Reachability Analysis of Complex Planar Hybrid Systems

Hallstein A. Hansen*

Faculty of Technology, Buskerud University College, Norway

Gerardo Schneider

Dept. of Computer Science and Engineering, Chalmers | University of Gothenburg, Sweden, and Dept. of Informatics, University of Oslo, Norway

Martin Steffen

Dept. of Informatics, University of Oslo, Norway

Abstract

Hybrid systems are systems that exhibit both discrete and continuous behavior. Reachability, the question of whether a system in one state can reach some other state, is undecidable for hybrid systems in general. In this paper we are concerned with GSPDIs, 2-dimensional systems generalizing SPDIs (planar hybrid systems based on "simple polygonal differential inclusions"), for which reachability have been shown to be decidable. GSPDIs are useful to approximate 2-dimensional control systems, allowing the verification of safety properties of such systems.

In this paper we present the following two contributions: i) An optimized algorithm that answers reachability questions for GSPDIs, where all cycles in the reachability graph are accelerated. ii) An algorithm by which more complex planar hybrid automata are over-approximated by GSPDIs subject to two measures of precision. We prove soundness, completeness, and termination of both algorithms, and discuss their implementation.

Keywords: hybrid systems, reachability checking, safety verification, non-linear systems, differential inclusions

1. Introduction

Hybrid automata is a well-known formal model for *hybrid systems,* i.e., systems which combine discrete and continuous behavior. Traditionally, their con-

*Corresponding author

Email addresses: hallsteinh@hibu.no (Hallstein A. Hansen), gersch@chalmers.se (Gerardo Schneider), msteffen@ifi.uio.no (Martin Steffen)

tinuous part is described by *differential equations*, or more generally by differential *inclusions*, capturing the system's evolution over time. The discrete part usually corresponds to switches between different *modes*, where each mode is characterized by differential inclusions.[1] Many interesting physical systems can be modelled by hybrid automata. One prominent example are control systems [1] where a controller device with discrete states affects the system, e.g., a plant, to assure that it adheres to given requirements. A simple thermostat is a typical control system where there is a discrete change between two modes, each modelled by specific differential inclusions: one mode representing the heater being turned *on*, and one for being *off*. For such systems, we are interested in *reachability*: starting in an initial state, can the system reach a given target state? Often, one is interested in whether some undesirable configuration is reachable; if not, the system is called *safe*. E.g., one may require that, when starting from a room temperature less than 30 degrees, the temperature never exceeds 30 degrees.

For general hybrid automata, the verification problem for many classes of properties, including reachability, is known to be undecidable [2]. Hence, various restrictions of hybrid systems have been proposed and investigated. In this paper we deal with planar hybrid systems, in particular with so-called *Generalized Polygonal Hybrid Systems* (GSPDIs)[2] [3]. A GSPDI consists of a finite partition of the plane into polygonal regions, each governed by a specific differential inclusion. For that model, we are not merely interested in that reachability is decidable, but to obtain an algorithm efficient enough to be used in practice. Secondly, we use GSPDIs to approximate more complex planar systems.

As mentioned, reachability for GSPDIs is decidable and [3] gives an algorithm with a doubly exponential complexity. The use of differential *inclusions* renders the behavior of GSPDIs non-deterministic and their reachability graphs in general include many complex, non-simple cycles. Though reachability searches can be optimized by considering only simple cycles and furthermore by using *acceleration* so that many such cycles can be analyzed without iteration, many of them still need to be iterated. Moreover, to be exhaustive, the search needs to analyze *all* possible cycles in the worst case. In fact, to prevent excessive iteration, earlier implementations of the reachability algorithm in the GSPeeDI tool only generate cycles as a last resort [4]. To make the approach more feasible in practice, it would be desirable to accelerate all the cycles, thus further reducing the time complexity. Though this is not possible in general for hybrid systems we will show later that we can indeed accelerate all simple

[1] As differential inclusions subsume differential equations as a particular case we will in general use the first term unless it is necessary to make the distinction.

[2] GSPDIs came first as an abbreviation for *Generalized SPDIs*, where the term SPDI was originally the acronym of *Simple Polygonal Differential Inclusions system*. For historical reason SPDIs where later called *Polygonal Hybrid Systems*. To keep with the later literature (only the first paper on the topic used 'Simple Polygonal Differential Inclusions') in what follows we will use the original acronyms (G)SPDIs meaning (Generalized) Polygonal Hybrid Systems.

cycles in GSPDIs.

Though not many real systems can directly be expressed as GSPDIs, there has been a theoretical interest in their study as GSPDIs are a class of hybrid automata lying on the border between decidability and undecidability [5, 6]. With reachability being decidable, one can use GSPDIs to over-approximate other systems, and since the underlying continuous dynamics of GSPDIs is quite rich, one still may obtain a realistic model which allows to derive properties for the underlying concrete system. Conservatively over-approximating a system gives a semi-decision procedure for reachability: If unreachable in the approximating GSPDI, the corresponding state is unreachable in the underlying system as well, while reachability in the abstract GSPDI does not in general imply reachability in the concrete system. In such an inconclusive outcome it is possible to use series of automatic refinements to get better, i.e., more precise approximations. Moreover, GSPDIs have been used to *approximate* differential equations [7], and algorithms and tools have been developed for that purpose [3, 4, 8].

This paper is a revised and extended version of the earlier papers [8, 7, 9]. Besides including the full proofs and more examples, we present the following new results: i) A reachability algorithm for GSPDIs which avoids iterating cycles more than twice. We prove that the algorithm is sound, complete, and that it terminates. This result dramatically reduces the complexity of the algorithm and, to our knowledge, there are no other similar results in the formal analysis of hybrid automata. ii) An implementation of the algorithm as part of the tool GSPeeDI, and showing empirical evidence of how cycle acceleration results in increases in performance.

The rest of the paper is organized as follows. Section 2 gives the mathematical background needed for the rest of the paper, while Section 3 includes previously known results pertaining to GSPDIs. Section 4 describes the new reachability algorithm for GSPDIs and proves that arbitrary cycle iterations can be avoided. We also describe the tool GSPeeDI [10], implementing the reachability algorithm for GSPDIs, and a semi-decision procedure for the reachability analysis of differential equations. In Section 5 we present an algorithm to over-approximate a particular class of nonlinear control systems using GSPDIs, proving that this over-approximation is a semi-decision procedure. We discuss related work in Section 6 and we conclude in Section 7 with directions for future work.

2. Mathematical preliminaries

We start with a short reminder of facts about Euclidean geometry on the plane, and two-dimensional vectors. Then we present differential equations and inclusions, and control systems. We proceed by introducing hybrid automata, a useful model for hybrid systems. We finish this section with truncated affine multi-valued functions, the underlying functions for computing reachability of GSPDIs, which are introduced in the next section.

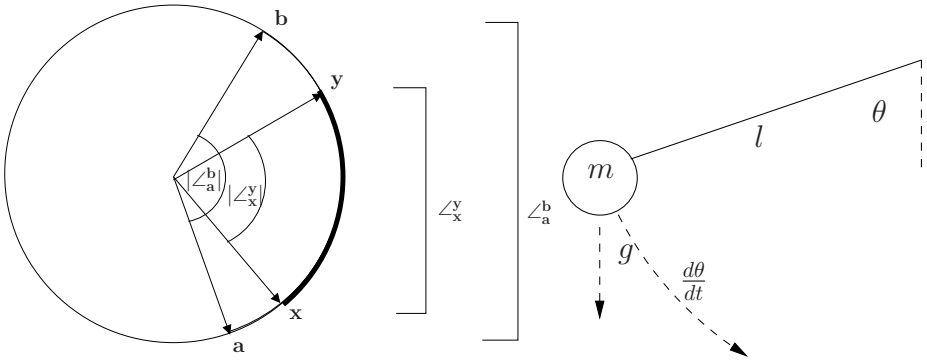

Figure 1: Left: A unit circle, illustrating angles and arcs. Right: A damped pendulum.

2.1. Vectors and planar geometry

In the following we assume that, unless stated otherwise, vectors are normalized so that two vectors are equal if and only if their directions are equal. A number of the results in this paper will concern points and intervals on the unit circle as a consequence.

Definition 1 (Unit circle and arcs)**.** *The* unit circle $\mathbb{T} = \{(x, y) \mid x \in \mathbb{R}, y \in \mathbb{R}, \sqrt{x^2 + y^2} = 1\}$ *is the circle with center at the origin and radius* 1 *(see Figure 1). For a normalized, non-zero vector* $\mathbf{x}$ *we have that* $\mathbf{x} \in \mathbb{T}$*. An* arc*, written* $\angle^{\mathbf{b}}_{\mathbf{a}}$ *or* $[\mathbf{a}, \mathbf{b}]$*, is a closed segment of the unit circle starting in endpoint* $\mathbf{a}$ *and, moving in the counter-clockwise direction, terminating in endpoint* $\mathbf{b}$*. The length of an arc, written* $|\angle^{\mathbf{b}}_{\mathbf{a}}|$ *is the* angle *between* $\mathbf{a}$ *and* $\mathbf{b}$*, measured in the interval* $[0, 2\pi)$*. We write* $\mathbf{x} \in \angle^{\mathbf{b}}_{\mathbf{a}}$ *if vector* $\mathbf{x} \in \mathbb{T}$ *and when starting in endpoint* a *and moving in counter-clockwise direction we first reach* x *before* b*. We write* $\angle^{\mathbf{y}}_{\mathbf{x}} \subseteq \angle^{\mathbf{b}}_{\mathbf{a}}$ *if, when starting in endpoint* a *and moving in counter-clockwise direction, we first reach* x *before* y *before* b*, and* $\angle^{\mathbf{y}}_{\mathbf{x}} \subset \angle^{\mathbf{b}}_{\mathbf{a}}$ *if the inclusion is proper.*

2.2. Differential inclusions

Differential equations are an important mathematical tool for modeling, simulating, and analyzing physical phenomena. They describe the relationship between the value of some physical quantitiy such as position, velocity, temperature, etc., and their rate of change with respect to time. A differential equation does not include any representation of uncertainty in the model. The generalization to differential inclusions allows to model systems whose behavior is non-deterministic or not exactly known.

Definition 2 (Differential equation)**.** *An* ordinary first-order differential equation *(ODE) relates a function* $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ *to its derivative* $\frac{dx}{dt}$*, expressed as*

$\frac{dx}{dt} = f(t, x(t))$, *where* $t \in \mathbb{R}_{\geq 0}$ *is often interpreted as time and* $x(t)$ *is the value of* x *at* t. *A* system *of first-order ODEs relates several functions* $x_1, \ldots, x_n$ *to their derivatives* $\frac{dx_1}{dt}, \ldots, \frac{dx_n}{dt}$*:*

$$\begin{aligned} \frac{dx_1}{dt} &= f_1(t, x_1(t), \ldots, x_n(t)) \\ &\vdots \\ \frac{dx_1}{dt} &= f_n(t, x_1(t), \ldots, x_n(t)) \end{aligned}$$

An n*-th order ODE can be transformed into a system of* n *first-order ODEs. The system is* linear *if the functions* x *appear to the power of one,* non-linear *otherwise.*

Example 1 (Pendulum)**.** *A damped pendulum of mass* m*, length* l*, and gravitational acceleration* g*, see Figure 1, can be modeled as a second-order non-linear ODE relating the angle* $\theta(t)$*, angular velocity* $\frac{d\theta}{dt}$*, and angular acceleration* $\frac{d^2\theta}{dt^2}$ *of the pendulum. The damping by friction is represented by a constant* c*:*

$$\frac{d^2\theta}{dt^2} = -\frac{c}{ml}\frac{d\theta}{dt} - \frac{g}{l}\sin\theta(t) \ .$$

General non-deterministic behavior is meant to be captured by differential equations. Systems with uncertainties or perturbations are inherently non-deterministic as there are many possible evolutions for a given initial state. This requires a more general definition.

Definition 3 (Differential inclusion [11])**.** *A* differential inclusion system *is of the form*

$$\begin{aligned} \frac{dx_1}{dt} &\in F_1(t, x_1(t), \ldots, x_n(t)) \\ &\vdots \\ \frac{dx_1}{dt} &\in F_m(t, x_1(t), \ldots, x_n(t)) \end{aligned}$$

where $F_i(t, x_1(t), \ldots, x_n(t))$ *is a subset of elements from* $\mathbb{R}^n$*.*

The parameter t, usually representing time, does not necessarily have to be an independent variable. A system given by $y(t) = f(t, x(t))$ is *time-invariant* if the system state with time-shifted input $f(t, x(t + \delta t))$ is equal to the system state with time-shifted output $y(t+\delta t) = f(t+\delta t, x(t+\delta t))$, that is $f(t, x(t+\delta t) = f(t + \delta t, x(t + \delta t))$ for all t and δt.

Example 2 (Pendulum)**.** *The damped pendulum from Example 1 is deterministic and described by a differential equation. If we now let the damping vary a little by substituting the coefficient* c *by* $c+e$ *where* e *is drawn non-deterministically from some interval* $E \subseteq \mathbb{R}$*, we get:*

$$\frac{d^2\theta}{dt^2} \in \{-\frac{c+e}{ml}\frac{d\theta}{dt} - \frac{g}{l}\sin\theta(t) \mid \ e \in E\} \ .$$

Later we will work with a particular class of differential inclusion systems, defined next.

Definition 4 (Time-invariant differential inclusion system (TIDIS))**.** *Let E be a subset of $\mathbb{R}$. Let $x(t), y(t) \in \mathbb{R}$ be state variables of the (unknown) functions x and y, and f and g be first order, time-invariant, possibly non-linear ODEs in x and y. We define the differential inclusions F and G as $F(x(t), y(t)) = \{f(x(t), y(t), e) \mid e \in E\}$ and $G(x(t), y(t)) = \{g(x(t), y(t), e) \mid e \in E\}$. A* time-invariant differential inclusion system *(TIDIS) is a tuple $\mathcal{S} = \langle Q,\ F,\ G \rangle$, where the domain $Q \subseteq \mathbb{R}^2$ is a convex polygon. Furthermore*

$$\frac{dx}{dt} \in F(x(t), y(t)) \quad \textit{and} \quad \frac{dy}{dt} \in G(x(t), y(t))$$

where $(x(t), y(t)) \in Q$.

The possible *behaviors* or *dynamics* of a TIDIS $\mathcal{S}$ at a given point $(x(t_i), y(t_i))$ is the set of vectors $F(x(t_i), y(t_i)) \times G(x(t_i), y(t_i)) \subseteq \mathbb{R}^2$. If $(0, 0)$ is an element of $F(x(t_i), y(t_i)) \times G(x(t_i), y(t_i))$, then $(x(t_i), y(t_i))$ is an *equilibrium point*. If $F(x(t_i), y(t_i)) \times G(x(t_i), y(t_i)) = \{(0, 0)\}$, then $\mathcal{S}$ cannot change its state from $(x(t_i), y(t_i))$.

Example 3 (Pendulum)**.** *For the damped pendulum, if we let $x = \theta$ and $y = \frac{d\theta}{dt}$ we can model the pendulum by the following TIDIS:*

$$\frac{dx}{dt} \in \{y(t)\} \quad \textit{and} \quad \frac{dy}{dt} \in \{-\frac{c+e}{ml} y(t) - \frac{g}{l} \sin x(t) \mid e \in E\}\ .$$

For reachability it is only relevant whether, not when, some point is reached. Hence the length of the behavior vectors is unimportant and we can normalize the behavior of a TIDIS as follows:

Definition 5 (Normalization)**.** *Let $\mathbb{T}$ be the unit circle. Then the* normalized behavior *of a TIDIS $\mathcal{S}$ is given by the function $N : \mathbb{R}^2 \to 2^{\mathbb{T}}$:*

$$N(x(t), y(t)) = \{\frac{f(x(t), y(t), e)}{r}, \frac{g(x(t), y(t), e)}{r}) \mid e \in E, r \neq 0\}$$

where

$$r = \sqrt{f(x(t), y(t), e)^2 + g(x(t), y(t), e)^2}.$$

Note that the function N is undefined for points where $r = 0$, i.e. where both $f(x(t), y(t), e)$ and $g(x(t), y(t), e)$ equal 0. To simplify notation we refer to the point $(x(t_i), y(t_i))$ as $p_i = (x_i, y_i)$, the set of normalized dynamics of p_i as $\widehat{p_i}$ decomposed as $\widehat{x_i}$ and $\widehat{y_i}$, and the normalized dynamic vector as $\dot{p_i} = (\dot{x_i}, \dot{y_i})$, $\dot{p_i} \in \widehat{p_i}$. See Figure 2-a) and 2-b) for an illustration.

We denote by p_i^+ and p_i^- the *upper* and *lower behavior limit vectors* of $\widehat{p_i}$, the vectors in $\widehat{p_i}$ such that for all other vectors $\dot{p_i} \in \widehat{p_i}$, we have $\dot{p_i} \in \angle_{p_i^-}^{p_i^+}$. See Figure 2-c) for a visualization. If $\widehat{p_i}$ contains one element only, then the behavior is deterministic at point p_i and $p_i^+ = p_i^-$.

