

# Guaranteed control synthesis for continuous systems in Uppaal-Tiga

CyPhy 2018

Kim G. Larsen, Adrien Le Coënt,  
Jakob H. Taankvist, Marius Mikučionis



## Context: control systems



## Context: control systems



Issues: guaranteed control synthesis for safety/stability/reachability of safety critical systems



# Outline

## 1 Guaranteed control of switched systems

- Switched systems
- Control of switched systems

## 2 Safety controller synthesis using timed games

## 3 Further optimization using Uppaal Stratego



# Switched Systems

A continuous-time **switched system**

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \quad (1)$$

is a family of continuous-time dynamical systems with a rule  $\sigma$  that determines at each time which one is active



## Switched systems

# Switched Systems

A continuous-time **switched system**

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \quad (1)$$

is a family of continuous-time dynamical systems with a rule  $\sigma$  that determines at each time which one is active

- state  $x \in \mathbb{R}^n$



## Switched systems

# Switched Systems

A continuous-time **switched system**

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \quad (1)$$

is a family of continuous-time dynamical systems with a rule  $\sigma$  that determines at each time which one is active

- state  $x \in \mathbb{R}^n$
- bounded perturbation  $d(\cdot) : \mathbb{R}^+ \longrightarrow D \subset \mathbb{R}^m$



## Switched systems

# Switched Systems

A continuous-time switched system

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \quad (1)$$

is a family of continuous-time dynamical systems with a rule  $\sigma$  that determines at each time which one is active

- state  $x \in \mathbb{R}^n$
- bounded perturbation  $d(\cdot) : \mathbb{R}^+ \longrightarrow D \subset \mathbb{R}^m$
- switching signal  $\sigma(\cdot) : \mathbb{R}^+ \longrightarrow U$  (piecewise constant)



## Switched Systems

A continuous-time **switched system**

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \quad (1)$$

is a family of continuous-time dynamical systems with a rule  $\sigma$  that determines at each time which one is active

- state  $x \in \mathbb{R}^n$
- bounded perturbation  $d(\cdot) : \mathbb{R}^+ \longrightarrow D \subset \mathbb{R}^m$
- switching signal  $\sigma(\cdot) : \mathbb{R}^+ \longrightarrow U$  (piecewise constant)
- $U = \{1, \dots, N\}$  finite set of **modes**, associated with the dynamics

$$\dot{x}(t) = f_u(x(t), d(t)), \quad u \in U$$

## Switched Systems

A continuous-time **switched system**

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \quad (1)$$

is a family of continuous-time dynamical systems with a rule  $\sigma$  that determines at each time which one is active

- state  $x \in \mathbb{R}^n$
- bounded perturbation  $d(\cdot) : \mathbb{R}^+ \rightarrow D \subset \mathbb{R}^m$
- switching signal  $\sigma(\cdot) : \mathbb{R}^+ \rightarrow U$  (piecewise constant)
- $U = \{1, \dots, N\}$  finite set of **modes**, associated with the dynamics

$$\dot{x}(t) = f_u(x(t), d(t)), \quad u \in U$$

We focus on sampled switched systems: switching instants occur periodically every  $\tau$ , i.e.  $\sigma$  is constant on  $[i\tau, (i+1)\tau)$



## Switched systems

## Examples of switched systems





## Control of switched systems: possible approaches

- Optimal control [Trélat, Rubio, Fattorini,...]
  - Minimization under constraints of a cost function



## Control of switched systems: possible approaches

- Optimal control [Trélat, Rubio, Fattorini,...]
  - Minimization under constraints of a cost function
- Lyapunov approaches [Krstic, Teel, Coron, Sigalotti, Liberzon...]
  - Stability analysis and stabilization using Lyapunov functions



## Control of switched systems: possible approaches

- Optimal control [Trélat, Rubio, Fattorini,...]
  - Minimization under constraints of a cost function
- Lyapunov approaches [Krstic, Teel, Coron, Sigalotti, Liberzon...]
  - Stability analysis and stabilization using Lyapunov functions
- Symbolic methods



## Control of switched systems: possible approaches

- Optimal control [Trélat, Rubio, Fattorini,...]
  - Minimization under constraints of a cost function
- Lyapunov approaches [Krstic, Teel, Coron, Sigalotti, Liberzon...]
  - Stability analysis and stabilization using Lyapunov functions
- Symbolic methods
  - Finite state abstraction methods:
    - Approximate bisimulation (PESSOA/CoSyMA) [Tabuada, Girard...]
    - Feedback refinement relations (SCOTS) [Zamani, Rungger, Reissig...]



