HAL
open science

Algorithmic Analysis of Polygonal Hybrid Systems, Part II: Phase Portrait and Tools

Eugene Asarin, Gordon Pace, Gerardo Schneider, Sergio Yovine

To cite this version:

Eugene Asarin, Gordon Pace, Gerardo Schneider, Sergio Yovine. Algorithmic Analysis of Polygonal Hybrid Systems, Part II: Phase Portrait and Tools. Theoretical Computer Science, 2007, To appear. Manuscript Number: TCS-D-07-00065R1. hal-00172768v2

HAL Id: hal-00172768

https://hal.science/hal-00172768v2

Submitted on 21 Sep 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Algorithmic Analysis of Polygonal Hybrid Systems, Part II: Phase Portrait and Tools

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

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

[b] *Dept. of Computer Science and AI, University of Malta, Msida, Malta*

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

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

Abstract

Polygonal differential inclusion systems (SPDI) are a subclass of planar hybrid automata which can be represented by piecewise constant differential inclusions. The reachability problem as well as the computation of certain objects of the phase portrait is decidable. In this paper we show how to compute the viability, controllability and invariance kernels, as well as semi-separatrix curves for SPDIs. We also present the tool SPeeDI$^+$, which implements a reachability algorithm and computes phase portraits of SPDIs.

Key words: Hybrid systems, differential inclusions, verification, phase portrait

* Corresponding author.

Email addresses: `Eugene.Asarin@liafa.jussieu.fr` (Eugene Asarin), `gordon.pace@um.edu.mt` (Gordon Pace), `gerardo@ifi.uio.no` (Gerardo Schneider), `Sergio.Yovine@imag.fr` (Sergio Yovine).

Contents

1	Introduction	3
2	Theoretical Background	5
2.1	SPDI	6
2.2	Successors and predecessors	8
2.3	Qualitative analysis of simple edge-cycles	9
3	Phase Portrait	11
3.1	Viability Kernel	11
3.2	Controllability Kernel	15
3.3	Invariance Kernel	19
3.4	Semi-Separatrix Curve	21
3.5	Further Properties of the Kernels	24
3.6	Phase Portrait Construction	26
4	SPeeDI$^+$	28
4.1	Description of the Tool	28
4.2	Implementation Issues	30
4.3	Example	35
4.4	Comparison with HyTech	36
5	Concluding Remarks	39
References		40

1 Introduction

Hybrid systems combining discrete and continuous dynamics arise as mathematical models of various artificial and natural systems, and as approximations to complex continuous systems. They have been used in various domains, including avionics, robotics and bioinformatics. Reachability analysis has been the principal research question in the verification of hybrid systems, even if it is a well-known result that for most non-trivial subclasses of hybrid systems reachability and most verification questions are undecidable. Various decidable subclasses have, subsequently, been identified, including timed [AD94] and initialized rectangular automata [HKPV95], hybrid automata with linear vector fields [LPY01], piecewise constant derivative systems (PCDs) [MP93] and polygonal differential inclusion systems (SPDIs) [1] [ASY01].

Compared to reachability verification, qualitative analysis of hybrid systems is a relatively neglected area [ALQ+01b,DV95,KdB01,MS00,SP02,SJSL00]. Typical qualitative questions include: "Are there 'sink' regions where a trajectory can never leave once it enters the region?"; "Which are the basins of attraction of such regions?"; "Are there regions in which every point in the region is reachable from every other point in the region without leaving it?". To answer such questions one usually gives a collection of objects characterizing these sets, hence providing useful information about the qualitative behavior of the hybrid system. We call the set of all such objects for a given system its *phase portrait*, in accordance with the usual meaning of this term.

In this work we will concentrate on SPDIs. An *SPDI* (Fig. 1) is a finite partition $\mathbb{P}$ of the plane (into convex polygonal areas), with a pair of vectors $\mathbf{a}_P$ and $\mathbf{b}_P$ associated to each polygonal area $P \in \mathbb{P}$. At any position on the plane $\mathbf{x}$, where $\mathbf{x} \in P$, the dynamics of the system are defined by the differential inclusion $\dot{\mathbf{x}} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ (where $\angle_{\mathbf{a}}^{\mathbf{b}}$ denotes the angle on the plane between the vectors $\mathbf{a}$ and $\mathbf{b}$).

In [ASY07] it has been proved that edge-to-edge and polygon-to-polygon reachability in SPDIs is decidable by exploiting the topological properties of a subset of the plane, extending the method introduced in [MP93]. The procedure is not based on the computation of the reach set but rather on the exploration of a finite number of types of qualitative behaviors obtained from the edge-signatures of trajectories (the sequences of their intersections with the edges of the polygons). Such types of signatures may contain loops which can be very expensive (or impossible) to explore naïvely. However, it has been shown that loops have structural properties that can be exploited to efficiently compute their effect. In summary, the novelty of the approach

[1] In the literature the terms *polygonal hybrid system* and *simple planar differential inclusion* have also been used for SPDI.

is the combination of several techniques, namely, (i) the representation of the two-dimensional continuous dynamics as a one-dimensional discrete dynamical system, (ii) the characterization of the set of qualitative behaviors of the latter as a finite set of types of signatures, and (iii) the "acceleration" of the iterations in the case of cyclic signatures.

Given a cycle on a SPDI, we can speak about a number of kernels pertaining to that cycle. The *viability* kernel is the largest set of points in the cycle which may loop forever within the cycle. The *controllability* kernel is the largest set of strongly connected points in the cycle (such that any point in the set may be reached from any other). An *invariant set* is a set of points such that each point must keep rotating within the set forever. The *invariance kernel* is the largest of such sets. Separatrices are convex polygons dissecting the plane into two mutually non-reachable subsets. The notion of separatrix can be relaxed, obtaining *semi-separatrix curves* (or simply, *semi-separatrices*), such that some points in one set may be reachable from the other set, but not vice-versa.

An important property of a dynamical system is *controllability* which refers to the ability of making the system to go from one state to another. If we think of the first state as being a "bad situation" (e.g., faulty state) and the second one as "good", the importance of this notion in control theory is clear. Besides, controllability kernels are important elements of the phase portrait of an SPDI yielding an analog of Poincaré-Bendixson theorem (see for example [HS74]) for simple trajectories, and the viability kernels are their basins of attraction [Aub90]. Invariance kernels are, on the other hand, "sinks" while semi-separatrices are filters allowing trajectories to traverse regions in one "direction". The information gathered for computing reachability turns out to be useful for computing viability, controllability and invariance kernels of such systems. Algorithms for computing these kernels have been presented in [ASY02,Sch04] and are implemented in the tool set SPeeDI$^+$[PS06b].

This paper is the second part of [ASY07], which describes a reachability algorithm for SPDIs. The contributions of the current paper are the following. We first show how to compute viability, controllability and invariance kernels for SPDIs and we present some properties of such phase portrait objects. We then continue by giving an algorithm to compute semi-separatrices of SPDIs. Finally, we present the tool SPeeDI$^+$, which implements the reachability algorithm presented in [ASY01,ASY07], and the computation and visualization of the above-mentioned phase portrait objects.

This work is an extended and revised version of [Sch02, chapter 6,8] and a number of conference papers on SPDIs. We have shown how to compute viability and controllability kernels in [ASY02] and invariance kernels in [Sch04]. The computation of semi-separatrices was presented in [PS06c]. A short presentation of the tool SPeeDI appeared in [APSY02], while the description of

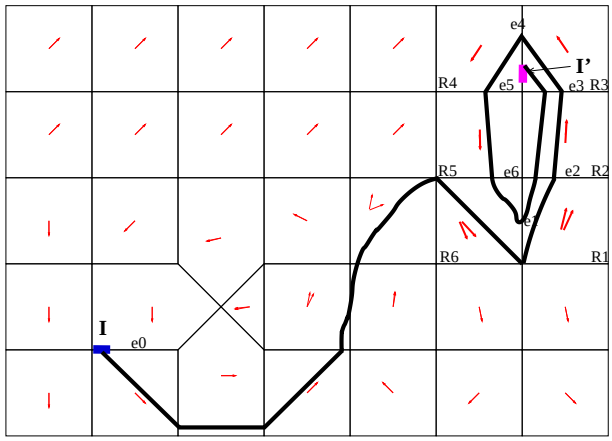


Fig. 1. An SPDI and its trajectory segment.

SPeeDI$^+$ is still unpublished [PS06b].

The paper is structured as follows. In the next section we introduce the necessary theoretical background, including the definition of SPDI and some of its properties; a more detailed and complete presentation can be found in [ASY07]. In section 3 we show how to compute viability, controllability and invariance kernels and semi-separatrices. In section 4 we present SPeeDI$^+$. The last section concludes our presentation.

2 Theoretical Background

A (positive) *affine* function $f : \mathbb{R} \to \mathbb{R}$ is such that $f(x) = ax + b$ with $a > 0$. An *affine multivalued* function $F : \mathbb{R} \to 2^{\mathbb{R}}$, denoted $F = \langle f_l, f_u \rangle$, is defined by $F(x) = \langle f_l(x), f_u(x) \rangle$ where f_l and f_u are affine and $\langle \cdot, \cdot \rangle$ denotes an interval, with $\mathsf{Dom}(F) = \{x \mid f_l(x) \leq f_u(x)\}$. For notational convenience, we do not make explicit whether intervals are open, closed, left-open or right-open, unless required for comprehension. For an interval $I = \langle l, u \rangle$ we have that $F(\langle l, u \rangle) = \langle f_l(l), f_u(u) \rangle$. The *inverse* of F is defined by $F^{-1}(x) = \{y \mid x \in F(y)\}$. The *universal inverse* of F is defined by $\tilde{F}^{-1}(I) = I'$ if and only if I' is the greatest non-empty interval such that for all $x \in I'$, $F(x) \subseteq I$.

It is not difficult to show that $F^{-1} = \langle f_u^{-1}, f_l^{-1} \rangle$ and similarly that $\tilde{F}^{-1} = \langle f_l^{-1}, f_u^{-1} \rangle$, provided that $\langle f_l^{-1}, f_u^{-1} \rangle \neq \emptyset$. Notice that if I is a singleton then $\tilde{F}^{-1}$ is defined only if $f_l = f_u$. These classes of functions are closed under composition.

A *truncated affine multivalued* function (TAMF) $\mathcal{F} : \mathbb{R} \to 2^{\mathbb{R}}$ is defined by an affine multivalued function F and intervals $S \subseteq \mathbb{R}^+$ and $J \subseteq \mathbb{R}^+$ as follows: $\mathcal{F}(x) = F(x) \cap J$ if $x \in S$, otherwise $\mathcal{F}(x) = \emptyset$. For convenience we write $\mathcal{F}(x) = F(\{x\} \cap S) \cap J$. For an interval I, $\mathcal{F}(I) = F(I \cap S) \cap J$ and $\mathcal{F}^{-1}(I) =$

$F^{-1}(I \cap J) \cap S$. The *universal inverse* of $\mathcal{F}$ is defined by $\tilde{\mathcal{F}}^{-1}(I) = I'$ if and only if I' is the greatest non-empty interval such that for all $x \in I'$, $F(x) \subseteq I$ and $F(x) = \mathcal{F}(x)$.

We say that $\mathcal{F}$ is *normalized* if $S = \mathsf{Dom}(\mathcal{F}) = \{x \mid F(x) \cap J \neq \emptyset\}$ (thus, $S \subseteq F^{-1}(J)$) and $J = \mathsf{Im}(\mathcal{F}) = \mathcal{F}(S)$.

The following theorem states that TAMFs are closed under composition.

Theorem 2.1 *The composition of two TAMFs $\mathcal{F}_1(I) = F_1(I \cap S_1) \cap J_1$ and $\mathcal{F}_2(I) = F_2(I \cap S_2) \cap J_2$, is the TAMF $(\mathcal{F}_2 \circ \mathcal{F}_1)(I) = \mathcal{F}(I) = F(I \cap S) \cap J$, where $F = F_2 \circ F_1$, $S = S_1 \cap F_1^{-1}(J_1 \cap S_2)$ and $J = J_2 \cap F_2(J_1 \cap S_2)$.* $\square$

2.1 SPDI

An *angle* $\angle_{\mathbf{a}}^{\mathbf{b}}$ on the plane, 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 $\alpha, \beta \geq 0$, and $\alpha + \beta > 0$. We can always assume that $\mathbf{b}$ is situated in the counter-clockwise direction from $\mathbf{a}$.

A *polygonal differential inclusion system* (SPDI) is defined by giving a finite partition $\mathbb{P}$ of the plane into convex polygonal sets, and associating with each $P \in \mathbb{P}$ a couple of vectors $\mathbf{a}_P$ and $\mathbf{b}_P$. Let $\phi(P) = \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$. The SPDI is determined by $\dot{\mathbf{x}} \in \phi(P)$ for $\mathbf{x} \in P$.

Let $E(P)$ be the set of edges of P. We say that an edge e is an *entry* of P if for all $\mathbf{x} \in e$ 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.

Assumption 1 *All the edges in $E(P)$ are either entries or exits, that is, $E(P) = in(P) \cup out(P)$.*

Reachability for SPDIs is decidable provided the above assumption holds; without such assumption it is not known whether reachability is decidable.

A *trajectory segment* of an SPDI is a continuous function $\xi : [0, T] \to \mathbb{R}^2$ which is smooth everywhere except in a discrete set of points, and such that for all $t \in [0, T]$, if $\xi(t) \in P$ and $\dot{\xi}(t)$ is defined then $\dot{\xi}(t) \in \phi(P)$. The *signature*, denoted $\mathsf{Sig}(\xi)$, is the ordered sequence of all the edges traversed by the trajectory segment, that is, $e_1, e_2, \ldots$, where $\xi(t_i) \in e_i$ and $t_i < t_{i+1}$. If $T = \infty$, a trajectory segment is called a *trajectory*.

