Time-Triggered Conversion of Guards for Reachability Analysis of Hybrid Automata

Stanley Bak[1], Sergiy Bogomolov[2], and Matthias Althoff[3]

[1] Air Force Research Laboratory, Dayton, OH, USA
stanleybak@gmail.com
[2] Australian National University, Australia
sergiy.bogomolov@anu.edu.au
[3] Technische Universität München, Germany
althoff@in.tum.de

Abstract. A promising technique for the formal verification of embedded and cyber-physical systems is flow-pipe construction, which creates a sequence of regions covering all reachable states over time. Flow-pipe construction methods can check whether specifications are met for all states, rather than just testing using a finite and incomplete set of simulation traces. A fundamental challenge when using flow-pipe construction on high-dimensional systems is the cost of geometric operations, such as intersection and convex hull. We address this challenge by showing that it is often possible to remove the need to perform high-dimensional geometric operations by combining two model transformations, direct time-triggered conversion and dynamics scaling. Further, we prove the overapproximation error in the conversion can be made arbitrarily small. Finally, we show that our transformation-based approach enables the analysis of a drivetrain system with up to 51 dimensions.

1 Introduction

Hybrid automata [6] are often used to model embedded and cyber-physical systems with a combination of discrete and continuous dynamics. Due to their expressiveness, however, hybrid automata can be difficult to verify. The flow-pipe construction technique [39] performs analysis with *regions* of states; it starts with a given initial set of states and propagates the set forward in time, constructing a sequence of regions that overapproximate the reachable set of states up to a time bound. To check which states can take a discrete transition, a geometric intersection is performed between the continuous reachable region and a transition's guard set. Afterwards, the intersected states are combined together in an aggregation step, often done by taking their convex hull or performing a template polytope overapproximation. Without guards, methods exist which can scale to analyze purely continuous systems with thousands of state variables [10], but no such scalability results exist for systems with guards, due to the complexity

DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA #88ABW-2017-1923, 25 APR 2017)

of high-dimensional intersection and aggregation. For this reason, we propose a new method to try to remove the need for these costly geometric operations.

In particular, we leverage time-triggered transitions [2, 8]. In a system with time-triggered transitions, the discrete mode change occurs after a certain amount of time has elapsed. In contrast, space-triggered transitions have mode changes based on the system's continuous state, and may arise from models of switched systems, or continuous systems with gain-scheduled controllers. Propagating sets of states through time-triggered transitions is practically free; it does not require performing high-dimensional intersection or convex hull.

In this paper, we present a transformation that can, under certain assumptions, convert a space-triggered transition to a series of time-triggered transitions. The main assumption for this conversion is that all executions of the hybrid automaton must pass through the guard completely (partial intersections with the guard are not considered here), and resets along transitions are not allowed. The main contributions of this paper are as follows:

- We present a new transformation process to convert a space-triggered transition to a series of time-triggered transitions;
- We prove that, in theory, the overapproximation error due to the proposed transformation can be reduced to an arbitrarily small constant;
- We demonstrate that, in practice, the approach works well on a numerical example of a high-dimensional drivetrain system.

This paper first introduces key definitions (Section 2), provides descriptions of the proposed transformations and accuracy proof (Section 3), and then evaluates the approach on a numerical example (Section 4). Related approaches are then discussed (Section 5), followed by a conclusion (Section 6).

2 Preliminaries

In order to define and justify the soundness of the model transformation steps used in our approach, we need to first precisely define the syntax and semantics of hybrid automata and some related concepts.

Definition 1 (Hybrid Automaton). *A* **hybrid automaton** $\mathcal{H}$ *is a tuple* $\mathcal{H} \triangleq (Modes, Var, Init, Flow, Trans, Inv)$*, where: (a)* $Modes$ *is a finite set of discrete elements, each of which we call a mode; (b)* $Var = (x_1, \ldots, x_n)$ *is a list of real-valued variables. (c)* $Init(m) \subseteq \mathbb{R}^n$ *is a bounded set of initial values for* Var *for each mode* $m \in Modes$*; (d) For each* $m \in Modes$*, the flow relation* $Flow(m)$ *has the form of* $\dot{x} \in f_m(x)$*, where* $x \in \mathbb{R}^n$ *and* $f_m : \mathbb{R}^n \to 2^{\mathbb{R}^n}$*. (e)* $Trans$ *is a set of discrete transitions, each of which is a 4-tuple* (m, G, υ, m')*, where* m *and* m' *are the source and the target modes,* $G \subseteq \mathbb{R}^n$ *is the guard, and* $\upsilon : \mathbb{R}^n \to \mathbb{R}^n$ *is the update or reset of the transition; (f)* $Inv(m) \subseteq \mathbb{R}^n$ *is an invariant for each mode* $m \in Modes$*.*

When a hybrid automaton has n real-valued variables, we say that $\mathcal{H}$ is n-dimensional, and has states with continuous part in $\mathbb{R}^n$. Note that f_m is a set-valued function, i.e., differential inclusions, $\dot{x} \in f_m(x)$, are allowed [46]. When the flows are deterministic, we may simply write them as a differential equation $\dot{x} = f_m(x)$. Now, we introduce a notion of distance which is useful to quantify properties of hybrid automata as well as errors.

Definition 2 (Distance). *Let $\|\cdot\|$ be the L^2 norm of a point in $\mathbb{R}^n$ and $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$ be the* **distance** *between two points in $\mathbb{R}^n$. We will write $d_S(\mathbf{x})$ to mean the lower bound on the distance between the point $\mathbf{x}$ and a set S, $d_S(\mathbf{x}) = \inf\{d(\mathbf{x}, \mathbf{x}') \mid \mathbf{x}' \in S\}$.*

Although the flows can be non-deterministic, they must obey a Lipschitz continuity property: in each mode m, there exists a constant L_m such that, for any two points $\mathbf{x_1}, \mathbf{x_2} \in \mathbb{R}^n$, for any $\mathbf{y_1} \in f_m(\mathbf{x_1})$, there exists a $\mathbf{y_2} \in f_m(\mathbf{x_2})$ with $d(\mathbf{y_1}, \mathbf{y_2}) \leq L_m \cdot d(\mathbf{x_1}, \mathbf{x_2})$. To define the formal semantics of hybrid automata, we introduce the notion of a state.

Definition 3 (State). *A* **state** *$s \in States$ of an n-dimensional hybrid automaton is a pair $(m, \mathbf{x})$, with mode $m \in Modes$ and continuous part $\mathbf{x} \in \mathbb{R}^n$.*

The semantics of hybrid automata is defined in terms of executions where an execution is a sequence of states. A state can change either due to a continuous flow or discrete transition, which requires the definition of successors.