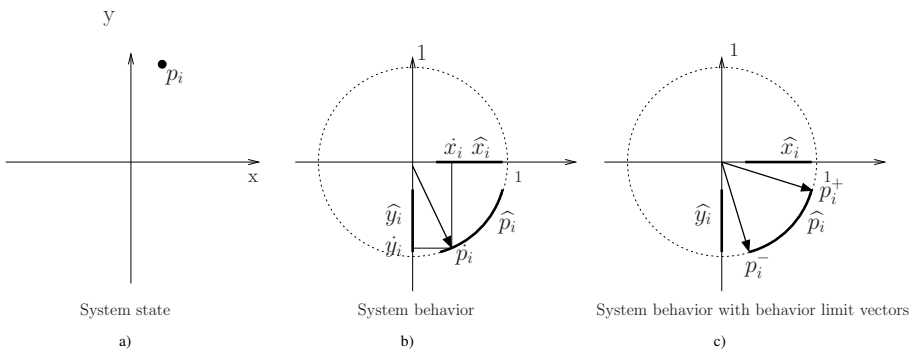

Figure 2: a) A point p_i; b) A possible future evolution, given by $\dot{p}_i$, included in the normalized dynamics $\widehat{p}_i$; c) the behavior limit vectors ($\angle_{p_i^-}^{p_i^+}$).

2.3. Lipschitz continuity

The more rapidly a system changes, the less precise the analysis may be and the more costly it is to analyze its behavior to obtain a given precision. Systems are called *locally Lipschitz continuous* if there exist areas with a finite upper bound on the change of behavior, and this corresponding bound is a measure of the state change. For this purpose the following definition provides a way of comparing two arcs on the unit circle.

Definition 6 (Behavior distance)**.** *Let* $A = [\underline{a}, \overline{a},]$ *and* $B = [\underline{b}, \overline{b}]$ *be arcs on the unit circle. Then the* behavior distance $d[A, B]$ *is defined as* $|\overline{a} - \overline{b}| + |\underline{a} - \underline{b}|$.

Lemma 1 (Metric)**.** *The behavior distance is a metric.*

Proof. A metric must obey the following 4 conditions:

1. $d[A, B] \geq 0$ (Non-negativity)
2. $d[A, B] = 0 \Leftrightarrow A = B$ (Identity)
3. $d[A, B] = d[B, A]$ (Symmetry)
4. $d[A, C] \leq d[A, B] + d[B, C]$, for all arcs B (Triangle inequality)

The conditions are easy to check: non-negativity holds as the distance is defined as the sum of two absolute values. The fact $|\overline{a} - \overline{b}| + |\underline{a} - \underline{b}| = 0$ iff $\overline{a} = \overline{b}$ and $\underline{a} = \underline{b}$ gives the second condition. Symmetry follows directly from the definition of absolute values. The triangle inequality holds on $\mathbb{R}$. Thus $|\overline{a} - \overline{c}| \leq |\overline{a} - \overline{b}| + |\overline{b} - \overline{c}|$ and $|\underline{a} - \underline{c}| \leq |\underline{a} - \underline{b}| + |\underline{b} - \underline{c}|$ yields $|\overline{a} - \overline{c}| + |\underline{a} - \underline{c}| \leq |\overline{a} - \overline{b}| + |\underline{a} - \underline{b}| + |\overline{b} - \overline{c}| + |\underline{b} - \underline{c}|$. □

With a metric for the image of the normalized dynamics $\widehat{p}$ of a point, we define what it means for the normalized behavior of a TIDIS to be Lipschitz continuous.

Definition 7 (Lipschitz continuity)**.** *Let $P \subseteq \mathbb{R}^2$ be a convex polygon. A function $f : \mathbb{R}^2 \to 2^{\mathbb{T}}$ is* Lipschitz continuous *(or just* Lipschitz *for short) on P if there exists a constant $K \in \mathbb{R}$ such that for all points p_i and p_j in P,*

$$d[f(p_i), f(p_j)] \leq K\|p_i - p_j\|,$$

where $d[\cdot]$ is a metric.

Example 4. *The normalized behavior of the (deterministic) damped pendulum given by*

$$\frac{dx}{dt} = y(t) \quad \textit{and} \quad \frac{dy}{dt} = -0.25y(t) - \sin x$$

is not *Lipschitz continuous at the origin. Consider, e.g., the behavior at the points $(x, 0)$ and $(-x, 0)$ and let x approach 0. Hence $\|(x, 0) - (-x, 0)\|$ approaches 0, but for any x the normalized behavior at $(x, 0)$ is always $(0, -1)$, and at $(-x, 0)$ it is $(0, 1)$. For the system to be Lipschitz continuous we require $d[\widehat{p_i}, \widehat{p_j}] \leq K\|p_i - p_j\|$ for some fixed K, that is $\pi \leq K\|(x, 0) - (-x, 0)\|$ for all x. Since $\|(x, 0) - (-x, 0)\|$ can be infinitely small we can always disprove this inequality.*

2.4. Hybrid automata

We now introduce hybrid automata [12], a common mathematical model for hybrid systems, i.e., systems that exhibit both continuous and discrete behavior [13].

Definition 8 (Hybrid automata)**.** *A* hybrid automaton $\mathcal{H}$ *is a tuple* $(Loc, Var, Tra, Act, Inv, Guard, Asg)$*, where*

- $Loc = \{l_1, \ldots, l_m\}$ *is a finite set of locations.*
- $Var = \{x_1, \ldots, x_n\}$ *is a finite set of real-valued variables and $V \subseteq \mathbb{R}^n$ the set of their valuations. The* state $(l_i, x_1, \ldots, x_n)$ *of a hybrid automaton is the current location and current valuation of the variables.*
- *Inv, the invariants, is a mapping from the locations to predicates on variable valuations, i.e., $Inv(l) \subseteq V$, restricting the possible valuations of the variables.*
- $Tra \subseteq Loc \times Loc$ *is a set of transitions.*
- *The function Guard maps transitions to guards, where each guard $G_{(l,l')} \subseteq V$.*
- *Asg is a function that maps each transition to a set of assignments, where each assignment $A_{(l,l')} \subseteq G_{(l,l')} \times Inv(l')$*
- *The function Act ("activities") maps each location to a set of continuous functions of type $\mathbb{R}_{\geq 0} \to V$.*

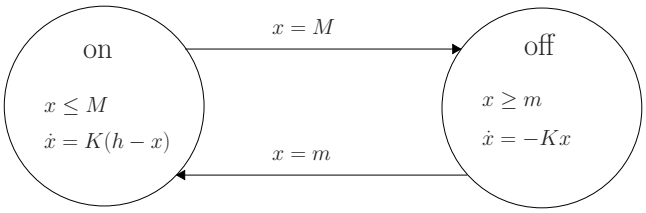

Figure 3: A thermostat.

We say that a transition is *enabled* whenever the transition may be taken (i.e., the guard is satisfied).

We assume in what follows that the activities of each location can be written as a TIDIS of the n variables of Var.

Example 5 (Thermostat)**.** *A classical example of a hybrid system is the thermostat. In Figure 3 we show a model of a thermostat as a hybrid automaton where x is the current temperature. Here $\dot{x} = K(h-x)$ and $\dot{x} = -Kx$, where the constant K is dependent on the environment and the constant h on the heater, written as two TIDISs of one equation each. The constraints $x \leq M$ and $x \geq m$ are the invariants of the* on *and* off *locations, and the transition guards are $x = M$ and $x = m$, where M is the upper bound and m the lower bound on the environment temperature.*

In general, the more expressive a class of hybrid automata is, the less properties are decidable [13]. In this paper we focus on hybrid automata subject to certain restrictions, e.g., that the interiors of the invariants of each location are disjoint. We define this using the notion of *mesh* of a polygon [14], which basically constitutes a partition of the polygon, except of for overlaps at the borders between different location invariants:

Definition 9. *A* mesh *of a convex polygon Q is a collection $\mathcal{M} = \{\mathcal{M}_l \mid l \in Loc\}$ of subsets of Q such that*

- *Each $\mathcal{M}_l$ is a convex polygon*
- $\cup_{l \in Loc} \mathcal{M}_l = Q$
- *For all $l \neq l'$, $\mathcal{M}_l \cap \mathcal{M}_{l'} = \beta(\mathcal{M}_l) \cap \beta(\mathcal{M}_{l'})$, where $\beta(\mathcal{M}_l)$ denotes the border (or boundary) of set $\mathcal{M}_l$.*

Example 6. *In Figure 4-a) the following 4 sets constitute a mesh of $[-\pi, \pi] \times [-\pi, \pi]$:*

- $\mathcal{M}_{off} = \{(x, y) \mid \neg(x < 0 \wedge y > 0)\}$
- $\mathcal{M}_{max} = \{(x, y) \mid \sqrt{x^2 + y^2} \leq \frac{1}{3} \wedge (x \leq 0 \wedge y \geq 0)\}$

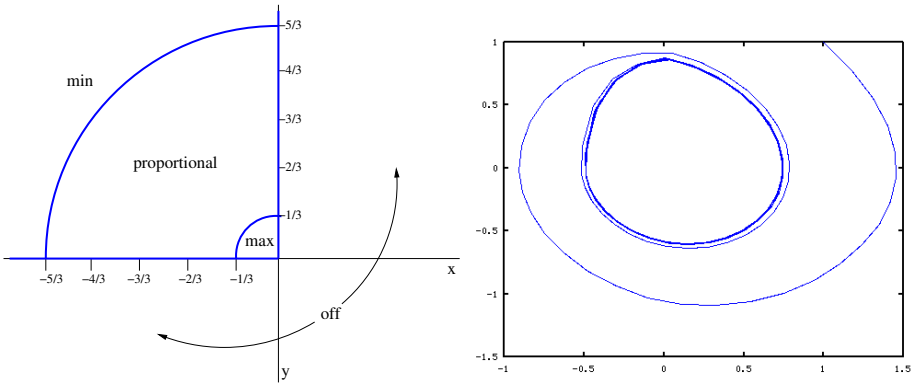

Figure 4: a) A geometric representation of the locations of a hybrid automaton modeling a pendulum with proportional control. b) Example trajectory.

- $\mathcal{M}_{proportional} = \{(x,y) \mid \sqrt{x^2+y^2} \in [\frac{1}{3}, \frac{5}{3}] \wedge (x \leq 0 \wedge y \geq 0)\}$
- $\mathcal{M}_{min} = \{(x,y) \mid \sqrt{x^2+y^2} \geq \frac{5}{3} \wedge (x \leq 0 \wedge y \geq 0)\}$

We see, for example, that $\mathcal{M}_{off} \cap \mathcal{M}_{max} = \beta(\mathcal{M}_{off}) \cap \beta(\mathcal{M}_{max}) = \{(x,y) \mid (x \in [\frac{-1}{3}, 0] \wedge y = 0) \vee (y \in [0, \frac{1}{3}] \wedge x = 0)\}$.

In this paper we limit ourselves to a subclass of hybrid automata, namely planar, non-overlapping systems without resets. Resetting a variable in a transition means to assign a (new) value to the variable when taking the transition, so in systems without resets, assignments in the transitions are represented by the identity relation. On the other hand, we consider systems with changing, non-deterministic, non-linear behavior. The precise definition of the systems is given below.

Definition 10 (Continuous non-overlapping hybrid automaton)**.** *Given a mesh* $\mathcal{M}$ *of a convex polyhedron* Q*, a* continuous non-overlapping hybrid automaton *(CN-HA)* $\mathcal{C}$ *is a hybrid automaton where*

- *For each* $l \in Loc$*,* $Inv(l) = \mathcal{M}_l$*.*
- $(l, l') \in Tra$ *if and only if* $\beta(\mathcal{M}_l) \cap \beta(\mathcal{M}_{l'}) \neq \emptyset$*.*
- *For every transition* $(l, l') \in Tra$
  - *the guard* $G_{(l,l')}$ *is given by* $\beta(\mathcal{M}_l) \cap \beta(\mathcal{M}_{l'})$*.*
  - *the assignment* $A_{(l,l')}$ *is the identity map* $\{(x,x) \mid x \in G_{(l,l')}\}$*.*

The pendulum of Figure 4, with the location invariants plotted geometrically, illustrates the restrictions of CN-HAs. The CN-HA has four locations, *min*, *max*, *proportional*, and *off*, and the location invariants are not overlapping

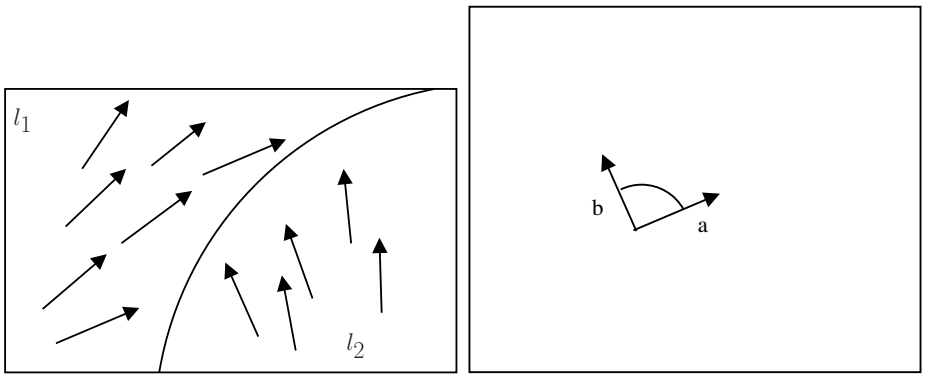

Figure 5: a) A border region with a curved border between locations l_1 and l_2, with example behavior vectors. b) The resulting behavior in the region for a hypothetical GSPDI.

except at the borders, and the trajectories are continuous. Figure 5-a) illustrates two regions l_1 and l_2 separated by a curved line representing the border between the regions. The figure exemplifies that the dynamics of l_1 and l_2 are in general independent of each other and thus not necessarily differentiable or Lipschitz continuous along the border. In this case the resulting approximation will contain the smallest arc that in turn contains the dynamics of both l_1 and l_2, see Figure 5-b). A single evolution of a hybrid automaton, a sequence of continuous evolutions in a location and discrete transitions between locations, is called a *run*:

Definition 11 (Run)**.** *The state of a hybrid automaton may change in two different ways:*

- *Discrete transitions:* $(l_i, v_i) \rightarrow (l_{i+1}, v_{i+1})$.
- *Time delay, or continuous evolution:* $(l_i, v_i) \rightarrow^{t_i} (l_i, f_i(t_i))$, *where* $f_i \in Act(l_i)$, $t_i \in \mathbb{R}_{\geq 0}$

A run *is an alternating sequence of continuous evolutions and discrete transitions:* $s_0 \mapsto_{t_0}^{f_0} s_1 \mapsto_{t_1}^{f_1} \ldots s_N$, *where* N *can be* ∞, *subject to*

- $f_i(0) = v_i$
- $f_i(t_i) \in Inv(l_i)$, *for all* $0 \leq t \leq t_i$.
- *If* $s'_i = (l_i, f(t_i))$ *is the continuous evolution of* s_i, *then we call* s_{i+1} *the discrete transition successor of* s'_i.

An example of a run of a hybrid automaton is shown in Figure 6-a). A geometric representation is used to illustrated that the continuous evolution is

given by three parameterized curves on the plane, while the discrete transitions are shown as discontinuous jumps. In the following we consider only *non-Zeno* hybrid automata, i.e., time does not converge. When investigating a CN-HA we are only interested in the continuous evolutions of the runs: The discrete transitions are restricted to the identity map, and any valuation of the variables belongs to a single location only, except for the borders. Thus we introduce the notion of a continuous *trajectory* [15, 14], see Figure 6-b).

Definition 12. *A run ρ of a hybrid automaton $\mathcal{H}$, where f_i is an activity of location l_i, has corresponding* trajectory *$\tau(I, \xi)$, where $I \subseteq \mathbb{R}_{\geq 0}$ and ξ is a continuous and almost-everywhere differentiable function $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$ if ρ satisfies:*

- $I = \sum_{i=0}^{N-1} [0, t_i]$.
- *For all $0 \leq i \leq N$ and all $t \in [0, t_i]$, $\xi(t_{i-1} + t) = f_i(t)$, letting $t_{-1} = 0$.*
- *For all $0 \leq i < N$ we have $f_i(t_i) = f_{i+1}(0)$.*

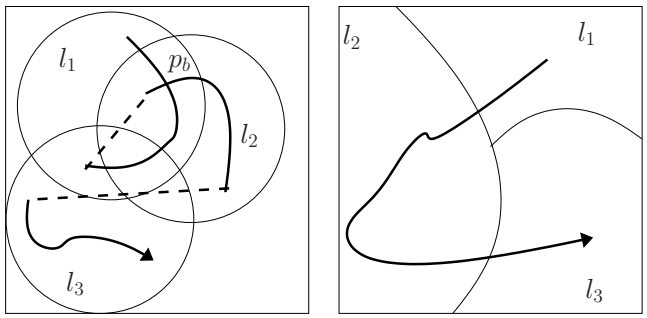

Figure 6: a) Run of a hybrid automaton, geometric representation, where the dashed lines represent discrete assignments. p_b refers to two states (l_1, x_b, y_b) and (l_2, x_b, y_b). b) Trajectory of a CN-HA.

