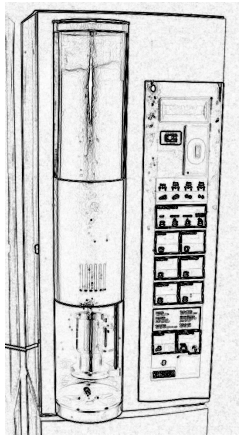
Quantitative Verification and Synthesis, of Embedded Systems

Kim G. Larsen

CISS – Aalborg University

DENMARK

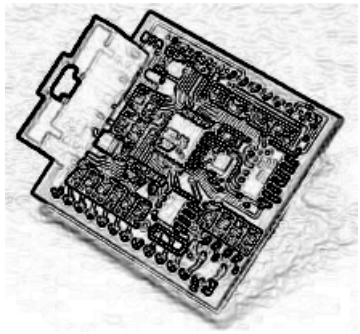
Embedded Systems



Plant
Continuous

sensors

actuators



Controller Program
Discrete

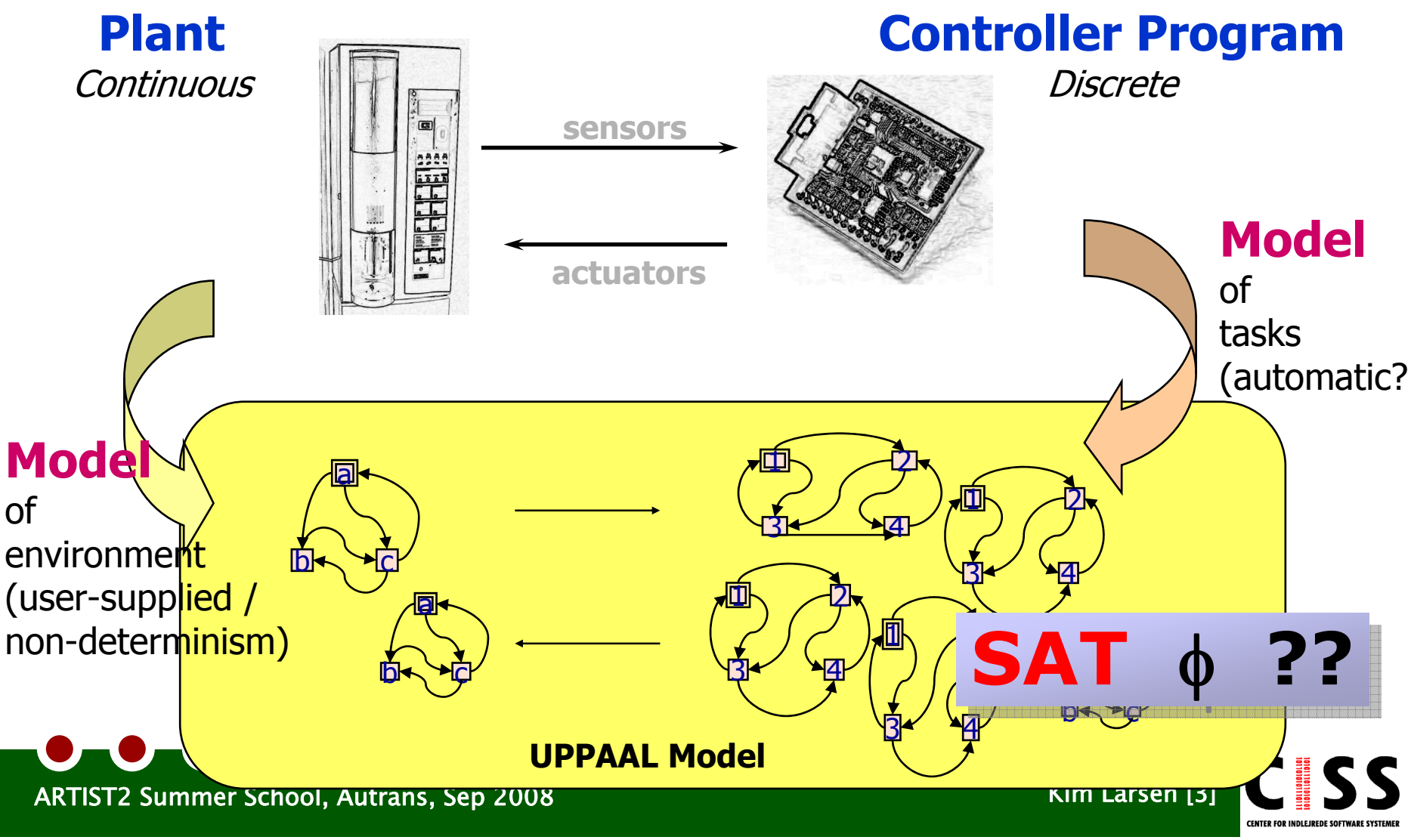
Eg.:

Realtime Protocols
Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

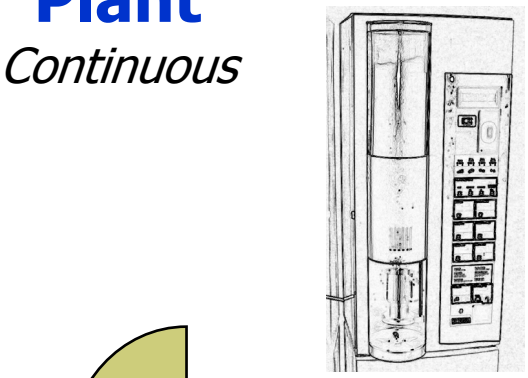
Quantities:

- timing
- energy
- memory
- bandwidth
- uncertainties

Verification



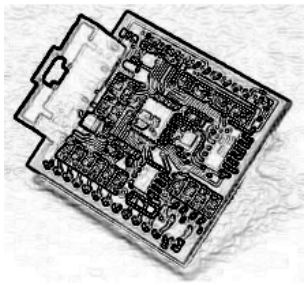
Synthesis



Plant
Continuous

sensors

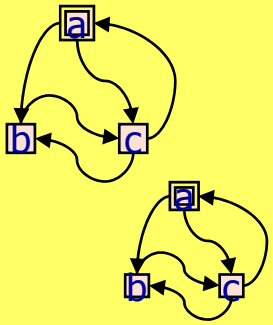
actuators

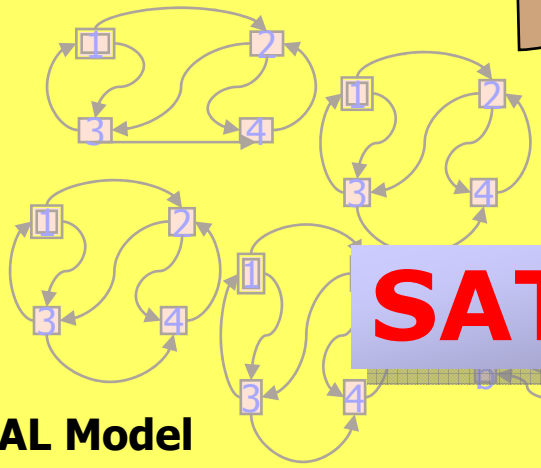


Controller Program
Discrete

Synthesis
of
tasks
(automatic)

Model
of
environment
(user-supplied)

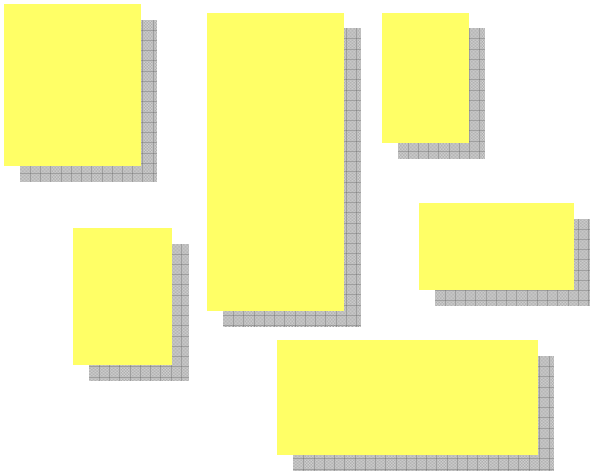




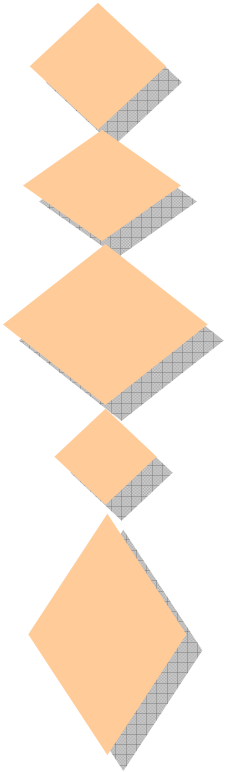
SAT ϕ !!

Partial UPPAAL Model

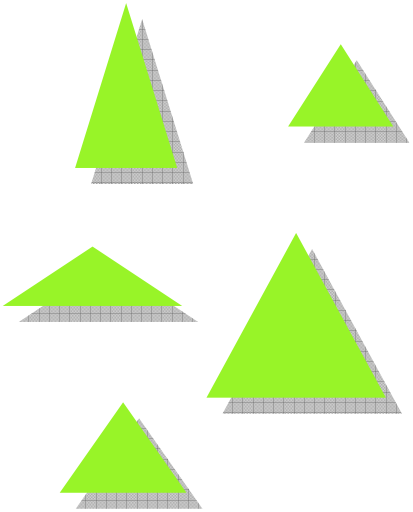
Embedded Systems → Scheduling



Tasks:
Computation times
Deadlines
Dependencies
Arrival patterns
uncertainties



Scheduling Principles (OS)
EDF, FPS, RMS, DVS, ..



Resources
Execution platform
PE, Memory
Networks
Drivers
uncertainties

Approach – Timed Automata

- Tasl
- Sch
  - Verify that given SP ensures deadlines.

- Performance Evaluation
  - Estimate resources (e.g. energy) required by given
- Sc Synthesis
  - (optimal) SP
  - given objective.
  - *Scheduling*: SP controls

CLASSIC

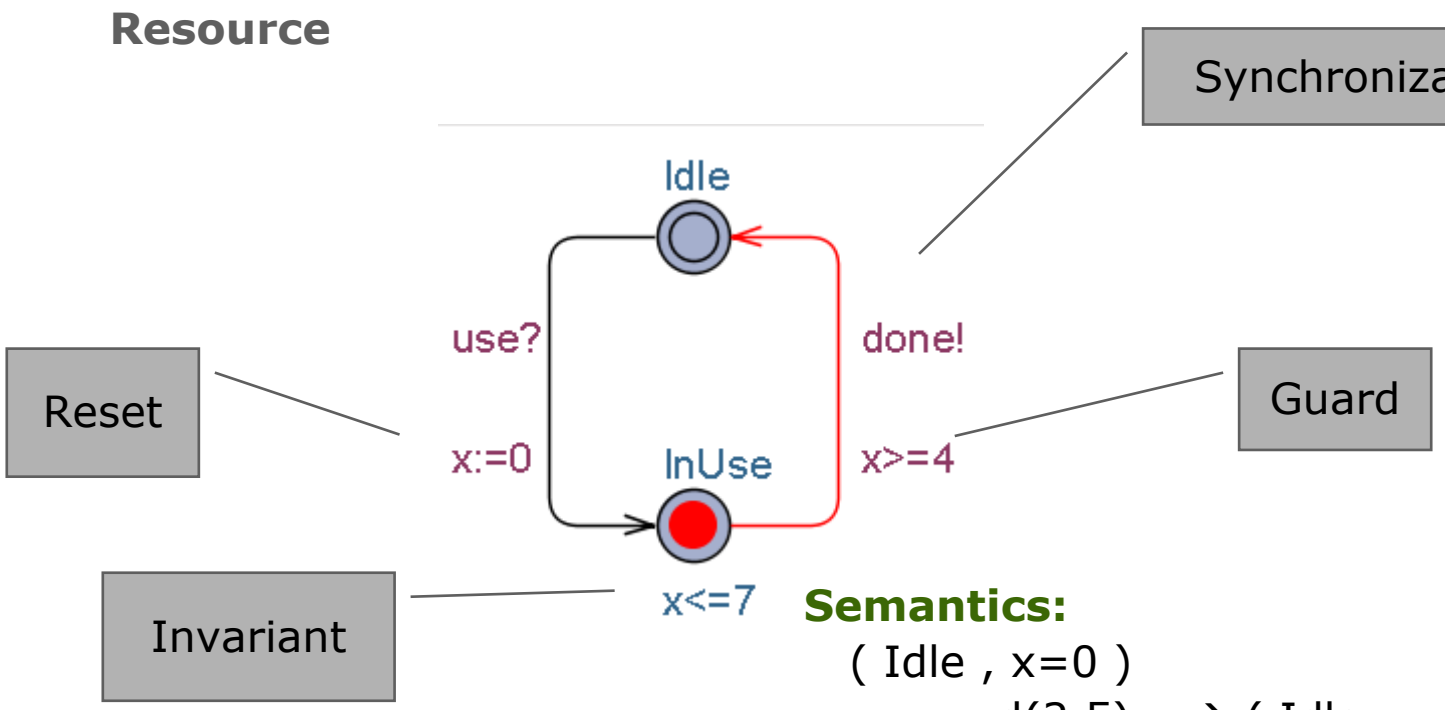
CORA

TIGA

TALK:
What can we do?
What can we do efficiently?
What would we like to do?
What can not be done?

