Deciding Reachability for 3-Dimensional Multi-Linear Systems [*]

Olga Tveretina

Karlsruhe Institute of Technology
Karlsruhe, Germany

olga.tveretina@kit.edu

Daniel Funke

Karlsruhe Institute of Technology
Karlsruhe, Germany

daniel.funke@student.kit.edu

This paper deals with the problem of point-to-point reachability in multi-linear systems. These systems consist of a partition of the Euclidean space into a finite number of regions and a constant derivative assigned to each region in the partition, which governs the dynamical behavior of the system within it. The reachability problem for multi-linear systems has been proven to be decidable for the two-dimensional case and undecidable for the dimension three and higher. Multi-linear systems however exhibit certain properties that make them very suitable for topological analysis. We prove that reachability can be decided exactly in the 3-dimensional case when systems satisfy certain conditions. We show with experiments that our approach can be orders of magnitude more efficient than simulation.

1 Introduction

During the last decades a lot of devices have been developed that consist of computers interacting with a physical environment. Computers perform discrete operations, while a physical environment has continuous dynamics. Such systems are called hybrid systems. Many of the applications of hybrid systems, such as intelligent highway systems, air traffic management systems and others are *safety critical* and require the guarantee of a safe operation.

Formally verifying safety properties of hybrid systems consists of building a set of reachable states and checking if this set intersects with a set of unsafe states. Therefore one of the most fundamental problems in the analysis of hybrid systems is the *reachability* problem.

The reachability problem is known as being difficult. It has been shown to be decidable for special kinds of hybrid automata [1, 7, 9, 10, 11] including timed automata [1], some classes of rectangular hybrid automata [7] and o-minimal hybrid automata [9].

Since only certain kinds of hybrid systems allow for the exact computation of the reachable set, approaches for safety verification include the approximation of reachability analysis and abstraction techniques. But these techniques are easy to fail when applied to large systems since the complexity rises up very quickly with an increase in system size.

One of the drawbacks of approximation and propagation techniques is that too little attention is paid to the geometric properties of the systems under analysis [3]. There are two main approaches in this direction: 1) methods that use topological properties of the plane [12], and 2) techniques based on the existence of integrals and the ability to compute them [5].

In this paper we consider *multi-linear systems* (ML) also often called *piecewise constant derivative systems* (PCDs) in the literature. They are a special kind of hybrid system, where the number of dimensions refers to the number of continuous variables. Such systems satisfy the following restrictions: A

---

discrete state is defined by a set of linear constraints and discrete transitions do not change continuous variables. ML systems have been proven to be decidable for the two-dimensional case [12], whereas the results presented in [2] state that such systems are undecidable for the dimension three and higher.

The decidability results for the 2-dimensional case rely on the existence of a periodic trajectory after a finite number of steps. This property does not hold for higher dimensions. Nevertheless, 3-dimensional systems also feature some sort of regularity. And, as in the 2-dimensional case, 3-dimensional multi-linear systems exhibit certain properties that make them very suitable for topological analysis.

Contribution. We consider a subclass of multi-linear systems, that we call multi-linear λ-systems. These systems satisfy the following property: If there is a cyclic trajectory, then the the points of each cycle iteration intersecting the same boundary element of a polyhedron lie on a straight line. A straight-forward consequence of this assumption is that the distances between the corresponding boundary points of different rounds are proportional (λ-property). We introduce the notion of a hypercycle, a generalization of a cycle. The infinity criterion for 2-dimensional case, refer to [12], has an analog in 3 dimensions. We show that the λ-property holds also for hypercycles, and the reachability can be decided exactly if the derived infinity criterion for 3 dimensions holds for a hypercycle.

We have implemented our approach and compared it with simulation. As soon as our algorithm detects a cycle (or a hypercycle) for which the given infinity criterion holds, the algorithm requires constant number of steps. While the number steps for simulation grows exponentially with the distance between points. Algorithms for computing reachable states are often based on floating point computations that involve rounding errors and the correctness of such algorithms can be violated. Since our algorithm takes significantly less steps, it leads to more exact computations.

A complete version of the paper containing all proofs and the details of the benchmarks is available at [14].

2 Multi-Linear Systems

Multi-linear systems consist of a partition of the Euclidean space into a finite number of regions and a constant derivative assigned to each region in the partition. In this section we define these systems in a way similar to [12].

We consider an *n*-dimensional Euclidean space $\mathscr{R}^n$ with a metric d and points in it denoted by *x* and *y*. In the following, we specify the position of any point in 3-dimensional space by three Cartesian coordinates. A *linear half space* is a set of all point in $\mathscr{R}^n$ satisfying $Ax + B \bowtie 0$, where $\bowtie \in \{<, \leq, >, \geq\}$, *A* is a rational vector and *B* is a rational number. A polyhedron is a subset of $\mathscr{R}^n$ obtained by intersecting a finite number of linear half spaces. Since we have a finite number of linear half spaces that divide the complete *n*-dimensional Euclidean space, there are polyhedra that are not bounded from all sides.

Definition 2.1 (Polyhedral partition) *Given a finite set of linear half spaces $\mathscr{S} = \{A_i x + B_i \bowtie 0, 1 \leq i \leq n\}$, we say that $\mathscr{P}(\mathscr{S}) = \{P_1, \ldots, P_m\}$ is a polyhedral partition of $\mathscr{R}^n$ by $\mathscr{S}$ if: 1) $\bigcup_{i=1}^m P_i = \mathscr{R}^n$, and 2) $P_i \cap P_j = \emptyset$ for distinct $i, j \in \{1, \ldots, m\}$.*

When it is convenient we will use $\mathscr{P}$ instead of $\mathscr{P}(\mathscr{S})$ to denote a polyhedral partition. Given a polyhedral partition $\mathscr{P}(\mathscr{S})$, we define the set of its boundary points as

$$\mathrm{Bd}(\mathscr{P}(\mathscr{S})) = \{y \in \mathscr{R}^n \mid \exists (Ax + B \bowtie 0) \in \mathscr{S} : Ay + B = 0\}.$$

