Computing Invariance Kernels of Polygonal Hybrid Systems

GERARDO SCHNEIDER

gerardos@it.uu.se

UPPSALA UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

UPPSALA, SWEDEN

UPPSALA UNIVERSITET

Overview of the presentation

- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Motivation and Related Work

- For Hybrid Systems
  - Verification (reachability, ...):
  - Qualitative behavior (Phase Portrait, ...)

Motivation and Related Work

- For Hybrid Systems
  - Verification (reachability, ...):
  - Qualitative behavior (Phase Portrait, ...)
- For a class of non-deterministic systems (SPDI)
  - Verification (HSCC'01)
  - Undecidability of some extensions (CONCUR'02)
  - Phase Portrait (HSCC'02):
    - Viability Kernel
    - Controllability Kernel
  - Invariance Kernels

Why Invariance Kernels?

- Important objects for giving SPDIs
- Crucial for proving termination of a BFS reachability algorithm for SPDI

Overview of the presentation

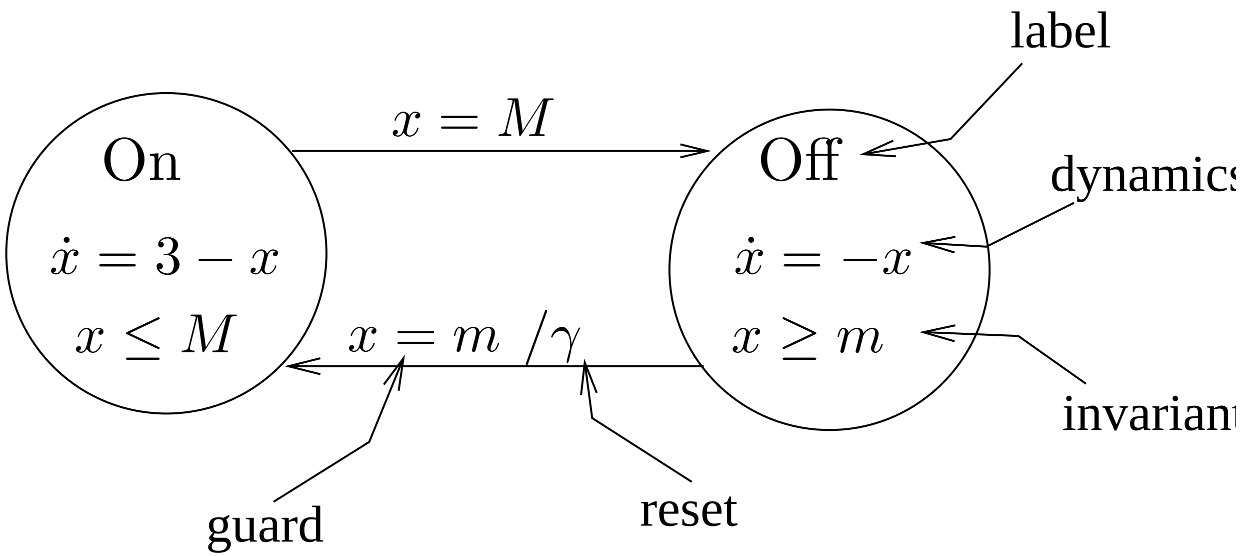
- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Hybrid Systems

- Hybrid Systems: interaction between discrete and continuous behaviors
- Examples: thermostat, automated highway systems, air traffic management systems, robotic systems, chemical plants, etc.