Definition 4 (Continuous Successor). *A state $(m', \mathbf{x}')$ is a* **continuous successor** *of another state $(m, \mathbf{x})$ if $m' = m$ and there exists a positive time t and a differentiable function $g : [0, t] \to \mathbb{R}^n$ such that the following holds: (1) $g(0) = \mathbf{x}$, (2) $g(t) = \mathbf{x}'$ and (3) for all $\delta \in (0, t)$: $g(\delta) \in Inv(m)$ and $(g(\delta), \dot{g}(\delta)) \in Flow(m)$.*

We refer to the time t as a dwell time spent in the mode m.

Definition 5 (Discrete Successor). *A state $(m, \mathbf{x}')$ is a* **discrete successor** *of another state $(m, \mathbf{x})$ if there exists a transition $(m, G, \upsilon, m') \in Trans$ such that $\mathbf{x} \in G$ and $\upsilon(\mathbf{x}) = \mathbf{x}'$.*

Based on these definitions, we can define an execution of a hybrid automaton:

Definition 6 (Execution). *An* **execution** *$\xi = s_0 s_1 \ldots$ of a hybrid automaton $\mathcal{H}$ is a (finite or infinite) sequence which starts at an initial state s_0 of $\mathcal{H}$, i.e. for $s_0 = (m, \mathbf{x})$ it holds that $\mathbf{x} \in Init(m)$. Each of s_{i+1} is either a continuous or discrete successor of s_i. For finite executions, we call the last state in the sequence the end state, and can refer to the duration of the execution as the sum of the dwelling times over all continuous successors in the execution. Given two executions of a hybrid automaton ξ and ξ', where ξ is finite, we say ξ is a prefix of ξ' if the sequence ξ' begins with ξ.*

The consideration of all possible executions defines the reachable states:

Definition 7 (Reachable States). *A state s is a* **reachable state** *of a hybrid automaton $\mathcal{H}$ if s is the end state of some execution of $\mathcal{H}$. In this paper, we primarily compare the continuous parts of reachable states. The set of all points which are the continuous part of some reachable state of $\mathcal{H}$ is written as $\mathsf{Reach}(\mathcal{H})$. The set of time-bounded continuous reachable states, which corresponds to the continuous parts of end states of all executions with durations no longer than T, is written as $\mathsf{Reach}_{\leq T}(\mathcal{H})$.*

To analyze our time-triggered transformation, we introduce overapproximating hybrid automata as well as a definition of error in the overapproximation. Note that, when considering continuous overapproximations (and later their errors), if $\mathcal{H}'$ has more variables than $\mathcal{H}$, the variables exclusive to $\mathcal{H}'$ are projected away before doing the comparison. That is, the projection of $\mathsf{Reach}(\mathcal{H}')$ onto Var, $\mathsf{Reach}(\mathcal{H}') \downarrow_{Var}$, is used instead of $\mathsf{Reach}(\mathcal{H}')$, where Var are the variables of the original automaton $\mathcal{H}$.

Definition 8 (Continuous Overapproximation). *A hybrid automaton $\mathcal{H}'$ is a* **continuous overapproximation** *of another hybrid automaton $\mathcal{H}$ if the inclusion $\mathsf{Reach}(\mathcal{H}') \supseteq \mathsf{Reach}(\mathcal{H})$ holds. A time-bounded version can also be defined. We say that $\mathcal{H}'$ with time bound T' is a time-bounded continuous overapproximation of automaton $\mathcal{H}$ with time bound T if $\mathsf{Reach}_{\leq T}(\mathcal{H}) \subseteq \mathsf{Reach}_{\leq T'}(\mathcal{H}')$.*

Definition 9 (Time-Bounded Continuous Overapproximation Error). *Let $\mathcal{H}$ be a hybrid automaton with a time bound T, and $\mathcal{H}'$ with time bound T' be a time-bounded continuous overapproximation of $\mathcal{H}$. Then, the* **time-bounded continuous overapproximation error** *is $\sup_{\mathbf{x}' \in \mathsf{Reach}_{\leq T'}(\mathcal{H}')} d_{\mathsf{Reach}_{\leq T}(\mathcal{H})}(\mathbf{x}')$.*

This measurement of error is equal to the asymmetric Hausdorff distance from $\mathsf{Reach}_{\leq T'}(\mathcal{H}')$ to $\mathsf{Reach}_{\leq T}(\mathcal{H}')$. Further, such a measurement is relevant for model transformations, such as the ones proposed in this paper. Rather than computing the reachable set of states of an automaton $\mathcal{H}$ with time bound T, we can instead compute reachable sets on a modified automaton $\mathcal{H}'$ with a different time bound T'. The time-bounded continuous overapproximation error can be used to measure the amount of error, in space, that an ideal reachability computation would have due to the use of $\mathcal{H}'$ instead of the original $\mathcal{H}$.

3 Transformations

In this section, we describe a way to convert certain space-triggered transitions into time-triggered ones. This is beneficial for reachability analysis algorithms, since time-triggered transitions can be handled efficiently. In order to do this, we first describe a direct time-triggered conversion transformation in Section 3.1, followed by a dynamics scaling transformation in Section 3.2. We combine the two transformations to construct the final automaton in Section 3.3, which we prove is an overapproximation with an error that can be made arbitrarily small in Theorem 1. We make four assumptions about the original automaton.

Assumption 1 *We present the conversion assuming that the original automaton $\mathcal{H}$ consists of two modes m_1 and m_2 with deterministic dynamics connected by a single transition (see top of Figure 1), and all the initial states are in m_1.*

It is often possible to apply the transformation to a more general hybrid automaton, by adapting the proposed process and considering a single transition at a time for the finite time-bound, as will be shown later in our evaluation in Section 4. In this conversion, there are two cases to consider: 1) The reachable set hits one guard set at a time. 2) Several guard sets are hit at once. Again, by removing the parts of the reachable set that are already hit by other guards, one can extract cases with a single guard intersection, as studied in this work (also see Sec. 5.5 of [2]). The only difference is that now a tree of possible next discrete states is spanned instead of consecutive next discrete states as in case 1). Notice that due to the finite time bound, and under a non-Zeno assumption, it is possible to unroll any loops in the automaton.

Assumption 2 *The single transition of $\mathcal{H}$ is space-triggered with a single linear condition, $G = \{\mathbf{x} \mid a \cdot \mathbf{x} = b\}$. Since the transition is space-triggered, there is no reset and the invariant of m_1 is $a \cdot \mathbf{x} \leq b$ (one side of the guard).*

The approach may be generalizable to more complex guards, but it would require a more complicated dynamics scaling process.

Assumption 3 *At some time t_{max}, all executions have taken the transition.*

Not all transitions satisfy Assumption 3, and it is one of the main restrictions of the approach.

Assumption 4 *For any amount of time t_γ, there exists a distance γ from the guard G, such that any execution that gets within distance γ of the guard must take the transition before t_γ time.*

Assumption 4 ensures that there are no executions that can touch the guard set and then back away without crossing the guard. One way to ensure this is by examining the Lie derivative of the guard level-set function, $B(\mathbf{x}) = a \cdot \mathbf{x} - b$ with respect to the mode's flow vector field. Due to the continuity of the flows, the condition is satisfied if there is some constant $\epsilon > 0$, such that at every point $\mathbf{x}$ where a transition might occur, $\frac{\partial B}{\partial \mathbf{x}} f_{m_1}(\mathbf{x}) \geq \epsilon$.

