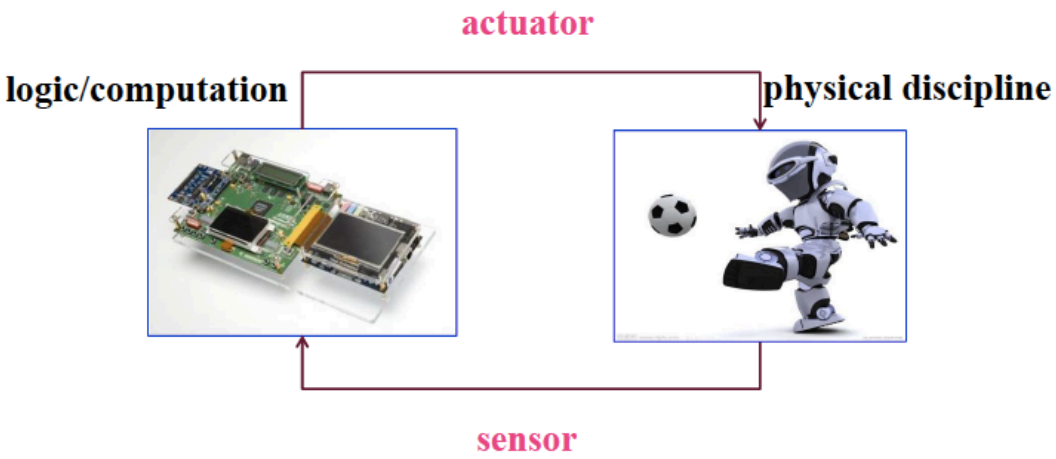
MPRI 2.8.2:
Foundations of real-time and hybrid systems

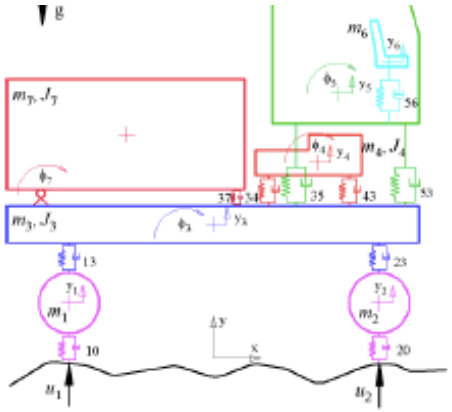
Eugene Asarin (IRIF)
Laurent Fribourg (LSV)

Thursday 12h45-15h15, salle 2035 (period 1)

Embedded Control Systems



Plant

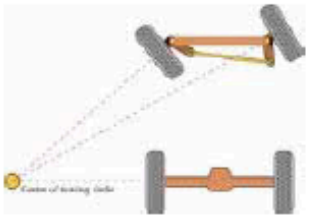


System Dynamics

Real-time Constraints

Controller

System Constraints



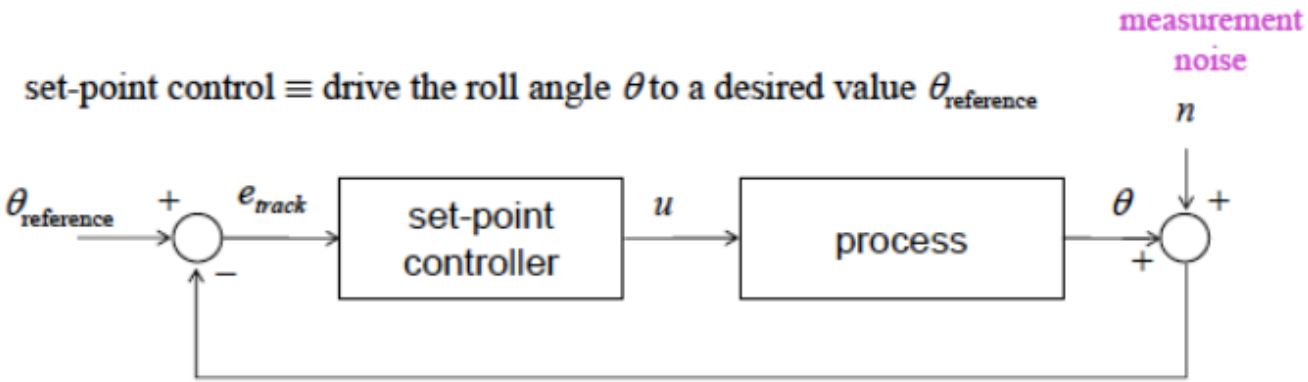
Classical view of control

Example #11: Roll-angle control

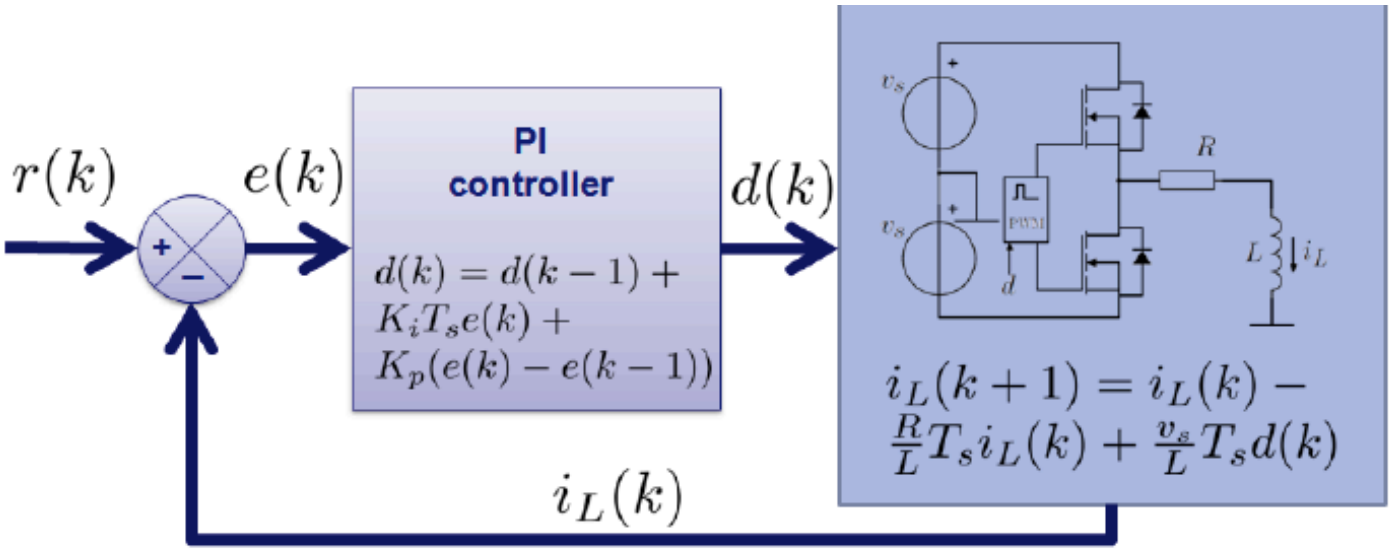
UCSB

$$\dddot{\theta} + 50.875\ddot{\theta} + 43.75\dot{\theta} = -1000u$$

set-point control $\equiv$ drive the roll angle θ to a desired value $\theta_{\text{reference}}$



