Hybrid systems and computer science a short tutorial

Eugene Asarin

Université Paris 7 - LIAFA

Introductory equations

- **Hybrid Systems** = Discrete+Continuous

Introductory equations

- **Hybrid Systems** = Discrete+Continuous
- **Hybrid Automata** = A class of models of Hybrid systems

Introductory equations

- **Hybrid Systems** = Discrete+Continuous
- **Hybrid Automata** = A class of models of Hybrid systems
- **Original motivation (1990)**= physical plant + digital controller

Introductory equations

- **Hybrid Systems** = Discrete+Continuous
- **Hybrid Automata** = A class of models of Hybrid systems
- **Original motivation (1990)**= physical plant + digital controller
- **New applications** = also scheduling, biology, economy, numerics, and more

Introductory equations

- **Hybrid Systems** = Discrete+Continuous
- **Hybrid Automata** = A class of models of Hybrid systems
- **Original motivation (1990)**= physical plant + digital controller
- **New applications** = also scheduling, biology, economy, numerics, and more
- **Hybrid community** = Control scientists' + Applied mathematicians + Some computer scientists'

Outline

1. Hybrid automata - the model
2. Verification
3. Conclusions and perspectives

1. The Model

Outline

1. Hybrid automata - the model
    - The definition
    - Semantic issues
    - Modeling with hybrid automata
    - “Hybrid” languages
    - Running a hybrid automaton
2. Verification
3. Conclusions and perspectives

The first example

I’m sorry, a thermostat.

The first example

I’m sorry, a thermostat.

- When the heater is OFF, the room cools down :

$$\dot{x} = -x$$

- When it is ON, the room heats:

$$\dot{x} = H - x$$

The first example

I'm sorry, a thermostat.

- When the heater is OFF, the room cools down :

$$\dot{x} = -x$$

- When it is ON, the room heats:

$$\dot{x} = H - x$$

- When t>M it switches OFF
- When t<m it switches ON

The first example

I'm sorry, a thermostat.

- When the heater is OFF, the room cools down :

$$\dot{x} = -x$$

- When it is ON, the room heats:

$$\dot{x} = H - x$$

- When t>M it switches OFF
- When t<m it switches ON

A strange creature. . .

A bad syntax

Some mathematicians prefer to write

$$\dot{x} = f(x, q)$$

where

$$\begin{aligned} f(x, \mathrm{Off}) &= -x \\ f(x, \mathrm{On}) &= H - x \end{aligned}$$

with some switching rules on q.

A bad syntax

Some mathematicians prefer to write

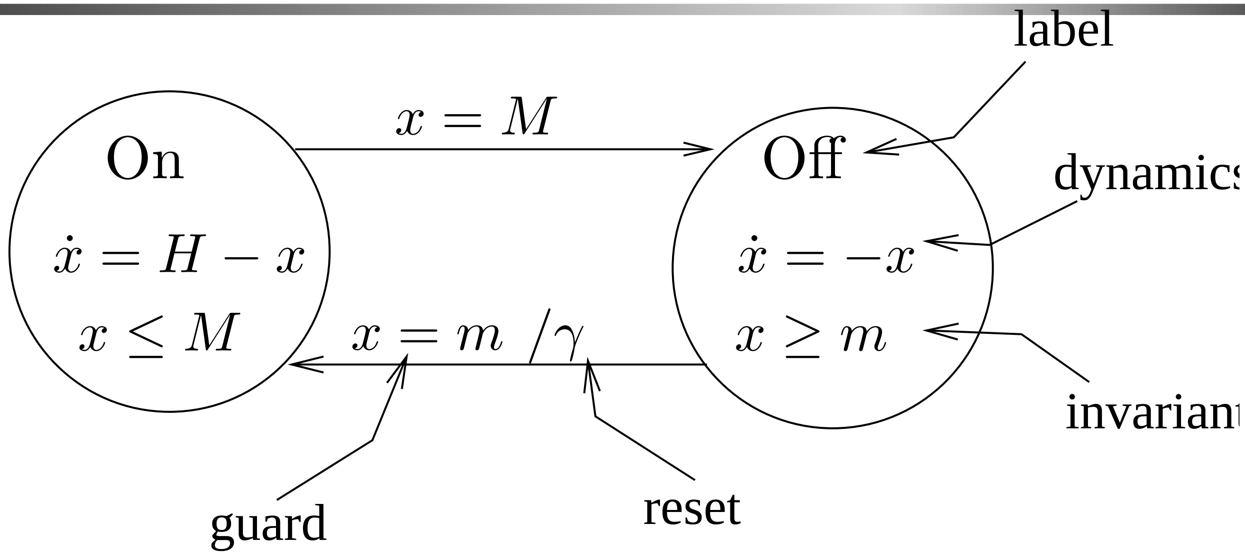
$$\dot{x} = f(x, q)$$

where

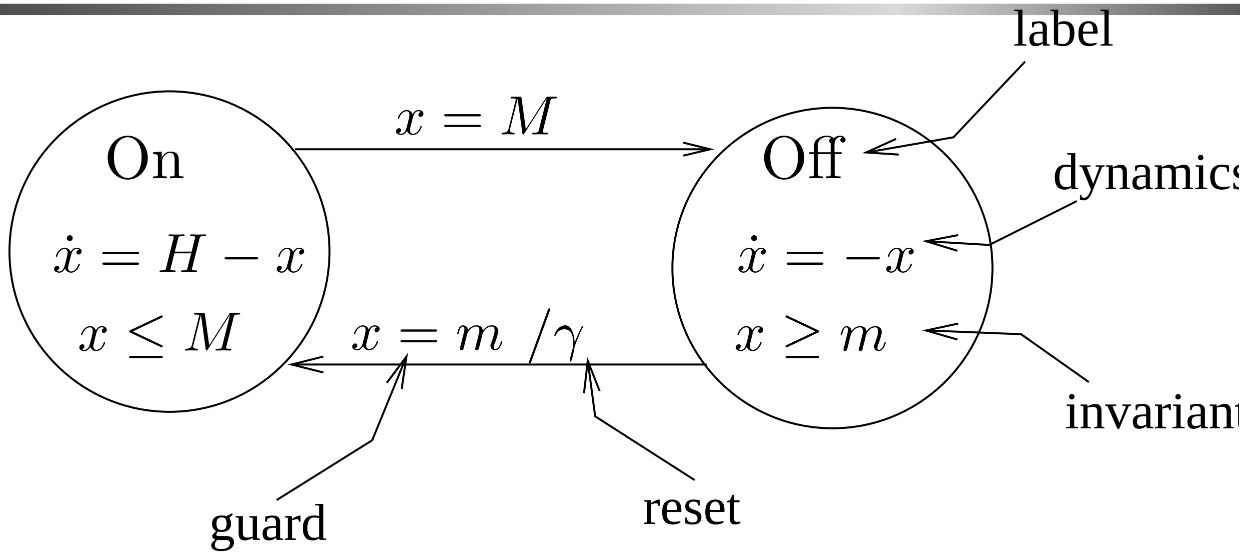
$$\begin{aligned} f(x, \mathrm{Off}) &= -x \\ f(x, \mathrm{On}) &= H - x \end{aligned}$$

with some switching rules on q.
But we will draw an automaton!