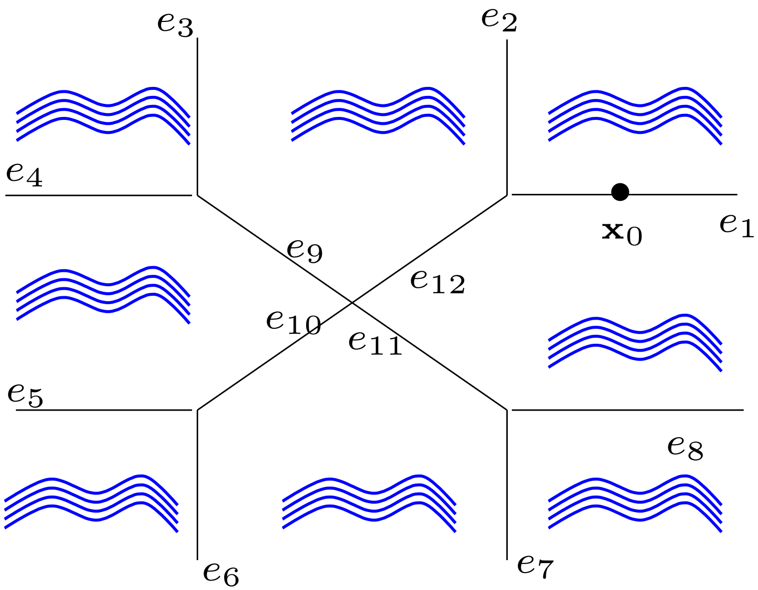
Hybrid Systems

Model: Hybrid Automata



Hybrid Systems

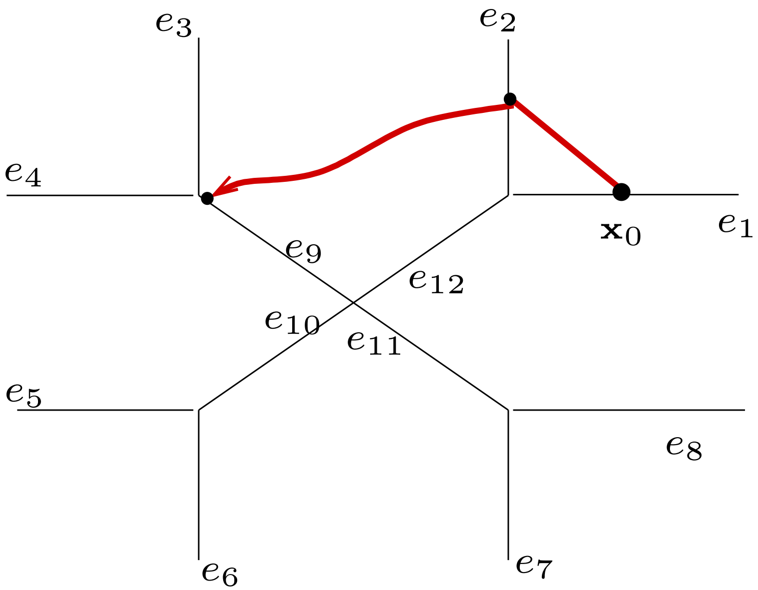
Example: Swimmer in a whirlpool



UPPSALA UNIVERSITET

Hybrid Systems

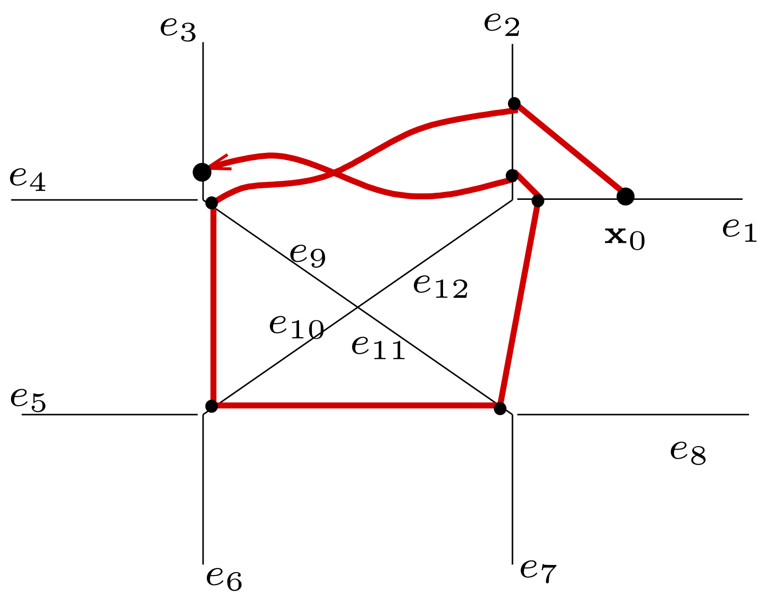
Example: Swimmer in a whirlpool



UPPSALA UNIVERSITET

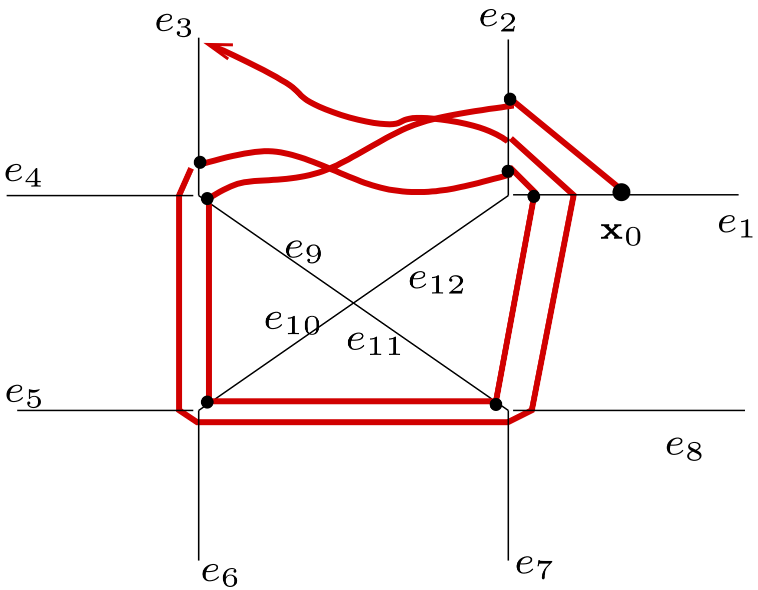
Hybrid Systems

Example: Swimmer in a whirlpool

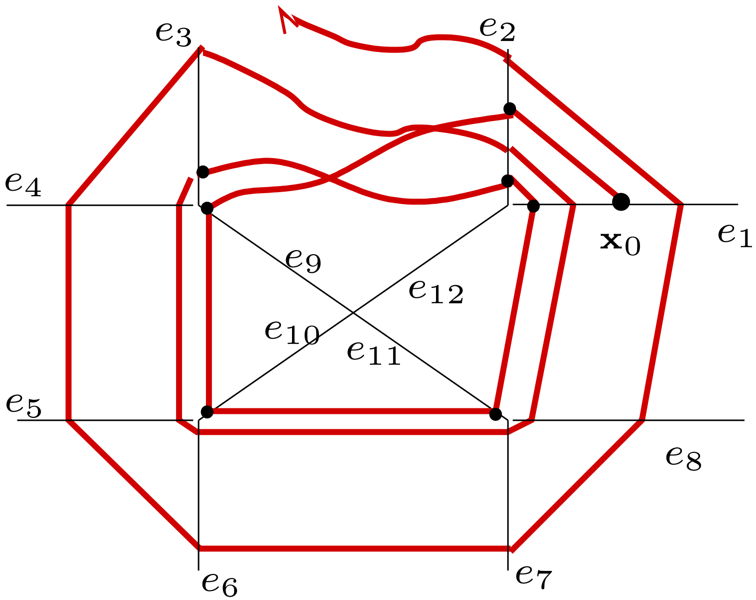


UPPSALA UNIVERSITET

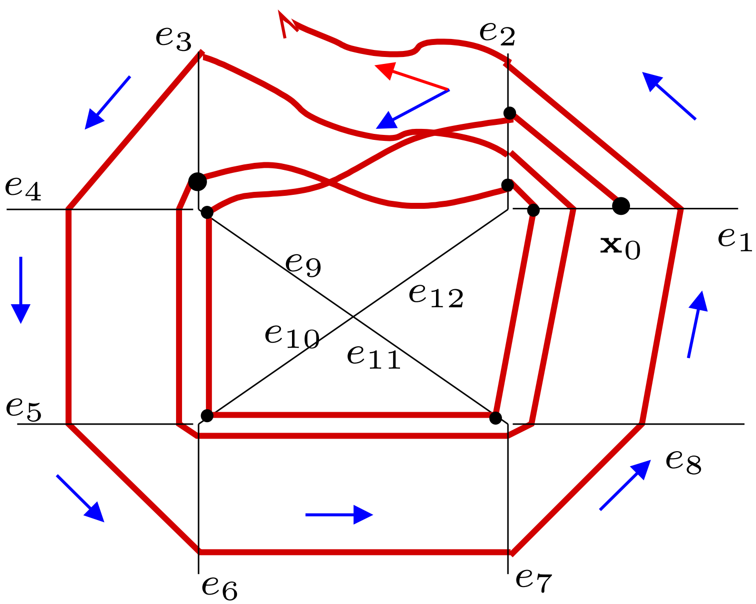
Example: Swimmer in a whirlpool



Example: Swimmer in a whirlpool



Example: Swimmer in a whirlpool



UPPSALA UNIVERSITET

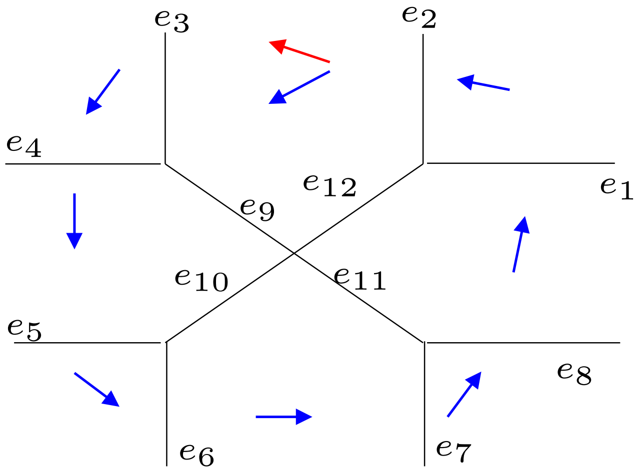
Overview of the presentation

- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Polygonal Differential Inclusion Systems (SPDIs)