Classical view of control



State-space realization of the closed-loop system

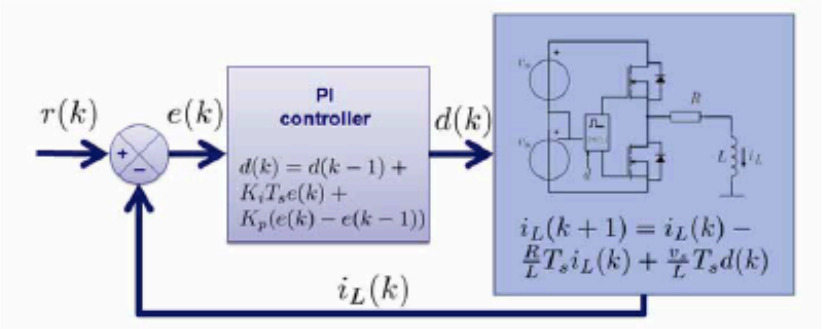
$$x(k+1) = Ax(k) + Dr(k)$$

$$x := \begin{bmatrix} i_L \\ \xi \end{bmatrix}, \quad \xi(k) := K_i T_s \sum_{i=0}^{k} e(i)$$

How to guarantee safety of the closed loop system ?

How to guarantee safety of the closed loop system by design?

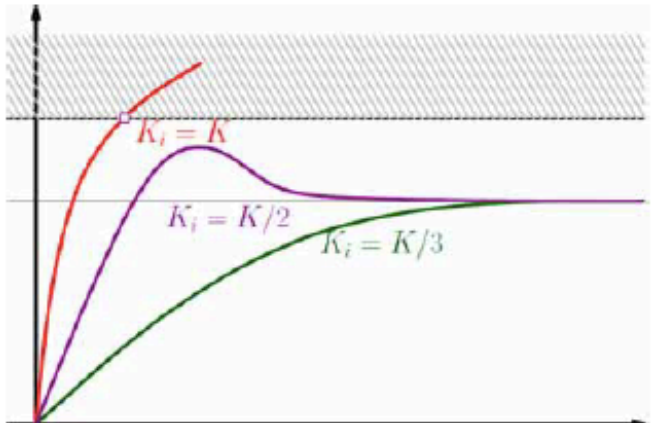
PI is valid only locally



-Saturation $d \in [-1, 1]$

-Hard constraints $i_L \leq i_{max}$

-Numeric representation



Industrial Solution

Extensive simulation

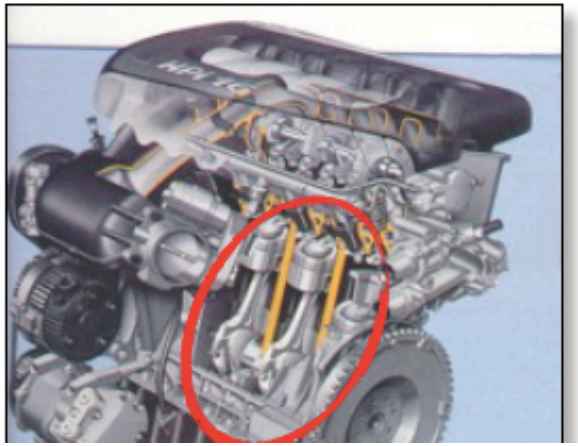
New context: interplay between discrete and continuous commands

- Transmission

discrete command
(R,N,1,2,3,4,5)

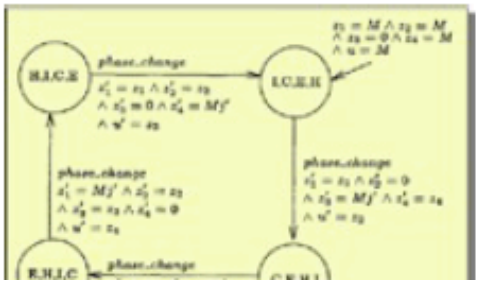
+

continuous
dynamical variables
(velocities, torques)



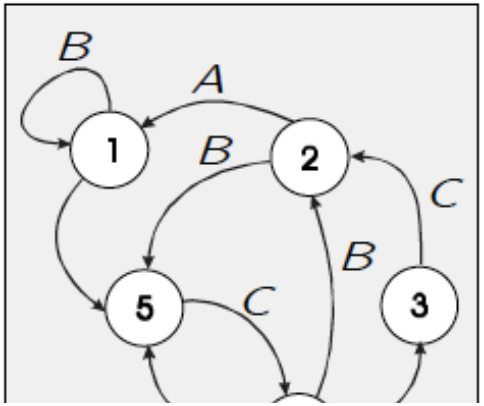
- Four-stroke engines

automaton,
dependent on
crankshaft angle



A suitable model: Hybrid systems

$\mathbf{x} \in \{1, 2, 3, 4, 5\}$
$\mathbf{u} \in \{A, B, C\}$



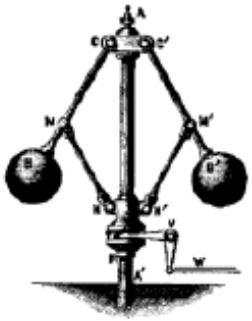
Computer Science

Control Theory

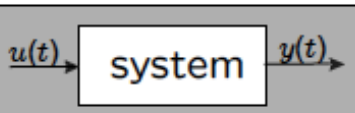
Finite state machines

Continuous dynamical systems

Hybrid systems



$x \in \mathbb{R}^n$
$u \in \mathbb{R}^m$
$y \in \mathbb{R}^p$

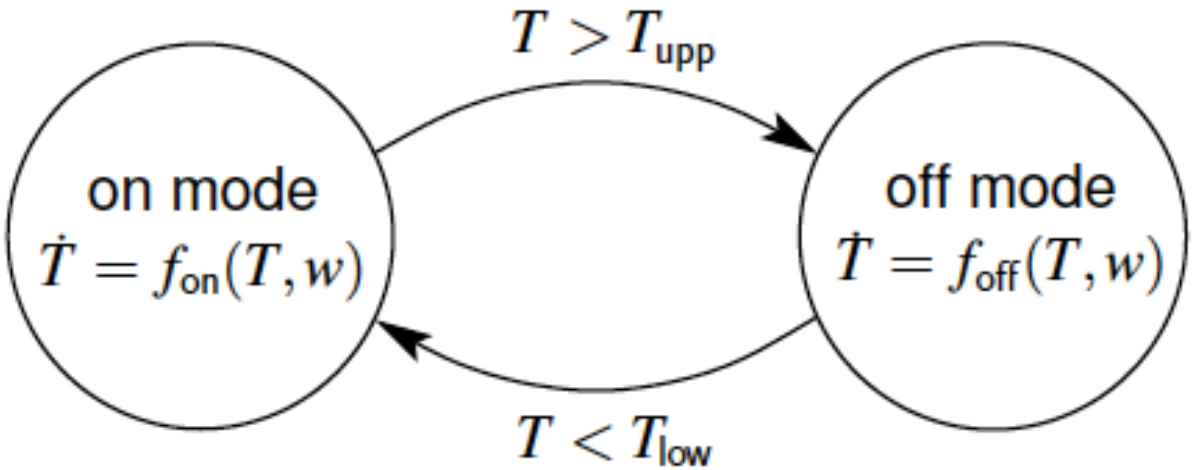


$$\begin{cases} \frac{dx(t)}{dt} = f(x(t), u(t)) \\ y(t) = g(x(t), u(t)) \end{cases}$$

