Algorithmic Analysis of Polygonal Hybrid Systems, Part I: Reachability

Eugene Asarin [a], Gerardo Schneider [b,*], Sergio Yovine [c]

[a] *LIAFA, Case 7014, 2 pl. Jussieu, 75251 Paris Cedex 5, France*

[b] *Dept. of Informatics, University of Oslo, P.O. Box 1080 Blindern, NO-0316 Oslo, Norway*

[c] *CNRS-VERIMAG, Centre Equation, 2 Ave. Vignate, 38610 Gières, France*

Abstract

In this work we are concerned with the formal verification of two-dimensional non-deterministic hybrid systems, namely *polygonal differential inclusion systems* (SPDIs). SPDIs are a class of nondeterministic systems that correspond to piecewise constant differential inclusions on the plane, for which we study the reachability problem.

Our contribution is the development of an algorithm for solving exactly the reachability problem of SPDIs. We extend the geometric approach due to Maler and Pnueli [MP93] to non-deterministic systems, based on the combination of three techniques: the representation of the two-dimensional continuous-time dynamics as a one-dimensional discrete-time system (using Poincaré maps), the characterization of the set of qualitative behaviors of the latter as a finite set of *types of signatures*, and acceleration used to explore reachability according to each of these types.

Key words: Hybrid systems, differential inclusions, verification, decision algorithm, reachability analysis

---

* Corresponding author.

Email addresses: `Eugene.Asarin@liafa.jussieu.fr` (Eugene Asarin), `gerardo@ifi.uio.no` (Gerardo Schneider), `Sergio.Yovine@imag.fr` (Sergio Yovine).

Contents

1 Introduction 3

2 Polygonal Differential Inclusions 5

2.1 Preliminaries 5

2.2 Polygonal differential inclusions 7

2.3 SPDI and hybrid systems 9

3 Simplification of Trajectory Segments 10

3.1 Straightening trajectory segments 10

3.2 Removing self-crossings 12

4 Qualitative Analysis of Simplified Trajectory Segments 16

4.1 From Simplified Trajectory Segments to Factorized Signatures 16

4.2 From Factorized Signatures to Types of Signatures 19

4.3 Properties of Types of Feasible Signatures 20

5 Affine Multivalued Operators 22

6 Successor Function 26

7 Reachability Analysis 32

7.1 Main algorithm 32

7.2 Main result 38

7.3 Examples 39

8 Conclusion 41

References 43

A Affine Operators (properties) 47

B Soundness, termination and completeness of $Exit_*$ and $Test_*$ functions 54

1 Introduction

In the last decades daily life has been dominated by technological devices using computers or digital controllers. One source of complexity in such systems arises because these computers perform discrete operations while interacting with a physical environment which, in turn, has continuous dynamics. These systems are called *hybrid systems* because both continuous and discrete behaviors interact with each other. A typical example is given by a discrete program that interacts with (controls, monitors, supervises) a continuous physical environment. Most hybrid systems are *critical* systems in which errors can have serious consequences: air traffic management systems [TLS98], robotic systems [AGH+00], manufacturing plants [FvS99], automobiles [BBM+00], automated highway systems [PAT] and chemical plants [BKS00]. To ensure correctness, the behavior of hybrid systems must be formally modeled and verified.

Hybrid systems have been extensively studied in the last decade (for instance, [GNRR93,AKNS95,AHS96,AKNS97,AKL+98,VvS99,LK00,dBSV01,TG02]). One widely used formalization for hybrid systems are *hybrid automata* [ACH+95] which are finite-state machines enriched with differential equations or inclusions. Hybrid automata allow to model the discrete part of a hybrid system as transitions between the states of the machine and the continuous part with differential equations or inclusions.

Most decidability results on algorithmic verification of hybrid systems proved in the literature are based on the existence of a finite and computable partition of the state space into classes of states which are equivalent with respect to reachability. This is the case for timed automata [AD94], certain classes of rectangular automata [HKPV95] and hybrid automata with linear vector fields of a special form [LPY01]. Except for timed automata, these results rely on stringent hypothesis such as the resetting of variables along transitions.

Most implemented computational procedures resort to (forward or backward) propagation of constraints, typically (unions of convex) polyhedra or ellipsoids (e.g., [ACH+95,ABDM00,BT00,DM98,GM99,KV00,CK98,Dan00,HPHHt97]). In general, these techniques provide semi-decision procedures: if the given final set of states is reachable, they will eventually terminate, otherwise they may fail to do so. This is a property of the techniques, not of the problem. In other words, these algorithms may not terminate for certain systems for which the reachability problem is indeed decidable. Nevertheless, they provide tools for computing (approximations of) the reach-set for large classes of hybrid systems with linear and non-linear vector fields.

Maybe the major drawback of set-propagation, reach-set approximation procedures is that little attention is paid to the geometric properties of the specific system or the class of systems under analysis. To our knowledge, in the context of hybrid systems there are two lines of work in the direction of developing more "geometric" approaches. One is based on the existence of (enough) integrals and the ability to compute them all [Bro99,DY01]. These methods, however, do not necessarily result in decision procedures. The other, applicable to two-dimensional hybrid dynamical systems, relies on the topological properties of the plane, and explicitly focuses on decidability issues. This method, originally introduced in [MP93], is the one used in our paper.

In this work we are concerned with the formal verification of two-dimensional non-deterministic hybrid systems, namely *polygonal differential inclusion systems* (SPDIs). SPDIs are a class of nondeterministic systems that correspond to piecewise constant differential inclusions on the plane, for which we study the reachability problem.

Previous studies on planar hybrid systems are the following. [GJ94] presents many examples and a general theory for modeling hybrid systems but no decidability issues are discussed. The starting point for our research was [MP93] that shows that the reachability problem for two-dimensional piecewise constant systems (PCDs) is decidable. The approach there is based on several ideas. First, as suggested by Poincaré, the "essence" of the two-dimensional continuous-time dynamics can be represented as a one-dimensional discrete-time system (a collection of so-called Poincaré maps [HS74,NS60]). Next, in the case of PCDs, these maps are particularly simple, they are just scalar affine functions. Last, due to the topological properties of the plane, the global behavior of a planar trajectory is never chaotical and always belongs to a finite set of qualitative types, and these types can be distinguished and analyzed using the explicit formulas for Poincaré maps. This result has been extended in [uV96] for planar piecewise Hamiltonian systems.

Our contribution is the development of an algorithm for solving exactly the reachability problem of SPDIs. This required the introduction of multi-valued Poincaré maps, an algorithmics allowing to work with them, and specific topological considerations, since trajectories of a differential inclusion behave much "worse" than those of a differential equation. Our approach considers in fact a subset of "nice" trajectories which is sufficient to obtain the correct reachability relation. This work is an extended and revised version of [ASY01].

On the other hand, using a terminology from the verification community, both the algorithm of [MP93] for PCDs as well as ours for SPDIs, are based on *acceleration* of simple cycles. *Acceleration* is a well-known technique in verification that consists in computing, in one step, all the possible (maybe infinite) states that would be reachable in an unbounded number of steps, clearly saving computation time and space. This technique was applied in many contexts, e.g. for automata with counters [BW94] and for automata with queues [AAB99,BGWW97,BH97]. Acceleration for hybrid systems was considered in [BHJ03], but without applications to decidability.

Outline

In Section 2 we describe the class of two dimensional non-deterministic *hybrid systems* studied in this article, namely *polygonal differential inclusion systems* (SPDIs). We also give some motivation for studying this model, and compare it to other classes of hybrid systems

In Sections 3 and 4 we present the difficulties that arise when trying to solve the reachability problem for SPDIs and we show how to overcome these difficulties first by simplifying trajectories, and the performing their qualitative analysis. We abstract trajectories to *types of signatures* and we show how this abstraction allows to split the reachability problem into finitely many simpler subproblems.

In Section 5 we present a useful class of functions called *truncated affine multivalued operators* (TAMF) that serves as a technical basis for characterizing *successor* and *predecessor* operators in Section 6.

In Section 7 we present the main contribution of this article, namely the decision procedure for the reachability problem of SPDIs. Given, for instance, two points in an SPDI, the reachability question is: Is one point reachable from the other? We show how a case analysis simplifies the treatment of cycles and how to take advantage of the fact that *successors* are TAMF in order to *accelerate* cycles. We finally present our reachability algorithm, we prove its soundness and completeness, and illustrate it with several examples.

Finally, in Section 8 we present the conclusions.

2 Polygonal Differential Inclusions

2.1 Preliminaries

We first introduce several notations:

- We denote the inner (scalar) product of two vectors $\mathbf{x}$, $\mathbf{y}$ by $\mathbf{x}\,\mathbf{y}$;
- Given $\mathbf{x} = (x_1, x_2)$ we denote by $\hat{\mathbf{x}}$ the vector $(x_2, -x_1)$ obtained from $\mathbf{x}$ by rotating clockwise by the angle $\pi/2$;
- We denote the Euclidean norm of $\mathbf{x}$ by $|\mathbf{x}|$;
- The ϵ-*neighborhood* of $\mathbf{x}$ is $B_\epsilon(\mathbf{x}) = \{\mathbf{y} \mid |\mathbf{x} - \mathbf{y}| < \epsilon\}$.

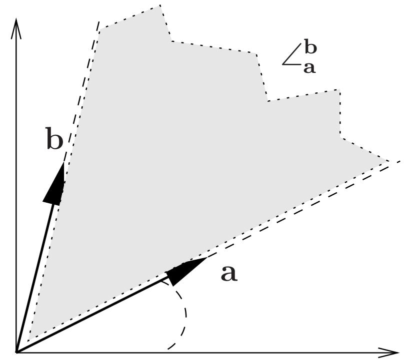


Fig. 1. Positive hull of $\{\mathbf{a}, \mathbf{b}\}$ with $\hat{\mathbf{a}}\,\mathbf{b} < 0$.

- The *interior* of $X \subseteq \mathbb{R}^2$ is the set of $\mathbf{x} \in X$ for which there exists $\epsilon > 0$ such that $B_\epsilon(\mathbf{x}) \subseteq X$. It is denoted by $int(X)$.
- For a line segment e on the plane we denote by $int_1(e)$ its relative interior, that is e without its endpoints.

For $\mathbf{x}_1, \ldots \mathbf{x}_n \in \mathbb{R}^2$ a *linear* combination is a vector $\mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{x}_i$ for some $\lambda_i \in \mathbb{R}$. A *positive combination* is a linear combination with $\lambda_i \geq 0$ for every i. The positive *hull* of a set $X \subseteq \mathbb{R}^2$ is the set of all positive combinations of points in X. A (closed) *half-plane* is the set of all points $\mathbf{x}$ satisfying $\mathbf{a}\,\mathbf{x} \leq b$. A *convex closed polygonal set* P is the intersection of finitely many half-planes. An *edge* e is a line segment in $\mathbb{R}^2$.

Let S be a finite index set and $\mathbb{P} = \{P_s\}_{s \in S}$ be a finite set of convex closed polygonal sets, called *regions*, such that:

(1) For all $s \in S$, $int(P_s) \neq \emptyset$;
(2) For all $s \neq r \in S$, $int(P_s \cap P_r) = \emptyset$.

Condition 1 states that regions are full dimensional. Condition 2 says that the intersection between two regions is empty, an edge, or a point. Thus, $\mathbb{P}$ is a polygonal *partition* of a subset of the plane.

We denote by $E(P)$ the set of edges of the form $e = P \cap P'$ with $P \neq P'$ and by $V(P)$ the set of vertices of the form $v = e \cap e'$ with $e, e' \in E(P)$. Let $int_1(E(P)) = \{int_1(e) \mid e \in P\}$ be the set of all the *open* edges of P, then let $EV(P) = int_1(E(P)) \cup V(P)$ be the set of all the vertices and open edges of P.

Angles on the plane play a special role in this article. An angle $\angle_{\mathbf{a}}^{\mathbf{b}}$ (Fig. 1), defined by two non-zero vectors $\mathbf{a}, \mathbf{b}$ is the set of all positive linear combinations $\mathbf{x} = \alpha\ \mathbf{a} + \beta\ \mathbf{b}$, with $\alpha, \beta \geq 0$. We can always assume that $\mathbf{b}$ is situated in the counter-clockwise direction from $\mathbf{a}$ (that is $\hat{\mathbf{a}}\,\mathbf{b} < 0$).

Informally, a *polygonal differential inclusion system* (SPDI) consists of a partition of a plane subset into convex polygonal regions, together with a constant differential inclusion associated with each region.

Let $\mathbb{P} = \{P_s\}_{s\in S}$ be a partition, and $\mathbb{F} = \{\phi_s\}_{s\in S}$ be such that each ϕ_s is an angle between two vectors $\mathbf{a}_s$ and $\mathbf{b}_s$ with $\hat{\mathbf{a}}_s\ \mathbf{b}_s < 0$ and $\mathbb{P}$ be a partition of the plane.

A *polygonal differential inclusion system* (SPDI) consists of a partition of the plane into convex polygonal regions, together with a differential inclusion associated with each region. More formally,

Definition 2.1 *A* polygonal differential inclusion system *(SPDI) is a pair $\mathcal{H} = (\mathbb{P}, \mathbb{F})$. Each region P_s has dynamics $\dot{\mathbf{x}} \in \phi_s$ for $\mathbf{x} \in P_s$ (given a generic region P we also use the notation $\phi(P)$).*

As an example consider the problem of a swimmer trying to escape from a whirlpool in a river.

Example 2.2 The dynamics $\dot{\mathbf{x}}$ of the swimmer around the whirlpool is approximated by the piecewise differential inclusion defined as follows. The zone of the river nearby the whirlpool is divided into 8 regions $R_1, \ldots, R_8$. To each region R_i we associate a pair of vectors $(\mathbf{a}_i, \mathbf{b}_i)$ meaning that $\dot{\mathbf{x}}$ belongs to their positive hull:

- $\mathbf{a}_1 = \mathbf{b}_1 = (1, 5)$,
- $\mathbf{a}_2 = \mathbf{b}_2 = (-1, \frac{1}{2})$,
- $\mathbf{a}_3 = (-1, \frac{11}{60})$ and $\mathbf{b}_3 = (-1, -\frac{1}{4})$,
- $\mathbf{a}_4 = \mathbf{b}_4 = (-1, -1)$,
- $\mathbf{a}_5 = \mathbf{b}_5 = (0, -1)$,
- $\mathbf{a}_6 = \mathbf{b}_6 = (1, -1)$,
- $\mathbf{a}_7 = \mathbf{b}_7 = (1, 0)$,
- $\mathbf{a}_8 = \mathbf{b}_8 = (1, 1)$.

The corresponding SPDI is illustrated in Fig. 2-(a). ■

Let P be a region and $e \in E(P)$ an edge. We say that e is an *entry* of P if for all $\mathbf{x} \in int_1(e)$ and for all $\mathbf{c} \in \phi(P)$, $\mathbf{x} + \mathbf{c}\epsilon \in P$ for some $\epsilon > 0$. We say that e is an *exit* of P if the same condition holds for some $\epsilon < 0$. We denote by $In(P) \subseteq E(P)$ the set of all entries of P and by $Out(P) \subseteq E(P)$ the set of all exits of P.

Definition 2.3 *A* trajectory segment *on some interval $[0, T] \subseteq \mathbb{R}$, with initial condition $\mathbf{x} = \mathbf{x}_0$, is a continuous and almost-everywhere (everywhere except on finitely many points) differentiable function $\xi(\cdot)$ such that $\xi(0) = \mathbf{x}_0$ and for all $t \in (0, T)$:*

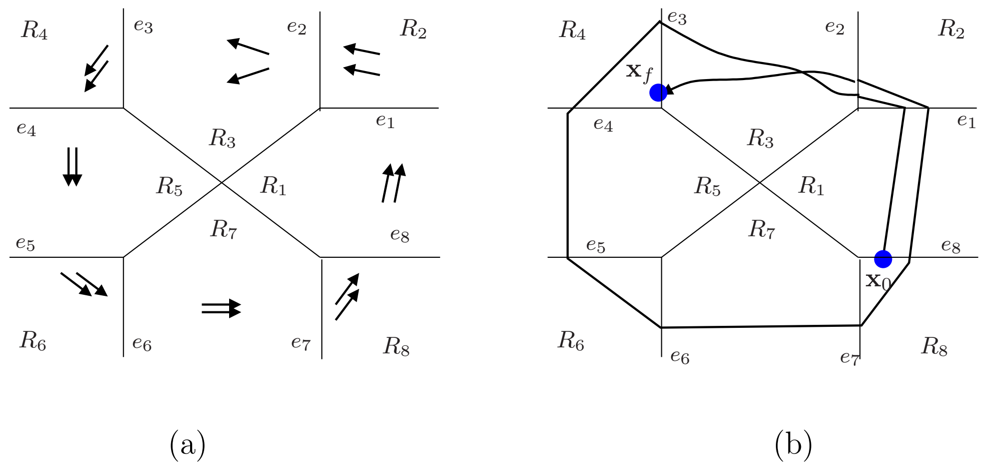


Fig. 2. (a) The SPDI of the swimmer; (b) A typical trajectory segment.

(1) if $\xi(t) \in int(P)$ then $\dot{\xi}(t)$ is defined and $\dot{\xi}(t) \in \phi(P)$;
(2) if $\xi(t) \in e$ and $e \in \mathit{In}(P)$ then $\dot{\xi}^+(t)$ is defined and $\dot{\xi}^+(t) = \phi(P)$, where $\dot{\xi}^+(t) = \frac{d^+\xi}{dt}$ is the right derivative of ξ.

If $T = \infty$, a trajectory segment is called a trajectory.

Example 2.4 Figure 2-(b) shows a typical trajectory of the SPDI presented in Example 2.2 from point $\mathbf{x}_0$ to $\mathbf{x}_f$.

Edges, vertices, entry edges, exit edges and the corresponding sets are defined as for PCDs. The set of all edges of an SPDI will be denoted by $\mathcal{E}$, i.e., $\mathcal{E} = \bigcup_{s \in S} EV(P_s)$.

In general, $E(P) \neq \mathit{In}(P) \cup \mathit{Out}(P)$. We say that P is a *good* region iff all the edges in $E(P)$ are entries or exits, that is,

Definition 2.5 *A region P of an SPDI is* good *if and only if $E(P) = \mathit{In}(P) \cup \mathit{Out}(P)$.*

Notice that, if P is a good region, then for all $e \in E(P)$, $e \notin \phi(P)$.

Assumption 2.6 (Goodness) *In the following we assume that all the regions of the SPDI considered are good.*

Example 2.7 In Figure 3-(a), region P (with $\phi(P) = \angle_{\mathbf{a}}^{\mathbf{b}}$) is good, since all are entry or exit edges. Figure 3-(b) shows a region that is not good: edges e_2 and e_5 are not in $\mathit{In}(P) \cup \mathit{Out}(P)$. ■

The reachability problem for an SPDI $\mathcal{H}$ can be defined as a predicate

$$\mathit{Reach}(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \equiv \exists \xi \, \exists t \geq 0 \, . \, (\xi(0) = \mathbf{x}_0 \wedge \xi(t) = \mathbf{x}_f).$$

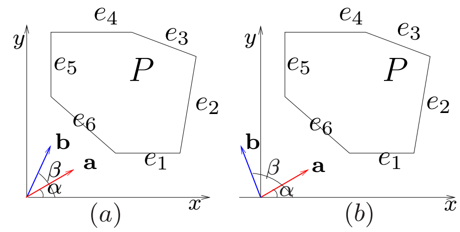


Fig. 3. a) A good region; b) A bad region.

The edge-to-edge reachability problem is the following: Given two edges e and e' of $\mathcal{H}$, is there $\mathbf{x}_0 \in e$ and $\mathbf{x}_f \in e'$ such that $\mathbf{x}_f$ is reachable from $\mathbf{x}_0$? The region-to-region reachability problem is defined similarly.

