Model Checking Polygonal Differential Inclusions Using Invariance Kernels*

Gordon J. Pace[1] and Gerardo Schneider[2]**

[1] Department of Computer Science and AI, University of Malta.
[2] Department of Information Technology, Uppsala University, Sweden.
{gordon.pace@um.edu.mt; gerardos@it.uu.se}

Abstract. Polygonal hybrid systems are a subclass of planar hybrid automata which can be represented by piecewise constant differential inclusions. Here, we identify and compute an important object of such systems' phase portrait, namely *invariance kernels*. An *invariant set* is a set of initial points of trajectories which keep rotating in a cycle forever and the *invariance kernel* is the largest of such sets. We show that this kernel is a non-convex polygon and we give a non-iterative algorithm for computing the coordinates of its vertices and edges. Moreover, we present a breadth-first search algorithm for solving the reachability problem for such systems. Invariance kernels play an important role in the algorithm.

1 Introduction

A *hybrid system* is a system where both continuous and discrete behaviors interact with each other. A typical example is given by a discrete program that interacts with (controls, monitors, supervises) a continuous physical environment. In the last decade many (un)decidability results for a variety of problems concerning classes of hybrid systems have been given [ACH+95,ABDM00,BT00], [DM98,GM99,KV00]. One of the main research areas in hybrid systems is reachability analysis. Most of the proved decidability results are based on the existence of a finite and computable partition of the state space into classes of states which are equivalent with respect to reachability. This is the case for timed automata [AD94], and classes of rectangular automata [HKPV95] and hybrid automata with linear vector fields [LPY99]. For some particular classes of two-dimensional dynamical systems a *geometrical* method, which relies on the analysis of topological properties of the plane, has been developed. This approach has been proposed in [MP93]. There, it is shown that the reachability problem for two-dimensional systems with piece-wise constant derivatives (PCDs) is decidable. This result has been extended in [CV96] for planar piece-wise Hamiltonian systems and in [ASY01] for polygonal hybrid systems, a class of nondeterministic systems that correspond to piecewise constant differential inclusions on the

* Published in VMCAI'04 (Fifth International Conference on Verification, Model Checking and Abstract Interpretation). LNCS Nro. 2937. Venice, Italy, p.110–121. January 11-13, 2004.
** Supported by European project ADVANCE, Contract No. IST–1999–29082.

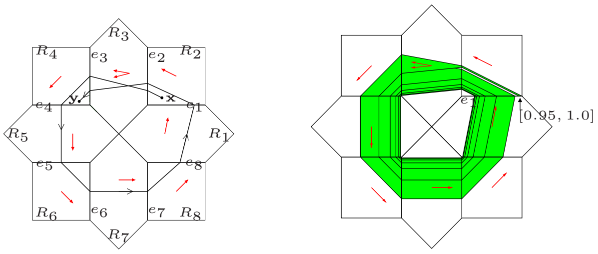

Fig. 1. (a) An SPDI and its trajectory segment; (b) Reachability analysis of the SPDI

plane, see Fig. 1(a). For historical reasons we call such a system an SPDI [Sch02]. In [AMP95] it has been shown that the reachability problem for PCDs is undecidable for dimensions higher than two.

Another important issue in the analysis of a (hybrid) dynamical system is the study of its qualitative behavior, namely the construction of its *phase portrait*. Typical questions one may want to answer include: "does every trajectory (except for the equilibrium point at the origin) converge to a limit cycle?", and "what is the biggest set such that any point on it is reachable from any other point on the set?". There are very few results on the qualitative properties of trajectories of hybrid systems [ASY02,Aub01,DV95,KV96,KdB01,MS00,SJSL00]. In particular, the question of defining and constructing phase portraits of hybrid systems has not been directly addressed except in [MS00], where phase portraits of deterministic systems with piecewise constant derivatives are explored and in [ASY02] where viability and controllability kernels for polygonal differential inclusion systems have been computed.