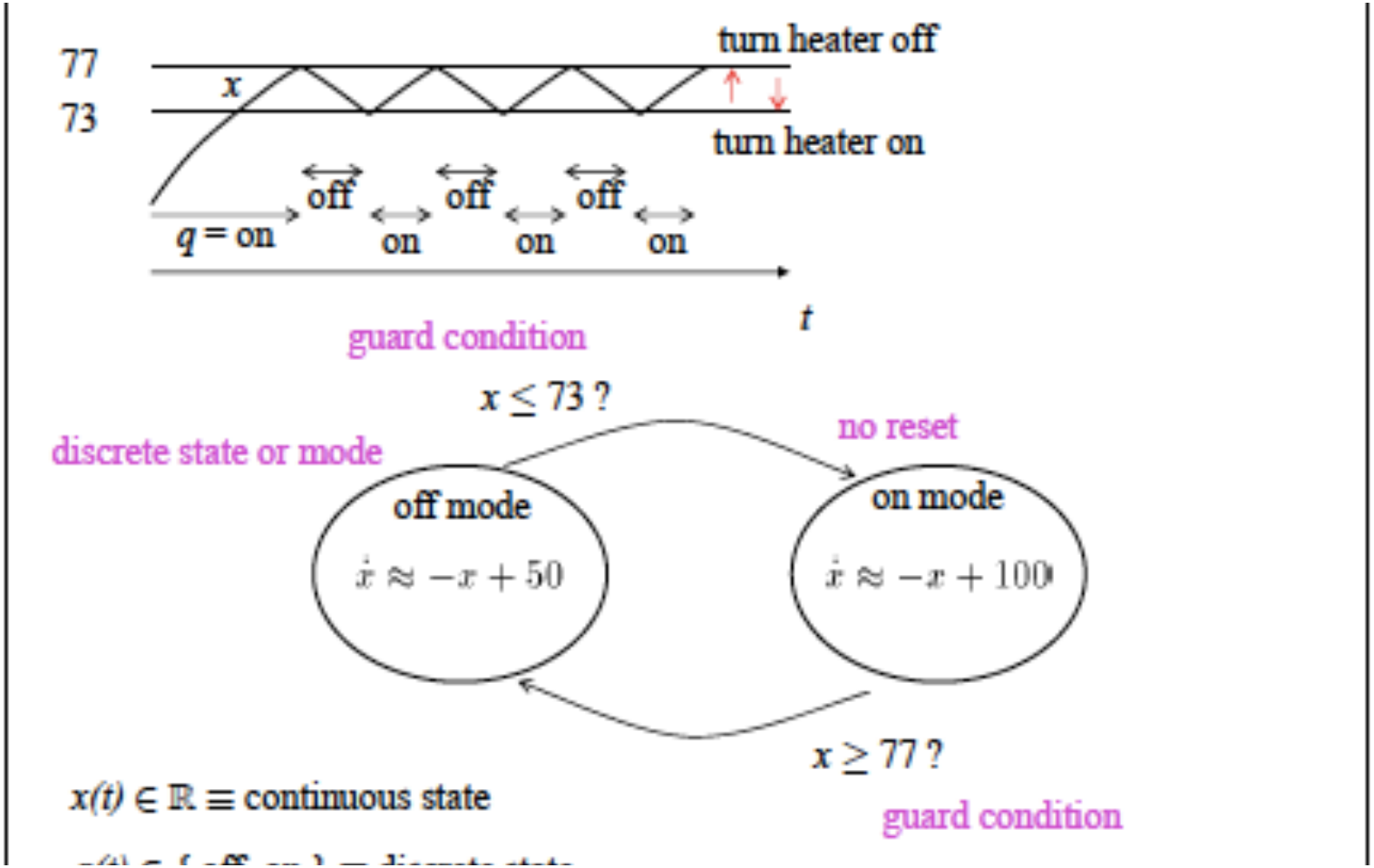
The thermostat example

. Hybrid systems: Motivating examples

- Hybrid: combination of continuous and discrete dynamics
- Temperature control system:

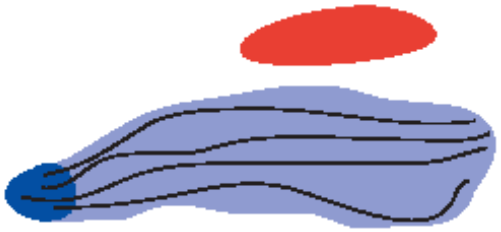


The thermostat example

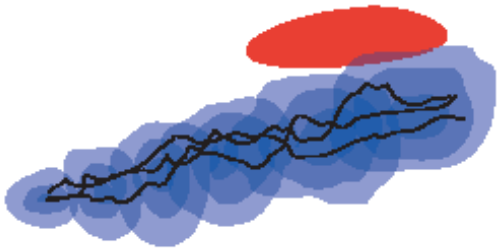


A basic verification technique: Reachability analysis

- **Main Talk: Reachability Analysis**
  - Linear Systems with Uncertain Parameters
  - Nonlinear Systems
  - Hybrid Systems



- **Stochastic Reachability Analysis of Linear Systems**
  - Basic Idea
  - Examples



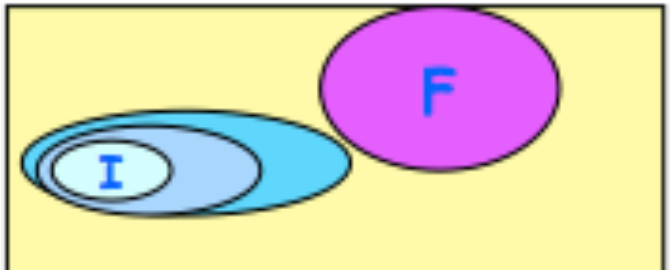
- **Safety Assessment of Autonomous Cars**
  - Basic Idea
  - Examples

Computation of Post (« Successor Set »)