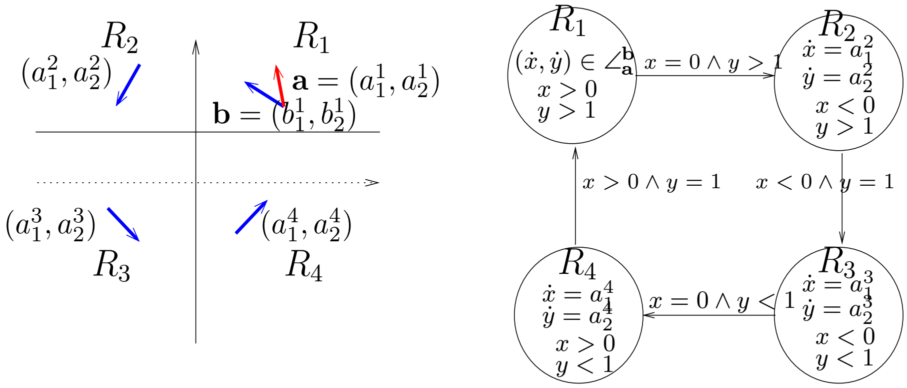
2.3 SPDI and hybrid systems

The notion of SPDI is a straightforward generalization of PCD (piecewise-constant derivatives) systems introduced and studied in [AMP95,AM94,MP93]. PCDs can be seen as deterministic linear hybrid automata (see [ACH+95]) with an additional constraint of having continuous trajectories. Mathematically, PCDs are differential equations with piecewise-constant right-hand side. As established in the references above, reachability is decidable for planar PCDs, and undecidable in dimensions 3 and more.

Our aim was to find a class of systems richer than planar PCD, but still with decidable reachability problem. The novel feature of SPDIs with respect to PCDs is the non-determinism. Technically, differential equations are replaced by differential inclusions.

In control and applied mathematics, inclusions are used to model systems with uncertainties and disturbances. One can model such systems using differential equations of the form $\dot{x} = f(x, u)$ where $u \in U$ is a control or a disturbance. An alternative representation is a differential inclusion $\dot{x} \in g(x)$ where $g(x) = \{f(x, u) \mid u \in U\}$ [PVB96]. The differential inclusion $\dot{x} \in g(x)$ captures every possible behavior of f. Moreover, polygonal differential inclusions allow to obtain conservative approximations of complicated nonlinear dynamics.

The class SPDI is also related to hybrid automata. In fact it is not difficult to show that any SPDI can be represented as a non-deterministic linear hybrid automaton with continuous trajectories[ACH+95], as illustrated on Fig 4.



(a) (b)

Fig. 4. From an SPDI (a) to a linear hybrid automaton (b).

3 Simplification of Trajectory Segments

In this section we prove that when solving the reachability question we can restrict the analysis to rectilinear trajectories without self-crossings.

3.1 Straightening trajectory segments

We show here how to transform trajectory segments into rectilinear ones by *straightening* them. W.l.o.g. we consider in what follows that $\xi(0) \in e$ for some edge $e \in \mathcal{E}$. We have the following objects associated to a trajectory (or a trajectory segment):

Definition 3.1 *An* edge signature *of an SPDI is a sequence of edges. The* edge signature of a trajectory ξ *is the ordered sequence of edges traversed by this trajectory:* $\mathsf{Sig}(\xi) = e_0 e_1 \ldots$. *The* trace *of* ξ *is the sequence* $\mathtt{trace}(\xi) = \mathbf{x}_0 \mathbf{x}_1 \ldots$ *of the intersection points of* ξ *with the set of edges* $\mathcal{E}$ *(notice that* $\mathbf{x}_i \in e_i$*). The* region signature *of* ξ *is the sequence* $\mathsf{RSig}(\xi) = P_0 P_1 \ldots$ *of traversed regions, that is,* $e_i \in \mathit{In}(P_i)$.

Definition 3.2 *Given a signature* $\mathsf{Sig}(\xi) = e_0 e_1 \ldots e_h \ldots e_n \ldots$, *the sequence of edges* $\sigma = e_h \ldots e_n$ *is a* cycle *iff* $e_h = e_n$*, and* σ *is a* simple edge-cycle *if additionally for all* $h < i \neq j < n$, $e_i \neq e_j$. *A region signature* $\mathsf{RSig}(\xi) = P_0 P_1 \ldots P_n$ *is a* region cycle *iff* $P_0 = P_n$ *and it is a* simple region cycle *if in addition for all* $0 < i \neq j < n$, $P_i \neq P_j$.

Example 3.3 Let us consider the trajectory segment ξ from point $\mathbf{x}_0$ to point $\mathbf{x}_7$ shown in Figure 5-(a). Its edge signature is the sequence $\mathsf{Sig}(\xi) = e_1 e_2 e_9 e_{10} e_{11} e_1 e_2 e_3$, its trace is $\mathtt{trace}(\xi) = \mathbf{x}_0 \mathbf{x}_1 \ldots \mathbf{x}_6 \mathbf{x}_7$, and its region signature is $\mathsf{RSig}(\xi) = R_1 R_2 R_4 R_6 R_8 R_1 R_2$. ∎

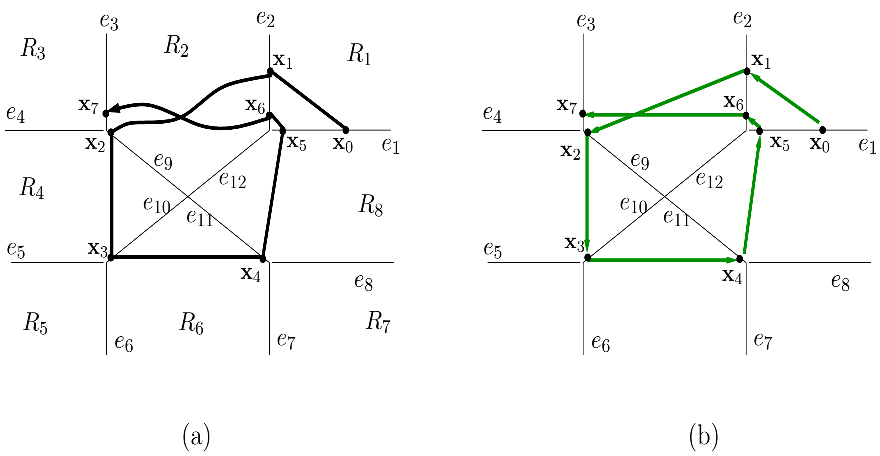


Fig. 5. (a) A trajectory segment with its trace; (b) The straightened trajectory segment.

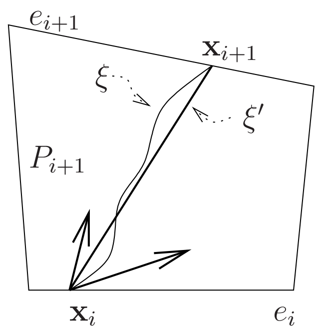


Fig. 6. Piecewise constant trajectory.

The following result expresses that any segment of trajectory in a given region can be straightened, preserving its initial and final points (see Fig. 6).

Proposition 3.4 *For every trajectory segment ξ there exists a trajectory segment ξ' with the same initial and final points, and edge and region signatures, such that for each P_i in the region signature, there exists $\mathbf{c}_i \in \phi(P_i)$, such that $\dot{\xi}'(t) = \mathbf{c}_i$ for all $t \in (t_i, t_{i+1})$. Moreover, $\mathtt{trace}(\xi) = \mathtt{trace}(\xi')$.*

PROOF. Let ξ be a trajectory segment whose trace is $\mathtt{trace}(\xi) = \mathbf{x}_0 \ldots \mathbf{x}_k$. Let $0 = t_0 < t_1 < \ldots < t_k$ be such that $\xi(t_i) = \mathbf{x}_i$. Consider an interval (t_i, t_{i+1}), on this interval $\xi(t)$ stays in some region P_i, hence it satisfies the inclusion $\dot{\xi} \in \angle_{\mathbf{a}_i}^{\mathbf{b}_i}$, where $\angle_{\mathbf{a}_i}^{\mathbf{b}_i} = \phi(P_i)$. This means that for some non-negative functions α, β the following equality holds:

$$\dot{\xi}(t) = \alpha(t)\mathbf{a}_i + \beta(t)\mathbf{b}_i, \quad \forall t \in (t_i, t_{i+1}). \tag{1}$$

Consider now the mean value of the right-hand side:

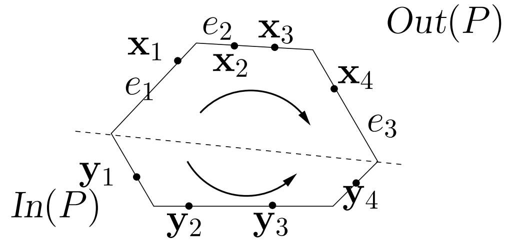


Fig. 7. Ordering: $\mathbf{x}_1 \preceq \mathbf{x}_2 \preceq \mathbf{x}_3 \preceq \mathbf{x}_4$; $\mathbf{y}_1 \preceq \mathbf{y}_2 \preceq \mathbf{y}_3 \preceq \mathbf{y}_4$.

$$\begin{aligned}\mathbf{c}_i &= \frac{1}{t_{i+1}-t_i}\int_{t_i}^{t_{i+1}} (\alpha(t)\mathbf{a}_i + \beta(t)\mathbf{b}_i)\, dt = \\ &= \mathbf{a}\frac{1}{t_{i+1}-t_i}\int_{t_i}^{t_{i+1}} \alpha(t)\, dt + \mathbf{b}\frac{1}{t_{i+1}-t_i}\int_{t_i}^{t_{i+1}} \beta(t)\, dt. \qquad (2)\end{aligned}$$

We have just shown that $\mathbf{c}_i$ is a positive linear combination of $\mathbf{a}_i$ and $\mathbf{b}_i$, and hence $\mathbf{c}_i \in \angle_{\mathbf{a}_i}^{\mathbf{b}_i}$.

Consider now a "piecewise straight" continuous line $\zeta(t)$ such that $\zeta(t_0) = x_0$ and

$$\dot{\zeta}(t) = \mathbf{c}_i, \quad \forall t \in (t_i, t_{i+1}).$$

It is easy to see now, that

- $\forall i.\ \zeta(t_i) = \xi(t_i) = x_i$, indeed this holds for t_0 and, in virtue of (1) and (2)

$$\xi(t_{i+1}) - \xi(t_i) = \zeta(t_{i+1}) - \zeta(t_i) = \int_{t_i}^{t_{i+1}} (\alpha(t)\mathbf{a}_i + \beta(t)\mathbf{b}_i)\, dt,$$

  which insures the inductive step;
- $\forall t \in (t_i, t_{i+1})$. $\zeta(t) \in P_i$ since P_i is convex;
- hence ζ satisfies the differential inclusion;
- in conclusion ζ is a trajectory segment with the same trace as ξ. □

Example 3.5 In Figure 5-(b) it is shown the straightened trajectory segment of the one given in Figure 5-(a). ■

Hence, in order to solve the reachability problem it is enough to consider trajectory segments having piecewise constant slopes. Notice that, however, such slopes need not be the same for each occurrence of the same region in the region signature. Hereinafter, we only consider trajectory segments whose derivatives are piecewise constant.

3.2 Removing self-crossings

Before proceeding to the removing of self-crossing trajectory segments we need to introduce an order relation which will be intensively used in the sequel.

Given a region P we define a dense linear order on $Out(P)$ as follows: let $\mathbf{x}_1, \mathbf{x}_2 \in P$, we say that $\mathbf{x}_1 \prec \mathbf{x}_2$ if $\mathbf{x}_2$ lies in the clockwise direction from $\mathbf{x}_1$ w.r.t P. Similarly, on $In(P)$ we say that $\mathbf{y}_1 \prec \mathbf{y}_2$ if $\mathbf{y}_2$ lies in the counter-clockwise direction from $\mathbf{y}_1$ w.r.t P (see Fig. 7). Notice, that these orders are compatible, in the sense that if $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to both $In(P)$ and $Out(Q)$, then the ordering between them with respect to the two regions will be the same.

We say that a trajectory ξ crosses itself if there exist $t \neq t'$ such that $\xi(t) = \xi(t')$. If a trajectory does not cross itself, the sequence of consecutive intersection points with $In(P)$ or $Out(P)$ is monotone with respect to $\preceq$. That is, for every three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ (visited in this order), if $\mathbf{x}_1 \prec \mathbf{x}_2 \prec \mathbf{x}_3$ the trajectory is a "counterclockwise expanding spiral"(Fig. 8(a)) or a "clockwise contracting spiral" (Fig. 8(b)) and if $\mathbf{x}_3 \prec \mathbf{x}_2 \prec \mathbf{x}_1$, the trajectory is a "counterclockwise contracting spiral" (Fig. 8(c)) or a "clockwise expanding spiral" (Fig. 8(d)).

Lemma 3.6 ([AMP95]) *For every trajectory ξ, if ξ does not cross itself, then for every edge e, the sequence* $\texttt{trace}(\xi) \cap e$ *is monotone (with respect to $\preceq$).*

We prove now that self-crossings can be removed from trajectory segments, preserving the reachability problem, by showing first that we can always diminish the number of self-crossings.

Lemma 3.7 *For every trajectory segment ξ that crosses itself at least once, there exists a trajectory segment ξ' with the same initial and final points as ξ having a number of self-crossings strictly smaller.*

PROOF. Suppose that the trajectory segment ξ with $\texttt{trace}(\xi) = \mathbf{x}_0 \ldots \mathbf{x}_f$ crosses itself *once* inside the region P. Let $e_1, e_2 \in In(P)$ be the input edges and $e'_1, e'_2 \in Out(P)$ be the output ones. Let $\mathbf{x} = \mathbf{x}_i \in e_1$ and $\mathbf{y} = \mathbf{x}_j \in e_2$, with $i < j$, be the points in $\texttt{trace}(\xi)$ where ξ enters P for the first and the second times, and let $\mathbf{x}' = \mathbf{x}_{i+1} \in e'_2$ and $\mathbf{y}' = \mathbf{x}_{j+1} \in e'_1$ be the corresponding output points. Let $\mathbf{c}_x, \mathbf{c}_y \in \phi(P) = \angle_{\mathbf{a}}^{\mathbf{b}}$ be the derivatives of ξ in the time intervals (t_i, t_{i+1}) and (t_j, t_{j+1}), respectively. Indeed, $\mathbf{c}_x$ and $\mathbf{c}_y$ are the vectors of the segments $\overline{\mathbf{x}\mathbf{x}'}$ and $\overline{\mathbf{y}\mathbf{y}'}$, respectively (Fig. 9(a)). Consider now the segment $\overline{\mathbf{x}\mathbf{y}'}$. Notice that the vector $\mathbf{c}'_x$ of this segment can be obtained as a positive combination of the vectors $\mathbf{c}_x$ and $\mathbf{c}_y$. That is, there exist $\alpha_1, \alpha_2 > 0$ such that $\mathbf{c}'_x = \alpha_1 \mathbf{c}_x + \alpha_2 \mathbf{c}_y$ (see Fig. 9(b)). Since $\phi(P) = \angle_{\mathbf{a}}^{\mathbf{b}}$ is closed under positive combinations, $\mathbf{c}'_x \in \phi(P)$. Similarly we can prove that $\mathbf{c}'_y$ is a positive combination of $\mathbf{a}$ and $\mathbf{b}$. Hence, there exists a trajectory ξ' that does not cross itself in P having $\texttt{trace}(\xi') = \mathbf{x}_1 \ldots \mathbf{x}\mathbf{y}' \ldots \mathbf{x}_f$ (Fig. 10). Notice that the result also works for the *degenerate* case when the trajectory segment crosses itself

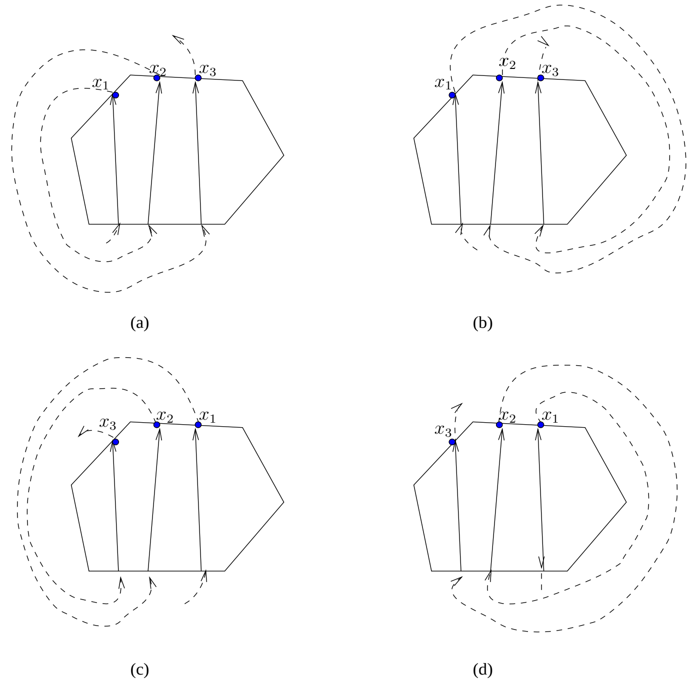


Fig. 8. (a) $\mathbf{x}_1 \prec \mathbf{x}_2 \prec \mathbf{x}_3$: counterclockwise expanding spiral; (b) $\mathbf{x}_1 \prec \mathbf{x}_2 \prec \mathbf{x}_3$: clockwise contracting spiral; (c) $\mathbf{x}_3 \prec \mathbf{x}_2 \prec \mathbf{x}_1$: counterclockwise contracting spiral; (d) $\mathbf{x}_3 \prec \mathbf{x}_2 \prec \mathbf{x}_1$: clockwise expanding spiral.

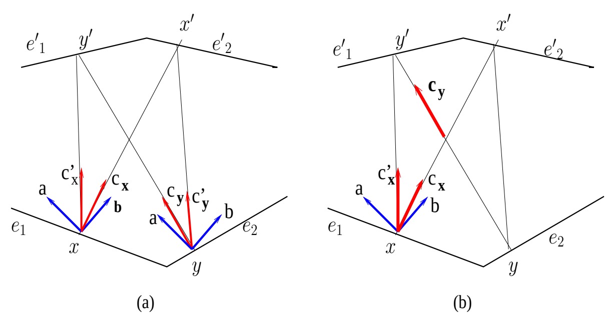


Fig. 9. A trajectory that crosses itself.

at an edge (or vertex) (see Fig. 11-(a)). If the trajectory segment ξ crosses itself more than once in region P, then the number of times the trajectory segment ξ', obtained by cutting away the loop (Fig. 10(c)), crosses itself in P is strictly smaller than the number of times ξ does it (see Fig. 12). After replacing $\overline{\mathbf{xx'}}$ and $\overline{\mathbf{yy'}}$ by $\overline{\mathbf{xy'}}$, the intersection q of $\overline{\mathbf{xx'}}$ and $\overline{\mathbf{yy'}}$ disappears. If the new segment of line $\overline{\mathbf{xy'}}$ crosses another segment $\overline{\mathbf{zz'}}$ (say at a point t), then $\overline{\mathbf{zz'}}$ necessarily crosses either $\overline{\mathbf{xx'}}$ (at r) or $\overline{\mathbf{yy'}}$ (at s) -or both-, before the transformation. The above is due to the fact that if $\overline{\mathbf{zz'}}$ crosses one side of the triangle $\mathbf{xy'}q$ then it must also cross one of the other sides of the triangle, say at r. Thus, no new crossing can appear and the number of crossings in the new configuration is always less than in the old one.

Notice that in the degenerate case shown in Figure 11-(b) there can be in-

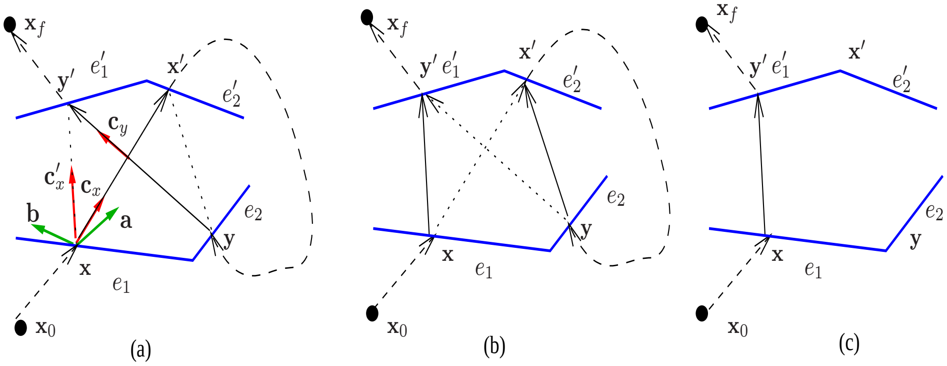
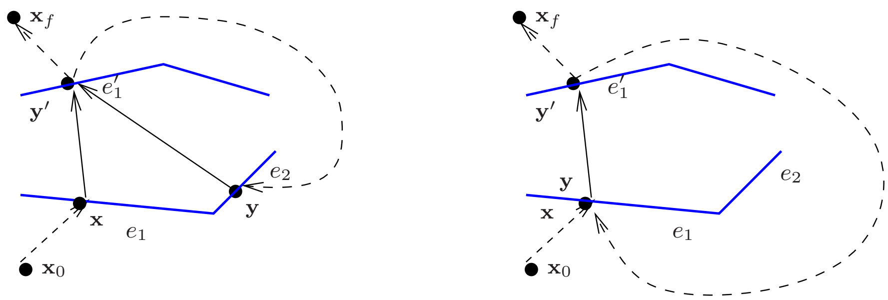


Fig. 10. Obtaining a non-crossing trajectory.



