Reachability Analysis of GSPDIs: Theory, Optimization, and Implementation

Hallstein A. Hansen
Buskerud University College, Norway
University of Oslo, Norway
Hallstein.Asheim.Hansen@hibu.no

Gerardo Schneider
University of Gothenburg, Sweden
University of Oslo, Norway
gersch@chalmers.se

ABSTRACT

Analysis of systems containing both discrete and continuous dynamics, hybrid systems, is a difficult issue. Most problems have been shown to be undecidable in general, and decidability holds only for few classes where the dynamics are restricted and/or the dimension is low. In this paper we present some theoretical results concerning the decidability of the reachability problem for a class of planar hybrid systems called Generalized Polygonal Hybrid Systems (GSPDI). These new results provide means to optimize a previous reachability algorithm, making the implementation feasible. We also discuss the implementation of the algorithm into the tool GSPeeDI.

Categories and Subject Descriptors

I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Verification of hybrid systems

Keywords

Hybrid systems, reachability, differential inclusions

1. INTRODUCTION

Hybrid systems combine dynamic and discrete behaviors, and mathematical models can be defined for systems arising from real scenarios (e.g., a chemical plant) as well as for artificial constructions (e.g., by *hybridizing* a complex differential equation into connected piece-wise smaller equations). These systems are generally hard to analyze: most important verification problems are undecidable for non-trivial classes of hybrid systems. In this paper we deal with a class of planar hybrid systems whose dynamics is given by differential inclusions: *generalized polygonal hybrid systems* (GSPDIs). Informally, a GSPDI consists of a partition of the plane where each *region* (convex polygon) has associated a pair of vectors determining a constant differential inclusion restricting the dynamics of trajectories. An example of a GSPDI and of a typical trajectory is given in Fig. 1. A complicating factor in the reachability analysis of GSPDIs is the presence of regions where the trajectory is allowed to enter and leave the region through the same edge (boundary of a region), to *slide* along, or *bounce* off a given edge. In Fig. 1 the dynamics of region P allow the trajectory to slide and bounce off the edge e. A region where no trajectory can enter and leave through the same edge is said to be *good*, and a GSPDI where all the regions are good is called an SPDI (the SPDI satisfies the *goodness assumption*).

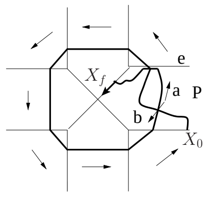

Figure 1: Example GSPDI, and trajectory

The reachability problems for SPDIs and GSPDIs have been shown to be decidable in [1], and [5] respectively. In the latter, there is a need to take special considerations in order to treat simple cycles containing bounces, to analyze the sliding on edges, and to prove termination. Moreover, in order to simplify some proofs many assumptions are made and left implicit.

In this paper we further investigate the theoretical results concerning reachability analysis of GSPDIs, which serve as basis to important optimizations making the implementation of the algorithm feasible. In particular, the contributions of this paper are: (i) A proof showing that the special treatment of simple cycles containing bounces can be avoided, and other theoretical results; (ii) The application of such results in order to optimize the reachability algorithm; (iii) All the assumptions are made explicit, making the implementation feasible.

The paper is organized as follows. In the next section we recall GSPDIs and other preliminary results. In section 3 we informally discuss the decidability results known for SPDIs and GSPDIs previously presented, and we sketch our solution. In sections 4 and 5 we present our main theoretical results. We discuss how these results are applied to optimize the algorithm and we briefly discuss the tool GSPeeDI in section 6, before concluding.

2. GENERALIZED POLYGONAL HYBRID SYSTEMS (GSPDI)

In this section we recall the main definitions and concepts required in the rest of the paper. For a more detailed presentation see [2, 5].

A vector $\mathbf{x} \in \mathbb{R}^2$ is denoted (x_1, x_2). The inner (scalar) product $\mathbf{xy} = x_1y_1 + x_2y_2$, while $\lambda\mathbf{x} = (\lambda x_1, \lambda x_2)$. The vector $\hat{\mathbf{x}}$ is defined as $(x_2, -x_1)$ ($\mathbf{x}$ rotated clockwise $\frac{\pi}{2}$ degrees), and $\hat{\mathbf{x}}\mathbf{x} = 0$. The angle $\angle_{\mathbf{y}}^{\mathbf{x}}$, where $\mathbf{y}$ is situated clockwise from $\mathbf{x}$, denotes a differential inclusion. $\mathbf{z} \in \angle_{\mathbf{y}}^{\mathbf{x}}$ means that $\mathbf{z}$ is a (positive) linear combination of $\mathbf{x}$ and $\mathbf{y}$: $\mathbf{z} = \alpha\mathbf{x} + \beta\mathbf{y}, \alpha \geq 0, \beta \geq 0$.