For each polyhedron $P \in \mathscr{P}(\mathscr{S})$, we define the set of the boundary points as

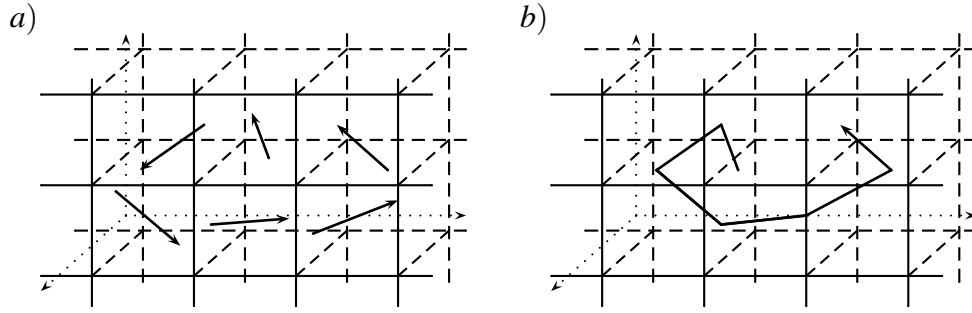$$\mathrm{Bd}(P) = \mathrm{Bd}(\mathscr{P}(\mathscr{S})) \cap P.$$

Figure 1: a) A simple 3-ML and b) a possible trajectory

Note that, depending on the partition, the set of boundary points of some polyhedra can be empty.

Definition 2.2 (Boundary element) *Given a polyhedral partition $\mathscr{P}(\mathscr{S})$ and a polyhedron $P \in \mathscr{P}(\mathscr{S})$, we say that e is a boundary element of P if the following holds.*

(1) $e \subseteq Bd(P)$, and

(2) There is $(Ax + B \bowtie 0) \in \mathscr{S}$ such that if $y \in e$ then $Ay + B = 0$.

An *n*-dimensional multi-linear system consists of a partitioning $\mathscr{P}(\mathscr{S}) = \{P_1, \ldots, P_m\}$ of the space $\mathscr{R}^n$ into a finite set of polyhedral regions and a constant derivative c_i assigned to each region P_i. We define such systems and a trajectory similar to [12].

Definition 2.3 (Multi-linear system) *We define a multi-linear system on $\mathscr{R}^n$ as a pair $\mathscr{H} = (\mathscr{P}, f)$, where $\mathscr{P}$ is a polyhedral partition of $\mathscr{R}^n$ and $f : \mathscr{P} \to \mathscr{R}^n$ is a function that assigns a vector c to each $P \in \mathscr{P}$.*

In the following we concentrate on 3-dimensional multi-linear systems. A simple 3-dimensional multi-linear system is depicted in Figure 1. The trajectories of such systems are sequences of line segments, where the break points belong to the boundaries of polyhedra. Multi-linear systems are deterministic in a sense that for each initial point there is exactly one corresponding trajectory.

We assume that the assigned derivative vectors of two neighboring polyhedra may not be directed towards the same boundary, since this would lead to Zeno behavior when a system performs infinitely many transitions in a finite period of time.

In the rest of the paper we use the following notations. By ε we denote the empty sequence. We use $s_1.s_2$ to denote the concatenation of sequences s_1 and s_2, $(s^i)_{i=1}^m$ is a shortcut for the sequence $s^1.\ldots.s^m$. Given a sequence s, we denote by $s^{\overline{x}}$ a (possibly infinite) sequence $s.s.\ldots$ if s is repeated at least two times. By $s_1 \sqsubset s_2$ we mean that $s_2 = s'_2.s_1.s''_2$ for some sequences s_1, s_2, s'_2, s''_2 and at most one of s'_2 and s''_2 is not the empty sequence.

In the following definitions for simplicity and without loss of generality we can assume that a trajectory always starts at a boundary element.

Definition 2.4 (Trajectory) *Let $\mathscr{H}$ be a ML, and $x_0 \in \mathscr{X}$ be a point.*

1. A trajectory starting at x_0 is a sequence $\tau = x_0.x_1.x_2.\ldots$ where for $i \geq 0$ there is $P_i \in \mathscr{P}$ such that $x_i \in Bd(P_i)$ and for $y \in]x_{i-1}, x_i[$ there is no $P' \in \mathscr{P}$ such that $y \in Bd(P')$. We denote by $Tr(\mathscr{H})$ the set of all trajectories of $\mathscr{H}$.
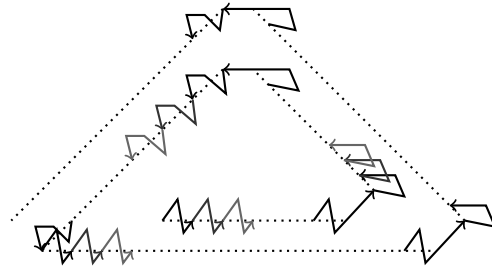
Figure 2: A schematic representation of a hypercycle

   2. *A sub-trajectory of τ, written as $\tau^s \sqsubset \tau$, is a finite (possibly empty) sequence $\tau^s = x_i.x_{i+1}.\ldots.x_j$. We denote by Tr^s the set of all sub-trajectories of $\mathcal{H}$.*

Definition 2.5 (Signature of a trajectory) *Let $\mathcal{H}$ be a ML, and $x_0 \in \mathcal{X}$ be a point. We assume a trajectory $\tau = x_0.x_1.x_2.\ldots$. We say that a sequence of boundary elements $\sigma(\tau) = e_0.e_1.e_2.\ldots$ is a signature of τ if $x_i \in e_i$ for $i \geq 0$. We denote by $\Sigma(\mathcal{H})$ the set of signatures of all trajectories of $\mathcal{H}$ and by $\Sigma^s(\mathcal{H})$ the set of signatures of all sub-trajectories of $\mathcal{H}$.*

Definition 2.6 (Simple trajectory) *Let $\mathcal{H}$ be a ML. We say that $\tau \in Tr^s(\mathcal{H})$ is a simple trajectory if $\sigma(\tau').\sigma(\tau') \not\sqsubset \sigma(\tau)$ for each $\tau' \in Tr^s(\mathcal{H})$ such that $\sigma(\tau') \sqsubset \sigma(\tau)$ and $\tau' \neq \varepsilon$. We denote by $\overline{Tr}(\mathcal{H})$ the set of all simple trajectries and by $\overline{\Sigma}(\mathcal{H})$ the set of signatures of all simple trajectories of $\mathcal{H}$.*

   For each multi-linear system, the number of polyhedra in the corresponding polyhedral partition is finite. Hence, we conclude that the number of signatures corresponding to the simple trajectories is also finite.