In this paper we show how to compute another important object of phase portraits of SPDIs, namely the *invariance kernel*. In general, an *invariant set* is a set of points such that every trajectory starting in the set remains within the set forever and the *invariance kernel* is the largest of such sets. We show that, for SPDIs, this kernel for a particular cycle is a non-convex polygon and we give a non-iterative algorithm for computing the coordinates of its vertices and edges[3]. Clearly, the invariance kernel provides useful insight about the behavior of the SPDI around simple cycles. Furthermore, we present an alternative algorithm to the one presented in [ASY01] for solving the reachability problem for SPDIs. This algorithm is a breadth-first search, in the spirit of traditional model checking algorithms. Invariance kernels play a key role in the algorithm.

[3] Notice that since SPDIs are partially defined over the plane, their invariance kernels are in general different from the whole plane.

2 Preliminaries

2.1 Truncated affine multivalued functions

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. In what follows we will consider only well-formed intervals, i.e. $\langle l, u \rangle$ is an interval iff $l \leq u$. 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 $F^{-1}(x) = \{y \mid x \in F(y)\}$. It is not difficult to show that $F^{-1} = \langle f_u^{-1}, f_l^{-1} \rangle$. The *universal inverse* of F is defined by $\tilde{F}^{-1}(I) = I'$ iff I' is the greatest non-empty interval such that for all $x \in I'$, $F(x) \subseteq I$. 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 $\mathcal{F}^{-1}(I) = F^{-1}(I \cap J) \cap S$. 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 $J = \mathsf{Im}\mathcal{F} = \mathcal{F}(S)$. In what follows we only consider normalized TAMFs. The *universal inverse* of $\mathcal{F}$ is defined by $\tilde{\mathcal{F}}^{-1}(I) = I'$ iff I' is the greatest non-empty interval such that for all $x \in I'$, $F(x) \subseteq I$ and $F(x) = \mathcal{F}(x)$.

TAMFs are closed under composition [ASY01]:

Theorem 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 $J = J_2 \cap F_2(J_1 \cap S_2)$.

2.2 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 $\dot{\mathbf{x}} \in \phi(P)$ for $\mathbf{x} \in P$.

Let $E(P)$ be the set of edges of P. We say that $e \in E(P)$ 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, $E(P) = In(P) \cup Out(P)$.*

Example 1. Consider the SPDI illustrated in Fig. 1(a). For each region R_i, $1 \leq i \leq 8$, there is a pair of vectors $(\mathbf{a}_i, \mathbf{b}_i)$, where: $\mathbf{a}_1 = \mathbf{b}_1 = (1, 5)$, $\mathbf{a}_2 = \mathbf{b}_2 = (-1, \frac{1}{2})$, $\mathbf{a}_3 = (-1, \frac{11}{60})$ and $\mathbf{b}_3 = (-1, -\frac{1}{10})$, $\mathbf{a}_4 = \mathbf{b}_4 = (-1, -1)$, $\mathbf{a}_5 = \mathbf{b}_5 = (0, -1)$, $\mathbf{a}_6 = \mathbf{b}_6 = (1, -1)$, $\mathbf{a}_7 = \mathbf{b}_7 = (1, 0)$, $\mathbf{a}_8 = \mathbf{b}_8 = (1, 1)$. ■

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 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*.

Assumption 2 *We will only consider trajectories with infinite signatures.*

2.3 Successors and predecessors

Given an SPDI, we fix a one-dimensional coordinate system on each edge to represent points laying on edges [ASY01]. 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 $P \in \mathbb{P}$, $e \in \mathit{In}(P)$ and $e' \in \mathit{Out}(P)$. For $I \subseteq e$, $\mathsf{Succ}_{ee'}(I)$ 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., $\xi(0) \in I \wedge \xi(t) \in e' \wedge \mathsf{Sig}(\xi) = ee'$). We have shown in [ASY01] that $\mathsf{Succ}_{ee'}$ is a TAMF[4].