Hybrid automaton

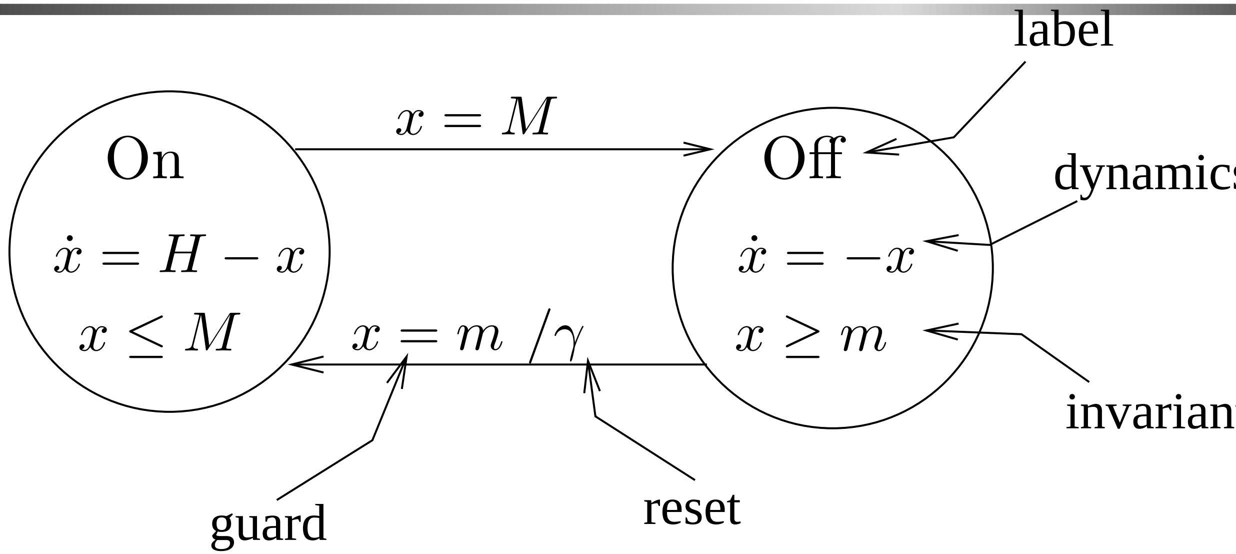


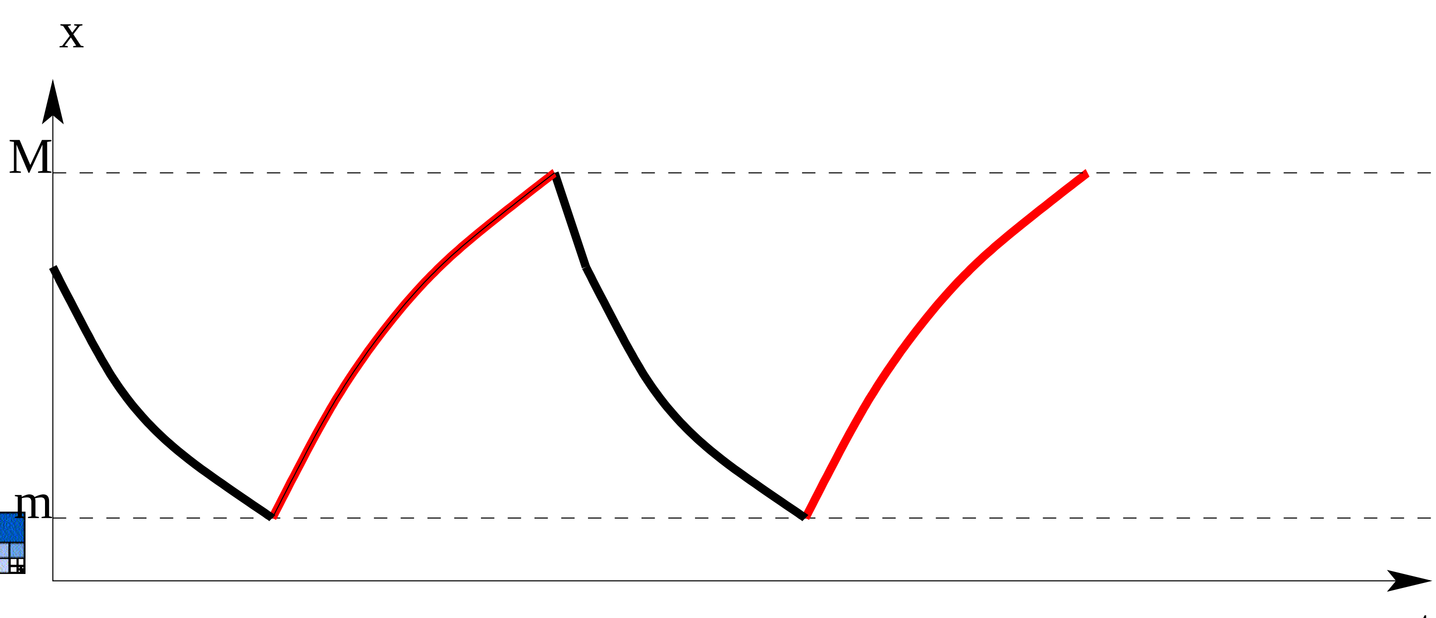
Hybrid automaton



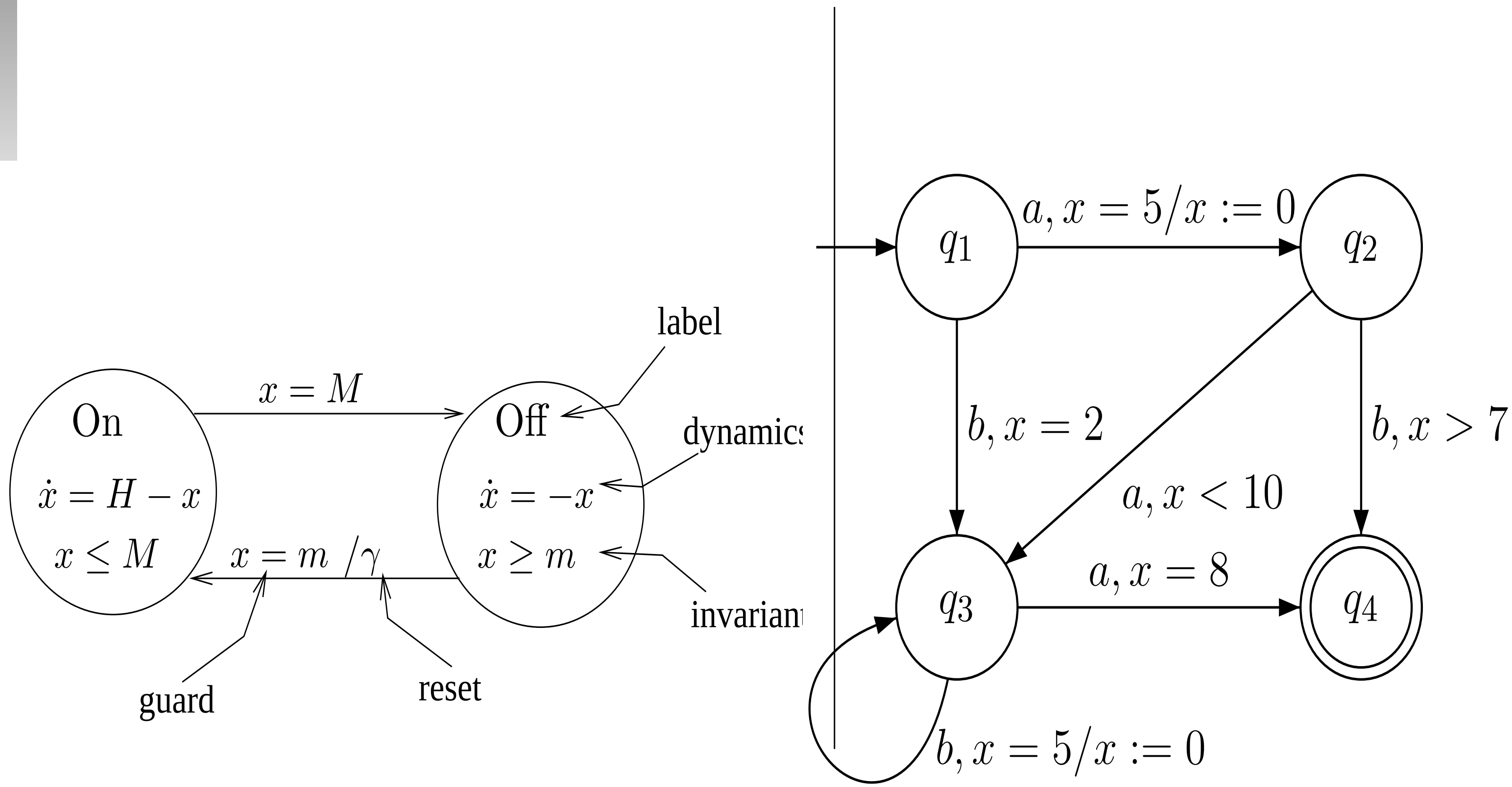
A formal definition: It is a tuple . . .

Hybrid automaton





Hybrid versus timed



Hybrid versus timed

Element	Timed Aut.	Hybrid Aut.
Discrete locations	$q \in Q$ (finite)	$q \in Q$ (finite)
Continuous variables	$\vec{x} \in \mathbb{R}^n$	$\vec{x} \in \mathbb{R}^n$
x dynamics	$\dot{x} = 1$	$\dot{x} = f(x)$ (and more)
Guards	bool. comb. of $x_i \leq c_i$	$\vec{x} \in G$