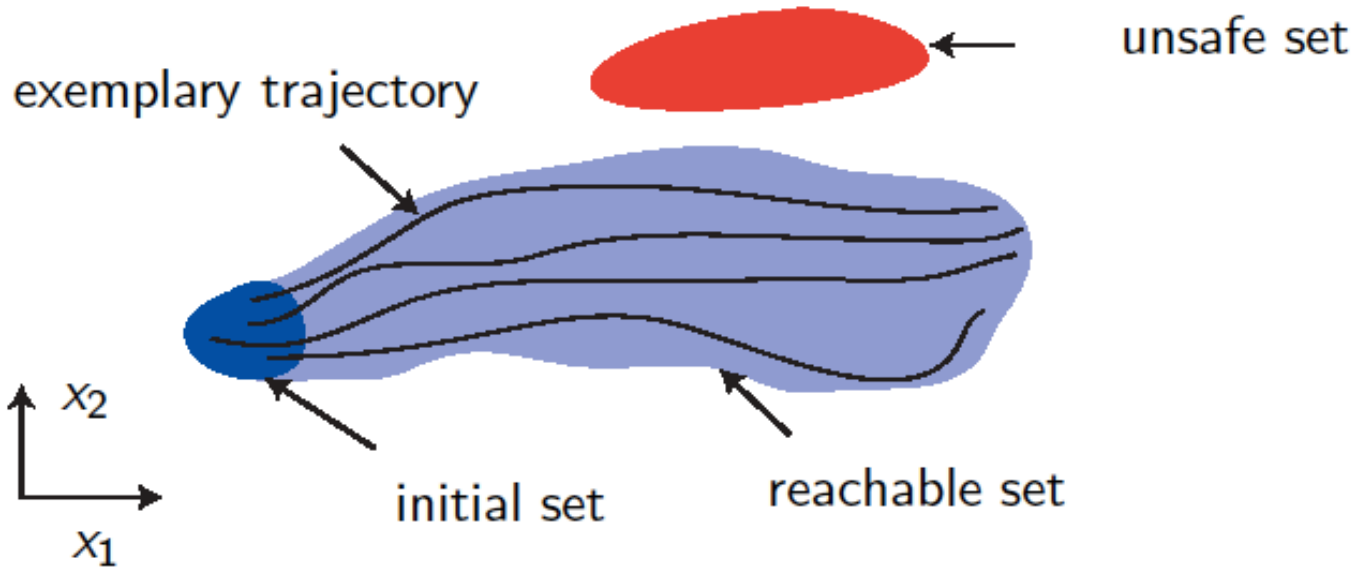
Goal: Given an initial set I, compute the set of reachable states to check if a bad state in F is reachable

Symbolic breadth-first search by applying Post:

Post(R): Set of successors of states in R



Trajectory tube



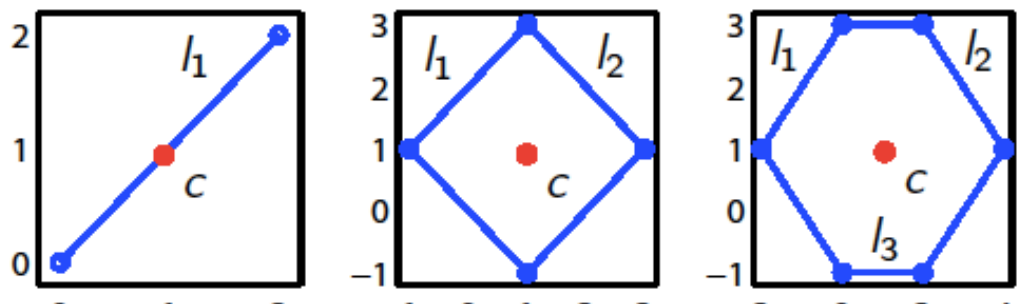
- System is safe, if no trajectory enters the unsafe set

Efficient Data structures: Zonotopes

Definition of a zonotope Z

$$Z = \left\{ x \in \mathbb{R}^n \middle| x = c + \sum_{i=1}^{p} \beta_i g^{(i)}, \quad -1 \leq \beta_i \leq 1 \right\}, \quad c, g^{(i)} \in \mathbb{R}^n$$

- Interpretation: Minkowski sum of line segments $l_i = [-1, 1]g^{(i)}$.
- Zonotopes are centrally symmetric to c.
- Short notation: $Z = (c, g^{(1 \ldots p)})$.



Symbolic integration

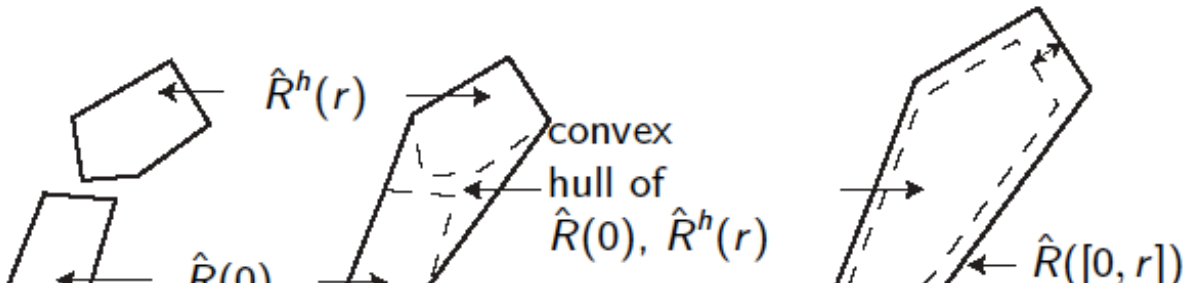
① Compute reachable set $\hat{R}^h(r)$ at time r (without input).

② Obtain convex hull of $\hat{R}(0)$ and $\hat{R}^h(r)$.

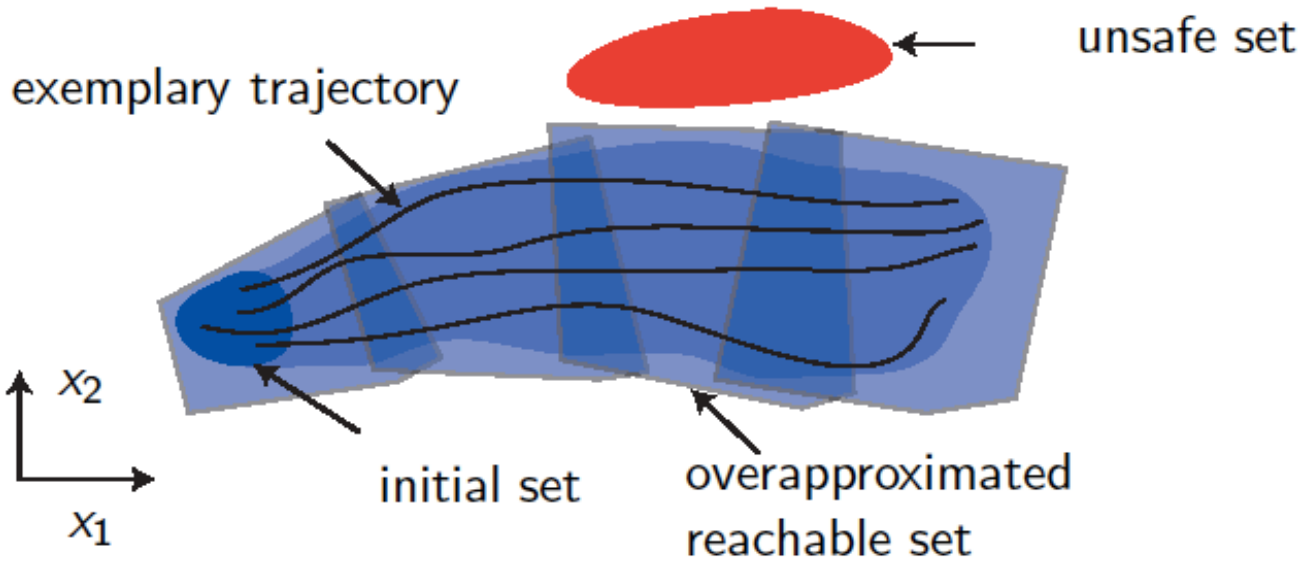
③ Enlarge reachable set to guarantee enclosure of all trajectories.

$$\hat{R}([0,r]) = \underbrace{\mathcal{CH}(\hat{R}(0), \overbrace{e^{\mathcal{A}r}\hat{R}(0)}^{①})}_{②} + \underbrace{F\hat{R}(0) + \hat{R}^i([0,r])}_{③}$$

F : Error interval due to the curvature of trajectories within $t \in [0, r]$.
$\hat{R}^i([0,r])$: Reachable set of the input (inhomogeneous solution).