Timed Automata

Timed Automata [Alur & Dill'89]

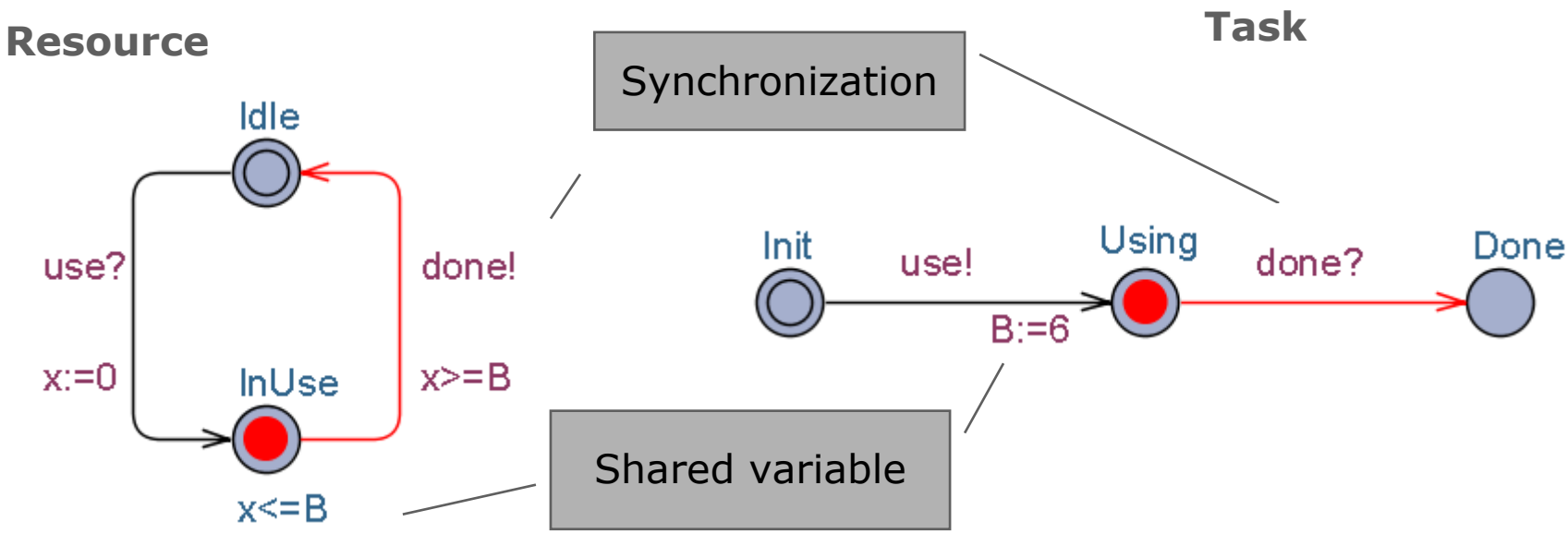


Semantics:

(Idle , x=0)

d(2.5)	→ (Idle , x=2.5)
use?	→ (InUse , x=0)
d(5)	→ (InUse , x=5)
done!	→ (Idle , x=5)
d(3)	→ (Idle , x=8)
use?	→ (InUse , x=0)

Composition



Semantics:

(Idle , Init , B=0, x=0)

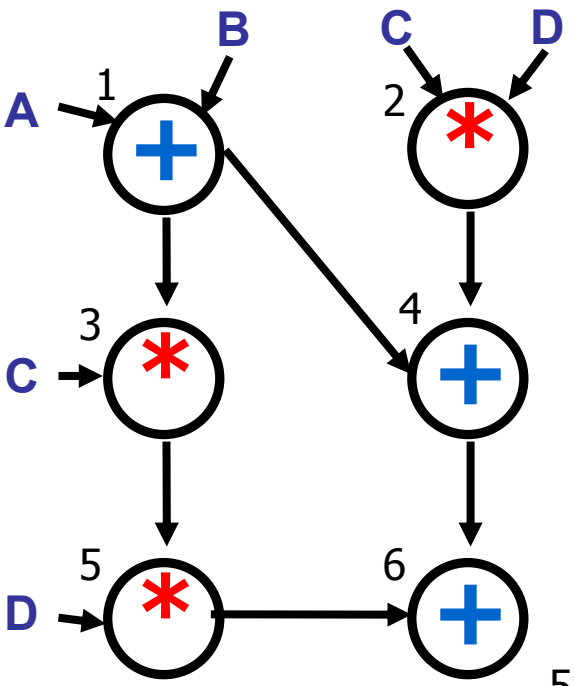
d(3.1415) → (Idle , Init , B=0 , x=3.1415)

use → (InUse , Using , B=6, x=0)

d(6) → (InUse , Using , B=6, x=6)

done → (Idle , Done , B=6 , x=6)

Task Graph Scheduling – Example



Compute :

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

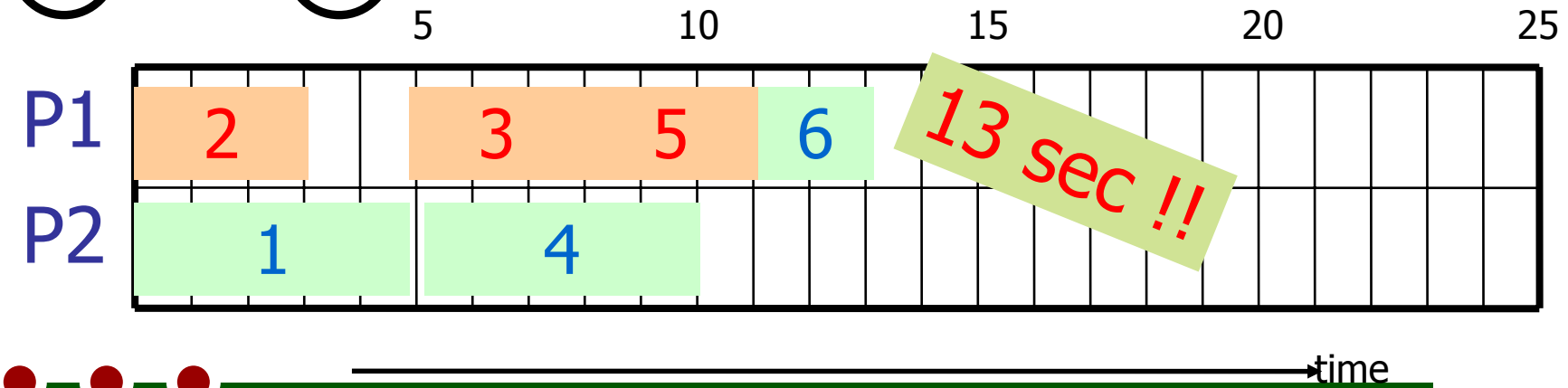
using 2 processors

P1 (fast)

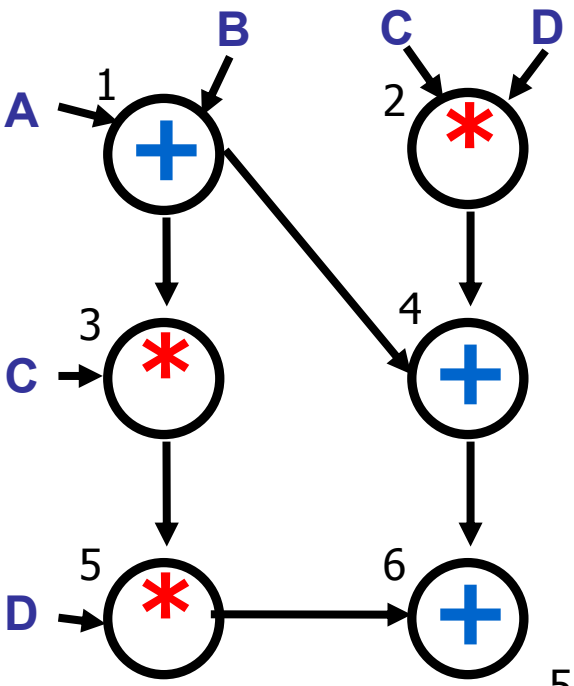
+	2s
*	3s

P2 (slow)

+	5s
*	7s



Task Graph Scheduling – Example



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

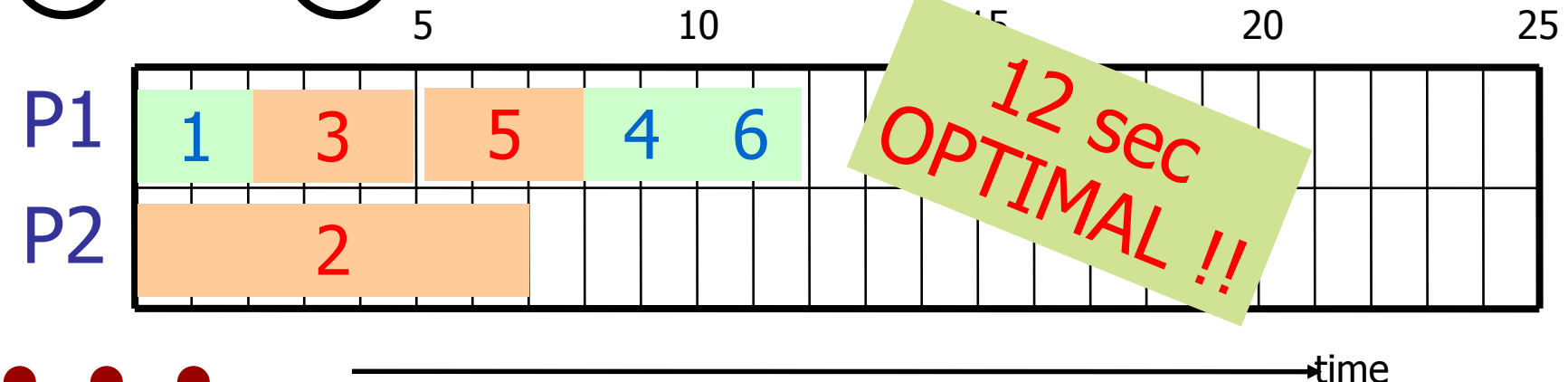
using 2 processors

P1 (fast)

+	2s
*	3s

P2 (slow)

+	5s
*	7s



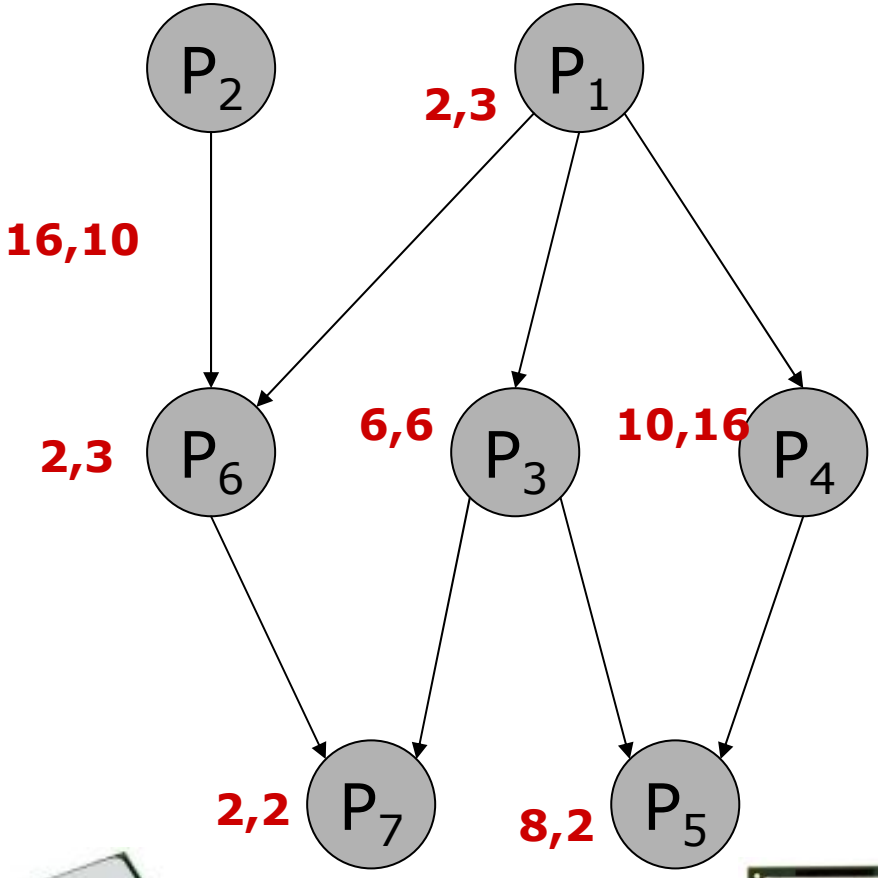
Task Graph Scheduling

Optimal Static Task Scheduling

- Task **P**={P_1,.., P_m}
- Machines **M**={M_1,..,M_n}
- Duration Δ : **(P£M) ! N_1**
- < : p.o. on **P** (pred.)

