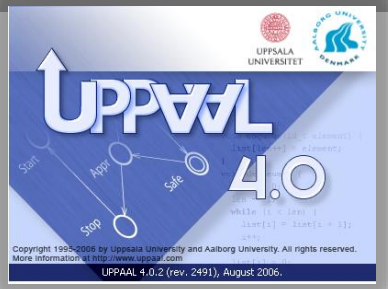
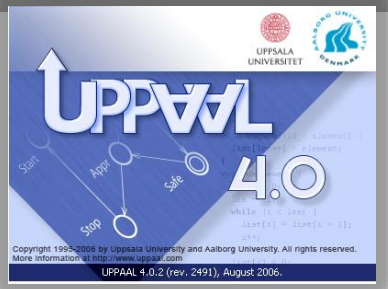
Model Checking, Performance Evaluation, Synthesis and Optimization of Cyber-Physical Systems



Kim G. Larsen
Aalborg University, DENMARK

Model Checking, Performance Evaluation, Synthesis and Optimization of Cyber-Physical Systems



Kim G. Larsen
Aalborg University, DENMARK

Overview

- Timed Automata / UPPAAL
  - Verification

Train Crossing

- Stochastic Priced Timed Automata / UPPAAL SMC
  - Performance Evaluation
  - SMC in a Nutshell
  - Stochastic Hybrid Automata

Train Crossing
Schedulability Analysis
Wireless Sensor Networks
Energy Aware Building

- Timed Games / UPPAAL TIGA
  - Controller Syntesis

Train Crossing
Brick Sorting,Production Cell, HYDAC Case

- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Syntheses

Train Crossing
Go-To-Work
Adaptive Cruice Control
Floor Heating

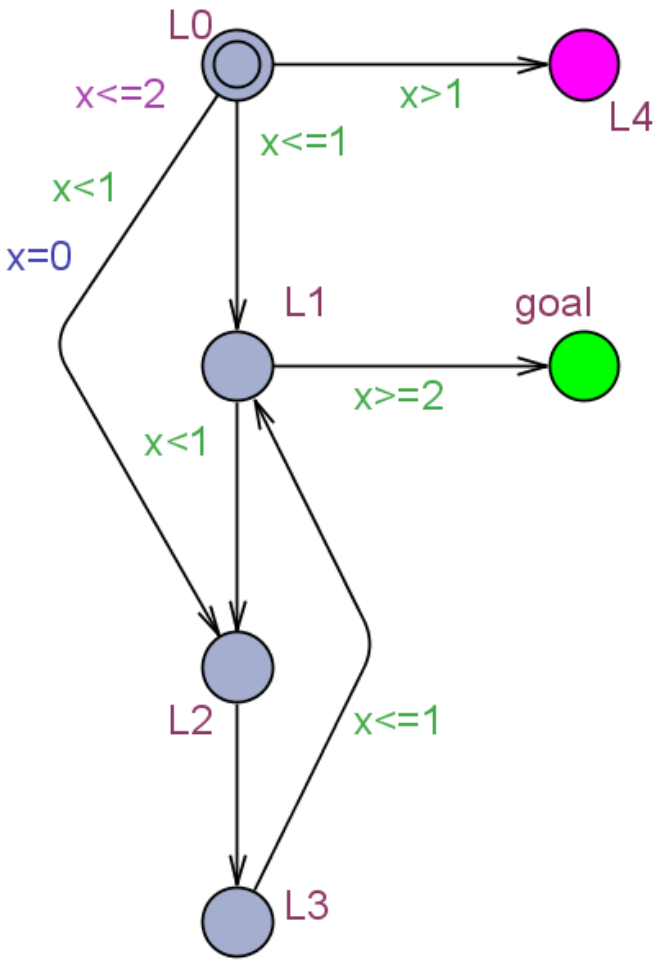
- Conclusion

Timed Games

TIGA

AALBORG UNIVERSITY DENMARK

Timed Automata & Model Checking



State (L1, x=0.81)

Transitions

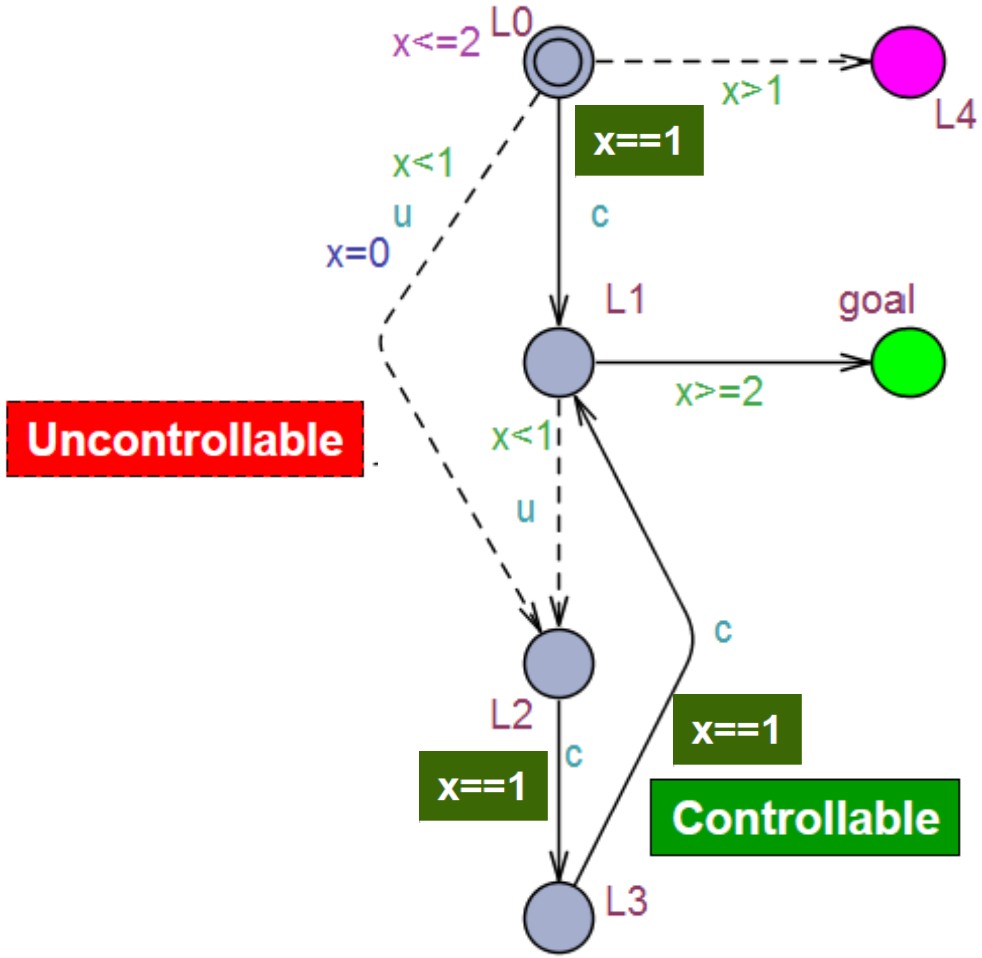
(L1 , x=0.81)
- 2.1 ->
(L1 , x=2.91)
->
(goal , x=2.91)

E⟨⟩ goal ?

A⟨⟩ goal ?

A[ ] ¬ L4 ?

Timed Games & Synthesis



Question
Does their exist a **strategy** that guarantees
A<> Goal ?

Strategy:
$\sigma: (\ell, v) \mapsto \{\lambda, c_{act}\}$

Decidability of Timed Games

Theorem [AMPS98,HK99]

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and "region-based" strategies are sufficient.

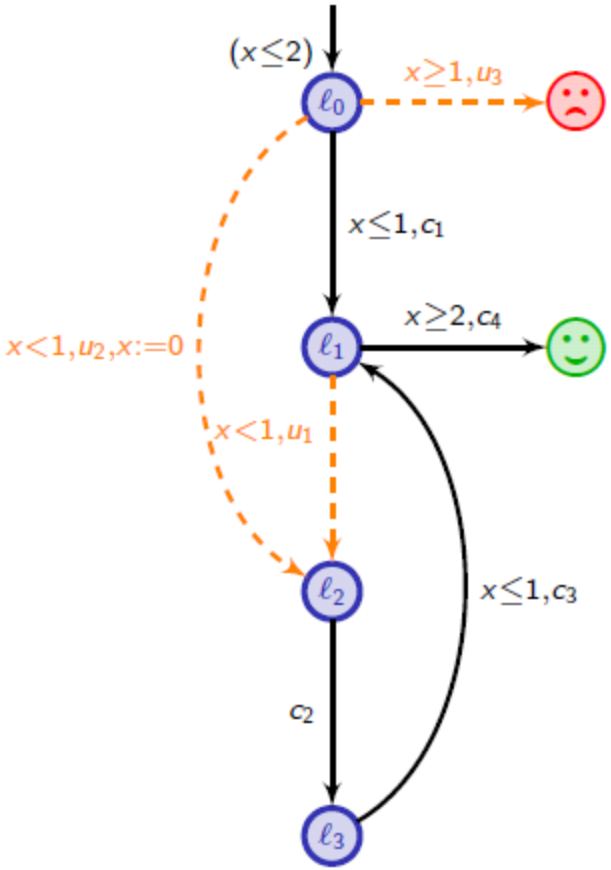
$\rightsquigarrow$ classical regions are sufficient for solving such problems

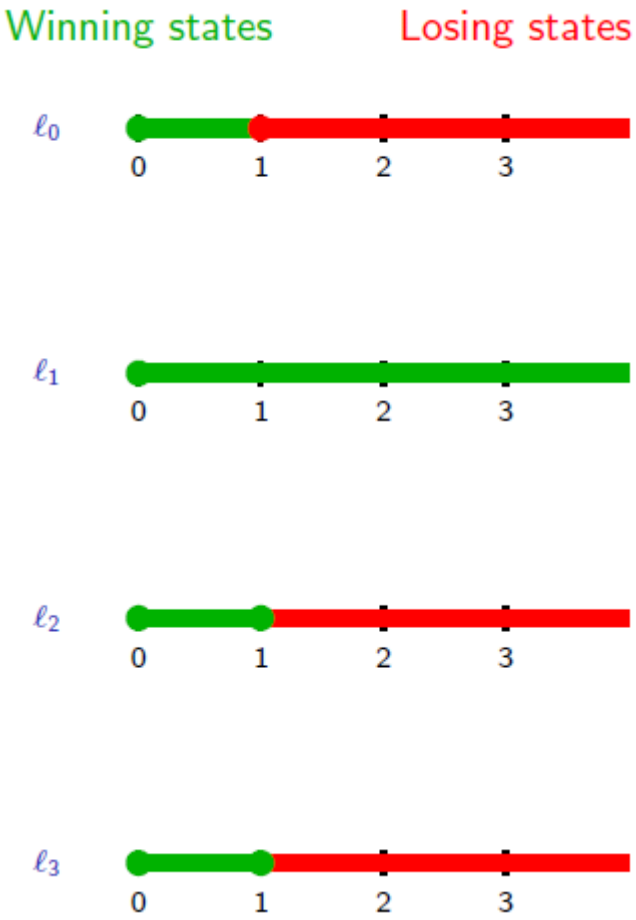
Theorem [AM99,BHPR07,JT07]

Optimal-time reachability timed games are decidable and EXPTIME-complete.

[AM99] Asarin, Maler. As soon as possible: time optimal control for timed automata *(HSCC'99)*.
[BHPR07] Brihaye, Henzinger, Prabhu, Raskin. Minimum-time reachability in timed games *(ICALP'07)*.
[JT07] Jurdziński, Trivedi. Reachability-time games on timed automata *(ICALP'07)*.

Computing Winning States



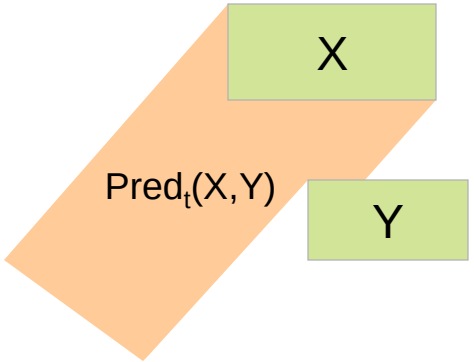


Reachability Games

Definitions

$$\begin{aligned} \text{cPred}(X) &= \{ q \in Q \mid \exists\, q' \in X.\ q \rightarrow_c q' \} \\ \text{uPred}(X) &= \{ q \in Q \mid \exists\, q' \in X.\ q \rightarrow_u q' \} \\ \text{Pred}_t(X,Y) &= \{ q \in Q \mid \exists\, t.\ q^t \in X \ \text{ and } \ \forall\, s \leq t.\ q^s \in Y^C \} \end{aligned}$$

$$\pi(X) = \text{Pred}_t[\, X \cup \text{cPred}(X) \,,\, \text{uPred}(X^C) \,]$$



Theorem:
The set of winning states is obtained as the least fixpoint of the function: $\mathbf{X \mapsto \pi(X) \cup Goal}$

Symbolic On-the-fly Algorithms for Timed Games [CDF+05, BCD+07]

- $S, S', \ldots$
  are *symbolic* states, i.e. sets of concrete states;
- G
  is the set of (concrete) goal states;
- $E = \{S \xrightarrow{c} S', S \xrightarrow{u} S'\}$
  the (finite) set of symbolic transitions (controlla
- $Waiting \subseteq E$
  is the list of symbolic transitions waiting to be p
- $Passed$
  is the list of the passed symbolic states;
- $Win[S] \subseteq S$
  is the subset of S currently known to be winnin
- $Depend[S] \subseteq E$
  indicates the edges (predecessors) of S which mu
  information about S is obtained.

symbolic version of on-the-fly MC algorithm for modal mu-calculus
Liu & Smolka 98

Initialization:

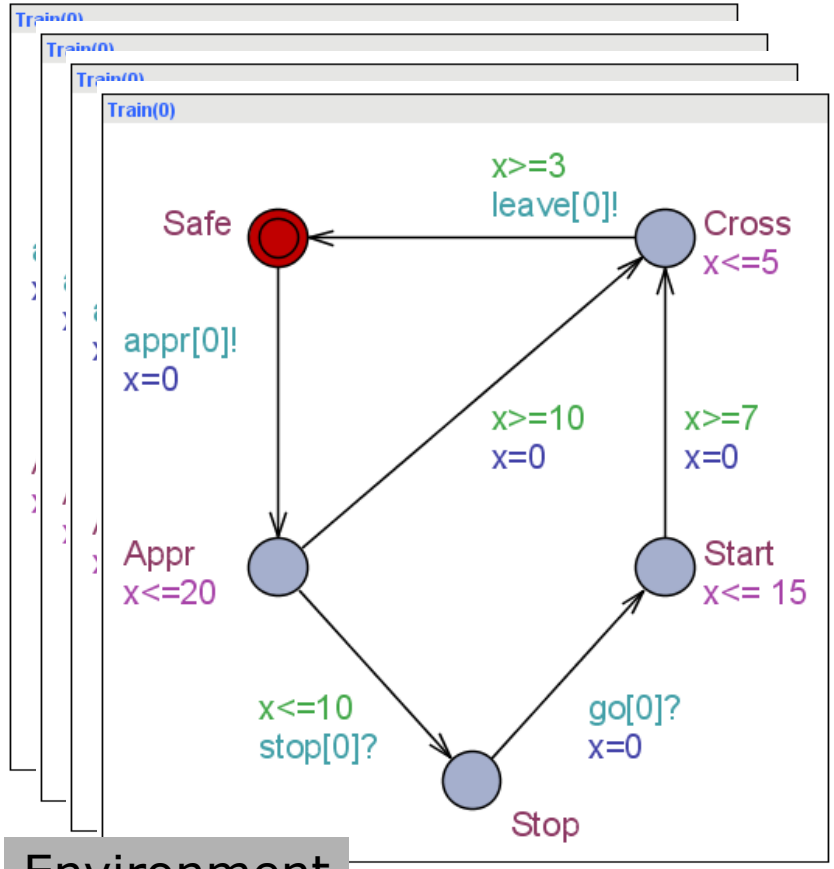
$Passed \leftarrow \{S_0\}$ **where** $S_0 = \{(\ell_0, \vec{0})\}^{\nearrow}$;
$Waiting \leftarrow \{(S_0, \alpha, S') \mid S' = \mathsf{Post}_\alpha(S_0)^{\nearrow}\}$;
$Win[S_0] \leftarrow S_0 \cap (\{\mathsf{Goal}\} \times \mathbb{R}^X_{\geq 0})$;
$Depend[S_0] \leftarrow \emptyset$;

Main:

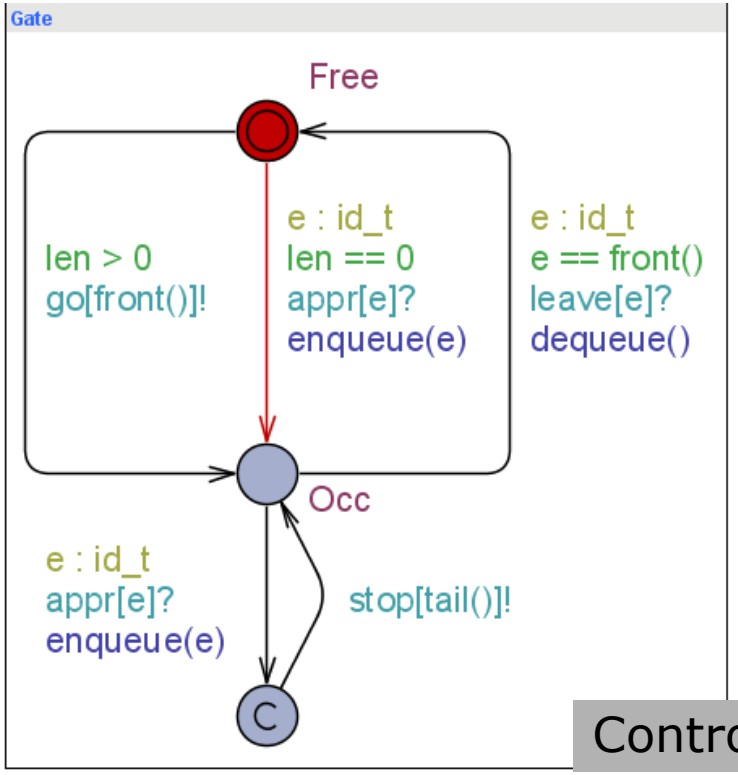
while $((Waiting \neq \emptyset) \wedge (s_0 \notin Win[S_0]))$ **do**
  $e = (S, \alpha, S') \leftarrow pop(Waiting)$;
  if $S' \notin Passed$ **then**
    $Passed \leftarrow Passed \cup \{S'\}$;
    $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
    $Win[S'] \leftarrow S' \cap (\{\mathsf{Goal}\} \times \mathbb{R}^X_{\geq 0})$;
    $Waiting \leftarrow Waiting \cup \{(S', \alpha, S'') \mid S'' = \mathsf{Post}_\alpha(S')^{\nearrow}\}$;
    if $Win[S'] \neq \emptyset$ **then** $Waiting \leftarrow Waiting \cup \{e\}$;
  else (* **reevaluate** *)^a
    $Win^* \leftarrow \mathsf{Pred}_t(Win[S] \cup \bigcup_{S \xrightarrow{c} T} \mathsf{Pred}_c(Win[T]),$
      $\bigcup_{S \xrightarrow{u} T} \mathsf{Pred}_u(T \setminus Win[T])) \cap S$;
    if $(Win[S] \subsetneq Win^*)$ **then**
      $Waiting \leftarrow Waiting \cup Depend[S]$; $Win[S] \leftarrow Win^*$;
    $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
  endif
endwhile

[CDF+05] Cassez, David, Fleury, Larsen, Lime. Efficient on-the-fly algorithms for the analysis of timed games *(CONCUR'05)*.
[BCD+07] Berhmann, Cougnard, David, Fleury, Larsen, Lime. Uppaal-Tiga: Time for playing games! *(CAV'07)*.