Semantic issues

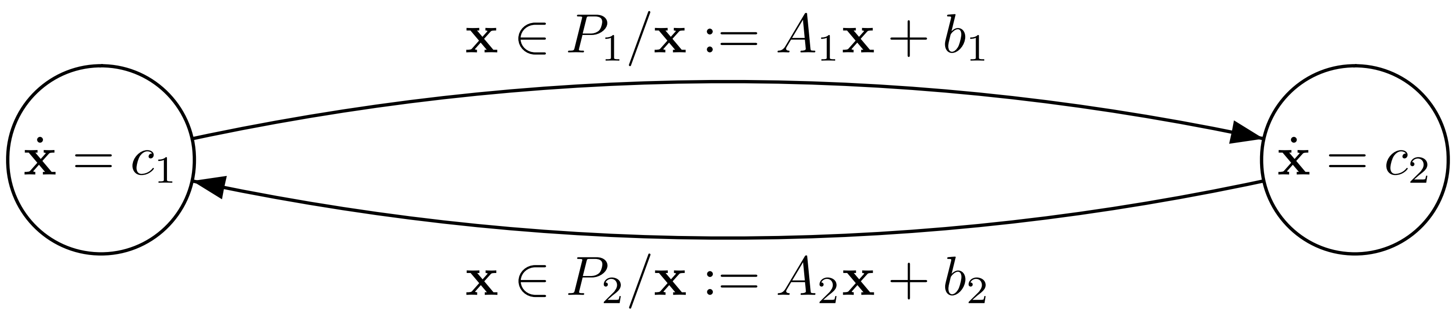
- A trajectory (run) is an $f : \mathbb{R} \rightarrow Q \times \mathbb{R}^n$
- Some mathematical complications (notion of solution, existence and unicity not so evident).
- Zeno trajectories (infinitely many transitions in a finite period of time).
  - can be forbidden
  - one can consider trajectories up to the first anomaly (Sastry et al., everything OK)
  - one can consider the complete Zeno trajectories (very funny : Asarin-Maler 95)

Variants

- Discrete-time ($x_{n+1} = f(x_n)$) or continuous-time $\dot{x} = f(x)$
- Deterministic (e.g. $\dot{x} = f(x)$) or non-deterministic (e.g. $\dot{x} \in F(x)$)
- Eager or lazy.
- With control and/or disturbance (e.g. $\dot{x} = f(x, u, d)$)
- Various restrictions on dynamics, guards and resets: “Piecewise trivial dynamics”. LHA, RectA, PCD, PAM, SPDI . . . They are still highly non-trivial.

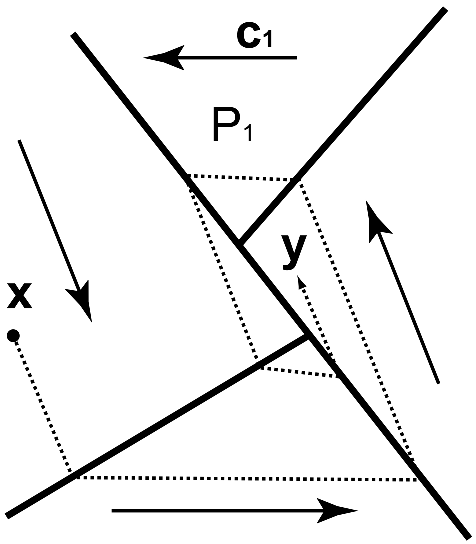
Special classes of Hybrid Automata 1

- The famous one: *Linear Hybrid Automata*



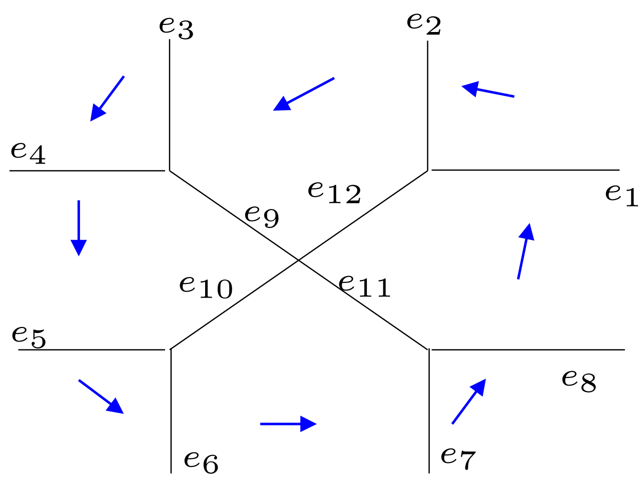
Special classes of Hybrid Automata 2

- My favorite: *PCD* = Piecewise Constant Derivatives

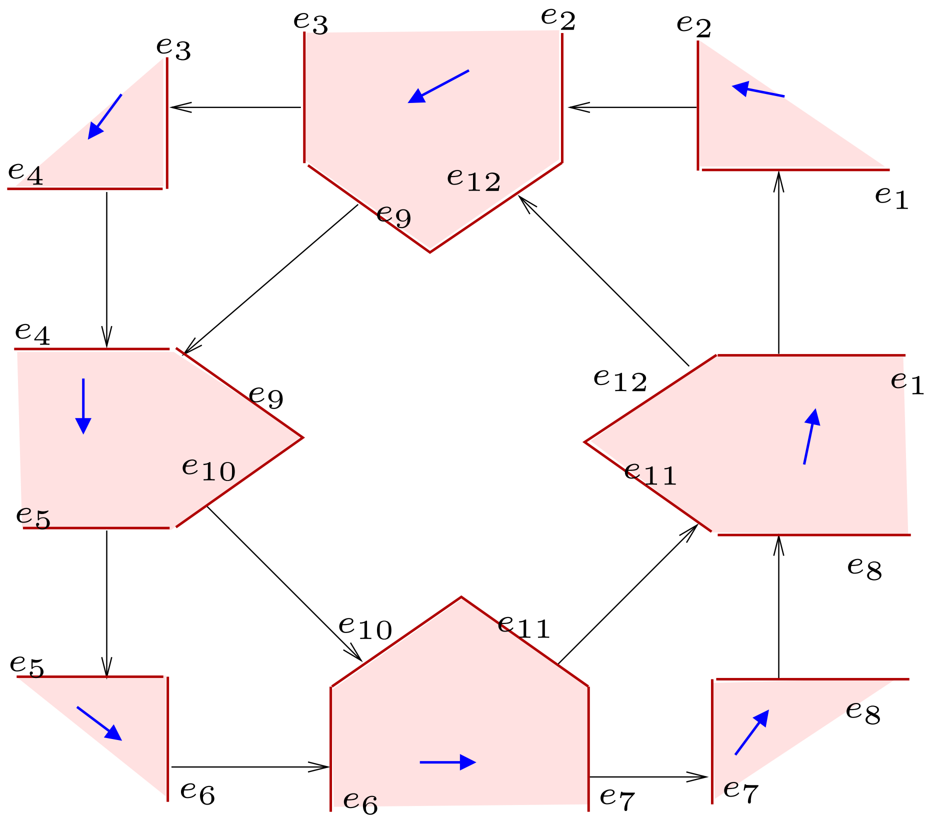


$\dot{\mathbf{x}} = \mathbf{c}_i$ for $\mathbf{x} \in P_i$

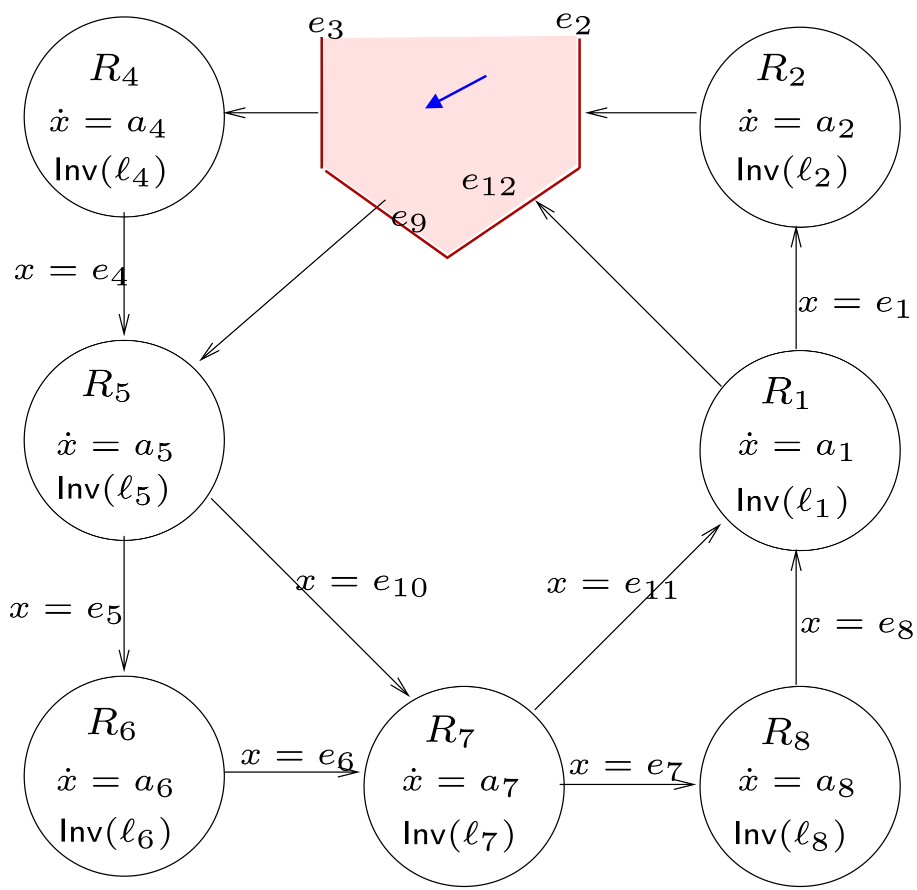
PCD is a linear hybrid automaton (LHA)