Example 2. Let $e_1, \ldots, e_8$ be as in Fig. 1(a) and $I = [l, u]$. We assume a one-dimensional coordinate system such that $e_i = S_i = J_i = (0, 1)$. We have that:

$$F_{e_1e_2}(I) = \left[\frac{l}{2}, \frac{u}{2}\right] \quad F_{e_2e_3}(I) = \left[l - \frac{1}{10}, u + \frac{11}{60}\right]$$
$$F_{e_ie_{i+1}}(I) = I \quad 3 \leq i \leq 7 \quad F_{e_8e_1}(I) = \left[l + \frac{1}{5}, u + \frac{1}{5}\right]$$

with $\mathsf{Succ}_{e_ie_{i+1}}(I) = F_{e_ie_{i+1}}(I \cap S_i) \cap J_{i+1}$, for $1 \leq i \leq 7$, and $\mathsf{Succ}_{e_8e_1}(I) = F_{e_8e_1}(I \cap S_8) \cap J_1$. ■

Given a sequence $w = e_1, e_2, \ldots, e_n$, Theorem 1 implies that the successor of I along w defined as $\mathsf{Succ}_w(I) = \mathsf{Succ}_{e_{n-1}e_n} \circ \ldots \circ \mathsf{Succ}_{e_1e_2}(I)$ is a TAMF.

Example 3. Let $\sigma = e_1 \cdots e_8 e_1$. We have that $\mathsf{Succ}_\sigma(I) = F(I \cap S) \cap J$, where:

$$F(I) = \left[\frac{l}{2} + \frac{1}{10}, \frac{u}{2} + \frac{23}{60}\right]$$

$S = (0, 1)$ and $J = (\frac{1}{5}, \frac{53}{60})$ are computed using Theorem 1. ■

For $I \subseteq e'$, $\mathsf{Pre}_{ee'}(I)$ is the set of points in e that can reach a point in I by a trajectory segment in P. We have that[ASY01]: $\mathsf{Pre}_{ee'} = \mathsf{Succ}_{ee'}^{-1}$ and $\mathsf{Pre}_\sigma = \mathsf{Succ}_\sigma^{-1}$.

[4] In [ASY01] we explain how to choose the positive direction on every edge in order to guarantee positive coefficients in the TAMF.

Example 4. Let $\sigma = e_1 \dots e_8 e_1$ be as in Fig. 1(a) and $I = [l, u]$. We have that $\mathsf{Pre}_{e_i e_{i+1}}(I) = F^{-1}_{e_i e_{i+1}}(I \cap J_{i+1}) \cap S_i$, for $1 \le i \le 7$, and $\mathsf{Pre}_{e_8 e_1}(I) = F^{-1}_{e_8 e_1}(I \cap J_1) \cap S_8$, where:

$$F^{-1}_{e_1 e_2}(I) = [2l, 2u] \quad F^{-1}_{e_2 e_3}(I) = \left[l - \tfrac{11}{60}, u + \tfrac{1}{10}\right]$$
$$F^{-1}_{e_i e_{i+1}}(I) = I \quad 3 \le i \le 7 \quad F^{-1}_{e_8 e_1}(I) = \left[l - \tfrac{1}{5}, u - \tfrac{1}{5}\right]$$

Besides, $\mathsf{Pre}_{\sigma}(I) = F^{-1}(I \cap J) \cap S$, where $F^{-1}(I) = [2l - \frac{23}{30}, 2u - \frac{1}{5}]$. ■

2.4 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 $1 \le i \neq j \le k$. Let $\mathsf{Succ}_{\sigma}(I) = F(I \cap S) \cap J$ 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 3 *None of the two functions f_l, f_u is the identity.*

Let l^* and u^* be the fixpoints[5] of f_l and f_u, respectively, and $S \cap J = \langle L, U \rangle$. We have shown in [ASY01] that a simple cycle is of one of the following types:

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

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, $l^* < L \le u^* \le U$.

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

Example 5. Let $\sigma = e_1 \cdots e_8 e_1$. We have that $S \cap J = \langle L, U \rangle = (\frac{1}{5}, \frac{53}{60})$. The fixpoints of the equation in example 3 are such that $L = l^* = \frac{1}{5} < u^* = \frac{23}{30} < U$. Thus, σ is STAY. ■

The classification above gives us some 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 very useful for solving the reachability problem [ASY01].