## Control of switched systems: possible approaches

- Optimal control [Trélat, Rubio, Fattorini,...]
  - Minimization under constraints of a cost function
- Lyapunov approaches [Krstic, Teel, Coron, Sigalotti, Liberzon...]
  - Stability analysis and stabilization using Lyapunov functions
- Symbolic methods
  - Finite state abstraction methods:
    - Approximate bisimulation (PESSOA/CoSyMA) [Tabuada, Girard...]
    - Feedback refinement relations (SCOTS) [Zamani, Rungger, Reissig...]
  - Tiling methods:
    - Uniform tiling [Girard, Witrant, Arcak...]
    - Adaptive state-space bisection (MINIMATOR) [Soulat, Fribourg]



## Control Synthesis Problem

We consider the [state-dependent control](#) problem of synthesizing  $\sigma$ :



## Control Synthesis Problem

We consider the [state-dependent control](#) problem of synthesizing  $\sigma$ :

At each sampling time  $k\tau$ , find the appropriate switched mode  $u \in U$  according to the current value of  $x$ , in order to achieve some objectives:



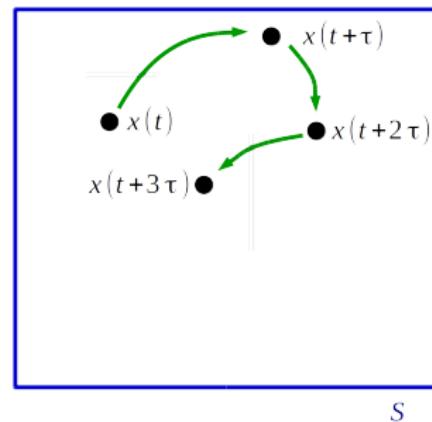
## Control Synthesis Problem

We consider the **state-dependent control** problem of synthesizing  $\sigma$ :

At each sampling time  $k\tau$ , find the appropriate switched mode  $u \in U$  according to the current value of  $x$ , in order to achieve some objectives:

Given a set  $S$ :

- Safety in  $S$ :  $x(t)$  stays in  $S$  forever





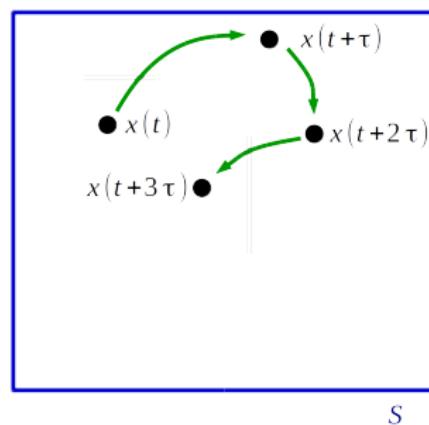
## Control Synthesis Problem

We consider the **state-dependent control** problem of synthesizing  $\sigma$ :

At each sampling time  $k\tau$ , find the appropriate switched mode  $u \in U$  according to the current value of  $x$ , in order to achieve some objectives:

Given a set  $S$ :

- Safety in  $S$ :  $x(t)$  stays in  $S$  forever



NB: classic stabilization impossible here (no common equilibrium pt)  
 $\rightsquigarrow$  practical stability



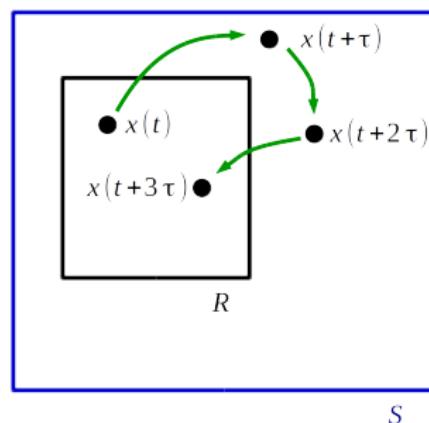
## Control Synthesis Problem

We consider the **state-dependent control** problem of synthesizing  $\sigma$ :

At each sampling time  $k\tau$ , find the appropriate switched mode  $u \in U$  according to the current value of  $x$ , in order to achieve some objectives:

Given two sets  $R, S$ :

- $(R, S)$ -stability:  $x(t)$  returns in  $R$  infinitely often, at some multiples of sampling period  $\tau$ , and always stays in  $S$