3.1 Direct Time-Triggered Conversion Transformation

First, we aim to replace the space-triggered transition of $\mathcal{H}$ with two time-triggered transitions. This is done by constructing a new automaton $\mathcal{H}_{tt}$, which is a continuous overapproximation of $\mathcal{H}$. We proceed with the following steps to transform the original hybrid automaton $\mathcal{H}$ to $\mathcal{H}_{tt}$:

1. We remove the space-triggered transition between m_1 and m_2.

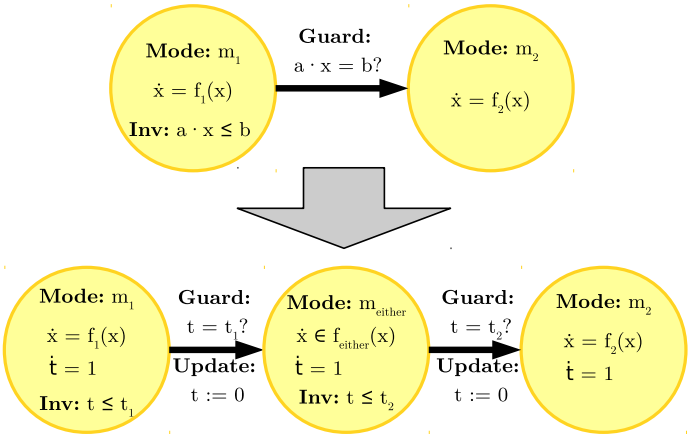

Fig. 1: The direct time-triggered conversion transformation, described in Section 3.1, converts the original automaton $\mathcal{H}$ (top), to an overapproximating automaton that only has time-triggered transitions, $\mathcal{H}_{tt}$ (bottom), using the parameters t_1 and t_2.

2. We add a new intermediate mode m_{either} and transitions such that the hybrid automaton switches from m_1 to m_{either} and then to m_2.
3. We add a new time variable, t, to the automaton with derivative $\dot{t} = 1$ in each mode. This will be used to force certain dwell times (exact times spent in each mode) as part of the new time-triggered transitions.
4. We equip the newly-introduced transitions with *time-triggered* guards. In other words, the guards are of the form $t = t_1$ and $t = t_2$, with invariants of the modes set to when t is less than t_1 or t_2, and resets $t := 0$ upon entering each mode. The first dwell time t_1 is selected to be the minimum duration when, in the original automaton, every finite execution with duration up to t_1 has an end state still in mode m_1. Similarly, t_2 is selected to be the smallest time such that the sum $t_1 + t_2$ is a time after which every finite execution with duration greater than or equal to $t_1 + t_2$ has an end state with mode m_2. Time t_2 exists because by Assumption 3, all executions eventually take the transition.
5. We assign the continuous dynamics in the mode m_{either} so that it *overapproximates* the dynamics in m_1 and m_2. This means that for any state $\mathbf{x}$, the flow in m_{either} contains the flows in m_1 and m_2, $f_1(\mathbf{x}) \cup f_2(\mathbf{x}) \subseteq f_{either}(\mathbf{x})$. In this way, we express the fact that executions of $\mathcal{H}$ with durations in the range $[t_1, t_1 + t_2]$ end in states that can be in either mode.

The conversion of $\mathcal{H}$ to $\mathcal{H}_{tt}$ is illustrated in Figure 1. Notice that, in practice, the times t_1 and t_2 would be available during flow-pipe construction since a tool must check at each step if a guard can be reached. This transformation results in a time-bounded overapproximation, which we prove next.

Lemma 1. *For any time bound T, the constructed $\mathcal{H}_{tt}$ is a time-bounded continuous overapproximation of the original automaton $\mathcal{H}$ with the same time bound.*

Proof. Consider any execution ξ of $\mathcal{H}$ which ends at a state s with continuous part $\mathbf{x}$. If the mode of s is m_1, then ξ is also directly an execution of $\mathcal{H}_{tt}$, and so an execution exists that ends with a state with continuous part equal to $\mathbf{x}$. In the other case, if the mode of s is m_2, then let t_{trans} be the maximum duration of any prefix of ξ ending in a state with mode m_1 (t_{trans} is the time of the transition). By the construction of $\mathcal{H}_{tt}$, $t_{trans} \geq t_1$.

The duration of ξ is either (1) less than or (2) greater than or equal to $t_1 + t_2$. In the first case, there is an execution of $\mathcal{H}_{tt}$ which first spends t_1 time in m_1, then spends $t_{trans} - t_1$ using the dynamics of m_1 in mode m_{either} (because m_{either}'s dynamics are a differential inclusion containing the dynamics of m_1), and finally spends the remaining time using the dynamics of m_2 in mode m_{either} (again, because, m_{either}'s dynamics contain m_2's dynamics). This execution ends in a state with continuous part equal to $\mathbf{x}$. In the second case, the execution would spend $t_2 - (t_{trans} - t_1)$ in mode m_{either} using the dynamics of m_2, and then use the remaining time in mode m_2 to also end at a state with continuous part $\mathbf{x}$. In all cases, we have constructed an execution of $\mathcal{H}_{tt}$ of equal duration with an end state with continuous part equal to $\mathbf{x}$, and this holds for any execution ξ of $\mathcal{H}$, and so $\mathsf{Reach}_{\leq T}(\mathcal{H}) \subseteq \mathsf{Reach}_{\leq T}(\mathcal{H}_{tt})$. □

The time-bounded continuous overapproximation error of $\mathcal{H}_{tt}$ crucially depends on the dwell time t_2 spent in the intermediate mode m_{either}. This follows from the fact that the dynamics in m_{either} is nondeterministic, subsuming both the dynamics of m_1 and m_2. This can be reduced by choosing the dynamics f_{either} to be as small as possible while still containing both f_1 and f_2. In general, however, the error cannot be eliminated without a further transformation.

3.2 Dynamics Scaling Transformation

Next, we introduce a dynamics scaling transformation that is later used to substantially reduce the overapproximation error in $\mathcal{H}_{tt}$. We first describe this transformation in isolation since it is quite general and we can show that it theoretically does not modify the continuously reachable states (Lemma 2).

Let $\mathcal{H}_{single}$ be a hybrid automaton with a single mode m with continuous dynamics $\dot{\mathbf{x}} = f(\mathbf{x})$. We proceed with the following steps to transform to construct a new automaton $\mathcal{H}_{scaling}$ from a copy of $\mathcal{H}_{single}$:

1. We create two additional copies of m: $m_{scaling}$ and m'.
2. We add a new time variable t in the automaton to measure the dwell time (unless such a variable already exists), with $\dot{t} = 1$ in all three modes.
3. We equip the automaton with time-triggered transitions with dwell times t_{begin} (from m to $m_{scaling}$) and $t_{scaling}$ (from $m_{scaling}$ to m'), where $t_{begin} > 0$ and $t_{scaling} > 0$ are parameters of the transformation.