The set of all trajectories of a hybrid automaton $\mathcal{H}$ is denoted as $[\mathcal{H}]$. For valuations x_s and x_f, a trajectory $\tau(I, \xi)$ with $\xi(0) = x_s$ and $\xi(t) = x_f$ for some $t \in I$ is denoted as $x_s \mapsto_\xi x_f$. When I is implicit, we write $\xi \in [\mathcal{H}]$. An important problem in hybrid systems is that of *reachability*, whether a state x_f can reached from an initial state x_i. Here we define reachability in terms of trajectories.

Definition 13 (Reachability)**.** *Given a hybrid automaton $\mathcal{H}$, an initial state x_s and a final state x_f,* reachability *is defined as:*

$$Reach(\mathcal{H}, x_s, x_f) \equiv \exists \tau(I, \xi) \in [\mathcal{H}] \,.\, x_s \mapsto_\xi x_f.$$

2.5. Control systems

A control system consists of a plant, a system which performs some task, and a controller, a device that modifies the behavior of the plant to ensure correct operation. The proportional controller is a much-used controller [1], and we will show how we can model it as a hybrid automaton.

Definition 14 (Proportional controller)**.** *Let $\mathcal{S}$ be a TIDIS and let the constant k_p represent the ability of the controller to change the state of the TIDIS (called the* gain*), and let the constants u_{min} and u_{max} represent the limits of the range of influence of the controller. At time t, let $\epsilon(t)$ be the difference between the desired state $SP(t)$ of $\mathcal{S}$ (called the set point) and the actual state $PV(t)$ of $\mathcal{S}$ (called the process value) such that $\epsilon(t) = SP(t) - PV(t)$.*

Then a proportional controller *u is defined as*

$$u = \begin{cases} 0 & \text{if } \neg a \\ u_{max} & \text{if } a \wedge \epsilon(t) \geq \epsilon_{max} \\ k_p\epsilon(t) & \text{if } a \wedge \epsilon_{min} < \epsilon(t) < \epsilon_{max} \\ u_{min} & \text{if } a \wedge \epsilon(t) \leq \epsilon_{min} \end{cases}$$

where $\epsilon_{min} = \frac{u_{min}}{k_p}$, and $\epsilon_{max} = \frac{u_{max}}{k_p}$. The a is a boolean predicate on the state variables of the system.

Based on the above we give a definition of a TIDIS controlled by a proportional controller as a hybrid automaton, a representation which will facilitate approximation of the TIDIS by the GSPDI systems, see Section 3. In the following we will use the notation $F(x, y, E, u)$ to refer to the differential inclusion $F(x(t), y(t)) = \{f(x(t), y(t), e, c) \mid e \in E \subseteq \mathbb{R}^2, c \in \mathbb{R}\}$, where the set E represents the uncertainty of the system, and the constant c the control input. The predicate a is changed into the following membership test for technical reasons: Split the domain $Q \subseteq \mathbb{R}^2$ of the TIDIS into two path-connected, open sets Q_a and $Q_{\neg a}$ with a non-empty border. a is now defined as $\{(x(t), y(t)) \mid (x(t), y(t)) \in Q_a\}$ and $\neg a$ as $\{(x(t), y(t)) \mid (x(t), y(t)) \in Q_{\neg a}\}$. This enables transitions to be taken as Q_a and $Q_{\neg a}$ intersect, i.e. there are points $(x(t), y(t)) \in Q_a \cap Q_{\neg a}$.

Definition 15 (Proportionally controlled TIDIS)**.**
Given a TIDIS $\mathcal{S} = \langle Q,\ F,\ G\rangle$, a proportionally controlled TIDIS *(PC-TIDIS) $\mathcal{S}' = \langle Q,\ F,\ G,\ A\rangle$ is a hybrid automaton A restricted to domain Q and with $Act = \{F, G\}$ for all locations $l \in A$, as shown in Figure 7:*

- *$Var = \{x, y\}$.*
- *The locations, with invariants and activities are as follows:*
  - *off*
    - *Invariant: $Inv(off) = \{\neg a\}$*
    - *Activity: $\{\frac{dx}{dt} = F(x, y, E, 0), \frac{dy}{dt} = G(x, y, E, 0)\}$*

- max
  - *Invariant:* $Inv(max) = \{a \wedge \epsilon \geq \epsilon_{max}\}$
  - *Activity:* $\{\frac{dx}{dt} = F(x, y, E, u_{max}), \frac{dy}{dt} = G(x, y, E, u_{max})\}$
- min
  - *Invariant:* $Inv(min) = \{a \wedge \epsilon \leq \epsilon_{min}\}$
  - *Activity:* $\{\frac{dx}{dt} = F(x, y, E, u_{min}), \frac{dy}{dt} = G(x, y, E, u_{min})\}$
- $proportional$
  - *Invariant:* $Inv(proportional) = \{a \wedge \epsilon_{max} \geq \epsilon \geq \epsilon_{min}\}$
  - *Activity:* $\{\frac{dx}{dt} = F(x, y, E, k_p\epsilon), \frac{dy}{dt} = G(x, y, E, k_p\epsilon)\}$

- *The transitions and guards are:*
  - (max, off) – *Guard:* $(\neg a)$
  - (off, max) – *Guard:* $(a \wedge \epsilon \geq \epsilon_{max})$
  - $(max, proportional)$ – *Guard:* $(a \wedge \epsilon_{min} \leq \epsilon \leq \epsilon_{max})$
  - $(proportional, max)$ – *Guard:* $(a \wedge \epsilon \geq \epsilon_{max})$
  - $(proportional, off)$ – *Guard:* $(\neg a)$
  - $(off, proportional)$ – *Guard:* $(a \wedge \epsilon_{min} \leq \epsilon \leq \epsilon_{max})$
  - $(proportional, min)$ – *Guard:* $(a \wedge \epsilon \leq \epsilon_{min})$
  - $(min, proportional)$ – *Guard:* $(a \wedge \epsilon_{min} \leq \epsilon \leq \epsilon_{max})$
  - (min, off) – *Guard:* $(\neg a)$
  - (off, min) – *Guard:* $(a \wedge \epsilon \leq \epsilon_{min})$

Note that the relational operators $<, >$ have been changed to $\leq, \geq$ to ensure that the invariant $Inv(l)$ and border $\beta(l)$ are closed sets for all $l \in Loc$.

Example 7 (Proportionally controlled pendulum)**.** *We want to force the pendulum to move around the equilibrium point* $(0, 0)$ *in a circle of radius* 1*,* $\epsilon(t) = 1 - \sqrt{x(t)^2 + y(t)^2}$*. By setting* $u_{min} = -2$*,* $u_{max} = 2$ *and* $k_p = 3$ *we get* $\epsilon_{min} = -\frac{2}{3}$ *and* $\epsilon_{max} = \frac{2}{3}$*. The controller operates by increasing the acceleration of the pendulum as it descends in one direction, i.e.* $a = (x < 0 \wedge y > 0)$*. The invariants of the locations of the resulting hybrid automaton are illustrated in Figure 4-a), and an example trajectory in Figure 4-b).*

2.6. Truncated affine multi-valued functions (TAMFs)

In order to perform reachability analysis of GSPDIs we need to iteratively compute a successor function. Given the geometrical nature of such systems it turns out that successors could be expressed as special multi-valued functions operating on intervals. This section introduces the mathematical background for GSPDIs successor functions.

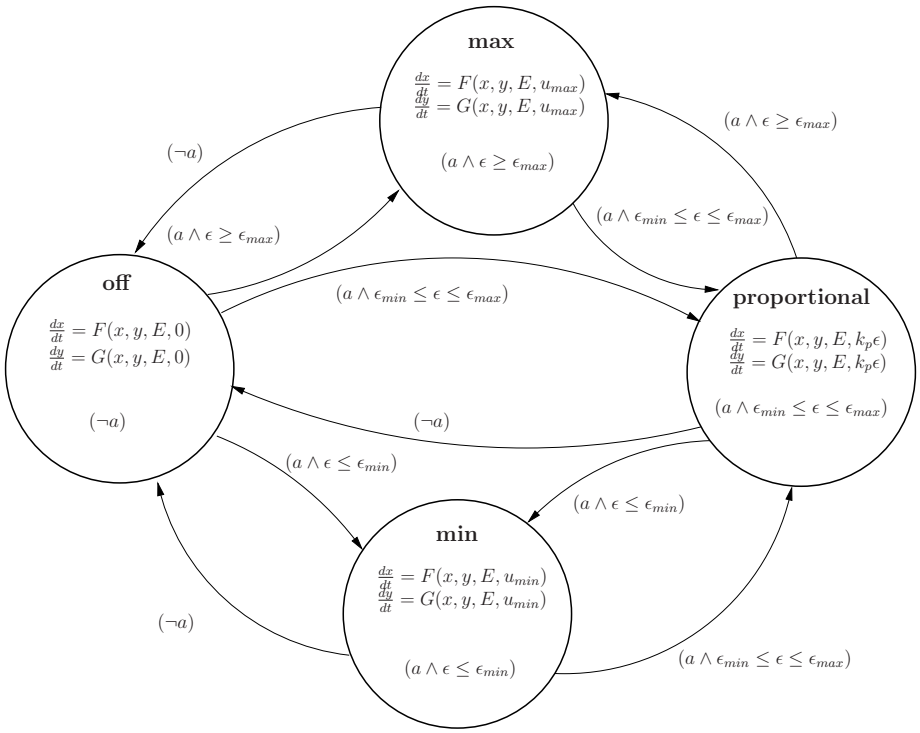

Figure 7: Proportionally controlled TIDIS.

Definition 16. *A* positive affine function $f : \mathbb{R} \to \mathbb{R}$ *is a function such that* $f(x) = ax + b, a > 0$. *The inverse of* f *is the positive affine function* $f^{-1}(x) = \frac{1}{a}x - \frac{b}{a}$.

In the vein of interval arithmetic [16], we can use two affine functions for over-approximation.

Definition 17. *Let* $X \subseteq 2^{\mathbb{R}}$ *be the set of closed intervals on* $\mathbb{R}$. *An* affine multivalued function *(AMF)* $F : X \to X$, *written* $F = [f_l, f_u]$, *is defined by* $F([l, u]) = [f_l(l), f_u(u)]$ *where* f_l *and* f_u *are positive affine functions. An* inverted affine multivalued function $F^{-1} : X \to X$, *is defined by* $F^{-1}([l, u]) = [f_u^{-1}(l), f_l^{-1}(u)]$.

We recall a useful result about the fixpoints of AMFs:

Lemma 2 ([15])**.** *Let* $[l_0, u_0] \subseteq \mathbb{R}$ *be any interval and* $F^n([l_0, u_0]) = [l_n, u_n]$. *Then the following properties hold:*

(1) The sequences l_n *and* u_n *are monotonous;*

(2) They converge to limits l^* *and* u^* *(finite or infinite), which can be effectively computed.*

In particular we are interested in AMFs with restricted inputs and outputs, since we are working with bounded real intervals.

Definition 18. *Let $X \subseteq 2^{\mathbb{R}}$ be the set of closed intervals on $\mathbb{R}$. Given an AMF F and two intervals $S \subseteq \mathbb{R}^+$ and $J \subseteq \mathbb{R}^+$ restricting the domain and range of F respectively, a* truncated affine multivalued function *(TAMF) $\mathcal{F}_{F,S,J} : X \to X$ is defined as follows: $\mathcal{F}_{F,S,J}(I) = F(I \cap S) \cap J$ if $I \cap S \neq \emptyset$, otherwise $\mathcal{F}_{F,S,J}(I) = \emptyset$. In what follows we will write $\mathcal{F}$ instead of $\mathcal{F}_{F,S,J}$. For convenience we write $\mathcal{F}(x) = \mathcal{F}([x, x])$.*

We say that $\mathcal{F}$ is normalized if $S = Dom(\mathcal{F}) = \{x \mid F(x) \cap J \neq \emptyset\}$ and $J = Im(\mathcal{F}) = \mathcal{F}(S)$, and will henceforth assume that all TAMFs are normalized. Unlike a differential inclusion, a multi-valued function is deterministic: The same input gives the same output. Thus an affine multi-valued function can be thought of as representing the set of all possible evolutions of a non-deterministic system.

3. Generalized polygonal hybrid systems (GSPDIs)

So far we have introduced concepts regarding the continuous evolution of hybrid automata, TIDISs, and CN-HAs. This section introduces a special kind of hybrid systems (a subclass of hybrid automata), namely GSPDIs, and provides some related definitions and results. We describe an algorithm for deciding reachability in next section.

Definition 19 (GSPDI)**.** *A* Generalized Polygonal Hybrid System *(GSPDI) is a pair $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$, where $\mathbb{P}$ is a finite mesh of some bounded subset of $\mathbb{R}^2$. Each $P \in \mathbb{P}$, called a* region, *is a convex polygon with area area(P). The union $\bigcup \mathbb{P}$ of all regions is called the* domain *of the GSPDI and assumed to be a convex polygon itself. $\mathbb{F}$ is a function associating a pair of* vectors *to each region, i.e., $\mathbb{F}(P) = (\mathbf{a}_P, \mathbf{b}_P)$, which describes an affine differential inclusion. Every point on the plane has its dynamics defined according to which polygon it belongs to: if $p \in P$, then $\dot{p} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$.*

Given a $P \in \mathbb{P}$, then for each $P' \in \mathbb{P}$, $P \neq P'$, such that $\beta(P) \cap \beta(P') \neq \emptyset$, we say that $\beta(P) \cap \beta(P')$ is an *edge* of P. Let $E(P)$ denote the set of all edges of region P.

Example 8. *Figure 8 shows an example GSPDI. For instance, the polygonal* region *R in the right has four* edges *$e_1, \ldots, e_4$ and the* arc *$\angle_{\mathbf{a}}^{\mathbf{b}}$ limits the behavior in the region. The trajectory ξ starts in point x_s on e_1, traverses the GSPDI to return to x_f on e_1.*

The diameter of the smallest circle that contains a region P is denoted *diam*(P). The continuous evolution of a GSPDI is in general non-deterministic, and without jumps we can extend the definition of a trajectory to GSPDIs.

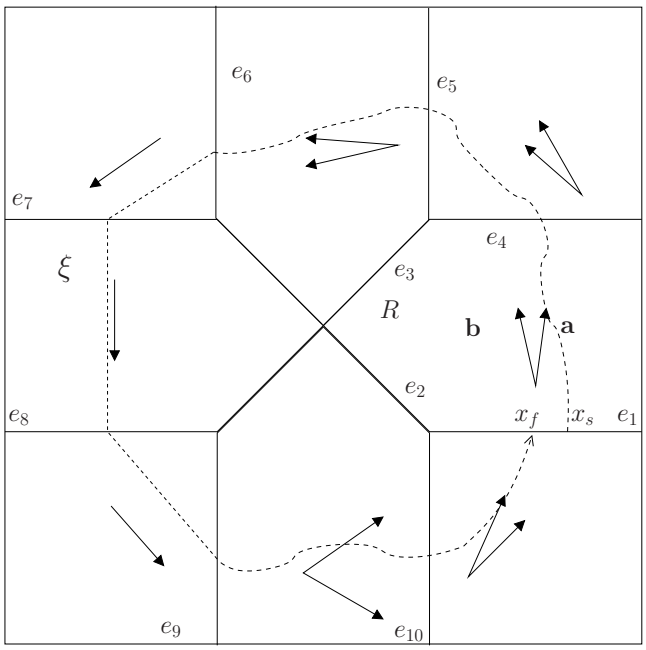

Figure 8: A GSPDI.

Definition 20 (GSPDI trajectory)**.** *A* trajectory ξ *of a GSPDI* $\mathcal{G}$ *is a continuous and almost-everywhere differentiable function* $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^2$ *s.t. the following holds: whenever* $\xi(t) \in P$ *for some* $P \in \mathbb{P}$*, then its derivative* $\dot{\xi}(t) \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$*. We write* $[\mathcal{G}]$ *for all the trajectories of* $\mathcal{G}$*.*

In Figure 8 the trajectory ξ obeys the dynamics of the regions of the GSPDI. Due to the restrictions on their dynamics, not many systems can be directly modeled as a GSPDI. Instead, we show how a CN-HA, which allows non-linear dynamics, can be approximated by a GSPDI.

Definition 21 (Approximation)**.** *A GSPDI* $\mathcal{G}$ approximates *a CN-HA* $\mathcal{C}$ *(written* $\mathcal{G} \geq \mathcal{C}$*) if* $\xi \in [\mathcal{C}]$ *implies* $\xi \in [\mathcal{G}]$*.*

In the following we assume that $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| \leq \pi$ for all $P \in \mathbb{P}$. If $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| > \pi$, then the region is *reach-all*, meaning that all points in $E(P)$ are reachable from any other in the region [8].[3] The trajectories of all GSPDI regions that are not reach-all can be *straightened* without loss of generality [15], turning them into a collection of lines traversing the edges of the GSPDI. In order to build a directed

[3]Whenever $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| > \pi$ then from the reachability point of view it is the same as $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| = 2\pi$, or in other words any trajectory is allowed in P.

reachability graph with edges as nodes, we characterize the edges according to the way trajectories may traverse them.