(a) (b)

Fig. 11. “Degenerate” self crossings.

finitely many crossing points. In such a case the construction above is still valid, but the induction proceeds over the number of crossing *points and intervals*. □

We have then the following proposition.

Proposition 3.8 (Existence of a non-crossing trajectory) *If there exists an arbitrary trajectory segment from point* $\mathbf{x}_0 \in e_0$ *to* $\mathbf{x}_f \in e_f$ *then there always exists a non-crossing trajectory segment between them.*

PROOF: By induction on the number n of times the trajectory segment crosses itself using Lemma 3.7 in the induction step. □

Example 3.9 Given the trajectory segment of Figure 5-(b), after eliminating the self-crossing we obtain the trajectory segment of Figure 13. ■

Hence, in order to solve the reachability problem we only need to consider non-crossing trajectory segments with piecewise constant derivatives. In what follows, we only deal with trajectory segments of this kind.

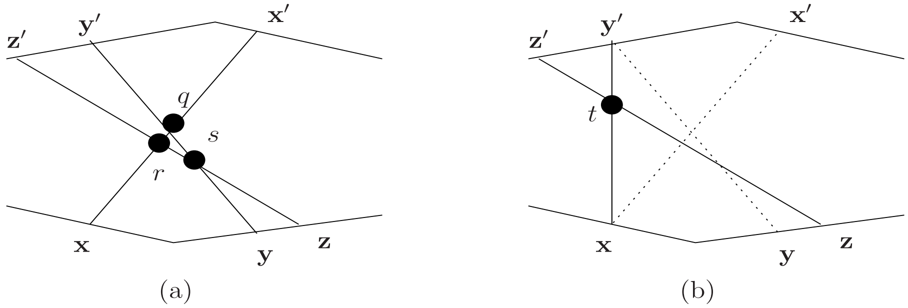


Fig. 12. The number of self-crossings decreases after eliminating a loop. (a) Before (3 crossings); (b) After (1 crossing).

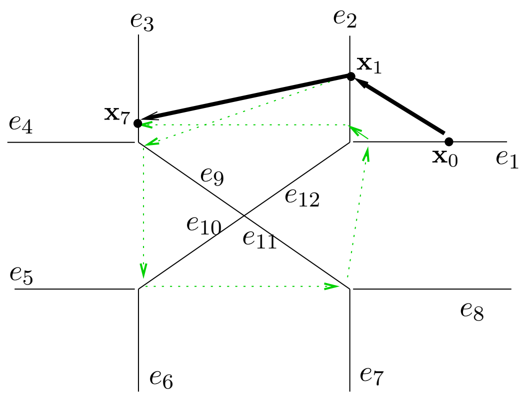


Fig. 13. A trajectory segment without self-crossing.

4 Qualitative Analysis of Simplified Trajectory Segments

Even considering simplified trajectory segments, there are infinitely many of them, and of a very different qualitative behavior. We show in this section that *signatures* provide a good "symbolic" abstraction of such trajectory segments. We also prove that there exist finitely many "types" of signatures, laying down the basis for a reachability algorithm.

4.1 From Simplified Trajectory Segments to Factorized Signatures

Given a trajectory segment ξ of an SPDI considering its edges signature $\mathsf{Sig}(\xi) = e_0, \ldots, e_i, \ldots, e_f$ provides information on its qualitative behavior.

In what follows we present a *representation theorem* that allows to express signatures in a *factorized* way.

Given a sequence w, ε denotes the empty sequence whereas $\mathsf{first}(w)$ and $\mathsf{last}(w)$ are the first and last elements of the sequence respectively. An edge signature σ can be expressed as a sequence of edges and cycles of the form

$r_1 s_1^{k_1} r_2 s_2^{k_2} \dots r_n s_n^{k_n} r_{n+1}$, where

(1) For all $1 \leq i \leq n+1$, r_i is a sequence of pairwise different edges;
(2) For all $1 \leq i \leq n$, s_i is a simple cycle (i.e., without repetition of edges) repeated k_i times;

This representation can be obtained by the following procedure of *greedy cycle decomposition.*

Algorithm $\mathcal{A}$. Let $\sigma = e_1 \dots e_{p-1} e_p$ be an edge signature. Starting from e_{p-1} and traversing backwards, take the first edge that occurs the second time. If there is no such edge, then trivially the signature can be expressed as a sequence of different edges. Otherwise, suppose that the edge e_j occurs again at position i (i.e. $e_i = e_j$ with $i < j$), thus $\sigma_{\mathcal{A}} = wsr$, where w, s and r are obtained as follows, depending on the repeated edge:

$$w = e_0 \dots e_i, \qquad s = e_{i+1} \dots e_j, \qquad r = e_{j+1} \dots e_{p-1}.$$

Clearly r is not a cycle and s is a simple cycle with no repeated edges. Let $k_m = \max\{l \mid s^l \text{ is a suffix of } w\}$. Thus, $\sigma_{\mathcal{A}} = w' s^k r$ with $w' = e_0 \dots e_h$ (a prefix of w) and $k = k_m + 1$. We repeat recursively the procedure above with w'. Adding the edge e_p to the last r (at the end) we obtain $\sigma_{\mathcal{A}} = r_1 s_1^{k_1} \dots r_n s_n^{k_n} r_{n+1}$ that is a representation of signature σ. $\square$

Notice that the "preprocessing" (taking away the last edge e_p) is done in order to differentiate edge signatures that end with a cycle from those that do not. There exists many other (maybe easier) ways of decomposing a signature σ (in particular, traversing forwards instead of backwards), but the one chosen here permits a clearer and simpler presentation of the reachability algorithm. In fact, using the above representation, the last visited edge in a cycle $e_1 \dots e_k$ is always the last one (e_k). The representation obtained by the above algorithm gives rise to the following theorem.

Theorem 4.1 (Representation Theorem) *Let $\sigma = e_1 \dots e_p$ be an edge signature, then it can always be written as $\sigma_{\mathcal{A}} = r_1 s_1^{k_1} \dots r_n s_n^{k_n} r_{n+1}$, where for any $1 \leq i \leq n+1$, r_i is a sequence of pairwise different edges and for all $1 \leq i \leq n$, s_i is a simple cycle (i.e., without repetition of edges).* $\square$

Each edge signature can then be represented as a sequence of edges and simple cycles.

Example 4.2 Let us consider the following examples. Suppose that

$$\sigma = abcdbcefgefgefgefhi.$$

Then, after applying once the above procedure of the algorithm we obtain

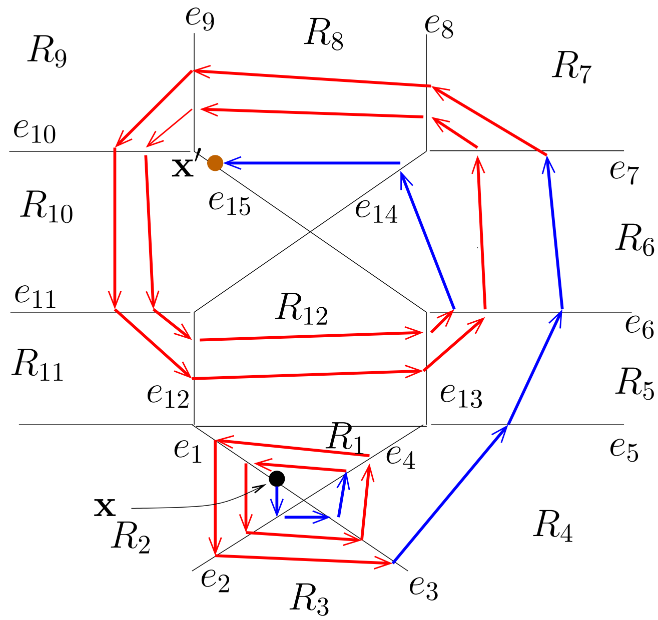


Fig. 14. A trajectory segment from $\mathbf{x}$ to $\mathbf{x}'$.

that

$$\sigma_{\mathcal{A}} = w(s_2)^3 r_1,$$

with $w = abcdbcef$; $s_2 = gef$; $r_1 = h$. Applying the procedure once more to w we obtain

$$w = w'(s_3)^1 r_2$$

with $w' = r_3 = abc$; $s_3 = dbc$; $r_2 = ef$. Putting all together and adding the last edge (i) gives

$$\sigma_{\mathcal{A}} = abc(dbc)^1 ef(gef)^3 hi.$$

Suppose now, that the signature ends with a cycle:

$$\sigma = abcdbcefgefgefgefgef.$$

In this case we apply the preprocessing obtaining

$$\sigma_{\mathcal{A}} = w(s_2)^4 r_1$$

with $w = abcdbce$; $s_2 = fge$; $r_1 = \varepsilon$. Applying the procedure to w we finally obtain

$$w = w'(s_3)^1 r_2$$

with $w' = r_3 = abc$; $s_3 = dbc$; $r_2 = e$ and that gives (adding f to the end)

$$\sigma_{\mathcal{A}} = abc(dbc)^1 e(fge)^4 f. \qquad \blacksquare$$

Example 4.3 Let us consider an SPDI and its trajectory segment from a point $\mathbf{x} \in e_1$ to a point $\mathbf{x}' \in e_{15}$ shown in Figure 14. The edge signature of the trajectory segment is $\sigma = e_1 e_2 e_3 \ldots e_6 e_7 \ldots e_{13} e_6 e_{14} e_{15}$. Applying Algorithm $\mathcal{A}$ above we obtain the following representation:

$$\sigma_{\mathcal{A}} = e_1 e_2 e_3 (e_4 e_1 e_2 e_3)^2 e_5 e_6 (e_7 \cdots e_{13} e_6)^2 e_{14} e_{15}. \qquad \blacksquare$$

Even when considering signatures, their number is still infinite. Our representation theorem simplifies the analysis but does not decrease the number of signatures to be considered. The problem is that in principle all the simple cycles can be iterated an unbounded number of times. Hence, the following natural step is to abstract away the number of times each simple cycle is iterated.

4.2 From Factorized Signatures to Types of Signatures

In this section we show how to abstract the signatures obtained in the previous section via the representation theorem to *types of signatures*. Given a representation of a signature, obtained as before, we have the following definition.

Definition 4.4 *Let $\sigma = e_1 \ldots e_p$ be an edge signature and $\sigma_{\mathcal{A}} = r_1 s_1^{k_1} \ldots r_n s_n^{k_n} r_{n+1}$ be its representation (obtained by Algorithm $\mathcal{A}$). Then we define the* type of a signature σ *as* $\texttt{type}(\sigma) = r_1, s_1, \ldots, r_n, s_n, r_{n+1}$. $\blacksquare$

When referring to the type of a signature, we will always mean the type being generated as in Theorem 4.1 (i.e., by Algorithm $\mathcal{A}$). The set of all the types of signatures of an SPDI will be denoted by $\mathcal{T}$. The set of types of signatures from one edge e_0 to other edge e_f will be denoted by $\mathcal{T}(e_0, e_f)$.

Example 4.5 The type of the signature $\sigma_{\mathcal{A}} = abc(dbc)^1 ef(gef)^3 hi$ of Example 4.2 is $\texttt{type}(\sigma) = abc, (dbc), ef, (gef), hi$. The type of $\sigma_{\mathcal{A}} = abc(dbc)^1 e(fge)^4 f$ is $\texttt{type}(\sigma_{\mathcal{A}}) = abc, (dbc), e, (fge), f$. And the type of the signature of Example 4.3 is $\texttt{type}(\sigma) = e_1 e_2 e_3, (e_4 e_1 e_2 e_3), e_5 e_6, (e_7 \cdots e_{13} e_6), e_{14} e_{15}$. $\blacksquare$

We have defined signatures as being arbitrary sequences of edges but we are particularly interested in signatures that correspond to trajectory segments.

Definition 4.6 *We say that a signature σ is* feasible *if and only if there exists a trajectory segment ξ with signature σ, i.e.,* $\mathsf{Sig}(\xi) = \sigma$.

The set of all the types of feasible signatures will be denoted by $\mathcal{T}_{feasible}$.

Given a type of signature we want to characterize the set of all the signatures with such type, that is the set of signatures that *concretize* the type.

Definition 4.7 *Given a type of signature $\tau = r_1, s_1, \ldots, s_n, r_{n+1}$, the* concretization *of τ is the set of all edge signatures with type τ, i.e.,*

$$\texttt{Concr}(\tau) = \{r_1 s_1^{k_1} \dots s_n^{k_n} r_{n+1} \mid k_i \in \mathbb{N}^+, 1 \leq i \leq n\}.$$

4.3 Properties of Types of Feasible Signatures

Let ξ be a trajectory segment with edge signature $\mathsf{Sig}(\xi) = e_0 \dots e_p$, and region signature $\mathsf{RSig}(\xi) = P_0 \dots P_p$.

Definition 4.8 *An edge e is said to be* abandoned *by ξ after position i, if $e_i = e$ and for some j, k, $i \leq j < k$, $P_j \dots P_k$ forms a region cycle and $e \notin \{e_{i+1}, \dots, e_k\}$. Since trajectory segments are* finite *we allow also the trivial case when $e \neq e_j$ for all $j, j > i$.*

Intuitively, the following lemma guarantees that any edge that occurs in a prefix of an edge signature but does not appear in a cycle following this prefix cannot occur anymore in any postfix (starting with the cycle) of the edge signature.

Lemma 4.9 (Abandonment is Irreversible) *For every trajectory segment ξ and edge e, if e is abandoned by ξ after position i, e will not appear in $\mathsf{Sig}(\xi)$ at any position $j > i$.*

SKETCH OF THE PROOF. Let us consider a trajectory ξ that abandons e. Since ξ is not self-crossing by Lemma 3.6 the sequence of points determined by the intersection of ξ with e (a prefix of its signature) is monotone. After abandoning the edge e the only possibility to "visit" the same edge again is by violating the monotonicity property. See Claim 2 in [AMP95] for a complete proof. □

Example 4.10 Let us consider the trajectory segment from $\mathbf{x}$ to $\mathbf{x}'$ of Figure 14, with signature $\sigma_{\mathcal{A}} = e_1 e_2 e_3 (e_4 e_1 e_2 e_3)^2 e_5 e_6 (e_7 \cdots e_{13} e_6)^2 e_{14} e_{15}$. In order to visualize the position, we unfold the above signature and we write the occurrence position of each edge as a superscript [1] :

$$\sigma_{\mathcal{A}} = e_1^1 e_2^2 e_3^3 (e_4^4 e_1^5 e_2^6 e_3^7)(e_4^8 e_1^9 e_2^{10} e_3^{11}) e_5^{12} e_6^{13} (e_7^{14} \cdots e_{13}^{20} e_6^{21})(e_7^{22} \cdots e_{13}^{28} e_6^{29}) e_{14}^{30} e_{15}^{31}.$$

Notice that $R_6 R_7 \dots R_{11} R_{12} R_5$ forms a region cycle with positions 13, 14, ..., 19 and 20 respectively. Edge e_5, for instance, is abandoned after position 12 since it does not belong to the set of edges $\{e_6, e_7, \dots e_{13}\}$ (that have positions 13, 14, ..., 20 respectively). Moreover, e_5 cannot appear in any extension of the above trajectory segment from $\mathbf{x}'$. Moreover, edges e_1 to e_4 are also abandoned at positions 9, 10, 11, and 8, respectively. ■

[1] We have kept the parentheses in order to visualize the cycles.

We have that the types of feasible signatures have the following properties.

Lemma 4.11 *Let $\sigma = e_0 \dots e_p$ be a feasible signature, then its type,* $\texttt{type}(\sigma) = r_1, s_1, \dots, r_n, s_n, r_{n+1}$ *satisfies the following properties:*

$\mathbf{P_1}$. *For every $1 \leq i \neq j \leq n+1$, r_i and r_j are disjoint;*
$\mathbf{P_2}$. *For every $1 \leq i \neq j \leq n$, s_i and s_j are different.*

PROOF.

$\mathbf{P_1}$. Let $e \in r_i$; we consider two cases:
  (1) $e \notin s_i$: The result follows immediately from Lemma 4.9 (e cannot occur in any r_j, $j > i$);
  (2) $e \in s_i$: Suppose that $e \in r_{i+1}$. Then we have $s_i = e_1 \cdots e_i \cdots e_k$ and $r_{i+1} = e_{k+1} \cdots e_j \cdots e_l$, with $e_i = e_j$, but this is not possible: the construction of σ was done backwards, and in this case we should have a cycle $s = e_{i+1} \cdots e_k e_{k+1} \cdots e_j$. If $e \in r_j$ (for any $j > i+1$) then again we have two cases: $e \in s_{j-1}$ or $e \notin s_{j-1}$; the first case is not possible by construction and the latter contradicts Lemma 4.9.

$\mathbf{P_2}$. Let $s_i = e_1, \dots, e_k$ be a simple cycle. After cycling k_i times the cycle is abandoned by edge e_k (by construction of $\sigma_{\mathcal{A}}$). Let P be a region s.t. $e_k \in \mathit{In}(P)$ and consider the unfolding of the last iteration and its continuation: $\dots, e_1, e_2, \dots, e_k, e, \dots$, where, by feasibility, $e = \mathsf{first}(r_{i+1})$, $e_k \in \mathit{In}(P)$ and $e_1, e \in \mathit{Out}(P)$ ($e_1 \neq e$). By the ordering between edges we have that either $e \prec e_1$ or $e_1 \prec e$. By the monotonicity of the trajectory, in both cases e_1 cannot occur after e in σ. Thus, any other cycle s_j, with $i < j$, differs from s_i at least on e_1. Hence, all the cycles are different. $\square$

We denote the set of types of signatures satisfying properties $\mathbf{P_1}$ and $\mathbf{P_2}$ by $\mathcal{T}_P$. By Lemma 4.11,

$$\mathcal{T}_{feasible} \subseteq \mathcal{T}_P.$$

We have the following proposition.

Proposition 4.12 *The set $\mathcal{T}_P$, and hence the set of types of feasible signatures $\mathcal{T}_{feasible}$ are finite.* $\square$

In our reachability algorithm we will use the larger but still finite set of types of signatures $\mathcal{T}_P$ instead of $\mathcal{T}_{feasible}$, because the former one is described by simple syntactic properties $\mathbf{P_1}$ and $\mathbf{P_2}$ and can be easily enumerated.

Remember that the point-to-point reachability for SPDIs can be stated as:

$$\mathit{Reach}(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \equiv \exists \xi \, \exists t \geq 0 \; . \; (\xi(0) = \mathbf{x}_0 \wedge \xi(t) = \mathbf{x}_f),$$

and for a given ξ, we have the following predicate:

$$Reach_\xi(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \equiv \exists t \geq 0 \;.\; (\xi(0) = \mathbf{x}_0 \wedge \xi(t) = \mathbf{x}_f).$$

Let us define the reachability following a given signature as:

$$Reach_\sigma(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \equiv \exists \xi \;.\; (\mathsf{Sig}(\xi) = \sigma \wedge Reach_\xi(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)).$$

Finally, the following predicate defines the point-to-point reachability for a given type of signature τ:

$$Reach_\tau(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \equiv \exists \sigma \in \mathtt{Concr}(\tau) \;.\; Reach_\sigma(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)).$$

Putting together the steps presented in this section we obtain the following result.

Theorem 4.13 *Given an SPDI $\mathcal{H}$ and two points $\mathbf{x}_0$ and $\mathbf{x}_f$, then the following holds:*

$$Reach(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \;\; \textit{iff} \;\; Reach_\tau(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \textit{ for some } \tau \in \mathcal{T}_P.$$

Thus, by Proposition 4.12, to solve the reachability problem we can proceed by examining one by one the types of signatures that guarantee to preserve reachability by the above theorem.

