

Timed Games & Timed Interfaces

TIGA

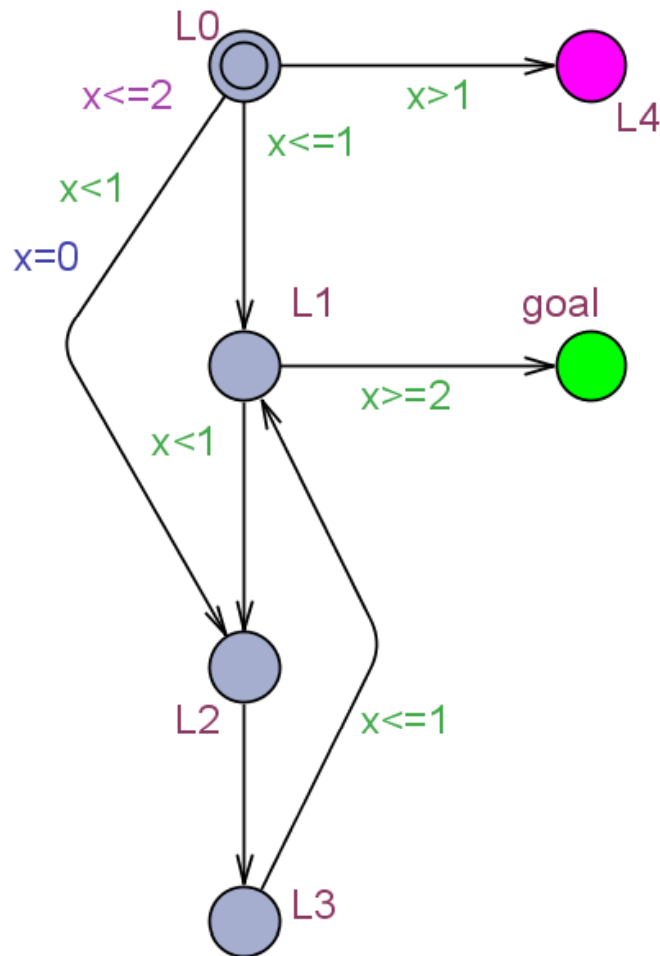
ECDAR

Kim G. Larsen

CISS – Aalborg University
DENMARK



Timed Automata & Model Checking

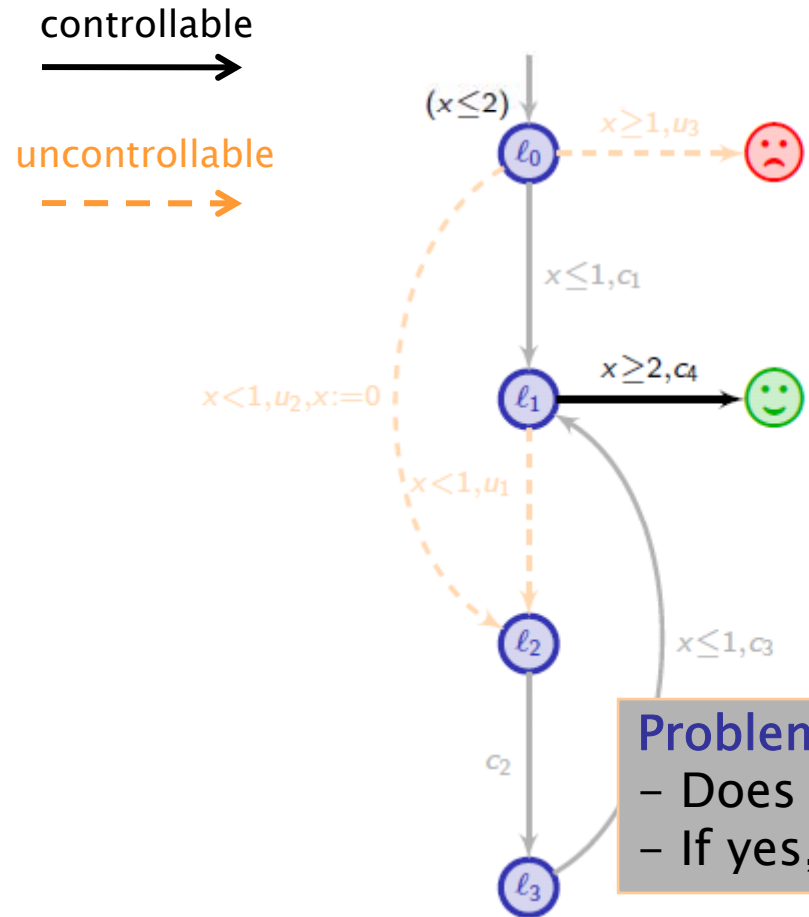


State (L1, x=0.81)
 Transitions
 (L1, x=0.81)
 - 2.1 ->
 (L1, x=2.91)
 ->
 (goal, x=2.91)

$E\langle \rangle$ goal ?
 $A\langle \rangle$ goal ?
 $A[] \neg L4$?



Timed Game Automata & Synthesis



A (memoryless) winning strategy

- from $(l_0, 0)$, play $(0.5, c_1)$

\leadsto can be preempted by u_3

Problems to be considered:

- Does there exist a winning strategy?
- If yes, compute one (as simple as possible)



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.

↪ classical regions are sufficient for solving such problems

Theorem [AM99, BHPR07, JT07]

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

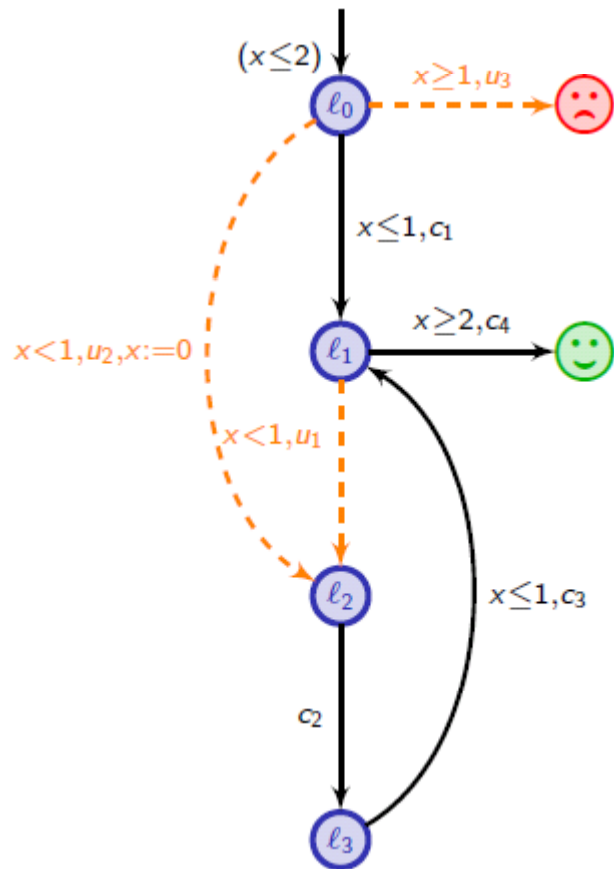
[AM99] Asarin, Maler. As soon as possible: time optimal control for timed automata (*HSCC'99*).

[BHPR07] Brihaye, Henzinger, Prabhu, Raskin. Minimum-time reachability in timed games (*ICALP'07*).

[JT07] Jurdziński, Trivedi. Reachability-time games on timed automata (*ICALP'07*).

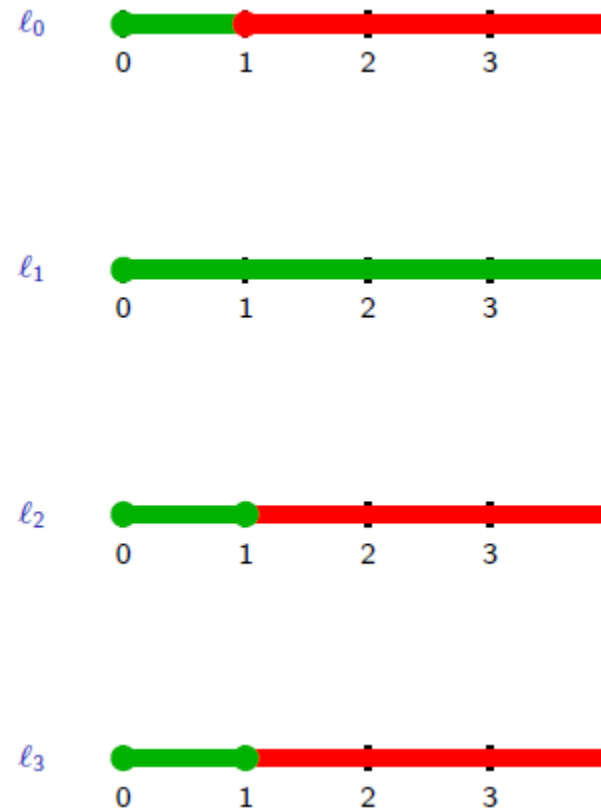


Computing Winning States



Winning states

Losing states



Reachability Games

Backwards Fixed-Point Computation

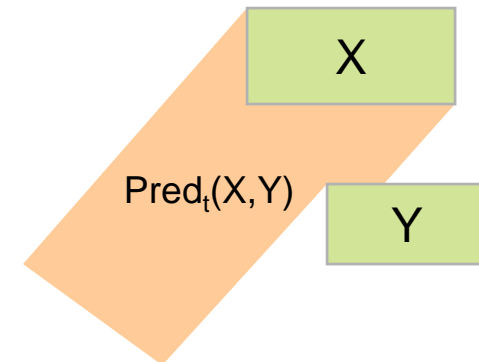
Definitions

$$\text{cPred}(X) = \{ q \in Q \mid \exists q' \in X. q \xrightarrow{c} q' \}$$

$$\text{uPred}(X) = \{ q \in Q \mid \exists q' \in X. q \xrightarrow{u} q' \}$$

$$\text{Pred}_t(X, Y) = \{ q \in Q \mid \exists t. q^t \in X \text{ and } \forall s \leq t. q^s \in Y^c \}$$

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



Theorem:

The set of winning states is obtained as the least fixpoint of the function:

$$X \mapsto \pi(X) \cup \text{Goal}$$


Symbolic On-the-fly Algorithms for Timed Games

[CDF+05, BCD+07]

- S, S' 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 (controllable/uncontrollable);
- $Waiting \subseteq E$ is the list of symbolic transitions waiting to be processed;
- $Passed$ is the list of the passed symbolic states;
- $Win[S] \subseteq S$ is the subset of S currently known to be winning;
- $Depend[S] \subseteq E$ indicates the edges (predecessors) of S which must be processed before information about S is obtained.

Initialization:

```

Passed ← {S0} where S0 = {(l0, 0)}↗;
Waiting ← {(S0, α, S') | S' = Postα(S0)↗};
Win[S0] ← S0 ∩ ({Goal} × ℝ≥0X);
Depend[S0] ← ∅;
    
```

Main:

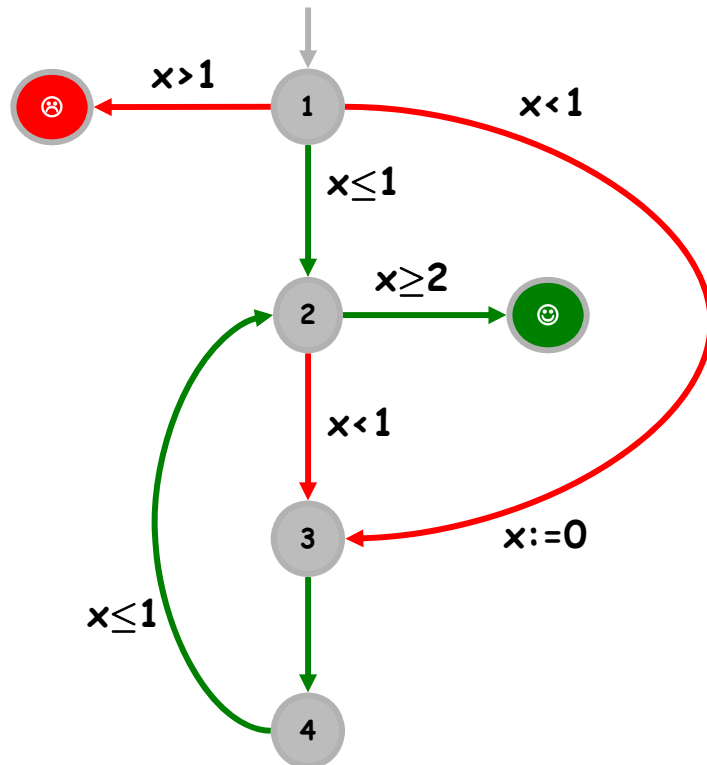
```

while ((Waiting ≠ ∅) ∧ (s0 ∉ Win[S0])) do
  e = (S, α, S') ← pop(Waiting);
  if S' ∉ Passed then
    Passed ← Passed ∪ {S'};
    Depend[S'] ← {(S, α, S')};
    Win[S'] ← S' ∩ ({Goal} × ℝ≥0X);
    Waiting ← Waiting ∪ {(S', α, S'') | S'' = Postα(S')↗};
    if Win[S'] ≠ ∅ then Waiting ← Waiting ∪ {e};
  reevaluate(*);
  Win* ← Predt(Win[S] ∪ ∪S→Tc Predc(Win[T]),
              ∪S→Tu Predu(T \ Win[T])) ∩ S;
  if (Win[S] ⊂ Win*) then
    Waiting ← Waiting ∪ Depend[S]; Win[S] ← Win*;
    Depend[S'] ← Depend[S'] ∪ {e};
  endif
endwhile
    
```

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



Symbolic On-the-fly Algorithms for Timed Games



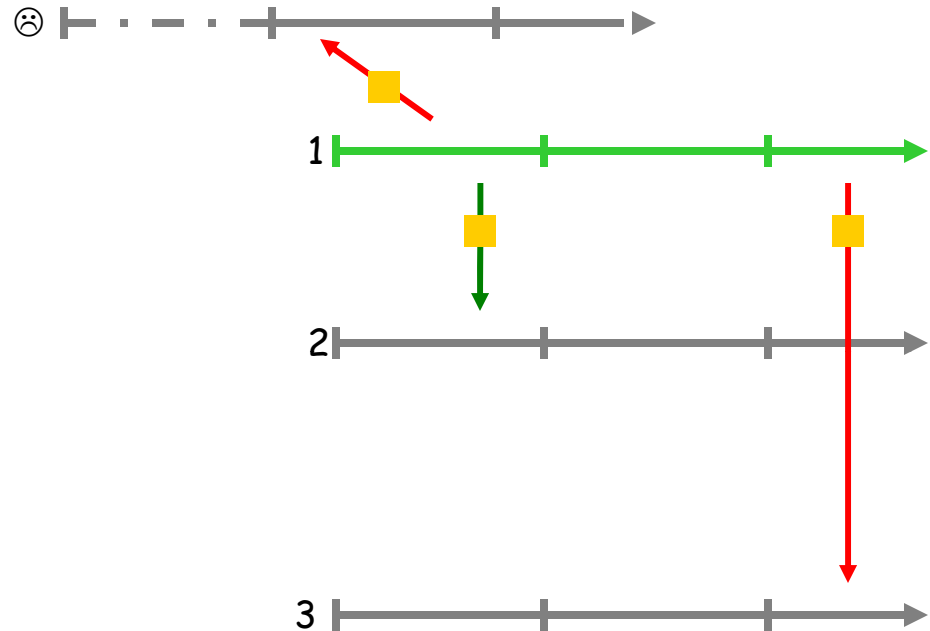
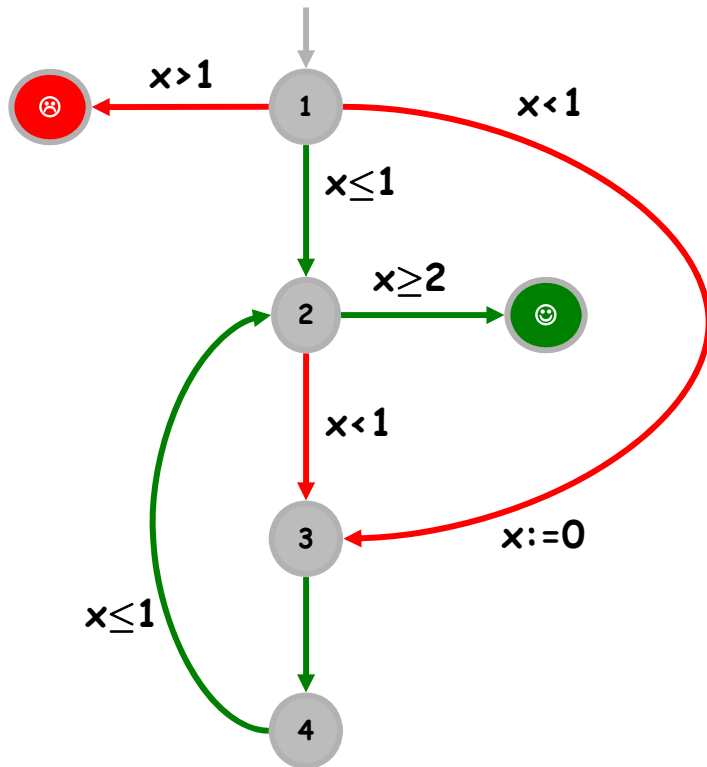
Passed