Lemma 2.7 *For each ML $\mathcal{H}$, $\overline{\Sigma}(\mathcal{H})$ is a finite set.*

   The notion of a cycle plays an important role in the next section. Due to the finiteness of the number of polyhedra in the polyhedral partition of each multi-linear system, each trajectory either reaches a region it never leaves or its subtrajectories form cycles of boundary elements.

Definition 2.8 (Cycle) *Let $\mathcal{H}$ be a ML. We say that τ, a (sub)trajectory of $\mathcal{H}$, is a cycle if $Sign(\tau) = \sigma^{\overline{\star}}$ for $\sigma \in \overline{\Sigma}(\mathcal{H})$. We denote by $C(\mathcal{H})$ the set of all cycles of $\mathcal{H}$.*

   Multi-linear systems for the dimension two have a nice property that makes the analysis simpler: Each trajectory has an ultimately periodic structure, i.e. after finite number of steps it forms a cycle in terms of visited boundary elements. This property does not hold for higher dimensions. Therefore, we introduce a notion of a hypercycle. This is a generalization of a cycle in the following sense: a hypercycle contains (several) cycles adjoined by simple trajectories. In each iteration of the hypercycle the number of passes through each cycle may vary but the sequence of visited boundary elements is preserved.

Definition 2.9 (Hypercycle) *Let $\mathcal{H} = (\mathcal{P}, f)$ be a multi-linear system. We say that a trajectory τ is a hypecycle if $\sigma(\tau) = (\sigma_1'.\sigma_1^{\overline{\star}}.\ldots.\sigma_m'.\sigma_m^{\overline{\star}})^{\overline{\star}}$ for $m \geq 1$, $\sigma_i \in \overline{\Sigma}(\mathcal{H})$ and $\sigma_i' \in \overline{\Sigma}(\mathcal{H}) \cup \{\varepsilon\}$ and at least one of the following holds.*

   • *There is $1 \leq i \leq m$ such that $\sigma_i' \neq \varepsilon$,*

   • *$m \geq 2$.*

We denote by $C^h(\mathcal{H})$ the set of all hypercycles of $\mathcal{H}$.

   In fact, the notion of a hypercycle can be generalized further by considering cycles of hypercycles. But in this paper we restrict the class of systems under consideration to the systems such that each trajectory is either a cycle or a hypercycle after finite number of steps.
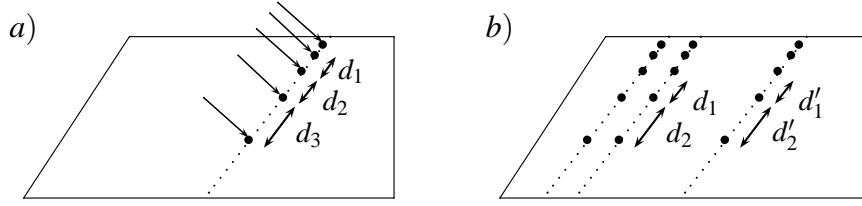
Figure 3: λ-cycle: a) $d_1/d_2 = d_2/d_3$, and b) $d_1/d_2 = d_1'/d_2'$

3 Deciding Reachability for a Special Class of Multi-Linear Systems

In this section we analyze topological properties of a subclass of multi-linear systems. This subclass is defined by a generalization of properties of 2-dimensional ML systems.

Namely, we assume that if there is a cyclic trajectory, then the points of each cycle iteration intersecting the same boundary element of a polyhedron lie on a straight line, called the λ-property. A straightforward consequence of this assumption is that the distances between the corresponding boundary points of different rounds are proportional.

If there is a cyclic trajectory, then the points of each cycle iteration intersecting the same boundary element of a polyhedron not necessarily lie on a straight line. In general case, even the angle between the corresponding line segments is not preserved. Nevertheless, we tend to think that for sufficiently many systems, especially for systems having some symmetry in their description, the trajectories obey the λ-property.

3.1 The Reachability Problem

In the following to be able to perform exact computations, we assume that all coefficients in a system are rationals.

Since a solution of a differential equation is unique for a given initial point in combination with rationality of coefficients, we obtain the following property. Given a multi-linear system and a rational initial point x, it is possible to compute the point y reachable from x after time interval Δt exactly.

Definition 3.1 (Reachability problem) *Given a multi-linear system $\mathcal{H} = (\mathcal{P}, f)$ and two points x and y, the problem of point-to-point reachability Reach$(\mathcal{H}, x, y)$ is stated as follows: Given two points $x, y \in \mathcal{R}^3$, is there a trajectory $\tau(\mathcal{H}, x)$ such that $y \in \tau(\mathcal{H}, x)$.*

3.2 Reachability for Multi-Linear λ-Systems

Now we define formally a subclass of multi-linear systems we consider.