[5] Obviously, the fixpoint x^* is computed by solving a linear equation $f(x^*) = x^*$, which can be finite or infinite (see Lemma 6, page 45 of [Sch02]).

Example 6. Consider again the cycle $\sigma = e_1 \cdots e_8 e_1$. Fig. 1(b) shows the reach set of the interval $[0.95, 1.0] \subset e_1$. Notice that the leftmost trajectory "converges to" the limit $l^* = \frac{1}{5}$. Fig. 1(b) has been automatically generated by the SPeeDI toolbox [APSY02] we have developed for reachability analysis of SPDIs. ■

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) whether any trajectory (if it exists) starting in the point remains in the cycle forever. In order to do this, we need to further study the properties of the system around simple edge-cycles and in particular STAY cycles. See [Sch03] for some important properties of STAY cycles.

3 Invariance Kernel

In this section we define the notion of *invariance kernel* and we show how to compute it. 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. 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 $K \subset \mathbb{R}^2$. We recall the definition of *viable* trajectory. 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.

Definition 1. *We say that a set K is* invariant *if for any $x \in K$ such that there exists at least one trajectory starting in it, every trajectory starting in x is viable in K. Given a set K, its largest invariant subset is called the* invariance kernel *of K and is denoted by $\mathsf{Inv}(K_\sigma)$.* ■

We denote by $\mathcal{D}_\sigma$ the one-dimensional discrete-time dynamical system defined by Succ_σ, that is $x_{n+1} \in \mathsf{Succ}_\sigma(x_n)$. The concepts above 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 $i \geq 0$. Similarly we say that an interval I in an edge e is invariant if *any* trajectory starting on $x_0 \in I$ is viable in I.

Before showing how to compute the invariance kernel of a cycle, we give a characterization of one-dimensional discrete-time invariant.

Lemma 1. *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 $F(I) = \mathcal{F}(I)$.*

Proof: Suppose that $F(I) = \mathcal{F}(I)$ and $F(I) \subseteq I$, then $\mathcal{F}(I) \subseteq I$, thus by definition of STAY and monotonicity of $\mathcal{F}$, we know that for all n, $\mathcal{F}^n(I) \subseteq I$.

Hence I is invariant. Let suppose now that I is invariant, then for any trajectory starting on $x_0 \in I$, $x_0 x_1 \ldots$ is in I and trivially $F(I) = \mathcal{F}(I)$. $\square$

Given two edges e and e' and an interval $I \subseteq e'$ we define the $\forall$-predecessor $\widetilde{\mathsf{Pre}}(I)$ 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. We have that $\widetilde{\mathsf{Pre}}_{ee'} = \widetilde{\mathsf{Suc}}_{ee'}^{-1}$ and $\widetilde{\mathsf{Pre}}_{\sigma} = \widetilde{\mathsf{Suc}}_{\sigma}^{-1}$.

Theorem 2. *For $\mathcal{D}_\sigma$, if $\sigma = e_1 \ldots e_n e_1$ is STAY then $\mathsf{Inv}(e_1) = \widetilde{\mathsf{Pre}}_{\sigma}(J)$, else $\mathsf{Inv}(e_1) = \emptyset$.*

