A Compositional Algorithm for Parallel Model Checking of Polygonal Hybrid Systems

Gordon Pace[1]   Gerardo Schneider[2]

[1]Dept. of Computer Science and AI – University of Malta

[2]Dept. of Informatics – University of Oslo

ICTAC'06
November 20–24, 2006 - Tunis

Outline

1. Introduction
   - Hybrid Systems
   - Polygonal Hybrid Systems (SPDIs)
   - Motivation
2. Phase Portrait of SPDIs
   - Controllability and Viability Kernels
3. Independent Questions and Parallelization
   - SPDI Decomposition
   - Unavoidable Kernels
   - Counting Subproblems
4. Parallel Algorithm for Reachability

Outline

1 Introduction
- Hybrid Systems
- Polygonal Hybrid Systems (SPDIs)
- Motivation

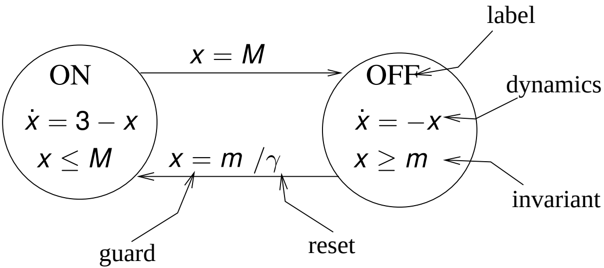
2 Phase Portrait of SPDIs
- Controllability and Viability Kernels

3 Independent Questions and Parallelization
- SPDI Decomposition
- Unavoidable Kernels
- Counting Subproblems

4 Parallel Algorithm for Reachability

Hybrid Systems

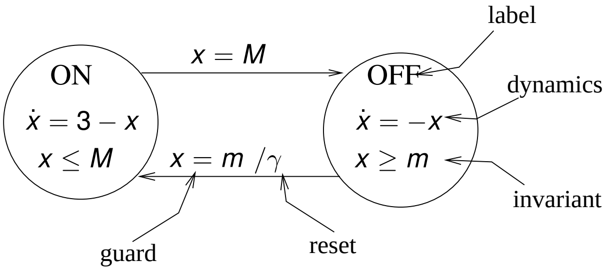
- Usual representation: Hybrid Automata



In general, we can have *differential inclusions* instead of differential equations

Hybrid Systems

- Usual representation: Hybrid Automata



In general, we can have *differential inclusions* instead of differential equations

Outline

1 Introduction
- Hybrid Systems
- Polygonal Hybrid Systems (SPDIs)
- Motivation

2 Phase Portrait of SPDIs
- Controllability and Viability Kernels

3 Independent Questions and Parallelization
- SPDI Decomposition
- Unavoidable Kernels
- Counting Subproblems

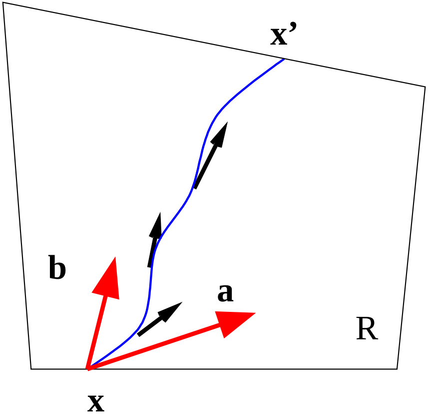
4 Parallel Algorithm for Reachability

Polygonal Hybrid Systems (SPDIs)

- A finite partition of the plane into convex polygonal sets (regions)
- Dynamics given by the angle determined by two vectors: $\dot{x} \in \angle_{\mathbf{a}}^{\mathbf{b}}$

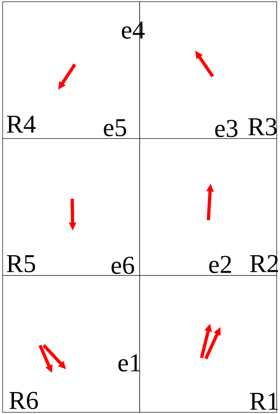
Polygonal Hybrid Systems (SPDIs)

- A finite partition of the plane into convex polygonal sets (regions)
- Dynamics given by the angle determined by two vectors: $\dot{x} \in \angle_{\mathbf{a}}^{\mathbf{b}}$



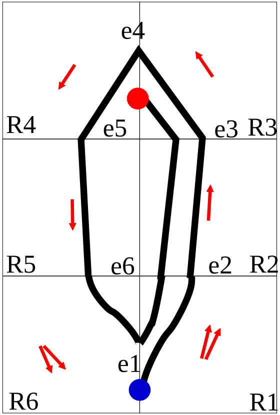
Polygonal Hybrid Systems (SPDIs)

- A finite partition of the plane into convex polygonal sets (regions)
- Dynamics given by the angle determined by two vectors: $\dot{x} \in \angle_{\mathbf{a}}^{\mathbf{b}}$



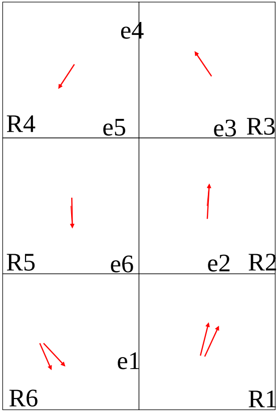
Polygonal Hybrid Systems (SPDIs)

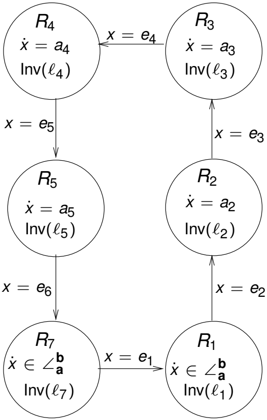
- A finite partition of the plane into convex polygonal sets (regions)
- Dynamics given by the angle determined by two vectors: $\dot{x} \in \angle_{\mathbf{a}}^{\mathbf{b}}$



Polygonal Hybrid Systems (SPDIs)