Example 1 Consider the SPDI illustrated in Fig. 1. For sake of simplicity we will only show the dynamics associated to regions R_1 to R_6 in the picture. For each region R_i, $1 \leq i \leq 6$, there is a pair of vectors $(\mathbf{a}_i, \mathbf{b}_i)$, where: $\mathbf{a}_1 = (45, 100), \mathbf{b}_1 = (1, 4)$, $\mathbf{a}_2 = \mathbf{b}_2 = (1, 10)$, $\mathbf{a}_3 = \mathbf{b}_3 = (-2, 3)$, $\mathbf{a}_4 = \mathbf{b}_4 = (-2, -3)$, $\mathbf{a}_5 = \mathbf{b}_5 = (1, -15)$, $\mathbf{a}_6 = (1, -2), \mathbf{b}_6 = (1, -1)$.
A trajectory segment starting on interval $I \subset e_0$ and finishing in interval $I' \subset e_4$ is depicted. ■

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

From this definition, it immediately follows that extending an unfeasible signature, can never make it feasible:

Proposition 2.2 *If a signature σ is not feasible, then neither is any extension of the signature — for any signatures σ' and σ'', the signature $\sigma'\sigma\sigma''$ is not feasible.* □

Given an SPDI $\mathcal{S}$, let $\mathcal{E}$ be the set of edges of $\mathcal{S}$, then we can define a graph $\mathcal{G}_\mathcal{S}$ where nodes correspond to edges of $\mathcal{S}$ and such that there exists an arc from one node to another if there exists a trajectory segment from the first edge to the second one without traversing any other edge. More formally: Given an SPDI $\mathcal{S}$, the *underlying graph of* $\mathcal{S}$ (or simply the *graph of* $\mathcal{S}$), is a graph $\mathcal{G}_\mathcal{S} = (N_\mathcal{G}, A_\mathcal{G})$, with $N_\mathcal{G} = \mathcal{E}$ and $A_\mathcal{G} = \{(e, e') \mid \exists \xi, t \,.\, \xi(0) \in e \wedge \xi(t) \in e' \wedge \mathsf{Sig}(\xi) = ee'\}$. We say that a sequence $e_0 e_1 \ldots e_k$ of nodes in $\mathcal{G}_\mathcal{S}$ is a *path* whenever $(e_i, e_{i+1}) \in A_\mathcal{G}$ for $0 \leq i \leq k-1$.

The following lemma shows the relation between edge signatures in an SPDI and paths in its corresponding graph.

Lemma 2.3 *If ξ is a trajectory segment of $\mathcal{S}$ with edge signature $\mathsf{Sig}(\xi) = \sigma = e_0 \ldots e_p$, it follows that σ is a path in $\mathcal{G}_\mathcal{S}$.* □

Remark. Notice that the converse of the above lemma is not true in general. It is possible to find a counter-example where there exists a path from node e to e', but there does not exist a trajectory segment form edge e to edge e' on the SPDI.

Throughout the paper, similarly to [ASY07], we assume that all the constants involved in the definition of the SPDI (coordinates of vectors, coordinates of vertices, etc.) are rational.

Given an SPDI, we fix a one-dimensional coordinate system on each edge to represent points laying on edges. For notational convenience, we will use e to denote both the edge and its one-dimensional representation. Accordingly, we write $\mathbf{x} \in e$ or $x \in e$, to mean "point $\mathbf{x}$ in edge e with coordinate x in the one-dimensional coordinate system of e". The same convention is applied to sets of points of e represented as intervals (e.g., $\mathbf{x} \in I$ or $x \in I$, where $I \subseteq e$) and to trajectories (e.g., "ξ starting in x" or "ξ starting in $\mathbf{x}$").

Now, let $P \in \mathbb{P}$, $e \in in(P)$ and $e' \in out(P)$. For $I \subseteq e$, $\mathsf{Succ}_{e,e'}(I)$ is the set of all points in e' reachable from some point in I by a trajectory segment $\xi : [0,t] \to \mathbb{R}^2$ in P (i.e., $\xi(0) \in I \wedge \xi(t) \in e' \wedge \mathsf{Sig}(\xi) = ee'$). It can be shown that $\mathsf{Succ}_{e,e'}$ is a TAMF.

Example 2 Let $e_1, \ldots, e_6$ be as in Fig. 1 and $I = [l, u]$ on e_1. We assume a one-dimensional coordinate system; here all the edges have local coordinates $0 \leq x \leq 10$. We have:

$$
\begin{aligned}
F_{e_1e_2}(I) &= \left[\frac{l}{4}, \frac{9}{20}u\right], & S_1 &= [0,10], & J_1 &= \left[0, \frac{9}{2}\right] \\
F_{e_2e_3}(I) &= [l+1, u+1], & S_2 &= [0,9], & J_2 &= [1,10] \\
F_{e_3e_4}(I) &= \left[\frac{3}{2}l, \frac{3}{2}u\right], & S_3 &= \left[0, \frac{20}{3}\right], & J_3 &= [0,10] \\
F_{e_4e_5}(I) &= \left[\frac{2}{3}l, \frac{2}{3}u\right], & S_4 &= [0,10], & J_4 &= \left[0, \frac{20}{3}\right] \\
F_{e_5e_6}(I) &= \left[l - \frac{2}{3}, u - \frac{2}{3}\right], & S_5 &= \left[\frac{2}{3}, 10\right], & J_5 &= \left[0, \frac{28}{3}\right] \\
F_{e_6e_1}(I) &= [l, 2u], & S_6 &= [0,10], & J_6 &= [0,10]
\end{aligned}
$$

with $\mathsf{Succ}_{e_ie_{i+1}}(I) = F_{e_ie_{i+1}}(I \cap S_i) \cap J_i$, for $1 \leq i < 6$; S_i and J_i are computed as shown in Theorem 2.1. ■

Given a sequence $w = e_1, e_2, \ldots, e_n$, Theorem 2.1 implies that the successor of I along w defined as $\mathsf{Succ}_w(I) = \mathsf{Succ}_{e_{n-1},e_n} \circ \cdots \circ \mathsf{Succ}_{e_1,e_2}(I)$ is a TAMF.

Example 3 Let $\sigma = e_1 \cdots e_6 e_1$. It results that $\mathsf{Succ}_\sigma(I) = F(I \cap S_\sigma) \cap J_\sigma$, where:

$$
F(I) = \left[\frac{l}{4} + \frac{1}{3}, \frac{9}{10}u + \frac{2}{3}\right] \tag{1}
$$

$S_\sigma = [0, 10]$ and $J_\sigma = [\frac{1}{3}, \frac{29}{3}]$ are computed using Theorem 2.1. ■

For $I \subseteq e'$, $\mathsf{Pre}_{e,e'}(I)$ is the set of points in e that can reach a point in I by a trajectory segment in P. The $\forall$-*predecessor* $\widetilde{\mathsf{Pre}}(I)$ is defined in a similar way to $\mathsf{Pre}(I)$ using the universal inverse instead of just the inverse: For $I \subseteq e'$, $\widetilde{\mathsf{Pre}}_{ee'}(I)$ is the set of points in e such that *any* successor of such points are in I by a trajectory segment in P. Both definitions can be extended straightforwardly to signatures $\sigma = e_1 \cdots e_n$: $\mathsf{Pre}_\sigma(I)$ and $\widetilde{\mathsf{Pre}}_\sigma(I)$. Therefore, the successor operator has two inverse operators.

Example 4 Let $\sigma = e_1 \ldots e_6 e_1$ be as in Fig. 1 and $I = [l, u]$. Now, $\mathsf{Pre}_{e_i e_{i+1}}(I) = F^{-1}_{e_i e_{i+1}}(I \cap J_i) \cap S_i$, for $1 \leq i < 6$, where:

$$F^{-1}_{e_1e_2}(I) = \left[\frac{20}{9}l, 4u\right] \qquad F^{-1}_{e_2e_3}(I) = [l-1, u-1]$$
$$F^{-1}_{e_3e_4}(I) = \left[\frac{2}{3}l, \frac{2}{3}u\right] \qquad F^{-1}_{e_4e_5}(I) = \left[\frac{3}{2}l, \frac{3}{2}u\right]$$
$$F^{-1}_{e_5e_6}(I) = \left[l + \frac{2}{3}, u + \frac{2}{3}\right] \qquad F^{-1}_{e_6e_1}(I) = \left[\frac{l}{2}, u\right]$$

Besides, $\mathsf{Pre}_\sigma(I) = F^{-1}(I \cap J_\sigma) \cap S_\sigma$, where $F^{-1}(I) = [\frac{10}{9}l - \frac{20}{27}, 4u - \frac{4}{3}]$. Similarly, we compute $\widetilde{\mathsf{Pre}}_\sigma(I) = \tilde{F}^{-1}(I \cap J_\sigma) \cap S_\sigma$, where $\tilde{F}^{-1}(I) = \left[4l - \frac{4}{3}, \frac{10}{9}u - \frac{20}{27}\right]$ if $4l - \frac{4}{3} \leq \frac{10}{9}u - \frac{20}{27}$, and $\tilde{F}^{-1}(I)$ is equal to the empty interval otherwise. ■

2.3 Qualitative analysis of simple edge-cycles

Let $\sigma = e_1 \cdots e_k e_1$ be a simple edge-cycle, i.e., $e_i \neq e_j$ for all $1 \leq i \neq j \leq k$. Let $\mathsf{Succ}_\sigma(I) = F(I \cap S_\sigma) \cap J_\sigma$ with $F = \langle f_l, f_u \rangle$ (we suppose that this representation is normalized). We denote by $\mathcal{D}_\sigma$ the one-dimensional discrete-time dynamical system defined by Succ_σ, that is $x_{n+1} \in \mathsf{Succ}_\sigma(x_n)$.

Assumption 2 *None of the two functions f_l, f_u is the identity function.*

Without the above assumption the definition of the kernels given in the next section should have to be slightly modified to consider the particular case whenever f_l or f_u are the identity. The results could be extended to take this into account but the presentation would be rather complicated.

Let σ be a simple cycle, and l^* and u^* be the fix-points[2] of f_l and f_u, respectively, and $S_\sigma \cap J_\sigma = \langle L, U \rangle$. Then σ is of one of the following types:

[2] The fix-point x^* is computed by solving the equation $f(x^*) = x^*$, where $f(\cdot)$ is positive affine.

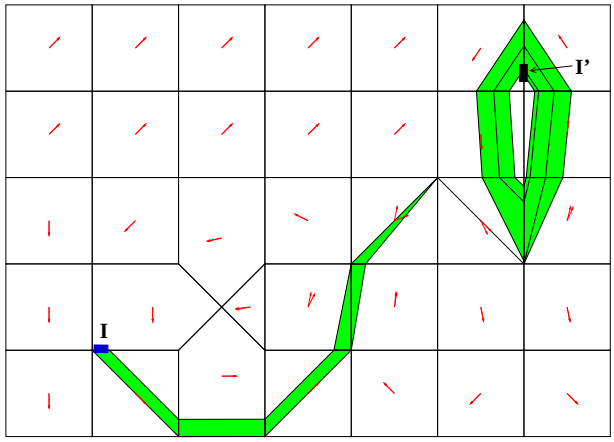


Fig. 2. Reachability analysis.

STAY. The cycle is not abandoned neither by the leftmost nor the rightmost trajectory, that is, $L \leq l^* \leq u^* \leq U$.

DIE. The rightmost trajectory exits the cycle through the left (consequently the leftmost one also exits) or the leftmost trajectory exits the cycle through the right (consequently the rightmost one also exits), that is, $u^* < L \vee l^* > U$.

EXIT-BOTH. The leftmost trajectory exits the cycle through the left and the rightmost one through the right, that is, $l^* < L \wedge u^* > U$.

EXIT-LEFT. The leftmost trajectory exits the cycle (through the left) but the rightmost one stays inside, that is, $l^* < L \leq u^* \leq U$.

EXIT-RIGHT. The rightmost trajectory exits the cycle (through the right) but the leftmost one stays inside, that is, $L \leq l^* \leq U < u^*$.

Example 5 Let $\sigma = e_1 \cdots e_6 e_1$. We have $S_\sigma \cap J_\sigma = \langle L, U \rangle = [\frac{1}{3}, \frac{29}{3}]$. The fix-points of Eq. (1) are such that $\frac{1}{3} < l^* = \frac{11}{25} < u^* = \frac{20}{3} < \frac{29}{3}$. Thus, σ is a STAY. ■

The classification above gives us some useful information about the qualitative behavior of trajectories. Any trajectory that enters a cycle of type DIE will eventually quit it after a finite number of turns. If the cycle is of type STAY, all trajectories that happen to enter it will keep turning inside it forever. In all other cases, some trajectories will turn for a while and then exit, and others will continue turning forever. This information is crucial for proving decidability of the reachability problem.

Example 6 Consider the SPDI of Fig. 1. Fig. 2 shows part of the reach set of the interval $[8, 10] \subset e_0$, answering positively to the reachability question: Is $[1, 2] \subset e_4$ reachable from $[8, 10] \subset e_0$? Fig. 2 has been automatically generated by the SPeeDI$^+$ toolbox. ■

The above result does not allow us to directly answer other questions about the behavior of the SPDI such as determine for a given point (or set of points)

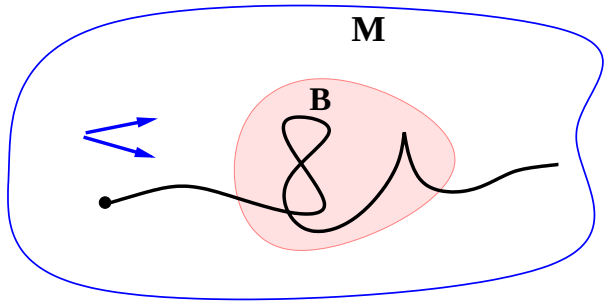


Fig. 3. Example.

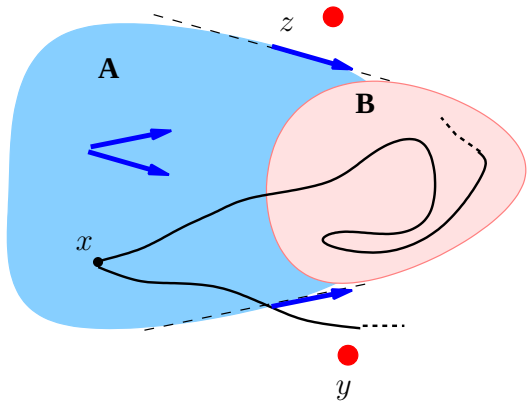


Fig. 4. Example 7: Viability kernel.

whether: (a) there exists at least one trajectory that remains in the cycle, and (b) it is possible to control the system to reach any other point. In order to do this, we need to further study the properties of the system around simple edge-cycles.