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 us consider a STAY cycle with signature σ. Let $I_K = \tilde{\mathcal{F}}^{-1}(J) = \widetilde{\mathsf{Pre}}_{\sigma}(J)$. We know that $F(\tilde{\mathcal{F}}^{-1}(J)) = \mathcal{F}(\tilde{\mathcal{F}}^{-1}(J))$[6] and by STAY property, $F(\tilde{\mathcal{F}}^{-1}(J)) \subseteq \tilde{\mathcal{F}}^{-1}(J)$, thus by Lemma 1 we have that I_K is invariant. We prove now that I_K is indeed the greatest invariant. Let suppose that there exists an invariant $H \subseteq S$ strictly greater than I_K. By assumption we have that $I_K = \tilde{\mathcal{F}}^{-1}(J) \subset H$, then by monotonicity of $\mathcal{F}$, $\mathcal{F}(\tilde{\mathcal{F}}^{-1}(J)) \subset \mathcal{F}(H)$ and since $\mathcal{F}(\tilde{\mathcal{F}}^{-1}(J)) = J$[7] we have that $J \subset \mathcal{F}(H)$, but this contradicts the monotonicity of $\mathcal{F}$ since $J = \mathcal{F}(S) \subset \mathcal{F}(H)$ and then $S \subset H$ which contradicts the hypothesis that $H \subseteq S$. Hence, $\mathsf{Inv}(e_1) = \widetilde{\mathsf{Pre}}_{\sigma}(J)$. $\square$

The invariance kernel for the continuous-time system can be now found by propagating $\widetilde{\mathsf{Pre}}(J)$ from e_1 using the following operator. 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 2. *Let R be a region and e be an edge in $Out(R)$. The e-*extended $\forall$-predecessor *of I, $\widetilde{\overline{\mathsf{Pre}}}_e(I)$ is defined as:*
$\widetilde{\overline{\mathsf{Pre}}}_e(I) = \{\mathbf{x} \mid \forall \xi \ . \ (\xi(0) = \mathbf{x} \Rightarrow \exists t \geq 0 \ . \ (\xi(t) \in I \wedge \mathsf{Sig}(\xi[0,t]) = e))\}$. $\blacksquare$

The above notion can be extended to cyclic signatures (and so to edge-signatures) as follows. Let $\sigma = e_1, \ldots, e_k$ be a cyclic signature. For $I \subseteq e_1$, the *σ-extended $\forall$-predecessor* of I, $\widetilde{\overline{\mathsf{Pre}}}_{\sigma}(I)$ is the set of all $\mathbf{x} \in \mathbb{R}^2$ for which any trajectory segment ξ starting in $\mathbf{x}$, reaches some point in I, such that $\mathsf{Sig}(\xi)$ is a suffix of $e_2 \ldots e_k e_1$.

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 $1 \leq i \leq n$ 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 $2 \leq i \leq k-1$.

[6] See Lemma 13 in [Sch03] for a proof.
[7] See Lemma 12 in [Sch03] for a proof.

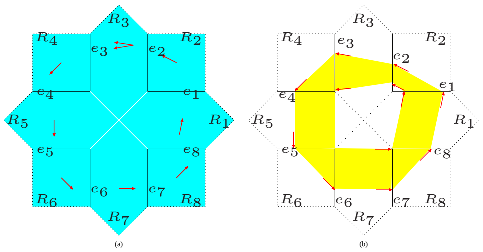

Fig. 2. Invariance kernel.

Now, let define the following set:

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

where P_i is such that $e_{i-1} \in \mathit{In}(P_i)$, $e_i \in \mathit{Out}(P_i)$ and $\mathtt{int}(P_i)$ is the interior of P_i.

We can now compute the invariance kernel of K_σ.

Theorem 3. *If σ is STAY then $\mathsf{Inv}(K_\sigma) = \widetilde{\overline{\mathsf{Pre}}}_\sigma(\widetilde{\mathsf{Pre}}_\sigma(J))$, otherwise $\mathsf{Inv}(K_\sigma) = \emptyset$.*

Proof: Trivially $\mathsf{Inv}(K_\sigma) = \emptyset$ for any type of cycle but STAY. That $\mathsf{Inv}(K_\sigma) = \widetilde{\overline{\mathsf{Pre}}}_\sigma(\widetilde{\mathsf{Pre}}_\sigma(J))$ for STAY cycles, follows directly from Theorem 2 and definition of $\widetilde{\overline{\mathsf{Pre}}}$. □

Example 7. Let $\sigma = e_1 \ldots e_8 e_1$. Fig. 2 depicts: (a) K_σ, and (b) $\widetilde{\overline{\mathsf{Pre}}}_\sigma(\widetilde{\mathsf{Pre}}_\sigma(J))$ ■