Model Checking (ex Train Gate)



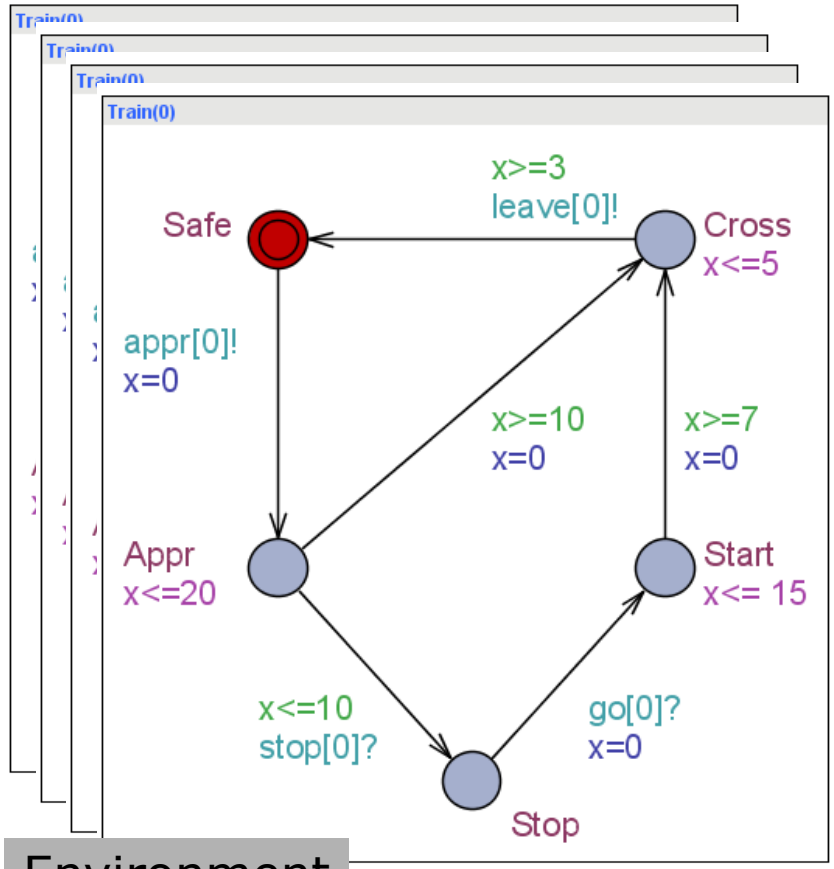
Environment



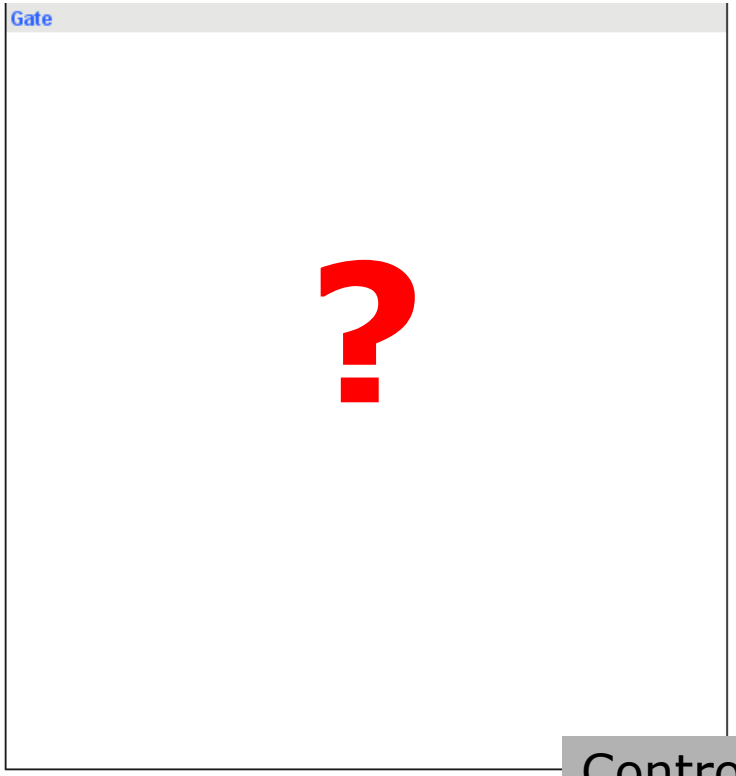
Controller

ϕ: Never two trains at the crossing at the same time

Synthesis (ex Train Gate)



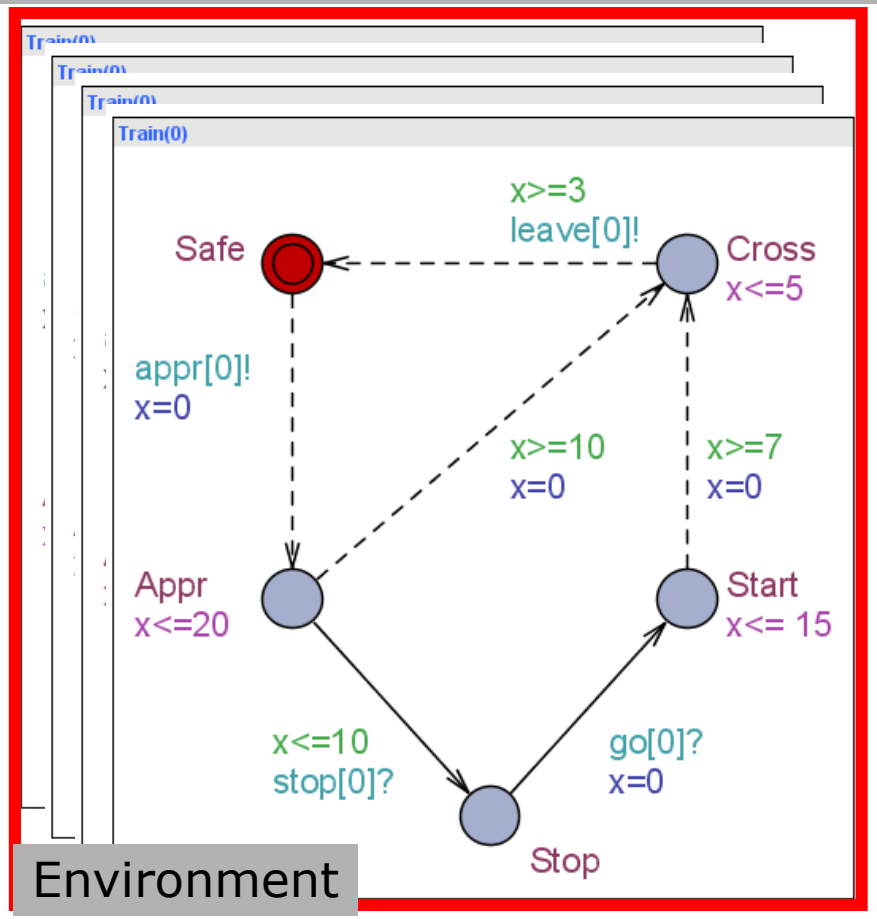
Environment



Controller

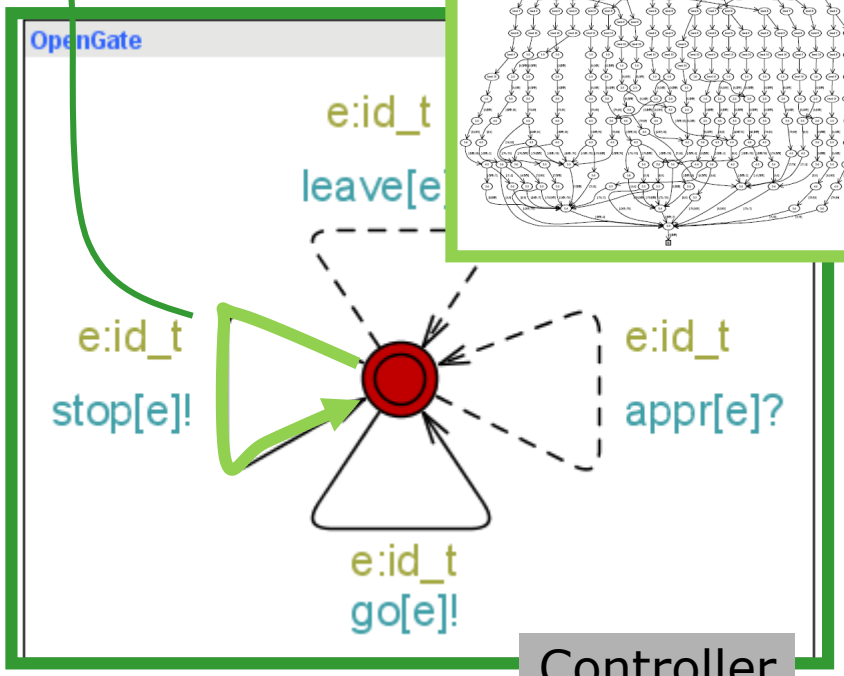
ϕ: Never two trains at the crossing at the same time

Timed Games



Environment

Controllable



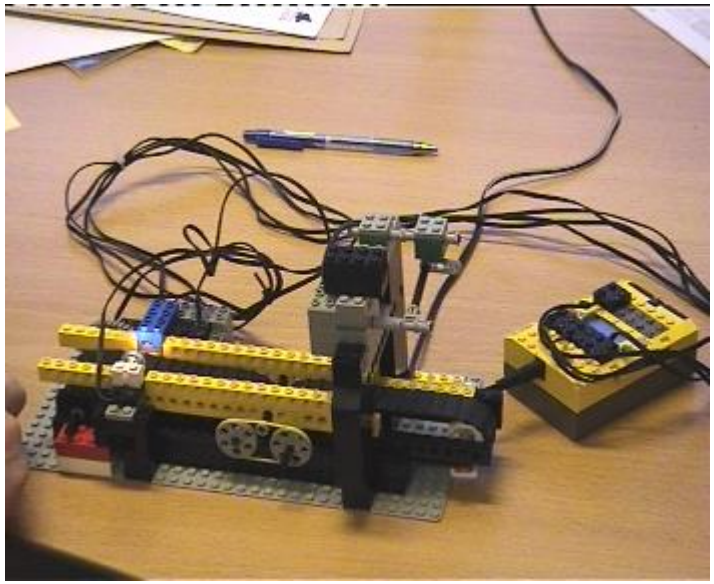
Controller

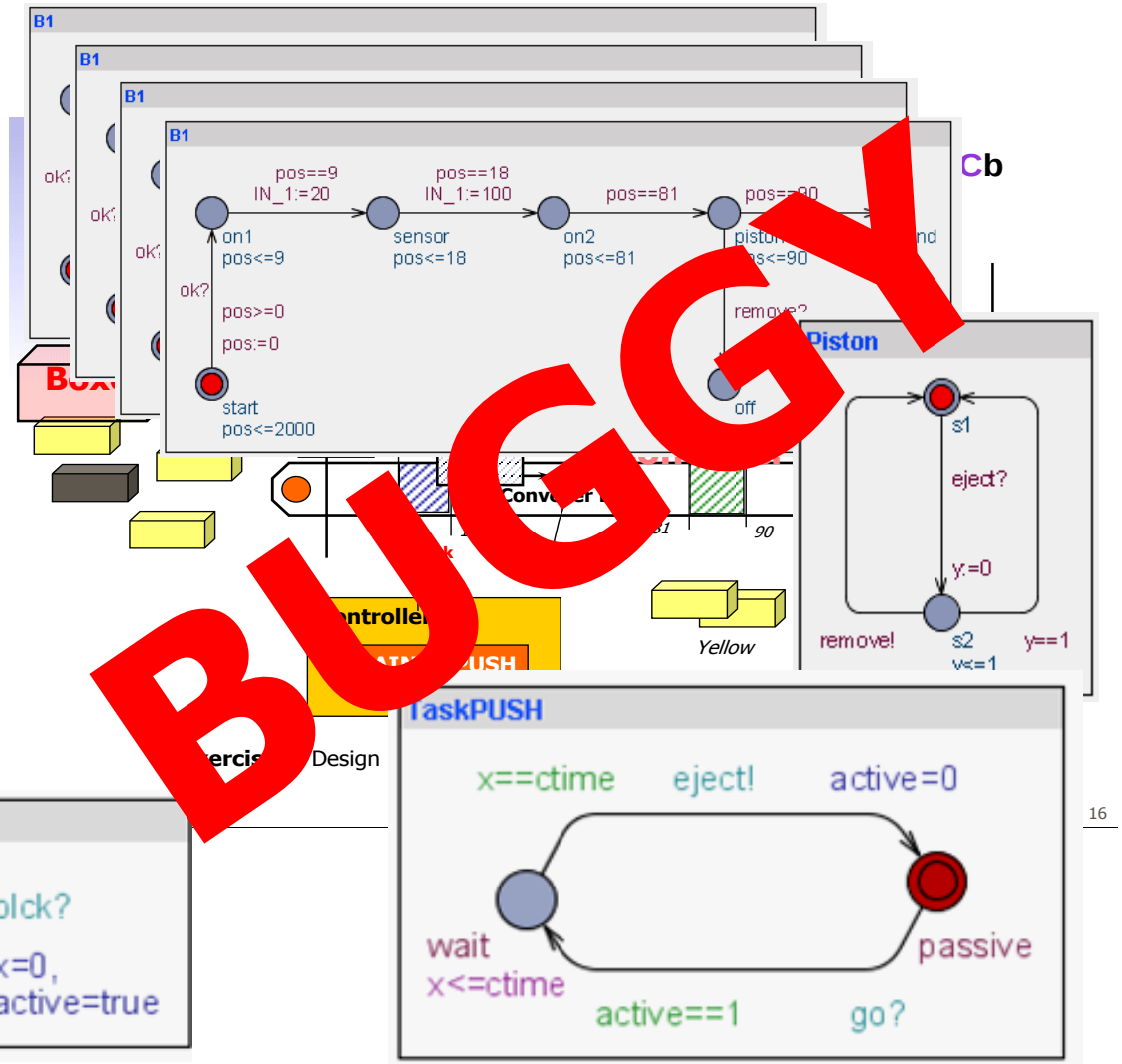
Find strategy for controllable actions st behaviour satisfies ϕ