Waiting

Depend



Symbolic On-the-fly Algorithms for Timed Games



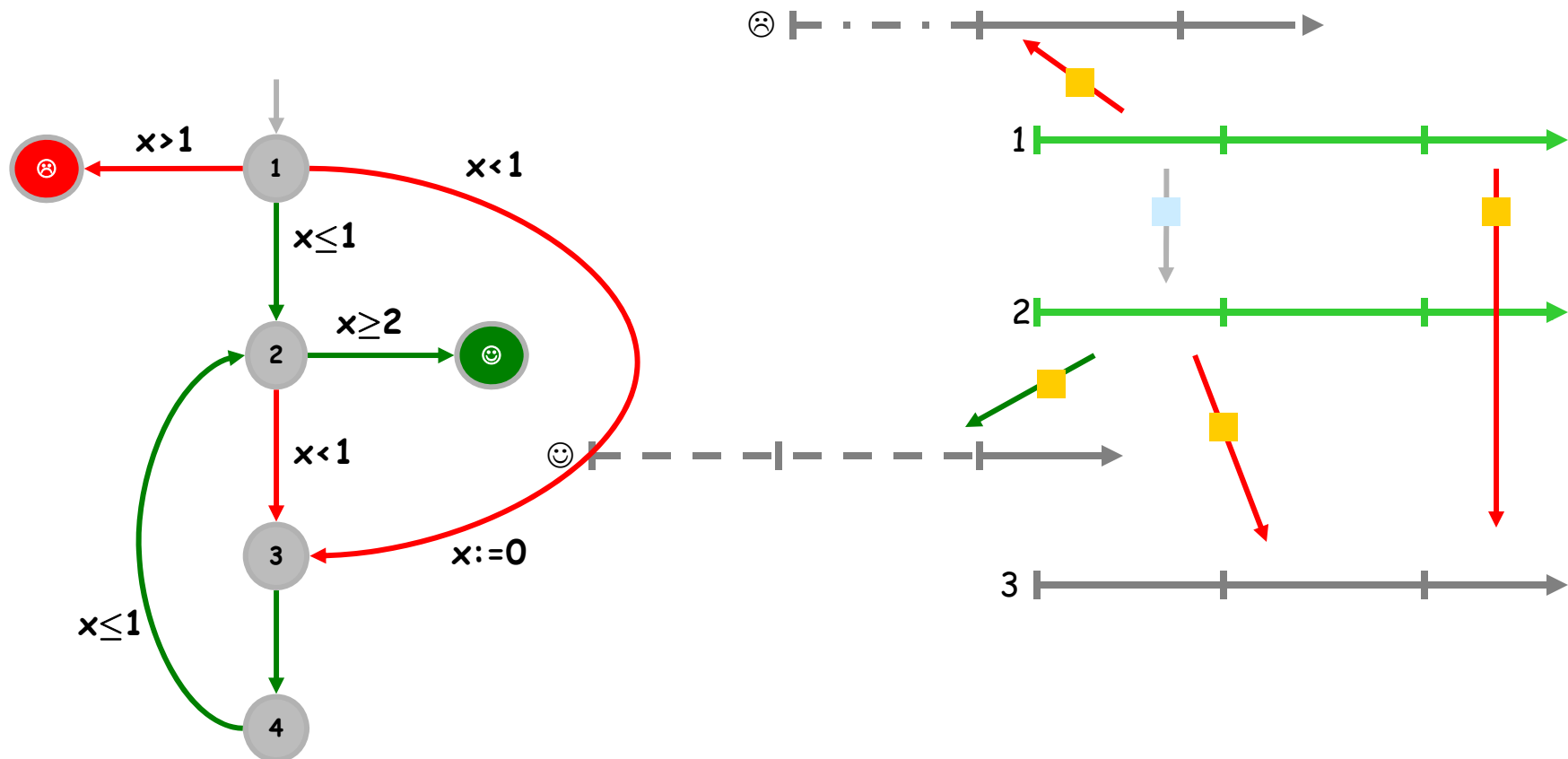
Passed

Waiting

Depend



Symbolic On-the-fly Algorithms for Timed Games



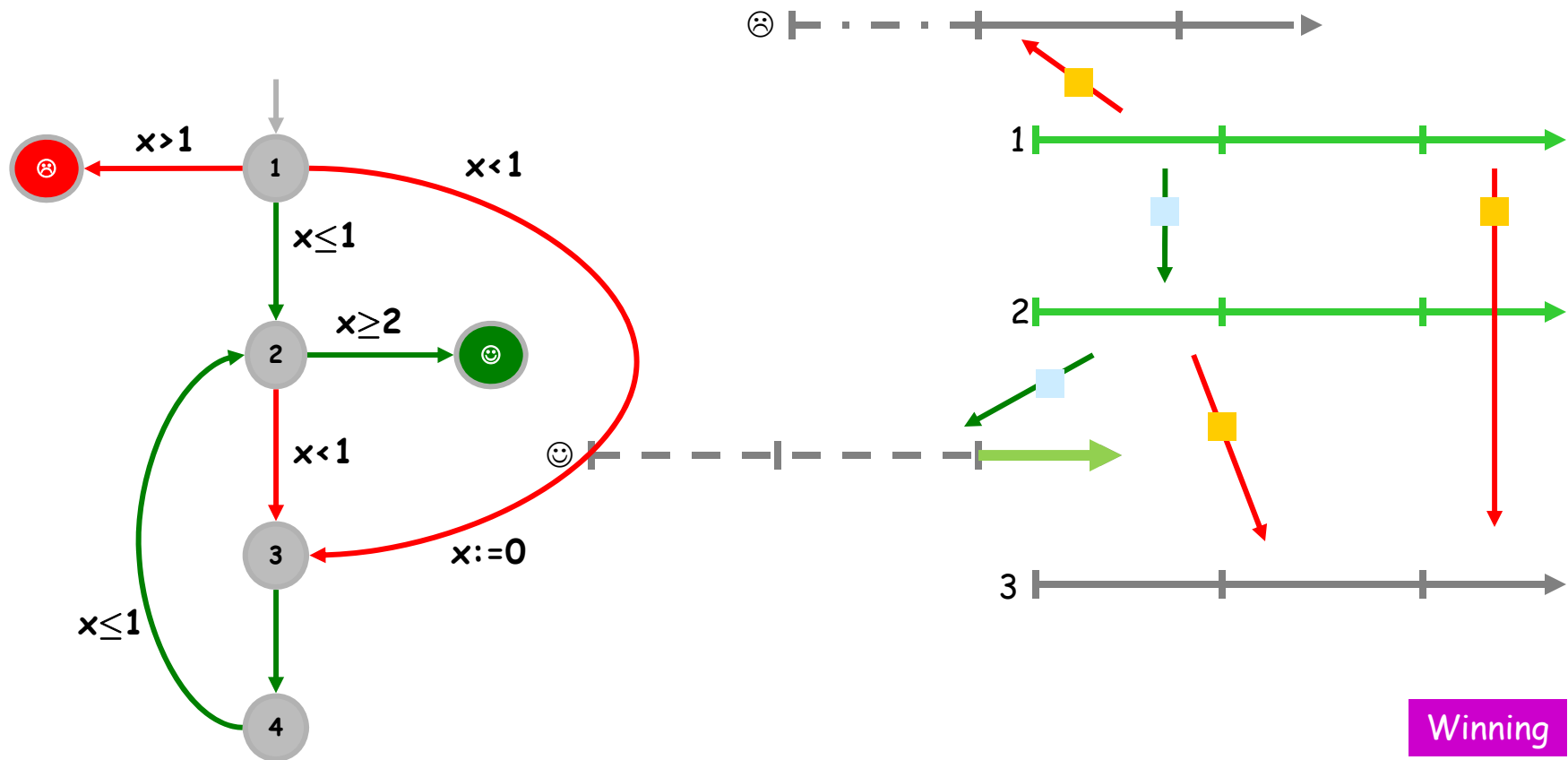
Passed

Waiting

Depend



Symbolic On-the-fly Algorithms for Timed Games



Winning

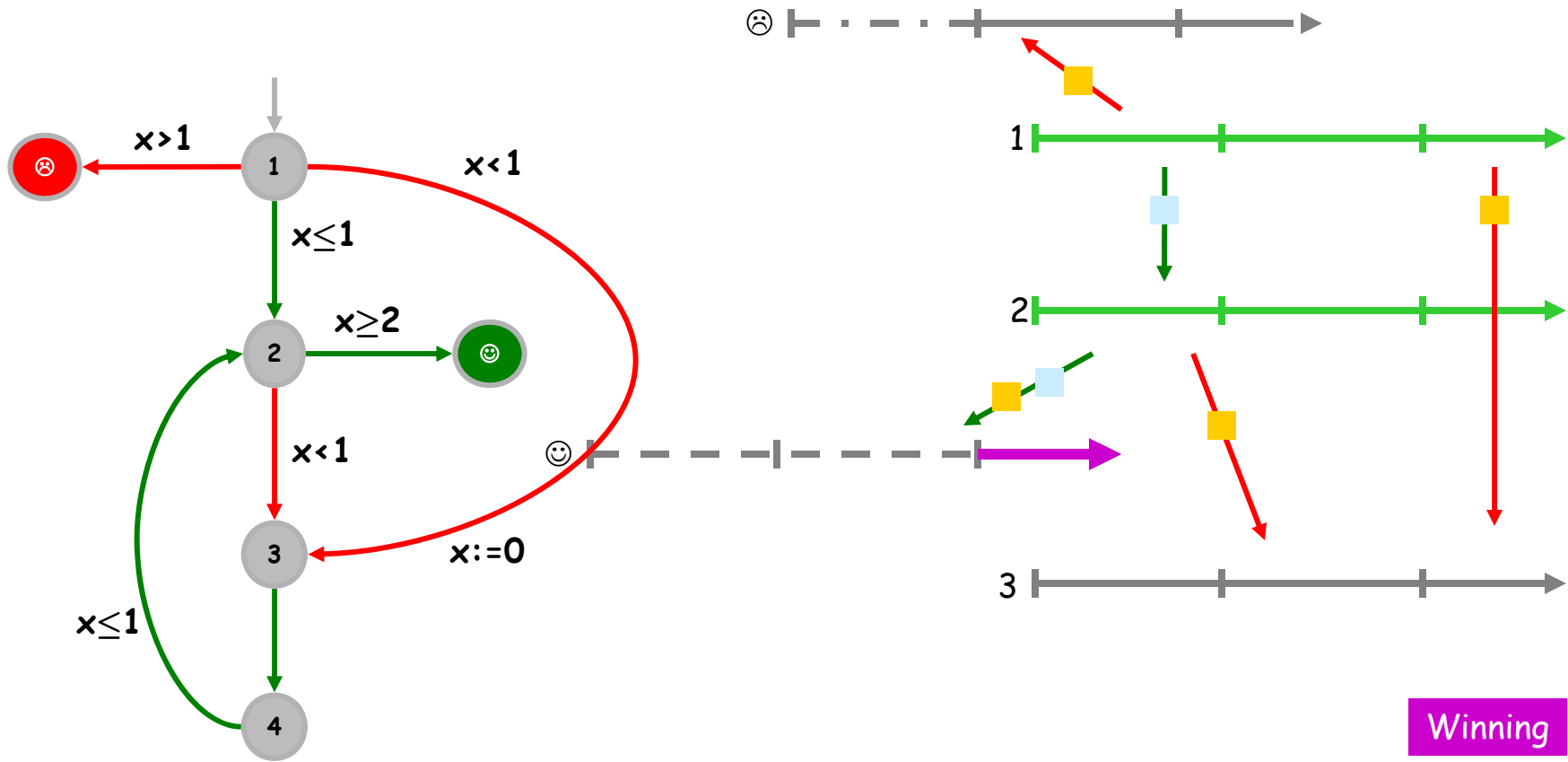
Passed

Waiting

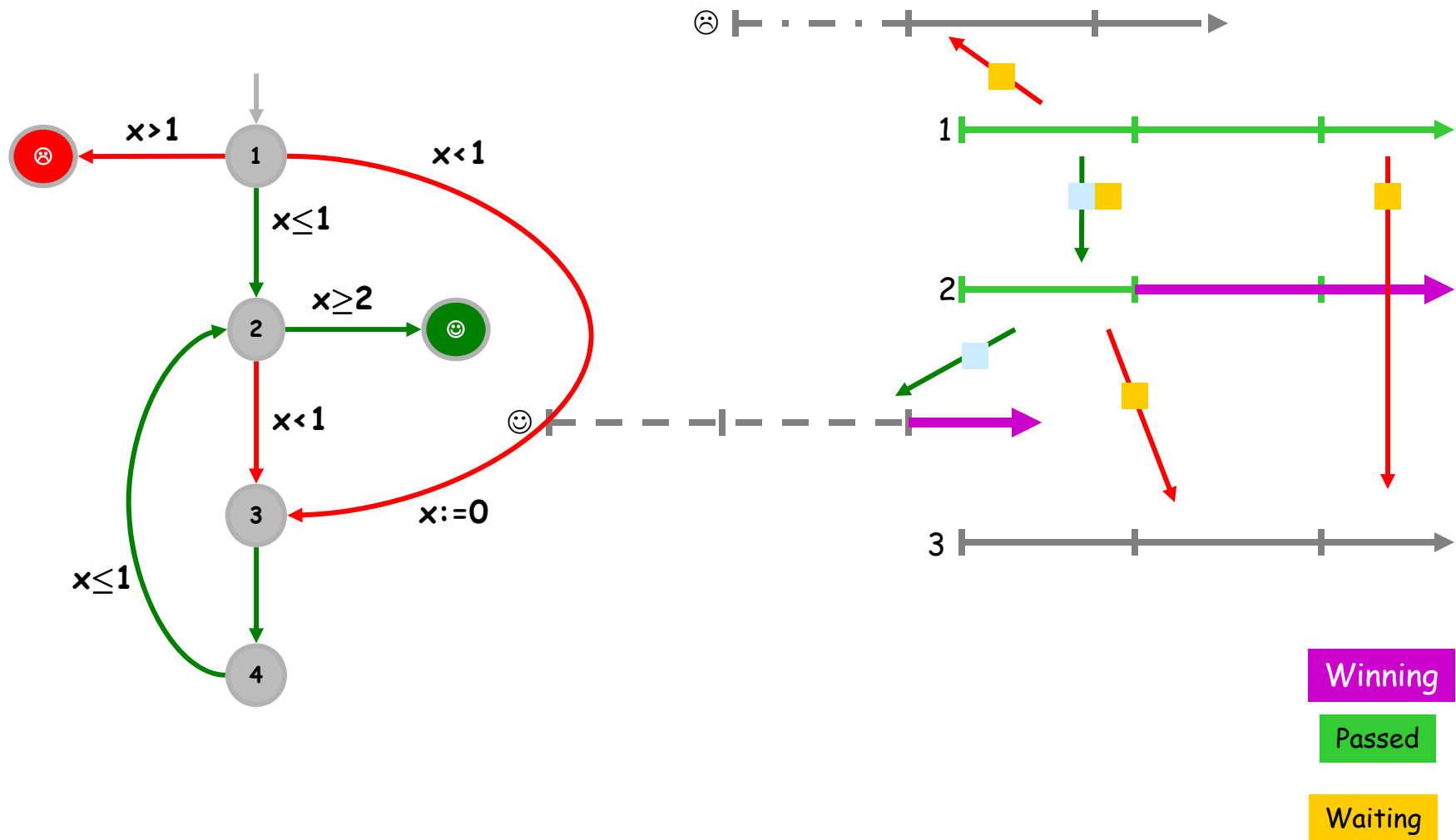
Depend



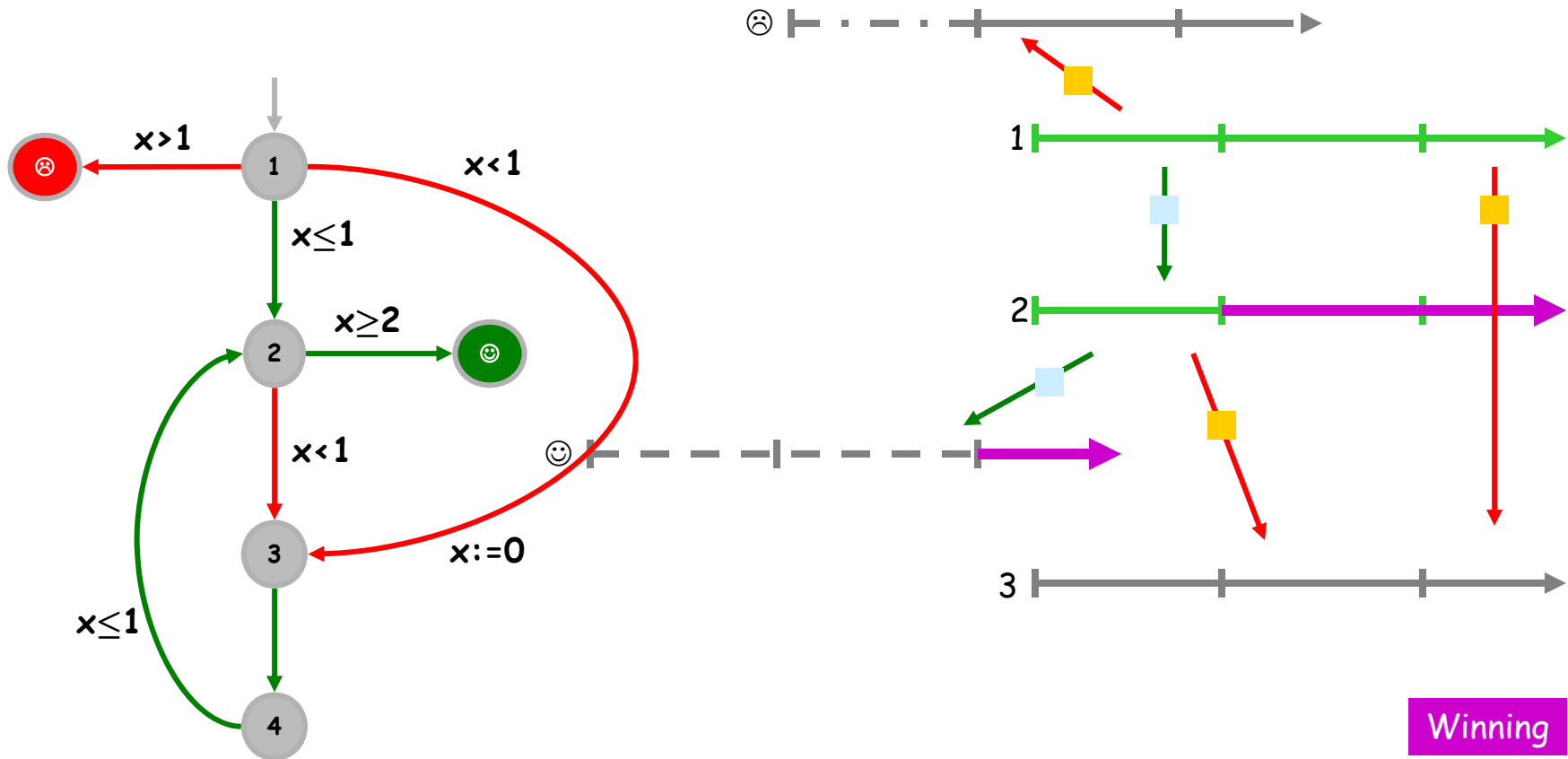
Symbolic On-the-fly Algorithms for Timed Games



Symbolic On-the-fly Algorithms for Timed Games



Symbolic On-the-fly Algorithms for Timed Games



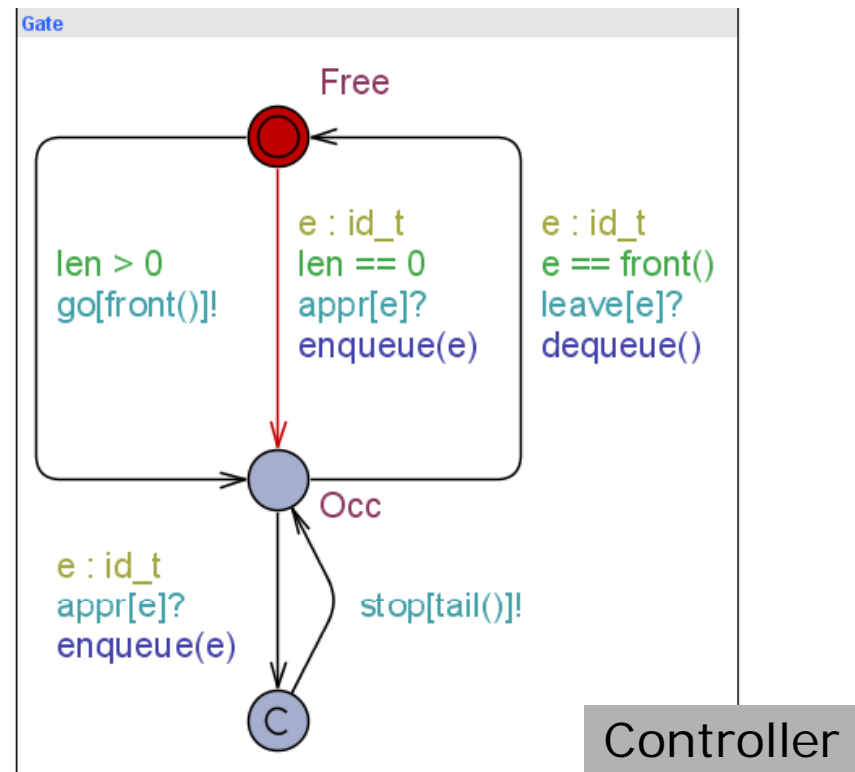
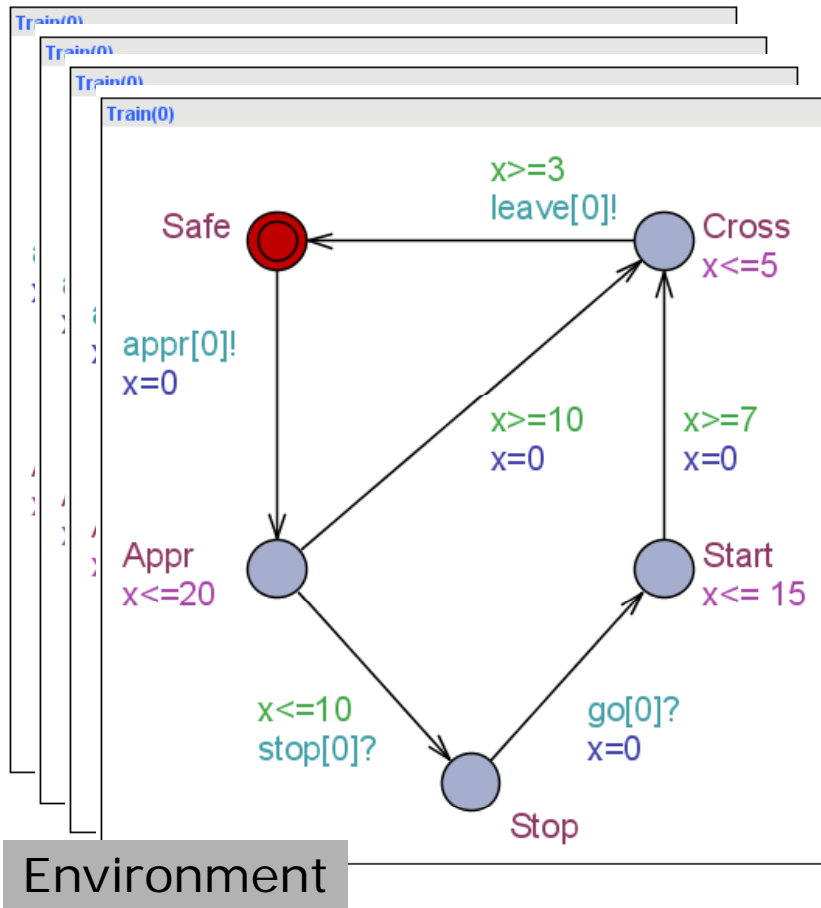
- **Reachability properties:**
 - control: $A[p U q]$ *until*
 - control: $A[\langle \rangle q] \Leftrightarrow \text{control: } A[\text{true} U q]$
- **Safety properties:**
 - control: $A[p W q]$ *weak until*
 - control: $A[] p \Leftrightarrow \text{control: } A[p W \text{false}]$
- **Time-optimality :**
 - $\text{control_t}^*(u,g): A[p U q]$
 - u is an upper-bound to prune the search
 - g is the time to the goal from the current state

[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*).



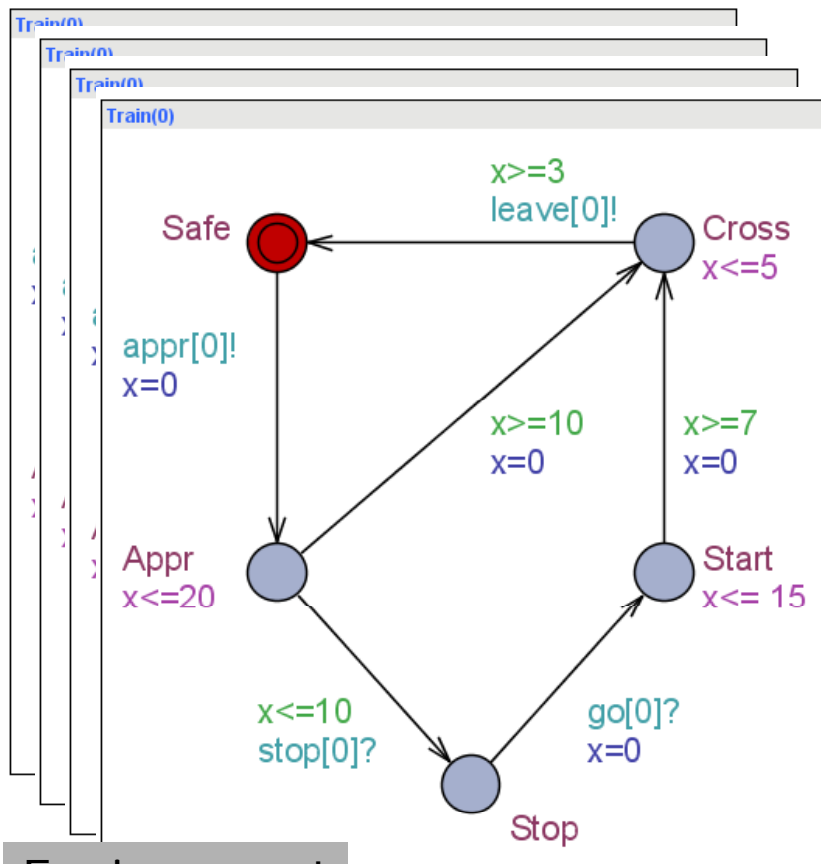
Model Checking (ex Train Gate)



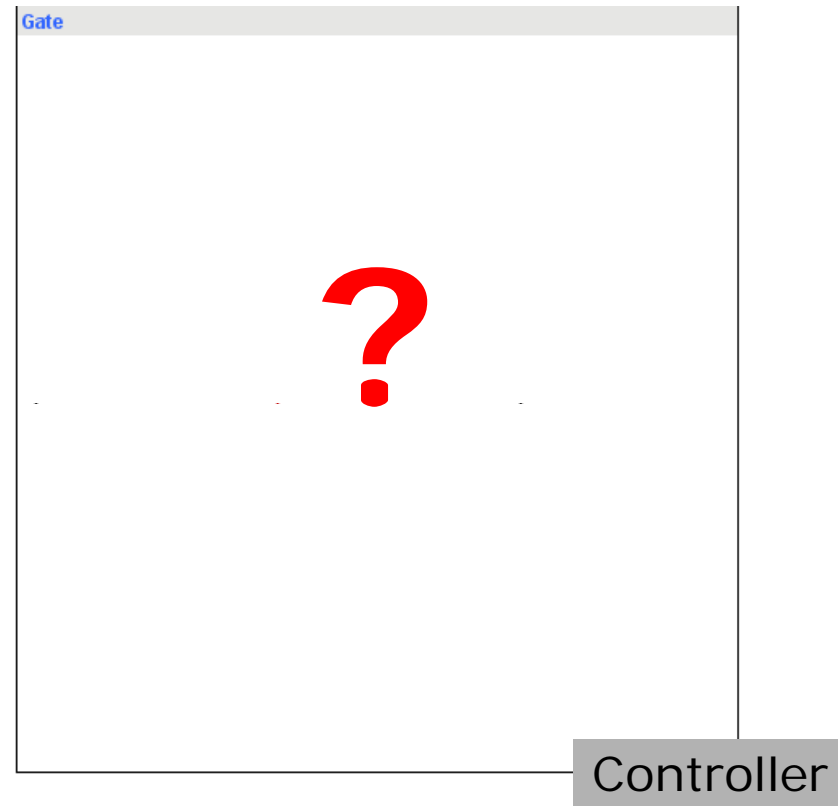
ϕ : Never two trains at the crossing at the same time



Synthesis (ex Train Gate)



Environment

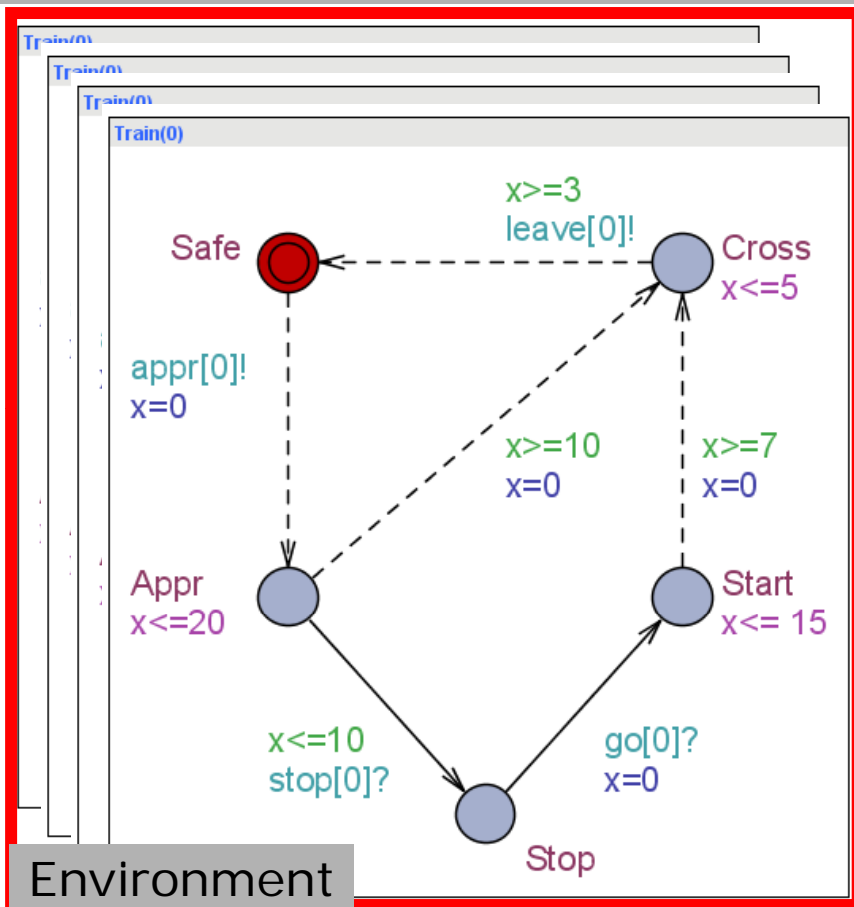


Controller

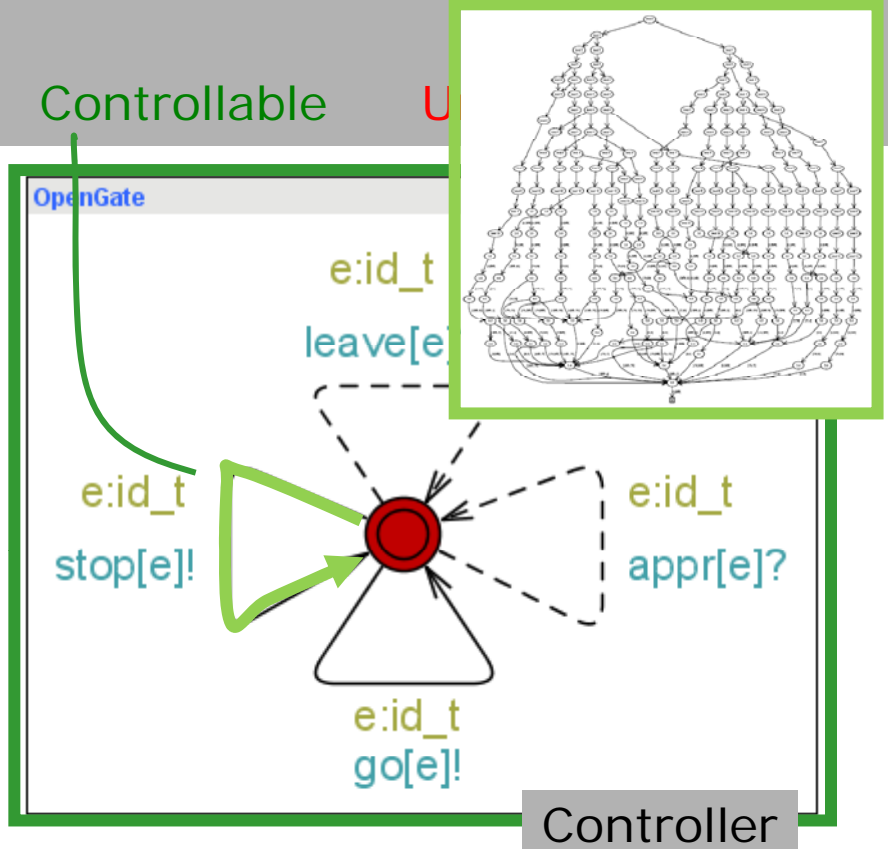
ϕ : Never two trains at the crossing at the same time



Timed Games



Find strategy for controllable actions st behaviour satisfies ϕ

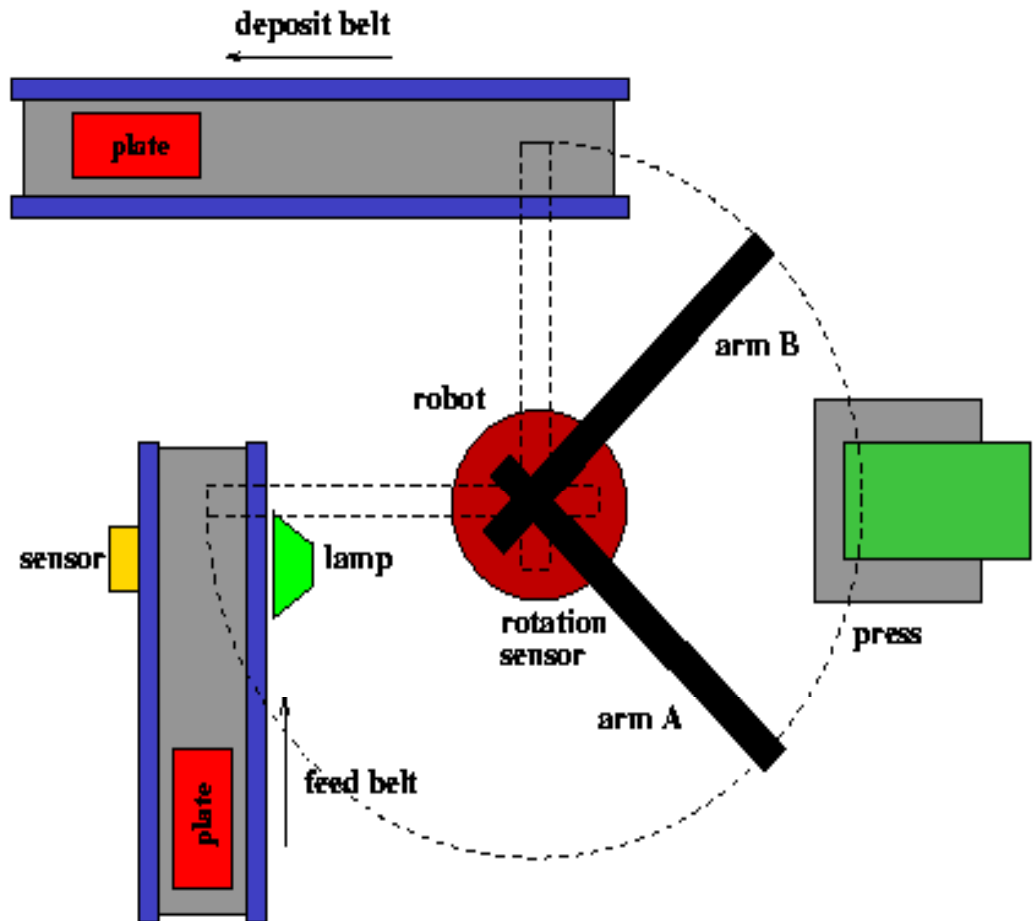


ϕ : Never two trains at the crossing at the same time



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

C:\Documents and Settings/kg\Desktop/DESKTOP FEB 2007/UPPAAL/uppaal-tiga-0.14/demo/prodcell-control-nondet.xml - UPPAAL

File Edit View Tools Options Help

Editor Simulator Verifier

Drag out

Transition chooser

0.0 1.0 2.0 3.0 4.0

Robot2
Plate(0)

Delay: 0 Reset

Take transition

Trace controls

First 0 Last

Prev Play Next

Speeder

Slow Fast

Random

Simulation Trace

atA, NotReady, NotReady, NotReady, No

Robot2

atA $x=0$ $x \leq \text{MAXROT}$ toB $x \geq \text{MINROT}$ $x=0$

IB && IPress && $x \geq 1$ takeP2! $x=0, B=\text{true}, \text{Press}=\text{false}$

B && IPress && $x \geq 1$ leaveP2! $x=0, B=\text{false}, \text{Press}=\text{true}$

IA && $x \geq 1$ takeA! $x=0, A=\text{true}$

IB && Press takeP2! $B=\text{true}, \text{Press}=\text{false}$

B && IPress leaveP2! $B=\text{false}, \text{Press}=\text{true}$

IA && IPress && $x \geq 1$ leaveP1! $x=0, A=\text{false}, \text{Press}=\text{true}$

IA && Press takeP1! $A=\text{true}, \text{Press}=\text{false}$

atB $x=0$ $x \leq \text{MAXROT}$ toA $x \geq \text{MINROT}$ $x=0$

IA && IPress && $x \geq 1$ leaveP1! $x=0, A=\text{false}, \text{Press}=\text{true}$

A && IPress && $x \geq 1$ leaveP1! $x=0, A=\text{true}, \text{Press}=\text{false}$

IA && Press takeP1! $A=\text{true}, \text{Press}=\text{false}$

leaveP1! $A \&\& \text{IPress}$ $A=\text{false}, \text{Press}=\text{true}$

leaveB! $x=0, B=\text{false}$

Plate(0)

Init $x=0$ $0 < \text{PLATES}-1 \&\& x \geq \text{MINWAIT}$ $\text{go}[0+1]!$

Ariving $x=12$ $0 == \text{PLATES}-1 \&\& x \geq \text{MINWAIT}$ $x=0$

Available $x=2$ $x < 2$ takeA?

TakenA leaveP1?

Press takeP2?

TakenB leaveB?

Safe

NotReady $0=0$ $x=0$ $\text{go}[0]?$ $x=0$

BAD $x=2$

Plate(1)

Init $x=0$ $1 < \text{PLATES}-1 \&\& x \geq \text{MINWAIT}$ $\text{go}[1+1]!$

Ariving $x=12$ $1 == \text{PLATES}-1 \&\& x \geq \text{MINWAIT}$ $x=0$

Available $x=2$ $x < 2$ takeA?

TakenA leaveP1?

Press takeP2?

TakenB leaveB?