4 Reachability Algorithms for SPDIs

4.1 Previous algorithm [ASY01]

The decidability proof of [ASY01] already provides an algorithmic way of deciding reachability in SPDIs, which was implemented in our tool SPeeDI [APSY02]. We will give an overview of the algorithm to be able to compare and contrast it with the new algorithm that we are proposing. The decidability proof is split into three steps:

1. Identify a notion of *types of signatures,* each of which 'embodies' a number of signatures through the SPDI.
2. Prove that a finite number of types suffice to cover all edge signatures. Furthermore, given an SPDI, this set is computable.
3. Give an algorithm which decides whether a given type includes a signature which is feasible under the differential inclusion constraints of the SPDI.

We will not go into the details (see [ASY01] for more details), but will outline a number of items which will allow us to compare the algorithms.

Definition 3. *A* type signature *is a sequence of edge signatures with alternating loops:* $r_1 s_1^+ r_2 s_2^+ \ldots s_n^+ r_{n+1} s^*$. *The* r_i *parts of the type signature are called the* sequential paths *while the* s_i *parts called* iteration paths. *The last iteration path* s *is always a STAY loop. The interpretation of a type is similar to that of regular expressions:*
$\mathsf{signatures}(r_1 s_1^+ r_2 s_2^+ \ldots s_n^+ r_{n+1} s^*) \stackrel{\text{df}}{=} \{r_1 s_1^{k_1} r_2 s_2^{k_2} \ldots s_n^{k_n} r_{n+1} s^k \mid k_i > 0,\ k \geq 0\}$ ■

In [ASY01], one can find details of how to decide whether a given type signature includes an edge signature which is feasible. Clearly, given a source edge e_s and a destination edge e_f, there potentially exists an infinite number of type signatures from e_s to e_f. To reduce this to a finite number, [ASY01] applies a number of syntactic constraints which ensure finiteness, but do not leave out any possibly feasible edge signatures. Using these constraints it is easy to implement a depth-first traversal of the SPDI to check all possible type signatures. Note that a breadth-first traversal would require excessive storage requirements of all intermediate nodes.
From our experience in using SPeeDI, our implementation of this algorithm, the main deficiency of this approach is that incorrect systems which may have 'short' counter-examples (in terms of type signature length) end up lost in the exploration of long paths — either taking an excessive amount of time to find the counter-example, or coming up with a long counter-example difficult to use for debugging the hybrid system. Ideally, we should be able to find a shortest counter-example without the need of exhaustive exploration of the SPDI.

4.2 A Breadth-First Algorithm

As is evident from the previous section, it is desirable to have a breadth-first algorithm to be able to identify shortest[8] counter-examples and be able to use standard algorithms for optimisation.

Definition 4. *The* edge graph *of an SPDI with partition* $\mathbb{P}$ *is the graph with the region edges as nodes:* $N = \cup_{P \in \mathbb{P}} E(P)$*; and transitions between two edges in the same partition with the first being an input and second an output edge:* $T = \{(e, e') \mid \exists P \in \mathbb{P} \,.\, e, e' \in P,\ e \in \mathit{In}(P),\ e' \in \mathit{Out}(P)\}$. ■

Example 8. To illustrate the notion of an edge-graph, Fig. 3 illustrates the edge-graph corresponding to the SPDI representing the swimmer example given in Fig. 1(a).

Definition 5. *The* meta-graph *of an SPDI* S *is its edge graph augmented with the loops in the SPDI:*

1. *An unlabelled transition for every transition in the original graph:* $\{e \to e' \mid$ *input edge* e *and output edge* e' *belong to the same region* $\}$

[8] Note that shortest, in this context, is not in terms or length of path on the SPDI, or number of edges visited, but on the length of the abstract signature which includes a counter-example.