We say that an edge $e \in E(P)$ is an *entry-only* edge of P if for all $\mathbf{x} \in e$ and for all $\mathbf{c} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$, we have $\mathbf{x} + \mathbf{c}t \in P$ for some $t > 0$. e is *exit-only* if the same condition holds for some $t < 0$. Intuitively, an entry-only (exit-only) edge of a region P allows at least one trajectory in P starting (terminating) on edge e, but allows no trajectories in P terminating (starting) on edge e.

We write $in(P)$ to denote the set of all entry-only edges of P, and $out(P)$ to denote the set of exit-only edges of P. We call the set $E(P) \setminus (in(P) \cup out(P))$ the *inout* edges of P, $inout(P)$. The line determined by an edge e is denoted as $line(e)$. If $in(P)$ and $out(P)$ do not intersect for some $GSPDI$, then the system is a member of a strict subclass of $GSPDI$s, called SPDIs.

Definition 22 (SPDI)**.** *A region P such that $inout(P) = \emptyset$ is called a* good *region. For a GSPDI where all $P \in \mathbb{P}$ are good we say that the* goodness assumption *holds, and refer to the system as an* SPDI *[15].*

The definition of the goodness assumption for SPDIs is equivalent to the fact that for any region P the arc $\angle_{\mathbf{a}}^{\mathbf{b}}$ does not contain *any* vector parallel to any of the edges of P. Essentially the relaxation of the goodness assumption is the main difference between SPDIs and the more general GSPDIs.

If we abstract away the exact path of a trajectory, we can characterize it by the edges it traverses, a property that will be exploited for reachability computations.

Definition 23 (Signature)**.** *For a GSPDI $\mathcal{G}$, the* signature *of a trajectory $\xi \in [\mathcal{G}]$ is the ordered sequence of edges $\mathsf{Sig}(\xi) = e_1 \dots e_n \dots$ traversed by ξ.*

As an example, the signature of trajectory ξ in Figure 8 is $e_1e_4e_5e_6e_7e_8e_9e_{10}$.

Given a region P, we introduce a one-dimensional coordinate system on each edge $e \in E(P)$. For this edge we choose a point of origin, given by a vector $\mathbf{v}$, and a directional vector $\mathbf{e}$. The vector $\mathbf{e}$ has a clockwise direction with respect to the border of P for edges in $out(P)$, and counter-clockwise for edges in $in(P)$. Thus an inout edge e will have two distinct characterizations depending on whether it is considered as an input edge or as an output edge, e_i and e_o.

Each edge may be represented by a one-dimensional coordinate system [17]. For each edge e we choose (1) a point on it (the origin) with radius-vector $\mathbf{v}$, and (2) a director vector $\mathbf{e}$ going in a predefined direction. To characterize e we need the coordinates of its extreme points, namely, $e^l, e^u \in \mathbb{Q} \cup \{-\infty, \infty\}$ such that $e = \{\mathbf{x} = \mathbf{v} + x\mathbf{e} \mid e^l < x < e^u\}$. That is, an edge $e \in E$ can be represented by a triplet $(\mathbf{v}, \mathbf{e}, (e^l, e^u))$. In the following we will use $\mathbf{x} \in \mathbb{R}^2$ to denote a *point* on an edge e, and (e, x) to denote the local coordinate of $\mathbf{x}$ with respect to e. An *edge-interval* $(e, [x, y])$ denotes the interval between two local coordinates x and y of e, where we note that the coordinates are the same if e is seen as an output edge with respect to some region P, or as an input edge with respect to the other region P'. We assume in the following that $e = \{\mathbf{v} + x\mathbf{e} \mid 0 \leq x \leq 1\}$.

Thus, the largest possible edge-interval for any edge is $[0,1]$. We call $(e,[0,1])$ a *full* edge-interval.

Since a GSPDI does not have discrete evolutions we will focus on the continuous evolution and the time-successors of the systems. Specifically we will look at *edge-to-edge reachability*; how to, from a subset of an input edge e_i, compute the points reachable on an output edge e_o. First we define what we mean by reachability.

Definition 24 (Point-to-point reachability)**.** *For a region P, vector $\mathbf{c}$, and $e_i \in in(P), e_o \in out(P)$ and points $x_i \in e_i, x_o \in e_o$, we define the predicate $x_i \xrightarrow{\mathbf{c}} x_o$ to hold if there exists a $t \in \mathbb{R}^+$ such that $x_o = t\mathbf{c} + x_i$.*

If, for $e_i \in in(P)$ and $e_o \in out(P)$, $x_i \xrightarrow{\mathbf{c}} x_o$ for some $x_i \in e_i, x_o \in e_o$, i.e. it is *possible* to reach x_o from x_i, then we can compute a *successor* function mapping points on e_i to points on e_o.

Definition 25. *Let $\mathcal{G}$ be a GSPDI, $P \in \mathbb{P}$, $e_i \in in(P)$, $e_o \in out(P)$, $x_i \in e_i$, $x_o \in e_o$, and $\mathbf{c} \in \angle^{\mathbf{b}_P}_{\mathbf{a}_P}$. The* point-to-point *successor of x_i following $\mathbf{c}$ is the point $x_o = \mathsf{Succ}^{\mathbf{c}}_{e_i e_o}(x_i)$, which exists if $x_i \xrightarrow{\mathbf{c}} x_o$. We say that the vector $\mathbf{c}$* points in *(into P) across e_i, and that it* points out *(of P) across e_o. We also say that $\mathbf{c}$ is* good with respect to e_i and e_o.

Note that in the following we are restricting ourselves to vectors that are *good* with respect to some input and output edges. Later we will relax this restriction. Given the above restriction we can easily compute x_o given x_i.

Lemma 3 ([15])**.** *Assume a region P with $e_i \in In(P), e_o \in Out(P)$, a point $x_i \in e_i$, and a vector $\mathbf{c} \in \angle^{\mathbf{b}_P}_{\mathbf{a}_P}$ which is good with respect to e_i and e_o. Let $\hat{\mathbf{c}}$ represent the right rotation (a clock-wise rotation by $\pi/2$ radians) of $\mathbf{c}$. Then the following function is a successor:*

$$\mathsf{Succ}^{\mathbf{c}}_{e_i e_o}(x_i) = \frac{\mathbf{e}_i \hat{c}}{\mathbf{e}_o \hat{c}} x_i + \frac{(\mathbf{v}_i - \mathbf{v}_o)\hat{c}}{\mathbf{e}_o \hat{c}}.$$

We call this the standard construction *for successors.*

We use these positive affine functions to define truncated multi-valued functions (TAMFs) that are used to compute reachability for intervals of GSPDIs as opposed to point-to-point-reachability.

Definition 26 (Edge-to-edge successor)**.** *For a region P, arc $\angle^{\mathbf{b}}_{\mathbf{a}}$, output edge e_o and input edge e_i with some edge-interval (e_i, I), a truncated affine function $\mathsf{Succ}_{e_i e_o}$ is an* edge-to-edge successor *if for all intervals $S' \subseteq S$, $J' \subseteq J$ where S and J are the restrictions of the domain and range of the successor respectively, we have that $\mathsf{Succ}_{e_i e_o}(S') = J'$ if and only if there exists $x_i \in S'$, $x_o \in J'$ and $\mathbf{c} \in \angle^{\mathbf{b}}_{\mathbf{a}}$ such that $x_i \xrightarrow{\mathbf{c}} x_o$ holds.*

The following lemma shows how the positive affine successor $\mathsf{Succ}^{\mathbf{c}}_{e_i e_o}$ is used to construct the successor $\mathsf{Succ}_{e_i e_o}$ as a TAMF. For arc $\angle^{\mathbf{b}_P}_{\mathbf{a}_P}$ and edge-interval $(e_i, [l,u])$ where $[l,u] \subseteq [0,1]$ we have:

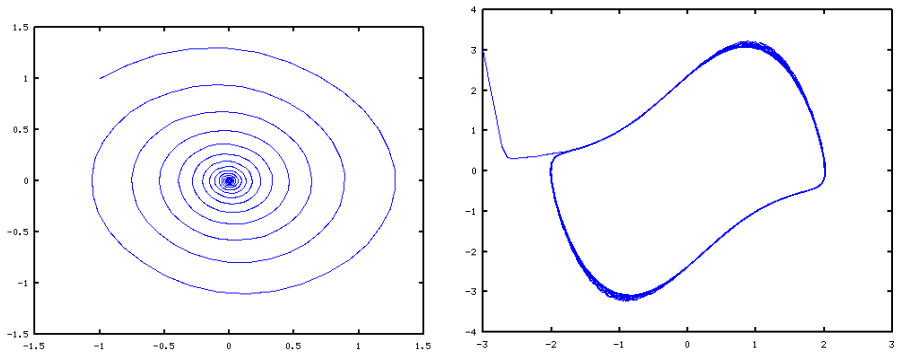

Figure 9: Trajectories of non-linear systems, the damped pendulum with a trajectory spiraling towards an equilibrium point on the left, and the van der Pol oscillator with a trajectory approaching a limit cycle on the right.

Lemma 4 ([15]). $\mathsf{Succ}_{e_i e_o}([l,u]) = [\mathsf{Succ}^{\mathbf{b}}_{e_i e_o}(l), \mathsf{Succ}^{\mathbf{a}}_{e_i e_o}(u)] \cap [0,1]$.

The signature of a trajectory of a GSPDI may include one or more *cycles*, a repetition of edges traversed. Cycles are of paramount importance when it comes to solving instances of the reachability problem for GSPDIs.

Definition 27 (Simple cycle). *An alternating sequence of distinct edges and interval successors* $e_1, \mathsf{Succ}_{e_1 e_2}, e_2, \mathsf{Succ}_{e_2 e_3}, \ldots \mathsf{Succ}_{e_{n-1} e_n}, e_n, \mathsf{Succ}_{e_n e_1}$*, is a* simple cycle*, and we denote it* $(e_1 \ldots e_n)$*. The successor obtained as* $\mathsf{Succ}_{e_n e_1} \circ \cdots \circ \mathsf{Succ}_{e_1 e_2}$ *is called the* cycle successor *of the cycle.*

We can characterize cycles as to whether the successively computed intervals on the cycle edges form one continuous interval, or a set of disjoint intervals. This distinction will affect reachability computation, see Definition 29 below.

Definition 28 (Continuous and disjoint cycles). *Let us assume a simple cycle* $\sigma = (e_1 \ldots e_n)$*, edge-interval* (e_1, I_1)*, with edge-intervals* (e_i, I_i) *defined as* $\mathsf{Succ}_\sigma(e_{i-1}, I_{i-1})$ *for* $1 < i \leq n$*. If* $\mathsf{Succ}_{\sigma_i}(I_i)$ *and* I_i *are adjacent or overlapping intervals then we say that the cycle* σ *is* continuous *with respect to* (e_1, I_1)*, otherwise it is* disjoint*.*

Cycles present a problem when performing reachability searches. Many non-linear systems exhibit phenomena such as equilibrium points, or limit cycles, which cannot be left by any trajectory. It is not possible to *reach* neither equilibrium points, nor limit cycles, they can only be approached as limits, leading to trajectories looping infinitely. For instance, the pendulum and the Van-der-Pol equation exhibit this kind of behavior, see Figure 9. In many cases the reachable set of a cycle can be computed without iteration, by analyzing the cycle successor. This is called *acceleration* [15].

Definition 29 (Continuous cycle acceleration)**.** *Let us consider a simple cycle σ with cycle successor Succ_σ, which is continuous with respect to some edge-interval $(e, [l, u])$. Assume Succ_σ consists of the positive affine functions f_l and f_u and define the interval $[L, U]$ as $S \cap J$. Given also the fixpoints l^* and u^* of Succ_σ. Then an interval I on e is said to be computed by a* continuous cycle acceleration *if the following holds:*

$$I = [max(L, min(l, l^*)), min(U, max(u, u^*))].$$

Though GSPDIs may be represented as hybrid automata, GSPDI's reachability algorithm is not performed on the underlying hybrid automaton but on a reachability graph having the edges as nodes (and not regions).

Definition 30 (Edge graph, [18])**.** *Given a $GSPDI$ $\mathcal{G}$, the* reachability graph *of $\mathcal{G}$ with mesh $\mathbb{P}$ is the graph (N, E) where $E \subseteq N \times N$ with the region edges as nodes: $N = \bigcup_{P \in \mathbb{P}} E(P)$; and two types of transitions:*

- *Edge-to-edge transitions: $(e_1, e_2) \in E$ if there exists a successor $\mathsf{Succ}_{e_1 e_2}$ with $S \cap J \neq \emptyset$, where S and J are the restrictions of the domain and range of F respectively.*
- *Cycle transitions: For all edges e and all cycles σ with e as the first node, $(e, e) \in E$ if $S \cap J \neq \emptyset$ for Succ_σ.*

Example 9. *In Figure 10 we see an edge-interval N. It has three successor intervals $E1$, $E2$, and $E3$ on three different edges, plus the cycle successor interval C computed by acceleration.*

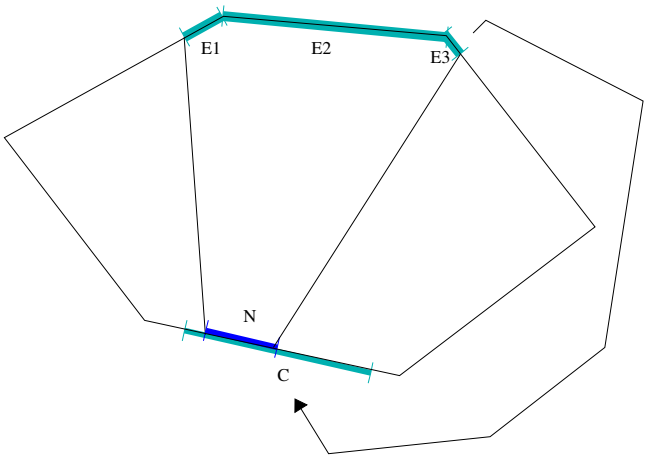

Figure 10: An edge-node, N, its edge successors $E1, E2, E3$, and cycle successor C.

4. Reachability analysis of GSPDIs

In this section we present new results concerning the reachability analysis of GSPDIs. We start by giving an informal description of earlier algorithms we have developed [3, 8]. We proceed by giving a special construction of edge-to-edge successors for edges that are not good, key to prove one of our main result in this section, namely that reachability may be performed without iterating any simple cycle more than at most twice (i.e., we *accelerate* all cycles). We finally provide our new reachability algorithm and prove that it is sound, complete, and that it terminates.

4.1. Informal description of previous GSPDI reachability algorithms

To better understand our new reachability algorithm we informally explain in what follows the original reachability algorithm for GSPDIs proposed in [3], and improved in [8].

The reachability algorithm of [3] gave special treatment to *inout* edges, using directed edges to differentiate between the edge used as an input, and when it is used as an output. Depending on in which direction the trajectory traverses an inout edge e_1, the edge will be considered as an input edge in for one region, but as an output edge for the adjacent region, and similarly the *inverse* edge e_1^{-1} (i.e., the edge considering the opposite order to edge e_1) would be an output edge in the first region and an input edge in the second one. In other words, any path passing through edges such as $\sigma = e_0 e_1 e_2 \dots e_n e_1^{-1} e_{n+1}$ could in principle be analyzed without problem. Since e_1 and e_1^{-1} are considered distinct edges the above path does not contain any cycle.

The problem with such paths is that it allows to 'bounce' off an edge. Note that any pair of edges $e_0 e_1$ is part of some path if e_0 is an input edge of a region, and e_1 is an output edge of the same region. One could then calculate the TAMF for such a trajectory. However, ee^{-1} can now be part of a valid path, whose behavior cannot be expressed as a normal TAMF, rather by a TAMF which needs to be manipulated by applying an auxiliary function (called *Flip* in [3]) in order to facilitate the treatment of such bounces in paths. There are some problems with the solution sketched in [3]: (i) Simple cycles containing bounces need special treatment; (ii) There are many implicit assumptions in the theoretical results, making unfeasible the implementation of the algorithm.

