Priced Timed Automata

Optimal Scheduling

Kim G. Larsen

CISS – Aalborg University

DENMARK

Overview

- Timed Automata
  - Scheduling
- Priced Timed Automata
  - Optimal Reachability
  - Optimal Infinite Scheduling
  - Multi Objectives
- Energy Automata

CLASSIC

CORA

TIGA

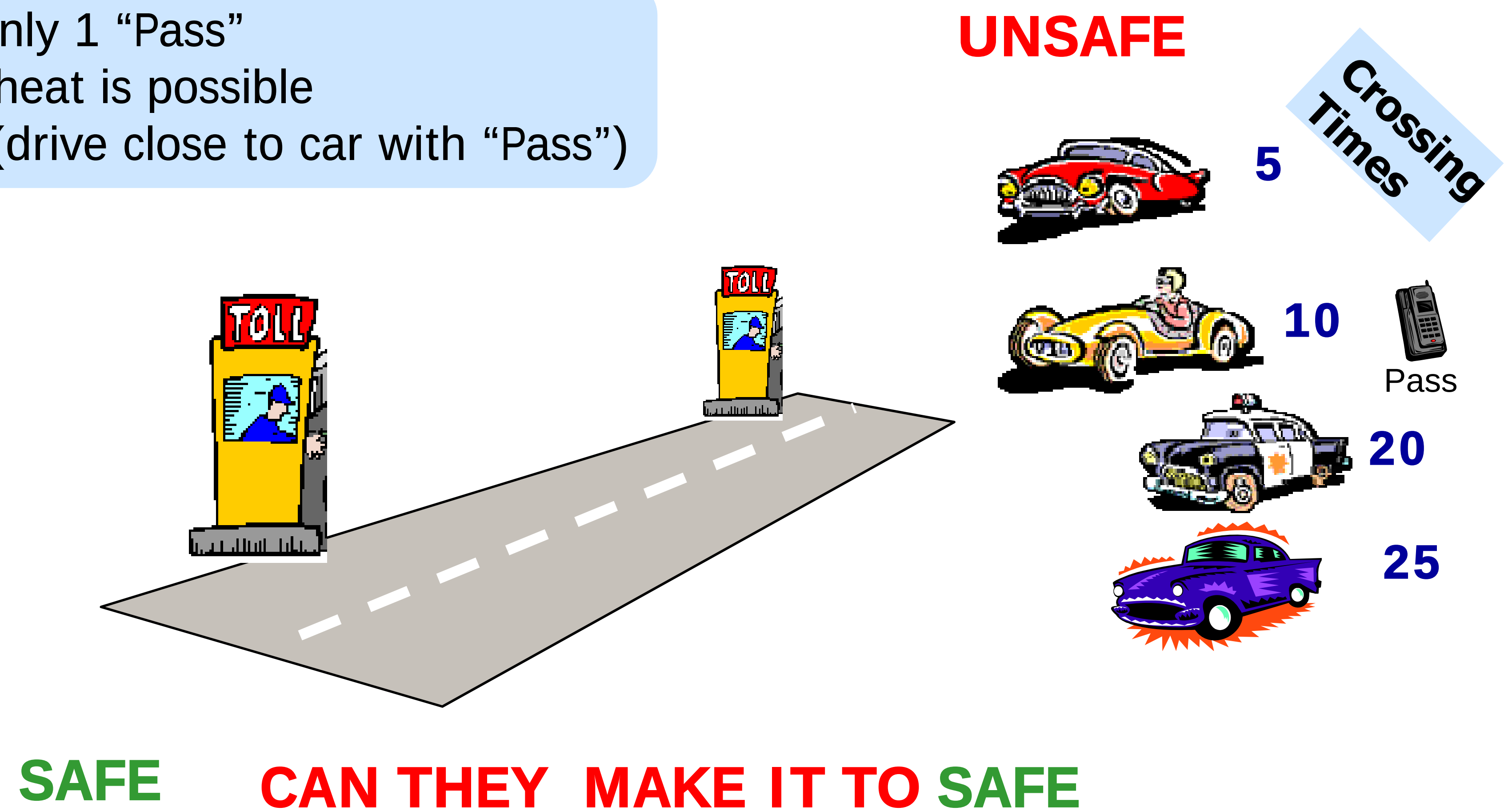
ECDAR

TRON

SMC

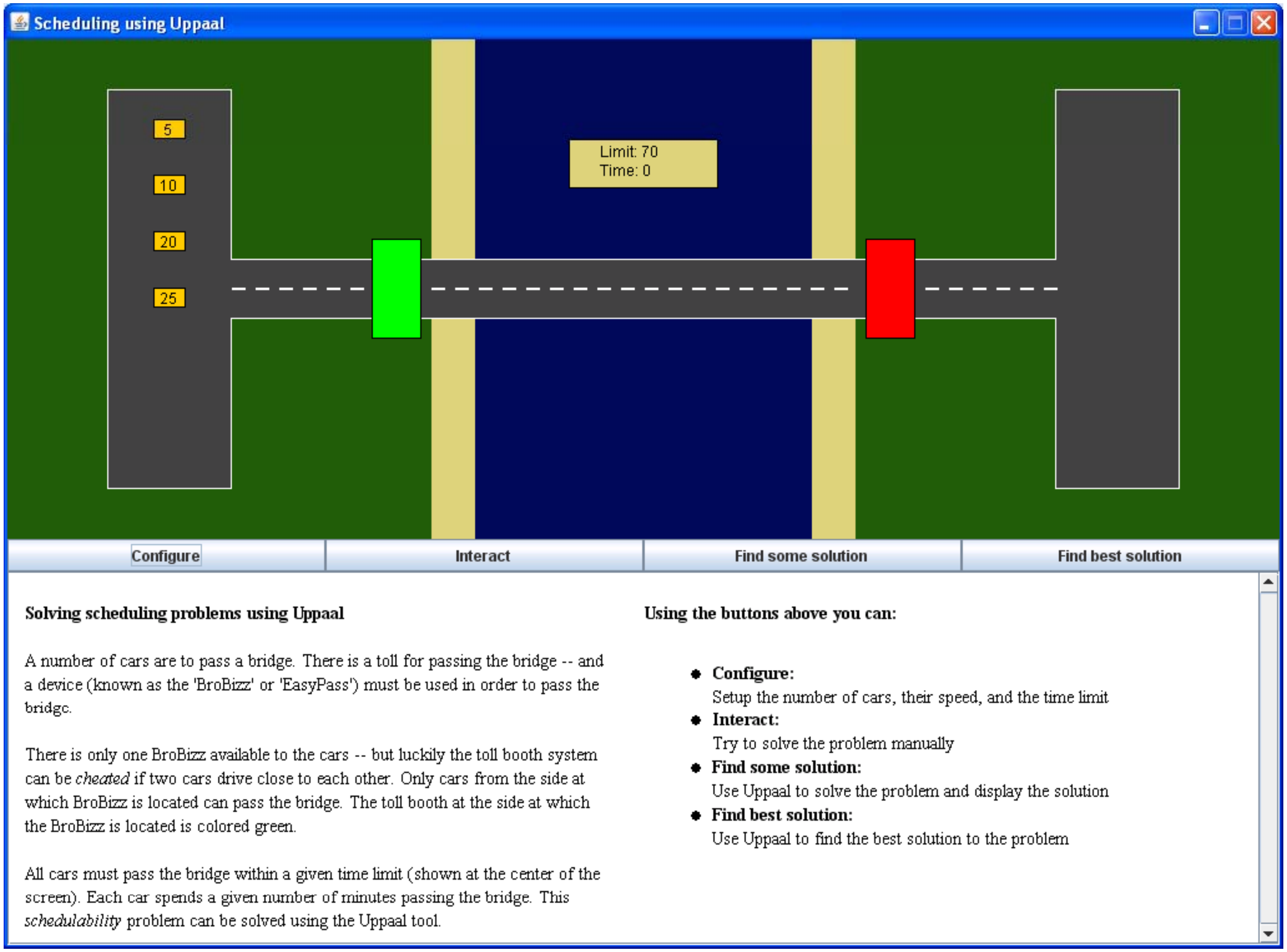
Real Time Scheduling

- Only 1 “Pass”
- Cheat is possible
  (drive close to car with “Pass”)



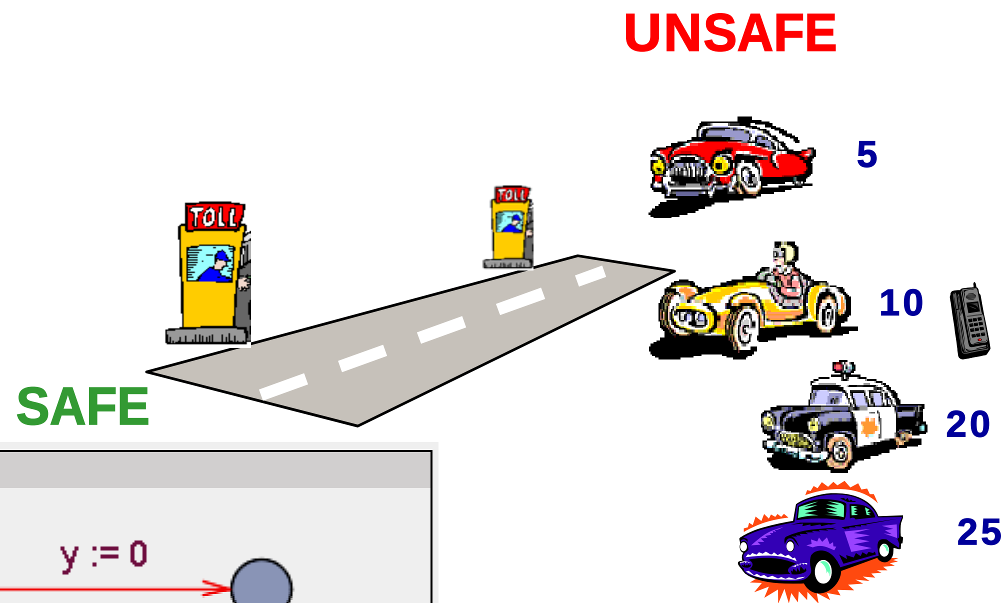
CAN THEY MAKE IT TO SAFE WITHIN 70 MINUTES ???

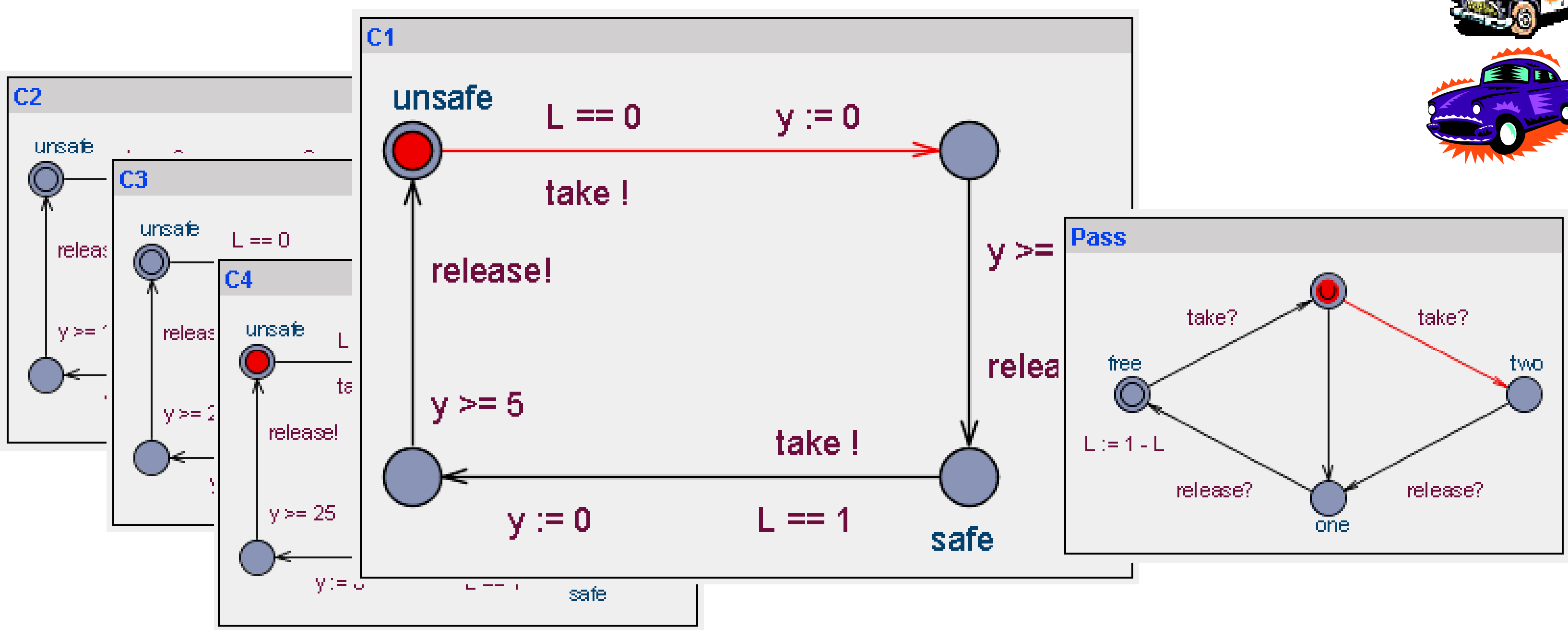
Let us play!


Scheduling using Uppaal
5
10
20
25
Limit: 70
Time: 0
Configure
Interact
Find some solution
Find best solution
Solving scheduling problems using Uppaal
A number of cars are to pass a bridge. There is a toll for passing the bridge -- and a device (known as the 'BroBizz' or 'EasyPass') must be used in order to pass the bridge.
There is only one BroBizz available to the cars -- but luckily the toll booth system can be cheated if two cars drive close to each other. Only cars from the side at which BroBizz is located can pass the bridge. The toll booth at the side at which the BroBizz is located is colored green.
All cars must pass the bridge within a given time limit (shown at the center of the screen). Each car spends a given number of minutes passing the bridge. This schedulability problem can be solved using the Uppaal tool.
Using the buttons above you can:
Configure:
Setup the number of cars, their speed, and the time limit
Interact:
Try to solve the problem manually
Find some solution:
Use Uppaal to solve the problem and display the solution
Find best solution:
Use Uppaal to find the best solution to the problem

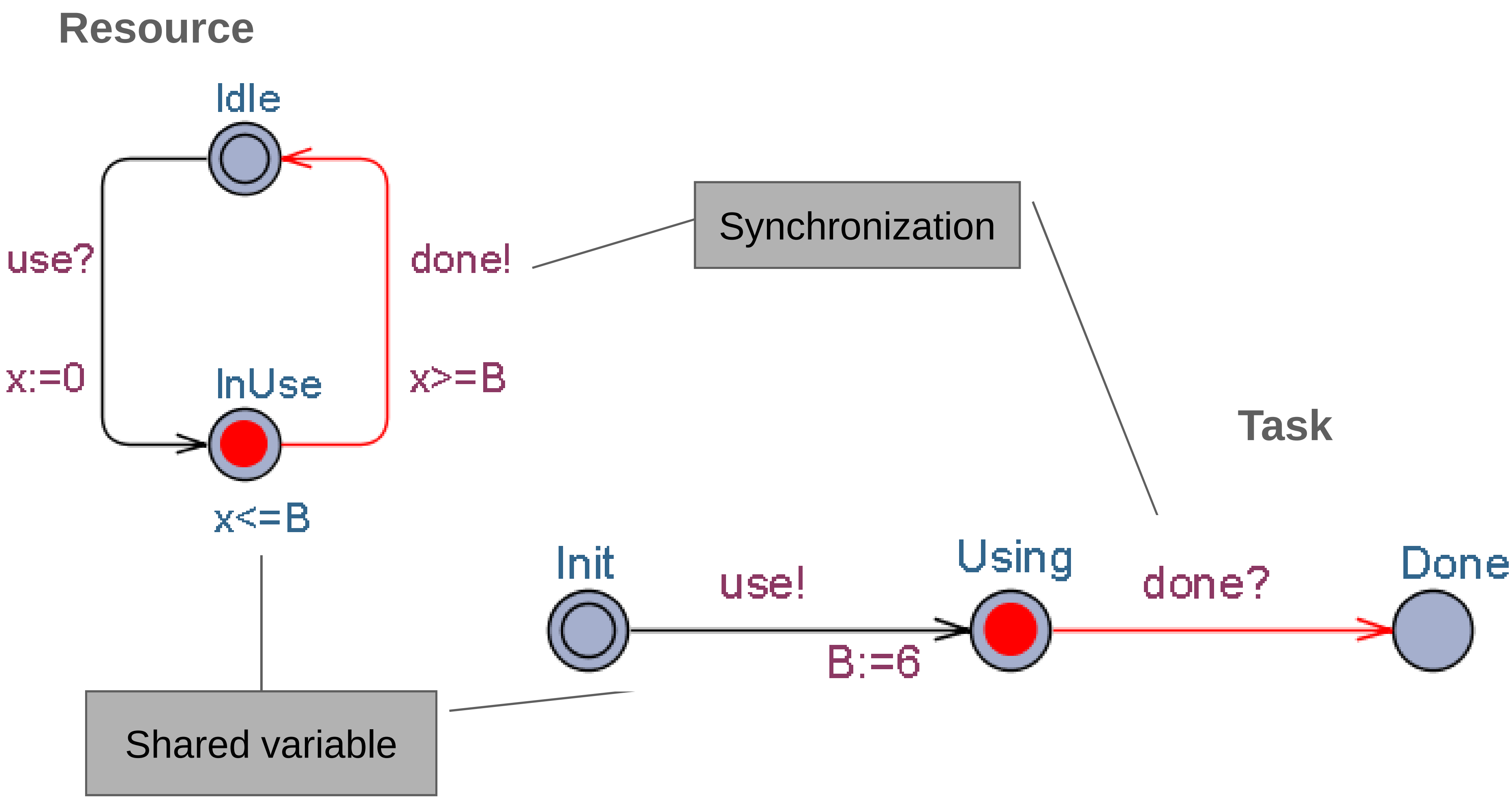
Real Time Scheduling

Solve Scheduling Problem using UPPAAL

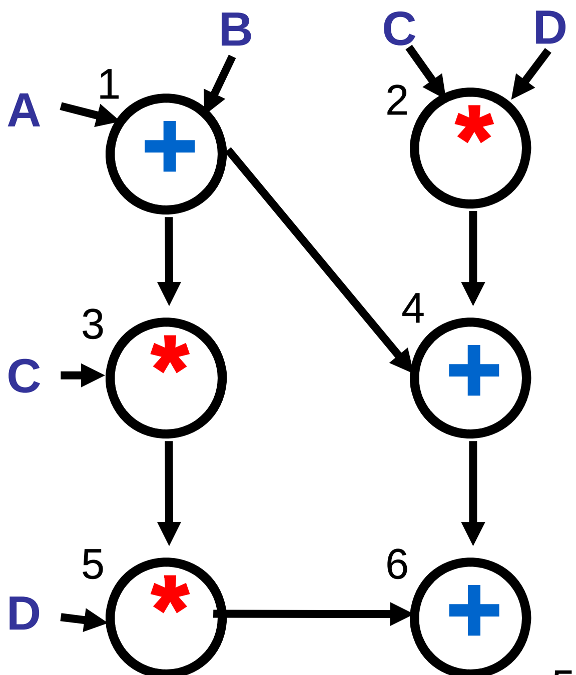




Resources & Tasks



Task Graph Scheduling – Example



Compute :