- A task can be executed only if all predecessors have completed
- Each machine can process at most one task at a time
- Task cannot be preempted.

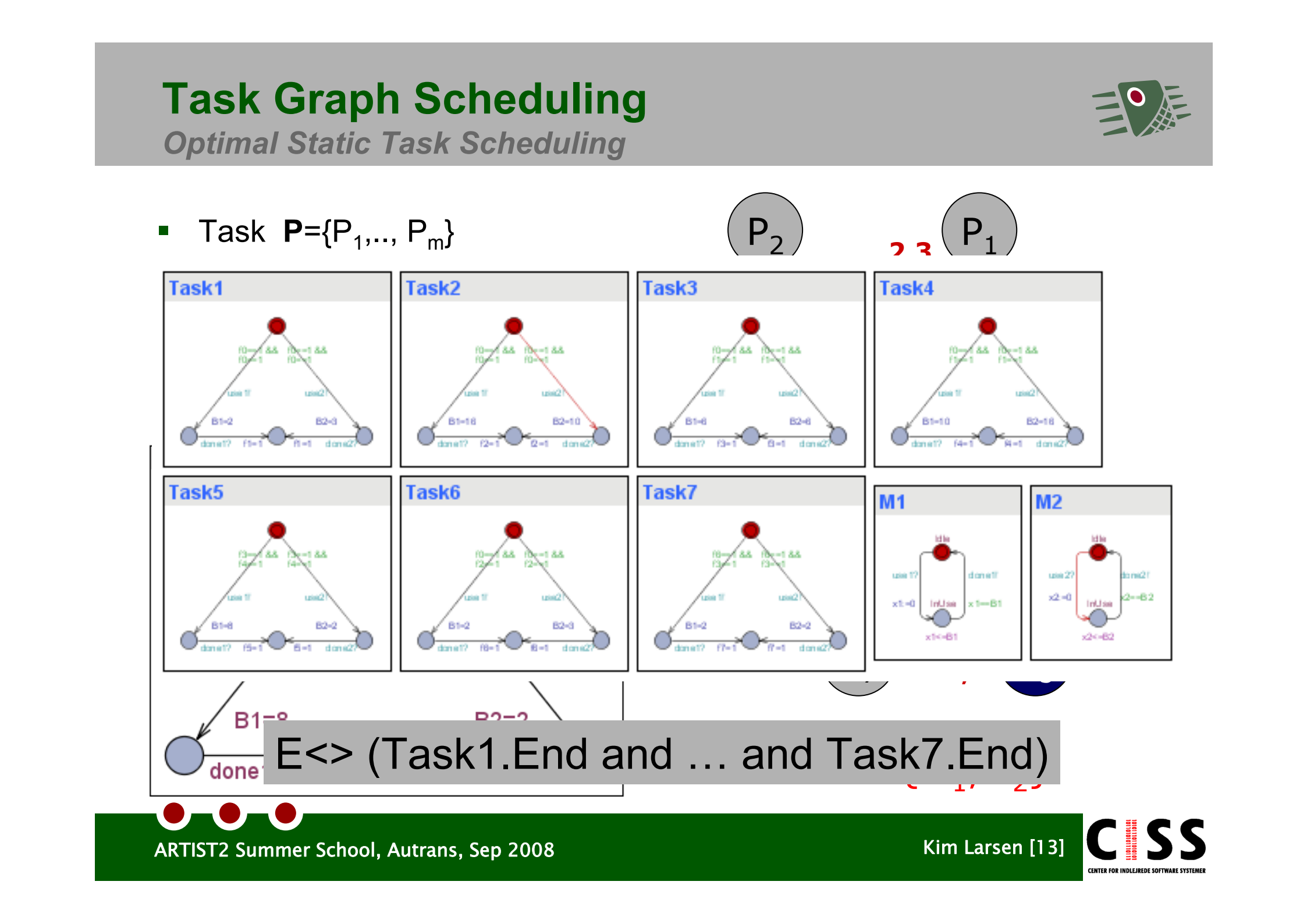
- Compute schedule with minimum completion-time!



M = {M_1,M_2}

Task Graph Scheduling

Optimal Static Task Scheduling



- Task **P**={P_1,.., P_m}

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

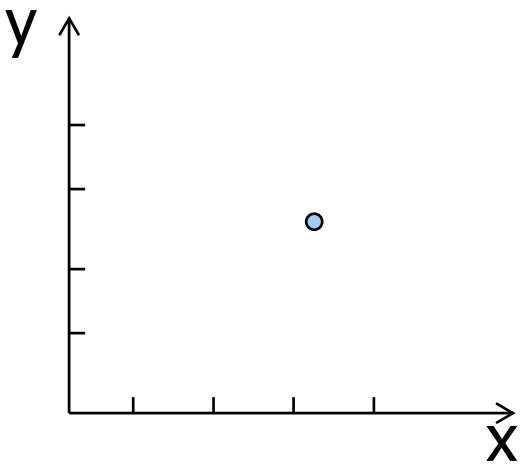
Symbolic A*
Branch-&-Bound
60 sec

Abdeddaïm, Kerbaa, Maler

Zones -- *From infinite to finite*

State

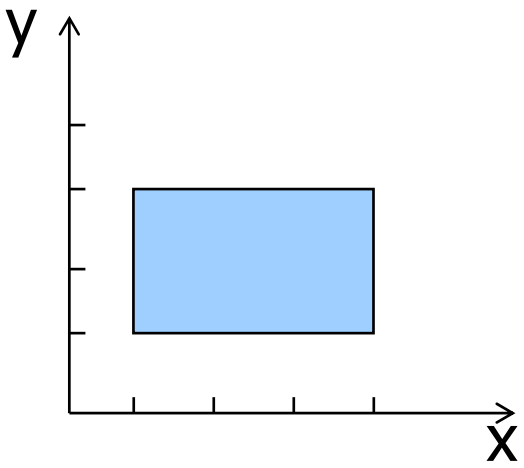
(n, x=3.2, y=2.5)



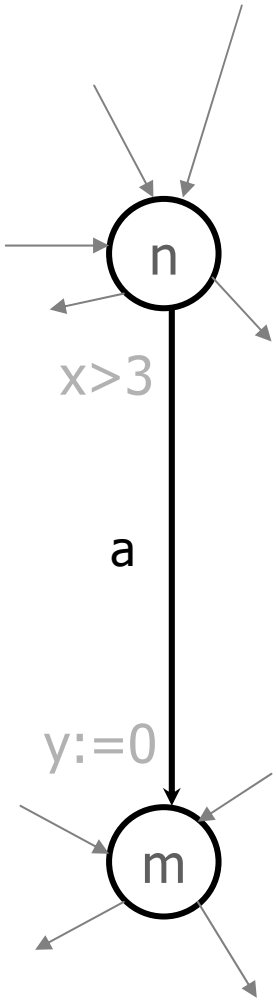
Symbolic state (set)

(n, 1 ≤ x ≤ 4, 1 ≤ y ≤ 3)

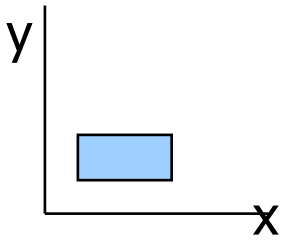
Zone:
conjunction of
x-y<=n, x<=>n



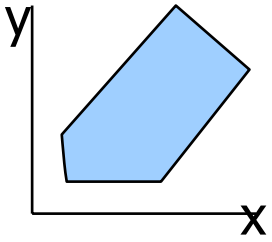
Symbolic Transitions *using Zones*



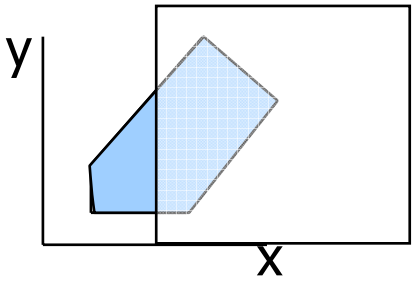
1<=x<=4
1<=y<=3



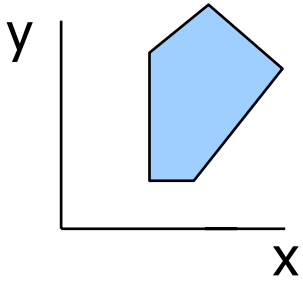
delays to



1<=x, 1<=y
-2<=x-y<=3



conjuncts to



3<x, 1<=y
-2<=x-y<=3

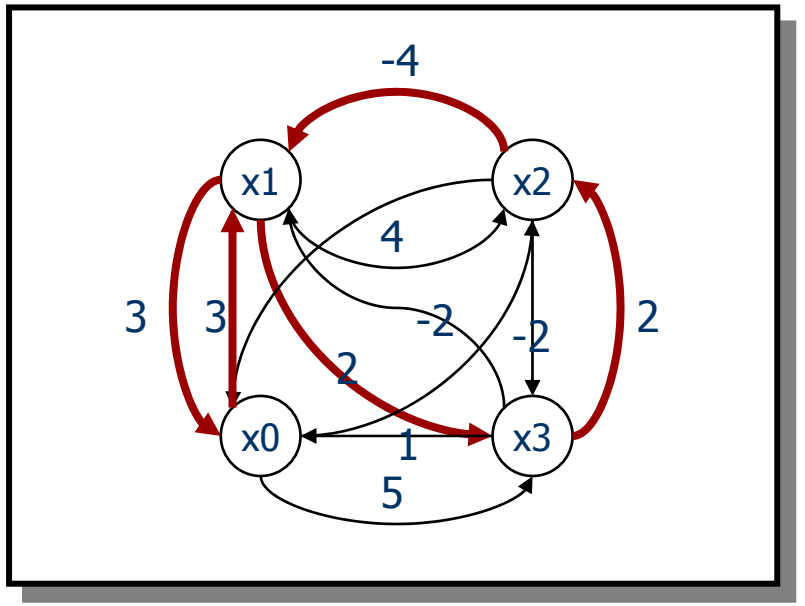
projects to

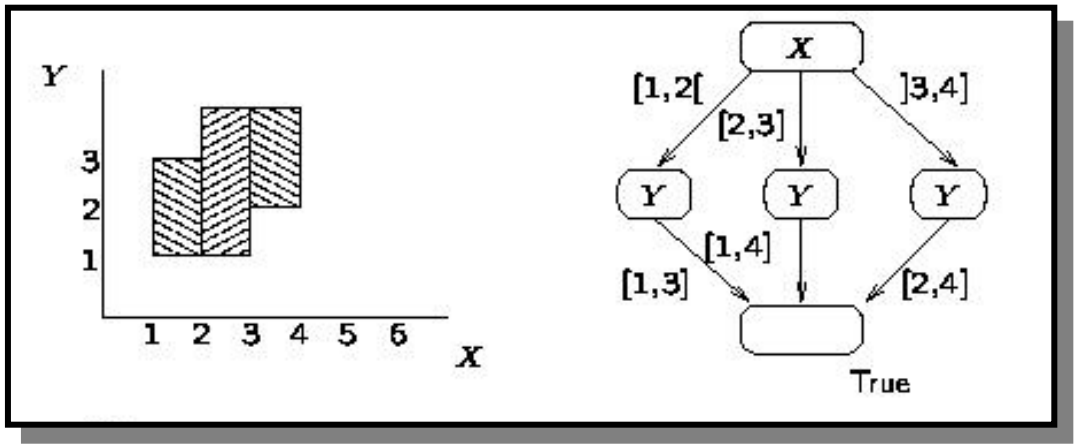
3<x, y=0

Thus (n,1<=x<=4,1<=y<=3) =a => (m,3<x, y=0)

Datastructures for Zones

- Difference Bounded Matrices (DBMs)
- Minimal Constraint Form
  [RTSS97]
- Clock Difference Diagrams
  [CAV99]
- PW List
  [SPIN03]