The solution introduced in [8] to the above problems were to: (i) Prove that the treatment of simple cycles containing bounces can be avoided; (ii) Make all the assumptions explicit, allowing an effective implementation of the algorithm. The reachability solution given in [8] is not based on the one presented in [3] (which is a depth-first search algorithm), but rather on an adapted version of the breadth-first search algorithm for SPDIs shown in [18]. The algorithm works in a standard manner on a directed graph where the edges are nodes and successors are transitions (cf. Definition 30). From an initial edge-interval all possible child edge-intervals are generated and put in a queue. These are then handled in turn. The search is finished whenever some goal edge-interval is reached (success), or

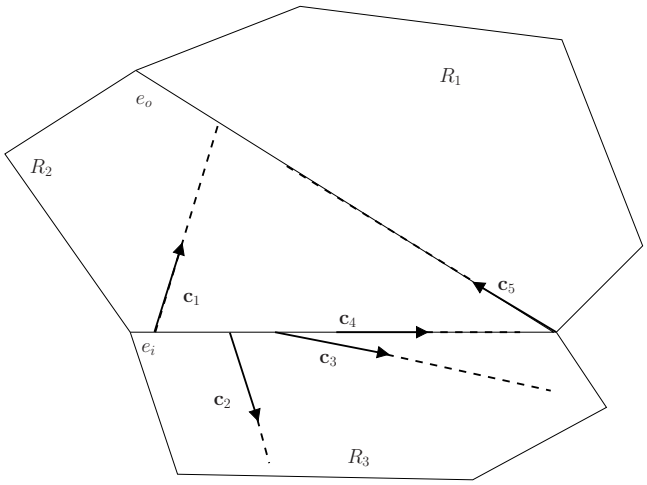

Figure 11: Vectors illustrating the problems in creating successors from GSPDIs.

the queue is empty (failure). There are two kinds of transitions: Those that represent ordinary successors $\mathsf{Succ}_{e_i e_o}$, and those that represent the successor Succ_{s^k}, iterating a cycle s any k number of times.

In the rest of this section we present an improved reachability algorithm, following a breadth-first search strategy as in [8], but with the additional interesting feature that all (simple) cycles will be treated without needing to iterate them (i.e., all cycles can be *accelerated*).

4.2. Edge-to-edge successors for inout edges

The *standard construction* of an edge-to-edge successor $\mathsf{Succ}_{e_i e_o}$, see Lemma 3, requires e_i to be entry-only and e_o to be exit-only. The presence of *inout* edges in a GSPDI complicates the construction of edge-to-edge successors, as the construction requires positive affine functions.

Figure 11 illustrates the problem. Any one of the five vectors $c_1 \dots c_5$ might possibly be in the dynamics of region R_2. Following a good vector, $\mathbf{c}_1$, in the positive direction maps a single point on e_i to a single point on e_o, and the standard construction can be used. However, following $\mathbf{c}_2$ in a positive direction will never cause an intersection with $\mathit{line}(e_o)$, where $\mathit{line}(\mathbf{x})$ denotes the *line* of infinite length overlapping $\mathbf{x}$. Following $\mathbf{c}_3$ leads us out of the region, and the result is a *negative* affine function. Following both $\mathbf{c}_4$ and $\mathbf{c}_5$ from some points on e_i we reach points on e_o, but not through some positive, affine function.

We will handle this problem by first giving a definition of a *total* arc that allows any point to be reached from any other.

Definition 31. *Given a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$, a region $P \in \mathbb{P}$ and two edges $e_i, e_o \in E(P)$, an arc α is a* total arc *if for all $x_i \in e_i$ and all $x_o \in e_o$ there exists a $\mathbf{c} \in \alpha$ such that $x_i \xrightarrow{\mathbf{c}} x_o$ holds.*

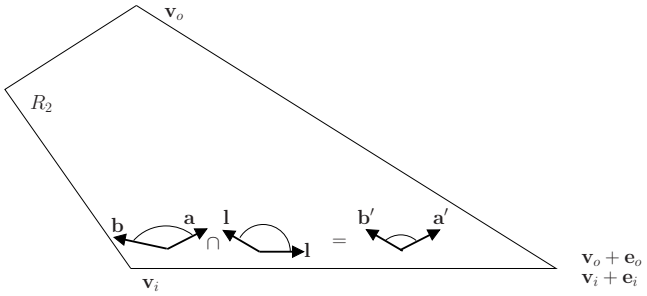

Figure 12: A total arc and the pruning computation.

The following lemma shows that a total arc preserves reachability for any arc $\angle^{\mathbf{b}}_{\mathbf{a}}$:

Lemma 5. *Given a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$, a region $P \in \mathbb{P}$ and two edges $e_i, e_o \in E(P)$ with total arc α. Then if $x_i \xrightarrow{\mathbf{c}} x_o$ for some $\mathbf{c} \in \angle^{\mathbf{b}}_{\mathbf{a}}$, then $\mathbf{c} \in \angle^{\mathbf{b}_P}_{\mathbf{a}_P} \cap \alpha$.*

Proof. Any $\mathbf{c} \in \angle^{\mathbf{b}_P}_{\mathbf{a}_P}$ for which $x_i \xrightarrow{\mathbf{c}} x_o$ holds, $x_i \in e_i$, $x_o \in e_o$, is also in α. □

When computing $\angle^{\mathbf{b}_P}_{\mathbf{a}_P} \cap \alpha$, we say that we are *pruning* the behavior of P with respect to e_i and e_o, and we denote this pruned behavior as $\angle^{\mathbf{b}'}_{\mathbf{a}'}$. An example of pruning may be seen in Figure 12, where $\mathbf{u}$ denotes the vector from the 'left' endpoint of e_i, $\mathbf{v}_i$, to the 'right' endpoint of e_o, $\mathbf{v}_o + \mathbf{e}_o$, and $\mathbf{l}$ the vector between the two other endpoints, $\mathbf{v}_i + \mathbf{e}_i$ to $\mathbf{v}_o$.

Lemma 6. *Given a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$, a region $P \in \mathbb{P}$, and edges $e_i, e_o \in E(P)$. Let $\mathbf{u}$ be the vector from $\mathbf{v}_i$ to $\mathbf{v}_o + \mathbf{e}_o$, and $\mathbf{l}$ be the vector from $\mathbf{v}_i + \mathbf{e}_i$ to $\mathbf{v}_o$. Then $\angle^{\mathbf{u}}_{\mathbf{l}}$ is a total arc for e_i and e_o.*

Proof. $\mathsf{Succ}^{\mathbf{c}}_{e_i e_o}(x)$ is necessarily contained in $[\mathsf{Succ}^{\mathbf{l}}_{e_i e_o}(x), \mathsf{Succ}^{\mathbf{u}}_{e_i e_o}(x)]$ for any $x \in [0, 1]$, since $\mathbf{l}$ and $\mathbf{u}$ represent the extremal lines between points in e_i and e_o. □

After pruning, the two kinds of vectors left in $\angle^{\mathbf{b}'}_{\mathbf{a}'}$ are good vectors or vectors parallel to either or both of $\mathbf{e}_i$ and $\mathbf{e}_o$. We will compute reachability for the pruned arcs by constructing point-to-point successors for the parallel vectors, and constructing interval successors for the rest of the vectors.

If we consider all vectors in $\angle^{\mathbf{b}'}_{\mathbf{a}'}$ that are good, that is, they all point in across e_i and out across e_o, then it should be trivial to construct a good interval successor for this arc. However, $\angle^{\mathbf{b}'}_{\mathbf{a}'}$ may contain vectors that intersect $line(e_o)$ at some point at infinity. In an earlier work we showed how we could use $\pm\infty$ as constant approximations for the interval successors [8].

4.3. Point-point-successors

The standard construction of $\mathsf{Succ}^{\mathbf{c}}_{e_i e_o}$(Lemma 3) is computed from the expression below, where $\hat{\mathbf{c}}$ represents the right rotation (a clock-wise rotation by $\pi/2$ radians) of $\mathbf{c}$:

$$\mathbf{v}_o\hat{\mathbf{c}} + x_o\mathbf{e}_o\hat{\mathbf{c}} = \mathbf{v}_i\hat{\mathbf{c}} + x_i\mathbf{e}_i\hat{\mathbf{c}}.$$

An assumption in the standard construction is that neither $\mathbf{e}_i\hat{c}$ nor $\mathbf{e}_o\hat{c}$ are zero, or in other words that $\mathbf{c}$ is parallel to neither edges, guaranteeing to have well-formed AMFs. However, as this is not the case in general (for non-good regions) we will need to consider the problematic cases in order to extract the conditions to preserve the edge-to-edge reachability (for $x_i \xrightarrow{\mathbf{c}} x_o$). We consider the following three cases.

Case $\mathbf{e}_i||\mathbf{c}$: In this case we get that all input values give the same constant value x_o,

$$\mathbf{v}_o\hat{\mathbf{c}} + x_o\mathbf{e}_o\hat{\mathbf{c}} = \mathbf{v}_i\hat{\mathbf{c}} + x_i\mathbf{e}_i\hat{\mathbf{c}}$$

$$\mathbf{v}_o\hat{\mathbf{c}} + x_o\mathbf{e}_o\hat{\mathbf{c}} = \mathbf{v}_i\hat{\mathbf{c}}$$

$$x_o = \frac{\mathbf{v}_i\hat{\mathbf{c}}}{\mathbf{e}_o\hat{\mathbf{c}}} - \frac{\mathbf{v}_o\hat{\mathbf{c}}}{\mathbf{e}_o\hat{\mathbf{c}}}.$$

Case $\mathbf{e}_o||\mathbf{c}$: In this case only the intersection point of $line(e_o)$ and $line(e_i)$ causes $line(e_o)$ to be reached,

$$\mathbf{v}_o\hat{\mathbf{c}} + x_o\mathbf{e}_o\hat{\mathbf{c}} = \mathbf{v}_i\hat{\mathbf{c}} + x_i\mathbf{e}_i\hat{\mathbf{c}}$$

$$\mathbf{v}_o\hat{\mathbf{c}} = \mathbf{v}_i\hat{\mathbf{c}} + x_i\mathbf{e}_i\hat{\mathbf{c}}$$

$$x_i = \frac{v_0\hat{c}}{e_i\hat{c}} - \frac{v_i\hat{c}}{e_i\hat{c}}$$

Case $\mathbf{e}_o||\mathbf{c}$ and $\mathbf{e}_i||\mathbf{c}$: $\mathbf{e}_o$ is reachable from $\mathbf{e}_i$ only if $line(e_i) = line(e_o)$,

$$\mathbf{v}_o\hat{\mathbf{c}} = \mathbf{v}_i\hat{\mathbf{c}}.$$

From all of the above we see that it is always possible to construct a (non-standard) successor that is conservative, in the sense that reachability is preserved.

Theorem 1. *Given a GSPDI $\mathcal{G} = \langle \mathbb{P},\ \mathbb{F}\rangle$, a region $P \in \mathbb{P}$ with dynamics $\angle^{\mathbf{b}}_{\mathbf{a}}$ such that $\angle^{\mathbf{b}}_{\mathbf{a}} \leq \pi$, and two edges $e_i, e_o \in E(P)$. Then we can construct successors (point-to-point and interval) that preserve edge-to-edge reachability.*

Example 10. *Consider the partial GSPDI of Figure 13, with the cycle $(e_1e_2e_3)$. The successors $\mathsf{Succ}_{e_1e_2}$ and $\mathsf{Succ}_{e_3e_1}$ are good, but in region R_2 we see that neither $\mathbf{a}$ nor $\mathbf{b}$ are good. We prune $\angle^{\mathbf{b}}_{\mathbf{a}}$ and get $\angle^{\mathbf{b}'}_{\mathbf{a}'} = \angle^{\mathbf{u}}_{\mathbf{l}}$ (the two dashed lines), and subsequently we are able to compute an interval successor through the standard construction, using the pruned dynamics $\angle^{\mathbf{b}'}_{\mathbf{a}'}$.*

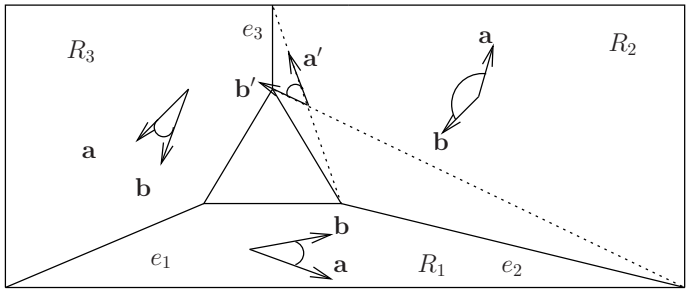

Figure 13: The region R_2 shows both original dynamics $\angle^{\mathbf{b}}_{\mathbf{a}}$ and modified dynamics $\angle^{\mathbf{b}'}_{\mathbf{a}'}$.

4.4. Cycle acceleration

As is well known in reachability, iterating cycles may lead to algorithmic solutions with high computational complexity. However, with the ability to compute and compose edge-to-edge successors and consequently to compute successor functions for cycles, we can accelerate cycles: A simple computation determines the exit set of the cycle and likewise whether a point on the cycle is reachable or not. Before presenting our main result concerning cycle acceleration (Theorem 2), we present a series of auxiliary lemmas used in the proof of the theorem.

Lemma 7. *Given a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$, a region $P \in \mathbb{P}$, and two edges $e_i, e_o \in E(P)$. For a successor $\mathsf{Succ}_{e_i e_o}$, if either of the functions $\mathsf{Succ}^{\mathbf{b}}_{e_i e_o} = c_l$ and/or $\mathsf{Succ}^{\mathbf{a}}_{e_i e_o} = c_u$, where c_l and c_u are real constants, then $c_l \leq 0$ and/or $c_u \geq 1$.*

Proof. From Lemma 5 and the procedure from constructing successors from [8], we know that c_l and c_u are either $\pm\infty$, or are defined by the value $\frac{(\mathbf{v}_i - \mathbf{v}_o)\hat{\mathbf{c}}}{\mathbf{e}_o \hat{\mathbf{c}}}$, the point where $line(e_i)$ and $line(e_o)$ intersects, which is a point not in the interior of P. □

Lemma 8. *Given a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$, a region $P \in \mathbb{P}$ and σ a simple cycle with $e \in E(P)$ as the first edge of σ. If σ is continuous with respect to some edge-interval $(e, [l, u])$, then acceleration computes exactly the interval reachable on e by iteration starting from $(e, [l, u])$.*

Proof. We want to show that $[max(L, min(l, l^*)), min(U, max(u, u^*))]$ contains exactly the reachable set of a cycle σ on the edge e.

First we show that all trajectories iterating the cycle are contained in the acceleration interval: Assume a trajectory $\xi \in [\mathcal{G}]$, where $\xi(0) \in (e, [l, u])$. If $L > l$ or $U < u$, then either l, u or both are not part of the cycle. Since, by monotonicity [15], we know that $\mathsf{Succ}^n_\sigma(l) \leq \mathsf{Succ}^n_\sigma(\xi(0)) \leq \mathsf{Succ}^n_\sigma(u)$, the trajectory will never leave $[min(l, l^*), max(u, u^*)]$. Since any value outside $[L, U]$ does not belong in the reachable set from iterating the cycle, this limitation also holds.

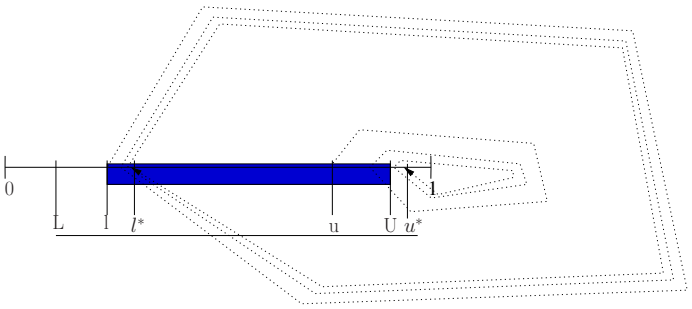

Figure 14: Cycle acceleration, reachable set on cycle in black (blue in color printing).

Then we show that the acceleration interval contains only the intervals generated by iterating. We see that $[l, u]$ contains what is already reached, and since σ is a continuous cycle we know that $[l^*, u^*]$ are the limits of the continuous interval that can (eventually) be reached. The interval $[min(l, l^*), max(u, u^*)]$ contains only this reachable set, while the limitation to $[L, U]$ ensures that only trajectories on the cycle are included. □

Example 11. *An example of cycle acceleration is given in Figure 14. We see that $max(L, min(l, l^*))$ in this case is l, which determines the lower limit of the reachable interval: The iteration does not increase the reached set, since $l^* > l$. The upper limit is given by $min(U, max(u, u^*))$, in this case U: The iteration increases the reachable set until the cycle is left at U.*

The following lemma shows that it is possible to calculate how many iterations are needed to reach/pass a given point when iterating a a cycle.

Lemma 9 (Iterated value of positive affine function)**.** *Given a positive affine function $f(x) = ax + b$ and a value $x_0 \in \mathbb{R}$ such that the sequence $x_0, \ldots$ is increasing with fixpoint x^*. Let X be any number between x_0 and x^*. Then the number of iterations required to reach X is given by $n = \eta_f(x_0, X)$, where*

$$\eta_f(x_0, X) = \log_a \frac{X - aX - b}{x_0 - ax_0 - b}$$

If n is an integer then $f^n(x_0) = X$, otherwise $f^{floor(n)}(x_0) < X$ and $f^{ceiling(n)}(x_0) > X$ gives the closest values, smaller and larger, to X. The corresponding case also holds when $x_0, \ldots$ is a decreasing sequence.