Definition 3.2 (λ-cycle and λ-line) *Let $\mathcal{H} = (\mathcal{P}, f)$ be a ML. Suppose for $\tau \in Tr^s$ the following holds.*

- *$\tau = (x_1^j, \ldots, x_s^j)_{j=1}^t$ where for $1 \leq i \leq s$, $1 \leq j \leq t$ there are $P_i \in \mathcal{P}$ and $e_i \in Bd(P_i)$ such that $x_i^j \in e_i$.*

- *$\overrightarrow{x_i^j x_i^{j+1}} = \lambda_i \cdot \overrightarrow{x_i^{j-1} x_i^j}$ for $x_i^{j-1}, x_i^j, x_i^{j+1}$, $1 \leq i \leq s$, $1 < j < t$.*

Then we say that τ is a λ-cycle. We say that a line L_i is a λ-line of τ with respect to e_i if $x_i^{j-1}, x_i^j, x_i^{j+1} \in L_i$, $1 \leq i \leq s$, $1 < j < t$.
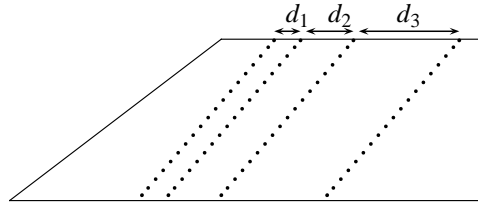
Figure 4: λ-hypercycle: The ratio of the distances between λ-lines of consequitive rounds of a hyper-cycle is preserved: $d_1/d_2 = d_2/d_3$

The notion of λ-cycle can be extended to a hypercycle. In the following, given two parallel lines L_1 and L_2, we denote by $\mathrm{Dist}(L_1,L_2)$ the distance between L_1 and L_2, i.e. the length of a line segment $[x_1,x_2]$ such that $x_1 \in L_1$, $x_2 \in L_2$ and $[x_1,x_2] \perp L_1$.

Definition 3.3 (λ-hypercycle) *Let $\mathcal{H} = (\mathcal{P}, f)$ be a ML. Suppose a trajectory τ is a hypercycle, i.e. $\sigma(\tau) = (\sigma_1'.\sigma_1^{\overline{\star}}.\ldots.\sigma_m'.\sigma_m^{\overline{\star}})^{\overline{\star}}$ for $\sigma_i, \sigma_i' \in \overline{\Sigma}(\mathcal{H})$. Let for $1 \le i \le m$, $e \in \sigma_i$, and L_{b-2}, L_{b-1}, L_b and L_{b+1} are λ-lines for e of the corresponding consecutive rounds $b-2$, $b-1$, b and $b+1$ of τ. We say that τ is a λ-hypercycle if*

$$\frac{d_{b-1}}{d_b} = \frac{d_b}{d_{b+1}},$$

where $d_a = \mathrm{Dist}(L_{a-1}, L_a)$, $b-2 < a \le b+1$.

Definition 3.4 (λ-system) *Let $\mathcal{H} = (\mathcal{P}, f)$ be a ML. We say that $\mathcal{H}$ is a λ-system if for each τ the following holds: 1) If τ is a cycle then τ is a λ-cycle. 2) If τ is a hypercycle then τ is a λ-hypercycle.*

As the next step, we define computable properties that would allow us to check whether a system is a λ-system. Lemma 3.5 defines conditions sufficient for a cycle to be a λ-cycle: As soon as three points of consecutive cycle iterations lie on a straight line, the ratio of the distances between consecutive points of different rounds is preserved.

Lemma 3.5 *Let $\mathcal{H} = (\mathcal{P}, f)$ be a ML. Suppose $\sigma(\tau) = \sigma^{\overline{\star}}$ for $\tau \in Tr^s$ and $\sigma \in \Sigma(\mathcal{H})$. Assume that $x_1, x_2, x_3 \in e$, $e \in \sigma$, are consecutive points of intersection of τ and e. If $x_1, x_2, x_3 \in L$ for some line L then τ is a λ-cycle.*

As we see in Lemma 3.6, it is sufficient for the λ-property to hold for two distinct trajectories going through the same cycle of boundary elements. Then it holds for each cycling trajectory going through the same cycle. Note that it is sufficient to compute for each trajectory whether three points of the consecutive rounds are in one line.

Lemma 3.6 *Let $\mathcal{H} = (\mathcal{P}, f)$ be a ML. Suppose $\sigma(\tau_i) = \sigma^{\overline{\star}}$ for a trajectory τ_i, $1 \le i \le 3$. Assume that τ_1 and τ_2 are λ-cycles. Then the following holds.*

- *τ_3 is a λ-cycle.*
- *$L_1^e \parallel L_2^e \parallel L_3^e$, where L_i^e is a λ-line of τ_i with respect to e for each $e \in \sigma$.*

Lemma 3.7 *Let $\mathcal{H} = (\mathcal{P}, f)$ be a ML. Suppose each $\tau \in C(\mathcal{H})$ is a λ-cycle. Then each $\tau' \in C^h(\mathcal{H})$ is a λ-hypercycle.*

Algorithm 1 POINT-TO-POINT REACHABILITY

Input: points $x_0, y \in \mathscr{R}^3$, ML system $\mathscr{H}$, maximal simulation steps $n \in \mathscr{N}$
Output: $\exists \tau(\mathscr{H}, x_0) = x_0, x_1, \ldots, y$

  1: $y' = \mathscr{H}(y)$ $\{y \in P_i$ and $y' \in \mathrm{Bd}(P_i)$ for some partition $i\}$
  2: $x \leftarrow x_0$ $k \leftarrow 0$
  3: **while** $k \leq n$ **do**
  4: $x \leftarrow \mathscr{H}(x)$ $k \leftarrow k+1$
  5: **if** cycle $\zeta = (x_i, \ldots, x_{i+s})^{\overline{\star}}$ detected **then**
  6:   **if** $\mathrm{Bd}(y') = \mathrm{Bd}(x_b) \in \zeta$ **then** {boundary element of y is in cycle}
  7:    **if** $y' = x_b^1 + t \cdot \overrightarrow{x_b^1 x_b^2}$ for $t = \frac{1-\lambda_b^k}{1-\lambda_b} | k \in \mathscr{N}^+$ **then**
  8:     **return true**
  9:    **end if**
10:   **end if**$\{y'$ is not reached by cycle $\zeta\}$
11:   **if** isinfinite(ζ) **then**
12:    **return false**
13:   **else**
14:    $x \leftarrow$ exitPoint(ζ)
15:   **end if**
16: **end if**{cycle detected}
17: **end while**
18: **return** $y' \in \tau(\mathscr{H}, x_0)$