Safe

Sunday, July 18, 2010



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

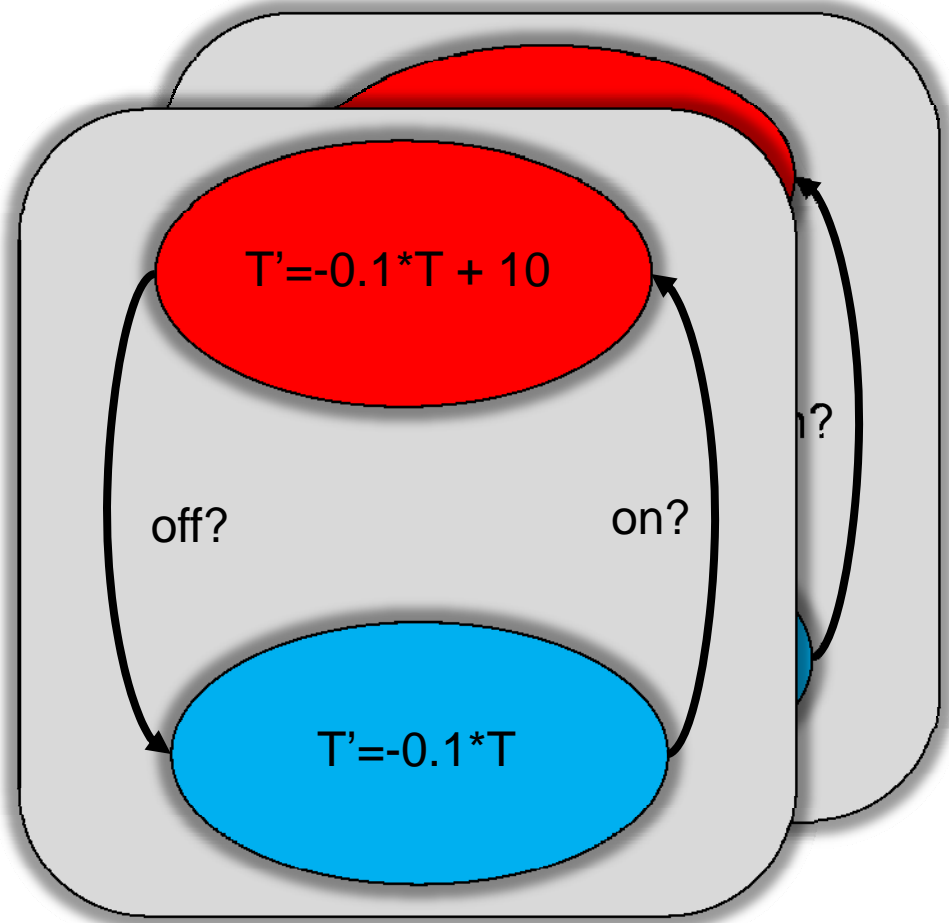
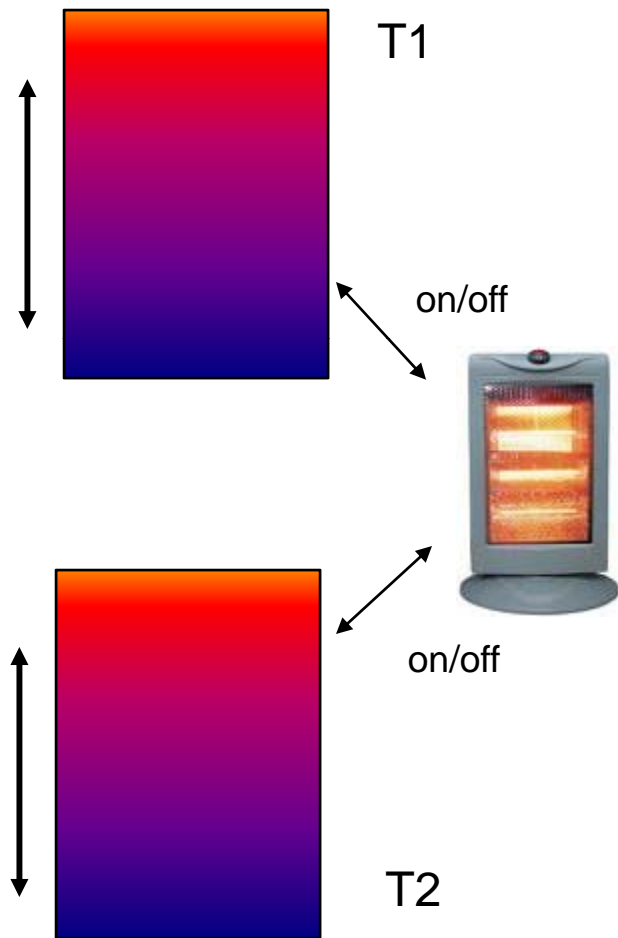
[BCD+07]

[CDF+05] Cassez, David, Fleury, Larsen, Lime. Efficient on-the-fly algorithms for the analysis of timed games (*CONCUR'05*).

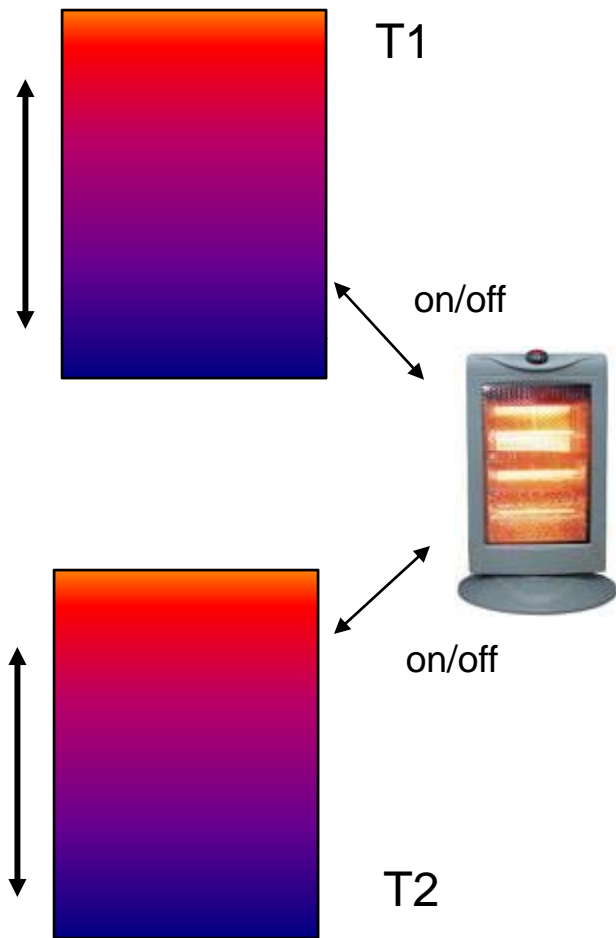
[BCD+07] Berhmann, Cougnard, David, Fleury, Larsen, Lime. Uppaal-Tiga: Time for playing games! (*CAV'07*).



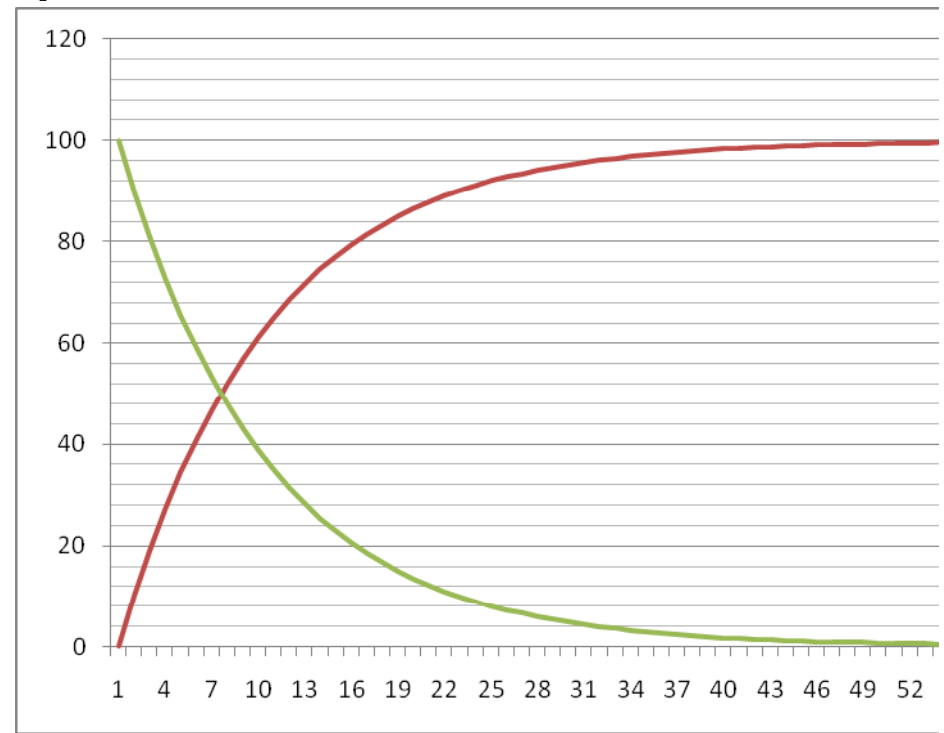
Two Tank Example



Two Tank Example



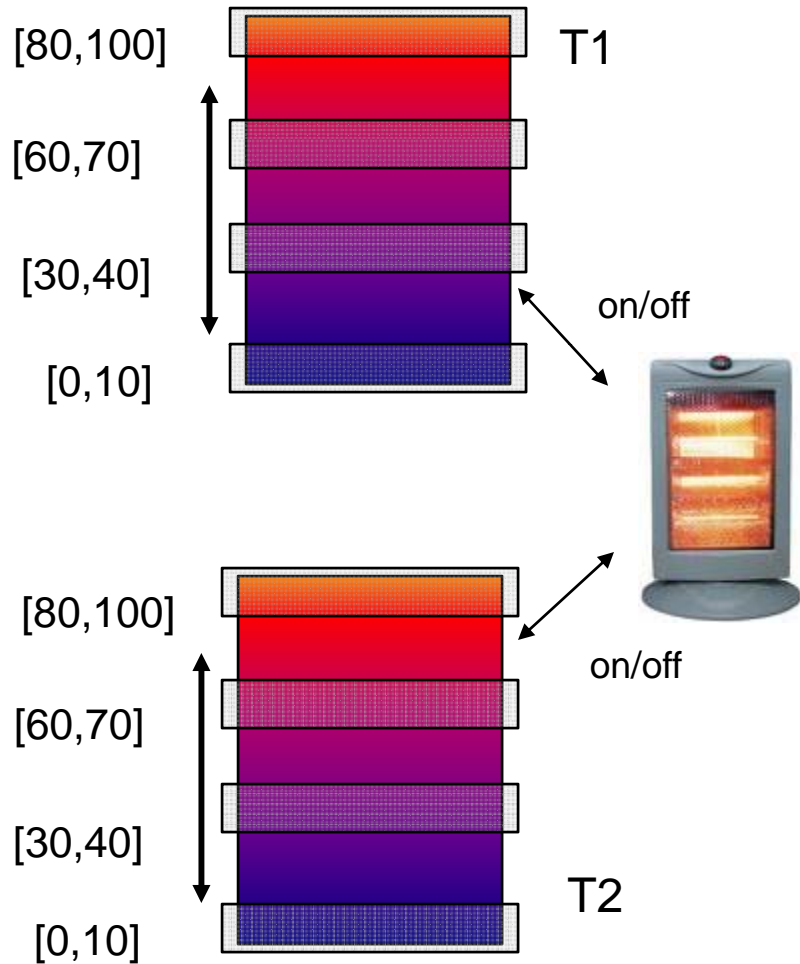
Temp



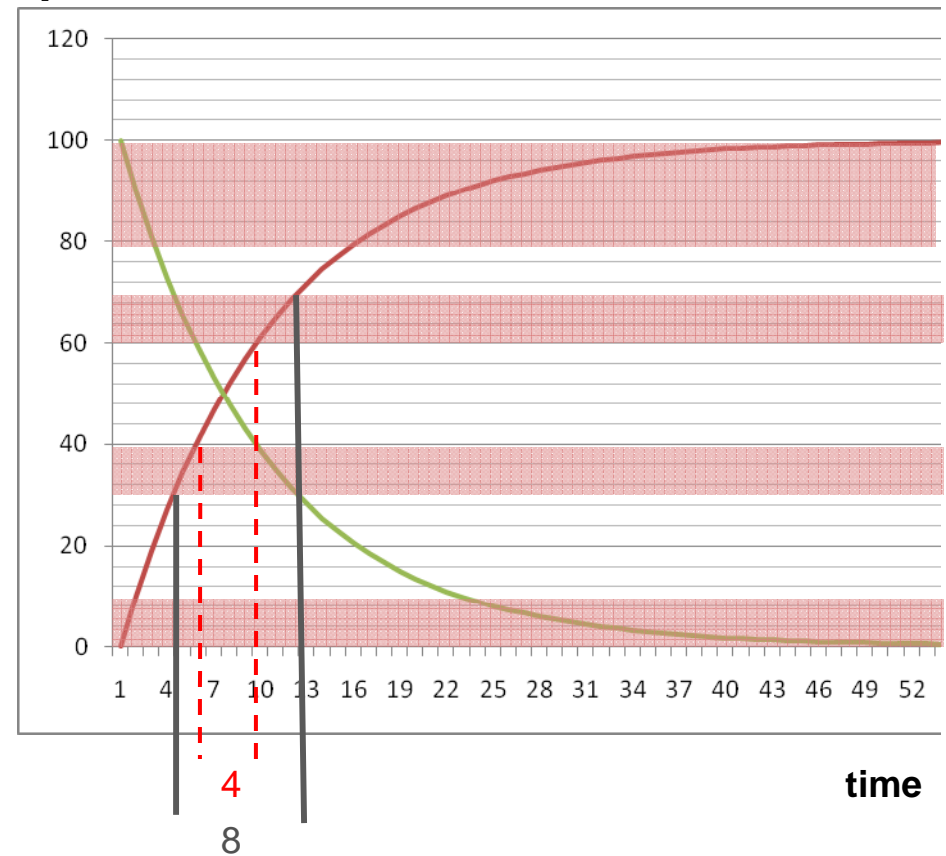
time



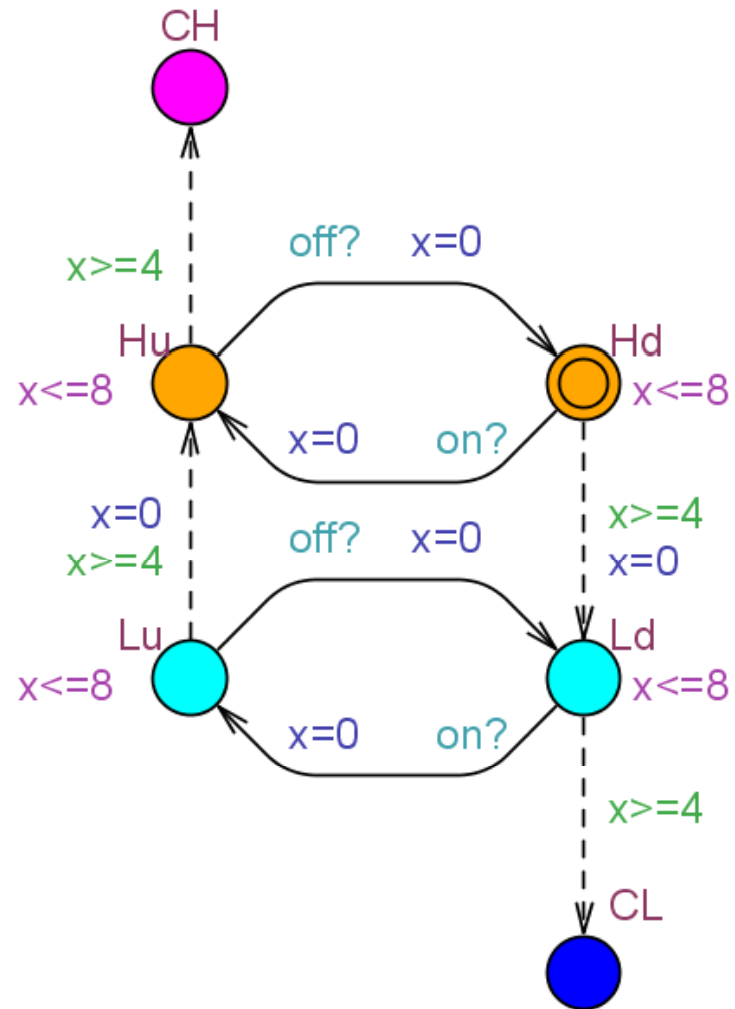
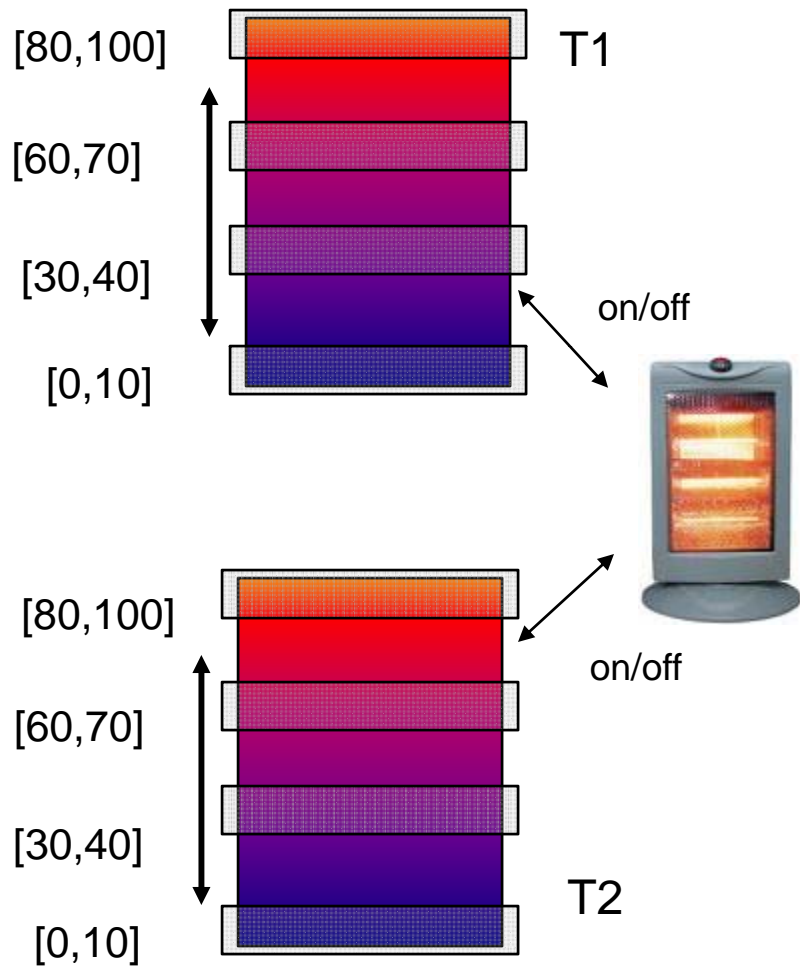
Two Tank Example



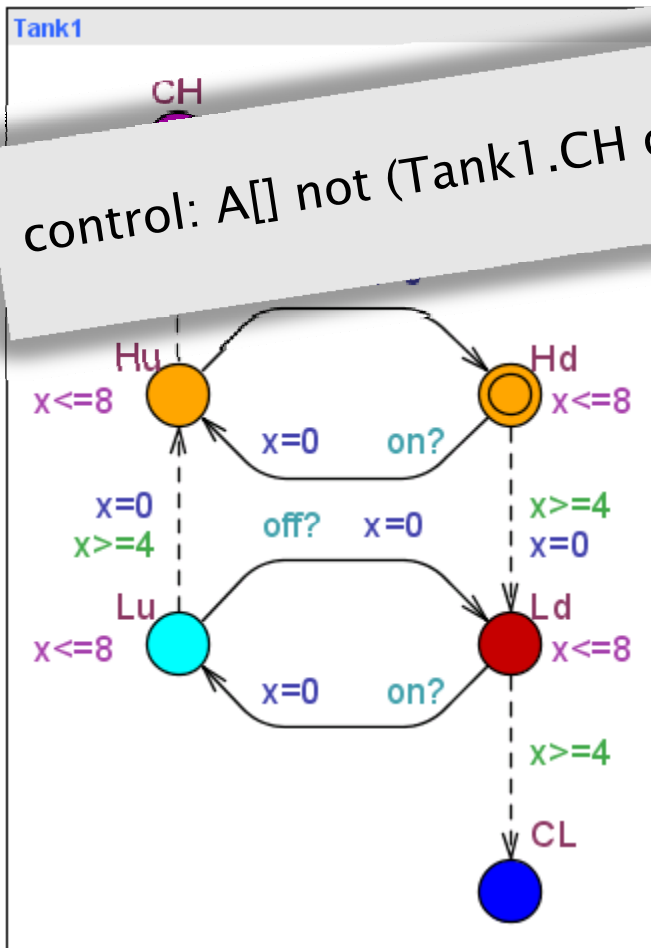
Temp



Two Tank Example



Two Tank Example



control: A[] not (Tank1.CH or Tank1.CL or Tank2.CH or Tank2.CL)

```

state: ( Tank1.Hu Tank2.Hu Controller.on )
when you are in
  (Tank1.x==3 && Tank1.x-Controller.z==7 && Tank2.x<3 && Controller.z==1) ||
  (Tank1.x==7 && 1<Tank2.x && Tank1.x-Controller.z==6 && Tank2.x<2 && Controller.z==1) ||
  (Tank1.x==5 && Tank1.x-Controller.z==5 && Tank2.x<1 && Controller.z==1),
take transition
  Controller.on->Controller.on { z == 1, tau, z := 0 }
when you are in
  (Tank1.x==7 && Tank1.x-Controller.z==5 && Tank2.x<1 && Controller.z==1) ||
  (Tank1.x==5 && Tank1.x-Tank2.x==3 && Tank2.x-Controller.z==1 && Controller.z==1) ||
  (Tank1.x==7 && Tank1.x-Tank2.x==4 && Tank2.x-Controller.z==2 && Controller.z==1) ||
  (Tank1.x==5 && Tank1.x-Tank2.x==3 && Tank2.x-Controller.z==2 && Controller.z==1) ||
  (Tank1.x==6 && Tank1.x-Tank2.x==1 && Tank2.x-Controller.z==1 && Controller.z==1) ||
  (Tank1.x==5 && Tank1.x-Tank2.x==2 && Tank2.x-Controller.z==2 && Controller.z==1) ||
  (Tank1.x==5 && Tank1.x-Tank2.x==3 && Tank2.x-Controller.z==1 && Controller.z==1) ||
  (Tank1.x==5 && Tank1.x-Tank2.x==4 && Tank2.x-Controller.z && Controller.z==1) ||
  (Tank1.x==4 && Tank1.x-Tank2.x==1 && Tank2.x-Controller.z==2 && Controller.z==1) ||
  (Tank1.x==4 && Tank1.x-Tank2.x==2 && Tank2.x-Controller.z==1 && Controller.z==1) ||
  (Tank1.x==4 && Tank1.x-Tank2.x==3 && Tank2.x-Controller.z && Controller.z==1) ||
  (Tank1.x==3 && Tank1.x-Tank2.x==1 && Tank2.x-Controller.z==1 && Controller.z==1) ||
  (Tank1.x==3 && Tank1.x-Tank2.x==2 && Tank2.x==Controller.z && Controller.z==1) ||
  (Tank1.x==2 && Tank1.x-Tank2.x==1 && Tank2.x==Controller.z && Controller.z==1),
take transition
  Controller.on->Controller.off { ? == 1, off!, ? := 0 }
  Tank2.Hu->Tank2.Hd { 1, off?, x := 0 }

state: ( Tank1.Lu Tank2.Ld Controller.on )
when you are in
  (Tank1.x==3 && 1<Tank2.x && Tank1.x-Controller.z==7 && Tank2.x<3 && Controller.z==1) ||
  (Tank1.x==7 && Tank1.x-Controller.z==5 && Tank2.x<3 && Controller.z==1) ||
  (Tank1.x==5 && Tank1.x-Controller.z==5 && Tank2.x<3 && Controller.z==1) ||
  (Tank1.x==4 && Tank1.x-Controller.z==3 && Tank2.x<3 && Controller.z==1) ||
  (Tank1.x==3 && Tank1.x-Controller.z==2 && Tank2.x<3 && Controller.z==1) ||
  (Tank1.x==2 && Tank1.x-Controller.z==1 && Tank2.x<3 && Controller.z==1) ||
  (Tank1.x==1 && Tank1.x==Controller.z && Tank2.x<3 && Controller.z==1),
take transition
  Controller.on->Controller.off { z == 1, off!, z := 0 }
  Tank1.Lu->Tank1.Ld { 1, off?, x := 0 }

state: ( Tank1.Ld Tank2.Ld Controller.off )
when you are in
  (Controller.z==1 && Tank1.x<2 && Tank2.x<4),
take transition

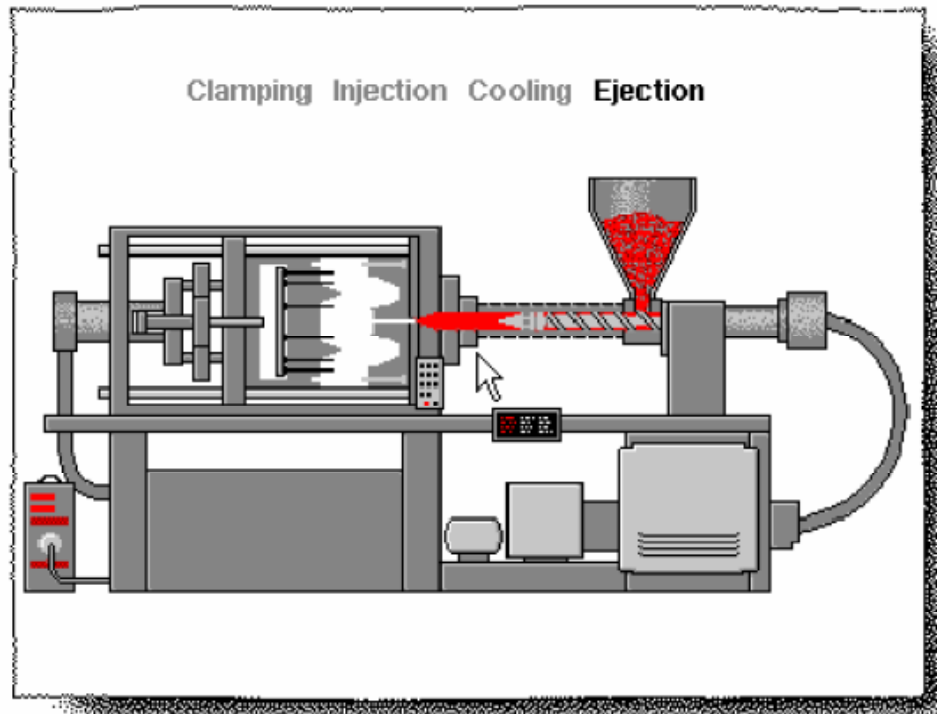
```



Plastic Injection Molding Machine



[CJL+09]

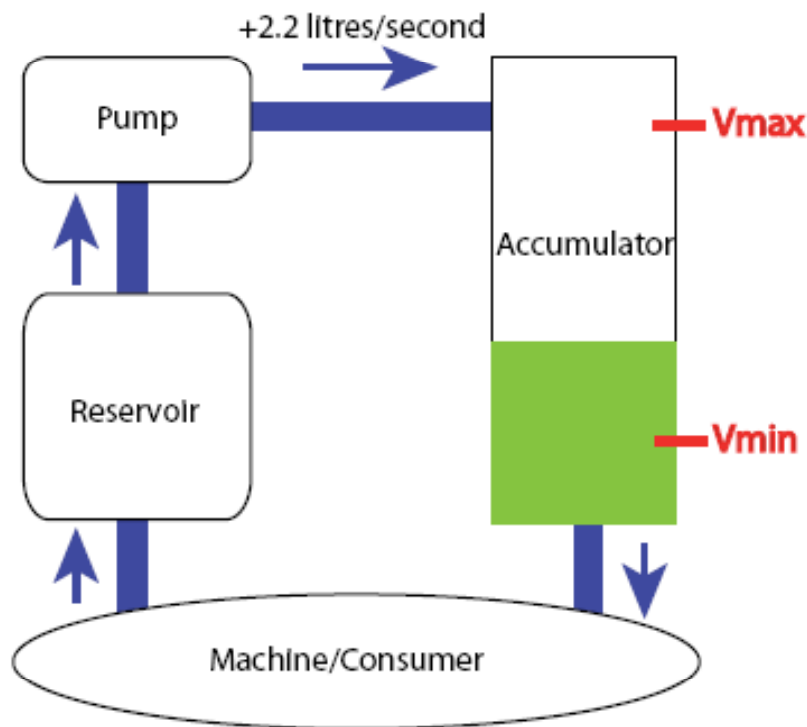


- Robust and optimal control
- Tool Chain
 - Synthesis: UPPAAL
TIGA
 - Verification: PHAVer
 - Performance: SIMULINK
- 40% improvement of existing solutions..