5 Affine Multivalued Operators

In this section we introduce a class of functions called *truncated affine multi-valued functions* (TAMFs) and we study some of its properties. TAMFs serve as a theoretical basis for the reachability analysis presented in section 7. See the Appendix for a proof of the results presented here and other auxiliary lemmas concerning TAMFs.

Definition 5.1 *A* positive affine function[2] *$f : \mathbb{R} \to \mathbb{R}$ is defined by a formula $f(x) = ax + b$ with $a > 0$.*

Affine functions can be extended to *multi-valued* functions.

Definition 5.2 *An* affine multi-valued operator (AMF) *$F : \mathbb{R} \to 2^{\mathbb{R}}$ is determined by two affine functions $f_l(x)$ and $f_u(x)$; it maps x to the interval $\langle f_l(x), f_u(x)\rangle$, where $\langle a, b\rangle$ means (a, b), $[a, b]$, $(a, b]$ or $[a, b)$:*

$$F(x) = \langle f_l(x), f_u(x)\rangle$$

[2] We will sometimes omit the word "positive".

with $\mathsf{Dom}(F) = \{x \mid f_l(x) \leq f_u(x)\}$.

We use the notation $F = \langle f_l, f_u\rangle$. Such an operator can be naturally extended to subsets of $\mathbb{R}$:

$$F(S) = \bigcup_{x \in S} F(x).$$

In particular, if $S = \langle l, u\rangle$ is an interval, then:

$$F(\langle l, u\rangle) = \langle f_l(l), f_u(u)\rangle,$$

where the domain of F is given by $\mathsf{Dom}(F) = \{\langle l, u\rangle \mid f_l(l) \leq f_u(u)\}$ (we consider just well-formed intervals $\langle l, u\rangle$, i.e. with $l \leq u$).

We are interested in considering a kind of affine function restricted with respect to some intervals.

Definition 5.3 *A* truncated affine multi-valued operator (TAMF) $\mathcal{F}_{F,S,J} : \mathbb{R} \to 2^{\mathbb{R}}$ *is determined by an affine multi-valued operator* F *and intervals* $S \subseteq \mathbb{R}^+$ *and* $J \subseteq \mathbb{R}^+$ *as follows:*

$$\mathcal{F}_{F,S,J}(x) = \begin{cases} F(x) \cap J & \text{if } x \in S \\ \emptyset & \text{otherwise.} \end{cases}$$

A TAMF can also be expressed as $\mathcal{F}_{F,S,J}(x) = F(\{x\} \cap S) \cap J$, or as $\mathcal{F}_{F,S,J}(x) = F\mid_S(x) \cap J$, where $F\mid_S$ stands for the restriction of F to S. We use calligraphic typeface to denote TAMF operators and in general we will write $\mathcal{F}$ instead of $\mathcal{F}_{F,S,J}$.

Truncated affine multi-valued functions can be also extended to sets and in particular to intervals, as shown in what follows.

$$\begin{aligned} \mathcal{F}(I) &= \textstyle\bigcup_{x \in I} \mathcal{F}(x) && \text{(by definition)} \\ &= \textstyle\bigcup_{x \in I} F(\{x\} \cap S) \cap J && \text{(by definition of } \mathcal{F}\text{)} \\ &= F(\cup_{x \in I}\{x\} \cap S) \cap J && = F(I \cap S) \cap J. \end{aligned}$$

We define the inverse of an AMF:

Definition 5.4 *The* inverse *of* F *is defined by* $F^{-1}(x) = \{y \mid x \in F(y)\}$.

It is not difficult to show that $F^{-1} = \langle f_u^{-1}, f_l^{-1}\rangle$ and the inverse of a TAMF $\mathcal{F}$ is given by the following Lemma:

Lemma 5.5 *Given a* $\mathcal{F}(I) = F(I \cap S) \cap J$, *then* $\mathcal{F}^{-1}(I) = F^{-1}(I \cap J) \cap S$.

Definition 5.6 *A TAMF $\mathcal{F}$ is* normalized *if $S = \mathsf{Dom}(\mathcal{F}) = \{x \mid F(x) \cap J \neq \emptyset\}$ and $J = \mathsf{Im}(\mathcal{F})$.*

Notice that, for normalized TAMFs, $S \subseteq F^{-1}(J)$ and $J = \mathcal{F}(S)$. In fact, any TAMF can be normalized as stated in the following lemma.

Lemma 5.7 *Every TAMF $\mathcal{F}$ can be represented in normal form.*

In what follows, we consider just TAMFs in normal form. The following result shows an important property of affine operators, that is the closure under composition.

Lemma 5.8 (composition of affine operations) *Affine functions, affine multi-valued operators, and truncated affine multi-valued operators are closed under composition.*

In particular, as proved in the Appendix (Lemma A.5), for

$$\mathcal{F}_1(x) = F_1(\{x\} \cap S_1) \cap J_1; \quad \mathcal{F}_2(x) = F_2(\{x\} \cap S_2) \cap J_2$$

we have that

$$\mathcal{F}_2 \circ \mathcal{F}_1(x) = F'(\{x\} \cap S') \cap J'$$

with

$$F' = F_2 \circ F_1; J' = J_2 \cap F_2(J_1 \cap S_2); S' = S_1 \cap F_1^{-1}(J_1 \cap S_2).$$

Example 5.9 Let $x \in J_0$ (where $J_0 = [0, 1]$), and

$$F_1(x) = \left(2x - \frac{3}{5},\, 3x + 5\right], \qquad F_2(x) = [5x + 2,\, 7x + 6]$$

be two (non-truncated) affine multi-valued functions, $\mathcal{F}_1 = F_1 \cap J_1$ (with $J_1 = (1,\, 6]$), and $\mathcal{F}_2 = F_2 \cap J_2$ (with $J_2 = [6,\, 10)$) their truncated versions. We have that

$$F_1^{-1}(y) = \left(\frac{y-5}{3}, \frac{5y+3}{10}\right], \qquad F_2^{-1}(y) = \left[\frac{y-6}{7}, \frac{y-2}{5}\right].$$

To obtain $\mathcal{F}_2 \circ \mathcal{F}_1(x)$ we need to compute F', S' and J' as in Lemma 5.8 but first we compute S_1 and S_2:

$$S_1 = F_1^{-1}(J_1) \cap J_0 = F_1^{-1}((1,6]) \cap [0,1] = \left(-\frac{4}{3}, \frac{33}{10}\right) \cap [0,1] = [0,1];$$

$$S_2 = F_2^{-1}(J_2) \cap J_1 = F_2^{-1}([6,10)) \cap (1,6] = \left[0, \frac{8}{5}\right) \cap (1,6] = \left(1, \frac{8}{5}\right);$$

$$S' = S_1 \cap F_1^{-1}(J_1 \cap S_2) = [0,1] \cap F_1^{-1}\left((1,6] \cap \left(1, \frac{8}{5}\right)\right) =$$

$$= [0,1] \cap F_1^{-1}\left(\left(1, \frac{8}{5}\right)\right) = [0,1] \cap \left(-\frac{4}{3}, \frac{11}{10}\right) = [0,1];$$

$$J' = J_2 \cap F_2(J_1 \cap S_2) = [6,10) \cap F_2\left((1,6] \cap \left(1, \frac{8}{5}\right)\right) =$$

$$= [6,10) \cap F_2\left(\left(1, \frac{8}{5}\right)\right) = [6,10) \cap (7,10) = (7,10);$$

$$F'(x) = F_2 \circ F_1(x) = \left(5\left(2x - \frac{3}{5}\right) + 2,\ 7(3x+5) + 6\right] = (10x - 1,\ 21x + 41].$$

Hence, the truncated affine multi-valued operator $\mathcal{F}_2 \circ \mathcal{F}_1(x)$ is

$$\mathcal{F}_2 \circ \mathcal{F}_1(x) = \begin{cases} (10x - 1,\ 21x + 41] \cap (7,10) & \text{if } x \in [0,1] \\ \emptyset & \text{otherwise.} \end{cases} \qquad \blacksquare$$

Another useful result gives the fixpoints of AMFs:

Lemma 5.10 *Let $\langle l_0, u_0 \rangle$ be any initial interval and $\langle l_n, u_n \rangle = F^n(\langle l_0, u_0 \rangle)$. 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.

We use the notation $\widehat{\mathcal{F}}$ for truncated affine multi-valued operators with $S = J$; i.e. the image and the domain coincide (we denote this set by H) and then $\widehat{\mathcal{F}}(I) = F(I \cap H) \cap H$. The following TAMF property will have a key role in the acceleration of cycles when computing successors for the reachability algorithm in section 7.

Lemma 5.11 (Fundamental lemma) *Let $\widehat{\mathcal{F}}$ be a truncated affine multi-valued operator. Then $\widehat{\mathcal{F}}^n(I) = F^n(I \cap H)\ \cap\ H$.*

Intuitively, what the above lemma says is that in order to obtain the iterated truncated affine multi-valued function truncated with an interval H (both the argument and the final result), we only need to iterate the non-truncated function intersecting the argument just once at the beginning and once at the end.

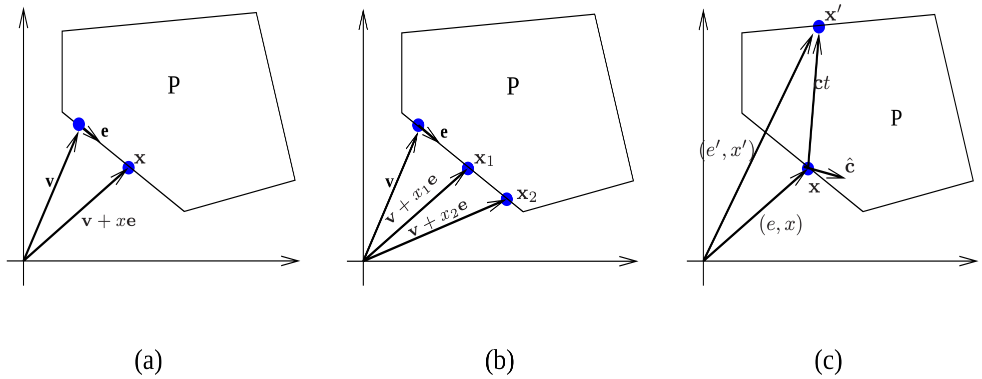


Fig. 15. (a) Representation of edges; (b) Representation of an interval; (c) One-step successor.

6 Successor Function

Let us introduce a one-dimensional coordinate system on each edge. For each edge e we chose a point on it (the origin) with radius-vector $\mathbf{v}$, and a director vector $\mathbf{e}$ going in the positive direction in the sense of the order $\prec$.

To characterize e we need the coordinates of its extreme points: two more numbers $e^l, e^u \in \mathbb{Q} \cup \{-\infty, \infty\}$ such that $e = \{\mathbf{v} + x\mathbf{e} \mid e^l < x < e^u\}$. Clearly, having fixed $\mathbf{v}$ and $\mathbf{e}$ for every edge we can represent every point $\mathbf{x} \in e$ by a pair (e, x) identifying the edge e and the coordinate x (see Fig.15(a)). Every interval $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$ contained in e is represented as $(e, \langle x_1, x_2 \rangle)$, where $\mathbf{x}_1 = (e, x_1)$ and $\mathbf{x}_2 = (e, x_2)$ (see Fig.15(b)). Notice that if e is a vertex, then $e = \{\mathbf{v}\}$, where $\mathbf{v}$ is the only vector that characterizes e. Moreover, all the vertices have local coordinates $x \in [0, 0]$, i.e. a vertex v is represented by a pair $(v, 0)$; hence, whenever e is a vertex, $e = \langle e^l, e^u \rangle$ must be understood as $e = [e^l, e^u]$ whereas if e is a "true" edge then $e = (e^l, e^u)$.

We define the *edge-to-edge successor* $\mathsf{Succ}^{\mathbf{c}}_{ee'}$ following a given vector $\mathbf{c}$.

Definition 6.1 *Let $e \in In(P)$ and $e' \in Out(P)$ be two edges, $\mathbf{x} = (e, x)$ a point, and $\mathbf{c} \in \phi(P)$ a given vector. The* edge-to-edge successor following a given vector $\mathbf{c}$ *is defined as*

$$\mathsf{Succ}^{\mathbf{c}}_{ee'}(x) = x',$$

where $\mathbf{x}' = (e', x')$ is a point such that $\mathbf{x}' = \mathbf{x} + \mathbf{c}t$ for some $t > 0$.

Notice that $\mathbf{x}'$ is unique. We say that the point (e', x') is the successor of (e, x) in the direction $\mathbf{c}$ (see Fig.15(c)). We prove now that successors are TAMFs.

Lemma 6.2 *The function $\mathsf{Succ}^{\mathbf{c}}_{ee'}$ is truncated affine.*

PROOF. Let $e = \langle e^l, e^u \rangle$ and $e' = \langle e'^l, e'^u \rangle$.

Expanding $\mathbf{x}' = \mathbf{x} + \mathbf{c}t$, we obtain $\mathbf{v}' + x'\mathbf{e}' = \mathbf{v} + x\mathbf{e} + t\mathbf{c}$. Multiplying both expressions by $\hat{\mathbf{c}}$ (the right rotation of $\mathbf{c}$) and eliminating x' we obtain $x' = \alpha(\mathbf{c})x + \beta(\mathbf{c})$ with $\alpha(\mathbf{c}) = \frac{\mathbf{e}\hat{\mathbf{c}}}{\mathbf{e}'\hat{\mathbf{c}}}$ and $\beta(\mathbf{c}) = \frac{\mathbf{v}-\mathbf{v}'}{\mathbf{e}'\hat{\mathbf{c}}}\hat{\mathbf{c}}$. With our choice of orientation of director vectors for e and e', the coefficient $\alpha(\mathbf{c})$ is always positive.

Notice that we have $x' = \mathsf{Succ}^{\mathbf{c}}_{ee'}(x)$ iff $x \in e$, $x' \in e'$ and $x' = \alpha(\mathbf{c})x + \beta(\mathbf{c})$. Thus, $x' = F(\{x\} \cap S) \cap J$ with $F(x) = \alpha(\mathbf{c})x + \beta(\mathbf{c})$, $S = \langle e^l, e^u \rangle$ and $J = \langle e'^l, e'^u \rangle$, i.e. $x' = \mathcal{F}_{F,\langle e^l,e^u \rangle,\langle e'^l,e'^u \rangle}$. $\square$

The notion of *successor* can be extended on all possible directions $\mathbf{c} \in \phi(P)$. $\mathsf{Succ}_{ee'}(x)$ is the set of all points in e' reachable from $\mathbf{x}$ by a trajectory segment in P. More formally,

Definition 6.3 *Let $P \in \mathbb{P}$, $e \in In(P)$ and $e' \in Out(P)$. For $\mathbf{x} = (e, x)$, the* edge-to-edge successor $\mathsf{Succ}_{ee'}(x)$ *is defined as*

$$\mathsf{Succ}_{ee'}(x) = \{x' \mid \mathbf{x}' = (e', x') \wedge \xi(0) = x \wedge \xi(t) = x' \wedge \mathsf{Sig}(\xi) = ee'\}.$$

$F^{\mathbf{c}}_{ee'}(x)$ will denote the *non-truncated* function $\alpha(\mathbf{c})x + \beta(\mathbf{c})$. The above notion of *successor* can be applied to any subset $A \subseteq \langle e^l, e^u \rangle$ and in particular to intervals $\langle l, u \rangle$:

Lemma 6.4 *Let $\phi(P) = \angle^{\mathbf{b}}_{\mathbf{a}}$, $\mathbf{x} = (e, x)$ and $\langle l, u \rangle \subseteq \langle e^l, e^u \rangle$. Then:*

(1) $\mathsf{Succ}_{ee'}(x) = \bigcup_{\mathbf{c} \in \phi(P)} \mathsf{Succ}^{\mathbf{c}}_{ee'}(x) = \left[F^{\mathbf{b}}_{ee'}(x), F^{\mathbf{a}}_{ee'}(x)\right] \cap \langle e'^l, e'^u \rangle$;
(2) $\mathsf{Succ}_{ee'}(\langle l, u \rangle) = \langle F^{\mathbf{b}}_{ee'}(l), F^{\mathbf{a}}_{ee'}(u) \rangle \cap \langle e'^l, e'^u \rangle$.

PROOF. It follows from the results given in section 5. $\square$

Therefore, $\mathsf{Succ}_{ee'}$ is truncated affine multivalued:

$$\mathsf{Succ}_{ee'}(\langle l, u \rangle) = F_{ee'}(\langle l, u \rangle \cap \langle e^l, e^u \rangle) \cap \langle e'^l, e'^u \rangle.$$

This lemma shows that in order to find a successor of an interval in an edge e, we should apply the rightmost dynamics ($\mathbf{a}$) to its right end and the leftmost ($\mathbf{b}$) to its left end, and intersect the result with the target edge. Fig. 16 shows the difference between non-truncated and truncated successors.

The successor operator will be used as a building block in the reachability algorithm. It can be naturally extended on edge signatures: for $\sigma_1 = e_1 e_2 \ldots e_n$

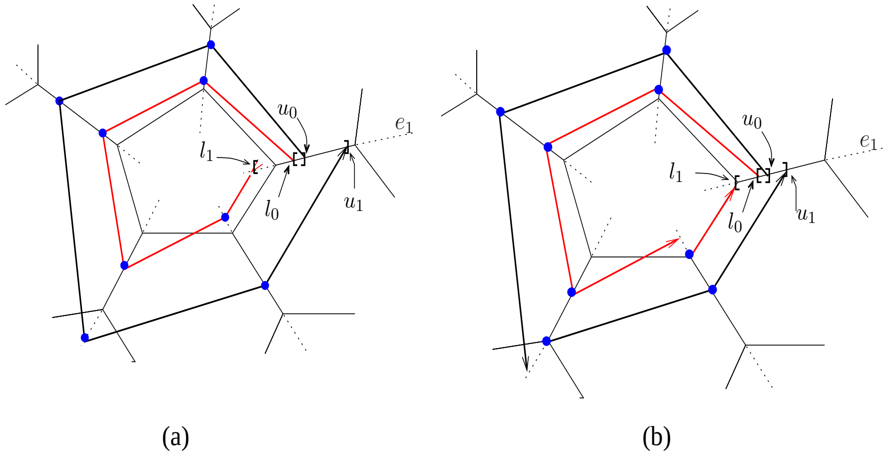


Fig. 16. (a) Non-truncated operator: $\mathsf{Succ}_{e_1}(l_0, u_0) = \langle l_1, u_1 \rangle$, with $l_1 < e_1^l < u_1 \leq e_1^u$; (b) Truncated successor: $\mathsf{Succ}_{e_1}(l_0, u_0) = \langle l_1, u_1 \rangle \in \langle e_1^l, e_1^u \rangle$.

let $\mathsf{Succ}_{\sigma_1}(I) = \mathsf{Succ}_{e_{n-1}e_n} \circ \cdots \circ \mathsf{Succ}_{e_2e_3} \circ \mathsf{Succ}_{e_1e_2}(I)$ that by Lemma 5.8 is truncated affine.

Notice that since we use edge signatures the semi-group property takes the following form.

Lemma 6.5 *For any edge signatures σ_1 and σ_2 and an edge e*

$$\mathsf{Succ}_{e\sigma_1} \circ \mathsf{Succ}_{\sigma_2 e} = \mathsf{Succ}_{\sigma_2 e \sigma_1}.$$

It is convenient to define a (trivial) successor Succ_e where e is a single edge. The only way to do it preserving the semi-group property is to put $\mathsf{Succ}_e(x) = x$.

In order to manipulate successor operators we should investigate their algebraic properties. Since one-step successors $\mathsf{Succ}_{e_1e_2}$ are truncated affine, Lemma 6.5 and Lemma 5.8 guarantee that all the multi-step Succ_u are truncated affine as well. In the sequel we will apply the iteration analysis to their non-truncated versions F_u.

The following result plays a technical role in the reachability algorithm.