ϕ: Never two trains at the crossing at the same time

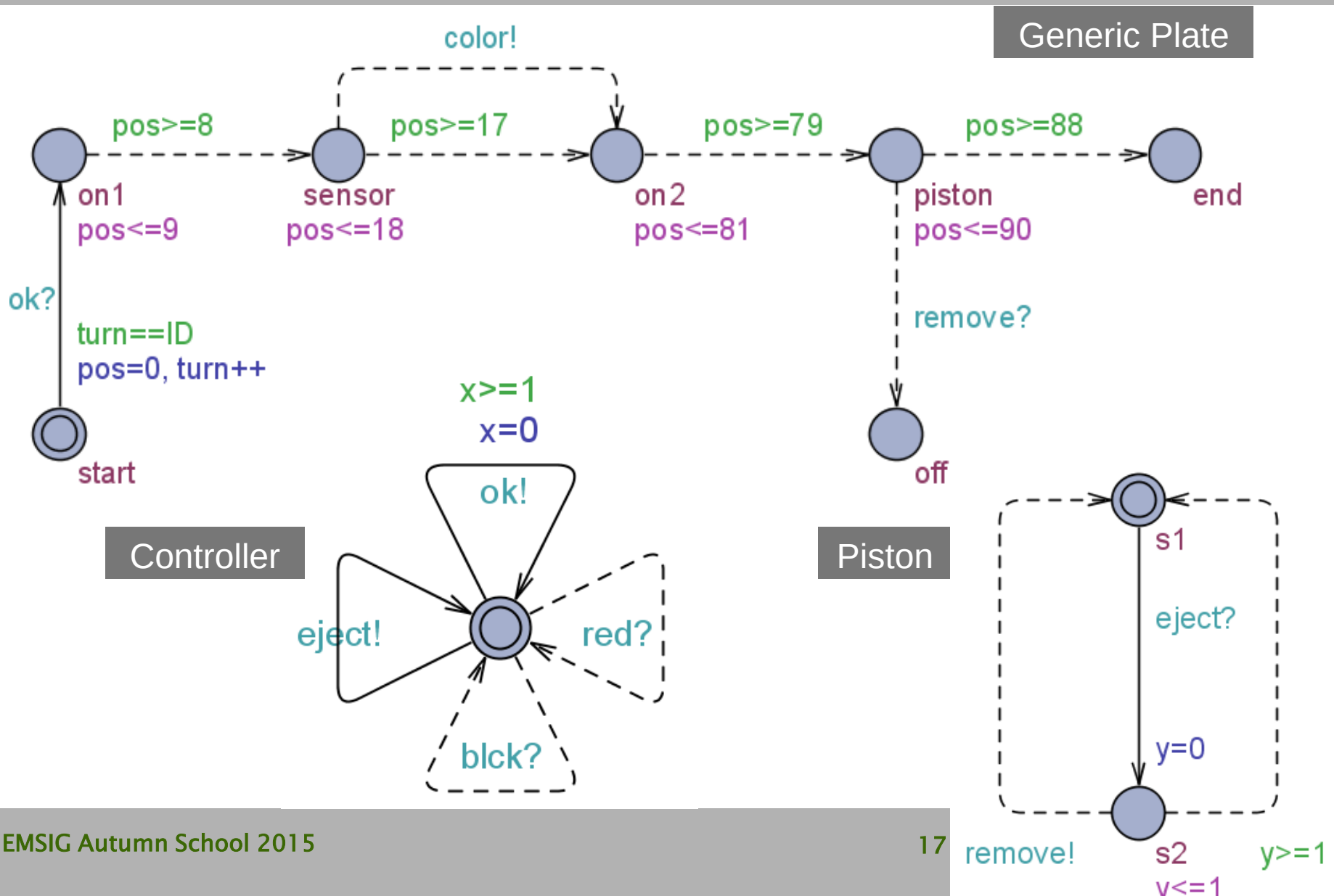
DEMO

A Buggy Brick Sorting Program



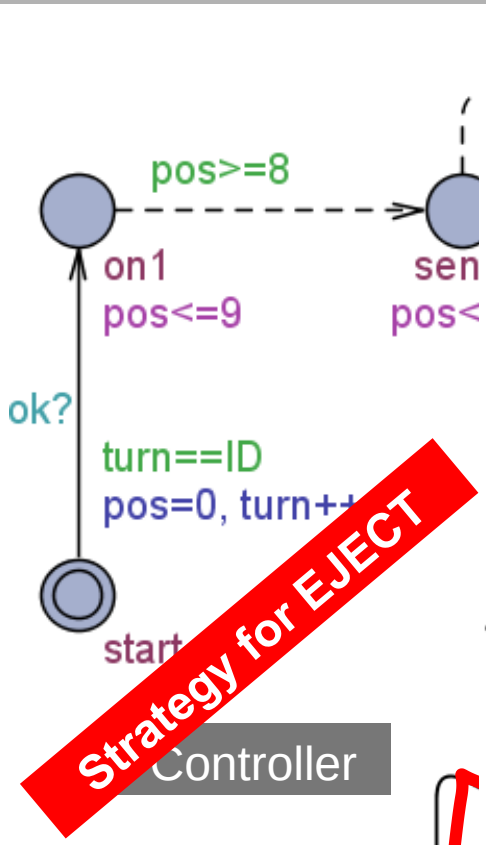


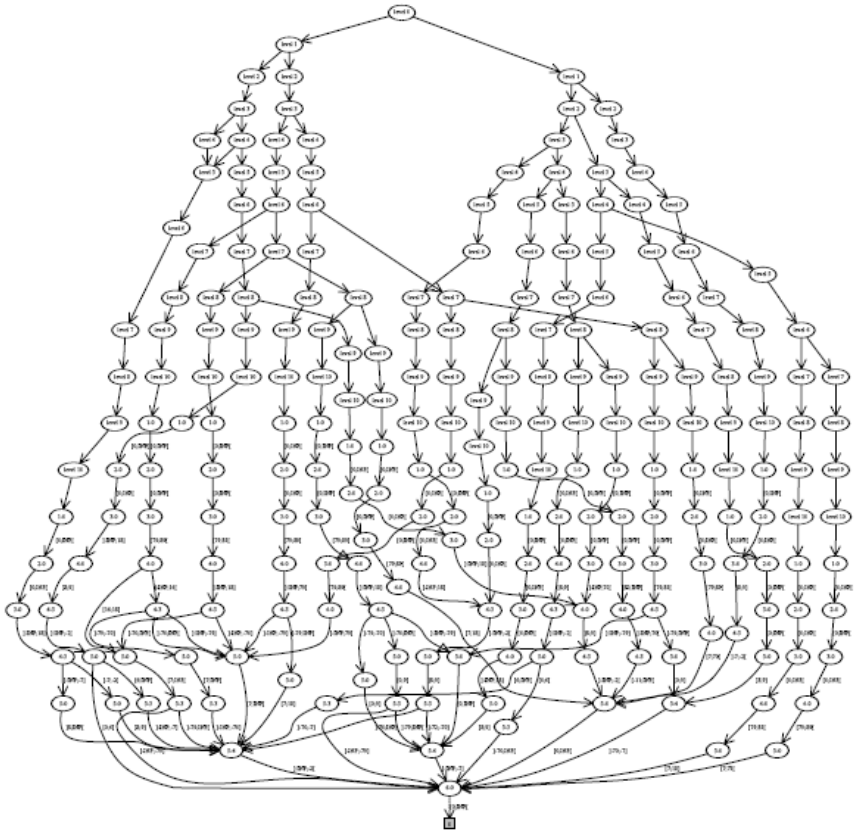
Brick Sorting

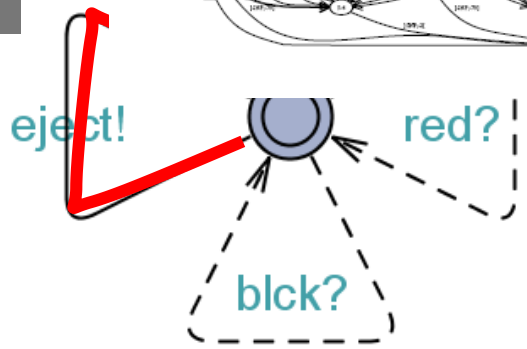


DEMO

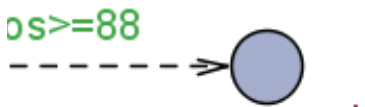
Brick Sorting

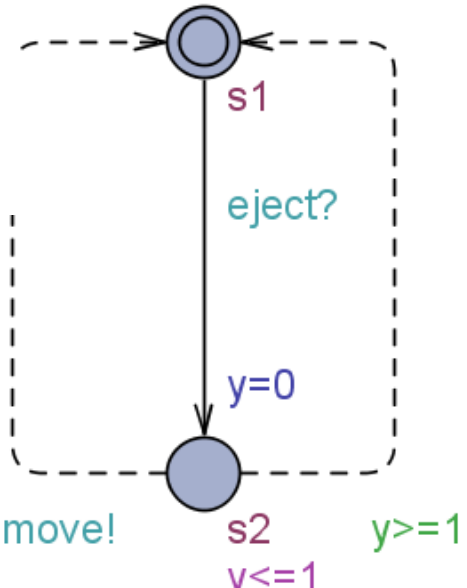






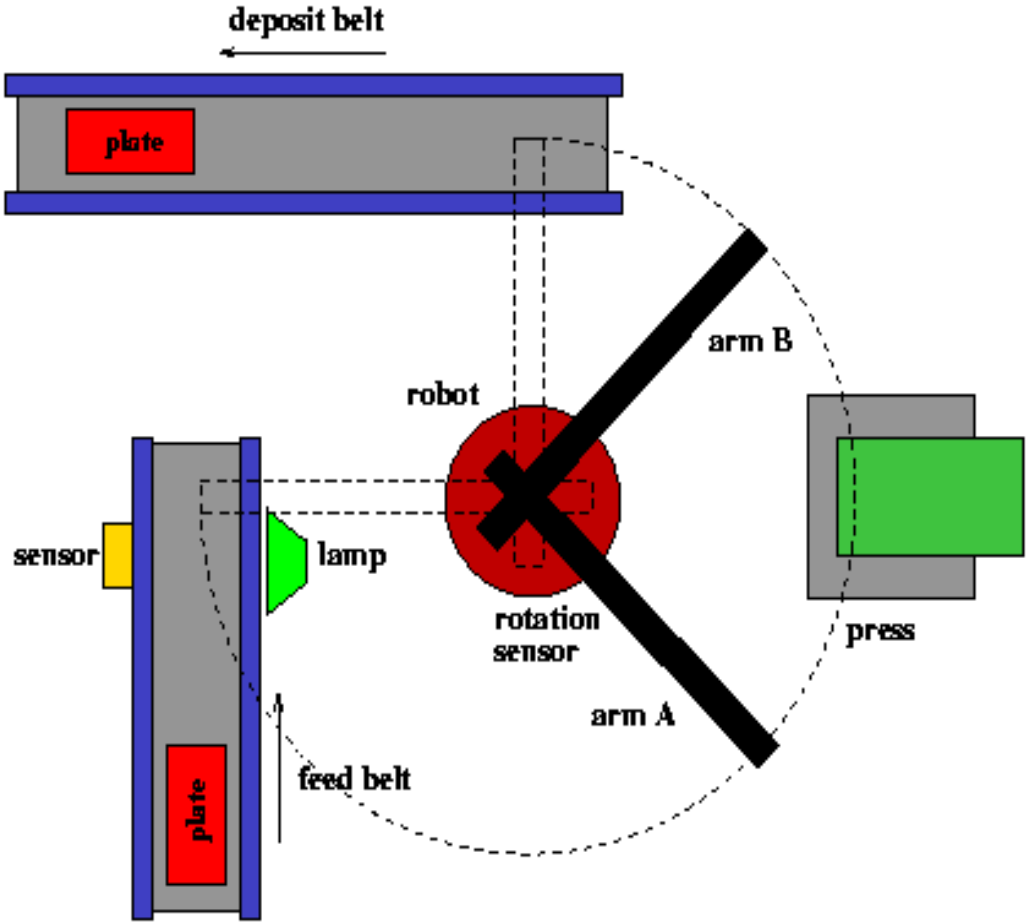
Generic Plate



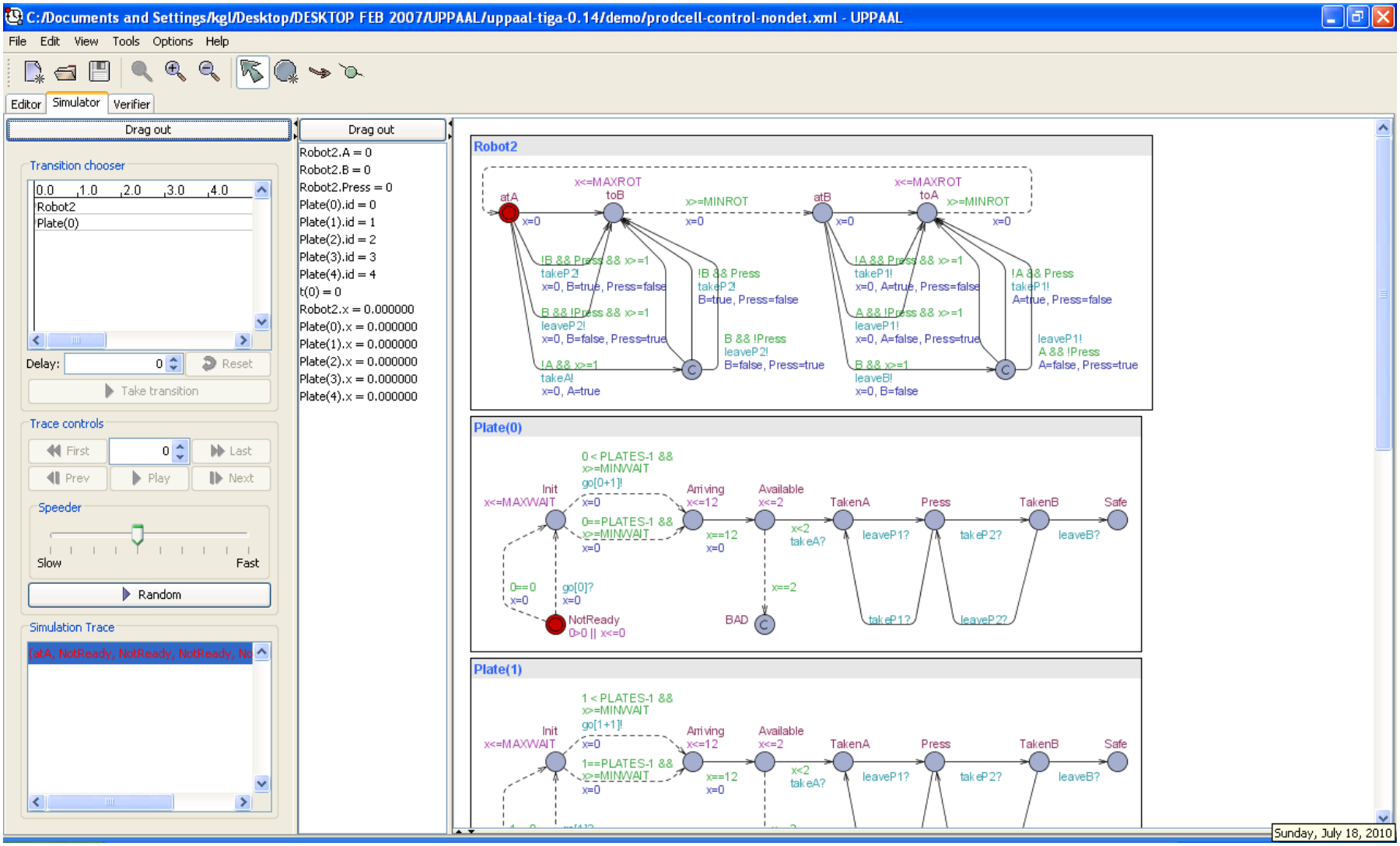


Production Cell Overview

- Realistic case-study described in several formalisms (1994 and later).
- **Objective**: stamp metal plates in press.
- feed belt, two-armed robot, press, and deposit belt.



Production Cell in UPPAAL Tiga



Experimental Results

[CDF+05]

Plates		Basic		Basic +inc		Basic +inc +pruning		Basic+lose +inc +pruning		Basic+lose +inc +topt	
		time	mem	time	mem	time	mem	time	mem	time	mem
2	win	0.0s	1M	0.0s	1M	0.0s	1M	0.0s	1M	0.04s	1M
	lose	0.0s	1M	0.0s	1M	0.0s	1M	0.0s	1M	n/a	n/a
3	win	0.5s	19M	0.0s	1M	0.0s	1M	0.1s	1M	0.27s	4M
	lose	1.1s	45M	0.1s	1M	0.0s	1M	0.2s	3M	n/a	n/a
4	win	33.9s	1395M	0.2s	8M	0.1s	6M	0.4s	5M	1.88s	13M
	lose	-	-	0.5s	11M	0.4s	10M	0.9s	9M	n/a	n/a
5	win	-	-	3.0s	31M	1.5s	22M	2.0s	16M	13.35s	59M
	lose	-	-	11.1s	61M	5.9s	46M	7.0s	41M	n/a	n/a
6	win	-	-	89.1s	179M	38.9s	121M	12.0s	63M	220.3s	369M
	lose	-	-	699s	480M	317s	346M	135.1s	273M	n/a	n/a
7	win	-	-	3256s	1183M	1181s	786M	124s	319M	6188s	2457M
	lose	-	-	-	-	16791s	2981M	4075s	2090M	n/a	n/a

Model	c3		c6		c12		u3		u6		u12	
Old	0.1s	1M	12s	63M	-	-	0.2s	3M	235s	273M	-	-
New	0.05s	3.5M	0.05s	3.5M	0.14s	55M	0.02s	3.5M	0.04s	3.5M	0.12s	55M

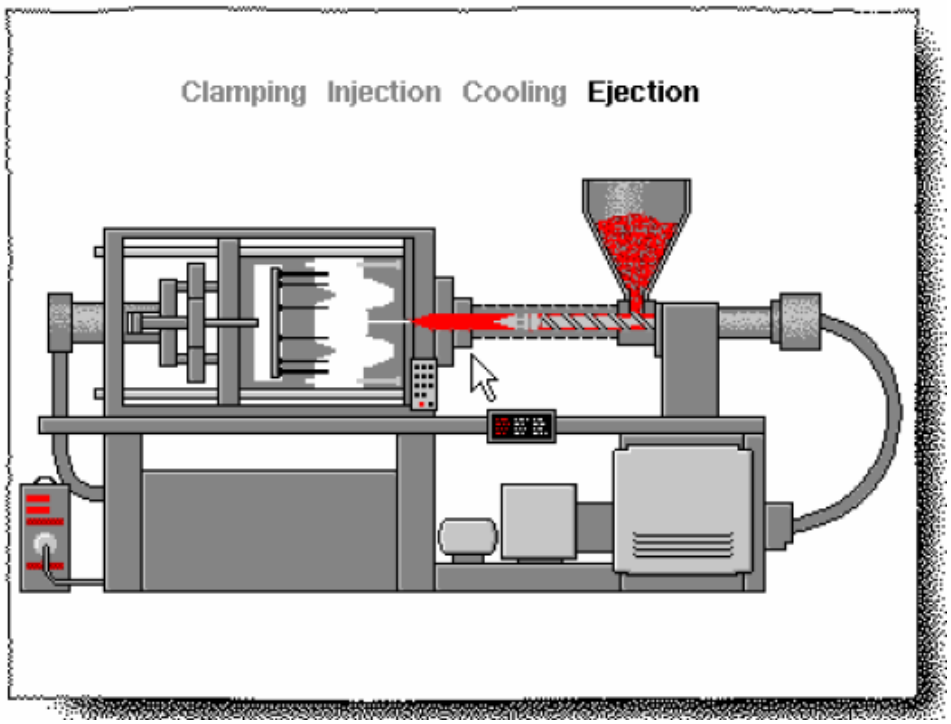
[BCD+07]

[CDF+05] Cassez, David, Fleury, Larsen, Lime. Efficient on-the-fly algorithms for the analysis of timed games *(CONCUR'05)*.
[BCD+07] Berhmann, Cougnard, David, Fleury, Larsen, Lime. Uppaal-Tiga: Time for playing games! *(CAV'07)*.

Plastic Injection Molding Machine

Quasiomodo

[CJL+09]

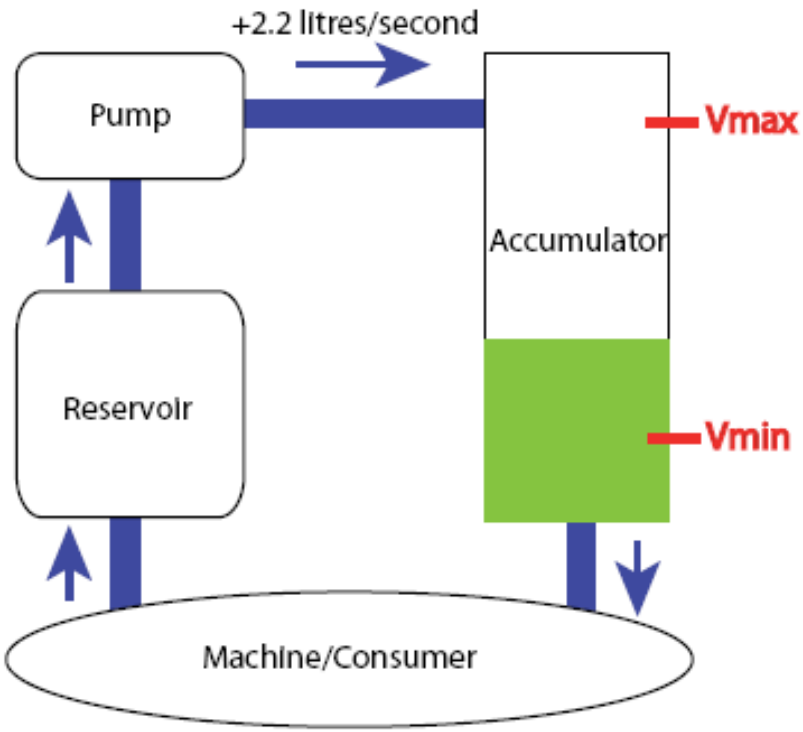


HYDAC

- Robust and optimal control
- Tool Chain
  - Synthesis: UPPAAL TIGA
  - Verification: PHAVer
  - Performance: SIMULINK
- 40% improvement of existing solutions..

[CJL+09] Cassez, Jessen, Larsen, Raskin, Reynier.Automatic Synthesis of Robust and Optimal Controllers – An Industrial Case Study *(HSCC'09)*.

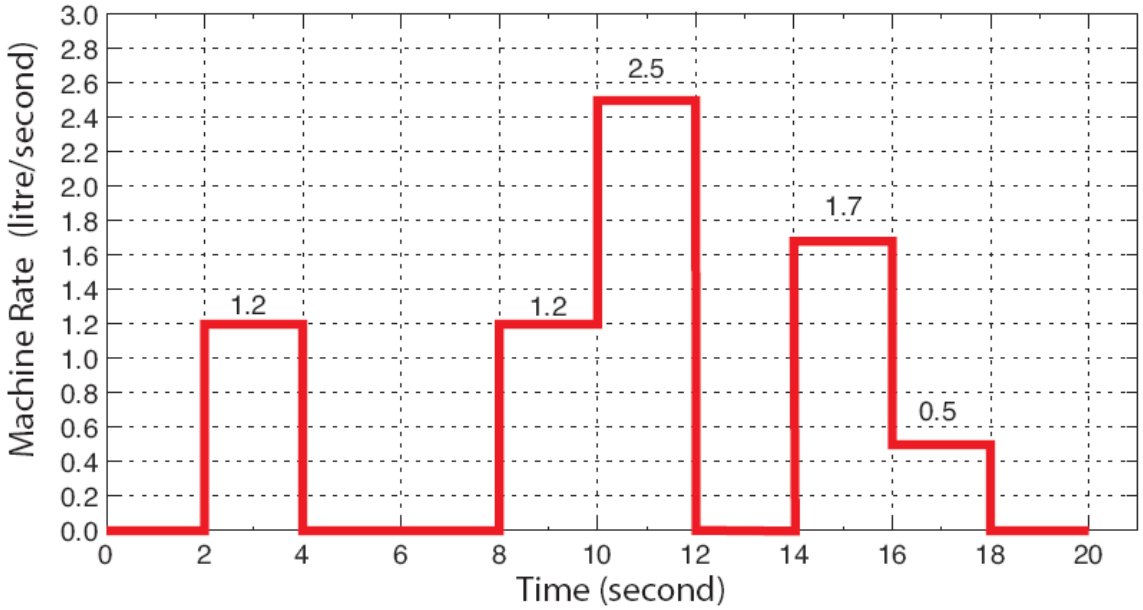
Oil Pump Control Problem



- R1: stay within safe interval [4.9,25.1]
- R2: minimize average/overall oil volume

$$\int_{t=0}^{t=T} v(t)dt \,/\, T$$

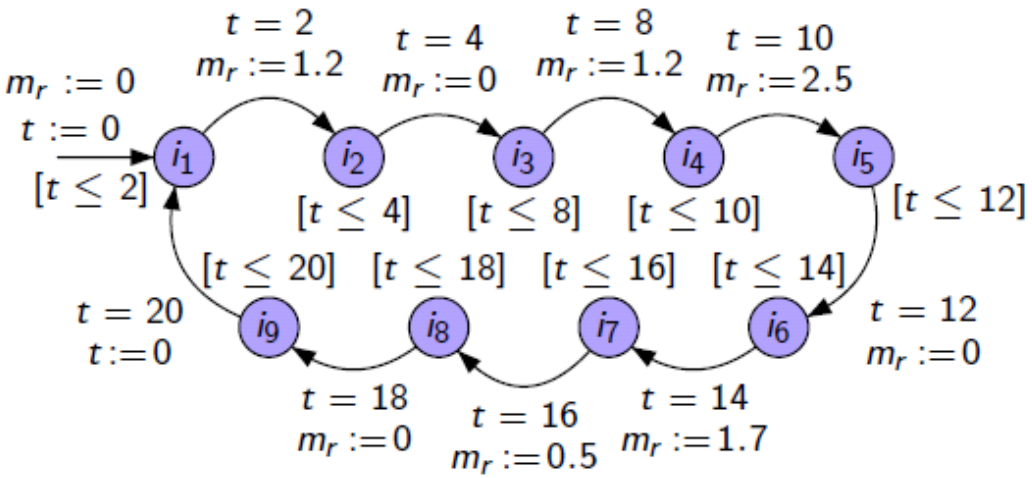
The Machine (consumption)



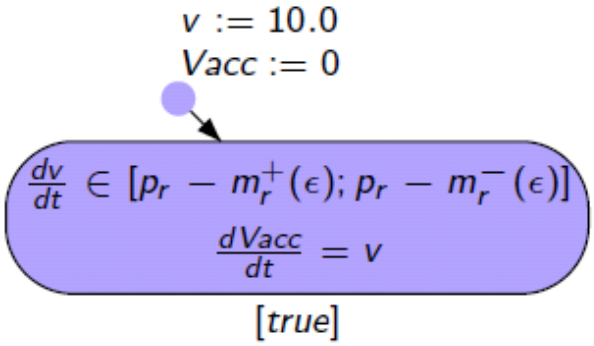
- Infinite cyclic demand to be satisfied by our control strategy.
- P: latency 2 s between state change of pump

- F: noise 0.1 l/s

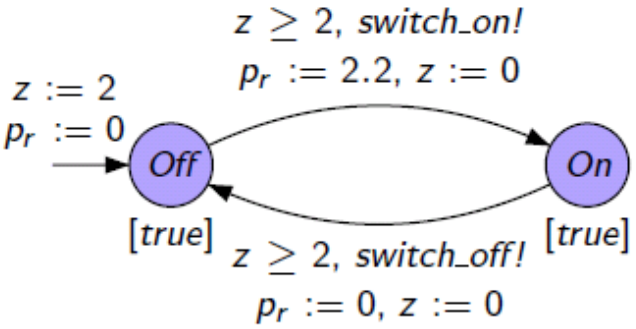
Hybrid Game Model



(a) The Machine



(b) The Accumulator



(c) The Pump

Undecidability

→

Discretization

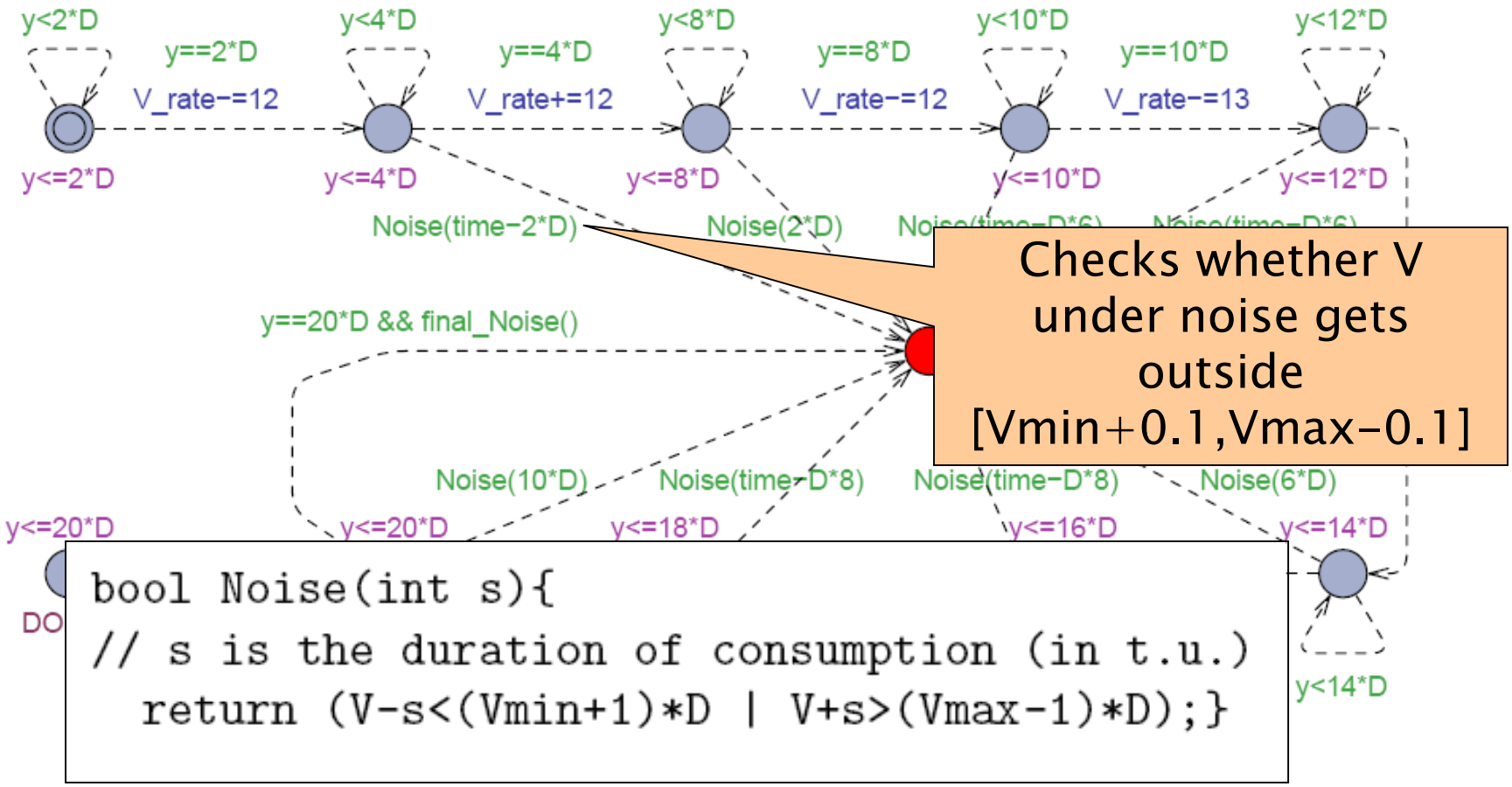
Abstract Game Model

- UPPAAL Tiga
  offers games of perfect information

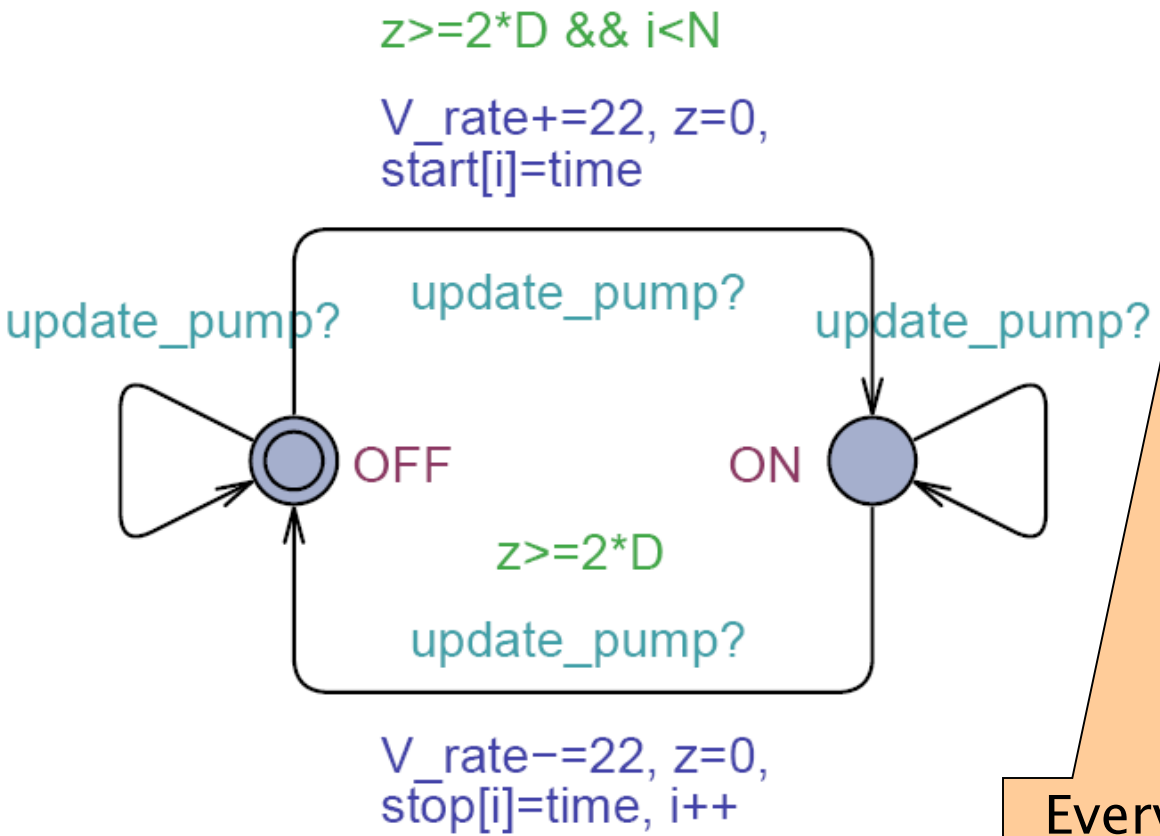
- Abstract game model such that states only contain information about:
  - Volume of oil at the beginning of cycle
  - The ideal volume as predicted by the consumption cycle
  - Current time within the cycle
  - State of the Pump (on/off)
  - Discrete model