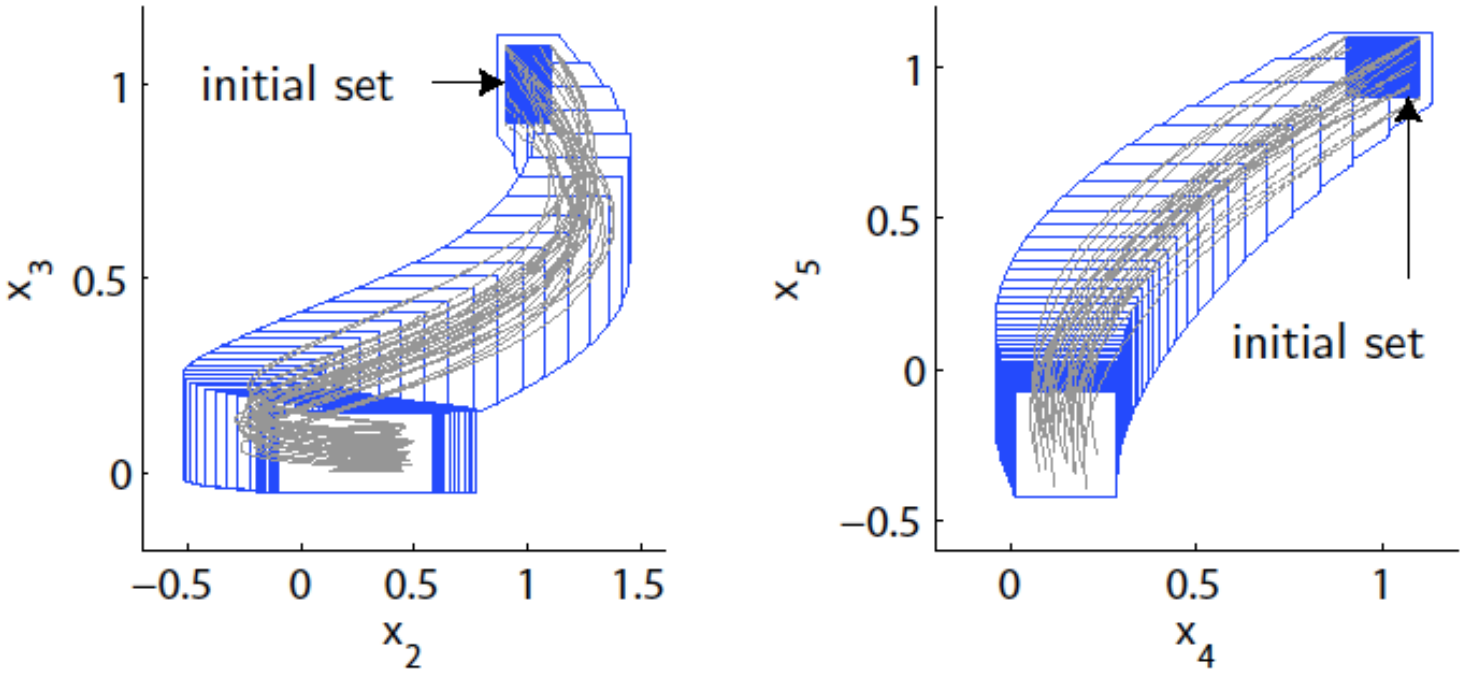


Overapproximation of tubes



- System is safe, if no trajectory enters the unsafe set.