Proof. We get this by rearranging $f^n(x_0) = a^n x_0 + \frac{a-a^n}{1-a} b + b$:

$$f^n(x_0)\frac{(1-a)}{b} = a^n \frac{x_0(1-a)}{b} + a - a^n + (1-a)$$

$$a^n \frac{x_0(1-a)}{b} = f^n(x_0)\frac{1-a}{b} + a^n - 1$$

$$a^n = \frac{f^n(x_0) - af^n(x_0) - b}{x_0 - ax_0 - b}$$

$$n = \log_a \frac{X - aX - b}{x_0 - ax_0 - b}, \text{ where } f^n(x_0) = X$$

□

Example 12. *The positive affine function* $0.85x+0.3$ *has fixpoint* 2.0*. We want to know, given* $x_0 = 0.1$*, at which iteration* 1.0 *is passed. So we use Lemma 9 to compute* $n \approx 3.95$*, which means* 1.0 *is passed between the third and fourth iteration of* f*. For the function* $1.5x - 0.5$ *which is decreasing for* $x_0 = 0.9$*, we get than* 0.0 *is passed at* $n \approx 5.68$*.*

The following two lemmas show results concerning reachability inside disjoint cycles, as well as what is the reachable set when leaving such cycles, i.e. the exit set.

Lemma 10 (Disjoint cycle reachability)**.** *Let* σ *with start interval* $(e, [l, u])$ *be a disjoint cycle, and let* $x \in [L, U]$*. Let* $\{I\}$ *denote the set of disjoint edge-intervals* $(e, I_1), \ldots, (e, I_n)$ *where* $I_{i+1} = \mathsf{Succ}_\sigma(I_i)$ *with* $I_1 = [l, u]$*. Then the question of whether* $(e, x) \in \{I\}$ *can be answered without computing* $\{I\}$ *explicitly.*

Proof. In the following we consider the case where $x_0 \ldots$ is increasing. We compute $n = \eta_{f_l}(l, x)$ as by Lemma 9, and the interval $[f_l^{floor(n)}(l), f_u^{floor(n)}(u)]$. We have that $f_l^{floor(n)}(l) \leq x < f_l^{floor(n)+1}(l)$, and thus if x is reachable, then it will be in the interval $[f_l^{floor(n)}(l), f_u^{floor(n)}(u)]$. □

Lemma 11 (Disjoint cycle exit)**.** *Given a disjoint cycle* σ*, then the exit set on all edges* e*,* $e \notin \sigma$*, can be computed without iterating the cycle.*

Proof. First we assume that the fixpoint of σ actually allows it to leave the cycle. The disjoint nature of the successor makes leaving the cycle impossible until either L or U are reached. We consider the case where the cycle is left by passing L. Then we can compute the *penultimate* interval $[f_l^{floor(n)}(l), f_u^{floor(n)}(u)]$ using $n = \eta_{f_l}(l, L)$, from which no trajectories can leave since L is not passed yet. Since this is the last interval before the cycle is left we only have to iterate once to leave. □

The following result shows that if the reachable set of an iterated cycle starts out being continuous, then it will never become disjoint (though the opposite is possible).

Lemma 12. *Given two positive affine functions* f_l *and* f_u *where* $f_u(x) \geq f_l(x)$ *for any* x*. If for any two values* $l \leq u$ *we have that* $f_l(l) \leq u \leq f_u(u)$ *(*$f_u(u) \geq l \geq f_l(l)$ *respectively), then* $f_l(f_l(l)) \leq f_u(u)$ *(*$f_u(f_u(u)) \geq f_l(l)$ *respectively).*

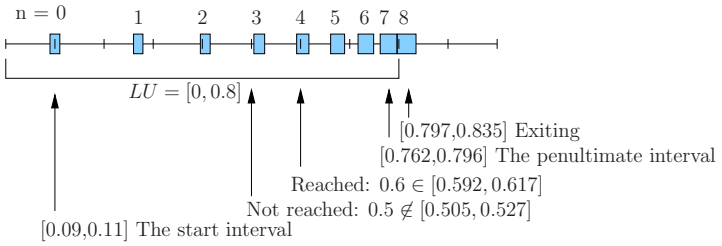

Figure 15: A successor function $[0.8x + 0.188, 0.81x + 0.19]$.

Proof. Call $f_l(l)$ for l'. Then we have $l' \leq u$ and subsequently $f_l(l') \leq f_u(u)$ which holds due to monotonicity and the requirement that $f_u(x) \geq f_l(x)$ for any x. The decreasing case can be proved in a similar manner. □

The proofs of Lemmas 10 and 11 give us the following procedure.

Procedure 1 (Disjoint cycle acceleration procedure)**.** *Given a simple cycle σ and edge-interval $(e, [l, u])$. If $\mathsf{Succ}_\sigma([l, u]) \cap [l, u]$ is non-empty, then the cycle is continuous. Otherwise perform the following two computations:*

- *Reachability: $(e, x) \in \{I\}$ is determined by whether $x \in [f_l^{floor(n)}(l), f_u^{floor(n)}(u)]$, where $n = \eta_{f_l}(l, x)$.*
- *Exit set: Perform acceleration as per Definition 29 to determine whether exiting σ is possible, and the limit L or U the cycle will cross. Then the interval produced by the penultimate iteration of σ before it reaches limit L or U is given by $[f_l^{floor(n)}(l), f_u^{floor(n)}(u)$, where either $n = \eta_{f_l}(l, L)$ or $n = \eta_{f_u}(u, U)$.*

Definition 32 (Disjoint cycle acceleration)**.** *Given a simple cycle σ and edge-interval $(e, [l, u])$. If $\mathsf{Succ}_\sigma([l, u]) \cap [l, u] = \emptyset$, then the computation as described in Procedure 1 is called the* disjoint cycle acceleration *of σ with respect to $(e, [l, u])$.*

For an illustration of the definition (and procedure) above, see Figure 15, where it is assumed that the successor function of a given cycle σ is $\mathsf{Succ}_\sigma = [0.8x + 0.188, 0.81x + 0.19]$, showing a point 0.5 that is not reached, a point 0.6 that is reached, and the penultimate and exiting edge-intervals with $U = 0.8$.

As a consequence of all the above we have the following result.

Theorem 2 (Cycle iteration)**.** *The reachability question for GSPDIs can be answered by iterating any simple cycle at most twice.*

Proof sketch. We will first show that we do not have to iterate to compute the fixpoints of any cycle, and then that we can decide reachability and exit sets also without iteration.

1. By Lemma 2 we know that we either can compute the fixpoints of the positive affine functions of the AMFs of any successor, or we know that $x^* \leq L \vee x^* \geq U$ by Lemma 7, in both cases without needing to iterate.

2. In order to prove that we can decide reachability and exit sets also without iteration, we separate our analysis in two cases, depending on whether the cycle is continuous or disjoint.

   (a) For a continuous cycle we can compute the reachable set by Lemma 8 and the exit sets directly by the edge-to-edge successors.

   (b) For a disjoint cycle we can compute the exit set as per Definition 32, derived from the proof of Lemma 11. We cannot compute the reachable set of the disjoint intervals $\{I\}$ without iterating. But we can however, given a point x, check whether $(e, x) \in \{I\}$, also as per Definition 32, derived from the proof of Lemma 10.

From the above, we have proved that we can decide reachability without needing to iterate (simple) cycles more than two times. □

4.5. Reachability algorithm

The high computational cost of iterating, and of generating cycles [19], gives us an incentive to analyze cycles as soon as they appear in a search. We have developed an algorithm for deciding reachability for GSPDIs based on a standard breadth first search algorithm (Algorithm 1). The algorithm iterates through a queue of edge-intervals, starting from the initial point *Src*. In a breadth-first search the children of each node being considered are computed. In our case we compute 1) the edge-to-edge successors of (e, I), 2) the set reachable on (e, I) due to acceleration of any continuous cycle, if applicable, and 3) the exit set due to acceleration of any disjoint cycle, also if applicable. The test $Dst \in children$ is to be interpreted as determining whether *Dst* is contained in the list *children* of edge-intervals and whether *Dst* is reachable from the disjoint cycle acceleration of any disjoint cycle, if applicable.

We finally prove that our reachability algorithm is sound, complete, and it terminates.

Theorem 3. *Algorithm 1 is sound, complete, and it terminates.*

Proof sketch. The algorithm is a breadth-first search method where new edge-intervals are added to the todo list if and only if they are visited by some trajectory, either by an edge-to-edge successor or a cycle successor. The question is thus whether these successors are sound and complete or not.

By Theorem 1 we know that the edge-to-edge successors are sound and complete. To reduce the run-time of the algorithm we know that we do not have to iterate any cycle more than at most twice, by Theorem 2, which forces us to consider soundness and completeness for acceleration. Acceleration of continuous cycles is sound and complete by Lemma 8. We must show soundness and completeness for the disjoint cycle acceleration of disjoint cycles. We know

Algorithm 1 GSPDI breadth-first reachability search algorithm.

```
1: Input: Src, Dst
2: visited := [Src]
3: todo := [Src]
4: while Not empty todo do
5:    (e, I) := todo.get()
6:    children := (e, I).successors() + (e, I).accelerated()
7:    if Dst ∈ children then
8:       Return REACHED
9:    end if
10:   visited.add(children)
11:   todo.add(children)
12: end while
13: Return NOT-REACHED
```

that given a disjoint cycle σ we can, by Lemma 10, resolve the question of whether $x \in (e, I)$ for any edge-interval (e, I), $e \in \sigma$ precisely, which is our definition of soundness and completeness. For any $e \notin \sigma$, Lemma 11 preserves the soundness of the exit set from σ and, since the method described is to compute edge-successors of the last iteration of the cycle before leaving, we also have completeness.

As there is only a finite number of cycles in a GSPDI, and since we do not need to iterate any cycle more than at most twice, we know that the algorithm terminates. □

4.6. The GSPeeDI tool

The tool GSPeeDI solves the reachability question for GSPDIs [10, 4]. It implements a tool chain of three separate stages:

- *System to GSPDI*: A system with possibly non-linear dynamics is approximated by a GSPDI. The current version of GSPeeDI, version 2.2, non-conservatively approximates non-linear autonomous systems. The approximation algorithm that is implemented, which is the topic of the next section, guarantees conservative results, but the implementation uses some non-conservative external subroutines.

- *GSPDI to edge-graph*: An edge-graph is built from a GSPDI, including generating the edge-to-edge successors from the region-wise arcs $\angle_{\mathbf{a}}^{\mathbf{b}}$.

- *Reachability search*: Given a GSPDI, a starting point and final point, the tool decides whether the final point is reachable from the initial one for the given GSPDI. This part of the tool chain is based on the theory presented in the previous section.

GSPeeDI can handle GSPDIs consisting of more than a thousand regions, such the one based on the damped pendulum as shown in Figure 16, generated by

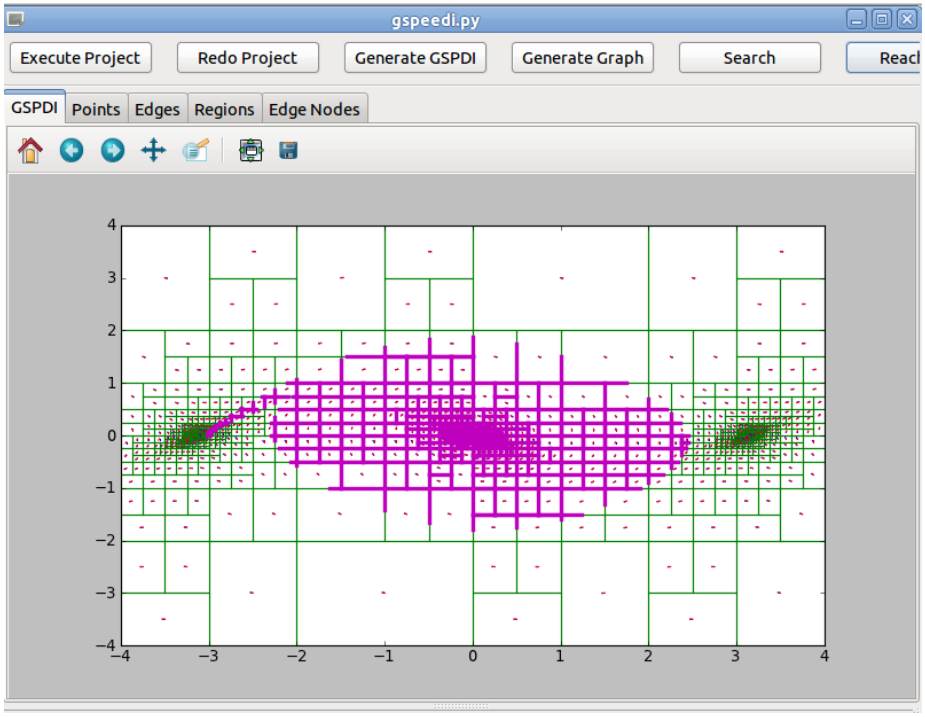

Figure 16: GSPeeDI screen shot: A GSPDI based on the pendulum.

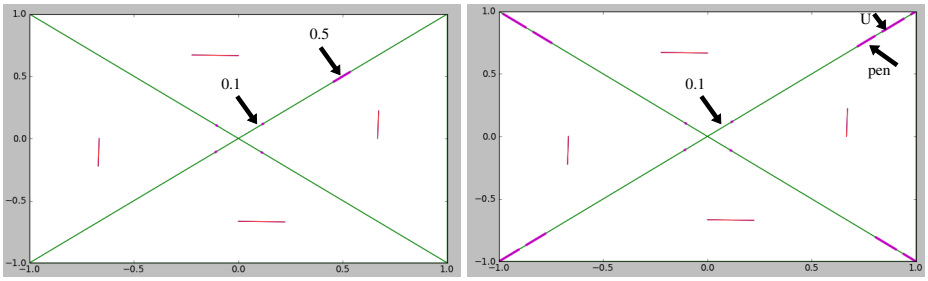

Figure 17: a) Disjoint cycle reachability. b) Disjoint cycle exit set.

the tool by (non-conservatively) hybridizing a system of autonomous differential equations, see [7]. In the screenshots, the GSPDI edges are marked by thin lines (green in the color printed version), while those parts of the edges that are in the current reachable set are marked by bold lines (purple in the color printed version). The arcs $\angle_{\mathbf{a}}^{\mathbf{b}}$ shown in the centers of the regions, but their directions are only discernible when the tool's zoom function is used. The reachable set

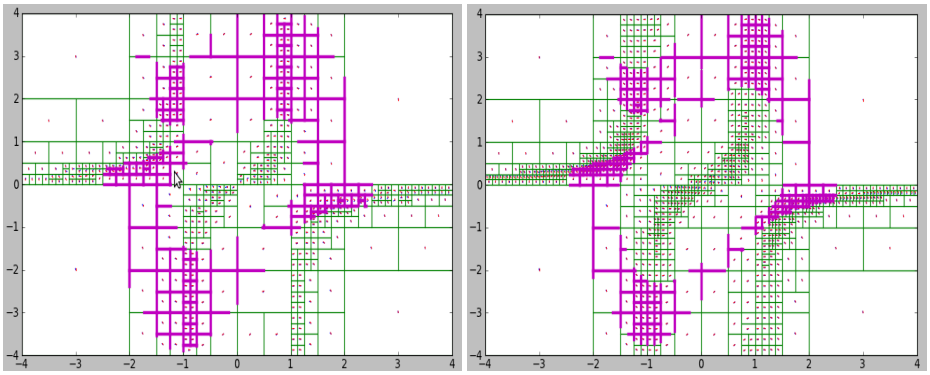

Figure 18: Van der Pol oscillator GSPDIs and reach sets. a) 378 regions. b) 789 regions.

of Figure 16 shows that from the initial point, located at $(-3.05, -0.05)$, the GSPDI only evolves in an ellipse around the equilibrium point $(0, 0)$, consistent with the damped pendulum's evolution slowly spiraling in towards $(0, 0)$.

The cycles accelerated in Figure 16 are all continuous, so we show how the tool handles disjoint cycles in Figure 17. Please note that demonstrating a disjoint cycle requires the angles $|\angle_{\mathbf{a}}^{\mathbf{b}}|$ to be so small as to appear to be single vectors in all regions in the figure. In Figure 17-a) the tool shows that point 0.5 is reachable from point 0.1 without iterating the cycle. The cycle is iterated once before the tool discovers the presence of a cycle, and then a total number of 9 iterations are skipped to verify that $0.5 \in [0.4573, 0.5323]$. The intervals drawn are the only intervals actually computed by the tool. In Figure 17-b) the tool shows how the cycle's penultimate iteration is calculated with start point 0.1. A total number of 12 iterations are skipped to arrive at the interval $[0.7215, 0.8006]$, and the cycle is left during the next iteration as the limit U is 0.8589.

As an illustration of the speedup in execution time we can get from using disjoint cycle acceleration we have run the tool on the *van der Pol* oscillator [20]:

$$\frac{dx}{dt} = y(t)$$

$$\frac{dy}{dt} = -\mu(x(t)^2 - 1)y(t) - x(t)$$

A typical trajectory of such a system is illustrated in Figure 9, and screenshots of two GSPDIs generated from such a system, along with example reachable sets, are shown in Figure 18. The GSPDIs differ in how coarse they are as approximations to the original system, a GSPDI with more regions is often a finer approximation that one with less regions. In Table 1 we list the times spent on building the reachable sets, starting from points both outside and inside the

GSPDI #	Initial point	Size (regions)	Previous version (2.1)	New version (2.2)
1	(-2, 3.5)	378	50s	6s
2	(-2, 3.5)	789	316s	45s
2	(0.25, 0.25)	789	5s	3.5s

Table 1: Time to build reach set for van der Pol oscillator GSPDIs.

limit cycle. In the table we see that the speedup from the previous version of tool, without disjoint cycle acceleration, to the new version which implements disjoint cycle acceleration, is substantial when disjoint cycle acceleration is applicable. As the reachability computation is performed on a GSPDI there are no other tools with which to directly compare GSPeeDI in order to evaluate its performance. In next section we present an approximation algorithm that outputs a GSPDI based on a non-linear input system.