Definition 1. A *Generalized Polygonal Hybrid System* (GSPDI) is a pair $H = \langle \mathbb{P}, \mathbb{F} \rangle$, where $\mathbb{P}$ is a finite partition of the plane (each $P \in \mathbb{P}$ is a convex polygon, called a *region* of the GSPDI), and $\mathbb{F}$ is a function associating a pair of vectors to each polygon: $\mathbb{F}(P) = (\mathbf{a}_P, \mathbf{b}_P)$. In a GSPDI every point on the plane has its dynamics defined according to which polygon it belongs to: if $\mathbf{x} \in P$, then $\dot{\mathbf{x}} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$.

Definition 2. A *trajectory segment* of a GSPDI is a continuous and almost-everywhere differentiable function $\xi : [0, T] \to \mathbb{R}^2$, such that for all $t \in [0, T]$, if $\xi(t) \in P$, then $\xi'(t) \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$. The *signature* of a trajectory segment, is an ordered sequence of edges $\mathsf{Sig}(\xi) = e_1 \dots e_n$ traversed by ξ.

We define $E(P)$ to be the set of edges of region P. We say that an edge $e \in E(P)$ is an *entry-only* of P if for all $\mathbf{x} \in e$ and for all $\mathbf{c} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$, $\mathbf{x}+\mathbf{c}\epsilon \in P$ for some $\epsilon > 0$. e is *exit-only* if the same condition holds for some $\epsilon < 0$. 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, and denote it by $InOut(()P)$

Definition 3. 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 it as an SPDI.

Given a region P, we introduce a one-dimensional coordinate system on each edge. For this edge we choose a point of origin, with a radius vector $\mathbf{v}$ and a director vector $\mathbf{e}$. The vector $\mathbf{e}$ has a clockwise direction with respect to the boundary 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 an input or an output edge, e and e^{-1}.

We characterize the edge e by its extreme points $e^l, e^u \in \mathbb{Q} \cup \{-\infty, \infty\}$, such that $e = \{\mathbf{v} + x\mathbf{e} | e^l \leq x \leq e^u\}$. 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* $(e, [x, y])$ denote the interval between two local coordinates x, y of e.

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.

Definition 4. A *positive affine function* $f : \mathbb{R} \to \mathbb{R}$ is a function such that $f(x) = ax + b, a > 0$. The inverse of this function is $f(x)^{-1} = \frac{1}{a}x - \frac{b}{a}$

Definition 5. An *affine multivalued function* (AMF) $F : \mathbb{R} \to 2^{\mathbb{R}}$, written $F = [f_l, f_u]$, is defined by $F(x) = [f_l(x), f_u(x)]$ where f_l and f_u are positive affine.

For an interval $I = [l, u]$ we have that $F(l, u) = [f_l(l), f_u(u)]$. An inverted affine multivalued function $F^{-1} : \mathbb{R} \to 2^{\mathbb{R}}$, is defined by $F^{-1}(x) = [f_u^{-1}(x), f_l^{-1}(x)]$.

We also need a way, given an inout edge e, to translate between local coordinates when the edge is seen as an input edge, and as an output edge. A point (e, x) is translated into $(e, flip(x))$, where $flip(x) = 1 - x$, is a negative affine function. An interval $(e, [l, u])$ is translated into $(e, Flip([l, u])$, where $Flip(l, u) = [flip(u), flip(l)]$ [5].

Definition 6. Given an AMF F and two intervals $S \subseteq \mathbb{R}^+$ and $J \subseteq \mathbb{R}^+$, a *truncated affine multivalued function* (TAMF) $\mathcal{F}_{F,S,J} : \mathbb{R} \to 2^{\mathbb{R}}$ is defined as follows: $\mathcal{F}_{F,S,J}(x) = F(x) \cap J$ if $x \in S$, otherwise $\mathcal{F}_{F,S,J}(x) = \emptyset$. In what follows we will write $\mathcal{F}$ instead of $\mathcal{F}_{F,S,J}$. For convenience we write $\mathcal{F}(x) = F(\{x\} \cap S) \cap J$ and $\mathcal{F}(I) = F(I \cap S) \cap J$, where I is an interval.

We say that F is normalized if $S = Dom(\mathcal{F}) = \{x | F(x) \cap J \neq \emptyset\}$ and $J = Im(\mathcal{F}) = \mathcal{F}(S)$. An inverted truncated affine multivalued function (inverted TAMF) is defined in terms of an inverted affine multivalued function. TAMFs are closed under composition [2].

In what follows we define the edge-to-edge successor for a particular vector $\mathbf{c}$, and for all vectors in $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$.