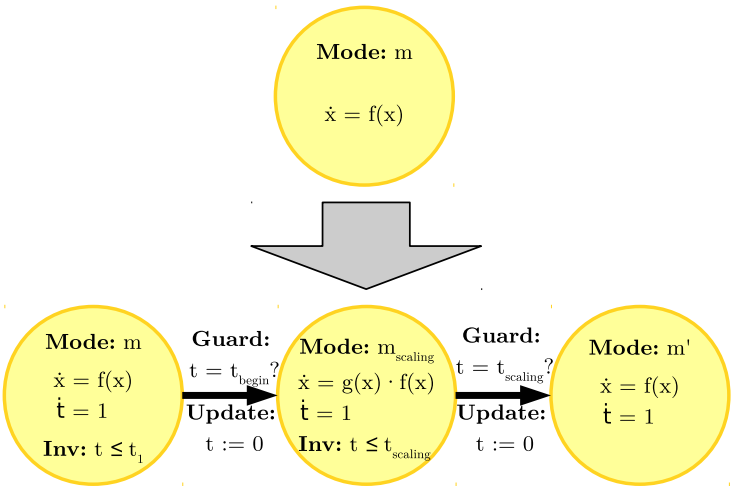

Fig. 2: The dynamics scaling transformation, described in Section 3.2, converts a single-mode automaton $\mathcal{H}_{single}$ (top), to an automaton with an identical continuous reachable set $\mathcal{H}_{scaling}$ (bottom).

4. We change the flow of $m_{scaling}$ to $\dot{\mathbf{x}} = g(\mathbf{x}) \cdot f(\mathbf{x})$ where $f(\mathbf{x})$ is the original dynamics in m, and $g(\mathbf{x})$, a user-defined function, is a scalar function that outputs a nonnegative number for every reachable state $\mathbf{x}$.

The dynamics scaling transformation is shown in Figure 2. It does not change the time-bounded continuous reachable set of states, which is proved next.

Lemma 2. *For any times T and $t_{scaling}$, the reachable set of the constructed $\mathcal{H}_{scaling}$ with time bound $T' = T + t_{scaling}$ is a zero-error time-bounded continuous overapproximation of the reachable set of $\mathcal{H}_{single}$ with time bound T.*

Proof. First, we show that $\mathcal{H}_{scaling}$ is a time-bounded continuous overapproximation, and then we analyze its error.

Consider any execution of $\mathcal{H}$ ending with a continuous state $\mathbf{x} \in \mathsf{Reach}_{\leq T}(\mathcal{H})$. The dynamics of each mode in $\mathcal{H}_{scaling}$ are identical to the original $\mathcal{H}_{single}$, except in $m_{scaling}$, where they get multiplied by a non-negative value at each point in space. This has the effect of scaling the vector field, without changing any of the directions. In the worst-case, the scaling factor in $m_{scaling}$, g can be zero, which effectively pauses the executions for at most $t_{scaling}$ time. Since the time bound of $\mathcal{H}_{scaling}$ is $T' = T + t_{scaling}$, we can ensure that $\mathbf{x} \in \mathsf{Reach}_{\leq T'}(\mathcal{H}_{scaling}) \downarrow Var$, and so $\mathcal{H}_{scaling}$ is a time-bounded continuous overapproximation. Notice that if ever $g(x) > 1$, more states may be reached by $\mathcal{H}_{scaling}$ than $\mathcal{H}$ for the same time-bound. However, any execution of $\mathcal{H}$ up to time T is still contained in (the larger) $\mathsf{Reach}_{\leq T'}(\mathcal{H}_{scaling})$.

In terms of error, consider any point in $\mathbf{x}' \in \mathsf{Reach}_{\leq T'}(\mathcal{H}_{scaling})$. Since the direction of the vector field in each of the modes of $\mathcal{H}_{scaling}$ is unchanged from

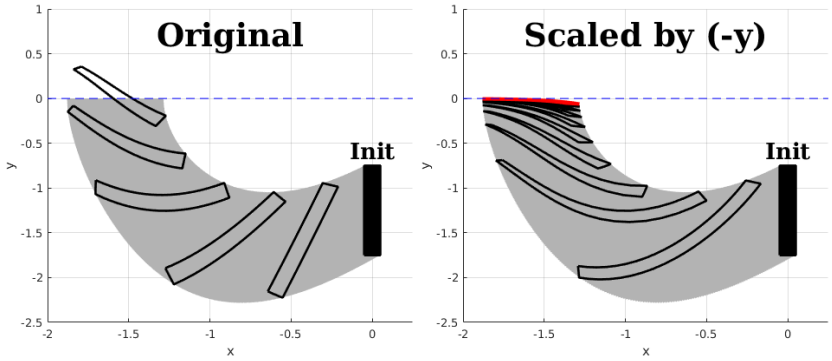

Fig. 3: For the Van der Pol dynamics, the currently-tracked set of states becomes flattened against the x axis when using a scaling function $g(\mathbf{x}) = -y$. The time-invariant set of reachable states below the x axis (grey states) is unchanged.

$\mathcal{H}$, the point $\mathbf{x}'$ will eventually be the continuous part of a reachable state of $\mathcal{H}$. Thus, $d_{\mathsf{Reach}(\mathcal{H})}(\mathbf{x}') = 0$, and so the overapproximation error is zero. □

Dynamics scaling can alter the set of states reachable with executions of a fixed duration, without altering the final (time-invariant) continuous reachable set. This is practically useful, because reachability algorithms usually perform their computations using sets which correspond to continuous trajectories of a fixed duration, which is sometimes called the *currently-tracked set of states* [8].

To reduce the error in the time-triggered conversion, we use dynamics scaling to flatten the currently-tracked set of states against a guard boundary. An illustration of this for the Van der Pol system, with dynamics $\dot{x} = y$ and $\dot{y} = (1 - x^2) \cdot y - x$, is shown in Figure 3. Here, when the dynamics is scaled based on the distance from the $y = 0$ guard, the currently-tracked set of states becomes flattened as it approaches the x axis.