- An SPDI can be seen as a hybrid automaton

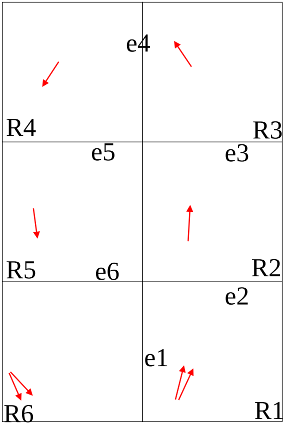




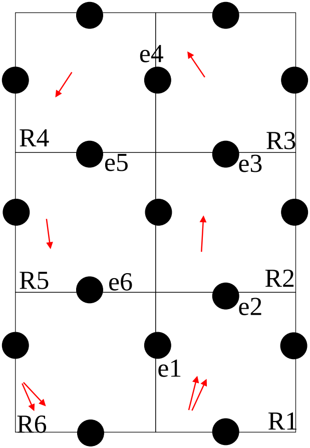
Polygonal Hybrid Systems (SPDIs)
Underlying Graph

- The reachability algorithm operates on a graph representation, not on the automaton

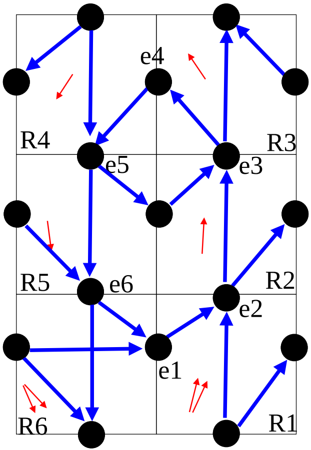
- The reachability algorithm operates on a graph representation, not on the automaton



- The reachability algorithm operates on a graph representation, not on the automaton

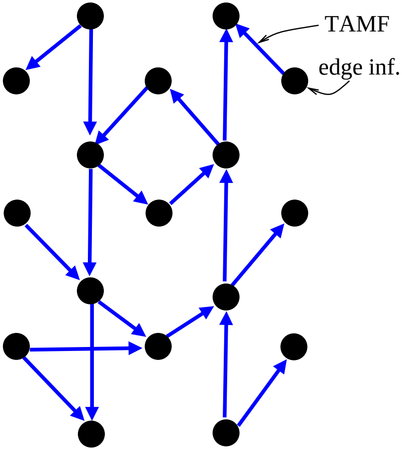


- The reachability algorithm operates on a graph representation, not on the automaton

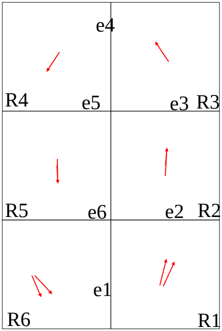


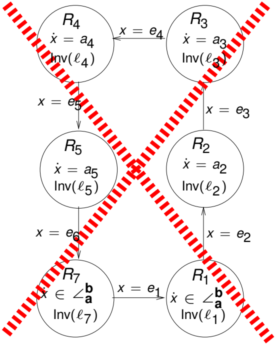
Polygonal Hybrid Systems (SPDIs)
Underlying Graph

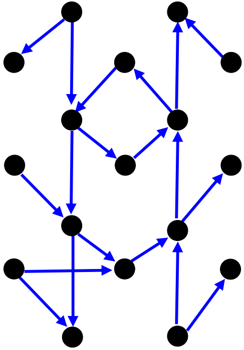
- The reachability algorithm operates on a graph representation, not on the automaton



Polygonal Hybrid Systems (SPDIs)
Three Views







- We will, however, use the geometrical representation in what follows instead for clarity of presentation

Known Results about SPDIs

- Reachability is decidable –in the plane
  (based on Poincaré maps, finite charact. of simple cycles, acceleration, ...)
  - DFS algorithm (HSCC'01)
  - BFS algorithm (VMCAI'04)
  - Tool: SPeeDI (CAV'02)
- Reachability is undecidable –3-dim and higher (ICALP'94)
- For slights extensions in 2-dim reachability is an open question, for others is undecidable (CONCUR'02, FSTTCS'05)
- Phase portrait computation
  - Viability and controllability kernels (HSCC'02)
  - Invariance kernels (NJC'04)
  - Semi-separatrices (FORMATS'06)

Contributors: E. Asarin, O. Maler, V. Mysore, G. Pace, A. Pnueli, G. Schneider, S. Yovine

Outline

1. Introduction
   - Hybrid Systems
   - Polygonal Hybrid Systems (SPDIs)
   - Motivation
2. Phase Portrait of SPDIs
   - Controllability and Viability Kernels
3. Independent Questions and Parallelization
   - SPDI Decomposition
   - Unavoidable Kernels
   - Counting Subproblems
4. Parallel Algorithm for Reachability

Motivation

- Application: Use of SPDIs for approximating non-linear differential equations
  - Triangulation of the plane: Huge number of regions
- Need to reduce the complexity ... without too much overhead
  - Static analysis to reduce the state space
  - Parallelizing the reachability algorithm

Motivation

- Application: Use of SPDIs for approximating non-linear differential equations
  - Triangulation of the plane: Huge number of regions
- Need to reduce the complexity ... without too much overhead
  - Static analysis to reduce the state space
  - Parallelizing the reachability algorithm

Parallelization

- Reduction of memory and time requirements are the main reasons for seeking parallelization
  - In verification the main bottleneck is usually memory
- The challenge:

  To partition the task among different processes keeping a balanced distribution of the use of memory and execution time... without a high communication cost

  And, if possible

  compositionally
- Remark: Hybrid system are, by nature, non compositional

Parallelization

- Reduction of memory and time requirements are the main reasons for seeking parallelization
  - In verification the main bottleneck is usually memory
- The challenge:

  To partition the task among different processes keeping a balanced distribution of the use of memory and execution time... without a high communication cost