Theorem 3.8 *Let $\mathscr{H} = (\mathscr{P}, f)$ be a multi-linear system. Then it is decidable whether $\mathscr{H}$ is a λ-system.*

Proof By Lemma 2.7, the set $\overline{\Sigma}(\mathscr{H})$ is finite. By Lemmas 3.5, 3.6 and 3.7 it is sufficient to perform the following steps: 1) For each $\sigma \in \overline{\Sigma}$ to choose two distinct τ_1 and τ_2 such that $\sigma(\tau_1) = \sigma^{\overline{\star}}$ and $\sigma(\tau_2) = \sigma^{\overline{\star}}$. 2) To compute three consecutive points of intersection of τ_1 and τ_2 with $e \in \sigma$. 3) To check whether these points are in one line.

 We have shown that if each cycle is a λ-cycle then the given system is a λ-system. The algorithm to check whether a (hyper)cycle is infinite is presented in the next section and it is an extension of the 2-dimensional case from [12].

4 Algorithm for Point-to-Point Reachability

 First we need to introduce some further notations. Let $e = \mathrm{Bd}(x) | x \in \mathscr{R}^3$ denote the border element e such that $x \in P_i \cap e$ for some partition i and $e \in \mathrm{Bd}(P_i)$. An edge e is given in the form of

$$e = \{v + \kappa \cdot \vec{u} | v, \vec{u} \in \mathscr{R}^3; l \leq \kappa \leq h : l, h \in \mathscr{R}\}.$$

Furthermore, given a cycle $\zeta = (x_i^j, \ldots, x_{i+s}^j)_{j=1}^k$, let x_b^a denote the point reached by the cycle in the ath iteration on border element $\mathrm{Bd}(x_b)$. Then $\{x_b^1 + t \cdot \overrightarrow{x_b^1 x_b^2}\}$ for $t \in \mathscr{R}$ is the line through the trajectory points on $\mathrm{Bd}(x_b)$ (λ-line), called a trace in the following.

 Algorithm 1 decides for a ML system $\mathscr{H}$ and a starting point $x_0 \in \mathscr{R}^3$ whether a point $y \in \mathscr{R}^3$ can be reached by a trajectory $\tau(\mathscr{H}, x_0) = x_0, x_1, \ldots, y$ through $\mathscr{H}$. The algorithm is allowed to perform $n \in \mathscr{N}$ simulations of $\mathscr{H}$, note that k can be much larger than n as our experiments will show.

Algorithm 2 INFINITY TEST

Input: cycle $\zeta = (x_i^j, \ldots, x_{i+s}^j)_{j=1}^3$
Output: ζ is infinite cycle

1: **for all** $x_b^j \in \zeta$ **do**
2:     **for all** $e \in \mathrm{Bd}(P_b)$ **do**
3:        **if** $e \cap \{x_b^1 + t \cdot \overrightarrow{x_b^1 x_b^2} | t \in \mathscr{R}^+\} \neq \emptyset$ **then** {trace intersects e}
4:          **if** $\lambda_b \geq 1$ **then**
5:            **return false**
6:          **else**
7:            **if** $x_b^\infty = x_b^1 + \frac{1}{1-\lambda_b} \cdot \overrightarrow{x_b^1 x_b^2} \notin P_b$ **then**
8:              **return false**
9:            **end if**
10:          **end if**
11:        **end if**
12:     **end for**
13: **end for**
14: **return true**

While the maximum number of evaluations is not reached, $\mathscr{H}$ is simulated stepwise until either y is reached or a cycle is detected. In our implementation we use the cycle detection algorithm due to Brent [4] which requires $O(\mu + \lambda)$ system evaluations.[1]

If a cycle $\zeta = (x_i^j, \ldots, x_{i+s}^j)_{j=1}^k$ is detected, several cases have to be distinguished:

a) $\mathrm{Bd}(y')$ is *not* part of the cycle.[2] If ζ is infinite according to Algorithm 2, y will never be reached.

b) $\mathrm{Bd}(y') = \mathrm{Bd}(x_b)$ for some $b \in [i, i+s]$ and hence is element of the cycle. In this case it needs to be checked whether $\exists a \in \mathscr{N}$ such that $y' = x_b^a$. If so then y' should be in

$$\{x_b^1 + t \cdot \overrightarrow{x_b^1 x_b^2}\}$$

for some $t_y \in \mathscr{R}^+$. For all x_b^a,

$$t_a = \sum_{i=1}^a \lambda_b^i = \frac{1 - \lambda_b^i}{1 - \lambda_b},$$

therefore

$$a_y = \frac{\log(1 - t_y(1 - \lambda_b))}{\log(\lambda_b)}$$

is in $\mathscr{N}^+$ iff y' is reached by a cycle iteration. If y' is not reached by ζ and the cycle is infinite, y is never reached by $\mathscr{H}$ from x_0.

c) y' is not reached by ζ and the cycle is *finite*. We calculate the point x_e where ζ is abandoned according to algorithm 3 and continue simulation and cycle detection there.

---

[1] μ denoting the first occurrence of the cycle, λ indicating the cycle length.
[2] y' is the border element reached from y by simulating $\mathscr{H}$, see algorithm 1 step 1.