NB: classic stabilization impossible here (no common equilibrium pt)  
 ↵ practical stability



# Outline

- 1 Guaranteed control of switched systems**
- 2 Safety controller synthesis using timed games**
- 3 Further optimization using Uppaal Stratego**



# Outline

## 1 Guaranteed control of switched systems

## 2 Safety controller synthesis using timed games

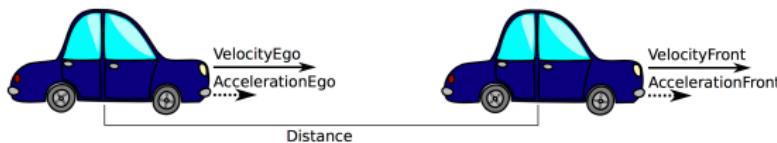
- From switched systems to timed games
- Integer approximation using a guaranteed Euler scheme

## 3 Further optimization using Uppaal Stratego



From switched systems to timed games

## A cruise control application



## A cruise control application



$$\dot{v}_f = a_f \quad (2)$$

$$\dot{v}_e = a_e \quad (3)$$

$$\dot{d} = v_f - v_e \quad (4)$$

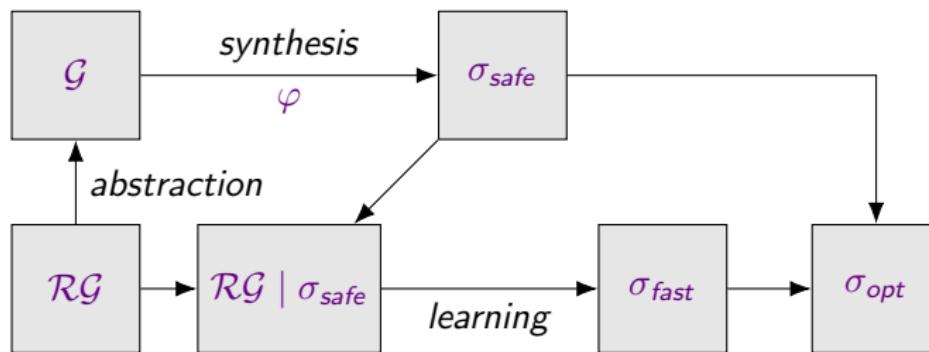
$a_f$  and  $a_e$  can take the values  $-2$ ,  $0$ , and  $2$ , resulting in an 8-mode switched system

Safety set:  $S = [-10, 20] \times [-10, 20] \times [5, 200]$ .



From switched systems to timed games

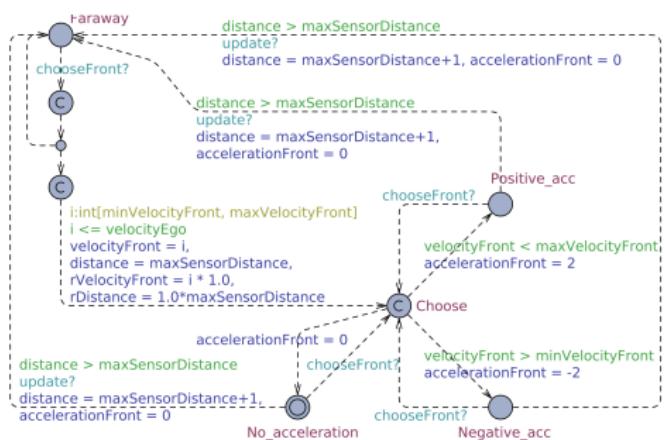
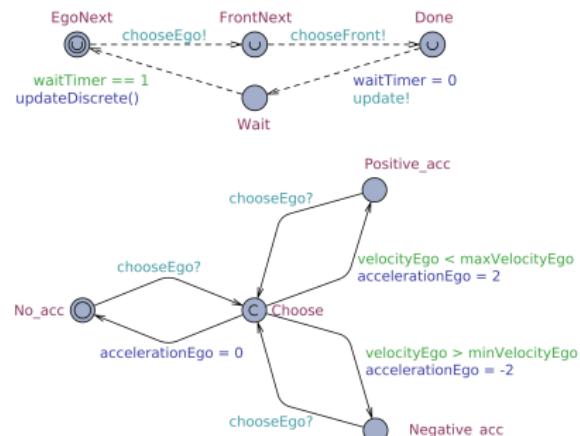
## Uppaal Stratego original workflow