- A partition of the plane into convex polygonal regions
- A constant differential inclusion for each region

$$\dot{x} \in \angle_{\mathbf{a}}^{\mathbf{b}} \text{ if } \mathbf{x} \in R_i$$

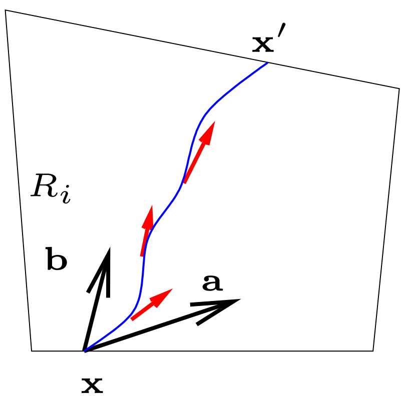


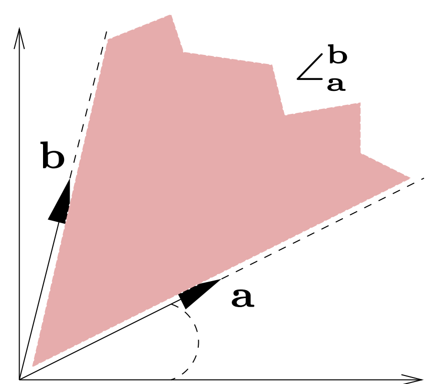
UPPSALA UNIVERSITET

Polygonal Differential Inclusion Systems (SPDIs)

- A partition of the plane into convex polygonal regions
- A constant differential inclusion for each region

$$\dot{x} \in \angle_{\mathbf{a}}^{\mathbf{b}} \text{ if } \mathbf{x} \in R_i$$





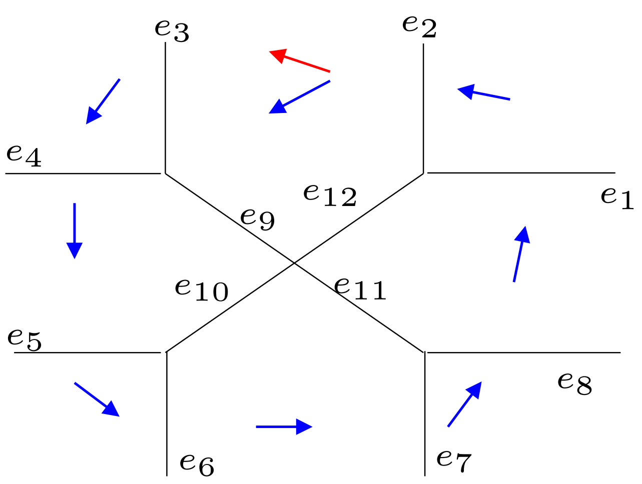
UPPSALA UNIVERSITET

Polygonal Differential Inclusion Systems (SPDIs)

- The “swimmer” is a hybrid system
- Hybrid Automata?

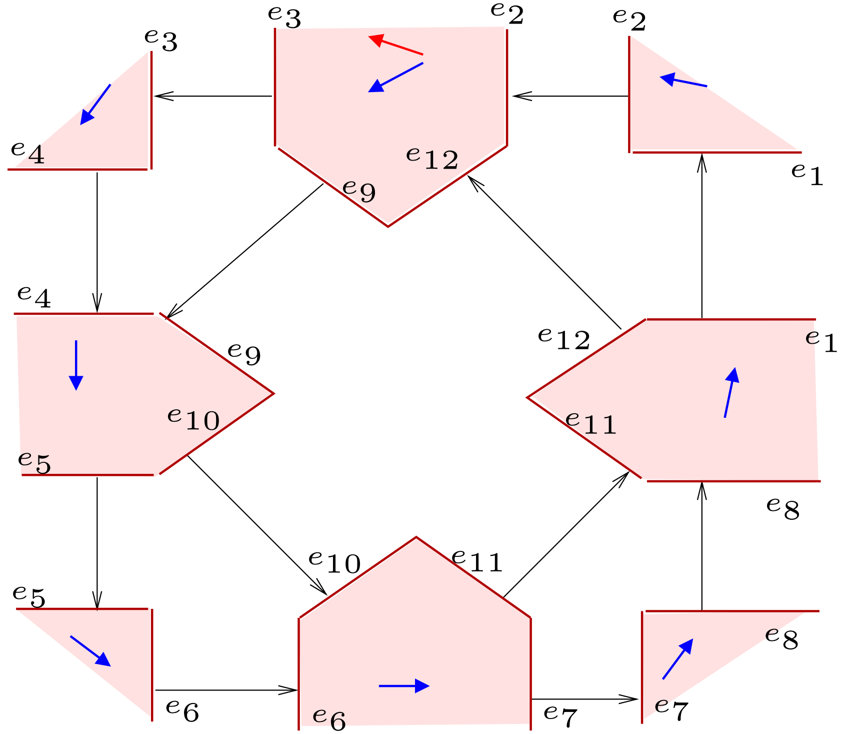
Polygonal Differential Inclusion Systems (SPDIs)

- The “swimmer” is a hybrid system
- Hybrid Automata?



Polygonal Differential Inclusion Systems (SPDIs)

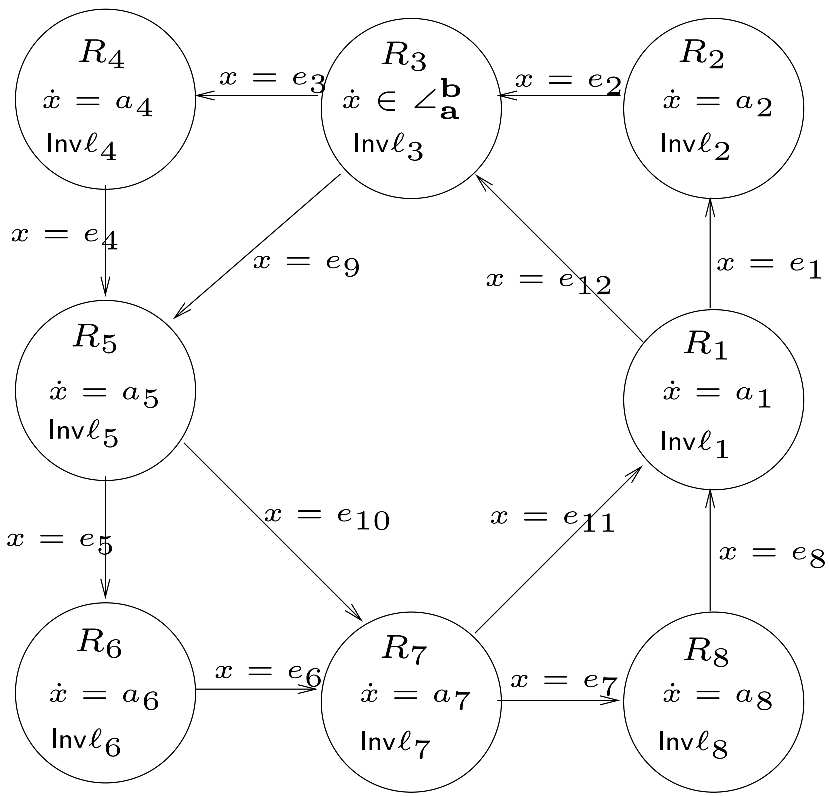
- The "swimmer" is a hybrid system
- Hybrid Automata?