  And, if possible

  compositionally

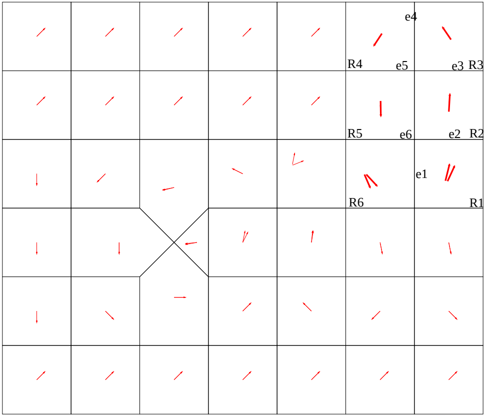
- Remark: Hybrid system are, by nature, non compositional

Outline

1. Introduction
   - Hybrid Systems
   - Polygonal Hybrid Systems (SPDIs)
   - Motivation
2. Phase Portrait of SPDIs
   - Controllability and Viability Kernels
3. Independent Questions and Parallelization
   - SPDI Decomposition
   - Unavoidable Kernels
   - Counting Subproblems
4. Parallel Algorithm for Reachability

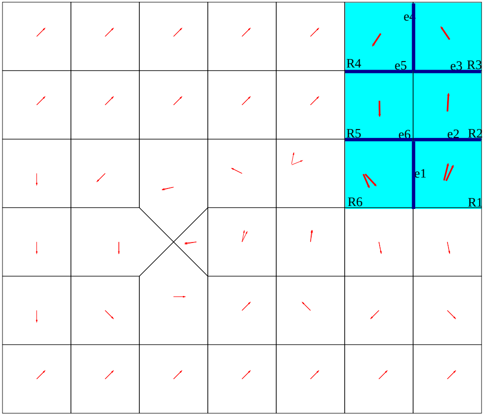
Few Preliminaries

- We only need to consider simple cycles
  - Given a sequence of non-repeating edges (except for the first and last edge) – e.g., $\sigma = e_1, \cdots, e_k, e_1$
  - Consider the polygonal subset of the SPDI determined by such sequence (denoted K_σ)



Few Preliminaries

- We only need to consider simple cycles
  - Given a sequence of non-repeating edges (except for the first and last edge) – e.g., $\sigma = e_1, \cdots, e_k, e_1$
  - Consider the polygonal subset of the SPDI determined by such sequence (denoted K_σ)

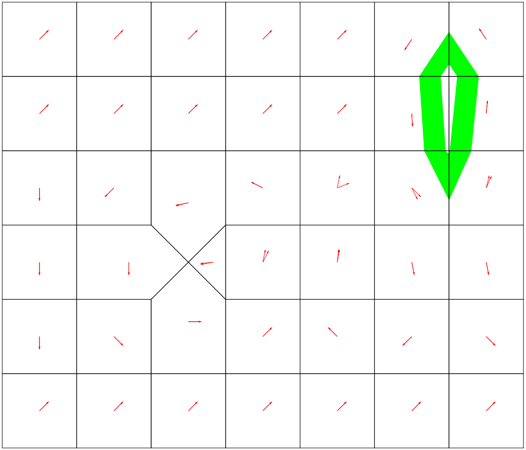


Controllability Kernels

- Given K_σ, its controllability kernel is the largest subset such that any two points are reachable from each other

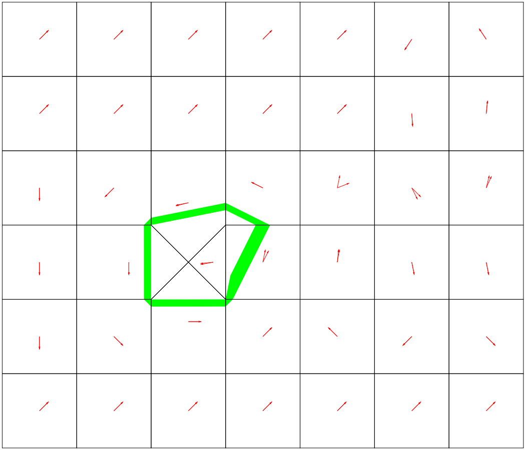
Controllability Kernels

- Given K_σ, its controllability kernel is the largest subset such that any two points are reachable from each other



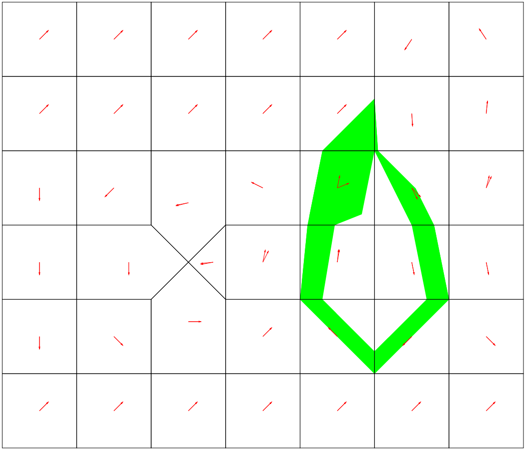
Controllability Kernels

- Given K_σ, its controllability kernel is the largest subset such that any two points are reachable from each other



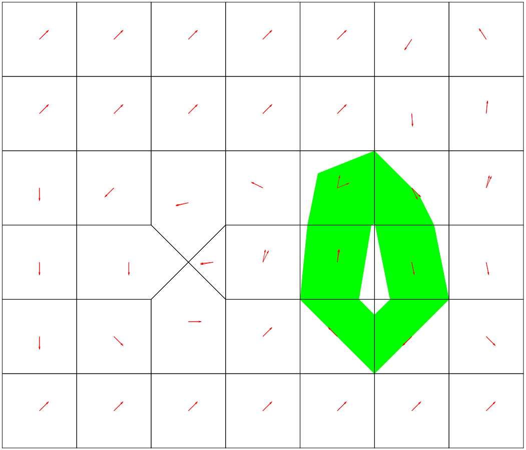
Controllability Kernels

- Given K_σ, its controllability kernel is the largest subset such that any two points are reachable from each other