Lemma 6.6 *Let P be a region, $\phi(P) = \angle_{\mathbf{a}}^{\mathbf{b}}$ its dynamics, $e \in In(P)$, $e_1, e_2 \in Out(P)$, and $F_{ee_i}(x) = \mathcal{F}_i(x) = F_i(\{x\} \cap S_i) \cap J_i$ be a truncated affine multi-valued function (with $F_i = [f_i^l, f_i^u]$ and $J_i = \langle L_i, U_i \rangle$). Given that $e_2 \prec e_1$ we have that*

(1) if $L_1 < f_1^l(x)$ then $\mathcal{F}_2(x) = \emptyset$;
(2) if $f_2^u(y) < U_2$ then $\mathcal{F}_1(x) = \emptyset$.

PROOF: (See Fig. 17).

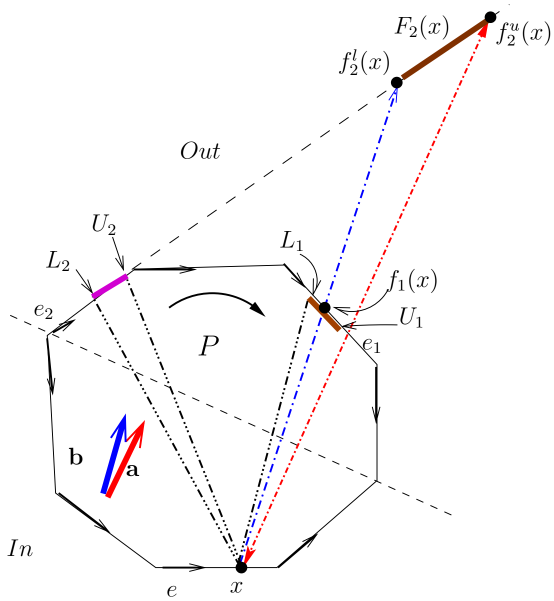


Fig. 17. Proof of Lemma 6.6.

(1) Looking from the point x, the directions to (e_2, L_2), (e_2, U_2), (e_1, L_1), the vector $\mathbf{b}$, the set $(e_2, F_2(x))$ and the vector $\mathbf{a}$ are situated in the clockwise order. This implies emptiness of $F_2(x) \cap \langle L_2, U_2 \rangle = \mathcal{F}_2(x)$.

(2) Similar. $\square$

Example 6.7 Let us come back to the example of the swimmer trying to escape from a whirlpool in a river (see Fig. 2). Suppose that the swimmer is following a trajectory with edge signature $(e_1 \dots e_8)^*$. It is not difficult to find a representation of the edges such that for each edge e_i, $(e_i^l, e_i^u) = (0, 1)$. Besides, the (non-truncated) affine successor functions are:

$$F_{e_1e_2}(x) = \left\{\frac{x}{2}\right\}; \qquad F_{e_ie_{i+1}}(x) = \{x\}, \text{ for all } i \in [3, 7];$$

$$F_{e_2e_3}(x) = \left[x - \frac{1}{4}, x + \frac{11}{60}\right]; \qquad F_{e_8e_1}(x) = \left\{x + \frac{1}{5}\right\}.$$

The truncated affine version of the functions above (normalized) are

$$\mathsf{Succ}_{e_1e_2}(x) = \begin{cases} \left\{\frac{x}{2}\right\} \cap (0,1) & \text{if } x \in (0,1) \\ \emptyset & \text{otherwise;} \end{cases}$$

$$\mathsf{Succ}_{e_2e_3}(x) = \begin{cases} \left[x - \frac{1}{4}, x + \frac{11}{60}\right] \cap (0,1) & \text{if } x \in (0,1) \\ \emptyset & \text{otherwise;} \end{cases}$$

$$\mathsf{Succ}_{e_ie_{i+1}}(x) = \begin{cases} \{x\} \cap (0,1) & \text{if } x \in (0,1) \\ \emptyset & \text{otherwise;} \end{cases}$$

$$\mathsf{Succ}_{e_8e_1}(x) = \begin{cases} \left\{x + \frac{1}{5}\right\} \cap (0,1) & \text{if } x \in (0, \frac{4}{5}) \\ \emptyset & \text{otherwise.} \end{cases}$$

The successor function for the loop $s = e_1 \ldots e_8$ is obtained by composition of the above functions as follows. Let us first compute $\mathsf{Succ}_{e_1e_2e_3}(x) = F(\{x\} \cap S) \cap J$, where

$$F = F_{e_2e_3} \circ F_{e_1e_2}, \quad S = S_1 \cap F^{-1}_{e_1e_2}(J_1 \cap S_2), \quad J = J_2 \cap F_{e_2e_3}(J_1 \cap S_2),$$

with

$$\begin{aligned} J_0 &= e_1 = (0,1), \\ J_1 &= e_2 = (0,1), \qquad & S_1 &= F^{-1}_{e_1e_2}(J_1) \cap J_0, \\ J_2 &= e_3 = (0,1), \qquad & S_2 &= F^{-1}_{e_2e_3}(J_2) \cap J_1, \end{aligned}$$

and

$$F^{-1}_{e_1e_2}(x) = \{2x\}, \quad F^{-1}_{e_2e_3}(x) = \left[x - \frac{11}{60}, x + \frac{1}{4}\right].$$

We compute now all the parameters above in order to obtain F, S and J

$$\begin{aligned} S_1 \quad &= F^{-1}_{e_1e_2}((0,1)) \cap (0,1) = (0,2) \cap (0,1) = (0,1); \\ S_2 \quad &= F^{-1}_{e_2e_3}((0,1)) \cap (0,1) = \left(-\tfrac{11}{60}, \tfrac{5}{4}\right) \cap (0,1) = (0,1); \\ F(x) &= \left[\tfrac{x}{2} - \tfrac{1}{4}, \tfrac{x}{2} + \tfrac{11}{60}\right]; \\ S \quad &= (0,1) \cap F^{-1}_{e_1e_2}((0,1) \cap (0,1)) = (0,1) \cap (0,2) = (0,1); \\ J \quad &= (0,1) \cap F_{e_2e_3}((0,1) \cap (0,1)) = (0,1) \cap \left(-\tfrac{1}{4}, \tfrac{71}{60}\right) = (0,1). \end{aligned}$$

We have then that

$$\mathsf{Succ}_{e_1e_2e_3}(x) = \begin{cases} \left[\frac{x}{2} - \frac{1}{4}, \frac{x}{2} + \frac{11}{60}\right] \cap (0,1) & \text{if } x \in (0,1) \\ \emptyset & \text{otherwise.} \end{cases}$$

Since $F_{e_ie_{i+1}}$ for $i \in [3,7]$ are the identity functions, we have that

$$\mathsf{Succ}_{e_3\ldots e_8}(x) = \begin{cases} \{x\} \cap (0,1) & \text{if } x \in (0,1) \\ \emptyset & \text{otherwise,} \end{cases}$$

and composing the functions above we obtain $\mathsf{Succ}_{e_1\ldots e_8} = \mathsf{Succ}_{e_1e_2e_3}$. We compute now $\mathsf{Succ}_{e_1\ldots e_8e_1}(x) = F'(\{x\} \cap S') \cap J'$, where

$$F' = F_{e_8e_1} \circ F_{e_1\ldots e_8}, \quad S' = S_1 \cap F^{-1}_{e_1\ldots e_8}(J_1 \cap S_2), \quad J' = J_2 \cap F_{e_8e_1}(J_1 \cap S_2),$$

with

$$\begin{aligned} J_0 &= e_1 = (0,1), \\ J_1 &= J = (0,1), & S_1 &= F^{-1}_{e_1\ldots e_8}(J_1) \cap J_0, \\ J_2 &= e_1 = (0,1), & S_2 &= F^{-1}_{e_8e_1}(J_2) \cap J_1, \end{aligned}$$

and

$$\begin{aligned} F^{-1}_{e_1\ldots e_8}(x) &= \left[2x - \tfrac{11}{30}, 2x + \tfrac{1}{2}\right], \\ F^{-1}_{e_8e_1}(x) &= \left\{x - \tfrac{1}{5}\right\}. \end{aligned}$$

We compute the parameters above to obtain F', S' and J':

$$\begin{aligned} S_1 &= F^{-1}_{e_1\ldots e_8}((0,1)) \cap (0,1) = \left(-\tfrac{11}{30}, \tfrac{5}{2}\right) \cap (0,1) = (0,1); \\ S_2 &= F^{-1}_{e_8e_1}((0,1)) \cap (0,1) = \left(-\tfrac{1}{5}, \tfrac{4}{5}\right) \cap (0,1) = \left(0, \tfrac{4}{5}\right); \\ F'(x) &= \left[\tfrac{x}{2} - \tfrac{1}{20}, \tfrac{x}{2} + \tfrac{23}{60}\right]; \\ S' &= (0,1) \cap F^{-1}_{e_1\ldots e_8}\left((0,1) \cap \left(0, \tfrac{4}{5}\right)\right) = (0,1) \cap (-\tfrac{11}{30}, \tfrac{21}{10}) = (0,1); \\ J' &= (0,1) \cap F_{e_8e_1}\left((0,1) \cap \left(0, \tfrac{4}{5}\right)\right) = (0,1) \cap (\tfrac{1}{5}, 1) = \left(\tfrac{1}{5}, 1\right). \end{aligned}$$

Hence,

$$\mathsf{Succ}_{e_1\ldots e_8e_1}(x) = \begin{cases} [\tfrac{x}{2} - \tfrac{1}{20}, \tfrac{x}{2} + \tfrac{23}{60}] \cap (\tfrac{1}{5}, 1) & \text{if } x \in (0,1) \\ \emptyset & \text{otherwise.} \end{cases}$$

Finally, by Lemma 5.10 we obtain the limits: $l^* = \left(-\frac{1}{20}\right) / \left(1 - \frac{1}{2}\right) = -\frac{1}{10}$, and $u^* = \left(\frac{23}{60}\right) / \left(1 - \frac{1}{2}\right) = \frac{23}{30}$. ■

The notion of *edge signature* introduced in the previous section allows to consider one dimensional discrete systems instead of the two dimensional continuous systems we are dealing with. The following evident lemma shows that a successor function computes the Poincaré map of a trajectory segment.

Lemma 6.8 *Given an SPDI $\mathcal{H}$ and two points $\mathbf{x}_0 = (e_0, x_0)$ and $\mathbf{x}_f = (e_f, x_f)$, the predicate $Reach_\sigma(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)$ holds iff $x_f \in \mathsf{Succ}_\sigma(x_0)$.*

function $Reach(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)$
  for each $\tau \in \mathcal{T}(e_0, e_f)$
    if $(Reach_{type}(x_0, x_f, \tau))$
      then $\longleftarrow$ true
  $\longleftarrow$ false

Fig. 18. Main algorithm.

function $Reach_{type}(x_0, x_f, \tau)$:
  $Z = \mathsf{Succ}_{r_1 f_1}(x_0)$
  for $i = 1$ *to* $n-1$
    $Z = \mathsf{Succ}_{r_{i+1} f_{i+1}}(Exit(Z, s_i, e_{x_i}))$
  if $\mathsf{loop}_{\mathsf{end}}(\tau)$
    then $\longleftarrow Test(Z, s_n, x_f)$
    else $\longleftarrow x_f \in \mathsf{Succ}_{r_{n+1}}(Exit(Z, s_n, e_{x_n}))?$

Fig. 19. $Reach_{type}$ function.

7 Reachability Analysis

In this section we present our main result, namely a decision procedure to solve the reachability problem for SPDIs. We adopt here the top-down programming style.

7.1 Main algorithm

Given an SPDI $\mathcal{H}$, we are interested in the reachability analysis between two points. We know that there exists a finite number of types of signatures in $\mathcal{T}_P$ of the form $r_1, s_1 \ldots\ r_n, s_n, r_{n+1}$. Moreover, the types of signatures are restricted to those with $e_0 = \mathsf{first}(r_1)$ and $e_f \in r_{n+1}$. Given such a set of types of signatures $\mathcal{T}(e_0, e_f)$, the algorithm shown in Fig. 18 is guaranteed to terminate, answering YES if $\mathbf{x}_f$ is reachable from $\mathbf{x}_0$ or NO otherwise:

Reachability from $\mathbf{x}_0$ to $\mathbf{x}_f$ with fixed type of signature τ is tested by the function $Reach_{type}(x_0, x_f, \tau)$, shown in Fig. 19.

Let the type τ have the form $\tau = r_1, s_1, \ldots, r_n, s_n, r_{n+1}$. Put $f_i = \mathsf{first}(s_i)$ and $e_{x_i} = \mathsf{first}(r_{i+1})$ if r_{i+1} is non-empty and f_{i+1} otherwise (i.e. e_{x_i} is the edge to which the trajectory exits from the loop s_i). Let us say that a type of signature τ has a $\mathsf{loop}_{\mathsf{end}}$ property if $\mathsf{first}(r_{n+1}) = \mathsf{first}(s_n)$, i.e. signatures of type τ terminate by several repetitions of the last loop.

$Reach_{type}(\cdot, \cdot, \cdot)$ uses two functions:

(1) $Test(Z, s, x)$ that answers whether x is reachable from a set Z (represented as a finite union of intervals) in the loop s. Formally, it checks whether $x \in \mathsf{Succ}_{s^+\mathsf{first}(s)}(I)$, i.e.,

$$\exists k \geq 1 \; . \; x \in \mathsf{Succ}_{s^k\mathsf{first}(s)}(I)?$$

(2) The function $Exit(Z, s, e)$ that for an initial set Z, a loop s, and an edge e (not in this loop) finds all the points on e reachable by making s several times and then exiting to e. Formally, it computes

$$\mathsf{Succ}_{s^+e}(I) = \bigcup_{k \geq 1} \mathsf{Succ}_{s^k e}(I),$$

which is always a finite union of intervals.

Since we know how to calculate the successor of a given interval in one and in several steps ($\mathsf{Succ}_{ee'}(\cdot)$ and $\mathsf{Succ}_r(\cdot)$), in order to implement $Test(\cdot)$ and $Exit(\cdot)$ it remains to show how to analyze the (simple) cycles s_i and eventually their continuation. Both algorithms $Test(\cdot)$ and $Exit(\cdot)$ start by doing qualitative analysis of the cycle (see next subsections for a detailed description of these algorithms). This analysis proceeds as follows.

Let s be a simple cycle, $f = \mathsf{first}(s)$ its first edge, and $I = \langle l, u \rangle \subset f$ an initial interval and $\mathsf{Succ}_{sf}(x) = F_{sf}(\{x\} \cap S) \cap J$. Notice that the successor can be iterated (applied again) only if $\mathsf{Succ}_{sf}(I)$ intersects with $S \cap J$, and only from this intersection. In what follows $\langle L, U \rangle$ will denote $S \cap J$.

The first thing to do is to determine the qualitative behavior of the leftmost and rightmost trajectories of the interval endpoints in the cycle. This can be done without iterating Succ_{sf}. Indeed, by Lemma 5.10, we can compute the limits $(l^*, u^*) = \lim_{n\to\infty} F^n_{sf}(\langle l, u \rangle)$ (notice that those are limits only for the *non-truncated operator* F), not taking into account that the edges are possible bounded (we use Lemma 5.11) and compare these limit points corresponding to unrestricted dynamics with L and U. There are five possibilities:

1. STAY The cycle is not abandoned by any of the two trajectories: $L \leq l^* \leq u^* \leq U$;
2. DIE The right trajectory exits the cycle through the left (consequently the left one also exits) or the left trajectory exits the cycle through the right (consequently the right one also exits). In symbols, $u^* < L \vee l^* > U$, see Fig. 20;
3. EXIT-BOTH Both trajectories exit the cycle (the left one through the left and the right one through the right): $l^* < L \wedge u^* > U$, see Fig. 21;
4. EXIT-LEFT The leftmost trajectory exits the cycle but not the other: $l^* < L \leq u^* \leq U$, see Fig. 22.
5. EXIT-RIGHT The rightmost trajectory exits the cycle but not the other: $L \leq l^* \leq U < u^*$.

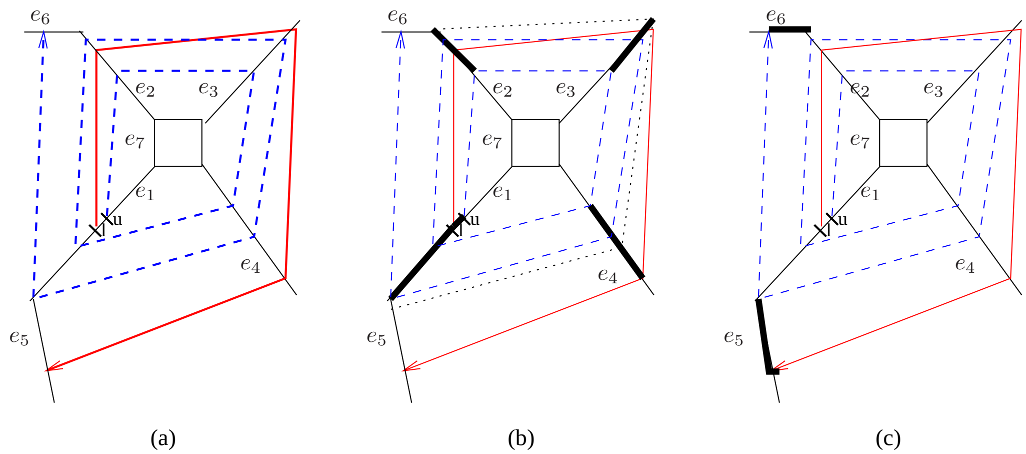


Fig. 20. [DIE] (a) Both trajectories leave the cycle $(e_1, e_2, e_3, e_4)^*$ through the left; (b) Reachable points on the cycle (in bold); (c) Possible continuation after leaving the cycle (in bold).

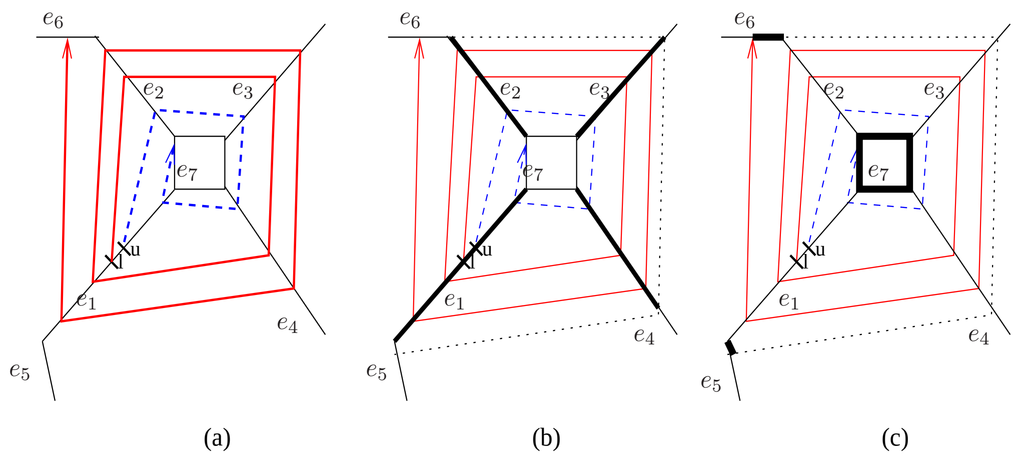


Fig. 21. [EXIT-BOTH]] (a) Both trajectories leave the cycle $(e_1, e_2, e_3, e_4)^*$; (b) Reachable points on the cycle (in bold); (c) Possible continuation after leaving the cycle (in bold).

This qualitative analysis is implemented in the function $Analyze(I, s)$ which returns the kind of qualitative behavior of the interval $I = \langle l, u \rangle$ under the loop s. See Fig. 23.

Notice that one (or both) of the successor functions can be the identity. In this case we have an infinite number of fixpoints but the analysis above continues to apply.