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



Oil Pump Control Problem

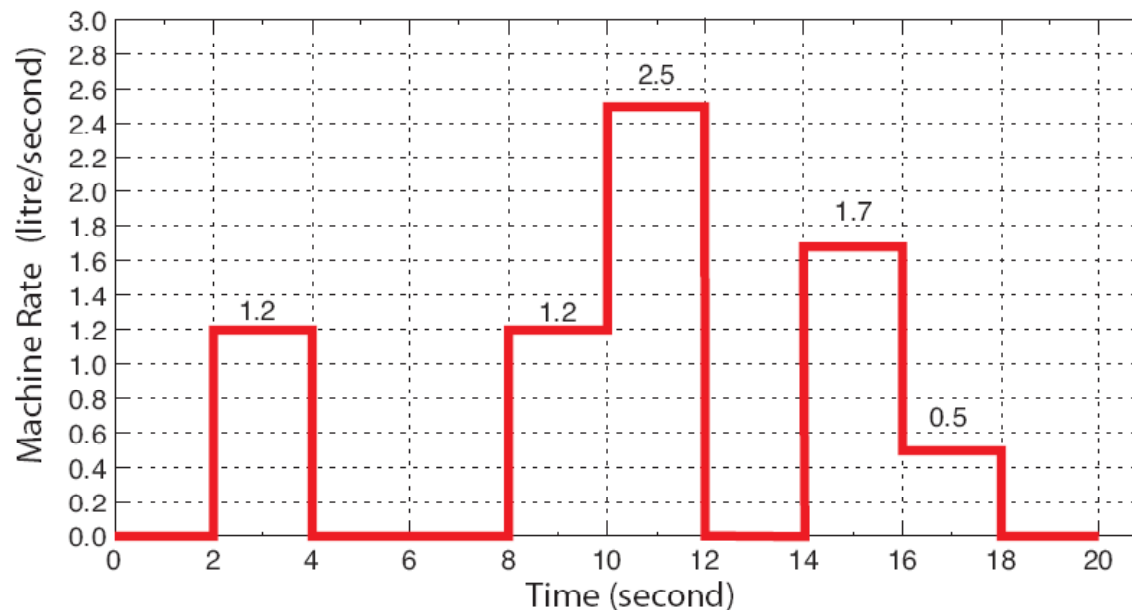


- **R1**: stay within safe interval [4.9,25.1]
- **R2**: minimize average/overall oil volume

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

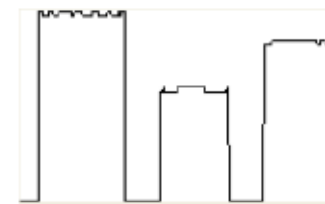


The Machine (consumption)

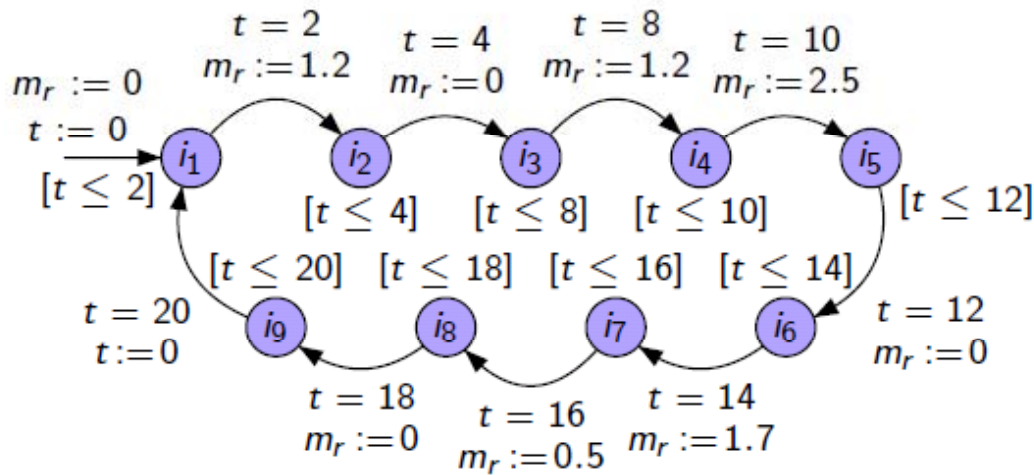


- Infinite cyclic demand to be satisfied by our control strategy.
- **P**: latency 2 s between state change of pump

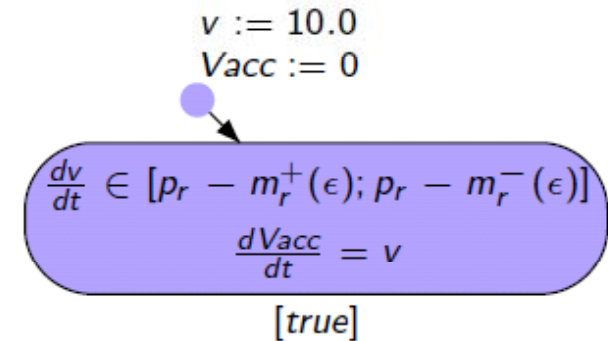
- **F**: noise 0.1 l/s



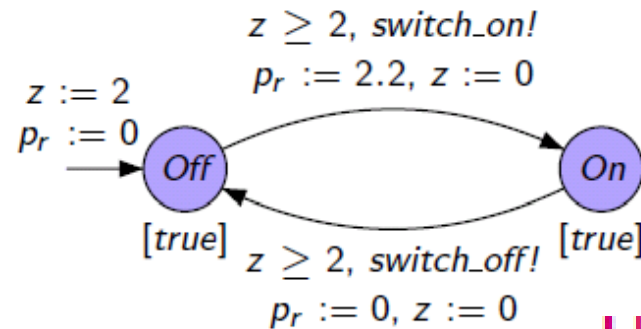
Hybrid Game Model



(a) The Machine



(b) The Accumulator



(c) The Pump

Undecidable problem!



Abstract Game Model

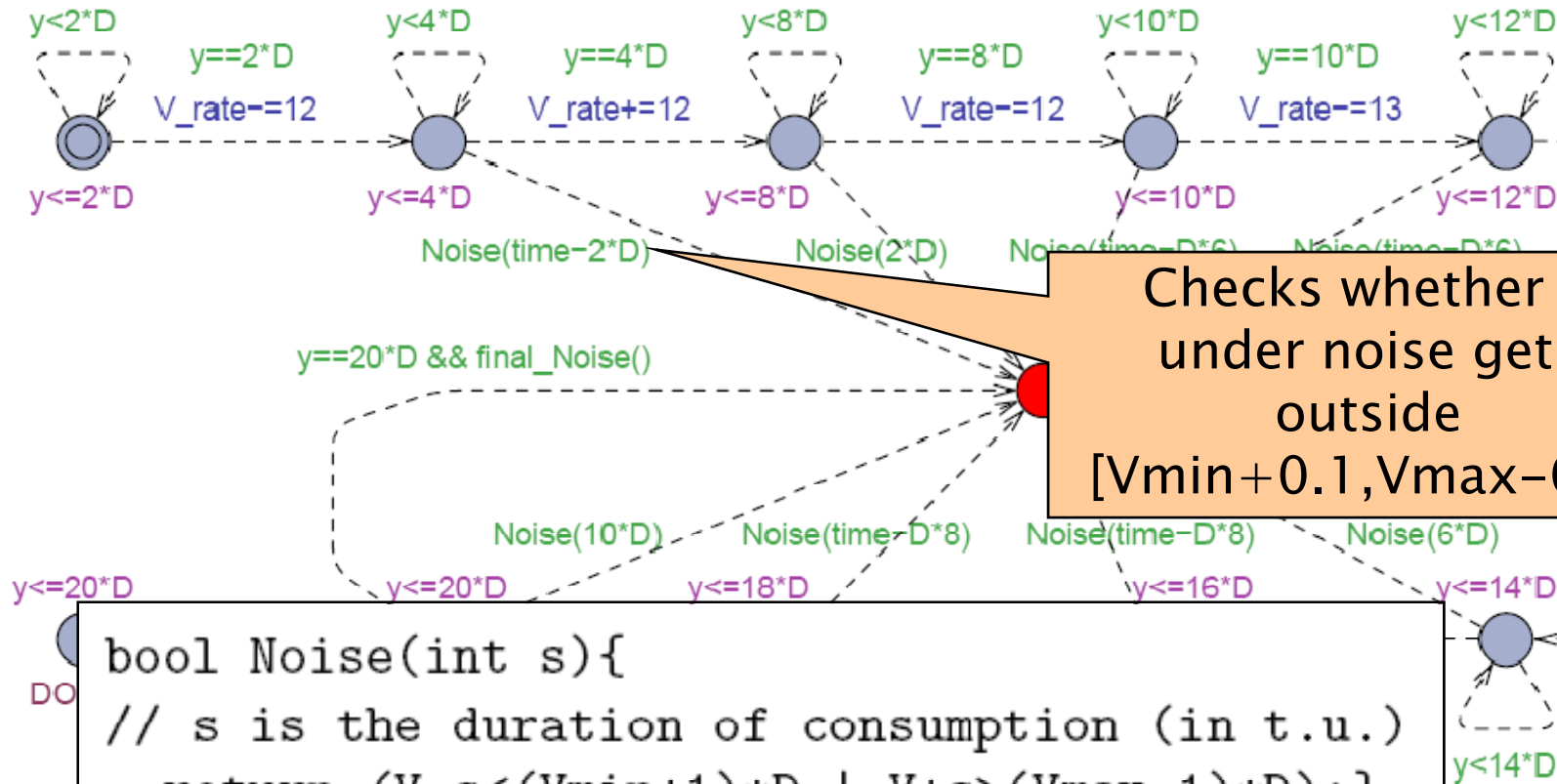


- UPPAAL Tiga offers games of perfect information
- **Abstract game** model such that states only contain information about:
 - Volume of oil at the beginning of cycle
 - The ideal volume as predicted by the consumption cycle
 - Current time within the cycle
 - State of the Pump (on/off)
 - Discrete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

D
V, V_rate
V_acc
time



Machine (uncontrollable)



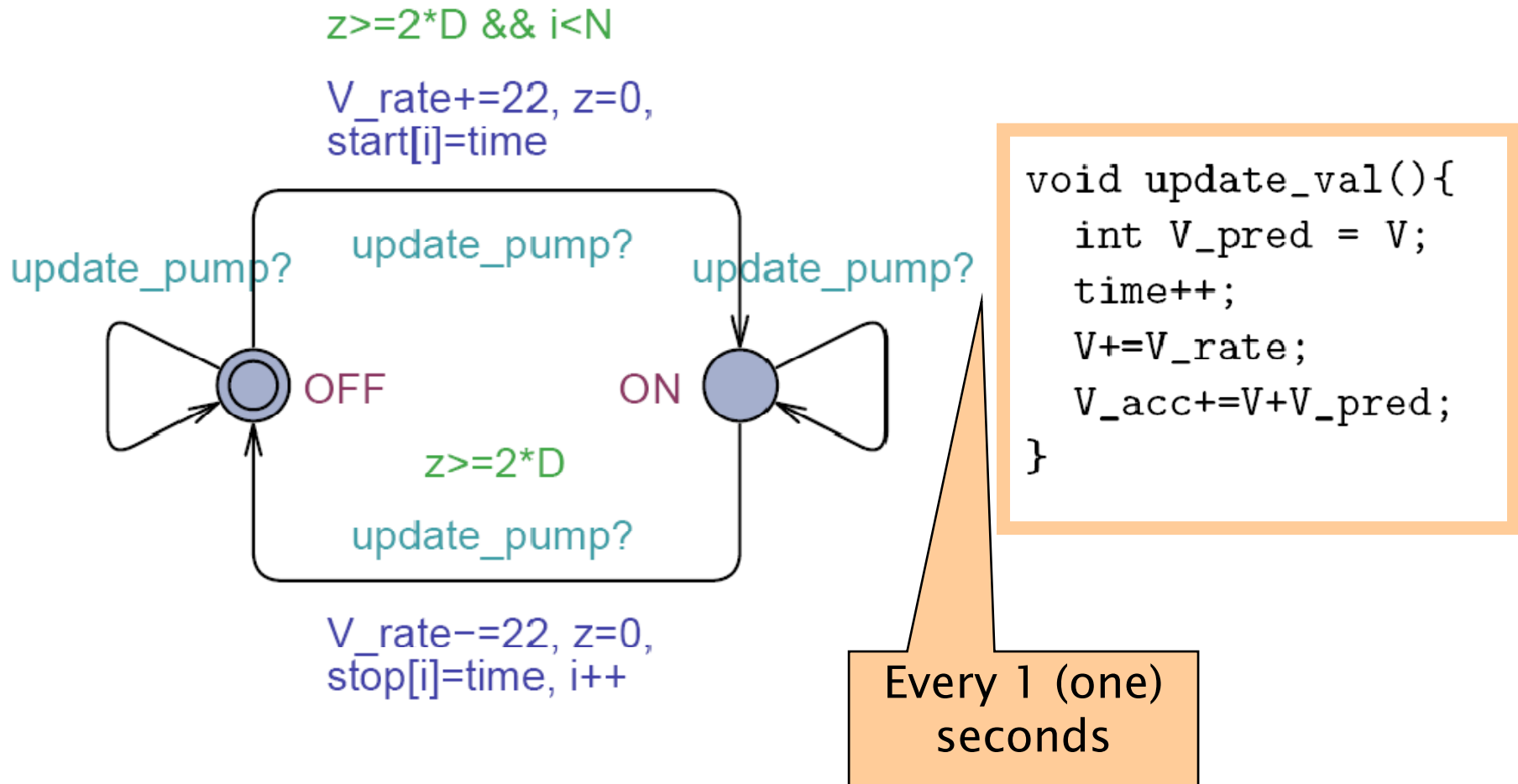
Checks whether V under noise gets outside $[V_{min}+0.1, V_{max}-0.1]$

```

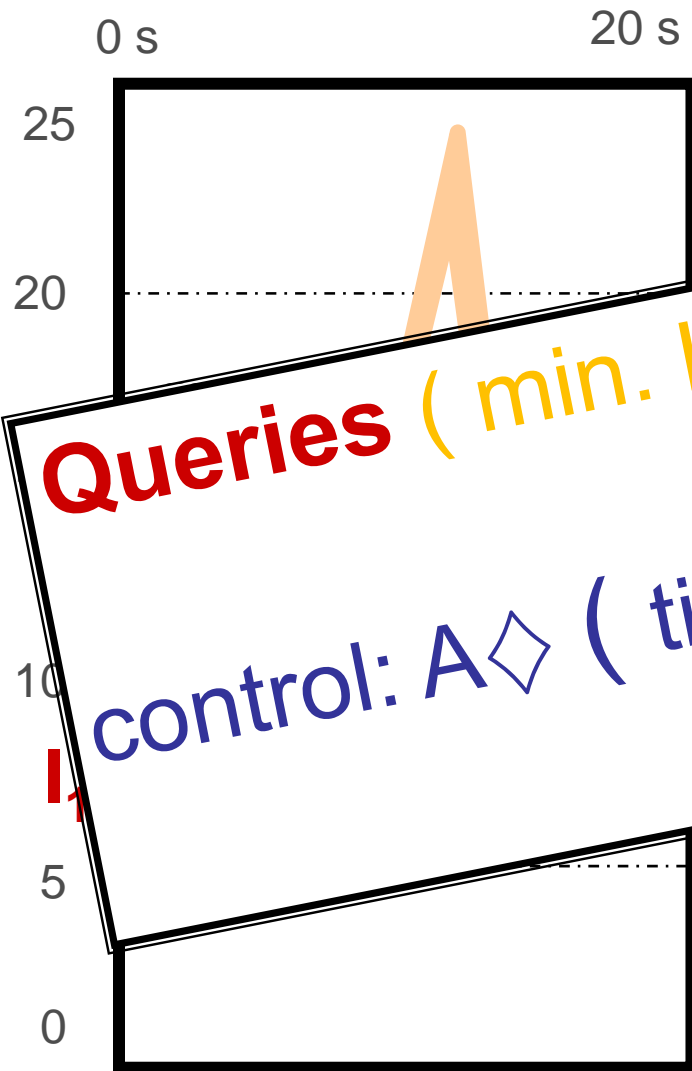
bool Noise(int s){
// s is the duration of consumption (in t.u.)
return (V-s<(Vmin+1)*D | V+s>(Vmax-1)*D);}
    
```



Pump (controllable)



Global Approach



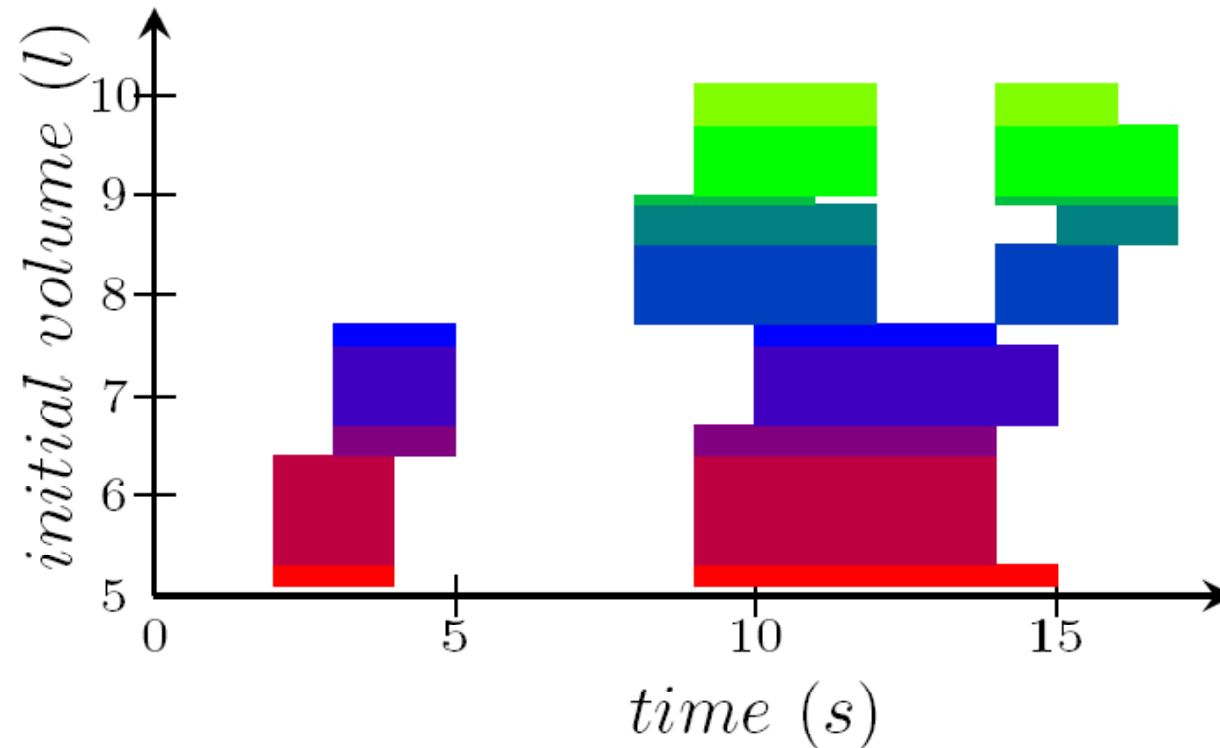
- Find some interval $I_1 = [V_1, V_2] \subseteq [4.9, 25.1]$

Queries (min. K)

control: $A \diamond (\text{time}=20 \wedge \text{not BAD} \wedge V \in I_2 \wedge V_{\text{acc}} \leq K)$

- I_1 is optimal among all m-stable intervals.

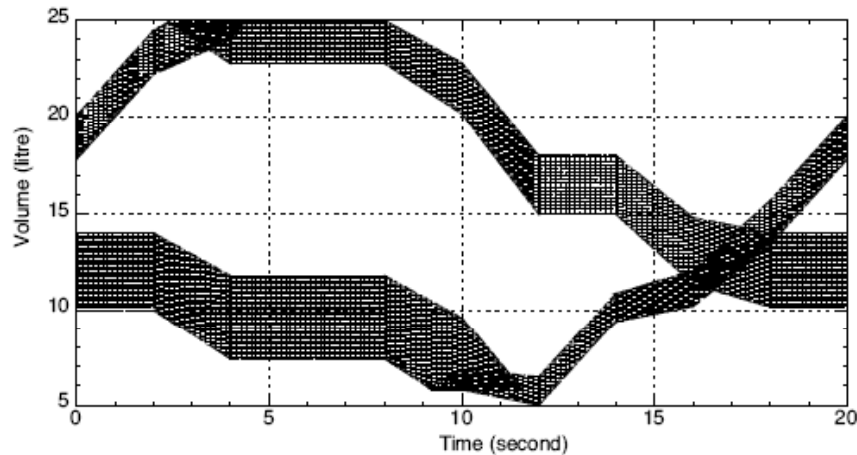
Synthesized Strategy



$D=1, m=0.4$: Optimal stable interval $I_1 = [5.1, 10]$

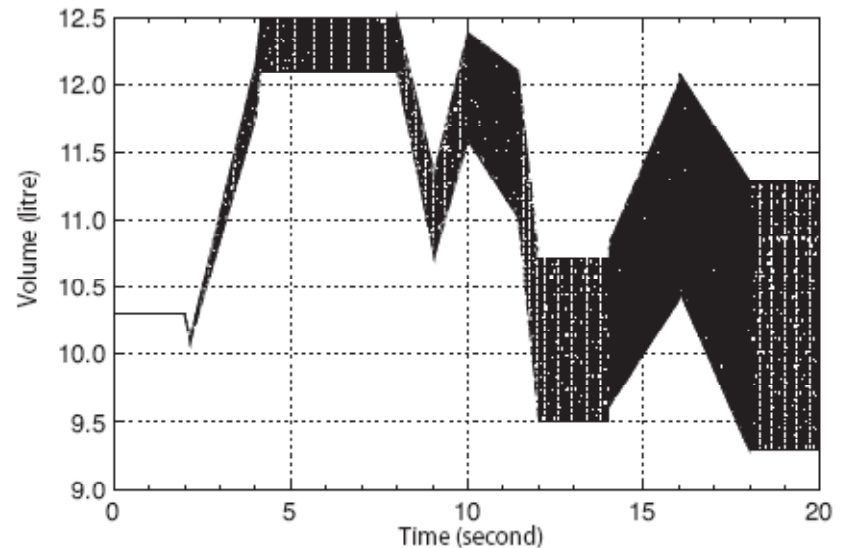
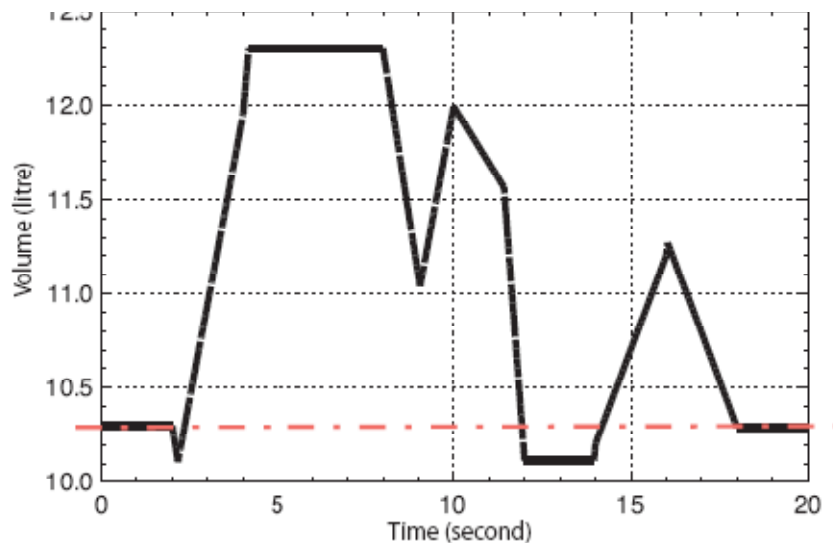


Verification Using PHAVER

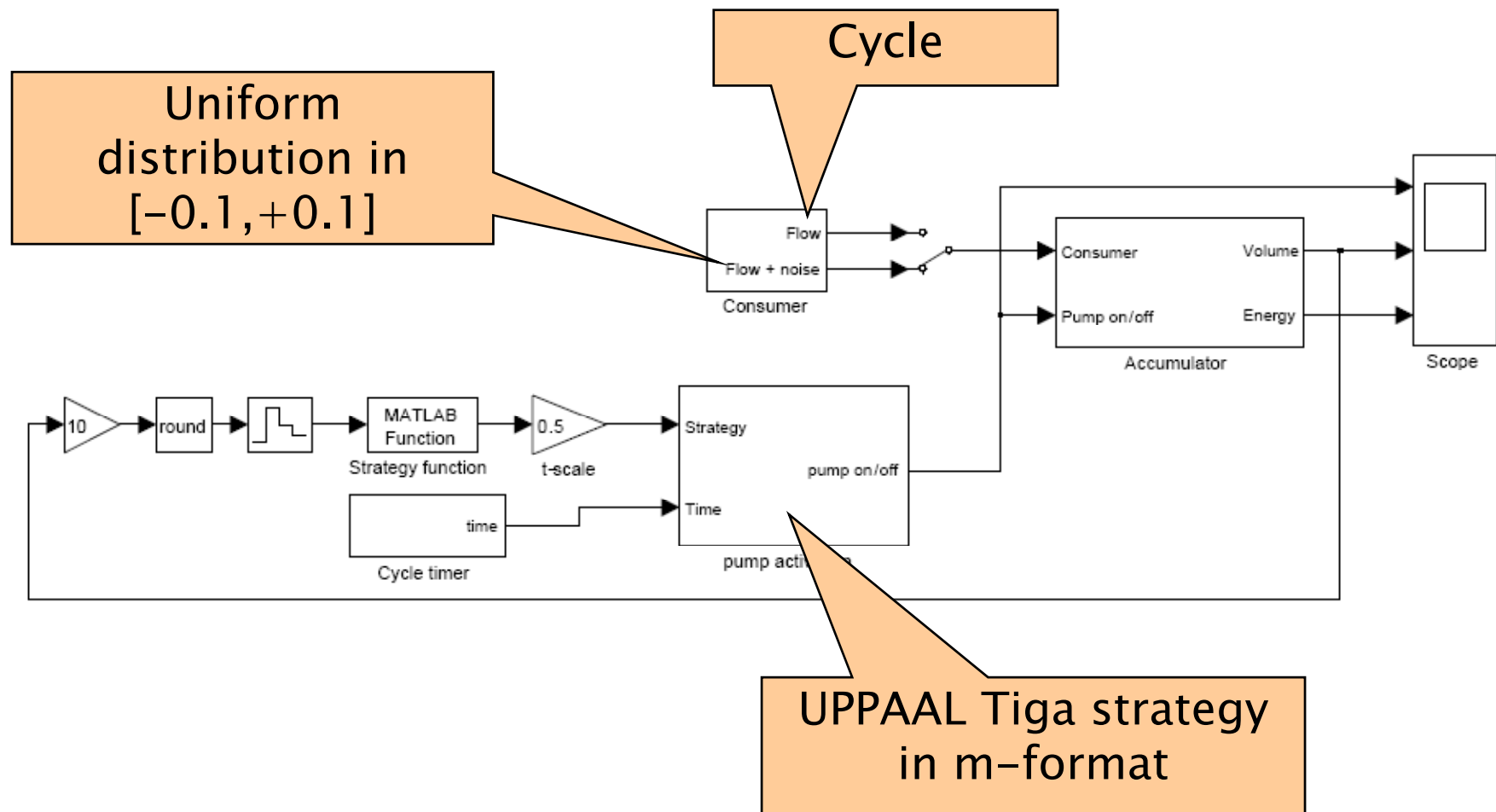


Bang-Bang safe and robust

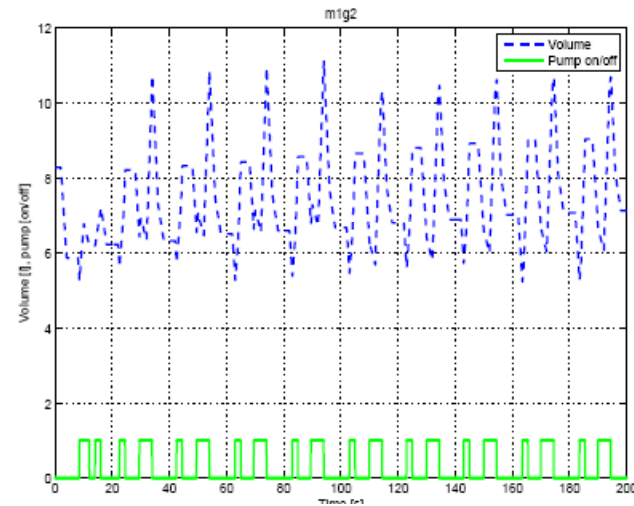
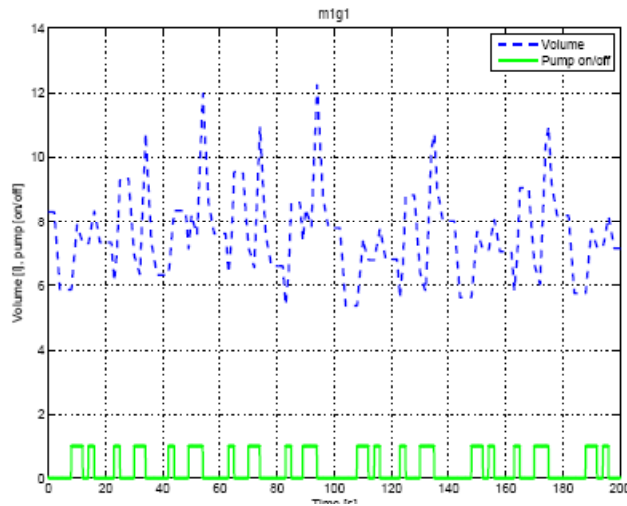
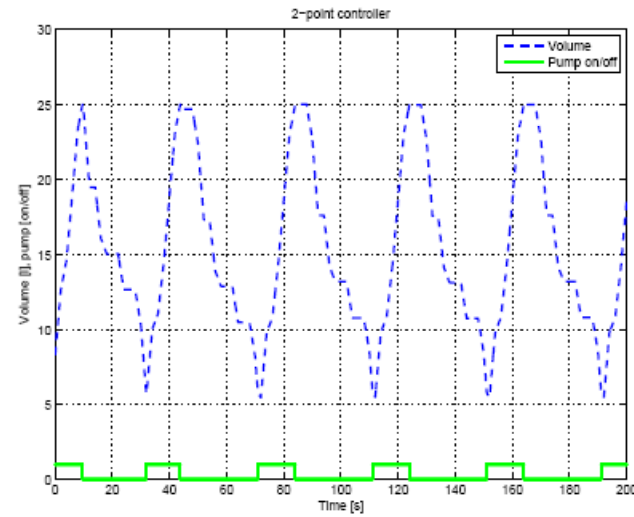
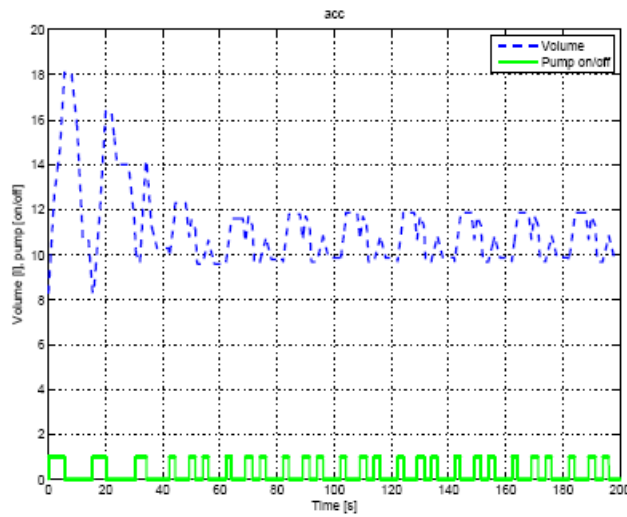
HyDAC optimized possibly unsafe under fluctuation