From switched systems to timed games

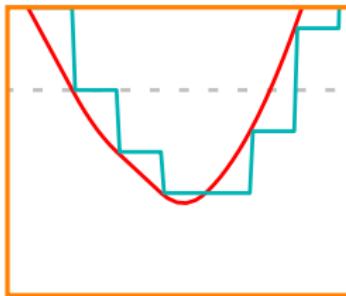
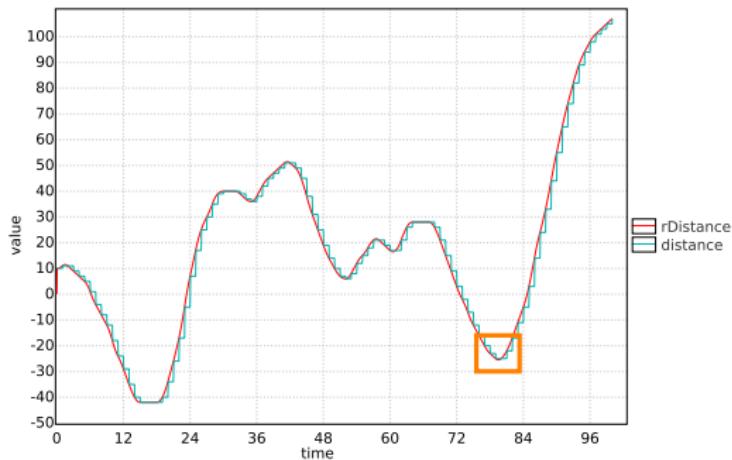
# Timed game model of the cruise control





From switched systems to timed games

# Standard discretization of the dynamics

Kim G. Larsen, Marius Mikučionis, Jakob H. Taankvist. *Safe and optimal adaptive cruise control*. 2015.



Integer approximation using a guaranteed Euler scheme

## Guaranteed Euler scheme

(H0) (Lipschitz): for all  $j \in U$ , there exists a constant  $L_j > 0$  such that:

$$\|f_j(y) - f_j(x)\| \leq L_j \|y - x\| \quad \forall x, y \in S.$$

(H1) (One-sided Lipschitz/Strong monotony): for all  $j \in U$ , there exists a constant  $\lambda_j \in \mathbb{R}$  such that

$$\langle f_j(y) - f_j(x), y - x \rangle \leq \lambda_j \|y - x\|^2 \quad \forall x, y \in T,$$

NB: constants computed by constrained optimization.

# Guaranteed Euler scheme

## Theorem

Given a sampled switched system satisfying (H0-H1), consider a point  $\tilde{x}^0$  and a positive real  $\delta$ . We have, for all  $x^0 \in B(\tilde{x}^0, \delta)$ ,  $t \in [0, \tau]$  and  $j \in U$ :