Overapproximation of tubes

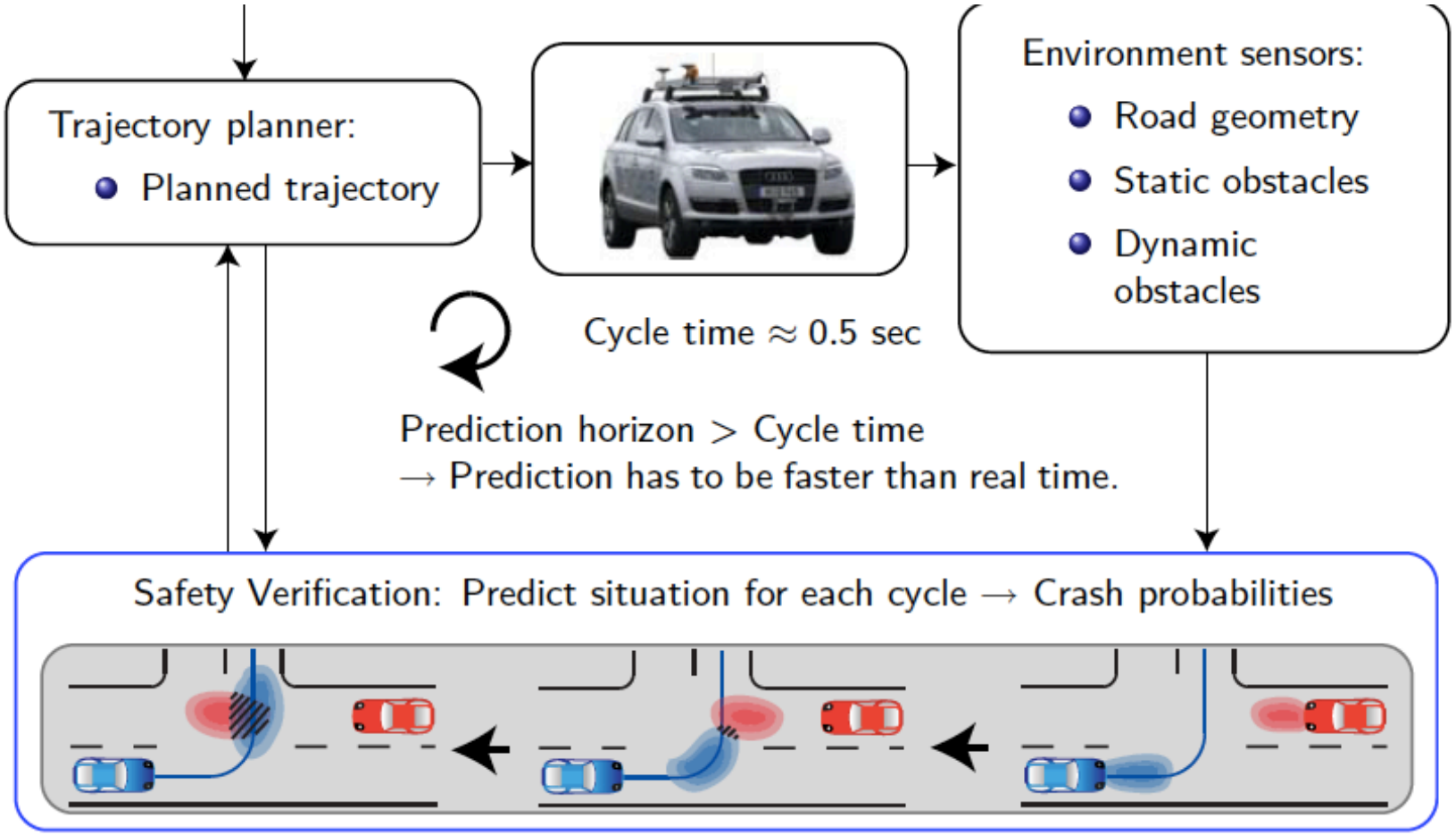


(a) Projection on x_2 and x_3 (b) Projection on x_4 and x_5

Computation with systems of higher dimensions for 125 time intervals:

Dimension n	5	10	20	50	100

Application to traffic control

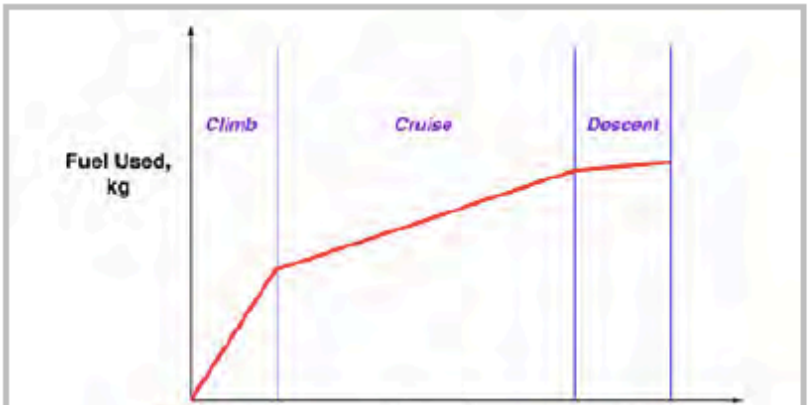


Optimisation in the symbolic context

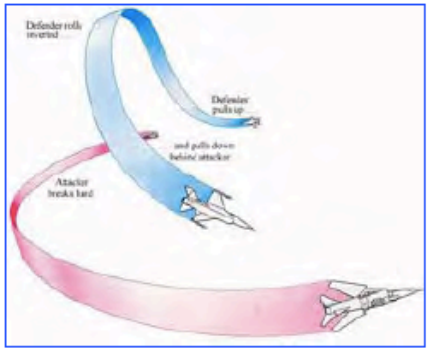
Minimize an Absolute Criterion

- Achieve a specific objective
  - Minimum time
  - Minimum fuel
  - Minimum financial cost
- to achieve a goal

- What is the control variable?



Game Theory

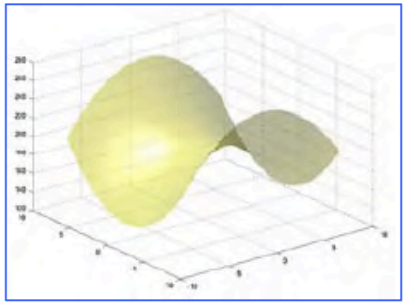


Example: Pursuit-Evasion: Competitive Optimization Problem

- Pursuer's goal: minimize final miss distance
- Evader's goal: maximize final miss distance

- "Minimax" (saddle-point) cost function
- Optimal control laws for pursuer and evader

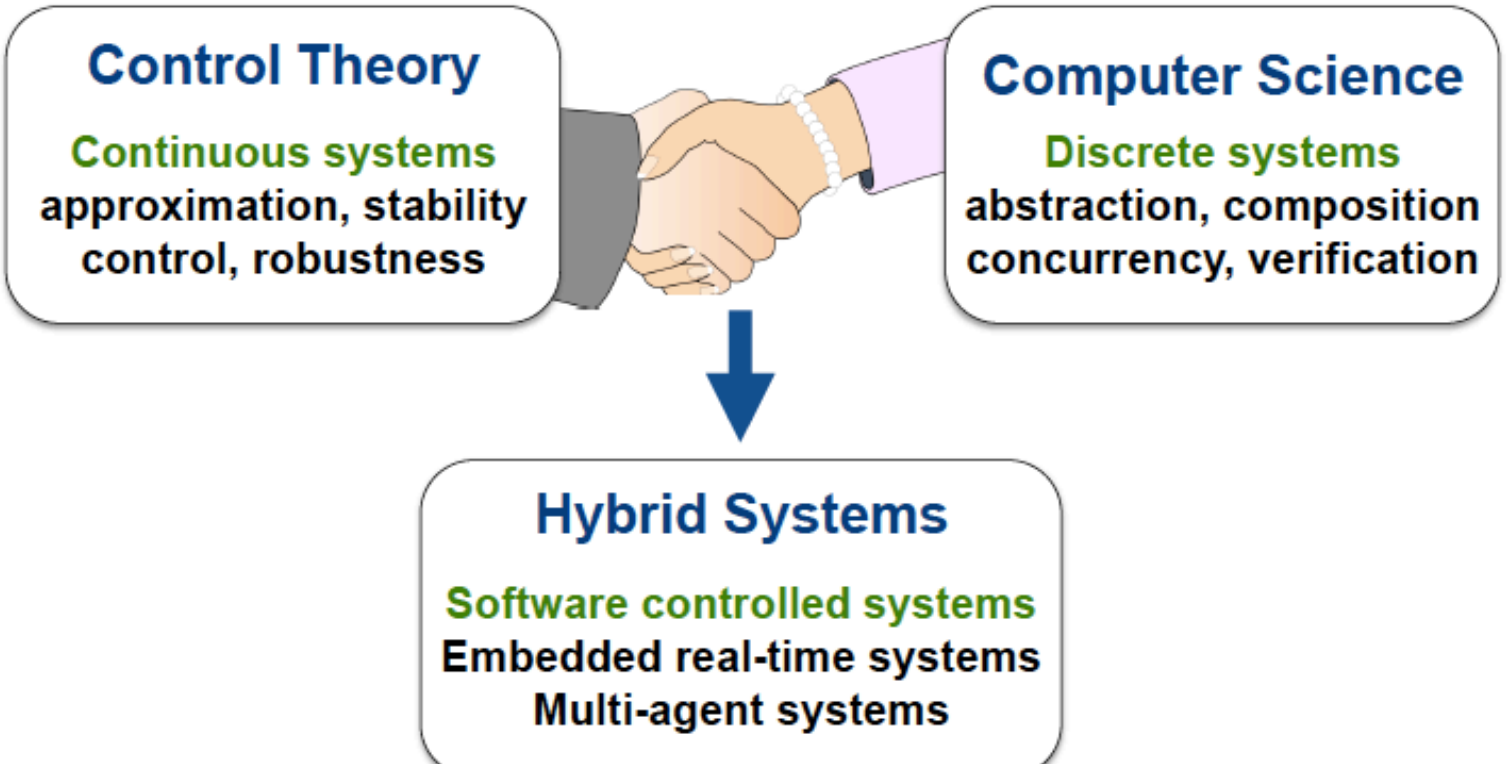
$$\mathbf{u}(t) = \begin{bmatrix} \mathbf{u}_P(t) \\ \mathbf{u}_E(t) \end{bmatrix} = -\begin{bmatrix} \mathbf{C}_P(t) & \mathbf{C}_{PE}(t) \\ \mathbf{C}_{EP}(t) & \mathbf{C}_E(t) \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_P(t) \\ \hat{\mathbf{x}}_E(t) \end{bmatrix}$$