D
V, V_rate
V_acc
time

Machine (uncontrollable)



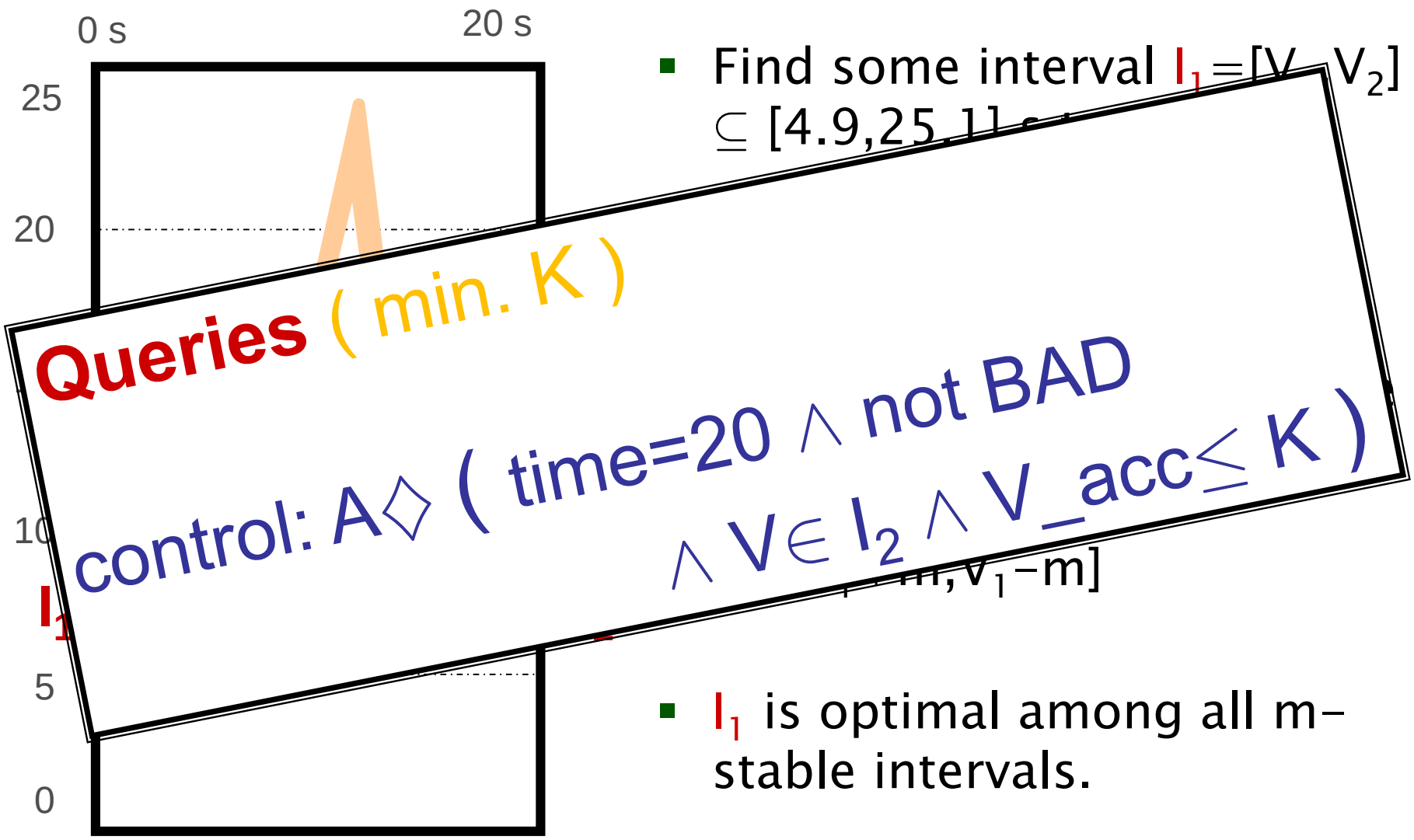
Pump (controllable)




```
void update_val(){
   int V_pred = V;
   time++;
   V+=V_rate;
   V_acc+=V+V_pred;
}
```

Every 1 (one) seconds

Global Approach



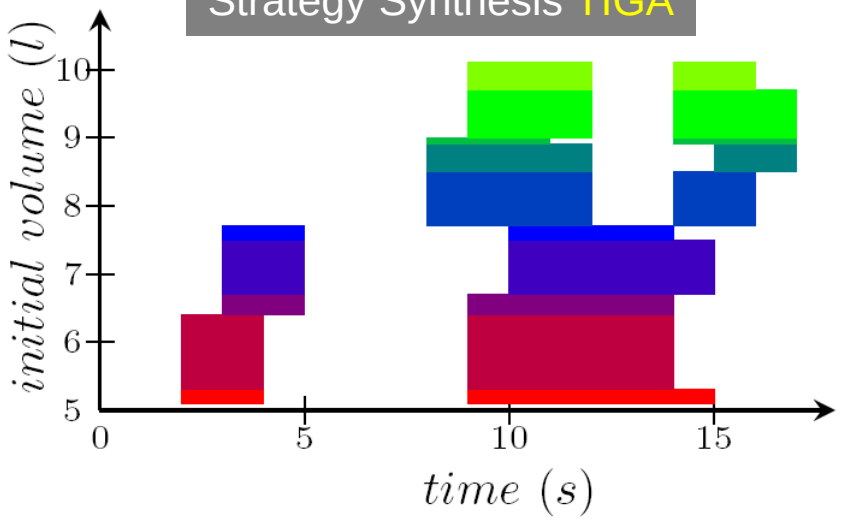
- Find some interval $l_1 = [V_1, V_2]$ ⊆ [4.9,25.1] s.t.
  $[V_1-m, V_1-m]$

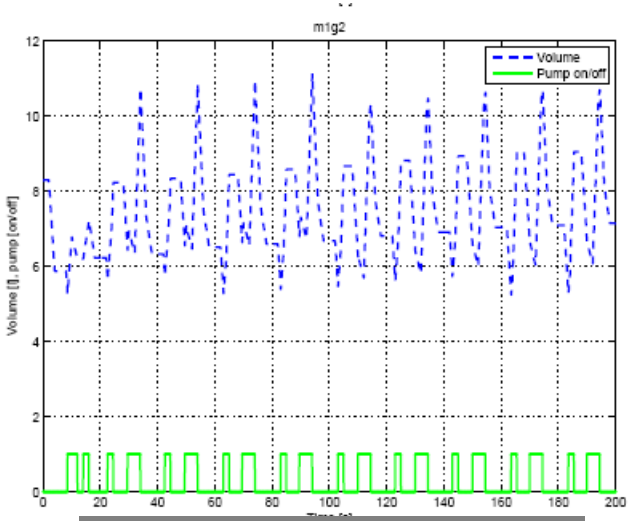
- l_1 is optimal among all m-stable intervals.

Tool Chain

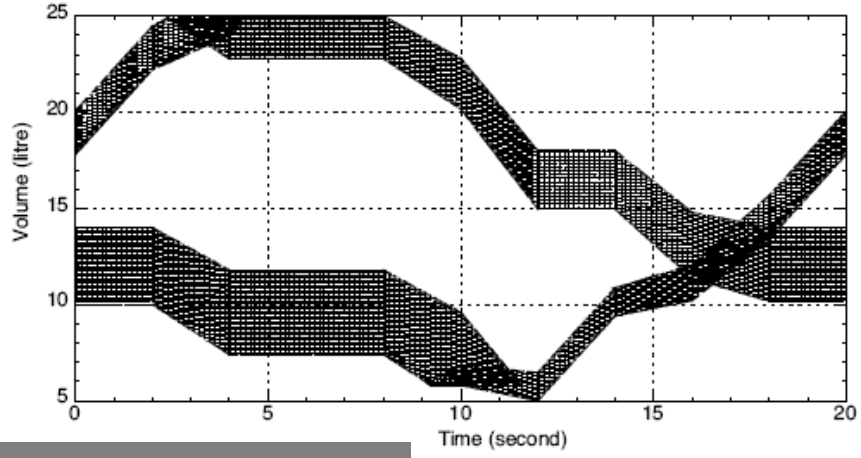
Quasiomodo

Strategy Synthesis TIGA





Performance Evaluation SIMULINK



Verification PHAVER

Guaranteed

Correctness

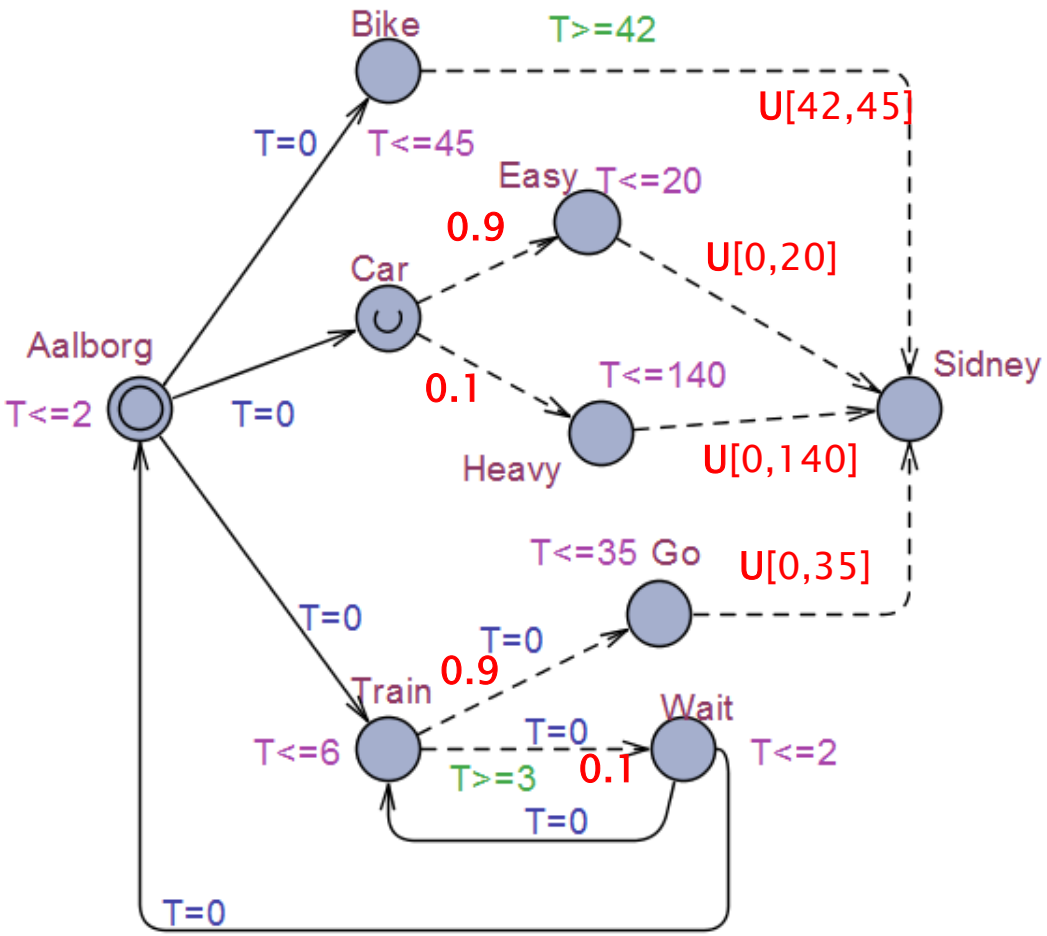
Robustness

with

40% Improvement

Stochastic Priced Timed Games

Going to Sydney – in 1 hour



Can I get to Sidney?
(1-player)

Will I always come to Sidney?
(1-player)

What is the optimal WC strategy?
(2-player)

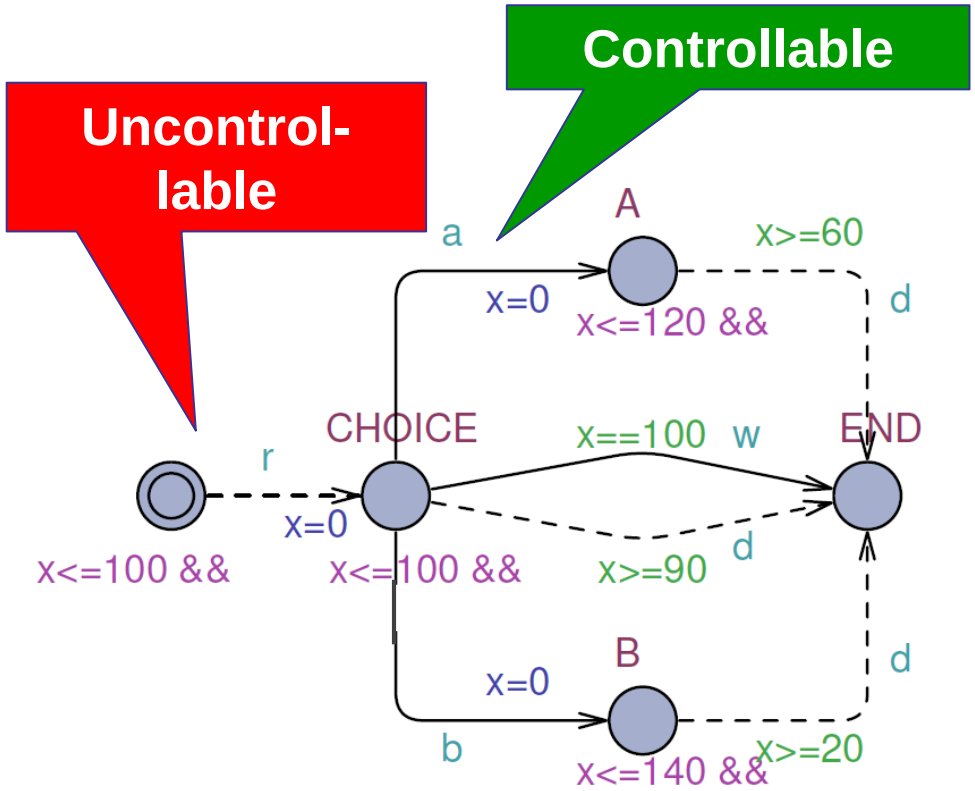
Is there a strategy guaranteeing WC <= 60?
(2-player)

What is the optimal strategy?
(1½-player)

What is the optimal strategy Guarenteeing WC <= 60?
(1½-player)

Timed Games

TIGA



- Strategy:

$$\sigma : Exec^{f}_{\mathcal{G}} \rightharpoonup \mathcal{P}\left(\Sigma_c \cup \{\lambda\}\right) \setminus \{\emptyset\}$$

- Memoryless, deterministic, most permissive.

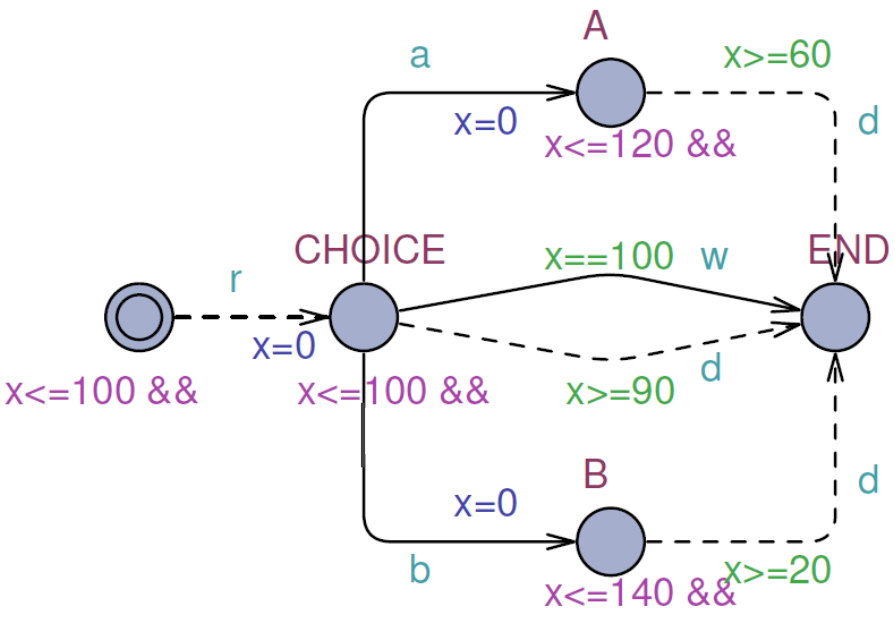
Run

$$\pi = (\text{INIT}, x = 0) \xrightarrow{50.1\ \text{r}} \rightarrow (\text{CHOICE}, x = 0) \xrightarrow{2.4\ \text{a}} \rightarrow (\text{A}, x = 0) \xrightarrow{20.3\ \text{d}} \rightarrow (\text{END}, x = 20.3)$$

$$\text{Total time} = 50.1 + 2.4 + 20.3 = 72.8$$

Timed Games –

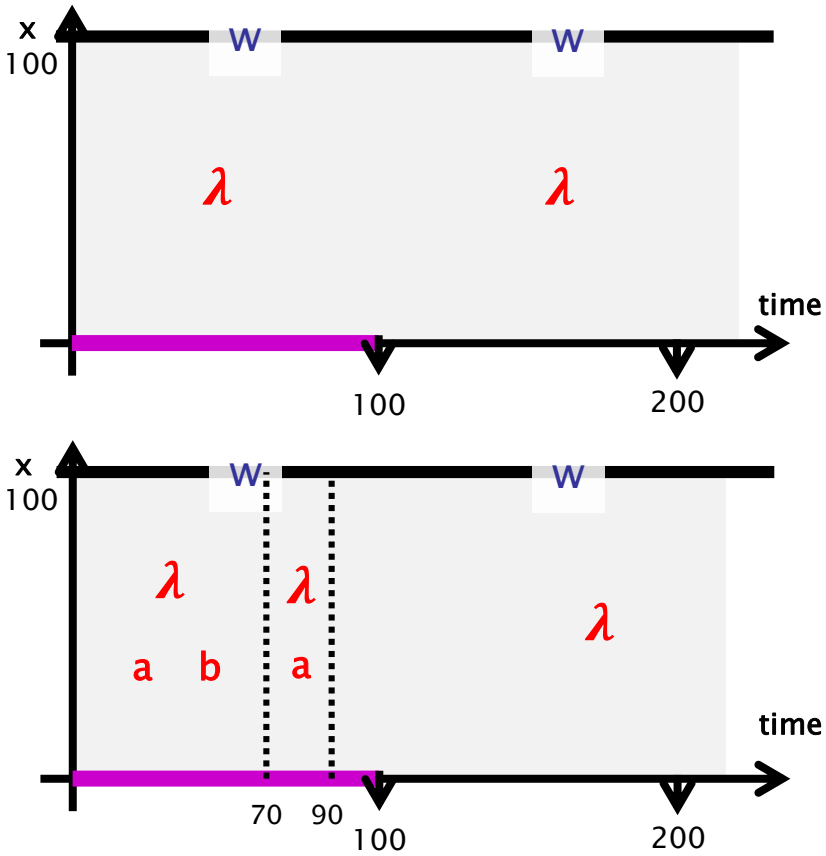
Time Bounded Reachability



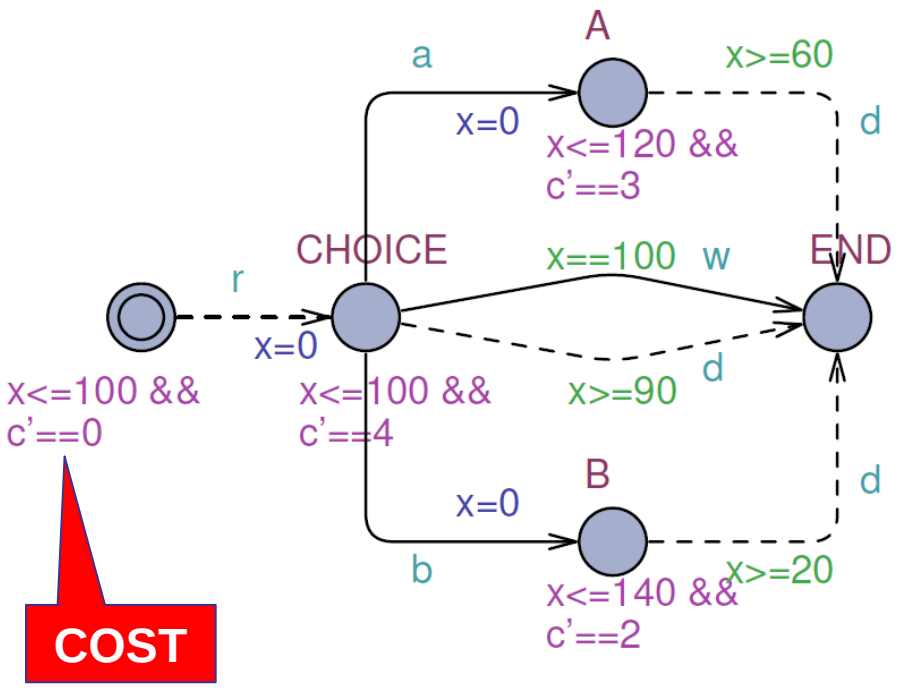
Most permissive, memoryless strategy

Objective: $A\langle\rangle$ **(END** $\wedge$ **time** ≤ 210**)**

Deterministic, memoryless strategy:



Priced Timed Games "CORA"