$$\phi_j(t; x^0) \in B(\tilde{\phi}_j(t; \tilde{x}^0), \delta_j(t)).$$

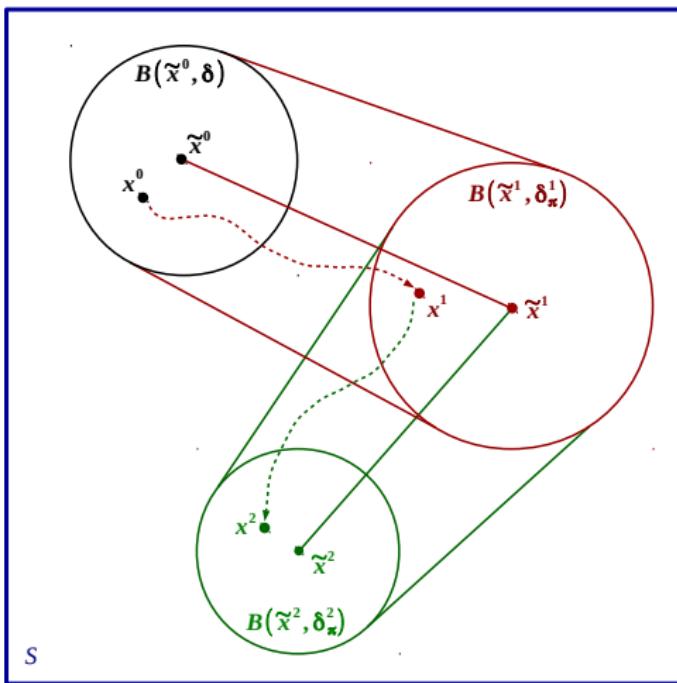
with

- if  $\lambda_j < 0$  :  $\delta_j(t) = \left( \delta^2 e^{\lambda_j t} + \frac{C_j^2}{\lambda_j^2} \left( t^2 + \frac{2t}{\lambda_j} + \frac{2}{\lambda_j^2} (1 - e^{\lambda_j t}) \right) \right)^{\frac{1}{2}}$
- if  $\lambda_j = 0$  :  $\delta_j(t) = (\delta^2 e^t + C_j^2 (-t^2 - 2t + 2(e^t - 1)))^{\frac{1}{2}}$
- if  $\lambda_j > 0$  :  $\delta_j(t) = \left( \delta^2 e^{3\lambda_j t} + \frac{C_j^2}{3\lambda_j^2} \left( -t^2 - \frac{2t}{3\lambda_j} + \frac{2}{9\lambda_j^2} (e^{3\lambda_j t} - 1) \right) \right)^{\frac{1}{2}}$



Integer approximation using a guaranteed Euler scheme

## Application to guaranteed integration





## Integer approximation using a guaranteed Euler scheme

# The guaranteed Euler scheme in UPPAAL TIGA

### Theorem

Consider system (1) satisfying (H0) and (H1), and an initial condition  $x_0 \in S$  at time 0.

- If  $x_0$  is an integer, let  $x'_0 = x_0$
- Otherwise, let  $x'_0 = \frac{\lfloor x_0 \rfloor + \lceil x_0 \rceil}{2}$

Let  $h = \tau / k$  be the step size of the Euler method. Given a controller  $C$ , let us compute iteratively two sequences of integer states as follows:

$$y_0^{\min} = \lfloor x'_0 \rfloor \quad \text{at } t = 0$$

$$y_0^{\max} = \lceil x'_0 \rceil \quad \text{at } t = 0$$

and

$$y_{i+1}^{\min} = h_C^k(i\tau) \left( \frac{y_i^{\max} + y_i^{\min}}{2}, \left\| \frac{y_i^{\max} - y_i^{\min}}{2} \right\| \right) \quad \text{at } t = (i+1)\tau$$

$$y_{i+1}^{\max} = g_C^k(i\tau) \left( \frac{y_i^{\max} + y_i^{\min}}{2}, \left\| \frac{y_i^{\max} - y_i^{\min}}{2} \right\| \right) \quad \text{at } t = (i+1)\tau$$

If controller  $C$  verifies, for all  $t = i\tau$  with  $i \in \mathbb{N}$ :

$$H_C^k \left( \frac{y_i^{\max} + y_i^{\min}}{2}, \left\| \frac{y_i^{\max} - y_i^{\min}}{2} \right\| \right) \in S \wedge G_C^k \left( \frac{y_i^{\max} + y_i^{\min}}{2}, \left\| \frac{y_i^{\max} - y_i^{\min}}{2} \right\| \right) \in S, \quad (5)$$

then system (1) is safe with respect to  $S$ .