7.1.1 Exit

In this section we describe the EXIT algorithm (see Fig. 24) and show its soundness and termination. The *exit set* on a given edge e_x after cycling on s,

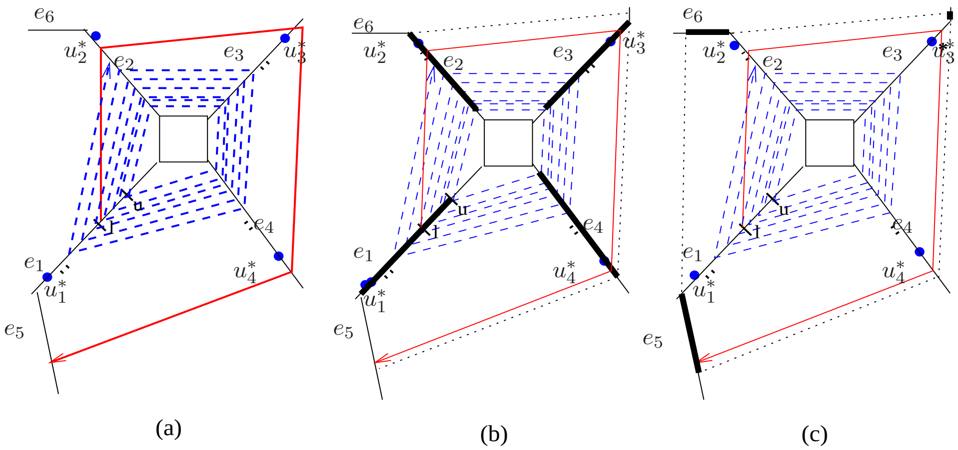


Fig. 22. [EXIT-LEFT] (a) The left trajectory leave the cycle $(e_1, e_2, e_3, e_4)^*$ through the left, whereas the right one tends to the limit u^*; (b) Reachable points on the cycle (in bold); (c) Possible continuation after leaving the cycle (in bold).

function *Analyze*(I, s)

  cases

  $L \leq l^* \leq u^* \leq U$: $\longleftarrow$ STAY

  $u^* < L \vee l^* > U$: $\longleftarrow$ DIE

  $l^* < L \wedge u^* > U$: $\longleftarrow$ EXIT-BOTH

  $L \leq l^* \leq U < u^*$: $\longleftarrow$ EXIT-RIGHT

  $l^* < L \leq u^* \leq U$: $\longleftarrow$ EXIT-LEFT

  endcases

Fig. 23. *Analyze* function.

for a given initial interval I, is

$$Ex = \bigcup_{m>0} \mathsf{Succ}_{se_x} \circ \mathsf{Succ}_{sf}^{m}(I).$$

The function $Exit(Z, s, e_x)$ should return $\mathsf{Succ}_{s+e_x}(Z)$. Both the argument Z and the result are finite collections of intervals. The exploration is made for each initial interval separately.

Notice that the call $\mathsf{Succ}_{sf}(I)$ ensures that $I \subseteq \langle L, U \rangle$. Preliminary analysis for each initial interval I is done by the function $Analyze(I, s)$ returning the kind of behavior k. After that, according to the result of this analysis, $Exit_k(I, s, e_x)$, that is one of five specialized procedures $Exit_{STAY}$, $Exit_{LEFT}$, $Exit_{RIGHT}$, $Exit_{BOTH}$, $Exit_{DIE}$, is launched and calculates the exit set. These specialized algorithms are presented in Fig. 25 (we only omit $Exit_{RIGHT}$ which is symmetrical to $Exit_{LEFT}$). Their termination and soundness will be established in the appendix. This will imply termination and soundness of the $Exit$ function itself.

$$
\begin{array}{l}
\textbf{function } \ Exit(Z, s, e_x) \\
\qquad\qquad E = \emptyset \\
\qquad\qquad \textbf{for each } \ I \in Z \\
\qquad\qquad\qquad\qquad \textbf{if } \ \mathsf{Succ}_{sf}(I) \cap S \neq \emptyset \\
\qquad\qquad\qquad\qquad \textbf{then } k = Analyze(I, s) \\
\qquad\qquad\qquad\qquad\qquad E = E \cup Exit_k(\mathsf{Succ}_{sf}(I) \cap S, s, e_x) \\
\qquad\qquad\qquad\qquad \textbf{else } E = E \cup \mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(I)) \\
\qquad\qquad \longleftarrow \text{E}
\end{array}
$$

Fig. 24. *Exit* function.

$$
\begin{array}{l}
\textbf{function } \ Exit_{STAY}(I, s, e_x) \\
\qquad\qquad \longleftarrow \emptyset \\
\\
\textbf{function } \ Exit_{DIE}(I, s, e_x) \\
\qquad\qquad Z = \emptyset \\
\qquad\qquad \textbf{repeat} \\
\qquad\qquad\qquad\qquad I = \mathsf{Succ}_{sf}(I) \\
\qquad\qquad\qquad\qquad Z = Z \bigcup \mathsf{Succ}_{se_x}(I) \\
\qquad\qquad \textbf{until } I = \emptyset \\
\qquad\qquad \longleftarrow \text{Z} \\
\\
\textbf{function } Exit_{BOTH}(I, s, e_x) \\
\qquad\qquad \longleftarrow \mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(\langle L, U \rangle)) \\
\\
\textbf{function } \ Exit_{LEFT}(I, s, e_x) \\
\qquad\qquad \longleftarrow \mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(\langle L, \max\{u, u^*\} \rangle))
\end{array}
$$

Fig. 25. Specialized *Exit* functions.

7.1.2 Test

In this section we describe the Test function and show its soundness and termination. In what follows, $l \uparrow$ means that the sequence $l, l_1, l_2, \ldots$ of successive successors of l is increasing whereas $l \downarrow$ means that the sequence is decreasing. Similarly for $u \uparrow$ and $u \downarrow$. Notice that detecting whether the sequences l_n and u_n are increasing or decreasing can be easily done at the stage of the preliminary analysis of the loop. The algorithm is shown in Fig. 26.

The upper-level structure is the same as for EXIT: each initial interval is treated separately, first by *Analyze* which detects the kind k of the loop and next by $Test_k$, which delegates all the remaining to one of the five specialized functions $Test_{STAY}$, $Test_{LEFT}$, $Test_{RIGHT}$, $Test_{BOTH}$, $Test_{DIE}$. The specialized $Test$ functions (except $Test_{RIGHT}$ symmetrical to $Test_{LEFT}$) are shown

```
function  Test(Z, s, x)
          for each  I ∈ Z such that Succ_sf(I) ∩ S ≠ ∅
                    k = Analize(I, s)
                    if Test_k(Succ_sf(I), s, x)
                          then ⟵ true
          ⟵ false
```

Fig. 26. $Test$ function.

```
function  Test_STAY(I, s, x)
          cases
               l* < x < u* :   ⟵ YES
               x ≤ l* ∧ l ↓ :  ⟵ NO
               x ≥ u* ∧ u ↑ :  ⟵ NO
               else :          ⟵ Search(I, x)
          endcases

function  Test_DIE(I, s, x)
          ⟵ Search(I, x)

function  Test_BOTH(I, s, x)
          ⟵ x ∈ Succ_sf(⟨L, U⟩)?

function  Test_LEFT(I, s, x)
          cases
               x ∈ Succ_sf(⟨L, u*⟩) :          ⟵ YES
               x < Succ_sf(⟨L, u*⟩) :          ⟵ NO
               Succ_sf(⟨L, u*⟩) < x ∧ u ↑ :    ⟵ NO
               else :                          ⟵ Search(I, x)
          endcases
```

Fig. 27. Specialized $Test$ functions.

in Fig. 27. Their soundness and correctness are stated in the appendix.

The five specialized $Test$ functions use the following two procedures (see Fig. 28): The function $Found(I, x)$ determines, if the current interval I contains x (YES), does not contain x and moves in the opposite direction (NO), or none of both these cases (NOTYET). The function $Search(I, x)$ iterates the loop s until the previous function $Found$ gives a definite answer YES or NO. Special measures will be taken to guarantee termination.

function $\mathit{Found}(I, x)$
cases
$x \in I :\quad \longleftarrow$ YES
$I = \emptyset :\quad \longleftarrow$ NO
$x < I \wedge l \uparrow :\quad \longleftarrow$ NO
$x > I \wedge u \downarrow :\quad \longleftarrow$ NO
else $:\quad \longleftarrow$ NOTYET
endcases

function $\mathit{Search}(I, x)$
while $\mathit{Found}(I, x) =$ NOTYET
$I = \mathsf{Succ}_{sf}(I)$
$\longleftarrow \mathit{Found}(I, x)$

Fig. 28. *Found* function.

7.2 Main result

Notice that the function $\mathit{Reach}_{type}(x_0, x_f, \tau)$ of the previous section computes $\mathit{Reach}_\tau(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)$ and hence the algorithm $\mathit{Reach}(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)$ computes the following:

$$\mathit{Reach}(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f) \equiv \exists \tau \in \mathcal{T}_P \,.\, \mathit{Reach}_\tau(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f).$$

From the previous section and the results of section 4 we have the following theorem.

Theorem 7.1 (Point–to–Point Reachability) *The algorithm above for deciding $\mathit{Reach}(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)$ is sound and complete. Hence point-to-point reachability is decidable for SPDI.*

PROOF. Soundness follows from the soundness of all the functions used in the algorithm that has already been proved. We have to prove that $\mathit{Reach}(\mathcal{H}, \mathbf{x}_0, \mathbf{x}_f)$ computes the good result for all the existing trajectory segments from $\mathbf{x}_0$ to $\mathbf{x}_f$, but this follows from Theorem 4.13 and the fact that all the types of feasible signatures are considered. $\square$

It is not difficult to see that the result also holds for edge–to–edge and region–to–region reachability.

Remark. Notice that the above decidability result holds for SPDIs under the goodness condition (see assumption 2.6). Non-good SPDIs can not be reduced to good SPDIs though we conjecture the reachability problem for non-good SPDIs is decidable.

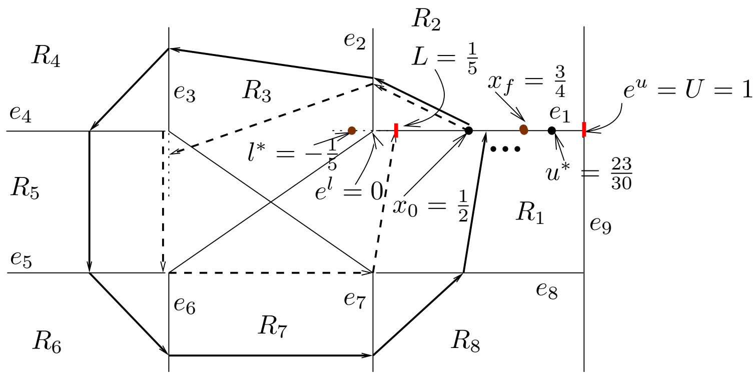


Fig. 29. $\mathbf{x}_f = (e_1, \frac{3}{4})$ is reachable from $\mathbf{x}_0 = (e_1, \frac{1}{2})$, i.e. $\frac{3}{4} \in \mathsf{Succ}_{e_1 e_8 \cdots e_1}(L, u^*)$.

7.3 Examples

In this section we present two examples of the application of the reachability algorithm for SPDIs.

Example 7.2 Consider again the swimmer of Figure 2 defined in section 2.2. Let $\mathbf{x}_0 = \left(e_1, \frac{1}{2}\right)$ be her initial position. We want to decide whether she is able to escape from the whirlpool and reach the final position $\mathbf{x}_f = (e_1, \frac{3}{4})$. Recall that $(L, U) = S \cap J = \left(\frac{1}{5}, 1\right)$, and

$$l^* = \left(-\frac{1}{20}\right) / \left(1 - \frac{1}{2}\right) = -\frac{1}{10}, \quad u^* = \left(\frac{23}{60}\right) / \left(1 - \frac{1}{2}\right) = \frac{23}{30}.$$

Thus, by the *Analyze* function we know that the cycle behaves as an Exit-LEFT and applying the function $Test_{LEFT}$ we obtain that $\mathbf{x}_f = (e_1, \frac{3}{4})$ is reachable from $\mathbf{x}_0 = (e_1, \frac{1}{2})$ because we have that

$$\mathsf{Succ}_{e_1 e_8 \cdots e_1}((L, u^*)) = \mathsf{Succ}_{e_1 e_8 \cdots e_1}\left(\left(\frac{1}{5}, \frac{23}{30}\right)\right) = \left(\frac{1}{20}, \frac{23}{30}\right),$$

and

$$\frac{3}{4} \in \left(\frac{1}{20}, \frac{23}{30}\right).$$

See Figure 29. ■

Example 7.3 Let us change the above example in order to show another behavior. For simplicity we consider the same partition as in the swimmer example but with the following differential inclusion dynamics:

- $R_1 : \mathbf{a} = \left(1, \frac{10}{3}\right), \mathbf{b} = (1, 5)$;
- $R_2 : \mathbf{a} = \mathbf{b} = (-1, 1)$;
- $R_3 : \mathbf{a} = \mathbf{b} = (-1, 0)$;
- $R_4 : \mathbf{a} = \mathbf{b} = (-1, -1)$;
- $R_5 : \mathbf{a} = \mathbf{b} = (0, -1)$;
- $R_6 : \mathbf{a} = \mathbf{b} = (1, -1)$;
- $R_7 : \mathbf{a} = \mathbf{b} = (1, 0)$;
- $R_8 : \mathbf{a} = \mathbf{b} = (1, 1)$.

We are interested in the edge signature $e_0(e_1 \ldots e_8)^* e_9$, and what matters for computing the reachable points of e_9 starting from $x_0 \in e_0$ are the following edge-to-edge successor functions:

$$\mathsf{Succ}_{e_0 e_1}(x) = \begin{cases} \left[\frac{1}{5}x, \frac{3}{10}x\right] \cap (0,1) \text{ if } x \in (0,1) \\ \emptyset \qquad\qquad\qquad \text{otherwise;} \end{cases}$$

$$\mathsf{Succ}_{e_i e_{i+1}}(x) = \begin{cases} \{x\} \cap (0,1) \text{ if } x \in (0,1) \\ \emptyset \qquad\qquad \text{otherwise;} \end{cases}$$

$$\mathsf{Succ}_{e_8 e_1}(x) = \begin{cases} \left[x + \frac{1}{5}, x + \frac{3}{10}\right] \cap (0,1) \text{ if } x \in (0, \frac{4}{5}) \\ \emptyset \qquad\qquad\qquad\qquad \text{otherwise;} \end{cases}$$

$$\mathsf{Succ}_{e_8 e_9}(x) = \begin{cases} \left[5x - 4, \frac{10}{3}x - \frac{7}{3}\right] \cap (0,1) \text{ if } x \in (\frac{7}{10}, 1) \\ \emptyset \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

Let x_0 be equal to $\frac{1}{2}$ on edge e_0 and x_f be $\frac{3}{10}$ on e_9; deciding whether exists a trajectory from $(e_0, \frac{1}{2})$ to $(e_9, \frac{3}{10})$ can be done following the steps:

(1) Compute the "enter interval" to the loop: $\mathsf{Succ}_{e_0 e_1}(\frac{1}{2}) = \left[\frac{1}{10}, \frac{3}{20}\right]$.
(2) Compute the successor function of the loop $(e_1 \ldots e_8)^*$ [3]:

$$\mathsf{Succ}_{e_1 \ldots e_8 e_1}(x) = \begin{cases} \left[x + \frac{1}{5}, x + \frac{3}{10}\right] \cap (\frac{1}{5}, 1) \text{ if } x \in (0, \frac{4}{5}) \\ \emptyset \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

(3) Compute the limits of the loop signature: By Lemma 5.10 we have that $u^* = l^* = \infty$ for both affine functions. We can then conclude that the trajectories will be counterclockwise expanding spirals and the *Analyze* function gives that the loop will behave as a DIE (see section 7.1).
(4) Execute the function $Exit_{DIE}(\left[\frac{1}{10}, \frac{3}{20}\right], e_1 \ldots e_8, e_9) = \{[0,1]\}$.

The execution trace is given in Table 1, where in the Z column we can see the set of (truncated) exit intervals over the edge e_9: in the third iteration the exit interval is the whole edge e_9. From the above we conclude that $(e_9, \frac{3}{10})$ is reachable from $(e_0, \frac{1}{2})$.

As an example of a non reachable point, consider the edge signature $(e_1 \ldots e_8)^*$ with $[\frac{9}{10}, \frac{19}{20}] \in e_1$ as initial interval and $x_f = \frac{3}{10}$ in e_9 as before. After computing

[3] Notice that in fact this function is the same as $\mathsf{Succ}_{e_8 e_1}$ since the other functions are the identity.

Iteration	I	Z
0	$[\frac{1}{10}, \frac{3}{20}]$	$\emptyset$
1	$[\frac{3}{10}, \frac{9}{20}]$	$\emptyset$
2	$[\frac{1}{2}, \frac{3}{4}]$	$\{[0, \frac{1}{6}]\}$
3	$[\frac{7}{10}, 1]$	$\{[0, 1]\}$
4	$[\frac{9}{10}, 1]$	$\{[0, 1]\}$

Table 1
Execution trace of the cycle $e_1 \ldots e_8$ starting from $[0.1, 0.15] \in e_1$. I represents the current interval (in e_1) and Z is the set of exit intervals (in e_9).

the corresponding functions we obtain that in the first iteration the loop is left and the exit interval on edge e_9 is $[\frac{1}{2}, \frac{5}{6}]$, from where we can conclude that $(e_9, \frac{3}{10})$ is not reachable from $(e_1, \left[\frac{9}{10}, \frac{19}{20}\right])$. ■

8 Conclusion

We have presented an algorithm for solving the reachability problem for polygonal differential inclusion systems. The novelty of the approach for the domain of hybrid systems is the combination of two techniques, namely, the representation of the two-dimensional continuous dynamics as a one-dimensional discrete system (due to Poincaré), and the characterization of the set of qualitative behaviors of the latter as a finite set of types of signatures. The enumeration of such a set is the base for proving decidability, which naturally gives a depth-first search algorithm. A breadth-first search algorithm has been given in [PS03].

An interesting issue is the complexity analysis of the algorithm. The algorithm is based on counting all "feasible" types of signatures; our finiteness argument (lemma 4.11) gives a doubly exponential estimation. In practice, the types of signatures are computed on-the-fly, and due to acceleration, the time for analyzing each type of signature is not significant. Moreover, by combining the space reduction techniques based on topological and geometrical optimizations recently presented in [PS06b] with the compositional parallel algorithm given in [PS06a], we envisage even greater gains in terms of space and time complexity.

Some other results on SPDIs and related systems have been given in the last years. SPDIs can be seen as non-deterministic piece-wise constant derivative systems, for which the reachability problem is decidable for two dimensional systems [MP93] and undecidable for three or higher dimensions [AM94]. The frontier between decidability and undecidability is not sharp. We can certainly find (stringent) conditions, such as planarity of the automaton, "memory-less" resets, etc., under which decidability follows almost straightforwardly from the decidability of SPDIs. On the other hand, it is not difficult to see that reachability for hybrid automata whose locations are equipped with SPDIs and similar classes of systems, which do not satisfy such conditions, is equivalent to deciding whether given a piece-wise linear map f on the unit interval and a point x in this interval, the sequence of iterates x, $f(x)$, $f(f(x))$, and so on, reaches some point y. This last question is still open [Koi]. The (un)decidability frontier has been studied in [AS02] and refined recently in [MP05]. Reachability of slight extensions of such classes turn out to be undecidable [AS02,MP05]. On another line of research, the qualitative behavior, i.e. the phase portrait, of SPDIs has been analyzed in [ASY02] and [Sch04]. The algorithm presented here has been implemented in a tool-kit called SPeeDI [APSY02] and recently extended to compute SPDIs phase portraits.

One open question on SPDIs is whether it could be possible to apply the same technique for solving the *parameter synthesis problem*, that is, for any two points, $\mathbf{x}_0$ and $\mathbf{x}_f$, assign a constant slope $\mathbf{c}_P \in \phi(P)$ to every region P such that $\mathbf{x}_f$ is reachable from $\mathbf{x}_0$, or conclude that such an assignment does not exist. Clearly, the decidability of the reachability problem does not imply the decidability of the parameter synthesis one.

Another question that naturally arises is decidability of the reachability problem for *General SPDIs*, i.e. SPDIs without goodness (assumption 2.6). We conjecture that the problem is indeed decidable. Preliminary works have shown, however, that if such a reachability algorithm exists it cannot be based on a reduction to the reachability of SPDIs; an extension of the technique presented here would be needed.

Though the class of SPDIs is rather simple from the modelling point of view, it is a rather complex one from the analysis point of view. Indeed, it is well known that even for slight extensions of this class of systems, reachability becomes undecidable, and adding jumps in 2-dim leads immediately to an "intermediate" complexity equivalent to a well-known open problem for which decidability analysis is difficult [AS02,MP05]. Moreover, SPDIs could be used for approximating complex non-linear differential equations on the plane, for which an exact solution is not known. The decidability of SPDI's reachability and of its phase portrait construction would be of invaluable help for the qualitative analysis of such equations. The challenge would be to find an "intelligent" partition of the plane in order to get an optimal approximation of the equations.