Performance **SIMULINK**



Results



Results

Controller	Acc. volume	Mean volume	Mean volume (TIGA)
Bang-Bang	2689	13.45	
Hy			
G1			
G1			
G1			
G1			
G2			
G2M			
G2M		7.5	7.95
G2M1	1489	7.44	7.95

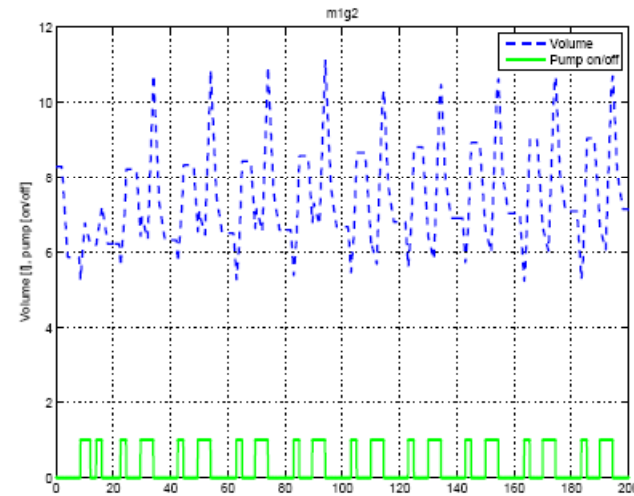
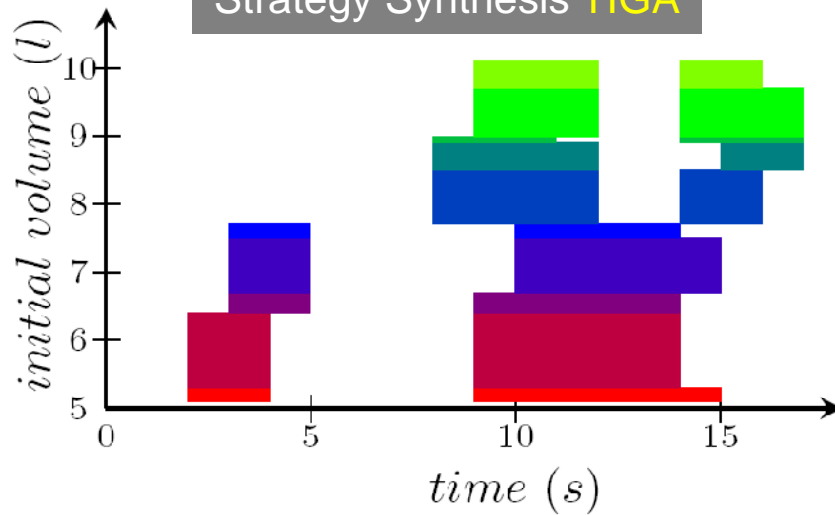
Guaranteed
Correctness
Robustness
 with
40% improvement in performance



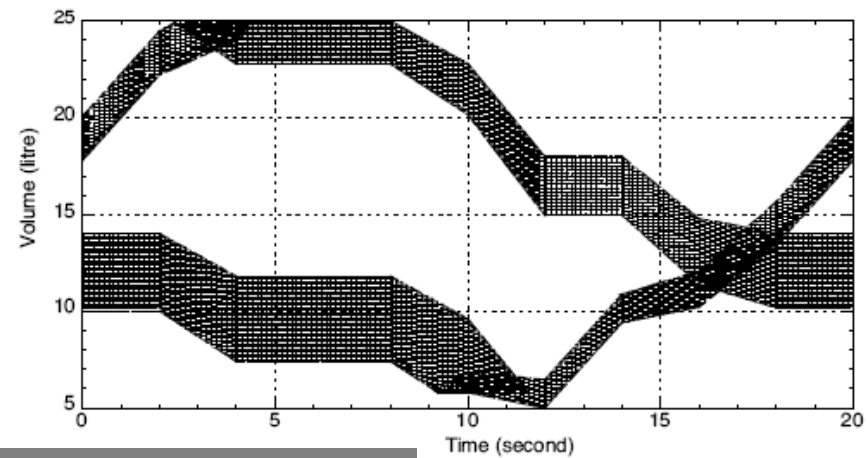
Tool Chain



Strategy Synthesis **TIGA**



Performance Evaluation **SIMULINK**



Verification **PHAVER**

Guaranteed
Correctness
Robustness
with
40% Improvement



What else ? What next ?

- Timed Games w Partial Observability
 - Action-based Observation: **undecidable** [BDMP03]
 - Finite-observation of states: **decidable** [CDL+07]
- Priced Timed Games:
 - Acyclic, cost non-zero: **decidable** [LTMM02] [BCFL04]
 - 1 clock: **decidable** [BLMR06]
 - >2 clocks: **undecidable** [BBR05, BBM06]
 - 2 clocks: **open**
- Energy Games:
 - Several Open Problems
 - Exponential Observers
- Climate Controller in Pig Stables [JRLD07]
- CHESS Way [Quasimodo@ESWEEK]



[BDMP03] Bouyer, D'Souza, Madhusudan, Petit. Timed control with partial observability (*CAV'03*).

[CDL+07] Cassez, David, Larsen, Lime, Raskin. Timed control with observation based and stuttering invariant strategies (*ATVA'07*).

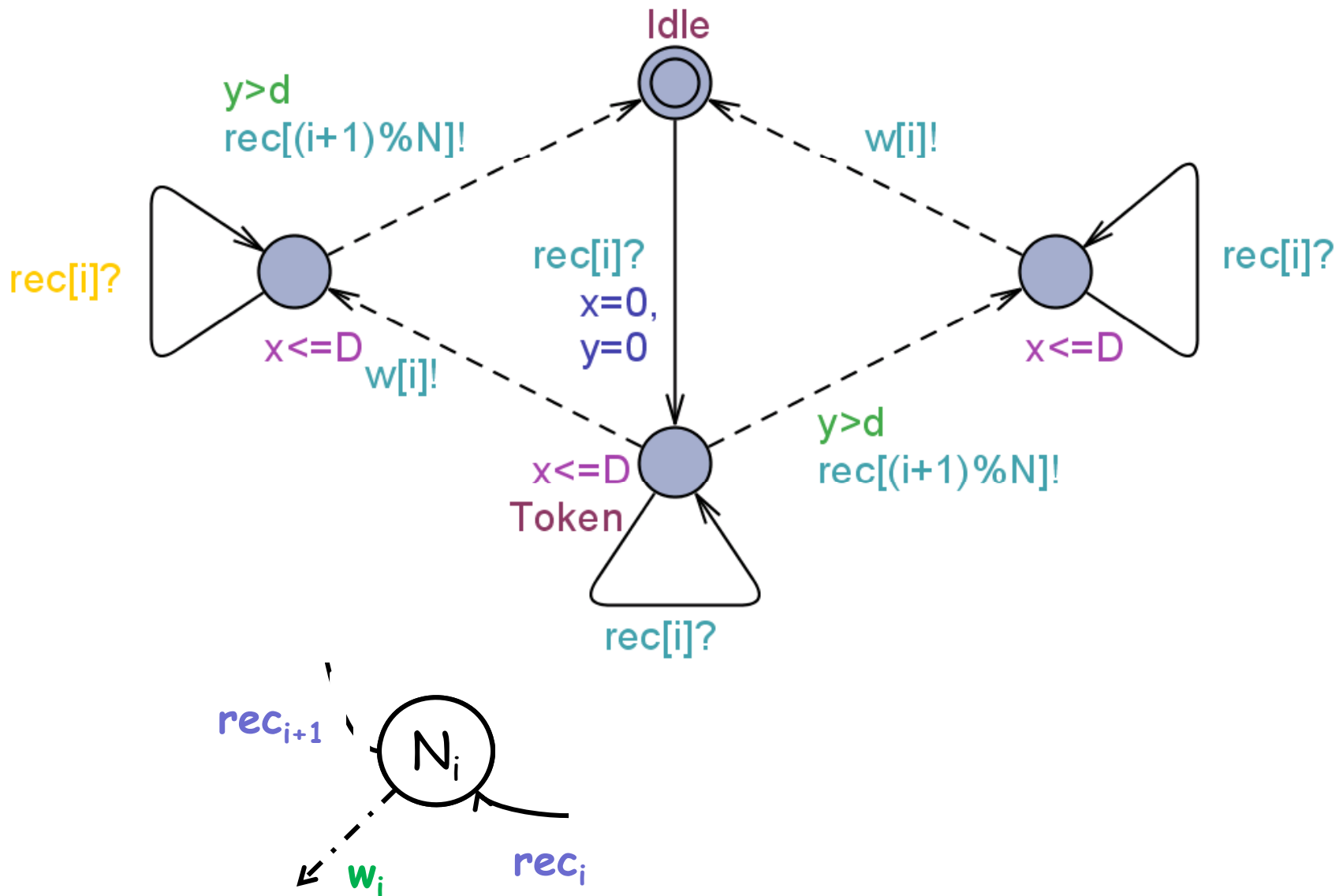
[JRLD07] Jessen, Rasmussen, Larsen, David. Guided Controller Synthesis for Climate Controller Using Uppaal Tiga (*FORMATS'07*).



Timed Interfaces & Compositionality



Real-Time version of Milner's Scheduler



Demo

C:\Documents and Settings\vg\Desktop\DESKTOP_FEB_2007\UPPAAL\UPPAAL_examples\Simple Leader Election\Winer-for-Verification.xml - UPPAAL

File Edit View Tools Options Help

Editor Simulator Verifier

Drag out

Transition chooser

0.0 6.0 12.0 18.0 24.0 30.0

rec[(9+1)%N]: Node(9) --> Node(0)

w[9]: Node(9) --> Env

Delay: 17.291 Reset

Take transition

Trace controls

First 715.995 Last

Prev Play Next

Speeder

Slow Fast

Random

Simulation Trace

```

(-, idle, idle, idle, idle, idle, idle, idle, idle
rec[0]: Starter --> Node(0)
(-, Token, Idle, Idle, Idle, Idle, Idle, Idle, I
rec[(0+1)%N]: Node(0) --> Node(1)
(-, -, Token, Idle, Idle, Idle, Idle, Idle, Idle
rec[(1+1)%N]: Node(1) --> Node(2)
(-, -, -, Token, Idle, Idle, Idle, Idle, Idle, Ic
w[2]: Node(2) --> Env
(-, -, -, -, Idle, Idle, Idle, Idle, Idle, Idle, Ic

```

Drag out

t(0) = 0
Node(0).x = 202.996685
Node(0).y = 202.996685
Node(1).x = 192.403760
Node(1).y = 192.403760
Node(2).x = 172.180987
Node(2).y = 172.180987
Node(3).x = 144.189037
Node(3).y = 144.189037
Node(4).x = 126.111239
Node(4).y = 126.111239
Node(5).x = 97.005302
Node(5).y = 97.005302
Node(6).x = 78.415511
Node(6).y = 78.415511
Node(7).x = 48.498608
Node(7).y = 48.498608
Node(8).x = 27.174265
Node(8).y = 27.174265
Node(9).x = 17.291171
Node(9).y = 17.291171
Env.x = 174.130219

Node(0) Node(1) Node(2) Node(3)

Node(4) Node(5) Node(6) Node(7)

Node(8) Node(9) Env

Gantt Chart

	0	17	33	50	66	83	99	116	132	149	165	182	198	215	231	248	264	280	297	313	330	346	363	379	396	412	429	445	462	478	495	511	528	544	560	577	593	610	626	643	659	676	692	709	725					
Node(0)																																																		
Node(1)																																																		
Node(2)																																																		
Node(3)																																																		
Node(4)																																																		
Node(5)																																																		
Node(6)																																																		
Node(7)																																																		
Node(8)																																																		
Node(9)																																																		

Compositional Verification

The screenshot displays the UPPAAL simulator interface for a system with 10 nodes (Node(0) to Node(9)) and an environment (Env). The nodes are arranged in a grid, with three sub-specifications highlighted in red:

- SubSpec₁**: Nodes 0, 1, 2, and 3.
- SubSpec₂**: Nodes 4, 5, 6, and 7.
- SubSpec₃**: Nodes 8 and 9.

The environment (Env) is shown in a separate window. The Gantt Chart at the bottom shows the execution timeline for each node, with time steps from 0 to 725. The simulation trace on the left shows the sequence of events, including transitions between nodes and the environment.

Transition chooser:

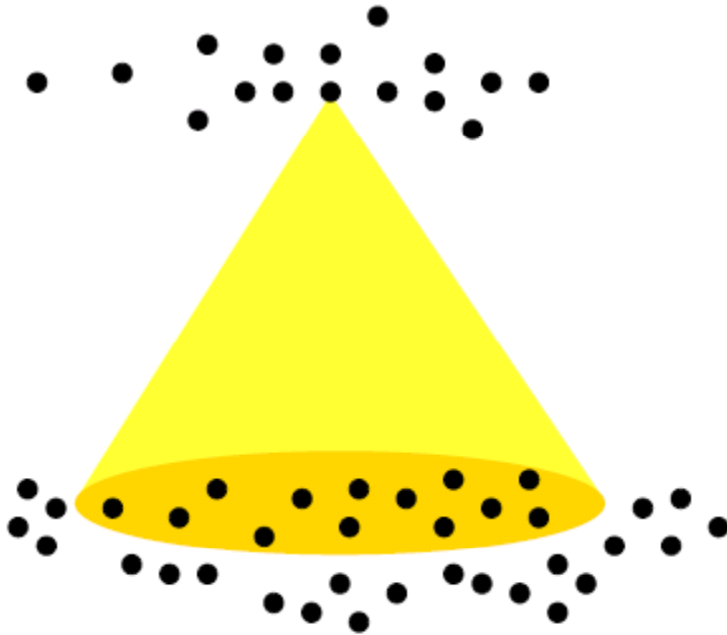
```
0,0 6,0 12,0 18,0 24,0 30,0
rec[(9+1)%N]: Node(9) --> Node(0)
w[9]: Node(9) --> Env
```

Simulation Trace:

```
<-, Idle, Idle, Idle, Idle, Idle, Idle, Idle, Idle
rec[0]: Starter --> Node(0)
<-, Token, Idle, Idle, Idle, Idle, Idle, Idle, I
rec[(0+1)%N]: Node(0) --> Node(1)
<-, -, Token, Idle, Idle, Idle, Idle, Idle, Idle
rec[(1+1)%N]: Node(1) --> Node(2)
<-, -, -, Token, Idle, Idle, Idle, Idle, Idle, Ic
w[2]: Node(2) --> Env
<-, -, -, -, Idle, Idle, Idle, Idle, Idle, Idle, Ic
```

Specification Theory

Spec: set of specifications



Imp: set of implementations

Specification Formalism

$$\text{SPF} = (\text{Imp}, \text{Spec}, \text{sat})$$

where

$$\text{sat} \subseteq \text{Imp} \times \text{Spec}$$

$$|S| = \{ I : I \text{ sat } S \}$$

Refinement:

$$S \leq T \text{ iff } |S| \subseteq |T|$$

Consistency:

$$|S| \neq \emptyset$$

$$|S| \cap |T| \neq \emptyset$$

Operations on Specifications

- **Logical Conjunction:**

- Given S_1 and S_2 construct $S_1 \wedge S_2$ such that
$$|S_1 \wedge S_2| = |S_1| \cap |S_2|$$

- **Structural Composition:**

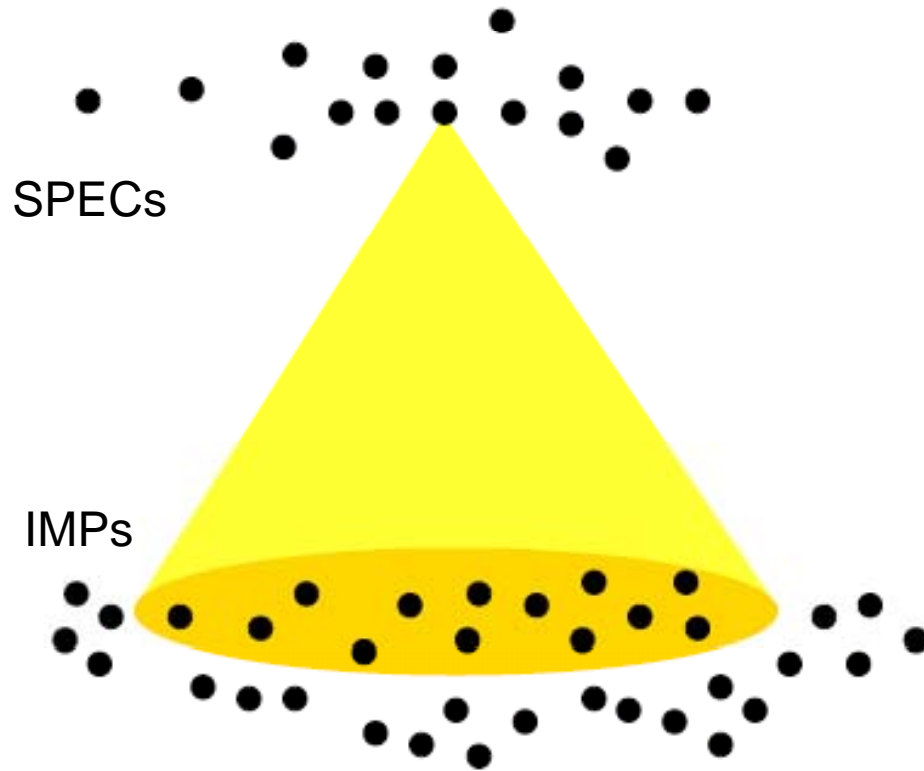
- Given S_1 and S_2 construct $S_1 \text{ par } S_2$ such that
$$|S_1 \text{ par } S_2| = |S_1| \text{ par } |S_2|$$
- \leq should be precongruence wrt **par** to allow for compositional analysis !

- **Quotienting:**

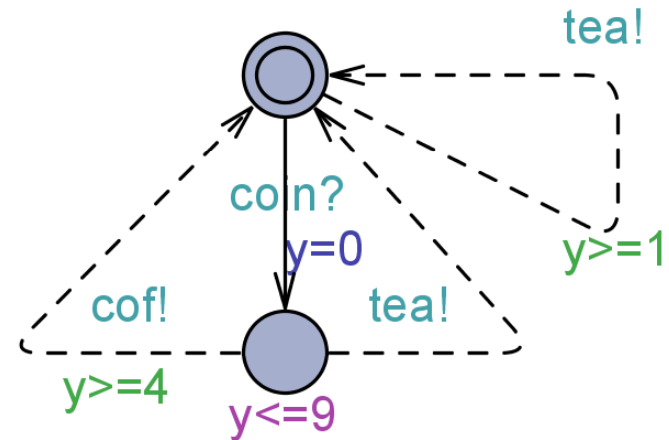
- Given overall specification T and component specification S construct the quotient specification $T \setminus S$ such that
$$S \text{ par } X \leq T \quad \text{iff} \quad X \leq T \setminus S$$

Specifications and Implementations

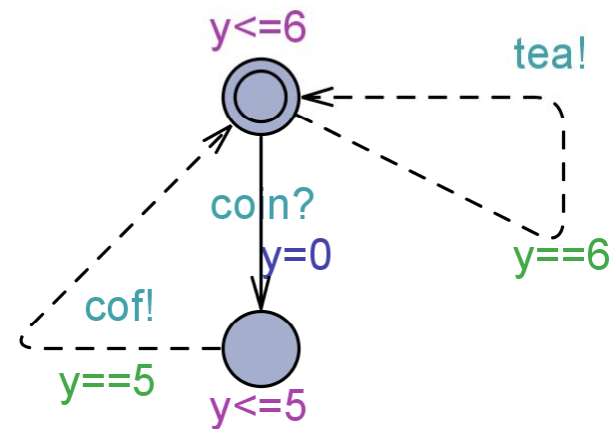
Timed I/O Transition Systems



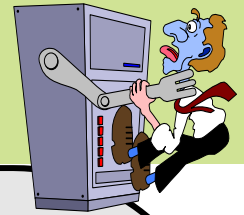
Timed I/O Transition Systems



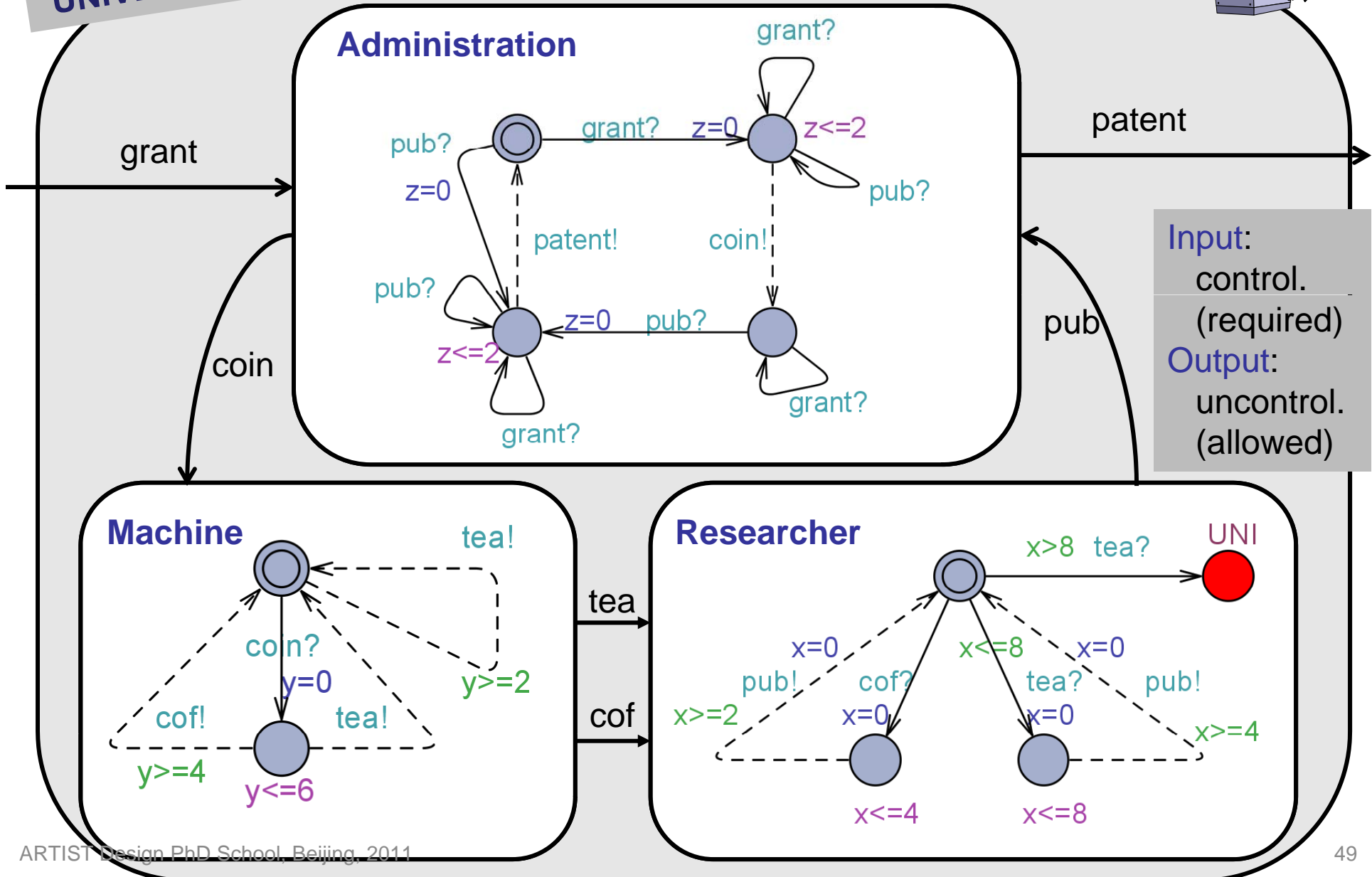
Timed I/O Automata



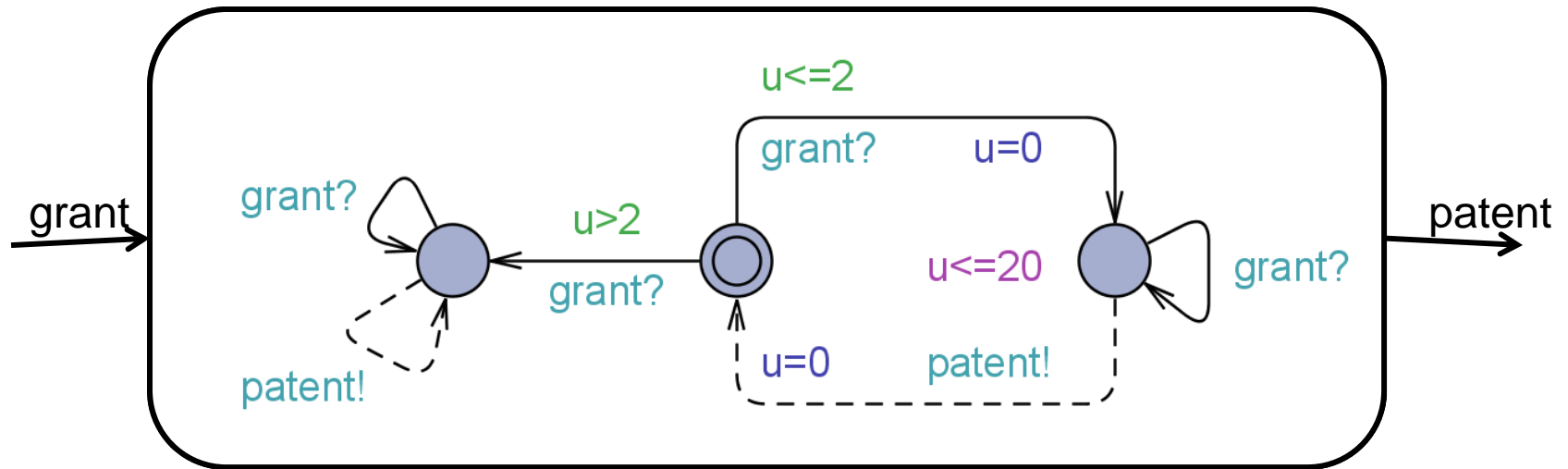
Timed Systems Specifications = Timed I/O Automata



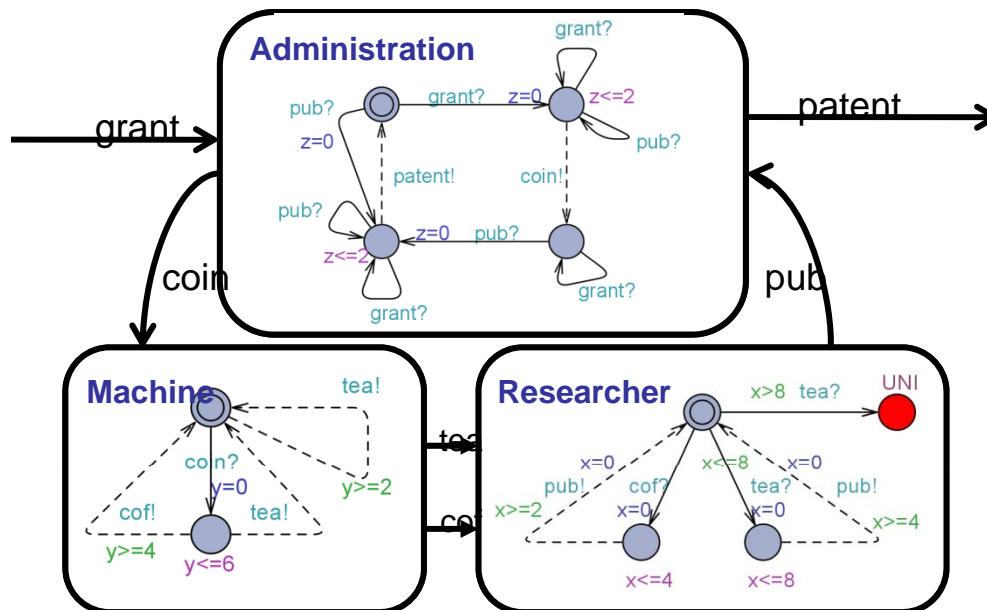
UNIVERSITY



Overall Specification



\bigvee
?