Definition 7. Let $e_i \in In(P)$, $e_o \in Out(P)$, $\mathbf{x}_i = (e_i, x_i)$ and $\mathbf{c} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$. The *edge-to-edge* successor following $\mathbf{c}$ is $\mathsf{Succ}_{e_i e_o}^{\mathbf{c}}(x_i) = x_o$, where $\mathbf{x}_o = (e_o, x_o)$ such that $\mathbf{x}_o = \mathbf{x}_i + t\mathbf{c}$ (for some $t \geq 0$). 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* $\mathsf{Succ}_{e_i e_o}^{\mathbf{c}}$.

Definition 8. For (e_i, I), $\mathsf{Succ}_{e_i e_o}(I)$ is the set of points on e_o reachable from some point in I by a trajectory segment ξ. That is, $\xi(0) \in (e_i, I), \xi(t) \in e_o, t > 0, \mathsf{Sig}(\xi) = e_i e_o$. If e_0 is entry-only and e_1 is exit-only then we say that $\mathsf{Succ}_{e_0 e_1}^{\mathbf{c}}$ is *good.*

The following lemma shows how the successor $\mathsf{Succ}_{e_i e_o}^{\mathbf{c}}$ is used to construct the successor $\mathsf{Succ}_{e_i e_o}$. For $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$, (e_i, x), $[l, u] \subseteq [0, 1]$ we have [2]:

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

Definition 9. Given a sequence of distinct edges $e_1 \dots e_n$, and the sequence $e_1 \dots e_n e_1 \dots e_n \dots e_1 \dots e_n$, we say that $e_1 \dots e_n e_1$ is a *simple cycle*, and we denote it $(e_1 \dots e_n)$.

3. REACHABILITY ANALYSIS OF GSPDIS

In order to better understand our contributions we informally explain in what follows the reachability algorithm for SPDIs [1] and GSPDIs [5].

The algorithm presented in [1] for deciding reachability on SPDI depends on the pre-processing of trajectory segments and a qualitative analysis to guarantee that we can treat all *signatures*, by looking at only a finite set of abstract signatures. Informally, this is achieved as follows: (1) Trajectory segments are simplified — it is sufficient to look at trajectories made up of straight segments across regions, and

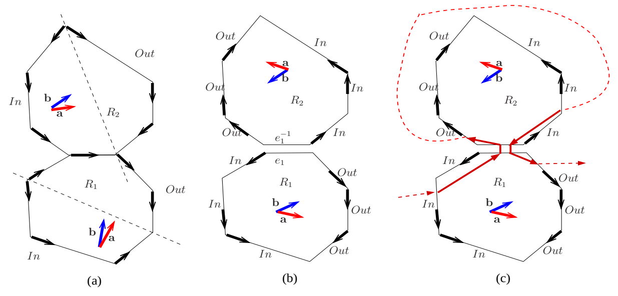

Figure 2: (a) An SPDI with matching order of edges; (b) a GSPDI with a duplicated inout edge; (c) a path through the GSPDI using edge e_1 in both directions.

which do not cross themselves; (2) Trajectory segments are abstracted into signatures, based on the *Poincaré map* that relates n-dimensional continuous-time systems with $(n-1)$-dimensional discrete-time systems; (3) It is shown that it is sufficient to look at signatures which consist only of sequences of edges and simple cycles; (4) Such signatures can be abstracted into *types of signatures* — signatures which do not take into account the number of times each simple cycle is iterated. Many of the lemmas for proving that the above guarantee the finiteness of types of signatures critically depend on the goodness assumption, which propagate this dependency to the constructive proof given for deciding reachability of SPDIs. More specifically, the proof of decidability depends on the concept of monotonicity of TAMFs and their composition. Before starting the analysis, the direction of the edges separating regions is fixed. An interesting result guarantees that the orientation of the edges in each polygon can be split into two contiguous sequences of edges — one being entry-only edges, the other being exit-only edges. Furthermore, the orientation of an edge in one region is guaranteed to match the orientation of the same edge in the adjacent region as shown in Fig. 2-(a).

When one moves on to GSPDIs, *inout* edges break this result since the direction of an edge when considered as an input edge clashes with the direction it is given when used as an output edge in the same region. (Entry-only edges and exit-only edges can be assigned in one fixed direction and do not cause any problem; see Fig. 2-(b).) To solve the above problem the solution suggested in [5] is to use directed edges, and differentiate between the edge used as an input, and when it is used as an output, just as though they were two different edges in the GSPDI. Fig. 2-(c) shows how an inout edge can be seen in this manner. Note that depending on in which direction the trajectory traverse the inout edge e_1, it is an input edge in region R_1, but an output edge in region R_2, and similarly, e_1^{-1} is an output edge in region R_1 and an input edge in region R_2 (hence, we did not draw the direction vector in the picture). In other words, any path passing through the edge such as $\sigma = e_0 e_1 e_2 \ldots e_3 e_1^{-1} e_4$ (see Fig. 2-(c)) can be analyzed as before. Since e_1 and e_1^{-1} are considered distinct edges the above path contains no cycle.