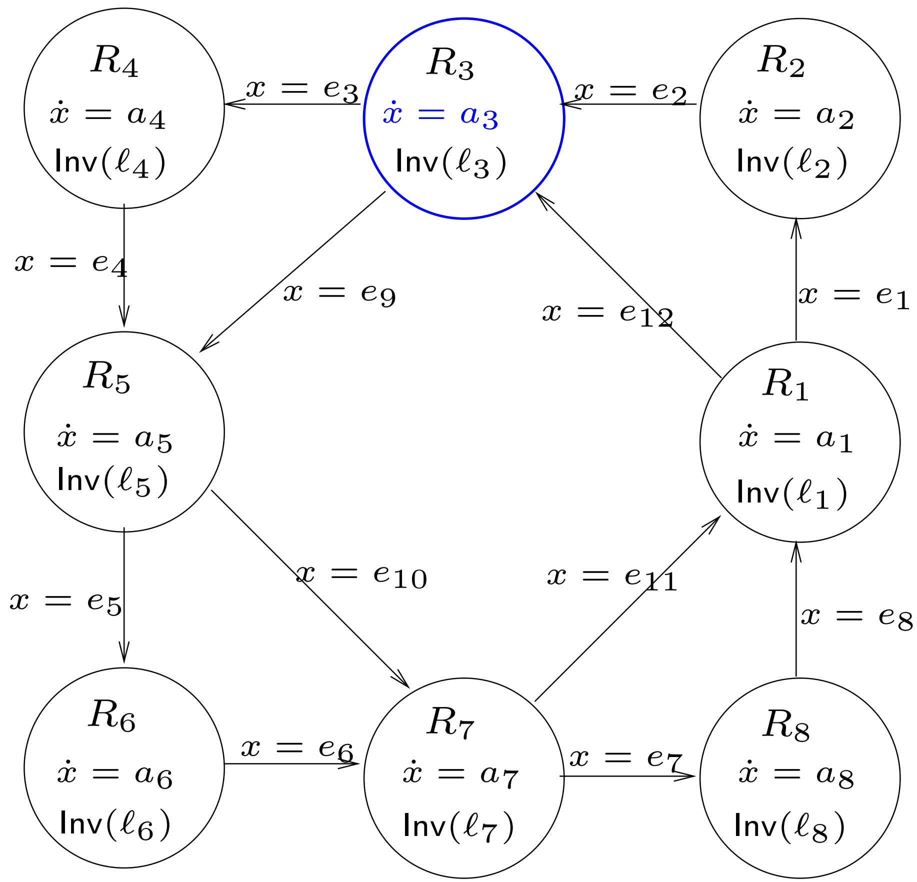
PCD is a linear hybrid automaton (LHA)



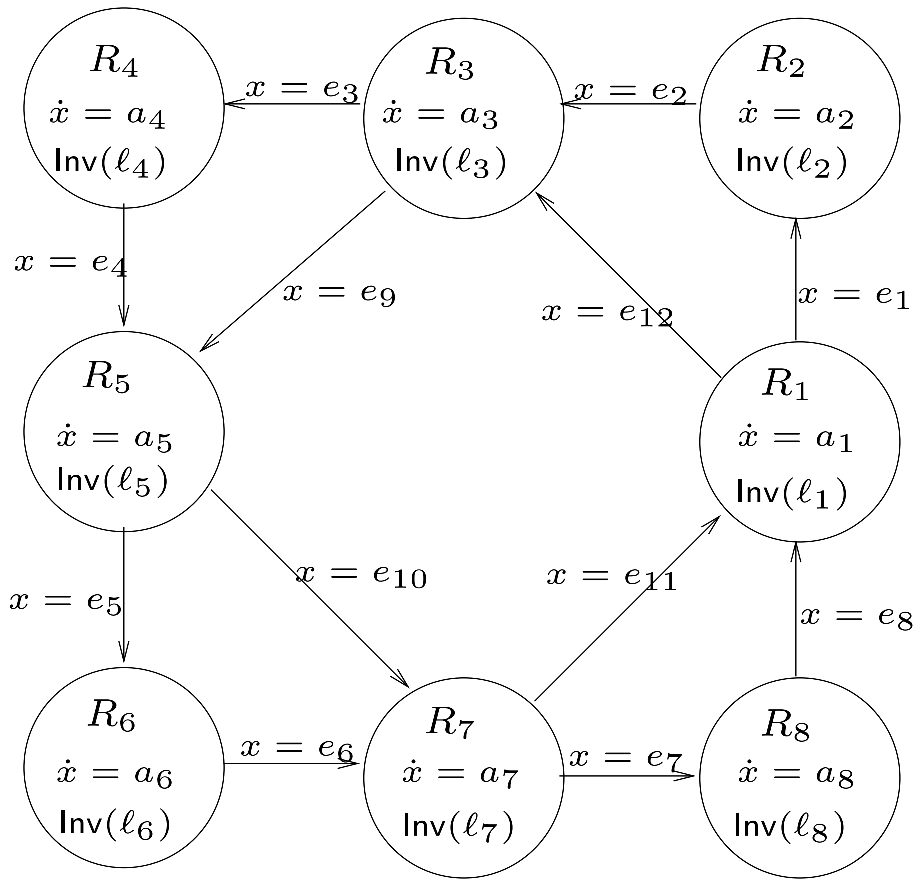
PCD is a linear hybrid automaton (LHA)



PCD is a linear hybrid automaton (LHA)

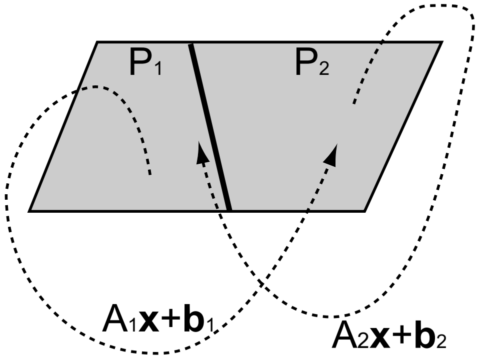


PCD is a linear hybrid automaton (LHA)



Special classes of Hybrid Automata 3

- The most illustrative: *Piecewise Affine Maps*



$$\mathbf{x} := A_i\mathbf{x} + \mathbf{b}_i \text{ for } \mathbf{x} \in P_i$$

How to model?

- a control system

How to model?

- a control system
- a scheduler with preemption

How to model?

- a control system
- a scheduler with preemption
- a genetic network

How to model?

- a control system
- a scheduler with preemption
- a genetic network

A network of interacting Hybrid automata

Hybrid languages

- SHIFT
- Charon
- Hysdel
- IF, Uppaal (Timed + ε)
- why not Simulink? or Simulink+CheckMate.

What to do with a hybrid model

- Simulate
  - With Matlab/Simulink
  - With dedicated tools
- Analyze with techniques from control science:
  - Stability analysis
  - Optimal control
  - etc..
- Analyze with your favorite techniques. The most important invention is the model.

2. Verification

1. Hybrid automata - the model ✓
2. Verification
   - Verification and reachability problems
   - Exact methods
     - The curse of undecidability
     - Decidable classes
     - Can realism help?
   - Approximate methods
     - The abstract algorithm
     - Data structures and concrete algorithms
   - Beyond reachability, beyond verification
   - Verification tools
3. Conclusions and perspectives

Verification and reachability problems

- Is automatic verification possible for HA?

Verification and reachability problems

- Is automatic verification possible for HA?
- *Safety:* are we sure that HA never enters a bad state?
- It can be seen as reachability : verify that

$$\neg \text{Reach}(Init, Bad)$$

Verification and reachability problems

- Is automatic verification possible for HA?
- *Safety:* are we sure that HA never enters a bad state?
- It can be seen as reachability : verify that

$$\neg \mathsf{Reach}(Init, Bad)$$

- It is a natural and challenging mathematical problem.
- Many works on decidability
- Some works on approximated techniques

The reachability problem

Given a hybrid automaton $\mathcal{H}$ and two sets $A, B \subset Q \times \mathbb{R}^n$, find out whether there exists a trajectory of $\mathcal{H}$ starting in A and arriving to B. All parameters rational.

Exact methods: Decidable classes

$\mathsf{Reach}(x, y) \Leftrightarrow \exists$ a trajectory from x to y

Reach is decidable for

- AD: timed automata
- HKPV95: initialized rectangular automata, extensions of timed automata
- LPY01: special linear equations + full resets.

Method : finite bisimulation
(stringent restrictions on the dynamics)

KPSY: Integration graphs???