UPPSALA UNIVERSITET

Polygonal Differential Inclusion Systems (SPDIs)

- The "swimmer" is a hybrid system
- Hybrid Automata?



UPPSALA UNIVERSITET

Polygonal Differential Inclusion Systems (SPDIs)

- The “swimmer” is a hybrid system
- Hybrid Automata?

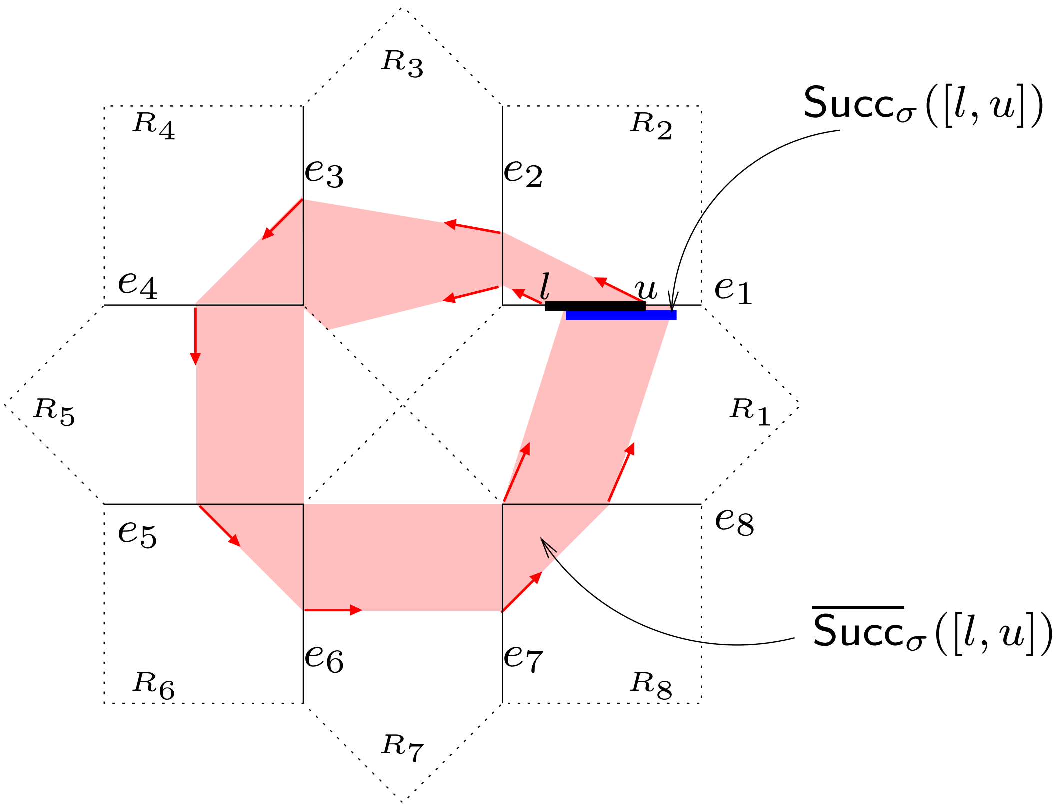
We will use the “geometric” representation instead of the hybrid automata

Overview of the presentation

- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Successor Operators

For a *signature* $\sigma = e_1 \ldots e_8 e_1$:



UPPSALA UNIVERSITET

Successor Operators

- Successors have the form

$$\mathsf{Succ}_\sigma(l,\ u) = [a_1 l + b_1,\ a_2 u + b_2] \cap J \text{ if } [l, u] \subseteq S$$

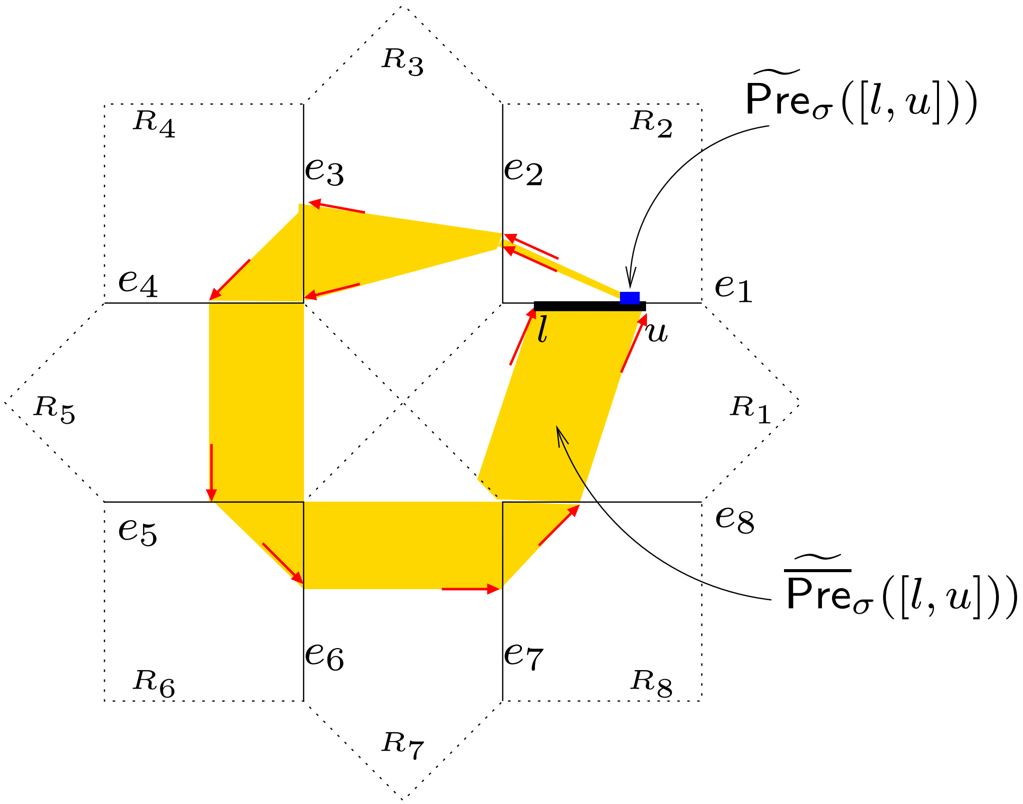
- Fixpoint equations

$$[a_1 l^* + b_1, a_2 u^* + b_2] = [l^*, u^*]$$

can be explicitely solved (without iterating).