3 Phase Portrait

In this section we define and show how to compute the viability, controllability and invariance kernels, as well as the semi-separatrices of an SPDI.

3.1 Viability Kernel

In this and the following sections, we will be studying the qualitative behavior of sets of trajectories having the same cyclic pattern, that is we consider only cyclic signatures. We rely on the information given by the classification given in the previous section (STAY, DIE, etc. cycles) to enable us to analyze better the qualitative behavior of the system. In this first part we introduce the *viability kernel* [Aub90,AC84] and we show how to compute it.

In general, a *viability domain* is a set of points such that for any point in the set, there exists at least one trajectory that remains in the set forever. The *viability kernel* is the largest of such sets.

Example 7 In Fig. 3 there are two disjoint sets, B and $M \setminus B$. The dynamics in B is given by a differential inclusion that allows the first derivative to be any value (i.e., $\angle_{\mathbf{a}}^{\mathbf{b}}$ is such that $\mathbf{a} = 0°$ and $\mathbf{b} = 360°$) whereas outside B, the dynamics is given by the two drawn vectors. Let us consider region A as in Fig. 4. Notice that for any point in A, there is a trajectory segment to a point in B from where it can remain for ever in B. On the other hand, outside A (and outside B), for example points y and z, are not starting points of infinite trajectories. Then, the viability kernel is given by $A \cup B$. ■

In particular, for SPDI, given a cyclic signature, the *viability domain* is a set of points which can keep rotating in the cycle forever and the *viability kernel* is the largest of such sets. We show that this kernel is a non-convex polygon (often with a hole in the middle) and we give a non-iterative algorithm for computing the coordinates of its vertices and edges.

In what follows, let $K \subset \mathbb{R}^2$.

Definition 3.1 *A trajectory ξ is* viable *in K if $\xi(t) \in K$ for all $t \geq 0$. K is a* viability domain *if for every $\mathbf{x} \in K$, there exists at least one trajectory ξ, with $\xi(0) = \mathbf{x}$, which is viable in K. The* viability kernel *of K, denoted $\mathsf{Viab}(K)$, is the largest viability domain contained in K.*

Remark. Differently from [Aub90], we do not require viability kernel to be closed. Indeed in our case sometimes the largest viable set is not closed, and the largest closed viable set does not exist.

3.1.1 One Dimensional Discrete-Time System

The same concepts can be defined for $\mathcal{D}_\sigma$, by setting that a trajectory $x_0 x_1 \ldots$ of $\mathcal{D}_\sigma$ is viable in an interval $I \subseteq \mathbb{R}$, if $x_i \in I$ for all $i \geq 0$.

Theorem 3.2 *For $\mathcal{D}_\sigma$, if σ is not DIE then $\mathsf{Viab}(e_1) = S_\sigma$, else $\mathsf{Viab}(e_1) = \emptyset$.* [3]

PROOF. If σ is DIE, $\mathcal{D}_\sigma$ has no viable trajectories. Therefore, $\mathsf{Viab}(e_1) = \emptyset$. Let σ be not DIE. We first prove that any viability domain is a subset of S_σ. Let I be a viability domain. Then, for all $x \in I$, there exists a trajectory starting in x which is viable in I. Then, $x \in \mathsf{Dom}(\mathsf{Succ}_\sigma) = S_\sigma$. Thus, $I \subseteq S_\sigma$.

[3] Notice that this theorem can be used to compute $\mathsf{Viab}(I)$ for any $I \subseteq e_1$.

Now, let us prove that S_σ is a viability domain. It suffices to show that for all $x \in S_\sigma$, $\mathsf{Succ}_\sigma(x) \cap S_\sigma \neq \emptyset$.
Let $x \in S_\sigma$.
If σ is STAY, we have that both l^* and u^* belong to $S_\sigma \cap J_\sigma$. It follows that both $f_l(x)$ and $f_u(x)$ are in S_σ.
If σ is EXIT-LEFT, we have that $l^* < S_\sigma \cap J_\sigma$ and $u^* \in S_\sigma \cap J_\sigma$. Then, $f_u(x) \in S$.
If σ is EXIT-RIGHT, we have that $l^* \in S_\sigma \cap J_\sigma$ and $u^* > S_\sigma \cap J_\sigma$. Then, $f_l(x) \in S$.
If σ is EXIT-BOTH, we have that $l^* < S_\sigma \cap J_\sigma$ and $u^* > S_\sigma \cap J_\sigma$. If $x \in J_\sigma$: then $x \in F(x)$. If $x < J_\sigma$: then $f_l(x) < x < S_\sigma \cap J_\sigma$, and either $f_u(x) \in S_\sigma \cap J_\sigma$ or $f_u(x) > S_\sigma \cap J_\sigma$ (the other case yields a contradiction). If $x > J_\sigma$: similar to the previous case.
Thus, for all $x \in S$, $\mathsf{Succ}_\sigma(x) \cap S_\sigma \neq \emptyset$. Hence, $\mathsf{Viab}(e_1) = S_\sigma$. $\square$

The following lemma will be useful when proving some results about convergence in the next section.

Lemma 3.3 *For $\mathcal{D}_\sigma$, if the trace $x_1 x_2 \ldots$ of ξ is viable in S_σ then $\forall n > 1 \,.\, x_n \in S_\sigma \cap J_\sigma$.*

PROOF. By Theorem 3.2, $x_1 \in S_\sigma$ and since $x_{n+1} \in \mathsf{Succ}_\sigma(x_n)$ we have that $x_n \in \mathsf{Dom}(\mathsf{Succ}_\sigma)$, i.e. $x_n \in S_\sigma$. On the other hand, $x_n \in \mathsf{Succ}_\sigma(x_{n-1})$ that is included in $\mathsf{Im}(\mathsf{Succ}_\sigma)$, hence $x_n \in J_\sigma$. $\square$

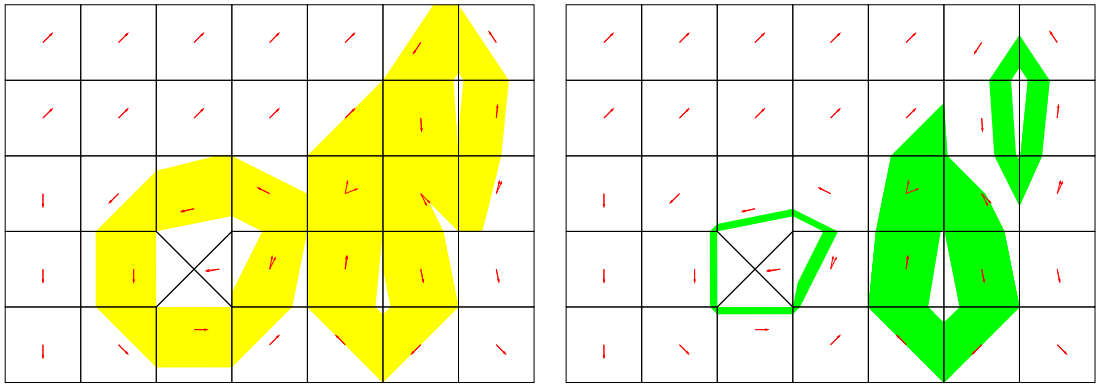
3.1.2 Continuous-Time System

The viability kernel for the continuous-time system can be now found by propagating S_σ from e_1 using the following operator. The *extended predecessor* of an output edge e of a region R is the set of points in R such that there exists a trajectory segment that reaches e without traversing any other edge. More formally:

Definition 3.4 *Let R be a region and e be an edge in $out(R)$. The e-*extended predecessor *of $I \subseteq e$, $\overline{\mathsf{Pre}}_e(I)$ is defined as:*

$$\overline{\mathsf{Pre}}_e(I) = \{\mathbf{x} \mid \exists \xi : [0,t] \to \mathbb{R}^2, t > 0 \,.\, \xi(0) = \mathbf{x} \wedge \xi(t) \in I \wedge \mathsf{Sig}(\xi) = e\}.$$

The above notion can be extended to cyclic signatures (and so to edge-signatures) as follows. Let $\sigma = e_1, \ldots, e_k e_1$ be a cyclic signature. For $I \subseteq e_1$, the σ-*extended predecessor* of I, $\overline{\mathsf{Pre}}_\sigma(I)$ is the set of all $\mathbf{x} \in \mathbb{R}^2$ for which there



(a) (b)

Fig. 5. (a) Viability Kernels; (b) Controllability Kernels.

exists a trajectory segment ξ starting in $\mathbf{x}$, that reaches some point in I, such that $\mathsf{Sig}(\xi)$ is a suffix of $e_2 \dots e_k e_1$.

It is easy to see that $\overline{\mathsf{Pre}}_\sigma(I)$ is a polygonal subset of the plane which can be calculated using the following procedure. First compute $\overline{\mathsf{Pre}}_{e_i}(I)$ for all $1 \leq i \leq n$ and then apply this operation k times:

$$\overline{\mathsf{Pre}}_\sigma(I) = \bigcup_{i=1}^{k} \overline{\mathsf{Pre}}_{e_i}(I_i)$$

with $I_1 = I$, $I_k = \mathsf{Pre}_{e_k e_1}(I_1)$ and $I_i = \mathsf{Pre}_{e_i e_{i+1}}(I_{i+1})$, for $2 \leq i \leq k-1$.

Given that the viability kernels (and the other kernels as well) are defined on cyclic signatures, we need to define a subset of the SPDI determined by such signatures. We thus define the following set:

$$K_\sigma = \bigcup_{i=1}^{k} (int(P_i) \cup e_i) \tag{2}$$

where P_i is such that $e_{i-1} \in in(P_i)$, $e_i \in out(P_i)$ and $int(P_i)$ is the interior of P_i. The segment of a trajectory with signature in σ^* necessarily stays in K_σ.

We can now compute the viability kernel of K_σ.

Theorem 3.5 *If σ is not DIE,* $\mathsf{Viab}(K_\sigma) = \overline{\mathsf{Pre}}_\sigma(S_\sigma)$*, otherwise* $\mathsf{Viab}(K_\sigma) = \emptyset$*.*

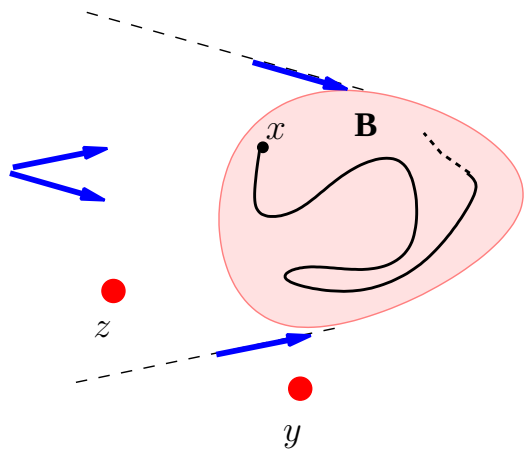


Fig. 6. Example 9: Controllability kernel.

PROOF. If σ is DIE, trivially $\mathsf{Viab}(K_\sigma) = \emptyset$.
Let σ be not DIE. We first prove that any viability domain K, with $K \subseteq K_\sigma$, is a subset of $\overline{\mathsf{Pre}}_\sigma(S_\sigma)$. Let $\mathbf{x} \in K$. Then, there exists a trajectory ξ such that $\xi(0) = \mathbf{x}$ and for all $t \geq 0$, $\xi(t) \in K$. Clearly, the sequence $x_1 x_2 \ldots$ of the intersections of ξ with e_1 is a trajectory of $\mathcal{D}_\sigma$. Then, by Theorem 3.2, $x_i \in S_\sigma$ for all $i \geq 1$. Thus, $\mathbf{x} \in \overline{\mathsf{Pre}}_\sigma(S_\sigma)$.
It remains to prove that $\overline{\mathsf{Pre}}_\sigma(S_\sigma)$ is a viability domain. Let $\mathbf{x} \in \overline{\mathsf{Pre}}_\sigma(S_\sigma)$. Then, there exists a trajectory segment $\bar{\xi} : [0, T] \to \mathbb{R}^2$ such that $\bar{\xi}(T) \in S_\sigma$ and $\mathsf{Sig}(\bar{\xi})$ is a suffix of σ. Theorem 3.2 implies that $\bar{\xi}(T)$ is the initial state of some trajectory ξ with $\mathsf{Sig}(\xi) = \sigma^\omega$. It is straightforward to show that for all $t \geq 0$, $\xi(t) \in \overline{\mathsf{Pre}}_\sigma(S_\sigma)$. Concatenating $\bar{\xi}$ and ξ, we obtain a viable trajectory starting in $\mathbf{x}$.
Hence, $\mathsf{Viab}(K_\sigma) = \overline{\mathsf{Pre}}_\sigma(S_\sigma)$. $\square$

This result provides a non-iterative algorithmic procedure for computing the viability kernel of K_σ.

Example 8 Fig. 5-(a) shows all the viability kernels of the SPDI given in Example 1. There are 4 cycles with viability kernels — in the picture two of the kernels are overlapping. ■

3.2 Controllability Kernel

In this section we define and we show how to compute the *controllability kernel* of a simple cycle.

We say $M \subset \mathbb{R}^2$ is *controllable* if for any two points $\mathbf{x}$ and $\mathbf{y}$ in M there exists a trajectory segment ξ starting in $\mathbf{x}$ that reaches an arbitrarily small neighborhood of $\mathbf{y}$ without leaving M.