It can be seen that the standard analysis for SPDIs works well for such cases. However, paths can now 'bounce' off an edge. Recall that any pair of edges $e_0 e_1$ is part of a path if e_0 is an input edge of a region, and e_1 is an output edge of the same region. One can 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 the *Flip* TAMF. This breaks the analysis used in SPDIs, to accelerate the analysis of simple cycles. For GPSDIs the standard SPDI analysis is then extended to handle such 'bounces' in paths. There are some problems with the solution sketched above [5]: (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.

Our solution to the above problems (contributions of this paper) include: (i) A proof showing that the treatment of simple cycles containing bounces can be avoided; (ii) All the assumptions are made explicit, making the implementation feasible.

Our reachability is not based on the one presented in [5] (which is a depth-first search algorithm), but rather on an adapted version of the breadth-first search algorithm for SPDIs shown in [4].

This algorithm is conducted in a standard manner on a directed graph where the edges are nodes and successors are transitions. From an initial edge-interval all possible child edge-intervals are generated and put into a queue. These are then handled in turn. The search is finished whenever some goal edge-interval is reached (success), or 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 $\mathsf{Succ}_{s e_m}$, iterating a cycle s some k number of times and then exiting to an edge e_m not on s.

4. EDGE-TO-EDGE SUCCESSORS FOR INOUT EDGES

The presence of inout edges in a GSPDI complicates the construction of edge-to-edge successors $\mathsf{Succ}^{\mathbf{a}}_{e_i e_o}$ and $\mathsf{Succ}^{\mathbf{b}}_{e_i e_o}$ where either or both of e_i or e_o are inout edges. If e_i is entry-only and e_o is exit-only we can easily see how positive affine functions can be created using what we will refer to as the *standard construction* [2]. In region R_1 of Fig. 2(a) we see that we can start at any point on any entry-only edge, follow either **a** or **b**, and intersect the line through any exit-only edge. But let us consider R_1 in Fig. 2-(b). The edge e_1 is inout, so if we consider it to be an entry edge, we see that we can never reach any exit edge by following **b** in its positive direction. If we consider it to be an exit edge, we see that we can never reach it from any entry edge by following **a** in its positive direction.

We want to develop an alternative construction that enables us to construct edge-to-edge successors for non-good vectors and still preserve reachability. Before we show that it is possible, and demonstrate the alternative construction, we will present the source of the complications. We will consider a single vector **c** in place of **a** and **b**, as the analysis is similar for the two vectors. The few differences will be explcitly mentioned.

We are interested in the following possibilities of vector **c** w.r.t. e_i: 1) it can point in across e_i $(0 < \angle^{\mathbf{c}}_{\mathbf{e}_i} < \pi)$, 2) it can be parallel to e_i $(\angle^{\mathbf{c}}_{\mathbf{e}_i} = 0$ or $\angle^{\mathbf{c}}_{\mathbf{e}_i} = \pi)$, 3) it can point out across e_i $(\pi < \angle^{\mathbf{c}}_{\mathbf{e}_i} < 2\pi)$.

We are also interested in the angle between e_i and e_o. We have: a) that $0 < \angle^{\mathbf{e}_o}_{\mathbf{e}_i} < \pi$, b) that $\angle^{\mathbf{e}_o}_{\mathbf{e}_i} = 0$ or $\angle^{\mathbf{c}}_{\mathbf{e}_i} = \pi$, and c) that $\pi < \angle^{\mathbf{e}_o}_{\mathbf{e}_i} < 2\pi$.

Combining the above we get the different cases we see in

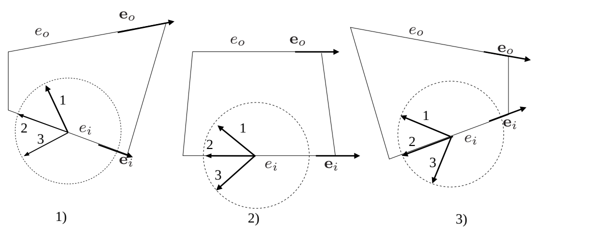

Figure 3: Three regions, showing arcs where vectors 1) point in across e_i, 2) are parallel to e_i, 3) and point out across e_i.

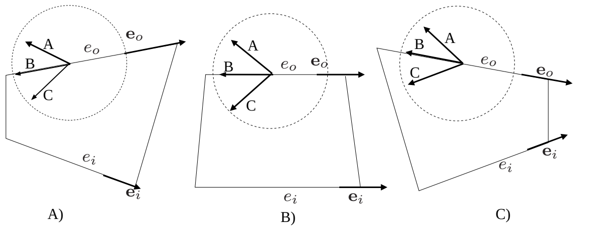

Figure 4: Three regions, showing arcs where vectors A) point out across e_o, B) are parallel to e_o, C) and point in across e_o.