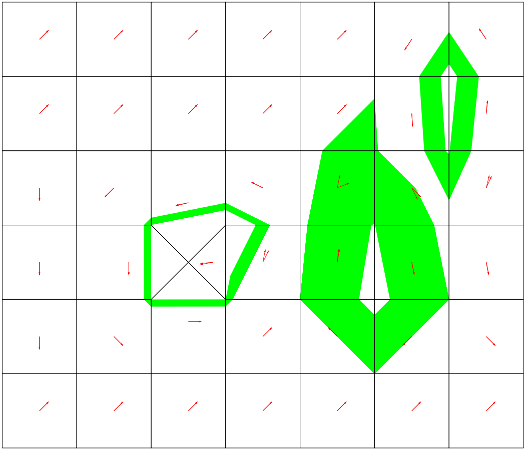
Controllability Kernels

- Given K_σ, its controllability kernel is the largest subset such that any two points are reachable from each other



Controllability Kernels

- Given K_σ, its controllability kernel is the largest subset such that any two points are reachable from each other

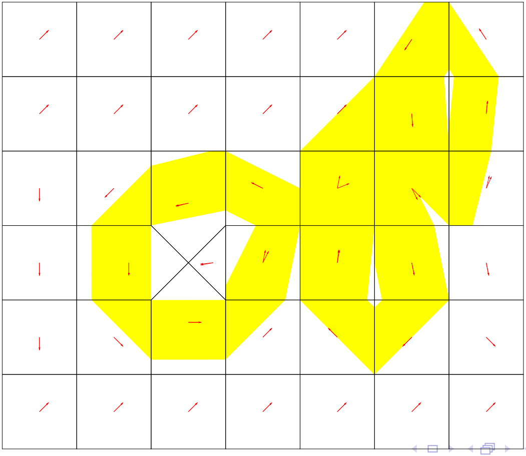


Viability Kernels

- Given K_σ, its viability kernel is the largest subset such that for any point in the set, there is at least one trajectory which remains in the set forever

Viability Kernels

- Given K_σ, its viability kernel is the largest subset such that for any point in the set, there is at least one trajectory which remains in the set forever



Outline

1 Introduction
- Hybrid Systems
- Polygonal Hybrid Systems (SPDIs)
- Motivation

2 Phase Portrait of SPDIs
- Controllability and Viability Kernels

3 Independent Questions and Parallelization
- SPDI Decomposition
- Unavoidable Kernels
- Counting Subproblems

4 Parallel Algorithm for Reachability

SPDI Decomposition

Given an SPDI and a reachability question, for each controllability kernel, we can:

1. Answer the reachability question without any further analysis;
2. Reduce the state space necessary for reachability analysis; or
3. Decompose the reachability question into two smaller, and independent reachability questions

1. Immediate Answer

- Two interesting properties:
  - Within the controllability kernel of a loop, any two points are mutually reachable
  - Any point on the viability kernel of the same loop can eventually reach the controllability kernel

Theorem 1

Given an SPDI $\mathcal{S}$, K_σ, and two points I and I', if

1. $I \subseteq \text{Viab}(K_\sigma)$, and
2. $I' \subseteq \text{Cntr}(K_\sigma)$

then $\text{REACH}(S, I, I')$.

1. Immediate Answer

- Two interesting properties:
  - Within the controllability kernel of a loop, any two points are mutually reachable
  - Any point on the viability kernel of the same loop can eventually reach the controllability kernel

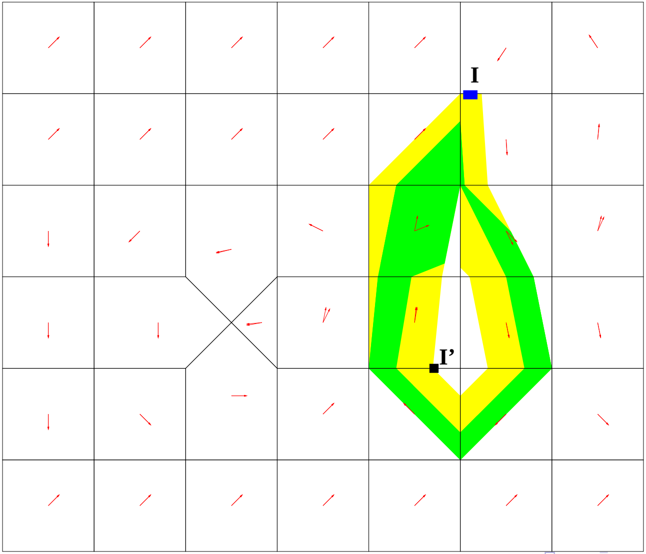
Theorem 1

Given an SPDI $\mathcal{S}$, K_σ, and two points I and I', if

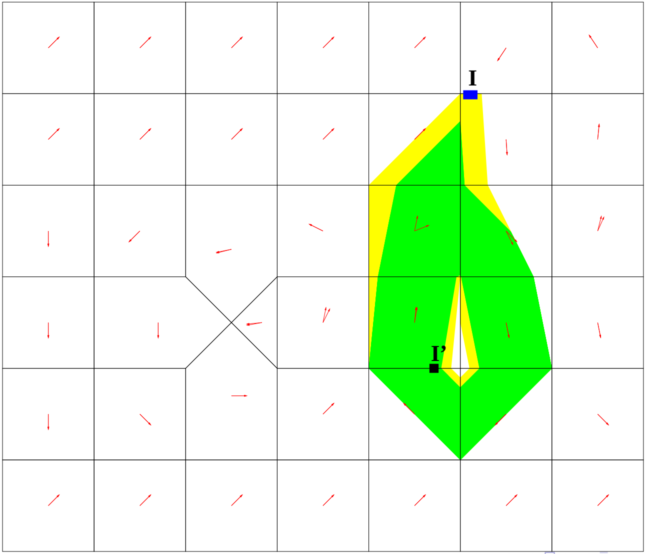
1. $I \subseteq \text{Viab}(K_\sigma)$, and
2. $I' \subseteq \text{Cntr}(K_\sigma)$

then $\textsc{Reach}(S, I, I')$.