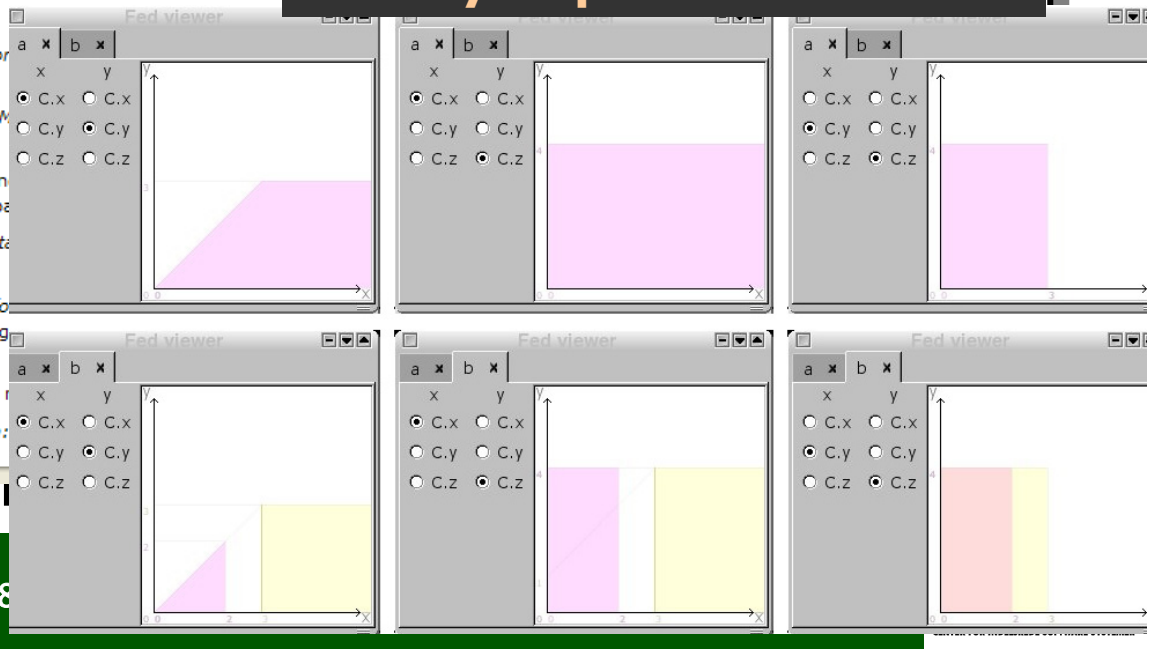


Datastructures for Zones

- Difference Bounded
-
-
-
-

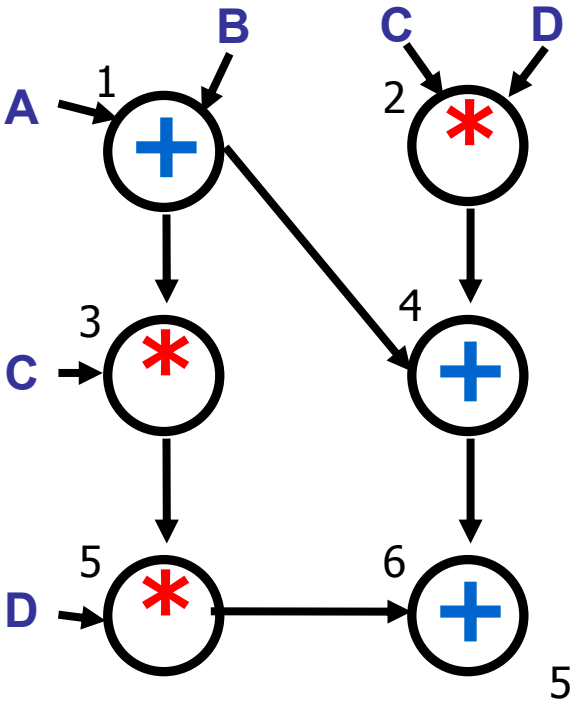
Alexandre David

Elegant RUBY bindings for easy implementations



Priced Timed Automata

Task Graph Scheduling – Revisited



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

using 2 processors

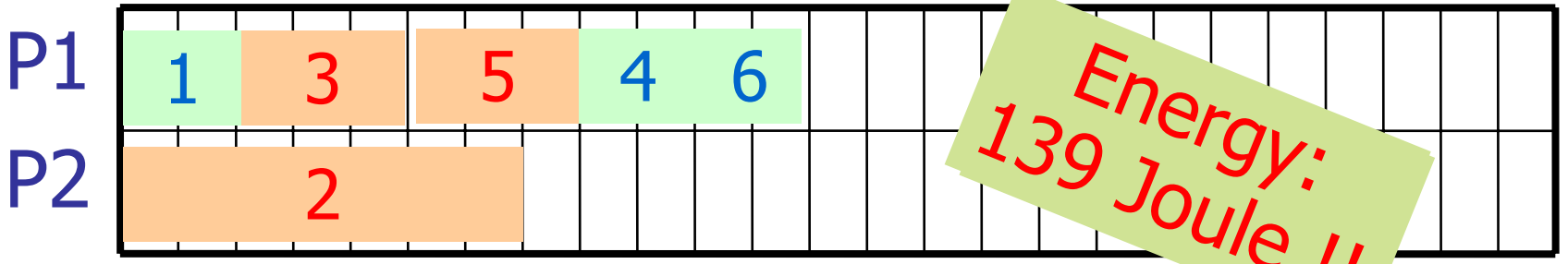
P1 (fast)

+	2s
*	3s
Idle	1W
In use	9W

P2 (slow)

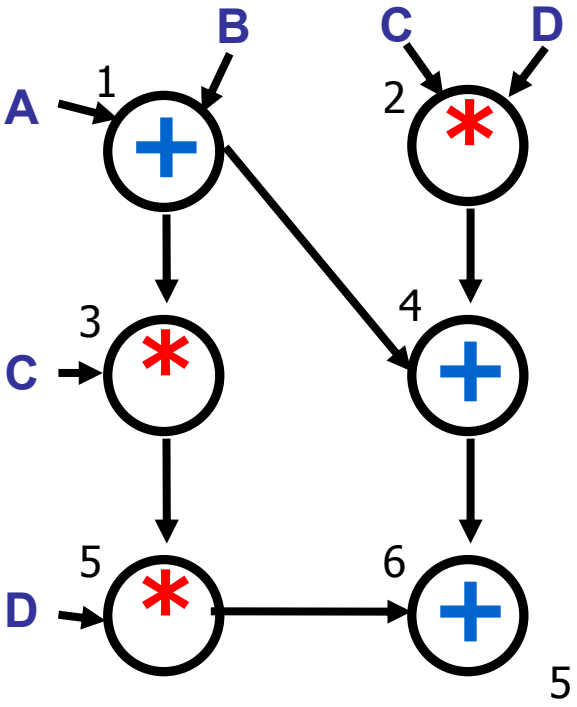
+	5s
*	7s
Idle	2W
In use	3W

ENERGY:
10
20



Energy:
139 Joule !!

Task Graph Scheduling – Revisited



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

using 2 processors

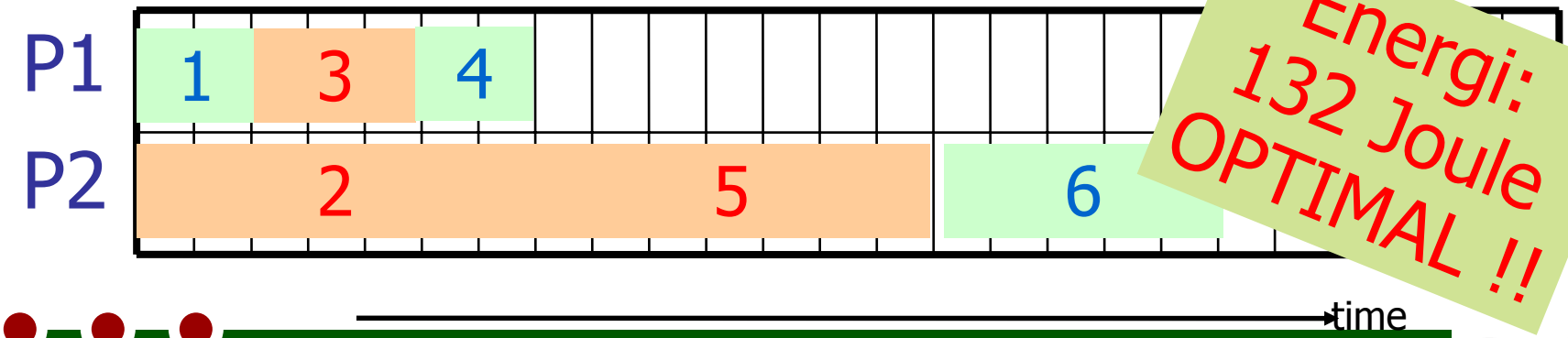
P1 (fast)

+	2s
*	3s
Idle	1W
In use	9W

P2 (slow)

+	5s
*	7s
Idle	2W
In use	3W

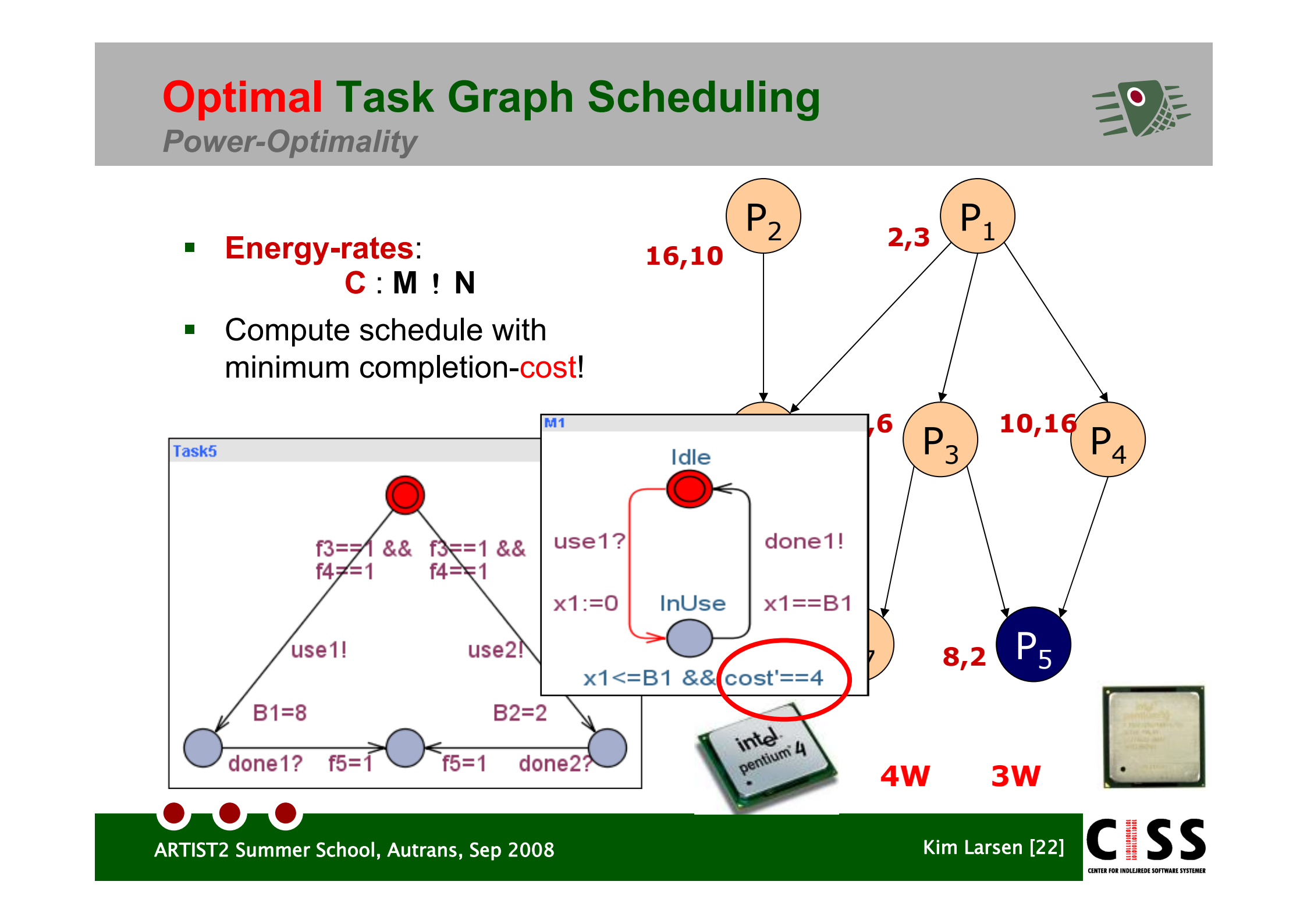
ENERGY:
10



Energi:
132 Joule
OPTIMAL !!