3.3 Combined Scaled Time-Triggered Transformation

We now combine the transformations from Section 3.1 and Section 3.2 into a single transformation, which can theoretically be done with arbitrary accuracy. The steps to produce the final automaton $\mathcal{H}_{final}$ starting from $\mathcal{H}$ are as follows:

1. We first apply the time-triggered transformation on $\mathcal{H}$ to produce $\mathcal{H}_{tt}$.
2. Next, we apply a version of the dynamics scaling transformation to mode m_1 of $\mathcal{H}_{tt}$. For the time t_{begin} of the transformation we use t_1. Time $t_{scaling}$ is a parameter of the transformation. For the scaling function $g(\mathbf{x})$, we use the minimum distance from the point to the guard set, $d_G(\mathbf{x})$. Since the guard set is defined with a single linear condition, $G = \{x \mid a \cdot \mathbf{x} = b\}$, we use the dynamics scaling function $g(\mathbf{x}) = -\hat{a} \cdot \mathbf{x} + b$, where $\hat{a} = \frac{a}{\|a\|}$ is the normal vector associated with a and $\cdot$ is the standard dot product. This function is

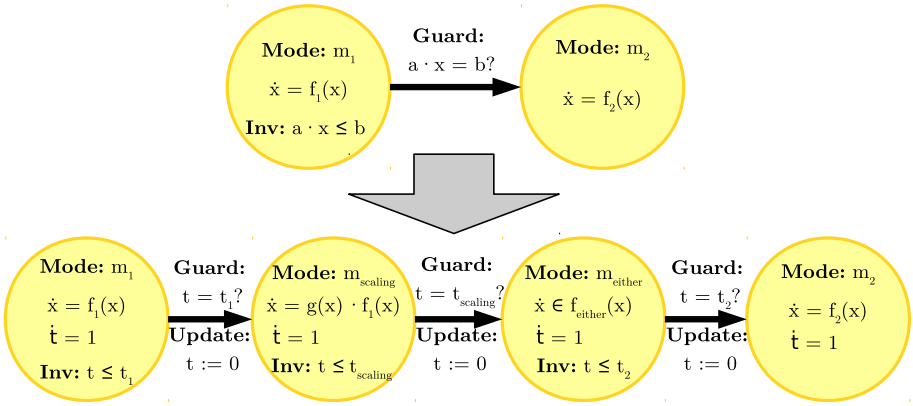

Fig. 4: The original automaton $\mathcal{H}$ (top) is transformed using both time-triggered conversion and dynamics scaling to produce the final automaton $\mathcal{H}_{final}$ (bottom). Theorem 1 proves that the error due to this transformation can be reduced to an arbitrarily small constant by increasing $t_{scaling}$.

nonnegative for any $\mathbf{x}$ in m_1, meeting the required condition for g in step 4 of the dynamics scaling transformation.

3. We directly transition from $m_{scaling}$ to m_{either}, deleting m_1' and its associated transitions.

The transformation is shown in Figure 4. Since the time-triggered transformation results in a time-bounded overapproximation (Lemma 1), and the application of the scaling transformation does not alter the reachable states but only the time at which they are reached (Lemma 2), the following corollary holds:

Corollary 1. *For any choice of $t_{scaling} \geq 0$, the constructed $\mathcal{H}_{final}$ with time bound $T' = T + t_{scaling}$ is a time-bounded continuous overapproximation of $\mathcal{H}$ with time bound T.*

Not only is $\mathcal{H}_{final}$ a time-bounded continuous overapproximation, but the overapproximiation error can also be reduced to an arbitrarily small constant. Intuitively, the reason why the error can be made arbitrarily small is that by increasing $t_{scaling}$, the set of states upon entering m_{either} becomes more and more flattened against the original guard's boundary. Since all states are then about to cross the guard, the time needed in m_{either} becomes arbitrarily small, reducing the only source of overapproximation error in the time-triggered conversion. Then, by the Lipschitz continuity of the dynamics, in finite time, the total divergence can also be made arbitrarily small as shown next.

Theorem 1. *For any time T and any desired error $\delta > 0$, there exists a $t_{scaling}$ such that the time-bounded continuous overapproximation error between $\mathcal{H}$ with time bound T and $\mathcal{H}_{final}$ with time bound $T' = T + t_{scaling}$ is less than δ.*

Proof. First, by Corollary 1, we know that $\mathcal{H}_{final}$ is a time-bounded overapproximation. Second, to show the error can be reduced to less than any $\delta > 0$, we will find a $t_{scaling}$ such that, given any point $\mathbf{x}' \in \mathsf{Reach}_{\leq T'}(\mathcal{H}_{final})$, there exists a point $\mathbf{x} \in \mathsf{Reach}_{\leq T}(\mathcal{H})$ such that $d(\mathbf{x}, \mathbf{x}') < \delta$.

Consider any $\delta > 0$. Let s' be the final state of an execution ξ' of $\mathcal{H}_{final}$ that has continuous part $\mathbf{x}'$, such that $s' = (m', \mathbf{x}')$. We proceed by showing there exists a $t_{scaling}$ for each of the four possible cases of m'.

Case $\mathbf{m' = m_1}$: ξ' is directly an execution of $\mathcal{H}$ for any value of $t_{scaling}$, so s' is reachable in $\mathcal{H}$. Thus, $\mathbf{x} = \mathbf{x}'$ with $d(\mathbf{x}, \mathbf{x}') = 0 < \delta$.

Case $\mathbf{m' = m_{scaling}}$: since the execution ξ' only reaches m_1 and $m_{scaling}$, we can apply the same reasoning as in Lemma 2, and for any value of $t_{scaling}$, find an execution of $\mathcal{H}$ that ends with continuous part $\mathbf{x} = \mathbf{x}'$. Again, $d(\mathbf{x}, \mathbf{x}') = 0 < \delta$.

Case $\mathbf{m' = m_{either}}$: the execution ξ' will contain a prefix execution ξ'_{prefix} of maximum duration that ends in mode $m_{scaling}$. Let $\mathbf{x'_{prefix}}$ be the continuous part of the end state of ξ'_{prefix}, and let t_{either} be the time ξ' spends in m_{either} (the duration of ξ' minus the duration of ξ'_{prefix}). Notice that by the same argument as in the second case, $\mathbf{x'_{prefix}}$ is the continuous part of some reachable state of $\mathcal{H}$. Let L_{either} be the Lipschitz constant of the flows in m_{either}. Using the definition of Lipschitz constants, the distance between $\mathbf{x'_{prefix}}$ and $\mathbf{x}'$ will be bounded by $\|\mathbf{x'_{prefix}}\|(e^{L_{either} \cdot t_{either}} - 1)$. Thus, if we can show that t_{either} can be made arbitrarily small, we can also make the distance between $\mathbf{x'_{prefix}}$ and $\mathbf{x}'$ less than δ. Notice that t_{either} is upper bounded by t_2, which is the maximum amount of time it takes for all executions reach the single guard G of the original automaton. Further, by Assumption 4 of the original automaton, for any amount of time t, there exists a distance γ from the guard G, such that any execution that gets within distance γ of G must take G's transition before t_γ time. We instantiate this assumption taking t_γ to be small enough such that $\|\mathbf{x'_{prefix}}\|(e^{L_{either} \cdot t_\gamma} - 1) < \delta$. Next, we assign a value of $t_{scaling}$ that ensures all continuous parts of executions are within γ distance of the guard set G upon entering m_{either}. By Assumption 3, all executions of $\mathcal{H}$ eventually take the transition. Let t_{max} be the maximum duration needed to take the transition using the original dynamics of m_1. Since the scaling function g was taken to be the distance from the guard set G (step 2 of the construction of $\mathcal{H}_{final}$), if we take $t_{scaling} \geq \frac{t_{max} - t_1}{\gamma}$, we can ensure that all executions, upon transitioning to m_{either}, are within γ distance of G (because if the dynamics of $m_{scaling}$ were used from the start, all executions would get within γ distance of G in at most $\frac{t_{max}}{\gamma}$ time). In this case, we have $t_{either} \leq t_2 \leq t_\gamma$, which ensures that $\|\mathbf{x'_{prefix}}\|(e^{L_{either} \cdot t_{either}} - 1) \leq \|\mathbf{x'_{prefix}}\|(e^{L_{either} \cdot t_\gamma} - 1) < \delta$. Taking $\mathbf{x} = \mathbf{x'_{prefix}}$, this ensures the desired $d(\mathbf{x}, \mathbf{x}') < \delta$.