Example 9 Let us consider again example of Fig. 3, where there are two disjoint sets B and $M \setminus B$. The dynamics in B is given by a differential inclusion that allows the first derivative to be any value (i.e., $\angle_{\mathbf{a}}^{\mathbf{b}}$ is such that $\mathbf{a} = 0°$ and $\mathbf{b} = 360°$) whereas outside B, the dynamics is given by the two drawn vectors. Notice that any point x in B is the starting point of a trajectory that reach any other point in B as shown in Fig. 6. Outside B points are not reachable one from the other, x is reachable from z but not vice-versa, for instance. Then, B is the controllability kernel. ■

For SPDIs and considering cyclic signatures, the controllability kernel is a cyclic polygonal stripe within which a trajectory can reach any point from any point. More formally,

Definition 3.6 *We say that M is* controllable *iff* $\forall \mathbf{x}, \mathbf{y} \in M, \forall \delta > 0, \exists \xi : [0,t] \to \mathbb{R}^2, t > 0 \,.\, (\xi(0) = \mathbf{x} \wedge |\xi(t) - \mathbf{y}| < \delta \wedge \forall t' \in [0,t] \,.\, \xi(t') \in M)$. *The* controllability kernel *of a set K, denoted $\mathsf{Cntr}(K)$, is the largest controllable subset of K.*

Notice that existence of such a largest set is not guaranteed in general. However, in the sequel we establish that controllability kernels always exist for K_σ sets in SPDIs satisfying Assumption 2. Moreover, we give an exact procedure allowing computation of the kernel.

3.2.1 One Dimensional Discrete-Time System

The above notions can be defined for the discrete dynamical system $\mathcal{D}_\sigma$. In order to compute the controllability kernel for the one-dimensional discrete-time dynamical system we need the following:

$$\mathcal{C}_{\mathcal{D}}(\sigma) = \begin{cases} \langle L, U \rangle & \text{if } \sigma \text{ is EXIT-BOTH} \\ \langle L, u^* \rangle & \text{if } \sigma \text{ is EXIT-LEFT} \\ \langle l^*, U \rangle & \text{if } \sigma \text{ is EXIT-RIGHT} \\ \langle l^*, u^* \rangle & \text{if } \sigma \text{ is STAY} \\ \emptyset & \text{if } \sigma \text{ is DIE} \end{cases}$$

We have then the following result for computing controllability kernels for the discrete-time system.

Theorem 3.7 *For $\mathcal{D}_\sigma$, $\mathsf{Cntr}(S_\sigma) = \mathcal{C}_{\mathcal{D}}(\sigma)$.*

PROOF. Controllability of $\mathcal{C}_{\mathcal{D}}(\sigma)$ follows from the reachability result given in [ASY07]. To prove that $\mathcal{C}_{\mathcal{D}}(\sigma)$ is maximal we reason by contradiction. Suppose it is not. Then, there should exist a controllable set $C \supset \mathcal{C}_{\mathcal{D}}(\sigma)$. Since $C \subseteq S_\sigma \cap J_\sigma$, there should exist $y \in C$ such that either $y < l^*$, or $y > u^*$. In any case, controllability implies that for all $l^* < x < u^*$, there exists a trajectory segment starting in x that reaches an arbitrarily small neighborhood of y. From the reachability algorithm given in [ASY07] we know that $Reach(x) \subset (l^*, u^*)$, which yields a contradiction. Hence, $\mathcal{C}_{\mathcal{D}}(\sigma)$ is the controllability kernel of S_σ. $\square$

3.2.2 Continuous-Time System

For $I \subseteq e_1$ let us define $\overline{\mathsf{Succ}}_\sigma(I)$ as the set of all points $\mathbf{y} \in \mathbb{R}^2$ for which there exists a trajectory segment ξ starting in some point $x \in I$, that reaches $\mathbf{y}$, such that $\mathsf{Sig}(\xi)$ is a prefix of $e_1 \dots e_k$. The successor $\overline{\mathsf{Succ}}_\sigma(I)$ is a polygonal subset of the plane which can be computed similarly to $\overline{\mathsf{Pre}}_\sigma(I)$, that is,

Definition 3.8 *Let R be a region and e be an edge in $in(R)$. The e-*extended successor *of $I \subseteq e$, $\overline{\mathsf{Succ}}_e(I)$ is defined as:*

$$\overline{\mathsf{Succ}}_e(I) = \{\mathbf{y} \mid \exists \xi, \mathbf{x} \in I, t > 0 \,.\, \xi(0) = \mathbf{x} \wedge \xi(t) = \mathbf{y} \wedge \mathsf{Sig}(\xi) = e\}.$$

The extended successors for cyclic signatures (and for edge-signatures) can be defined as follows. Let $\sigma = e_1, \dots, e_k e_1$ be a cyclic signature. For $I \subseteq e_1$, the *σ-extended successor* of I, $\overline{\mathsf{Succ}}_\sigma(I)$ is the set of all reachable points $\mathbf{y} \in \mathbb{R}^2$ via a trajectory segment ξ starting in $\mathbf{x} \in I$, such that $\mathsf{Sig}(\xi)$ is a prefix of $e_1 \dots e_k$.

As for extended predecessors, $\overline{\mathsf{Succ}}_\sigma(I)$ is a polygonal subset of the plane which can be calculated using the following procedure. First compute $\overline{\mathsf{Succ}}_{e_i}(I)$ for all $1 \leq i \leq n$ and then apply this operation k times:

$$\overline{\mathsf{Succ}}_\sigma(I) = \bigcup_{i=1}^{k} \overline{\mathsf{Succ}}_{e_i}(I_i)$$

where $I_1 = I$ and $I_{i+1} = \mathsf{Succ}_{e_i e_{i+1}}(I)$ for $1 \leq i \leq k-1$.

Let $\mathcal{C}(\sigma)$ be defined as follows:

$$\mathcal{C}(\sigma) \;=\; (\overline{\mathsf{Succ}}_\sigma \cap \overline{\mathsf{Pre}}_\sigma)(\mathcal{C}_{\mathcal{D}}(\sigma)).$$

In the following theorem we show how to compute controllability kernels for continuous-time systems:

Theorem 3.9 $\mathsf{Cntr}(K_\sigma) = \mathcal{C}(\sigma)$.

PROOF. Let $\mathbf{x}, \mathbf{y} \in \mathcal{C}(\sigma)$. Since $\mathbf{y} \in \overline{\mathsf{Succ}}_\sigma(\mathcal{C}_\mathcal{D}(\sigma))$, there exists a trajectory segment starting in some point $w \in \mathcal{C}_\mathcal{D}(\sigma)$ and ending in $\mathbf{y}$. Let ϵ be an arbitrarily small number and $B_\epsilon(\mathbf{y})$ be the set of all points $\mathbf{y}'$ such that $|\mathbf{y} - \mathbf{y}'| < \epsilon$. Clearly, $w \in \overline{\mathsf{Pre}}_\sigma(B_\epsilon(\mathbf{y})) \cap \mathcal{C}_\mathcal{D}(\sigma)$. Now, since $\mathbf{x} \in \overline{\mathsf{Pre}}_\sigma(\mathcal{C}_\mathcal{D}(\sigma))$, there exists a trajectory segment starting in $\mathbf{x}$ and ending in some point $z \in \mathcal{C}_\mathcal{D}(\sigma)$. Since $\mathcal{C}_\mathcal{D}(\sigma)$ is controllable, there exists a trajectory segment starting in z that reaches a point in $\overline{\mathsf{Pre}}_\sigma(B_\epsilon(\mathbf{y})) \cap \mathcal{C}_\mathcal{D}(\sigma)$. Thus, there is a trajectory segment that starts in $\mathbf{x}$ and ends in $B_\epsilon(\mathbf{y})$. Therefore, $\mathcal{C}(\sigma)$ is controllable. Maximality follows from the maximality of $\mathcal{C}_\mathcal{D}(\sigma)$ (Theorem 3.7) and the definition of $\overline{\mathsf{Succ}}_\sigma$ and $\overline{\mathsf{Pre}}_\sigma$. Hence, $\mathcal{C}(\sigma)$ is the controllability kernel of K_σ. $\square$

This result provides a non-iterative algorithmic procedure for computing the controllability kernel of K_σ.

Example 10 Fig. 5-(b) shows all the controllability kernels of the SPDI given in Example 1. There are 4 cycles with controllability kernels — in the picture two of the kernels are overlapping. ■

In what follows we provide some notations and definitions related to controllability kernels. Let $\mathsf{Cntr}^l(K_\sigma)$ be the closed curve obtained by taking the leftmost trajectory and $\mathsf{Cntr}^u(K_\sigma)$ be the closed curve obtained by taking the rightmost trajectory which can remain inside the controllability kernel. In other words, $\mathsf{Cntr}^l(K_\sigma)$ and $\mathsf{Cntr}^u(K_\sigma)$ are the two curves defining the controllability kernel. A non-empty controllability kernel $\mathsf{Cntr}(K_\sigma)$ of a given cyclic signature σ partitions the plane into three disjoint subsets: (1) the controllability kernel itself, (2) the set of points limited by $\mathsf{Cntr}^l(K_\sigma)$ (and not including $\mathsf{Cntr}^l(K_\sigma)$), and (3) the set of points limited by $\mathsf{Cntr}^u(K_\sigma)$ (and not including $\mathsf{Cntr}^u(K_\sigma)$).

Definition 3.10 *We define the* inner *of* $\mathsf{Cntr}(K_\sigma)$ *(denoted by* $\mathsf{Cntr}_{in}(K_\sigma)$*) to be the subset defined by (2) above if the cycle is counter-clockwise or to be the subset defined by (3) if it is clockwise. The* outer *of* $\mathsf{Cntr}(K_\sigma)$ *(denoted by* $\mathsf{Cntr}_{out}(K_\sigma)$*) is defined to be the subset which is not the inner nor the controllability itself.*

Remark: Notice that an edge in the SPDI may be split into parts by the controllability kernel — part inside, part on the kernel, and part outside. In such cases, we can generate a different SPDI, with the same dynamics but with the edge split into parts, such that each part is completely inside, on, or outside the kernel. Although the signatures will obviously change, it is trivial to prove that the behavior of the SPDI remains identical to the original. To simplify presentation, in the rest of the paper we will assume that all edges

are either completely inside, on, or completely outside the kernels. We note that in practice splitting is not necessary since we can just consider parts of edges.

3.3 Invariance Kernel

In general, an *invariant set* is a set of points such that for any point in the set, every trajectory starting in such point remains in the set forever and the *invariance kernel* is the largest of such sets. In particular, for SPDI, given a cyclic signature, an *invariant set* is a set of points which keep rotating in the cycle forever and the *invariance kernel* is the largest of such sets. More formally:

Definition 3.11 *A set M is said to be* invariant *if for any $x \in M$ there exists at least one trajectory starting in it and every trajectory starting in x is viable in M. Given a set K, its largest invariant subset is called the* invariance kernel *of K and is denoted by $\mathsf{Inv}(K)$.*

We need some preliminary definitions before showing how to compute the kernel. The *extended $\forall$-predecessor* of an output edge e of a region R is the set of points in R such that every trajectory segment starting in such point reaches e without traversing any other edge. More formally,

Definition 3.12 *Let R be a region and e be an edge in out(R), then the e-*extended $\forall$-predecessor *of I, $\widetilde{\overline{\mathsf{Pre}}}_e(I)$ is defined as:*

$$\widetilde{\overline{\mathsf{Pre}}}_e(I) = \{\mathbf{x} \mid \forall \xi \,.\, (\xi(0) = \mathbf{x} \Rightarrow \exists t \geq 0 \,.\, (\xi(t) \in I \wedge \mathsf{Sig}(\xi[0,t]) = e))\}.$$

It is easy to see that $\widetilde{\overline{\mathsf{Pre}}}_\sigma(I)$ is a polygonal subset of the plane which can be calculated using the following procedure. First compute $\widetilde{\overline{\mathsf{Pre}}}_{e_i}(I)$ for all $1 \leq i \leq k$ and then apply this operation k times:

$$\widetilde{\overline{\mathsf{Pre}}}_\sigma(I) = \bigcup_{i=1}^{k} \widetilde{\overline{\mathsf{Pre}}}_{e_i}(I_i)$$

with $I_1 = I$, $I_k = \widetilde{\mathsf{Pre}}_{e_k e_1}(I_1)$ and $I_i = \widetilde{\mathsf{Pre}}_{e_i e_{i+1}}(I_{i+1})$, for $2 \leq i \leq k-1$.

To prove the computability of invariance kernels, we need the following results (we only give here the proof of the main theorem; see [Sch04] for a complete proof of the auxiliary lemmas). Remember that F is an AMF and $\mathcal{F}$ is a TAMF.

Lemma 3.13 *For any STAY cycle σ, $\mathcal{F}(\tilde{\mathcal{F}}^{-1}(J_\sigma)) = J_\sigma$.*

Lemma 3.14 *For any STAY cycle σ, $F(\tilde{\mathcal{F}}^{-1}(J_\sigma)) = \mathcal{F}(\tilde{\mathcal{F}}^{-1}(J_\sigma))$.*

Lemma 3.15 *For any STAY cycle σ, $F(S_\sigma \cap J_\sigma) \subseteq S_\sigma \cap J_\sigma$.*

Lemma 3.16 *For any STAY cycle σ, $\mathcal{F}(S_\sigma \cap J_\sigma) = F(S_\sigma \cap J_\sigma)$.*

Lemma 3.17 *For any STAY cycle σ, $J_\sigma \subseteq S_\sigma$.*

Lemma 3.18 *For $\mathcal{D}_\sigma$ and σ a STAY cycle, the following is valid. If I is such that $F(I) \subseteq I$ and $F(I) = \mathcal{F}(I)$ then I is invariant. On the other hand if I is invariant then $F(I) = \mathcal{F}(I)$.*

We compute the invariance kernel of K_σ as follows:

Theorem 3.19 *If σ is STAY then $\mathsf{Inv}(K_\sigma) = \widetilde{\overline{\mathsf{Pre}}}_\sigma(\widetilde{\mathsf{Pre}}_\sigma(J_\sigma))$, otherwise $\mathsf{Inv}(K_\sigma) = \emptyset$.*

PROOF. That $\mathsf{Inv}(e_1) = \emptyset$ for any type of cycle but STAY follows directly from the definition of each type of cycle.