(D * (C * (A + B)) + ((A + B) + (C * D))

4

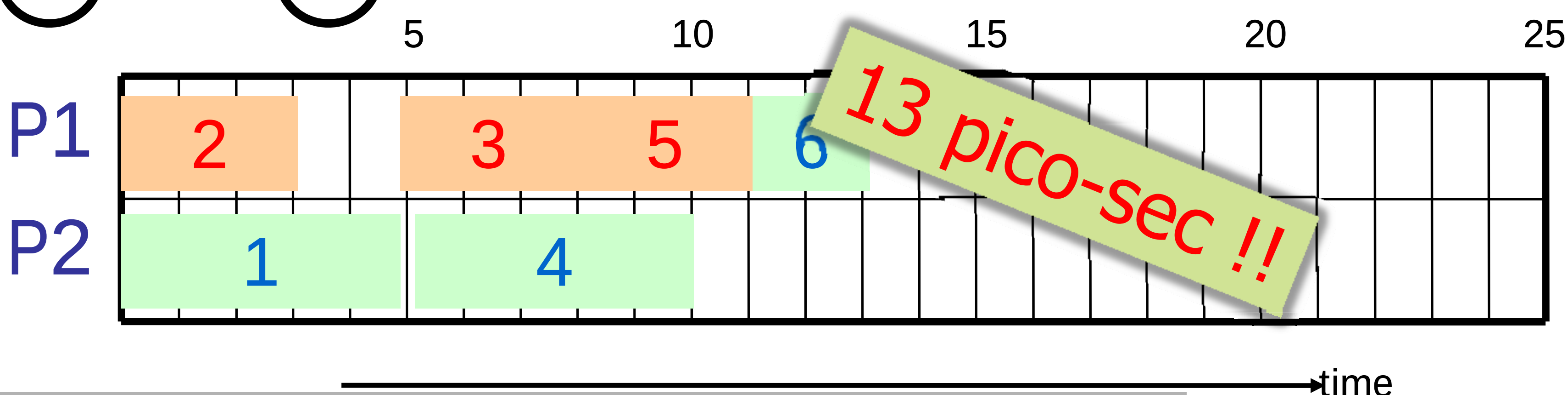
using 2 processors

P1 (fast)

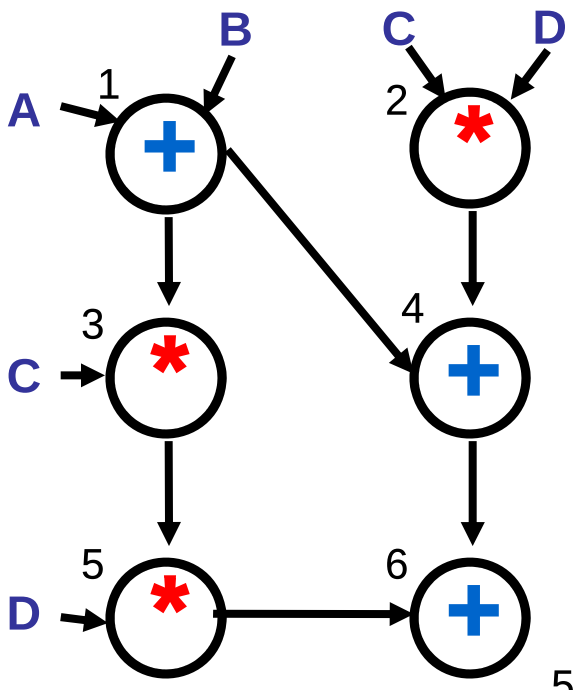
+	2ps
*	3ps

P2 (slow)

+	5ps
*	7ps



Task Graph Scheduling – Example



Compute :

(D * (C * (A + B)) + ((A + B) + (C * D))

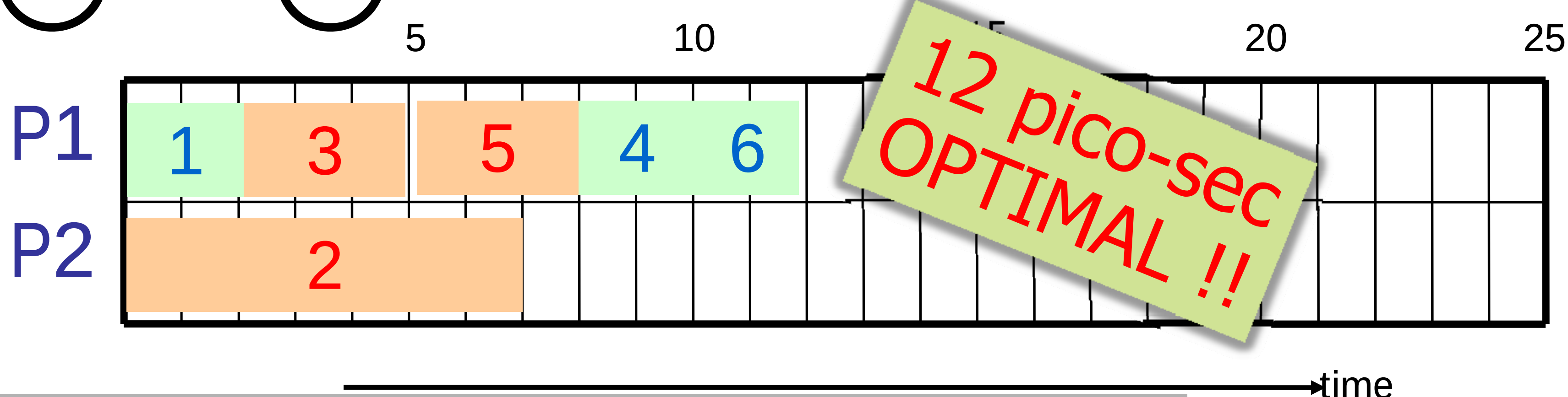
using 2 processors

P1 (fast)

+	2ps
*	3ps

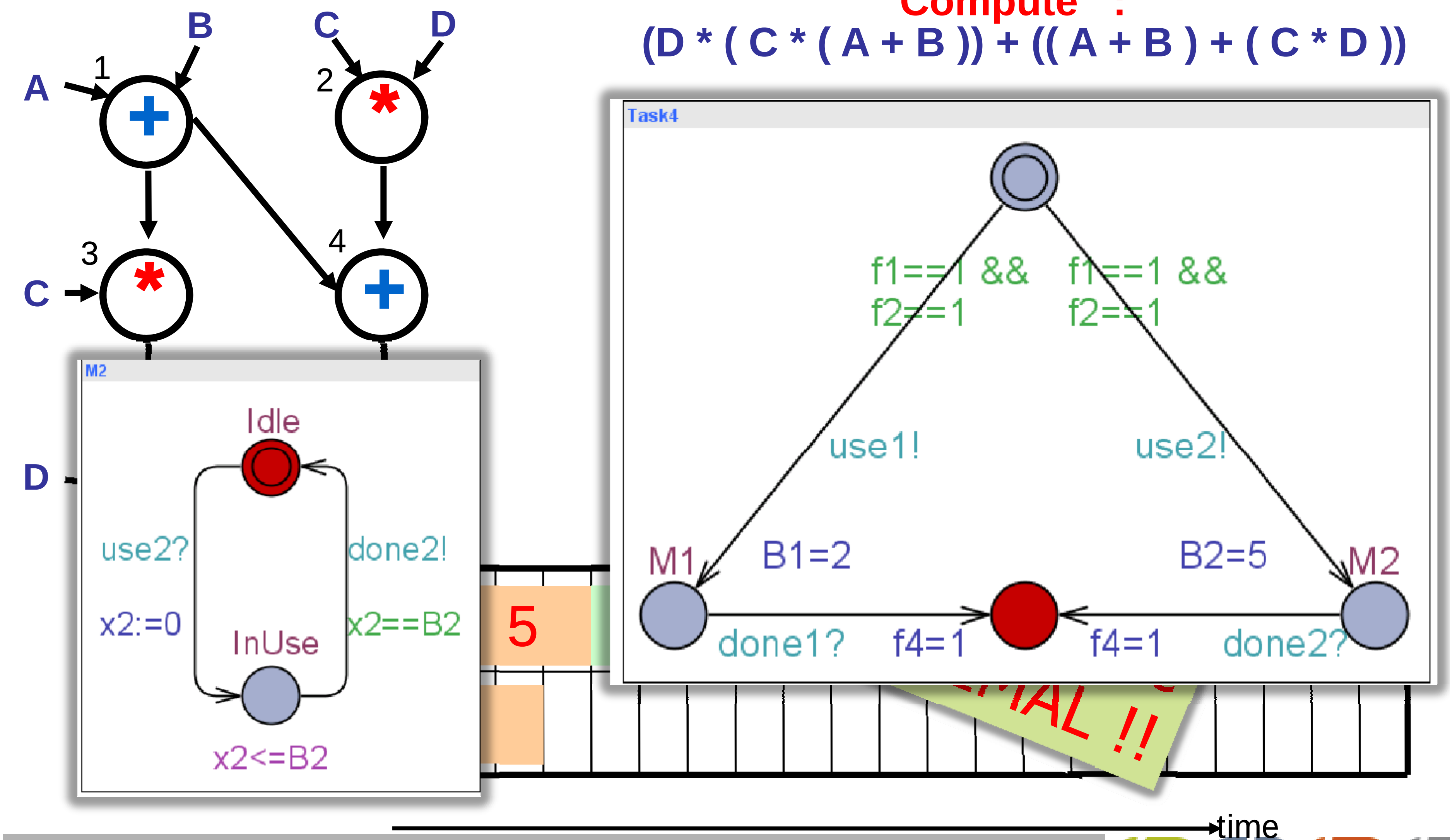
P2 (slow)

+	5ps
*	7ps

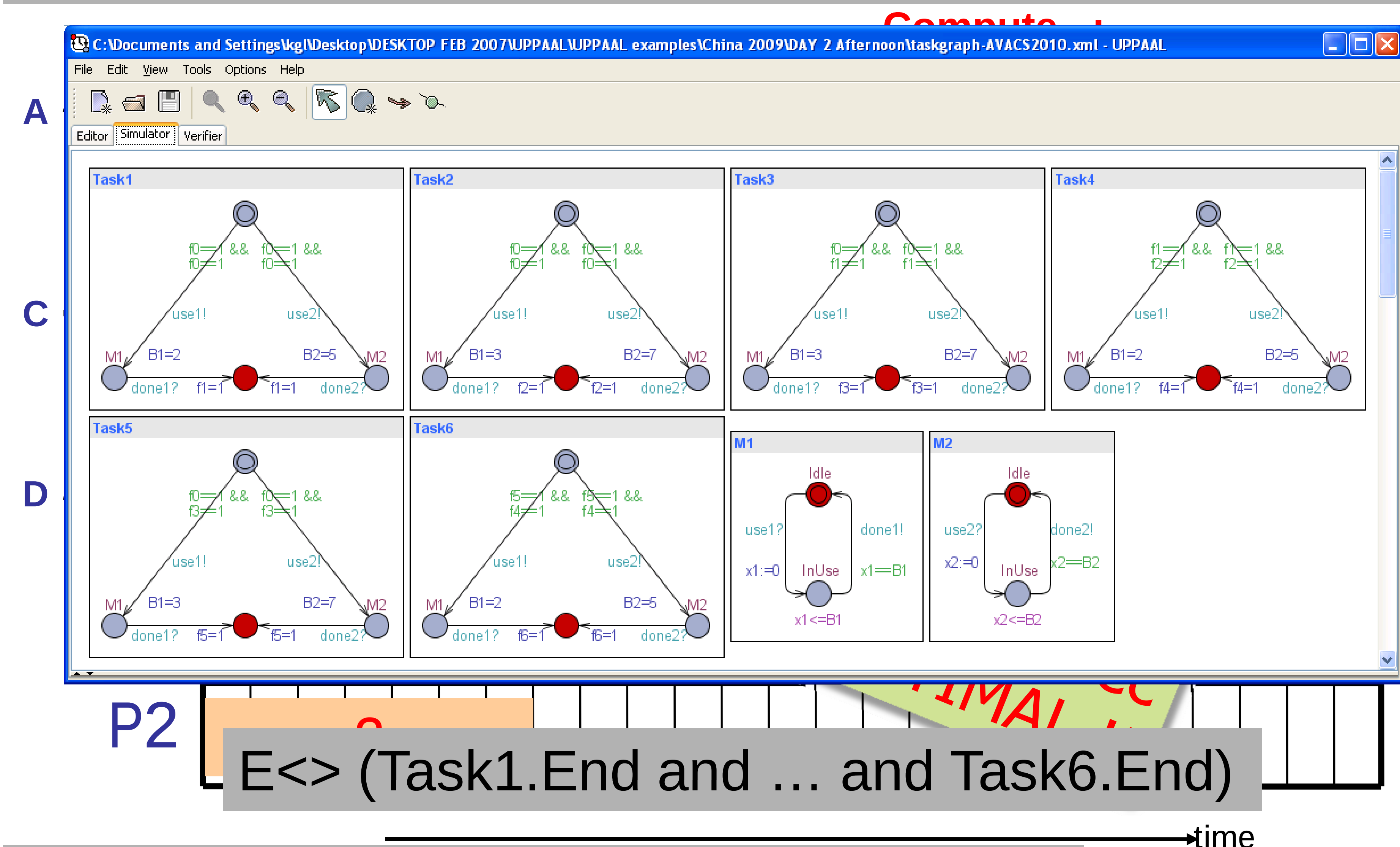


Task Graph Scheduling – Example

Compute :
(D * (C * (A + B)) + ((A + B) + (C * D))



Task Graph Scheduling – Example



E<> (Task1.End and … and Task6.End)

Experimental Results

name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.	1150
233	1980	207	19	1118	1121
294	2014	141	17	1257	1261
295	2168	965	18	1318	1322
292	2333	318	3	8009	8009
298	2399	303	10	2471	2473

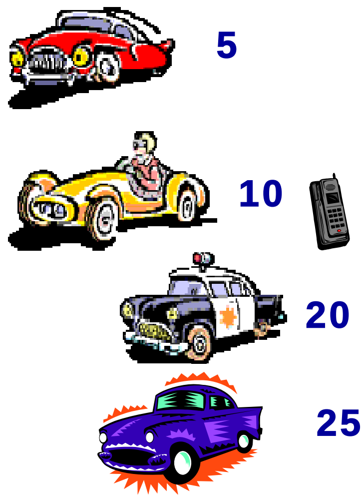
AMETIST
advanced methods for timed systems