1. Immediate Answer
Example



1. Immediate Answer
Example



2. Reduction of the State-Space

Theorem 2

Given an SPDI $\mathcal{S}$, two points I and I' and a controllability kernel $C = \text{Cntr}(K_\sigma)$, if

1. $I \subseteq C_{in}$, and
2. $I' \subseteq C_{in}$,

then $\quad \textsc{Reach}(\mathcal{S}, I, I')$ iff $\textsc{Reach}(\mathcal{S} \setminus C_{out}, I, I')$.

Similarly, if

1. $I \subseteq C_{out}$, and
2. $I' \subseteq C_{out}$

then $\quad \textsc{Reach}(\mathcal{S}, I, I')$ iff $\textsc{Reach}(\mathcal{S} \setminus C_{in}, I, I')$.

2. Reduction of the State-Space

Theorem 2

Given an SPDI $\mathcal{S}$, two points I and I' and a controllability kernel $C = \text{Cntr}(K_\sigma)$, if

1. $I \subseteq C_{in}$, and
2. $I' \subseteq C_{in}$,

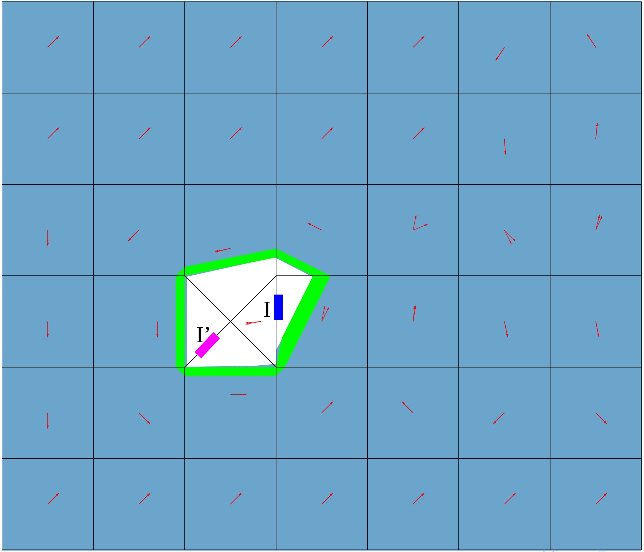
then $\textsc{Reach}(\mathcal{S}, I, I')$ iff $\textsc{Reach}(\mathcal{S} \setminus C_{out}, I, I')$.

Similarly, if

1. $I \subseteq C_{out}$, and
2. $I' \subseteq C_{out}$

then $\textsc{Reach}(\mathcal{S}, I, I')$ iff $\textsc{Reach}(\mathcal{S} \setminus C_{in}, I, I')$.

2. Reduction of the State-Space
Example



3. Decomposition into Independent Questions

Theorem 3

Given an SPDI $\mathcal{S}$, two points I and I' and a controllability kernel $C = \mathrm{Cntr}(K_\sigma)$, if

1. $I \subseteq C_{in}$ and
2. $I' \subseteq C_{out}$

then

$$\text{REACH}(\mathcal{S}, I, I')$$
$$\text{iff}$$
$$\text{REACH}(\mathcal{S} \setminus C_{out}, I, C) \wedge \text{REACH}(\mathcal{S} \setminus C_{in}, C, I').$$

Similarly, for $I \subseteq C_{out}$, and $I' \subseteq C_{in}$.

3. Decomposition into Independent Questions

Theorem 3

Given an SPDI $\mathcal{S}$, two points I and I' and a controllability kernel $C = \text{Cntr}(K_\sigma)$, if

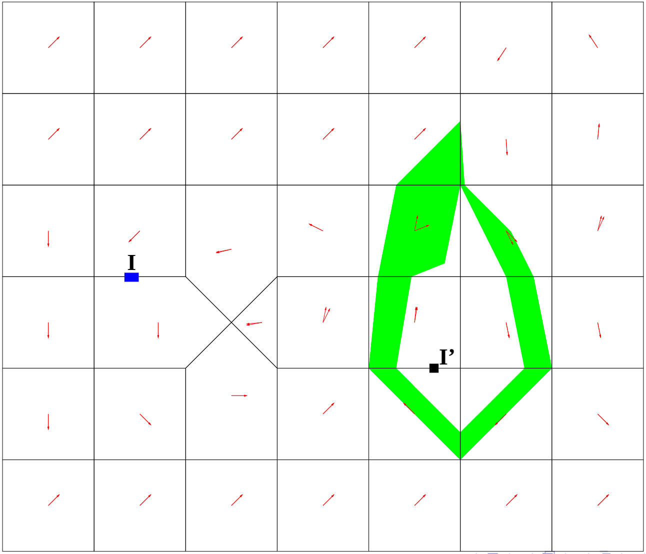
1. $I \subseteq C_{in}$ and
2. $I' \subseteq C_{out}$

then

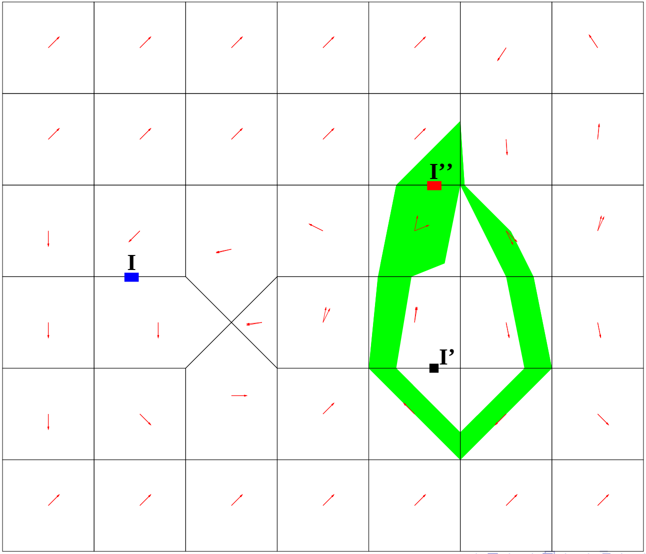
$$\textsc{Reach}(\mathcal{S}, I, I')$$
$$\text{iff}$$
$$\textsc{Reach}(\mathcal{S} \setminus C_{out}, I, C) \wedge \textsc{Reach}(\mathcal{S} \setminus C_{in}, C, I').$$

Similarly, for $I \subseteq C_{out}$, and $I' \subseteq C_{in}$.