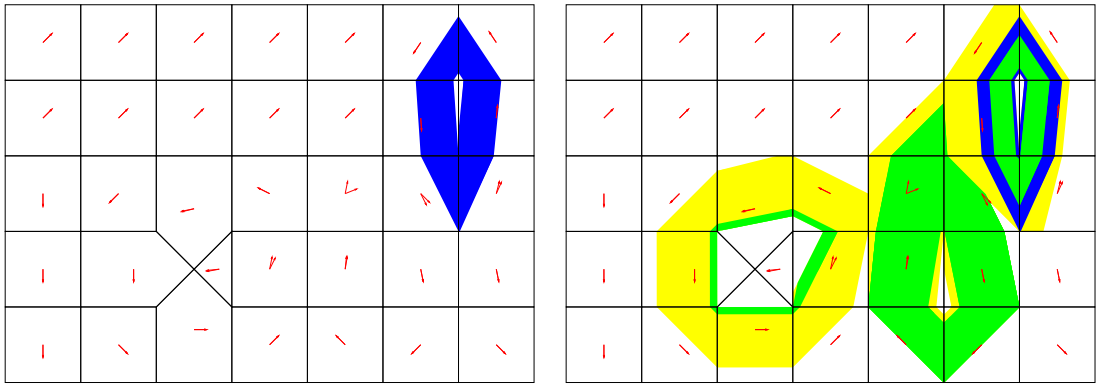
Let's consider a STAY cycle with signature σ. Let $I_K = \tilde{\mathcal{F}}^{-1}(J_\sigma) = \widetilde{\mathsf{Pre}}_\sigma(J_\sigma)$. We know that $F(\tilde{\mathcal{F}}^{-1}(J_\sigma)) = \mathcal{F}(\tilde{\mathcal{F}}^{-1}(J_\sigma)) = J_\sigma$ (see Lemmas 3.13 and 3.14). By Lemmas 3.15, 3.16 and 3.17, we have that $\mathcal{F}(J_\sigma) \subseteq J_\sigma$, so $J_\sigma \subseteq \tilde{\mathcal{F}}^{-1}(J_\sigma)$ and then $F(\tilde{\mathcal{F}}^{-1}(J_\sigma)) \subseteq \tilde{\mathcal{F}}^{-1}(J_\sigma)$. We have then, by Lemma 3.18, that I_K is invariant. We prove now that I_K is indeed the greatest invariant. Let suppose that there exists an invariant $H \subseteq S_\sigma$ strictly greater than I_K. By assumption we have that $I_K = \tilde{\mathcal{F}}^{-1}(J_\sigma) \subset H$, then by monotonicity of $\mathcal{F}$, $\mathcal{F}(\tilde{\mathcal{F}}^{-1}(J_\sigma)) \subset \mathcal{F}(H)$ and by Lemma 3.13 we have that $J_\sigma \subset \mathcal{F}(H)$, but this contradicts the monotonicity of $\mathcal{F}$ since $J_\sigma = \mathcal{F}(S_\sigma) \subset \mathcal{F}(H)$ and then $S_\sigma \subset H$ which contradicts the hypothesis that $H \subseteq S_\sigma$. Hence, $\mathsf{Inv}(e_1) = \widetilde{\mathsf{Pre}}_\sigma(J_\sigma)$. $\square$

Example 11 Fig. 7-(a) shows the unique invariance kernels of the SPDI given in Example 1. ■

An interesting property of invariance kernels is that the limits are included in the invariance kernel, i.e. $[l^*, u^*] \subseteq \mathsf{Inv}(K_\sigma)$. In other words:

Proposition 3.20 *The set delimited by the polygons defined by the interval $[l^*, u^*]$ is an invariance set of STAY cycles.* $\square$

In a similar way as for the controllability kernel, we define $\mathsf{Inv}^l(K_\sigma)$, $\mathsf{Inv}^u(K_\sigma)$, the inner $\mathsf{Inv}_{in}(K_\sigma)$ and outer $\mathsf{Inv}_{out}(K_\sigma)$ of an invariance kernel.



(a) (b)

Fig. 7. (a) Invariance kernel; (b) All the kernels.

3.4 Semi-Separatrix Curve

In this section we define the notion of *separatrix curves*, which are curves on $\mathbb{R}^2$ dissecting the plane into two mutually non-reachable subsets. We relax the notion of separatrix obtaining *semi-separatrix curves* such that some points in one set may be reachable from the other set, but not vice-versa.

Definition 3.21 *Let $K \subseteq \mathbb{R}^2$. A* separatrix *in K is a closed curve γ partitioning K into three pairwise disjoint sets K_A, K_B and γ itself, such that $K = K_A \cup K_B \cup \gamma$ and the following conditions hold:*

(1) For any point $\mathbf{x}_0 \in K_A$ and trajectory ξ, with $\xi(0) = \mathbf{x}_0$, there is no t such that $\xi(t) \in K_B$; and
(2) For any point $\mathbf{x}_0 \in K_B$ and trajectory ξ, with $\xi(0) = \mathbf{x}_0$, there is no t such that $\xi(t) \in K_A$.

If only one of the above conditions holds then we say that the curve is a semi-separatrix*. If only condition 1 holds, then we say that K_A is the* inner *of γ (written γ_{in}) and K_B is the* outer *of γ (written γ_{out}). If only condition 2 holds, K_B is the* inner *and K_A is the* outer *of γ.*

Remark: Notice that, as in the case of the controllability kernel, an edge of the SPDI may be split into two by a semi-separatrix — part inside, and part outside. As before, we can split the edge into parts, such that each part is completely inside, or completely outside the semi-separatrix.

The set of all the separatrices of $\mathbb{R}^2$ is denoted by $\mathsf{Sep}(\mathbb{R}^2)$, or simply Sep. The above notions are extended to SPDIs straightforwardly.

Now, let $\sigma = e_1 \dots e_n e_1$ be a simple cycle, $\angle^{\mathbf{b}_i}_{\mathbf{a}_i}$ $(1 \leq i \leq n)$ be the dynamics of the regions for which e_i is an entry edge and $I = [l, u]$ and interval on edge e_1. Remember that $\mathsf{Succ}_{e_1 e_2}(I) = F(I \cap S) \cap J$, where $F = [a_1 l + b_1, a_2 u + b_2]$. Let $\mathbf{l}$ be the vector corresponding to the point on e_1 with local coordinates l and $\mathbf{l}'$ be the vector corresponding to the point on e_2 with local coordinates $F(l)$ (similarly, we define $\mathbf{u}$ and $\mathbf{u}'$ for $F(u)$). We define first $\overline{\mathsf{Succ}}^{\mathbf{b}_1}_{e_1}(I) = \{\mathbf{x} \mid \mathbf{l}' = \alpha\mathbf{x} + \mathbf{l}, 0 < \alpha < 1\}$ and $\overline{\mathsf{Succ}}^{\mathbf{a}_1}_{e_1}(I) = \{\mathbf{x} \mid \mathbf{u}' = \alpha\mathbf{x} + \mathbf{u}, 0 < \alpha < 1\}$. We extend these definitions in a straight way to any (cyclic) signature $\sigma = e_1 \dots e_n e_1$, denoting them by $\overline{\mathsf{Succ}}^{\mathbf{b}}_{\sigma}(I)$ and $\overline{\mathsf{Succ}}^{\mathbf{a}}_{\sigma}(I)$, respectively; we can compute them similarly as for $\overline{\mathsf{Succ}}$. Whenever applied to the fix-point $I^* = [l^*, u^*]$, we denote $\overline{\mathsf{Succ}}^{\mathbf{b}}_{\sigma}(I^*)$ and $\overline{\mathsf{Succ}}^{\mathbf{a}}_{\sigma}(I^*)$ by ξ^l_σ and ξ^u_σ respectively. Intuitively, ξ^l_σ (ξ^u_σ) denotes the piece-wise affine closed curve defined by the leftmost (rightmost) fix-point l^* (u^*).

We show now how to identify semi-separatrices for simple cycles.

Theorem 3.22 *Given an SPDI, let σ be a simple cycle, then the following hold:*

(1) If σ is EXIT-RIGHT then ξ^l_σ is a semi-separatrix curve (filtering trajectories from "left" to "right");
(2) If σ is EXIT-LEFT then ξ^u_σ is a semi-separatrix curve (filtering trajectories from "right" to "left");
(3) If σ is STAY, then the two polygons defining the invariance kernel ($\mathsf{Inv}^l(K_\sigma)$ and $\mathsf{Inv}^u(K_\sigma)$), are semi-separatrices.

PROOF.

(1) By definition of EXIT-RIGHT, any trajectory is bounded to the left by ξ^l_σ, which is a piece-wise affine closed curve, partitioning $\mathbb{R}^2$ into three disjoint sets: K_B, the "right" part of ξ^l_σ; K_A, the "left" part of ξ^l_σ; and ξ^l_σ itself. By Jordan's theorem, any trajectory may pass from K_B to K_A if and only if it cross ξ^l_σ. However, by definition of EXIT-RIGHT, this is only possible from K_A to K_B but not vice-versa. Hence ξ^l_σ is a semi-separatrix curve.
(2) Symmetric to the previous case.
(3) Follows directly from the definition of invariance kernel, since any trajectory with initial point in $\mathsf{Inv}(K_\sigma) \cup \mathsf{Inv}_{in}(K_\sigma)$ cannot leave $\mathsf{Inv}(K_\sigma)$. If the trajectory cycles clockwise it cannot traverse $\mathsf{Inv}^l(K_\sigma)$ and if it cycles counter-clockwise it cannot traverse $\mathsf{Inv}^u(K_\sigma)$. In both cases no point on $\mathsf{Inv}_{out}(K_\sigma)$ can be reached. Symmetrically, trajectories starting in $\mathsf{Inv}(K_\sigma) \cup \mathsf{Inv}_{out}(K_\sigma)$ cannot reach any point on $\mathsf{Inv}_{in}(K_\sigma)$. $\square$

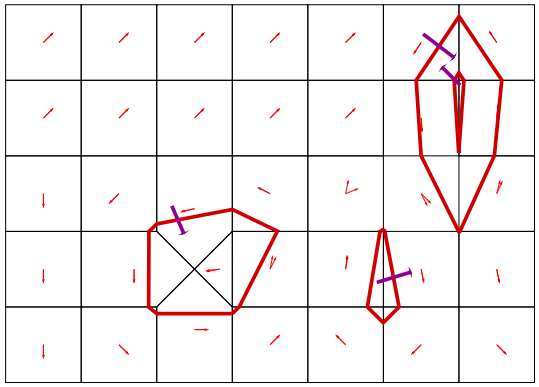


Fig. 8. Semi-separatrices.

Remark: In the case of STAY cycles, ξ_σ^l and ξ_σ^u are also semi-separatrices. Whenever the dynamics of the cycle σ is the identity, there is an infinite number of semi-separatrices. This is, however, disallowed by Assumption 2.

Notice that in the above result, computing a semi-separatrix depends only on one simple cycle, and the corresponding algorithm is then reduced to find simple cycles in the SPDI and checking whether it is STAY, EXIT-RIGHT or EXIT-LEFT.

Example 12 Fig. 8 shows all the semi-separatrices of the SPDI given in Example 1 obtained as shown in Theorem 3.22. The small arrows traversing the semi-separatrices show the inner and outer of each semi-separatrix: a trajectory may traverse the semi-separatrix following the direction of the arrow, but not vice-versa. ■

The following two results relate feasible signatures and semi-separatrices.

Proposition 3.23 *If, for some semi-separatrix γ, $e \in \gamma_{in}$ and $e' \in \gamma_{out}$, then the signature ee' is not feasible.* □

PROOF. Directly from the definition of semi-separatrix. □

Proposition 3.24 *Given a semi-separatrix γ and signature σ (of at least length 2), then σ is not feasible if $head(\sigma) \in \gamma_{in}$ and $last(\sigma) \in \gamma_{out}$.*

PROOF. The proof proceeds by induction on sequence σ. The base case, when σ is of length 2, reduces to Proposition 3.23. Now, assuming that the proposition is true for signatures of length n, we are required to prove that it is also true for signatures of length $n+1$. Consider the signature $\sigma' = ee'\sigma e''$, with $e \in \gamma_{in}$ and $e'' \in \gamma_{out}$. Clearly, either $e' \in \gamma_{in}$ or $e' \in \gamma_{out}$.

Case 1: $e' \in \gamma_{in}$. The signature $e'\sigma e''$ satisfies the conditions and is of length n. Therefore, the inductive property applies, and we can conclude that $e'\sigma e''$ is not feasible. However, since any extension of an unfeasible signature is itself unfeasible, it follows that σ' is not feasible.

Case 2: $e' \in \gamma_{out}$. The signature ee' is unfeasible by Proposition 3.23. Therefore, being an extension of ee', σ' is also unfeasible (Proposition 2.2). □

3.5 Further Properties of the Kernels

In this section we present some properties of controllability kernels, regarding convergence and its relation to fix-points in general. In particular, for STAY cycles we have stronger limit cycle properties.

3.5.1 Convergence

Definition 3.25 *A trajectory* ξ converges *to a set* $K \subset \mathbb{R}^2$ *if* $\lim_{t\to\infty} dist(\xi(t), K) = 0$.

For $\mathcal{D}_\sigma$, convergence is defined as $\lim_{n\to\infty} dist(\xi_n, I) = 0$. The following result says that the controllability kernel $\mathcal{C}_\mathcal{D}(\sigma)$ can be considered to be a kind of (weak) limit cycle of $\mathcal{D}_\sigma$.

Theorem 3.26 *For $\mathcal{D}_\sigma$, any viable trajectory in S_σ converges to $\mathcal{C}_\mathcal{D}(\sigma)$.*

PROOF. Let $x_1x_2\ldots$ be a viable trajectory. By Lemma 3.3, $x_i \in S_\sigma \cap J_\sigma$ for all $i \geq 2$. Recall that $\mathcal{C}_\mathcal{D}(\sigma) \subseteq S_\sigma \cap J_\sigma$. There are three cases: (1) There exists $N \geq 2$ such that $x_N \in \mathcal{C}_\mathcal{D}(\sigma)$. Then, for all $n \geq N$, $x_n \in \mathcal{C}_\mathcal{D}(\sigma)$. (2) For all n, $x_n < \mathcal{C}_\mathcal{D}(\sigma)$. Therefore, $x_n < l^*$. Let $\hat{x}_n$ be such that $\hat{x}_1 = x_1$ and for all $n \geq 1$, $\hat{x}_{n+1} = f_l(\hat{x}_n)$. Clearly, for all n, $\hat{x}_n \leq x_n < l^*$, and $\lim_{n\to\infty} \hat{x}_n = l^*$, which implies $\lim_{n\to\infty} x_n = l^*$. (3) For all n, $x_n > \mathcal{C}_\mathcal{D}(\sigma)$. Therefore, $u^* < x_n$. Let $\hat{x}_n$ be such that $\hat{x}_1 = x_1$ and for all $n \geq 1$, $\hat{x}_{n+1} = f_u(\hat{x}_n)$. Clearly, for all n, $u^* < x_n \leq \hat{x}_n$, and $\lim_{n\to\infty} \hat{x}_n = u^*$, which implies $\lim_{n\to\infty} x_n = u^*$. Hence, $x_1x_2\ldots$ converges to $\mathcal{C}(\sigma)$. □

Furthermore, $\mathcal{C}(\sigma)$ can be regarded as a (weak) limit cycle of the SPDI. The following result is a direct consequence of Theorem 3.5 and Theorem 3.26.