Case $\mathbf{m' = m_2}$: Any execution of $\mathcal{H}'$ will have an intermediate continuous state at the moment it takes the transition to m_2 which we call $\mathbf{x'_i}$. By the same reasoning as in the above case, when $t_{scaling}$ is large enough, we can guarantee there is an execution of $\mathcal{H}$ that ends at a state that has just transitioned to m_2 with continuous part $\mathbf{x_i}$, with $\|\mathbf{x_i} - \mathbf{x'_i}\|$ less than any positive constant. If L_{m_2} is a Lipschitz constant of m_2, then the divergence in trajectories between

two points $\mathbf{x_i}$ and $\mathbf{x'_i}$ under m_2's dynamics is bounded by $\|\mathbf{x_i} - \mathbf{x'_i}\|e^{L_{m_2} \cdot t}$, for any time t. Pick $t_{scaling}$ such that $\|\mathbf{x_i} - \mathbf{x'_i}\|e^{L_{m_2} \cdot T} < \delta$. Now, since t_{m_2}, the amount of time spent in mode m_2, is less than the time bound T, we have $\|\mathbf{x_i} - \mathbf{x'_i}\|e^{L_{m_2} \cdot t_{m_2}} < \|\mathbf{x_i} - \mathbf{x'_i}\|e^{L_{m_2} \cdot T} < \delta$. Thus, we can take $\mathbf{x}$ as the end point of the execution that goes through $\mathbf{x_i}$, and then uses the flow in m_2. This guarantees $d(\mathbf{x}, \mathbf{x'}) < \delta$.

In all cases, there exists a $t_{scaling}$ so that the error can be reduced to less than δ. □

Constructing $\mathbf{t_{scaling}}$: One way to construct the final value of $t_{scaling}$ is: Following the reasoning in the m_2 case, we must ensure $\|\mathbf{x_i} - \mathbf{x'_i}\|e^{L_{m_2} \cdot T} < \delta$. This can be done by ensuring the error after m_{either} (which bounds $\|\mathbf{x_i} - \mathbf{x'_i}\|$) is less than $\delta/(e^{L_{m_2} \cdot T})$. By the reasoning in the m_{either} case, this occurs when $\|\mathbf{x'_{prefix}}\|(e^{L_{either} \cdot t_{either}} - 1) < \delta/(e^{L_{m_2} \cdot T})$, where $\|\mathbf{x'_{prefix}}\|$ is the norm of the continuous part of the state upon entering m_{either}, which is less than $\|Init(m_1)\|e^{L_{m_1} t_1}$. Substituting and solving for t_{either}, we get the condition $t_{either} < (\ln(\delta/(e^{L_{m_2} \cdot T})/\|Init(m_1)\|e^{L_{m_1} t_1}) + 1)/L_{either}$. Using Assumption 4, we get the γ corresponding to the t_{either} condition, and then need to find $t_{scaling}$ to ensure all executions are within γ distance of the guard upon switching to m_{either}. This can be ensured by taking the maximum time an execution can remain in the first mode t_{max}, multiplied by the maximum slowdown due to scaling γ, resulting in the value of $t_{scaling}$ that ensures the desired error, $t_{scaling} > t_{max}\gamma$. Notice that although theoretically the error can be made arbitrarily small by choosing a large enough $t_{scaling}$, flow-pipe construction methods often have overapproximation error, which may prevent this in practice.

Also notice that the proposed transformations do not depend on Assumption 4, but only the proof that we can make the error small uses it. By using a different scaling function g, we may be able to remove it.

4 Evaluation

We evaluate the proposed approach using a drivetrain system model [38]. The complete system dynamics, controller, and initial set description are available in another work [2], and here we only provide a brief description.

The model is a parameterized vehicle drivetrain, where one can add any number θ of rotating masses, corresponding to gears and other parts of the drivetrain such as transmission shafts. Given θ rotating masses, the model contains $n = 7 + 2\theta$ dimensions. The hybrid behavior of the drivetrain originates from backlash [42], which is caused by a physical gap between two components that are normally touching, such as gears. When the rotating components switch direction, for a short time they temporarily disconnect, and the system is said to be in the *dead zone*. All flows are linear ODEs.

We analyze an extreme maneuver from an assumed maximum negative acceleration that lasts for 0.2 [s], followed by a maximum positive acceleration that lasts for 1.8 [s]. The initial states of the model are taken to be a zonotope with

Table 1: Computational times in seconds ($n = 2\theta + 7$).

Dimensions	**11**	**21**	**31**	**41**	**51**
SpaceEx (smaller init)	541	1669	T/O	T/O	T/O
CORA Total	75	264	475	654	1073
CORA 1st guard					
Scaling Mode	36.95	132.00	281.44	377.25	620.54
Either Mode	0.06	0.11	0.37	1.49	2.96
CORA 2nd guard					
Scaling Mode	28.87	122.17	182.22	259.96	427.98
Either Mode	0.05	0.12	0.19	0.72	1.68

a single generator (a line segment in the n-dimensional space). We can make the reachability problem easier by considering scaling down the initial states by some percentage. The model has the following specification: after the change of direction of acceleration, the drivetrain completely passes the dead zone before being able to transmit torque again. Due to oscillations in the torque transmission, the drivetrain should not re-enter the dead zone of the backlash. The system has three modes with two transitions between them, and so as mentioned after Assumption 1, we needed to apply the proposed transformation twice.