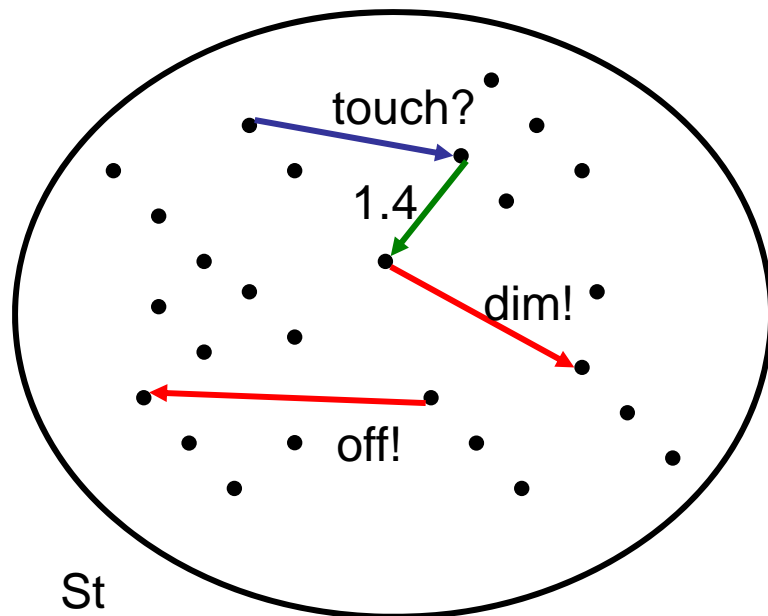
Timed I/O Transition Systems

TIOTS:

(St, Act, \rightarrow)

where $\rightarrow: St \times (Act \cup \mathbb{R}) \times St$

and $Act = \Sigma_i \cup \Sigma_o$



Time determinism ($d \in \mathbb{R}$)

if $s \xrightarrow{d} s'$ and $s \xrightarrow{d} s''$ then $s' = s''$

Input enabledness

for all s and $i \in \Sigma_i$. $s \xrightarrow{i}$

Determinism ($a \in Act \cup \mathbb{R}$)

if $s \xrightarrow{a} s'$ and $s \xrightarrow{a} s''$ then $s' = s''$

Output Urgency

whenever $s \xrightarrow{o}$

then $s \xrightarrow{d}$ implies $d = 0$

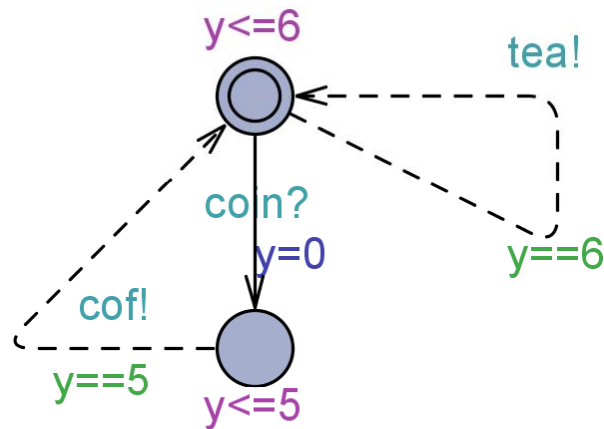
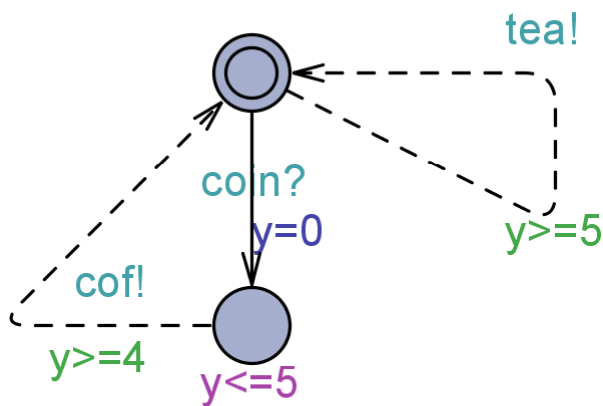
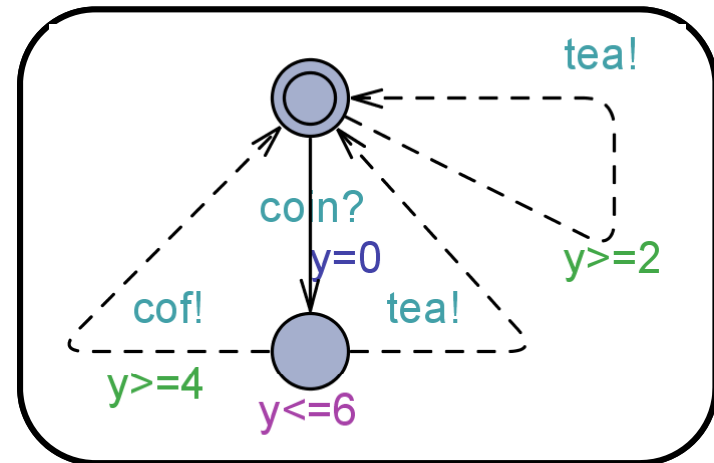
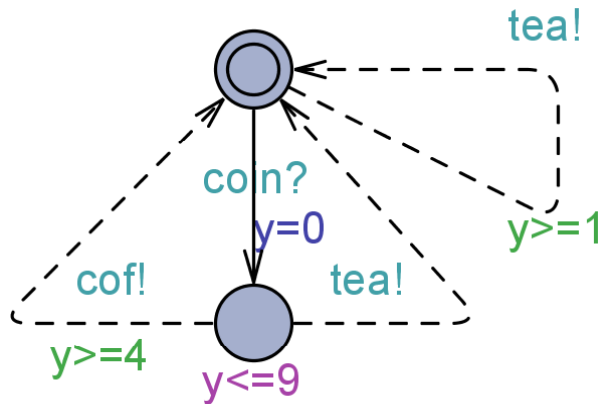
Independent Progress

Either $\forall d \geq 0$. $s \xrightarrow{d}$

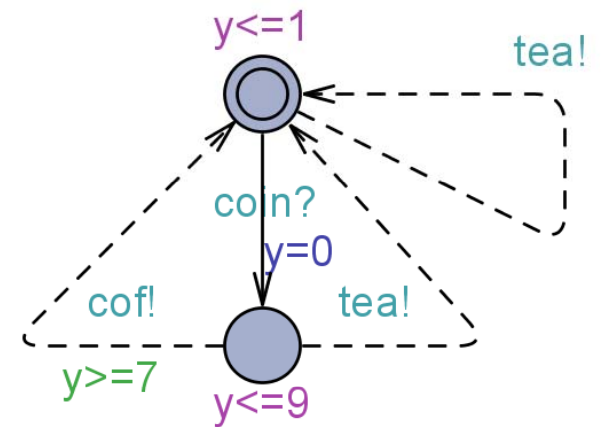
or $\exists d, o$. $s \xrightarrow{d} \xrightarrow{o}$

Implementations

Refinements, Implementations, Consistency



An Implementation



Inconsistent

Refinement = Timed Alternating Simulation

Let S and T be TIOGA.

$S \prec T$ iff

- i. $T \xrightarrow{i?} T'$ then $S \xrightarrow{i?} S'$ with $S' \leq T'$
- ii. $S \xrightarrow{o!} S'$ then $T \xrightarrow{o!} T'$ with $S' \leq T'$
- iii. $S \xrightarrow{d} S'$ then $T \xrightarrow{d} T'$ with $S' \leq T'$

Intuition:

S leaves less choices than T
for an implementation.

Definition:

$$I \text{ sat } S \Leftrightarrow^{\Delta} I \leq S$$

Theorem

Whenever $S \leq T$ then $|S| \subseteq |T|$

Theorem:

Whenever $\emptyset \neq |S| \subseteq |T|$ then $S \leq T$

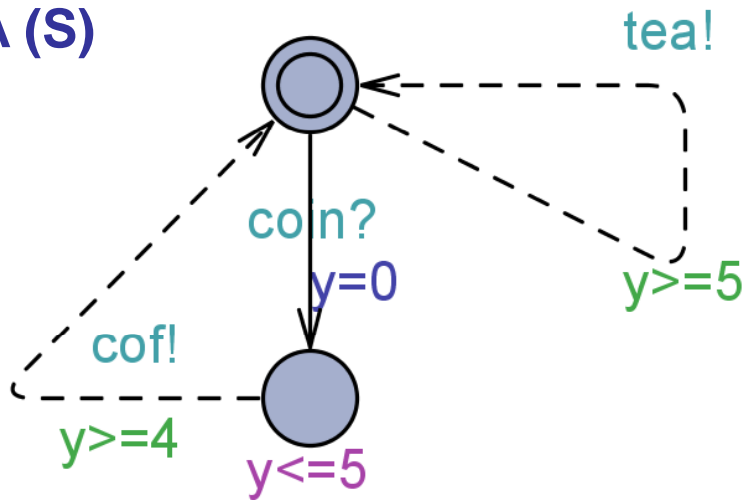
Theorem

$S \leq T \Rightarrow$

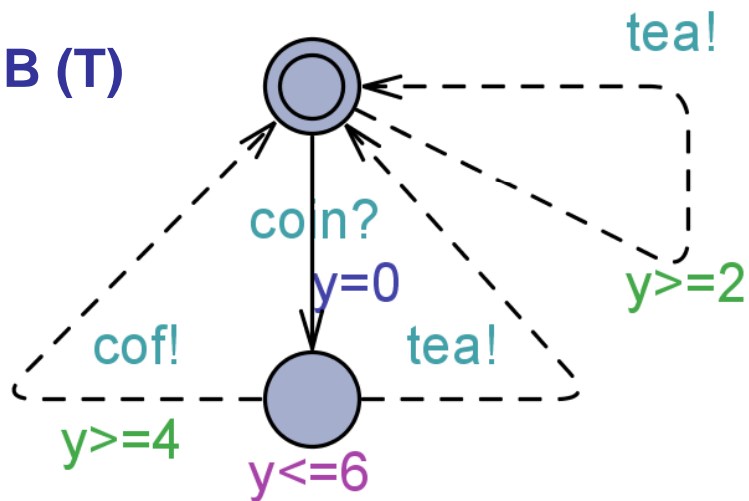
$(\forall \Phi \in \text{ATCTL}. T \text{ cntr } \Phi \Rightarrow S \text{ cntr } \Phi)$

Refinement (example)

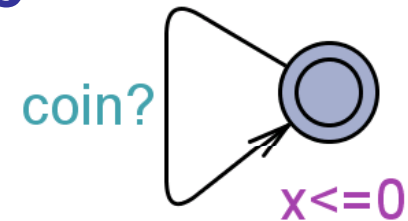
A (S)



B (T)



INC

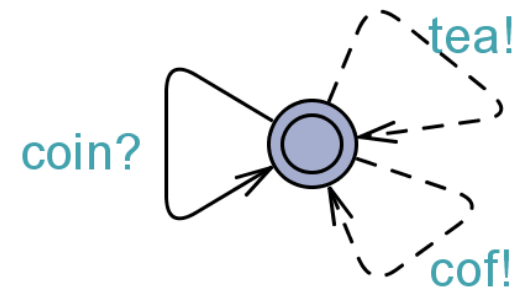


Let S and T be TIOGA.

$S \prec T$ iff

- i. $T \xrightarrow{i?} T'$ then $S \xrightarrow{i?} S'$ with $S' \leq T'$
- ii. $S \xrightarrow{o!} S'$ then $T \xrightarrow{o!} T'$ with $S' \leq T'$
- iii. $S \xrightarrow{d} S'$ then $T \xrightarrow{d} T'$ with $S' \leq T'$

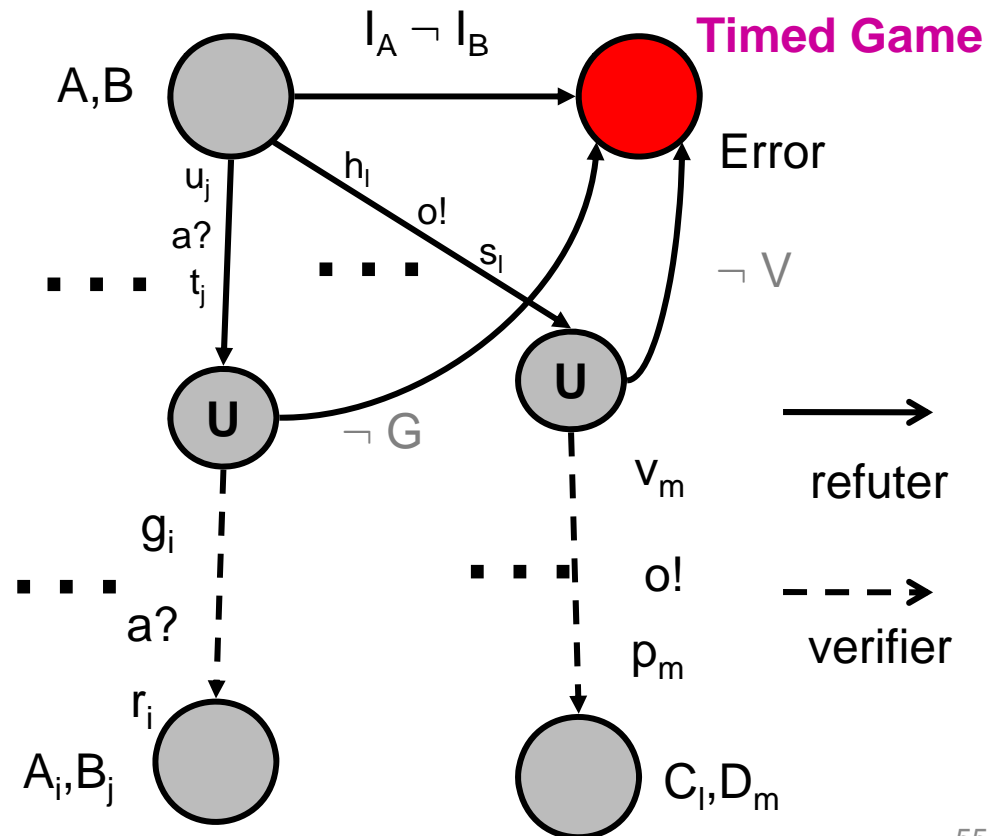
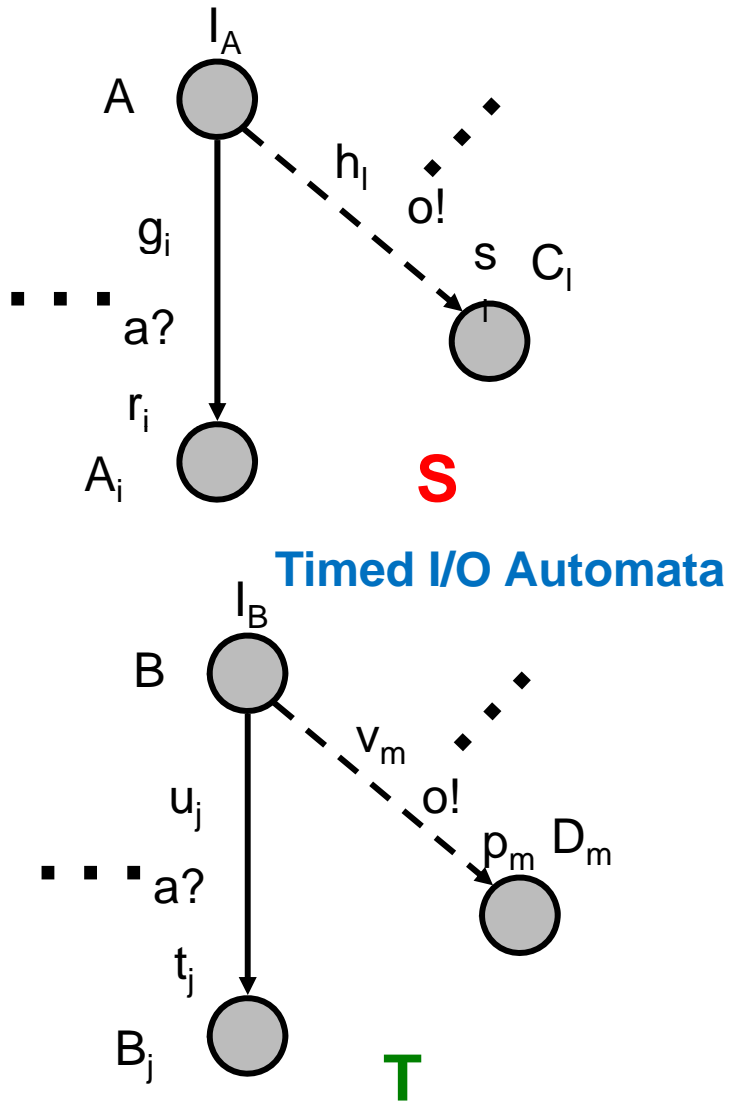
UNI



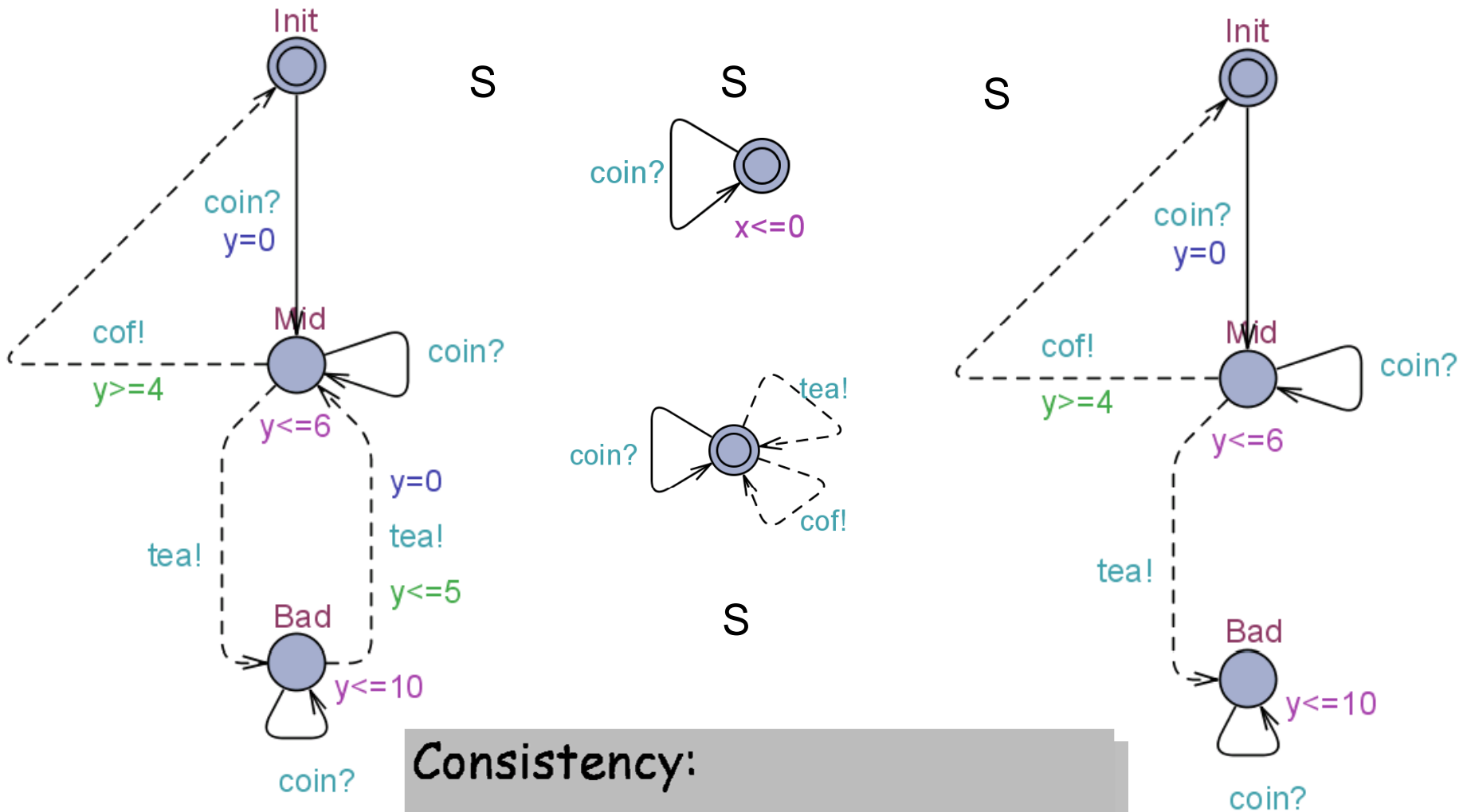
Refinement as a Game

FORMATS09
Optimized Refinement Algorithm

not $A \leq B$
iff
 $A \times B$ sat control: $A \leftrightarrow \text{Error}$

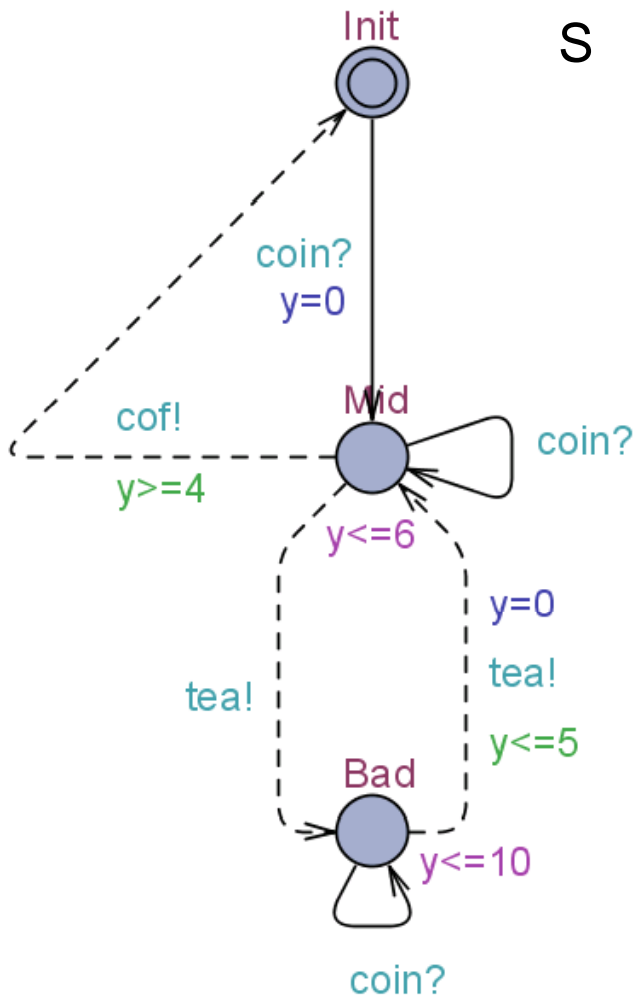


Consistency



Consistency:
Does there exist I such that
 $I \leq S$?

Consistency



Definitions

Err =

$$\{ s \mid \forall d > 0. \neg s \xrightarrow{d} \wedge \forall o. \neg s \xrightarrow{o} \}$$

$\pi(X) =$

Err \cup

$\text{Pred}_t[X \cup \text{iPred}(X) , \text{oPred}(X^c)]$

Theorem

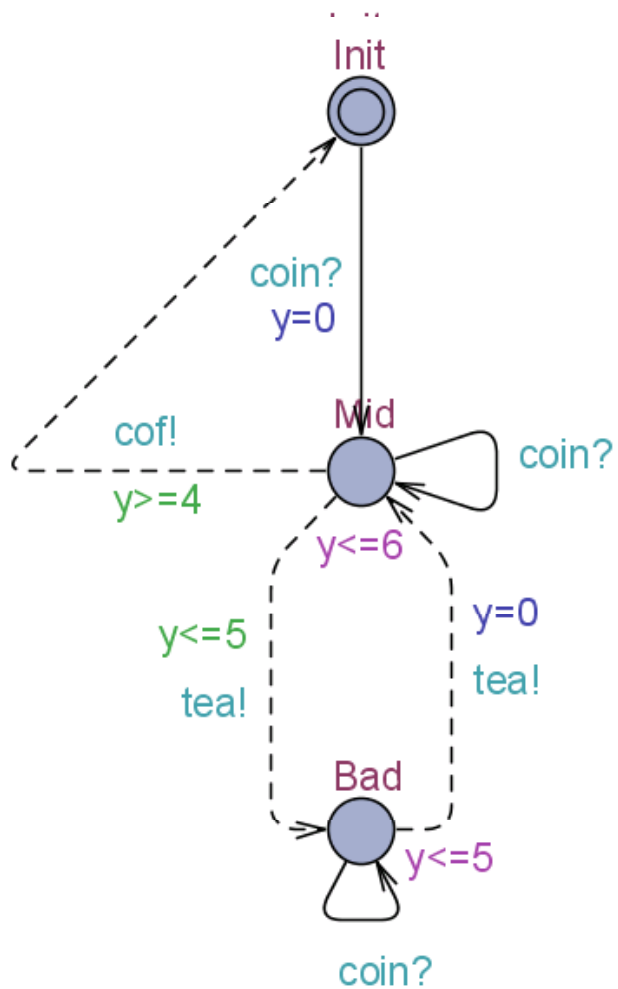
A specification (state) s is

inconsistent

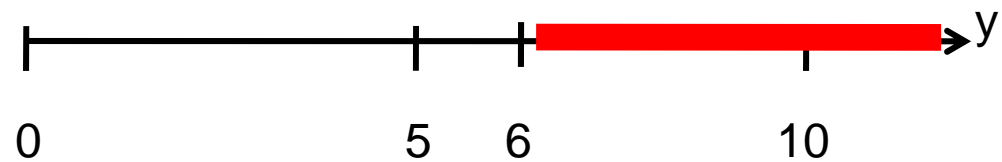
iff

$$s \in \mu X. \pi(X)$$

Consistency



Pruned Version



$$\pi(X) = \text{Err} \cup \text{Pred}_t[X \cup \text{iPred}(X) , \text{oPred}(X^c)]$$

$$\text{Err} = \{ s \mid \forall d > 0. \neg s \xrightarrow{d} \wedge \forall o. \neg s \xrightarrow{o} \}$$

Consistency in UPPAAL Tiga+

The image displays two windows from the UPPAAL Tiga+ environment. The left window is a WordPad document titled 'out - WordPad' containing the following text:

```
Maximal Output Strategy to avoid states
without Indendent Progress:
=====

State: ( PartInc.Mid )
When you are in (PartInc.y<=5),
    take transition PartInc.Mid->PartInc.Bad ( 1, tea!, 1 )
When you are in (4<=PartInc.y && PartInc.y<=6),
    take transition PartInc.Mid->PartInc.Init ( y >= 4, cof!, 1 )

State: ( PartInc.Init )
While you are in true,
    wait.

State: ( PartInc.Bad )
When you are in (PartInc.y<=5),
    take transition PartInc.Bad->PartInc.Mid ( y <= 5, tea!, y := 0 )
```

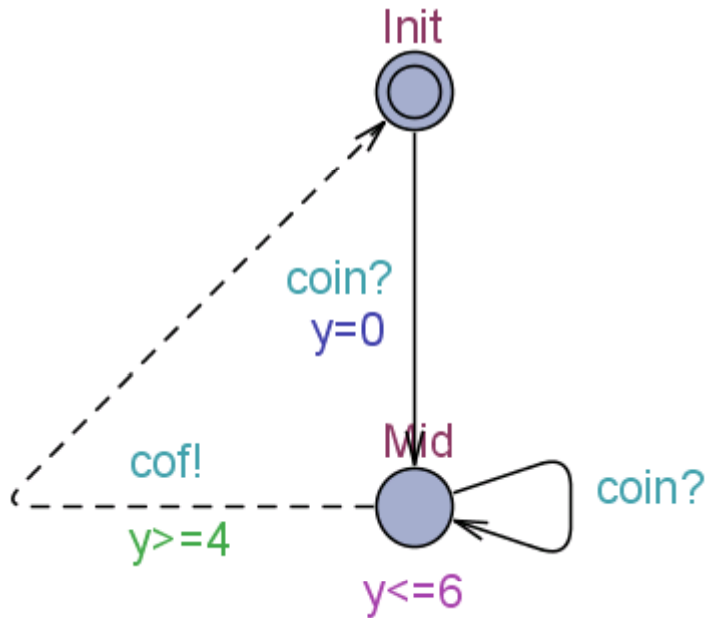
The right window is the UPPAAL GUI, showing a state transition diagram for the 'PartInc' process. The diagram features three states: 'Init' (a grey circle), 'Mid' (a red circle), and 'Bad' (a grey circle). Transitions are labeled with events and guards:

- From 'Init' to 'Mid': event 'coin?', guard 'y=0'.
- From 'Mid' to 'Init': event 'cof!', guard 'y>=4'.
- From 'Mid' to 'Bad': event 'tea!', guard 'y<=6'.
- From 'Bad' to 'Mid': event 'tea!', guard 'y<=5'.
- From 'Bad' to 'Bad': event 'coin!', guard 'y<=10'.
- From 'Mid' to 'Mid': event 'coin?' (self-loop).