Theorem 3.27 *Any viable trajectory in K_σ converges to $\mathcal{C}(\sigma) = \mathsf{Cntr}(K_\sigma)$.* □

3.5.2 STAY Cycles

The controllability kernels of STAY-cycles have stronger limit cycle properties. The following result is a corollary of the previous theorems.

Theorem 3.28 *Let σ be STAY. Then,*
(1) $\mathcal{C}(\sigma)$ is invariant.
(2) There exists a neighborhood K of $\mathcal{C}(\sigma)$ such that any viable trajectory starting in K converges to $\mathcal{C}(\sigma)$.

PROOF.

(1) Suppose that $\mathcal{C}(\sigma)$ is not invariant, then it exists $x \in \mathcal{C}(\sigma)$ and a trajectory ξ starting on x (i.e. $x = \xi(0)$) s.t. ξ is not viable. By definition of $\mathcal{C}(\sigma)$, exists $x' \in \langle l^*, u^* \rangle$ and $t \geq 0$ such that $x' = \xi(t)$. On the other hand, by our assumption of non invariance, it exists $T > t$ such that $\xi(T) \notin \mathcal{C}(\sigma)$, that means $\xi(T) \notin \overline{\mathsf{Pre}}(l^*, u^*)$ and then x' has a successor not in $\langle l^*, u^* \rangle$, contradicting the hypothesis that σ is STAY. Hence $\mathcal{C}(\sigma)$ must be invariant;
(2) It follows directly from Theorem 3.27. $\square$

From the above, the definition of invariance kernel and Theorem 3.19 the following result relating controllability and invariance kernels for STAY cycles follows:

Proposition 3.29 *If $\sigma = e_1 \ldots e_n e_1$ is STAY then $\mathsf{Cntr}(K_\sigma) \subseteq \mathsf{Inv}(K_\sigma)$.* $\square$

Example 13 Fig. 7-(b) shows the viability, controllability and invariance kernels of the SPDI given in Example 1. For any point in the viability kernel of a cycle there exists a trajectory which will converge to its controllability kernel (Theorem 3.27). It is possible to see in the picture that $\mathsf{Cntr}(\cdot) \subset \mathsf{Inv}(.)$ (Proposition 3.29). All the above pictures have been obtained with the toolbox SPeeDI$^+$ [PS06b]. ■

3.5.3 Fix-Points

Here we give an alternative characterization of the controllability kernel of a cycle in SPDI. As in [KV95], we define *fix-points* and *periodic points*.

Definition 3.30 *A point x in e_1 is a* fix-point *iff $x \in \mathsf{Succ}_\sigma(x)$. We call a point $\mathbf{x} \in K_\sigma$ a* periodic point *iff there exists a trajectory segment ξ starting and ending in $\mathbf{x}$, such that $\mathsf{Sig}(\xi)$ is a cyclic shift of σ.*

If $\mathbf{x} \in K_\sigma$ is a periodic point then there exists also an infinite periodic trajectory passing through some $x \in e_1$. The following result characterizes the set of fix-points and of periodic points for SPDIs.

Theorem 3.31 *For SPDIs,*
(1) $\mathcal{C}_\mathcal{D}(\sigma)$ is the set of all the fix-points in e_1.
(2) $\mathcal{C}(\sigma)$ is the set of all the periodic points in K_σ.

PROOF.

(1) Let $\sigma = e_1, \cdots e_k e_1$ be a cycle signature, $\langle L, U\rangle = S_\sigma \cap J_\sigma$ as before and x a fix-point of e_1.
If σ is DIE, trivial.
If σ is STAY, any fix-point of e_1 must be in $\langle l^*, u^*\rangle$, hence $x \in \langle l^*, u^*\rangle$.
If σ is EXIT-BOTH, notice that if x is a fix-point in e_1, then it exists a viable trajectory ξ starting on x such that for all $n > 1$, $x_n = x$, but by Lemma 3.3, $x_n = S_\sigma \cap J_\sigma$, i.e. any fix-point of e_1 must be in $S_\sigma \cap J_\sigma$.
If σ is EXIT-LEFT, from the above results any fix-point must be in $\langle L, U\rangle \cap \langle l^*, u^*\rangle$, hence $x \in \langle L, u^*\rangle$.
If σ is EXIT-RIGHT, as for EXIT-LEFT, we obtain that $x \in \langle l^*, U\rangle$.
(2) Let $\mathbf{x} \in K_\sigma$ be a periodic point, then any trajectory starting on $\mathbf{x}$ must intersect e_1 in a point x that is a fix-point, but by (1), $x \in \mathcal{C}_\mathcal{D}(\sigma)$, then $\mathbf{x} \in \overline{\mathsf{Pre}}(x)$ that implies $\mathbf{x} \in \mathcal{C}(\sigma)$. $\square$

As a direct consequence of the above theorem, the following result holds.

Corollary 3.32 *Given a cyclic signature $\sigma = e_1, \cdots e_k e_1$, all the fix-points in e_1 are included in $\langle L, U\rangle \cap \langle l^*, u^*\rangle$.* $\square$

3.6 Phase Portrait Construction

Let ξ be a trajectory without self-crossings.[4] Recall that ξ is assumed to have an infinite signature. An immediate consequence of [ASY07, Lemma 4.11] is that $\mathsf{Sig}(\xi)$ can be canonically expressed as a sequence of edges and cycles of the form $r_1 s_1^* \ldots r_n s_n^\omega$, with (among others) the following properties:

(1) For all $1 \leq i \leq n$, r_i is a sequence of pairwise different edges, and s_i is a simple cycle.
(2) For all $1 \leq i \neq j \leq n$, r_i and r_j are disjoint, and s_i and s_j are different.
(3) For all $1 \leq i \leq n-1$, s_i is repeated a finite number of times.

[4] A formal definition of "self-crossing" was introduced in [ASY07, Section 3.2].

(4) s_n is repeated forever.

Hence,

Theorem 3.33 *Every trajectory with an infinite signature and which does not have self-crossings converges to the controllability kernel of some simple edge-cycle.*

PROOF. This follows directly from the above properties and from Theorem 3.27. □

We now define the notions of the *limit set* and the *limit points* of a given trajectory.

Definition 3.34 *Given a trajectory ξ such that $\xi(0) = \mathbf{x}$, a point $\mathbf{y}$ is a* limit point *of $\mathbf{x}$ if there is a sequence $t_0, t_1, t_2, \ldots$ such that $t_n \to \infty$ and $\lim_{n\to\infty} \xi(t_n) = \mathbf{y}$. The set of all the limits points of $\mathbf{x}$ is its* limit set*, $\mathrm{limit}(\xi)$.*

Corollary 3.35 *(1) Any trajectory ξ with infinite signature without self-crossings is such that its limit set $\mathrm{limit}(\xi)$ is a subset of the controllability kernel $\mathcal{C}(\sigma)$ of a simple edge-cycle σ.*

(2) Any point in $\mathcal{C}(\sigma)$ is a limit point of a trajectory ξ with infinite signature without self-crossings

PROOF. The result is a direct consequence of Theorem 3.33. □

A sound algorithm to compute all the above mentioned phase portrait objects is obtained directly from theorems 3.5, 3.9, 3.19 and 3.22.

Example 14 Fig. 9 shows an SPDI with two edge cycles $\sigma_1 = e_1, \cdots, e_8, e_1$ and $\sigma_2 = e_{10}, \cdots, e_{15}, e_{10}$, and their respective controllability kernels. Every simple trajectory eventually arrives (or converges) to one of the two limit sets and rotates therein forever. ■

The phase portrait plays an important role on the optimization of the reachability algorithm for SPDIs [PS06c], and to obtain a compositional parallel reachability algorithm [PS06a].

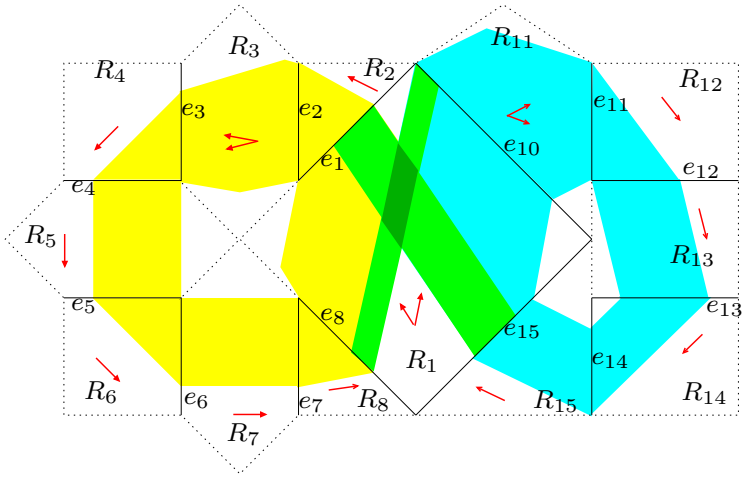


Fig. 9. Another SPDI and its "phase-portrait".

4 SPeeDI$^+$

In this section we discuss some issues related to the tool SPeeDI$^+$, which extends SPeeDI (implementing the reachability algorithm for SPDIs [APSY02,ASY07]) with the computation of the kernels introduced in the previous section.

4.1 Description of the Tool

The proof of the decidability of reachability questions for SPDIs given in [ASY07] is a constructive one, giving (i) a reduction of the infinite number of possible paths to be analyzed for a given reachability question to a finite set of abstract signatures; and (ii) a technique for calculating the effect of following an abstract signature. This approach lies at the core of SPeeDI$^+$ to answer reachability questions for a given SPDI. The resulting algorithm is thus essentially a depth-first search on the SPDI graph (but abstracting away loops in terms of the abstract signatures). Apart from the reachability algorithm, SPeeDI$^+$ comes with a number of other tools and utilities to visualize and analyze SPDIs:

Visualization aids: To help visualize systems, the tool can generate graphical representations of the SPDI, and particular trajectories and signatures within it.

Information gathering: SPeeDI$^+$ calculates edge-to-edge successor function composition and enlist signatures going from one edge to another.

Verification: The most important facet of the tool suite is that of verification. At the lowest level, the user may request whether, given a signature (with a possibly restricted initial and final edge), it is a feasible one or not. At a more general, and useful level, the user may simply give a restricted ini-

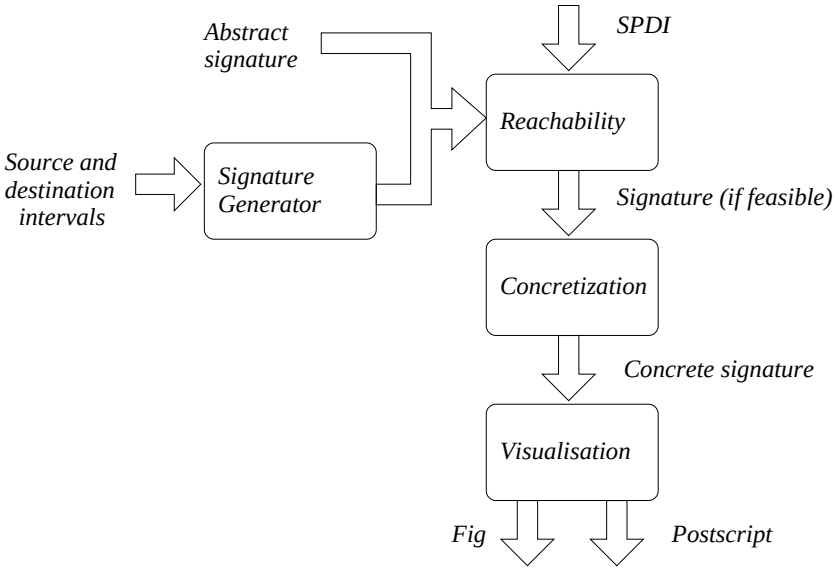


Fig. 10. Workflow of the tool.

tial edge and restricted final edge, and the tool attempts to answer whether the latter is reachable from the former.

Phase portrait: In SPeeDI$^+$ the user can also extract information about the phase portrait of an SPDI and visualize it. SPeeDI$^+$ allows the calculation on the viability, controllability and invariance kernels. Figures 5 and 7 have all been automatically generated the tool.

Trace generation: Whenever reachability succeeds SPeeDI$^+$ generates stripes of feasible trajectories using different strategies and graphical representation of them.

Exact arithmetic: An offshoot of SPeeDI$^+$ is a version using an exact representation of numbers to avoid rounding errors by using the Haskell's native rational number library.

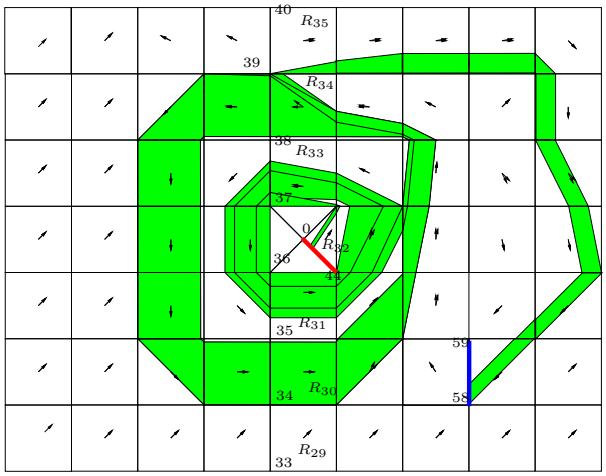
A typical usage sequence of the tool suite, concerning reachability analysis, is captured in Fig. 10.