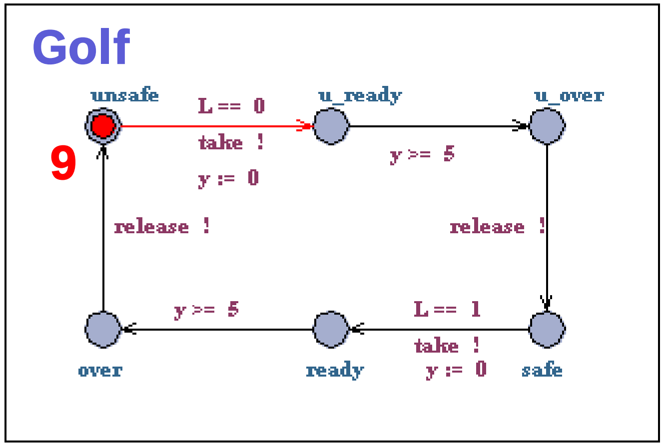
Symbolic A*
Branch-&-Bound
60 sec

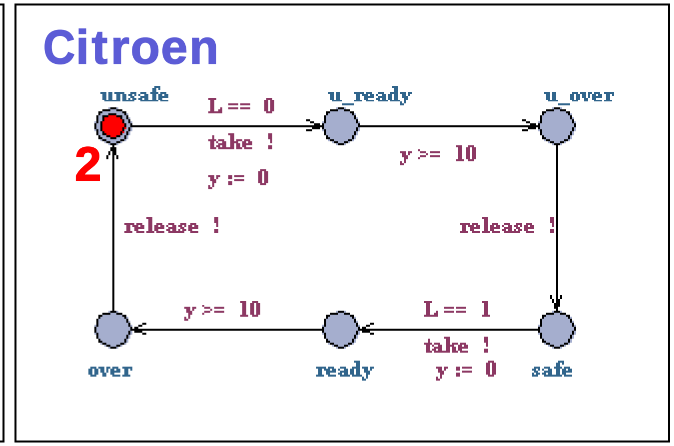
Abdeddaïm, Kerbaa, Maler

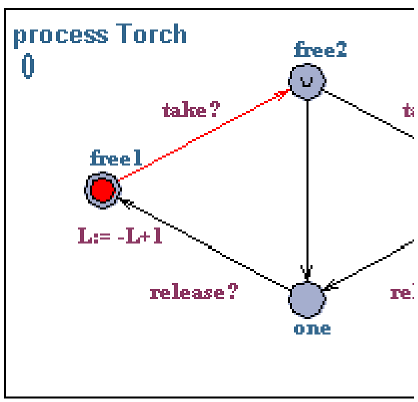
Priced Timed Automata

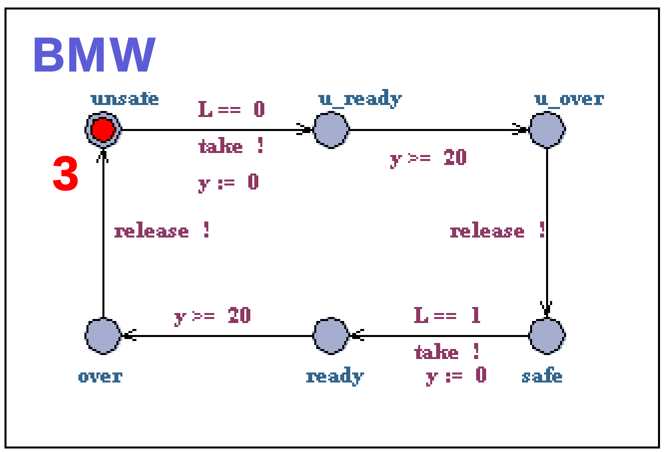
EXAMPLE: Optimal rescue plan for cars with different subscription rates for city driving !

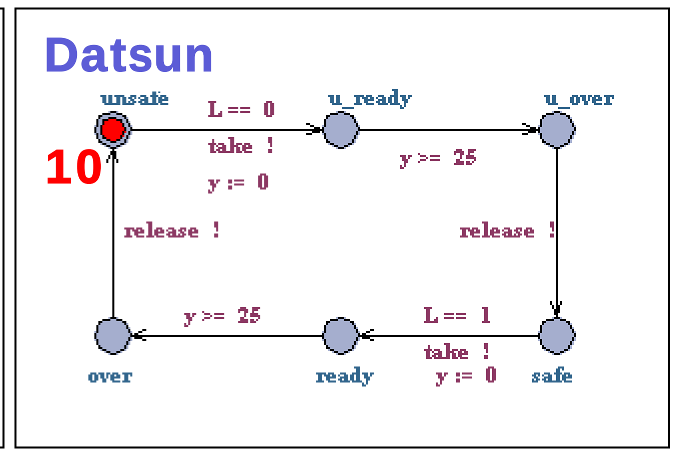










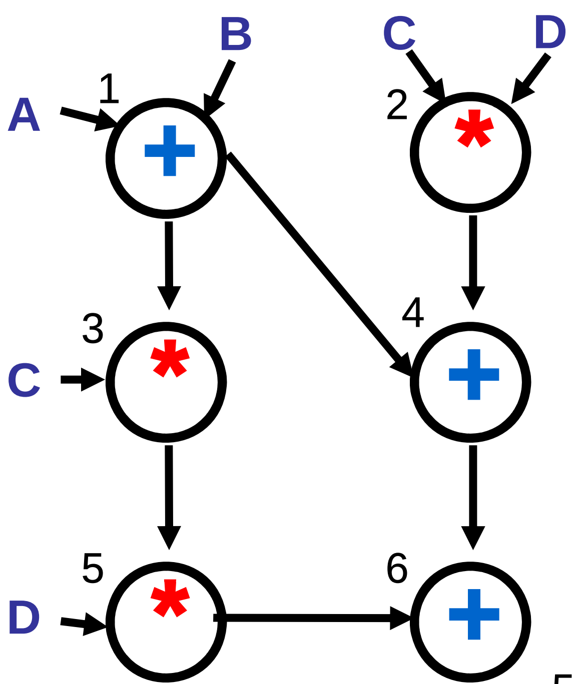


OPTIMAL PLAN HAS ACCUMULATED COST=195 and TOTAL TIME=65!

Experiments

COST-rates				SCHEDULE	COST	TIME	#Expl	#Pop'd
G	C	B	D					
Min Time				CG> G< BD> C< CG>		60	1762 1538	2638
1	1	1	1	CG> G< BG> G< GD>	55	65	252	378
9	2	3	10	GD> G< CG> G< BG>	195	65	149	233
1	2	3	4	CG> G< BD> C< CG>	140	60	232	350
1	2	3	10	CD> C< CB> C< CG>	170	65	263	408
1	20	30	40	BD> B< CB> C< CG>	975 1085	85 time<85	-	-
0	0	0	0	-	0	-	406	447

Task Graph Scheduling – Revisited



Compute :

(D * (C * (A + B)) + ((A + B) + (C * D))

using 2 processors

P1 (fast)

+	2ps
*	3ps
Idle	10W
In use	90W

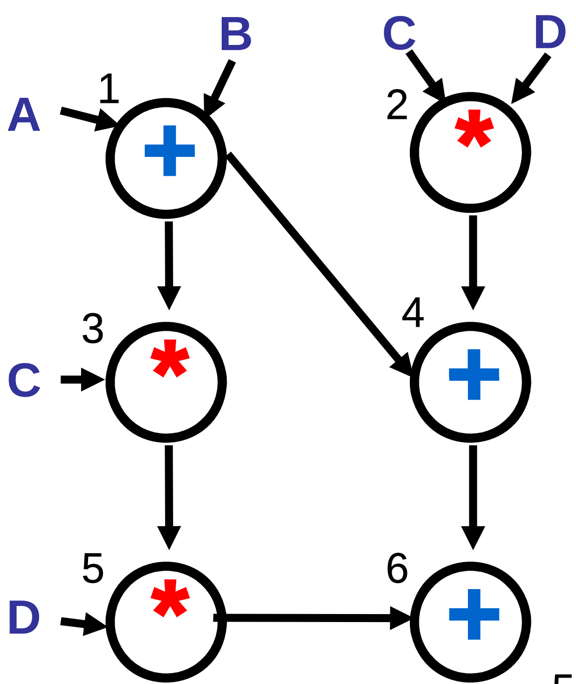
P2 (slow)

+	5ps
*	7ps
Idle	20W
In use	30W

ENERGY:

Energy: 1.39 nano-joule !!

Task Graph Scheduling – Revisited



Compute :

(D * (C * (A + B)) + ((A + B) + (C * D))

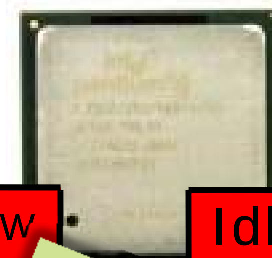
using 2 processors

P1 (fast)

+	2ps
*	3ps

Idle	10W
In use	90W

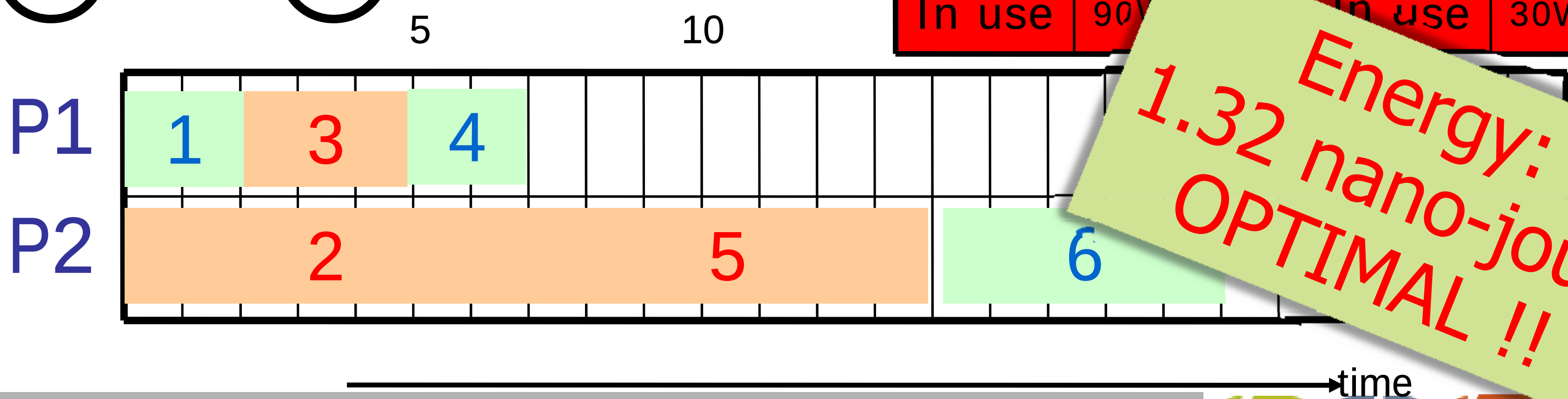
P2 (slow)



+	5ps
*	7ps

Idle	20W
In use	30W

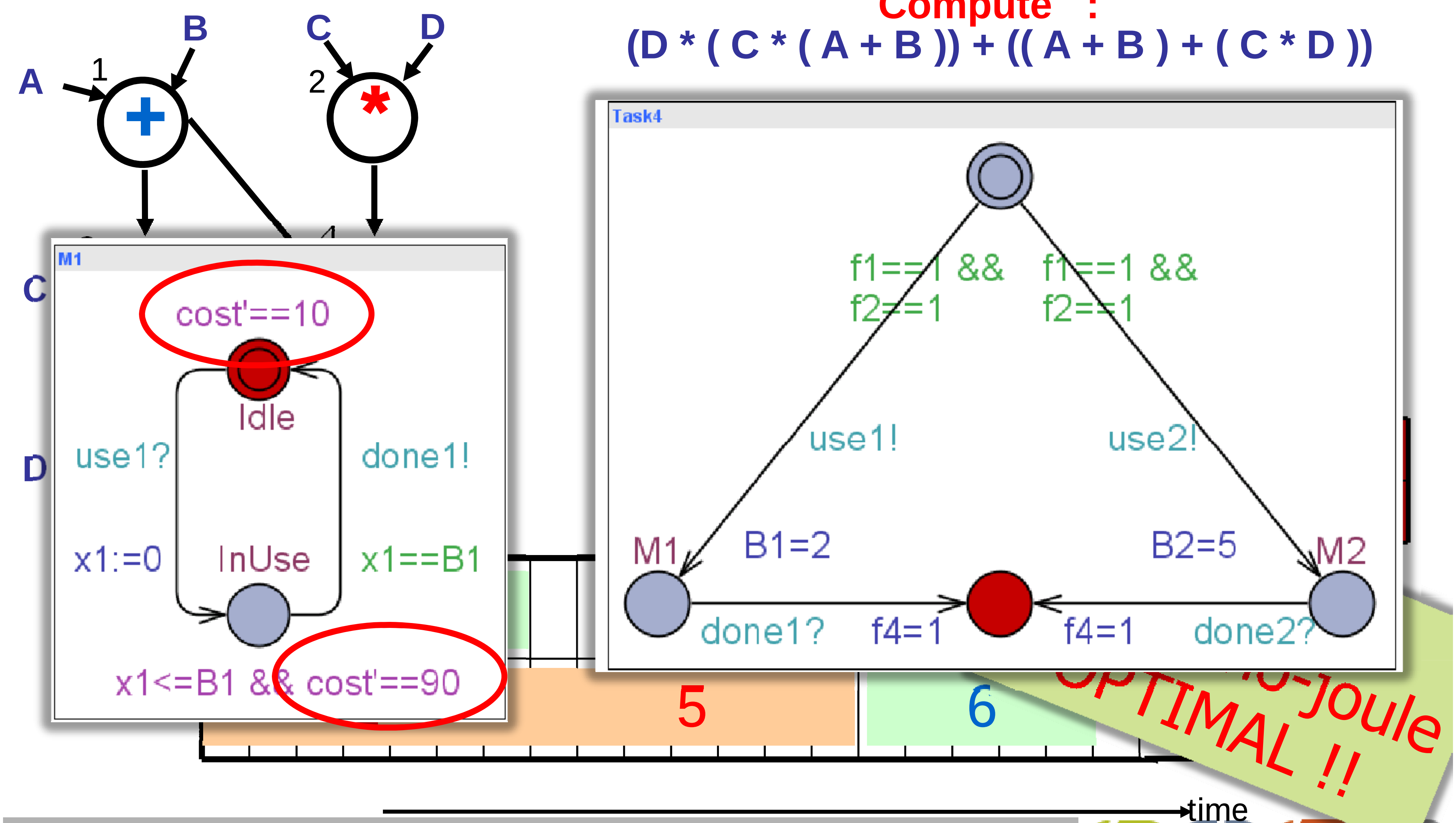
ENERGY:



Energy:
1.32 nano-joule
OPTIMAL !!

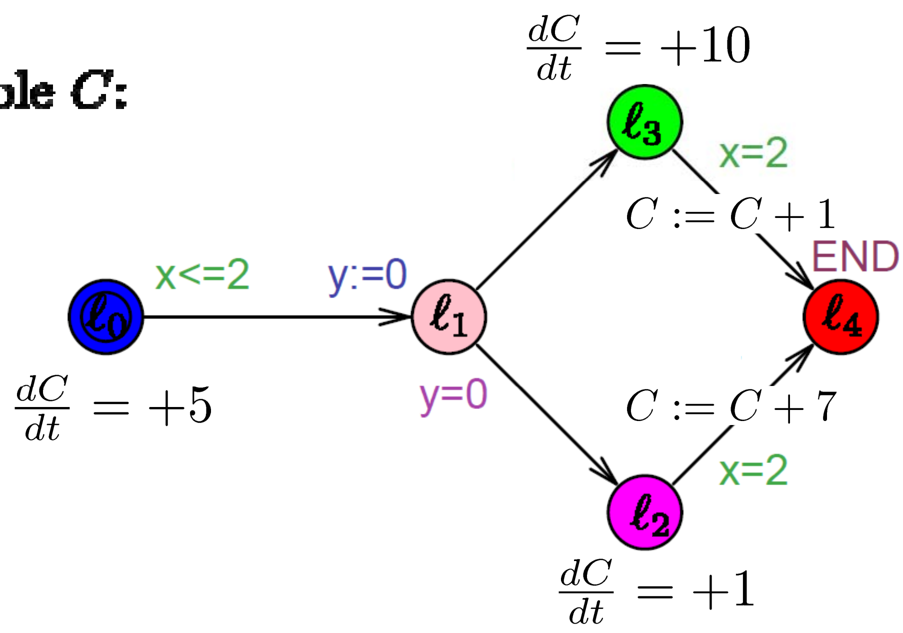
Task Graph Scheduling – Revisited

Compute :
(D * (C * (A + B)) + ((A + B) + (C * D))



A simple example

Observer variable C:



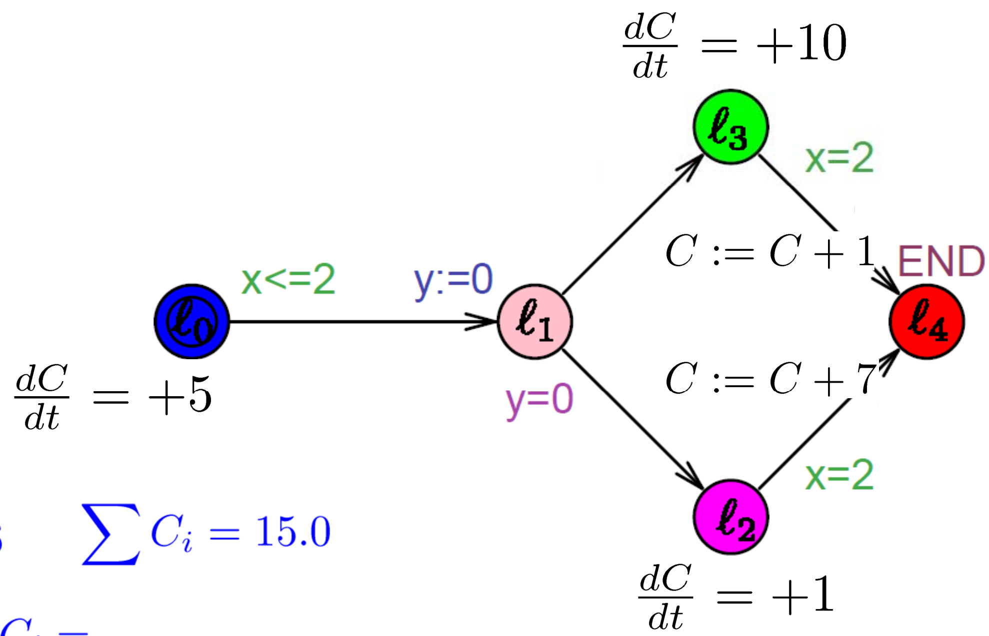
$$(\ell_0, [0,0]) \xrightarrow{1.9}_{9.5} (\ell_0, [1.9, 1.9]) \rightarrow_0 (\ell_1, [1.9, 0]) \rightarrow_0 (\ell_2, [1.9, 0]) \xrightarrow{0.1}_{0.1} (\ell_2, [2, 0.1]) \rightarrow_7 (\ell_4, [2, 0.1])$$

$$\sum C_i = 16.6$$

$$(\ell_0, [0,0]) \xrightarrow{1.2}_{6.0} (\ell_0, [1.2, 1.2]) \rightarrow_0 (\ell_1, [1.2, 0]) \rightarrow_0 (\ell_3, [1.2, 0]) \xrightarrow{0.8}_{8.0} (\ell_3, [2, 0.8]) \rightarrow_1 (\ell_4, [2, 0.8])$$

$$\sum C_i = 15.0$$

A simple example



$\sum C_i = 16.6$ $\sum C_i = 15.0$

$\sum C_i = ..$

Q: What is cheapest cost for reaching ℓ_4 ?