Algorithm 3 EXIT POINT

Input: cycle $\zeta = (x_i^j, \ldots, x_{i+s}^j)_{j=1}^3$
Output: point $x_e \in \mathcal{R}^3$ where cycle ζ is abandoned

1:   $PQ \leftarrow$ PriorityQueue : $\mathcal{N} \times \mathcal{R}^3$
2:   **for all** $x_b^j \in \zeta$ **do**
3:     **if** $\lambda_b < 1 \vee x_b^\infty \in P_b$ **then**
4:       **skip**
5:     **end if**
6:     **for all** $e \in \mathrm{Bd}^e(P_b)$ **do**
7:       $t_e \leftarrow \dfrac{(\overrightarrow{x_b^1 v} \times \vec{u}) \cdot (\overrightarrow{x_b^1 x_b^2} \times \vec{u})}{||\overrightarrow{x_b^1 x_b^2} \times \vec{u}||^2}$
8:     **end for**
9:     $t \leftarrow \min_{e \in \mathrm{Bd}^e(P_b)} t_e$
10:   **if** $\lambda_b \neq 1$ **then**
11:     $n \leftarrow \lfloor \frac{\log(1 - t(1 - \lambda_b))}{\log(\lambda_b)} \rfloor$
12:     $t \leftarrow \frac{1 - \lambda_b^n}{1 - \lambda_b}$
13:   **else**
14:     $t \leftarrow n \leftarrow \lfloor t \rfloor$
15:   **end if**
16:   $x_e \leftarrow x_b^1 + t \cdot \overrightarrow{x_b^1 x_b^2}$
17:   $PQ.\mathrm{put}(n, x_e)$
18: **end for**
19: **return** $PQ.\mathrm{pop}()$

    The infinity test (Algorithm 2) checks for every partition P_b in cycle ζ whether the trace line $\{x_b^1 + t \cdot \overrightarrow{x_b^1 x_b^2}\}$ intersects with some $e \in \mathrm{Bd}(P_b)$. If a intersection point $x_{IS_{b,e}}$ exists and $\lambda_b \geq 1$ then ζ must abandon P_b after some number of iterations and therefore can not be infinite. If no intersection occurs or $\lambda_b < 1$ and the convergence point

$$x_b^\infty = x_b^1 + \sum_{i=1}^\infty \lambda_b^i \cdot \overrightarrow{x_b^1 x_b^2} = x_b^1 + \frac{1}{1 - \lambda_b} \cdot \overrightarrow{x_b^1 x_b^2}$$

lies before $x_{IS_{b,e}}$ on the trace line, P_b is never abandoned. If no partition within the cycle is ever abandoned then ζ is infinite.

    For a finite cycle ζ, Algorithm 3 determines the exit point x_e from ζ. Recall that an edge can be represented as $e = \{v + \kappa \cdot \vec{u} | l \leq \kappa \leq h\}$. For every $e \in \mathrm{Bd}(P_b)$ the intersection point $x_{IS_{b,e}}$ with the trace line is determined.

    The intersection point with the smallest distance t to x_b^1 is the exit point to P_b. The number of cycle iterations fully contained in P_b is given by

$$n_b = \left\lfloor \frac{\log(1 - t(1 - \lambda_b))}{\log(\lambda_b)} \right\rfloor$$

with point

$$x_{e,b} = x_b^1 + \frac{1 - \lambda_b^n}{1 - \lambda_b} \cdot \overrightarrow{x_b^1 x_b^2}.$$
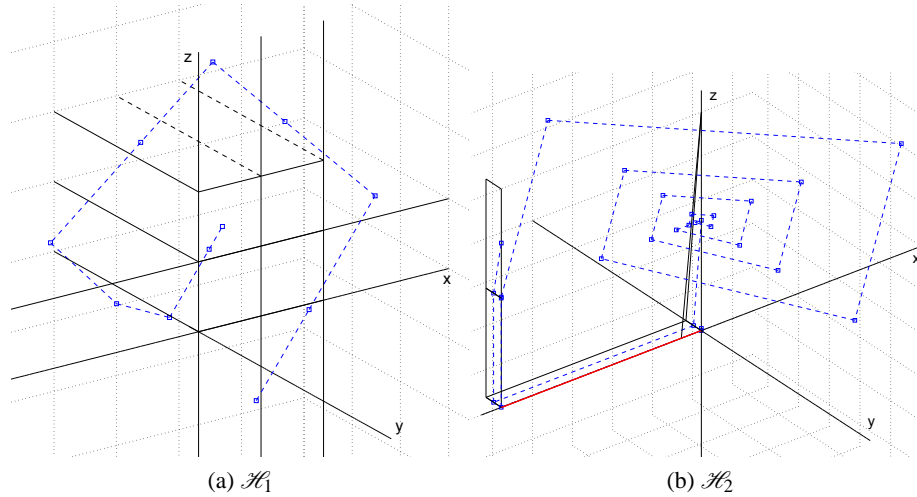
(a) $\mathscr{H}_1$ (b) $\mathscr{H}_2$

Figure 5: Example ML systems with trajectories. The convergence line of $\mathscr{H}_2$ is highlighted in red.

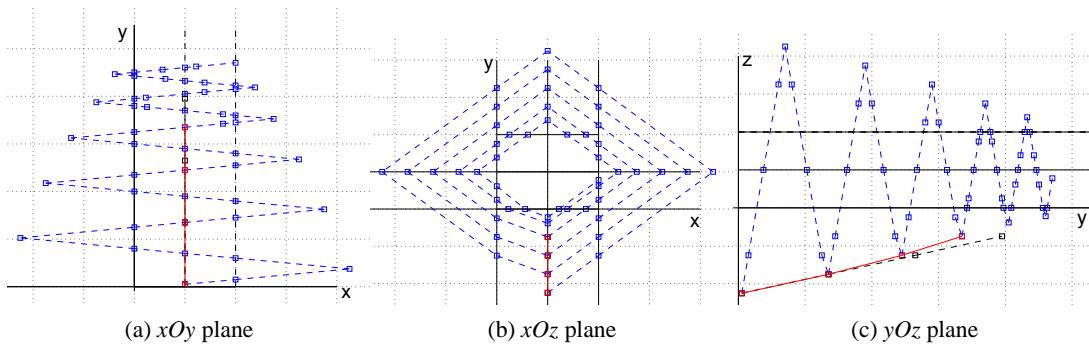(a) *xOy* plane   (b) *xOz* plane   (c) *yOz* plane

Figure 6: Projections of trajectory in $\mathscr{H}_1$. In c), the dashed black line indicates the trajectory if the points were to be on one line.