3. Decomposition into Independent Questions
Example



3. Decomposition into Independent Questions
Example



Outline

1. Introduction
   - Hybrid Systems
   - Polygonal Hybrid Systems (SPDIs)
   - Motivation
2. Phase Portrait of SPDIs
   - Controllability and Viability Kernels
3. Independent Questions and Parallelization
   - SPDI Decomposition
   - Unavoidable Kernels
   - Counting Subproblems
4. Parallel Algorithm for Reachability

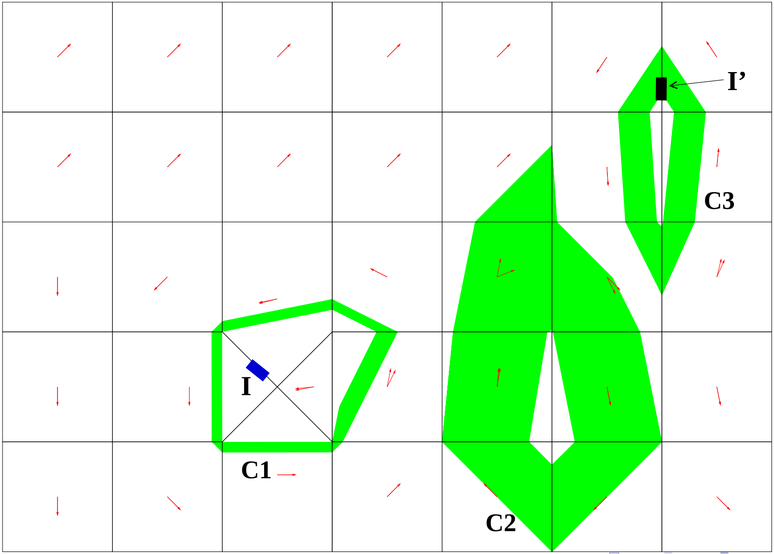
Unavoidable Kernels

Definition

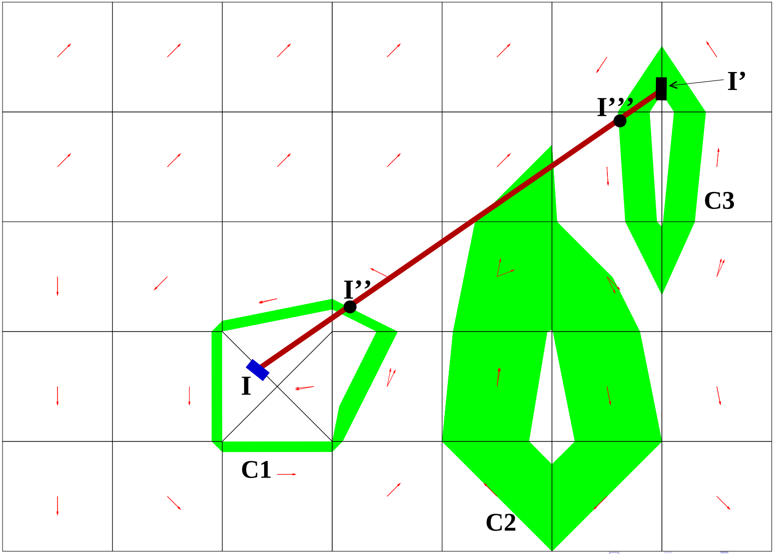
Given an SPDI $\mathcal{S}$ and two points I and I', we say that a controllability kernel $\text{Cntr}(K_\sigma)$ is unavoidable if any segment of line with extremes on points lying on I and I' intersects with both the edges of $\text{Cntr}^l(K_\sigma)$ and those of $\text{Cntr}^u(K_\sigma)$ an odd number of times (disregarding tangential intersections with vertices).

Unavoidable Kernels

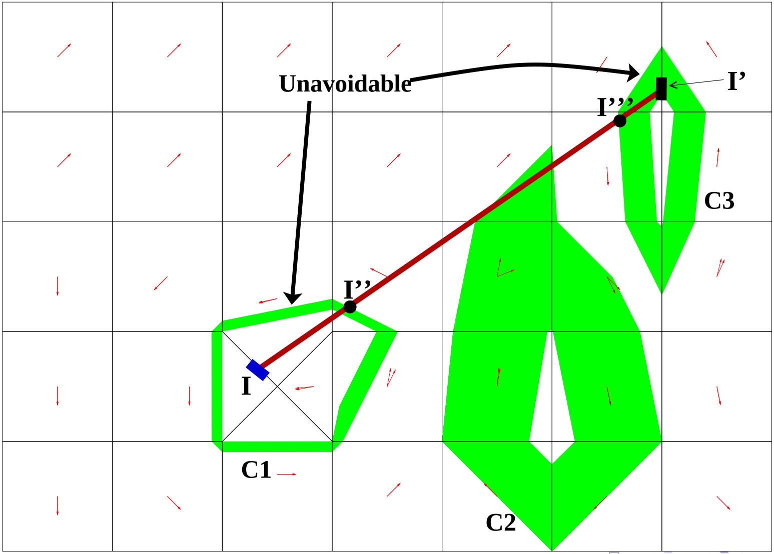
Example



Unavoidable Kernels
Example



Unavoidable Kernels
Example



Unavoidable Kernels

Theorem

Given an SPDI $\mathcal{S}$, two points I and I', and a controllability kernel $C = \text{Cntr}(K_\sigma)$, then C is an unavoidable kernel if and only if one of the following conditions holds

- $I \subseteq C_{in}$ and $I' \subseteq C_{out}$; or
- $I \subseteq C_{out}$ and $I' \subseteq C_{in}$.