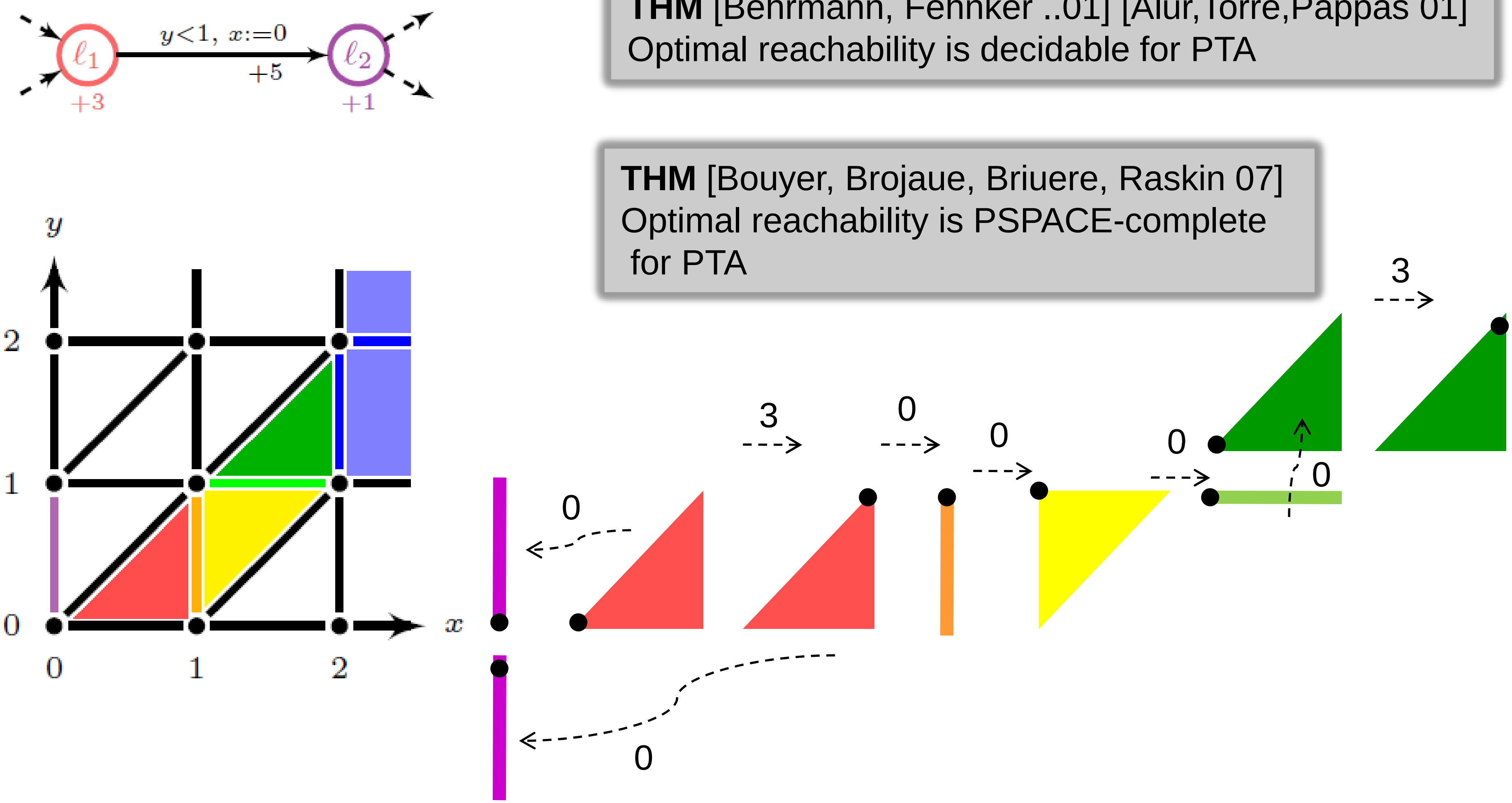
$$\inf_{0 \leq t \leq 2} \min\{5t + 10(2-t) + 1, 5t + (2-t) + 4\} = 9$$

➜ **strategy:** leave immediately ℓ_0, go to ℓ_3, and wait there 2 t.u.

Corner Point Regions

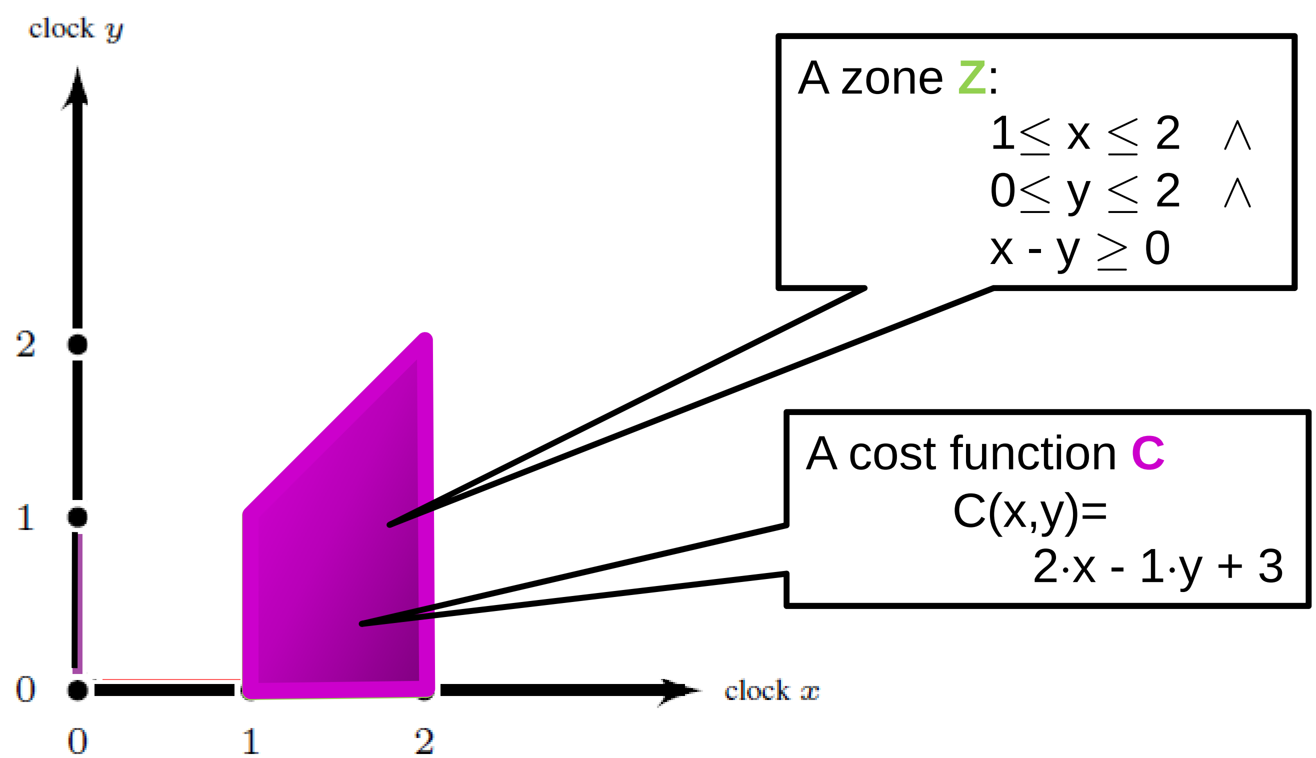
THM [Behrmann, Fehnker ..01] [Alur,Torre,Pappas 01]
Optimal reachability is decidable for PTA

THM [Bouyer, Brojaue, Briuere, Raskin 07]
Optimal reachability is PSPACE-complete
for PTA

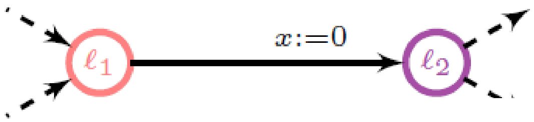


Priced Zones

[CAV01]



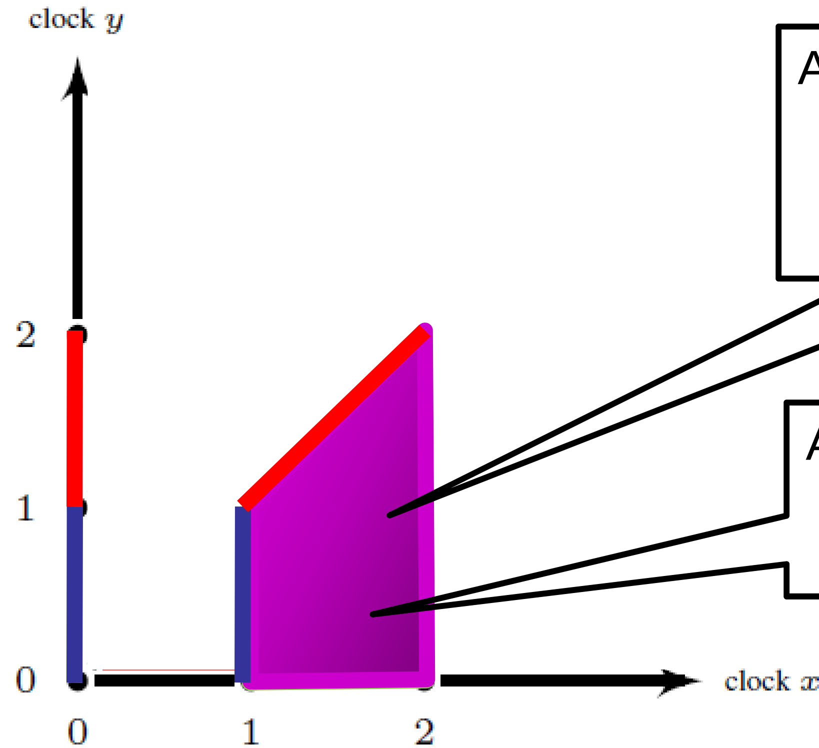
Priced Zones – Reset [CAV01]



Z[x=0]:
x=0 ∧
0≤ y ≤ 2

C = 1·y + 3

C= -1·y + 5



A zone **Z**:
1≤ x ≤ 2 ∧
0≤ y ≤ 2 ∧
x - y ≥ 0

A cost function **C**
C(x,y) =
2·x - 1·y + 3

Symbolic Branch & Bound Algorithm

Cost := ∞

Passed := $\emptyset$

Waiting := $\{(l_0, Z_0)\}$

while Waiting $\neq \emptyset$ **do**

  select (l, Z) from Waiting

  if $l = l_g$ and minCost(Z) < Cost **then**

    Cost := minCost(Z)

  if minCost(Z) + Rem$_{(l,Z)} \geq$ Cost **then** break

  if for all (l, Z') in Passed: $Z' \not\leq Z$ **then**

    add (l, Z) to Passed

    add all (l', Z') with $(l, Z) \rightarrow (l', Z')$

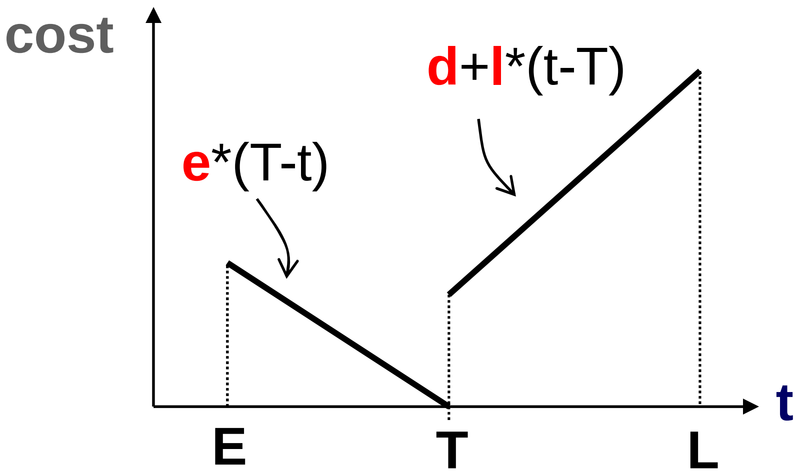
return Cost

$Z' \leq Z$

Z' **is bigger & cheaper than** ***Z***

$\leq$ **is a well-quasi ordering which guarantees termination!**

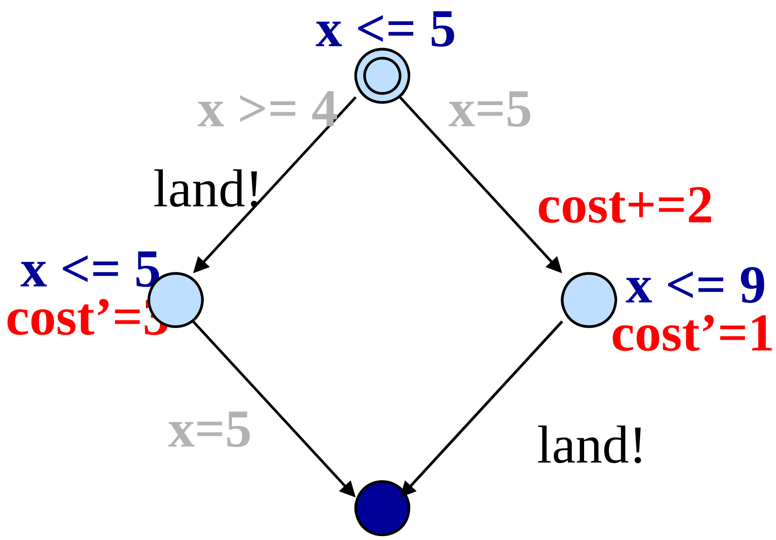
Example: Aircraft Landing



E earliest landing time
T target time
L latest time
e cost rate for being early
l cost rate for being late
d fixed cost for being late

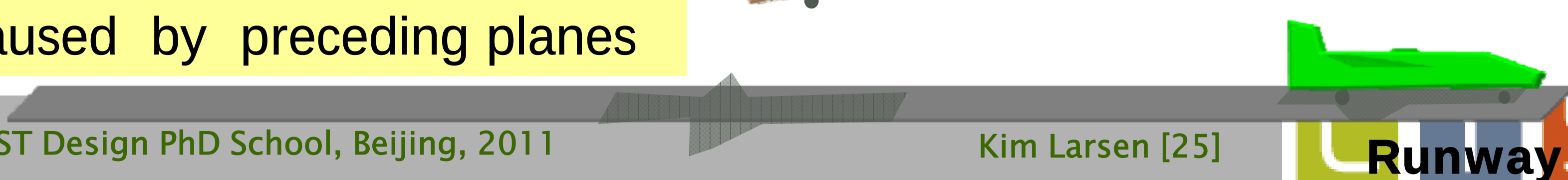
Planes have to keep separation distance to avoid turbulences caused by preceding planes

Example: Aircraft Landing



4 earliest landing time
5 target time
9 latest time
3 cost rate for being early
1 cost rate for being late
2 fixed cost for being late

Planes have to keep separation distance to avoid turbulences caused by preceding planes



Aircraft Landing

Source of examples:
Baesley et al'2000

	problem instance	1	2	3	4	5	6	7
	number of planes	10	15	20	20	20	30	44
	number of types	2	2	2	2	2	4	2
1	optimal value	700	1480	820	2520	3100	24442	1550
	explored states	481	2149	920	5693	15069	122	662
	cputime (secs)	4.19	25.30	11.05	87.67	220.22	0.60	4.27
2	optimal value	90	210	60	640	650	554	0
	explored states	1218	1797	669	28821	47993	9035	92
	cputime (secs)	17.87	39.92	11.02	755.84	1085.08	123.72	1.06
3	optimal value	0	0	0	130	170	0	N/A
	explored states	24	46	84	207715	189602	62	
	cputime (secs)	0.36	0.70	1.71	14786.19	12461.47	0.68	
4	optimal value	N/A	N/A	N/A	0	0	N/A	N/A
	explored states				65	64		
	cputime (secs)				1.97	1.53		