Fig. 11 illustrates a typical session of the tool on an example SPDI composed of 63 regions. The left part of the diagram shows selected portions of the input file, defining vectors, named points on the x-y plane, and regions (as sequences of point names, and pairs of differential inclusion vectors). The lower right-hand panel shows the signature generated by the tool `reachable` which satisfies the user's demand. The signature has two loops which are expressed with the star symbol. A trace is then generated from the signature using `simsig`. It traverses three times the first loop and two times the second one. The graphical representation of the SPDI and the trace is generated automatically using `simsig2fig`. The execution time for this example is a few seconds.

Input file

```
Points:
0. 0.0, 0.0
* ...
33. -5.0, -35.0
34. -5.0, -25.0
35. -5.0, -15.0
36. -5.0, -5.0
37. -5.0, 5.0
38. -5.0, 15.0
39. -5.0, 25.0
* ...
Vectors:
* ...
v3. -1,0.1833333333
v8. 1,0
v9. 1,1
v12. 1, 1.5
v20. -1, 0.001
v22. 1,-0.001
v25. -1,0.7
v28. 1, 0.001
*...
Regions:
* ...
* R29
33 ? 41 ! 42 ! 34 ? 33, v9, v9
* R30
34 ! 42 ! 43 ? 35 ? 34, v22, v22
* R31
35 ? 36 ? 0 ! 44 ! 43 ! 35, v8, v8
* R32
44 ! 45 ! 0 ? 44, v12, v12
* R33
0 ? 45 ? 46 ! 38 ! 37 ! 0, v3, v20
* R34
38 ? 46 ? 47 ! 39 ! 38, v25, v20
* ...
```

Generated Figure



Session log

```
Generating and trying signatures from edge 0-44 to 58-59
Starting interval:[1.0,2.0] Finishing interval:[0.0,10.0]

(0-44,45-44)  (45-53,45-46,37-38,...,36-35,44-43,44-52)*
(53-52,53-61,54-62,54-55,46-47)(38-39,..., 46-47)* (39-47,
...,67-59,58-59)   <REACHABLE>
```

Fig. 11. Example.

Fig. 12 for a short description of the different utilities of the tool. A more detailed explanation can be found in [Sch02, chapter 8] and the appendices of the same work.

4.2 Implementation Issues

SPeeDI$^+$ was implemented in Haskell [Jon03], a general-purpose, lazy, functional language [BW88,FH88]. Despite the fact that functional languages, especially lazy ones, have a rather bad reputation regarding performance (see for example [LNSW01] for a report on the experiences of writing verification tools in functional languages), we found that the performance we obtained was more than adequate for the magnitude of examples we had in mind. Furthermore, we feel that with the gain in the level of abstraction of the code, we have much more confidence in the correctness of our tool than had we used a lower level language. We found laziness particularly useful in separating control and data considerations. Quite frequently, optimizations dictated that we evaluate

Analysis:

getmafs Given an SPDI and a concrete signature, calculate the intermediate TAMFS along it.

looptype Given an SPDI and a loop, analyze the type and behavior of the loop.

showsigs Given an SPDI, a source and destination edge, list the abstract signatures that SPeeDI$^+$ will analyze for reachability.

trysig Given an SPDI and an abstract signature, apply the signature to calculate the behavior on the SPDI starting from a given part of the starting edge.

reachable Given an SPDI, an interval on a source edge and an interval on a destination edge, answers whether the destination is reachable from the source.

simsig Given an SPDI and an abstract signature, produce a corresponding feasible concrete signature (provided that the original abstract signature was feasible) through forward or backward analysis.

viability/invariance/controllability Given an SPDI and a loop, calculate the viability, invariance or controllability kernel for that loop (respectively).

Visualization:

spdi2ps Visualization tool, transforming a given SPDI into a Postscript image.

sig2path/sig2fig/sig2ps Given an SPDI and a concrete or abstract signature produce a graphical representation of the signature.

simsig2fig Given an SPDI and an abstract signature, produce a graphical visualization of a corresponding feasible concrete signature.

drawkernels Given an SPDI, produce a graphical representation of all the kernels in that SPDI.

Fig. 12. Description of the utilities.

certain complex expressions at most once, if at all. In most strict languages, this would have led to complex code which mixes data computations (which use the values of the expressions) with control computation (to decide whether this is the first time we are using the expression and, if so, evaluate it). Thanks to shared expressions and laziness, all this came for free — resulting in cleaner code, where the complex control is not done by the programmer.

SPeeDI$^+$ consists of the utilities described in the previous section plus a library

for intervals, vectors and truncated affine multi-valued functions.

The tools are available in two versions — one which uses floating point numbers, and one with exact arithmetic, which uses Haskell's rational number library. Obviously, the performance using exact arithmetic degrades the performance, but the fact that loop behaviors are analyzed and calculated in one go, thus limiting the length of the traces analyzed, meaning that the degradation in performance is reasonable.

4.2.1 Input Language

As shown in Fig. 11, the input file consists of three parts: description of points, description of vectors and description of regions.

4.2.2 SPDI Validation

Given an SPDI, SPeeDI$^+$ performs the following consistency checks:

(1) Regions must be well defined polygons;
(2) Vectors corresponding to a region differential inclusion must respect the fact that the `<a-vector>` corresponds to **a** and `<b-vector>` corresponds to **b**, such that **b** is situated in the counterclockwise direction of **a**;
(3) Each region is *good* (i.e., every edge is an entry or exit, but no both).

4.2.3 Data structures

An SPDI $\mathcal{H}$ can be represented as a graph $\mathcal{G}_{\mathcal{H}}$. Indeed, given $\mathcal{H}$, we can define a graph $\mathcal{G}_{\mathcal{H}}$ where nodes correspond to edges of $\mathcal{H}$ and such that there exists an arc from one node to another if there exists a trajectory segment from the first edge to the second one without traversing any other edge. $\mathcal{G}_{\mathcal{H}}$ is defined in Haskell as a list of edges identifiers and a transition function that associate to each pair of edges its TAMF if it exists or "Nothing" otherwise.

The graph is defined then in SPeeDI$^+$ as:

```
data Graph =                                                    (1)
  Graph {                                                       (2)
    transitionFunction :: EdgeId -> EdgeId -> Maybe TAMF,       (3)
    domain :: [EdgeId]                                          (4)
  }
```

The `Graph` datatype is a record consisting of (i) the transition function of the SPDI represented as a function, which given two edges, returns the TAMF

between the two edges if a transition is possible (see line 3); and (ii) a list of the nodes of the graph (in the field `domain` on line 4). Note that the transition function is a total one, since we return `Maybe TAMF`, to enable us to return `Nothing` when a direct transition is not possible, and `Just f` when a transition is possible with TAMF `f`. Note that underneath this clean transition relation description lies a standard efficient two dimensional array access.

4.2.4 Generation of Types of Signatures

Given two intervals $I_0 \in e_0$ and $I_f \in e_f$ SPeeDI$^+$ generates all the types of signatures $r_1, s_1, \cdots, r_n, s_n, r_{n+1}$ that satisfy the following properties:

(1) $first(r_1) = e_0$ and $last(r_{n+1}) = e_f$;
(2) For every $1 \leq i \neq j \leq n+1$, r_i is a path on the graph;
(3) For every $1 \leq i \neq j \leq n$, s_i is a simple loop on the graph;
(4) For every $1 \leq i \neq j \leq n+1$, r_i and r_j are disjoint;
(5) For every $1 \leq i \neq j \leq n$, s_i and s_j are different;
(6) For every $1 \leq i < n$, s_i and r_{i+1} are disjoint;
(7) For every $1 \leq i \leq n$, s_i is never a suffix of r_i.

The first property guarantees that only signatures from the initial edge to the final one are generated. The other properties ensure that there is only a finite number of types of signatures to be considered, which is one of the key observations for guaranteeing termination. See [Sch02] for the theoretical justification of these properties.

4.2.5 Pre-Optimizations

In this section we describe the optimizations done in order to minimize the graph (in terms of number of transitions and states) analyzed for reachability. The following optimizations are implemented on the current version of the tool.

(1) We eliminate some types of infeasible signatures: we only consider trajectories that have a non-empty TAMF. It can be the case that there is no trajectory segment from one edge to other of the same region even though there is a *path* on the graph. This is detected on SPeeDI$^+$ checking that the `transitionFunction` for the two given edges gives a non-empty TAMF when applied to the whole source edge.
(2) When considering reachability from edge e to edge e' clearly source nodes of the graph cannot be reachable from e (except from e itself). We recursively eliminate all the source nodes of the graph different from the node src corresponding to e.

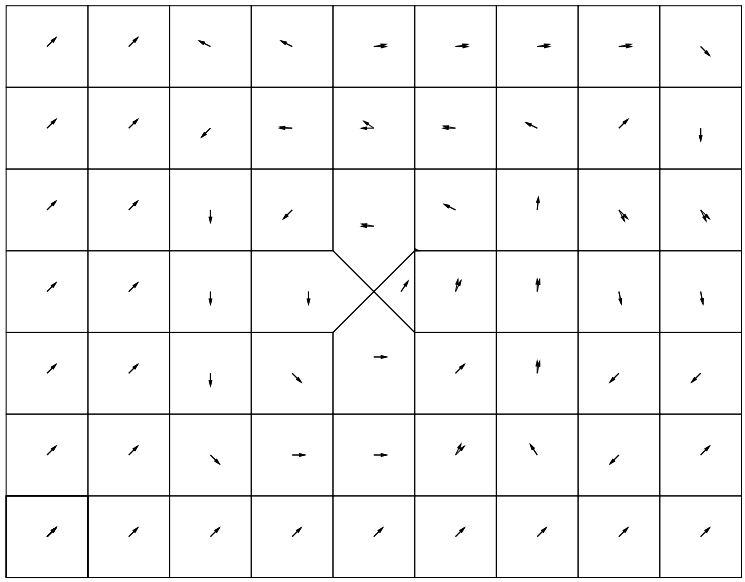


Fig. 13. The SPDI of Example 4.3.

(3) As in the previous point, we do the same for the sink nodes and the destination node *dst*, corresponding to e'.

4.2.6 Verification-Time Optimizations

We now describe the optimizations done in order to minimize the number of types of signatures analyzed for reachability.

(1) A number of properties of SPDIs (as proved in [Sch02]) are used to reduce the signatures explored. This includes the properties that, for instance, loops may not appear more than once in a signature (since whenever a concrete path with a repeated loop, there exists another path with the same source and destination but with no repeated loops), and that the paths between the loops pass through no edge more than once. These properties are part of the constraints given in section 4.2.4.
(2) The generation of the signatures is done concurrently with their analysis — it carries along the analysis of the application of the generated signature. As soon as a signature is not feasible (when applying the TAMF of the partial signature to the initial interval gives an empty interval as a result), it is not explored any further. This drastically reduces the signatures generated.

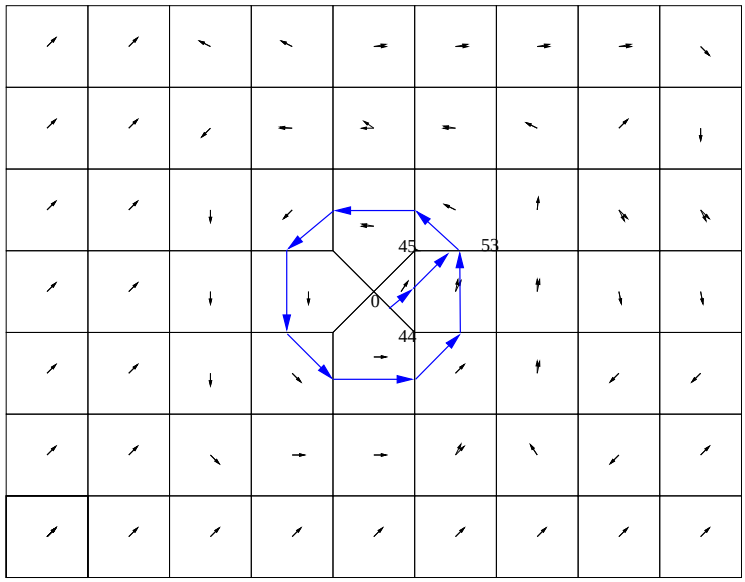


Fig. 14. A path manually analyzed using SPeeDI$^+$.

4.3 Example

In this section we present an example of an SPDI analyzed using the different utilities explained before. The SPDI we are going to consider has 63 regions and 162 edges as shown in Fig. 13. Note that all the figures shown in this section have been automatically created using SPeeDI$^+$ visualization tools, with the only additions being annotations used to refer to edges or regions.

Internally, SPeeDI$^+$ calculates composed TAMFs on adjacent regions along a given path. It is usually useful to see such composed TAMFs to understand the behavior of a path in an SPDI better. This is especially useful in the case of loops. Similarly, for a manual qualitative analysis of an SPDI, it is useful to be able to calculate the qualitative behavior of a cycle (see section 2.3). The tools `getmafs` and `looptype` can be used for these purposes, obtaining information as in the following example (for the path, which includes a cycle, as shown in Fig. 14):

```
The requested AMFs:
From edge 84 (0-44) to edge 86 (44-45):
AMF is [1.7677669529663687x-2.5, 1.7677669529663687x-2.5]
   (accumulated AMF is
       [1.7677669529663687x-2.5, 1.7677669529663687x-2.5])
From edge 86 (44-45) to edge 103 (45-53):
AMF is [0.2x, 0.5x]
   (accumulated AMF is
       [0.35355339059327373x-0.5, 0.8838834764831843x-1.25])
```

```
From edge 103 (45-53) to edge 88 (45-46):
AMF is [0.5x, 0.5x]
   (accumulated AMF is
       [0.17677669529663687x-0.25, 0.44194173824159216x-0.625])
...
Loop type: Exit right
```

Before proceeding directly to reachability analysis, we can get a list of all feasible types of signatures from one edge to another (on the symbolic graph) using the `getsigs` tool. For example, the tool lists 36 feasible types of signatures on the example shown in Fig. 11 from edge 0-44 to edge 58-59.