Special consideration is given to $\lambda_b = 1$, refer to Algorithm 3. The overall exit point to cycle ζ can therefore be determined by $x_e = x_{e,k}$ with $k = \mathrm{argmin}_{b \in [1,t]} n_b$.

5 Experiments

The experiments were performed on a server with two dual-core 2.8 GHz CPUs and 3 GB main memory under RedHat Linux. We implemented the algorithms in Java using the JAMA library for linear algebra operations [8]. The source code of our implementation is available at [13].

Two sample ML systems were used in our experiments and are depicted in Figure 5 alongside a sample trajectory for each. The details of the examples can be found in [14].

All partitions in $\mathscr{H}_1$ are unbounded in the Oy dimension. The *xOz* plane is divided into four inner partitions and 8 outer ones. In the inner partitions the trajectory "rotates" around the center with increasing radius, whereas the radius decreases in the outer partitions. Not all choices of **y** result in a λ-system. Note that in figure 6 the points of a trace through one border element (depicted in red) lie on one line for 2 projections (*xOy* and *xOz*), but violate the line criterion for the *yOz* projection.
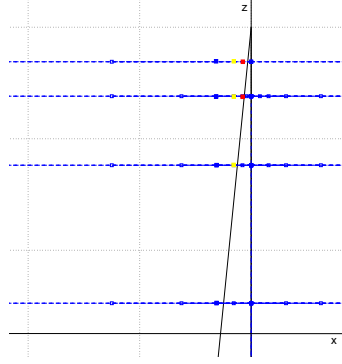
Figure 7: Projection on xOz plane of trajectory in $\mathscr{H}_2$. With each iteration of the hyper cycle, more iterations of the simple cycle through the four 3 dimensional polyhedra are required to reach P_1. Therefore no infinite simple cycle exits in $\mathscr{H}_2$.

System $\mathscr{H}_2$ consists of 5 two dimensional polygons and 4 unbounded three dimensional polyhedra. In the unbounded regions the system "rotates" around the Oz axis in ever shrinking circles until it reaches P_1. $\mathscr{H}_2$ then traverses all two dimensional polygons until it reaches P_5 where it is ejected into to the unbounded space again. With each iteration more and more rotations are required in the unbounded space to reach P_1 and the trajectory through P_3 converges towards the Ox axis. Therefore the system never reaches an infinite simple cycle as illustrated in figure 7.

In $\mathscr{H}_1$ placing the initial point x_0 at any distance n from the inner four partitions results in $\Theta(n)$ simulation steps until the inner portion is reached. Our algorithm reduces the complexity to $O(\mu + \lambda)$ in general, considering $\mathscr{H}_1$ even to $O(1)$. Only three cycle iterations are required to calculate λ_b for each $x_b \in \zeta = (x_i, \ldots, x_{i+s})^{\mp}$ and determine the exit point of the cycle. Experimental data is shown in table 1 and figure 8a. We attribute the decrease in running time of our algorithm in the first two iterations to the Java just in time compiler, optimizing code dynamically as it is executed [6]. Thereafter the algorithm exhibits constant execution time as anticipated.

Modifying $\mathscr{H}_1$ to $\mathscr{H}_1'$ so that the rotation radius decreases in the outer partitions as well as in the inner partitions,[3] produces a convergence line for all partitions at $Cl = \{(C, \cdot, C)\}$. Simulation alone may *never* determine whether $y \in Cl$ is reached, whereas our algorithm requires again three cycle iterations of length at most 12 to determine the reachability of y.

ML system $\mathscr{H}_2$ exhibits a similar behavior. With each pass through the two dimensional partitions the number of required rotations to reach the Oz axis again increases to infinity. Therefore the reachability of y on or close to the convergence line of the system is not feasibly determined by simulation alone. If y is reached after n hyper cycle iterations, at least $\Theta(n^2)$ simulation steps were required. Our algorithm reduces the complexity to $O(n(\mu + \lambda))$. With improved hyper cycle handling the complexity ought to be further reduced to $O(c \cdot (\mu + \lambda))$, since three passes of the simple cycle to reach the Oz axis suffice to determine the convergence line of the hyper cycle. Again, experimental data is shown in table 1 and figure 8b. The data exhibits the same behavior as for $\mathscr{H}_1$ regarding the Java just in time compilation. Due to its position $x_0 = (10^8, 10^8, C - C/10^8)$ reaches P_2 before P_1 and therefore requires fewer simulation steps than $x_0 = (10^7, 10^7, C - C/10^7)$.

---

[3]Specifically setting c_1 to $(\frac{1}{2}, y_1, -1)$.

$x_0 = (5, 0, -x)$	**Steps**	**Simulation**	**PTPR**	$x_0 = (x, x, C - C/x)$	**Steps**	**Simulation**	**PTPR**
x				x			
10^1	10	10	24	10^1	10	10	12
10^2	154	98	18	10^2	59	20	13
10^3	1594	1421	6	10^3	83	17	3
10^4	15994	2082	6	10^4	111	23	4
10^5	159994	22609	5	10^5	135	26	4
10^6	1599994	188276	5	10^6	149	30	3
10^7	15999994	1952252	5	10^7	161	33	3
10^8	159999994	19700805	7	10^8	143	29	3

(a) $\mathscr{H}_1$: y is first point to be reached in the inner four partitions.

(b) $\mathscr{H}_2$: $y = (0, 0, C - C/x)$.

Table 1: Comparison of our algorithm to pure simulation to decide reachability of y given x_0. Column *steps* lists the number of simulation steps required to reach y. Columns *simulation* and *PTPR* give the time in [ms] required to decide reachability by simulation and our algorithm, respectively.