Symbolic Branch & Bound Algorithm

Cost := ∞
Passed := $\emptyset$
Waiting := $\{(l_0, Z_0)\}$
while Waiting $\neq \emptyset$ **do**
  select (l, Z) from Waiting
  if $l = l_g$ and minCost(Z) < Cost **then**
    Cost := minCost(Z)
  if minCost(Z) + Rem$_{(l,Z)}$ $\geq$ Cost **then** break
  if for all (l, Z') in Passed: $Z' \not\subseteq Z$ **then**
    add (l, Z) to Passed
    add all (l', Z') with $(l, Z) \rightarrow (l', Z')$ to Waiting
return Cost

Zone based
Linear Programming Problems
→(dualize)
Min Cost Flow

Aircraft Landing (revisited)

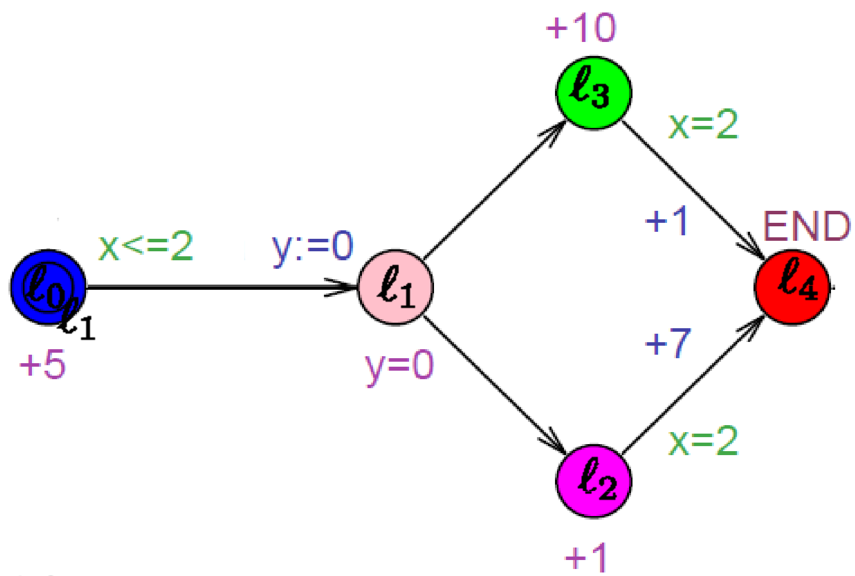
[TACAS04]

Using MCF/Netsimplex

RW	Planes	10	15	20	20	20	30	44
	Types	2	2	2	2	2	4	2
1	simplex	0.844s	5.210s	2.135s	17.888s	44.878s	0.451s	0.670s
	netsimplex	0.156s	0.657s	0.369s	2.363s	5.503s	0.127s	0.322s
factor		**5.41**	**7.93**	**5.79**	**7.57**	**8.16**	**3.55**	**2.08**
2	simplex	2.577s	7.436s	2.175s	94.357s	120.004s	2.322s	0.264s
	netsimplex	0.332s	1.036s	0.436s	13.376s	18.033s	0.600s	0.179s
factor		**8.00**	**7.18**	**4.99**	**7.054**	**6.65**	**3.87**	**1.474**
3	simplex	0.120s	0.181s	0.357s	740.100s	516.678s	0.166s	N/A
	netsimplex	0.064s	0.104s	0.129s	170.176s	124.805s	0.079s	N/A
factor		**1.87**	**1.74**	**2.77**	**4.34**	**4.14**	**2.10**	
4	simplex	N/A	N/A	N/A	1.603s	0.318s	N/A	N/A
	netsimplex	N/A	N/A	N/A	0.378s	0.093s	N/A	N/A
factor					**4.24**	**3.42**		

A. Loebel (2000). MCF Version 1.2 - A network simplex implementation. (http://www.zib.de)

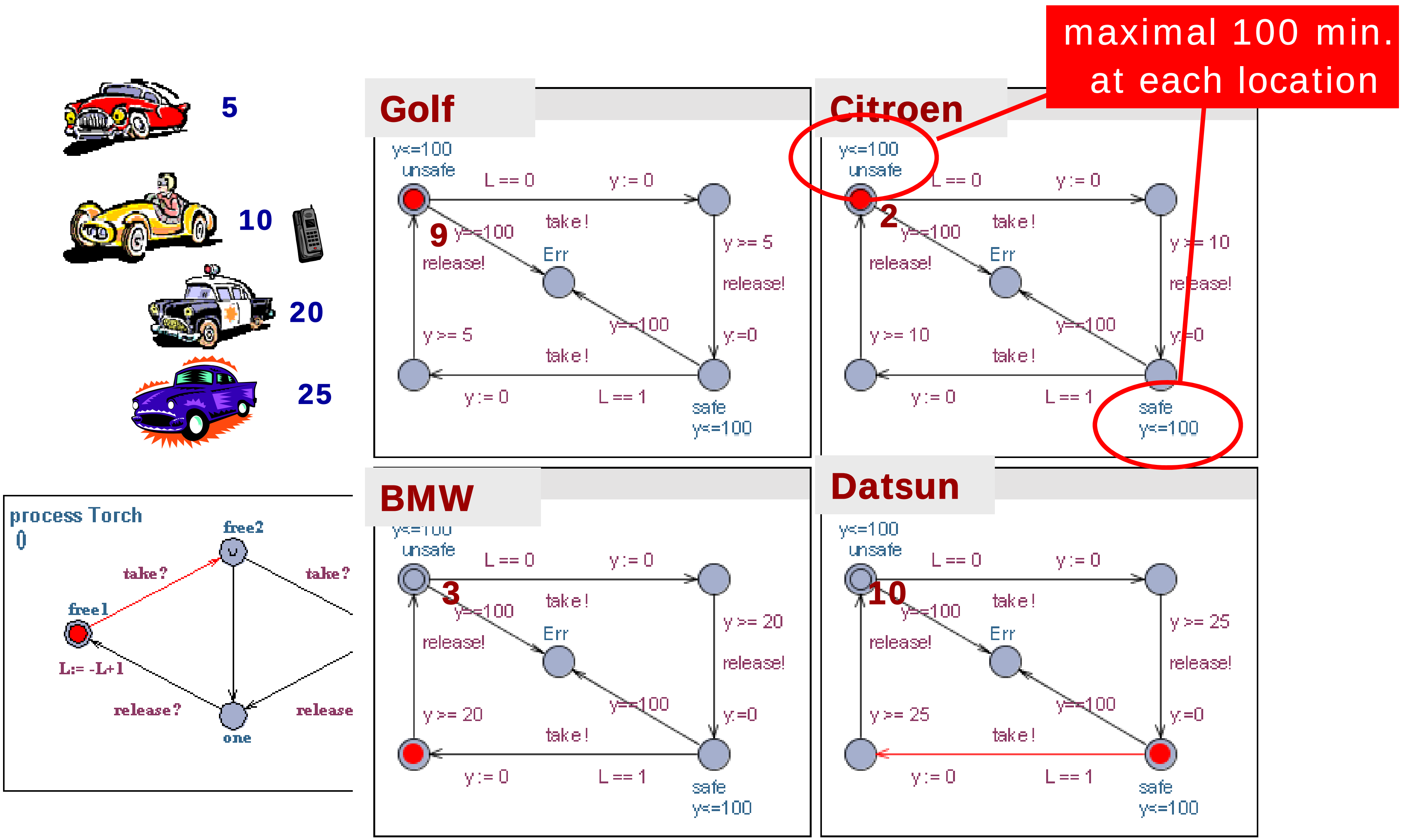
Optimal Schedule



$$(\ell_0,[0,0]) \xrightarrow{1.2}_{6.0} (\ell_0,[1.2,1.2]) \rightarrow_0 (\ell_1,[1.2,0]) \rightarrow_0$$
$$(\ell_3,[1.2,0]) \xrightarrow{0.8}_{8.0} (\ell_3,[2,0.8]) \rightarrow_1 (\ell_4,[2,0.8])$$
$$\rightarrow_{2.0} (\ell_0,[0,0])$$

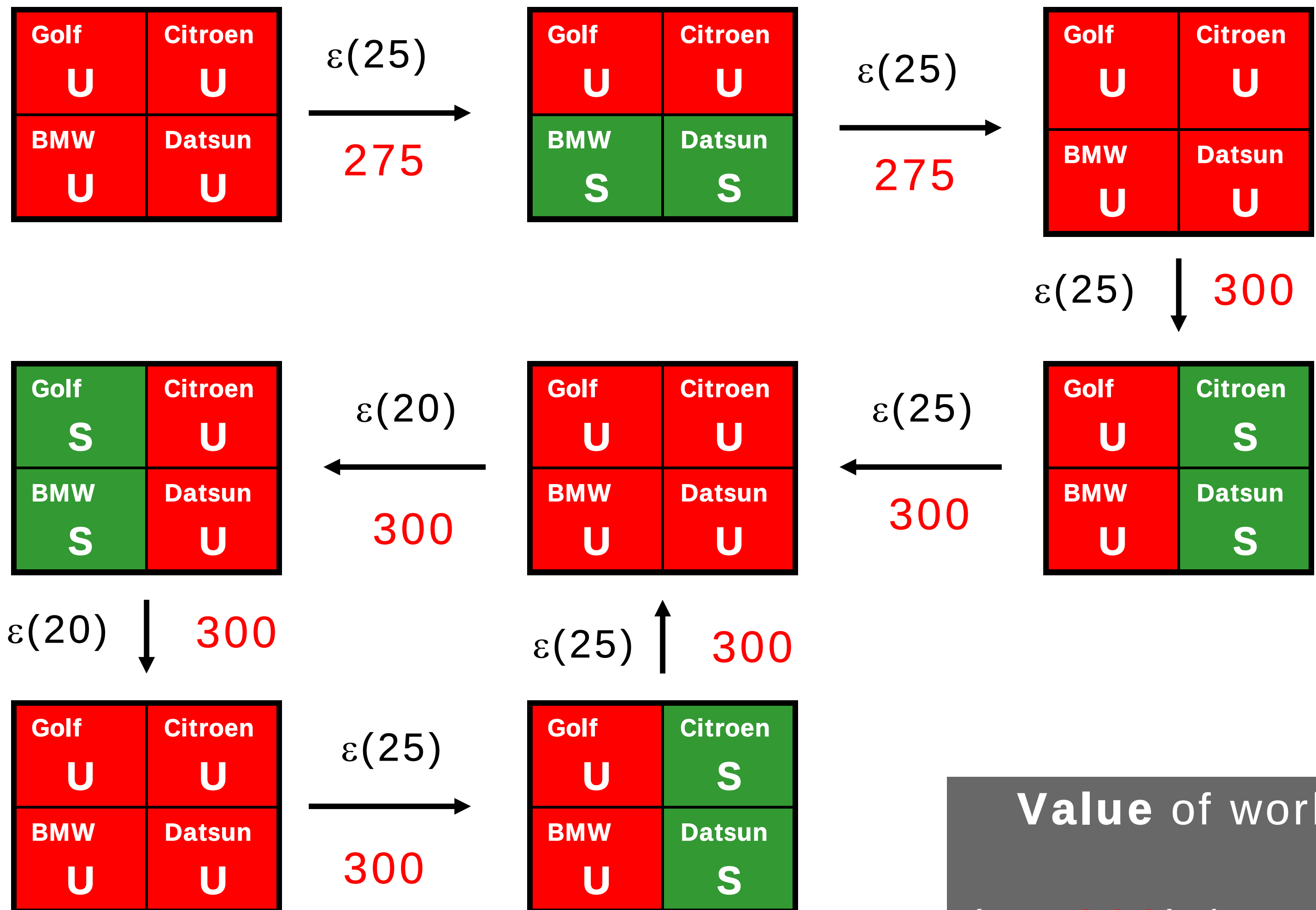
$$\sum_i C_i / \sum_i t_i = 17/2 = 8.5$$

EXAMPLE: Optimal WORK plan for cars with different subscription rates for city driving !



CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Workplan I



Value of workplan:

(4 x **300**) / 90 = **13.33**

UCb

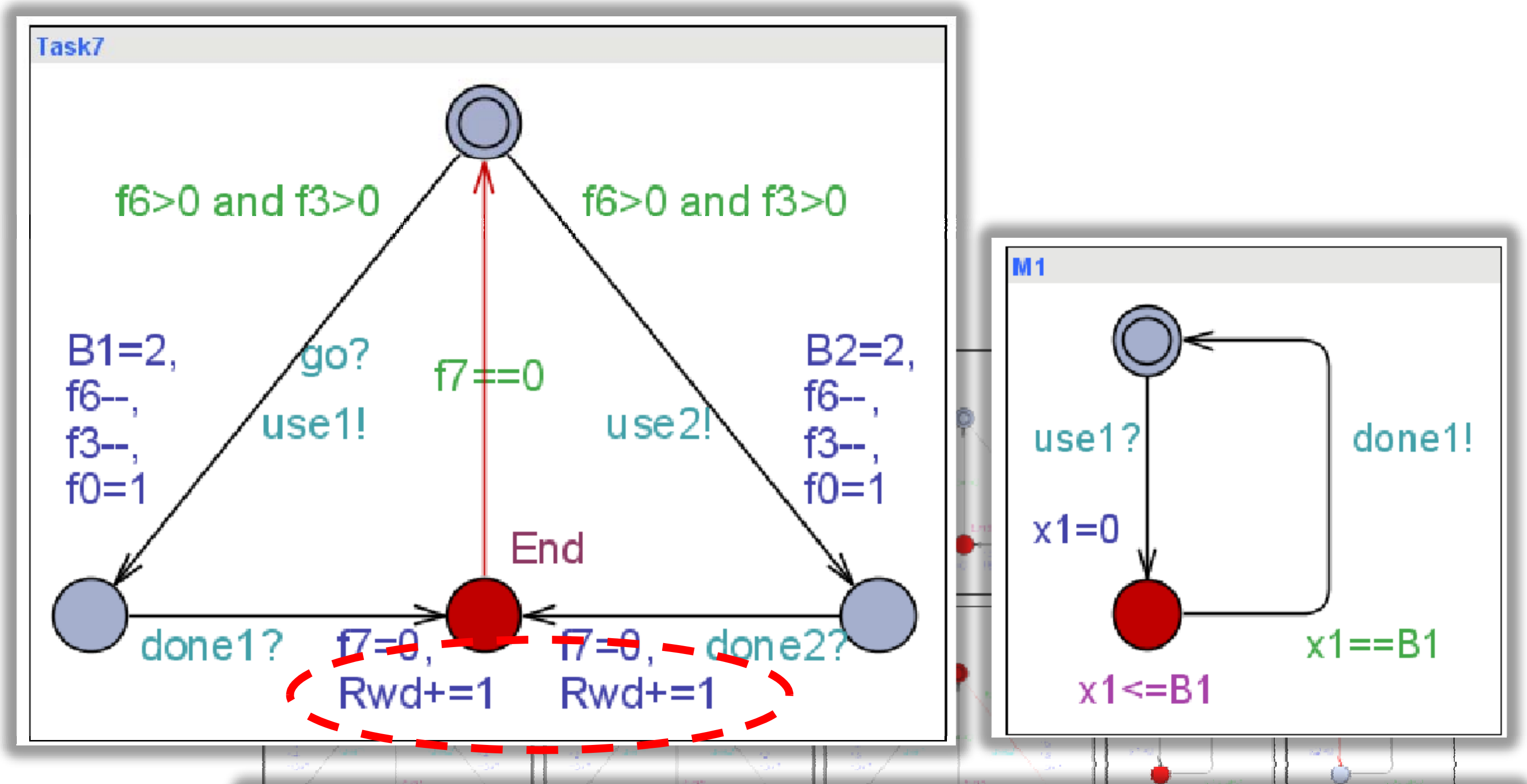
Workplan II

Value of workplan:

560 / 100 = **5.6**

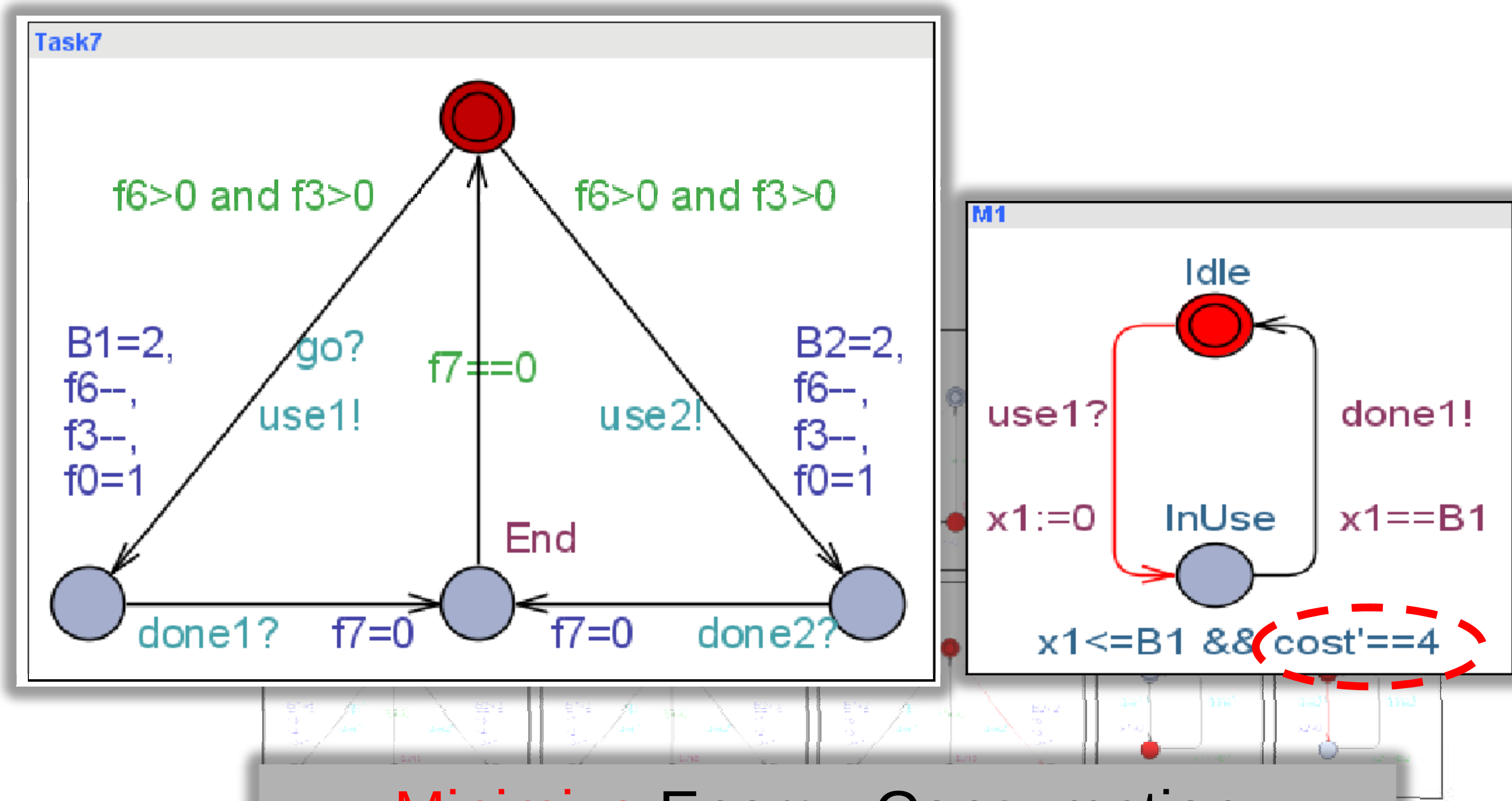
UCI

Optimal Infinite Scheduling



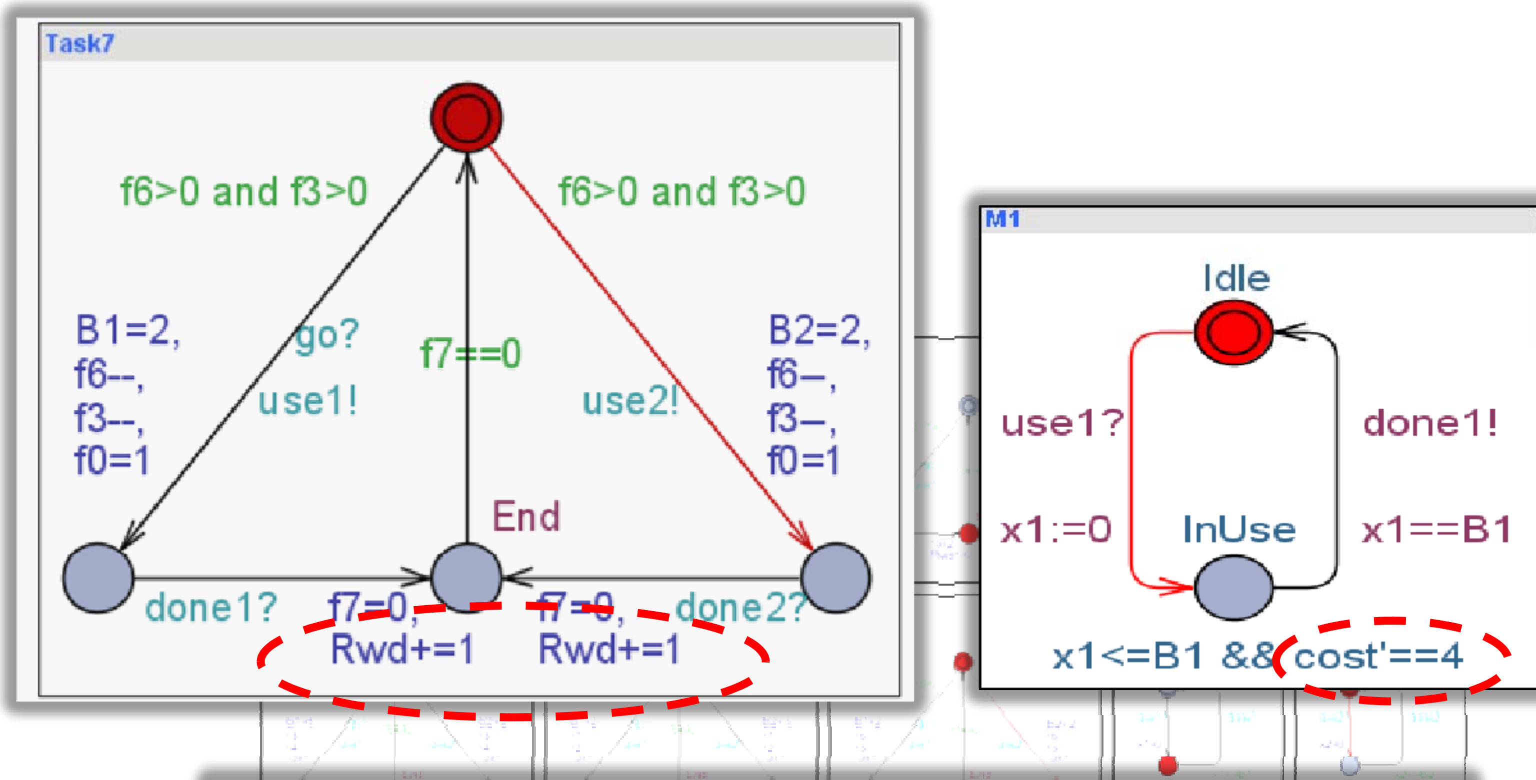
Maximize throughput:
i.e. maximize Reward / Time in the long run!

Optimal Infinite Scheduling



Minimize Energy Consumption:
i.e. minimize Cost / Time in the long run

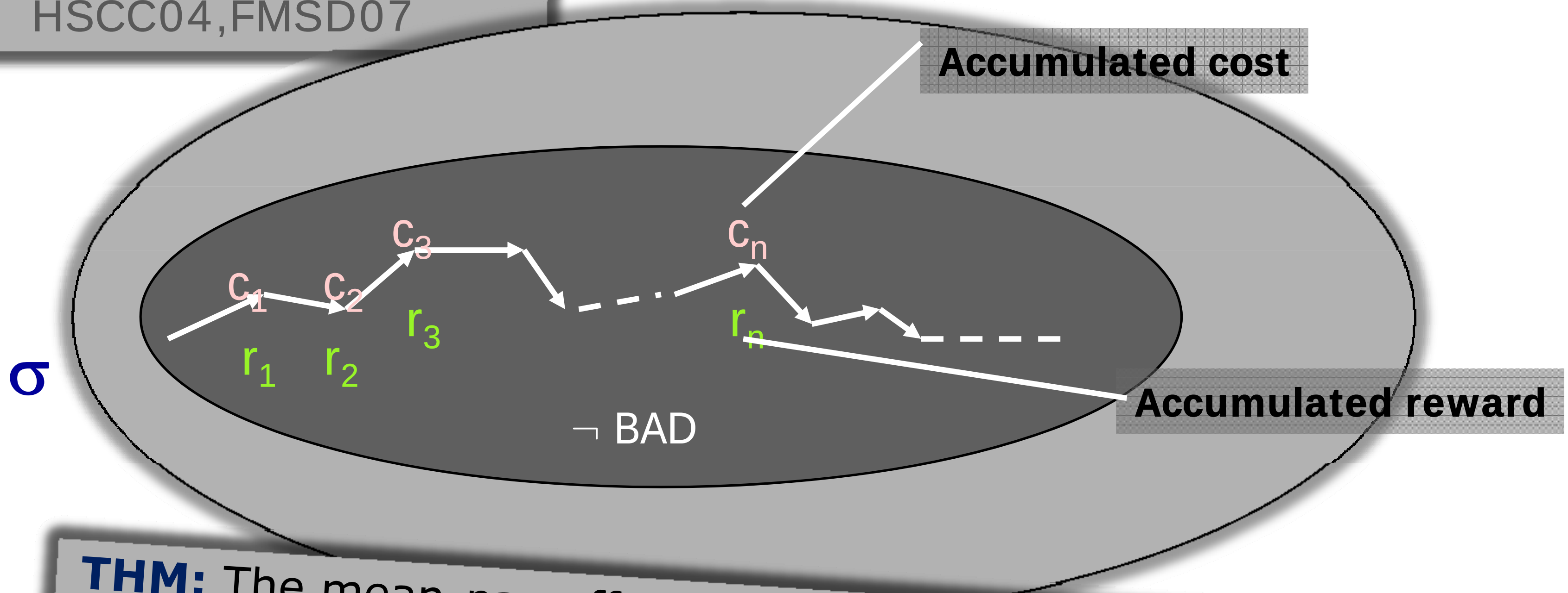
Optimal Infinite Scheduling



Maximize throughput:
i.e. maximize Reward / Cost in the long run

Mean Pay-Off Optimality

Bouyer, Brinksma, Larsen:
HSCC04,FMSD07



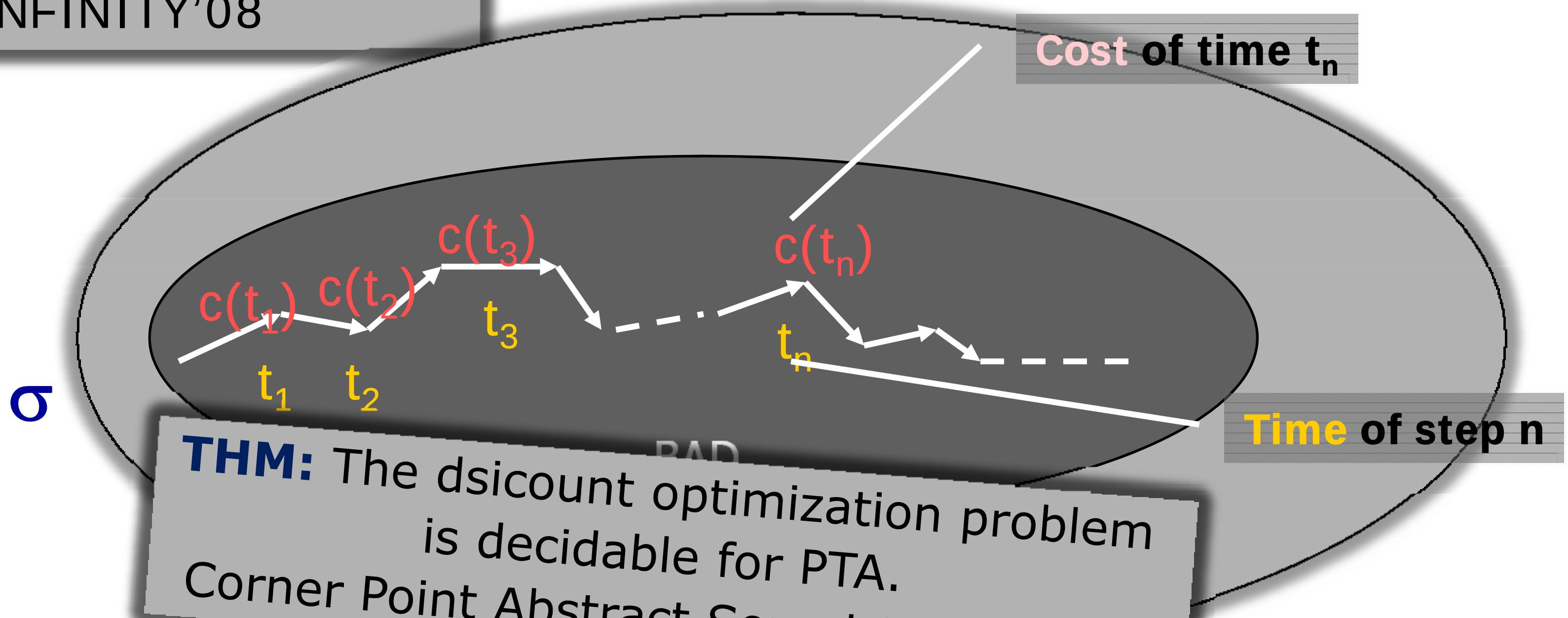
THM: The mean-pay off optimization problem is decidable (and PSPACE-complete) for PTA.
Corner Point Abstract Sound & Complete

Optimal Schedule σ^*: $val(\sigma^*) = \inf_\sigma val(\sigma)$

Discount Optimality

$\lambda < 1$: discounting factor

Larsen, Fahrenberg:
INFINITY'08



THM: The dsicount optimization problem is decidable for PTA.
Corner Point Abstract Sound & Complete

Value of path σ: $\text{val}(\sigma) = \int_{t=0}^{t=\infty} c(t)\lambda^t dt$

Optimal Schedule σ^*: $\text{val}(\sigma^*) = \inf_\sigma \text{val}(\sigma)$

Soundness of Corner Point Abstraction

Lemma
Let Z be a (bounded, closed) zone and let f be a(well-defined) function over Z defined by:

$$f : (t_1, \ldots, t_n) \mapsto \frac{a_1 t_1 + \cdots + a_n t_n + a}{c_1 t_1 + \cdots + c_n t_n + d}$$

then $\inf_Z f$ is obtained at a corner-point of Z (with integer coefficients).

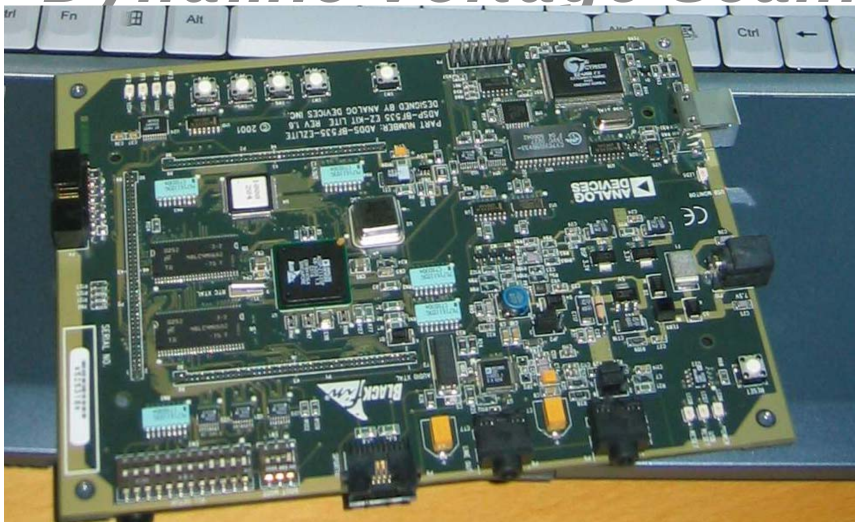
Lemma
Let Z be a (bounded, closed) zone and let f be a function over Z defined by:

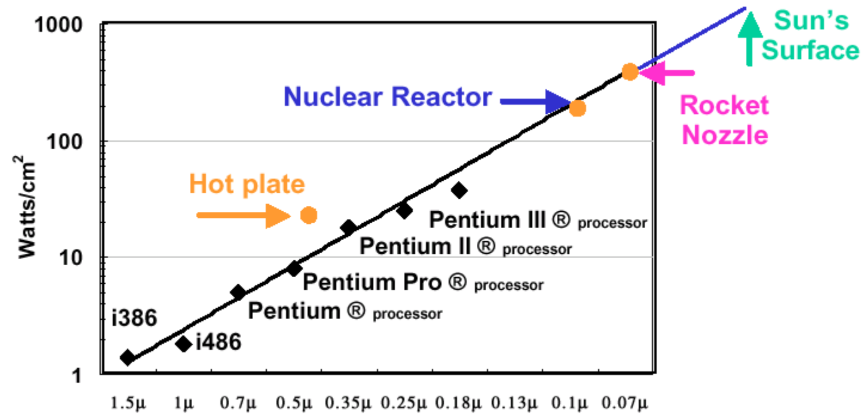
$$f : (t_1, \ldots, t_n) \mapsto a_1 \lambda^{t_1} + \cdots a_n \lambda^{t_n} + a$$

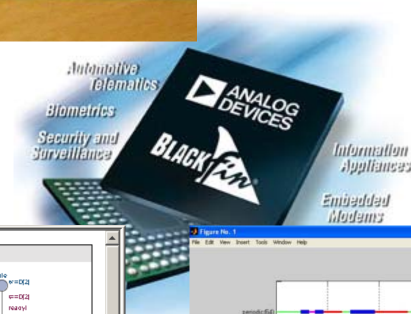
then $\inf_Z f$ is obtained at a corner-point of Z (with integer coefficients).