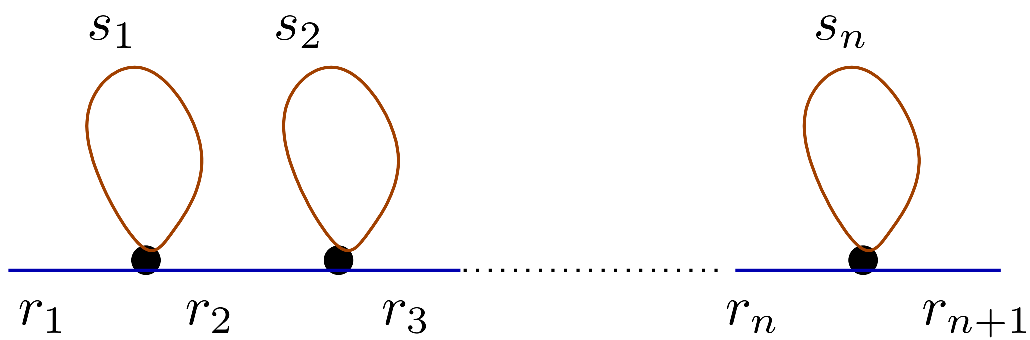
Decidability 2

Reach is decidable for

- MP94: 2d PCD.
- CV96: 2d multi-polynomial systems.
- ASY01: 2d “non-deterministic PCD”

Decidability 2 - geometric method

- consider signatures
- signatures are simple on the plane (Poincaré-Bendixson)



finitely many patterns

- exact acceleration of the cycles is possible.
- Algorithm: for each pattern compute successors, accelerate cycles.

Restrictions: planarity, no jumps

Exact methods: The curse of undecidability

- Koiran et al.: Reach is undecidable for 2d PAM.
- AM95: Reach is undecidable for 3d PCD.
- HPKV95 Many results of the type : “3clocks + 2 stopwatches = undecidable”

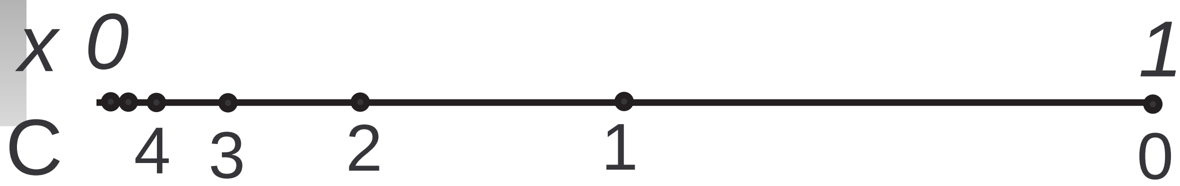
Proof method: simulation of 2-counter (Minsky) machine, TM etc...

- A counter: values in $\mathbb{N}$; operations: $C++$, $C--$; test $C > 0$?
- A Minsky (2 counter) machine

$$\begin{array}{lll} q_1: & D++; & \text{goto } q_2 \\ q_2: & C--; & \text{goto } q_3 \\ q_3: & \text{if } C>0 & \text{then goto } q_2 \text{ else } q_1 \end{array}$$

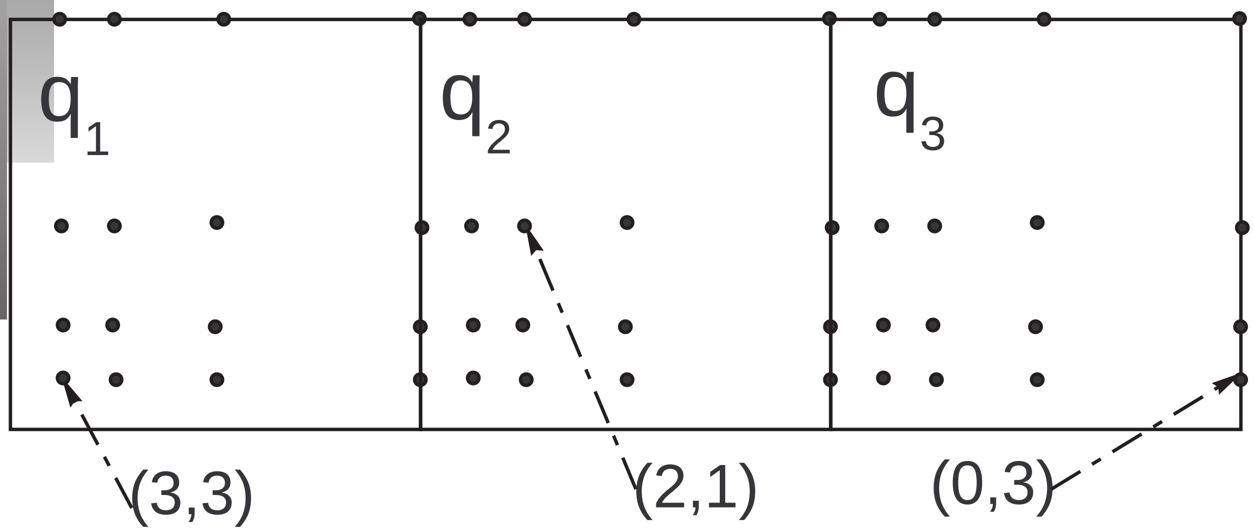
- Reachability is undecidable (and Σ^0_1-complete) for Minsky machines.

Simulating a counter



Counter	PAM
State space $\mathbb{N}$	State space $[0;1]$
State $C = n$	$x = 2^{-n}$
$C++$	$x := x/2$
$C--$	$x := 2x$
$C > 0?$	$x < 0.75?$

Encoding a state of a Minsky Machine



Minsky Machine	PAM
State space $\{q_1, \ldots, q_k\} \times \mathbb{N} \times \mathbb{N}$	State space $[1; k+1]$
State $(q_i, C = m, D = n)$	$x = i + 2^{-m}, y = 2^{-n}$

Simulating a Minsky Machine

Minsky Machine	PAM
State space $\{q_1, \ldots, q_k\} \times \mathbb{N} \times \mathbb{N}$	State space $[1; k+1] \times [0; 1]$
State $(q_i, C = m, D = n)$	$x = i + 2^{-m}, y = 2^{-n}$
q_1 : $D++$; goto q_2	$\begin{cases} x := x + 1 \\ y := y/2 \end{cases}$ if $1 < x \leq 2$
q_2 : $C--$; goto q_3	$\begin{cases} x := 2(x-2) + 3 \\ y := y \end{cases}$ if $2 < x \leq 3$
q_3 : if $C > 0$ then goto q_2 else q_1	$\begin{cases} x := x - 1 \\ y := y \end{cases}$ if $3 < x < 4$ $\begin{cases} x := x - 2 \\ y := y \end{cases}$ if $x = 4$

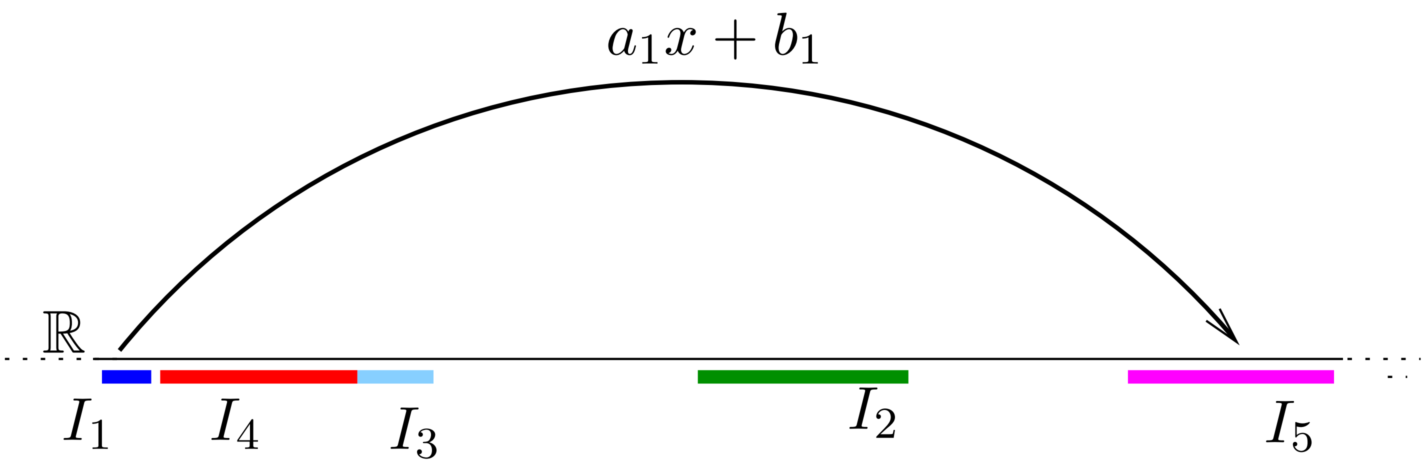
. . . finally

we have proved that Reach is undecidable for 2d PAMs.

Undecidability proofs for other classes of HA are similar.