Integer approximation using a guaranteed Euler scheme

## Example of UPPAAL implementaiton

```

File Edit View Tools Options Help
[Icons] Editor Simulator ConcreteSimulator Verifier
Project Declarations
  Ego
  Front
  Monitor
  System
  System declarations

if(d1 < d2){return d2;} else{return d1;}

void eulerDiscrete(){
    //double velEgo, velFront,
    double dist, velEgo, velFront, delta;
    double memdist_min, memdist_max, memVF_min, memVF_max, memVE_min, memVE_max;

    int i;

    dist = (distance_gua_evol[0]+distance_gua_evol[1])/2;
    velFront = (velocityFront_gua_evol[0]+velocityFront_gua_evol[1])/2;
    velEgo = (velocityEgo_gua_evol[0]+velocityEgo_gua_evol[1])/2;

    delta = sqrt((distance_gua_evol[1]-distance_gua_evol[0])*(distance_gua_evol[1]-distance_gua_evol[0]))/4 + (velocityFront_(
    memdist_min = dist - delta;
    memdist_max = dist + delta;
    memVF_min = velFront - delta;
    memVF_max = velFront + delta;
    memVE_min = velEgo - delta;
    memVE_max = velEgo + delta;

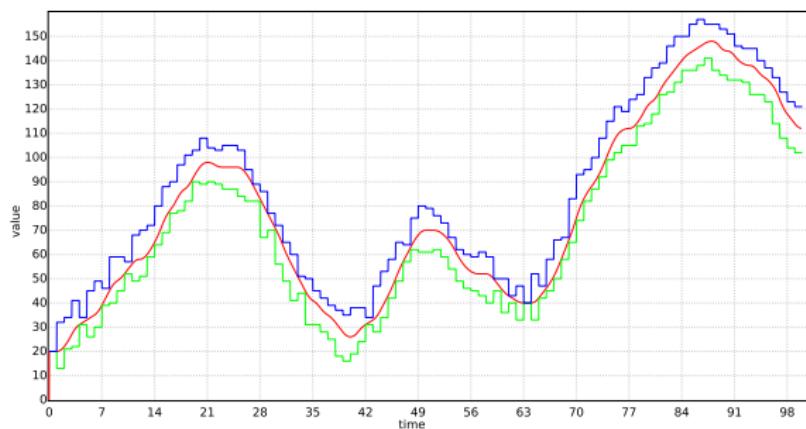
    for(i=0;i<euler_sub_step;i++){
        dist = dist + tau*(velFront - velEgo);
        velEgo = velEgo + tau*accelerationEgo;
        velFront = velFront + tau*accelerationFront;
        delta = delta_mode(delta,find_mode(accelerationFront,accelerationEgo));
        memdist_min = min(memdist_min,dist-delta);
        memdist_max = max(memdist_max,dist+delta);
        memVF_min = min(memVF_min,velFront-delta);
        memVF_max = max(memVF_max,velFront+delta);
        memVE_min = min(memVE_min,velEgo-delta);
        memVE_max = max(memVE_max,velEgo+delta);
    }
    distance_gua[0] = fce(memdist_min);
    distance_gua[1] = cei(memdist_max);
    velocityFront_gua[0] = flo(memVF_min);
    velocityFront_gua[1] = cei(memVF_max);
    velocityEgo_gua[0] = flo(memVE_min);
    velocityEgo_gua[1] = cei(memVE_max);
}

```

Integer approximation using a guaranteed Euler scheme

## Integer approximation of the continuous dynamics

Integer bounding of the real distance within time:



Requirement: all functions must use integers as inputs and return integers



Integer approximation using a guaranteed Euler scheme

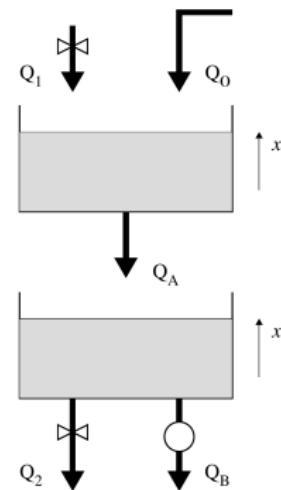
## Two tank case study

### ■ Dynamics:

$$\dot{x}_1 = -x_1 + c_1$$

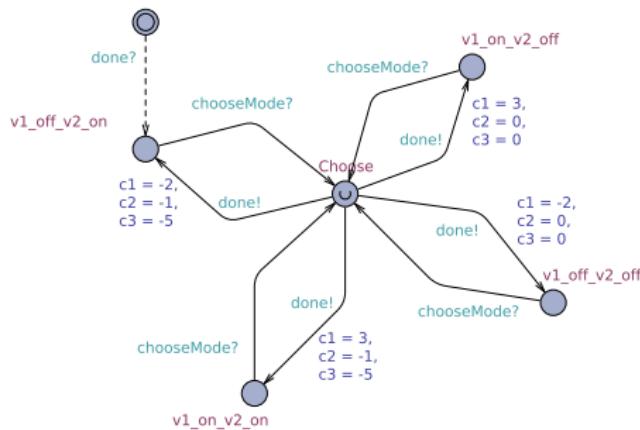
$$\dot{x}_2 = x_1 + c_2 x_2 + c_3$$

with  $c_1 \in \{-2, 3\}$ ,  $c_2 \in \{0, -1\}$ ,  
 $c_3 \in \{0, -5\}$ .