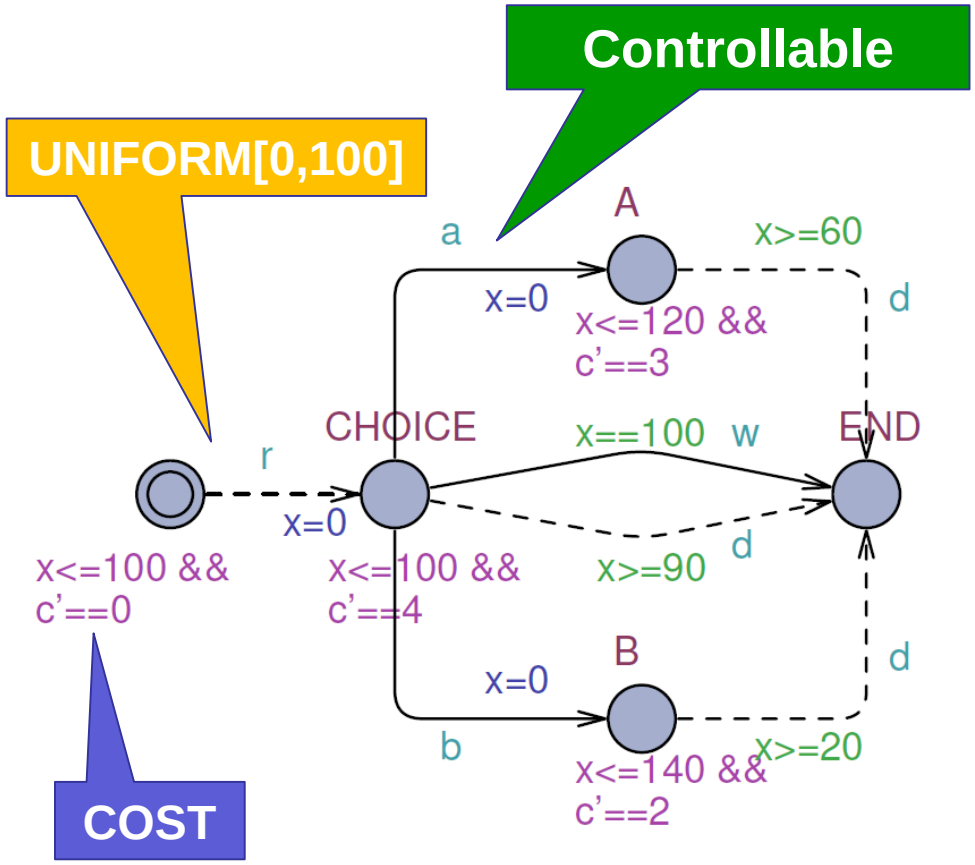
- Cost optimal strategy
  - take b immediately
    WC= 280

Priced Run

$$\pi = (\text{Init}, x = 0) \xrightarrow[0]{50.1} \xrightarrow{r} (\text{CHOICE}, x = 0) \xrightarrow[9.6]{2.4} \xrightarrow{a} (\text{A}, x = 0) \xrightarrow[60.9]{20.3} \xrightarrow{d} (\text{END}, x = 20.3)$$

$$\text{Total } cost = 0 + 9.6 + 60.9 = 75.5$$

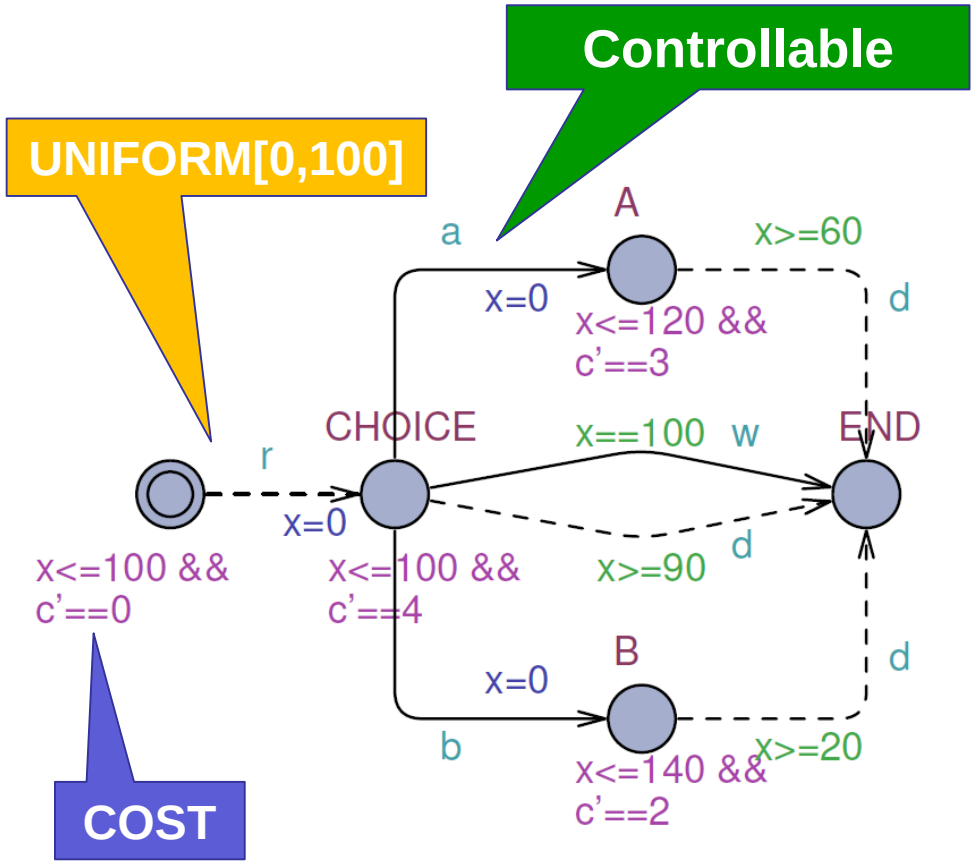
Priced Timed MDP "SMC"



- Cost optimal strategy
  - take b immediately
    WC= 280

- Priced Timed MDP
- Optimal expected cost str
  - take b immediately
    expectation = 160

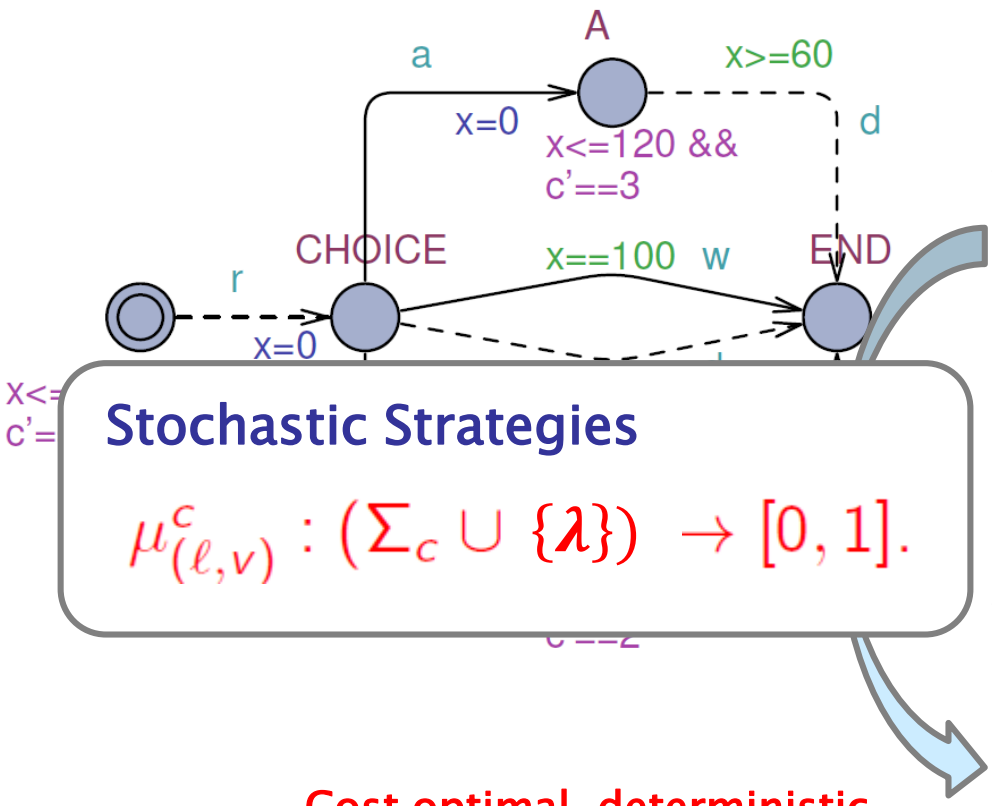
Priced Timed MDP "SMC"



- Cost optimal strategy
  - take b immediately
    overall = 280
- Priced Timed MDP
- Optimal expected cost str
  - take b immediately
    expectation = 160
- Minimal Expected Cost while guaranteeing END is reached within time 210:

  Strat.: t>90→ (100,w)
  t>70→ (0,a)
  t<70→ (0,b)
  = **204**

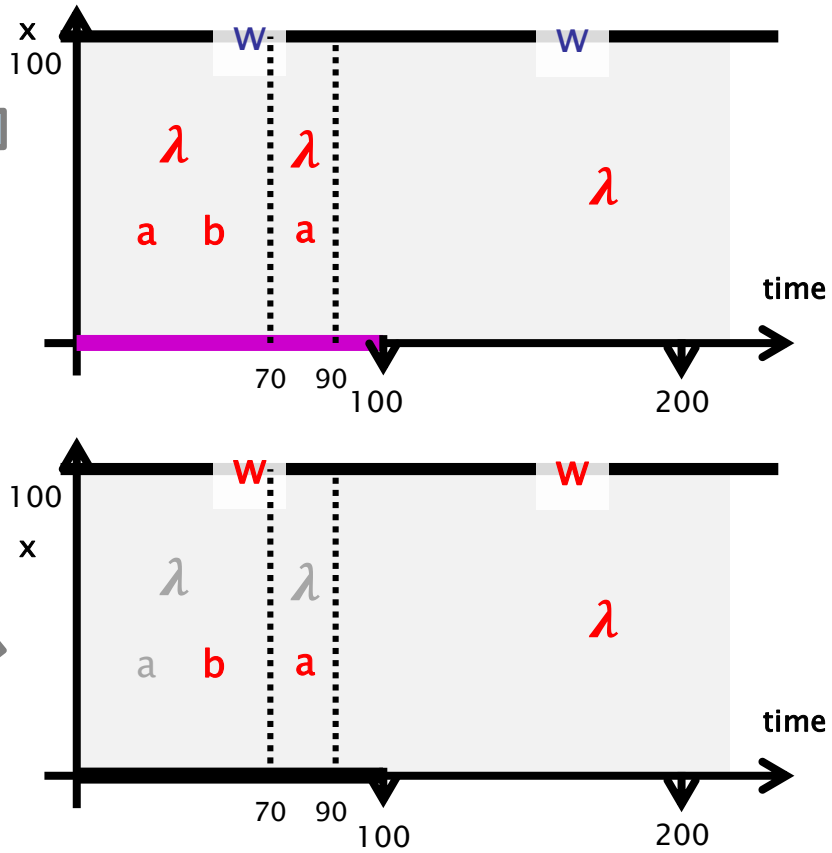
Stochastic Strategies for Learning!



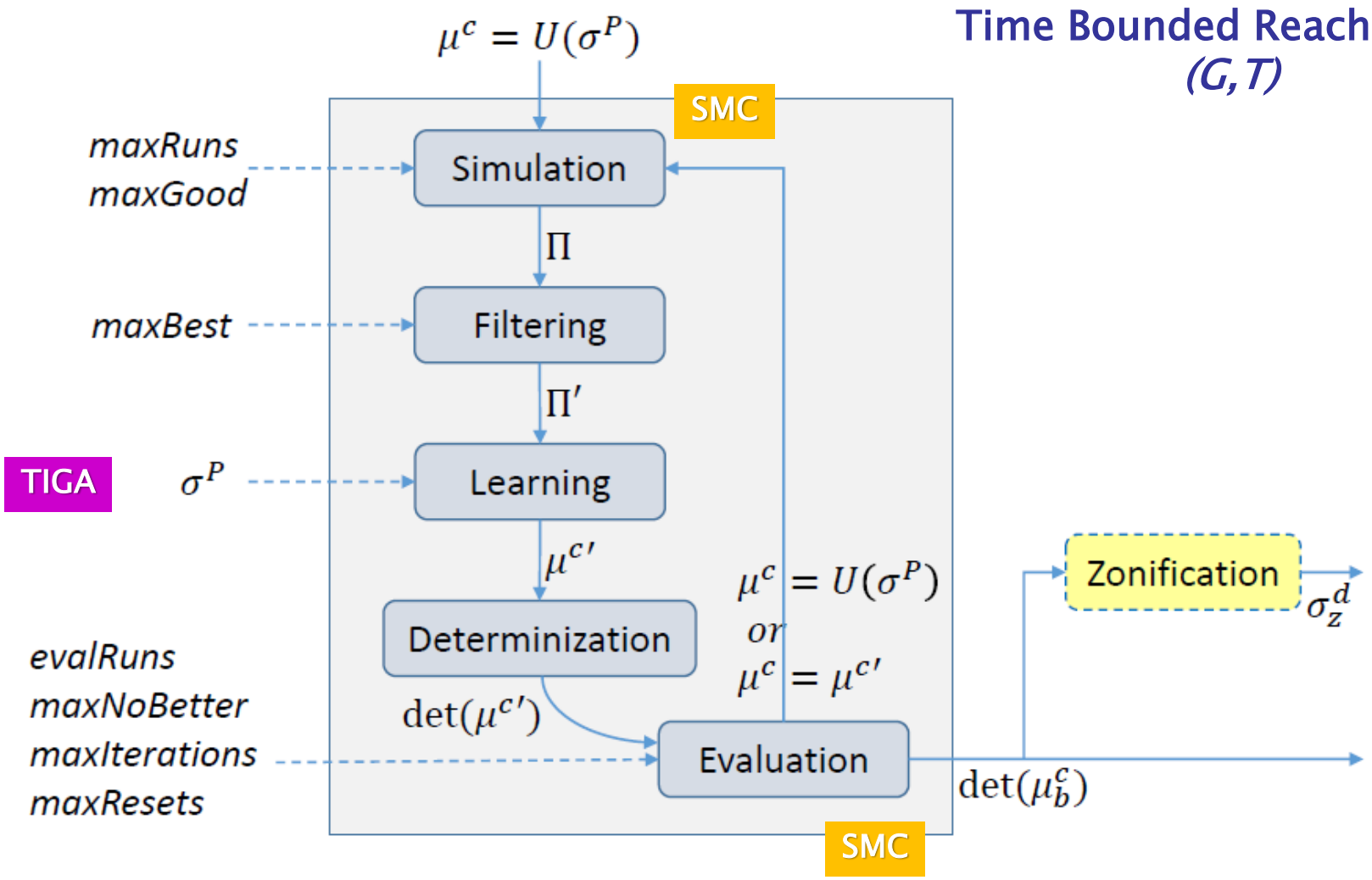
Cost optimal deterministic sub-strategy !

Objective: $A\langle\rangle$ **(END $\wedge$ time ≤ 210)**

Most permissive, memoryless strategy:



Reinforcement Learning



Strategies – Representation

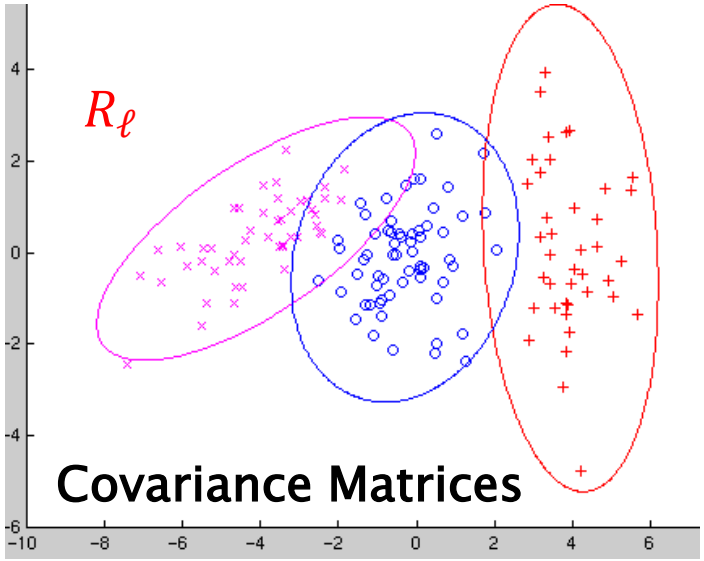
Nondeterministic Strategies

$$\sigma^{n}_{(\ell,v)} \subseteq (\Sigma_c \cup \{\lambda\})$$

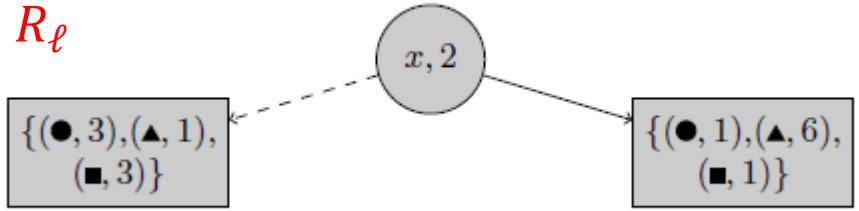
$$R_\ell = \{(Z_1, a_1), \ldots, (Z_k, a_k)\}, \text{ where } a_i \in \Sigma_c \cup \{\lambda\}.$$