Outline

1. Introduction
   - Hybrid Systems
   - Polygonal Hybrid Systems (SPDIs)
   - Motivation
2. Phase Portrait of SPDIs
   - Controllability and Viability Kernels
3. Independent Questions and Parallelization
   - SPDI Decomposition
   - Unavoidable Kernels
   - Counting Subproblems
4. Parallel Algorithm for Reachability

Counting Subproblems

Theorem (upper bound)

Given an SPDI $\mathcal{S}$ and two points I and I', the question $\textsc{Reach}(\mathcal{S}, I, I')$ can be split into no more than **k** reachability questions, **k** is the number of mutually-disjoint controllability kernels

Theorem (lower bound)

Given an SPDI $\mathcal{S}$ and two points I and I', the question $\textsc{Reach}(\mathcal{S}, I, I')$ can be split into at least **u+1** reachability questions, **u** is the number of mutually-disjoint unavoidable controllability kernels

Counting Subproblems

Theorem (upper bound)

Given an SPDI $\mathcal{S}$ and two points I and I', the question $\text{REACH}(\mathcal{S}, I, I')$ can be split into no more than **k** reachability questions, **k** is the number of mutually-disjoint controllability kernels

Theorem (lower bound)

Given an SPDI $\mathcal{S}$ and two points I and I', the question $\text{REACH}(\mathcal{S}, I, I')$ can be split into at least **u+1** reachability questions, **u** is the number of mutually-disjoint unavoidable controllability kernels

Algorithm

```
function ReachPar(𝒮, I, I') =
  ReachParKernels (𝒮, ControllabilityKernels(𝒮), I, I')
```

```
function ReachParKernels(𝒮, [], I, I') =
  Reach(𝒮, I, I');
```

Algorithm

```
function ReachPar(𝒮, I, I') =
  ReachParKernels (𝒮, ControllabilityKernels(𝒮), I, I')
```

```
function ReachParKernels(𝒮, [], I, I') =
  Reach(𝒮, I, I');
```

Algorithm

```
function ReachParKernels(𝒮, k:ks, I, I') =
  if (ImmedieteAnswer(𝒮, I, I')) then
     True;
  elsif (SameSideOfKernel(𝒮, k, I, I')) then
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I');
     ReachParKernels(𝒮_I, ks, I, I');
  else
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I);
     𝒮_I' := 𝒮 \ EdgesOnOtherSideOf(𝒮 k, I');
     parbegin
        r1 := ReachParKernels(𝒮_I, ks, I, viability(k));
        r2 := ReachParKernels(𝒮_I', ks, k, I');
     parend;
     return (r1 and r2);
```

Algorithm

```
function ReachParKernels(𝒮, k:ks, I, I') =
  if (ImmedieteAnswer(𝒮, I, I')) then     {Theorem 1}
     True;
  elsif (SameSideOfKernel(𝒮, k, I, I')) then
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I');
     ReachParKernels(𝒮_I, ks, I, I');
  else
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I);
     𝒮_I' := 𝒮 \ EdgesOnOtherSideOf(𝒮 k, I');
     parbegin
        r1 := ReachParKernels(𝒮_I, ks, I, viability(k));
        r2 := ReachParKernels(𝒮_I', ks, k, I');
     parend;
     return (r1 and r2);
```

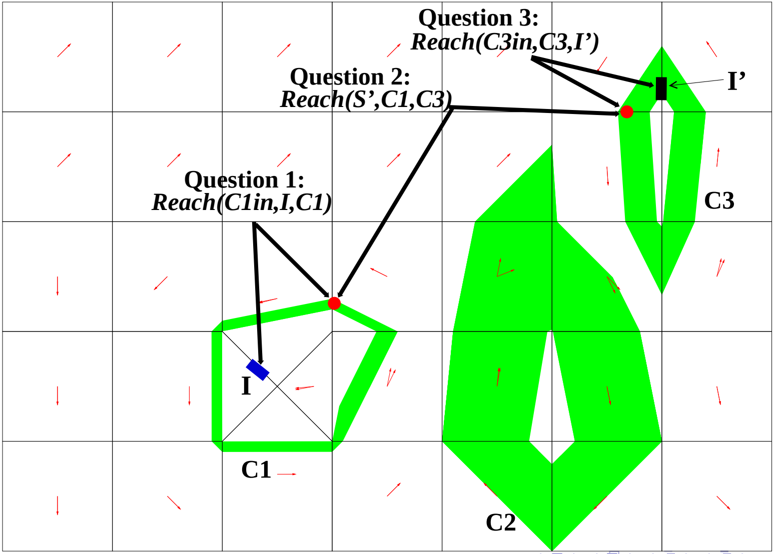
Algorithm

```
function ReachParKernels(𝒮, k:ks, I, I′) =
  if (ImmedieteAnswer(𝒮, I, I′)) then    {Theorem 1}
     True;
  elsif (SameSideOfKernel(𝒮, k, I, I′)) then   {Theorem 2}
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I′);
     ReachParKernels(𝒮_I, ks, I, I′);
  else
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I);
     𝒮_I′ := 𝒮 \ EdgesOnOtherSideOf(𝒮 k, I′);
     parbegin
        r1 := ReachParKernels(𝒮_I, ks, I, viability(k));
        r2 := ReachParKernels(𝒮_I′, ks, k, I′);
     parend;
     return (r1 and r2);
```