Example of a *differential game*, Isaacs (1965), Bryson & Ho (1969)

MPRI 2.8.2: Foundations of real-time and hybrid systems (Thursday 12h45)

Continuous AND Discrete Systems



MPRI 2.9.1: Mathematical foundations of the theory of infinite transitions

(Monday 16h15)

WELL QUASI-ORDERS FOR ALGORITHMS

Jean Goubault-Larrecq goubault@lsv.fr

Philippe Schnoebelen phs@lsv.fr

Background

Classical Approach

Finite-State Systems

Model Checking

Model $\models$ (safety) property

Challenge:

Infinite-State Systems

Sources of "Infiniteness":

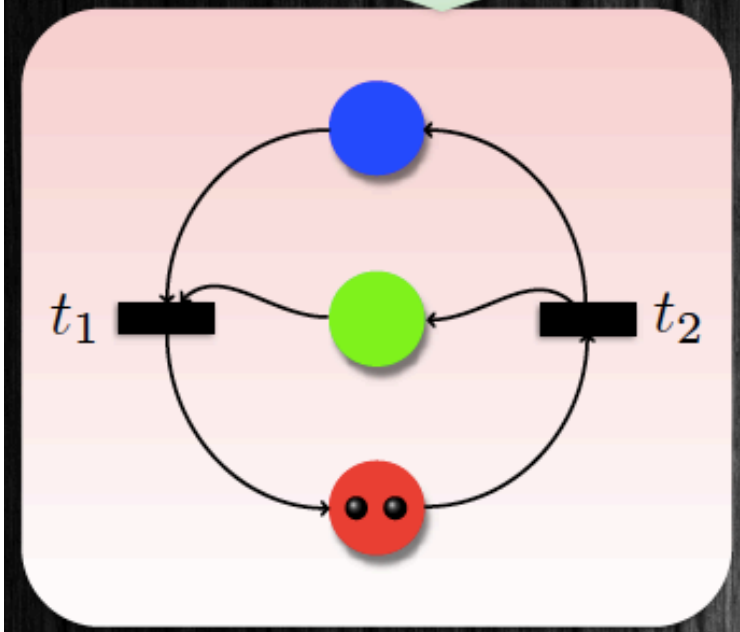
Unbounded Data Structures

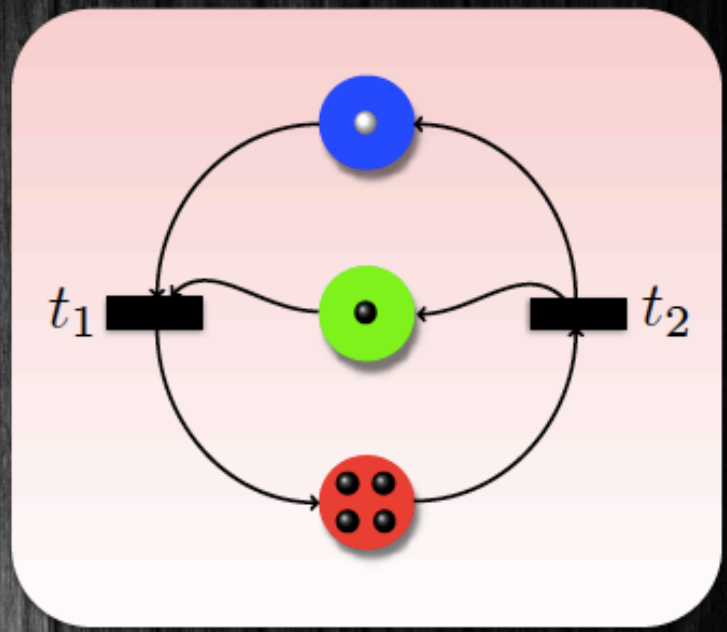
- stacks (recursion)
- queues (protocols)
- counters (programs)
- clocks (time)
- lists, trees, graphs (heaps)

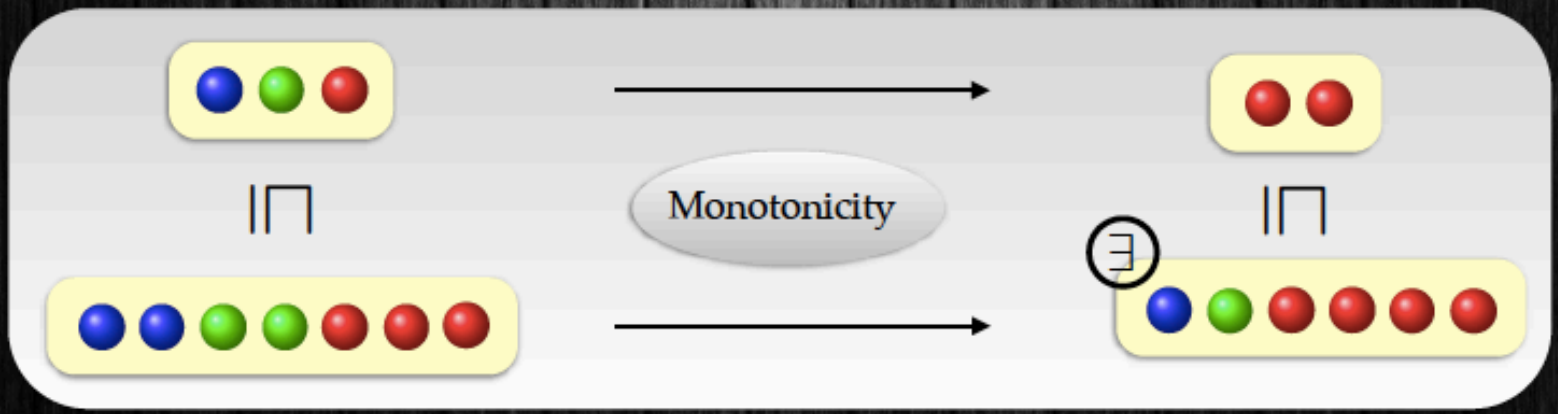
Unbounded Control Structures

- parameterized systems
- multithreaded programs
- concurrent libraries
- Petri nets