## Two tank case study

Timed game model:

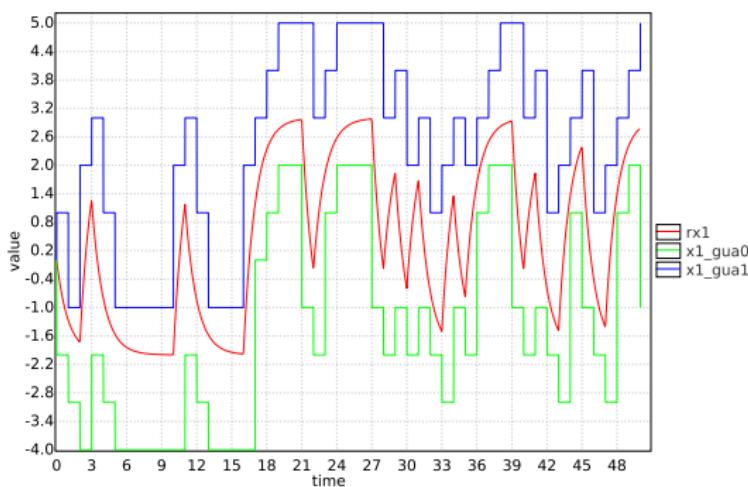




Integer approximation using a guaranteed Euler scheme

## Two tank case study

Guaranteed simulations:

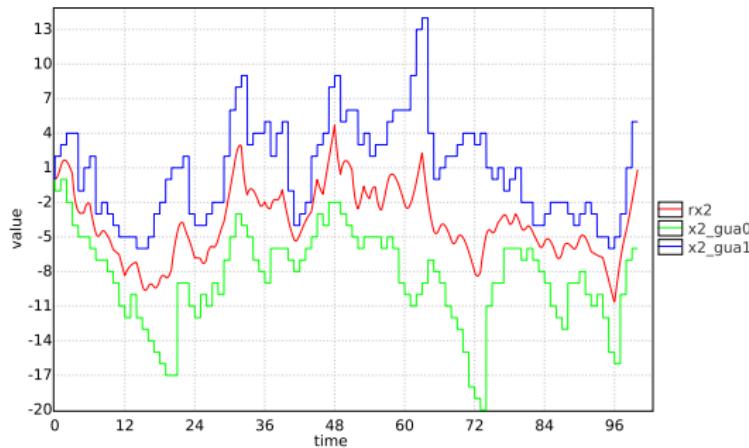




Integer approximation using a guaranteed Euler scheme

## Two tank case study

Guaranteed simulations:

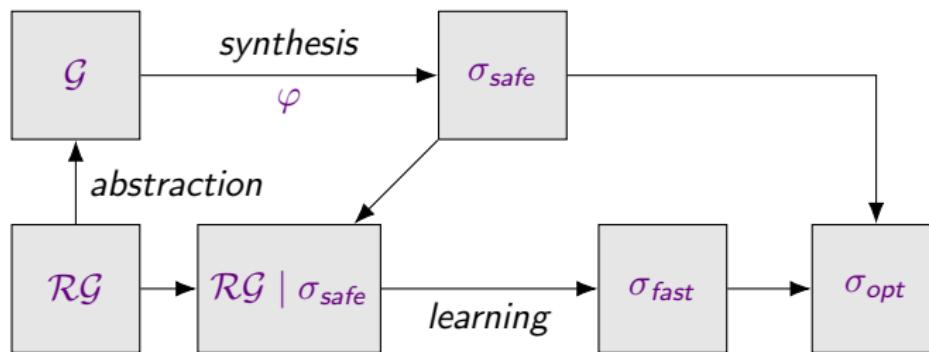




# Outline

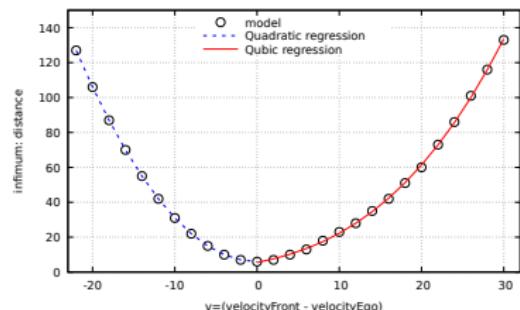
- 1 Guaranteed control of switched systems**
- 2 Safety controller synthesis using timed games**
- 3 Further optimization using Uppaal Stratego**
  - Learning from the safe strategy

## Uppaal Stratego original workflow

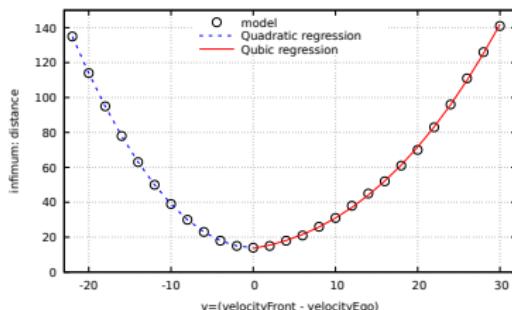


## Optimization of the safe strategy

Smallest achievable distance under the (guaranteed) safe strategy as a function of the relative velocity of the two cars:



(a) Strategy previously computed.