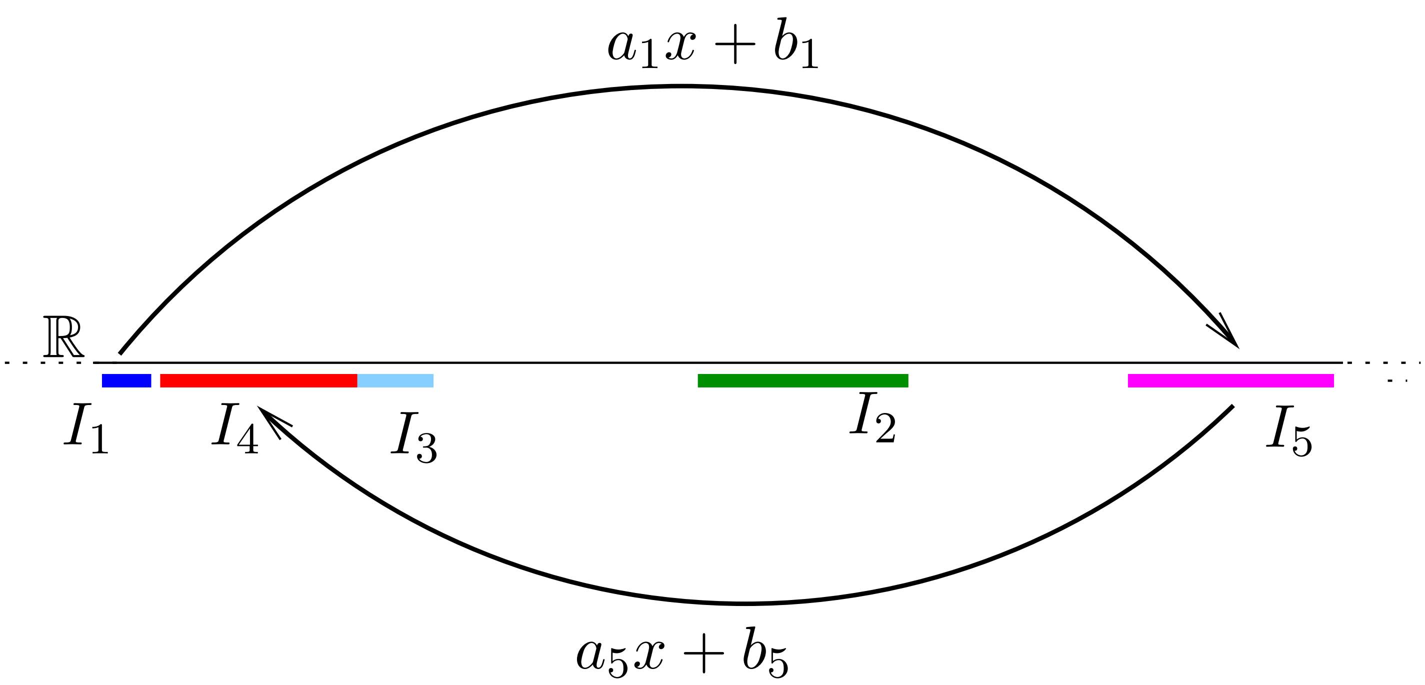
A difficult problem

- 1d piecewise affine maps (PAMs): $f : \mathbb{R} \to \mathbb{R}$
  $f(x) = a_i x + b_i$ for $x \in I_i$



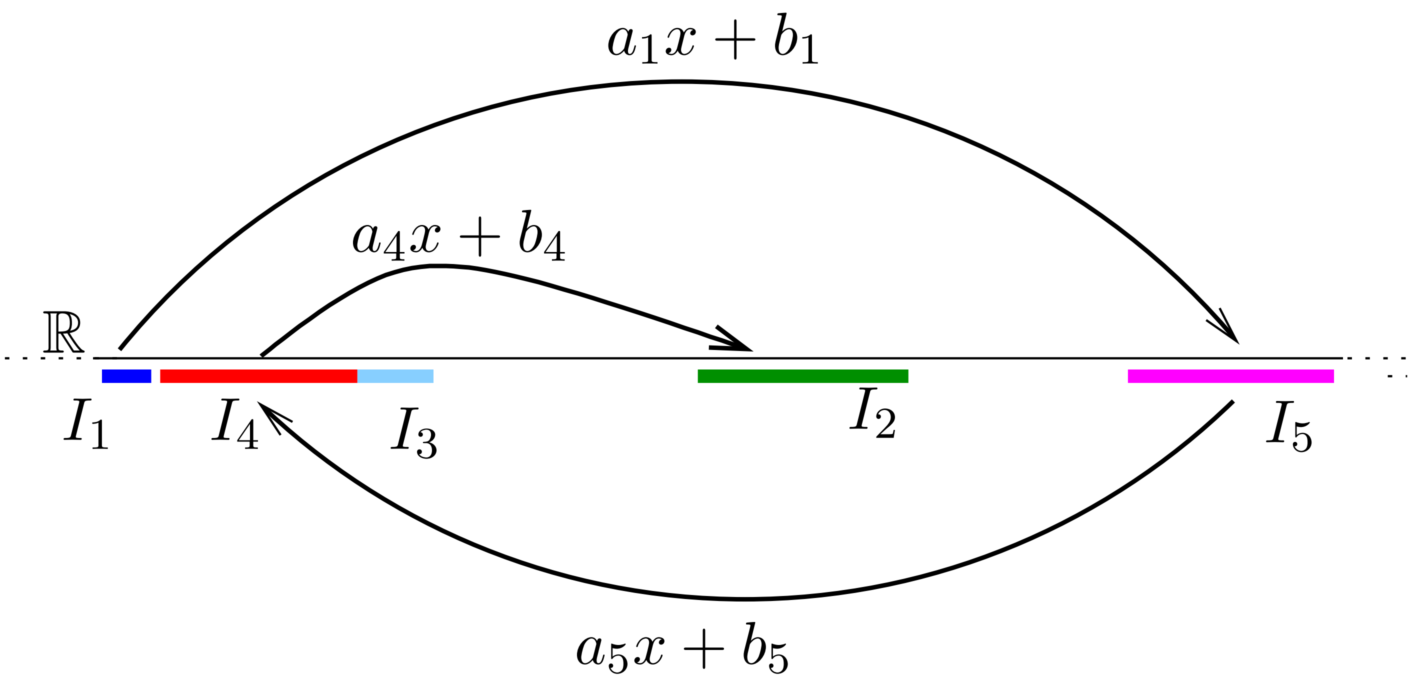
A difficult problem

- 1d piecewise affine maps (PAMs): $f : \mathbb{R} \to \mathbb{R}$
  $f(x) = a_i x + b_i$ for $x \in I_i$



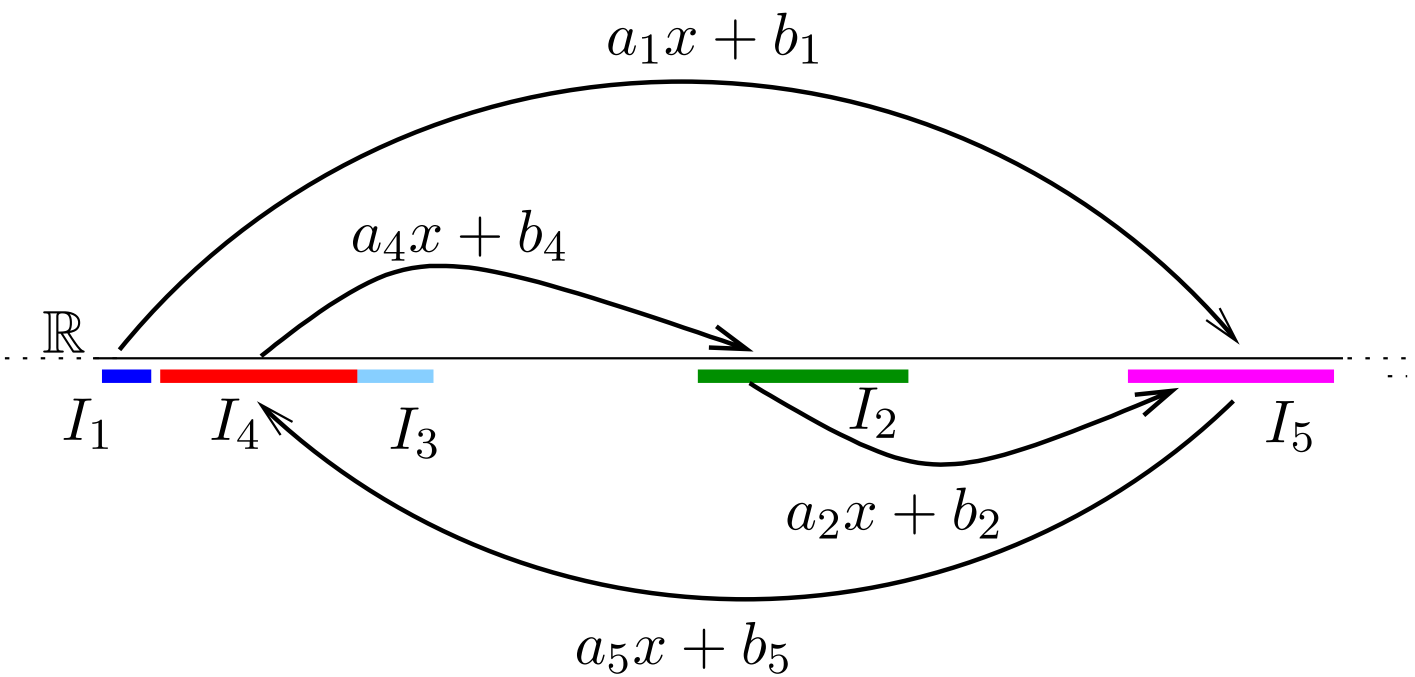
A difficult problem

- 1d piecewise affine maps (PAMs): $f : \mathbb{R} \to \mathbb{R}$
  $f(x) = a_i x + b_i$ for $x \in I_i$