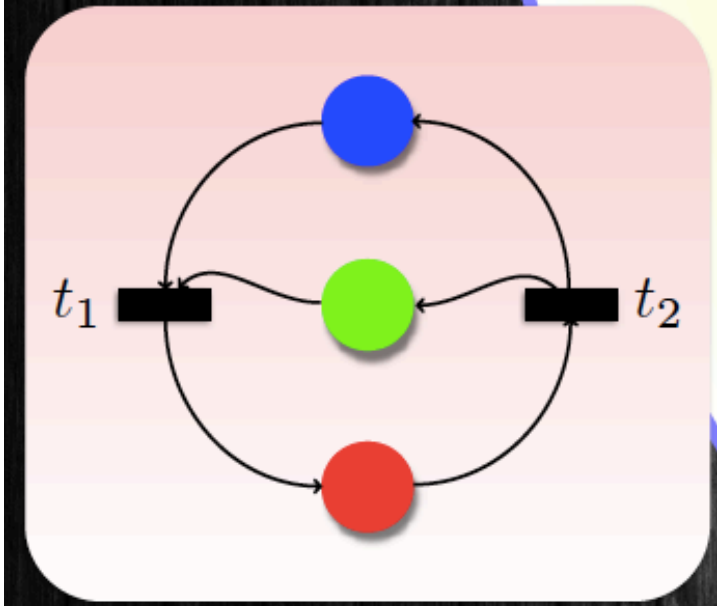
Monotonicity





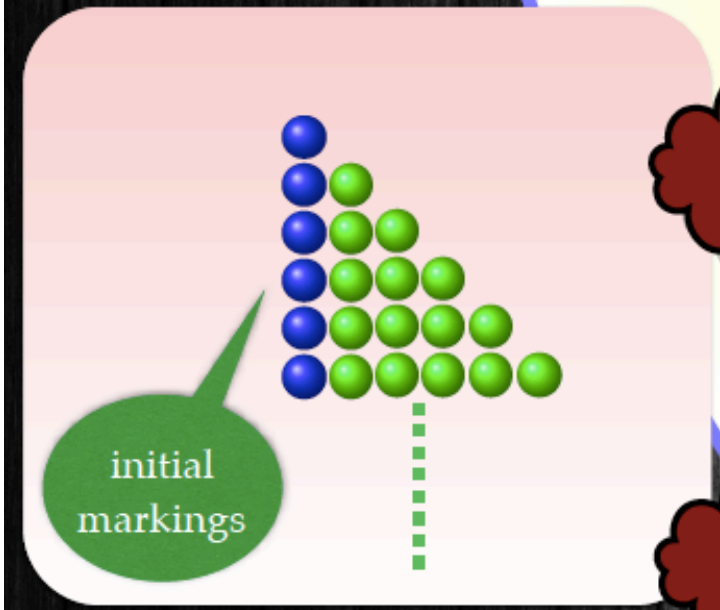


Backward Reachability



Backward Reachability

System Safe !



symbolic representation = finite multisets

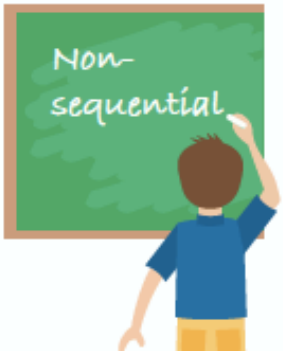
Termination: multisets well quasi-ordered

MPRI 2.8.1: Non-sequential theory of distributed systems

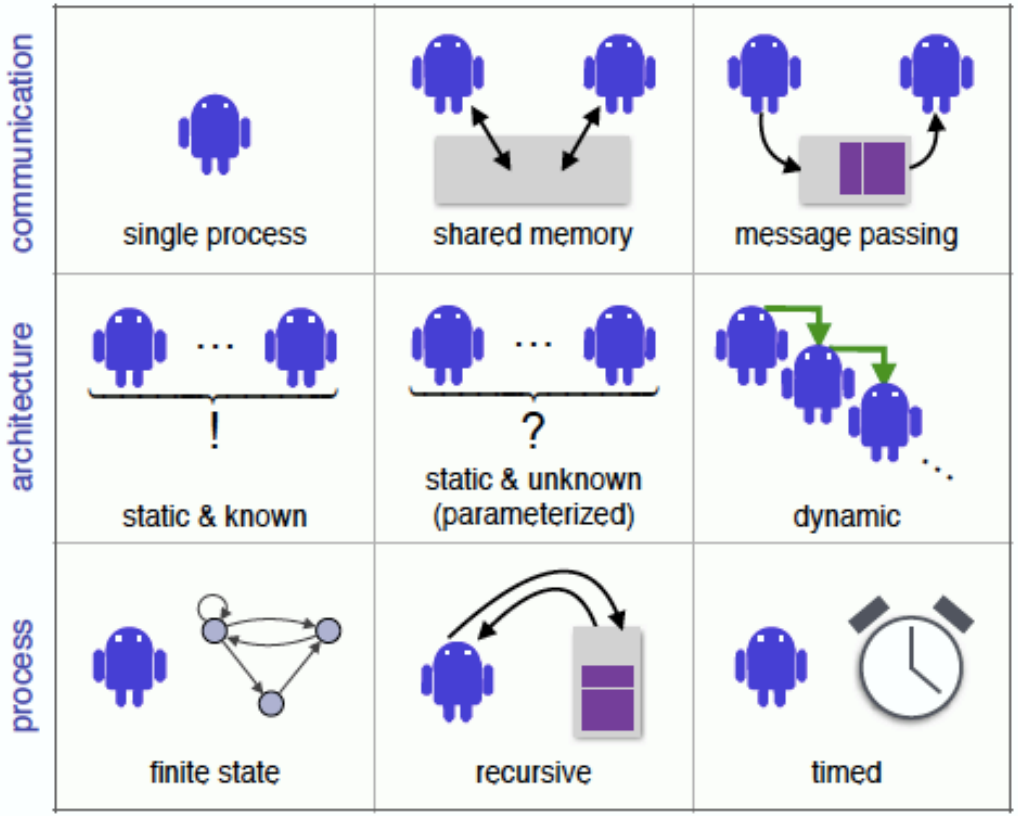
Wednesdays
1st Lecture on September 12, 2018

13:15 — 15:45 (2,5 hours per lecture)

Bat. Sophie Germain, room 2036



Graph automata



Behavior

- Graphs

System model

- Graph automata

Specification

- Linear-time temporal logic (LTL)
- Monadic second-order logic (MSO)

... in a uniform framework.

2.9.2. Algorithmic Verification of Programs

- **Teachers:** Ahmed Bouajjani (www.irif.fr/~abou), Constantin Enea (www.irif.fr/~cenea)
- **Course objectives:** algorithmic approach for the verification of concurrent and distributed software
  - correctness criteria for concurrent and distributed data structures
    - concurrent queues, hash maps, etc for multi-threading programming: java.util.concurrent, Intel Thread Building Blocks
    - distributed key-value stores for distributed applications (web services/cloud): Amazon SS3, MongoDB
    - correctness criteria: linearizability (atomicity), eventual/causal consistency, etc
  - analysis techniques based on automata, logic, model checking, deductive verification

MERCI !