Stochastic Strategies

$$\mu^{s}_{(\ell,v)} : (\Sigma_c \cup \{\lambda\}) \to [0,1]$$



Covariance Matrices

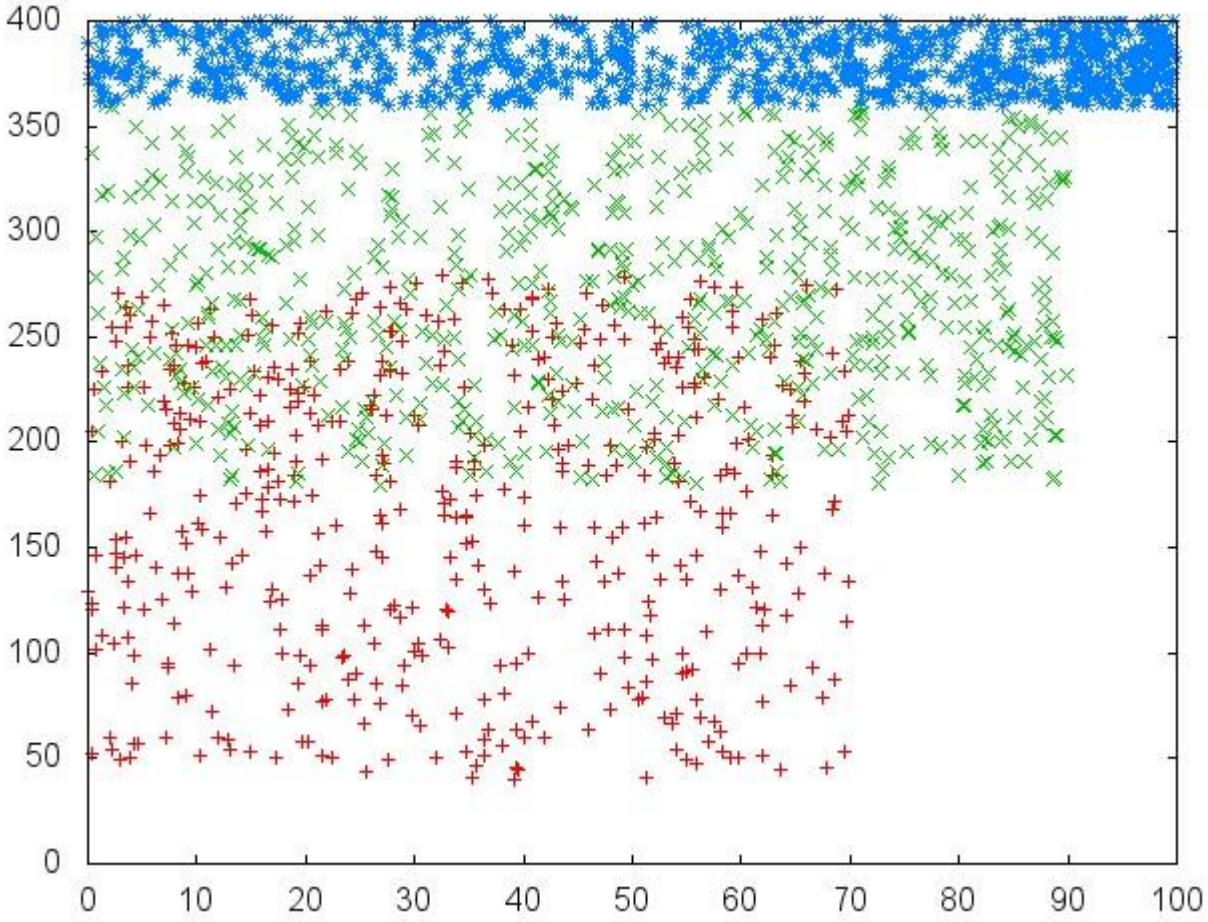


Splitting

Logistic Regression

Learned Strategies

Covariance Matrices



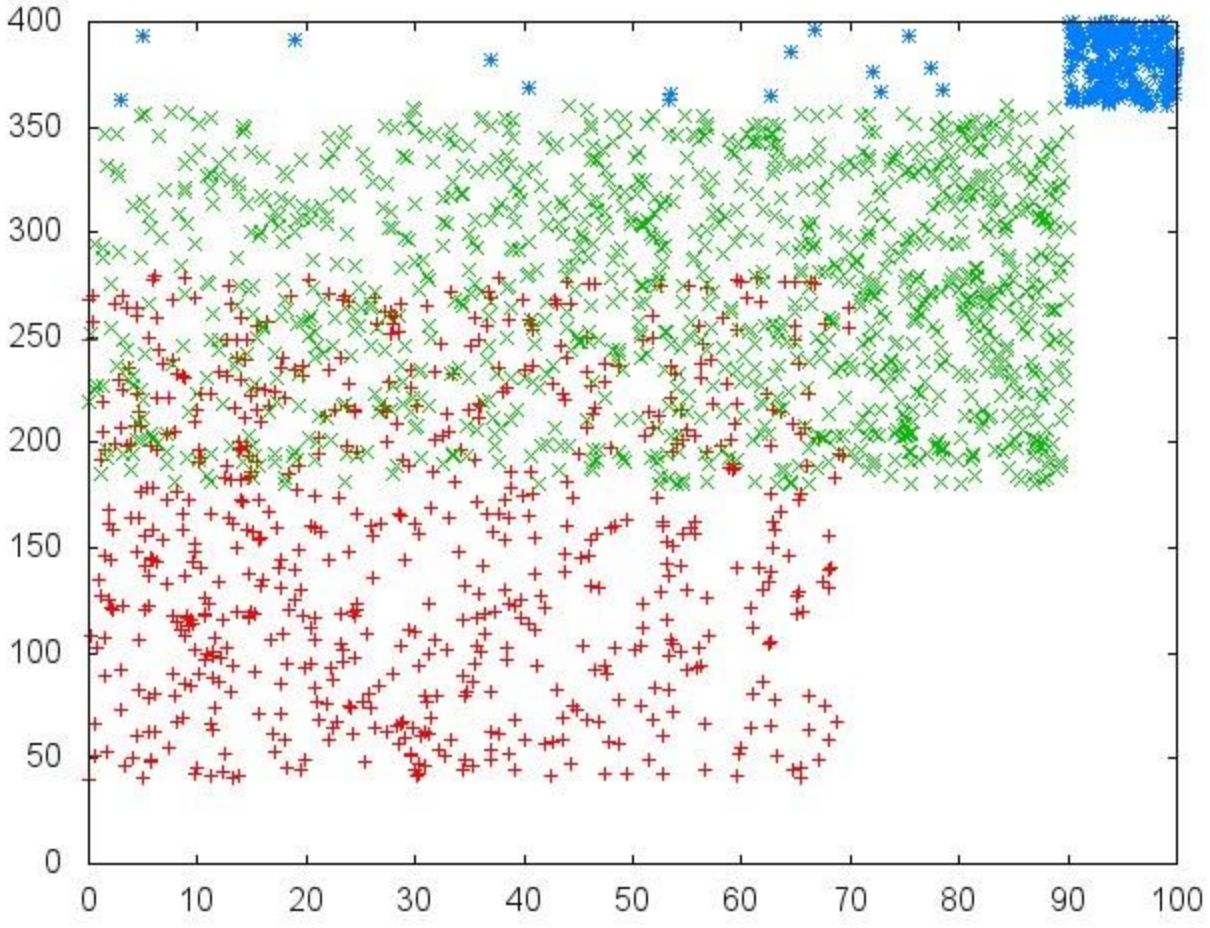
λ

a

b

Learned Strategies

Covariance Matrices



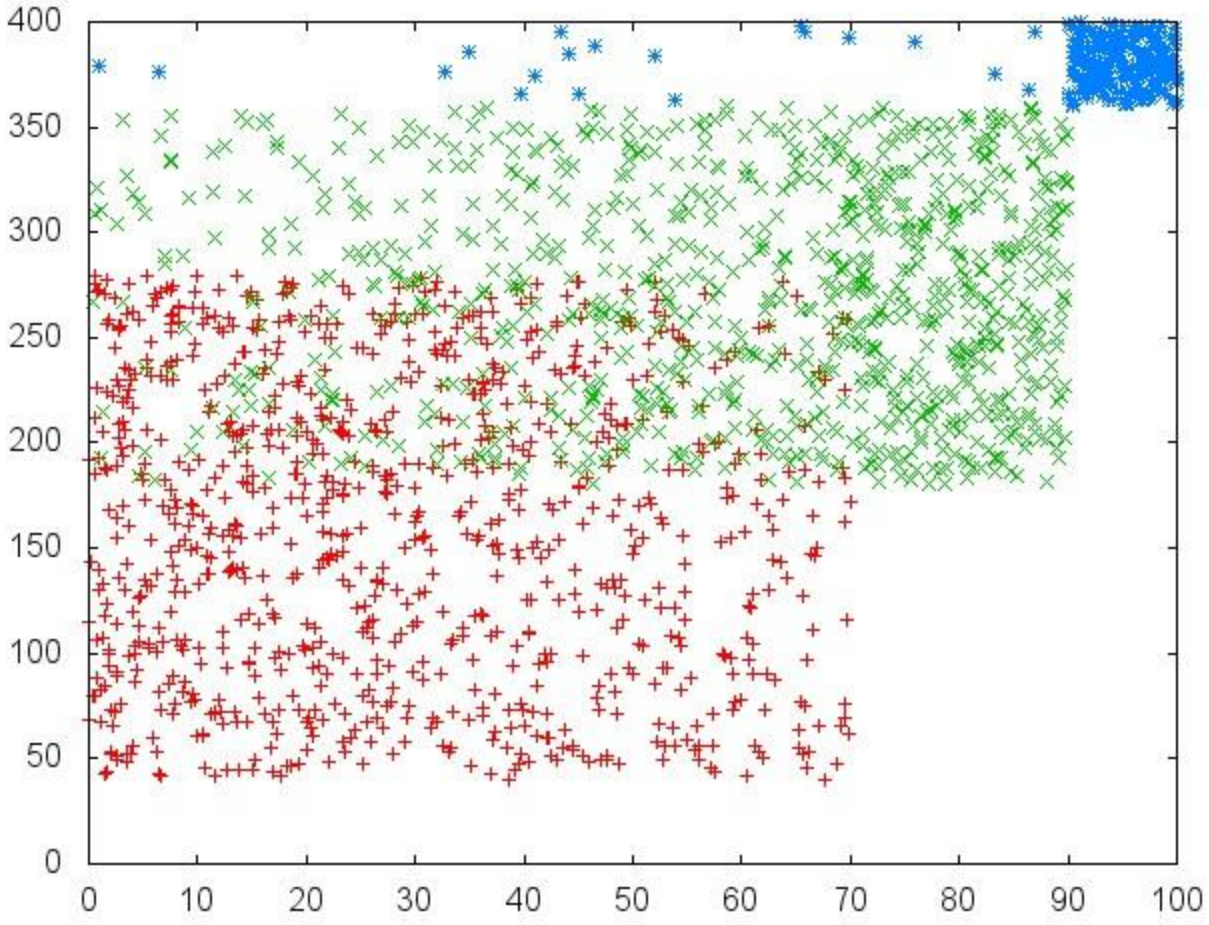
λ

a

b

Learned Strategies

Covariance Matrices



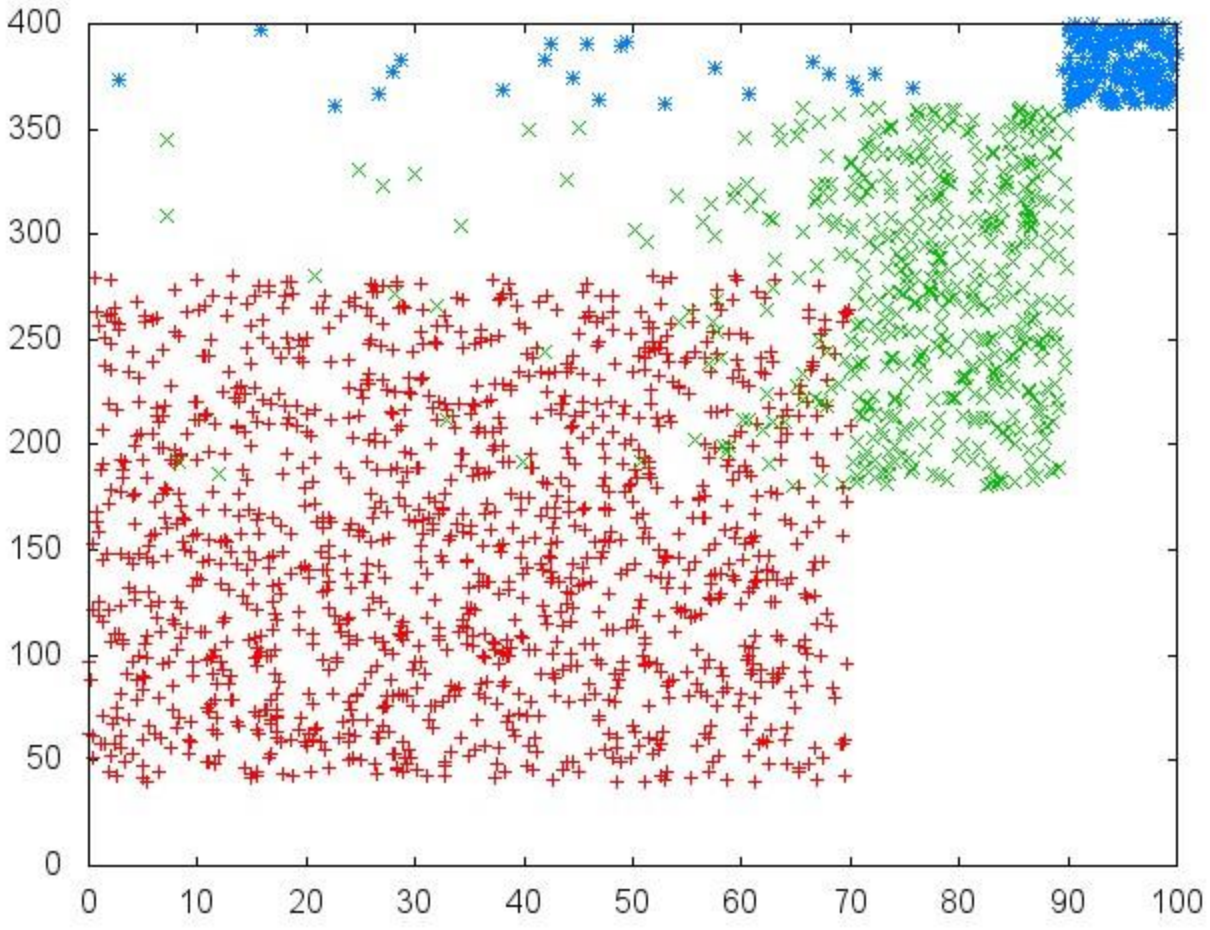
λ

a

b

Learned Strategies

Covariance Matrices



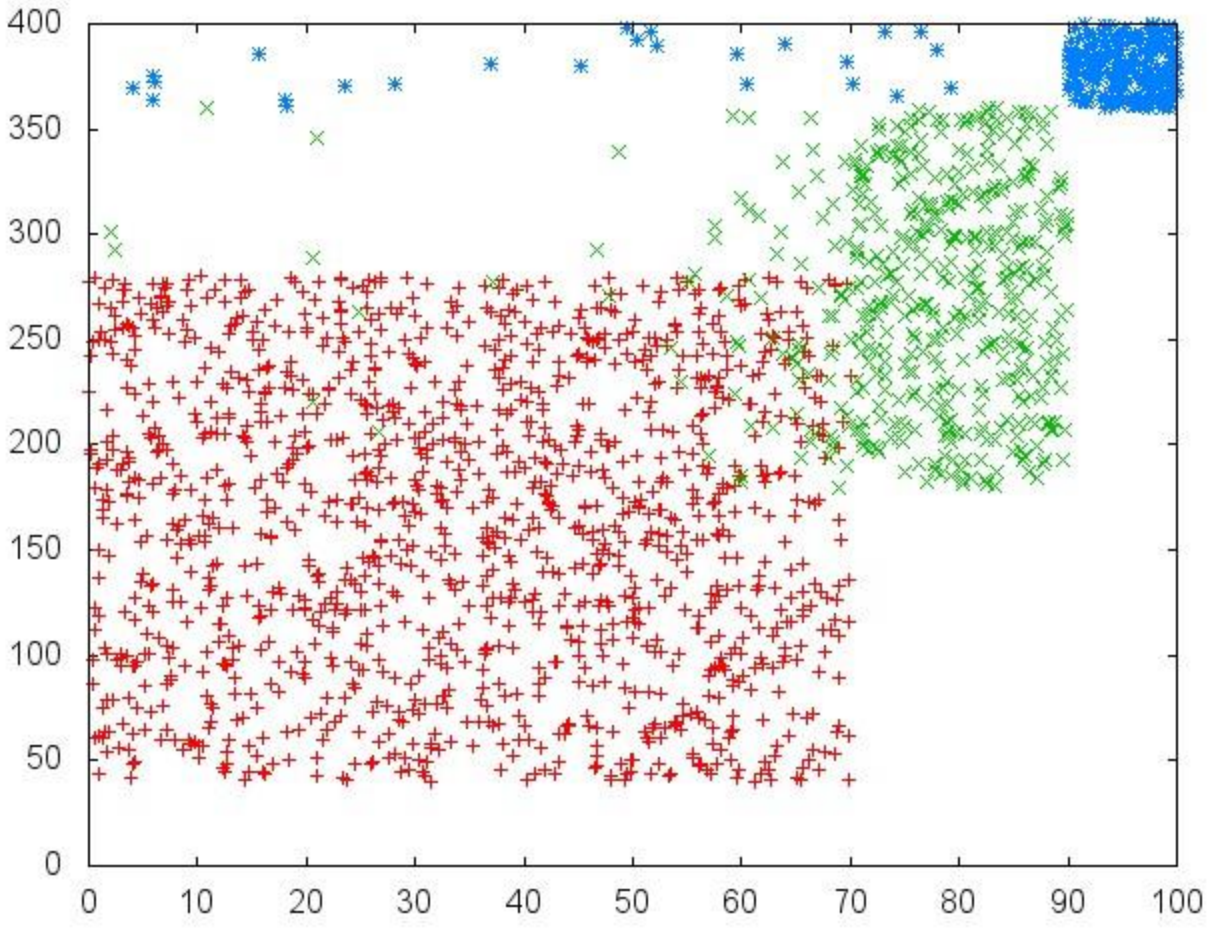
λ

a

b

Learned Strategies

Covariance Matrices



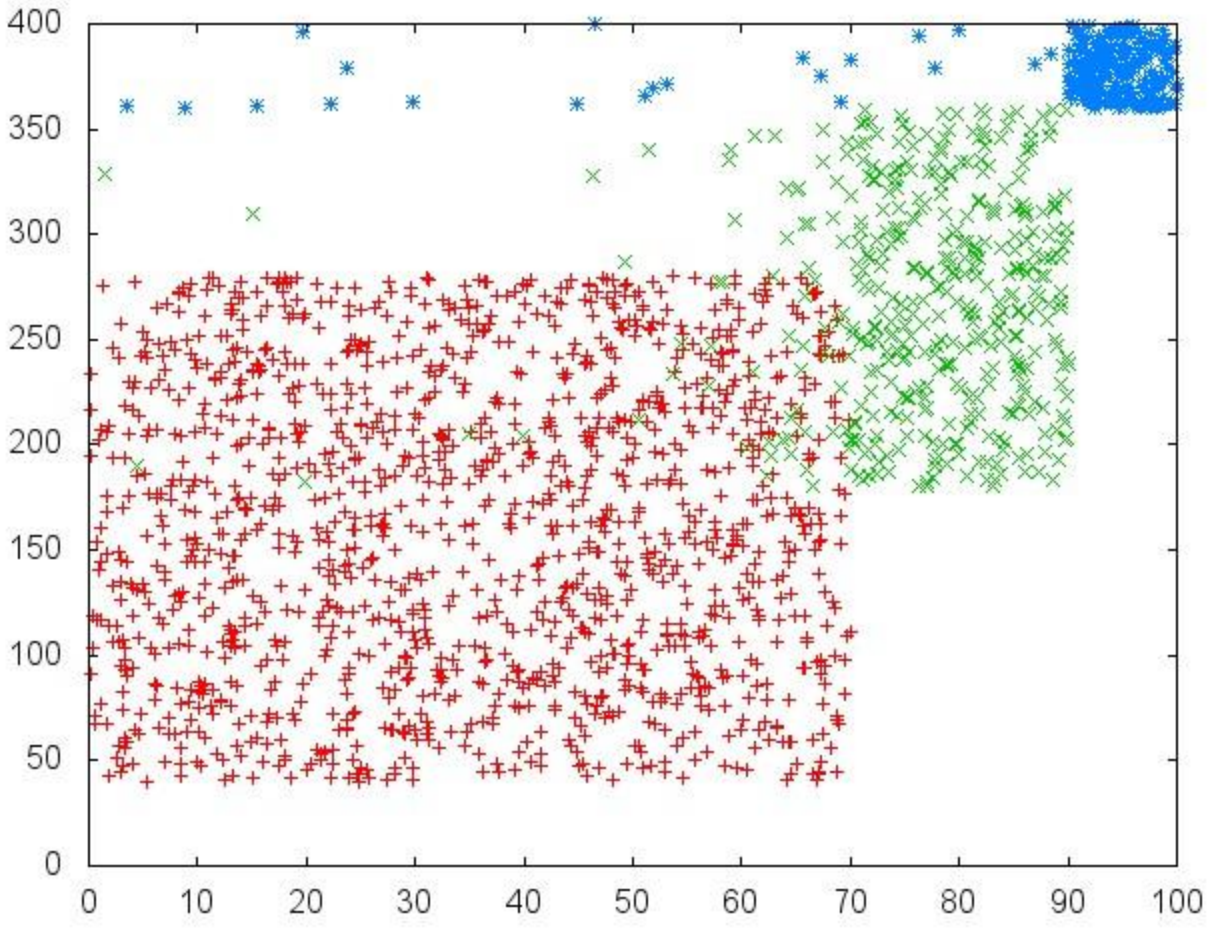
λ

a

b

Learned Strategies

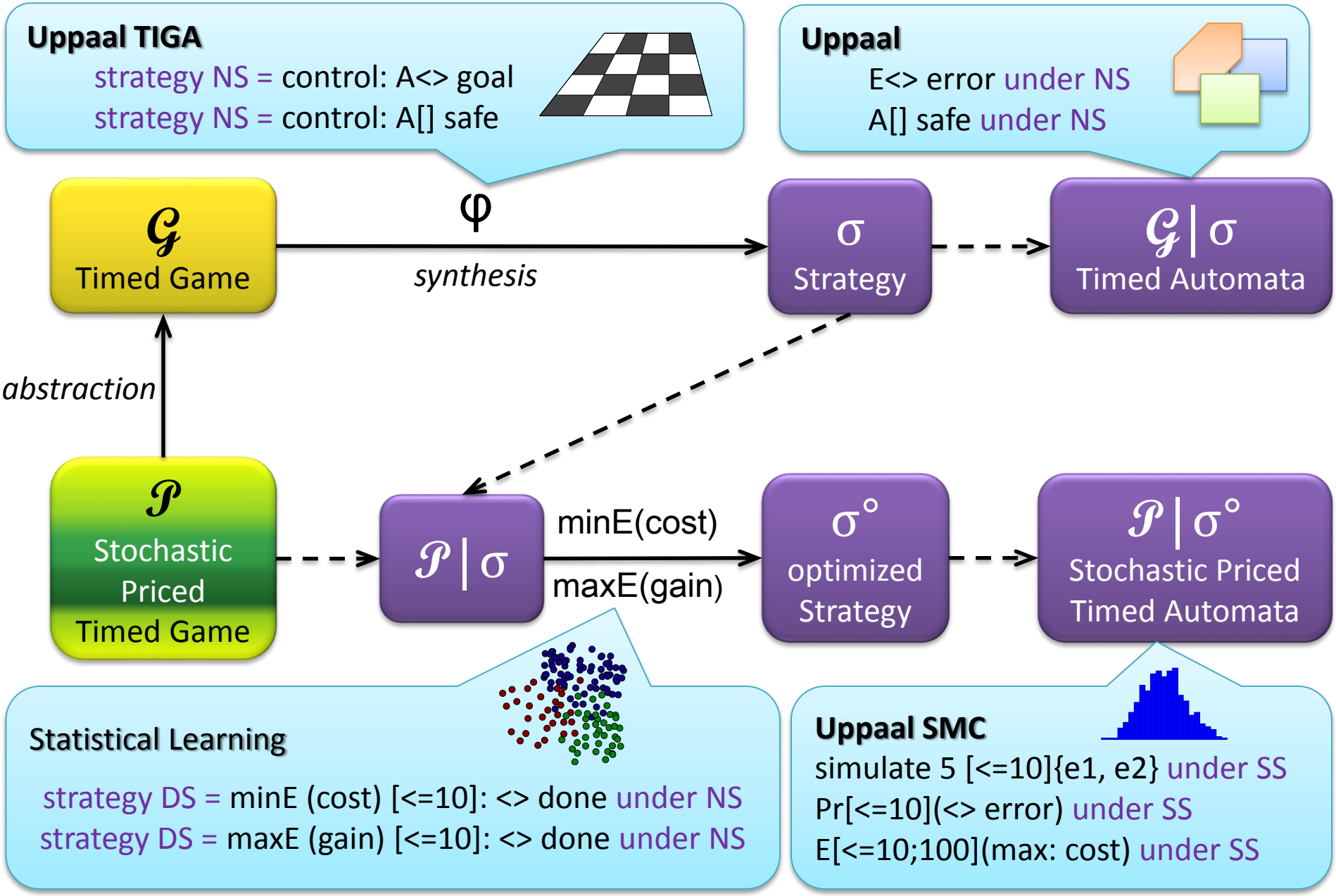
Covariance Matrices



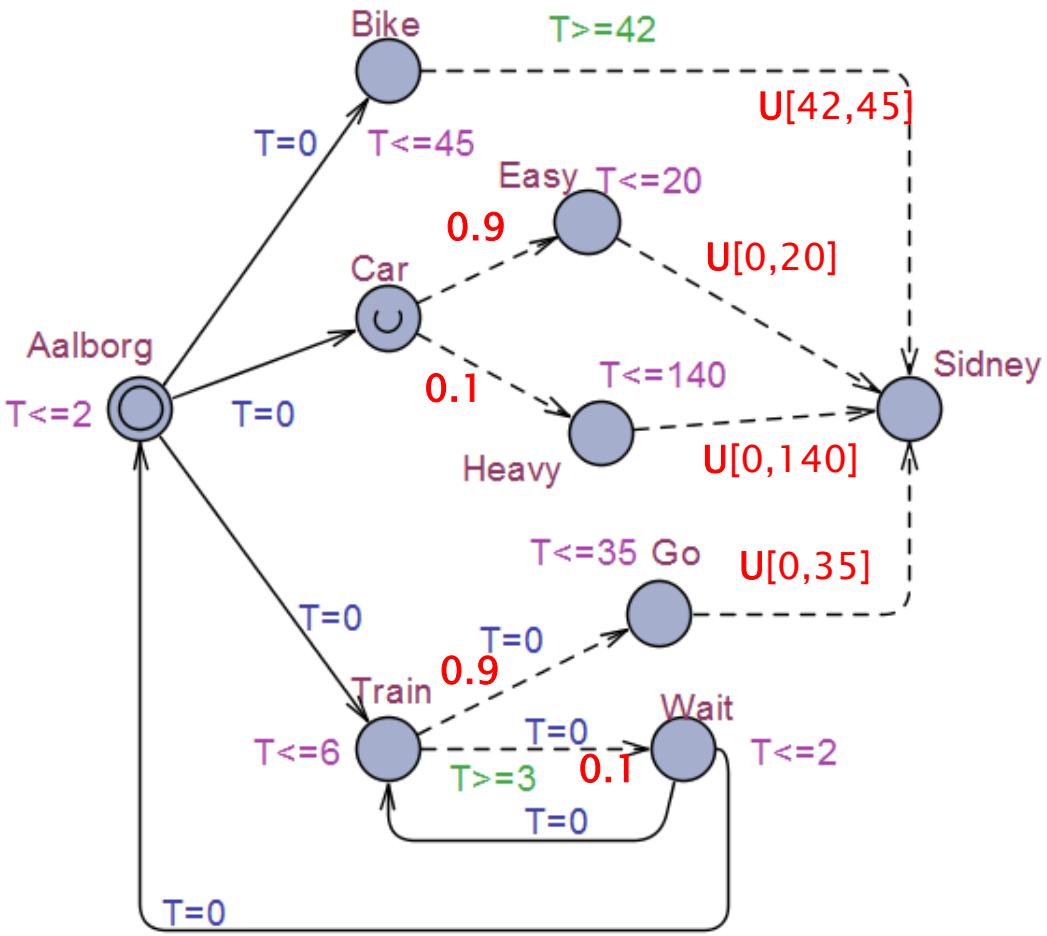
λ

a

b


Uppaal TIGA
strategy NS = control: A<> goal
strategy NS = control: A[] safe
Uppaal
E<> error under NS
A[] safe under NS
$\mathcal{G}$
Timed Game
φ
synthesis
σ
Strategy
$\mathcal{G} \mid \sigma$
Timed Automata
abstraction
$\mathcal{P}$
Stochastic
Priced
Timed Game
$\mathcal{P} \mid \sigma$
minE(cost)
maxE(gain)
σ°
optimized
Strategy
$\mathcal{P} \mid \sigma^\circ$
Stochastic Priced
Timed Automata
Statistical Learning
strategy DS = minE (cost) [<=10]: <> done under NS
strategy DS = maxE (gain) [<=10]: <> done under NS
Uppaal SMC
simulate 5 [<=10]{e1, e2} under SS
Pr[<=10](<> error) under SS
E[<=10;100](max: cost) under SS