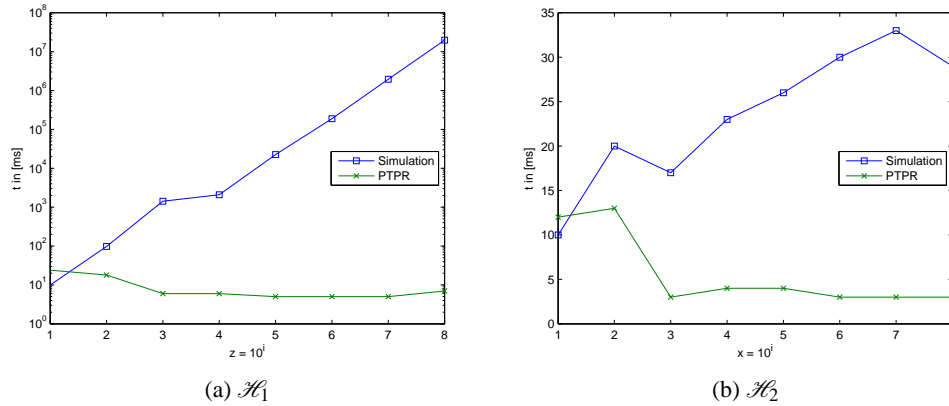(a) $\mathscr{H}_1$ (b) $\mathscr{H}_2$

Figure 8: Comparison of our algorithm (*PTPR*) to pure simulation to decide reachability of y given x_0.

6   Conclusions

The complexity of safety critical systems has increased dramatically over last decades. The safety properties of such systems can often not be checked exactly either due to theoretical boundaries or due to too large computational efforts required. One of the drawbacks of recent techniques is that too little attention is paid to the geometric properties of the systems under analysis.

A hybrid system (a hybrid automaton) is a formalism that can be used for modeling safety critical systems. ML systems constitute a rather simple class of hybrid systems but yet they are on the boundary of decidable and undecidable systems.

ML systems have certain properties that make them very suitable for a topological analysis. We have shown that on the one hand there are systems with acyclic behavior, and on the other hand if some properties of 2-dimensional systems hold in three dimensions then it is possible to answer exactly whether a point y is reachable from a point x. We have presented a prototype implementation of our approach for solving the reachability problem for a subclass of multi-linear systems which we have called

λ-systems. We compared our approach with simulation. The results suggest, that using geometrical properties of the systems can lead to orders of magnitude more efficient techniques than simulation. As soon as our algorithm detects a cycle (or a hypercycle) for which the infinity criterion holds, the algorithm requires constant number of steps. While the number steps for simulation grows exponentially with the distance between points. Also our algorithm can lead to more exact computations because of less rounding errors during the computation.

References

[1] R. Alur & D. L. Dill (1994): *A Theory of Timed Automata*. Theor. Comput. Sci. 126(2), pp. 183–235, doi:`10.1016/0304-3975(94)90010-8`.

[2] E. Asarin, O. Maler & A. Pnueli (1995): *Reachability Analysis of Dynamical Systems Having Piecewise-Constant Derivatives*. Theor. Comput. Sci. 138(1), pp. 35–65, doi:`10.1016/0304-3975(94)00228-B`.

[3] E. Asarin, G. Schneider & S. Yovine (2007): *Algorithmic analysis of polygonal hybrid systems, part I: Reachability*. Theor. Comput. Sci. 379(1-2), pp. 231–265, doi:`10.1016/j.tcs.2007.03.055`.

[4] R. P. Brent (1980): *An improved Monte Carlo factorization algorithm*. BIT Numerical Mathematics 20, pp. 176–184, doi:`10.1007/BF01933190`. 10.1007/BF01933190.

[5] M. Broucke (1999): *A geometric approach to bisimulation and verification of hybrid systems*. In: *Hybrid Systems: Computation and Control, volume 1569 of Lecture Notes in Computer Science*. Springer-Verlag, pp. 61–75, doi:`10.1007/3-540-48983-5_9`.

[6] T. Cramer, R. Friedman, T. Miller, D. Seberger, R. Wilson & M. Wolczko (1997): *Compiling Java just in time*. Micro, IEEE 17(3), pp. 36–43, doi:`10.1109/40.591653`.

[7] T. A. Henzinger, P. W. Kopke, A. Puri & P. Varaiya (1995): *What's decidable about hybrid automata?* In: *STOC*. pp. 373–382, doi:`10.1145/225058.225162`.

[8] J. Hicklin, C. Moler, P. Webb, R.F. Boisvert, B. Miller, R. Pozo & K. Remington (2000): *Jama: A Java matrix package*. Website. Available at `http://math.nist.gov/javanumerics/jama`.

[9] G. Lafferriere, G. J. Pappas & S. Sastry (2000): *O-Minimal Hybrid Systems*. Math. Control Signals Systems 13, pp. 1–21, doi:`10.1007/PL00009858`.

[10] G. Lafferriere, G. J. Pappas & S. Yovine (1999): *A New Class of Decidable Hybrid Systems*. In: *HSCC*. pp. 137–151, doi:`10.1007/3-540-48983-5_15`.

[11] G. Lafferriere, G.J. Pappas & S. Yovine (1999): *Reachability Computation For Linear Hybrid Systems*. In: *Proceedings of the 14th IFAC World Congress*. E, pp. 7–12.

[12] O. Maler & A. Pnueli (1993): *Reachability Analysis of Planar Multi-linear Systems*. In: *CAV*. pp. 194–209, doi:`10.1007/3-540-56922-7_17`.

[13] O. Tveretina & D. Funke (2011): *3-Dimensional Multi-Linear Systems*. Website. Available at `http://www.stud.uni-karlsruhe.de/~uucom/ml_systems/index.html`.

[14] O. Tveretina & D. Funke (2011): *Deciding Reachability for a Subclass of 3-Dimensional Multi-Linear Systems*. Technical Report 2011-19, Karlsruhe Institute of Technology.