References

[AAB99] P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *TACAS*, volume 1579 of *LNCS*, pages 208–222, 1999.

[ABDM00] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In Lynch and Krogh [LK00], pages 20–31.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AGH+00] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specification of hybrid systems in CHARON. In Lynch and Krogh [LK00], pages 6–19.

[AHS96] R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III*, volume 1066 of *LNCS*, Rutgers University in New Brunswick, NJ, USA, October 1996. Springer.

[AKL+98] P.J. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors. *Hybrid Systems V*, volume 1567 of *LNCS*, Notre Dame, Indiana, USA, September 1998. Springer.

[AKNS95] P.J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems II*, volume 999 of *LNCS*, Ithaca, NY, USA, October 1995. Springer.

[AKNS97] P.J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems IV*, volume 1273 of *LNCS*, Ithaca, NY, USA, October 1997. Springer.

[AM94] E. Asarin and O. Maler. On some relations between dynamical systems and transition systems. In S. Abiteboul and E. Shamir, editors, *ICALP'94*, number 820 in LNCS, pages 59–72. Springer, 1994.

[AMP95] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:35–65, 1995.

[APSY02] E. Asarin, G. Pace, G. Schneider, and S. Yovine. SPeeDI: a verification tool for polygonal hybrid systems. In *CAV'2002*, volume 2404 of *LNCS*, pages 354–358, Copenhagen, Denmark, July 2002. Springer-Verlag.

[AS02] E. Asarin and G. Schneider. Widening the boundary between decidable and undecidable hybrid systems. In *CONCUR'2002*, volume 2421 of

LNCS, pages 193–208, Brno, Czech Republic, August 2002. Springer-Verlag.

[ASY01] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In di Benedetto and Sangiovanni-Vincentelli [dBSV01], pages 89–104.

[ASY02] E. Asarin, G. Schneider, and S. Yovine. Towards computing phase portraits of polygonal differential inclusions. In Tomlin and Greenstreet [TG02], pages 49–61.

[BBM+00] A. Balluchi, L. Benvenuti, G.M. Miconi, U. Pozzi, T. Villa, M.D. Di Benedetto, H. Wong-Toi, and A.L. Sangiovanni-Vincentelli. Maximal safe set computation for idle speed control of an automotive engine. In Lynch and Krogh [LK00], pages 32–44.

[BGWW97] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *Static Analysis Symposium*, volume 1302 of *LNCS*, pages 172–186. Springer, September 1997.

[BH97] A. Bouajjani and P. Habermehl. Symbolic Reachability Analysis of FIFO Channel Systems with Nonregular Sets of Configurations (extended abstract). In *Automata, Languages and Programming, 24th International Colloquium*, volume 1256 of *LNCS*, pages 560–570. Springer-Verlag, July 1997.

[BHJ03] B. Boigelot, F. Herbreteau, and S. Jodogne. Hybrid acceleration using real vector automata. In *CAV*, volume 2725 of *LNCS*, pages 193–205. Springer, 2003.

[BKS00] N. Bauer, S. Kowalewski, and G. Sand. A case study: Multi product batch plant for the demonstration of control and scheduling problems. In *ADPM*, pages 969–974, Dortmund, Germany, 2000.

[Bro99] M. Broucke. A geometric approach to bisimulation and verification of hybrid systems. In Vaandrager and van Schuppen [VvS99], pages 61–75.

[BT00] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In Lynch and Krogh [LK00], pages 73–88.

[BW94] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proceedings of the 6th International Conference on Computer Aided Verification*, volume 818 of *LNCS*, pages 55–67, 1994.

[CK98] A. Chutinan and B.H. Krogh. Computing polyhedral approximations to dynamic flow pipes. In *Proc. of the 37th Annual International Conference on Decision and Control, CDC'98*. IEEE, 1998.

[Dan00] T. Dang. d/dt manual. Technical report, Verimag, Grenoble, 2000.

[dBSV01] M.D. di Benedetto and A. Sangiovanni-Vincentelli, editors. *Hybrid Systems: Computation and Control*, volume 2034 of *LNCS*, Rome, Italy, March 2001. Springer.

[DM98] T. Dang and O. Maler. Reachability analysis via face lifting. In *HSCC'98*, number 1386 in LNCS, pages 96–109. Springer Verlag, 1998.

[DY01] J. Della Dora and S. Yovine. Looking for a methodology for analyzing hybrid systems. In *European Control Conference*, Porto, Portugal, September 2001.

[FvS99] J.J.H. Fey and J.H. van Schuppen. VHS case study 4 - modeling and control of a juice processing plant. `http://www-verimag.imag.fr/VHS/CS4/dcs42.ps.gz`, 1999.

[GJ94] J. Guckenheimer and S. Johnson. Planar hybrid systems. In *Hybrid Systems and Autonomous Control Workshop*, pages 202–225, 1994.

[GM99] M. R. Greenstreet and I. Mitchell. Reachability analysis using polygonal projections. In Vaandrager and van Schuppen [VvS99], pages 103–116.

[GNRR93] R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *LNCS*. Springer-Verlag, 1993.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *27th Annual Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.

[HPHHt97] T.A. Henzinger, P.-H.Ho, and H.Wong-toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1):110–122, 1997.

[HS74] M.W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems and Linear Algebra*. Academic Press Inc., 1974.

[Koi] P. Koiran. My favourite problems. `http://www.ens-lyon.fr/~koiran/problems.html`.

[KV00] A.B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In Lynch and Krogh [LK00], pages 202–214.

[Lay82] S.R. Lay. *Convex sets and their applications*. John Wiley and Sons, New York, 1982.

[LK00] N. Lynch and B.H. Krogh, editors. *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*. Springer-Verlag, 2000.

[LPY01] G. Lafferriere, G. Pappas, and S. Yovine. Symbolic reachability computation of families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, September 2001.

[MP93] O. Maler and A. Pnueli. Reachability analysis of planar multi-linear systems. In C. Courcoubetis, editor, *CAV'93*, number 697 in LNCS, pages 194–209. Springer-Verlag, 1993.

[MP05] V. Mysore and A. Pnueli. Refining the undecidability frontier of hybrid automata. In *FSTTCS*, volume 3821 of *LNCS*, pages 261–272. Springer-Verlag, 2005.

[NS60] V.V. Nemytskii and V.V. Stepanov. *Qualitative theory of differential equations.* Princeton University Press, 1960.

[PAT] PATH Project. `http://paleale.eecs.berkeley.edu/`.

[PS03] G. Pace and G. Schneider. Model checking polygonal differential inclusions using invariance kernels. In *VMCAI'04*, number 2937 in LNCS, pages 110–121, Venice, Italy, December 2003. Springer Verlag.

[PS06a] G. Pace and G. Schneider. A compositional algorithm for parallel model checking of polygonal hybrid systems. In *ICTAC 2006*, volume 4281 of *LNCS*, pages 168–182. Springer-Verlag, 2006.

[PS06b] G. Pace and G. Schneider. Static analysis for state-space reduction of polygonal hybrid systems. In *FORMATS'06*, volume 4202 of *LNCS*, pages 306–321. Springer-Verlag, 2006.

[PVB96] A. Puri, P. Varaiya, and V. Borkar. Epsilon approximations of differential inclusions. In Alur et al. [AHS96], pages 362–376.

[Sch04] G. Schneider. Computing invariance kernels of polygonal hybrid systems. *Nordic Journal of Computing*, 11(2):194–210, 2004.

[TG02] C.J. Tomlin and M.R. Greenstreet, editors. *Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, Stanford, CA, USA, March 2002. Springer.

[TLS98] C. Tomlin, J. Lygeros, and S. Sastry. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.

[uV96] K. Čerāns and J. Vīksna. Deciding reachability for planar multi-polynomial systems. In Alur et al. [AHS96], pages 389–400.

[VvS99] F.W. Vaandrager and J.H. van Schuppen, editors. *Hybrid Systems : Computation and Control*, volume 1569 of *LNCS*, Berg en Dal, The Netherlands, March 1999. Springer-Verlag.

A Affine Operators (properties)

We will prove here the lemmas introduced in Section 5 as well as other interesting properties of iterations of affine operations.

To start with, we prove that to obtain the inverse of a truncated affine multi-valued function $\mathcal{F}$ we just need to inverse the corresponding non-truncated affine function and truncate it with the domain and co-domain of $\mathcal{F}$ interchanged.

Lemma A.1 (5.5) *Given a $\mathcal{F}(I) = F(I \cap S) \cap J$, then $\mathcal{F}^{-1}(I) = F^{-1}(I \cap J) \cap S$.*

PROOF. We prove first that $\mathcal{F}^{-1}(I) \subseteq F^{-1}(I \cap J) \cap S$. Let $y \in \mathcal{F}^{-1}(I)$, then it exists $x \in I$ such that $x \in \mathcal{F}(y) = F(\{y\} \cap S) \cap J$. It is immediate that $y \in S$ and $x \in J$, and hence $x \in I \cap J$. We deduce that $y \in F^{-1}(I \cap J)$, and conclude that $y \in F^{-1}(I \cap J) \cap S$.
For the other inclusion, given $y \in S$ and $y \in F^{-1}(I \cap J)$ we prove now that $y \in \mathcal{F}^{-1}(x)$. Indeed, it exists $x \in I \cap J$ such that $x \in F(y)$. We have then that $x \in J$, $x \in F(y)$ and $y \in S$, and hence $x \in \mathcal{F}(y) = F(\{y\} \cap S) \cap J$. We conclude that $y \in \mathcal{F}^{-1}(I)$. $\square$

Lemma A.2 (5.7) *Every TAMF $\mathcal{F}$ can be represented in normal form.*

PROOF. Let $\mathcal{F}(I) = F(I \cap S) \cap J$ be a TAMF. We show that there exists a TAMF $\mathcal{F}'(I) = F'(I \cap S') \cap J'$ such that $\mathcal{F} = \mathcal{F}'$ and $\mathcal{F}'$ is in normal form. Let $\mathcal{F}'$ be the above function with $F' = F$, $S' = S \cap F^{-1}(J)$ and $J' = \mathcal{F}(S)$. Clearly $S' = \mathsf{Dom}(\mathcal{F}')$ and $J' = \mathsf{Im}(\mathcal{F}')$. It remains to show that $\mathcal{F} = \mathcal{F}'$.

If $x \notin S$ or $x \notin F^{-1}(J)$, then the result follows, since $\mathcal{F}(x) = \emptyset$ and $\mathcal{F}'(x) = \emptyset$. Suppose now that $x \in S$ and $x \in F^{-1}(J)$. Hence $x \in S'$ and $\mathcal{F}(x) = F(x) \cap J$ and $\mathcal{F}'(x) = F(x) \cap J' = F(x) \cap F(S) \cap J = F(x) \cap J$. The two maps $\mathcal{F}$ and $\mathcal{F}'$ are identical for all x. $\square$

We will use in the sequel the one-dimensional case of a classical result from convex geometry:

Theorem A.3 (Helly, see [Lay82]) *If intervals $I_1, \ldots, I_k \subseteq \mathbb{R}$ intersect pairwise:*

$$\forall i, j : I_i \cap I_j \neq \emptyset,$$

then they have a common point.

Before showing that the class of functions above defined are closed under composition we prove the following lemma.

Lemma A.4 *Let F be a multi-valued affine operator. If $I \cap H \neq \emptyset$ or $I = \emptyset$ or $H = \emptyset$ then $F(I \cap H) = F(I) \cap F(H)$.*

PROOF. Clearly if $I = \emptyset$ or $H = \emptyset$ then $F(I \cap H) = \emptyset = F(I) \cap F(H)$. Suppose now that $I \cap H \neq \emptyset$. The inclusion $F(I \cap H) \subseteq F(I) \cap F(H)$ is trivial. To prove the other direction, suppose that $x \in F(I)$ and $x \in F(H)$. Then $F^{-1}(x) \cap I \neq \emptyset$, and $F^{-1}(x) \cap H \neq \emptyset$, and, by hypothesis $I \cap H \neq \emptyset$. In other words, the three intervals $F^{-1}(x)$, I and H intersect pairwise, and hence, by Helly's theorem they have a common point y. Immediately we have $x \in F(y) \subseteq F(I \cap H)$. $\square$

Now we can prove the closure under composition for the three classes of functions introduced before.

Lemma A.5 (5.8, composition of affine operations) *Affine functions, affine multi-valued operators, and truncated affine multi-valued operators are closed under composition.*

PROOF.

Affine functions: For $f(x) = ax + b$ and $g(x) = cx + d$ the composition $g \circ f(x) = c(ax + b) + d = (ca)x + (cb + d)$ has the required form. Notice, that the coefficient ca is positive since c and a are positive.

Affine multi-valued operators For $F = \langle f_l, f_u \rangle$ and $H = \langle h_l, h_u \rangle$, the composition $H \circ F$ is nothing other than $\langle h_l \circ f_l, h_u \circ f_u \rangle$.

Truncated affine multi-valued operators For

$$\mathcal{F}_1(x) = F_1(\{x\} \cap S_1) \cap J_1, \quad \mathcal{F}_2(x) = F_2(\{x\} \cap S_2) \cap J_2$$

we will establish that $\mathcal{F}_2 \circ \mathcal{F}_1(x) = \mathcal{F}_{F',S',J'}(x)$ with $F' = F_2 \circ F_1$, $J' = J_2 \cap F_2(J_1 \cap S_2)$ and $S' = S_1 \cap F_1^{-1}(J_1 \cap S_2)$.

Indeed, by definition of $\mathcal{F}_1$ and $\mathcal{F}_2$

$$\begin{aligned} \mathcal{F}_2 \circ \mathcal{F}_1(x) &= \mathcal{F}_2(F_1(\{x\} \cap S_1) \cap J_1) \\ &= F_2((F_1(\{x\} \cap S_1) \cap J_1) \cap S_2) \cap J_2. \end{aligned} \tag{A.1}$$

We split the proof into two cases:

(1) $x \in S'$, that reduces, using the formula for S', to two conditions: $x \in S_1$ and $F_1(x) \cap (J_1 \cap S_2) \neq \emptyset$. In this case $F_1(\{x\} \cap S_1) = F_1(x)$ and then

expression (A.1) is equal to $F_2((F_1(x) \cap J_1) \cap S_2) \cap J_2$ that is equal to

$$F_2(F_1(x) \cap (J_1 \cap S_2)) \cap J_2. \tag{A.2}$$

In this case the distributivity holds (see Lemma A.4) and expression (A.2) is equal to $F_2(F_1(x)) \cap F_2(J_1 \cap S_2) \cap J_2$, and hence to $\mathcal{F}_{F',S',J'}(x)$.

(2) $x \notin S'$ which splits into two subcases: $x \notin S_1$ or $F_1(x) \cap (J_1 \cap S_2) = \emptyset$. In both cases it is easy to see that $\mathcal{F}_2 \circ \mathcal{F}_1(x) = \emptyset$. This also matches with $\mathcal{F}_{F',S',J'}(x)$. $\square$

We show next that normalization is preserved by composition.

Lemma A.6 *If $\mathcal{F}_1$ and $\mathcal{F}_2$ are normalized, then $\mathcal{F}_2 \circ \mathcal{F}_1$, represented as stated in Lemma A.5 is also normalized.*

PROOF. By Lemma A.5,

$$\mathcal{F}_2 \circ \mathcal{F}_1(x) = \mathcal{F}_{F',S',J'}(x)$$

with $F' = F_2 \circ F_1$, $J' = J_2 \cap F_2(J_1 \cap S_2)$ and $S' = S_1 \cap F_1^{-1}(J_1 \cap S_2)$.

We have to prove that $S' = \mathsf{Dom}(\mathcal{F}') = F'^{-1}(J') \cap S'$ and $J' = \mathsf{Im}(\mathcal{F}') = F'(S') \cap J'$. This is equivalent to $S' \subseteq F'^{-1}(J')$ and $J' \subseteq F'(S')$.

$\mathbf{J'} \subseteq \mathbf{F'}(\mathbf{S'})$ We have to prove that

$$J_2 \cap F_2(J_1 \cap S_2) \subseteq F_2(F_1(S_1 \cap F_1^{-1}(J_1 \cap S_2))).$$

Indeed suppose that $x \in J_2$ and $x \in F_2(J_1 \cap S_2)$. Then $x \in F_2(y)$ for some $y \in J_1 \cap S_2$. By normalization of $\mathcal{F}_1$, for this y there exists a $z \in S_1$, such that $y \in F_1(z)$. Clearly this $z \in F_1^{-1}(y) \subseteq F_1^{-1}(J_1 \cap S_2)$. We have thus:

$$\begin{aligned} &z \in S_1 \cap F_1^{-1}(J_1 \cap S_2) \\ &y \in F_1(S_1 \cap F_1^{-1}(J_1 \cap S_2)) \\ &x \in F_2(F_1(S_1 \cap F_1^{-1}(J_1 \cap S_2))), \end{aligned}$$

which concludes the proof of the first inclusion.

$\mathbf{S'} \subseteq \mathbf{F'^{-1}}(\mathbf{J'})$ We have to prove that

$$S_1 \cap F_1^{-1}(S_2 \cap J_1) \subseteq [F_2 \circ F_1]^{-1}(J_2 \cap F_2(J_1 \cap S_2)).$$

Suppose that $x \in S_1$ and $x \in F_1^{-1}(S_2 \cap J_1)$, i.e.

$$F_1(x) \cap (S_2 \cap J_1) \neq \emptyset,$$

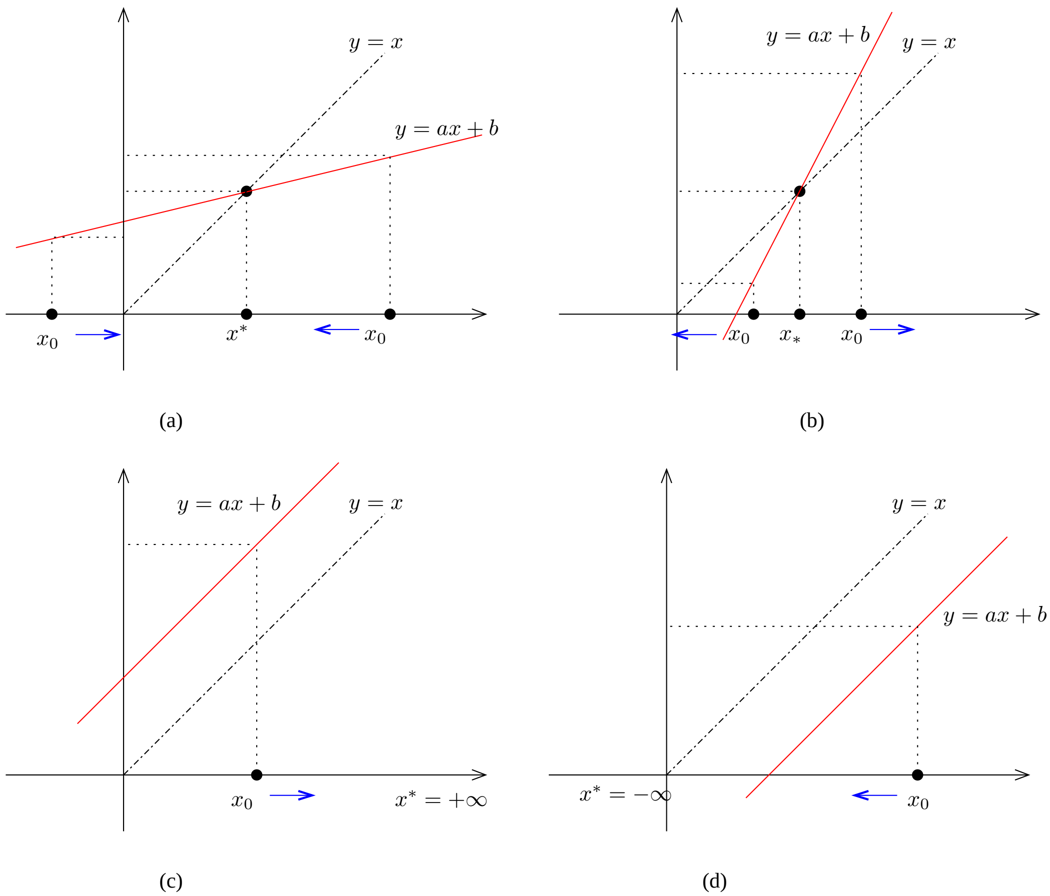


Fig. A.1. (a): $a < 1$, $x^* = b/(1-a)$; (b): $a > 1$, $x^* = -\infty$ if $x_0 < x_*$,$x^* = x_0$ if $x_0 = x_*$, $x^* = +\infty$ if $x_0 > x_*$; (c): $a = 1$ and $b > 0$, $x^* = +\infty$; (d): $a = 1$ and $b < 0$, $x^* = -\infty$

then there exists some $y \in F_1(x) \cap (S_2 \cap J_1)$, and by normalization of $\mathcal{F}_2$ we have that $F_2(y) \cap J_2 \neq \emptyset$, hence

$$F_2(F_1(x) \cap (S_2 \cap J_1)) \cap J_2 \neq \emptyset.$$

By Lemma A.4 we have that

$$F_2(F_1(x)) \cap F_2(S_2 \cap J_1) \cap J_2 \neq \emptyset.$$

Hence

$$x \in [F_2 \circ F_1]^{-1}(J_2 \cap F_2(J_1 \cap S_2)). \qquad \square$$

The following result shows how to compute fixpoints of affine functions [MP93].