Optimal Task Graph Scheduling

Power-Optimality



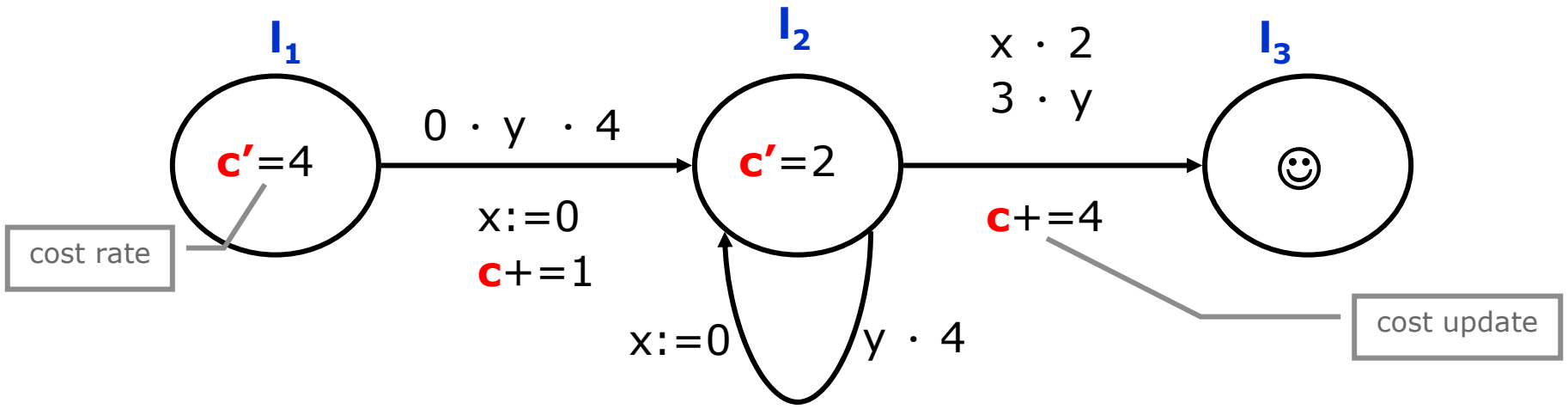
- **Energy-rates**:
  C : M ! N
- Compute schedule with minimum completion-cost!

Priced Timed Automata

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)

Timed Automata + **COST** variable

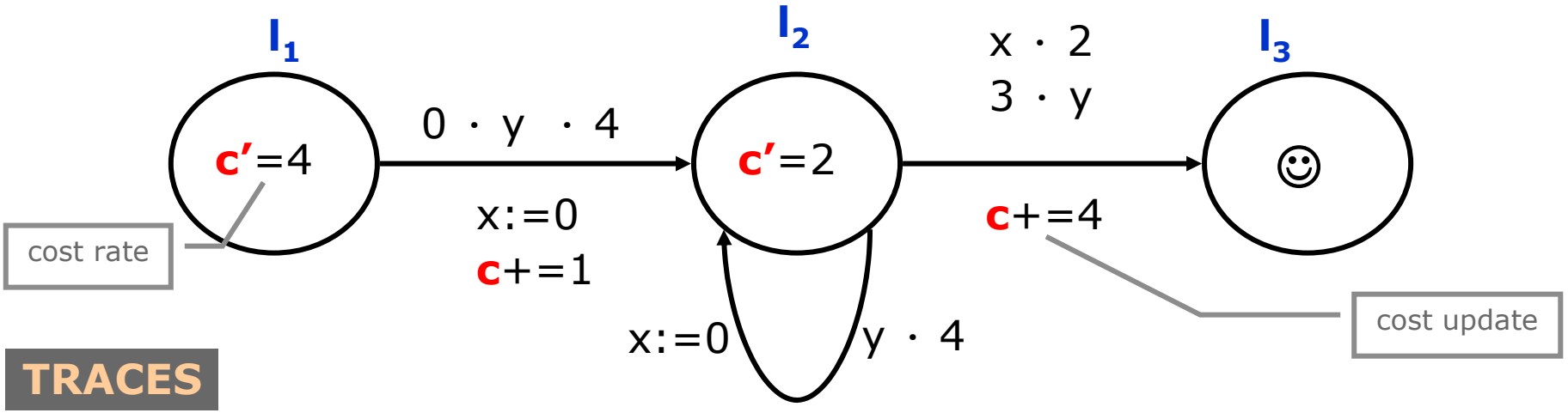


Priced Timed Automata

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)

Timed Automata + **COST** variable



TRACES

$$(l_1, x=y=0) \xrightarrow[12]{\varepsilon(3)} (l_1, x=y=3) \xrightarrow[1]{} (l_2, x=0, y=3) \xrightarrow[4]{} (l_3, \_, \_)$$

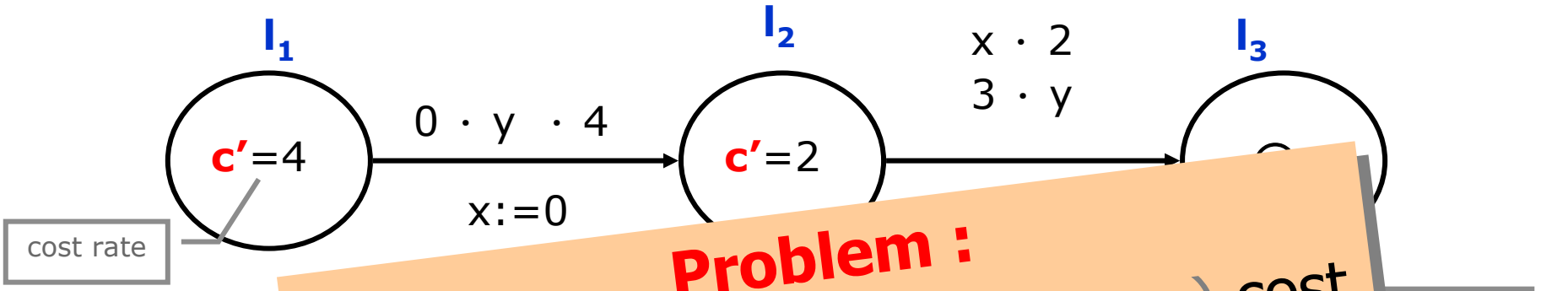
$\sum$ c=17

Priced Timed Automata

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)

Timed Automata + **COST** variable



Problem :