The implementation was done in CORA [1], a MATLAB-based tool, on an i7 Processor and 6GB memory. We computed reachable sets with a varying number of rotating masses, where the total number of dimensions $n \in \{11, 21, 31, 41, 51\}$ (plots are shown in Fig. 5). Using the transformation approach from this paper, we could successfully analyze the model using initial states up to 40% of the desired size, while provably meeting the specification. The overall computational time, as well as the individual intersection times with the two guard sets are listed in Tab. 1. The total CPU time for the largest system with 51 dimensions is about 18 minutes. The table also shows that the runtime is dominated by computation in the scaling modes. This demonstrates a trade-off of our approach: Although we can eliminate geometric intersections, the dynamics in the scaling mode becomes more complicated. Further, the error can be reduced by spending more time in the scaling mode, at the cost of additional computation time.

We also analyzed the same system using SpaceEx [27], the state-of-the art reachability tool for linear systems which performs geometric operations for guard intersection and successor aggregation. Note that SpaceEx is a more general-purpose tool, while the approach here requires that all executions eventually reach the guard set. For SpaceEx, we used the space-time clustering analysis scenario [29], and, for each of the models, we tried to maximize the size of the initial set while ensuring the specification was not violated and the analysis time was less than 20 minutes. For the $n = 11$ model, using a `flowpipe-tolerance` parameter of 0.0005, in 541 seconds we could successfully analyze the system with up to 1.1% of the desired initial set size (1.2% violated the error specifi-

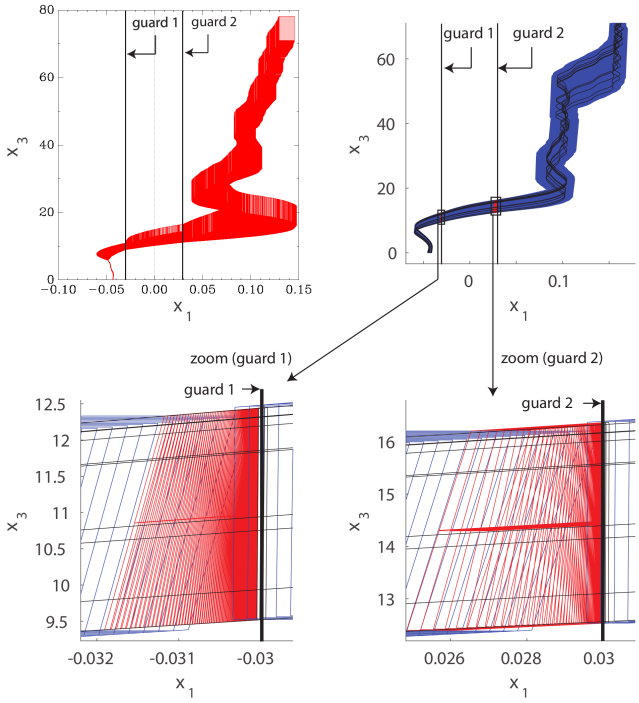

Fig. 5: Reachable set using SpaceEx with $n = 11$ (top left), and using our transformation approach with $n = 31$ (others).

cation). For the $n = 21$ model, using a `flowpipe-tolerance` value of 0.01, in 1669 seconds we could successfully analyze the system with up to 0.2% of the desired initial set size (0.3% exceeded the time bound). We did not find parameters which succeeded for the other models within the 20 minute timeout. This demonstrates that this model is particularly hard for techniques which do geometric intersection and aggregation as part of reachability analysis. A plot of the reachable states using SpaceEx and our technique with CORA is shown in Figure 5.

5 Related Work

Hybrid automata [6] can be analyzed by a number of methods [45]. These range from SMT [19,25,36], deduction [43], level sets [41], and simulation [24] based to flow-pipe construction based methods. In this paper, we compute time-bounded reachability [18] using flow-pipe construction. These methods work by propagating regions of states, which can be represented using constraint polyhedra [26], support functions [32], orthogonal polyhedra [23], zonotopes [5], Taylor models [21,44] or ellipsoids [17,37]. These representations have been implemented in

powerful analysis tools for hybrid automata including HyTech [34], Ariadne [13], Flow* [20], PHAVer [26], SpaceEx [27], CORA [1], and Hylaa [11].

Research on intersection in flowpipe construction involves techniques which avoid the intersection operation by employing a nonlinear mapping onto the guard [2]. Continuization methods [3,4,12] eliminate intersections using abstractions that get rid of fast-switching dynamics or eliminate guard intersections between similar continuous dynamics, as performed for m_{either} in this work. Frehse et al. [30] cast the intersection operator as a convex minimization problem. Other research examines the problem of efficiently computing geometric intersections for particular choices of data structures [31,33,35,40].

Our approach was presented using model transformations [9]. Model transformations can be used to derive abstractions [14–16,28]. Bak et al. [8] use model transformations to encode a hybridization process, i.e. reduction of the analysis of non-linear hybrid automata to linear ones, also using time-triggered transitions. The pseudo-invariants model transformation [7] can be used to reduce wrapping-effect error, which may also be possible with the dynamics scaling approach described in this work, without requiring geometric intersections.

In terms of applicability, benchmarks for various classes of hybrid systems have been proposed [22]. Of these proposed hybrid benchmarks, the main limit to applicability is the presence of resets along transitions. Some models, such as the filtered oscillator or glycemic control system, only use identity resets and do not have synchronization points, and so may be applicable for our method.

6 Conclusion

In this paper, we have presented a new way to handle certain types of discrete transitions when performing hybrid systems reachability analysis. We do this by creating an overapproximation abstraction of the original hybrid automaton that uses only time-triggered transitions. Given a space-triggered transition, our technique works in two steps: (1) we first add an intermediate mode which accounts for a "grey" zone when executions can be in either mode; (2) we scale the continuous dynamics in the first mode to decrease the time interval executions must spend in previously-mentioned "grey" zone. By applying these transformations, we remove the need to perform high-dimensional set intersection and set aggregation, which can be both time-consuming and error-prone.

The trade-off with this approach is that the system dynamics become more complicated when the dynamics are scaled. A system with linear ODEs, for example, becomes a quadratic system when scaling is being performed. The proposed method can also work for systems that originally have nonlinear dynamics, and so it is a promising approach to address part of the grand challenge of verifying high-dimensional nonlinear hybrid systems.

Acknowledgment. This work was partly supported by the ARC project DP140104219 (Robust AI Planning for Hybrid Systems), the German Research Foundation (DFG) grant number AL 1185/5-1, and the Air Force Office of Scientific Research (AFOSR).

References

[1] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[2] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Hybrid Systems: Computation and Control, HSCC'12, Beijing, China, April 17-19, 2012*, pages 45–54, 2012.

[3] M. Althoff, C. Le Guernic, and B. H. Krogh. Reachable set computation for uncertain time-varying linear systems. In *Hybrid Systems: Computation and Control*, pages 93–102, 2011.

[4] M. Althoff, A. Rajhans, B. H. Krogh, S. Yaldiz, X. Li, and L. Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. In *Proc. of the Int. Conference on Computer Aided Design*, pages 659–666, 2011.

[5] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.

[6] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[7] S. Bak. Reducing the wrapping effect in flowpipe construction using pseudo-invariants. In *4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems (CyPhy 2014)*, pages 40–43, 2014.