Going to Sydney – in 1 hour



Can I get to Sidney?
(1-player)

Will I always come to Sidney?
(1-player)

What is the optimal WC strategy?
(2-player)

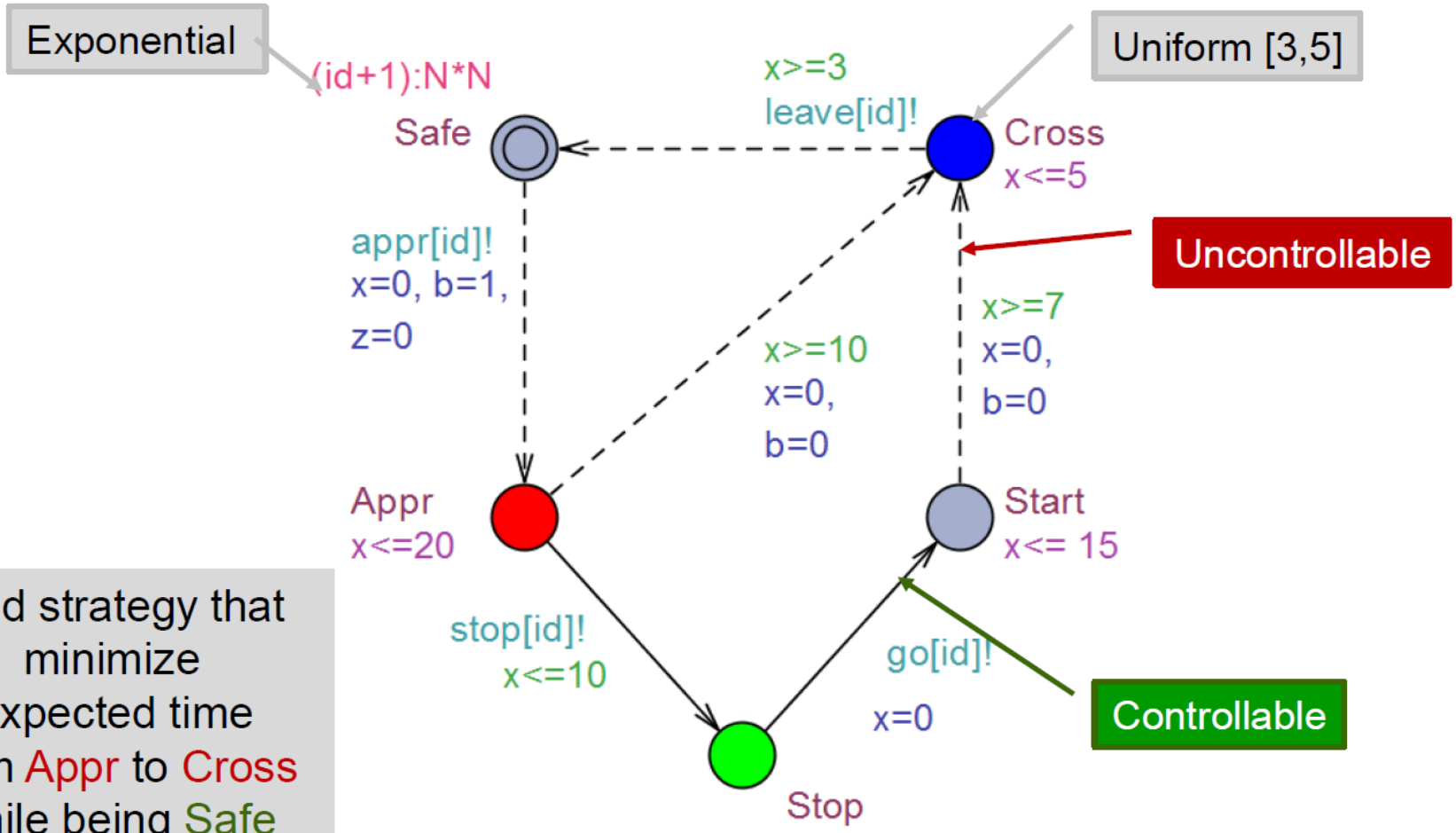
Is there a strategy guaranteeing WC <= 60?
(2-player)

What is the optimal strategy?
(1½-player)

What is the optimal strategy Guarenteeing WC <= 60?
(1½-player)

DEMO

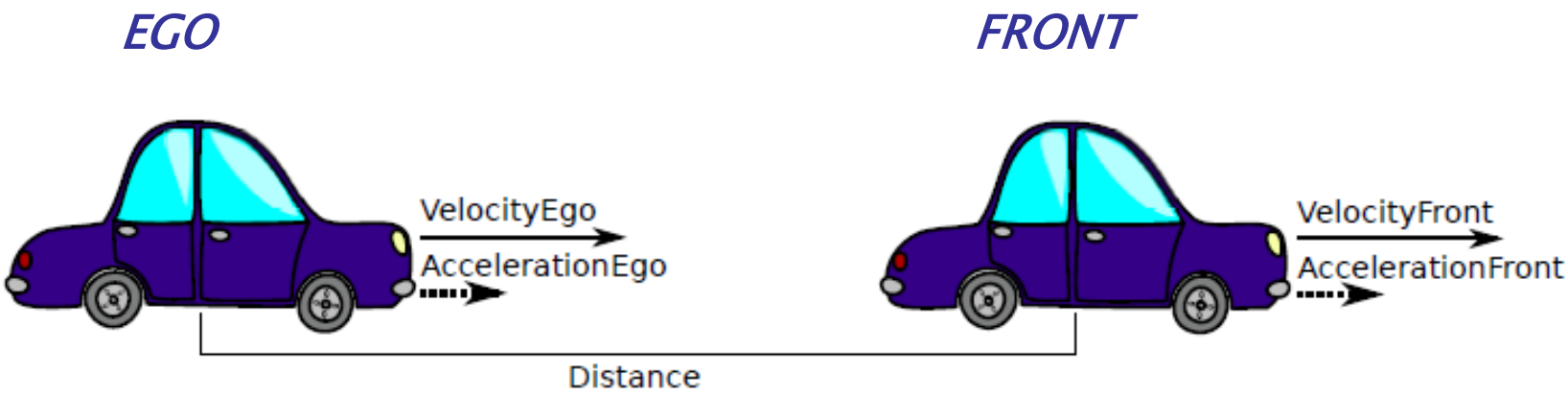
Safe and Optimal Train Gate



Find strategy that minimize Expected time from Appr to Cross while being Safe

DEMO

Safe & Adaptive Cruice Control



Q1: Find a safety **strategy** for *Ego* such no crash will ever occur no matter what *Front* is doing.

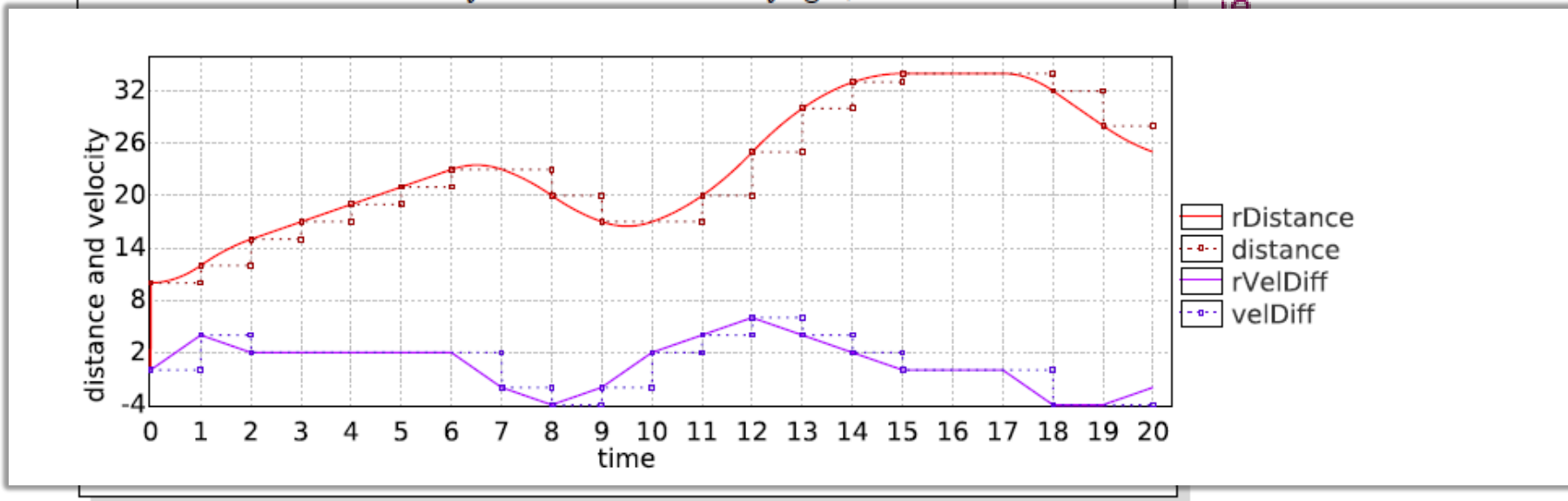
Q2: Find the **most permissive strategy** ensuring safety

Q3: Find the **optimal sub-strategy** that will allow *Ego* to go as far as possible (without overtaking).

Discretization

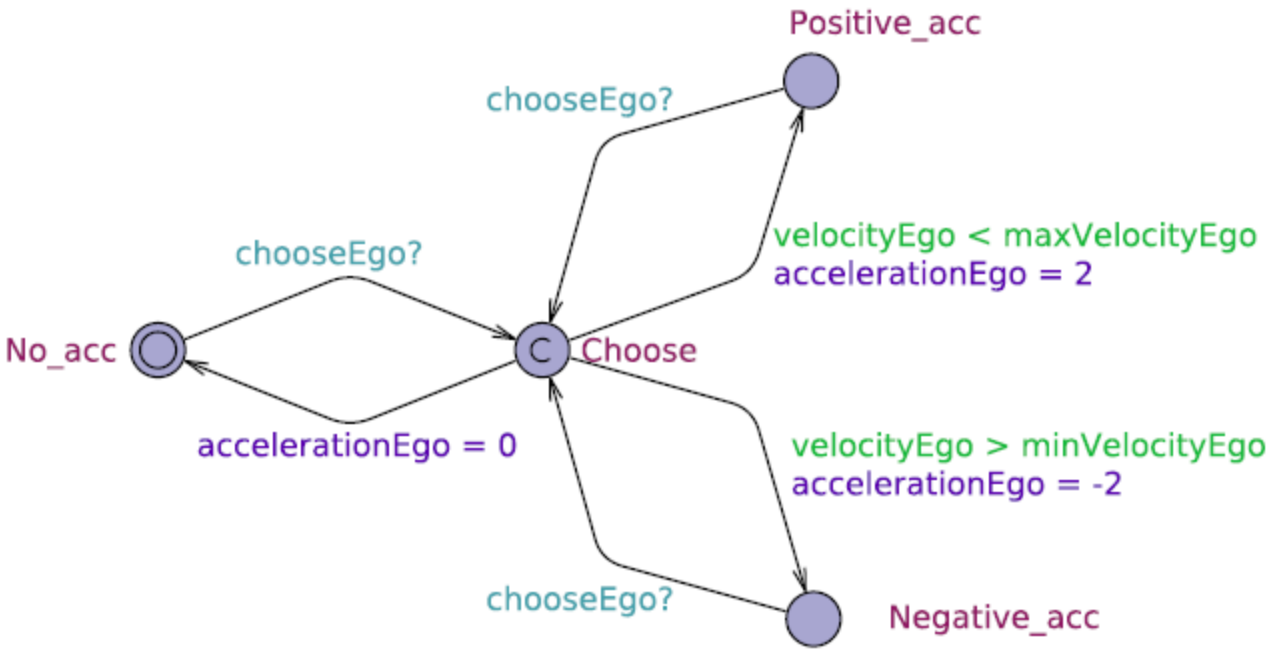
Discrete

```
void updateDiscrete(){
    int oldVel, newVel;
    oldVel = velocityFront - velocityEgo;
    velocityEgo = velocityEgo + accelerationEgo;
    velocityFront = velocityFront + accelerationFront;
    newVel = velocityFront - velocityEgo;
```

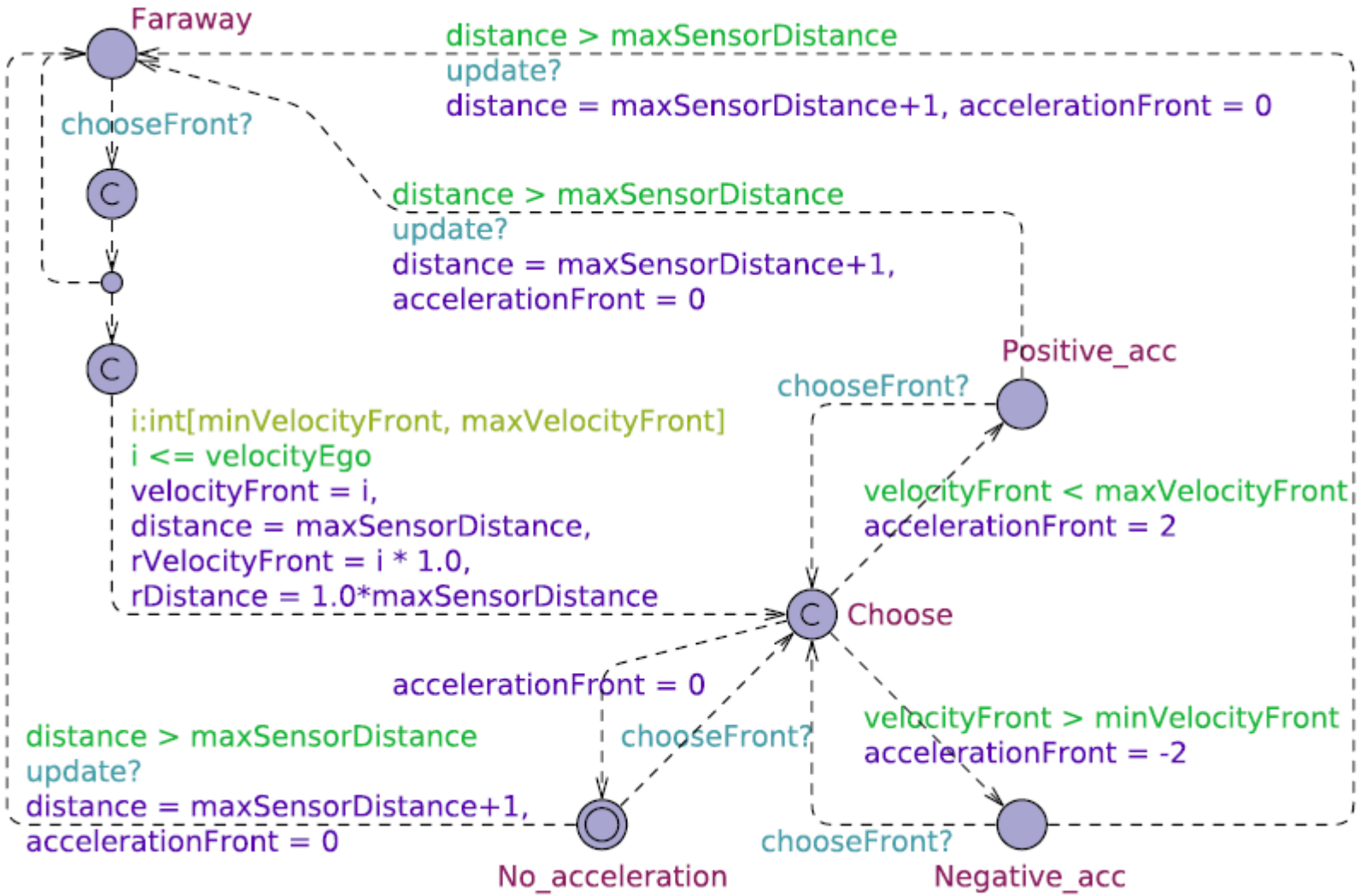


Continuous

Ego


Positive_acc
chooseEgo?
velocityEgo < maxVelocityEgo
accelerationEgo = 2
chooseEgo?
No_acc
C
Choose
accelerationEgo = 0
velocityEgo > minVelocityEgo
accelerationEgo = -2
chooseEgo?
Negative_acc

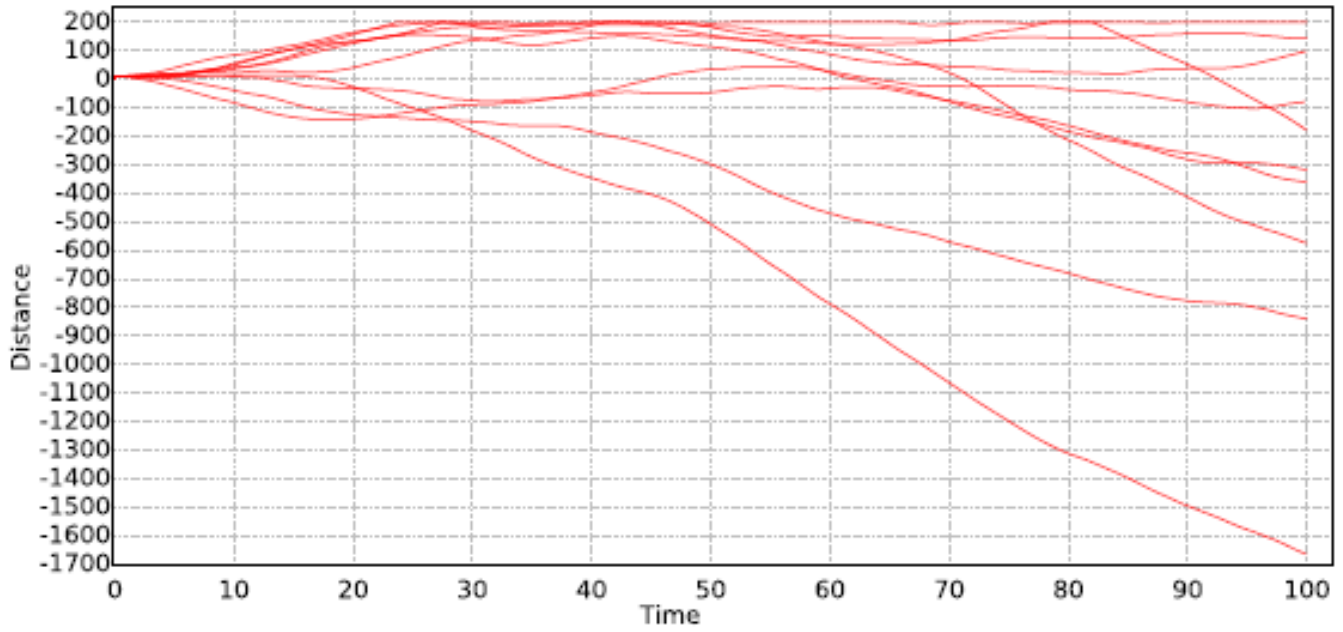
Front


Faraway
distance > maxSensorDistance
update?
distance = maxSensorDistance+1, accelerationFront = 0
chooseFront?
distance > maxSensorDistance
update?
distance = maxSensorDistance+1,
accelerationFront = 0
Positive_acc
chooseFront?
i:int[minVelocityFront, maxVelocityFront]
i <= velocityEgo
velocityFront = i,
distance = maxSensorDistance,
rVelocityFront = i * 1.0,
rDistance = 1.0*maxSensorDistance
velocityFront < maxVelocityFront
accelerationFront = 2
Choose
accelerationFront = 0
chooseFront?
velocityFront > minVelocityFront
accelerationFront = -2
distance > maxSensorDistance
update?
distance = maxSensorDistance+1,
accelerationFront = 0
chooseFront?
No_acceleration
Negative_acc