5. Approximation algorithm

In the previous section we presented an algorithm to efficiently perform reachability analysis of GSPDIs. In this section we present results concerning applying GSPDIs as an approximation model for other complex planar systems for which reachability is hard (undecidable or not known).

5.1. Proportionally controlled TIDISs

In particular we will be dealing with *proportionally controlled TIDISs* (PC-TIDIS, cf. definition 15). We first show that PC-TIDISs are a subclass of CN-HAs (cf. definition 10), and that it is possible to hybridize a CN-HA into a GSPDI. Then we introduce measures of precision which enable us to compare the respective precision of two approximating GSPDIs. Finally we give an algorithm which takes a CN-HA and precision bounds as input, and outputs an approximating GSPDI that respects the precision bounds.

In the rest of the paper we assume that CN-HA and GSPDI have the same domain and range (usually a convex polygon, unless otherwise specified).

Lemma 13 (CN-HAs)**.** *If a hybrid automaton $\mathcal{H}$ is a PC-TIDIS, then it is also a CN-HA.*

Proof. The lemma follows directly from Definition 15: A PC-TIDIS has identity maps as assignments, and the state uniquely determines the current location of the automaton, since any enabled transition is automatically taken. □

The runs of a CN-HA have the same properties as those of a GSPDI.

Lemma 14 (CN-HA trajectory)**.** *The* runs *ξ of a CN-HA $\mathcal{C}$ are continuous and almost-everywhere differentiable functions $\mathbb{R}_{\geq 0} \to \mathbb{R}^2$, and so trajectories.*

Proof. The runs of a CN-HA have $\mathbb{R}^2$ as their image, and time ($\mathbb{R}_{\geq 0}$) as their domain by Definition 12. From Definition 12 we also have that a run of a hybrid automaton consists of a sequence of intervals, where the run is continuous and almost everywhere differentiable in each interval. Since by Definition 10 we do not have assignments in a CN-HA, the runs will also be continuous across interval boundaries and, assuming non-zeno behavior, almost everywhere differentiable. □

Example 13. *The trajectory evolves in location l_1 during time interval $[0, t_1]$ until in reaches the border between l_1 and l_2, where $f_1(t_1) = f_2(0)$.*

Note that, compared to the run of the general hybrid automaton in Figure 6-b), the invariants of the locations do not overlap, and the value of the $CN-HA$ trajectory does not change at the border due to the identity map, although its behavior may do so.

In what follows we characterize what it means for a GSPDI to approximate a CN-HA.

Lemma 15 (Approximation)**.** *Let $\mathcal{C}$ be a CN-HA, and $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$ a GSPDI. If for any region $P \in \mathbb{P}$ and for all trajectories $\xi \in \mathcal{C}$ and points $\xi(t) \in P$ it is the case that $\dot{\xi}(t) \in \angle^{\mathbf{b}_P}_{\mathbf{a}_P}$, then $\mathcal{G} \geq \mathcal{C}$.*

Proof. The lemma follows directly from Lemma 14 and Definitions 12 and 21. □

In the following we assume for all regions $P \in \mathbb{P}$ that $\angle^{\mathbf{b}_P}_{\mathbf{a}_P}$ is the arc with the shortest length such that Lemma 15 holds. If we make finer and finer partitions $\mathbb{P}$ of the domain Q of $\mathcal{C}$, we can generate GSPDIs whose behaviors become more and more restricted while still being approximations of some CN-HA $\mathcal{C}$.

There will be some limit to how restricted the behavior of a GSPDI may be and still remain an over-approximation, as it must contain the behavior of the underlying CN-HA. If we consider the behavior of a single region P, the following definition is useful for finding a lower bound on this behavior.

Definition 33 (Minimal behavior)**.** *For a CN-HA $\mathcal{C}$ with domain Q and a region $P \subseteq Q$, a* minimal behavior point *min_P is a point $min_P \in P$ such that $|\angle^{min^+_P}_{min^-_P}| \leq |\angle^{p^+}_{p^-}|$ for all $p \in P$. The arc length $|\angle^{min^+_P}_{min^-_P}|$ is the* minimal behavior *of P.*

The normalized behavior in a region P can never be less than in one of its minimal behavior points, $|\angle^{\mathbf{b}_P}_{\mathbf{a}_P}| \not< |\angle^{min^+_P}_{min^-_P}|$. This lower bound on the normalized behavior does not get smaller as we partition P, since the resulting sub-partitions may have minimal behavior points with larger behavior.

Lemma 16 (Increasing minimal behavior)**.** *Let $\mathcal{C}$ be a CN-HA with domain Q, $P \subseteq Q$ be a region, and $P' \subseteq P$ be a sub-region of P, then $|\angle^{min^+_P}_{min^-_P}| \leq |\angle^{min^+_{P'}}_{min^-_{P'}}|$.*

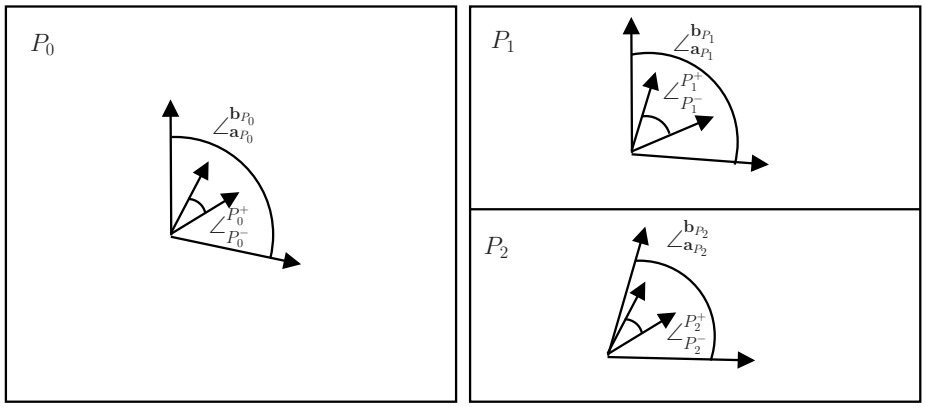

Figure 19: The effect of partitioning on the minimal behavior and arcs of regions.

Proof. By definition $|\angle^{min_P^+}_{min_P^-}| \leq |\angle^{p^+}_{p^-}|$ for all $p \in P$, and P' is contained in P. □

The lemma is illustrated in Figure 19. As we partition region P_0, we see that the difference in length between the minimal behavior and the arc $\angle^{\mathbf{b}}_{\mathbf{a}}$ is smaller in the resulting regions P_1 and P_2 than in P_0. This property forms the basis of the following definition:

Definition 34 (Measures for precision)**.** *Let us assume a CN-HA $\mathcal{C}$ and a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$ such that $\mathcal{G} \geq \mathcal{C}$, and two disjoint sets $\mathbb{X}, \mathbb{Y}$ such that $\mathbb{P} = \mathbb{X} \cup \mathbb{Y}$. Let $\theta : \mathbb{R}^2 \to [0, 2\pi]$ be a function that maps a region $P \in \mathbb{P}$ to $|\angle^{\mathbf{b}_P}_{\mathbf{a}_P}| - |\angle^{min_P^+}_{min_P^-}|$. We will overload this function symbol and let $\theta : 2^{\mathbb{R}^2} \to [0, 2\pi]$ and $\delta : 2^{\mathbb{R}^2} \to [0, 1]$ be functions such that*

1. *$\theta(\mathbb{X})$ is the maximum $\theta(X)$ of all $X \in \mathbb{X}$.*
2. *$\delta(\mathbb{Y})$ is the relative weight of the regions of $\mathbb{Y}$, $\frac{area(\cup \mathbb{Y})}{area(\cup \mathbb{P})}$.*

Let $\Theta \in [0, 2\pi]$ and $\Delta \in [0, 1]$. We say that $\mathcal{G}$ obeys the bounds *Θ and Δ if $\theta(\mathbb{X}) \leq \Theta$ and $\delta(\mathbb{Y}) \leq \Delta$ for partition $\mathbb{P}$ where $\mathbb{P} = \mathbb{X} \cup \mathbb{Y}$.*

Example 14. *For the pendulum given by*

$$\frac{dx}{dt} \in \{y(t)\}$$

$$\frac{dy}{dt} \in \{-\frac{25+e}{100}y(t) - \sin x \mid e \in [-1, 1]\}$$

Polygon P	min_P	$\lvert min_P \rvert$	$\lvert \angle^{\mathbf{b}_P}_{\mathbf{a}_P} \rvert$	$\theta(P)$
$[1,1] \times [2,2]$	$(\pi/2, 1)$	0.003	0.396	0.393
$[1,1.5] \times [2,2]$	$(\pi/2, 1.5)$	0.004	0.157	0.153
$[1.5,1.5] \times [2,2]$	$(\pi/2, 1.5)$	0.004	0.148	0.144
$[1.5,1.75] \times [2,2]$	$(\pi/2, 1.75)$	0.005	0.072	0.067
$[1.75,1.75] \times [2,2]$	$(1.75, 1.75)$	0.005	0.070	0.065
$[1.75,1.875] \times [2,2]$	$(1.75, 2.0)$	0.005	0.040	0.035
$[1.875,1.875] \times [2,2]$	$(1.875, 2.0)$	0.005	0.036	0.031
...				
$[1.999,1.999] \times [2,2]$	$(1.999, 2.0)$	0.005	0.0053	0.0003

Table 2: Partitioning of the damped pendulum.

we show the decreasing precision measure $\theta(P)$ by partitioning a region $[1,1] \times [2,2]$ as in Table 2. The increasing size of the minimal behavior $|min_P|$ and decreasing size of the arc $\angle^{\mathbf{b}_P}_{\mathbf{a}_P}$ together force the precision measure $\theta(P)$ to decrease.

Before we show the existence of GSPDIs that approximate any CN-HA $\mathcal{C}$ while still obeying bounds Θ and Δ, we need the following lemma.

Lemma 17 (Vanishing sub-partition)**.** *Let $\epsilon \in \mathbb{R}^+$ and let $X \subseteq Q$ be a set such that $area(X) = 0$. Then there exists a subpartition $\mathbb{Y}$ of Q where each $Y \in \mathbb{Y}$ is a convex polygon and $X \subseteq \cup\mathbb{Y}$, such that $area(\cup\mathbb{Y}) \leq \epsilon$.*

Proof. Since the area of X is 0, X is a collection of one- and zero-dimensional entities. We let the partition $\mathbb{Y}$ be a closer and closer approximation of lines and points respectively, until $area(\cup\mathbb{Y}) \leq \epsilon$. □

For a CN-HA there does exist a GSPDI that obeys any precision bounds.

Lemma 18 (Existence of approximation)**.** *Given an CN-HA $\mathcal{C}$ and bounds Θ and Δ, there exists a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$, $\mathcal{G} \geq \mathcal{C}$ such that $\mathcal{G}$ obeys Θ and Δ.*

Proof. The lemma imposes two conditions on the precision of $\mathcal{G}$, namely that both $\mathcal{G}$ and Θ obey Δ.

1. For the first condition we will consider a Lipschitz region $P \in \mathbb{P}_L$ of $\mathcal{C}$. Definition 7 of Lipschitz continuity gives us $d[\widehat{p_i}, \widehat{p_j}] \leq K\|p_i - p_j\|$ for all points $p_i, p_j \in P$, where K is the Lipschitz constant of P. The upper bound on $\|p_i - p_j\|$ is the diameter of the smallest disk containing P, $diam(P)$, thus $d[\widehat{p_i}, \widehat{p_j}] \leq K \cdot diam(P)$. If we make $diam(P) \to 0$ we have $d[\widehat{p_i}, \widehat{p_j}] \to 0$ since K is a constant, and in particular for some minimal behavior point min_P of P, $d[\widehat{p_i}, \angle^{min^+_P}_{min^-_P}] \to 0$. Because of this, and as a consequence of Lemma 16, the behavior arc of P approaches that of some minimal behavior point min_P as P shrinks, i.e. $\angle^{\mathbf{b}}_{\mathbf{a}} \to \angle^{p^+_{minP}}_{p^-_{minP}}$, and

consequently $\theta(P) = |\angle^{\mathbf{b}}_{\mathbf{a}}| - |\angle^{p^+_{minP}}_{p^-_{minP}}| \to 0$. If we repeat this for all $P \in \mathbb{P}_L$, we get $\theta(\mathbb{P}) \to 0$, thus $\theta(\mathbb{P})$ can be made smaller than any Θ.

2. The Lipschitz condition holds in all of Q, except for the arbitrarily small neighborhoods of the non-Lipschitz points, and the border $\beta(Q)$ (as illustrated in Figure 5-a) as the behavior at each side of the border is governed by a different TIDIS. We know by Lemma 17 that there exists a $\mathbb{P}_N$ with $area(\cup\mathbb{P}_N) \leq \Delta \cdot area(Q)$ that contains the non-Lipschitz points and the border, since both have area 0.

Thus, since $\theta(\mathbb{P}) \leq \Theta$ and $area(\cup\mathbb{P}_N) \leq \Delta \cdot area(Q)$, for any Θ and Δ we have that $\mathcal{G}$ obeys both bounds. □

Lemma 18 guarantees that there is always a GSPDI with $\theta(\mathbb{X})$ and $\delta(\mathbb{Y})$ arbitrarily small for sets $\mathbb{X}, \mathbb{Y}$, trivially by letting $\mathbb{P}_L = \mathbb{X}$ and $\mathbb{P}_N = \mathbb{Y}$. To actually arrive at such a GSPDI, one can iteratively partition the domain Q finer and finer with $diam(P) \to 0$ for all $P \in \mathbb{P}$. For that purpose, we assume a function *doPartition*, which when applied to a partition of Q produces a sub-partition of convex polygons, for instance by splitting one particular polygon of the current partition. In the following we will assume that the *doPartition* function is being applied in a breadth-first manner, but other strategies might be employed.

Lemma 19 (Partition)**.** *Assume a CN-HA $\mathcal{C}$, bounds Θ and Δ, and that the doPartition function is being employed following a breadth-first strategy on Q. Then in a finite number of steps a partition $\mathbb{P}$ is generated such that there exists a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$ with $Q = \cup\mathbb{P}$, and where $\theta(\mathbb{P}_L) \leq \Theta$, $\delta(\mathbb{P}_N) \leq \Delta$, and $\mathcal{G} \geq \mathcal{C}$.*

Proof. The lemma requires application of *doPartition* iteratively such that $\theta(\mathbb{P}_L)$ and $\delta(\mathbb{P}_N)$ get smaller than the given upper bounds. The breadth-first strategy, where each polygon is split in two equally-sized sub-polygons, guarantees that the regions of the partition of the domain of $\mathcal{G}$ get arbitrarily small, and so Lemma 18 will apply. □

In Algorithm 2 we present a method for realizing Lemma 19. The algorithm takes a CN-HA $\mathcal{C}$ and bounds Θ and Δ as input, and yields as output a partition $\mathbb{P}$ which forms part of a GSPDI $\mathcal{G} = \langle \mathbb{P}, \mathbb{F} \rangle$ with $\mathcal{G} \geq \mathcal{C}$ and where furthermore $\mathbb{P}$ can be divided into two sets, $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$, such that $\theta(\mathbb{P}_{OK}) \leq \Theta$ and $\delta(\mathbb{P}_{BAD}) \leq \Delta$ (cf. Algorithm 2).

To maintain the successively finer partitioning of the given domain Q, the algorithm uses two collections of regions $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$. As loop invariant of the *while* iteration, the union of $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$ is a partition of the initial convex polygon Q. The collection $\mathbb{P}_{OK}$ contains regions P where $\theta(P)$ is less than or equal to Θ. The collection $\mathbb{P}_{BAD}$, on the other hand, contains those regions whose angles are yet to be computed.

Algorithm 2 Construct a GSPDI from a CN-HA with bounds Θ and Δ.

Input: CN-HA $\mathcal{C}$, $\Theta \in [0, 2\pi]$, $\Delta \in [0, 1]$

Empty queue $\mathbb{P}_{BAD}$, and empty collection $\mathbb{P}_{OK}$

$\mathbb{P}_{BAD}.insert(Q)$
while $area(\mathbb{P}_{BAD}) > \Delta \cdot area(Q)$ **do**
  $P := \mathbb{P}_{BAD}.remove()$
  $\angle_{\mathbf{a}_P}^{\mathbf{b}_P} := P.getAngle(P.locations())$
  $|\angle_{min_P^-}^{min_P^+}| := P.getMinimalBehavior(P.locations())$
  if $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| - |\angle_{min_P^-}^{min_P^+}| \leq \Theta$ **then**
    $\mathbb{P}_{OK}.insert(P)$
  else
    $\{P_1, \ldots, P_n\} := P.doPartition()$
    $\mathbb{P}_{BAD}.insert(P_1, \ldots, P_n)$
  end if
end while
return $\mathbb{P}_{OK} \cup \mathbb{P}_{BAD}$

The collection $\mathbb{P}_{BAD}$ keeps the regions in a queue, and during each iteration, the first region P is removed from the head of the queue. For each region we compute the angle $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ and the minimal behavior $|\angle_{min_P^-}^{min_P^+}|$.