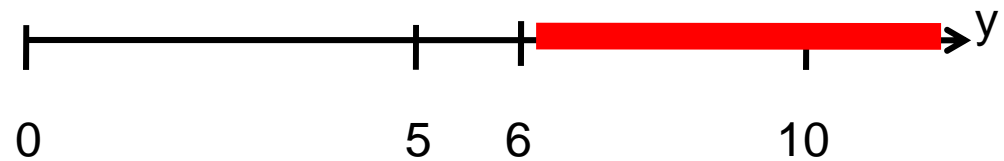
Below the GUI window, the text 'GUI' is written.

Command-Line

Consistency



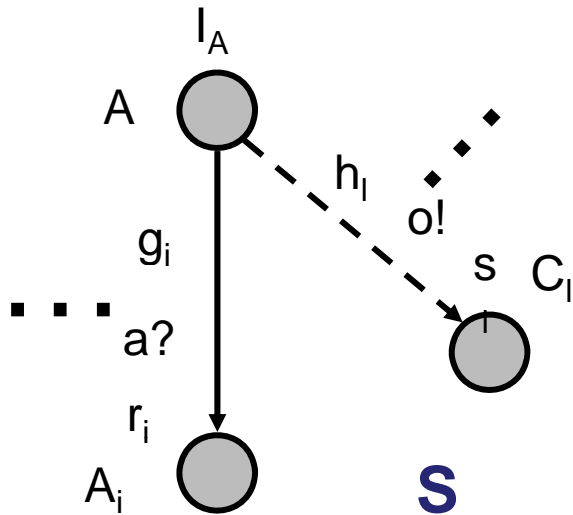
Pruned Version



$$\pi(X) = \text{Err} \cup \text{Pred}_t[X \cup \text{iPred}(X) , \text{oPred}(X^c)]$$

$$\text{Err} = \{ s \mid \forall d > 0. \neg s \xrightarrow{d} \wedge \forall o. \neg s \xrightarrow{o} \}$$

Conjunction, $S \wedge T$

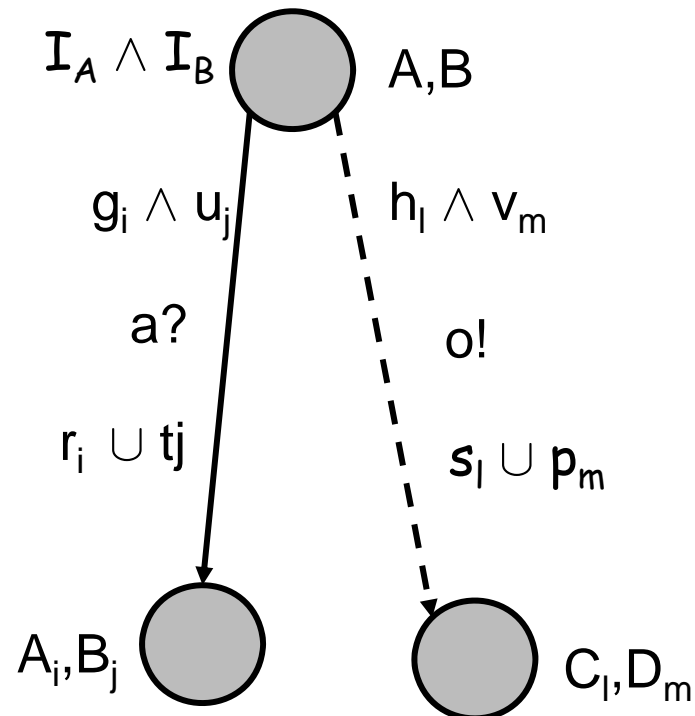
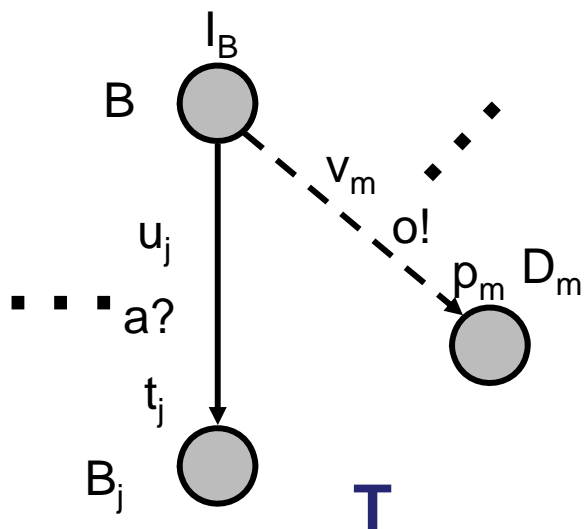


Theorem

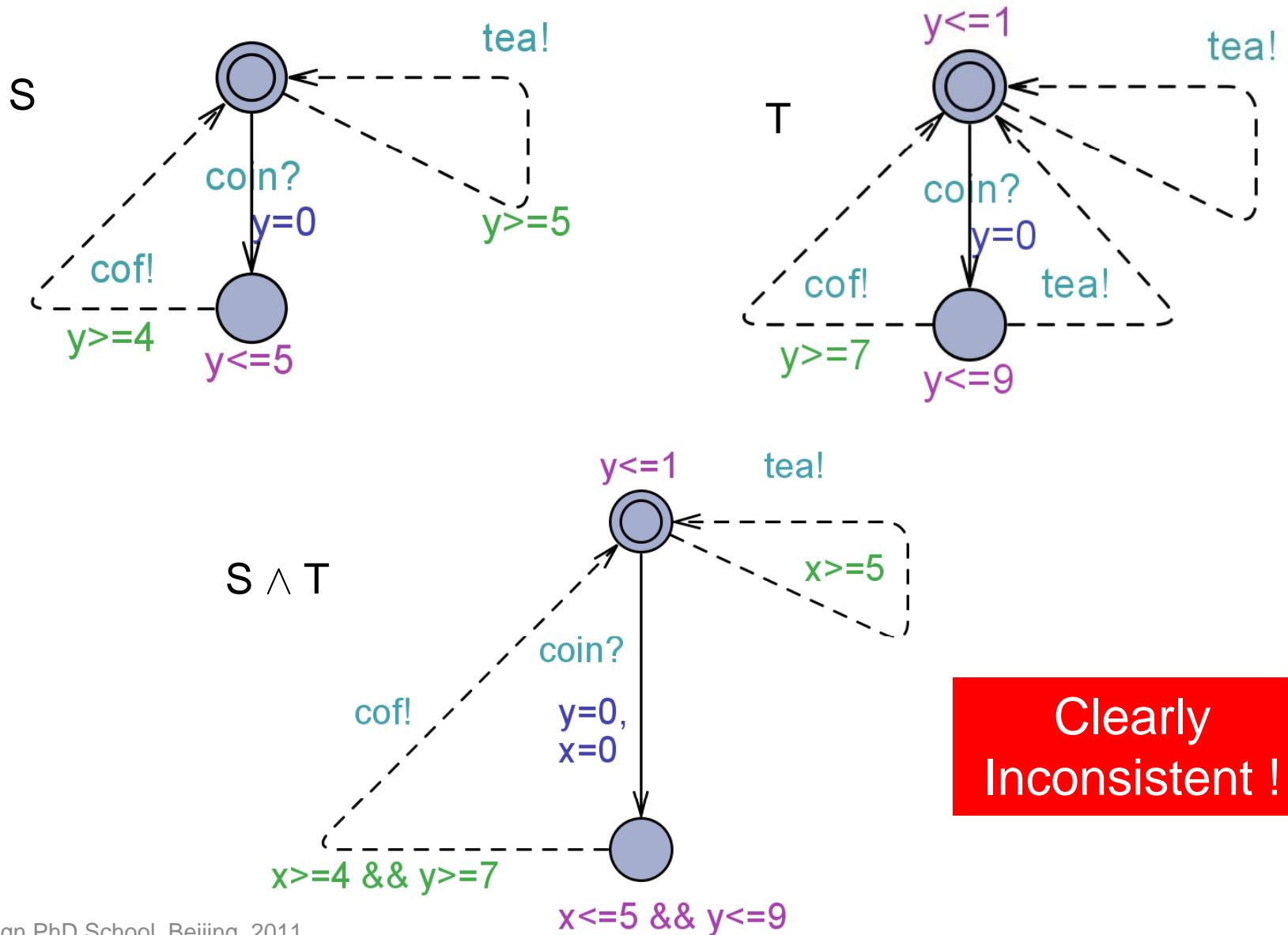
$$S \wedge T \leq S$$

$$S \wedge T \leq T$$

$$(U \leq S) \text{ and } (U \leq T) \Rightarrow U \leq (S \wedge T)$$



Conjunction, Ex.



Clearly Inconsistent !

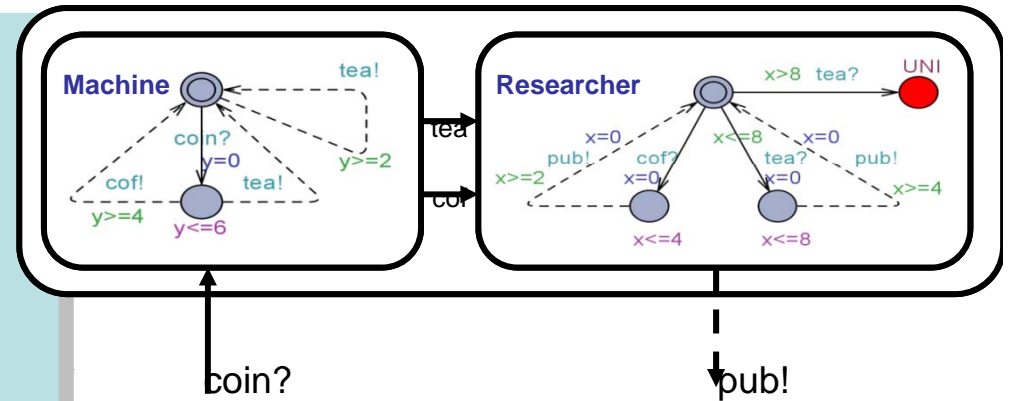
Composition, S|T

$$\frac{s_1 \xrightarrow{i?} s_1'}{s_1 | s_2 \xrightarrow{i?} s_1' | s_2} \quad i \in \Sigma_i^1 - \Sigma_0^2$$

$$\frac{s_1 \xrightarrow{o!} s_1'}{s_1 | s_2 \xrightarrow{o!} s_1' | s_2} \quad o \in \Sigma_o^1 - \Sigma_i^2$$

$$\frac{s_1 \xrightarrow{a!} s_1' \quad s_2 \xrightarrow{a?} s_2'}{s_1 | s_2 \xrightarrow{a!} s_1' | s_2'} \quad a \in \Sigma_o^1 \cap \Sigma_i^2$$

$$\frac{s_1 \xrightarrow{d} s_1' \quad s_2 \xrightarrow{d} s_2'}{s_1 | s_2 \xrightarrow{d} s_1' | s_2'} \quad d \in \mathcal{R}$$



Theorem

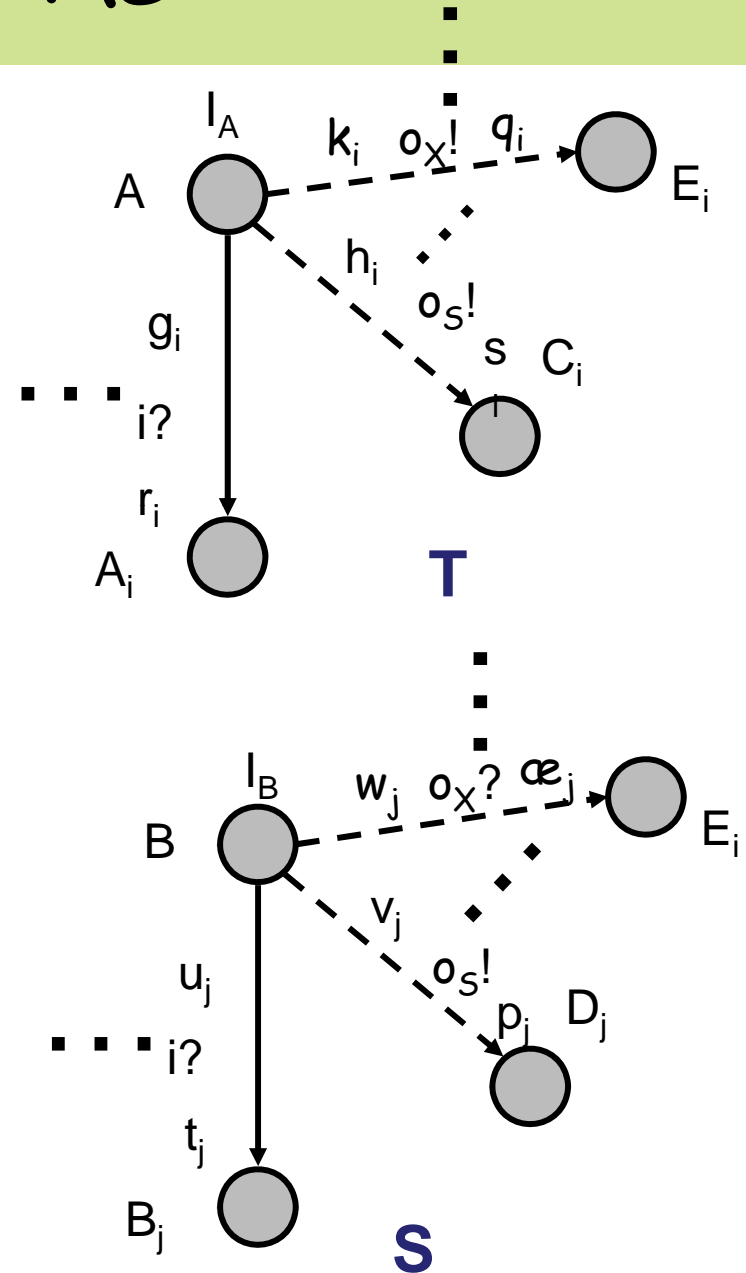
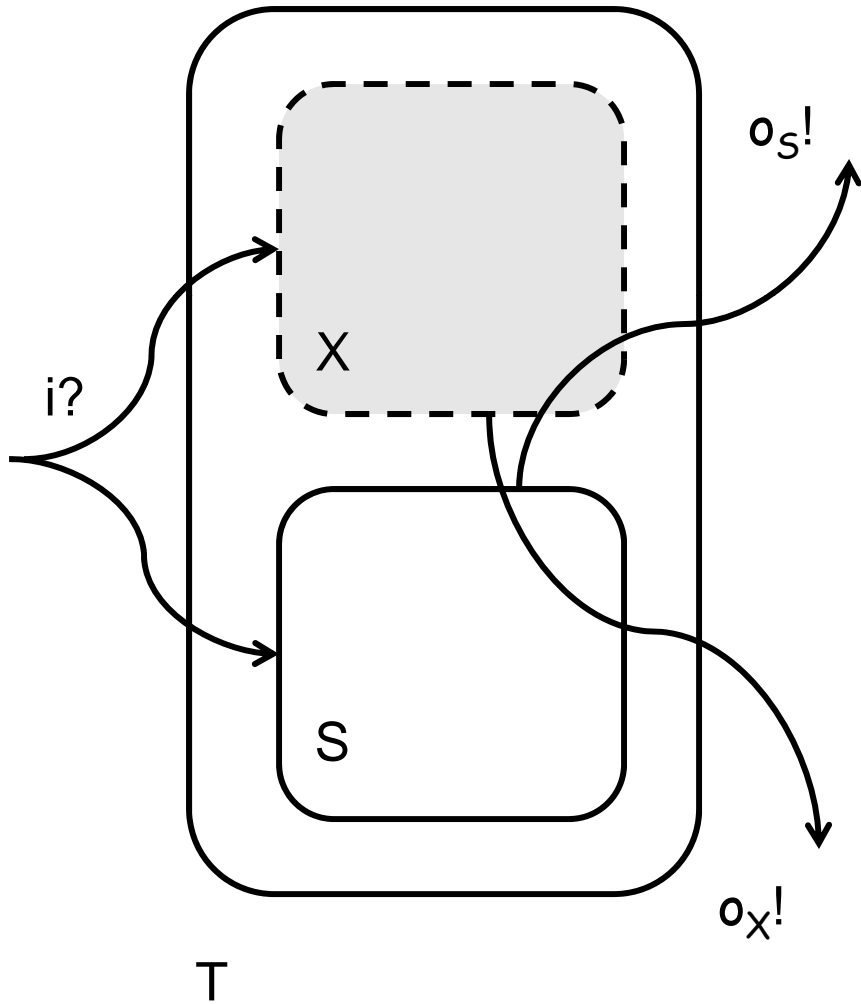
If $A_1 \leq B_1$ and $A_2 \leq B_2$

then

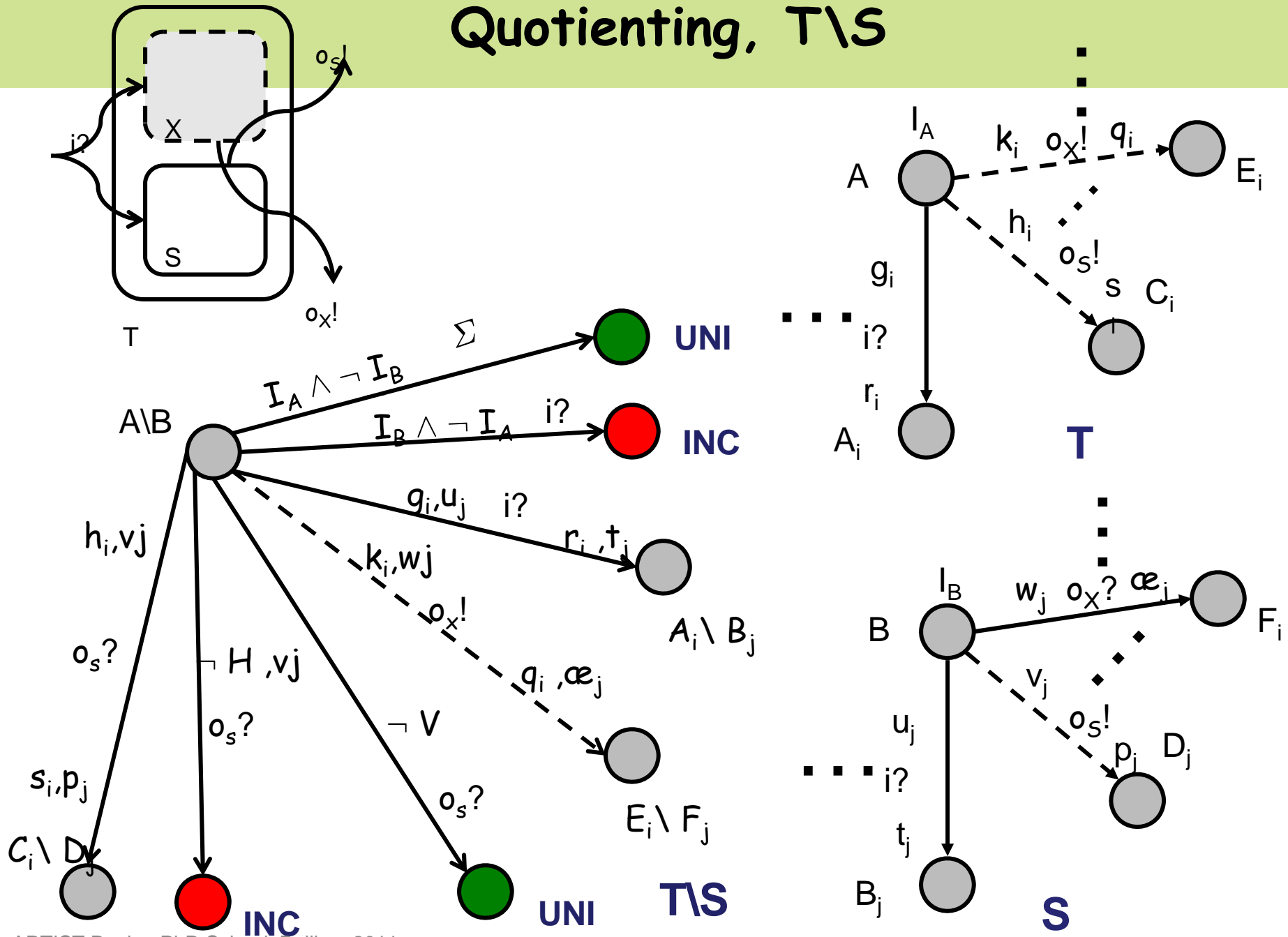
$$A_1 | A_2 \leq B_1 | B_2$$

Classical rules for
Composition of I/O transition
Systems

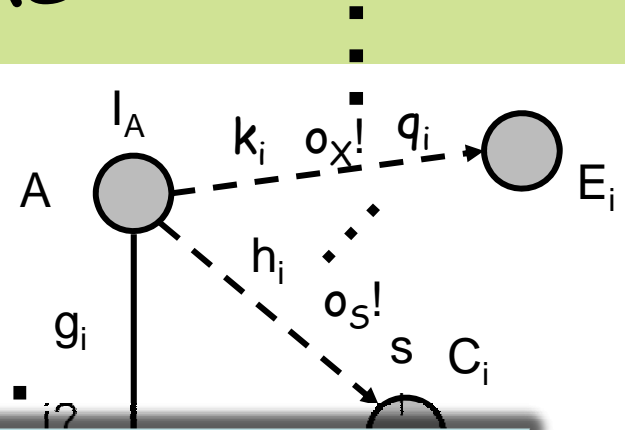
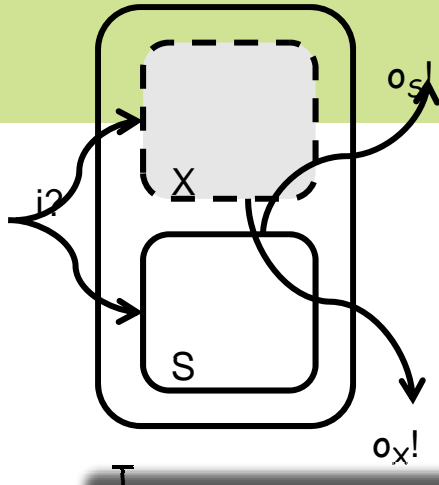
Quotienting, $T \setminus S$



Quotienting, $T \setminus S$

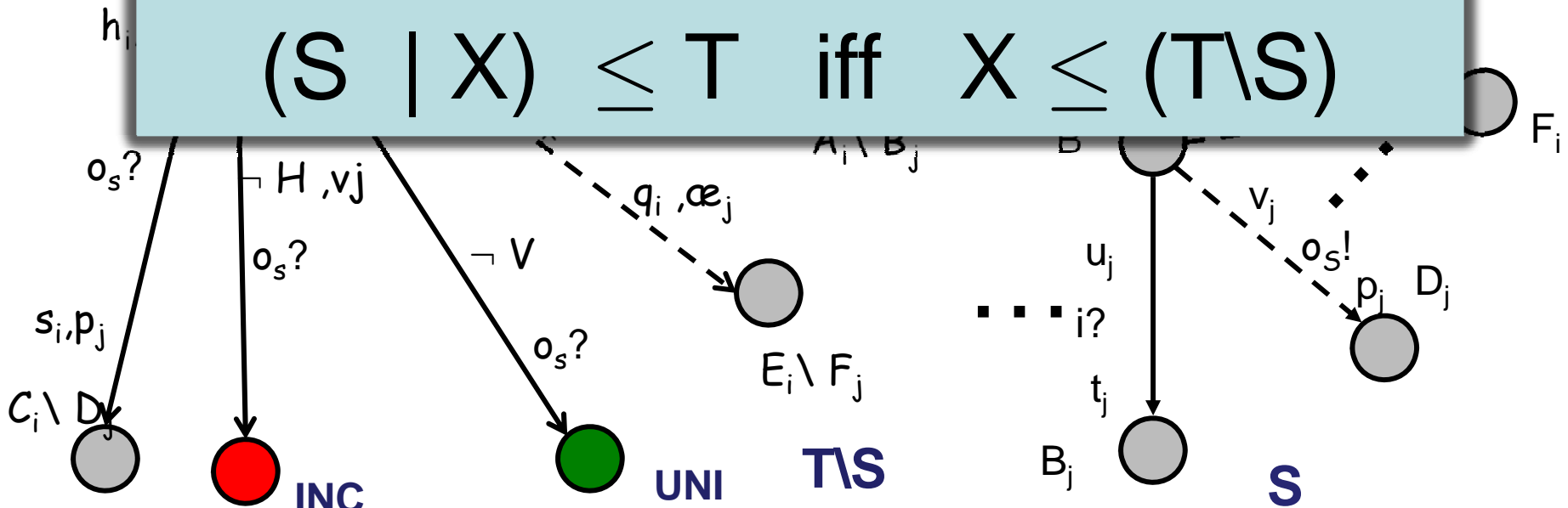


Quotienting, $T \setminus S$

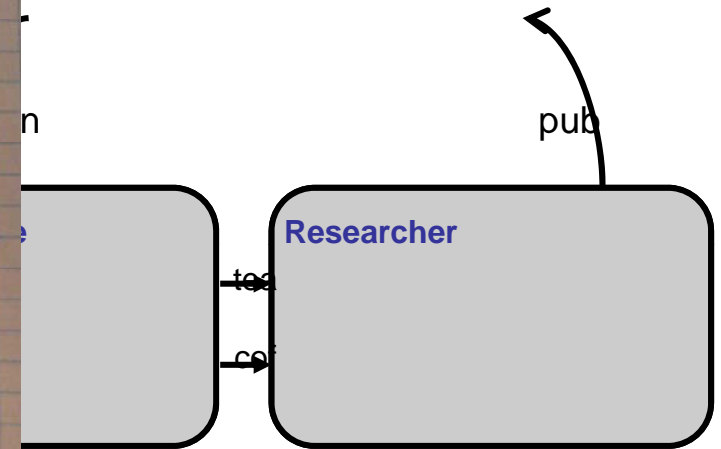
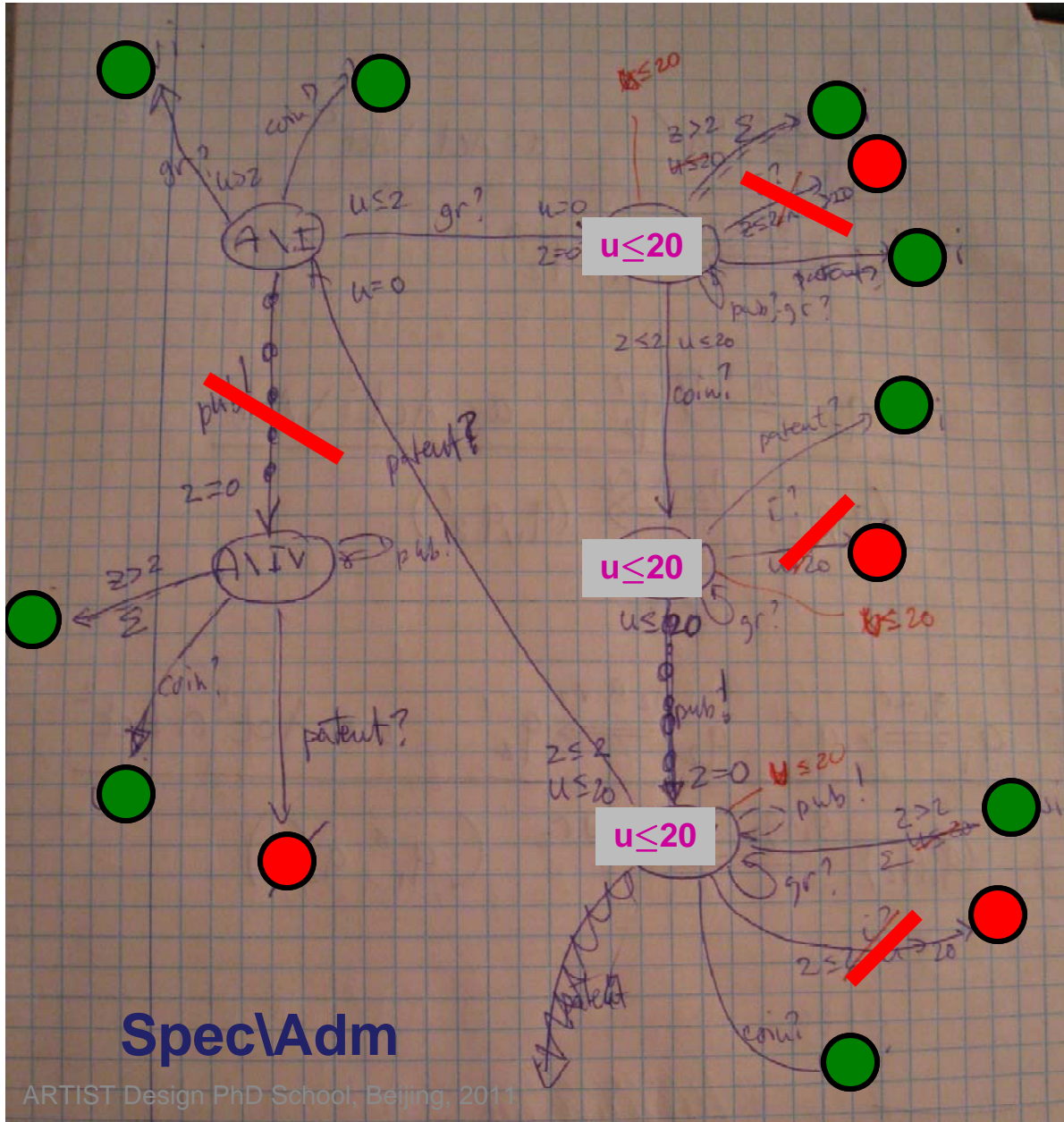