No Strategy

```
Pr[<=100] (<> distance <= 5)
```

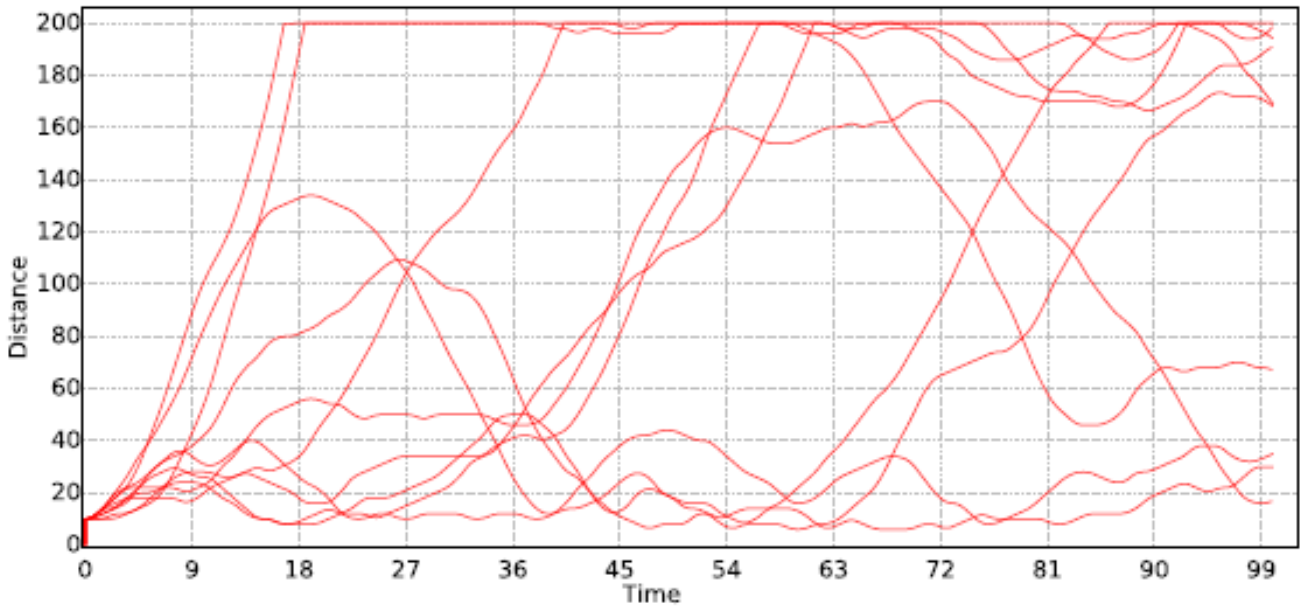
```
A[] distance > 5
```



Safety Strategy

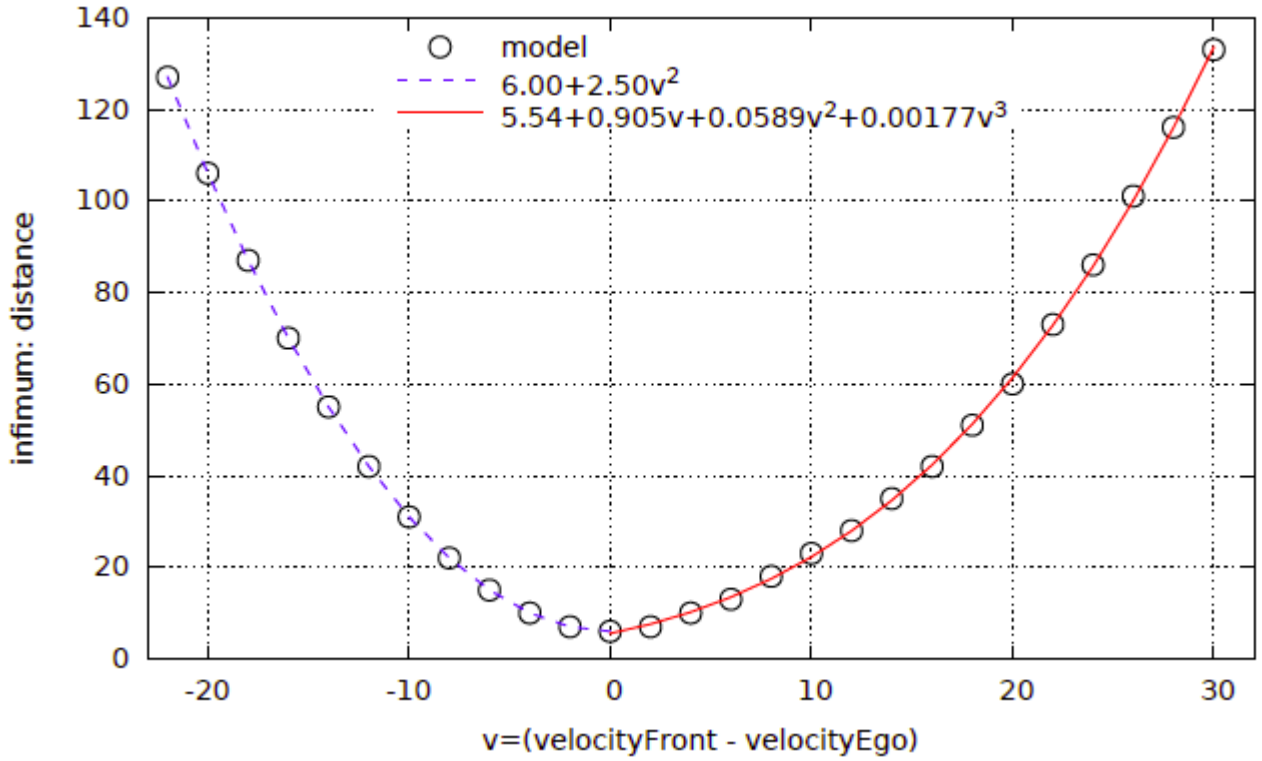
```
strategy safe = control: A[] distance > 5
```

```
A[] distance > 5 under safe
```



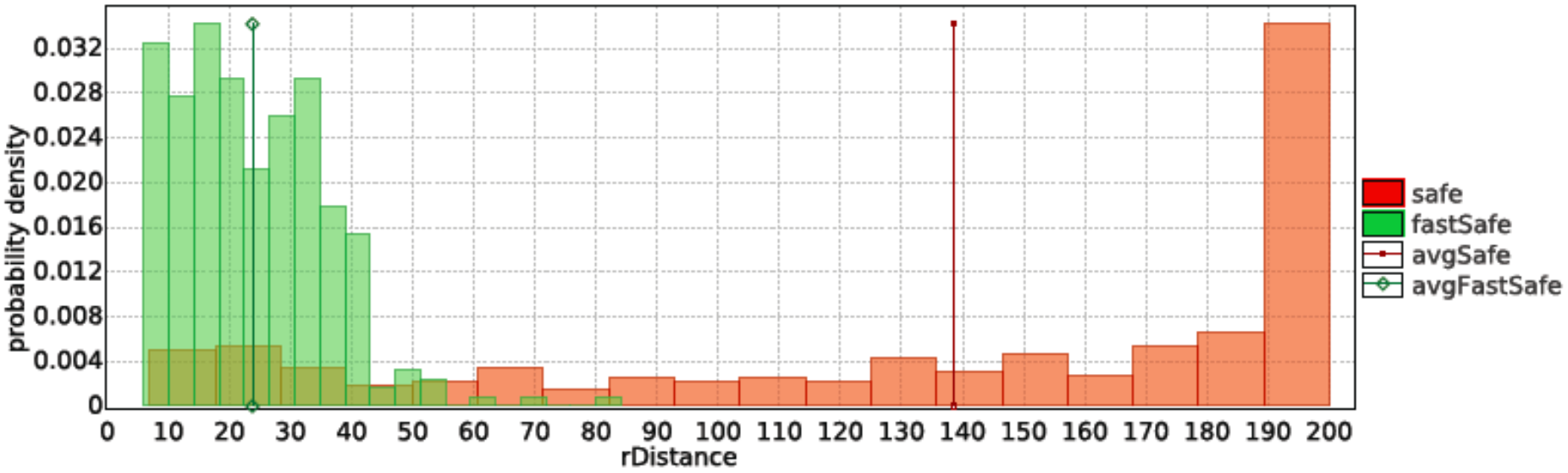
Safety Strategy

```
inf{velosityFront-velosityEgo==v}: distance under safe
```

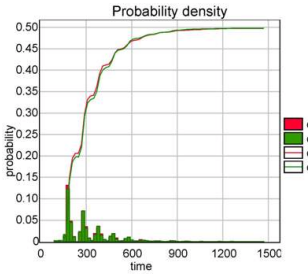


Optimal and Safe Strategy

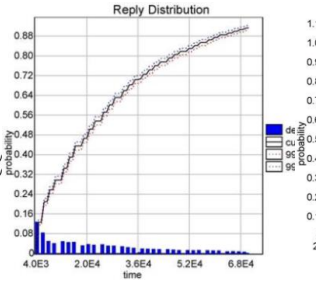
```
strategy safeFast = minE (D) [<=100]: <> time >= 100 under safe
```

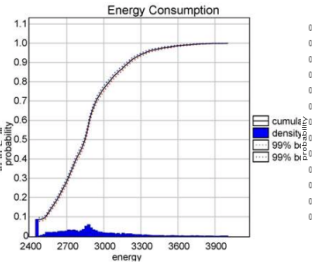


Other Case Studies

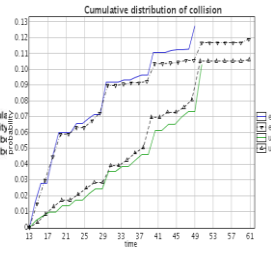


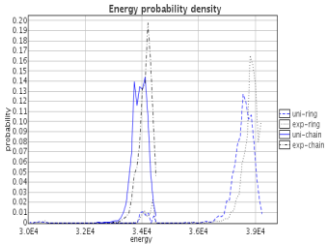
FIREWIRE



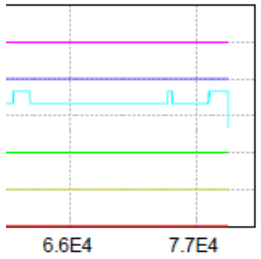


BLUETOOTH

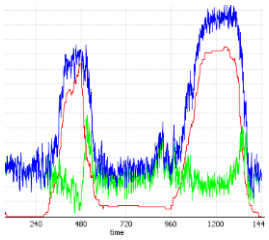




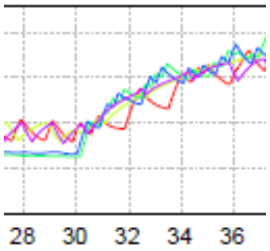
10 node LMAC



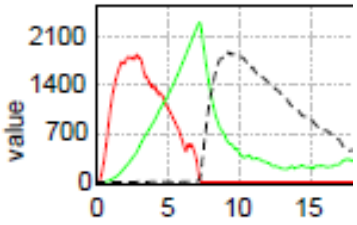
Schedulability Analysis for Mix Cr Sys



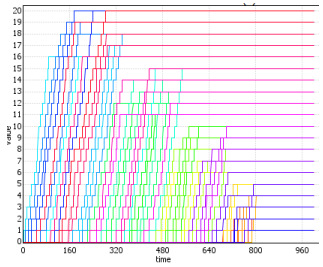
Smart Grid Demand / Response



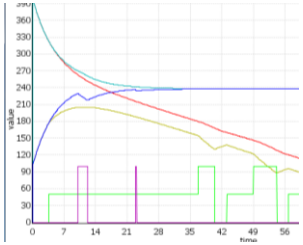
Energy Aware Buildings



Genetic Oscilator (HBS)



Passenger Seating in Aircraft



Battery Scheduling

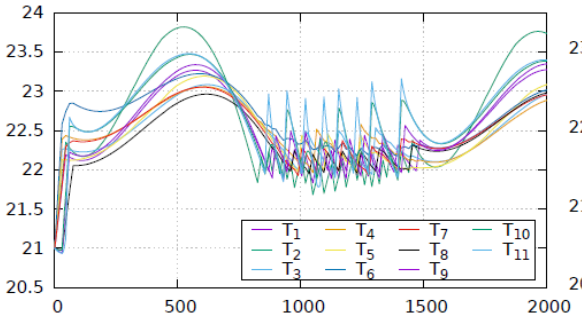
Floorheating

Cassting

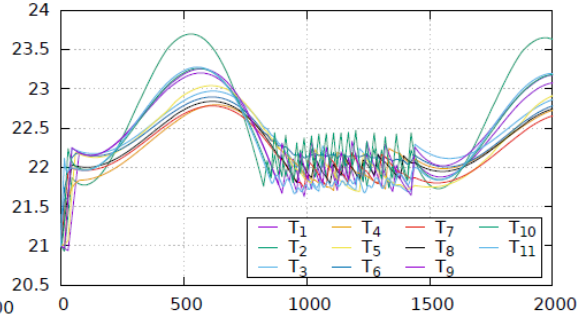
seluxit

Thursday Afternoon
Daniel Lux, Seluxit
Marco Muniz, AAU

Bang-Bang Controller



STRATEGO-ON-CL Controller



DiCyPS

CENTER FOR DATA-INTENSIVE CYBER-PHYSICAL SYSTEMS

2015-2021, 70MMDKK
Innovation Fund DK

FLEXDANMARK

Danfoss

RAMBØLL

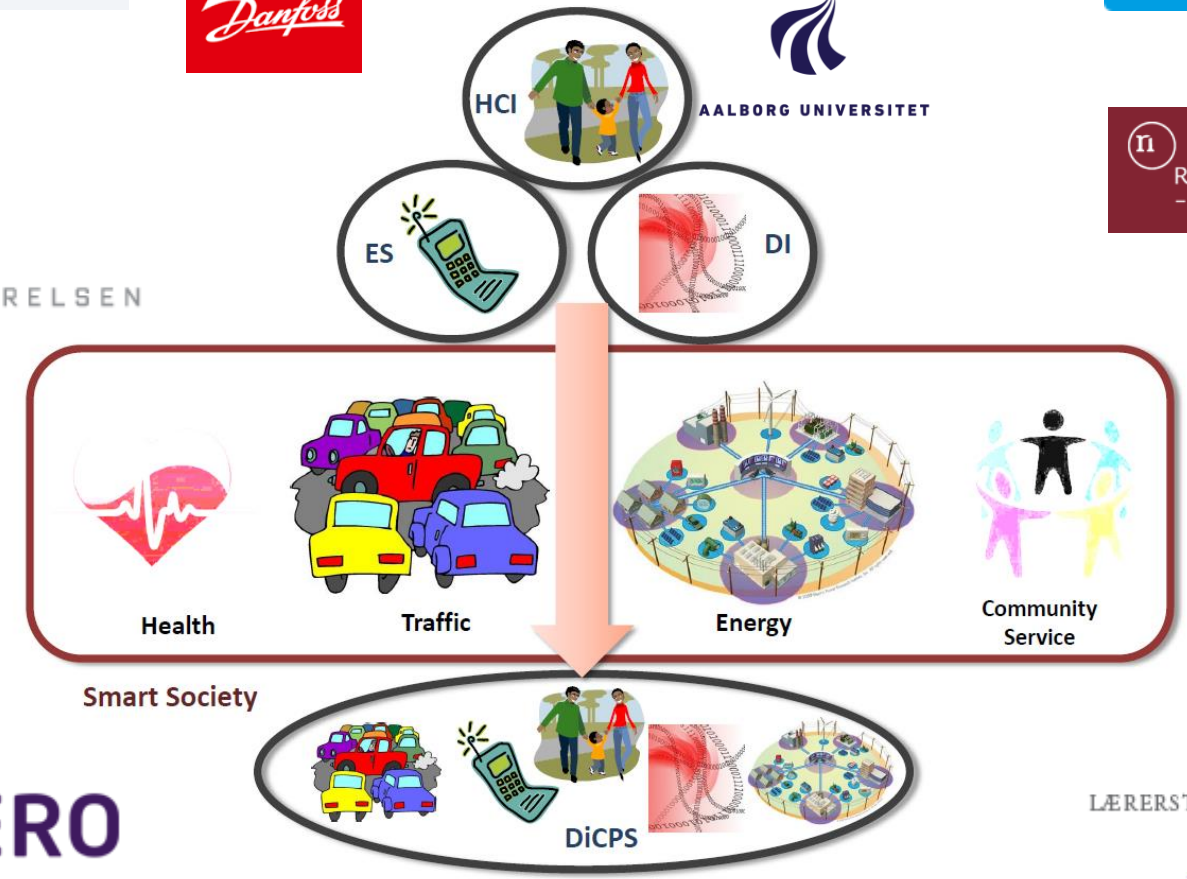
BLIP systems

NYFORS

AALBORG UNIVERSITET

REGION NORDJYLLAND
– i gode hænder

ENERGI STYRELSEN



AALBORG KOMMUNE

REJSEKORT

INSERO

LÆRERSTANDENS BRANDFORSIKRING

SCADA international

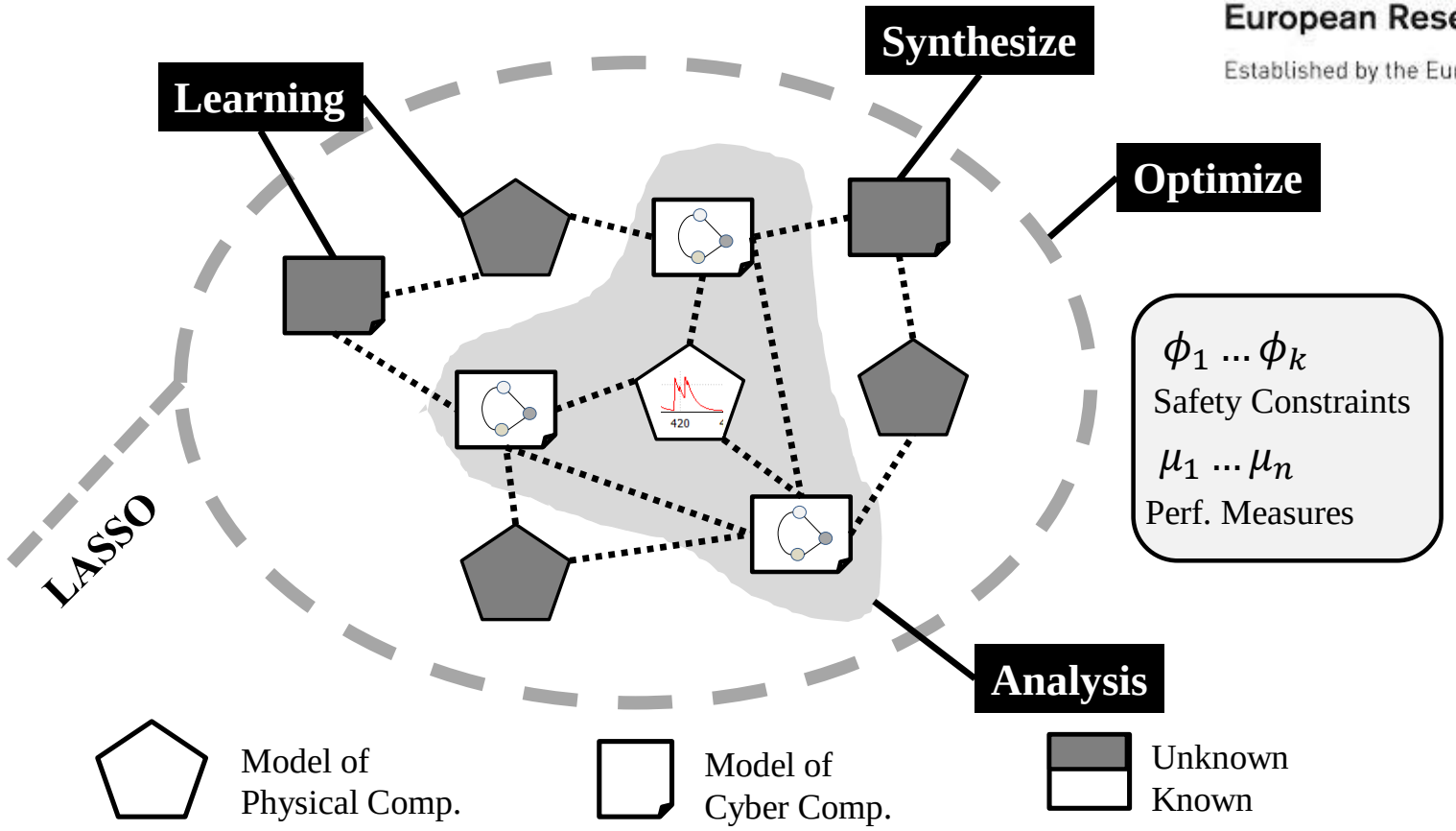
TU Dresden INRIA U College London

seluxit

LASSO

Learning, Analysis, SynthesiS and Optimization of Cyber-Physical Systems

European Research Council
Established by the European Commission



Contact: kgl@cs.aau.dk

Applications

Case Studies: Controllers

- **Memory Arbiter Synthesis and Verification for a Radar Memory Interface Card, 2005**
- Analyzing a χ model of a turntable system using Spin, CADP and Uppaal, 2006
- Designing, Modelling and Verifying a Container Terminal System Using UPPAAL, 2008
- Model-based system analysis using Chi and Uppaal: An industrial case study, 2008
- **Climate Controller for Pig Stables, 2008 (synth)**
- **Optimal and Robust Controller for Hydralic Pump, 2009 (synth)**

References

- Frits Vaandrager: A first introduction to UPPAAL
- Alexandre David, Kim G Larsen: More features in UPPAAL
- Alexandre David, Kim G Larsen, Axel Legay, Marius Mikucionis, Danny Bøgsted Poulsen: UPPAAL SMC Tutorial. To appear in Software Tools for Technology Transfer.
- Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, Zheng Wang: Time for Statistical Model Checking of Real-Time Systems. CAV 2011.
- Gerd Behrmann, Agnes Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime: UPPAAL-Tiga: Time for Playing Games! CAV 2007.
- Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikucionis, Jakob Haahr Taankvist: Uppaal Stratego. TACAS 2015

- For more see

http://people.cs.aau.dk/~kgl/SSFT2015/

Case Studies: Protocols

- Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal, 2006
- Formalizing SHIM6, a Proposed Internet Standard in UPPAAL, 2007
- Verifying the distributed real-time network protocol RTnet using Uppaal, 2007
- **Analysis of the Zeroconf protocol using UPPAAL, 2009**
- Analysis of a Clock Synchronization Protocol for Wireless Sensor Networks, 2009
- **Model Checking the FlexRay Physical Layer Protocol, 2010**

Using UPPAAL as Back-end

- Timed automata translator from Uppaal to PVS
- Component-Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008
- METAMOC: Modular WCET Analysis Using UPPAAL, 2010.
- TetaSARTS: a tool for modular timing analysis of safety critical Java systems, 2013

www.uppaal.org

RELATED SITES: TIMES | UPPAAL CORA | UPPAAL TRON

UPPAAL
Home

Home | About | Documentation | Download | Examples | Bugs

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

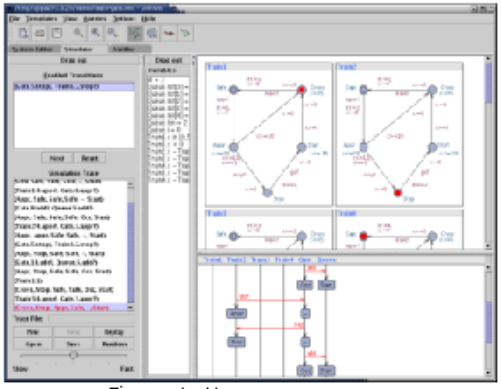


Figure 1: UPPAAL on screen.

Download

The current official release is UPPAAL 3.4.11 (Jun 23, 2005). A release of UPPAAL **3.6 alpha 3** (dec 20, 2005) is also available. For more information about UPPAAL version 3.4, we refer to this press release.

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

License

The UPPAAL tool is **free** for non-profit applications. For information about commercial licenses, please email sales(at)uppaal(dot)com.

To find out more about UPPAAL, read this short introduction. Further information may be found at this web site in the pages About, Documentation, Download, and Examples.

Mailing Lists

UPPAAL has an open discussion forum group at Yahoo!Groups intended for users of the tool. To join or post to the forum, please refer to the information at the discussion forum page. Bugs should be reported using the bug tracking system. To email the development team directly, please use uppaal(at)list(dot)it(dot)uu(dot)se.

www.uppaal.{org,com}