[8] S. Bak, S. Bogomolov, T. A. Henzinger, T. T. Johnson, and P. Prakash. Scalable static hybridization methods for analysis of nonlinear systems. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, pages 155–164, New York, NY, USA, 2016. ACM.

[9] S. Bak, S. Bogomolov, and T. T. Johnson. HYST: a source transformation and translation tool for hybrid automaton models. In *18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015)*, pages 128–133. ACM, 2015.

[10] S. Bak and P. S. Duggirala. Direct verification of linear systems with over 10000 dimensions. In *4th International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2017.

[11] S. Bak and P. S. Duggirala. Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 173–178. ACM, 2017.

[12] S. Bak and T. T. Johnson. Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In *36th IEEE Real-Time Systems Symposium (RTSS)*, San Antonio, Texas, Dec. 2015. IEEE Computer Society.

[13] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, and T. Villa. Assume-guarantee verification of nonlinear hybrid systems with ARIADNE. *International Journal of Robust and Nonlinear Control*, 24:699–724, 2014.

[14] S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C. S. Pasareanu, A. Podelski, and T. Strump. Assume-guarantee abstraction refinement meets hybrid systems. In *10th International Haifa Verification Conference (HVC 2014)*, volume 8855 of *LNCS*, pages 116–131. Springer, 2014.

[15] S. Bogomolov, G. Frehse, R. Grosu, H. Ladan, A. Podelski, and M. Wehrle. A box-based distance between regions for guiding the reachability analysis of SpaceEx. In *Computer Aided Verification (CAV 2012)*, volume 7358 of *LNCS*, pages 479–494. Springer, 2012.

[16] S. Bogomolov, C. Mitrohin, and A. Podelski. Composing reachability analyses of hybrid systems for safety and stability. In *Proc. of the 8th International Symposium on Automated Technology for Verification and Analysis*, pages 67–81, 2010.

[17] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Hybrid Systems: Computation and Control*, LNCS 1790, pages 73–88. Springer, 2000.

[18] T. Brihaye, L. Doyen, G. Geeraerts, J. Ouaknine, J.-F. Raskin, and J. Worrell. Time-bounded reachability for monotonic hybrid automata: Complexity and fixed points. In *International Symposium on Automated Technology for Verification and Analysis*, pages 55–70. Springer, 2013.

[19] L. Bu, Y. Li, L. Wang, X. Chen, and X. Li. Bach 2 : Bounded reachability checker for compositional linear hybrid systems. In *Proc. of Design, Automation & Test in Europe*, pages 1512–1517, 2010.

[20] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of Computer-Aided Verification*, LNCS 8044, pages 258–263. Springer, 2013.

[21] X. Chen, S. Sankaranarayanan, and E. Ábrahám. Taylor model flowpipe construction for non-linear hybrid systems. In *Proc. of the 33rd IEEE Real-Time Systems Symposium*, 2012.

[22] X. Chen, S. Schupp, I. B. Makhlouf, E. Ábrahám, G. Frehse, and S. Kowalewski. A benchmark suite for hybrid systems reachability analysis. In *NASA Formal Methods Symposium*, pages 408–414. Springer, 2015.

[23] T. Dang. *Vérification et synthèse des systèmes hybrides*. PhD thesis, Institut National Polytechnique de Grenoble, 2000.

[24] A. Donzé and O. Maler. Systematic simulations using sensitivity analysis. In *Hybrid Systems: Computation and Control*, LNCS 4416, pages 174–189. Springer, 2007.

[25] M. Fränzle and C. Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in System Design*, 30(3):179–198, 2007.

[26] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *Hybrid Systems: Computation and Control*, LNCS 3413, pages 258–273. Springer, 2005.

[27] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *Proc. of the 23rd International Conference on Computer Aided Verification*, LNCS 6806, pages 379–395. Springer, 2011.

[28] G. Frehse, S. K. Jha, and B. H. Krogh. A counterexample-guided approach to parameter synthesis for linear hybrid automata. In *International Workshop on Hybrid Systems: Computation and Control*, pages 187–200. Springer, 2008.

[29] G. Frehse, R. Kateja, and C. Le Guernic. Flowpipe approximation and clustering in space-time. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 203–212. ACM, 2013.

[30] G. Frehse and R. Ray. Flowpipe-guard intersection for reachability computations with support functions. In *Proc. of Analysis and Design of Hybrid Systems*, pages 94–101, 2012.

[31] K. Ghorbal, E. Goubault, and S. Putot. A logical product approach to zonotope intersection. In *Proc. of the 27th International Conference on Computer Aided Verification*, pages 212–226, 2010.

[32] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. In *Proc. of the 17th IFAC World Congress*, pages 8966–8971, 2008.

[33] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proc. of Hybrid Systems: Computation and Control*, LNCS 4981, pages 215–228. Springer, 2008.

[34] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: The next generation. In *Proc. of the 16th IEEE Real-Time Systems Symposium*, pages 56 – 65, 1995.

[35] F. Immler. A verified algorithm for geometric zonotope/hyperplane intersection. In *Proc. of the Conference on Certified Programs and Proofs*, pages 129–136, 2015.

[36] S. Kong, S. Gao, W. Chen, and E. Clarke. dReach: δ-reachability analysis for hybrid systems. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems, 200-205*, pages 200–205, 2015.

[37] A. B. Kurzhanski and P. Varaiya. *New Directions and Applications in Control Theory*, chapter Ellipsoidal Techniques for Hybrid Dynamics: the Reachability Problem, pages 193–205. Springer, 2005.

[38] A. Lagerberg. A benchmark on hybrid control of an automotive powertrain with backlash. Technical Report R005/2007, Signals and Systems, Chalmers University of Technology, 2007.

[39] C. Le Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[40] M. Maïga, N. Ramdani, L. Travé -Massuyès, and C. Combastel. A CSP versus a zonotope-based method for solving guard set intersection in nonlinear hybrid reachability. *Mathematics in Computer Science*, 8:407–423, 2014.

[41] I. M. Mitchell and Y. Susuki. Level set methods for computing reachable sets of hybrid systems with differential algebraic equation dynamics. In *Hybrid Systems: Computation and Control*, LNCS 4981, pages 630–633. Springer, 2008.

[42] M. Nordin and P.-O. Gutman. Controlling mechanical systems with backlash – a survey. *Automatica*, 38:1633 – 1649, 2002.

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

[44] N. Ramdani and N. S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149–162, 2010.

[45] S. Schupp, E. Ábrahám, X. Chen, I. Ben Makhlouf, G. Frehse, S. Sankaranarayanan, and S. Kowalewski. Current challenges in the verification of hybrid systems. In *Proc. of the Fifth Workshop on Design, Modeling and Evaluation of Cyber Physical Systems*, pages 8–24, 2015.

[46] G. V. Smirnov. *Introduction to the Theory of Differential Inclusions*. American Mathematical Society, 2002.