Fig. 3. The three thick arrows are example vectors of the cases i), ii) and iii) above, and the three regions illustrates the relationship between e_i and e_o.

Similarly, we can express the relation between vector $\mathbf{c}$ and e_o: A) it can point out across e_o ($0 < \angle^{\mathbf{c}}_{\mathbf{e}_i} < \pi$), B) it can be parallel to e_o ($\angle^{\mathbf{c}}_{\mathbf{e}_o} = 0$ or $\angle^{\mathbf{c}}_{\mathbf{e}_o} = \pi$), and C) it can point in across e_o ($\pi < \angle^{\mathbf{c}}_{\mathbf{e}_o} < 2\pi$). Fig. 4 shows the corresponding cases for e_o.

We take into consideration that $\mathbf{c}$ must be seen in relation to both e_i and e_o at the same time. We have nine different cases, $\{1, 2, 3\} \times \{A, B, C\}$, not all of which are valid for each kind of region, as shown in Fig. 5.

Having obtained these cases, we want to construct edge-to-edge-successors, noting the following:

LEMMA 2. *It is not possible to construct any affine or constant function following* $\mathbf{c}$ *when* $\angle^{\mathbf{c}}_{e_o} > \pi$ *and* $\mathbf{c}$ *points in across* e_o. *(Cases 1C, 2C and 3C).*

LEMMA 3. *It is not possible to construct any affine or constant function when* $\angle^{\mathbf{c}}_{e_o} = 0$ *or* $\angle^{\mathbf{c}}_{e_o} = \pi$*, and* $\mathbf{c}$ *is parallel to* e_o. *(Cases 1B, 2B and 3B). However, if* $0 < \angle^{\mathbf{c}}_{e_i} < \pi$ *(case 1B), then it is possible to construct the function* $f : \mathbb{R} \to \mathbb{R}^2$ *defined such that* $f(x) = I$ *if* $x = c$ *and* $f(x) = \emptyset$ *if* $x \neq c$*, for some* c*, where* I *is either* $[-\infty, d]$ *or* $[d, \infty]$*, for some* d.

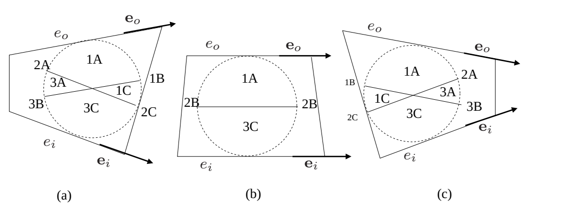

Figure 5: The arcs that show the different cases of vector c in relation to e_i and e_o.

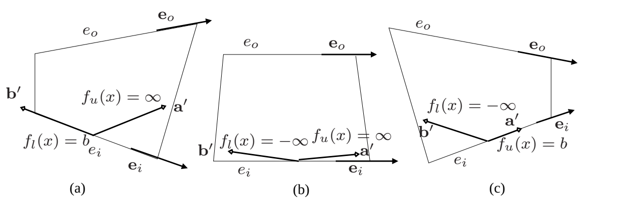

Figure 6: Approximate vectors $\mathbf{a}'$ and $\mathbf{b}'$ and their affine functions $f_u(x)$ and $f_l(x)$

In what follows we consider only the cases where $\mathbf{c}$ points out across e_o: the 'A' cases.

LEMMA 4. *If* $\angle^{\mathbf{c}}_{e_i} > \pi$ *(case 3A) the standard construction gives us a negative affine function:* $f(x) = ax+b, a < 0$.

LEMMA 5. *If* $\angle^{\mathbf{c}}_{e_i} = 0$ *or* $\angle^{\mathbf{c}}_{e_i} = \pi$ *(case 2A) the standard construction gives us a constant function:* $f(x) = b$.

In order to construct a successor $\mathsf{Succ}_{e_i e_o}$ when it may be that neither $\mathbf{a}$ nor $\mathbf{b}$ are good, we use the following procedure (we show this for $\mathbf{a}$ only, as $\mathbf{b}$ is handled similarly).

- If the vector is of case 1A, then it is good, and we can use the standard construction and get the positive affine function $f_u(x) = ax + b, a > 0$.
- Otherwise, exchange the vector $\mathbf{a}$ for a new vector $\mathbf{a}'$ which direction is moved in the counter-clockwise direction from $\mathbf{a}$, until is moved out of 1A into one of case 1B, 2A or 2B.
- If $\mathbf{a}'$ is parallel to $\mathbf{e}_i$ but not to $\mathbf{e}_o$ (case 2A), let $f_u(x) = b$.[1]
- For case 2B we approximate by a constant function $f_u(x) = \infty$, based on a vector that intersects the line through e_o at infinity.
- For case 1B, let $f_u(x) = \infty$. We may have to handle a special situation, according to lemma 3.