Theorem

$$(S \mid X) \leq T \text{ iff } X \leq (T \setminus S)$$

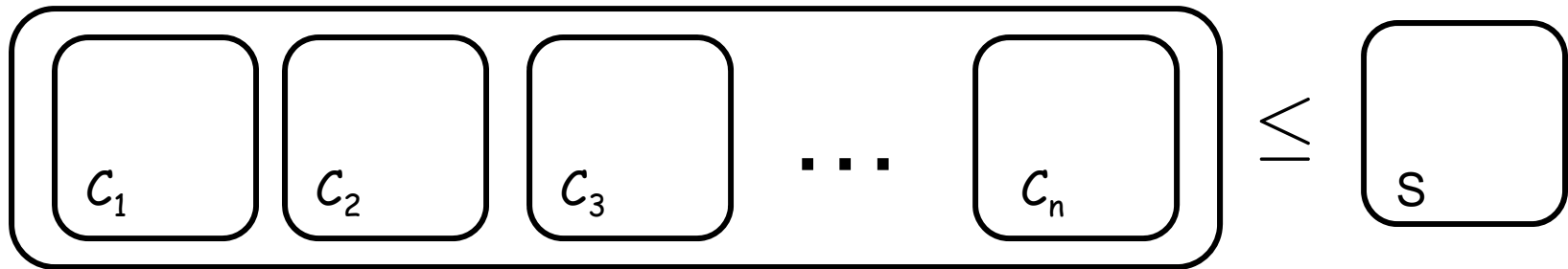


Quotienting, "Application"

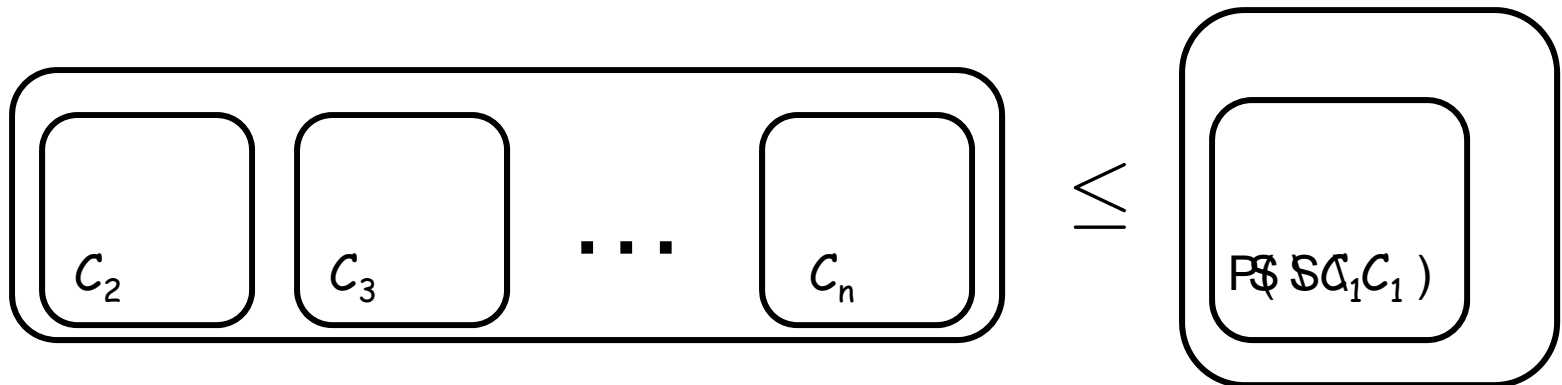


\leq
Spec \ Adm

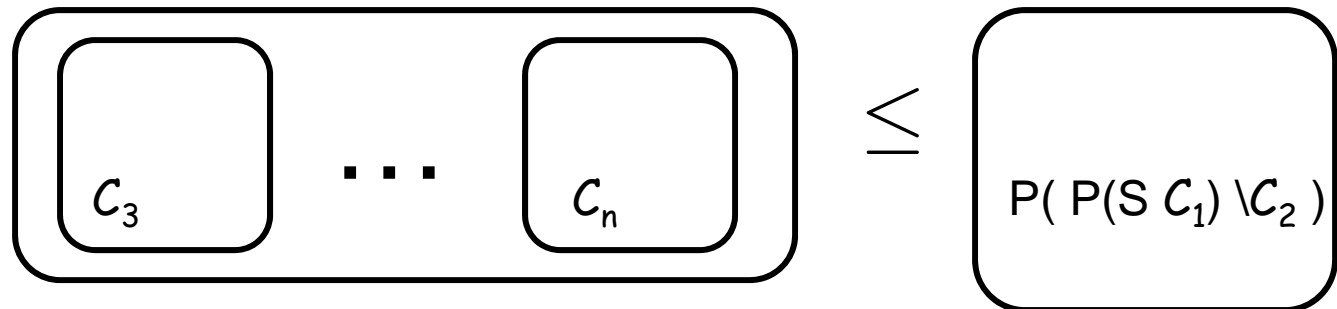
Compositional Refinement Checking



iff



iff

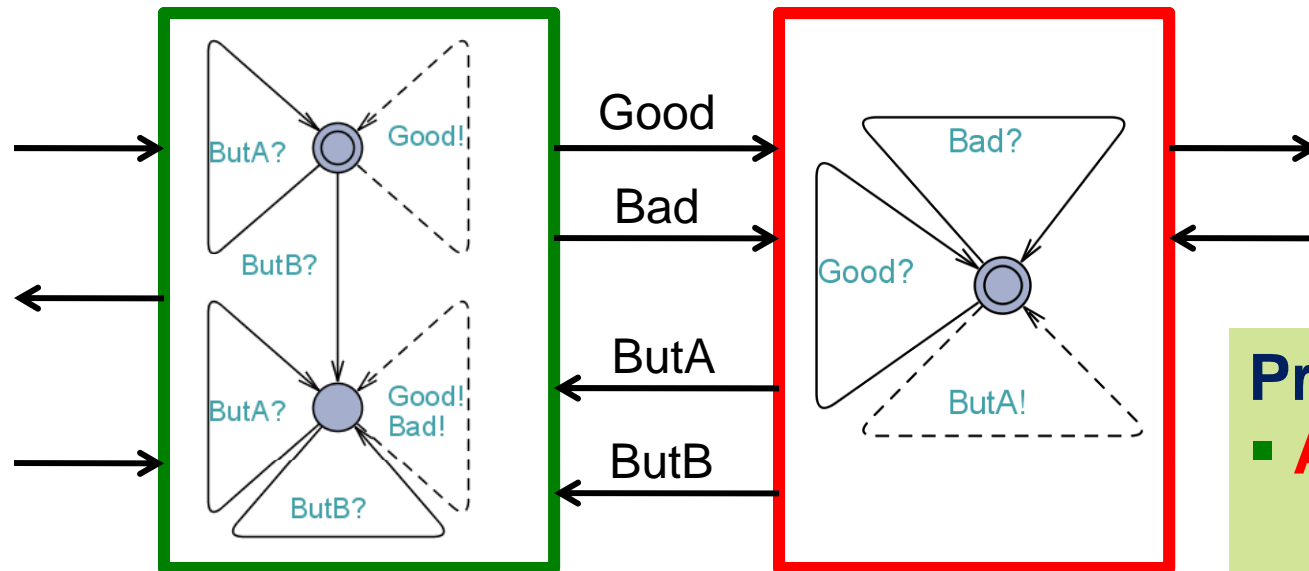


\dots \dots

Assume-Guarantee

Guarantee

Assumption

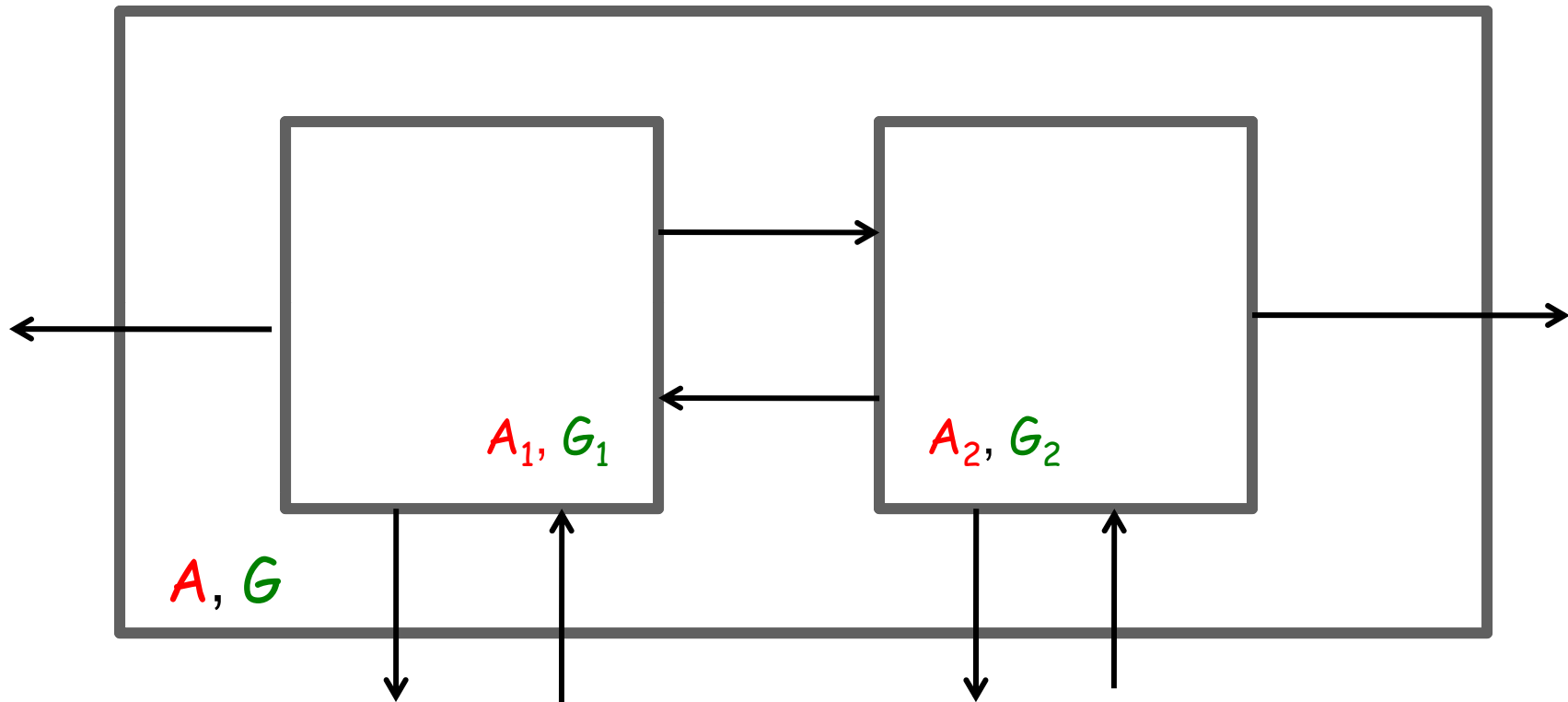


Properties

- $A \gg G \geq G$
- $A \leq A' \Rightarrow A \gg G \geq A' \gg G$
- $G \leq G' \Rightarrow A \gg G \leq A \gg G'$

$$A \gg G = (A \mid G) \setminus A$$

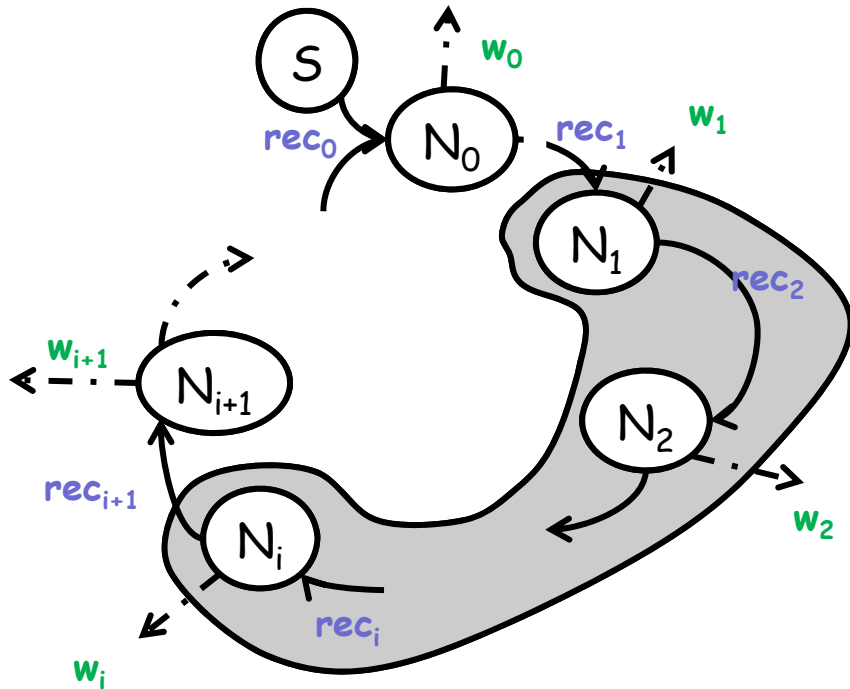
Assume-Guarantee Reasoning



Proof Rule:

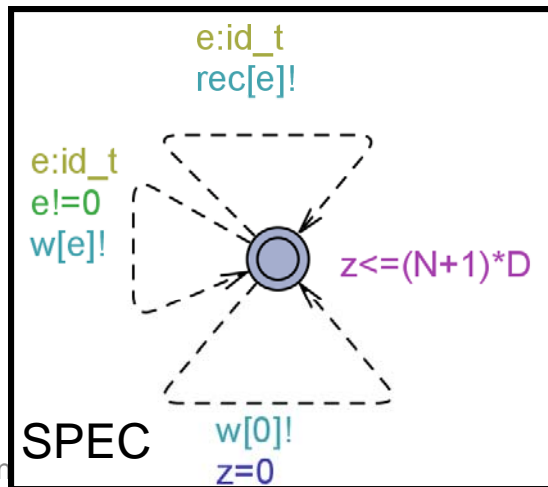
$$A \gg G \geq (A_1 \gg G_1 \mid A_2 \gg G_2)$$

Milner's Scheduler Compositionally

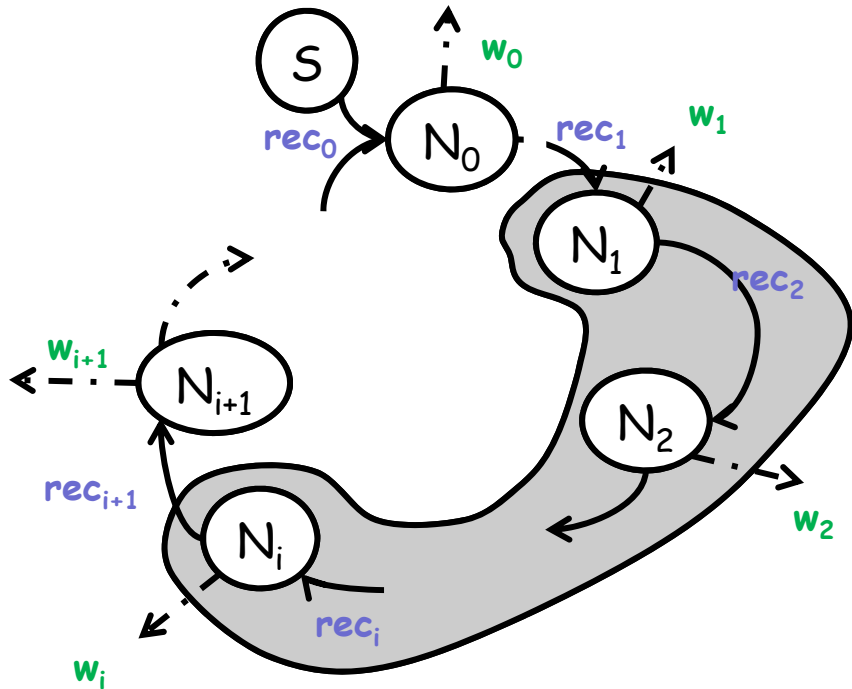


Find SS_i and verify:

1. $N_1 \leq SS_1$
2. $SS_1 \mid N_2 \leq SS_2$
3. $SS_2 \mid N_3 \leq SS_3$
- ...
- n. $SS_{n-1} \mid N_n \leq SS_n$
- n+1. $SS_n \mid N_0 \leq \text{SPEC}$

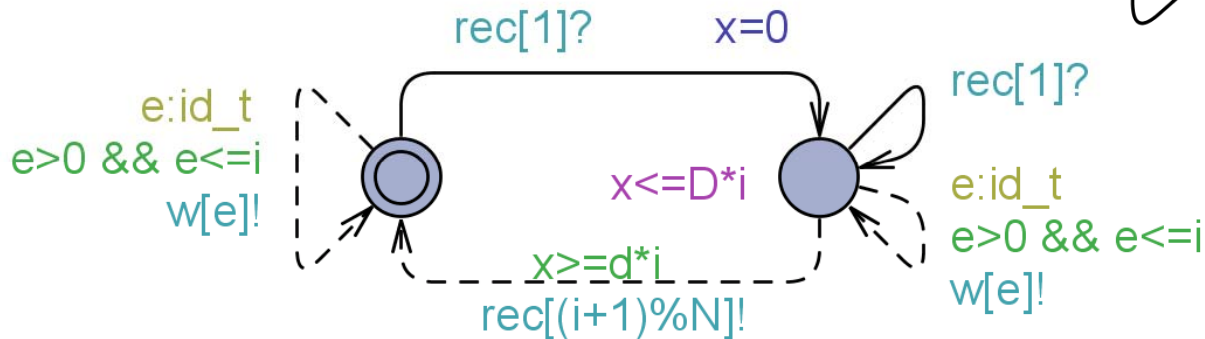
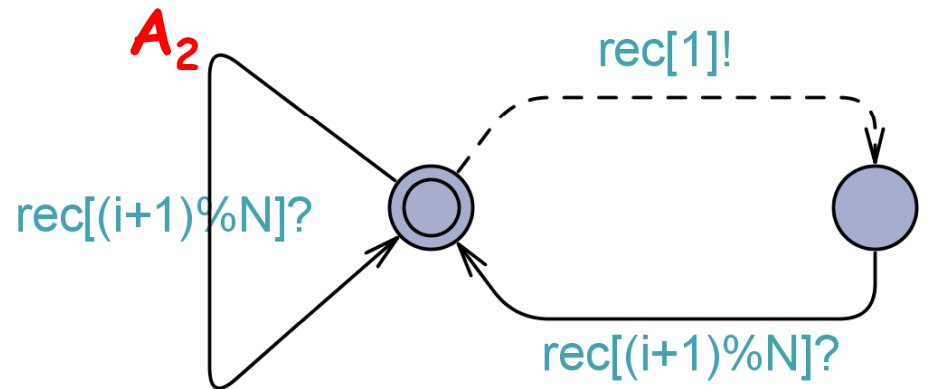
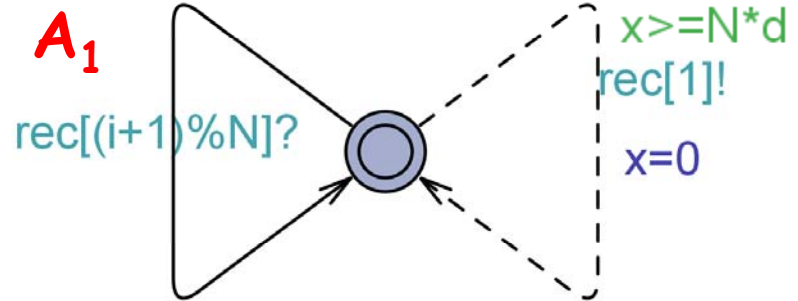


Milner's Scheduler Compositionally



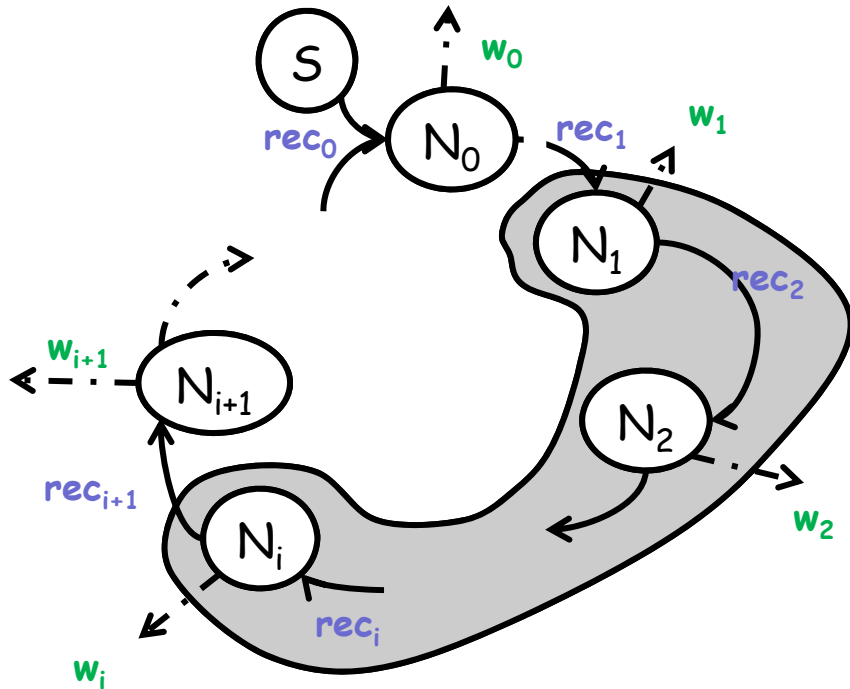
G

Find SS_i

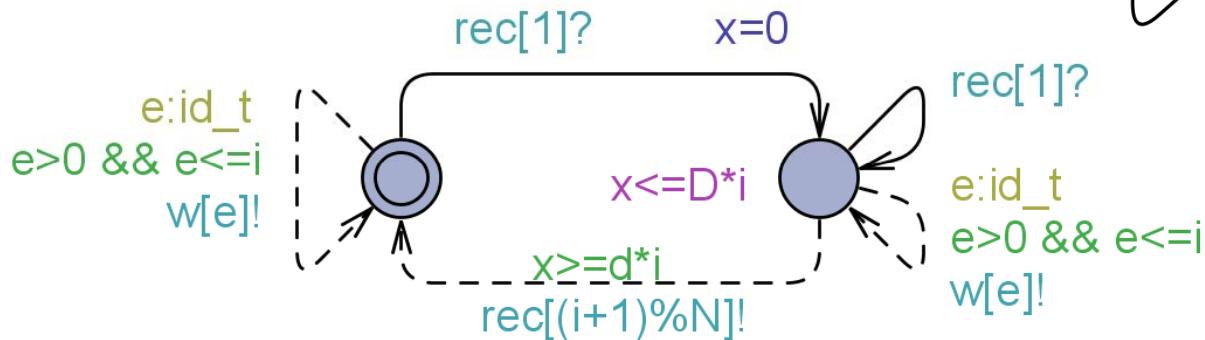
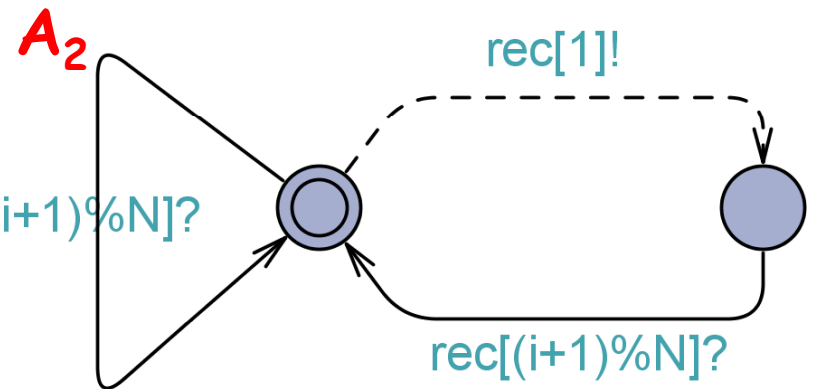
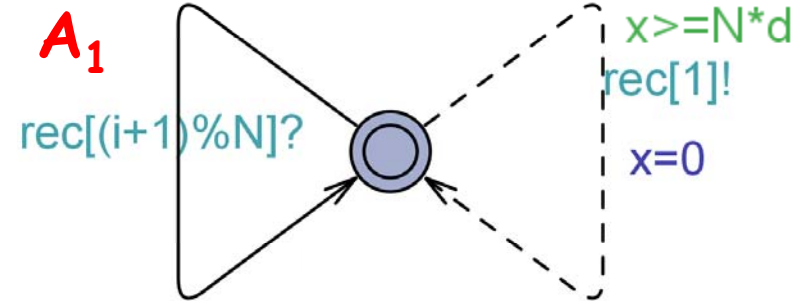


Milner's Scheduler Compositionally

Take $SS_i = (A_1 \ \& \ A_2) \gg G$



G



Experiments

	$d = 29$	20	10	9	8	6	4
$n = 5$	0.080	0.097	0.191	<i>0.169</i>	<i>0.172</i>	<i>0.151</i>	<i>0.205</i>
monolithic	0.034	0.034	0.073	1.191	1.189	64.933	> 600
$n = 6$	0.102	0.133	0.231	<i>0.228</i>	<i>0.238</i>	<i>0.238</i>	<i>0.294</i>
monolithic	0.040	0.043	0.095	6.786	6.791	> 600	> 600
$n = 8$	0.225	0.349	0.516	0.515	<i>0.540</i>	<i>0.600</i>	<i>0.582</i>
monolithic	0.076	0.076	0.230	88.542	88.642	> 600	> 600
$n = 12$	0.830	1.414	1.802	1.895	1.831	<i>2.079</i>	<i>2.181</i>
monolithic	0.220	0.223	0.843	> 600	> 600	> 600	> 600
$n = 20$	4.990	9.739	12.377	11.923	12.041	12.438	<i>12.764</i>
monolithic	1.038	1.030	4.523	> 600	> 600	> 600	> 600
$n = 30$	22.053	45.709	55.728	55.345	55.112	54.702	<i>56.164</i>
monolithic	3.791	3.778	17.652	> 600	> 600	> 600	om

Conclusion & Future Work

- Complete specification theory based on Timed I/O Automata
- Analysis: refinement, consistency, compatibility
- Operations: conjunction, parallel composition, quotienting
- Implemented in the tool ECDAR using the engine of UPPAAL Tiga.
- Non-determinism ?
- Unobservable actions ?
- Applications :
 - Milners Scheduler
 - Leader Election Protocol
 - Fischers Protocol
- **USE IT !!!**

Timed Games W Partial Observability

References

- HSCC'10
Timed I/O Automata: A Complete Specification Theory for Real-time Systems
- FMCO'09
Methodologies for Specification of Real-Time Systems Using Timed I/O Automata
- ATVA'10
Ecdar: An Environment for Compositional Design and Analysis of Real Time Systems
- WADT'10
An Interface Theory for Timed Systems
- www.cs.aau.dk/~adavid/ecdar
- www.uppaal.com