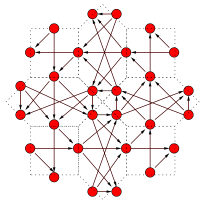

Fig. 3. The edge-graph of the swimmer SPDI example

2. *A set of labelled transitions, one for each simple loop in the original graph which is eventually left:* $\{e \overset{se}{\looparrowright} e' \mid \mathsf{head}(s) \neq e',\ esee' \text{ is a valid path in } S\}$
3. *A set of labelled sink edges, one for each simple loop of type STAY which is never left:* $\{e \overset{se}{\circlearrowleft} \mid ese \text{ is a valid path in } S,\ se \text{ is a STAY loop}\}$. ■

Note that reachability along a path through the meta-graph corresponds to that of a type signature as defined in the previous section. For example $e_1 \longrightarrow e_2 \overset{s_1}{\looparrowright} e_3 \overset{s_2}{\circlearrowleft}$ would correspond to $e_1 e_2 s_1^+ e_3 s_2^*$. From the result in [ASY01], which states that only a finite number of abstract signatures (of finite length) suffices to describe the set of all signatures through the graph, it immediately follows that for any SPDI, it suffices to explore the meta-graph to a finite depth to deduce the reachable set.

Proposition 1. *Given an SPDI S, reachability in S is equivalent to reachability in the meta-graph of S.* □

To implement the meta-graph traversal, we will define the functions corresponding to the different transitions which, given a set of edge-intervals already visited, return a new set of edge-intervals which will be visited along that transition:

$$\longrightarrow (E) \stackrel{df}{=} \{\mathsf{Succ}_{ee'}(i) \mid i \in E,\ i \subseteq e,\ e \rightarrow e'\}$$
$$\looparrowright (E) \stackrel{df}{=} \{\mathsf{Succ}_{p}(i) \mid i \in E,\ i \subseteq e,\ e \overset{\sigma}{\looparrowright} e',\ p \text{ prefix } e\sigma^{+}e'\}$$
$$\circlearrowleft (E) \stackrel{df}{=} \{\mathsf{Succ}_{p}(i) \cap \mathsf{Inv}(K_\sigma) \mid i \in E,\ i \subseteq e,\ e \overset{\sigma}{\circlearrowleft},\ p \text{ prefix } e\sigma^{*}\}.$$

Note that, using the techniques developed in [ASY01], we can always calculate the first two of the above. Furthermore, we are guaranteed that if E consists of a finite number of edge-intervals, so will $\longrightarrow (E)$ and $\looparrowright (E)$. Unfortunately, this is not the case with $\circlearrowleft (E)$. However, it is possible to compute whether a given set of edge-intervals and $\circlearrowleft (E)$ (E being a finite set of edge-intervals) overlap. If we consider the standard model checking approach, we can use a given SPDI with transitions $\longrightarrow$, meta-transitions $\looparrowright$, sink-transitions $\circlearrowleft$ and initial set I:

$$R_0 \stackrel{df}{=} I \qquad\qquad R_{n+1} \stackrel{df}{=} R_n \cup \longrightarrow (R_n) \cup \looparrowright (R_n) \cup \circlearrowleft (R_n)$$

We can terminate once nothing else is added: $R_{N+1} = R_N$. Edge-interval sets R_n can be represented enumeratively. However, as already noted, STAY loops represented by sinks may induce an infinite number of disjoint intervals. However, since sinks are dead end transitions, we can simplify the reachability analysis by performing the sinks only at the end:

$$R_{n+1} \stackrel{df}{=} R_n \cup \rightarrow (R_n) \cup \looparrowright (R_n)$$

Since the termination condition depended on the fact that we were also applying the sink transitions $\circlearrowleft$, we add this when we check the termination condition: $R_N \cup \circlearrowleft (R_N) = R_{N+1} \cup \circlearrowleft (R_{N+1})$. Although the problem has been simply moved to the termination test, we show that this condition can be reduced to the simpler, and testable: $R_{N+1} \setminus R_N \subseteq \mathsf{Inv}$, where Inv is the set of all invariance kernels $\bigcup_{\sigma \in STAY} \mathsf{Inv}(K_\sigma)$. The proof of correctness of the algorithm can be found in [Pac03].