The results of applying the procedure outlined above, for our three regions, can be seen in Fig. 6. We let the modified vectors $\mathbf{a}'$ and $\mathbf{b}'$ start in the center of e_i.

From all of the above we have the following result.

THEOREM 1. *For any region* P*, with dynamics* $\angle^{\mathbf{b}}_{\mathbf{a}}$*, such that* $\angle^{\mathbf{b}}_{\mathbf{a}} \leq \pi$*, and two edges* $e_i, e_o \in E(P)$*, we can always find two good vectors* $\mathbf{a}', \mathbf{b}'$ *such that* $\mathsf{Succ}^{\angle^{\mathbf{b}}_{\mathbf{a}}}_{e_i e_o}(I) = \mathsf{Succ}^{\angle^{\mathbf{b}'}_{\mathbf{a}'}}_{e_i e_o}(I)$*, for any* $I \subseteq [0, 1]$.

Example. Consider the partial GSPDI of Fig. 7, with the cycle $(e_1 e_2 e_3)$. The successors $\mathsf{Succ}_{e_1 e_2}$ and $\mathsf{Succ}_{e_3 e_1}$ are good. We assume they are based on the following (general) AMFs: $F_{e_1 e_2}([l, u]) = [a_1 l + b_1, c_1 u + d_1]$, and $F_{e_3 e_1}([l, u]) = [a_3 l + b_3, c_3 u + d_3]$. For successor $\mathsf{Succ}_{e_2 e_3}$ we need to apply the procedure described above, after which we end up with the following AMF: $F_{e_2 e_3}([l, u]) = [r, \infty]$. We want to compute $F_{e_1 e_2 e_3 e_1}$. The composition $F_{e_2 e_3} \circ F_{e_1 e_2}$ gives us: $F_{e_1 e_2 e_3}([l, u]) = [0(a_l + b_1) + r, 0(c_1 u + d_1) + \infty] = [r, \infty]$.

[1] b can be determined as in the proof of lemma 5, which is not presented here due to lack of space.

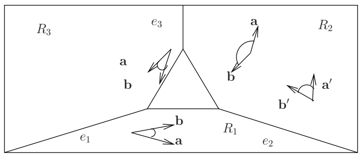

Figure 7: Part of GSPDI. The region R_2 shows both original dynamics $\angle_{\mathbf{a}}^{\mathbf{b}}$ and modified dynamcs $\angle_{\mathbf{a}'}^{\mathbf{b}'}$.

No matter which interval we start with, we end up at $[r, \infty]$. The composition $F_{e_3e_1} \circ F_{e_1e_2e_3}$ gives us: $F_{e_1e_2e_3e_1}([l,u]) = [a_3r + b_3, c_3\infty + d_3] = [a_3r + b_3, \infty]$.

Note that composition with a constant function always yields another constant function.

5. THEORETICAL CONTRIBUTIONS TO GSPDI REACHABILITY ANALYSIS

A directed graph may contain an unmanageably large number of simple cycles. In order to perform reachability analysis they may all have to be identified, and have their combined successors computed and analyzed in order to compute the *exit sets*, the edge-intervals that may be reached after iterating the cycles any k number of times, $k > 1$. We want to reduce this number by identifying properties of cycles that do not need to be iterated more than once. These cycles are thus not required to be explicitly identified and treated. In particular we want to avoid cycles containing bounces.

Definition 10. Given a signature $\sigma = e_0e_1 \dots e_n$, a pair of edges e_je_{j+1} is said to be a *bounce* if $e_{j+1} = e_j^{-1}$. That is, the edge e_j is visited twice in immediate succession.

The following result shows that no cycle needs to be iterated more than once as part of a search, if we can begin that iteration with a full edge-interval $(e, [0,1])$, for any edge e. Let s be the cycle $(e_1 \dots e_n)$. Let r_1 be the sequence of edges $e_i \dots e_ne_1$ $(1 < i \leq n)$, and r_2 be the sequence $e_1 \dots e_j$ $(1 \leq j \leq n)$. Let $P \in \mathbb{P}$, such that $e_j \in In(P)$, $e_x \in Out(P)$, and $\mathsf{Succ}_{e_je_x} \neq \emptyset$, and $e_m \notin s$.

LEMMA 6. $\mathsf{Succ}_{e_je_m}(\mathsf{Succ}_{r_1s^kr_2}(I)) \subseteq \mathsf{Succ}_{e_je_m}(\mathsf{Succ}_{r_2}([0,1]))$

We define the union of two closed intervals A and B, $A \cup B \neq \emptyset$, that are overlapping or adjacent to be a new closed interval $I = [l, u]$, where l is the lower end point, and u the upper end point, of A and B.