Predecessor Operators

For $\sigma = e_1 \ldots e_8 e_1$:



UPPSALA UNIVERSITET

Overview of the presentation

- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Classification of Simple Cycles

Given a cyclic signature $\sigma = e_1 \dots e_8 e_1$.
Let $e_1 = [L, U]$.

$$\mathsf{Succ}^*_\sigma = [l^*, u^*]$$

$$\mathsf{Succ}_\sigma([l, u]) \subseteq [l^*, u^*]$$

UPPSALA UNIVERSITET

Classification of Simple Cycles

Given a cyclic signature $\sigma = e_1 \ldots e_8 e_1$.
Let $e_1 = [L, U]$.

$$\mathsf{Succ}^*_\sigma = [l^*, u^*]$$

$$\mathsf{Succ}_\sigma([l, u]) \subseteq [l^*, u^*]$$

STAY:	$L \leq l^* \leq u^* \leq U$
DIE:	$u^* < L \vee l^* > U$
EXIT-BOTH:	$l^* < L \wedge u^* > U$
EXIT-LEFT:	$l^* < L \leq u^* \leq U$
EXIT-RIGHT:	$L \leq l^* \leq U < u^*$

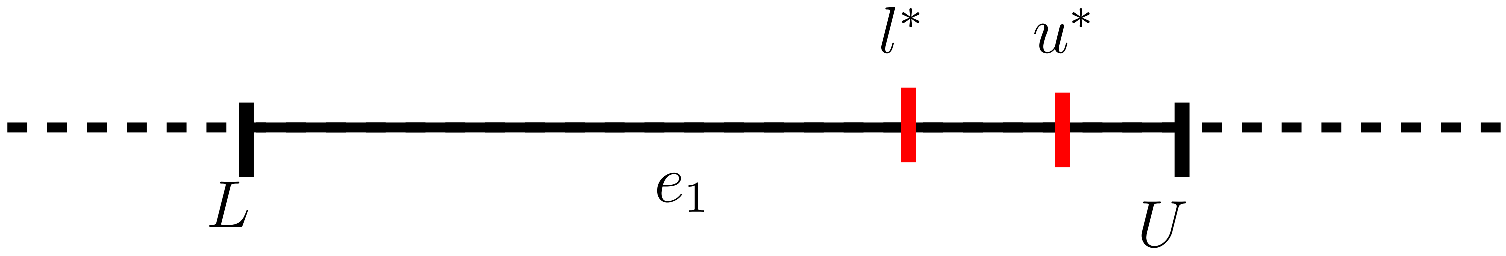
Classification of Simple Cycles

Given a cyclic signature $\sigma = e_1 \dots e_8 e_1$.
Let $e_1 = [L, U]$.

$$\mathsf{Succ}^*_\sigma = [l^*, u^*]$$

$$\mathsf{Succ}_\sigma([l, u]) \subseteq [l^*, u^*]$$

STAY:



UPPSALA UNIVERSITET

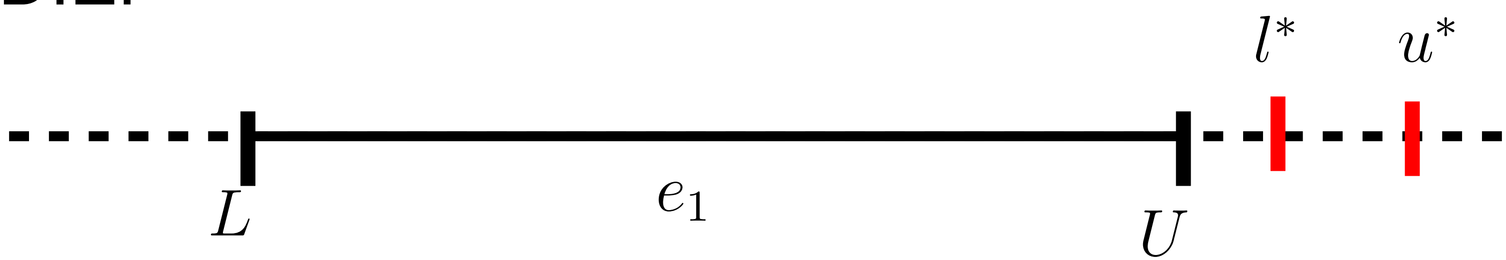
Classification of Simple Cycles

Given a cyclic signature $\sigma = e_1 \ldots e_8 e_1$.
Let $e_1 = [L, U]$.

$$\mathsf{Succ}^*_\sigma = [l^*, u^*]$$

$$\mathsf{Succ}_\sigma([l, u]) \subseteq [l^*, u^*]$$

DIE:



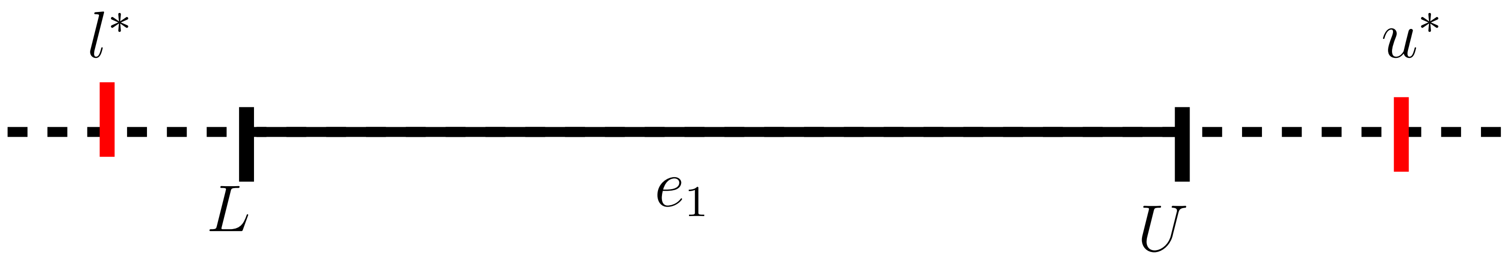
Classification of Simple Cycles

Given a cyclic signature $\sigma = e_1 \ldots e_8 e_1$.
Let $e_1 = [L, U]$.

$$\mathsf{Succ}^*_\sigma = [l^*, u^*]$$

$$\mathsf{Succ}_\sigma([l, u]) \subseteq [l^*, u^*]$$

EXIT-BOTH:



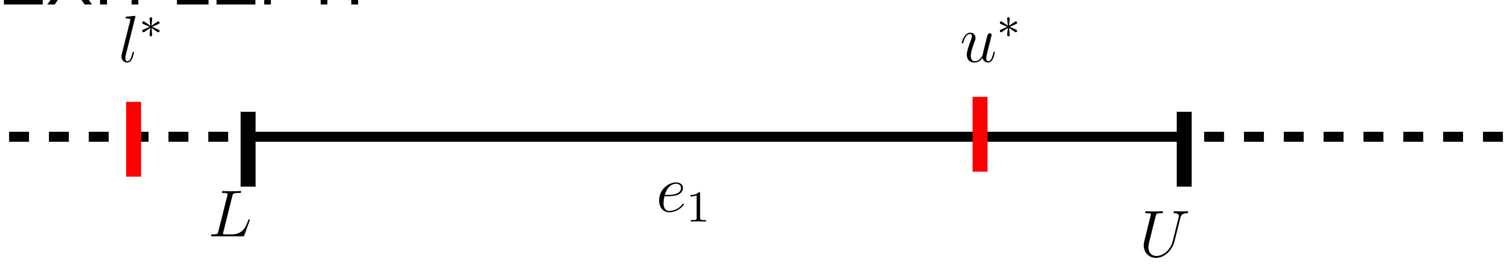
Classification of Simple Cycles

Given a cyclic signature $\sigma = e_1 \dots e_8 e_1$.
Let $e_1 = [L, U]$.