Find the minimum (maximum) cost of reaching location l_3

Efficient Implementation:
CAV'01 and TACAS'04

TRACES

$(l_1, x=y=0)$

$(l_1, x=y=0) \xrightarrow{\varepsilon(}$

$(l_1, x=y=0) \xrightarrow{1} (l_2, x=0, y=0$

$\xrightarrow{4}$

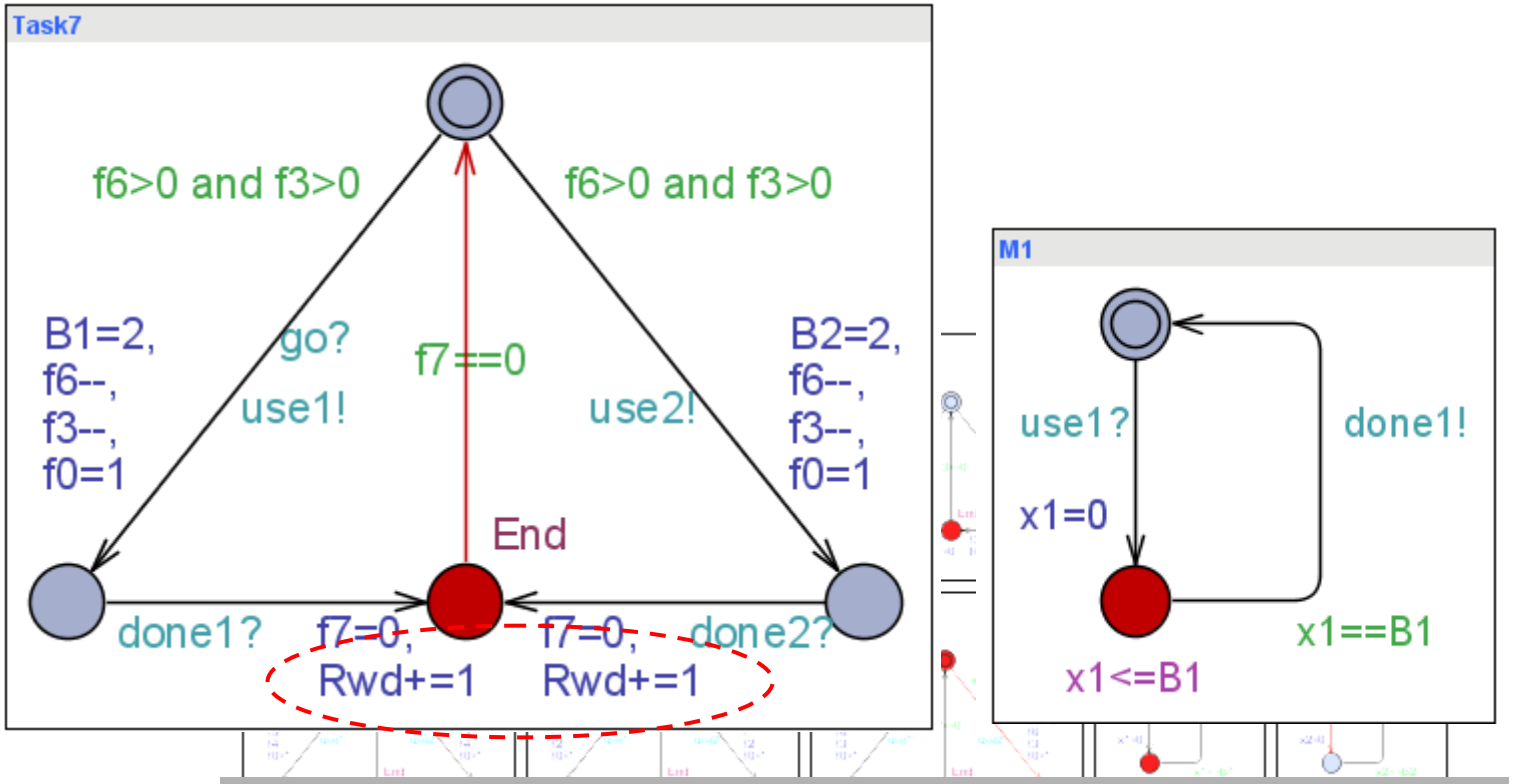
$(l_3, \_, \_)$ **Σ c=17**

$(l_3, \_, \_)$ **Σ c=16**

$(l_3, \_, \_)$ **Σ c=11**

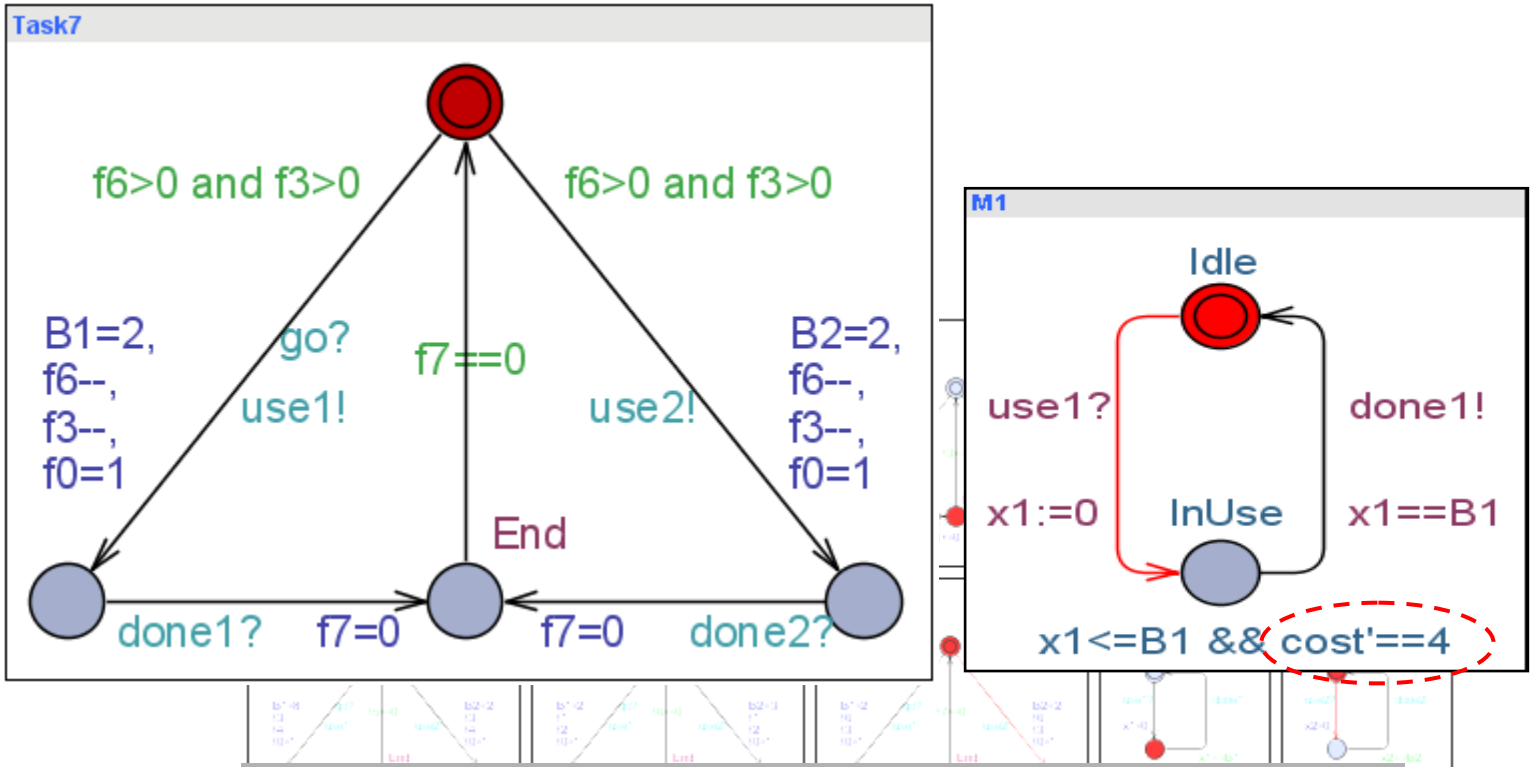
Competitive with MILP and commercial tool (Axxon)

Optimal Infinite Scheduling



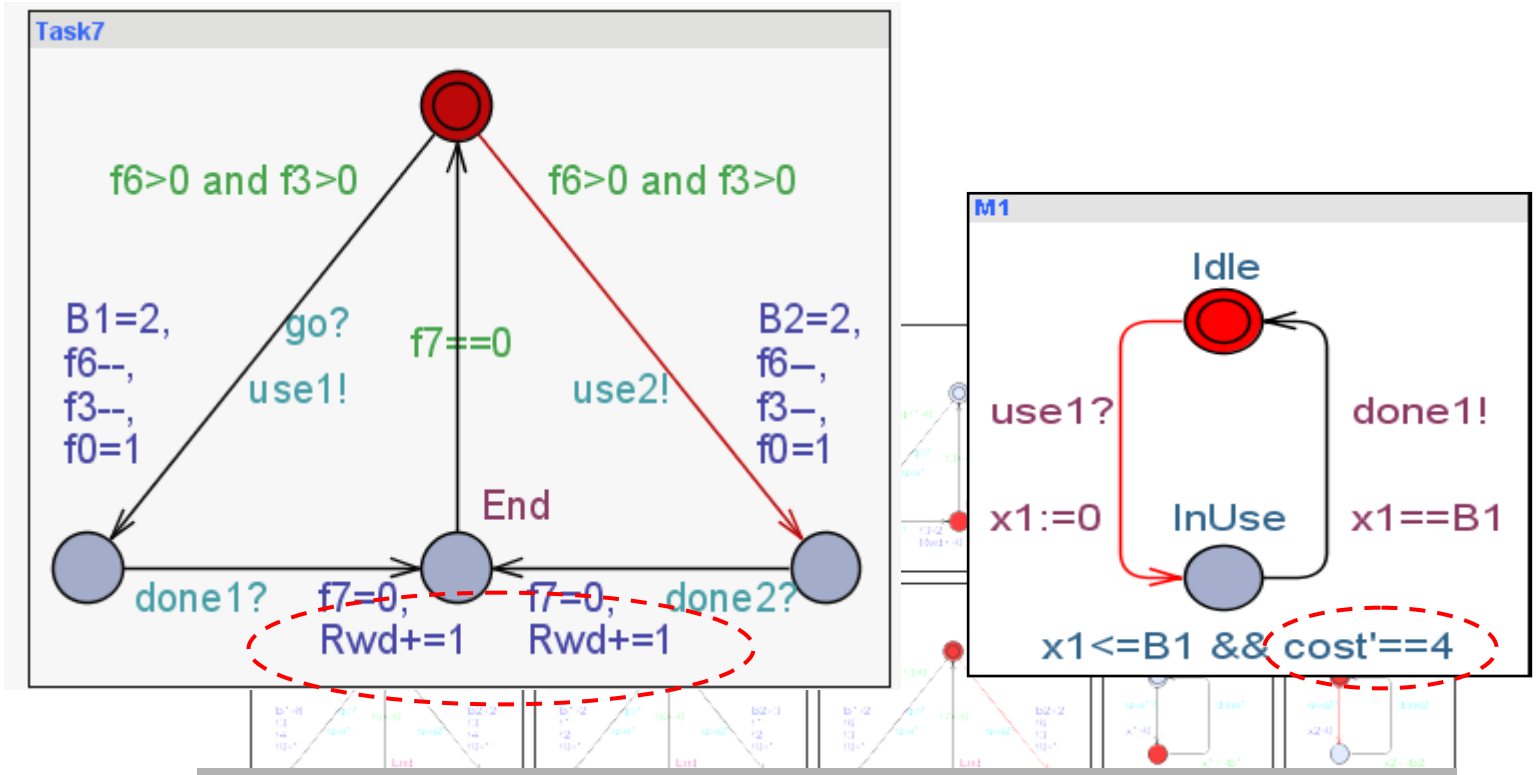
Maximize throughput:
i.e. maximize Reward / Time in the long run!