We want to detect as many full edge-intervals as possible, since their presence is beneficial to the analysis. The following lemma shows that if two edge-intervals are adjacent or overlapping, then we only need consider the successors of their *union.* This result is useful during reachability search, as we might reduce the number of edge-intervals, and increase the number of full edge-intervals.

LEMMA 7. *Let $A = [l, a]$ and $B = [b, u]$ be closed intervals such that $l < b \leq a < u$ (A and B are adjacent or overlapping). Let $I = [l, a] \cup [b, u]$. Then, for any sequence of edges $\sigma = e_1 \dots e_n$, $\mathsf{Succ}_\sigma(I) = \mathsf{Succ}_\sigma(A) \cup \mathsf{Succ}_\sigma(B)$.*

Some edge-intervals are full *a priori* - we can consider them full as soon as a search reaches the edge in question. The following kind of edges have this property:

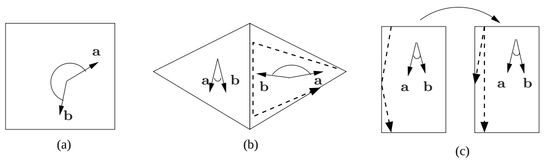

Figure 8: (a) Reach-all region (b) Reach-all edge with bouncing trajectory (c) Bouncing trajectory on the left made redundant by trajectories on the right.

Definition 11. An edge e is *reach-all* iff $\forall x, y \in e, \exists \xi : [0,t] \to \mathbb{R}^2, t \geq 0. \xi(0) = x \wedge \xi(t) = y \wedge Sig(\xi) = e$.

We also formulate a stronger property as well, that all edges of a special kind of region become full if just a single point on any of those edges is reached:

Definition 12. A reach-all region P is a region such that $\forall e, e' \in P \wedge \forall x \in e \wedge \forall y \in e' \exists \xi : [0,t] \to \mathbb{R}^2, t \geq 0. \xi(0) = x \wedge \xi(t) = y \wedge Sig(\xi) = ee'$.

It is obvious that all the edges of a reach-all region are reach-all as well. The following two lemmas let us identify reach-all regions and edges. Regions where the dynamics have angles larger than π are reach-all regions, and edges where we can slide in both directions are reach-all.

LEMMA 8. *For $P \in \mathbb{P}$, if $\angle_{\mathbf{a}}^{\mathbf{b}} > \pi$ then P is a reach-all region.*

An example of a reach-all region can be found in Fig. 8-(a). Let $P, P' \in \mathbb{P}$, e an edge of P and P', and $x, y, z \in e$, such that $x \leq y \leq z$.

LEMMA 9. *If $\exists \xi$, $\dot{\xi} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$, and $\exists \xi'$, $\dot{\xi}' \in \angle_{\mathbf{a}_{P'}}^{\mathbf{b}_{P'}}$, such that $\forall x, y, z, \exists t, t' \in \mathbb{R}, \xi(0) = y, \xi(t) = x, sig(\xi) = e$ and $\xi'(0) = y, \xi'(t') = z, sig(\xi) = sig(\xi') = e$, then e is a reach-all edge.*

See Fig. 8-(b) for an example of a reach-all edge.

We now return to handling bounces. We will show that bounces, for reachability purposes, either are redundant, or appear as the result of a reach-all edge. Thus no cycle needs to include a bounce to preserve reachability. First we look at redundant bounces, which only follow the dynamics of one region.

LEMMA 10. *Let ξ be a trajectory segment containing a bounce. Assume that $\dot{\xi} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$. Then reachability is preserved by the existence of trajectory segments ξ' and ξ'', such that $\xi'(0) = x_1, \xi'(t') = x_2, t' \in \mathbb{R}$ and $\xi''(0) = x_1, \xi''(t'') = x_3, t'' \in \mathbb{R}$ and $sig(\xi') = e_1e_2$ and $sig(\xi'') = e_1e_3$.*

Fig. 8-(c) shows a bounce as described in the above lemma, as well as the trajectory segments that make it redundant.

A bounce may also follow the dynamic of the region on the other side of the edge that is bounced off. Let $t_2' \in \mathbb{R}, t_2 < t_2' < t_3$. Let $x_2' \in e_2$ be a point outside the hull of $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$.

LEMMA 11. *If $\dot{\xi} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P} \cup \angle_{\mathbf{a}_{P'}}^{\mathbf{b}_{P'}}$, and $sig(\xi) = e_1e_2e_3$ and $\xi(0) = x_1, \xi(t_2) = x_2, \xi(t_2') = x_2', \xi(t_3) = x_3$, then e_2 is a reach-all edge.*

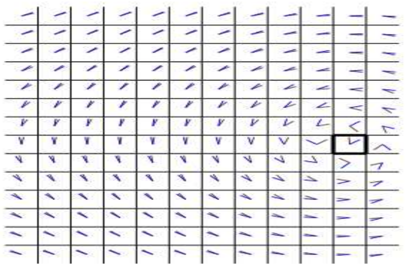

Figure 9: Example of a GSPDI. The picture is automatically generated by the GSPeeDI tool.