$$\mathsf{Succ}^*_\sigma = [l^*, u^*]$$

$$\mathsf{Succ}_\sigma([l, u]) \subseteq [l^*, u^*]$$

EXIT-LEFT:



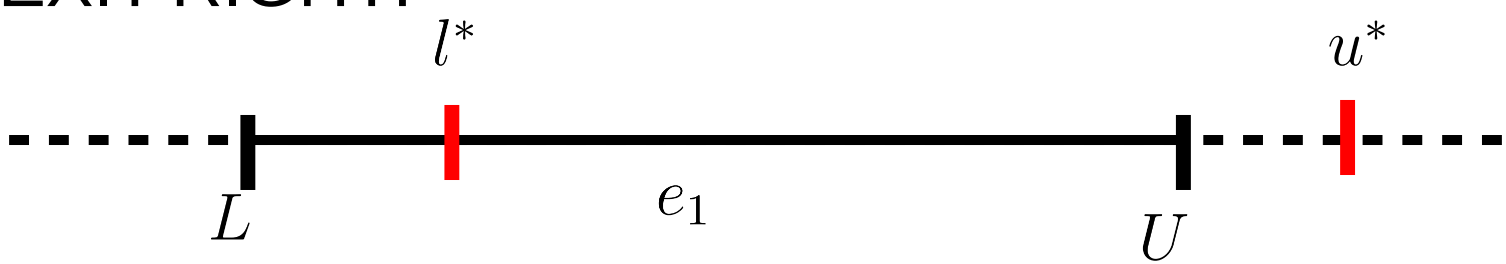
Classification of Simple Cycles

Given a cyclic signature $\sigma = e_1 \ldots e_8 e_1$.
Let $e_1 = [L, U]$.

$$\mathsf{Succ}^*_\sigma = [l^*, u^*]$$

$$\mathsf{Succ}_\sigma([l, u]) \subseteq [l^*, u^*]$$

EXIT-RIGHT:



Overview of the presentation

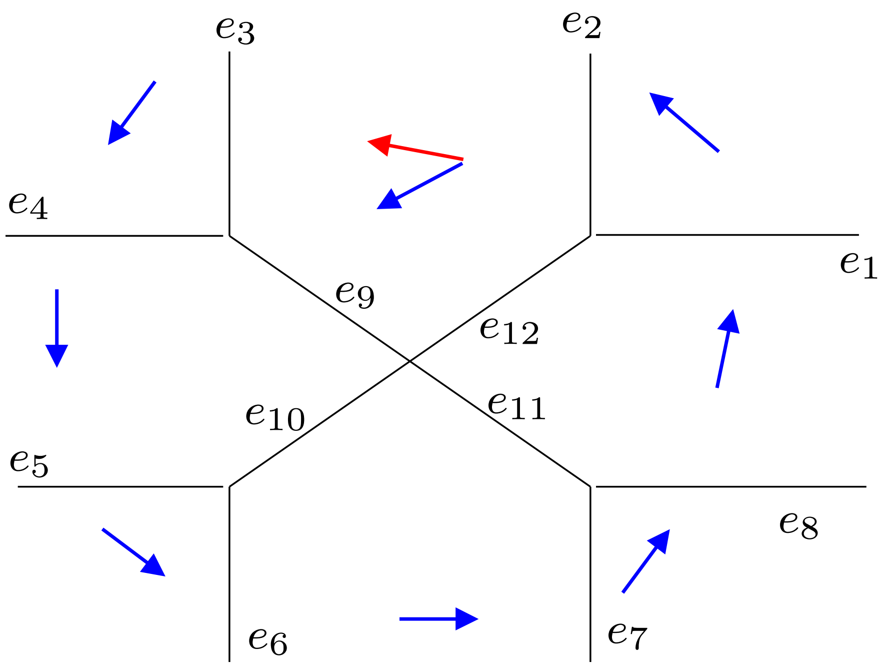
- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Phase Portrait

Phase Portrait: a picture of important objects of a dynamical system

Phase Portrait

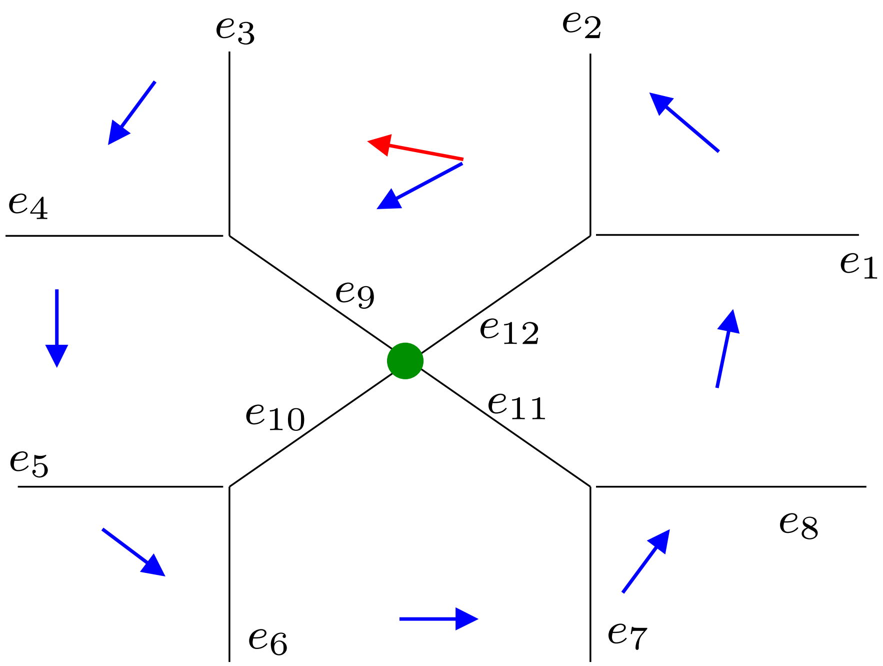
Phase Portrait: a picture of important objects of a dynamical system