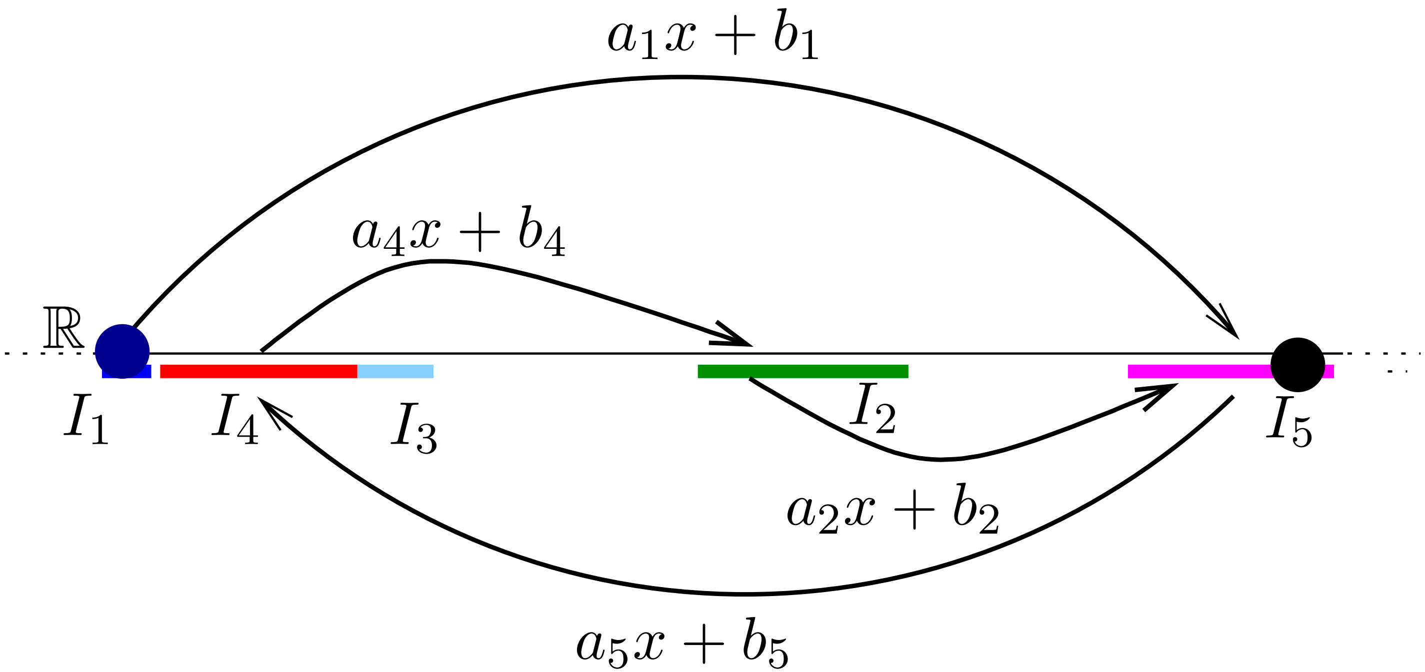
A difficult problem

- 1d piecewise affine maps (PAMs): $f : \mathbb{R} \to \mathbb{R}$
  $f(x) = a_i x + b_i$ for $x \in I_i$



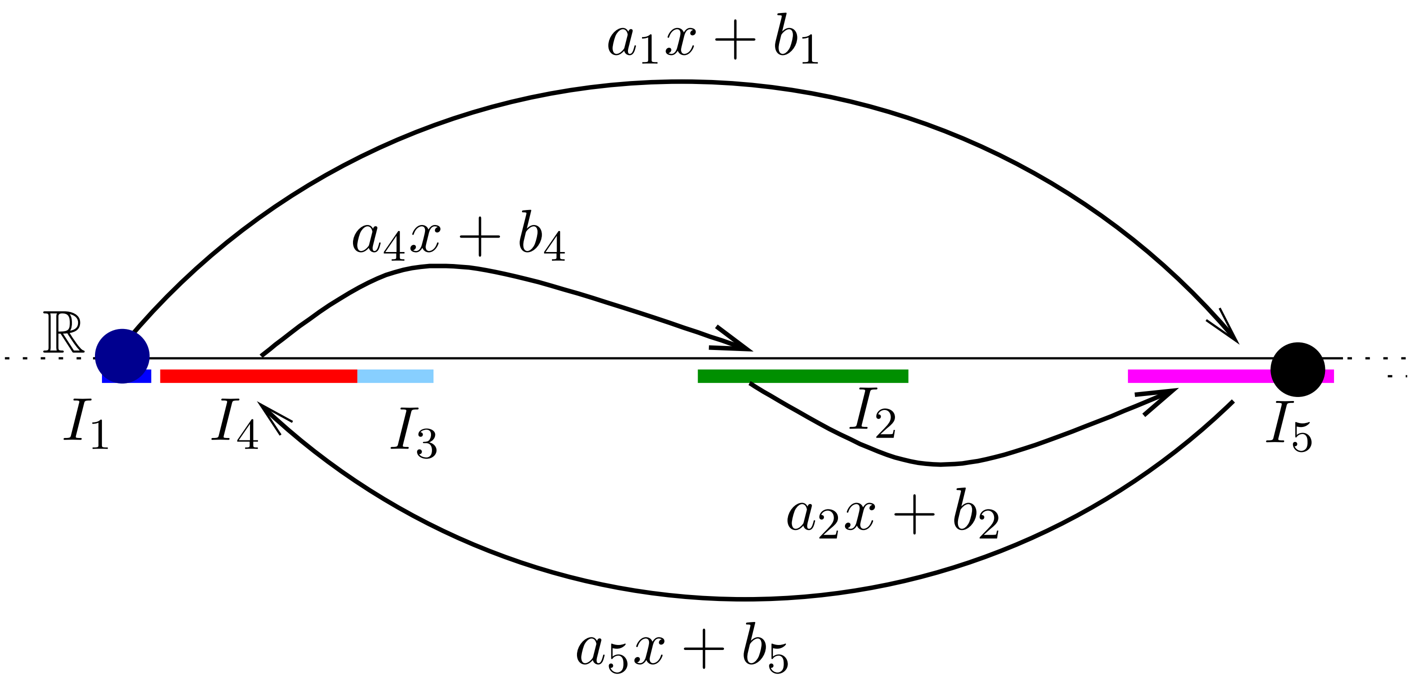
A difficult problem

- 1d piecewise affine maps (PAMs): $f : \mathbb{R} \to \mathbb{R}$
  $f(x) = a_i x + b_i$ for $x \in I_i$



A difficult problem

- 1d piecewise affine maps (PAMs): $f : \mathbb{R} \to \mathbb{R}$
  $f(x) = a_i x + b_i$ for $x \in I_i$



Old Open Problem. Is reachability decidable for 1d PAM?

Can realism help?

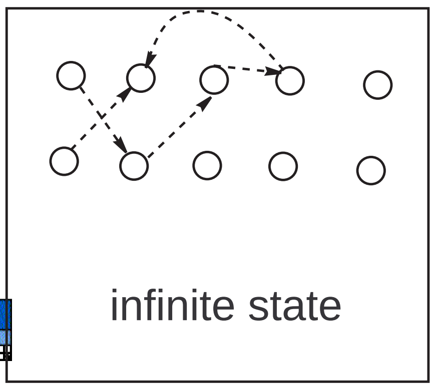
Maybe even undecidability is an artefact? Maybe it never occurs in real systems?