A bounce as described in the lemma is shown in Fig. 8-(b). From all the lemmas above we have the main result of this section:

THEOREM 2. *There is no need to iterate cycles containing reach-all edges (including bounces) more than once.*

As we will see in next section, this theorem has a practical impact on the performance of the reachability algorithm.

6. OPTIMIZATION AND IMPLEMENTATION

The theory in this paper was motivated by our development of the tool GSPeeDI [3], which, given a GSPDI and source and target edge-intervals, decides whether the target is reachable from the source.

The tool is written in Python, a general-purpose high-level programming language, which explicitly supports $\pm\infty$ as part of the *float* data type, enabling us to correctly represent the constant functions $f(x) = \pm\infty$.

The realization of affine function generation requires the theoretical result of section 4, as the procedure set out in the previous work [2] was inappropriate for certain special cases in the absence of the goodness assumption.

The results in section 5 were initially derived from the search for a practical way to deal with bounces and the accompanying *Flip* function, which is negative affine. The analysis of cycles requires the affine functions to be positive.

Computing all simple cycles may be infeasible for large graphs: The number of simple cycles in a complete, directed graph with n nodes is exactly $\sum_{i=1}^{n-1} \begin{pmatrix} n \\ n-i+1 \end{pmatrix} (n-i)!$.

For computing all simple cycles the tool uses an algorithm due to Tarjan [6], which has a time bound of $O((n+e)(c+1))$, where e is the number of edges and c the number of cycles in the graph. Clearly, the number of cycles is the factor determining the point at which the size of the problem becomes infeasible.

The tool has been run on examples with hundreds of nodes. Such examples become infeasible if optimizations are disabled, due to execution time and the memory requirements of the vast number of (unpermutated) cycles.

The properties of GPSDIs allow us to modify the algorithm in such a way as to stop exploring all paths (possible cycle prefixes) we such that: 1. $\mathsf{Succ}_{we}([0,1]) = \emptyset$: it is not possible to traverse this path; 2. e is reach-all; 3. The last edge of w is e^{-1}: a redundant bounce is found.

We will explain the optimizations' effectiveness on a large GSPDI (Fig. 9 show actual output of the GSPeeDI tool, which partially shows the GSPDI). In the figure there is just one reach-all region, and the surrounding reach-all edges can be seen drawn with thicker lines[2].

The resulting reachability graph contains 334 nodes. A run of the algorithm without optimizations finds that the total number of cycles (without permutations) is 181398. If we apply the first optimization, we reduce that number to 1041 cycles. Adding the second optimization reduces the number to 112 cycles, and applying all three leaves us with 85 cycles.

So, for this particular example, we find that more than 99% of the possible cycles are redundant. Computing the number of permutations gives us a total of 1100 cycles subsequently used as meta-transitions in the breadth-first search.

The execution time of the program which generates the cycles is less than a minute on a low-end, modern CPU. On the same system a reachability search returning *false* (thus having computed the entire reach-set for a particular start-interval) finishes execution in slightly over ten seconds.

7. CONCLUSION

In this paper we have presented further theoretical results concerning the reachability of GSPDIs, and we showed the implication of such results on the optimization of the algorithm, which has been implemented into the tool GSPeeDI.

The main application of GSPDIs is as approximation of planar (non-linear) differential equations, as discussed in [5]. We are currently working on the development of techniques to obtain such approximations.

8. ACKNOWLEDGMENTS

We thank Martin Steffen for useful discussion on early versions of the paper.

9. REFERENCES

[1] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In *HSCC'01*, volume 2034 of *LNCS*, pages 89–104, 2001.

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

[3] H. A. Hansen and G. Schneider. Gspeedi – a verification tool for generalized polygonal hybrid systems. In *ICTAC 2009*, volume 5684 of *LNCS*, pages 343–348, August 2009.

[4] G. Pace and G. Schneider. Model checking polygonal differential inclusions using invariance kernels. In *VMCAI'04*, volume 2937 of *LNCS*, pages 110–121, 2003.

[5] G. J. Pace and G. Schneider. Relaxing goodness is still good. In *ICTAC'08*, volume 5160 of *LNCS*, pages 274–289, 2008.

[6] R. E. Tarjan. Enumeration of the elementary circuits of a directed graph. Technical report, Ithaca, NY, USA, 1972.

[2]That the region is reach-all, due to $\angle_{\mathbf{a}}^{\mathbf{b}} > \pi$, is hard to see in black-and-white, as the tool uses different colors to distinguish $\mathbf{a}$ and $\mathbf{b}$.