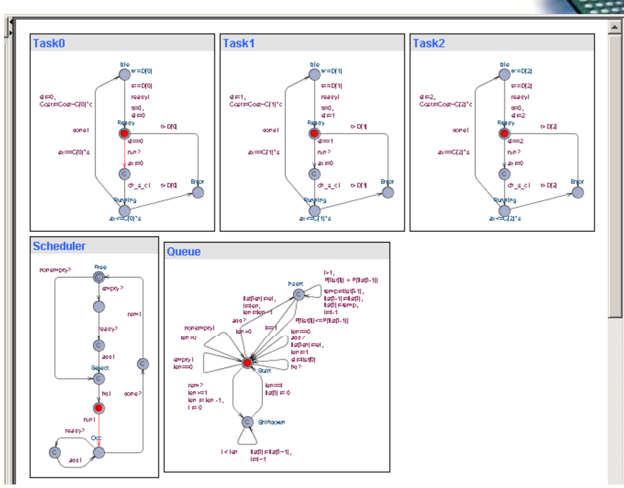
Application

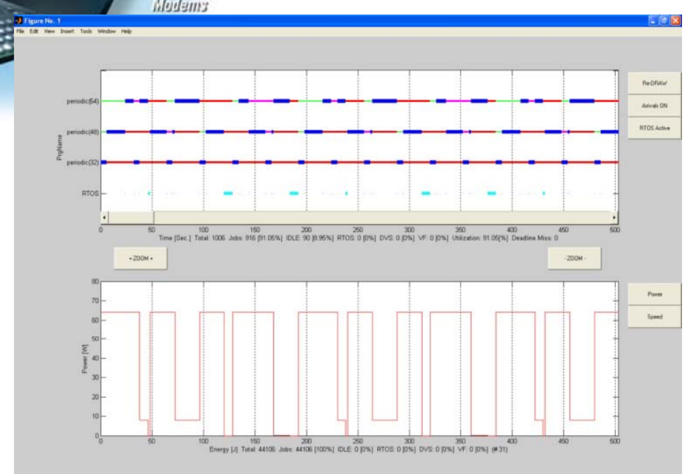
Dynamic Voltage Scaling

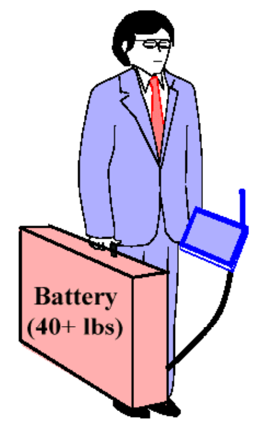




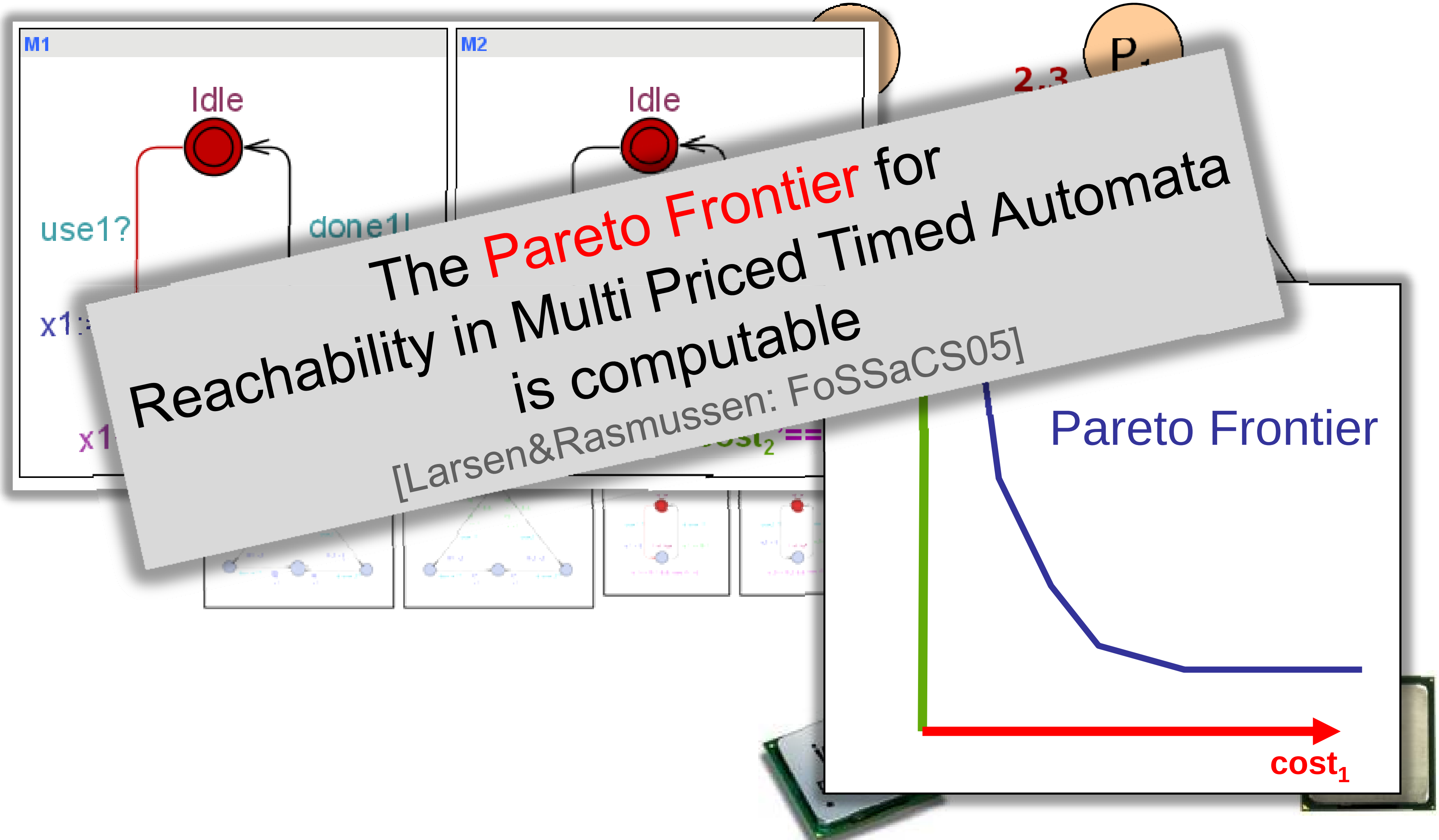








Multiple Objective Scheduling



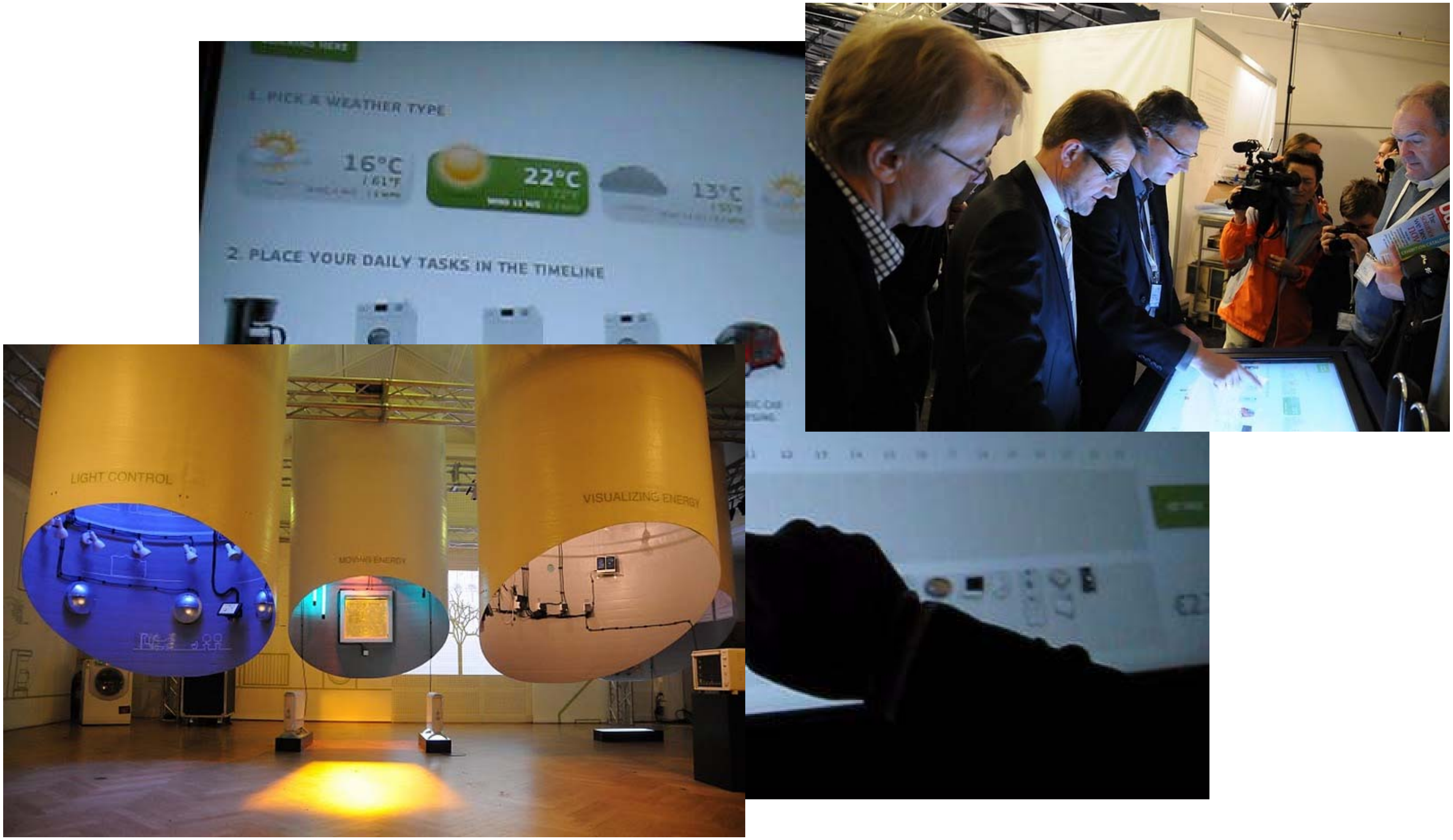
The Pareto Frontier for Reachability in Multi Priced Timed Automata is computable

[Larsen&Rasmussen: FoSSaCS05]

"Experimental" Results

Warehouse
iTunes

”Experimental” Results



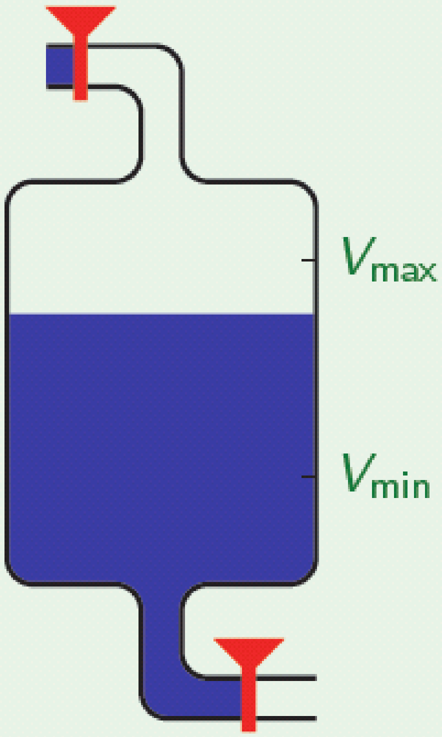
Energy Automata

Managing Resources

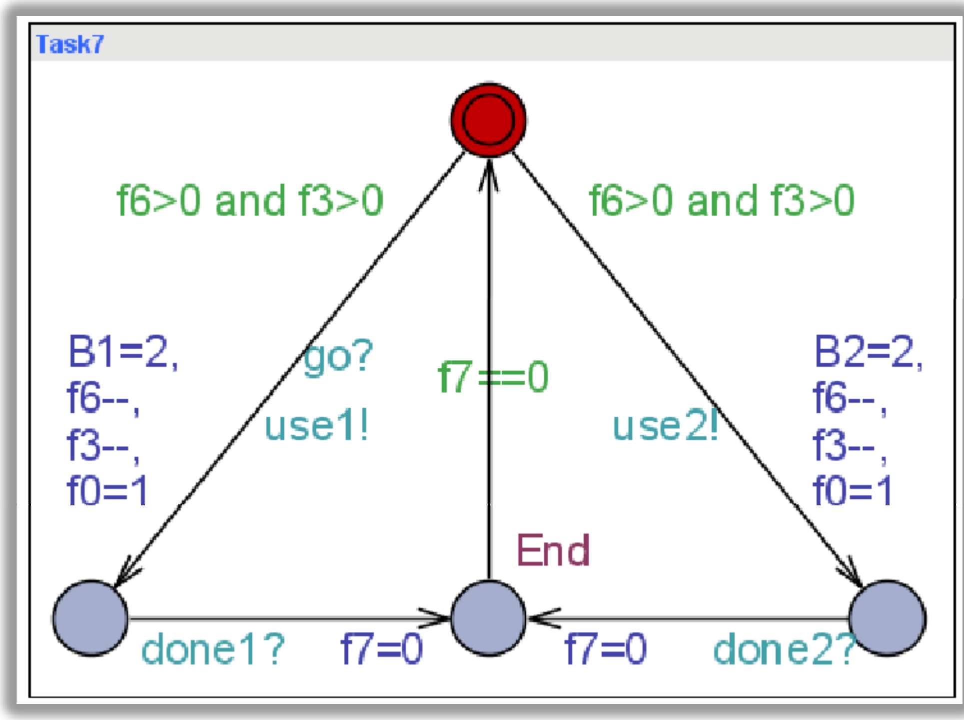
Example

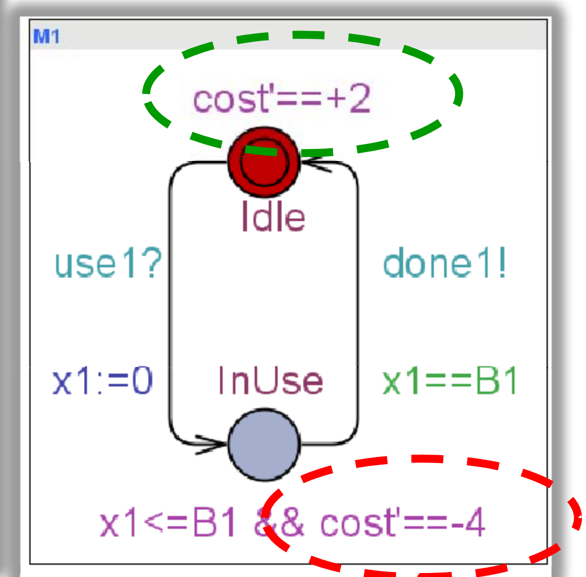
In some cases, resources can both be consumed and regained.

The aim is then to keep the level of resources within given bounds.



Consuming & Harvesting Energy

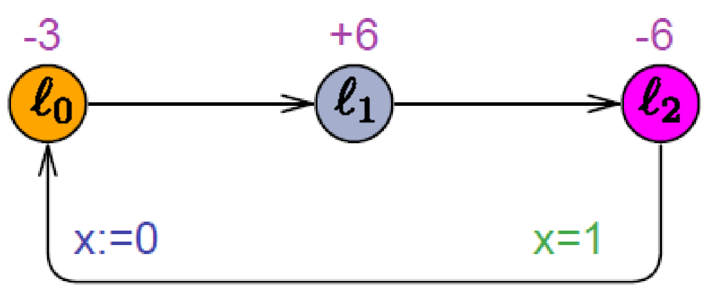


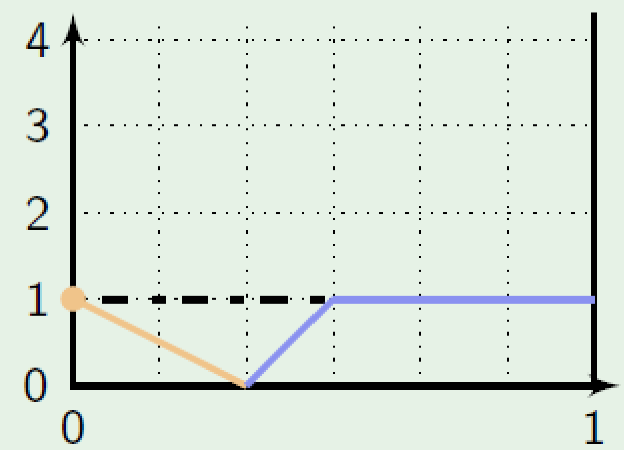


Maximize throughput
while respecting: $0 \leq \mathbf{E} \leq$ MAX

Energy Constrains

- Energy is not only consumed but may also be regained
- The aim is to continously satisfy some energy constriants





lower-weak-upper-bound problem

Results (so far)

Bouyer, Fahrenberg, Larsen, Markey, Srba: **FORMATS 2008**

Untimed

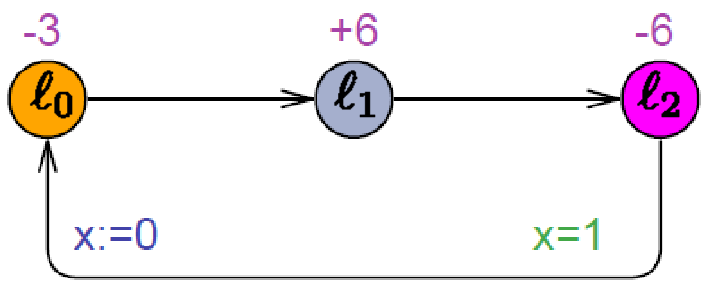
	games	existential problem	universal problem
L	$\in$ UP $\cap$ coUP P-h	$\in$ P	$\in$ P
L+W	$\in$ NP $\cap$ coNP P-h	$\in$ P	$\in$ P
L+U	EXPTIME-c	$\in$ PSPACE NP-h	$\in$ P

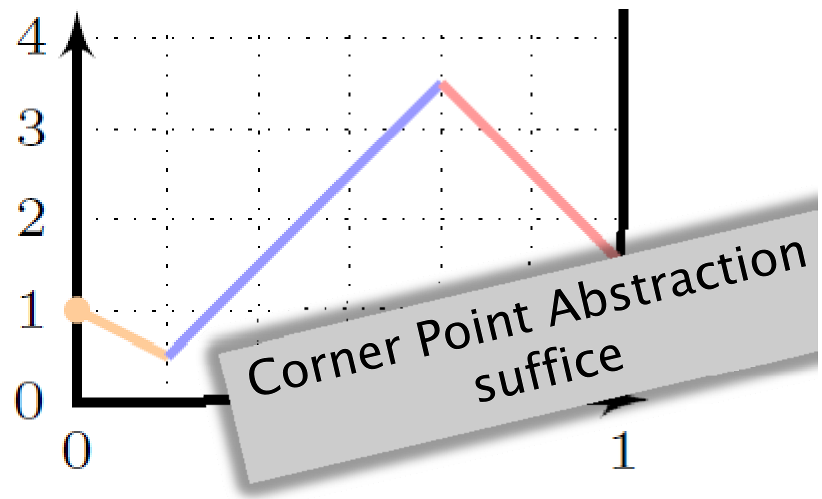
1 Clock

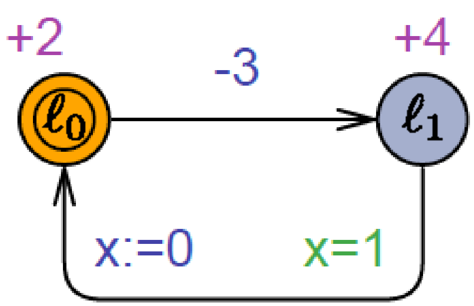
	games	existential problem	universal problem
L	?	$\in$ P	[illegible]
L+W	?	$\in$ P	$\in$ P
L+U	undecidable	?	?