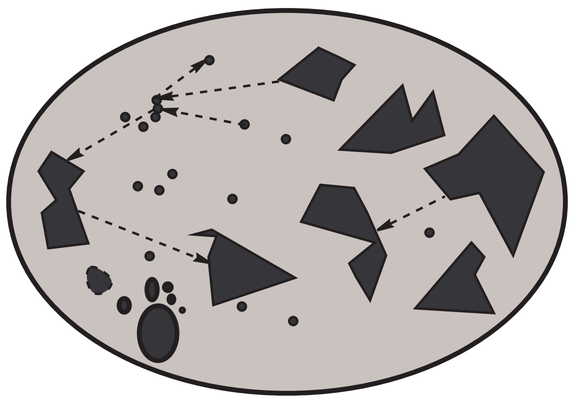
Proof method – Abstract View

- Proof by simulation of an infinite state machine by a DS
- State of machine $\leftrightarrow$ state of the DS
- Dynamics of DS simulates transitions of the machine

Consequences for bounded DS witnessing undecidability

- Important states (sets) of the DS are very dense (have accumulation points)
- Dynamics should be very precise (at least around accumulation points)
- It is difficult (impossible) to realize such systems physically
- ...and also: dynamics should be chaotic...





The Conjecture

Reachability is decidable for realistic, unprecise, noisy, “fuzzy”, “robust” systems

Arguments:

- The only known proof method uses unbounded precision (or unbounded state space)
- Noise could regularize...
- This world is nice and bad things never happen...
- Engineers design systems and never deal with undecidability.

Some Thoughts and Results

- All the arguments are weak
- The problem is interesting
- I know 3 natural formalizations of “realism”
  - Non-zero noise: undecidable (Σ_1-hard)
  - uniform noise: open problem
  - Infinitesimal noise: undecidable and co-r.e. (Π^0_1-complete)
- Both positive or negative solution would be interesting for the second one
- Most of these effects are not specific for a class of systems, they can be ported to any reasonable class.

Approximate methods for reachability

- In practice approximate methods should be used for safety verification.
- Several tools, many methods.
- General principles are easy, implementation difficult.

Abstract algorithm

For example consider forward breadth-first search.

F=Init
repeat
    F=F $\cup$ SuccFlow(F) $\cup$ SuccJump(F)
until fixpoint |(F$\cap$ Bad $\neq \emptyset$) | tired

A standard verification (semi-)algorithm.

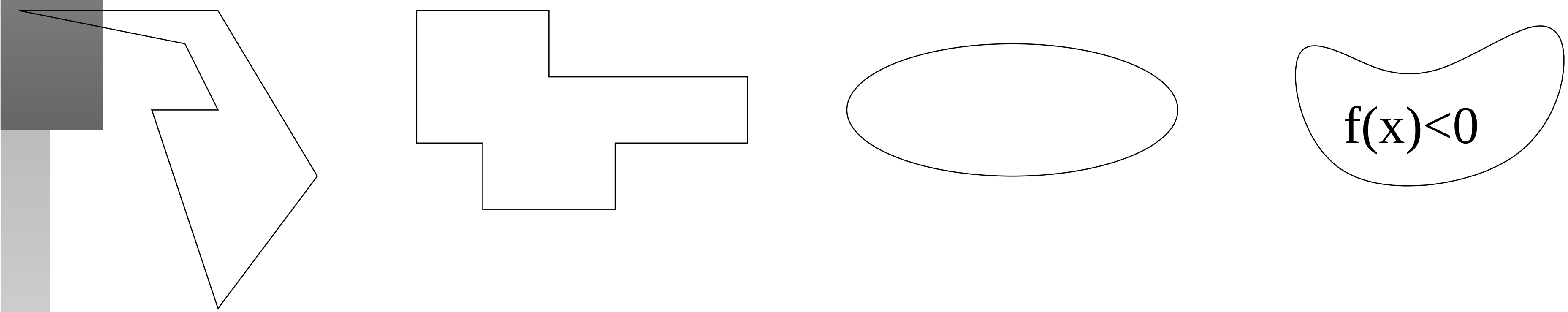
How to implement it

Needed data structure for (over-)approximate representation of subsets of $\mathbb{R}^n$, and algorithms for efficient computing of

- unions, intersections;
- inclusion tests;
- SuccFlow;
- SuccJump.

Known implementations

- Polyhedra (HyTech - exact. Checkmate)
- “Griddy polyhedra” (d/dt)
- Ellipsoids (Kurzhanski, Bochkarev)
- Level sets of functions (Tomlin)



Does it work?

Up to 10 dimensions. Sometimes.

Using advanced verification techniques

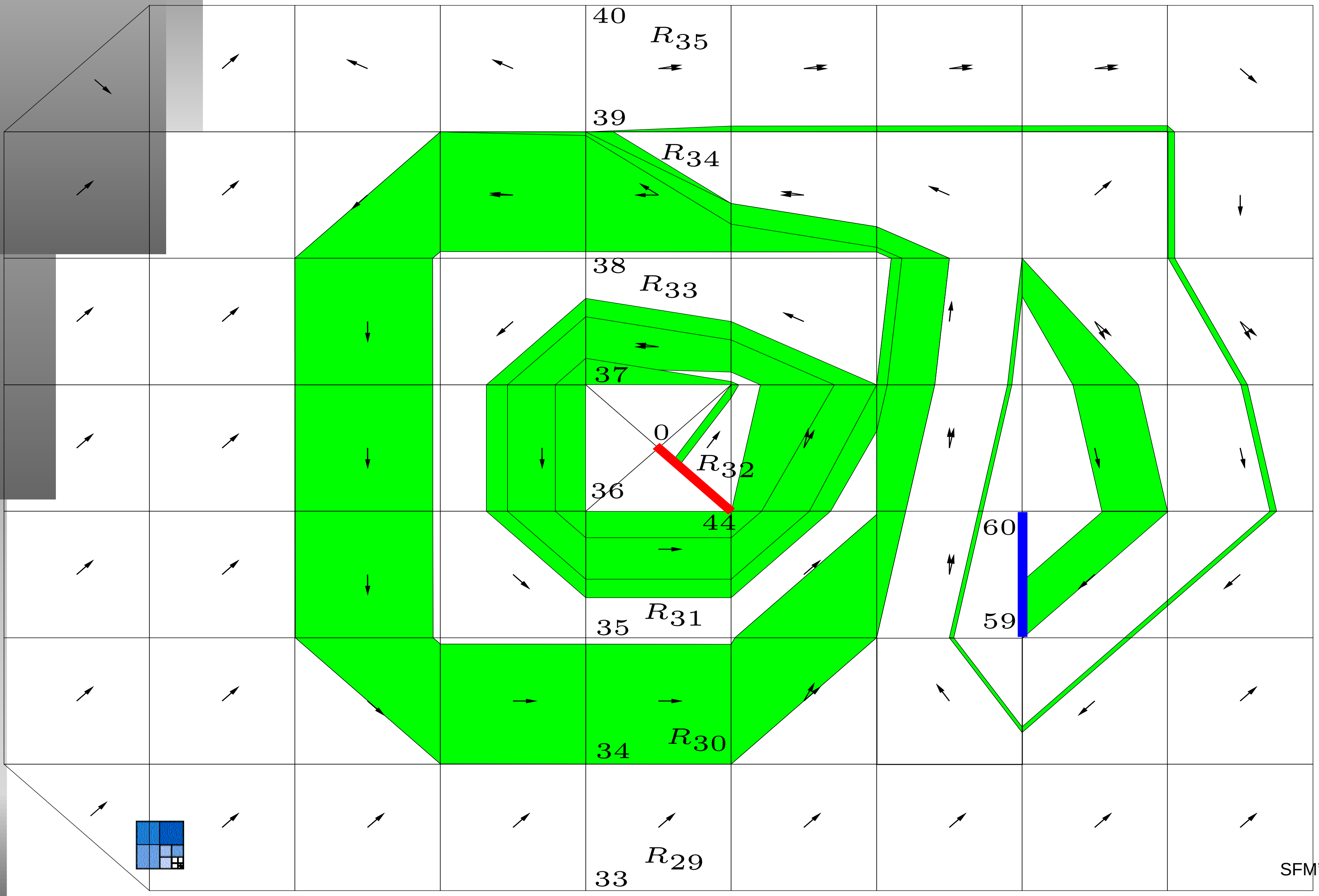
- Searching for better data-structures (SOS, *DD)
- Abstraction and refinement
- Combining model-checking and theorem proving
- Acceleration
- Bounded model-checking

Beyond verification

Generic verification algorithms + hybrid data structures allow:

- Model-checking
- Controller synthesis
- Phase portrait generation

A picture



3. Final Remarks

Outline

1. Hybrid automata - the model ✓
2. Verification ✓
3. Conclusions and perspectives
   - Conclusions for a pragmatical user
   - Conclusions for a researcher

Conclusions for a pragmatical user

- A useful and proper model : HA. Modeling languages available.
- Simulation possible with old and new tools
- No hope for exact analysis
- In simple cases approximated analysis (and synthesis) with guarantee is possible using verification paradigm. Tools available
- (Not discussed) Some control-theoretical techniques available (stability, optimal control etc).

Perspectives for a researcher

- Obtain new decidability results (nobody cares for undecidability).
- Explore noise-fuzziness-realism issues
- Apply modern model-checking techniques to approximate verification of HS
- Create hybrid theory of formal languages
- etc.