Optimal Infinite Scheduling



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

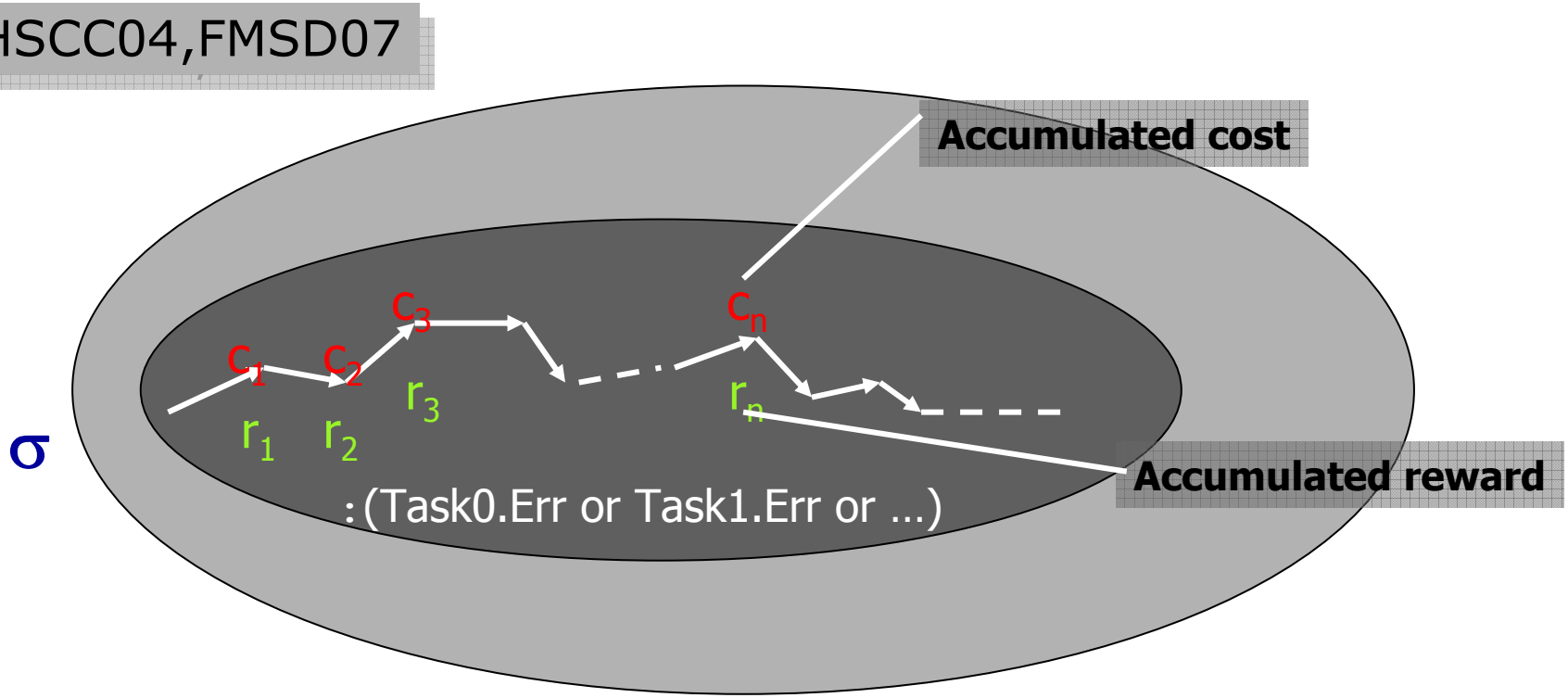
Optimal Infinite Scheduling



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

Cost Optimal Scheduling =

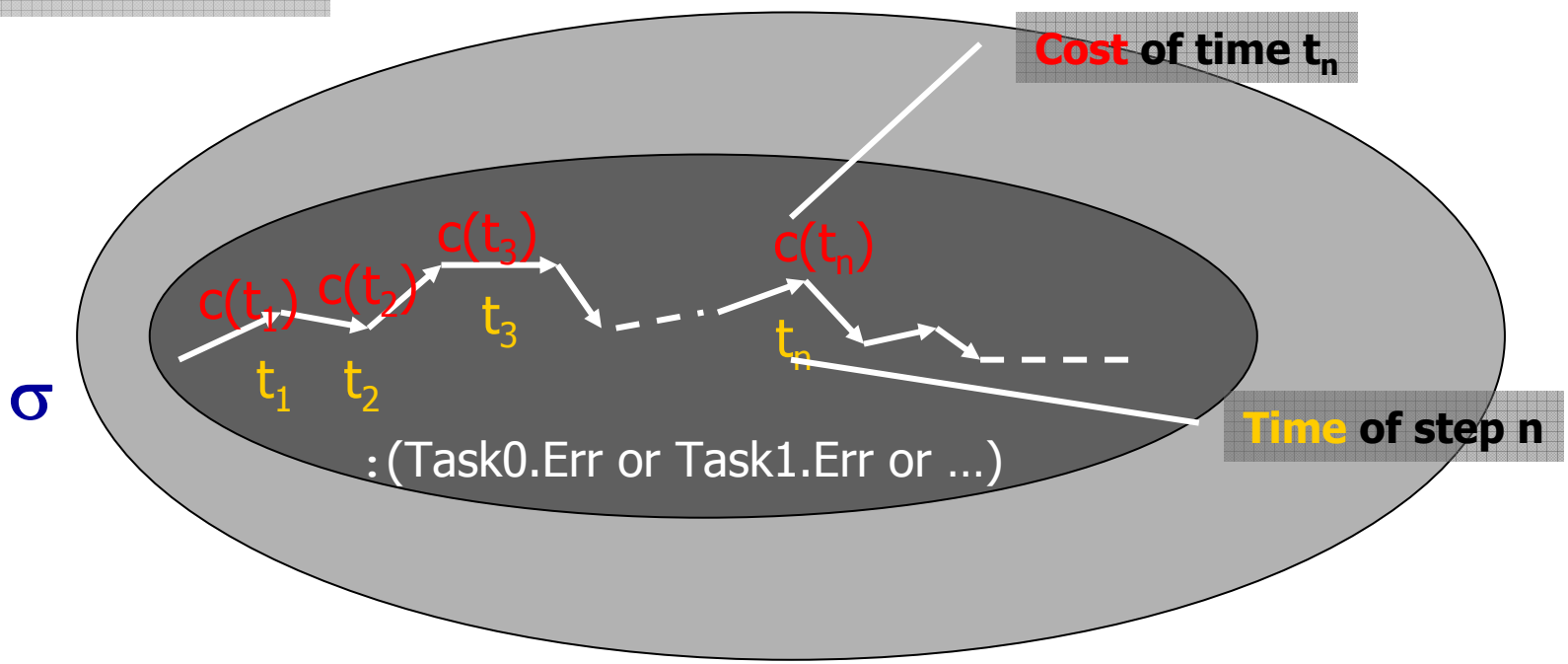
Mean Pay-off



Value of path σ: $val(\sigma) = \lim_{n \to \infty} c_n/r_n$

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

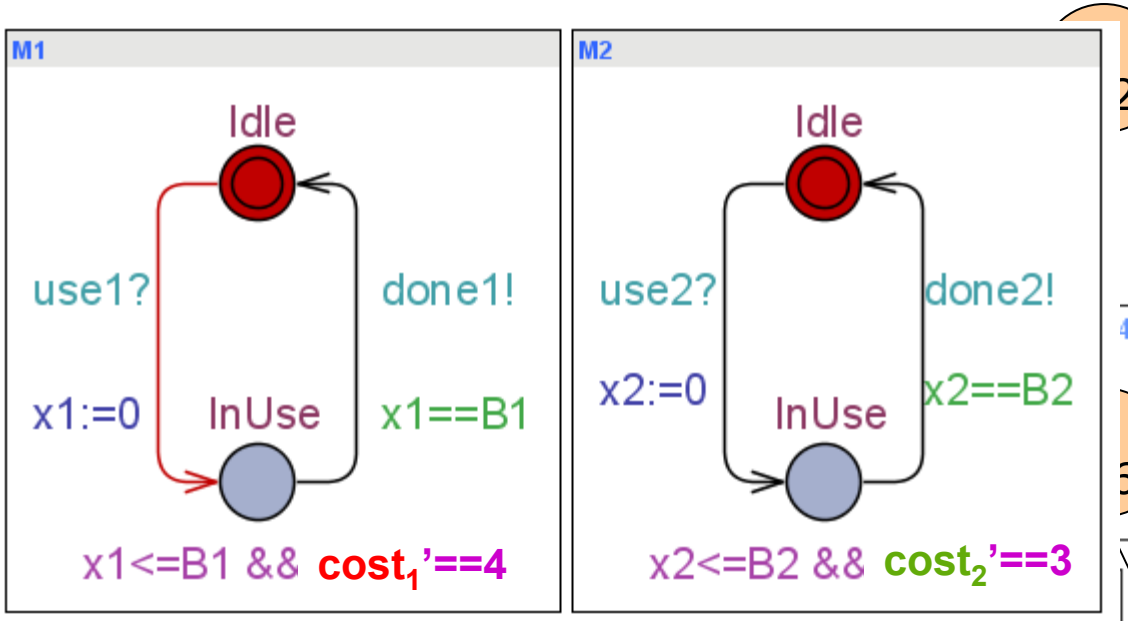
Discount Optimality $\lambda < 1$: discounting factor

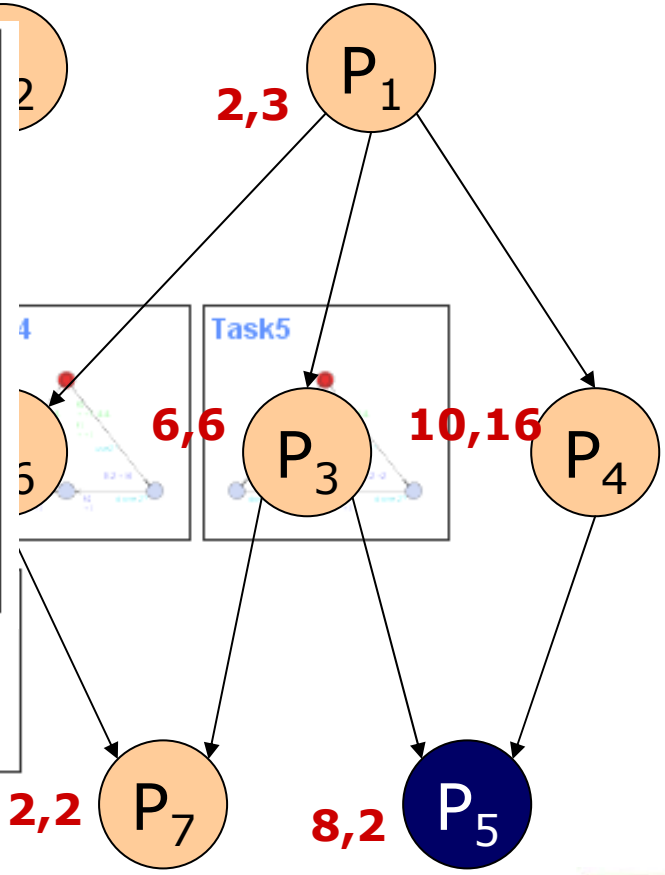


Value of path σ: $\text{val}(\sigma) = \int_{t=0}^{t=\infty} c(t)\lambda^t dt$

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

Multiple Objective Scheduling

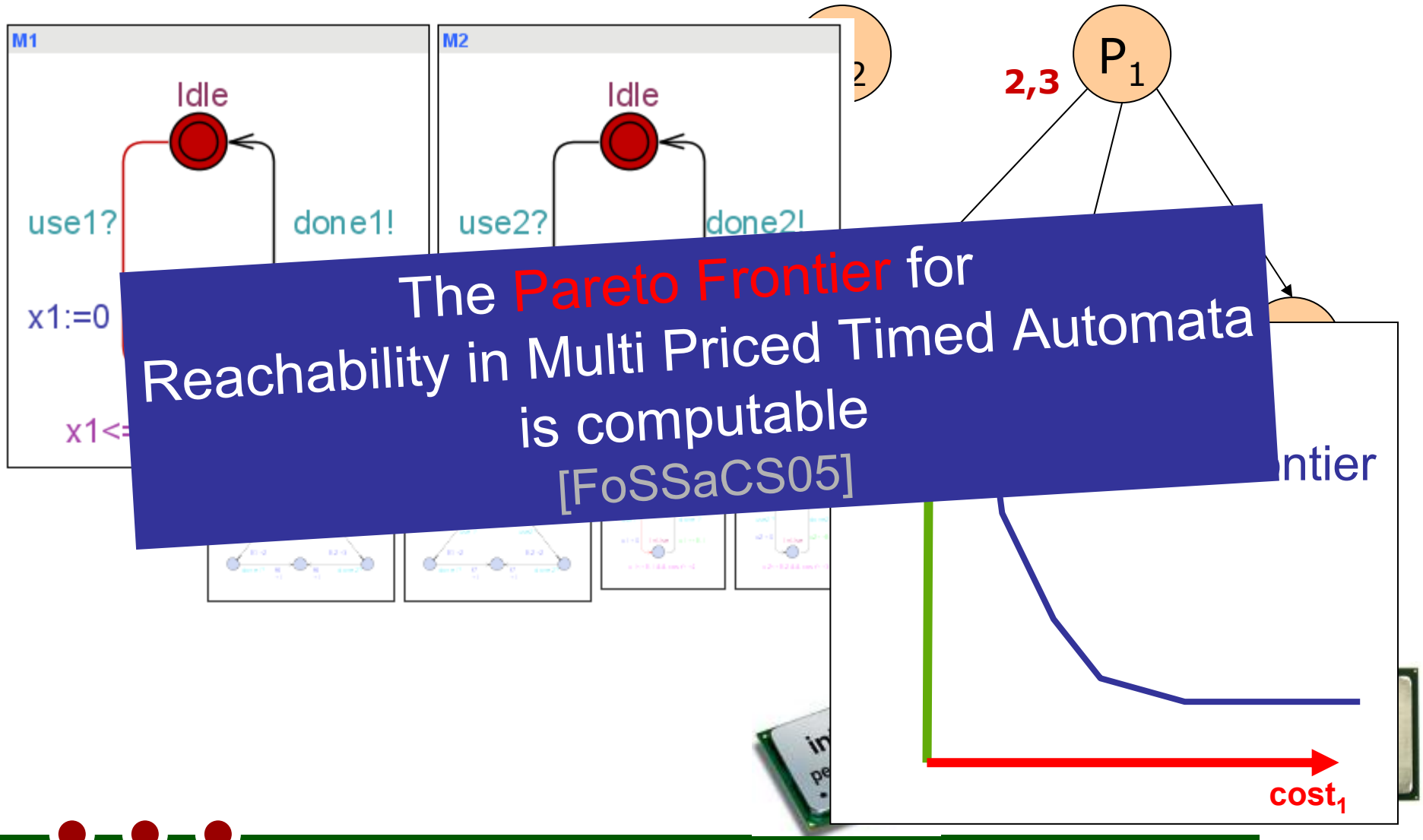




4W

3W

Multiple Objective Scheduling



The Pareto Frontier for
Reachability in Multi Priced Timed Automata
is computable
[FoSSaCS05]