Algorithm

```
function ReachParKernels(𝒮, k:ks, I, I') =
  if (ImmedieteAnswer(𝒮, I, I')) then     {Theorem 1}
     True;
  elsif (SameSideOfKernel(𝒮, k, I, I')) then   {Theorem 2}
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I');
     ReachParKernels(𝒮_I, ks, I, I');
  else        {Theorem 3}
     𝒮_I := 𝒮 \ EdgesOnOtherSideOf(𝒮, k, I);
     𝒮_I' := 𝒮 \ EdgesOnOtherSideOf(𝒮 k, I');
     parbegin
        r1 := ReachParKernels(𝒮_I, ks, I, viability(k));
        r2 := ReachParKernels(𝒮_I', ks, k, I');
     parend;
     return (r1 and r2);
```

Parallel (Independent) Reachability Questions
Example



Soundness and Completeness

Theorem

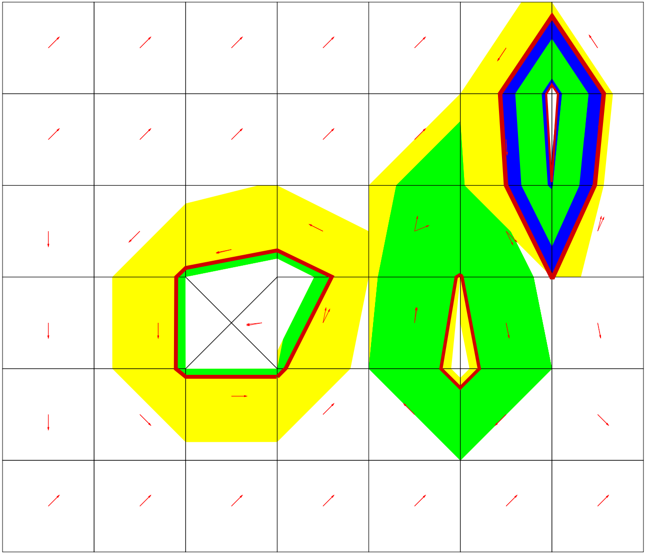
Given an SPDI $\mathcal{S}$ and two points $I \subseteq e$ and $I' \subseteq e'$,

$$\text{REACH}(\mathcal{S}, I, I')$$
iff
$$\text{REACH}_{||}(\mathcal{S}, I, I').$$

- A parallel algorithm for reachability analysis of polygonal hybrid systems
  - Compositional
  - Each parallel task is performed, in general in smaller independent state-spaces
  - No extra work needed to perform the computation of the kernels: identification and analysis of loops is performed in the first part of the reachability algorithm
    - The only extra work is the computation of unavoidable kernels
- Combination of techniques
  - The detection of *unavoidable* kernels may be done by using standard geometrical test (odd-parity test, used in computer graphics)
  - The analysis is then performed on the graph

- A parallel algorithm for reachability analysis of polygonal hybrid systems
  - Compositional
  - Each parallel task is performed, in general in smaller independent state-spaces
  - No extra work needed to perform the computation of the kernels: identification and analysis of loops is performed in the first part of the reachability algorithm
    - The only extra work is the computation of unavoidable kernels
- Combination of techniques
  - The detection of *unavoidable* kernels may be done by using standard geometrical test (odd-parity test, used in computer graphics)
  - The analysis is then performed on the graph

Thank you!



Extensions and Applications

- Not exact extensions to higher dimensions (undecidable)
  - Maybe use the idea for approximations
- Use of SPDIs for approximating non-linear differential equations on the plane
  - Approximation of phase portrait objects

Implementation

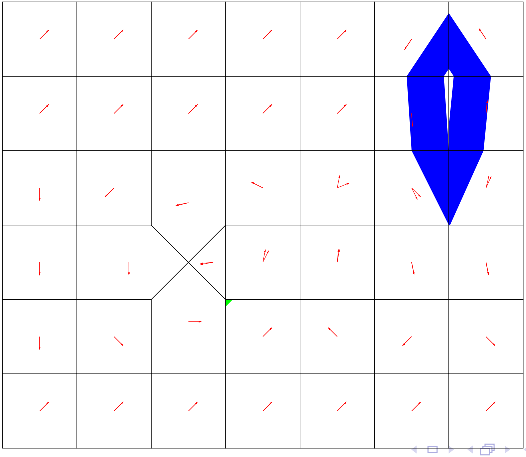
- Implementation in SPeeDI^{+}

Invariance Kernels

- Given K_σ, its invariance kernel is the largest subset such that for any point x in the set, there is at least one trajectory starting in it and every trajectory starting in x is viable

Invariance Kernels

- Given K_σ, its invariance kernel is the largest subset such that for any point x in the set, there is at least one trajectory starting in it and every trajectory starting in x is viable

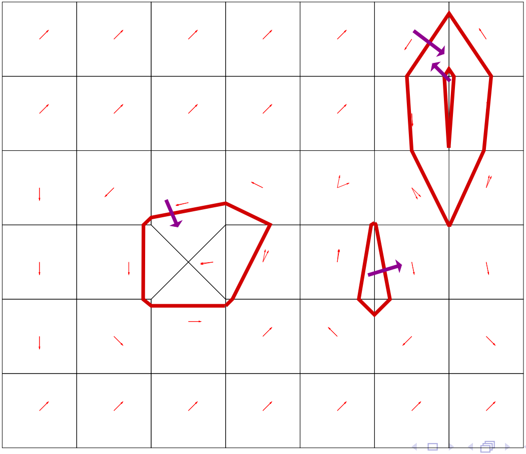


Semi-Separatrices

- A semi-separatrix is a closed curve dissecting the state space into two subsets such that one is reachable from the other but not vice-versa

Semi-Separatrices

- A semi-separatrix is a closed curve dissecting the state space into two subsets such that one is reachable from the other but not vice-versa



Semi-Separatrices

Based on properties of limit trajectories on simple cycles and the invariance kernel we have an algorithm for computing semi-separatrices

Theorem

The computation of semi-separatrices for SPDIs is decidable