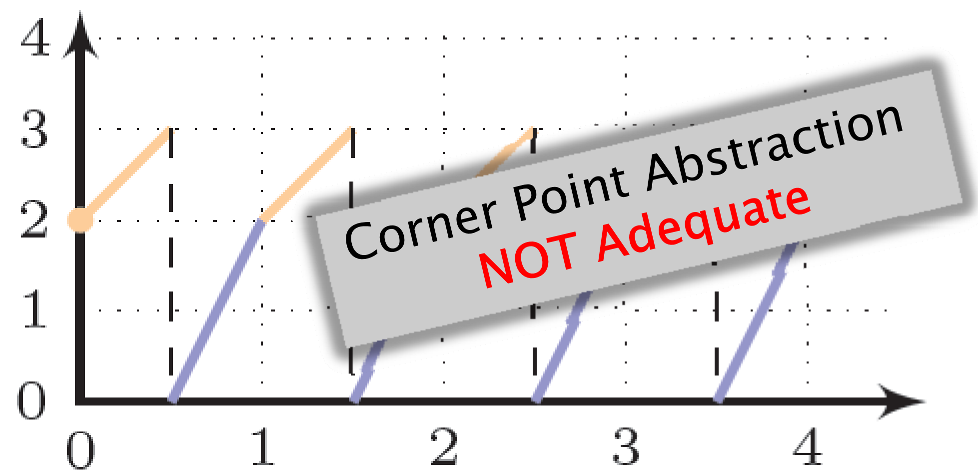
Corner Point Abstraction Suffice

Discrete Updates on Edges

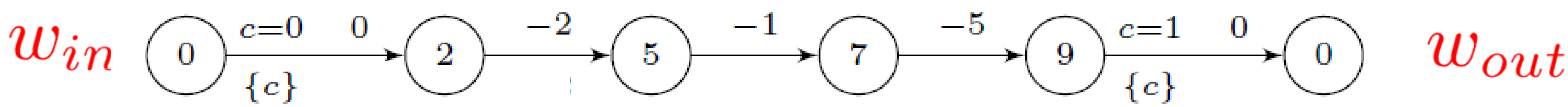




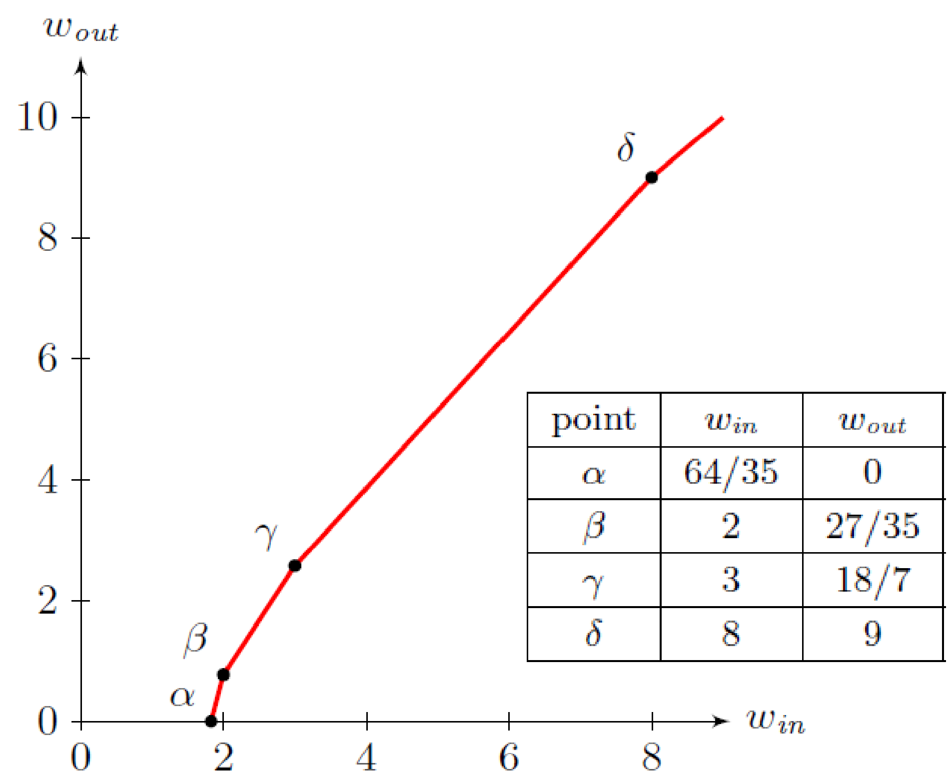




New Approach: Energy Functions

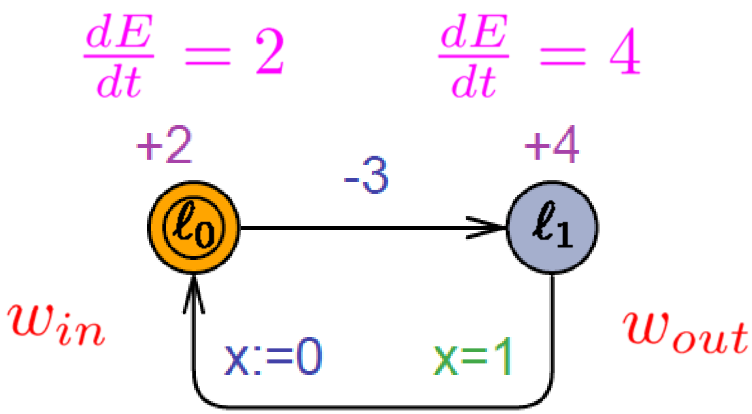


- Maximize energy along paths
- Use this information to solve general problem

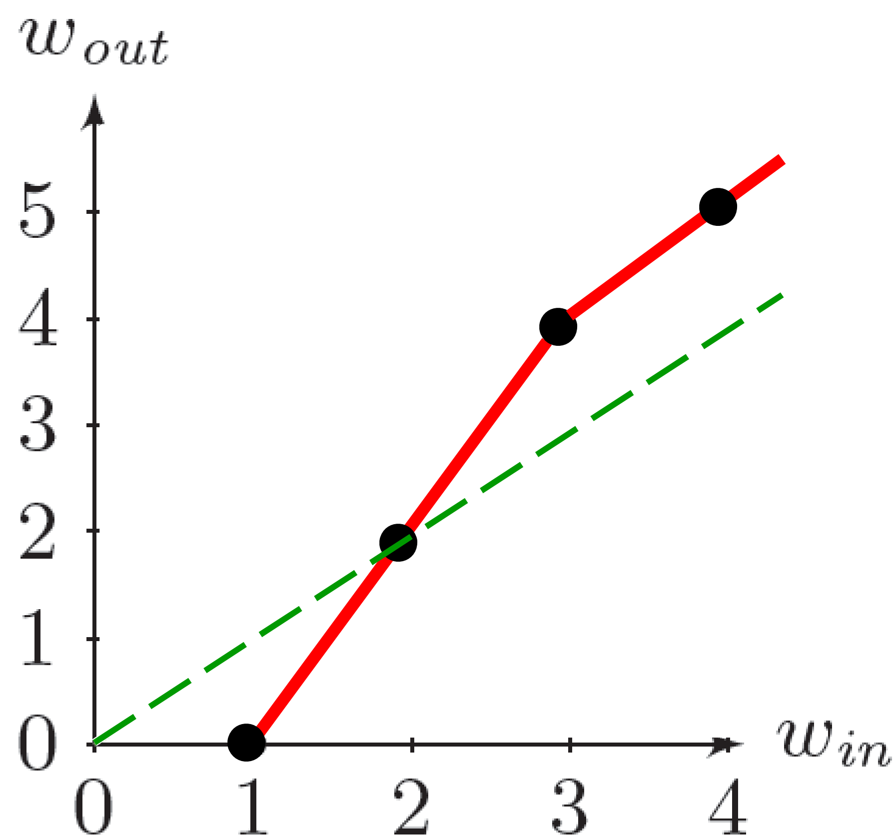


point	w_{in}	w_{out}
α	$64/35$	0
β	2	$27/35$
γ	3	$18/7$
δ	8	9

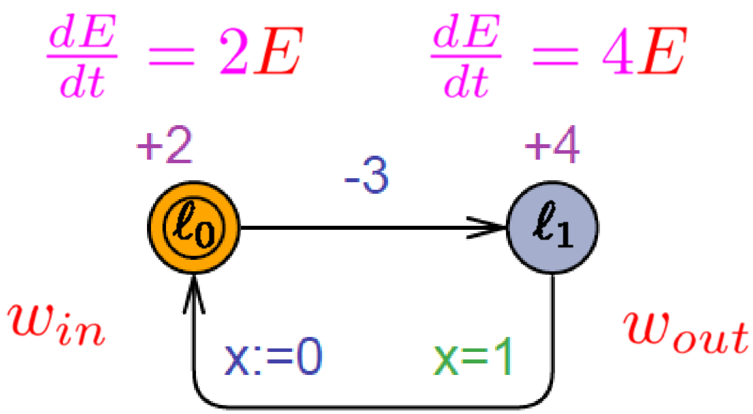
Energy Function

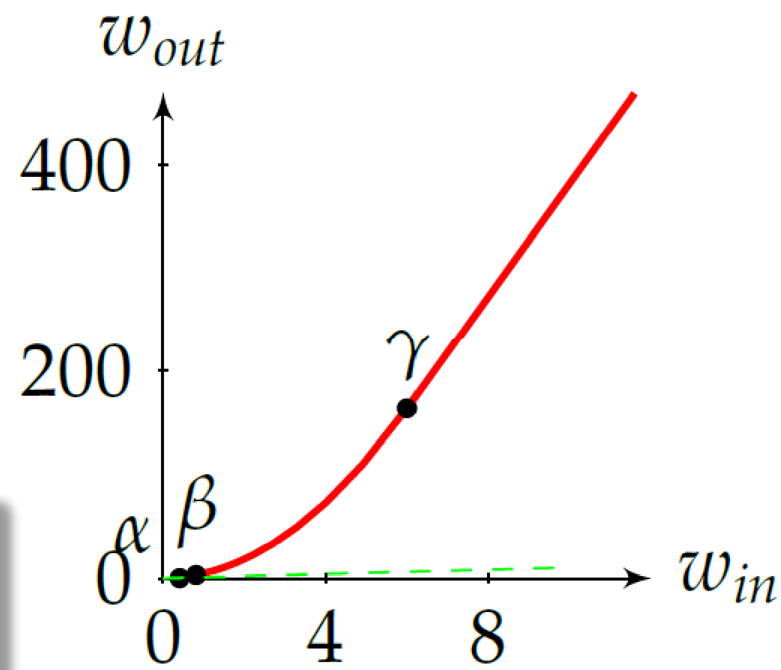


General Strategy
Spend just enough time to survive the next negative update



Exponential PTA





General Strategy

Spend just enough time

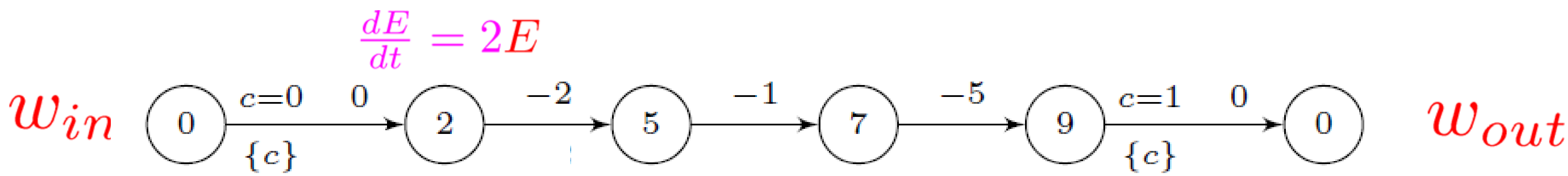
~~to survive the next negative update~~

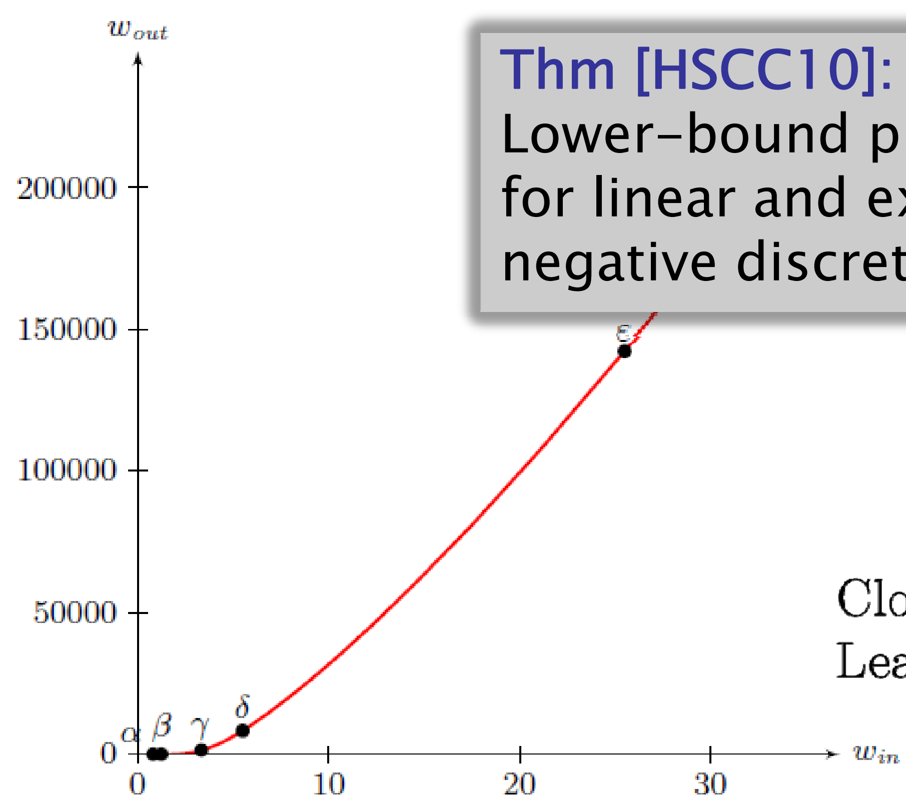
so that after next negative update there is a certain positive amount !

Minimal Fixpoint:

$$\frac{3}{e^2-1} \approx 0.47$$

Exponential PTA





Thm [HSCC10]:
Lower-bound problem is decidable for linear and exponential 1-clock PTAs with negative discrete updates.

- $f : x \mapsto \alpha \cdot x^r + \beta$ where r is rational
- $\frac{df}{dt} \geq 1$

Closed under max and composition.
Least fixed point computable.

Conclusion

- Priced Timed Automata a uniform framework for modeling and solving dynamic ressource allocation problems!
- Not mentioned here:
  - Model Checking Issues (ext. of CTL and LTL).
- Future work:
  - Zone-based algorithm for optimal infinite runs.
  - Approximate solutions for priced timed games to circumvent undecidablity issues.
  - Open problems for Energy Automata.
  - Approximate algorithms for optimal reachability