The type of signatures listed by the showsigs are the candidates for the reachability question: Is $I_f \subseteq e_f$ reachable from $I_0 \subseteq e_0$? Individual types of signatures can be checked for actual feasibility using the `trysig` tool, or one can check all possible types of signatures (hence determining reachability), using the `reachable` tool:

```
> reachable example.spdi [1,2] [0,10] 0-44 58-59

  SPDI Reachability Tool v2

  REACHABLE
  0-44,45-44 (45-53,45-46,37-38,37-29,36-28,36-35,44-43,44-52)*
  53-52,53-61,54-62,54-55,46-47 (38-39,30-31,30-22,29-21,28-20,
  27-19,27-26,35-34,43-42,43-51,52-51,52-60,53-61,54-62,54-55,
  46-47)* 39-47,48-47,56-55,64-63,72-71,79-71,78-70,77-69,76-68,
  67-68,67-59,58-59
```

Finally, given a type of signature, we usually desire to obtain an actual concrete signature (the corresponding signature with an unfolding of the cycles), which can be done using the `simsig` tool (or the `simsig2fig` tool which produces a graphical visualization of the path). Applying the tool to the path identified by the reachability run given above, one obtains the diagram shown in Fig. 11.

4.4 Comparison with HyTech

While SPeeDI$^+$ is, as far as we know, the only verification tool for hybrid systems implementing a decision algorithm (with the exception of timed automata), it is interesting to compare it to "semi-algorithmic" hybrid system verification tools such as HyTech [HPHHt97,HHW95]. HyTech is a tool capable of treating hybrid linear systems of any dimension, making it much more general than SPeeDI$^+$, which is limited to two-dimensional systems without

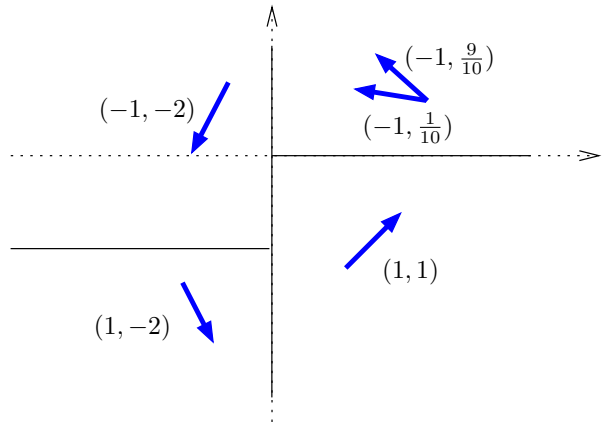


Fig. 15. SPDI of Example 4.4.1.

resets. On the other hand, SPeeDI$^+$ implements acceleration techniques (based on the resolution of fix-point equations) which yield a complete decision procedure for SPDIs. Also, SPeeDI$^+$ does not handle arbitrary polyhedra, but only polygons and line segments. For these reasons, comparing the performance of the two tools is meaningless and no fair benchmarking is really possible. However, we have explored a simple illustrative example.

4.4.1 Example

Consider the SPDI defined as follows (see Fig. 15) with $I_0 \equiv (y = 0 \wedge x \in [3; 4])$ as initial region:

Region	*Defining conditions*	*Vector*
R_0	$(x \geq 0) \wedge (y \geq 0)$	$\mathbf{a} = (-1, \frac{9}{10}), \mathbf{b} = (-1, \frac{1}{10})$
R_1	$(x \leq 0) \wedge (y \geq -10)$	$\mathbf{a} = \mathbf{b} = (-1, -2)$
R_2	$(x \leq 0) \wedge (y \leq -10)$	$\mathbf{a} = \mathbf{b} = (1, -2)$
R_3	$(x \geq 0) \wedge (y \leq 0)$	$\mathbf{a} = \mathbf{b} = (1, 1)$

We consider different final points x_f on the x axis and try to answer the question: Is x_f reachable from I_0?

The experimental results are given in Table 1.[5]

All the results above of HyTech were using the `reach backward` command. The `reach forward` gives "`Library overflow error in multiplication`" in all the cases.

[5] The column corresponding to SPeeDI$^+$ uses exact arithmetic.

Final Point	*HyTech*	*SPeeDI*	SPeeDI$^+$	*Reachable*
199	`overflow`	`0.05 sec`	`0.07 sec`	Yes
200	`overflow`	`0.05 sec`	`0.07 sec`	No
201	`overflow`	`0.01 sec`	`0.03 sec`	No
210	`overflow`	`0.05 sec`	`0.07 sec`	No
5	`0.04 sec`	`0.05 sec`	`0.07 sec`	No
20	`0.07 sec`	`0.05 sec`	`0.07 sec`	No
$\frac{200}{9}$	`0.10 sec`	`0.05 sec`	`0.07 sec`	Yes
$\frac{201}{9}$	`overflow`	`0.03 sec`	`0.05 sec`	Yes
$\frac{199}{9}$	`0.07 sec`	`0.04 sec`	`0.07 sec`	Yes
$\frac{1}{2}$	`0.06 sec`	`0.05 sec`	`0.08 sec`	No

Table 1
Comparison with HyTech.

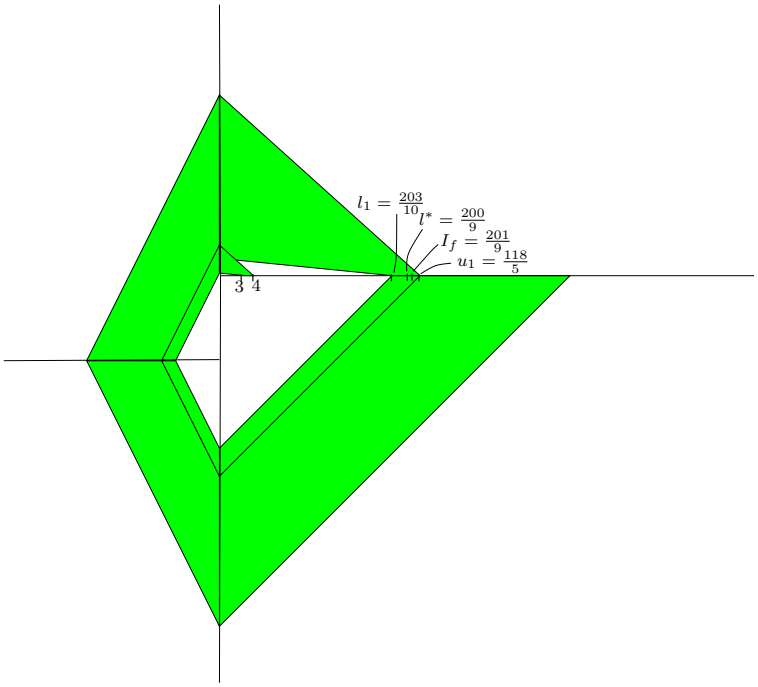


Fig. 16. Simulation of reachability for $x_f = \frac{201}{9}$.

Fig. 16 shows the simulation of the case whenever $x_f = \frac{201}{9}$. In the picture one can see that starting from the initial interval I_0 the system spirals anti-clockwise. The intersection of the spiral with the x-axis converges to the "fix-point interval" $I^* = (\frac{200}{9}; 200)$. SPeeDI$^+$ in fact computes the interval I^*, and whenever $x_f \in I^*$ it gives immediately the positive answer to the reachabil-

ity question. If $x_f \geq 200$ SPeeDI$^+$ says "no". The only case when it really computes successors is when x_f lies between I_0 and I^*.

Notice that the problems with HyTech are mainly whenever the final point I_f is close to the fix-points ($l^* = \frac{200}{9}$ and $u^* = 200$), and also whenever I_f is located in between the fix-points or when $x_f \geq u^*$.

We summarize here a number of qualitative conclusions taken from the above experiments, and others not presented in this paper, comparing HyTech with SPeeDI$^+$:

- It is well known that since HyTech uses exact rational arithmetic, it can easily run into overflow problems. This is particularly an issue when the path to the target passes through a large number of regions. This makes verification of non-trivial sized SPDIs (e.g. the one in Fig. 11) impossible, even though they are still possible using SPeeDI$^+$ with exact arithmetic.
- In the case of loops, SPeeDI$^+$ calculates the limit interval without repeatedly iterating the loop. It makes use of this interval to accelerate the reachability analysis, avoiding time consuming loop traversals. In contrast, HyTech performs these iterations. Following the loops explicitly, easily leads to overflow problems, and, more seriously, in certain (even simple) configurations, this analysis never terminates. The acceleration enables SPeeDI$^+$ to work even when using exact arithmetic, since the length of paths explored is much lower than had these loops to be unfolded.

While the first issue is limited to HyTech, the second is inherent to any tool based on non-accelerated reachability analysis. On examples which HyTech can handle, the two tools take approximately the same amount of time (a fraction of a second) to reach the result. SPeeDI$^+$, however, can handle much larger (planar) examples.

5 Concluding Remarks

We have first defined viability, controllability and invariance kernels as well as semi-separatrices for SPDIs and presented non-iterative algorithms to calculate them. These objects are not merely mathematical curiosity. It turns out that they can be used for optimizing the reachability algorithm [PS06c] and as basis for a compositional parallel algorithm for reachability analysis [PS06a].

We have presented a prototype tool for solving the reachability problem for the class of polygonal differential inclusions. The tool implements the algorithm presented in [ASY07] which is based on the analysis of a finite number of qualitative behaviors generated by a discrete dynamical system characterized

by positive affine Poincaré maps. Since the number of such behaviors may be very large, the tool uses several powerful heuristics that exploit the topological properties of planar trajectories for considerably reducing the set of actually explored signatures. When reachability is successful, the tool outputs a visual representation of the stripe of trajectories that go from the initial point (edge, polygon) to the final one.

We have also presented SPeeDI$^+$, an extension of SPeeDI with the computation and visualization of the different phase portrait objects presented in this paper. Regarding complexity, the critical part of the algorithm consists in counting all feasible types of signatures, which has a double exponential upper-bound on the size of the SPDI.[6] Though we cannot provide bounds on the required number of steps for analyzing simple cycles, our experiments show that our algorithm performs very well in practice. The main reason is that the analysis of most simple loops can be accelerated, i.e., the limit can be computed without iterating (see [ASY07] for more details). Moreover, the computation of the phase portrait itself does not add extra complexity, as most of the required information is already computed by the reachability algorithm.

Our work is obviously restricted to planar systems, which enables us to compute these kernels exactly. In higher dimensions and hybrid systems with higher complexity, calculation of kernels is not computable. Other related work is thus based on calculations of approximations of these kernels (e.g., [ALQ$^+$01b,ALQ$^+$01a,SP02]).

References

[AC84] J-P. Aubin and Arrigo Cellina. *Differential Inclusions*. Number 264 in A Series of Comprehensive Studies in Mathematics. Springer-Verlag, 1984.

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

[ALQ$^+$01a] J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Towards a viability theory for hybrid systems. In *European Control Conference*, 2001.

[ALQ$^+$01b] J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Viability and invariance kernels of impulse differential inclusions. In *Conference on Decision and Control*, volume 40 of *IEEE*, pages 340–345, 2001.

[6] This follows from [ASY07, Lemma 4.11].

[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. Springer-Verlag, 2002.

[ASY01] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In *HSCC'2001*, number 2034 in LNCS, pages 89–104. Springer-Verlag, 2001.

[ASY02] E. Asarin, G. Schneider, and S. Yovine. Towards computing phase portraits of polygonal differential inclusions. In *HSCC'02*, pages 49–61. LNCS 2289, Springer, 2002.

[ASY07] E. Asarin, G. Schneider, and S. Yovine. Algorithmic Analysis of Polygonal Hybrid Systems. Part I: Reachability. *Theoretical Computer Science*, 379(1-2):231–265, 2007.

[Aub90] J.-P. Aubin. A survey on viability theory. *SIAM J. Control and Optimization*, 28(4):749–789, July 1990.

[BW88] R. Bird and P. Wadler. *Introduction to Functional Programming*. Prentice Hall International, New York, 1988.

[DV95] A. Deshpande and P. Varaiya. Viable control of hybrid systems. In *Hybrid Systems II*, number 999 in LNCS, pages 128–147, 1995.

[FH88] A.J. Field and P.G. Harrison. *Functional Programming*. Addison Wesley, Reading, Massachusetts, 1988.

[HHW95] T.A. Henzinger, P-H. Ho, and H. Wong-Toi. Hytech: The next generation. In *Proc. IEEE Real-Time Systems Symposium RTSS'95*, 1995.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *STOC'95*, 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), 1997.

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

[Jon03] S. Peyton Jones. *Haskell 98 language and libraries: the Revised Report*. Cambridge University Press, 2003.

[KdB01] P. Kowalczyk and M. di Bernardo. On a novel class of bifurcations in hybrid dynamical systems. In *HSCC'01*, number 2034 in LNCS, pages 361–374. Springer, 2001.

[KV95] M. Kourjanski and P. Varaiya. Stability of hybrid systems. In *Hybrid Systems III*, number 1066 in LNCS, pages 413–423. Springer, 1995.

[LNSW01] M. Leucker, T. Noll, P. Stevens, and M. Weber. Functional programming languages for verification tools: Experiences with ML and Haskell. In *Proceedings of SFPW'01*, 2001.

[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, 2001.

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

[MS00] A. Matveev and A. Savkin. *Qualitative theory of hybrid dynamical systems*. Birkhäuser Boston, 2000.

[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. Computation and visualization of phase portraits for model checking SPDI. 2006.

[PS06c] 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.

[Sch02] G. Schneider. *Algorithmic Analysis of Polygonal Hybrid Systems*. PhD thesis, VERIMAG – UJF, Grenoble, France, 2002.

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

[SJSL00] S. Simić, K. Johansson, S. Sastry, and J. Lygeros. Towards a geometric theory of hybrid systems. In *HSCC'00*, number 1790 in LNCS, pages 421–436. Springer, 2000.

[SP02] P. Saint-Pierre. Hybrid kernels and capture basins for impulse constrained systems. In *HSCC'02*, volume 2289 of *LNCS*. Springer-Verlag, 2002.