If $\theta(P)$ is small enough, i.e., if $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| - |\angle_{min_P^-}^{min_P^+}| \leq \Theta$, then P is considered finished and moved to $\mathbb{P}_{OK}$. Otherwise P is partitioned, and the sub-polygons $P_1, \ldots, P_n$ are placed at the back of the queue $\mathbb{P}_{BAD}$. The while loop is executed until the area of $\mathbb{P}_{BAD}$ is less than or equal to the desired threshold, $\Delta \cdot area(Q)$. The return value is the union of $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$, which is a valid partition $\mathbb{P}$ of Q, satisfying both Θ and Δ.

Note that the algorithm does not compute two sets of convex polygons where the underlying CN-HA is Lipschitz in one and not in the other. Instead, these properties are implicitly used to allow the computation of two sets $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$ where $\theta(P) \leq \Theta$ for all $P \in \mathbb{P}_{OK}$ and where the area of $\cup\mathbb{P}_{BAD} \leq \Delta \cdot area(Q)$ (cf. also Definition 34 which gives the measures of precision).

By the properties of $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$, Algorithm 2 ensures that all the trajectories of the CN-HA are also trajectories of the generated approximating GSPDI (Lemma 15) that is, the algorithm is sound. It also satisfies that $\theta(\mathbb{P}) \leq \Theta$ and $\delta(\mathbb{P}) \leq \Delta$ (Lemma 19), which guarantees completeness, and also termination of the algorithm.

Theorem 4. *Algorithm 2 is sound, complete, and it terminates.*

Proof. The *soundness* of the algorithm is a direct consequence of the approximation Lemma 15: As an invariant, the domain Q is partitioned into regions

$\mathbb{P}$ (split into $\mathbb{P}_{BAD}$ and $\mathbb{P}_{OK}$). Initially, the partition consists of one polygon, Q, and the loop either keeps the partition or refines it by replacing one polygon by sub-polygons. Each iteration/partition corresponds to a GSPDI, which approximates the CN-HA by Lemma 15.

As for *completeness:* the algorithm works by successively partitioning the polygons of $\mathbb{P}_{BAD}$. For each P considered, there are two options: Either $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| \leq \Theta$, in which case it is moved from $\mathbb{P}_{BAD}$ to $\mathbb{P}_{OK}$, or not.

The question is whether the area of $\mathbb{P}_{BAD}$ eventually will be less than $\Delta \cdot area(Q)$. By Lemma 19 and its proof we know that our strategy for applying *doPartition* will generate two sets $\mathbb{P}_L$ and $\mathbb{P}_N$, the area of the latter which can be made arbitrarily small, and that we can find an arbitrarily small upper bound on $\theta(P)$, for each $P \in \mathbb{P}_L$. So we let Θ be an upper bound of these $\theta(P)$, eventually forcing $\mathbb{P}_{BAD} \subseteq \mathbb{P}_N$. By having the upper bound of $area(\cup\mathbb{P}_N)$ as $\Delta \cdot area(Q)$, we have that $\theta(\mathbb{P}_{OK}) \leq \Theta$ and $\delta(\mathbb{P}_{BAD}) \leq \delta(\mathbb{P}_N) \leq \Delta$.

Finally, the algorithm *terminates* when the area of $\mathbb{P}_{BAD}$ is less than $\Delta \cdot area(Q)$. The proof of completeness shows that this is always possible to achieve. In addition, Lemma 19 guarantees that the breadth-first strategy will generate a $\mathbb{P}_N$ with a sufficiently small area in a finite number of steps. □

6. Related work

In this section we will mostly focus on other works concerning the reachability computation for hybrid automata, which we discussed in Section 4, and on other approaches for approximating the behavior of non-linear dynamics, which was the topic of Section 5.

The seminal paper on algorithmic analysis of hybrid automata is [21]. Here the notion of *time simulation* is introduced, which give a formal definition of the relation between reachability in an original system, and reachability in an approximation. The idea of partitioning is introduced and defined as a finite set of predicates on the flow in each location, a so-called *flow split*. Each flow transition of the transformed automaton will have an alternating sequence of flow and silent jump transitions with only the identity map for assignments. It has been proved that every hybrid automaton is splittable. Nonlinear hybrid automata are approximated by linear hybrid automata, for which there is exists an efficient, though not necessarily terminating, algorithm for deciding reachability [13]. Two methods given in the paper for generating approximations are the *clock translation* and *linear phase-portrait* algorithms. Clock translation, replacing each variable x by a clock t_x, i.e. that $\frac{dt_x}{dt} = 1$, is applicable if x is an independent, monotonically determined variable. The reachability problem is recursively enumerable for the resulting hybrid automata. The *linear phase-portrait approximation* does not suffer from the restrictions of clock translation, and the resulting automaton over-approximates the original using either piecewise constant bounds as flow (rate translation) or differential inequalities of the form $Ax \geq b$, and linear inequalities to partition the state space. Here each variable is treated independently, whereas we over-approximate the behavior based on the state of both variables x and y.

The above theory has been implemented in the tool Hytech [22], which answers reachability questions in linear hybrid automata. The Hytech+ system [23], is an updated version of Hytech. Hytech+ extends the class of hybrid automata accepted by the system from linear to non-linear dynamics (polynomials, exponentials, and trigonometric functions), while the partitions are limited to hyper-rectangles. Interval ODE solvers are used to compute over-approximations of the continuous part of the hybrid automata.

A tool for solving reachability questions is d/dt [24], which takes input based on linear differential inclusions. This approach has later been extended and the concept of *hybridization* is introduced [14]: The dynamics of a (possibly non-hybrid) non-linear system is transformed into a hybrid automaton with simpler dynamics through over-approximation. In a sense, our work may be considered a special subclass of this technique, and so we have borrowed and adapted their use of meshes (which they restrict to hypercubes) and continuous traces. The continuous development is assumed to be a continuous vector field, and for each cell of the mesh a function is constructed by interpolation, and the approximation error is computed based on either the Lipschitz constant, or the second derivative if applicable.

The PHAVer tool tries [25], in addition to allowing piecewise affine dynamics to be over-approximated to linear dynamics, to remedy the slow convergence of this algorithm. This is done by conservatively limiting the bits of the coefficients of the linear terms, and pruning constraints.

The HSolver tool [26] is able to do safety verification of non-linear hybrid automata through approximation using interval arithmetic. The state space is broken up into hyper-rectangles, and these rectangles are refined into sub-rectangles if their refinement reduces the reachable set of the abstraction. Rectangles that can no longer be reached are pruned away. The abstraction the reachability search is performed in is a discrete transition system, and transitions between rectangles are based on whether there is a trajectory from one to the other in the original system, based on constraint propagation [27]. The tool is general: It supports any number of variables, assignments, and functions including the trigonometric functions. However, the algorithm includes iterating to find fixpoints, where termination of the iteration process is guaranteed by the finite precision of the floating point numbers of the implementation.

Any method that ensures that an over-approximation is truly conservative must also ensure that the results of the numerical computations used in an implementation must also be conservative. This is guaranteed by interval arithmetic [16]. In interval arithmetic the value of π could be represented as the interval $[3.14, 3.15]$, which contains the true value of π.

Methods that over-approximate the flow of non-linear dynamics include interval global optimization methods [28], which is directly applicable for our work, and interval solvers for ordinary differential equation [29]. In the former the extrema of the function to be optimized is sure to be contained in the interval returned by the method. In the latter, we have that while an ordinary ODE solver will compute the best approximation to the value of the input ODE at some time t, the interval ODE solver will compute an interval in which the

value of the input ODE at time t is sure to be contained.

A final comment on the relation between SPDIs and GSPDIs. A GSPDI is a generalization of the less expressive SPDIs (Polygonal Hybrid Systems) [17, 15]; a similar system with the added restriction that all vectors be good vis-a-vis each pair of input and output edge. The reason for having defined SPDIs in the first place (and not GSPDIs) was that the goodness assumption was crucial for proving reachability [17, 15]. It was believed at the beginning that relaxing goodness would make reachability undecidable for such systems (GSPDIs) [30], but it was proved otherwise later [3]. All the results presented in this paper also hold for SPDIs.

7. Conclusion and future work

In this paper we have presented a new reachability algorithm for GSPDIs, with the feature that all cycles in the reachability graph can be accelerated. This is, to our knowledge, a quite remarkable result given the complex nature of the trajectories of GSPDIs.

Besides, we have defined a restricted form of non-linear hybrid automata, the CN-HA, and shown how a proportional controller may be modeled using this representation. CN-HAs can be over-approximated by a class of hybrid automata, the GSPDIs, which have simpler dynamics. We presented an algorithm that takes a CN-HA as input and produces a GSPDI as output, obeying bounds derived from our precision measures.

Exploiting Lipschitz continuity for reachability checking and simulation is not new in itself. It is for instance inherent in the *hybridization* approach of [31], and is also used for hybrid computation [32]. A main difference is that we consider systems that may be Lipschitz continuous only in parts of the plane. A Lipschitz continuous system has an upper bound, the Lipschitz constant, on how fast the system's dynamics changes. We exploit the phenomenon that a system may be Lipschitz continuous almost everywhere, and have different Lipschitz constants for different subsets of the plane. We minimize the area where the system is not Lipschitz continuous, and treat areas where the Lipschitz constant is large more thoroughly than areas where it is small, to get as good an approximation as possible. This comes with a computational price, as we must identify the Lipschitz constant for each area we consider, using non-linear global optimization tools. We need, however, to identify these areas only once, and then we can perform multiple reachability computations.

One line of future work is incorporating support for enhancements, optimizations and utilities currently available for the SPeeDI tool [33], that have been already explored theoretically for SPDIs. This include the computation of the phase portrait of a system [34], which may allow both optimizations [35] and compositional parallelization [36] of the reachability analysis algorithm. Note that the implementation of such features will not add to the complexity of the tool as all the information needed to compute the phase portrait (invariance, viability and controllability kernels, and semi-separatrices) is already computed when analyzing simple cycles (see [36, 35] for more details).

In the future we would like to provide a prototype implementation based on interval global optimization methods [28], and integrate this into the reachability checker GSPeeDI [10, 4]. We can investigate whether other kinds of controllers, such as the Proportional, Integral, Derivative (PID) controllers, can be represented as CN-HAs. To facilitate the analysis of TIDISs regulated by controllers that cannot be represented as CN-HAs, we can look at extending the definition of a GSPDI to a hierarchical GSPDI [37].

References

[1] K. J. Åström, R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, Princeton University Press, 2008.

[2] T. Henzinger, P. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, in: STOC'95, ACM Press, 1995, pp. 373–382.

[3] G. J. Pace, G. Schneider, Relaxing goodness is still good, in: ICTAC'08, Vol. 5160 of LNCS, Springer, 2008, pp. 274–289.

[4] H. A. Hansen, G. Schneider, GSPeeDI –A tool for analyzing generalized polygonal hybrid systems, in: ICTAC'09, Vol. 5684 of LNCS, Springer, 2009, pp. 336–342.

[5] E. Asarin, G. Schneider, Widening the boundary between decidable and undecidable hybrid systems, in: CONCUR'02, Vol. 2421 of LNCS, Springer, 2002, pp. 193–208.

[6] V. Mysore, A. Pnueli, Refining the undecidability frontier of hybrid automata, in: FSTTCS'05, Vol. 3821 of LNCS, Springer, 2005, pp. 261–272.

[7] H. A. Hansen, G. Schneider, M. Steffen, Reachability analysis of non-linear planar autonomous systems, in: FSEN'11, Vol. 7141 of LNCS, Springer, 2012, pp. 206–220.

[8] H. A. Hansen, G. Schneider, Reachability analysis of GSPDIs: Theory, optimization, and implementation, in: SAC-SVT'10, ACM, 2010, pp. 2511–2516.

[9] H. A. Hansen, Safety verification of non-linear, planar proportional control systems with differential inclusions, in: IEEE ICESS-11, IEEE Computer Society, 2011, pp. 1146–1153.

[10] H. A. Hansen, GSPeeDI, `http://heim.ifi.uio.no/hallstah/gspeedi/`.

[11] J. P. Aubin, A. Cellina, Differential Inclusions: Set-Valued Maps and Viability Theory, Springer, 1984.

[12] T. A. Henzinger, The theory of hybrid automata, in: LICS'96, IEEE Computer Society, 1996, pp. 278–292.

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

[14] E. Asarin, T. Dang, A. Girard, Hybridization methods for the analysis of nonlinear systems, Acta Informatica 43 (2007) 451–476.

[15] E. Asarin, G. Schneider, S. Yovine, Algorithmic analysis of polygonal hybrid systems, part I: Reachability, Theoretical Computer Science 379 (1-2) (2007) 231–265.

[16] R. E. Moore, Interval arithmetic and automatic error analysis in digital computing, Ph.D. dissertation, Department of Mathematics, Stanford University, Stanford, CA, USA (Nov. 1962).

[17] E. Asarin, G. Schneider, S. Yovine, On the decidability of the reachability problem for planar differential inclusions, in: HSCC'01, Vol. 2034 of LNCS, Springer, 2001, pp. 89–104.

[18] G. Pace, G. Schneider, Model checking polygonal differential inclusions using invariance kernels, in: VMCAI'04, Vol. 2937 of LNCS, Springer, 2003, pp. 110–121.

[19] R. E. Tarjan, Enumeration of the Elementary Circuits of a Directed Graph, Tech. rep., Cornell University, Ithaca, NY, USA (1972).

[20] B. V. der Pol, J. V. der Mark, Frequency Demultiplication, Nature 120 (1927) 363–364.

[21] T. Henzinger, P.-H. Ho, H. Wong-Toi, Algorithmic analysis of nonlinear hybrid systems, IEEE Transactions on Automatic Control 43 (4) (1998) 540 –554.

[22] T. A. Henzinger, P.-H. Ho, H. Wong-Toi, HyTech: A model checker for hybrid systems, Software Tools for Technology Transfer 1 (1997) 110–122.

[23] T. A. Henzinger, B. Horowitz, R. Majumdar, H. Wong-Toi, Beyond HYTECH: Hybrid systems analysis using interval numerical methods, in: HSCC'00, Vol. 1790 of LNCS, Springer, 2000, pp. 130–144.

[24] E. Asarin, T. Dang, O. Maler, O. Bournez, Approximate reachability analysis of piecewise-linear dynamical systems, in: HSCC'00, Vol. 1790 of LNCS, Springer, 2000, pp. 20–31.

[25] G. Frehse, PHAVer: Algorithmic verification of hybrid systems past HyTech, in: HSCC'05, Vol. 3414 of LNCS, Springer, 2005, pp. 258–273.

[26] S. Ratschan, Z. She, Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement, ACM Transactions in Embedded Computing Systems 6 (1) (2007) 573–589.

[27] S. Ratschan, Efficient solving of quantified inequality constraints over the real numbers, ACM Transactions on Computational Logic 7 (4) (2006) 723–748.

[28] T. Weise, Global Optimization Algorithms Theory and Application , 2nd Edition, E-book, 2009, `http://www.it-weise.de/`.

[29] N. S. Nedialkov, Interval tools for ODEs and DAEs, in: SCAN'06, IEEE Computer Society, 2006.

[30] G. Schneider, Reachability analysis of Generalized Polygonal Hybrid Systems, in: SAC-SV'08, ACM, 2008, pp. 327–332.

[31] E. Asarin, T. Dang, A. Girard, Reachability analysis of nonlinear systems using conservative approximation, in: HSCC'03, Vol. 2623 of LNCS, Springer, 2003, pp. 20–35.

[32] J. D. Dora, A. Maignan, M. Mirica-Ruse, S. Yovine, Hybrid computation, in: ISSAC, 2001, pp. 101–108.

[33] G. Pace, G. Schneider, SPeeDI, `http://www.cs.um.edu.mt/~svrg/Tools/SPeeDI/index.html`.

[34] E. Asarin, G. Schneider, S. Yovine, Towards computing phase portraits of polygonal differential inclusions, in: HSCC'02, Vol. 2289 of LNCS, Springer, 2002, pp. 49–61.

[35] G. Pace, G. Schneider, Static analysis for state-space reduction of polygonal hybrid systems, in: FORMATS'06, Vol. 4202 of LNCS, Springer, 2006, pp. 306–321.

[36] G. Pace, G. Schneider, A compositional algorithm for parallel model checking of polygonal hybrid systems, in: ICTAC'06, Vol. 4281 of LNCS, Springer, 2006, pp. 168–182.

[37] G. Schneider, Algorithmic Analysis of Polygonal Hybrid Systems, Ph.D. thesis, VERIMAG – UJF, Grenoble, France (July 2002).