Lemma A.7 *Let f be an affine function, x_0 be any initial point and $x_n = f^n(x_0)$. The following properties hold*

(1) The sequence x_n is monotonous;

(2) It converges to a limit x^* (finite or infinite), which can be effectively computed knowing a, b and x_0.

PROOF. Monotonicity of x_n follows from the identity $x_{n+1} - x_n = a^n(x_1 - x_0)$:

$$x_n = f^n(x_0) \Longrightarrow f^n(x_0) = a^n x_0 + a^{n-1}b + \ldots + ab + b \Longrightarrow$$
$$\begin{aligned} x_{n+1} - x_n &= (a^{n+1}x_0 + a^n b + \ldots + ab + b) - (a^n x_0 + a^{n-1}b + \ldots + ab + b) \\ &= a^n(ax_0 + b - x_0) = a^n(x_1 - x_0) \end{aligned}$$

Existence of limit is immediate from the monotonicity. To calculate the limit several cases should be considered (see Fig. A.1):

$a < 1$: In this case the limit is finite and it is the unique fixpoint of the function f: $ax^* + b = x^*$, and hence $x^* = b/(1-a)$.

$a = 1$: In this case

$$x^* = \begin{cases} -\infty \text{ if } b < 0 \\ x_0 \text{ if } b = 0 \\ \infty \text{ if } b > 0 \end{cases}$$

$a > 1$: In this case we should calculate first the (unstable) fixpoint $x_* = b/(1-a)$. However in this case the limit is not necessary equal to x_* . Namely,

$$x^* = \begin{cases} -\infty \text{ if } x_0 < x_* \\ x_0 \text{ if } x_0 = x_* \\ \infty \text{ if } x_0 > x_* \end{cases} \qquad \square$$

This result can be easily extended to intervals and affine multi-valued operators.

Lemma A.8 (5.10) *Let $\langle l_0, u_0\rangle$ be any initial interval and $\langle l_n, u_n\rangle = F^n(\langle l_0, u_0\rangle)$. 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.

PROOF. Direct consequence of Lemma A.7 considering l_n and u_n. $\square$

The following result is a direct consequence of monotonicity of l_n and u_n.

Lemma A.9 (Convexity) *Let F be an affine multi-valued operator.*

(1) If $H \cap I \neq \emptyset$, and $H \cap F^n(I) \neq \emptyset$, then for all $k \in 0..n$ also $H \cap F^k(I) \neq \emptyset$.
(2) If $x \in I$, and $x \in F^n(I)$, then for all $k \in 0..n$ also $x \in F^k(I)$.

PROOF. We will use the following evident fact: for non-empty intervals $[a, b] \cap [c, d] \neq \emptyset$ if and only if $a \leq d$ and $b \geq c$.

(1) W.l.o.g. we suppose that I and H are closed intervals. Let $F^k(I) = [l_k, u_k]$, and $H = [L, U]$. Sequences l_k and u_k are monotonous (increasing or decreasing) due to Lemma A.8. From nonemptyness hypotheses $U \geq l_0$ and $U \geq l_n$, and by monotonicity also $U \geq l_k$ for all intermediate values of k. Similarly $L \leq u_k$, and hence $H \cap F^k(I) \neq \emptyset$.
(2) Apply the previous statement with $H = [x, x]$. □

Our next aim is to prove the Fundamental Lemma (Lemma 5.11) and a result (Lemma A.11) allowing to compute iterations of arbitrary TAMFs.

Lemma A.10 (5.11, Fundamental lemma) *Let $\widehat{\mathcal{F}}$ be a truncated affine multi-valued operator of the form $\widehat{\mathcal{F}}(I) = F(I \cap H) \cap H$. Then $\widehat{\mathcal{F}}^n(I) = F^n(I \cap H) \cap H$.*

PROOF.

Base case ($n = 1$): By definition $\widehat{\mathcal{F}}(I) = F(I \cap H) \cap H$.
Inductive step (from $n \geq 1$ to $n + 1$): Applying inductive hypothesis we have that

$$\begin{aligned} \widehat{\mathcal{F}}^{n+1}(I) &= \widehat{\mathcal{F}}(\widehat{\mathcal{F}}^n(I)) \\ &= \widehat{\mathcal{F}}(F^n(I \cap H) \ \cap \ H) && \text{(By inductive hypothesis)} \\ &= F(F^n(I \cap H) \ \cap \ H) \ \cap \ H && \text{(By definition of } \widehat{\mathcal{F}}\text{)} \end{aligned} \tag{A.3}$$

In order to prove the required

$$\widehat{\mathcal{F}}^{n+1}(I) = F^{n+1}(I \cap H) \cap H \tag{A.4}$$

we will establish two inclusions between the two expressions.
(1) $\subseteq$: This inclusion is easy, removing one intersection can only augment the set:

$$\widehat{\mathcal{F}}^{n+1}(I) = F(F^n(I \cap H) \cap H) \cap H \subseteq F(F^n(I \cap H)) \cap H = F^{n+1}(I \cap H) \cap H.$$

(2) $\supseteq$: This direction is more involved. Suppose that x belongs to the right-hand side of (A.4), that is $x \in F(F^n(I \cap H)) \cap H$. We have to deduce that it also belongs to the left-hand side. We notice the following three facts:

(a) Since $x \in H$ and $x \in F^{n+1}(H)$, by Convexity Lemma A.9 we have that $x \in F(H)$. We prefer to write it down as

$$F^{-1}(x) \cap H \neq \emptyset. \tag{A.5}$$

(b) Since $x \in F(F^n(I \cap H))$, then

$$F^{-1}(x) \cap F^n(I \cap H) \neq \emptyset. \tag{A.6}$$

(c) Notice that $H \cap I \cap H = I \cap H \neq \emptyset$ (otherwise x would not exist), and also $H \cap F^{n+1}(I \cap H) \neq \emptyset$ since it contains x. Then by the interval Convexity Lemma A.9 we have that

$$H \cap F^n(I \cap H) \neq \emptyset. \tag{A.7}$$

Equations (A.5-A.7) and Helly's Theorem A.3 guarantee that the three intervals $F^n(I \cap H)$, H, and $F^{-1}(x)$ have a common point z. Immediately $z \in F^n(I \cap H) \cap H$ and $x \in F(z)$. Hence $x \in F(F^n(I \cap H) \cap H)$, which together with the hypothesis $x \in H$ gives the required:

$$x \in F(F^n(I \cap H) \cap H) \cap H = \widehat{\mathcal{F}}^{n+1}(I). \qquad \square$$

Notice, that the Fundamental Lemma allows to compute the iteration of TAMFs of the special form $\widehat{\mathcal{F}}(I) = F(I \cap H) \cap H$. However the general case can be reduced to this special one. Indeed, for any TAMF $\mathcal{F}(I) = F(I \cap S) \cap J$ we can introduce $H = S \cap J$ and an auxiliary special form TAMF $\widehat{\mathcal{F}}(I) = F(I \cap H) \cap H$.

The following Lemma shows that in order to compute the iteration of $\mathcal{F}$ we need to apply it once at the beginning and once at the end and compose them with the iteration of $\widehat{\mathcal{F}}$ given by the Fundamental Lemma.

Lemma A.11 $\mathcal{F}^{n+2} = \mathcal{F} \circ \widehat{\mathcal{F}}^n \circ \mathcal{F}$.

PROOF. The following two identities can be proved by straightforward computation

$$\mathcal{F} \circ \mathcal{F} \circ \mathcal{F} = \mathcal{F} \circ \widehat{\mathcal{F}} \circ \mathcal{F} \tag{A.8}$$
$$\widehat{\mathcal{F}} \circ \mathcal{F} \circ \mathcal{F} = \widehat{\mathcal{F}} \circ \widehat{\mathcal{F}} \circ \mathcal{F} \tag{A.9}$$

For the first one:

$$\begin{aligned}
\mathcal{F}\circ\mathcal{F}\circ\mathcal{F}(I) &= F((F((F(I\cap S)\cap J)\cap S)\cap J)\cap S)\cap J = \\
&= F((F((F(I\cap S)\cap J)\cap (S\cap J))\cap (J\cap S))\cap S)\cap J = \\
&= F((F((F(I\cap S)\cap J)\cap H)\cap H)\cap S)\cap J = \\
&= \mathcal{F}\circ\widehat{\mathcal{F}}\circ\mathcal{F}(I).
\end{aligned}$$

The proof of the second identity is similar.

We can now prove the main statement by induction:

Base case ($n = 0$): Trivial.
Base case ($n = 1$): Immediate from (A.8).
Inductive step (from $n \geq 1$ to $n + 1$): Suppose $\mathcal{F}^{n+2} = \mathcal{F}\circ\widehat{\mathcal{F}}^n\circ\mathcal{F}$. Then applying (A.9) we can transform $\mathcal{F}^{n+3}$ to the required form:

$$\begin{aligned}
\mathcal{F}^{n+3} &= \mathcal{F}^{n+2}\circ\mathcal{F} = \mathcal{F}\circ\widehat{\mathcal{F}}^n\circ\mathcal{F}\circ\mathcal{F} = \mathcal{F}\circ\widehat{\mathcal{F}}^{n-1}\circ\widehat{\mathcal{F}}\circ\mathcal{F}\circ\mathcal{F} = \\
&= \mathcal{F}\circ\widehat{\mathcal{F}}^{n-1}\circ\widehat{\mathcal{F}}\circ\widehat{\mathcal{F}}\circ\mathcal{F} = \mathcal{F}\circ\widehat{\mathcal{F}}^{n+1}\circ\mathcal{F} \qquad \square
\end{aligned}$$

B Soundness, termination and completeness of $Exit_*$ and $Test_*$ functions

Notation. We recall the notations introduced before and we introduce others to simplify the proofs. As before, let s be a simple cycle, $f = \mathsf{first}(s)$ its first edge and $I = \langle l, u\rangle \subset f$ be the initial interval. Notice that the functions $Exit_*$ are always called with $I \subseteq \langle L, U\rangle$ (in fact this is the precondition for iterating, see Lemma A.10). Let $I_i = \langle l_i, u_i\rangle = \mathsf{Succ}^i_{sf}(I)$ and $\tilde{I}_i = \langle \tilde{l}_i, \tilde{u}_i\rangle = F^i_{sf}(I)$. The Fundamental Lemma (Lemma A.10) guarantees that $I_i = \tilde{I}_i \cap \langle L, U\rangle$. Remember that $\mathcal{F}(I) = \mathsf{Succ}_{sf}(I) = F_{sf}(I\cap S)\cap J$ and $\widehat{\mathcal{F}}(I) = F_{sf}(I\cap S\cap J)\cap S\cap J$. We use notation Ex for the set returned by $Exit_*$.

Exit-STAY: soundness By hypothesis, $L < l^* < u^* < U$. Hence, for all i, $\tilde{I}_i = \langle \tilde{l}_i, \tilde{u}_i\rangle \subseteq \langle L, U\rangle$, hence $I_i = \tilde{I}_i$ and by Lemma 6.6 we have that $\mathsf{Succ}^i_{se_x}(I) = \emptyset$.
Exit-STAY: termination Trivial.
Exit-DIE: soundness Trivial.
Exit-DIE: termination From the hypothesis we know that there exists an n s.t. $\tilde{I}_n \cap \langle L, U\rangle = \emptyset$ (either because $\tilde{u}_n < L$ if $u^* < L$ or because $U < \tilde{l}_n$ if $U < l^*$). Both cases imply that $\mathsf{Succ}^n_{sf}(I) = \emptyset$.
Exit-BOTH: soundness Notice that we call $Exit_{BOTH}$ with $\mathsf{Succ}_{sf}(I)\cap S = \mathcal{F}(I)$. On the other hand, because the limits are out of $\langle L, U\rangle$, we know that there exists an n such that $\langle L, U\rangle \subset \tilde{I}_n$ and by the Fundamental

Lemma (Lemma A.10), $\widehat{\mathcal{F}}^n(I) = I_n = \langle L, U\rangle$ (i.e. $\widehat{\mathcal{F}}^n \circ \mathcal{F}(I) = \langle L, U\rangle$). By Lemmay A.11 we have that $\mathcal{F}^n(I) = \mathcal{F} \circ \widehat{\mathcal{F}}^{n-2} \circ \mathcal{F}(I) = \mathcal{F}(\langle L, U\rangle) = \mathsf{Succ}_{sf}(\langle L, U\rangle)$.

(1) We prove first that the algorithm produces just 'exits':

$$\mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(\langle L, U\rangle)) \subseteq Ex.$$

This follows directly from the fact that $\mathsf{Succ}_{sf}(\langle L, U\rangle) = \mathsf{Succ}^n_{sf}(I) \subseteq \cup_{m>0}\mathsf{Succ}^m_{sf}(I)$;

(2) We prove now that all the 'exits' are computed ($Ex \subseteq \mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(\langle L, U\rangle))$). By definition, $Ex = \cup_{m>0}\mathsf{Succ}_{se_x} \circ \mathcal{F}^m(I)$, that can be written as $Ex = \mathsf{Succ}_{se_x} \circ \mathcal{F}(I) \cup \mathsf{Succ}_{se_x} \circ \mathcal{F} \circ \mathcal{F}(\cup_{m\geq 2}\mathcal{F}^{m-2}(I))$. Let A be the set $\cup_{m\geq 2}\mathcal{F}^{m-2}(I)$, thus $\mathcal{F} \circ \mathcal{F}(A) = \mathcal{F}(S \cap \mathcal{F}(A)) \subseteq \mathcal{F}(S \cap J) = \mathcal{F}(\langle L, U\rangle)$. On the other hand, $\mathsf{Succ}_{se_x} \circ \mathcal{F}(I) \subseteq \mathsf{Succ}_{se_x} \circ \mathcal{F}(\langle L, U\rangle)$, since $I \subseteq \langle L, U\rangle$ and by monotonicity of both functions. Hence, $Ex \subseteq \mathsf{Succ}_{se_x} \circ \mathcal{F}(\langle L, U\rangle)$.

Exit-BOTH: termination Trivial.

Exit-LEFT: soundness By hypothesis, $l^* < L < u^* \leq U$. Thus, there exists a natural number n s.t. $\tilde{l_n} \leq L$ and for all i, $u_i = \tilde{u}_i \leq U$. Let's consider the following two cases:

(1) If $f \prec e_x$ then $Ex = \emptyset$ (by definition of Exit-LEFT) and $\mathsf{Succ}_{se_x}(I_i) = \emptyset$ for any i (by Lemma 6.6-2), so $\mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(\langle L, \max\{u, u^*\}\rangle)) = \emptyset$;

(2) If $e_x \prec f$, we consider two cases:
  - (a) If $u < u^*$ then for all i, $u_i = \tilde{u}_i \leq u^*$ and then $\cup_{m>0}\mathsf{Succ}^m_{sf}(I) = \mathsf{Succ}_{sf}(L, u^*)$, thus $Ex = \mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(L, u^*))$;
  - (b) If $u^* < u$ then for all i, $u_i = \tilde{u}_i \leq u$ and $\cup_{m>0}\mathsf{Succ}^m_{sf}(I) = \mathsf{Succ}_{sf}(L, u)$. Consequently, $Ex = \mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(L, u))$;

From both cases we have that $Ex = \mathsf{Succ}_{se_x}(\mathsf{Succ}_{sf}(\langle L, \max\{u, u^*\}\rangle))$.

Exit-LEFT: Termination Trivial.

Exit-RIGHT Similar to the previous case.

Test-STAY: soundness We prove the soundness considering each case separately:

(1) We have to prove that if $l^* < x < u^*$ then $x \in Reach(I)$. By hypothesis $l^* < x_f < u^*$, then there exists a positive real number ϵ such that $l^* + \epsilon < x_f < u^* - \epsilon$. It's not difficult to see that exists two real numbers N_1 and N_2 such that for all n greater (or equal) than N_1, $u_n > u^* - \epsilon$ and for all n greater (or equal) than N_2, $l_n < l^* + \epsilon$. Let N be equal to the maximum between N_1 and N_2, then it follows that $l_N < l^* + \epsilon$ and $u^* - \epsilon < u_N$. Thus, $l_N < x_f < u_N$ and x_f is reachable.

(2) We have to prove that if $x \leq l^* \wedge l \downarrow$ then $x \notin Reach(I)$. Trivial, by definition of limit and monotonicity of the sequence.

(3) We have to prove that if $u^* \leq x \wedge u \uparrow$ then $x \notin Reach(I)$. Trivial, by definition of limit and monotonicity of the sequence.

(4) We have to prove that if $(x < l^* \wedge l \uparrow) \vee (u^* < x \wedge u \downarrow)$ then $Search(I, x) \equiv (x \in Reach(I)?)$. Computing $Search(I, x)$ gives a se-

quence of intervals $I, I_1, \ldots, I_n$ s.t. $Reach(I) = \bigcup_i I_i$. If $Search(I, x)$ terminates then $\exists i \cdot (Found(I_i, x) = \text{YES} \vee Found(I_i, x) = \text{NO})$ and $\forall j < i \cdot Found(I_i, x) = \text{NOTYET}$. We analyze then each of the cases of $Found(I, x)$:

(a) If $x \in I$ then $Found(I_i, x) = \text{YES}$ and $x \in I_i$, i.e. $x \in Reach(I)$.

(b) If $I = \emptyset$ then $Found(I_i, x) = \text{NO}$ and $\forall k \geq i \cdot I_k = \emptyset$ and $x \notin I_j$. Thus $x \notin Reach(I)$.

(c) If $x < I \wedge l \uparrow$ then $Found(I_i, x) = \text{NO}$ and $\forall k \geq i \cdot x < l_i < l_k$ and because $x \notin I_j$ then $x \notin I_k$ and hence $x \notin Reach(I)$.

(d) If $I < x \wedge u \downarrow$ then $Found(I_i, x) = \text{NO}$ and $\forall k \geq i \cdot u_k < u_i < x$ and because $x \notin I_j$ then $x \notin I_k$ and hence $x \notin Reach(I)$.

Test-STAY: termination We have to show termination just when $(x < l^* \wedge l \uparrow) \vee (u^* < x \wedge u \downarrow)$. If $x < l^* \wedge l \uparrow$ then $\exists i \cdot (x < l_i < l^* \wedge Found(I_i, x) = NO)$. Thus, it terminates. Similarly for the other case.

Test-DIE: soundness Trivial.

Test-DIE: termination Eventually I becomes empty. Hence, at this stage $Found(I, x) = \text{NO}$ and $Search$ terminates.

Test-BOTH: soundness Immediate from the proof of soundness of the Exit algorithm for EXIT-BOTH.

Test-BOTH: termination Trivial.

Test-LEFT: soundness The proof is similar to the STAY case.

Test-LEFT: termination We have to consider just the case when $u \downarrow$ and $\mathsf{Succ}_{sf}(\langle L, u^* \rangle) < x$. In this case we know that $\exists i \cdot u^* < u_i < x \wedge Found(I_i, x) = \text{NO}$. Thus the algorithm terminates.

Test-RIGHT The algorithm and its correctness proof are similar to the previous case.