We can now implement the algorithm in a similar manner as standard forward model checking:

```
preR := {}; R := Src;
while (R \ preR ⊈ Inv)
    preR := R; R := R∪ → (R)∪ ↬ (R);
    if (Dst overlaps R) then return REACHABLE;
if (Dst overlaps (R ∪ ↺ (R)))
    then return REACHABLE else return UNREACHABLE;
```

5 Conclusion

One of the contributions of this paper is an automatic procedure to obtain an important object of the phase portrait of simple planar differential inclusions (SPDIs [Sch02]), namely invariance kernels.

We have also presented a breadth-first search algorithm for solving the reachability problem for SPDIs. The advantage of such an algorithm is that it is much simpler than the one presented in [ASY01] and it reminds the classical model checking algorithm for computing reachability. Invariance kernels play a crucial role to prove termination of the BFS reachability algorithm.

We intend to implement the algorithm in order to empirically compare it with the previous algorithm for SPDIs [APSY02].

Acknowledgments. We are thankful to Eugene Asarin and Sergio Yovine for the valuable discussions.

References

[ABDM00] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control 2000*, LNCS 1790. Springer Verlag, 2000.

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

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

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

[APSY02] E. Asarin, G. Pace, G. Schneider, and S. Yovine. Speedi: a verification tool for polygonal hybrid systems. In E. Brinksma and K.G. Larsen, editors, *Computer-Aided Verification 2002*, LNCS 2404, 2002.

[ASY01] E. Asarin, G. Schneider, and S. Yovine. On the decidability of reachability for planar differential inclusions. In *Hybrid Systems: Computation and Control 2001*, LNCS 2034, 2001.

[ASY02] E. Asarin, G. Schneider, and S. Yovine. Towards computing phase portraits of polygonal differential inclusions. In *Hybrid Systems: Computation and Control 2002*, LNCS 2289, 2002.

[Aub01] J.-P. Aubin. The substratum of impulse and hybrid control systems. In *Hybrid Systems: Computation and Control 2001*, LNCS 2034, 2001.

[BT00] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Hybrid Systems: Computation and Control 2000*, LNCS 1790, 2000.

[CV96] K. Cerāns and J. Vīksna. Deciding reachability for planar multi-polynomial systems. In *Hybrid Systems III*, LNCS 1066, 1996.

[DM98] T. Dang and O. Maler. Reachability analysis via face lifting. In *Hybrid Systems: Computation and Control 1998*, LNCS 1386, 1998.

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

[GM99] M. R. Greenstreet and I. Mitchell. Reachability analysis using polygonal projections. In *HSCC '99*, LNCS 1569, 1999.

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

[KdB01] P. Kowalczyk and M. di Bernardo. On a novel class of bifurcations in hybrid dynamical systems. In *Hybrid Systems: Computation and Control 2001*, LNCS 2034. Springer, 2001.

[KV96] M. Kourjanski and P. Varaiya. Stability of hybrid systems. In *Hybrid Systems III*, LNCS 1066, pages 413–423. Springer, 1996.

[KV00] A.B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *Hybrid Systems: Computation and Control*, LNCS 1790, 2000.

[LPY99] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 1569. 1999.

[MP93] O. Maler and A. Pnueli. Reachability analysis of planar multi-linear systems. In *Computer-Aided Verification '93*. LNCS 697, 1993.

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

[Pac03] G. J. Pace. A new breadth first search algorithm for deciding spdi reachability. Technical Report CSAI2003-01, Department of Computer Science & AI, University of Malta, 2003.

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

[Sch03] G. Schneider. Invariance kernels of polygonal differential inclusions. Technical Report 2003-042, Department of IT, Uppsala University, 2003.

[SJSL00] S. Simić, K. Johansson, S. Sastry, and J. Lygeros. Towards a geometric theory of hybrid systems. In *Hybrid Systems: Computation and Control 2000*, LNCS 1790, 2000.