UPPSALA UNIVERSITET

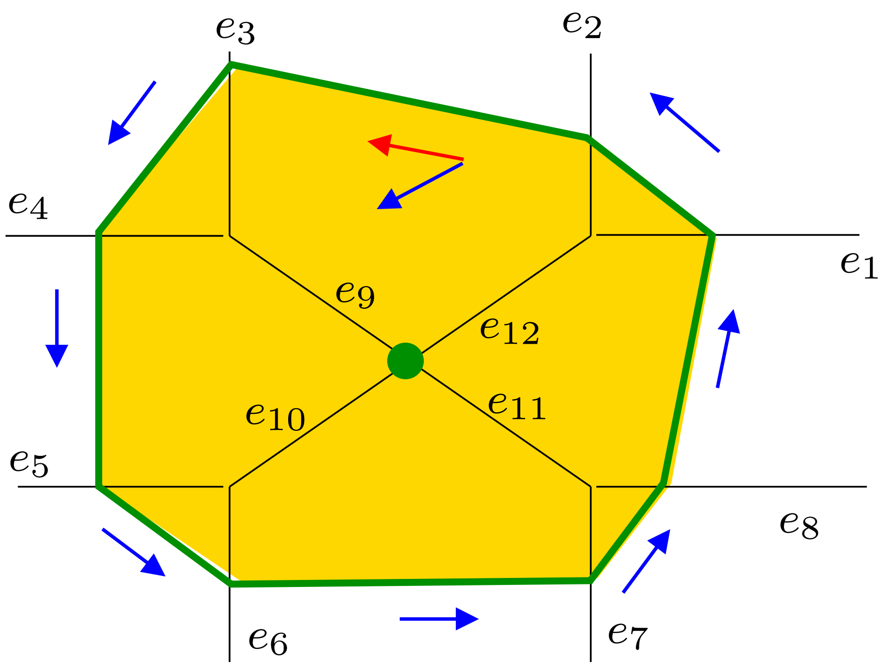
Phase Portrait

Phase Portrait: a picture of important objects of a dynamical system



Phase Portrait

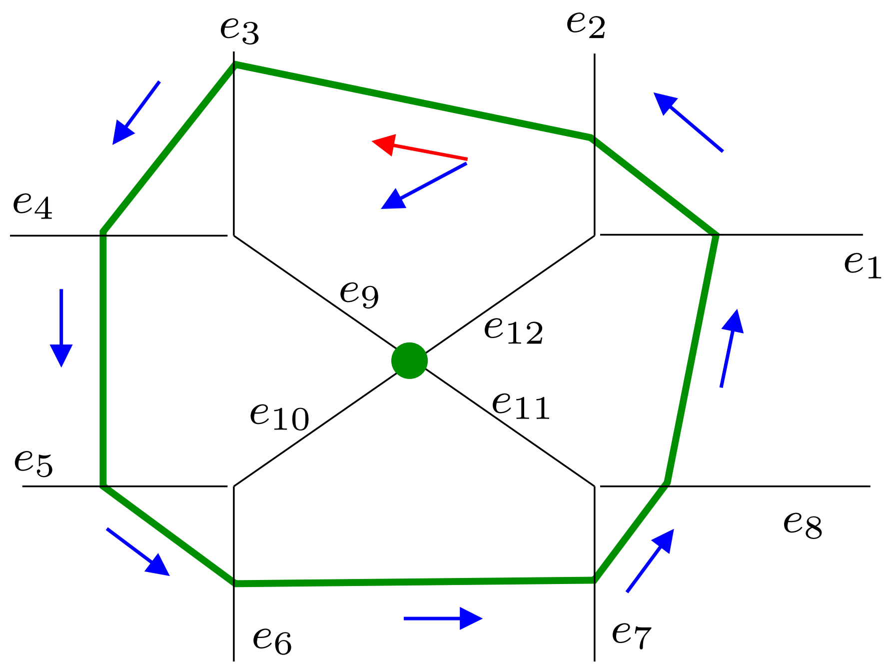
Phase Portrait: a picture of important objects of a dynamical system



UPPSALA UNIVERSITET

Phase Portrait

Phase Portrait: a picture of important objects of a dynamical system



Overview of the presentation

- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Invariance Kernel

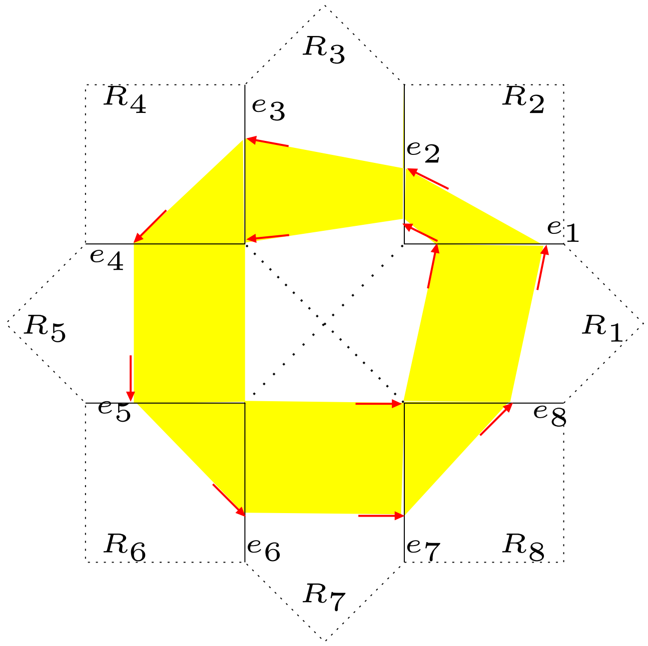
$\mathsf{Inv}(\sigma)$: Is the greatest set of points such that every trajectory starting in such points remains in the set forever.

UPPSALA UNIVERSITET

Invariance Kernel

Inv(σ): Is the greatest set of points such that every trajectory starting in such points remains in the set forever.

Example: $\sigma = e_1 e_2 \ldots e_8 e_1$



UPPSALA UNIVERSITET

Invariance Kernel

$\mathsf{Inv}(\sigma)$: Is the greatest set of points such that every trajectory starting in such points remains in the set forever.

Theorem: If σ is STAY: $\mathsf{Inv}(\sigma) = \overline{\widetilde{\mathsf{Pre}}_\sigma}(\widetilde{\mathsf{Pre}}_\sigma(J))$

Overview of the presentation

- Motivation
- Introduction: Hybrid System
- Polygonal Differential Inclusion System (SPDI)
- Successors and Predecessors
- Classification of Simple Cycles
- Phase Portrait of SPDIs
- Invariance Kernels
- Conclusions

Conclusions

ACHIEVEMENTS:

- Algorithm for obtaining a new object of SPDI's Phase Portrait: Invariance Kernel

APPLICATIONS:

- Find "sinks" of non-linear differential equations
- IK are important for proving termination of a BFS reachability algorithm for SPDIs

FUTURE WORK:

- Extend the tool SPeeDI to compute Invariance Kernels

Auxiliary Slides

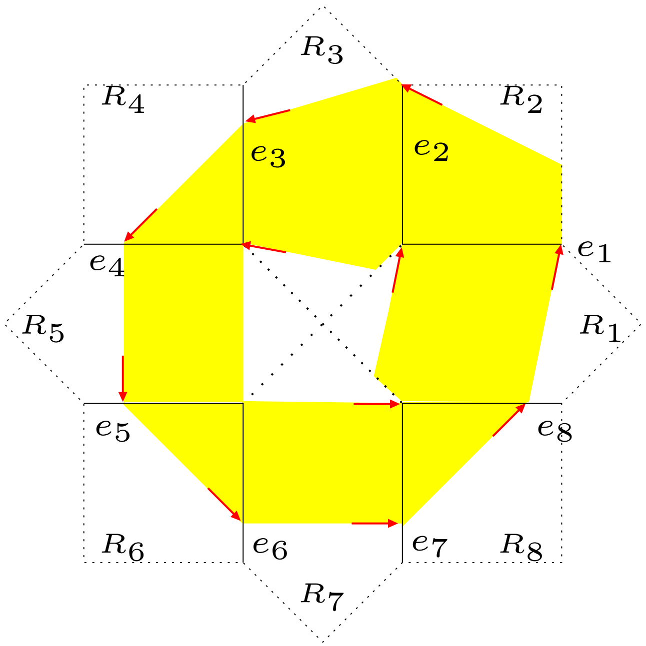
Viability Kernel

$\mathsf{Viab}(\sigma)$: Is the greatest set of initial points of trajectories which can cycle forever in σ