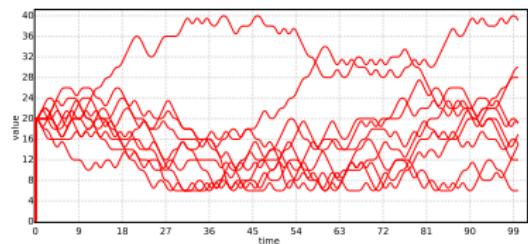
(b) Safe and guaranteed strategy.



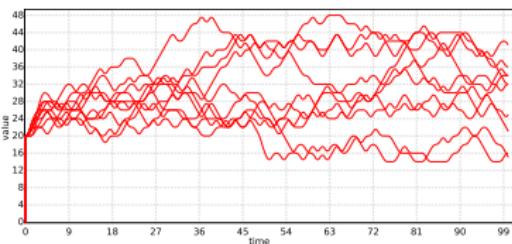
## Learning from the safe strategy

## Optimization of the safe strategy

Distance over time for 10 simulations under the two different optimised strategies:



(a) Optimised strategy computed previously.



(b) Optimised safe and guaranteed strategy.

## Conclusions

Our approach:

- Abstraction from SHG (Switched System) to TG
- Use of UPPAAL TIGA for synthesizing strategies for TG
- Any kind further optimization/model checking is possible

Pros:

- Implemented entirely in UPPAAL (no need of external libraries)
- Possibility of handling time-dependent objectives

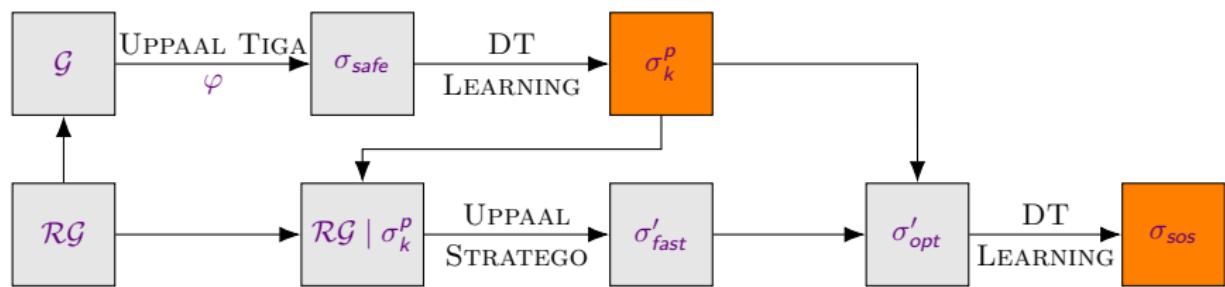
Cons:

- Twice as many variables as the original switched system



## Learning from the safe strategy

## Future work



## New framework

### Definition (Stochastic Hybrid Game)

A stochastic hybrid game  $G$  is a tuple  $(C, U, X, F, \delta)$  where:

- 1  $C$  is a controller with a finite set of (controllable) modes  
 $C = \{c_1, \dots, c_k\}$ ,
- 2  $U$  is the environment with a finite set of (uncontrollable) modes  
 $U = \{u_1, \dots, u_l\}$ ,
- 3  $X = \{x_1, \dots, x_n\}$  is a finite set of continuous (real-valued) variables,
- 4 for each  $c \in C$  and  $u \in U$ ,  $F_{c,u} : \mathbb{R}_{>0} \times \mathbb{R}^X \rightarrow \mathbb{R}^X$  is the flow-function that describes the evolution of the continuous variables over time in the combined mode  $(c, u)$ , and
- 5  $\delta$  is a family of probability functions  $\delta_\gamma : U \rightarrow [0, 1]$ . More precisely,  $\delta_\gamma(u')$  is the probability that  $u$  in the global configuration  $\gamma = (c, u, v)$  will change to the uncontrollable mode  $u'$ .