Viability Kernel

Viab(σ): Is the greatest set of initial points of trajectories which can cycle forever in σ

Example: $\sigma = e_1 e_2 \dots e_8 e_1$

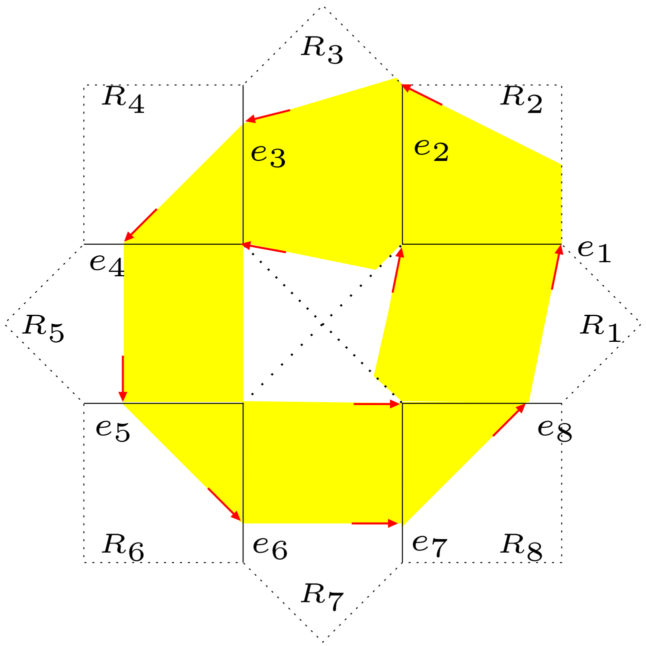


UPPSALA UNIVERSITET

Viability Kernel

$\textsf{\color{red}{Viab}}(\sigma)$: Is the greatest set of initial points of trajectories which can cycle forever in σ

Example: $\sigma = e_1 e_2 \ldots e_8 e_1$



Theorem: $\mathsf{Viab}(\sigma) = \overline{\mathsf{Pre}}_\sigma(\mathsf{Dom}(\mathsf{Succ}_\sigma))$

UPPSALA UNIVERSITET

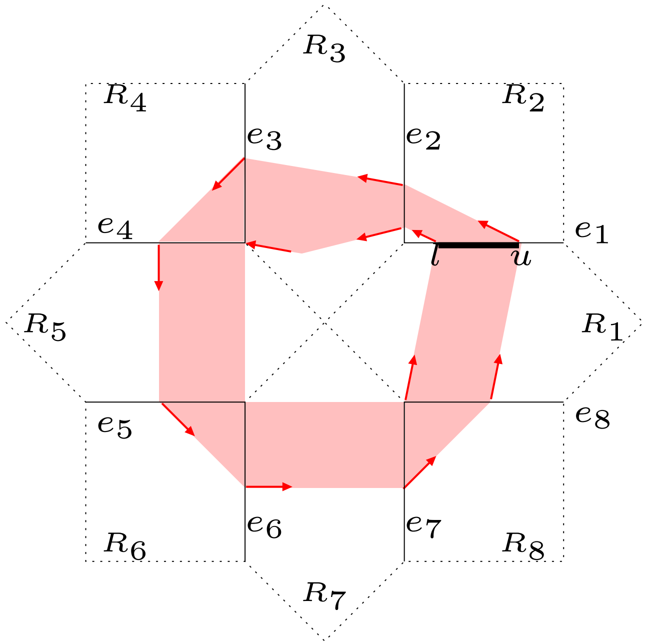
Controllability Kernel

$\mathsf{Cntr}(\sigma)$: Is the greatest set of mutually reachable points via trajectories that remain in the cycle

Controllability Kernel

Cntr(σ): Is the greatest set of mutually reachable points via trajectories that remain in the cycle

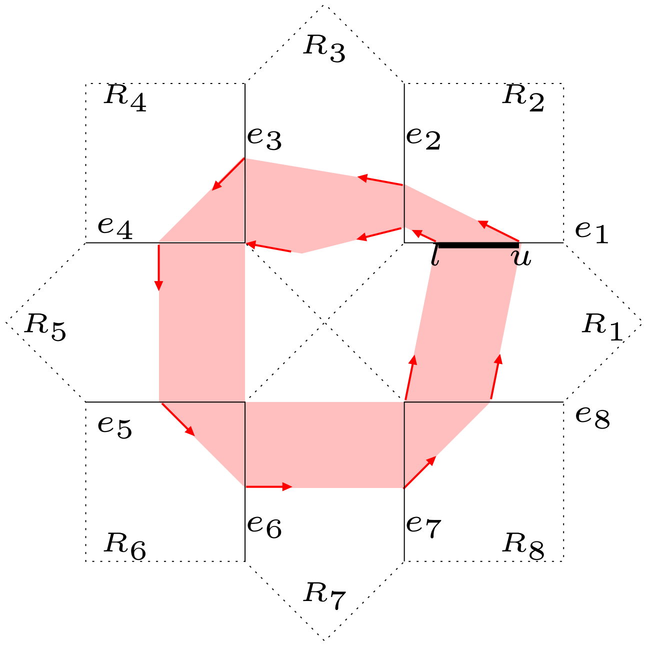
Example: $\sigma = e_1 e_2 \dots e_8 e_1$



Controllability Kernel

Cntr(σ): Is the greatest set of mutually reachable points via trajectories that remain in the cycle

Example: $\sigma = e_1 e_2 \dots e_8 e_1$



Theorem: $\mathsf{Cntr}(\sigma) = (\overline{\mathsf{Succ}}_\sigma \cap \overline{\mathsf{Pre}}_\sigma)(\mathcal{C}_\mathcal{D}(\sigma))$

UPPSALA UNIVERSITET

Phase Portrait of SPDIs

Algorithm: phase portrait for SPDIs

for each simple cycle σ **do**
Compute $\mathsf{Viab}(\sigma)$ (*viability kernel*)
Compute $\mathsf{Cntr}(\sigma)$ (*controllability kernel*)

Phase Portrait of SPDIs

Algorithm: phase portrait for SPDIs

for each simple cycle σ **do**
Compute $\mathsf{Viab}(\sigma)$ (*viability kernel*)
Compute $\mathsf{Cntr}(\sigma)$ (*controllability kernel*)

Both kernels are exactly computed by non-iterative algorithms!