Schedulability Analysis

ANALOG DEVICES

DaNES

AALBORG UNIVERSITY DENMARK

CISS

CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

CENTER FOR INDI

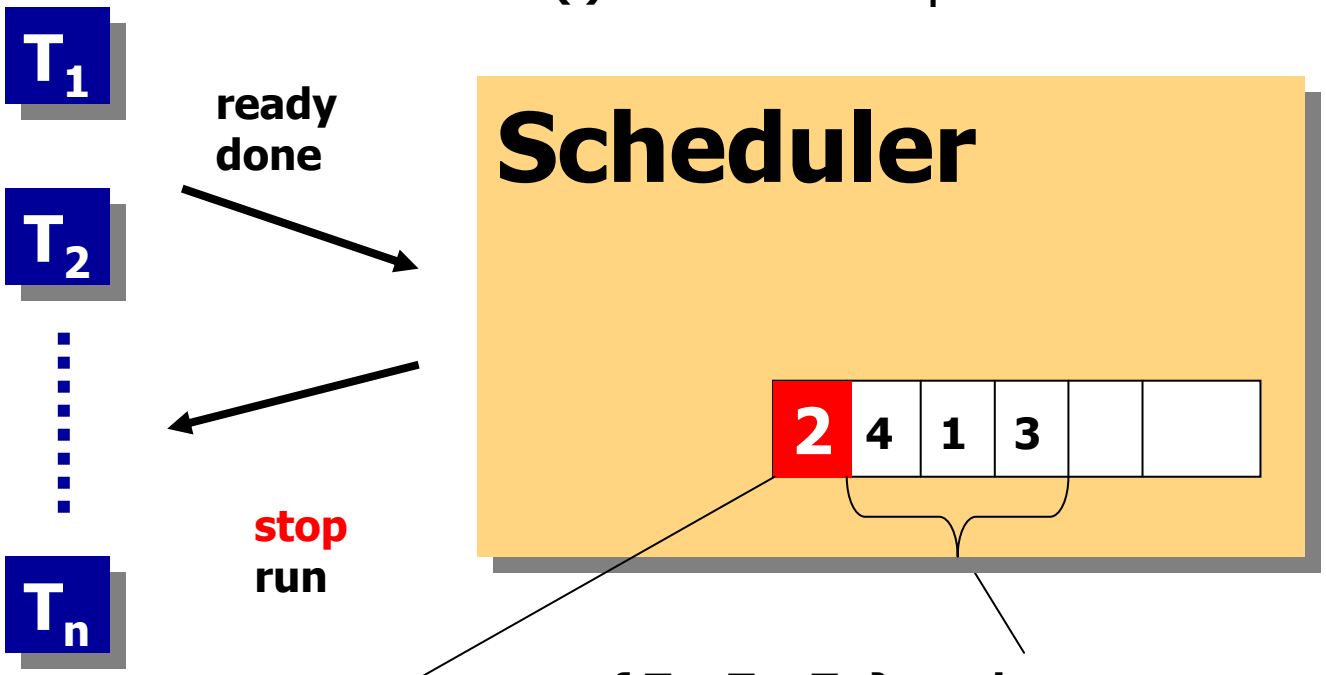
Task Scheduling

utilization of CPU

P(i), [E(i), L(i)], .. : period or earliest/latest arrival or .. for T_i

C(i): execution time for T_i

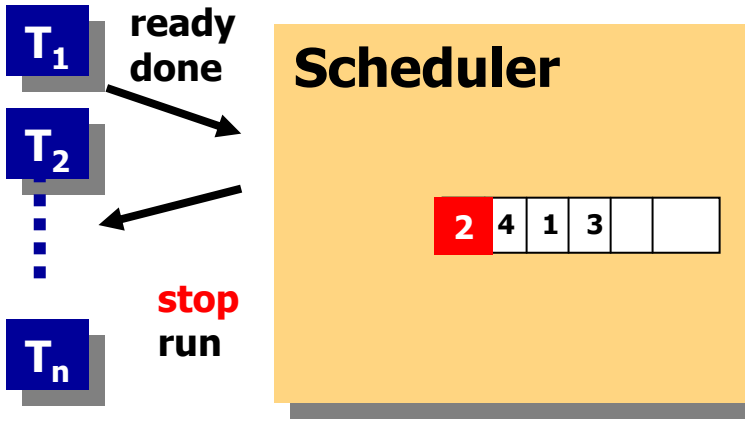
D(i): deadline for T_i

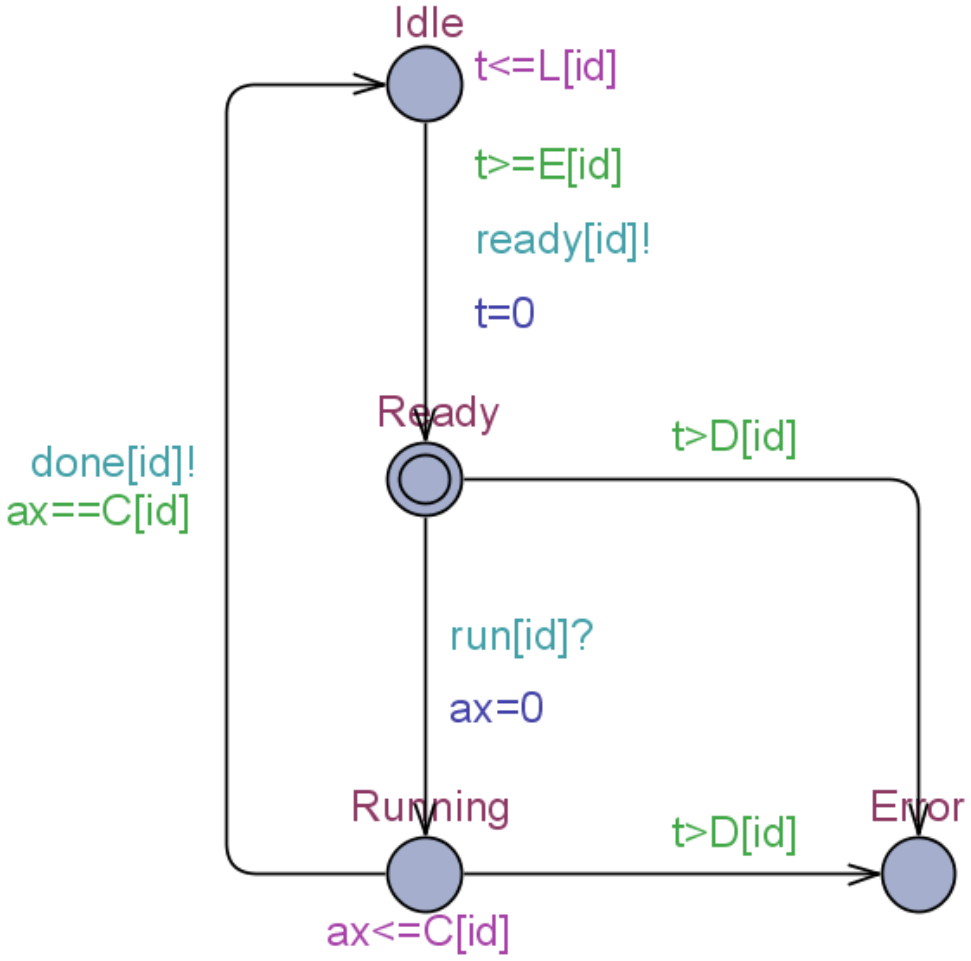


T_2 is running

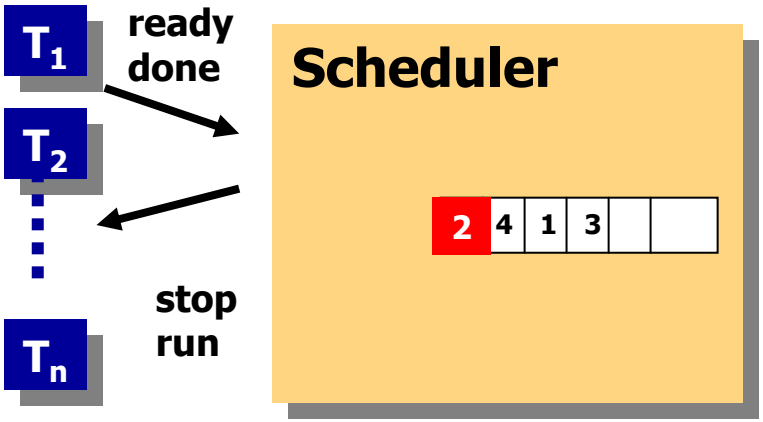
{ T_4 , T_1 , T_3 } ready ordered according to some given priority: (e.g. Fixed Priority, Earliest Deadline,..)

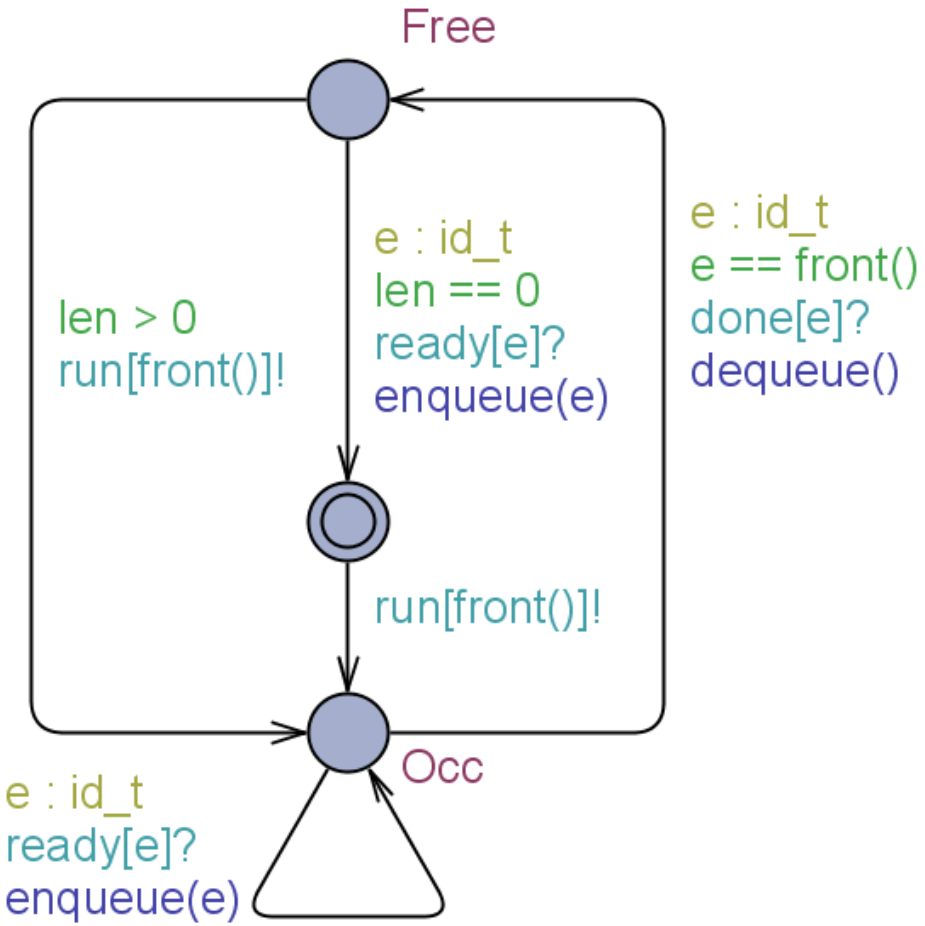
Modeling Task





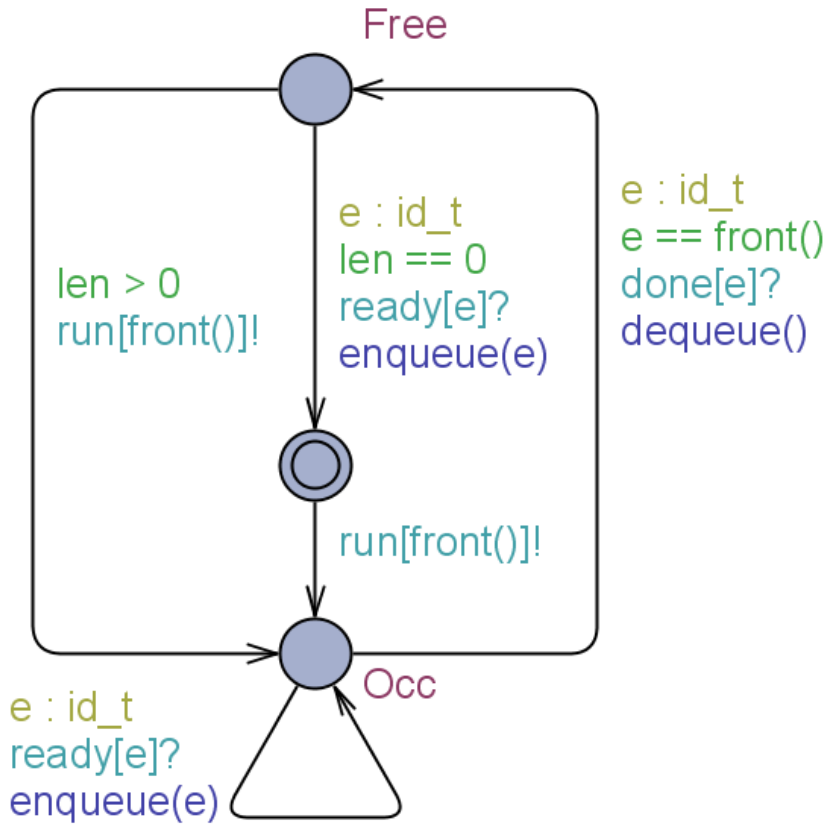
Modeling Scheduler





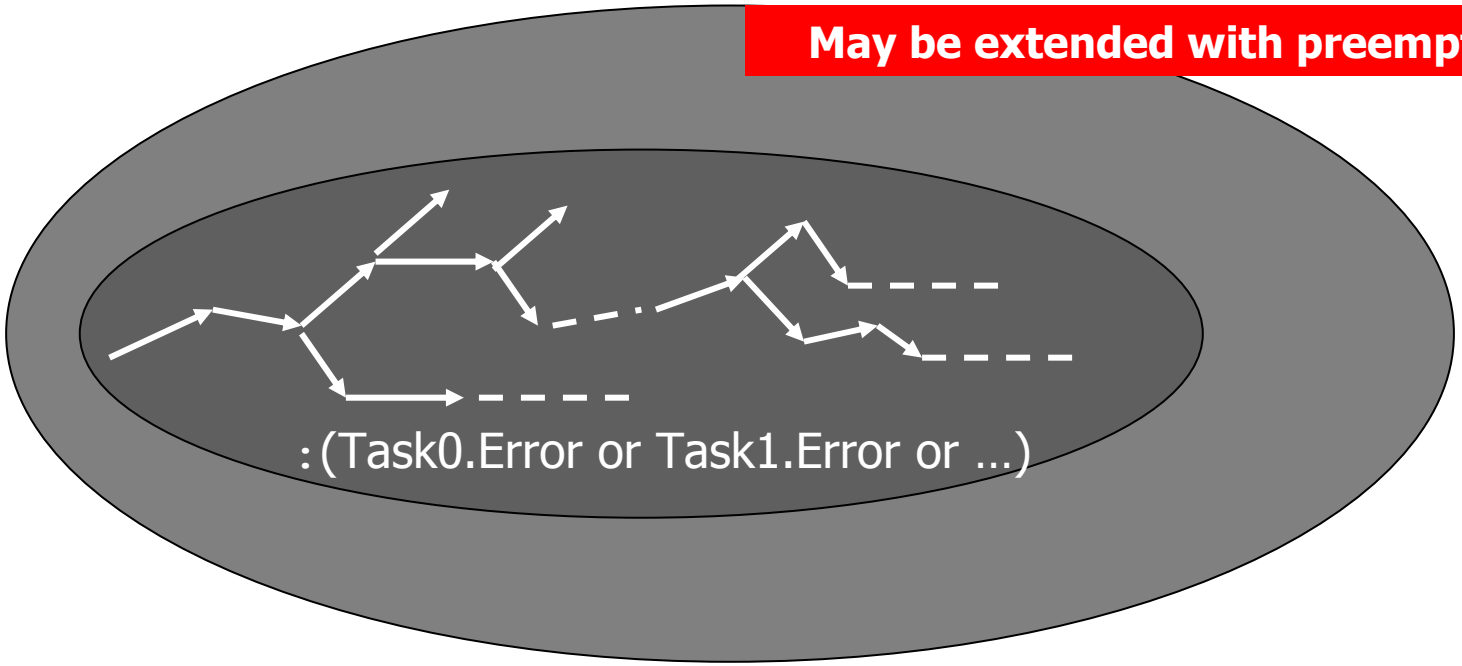
Modeling Queue

In UPPAAL 4.0
User Defined Function



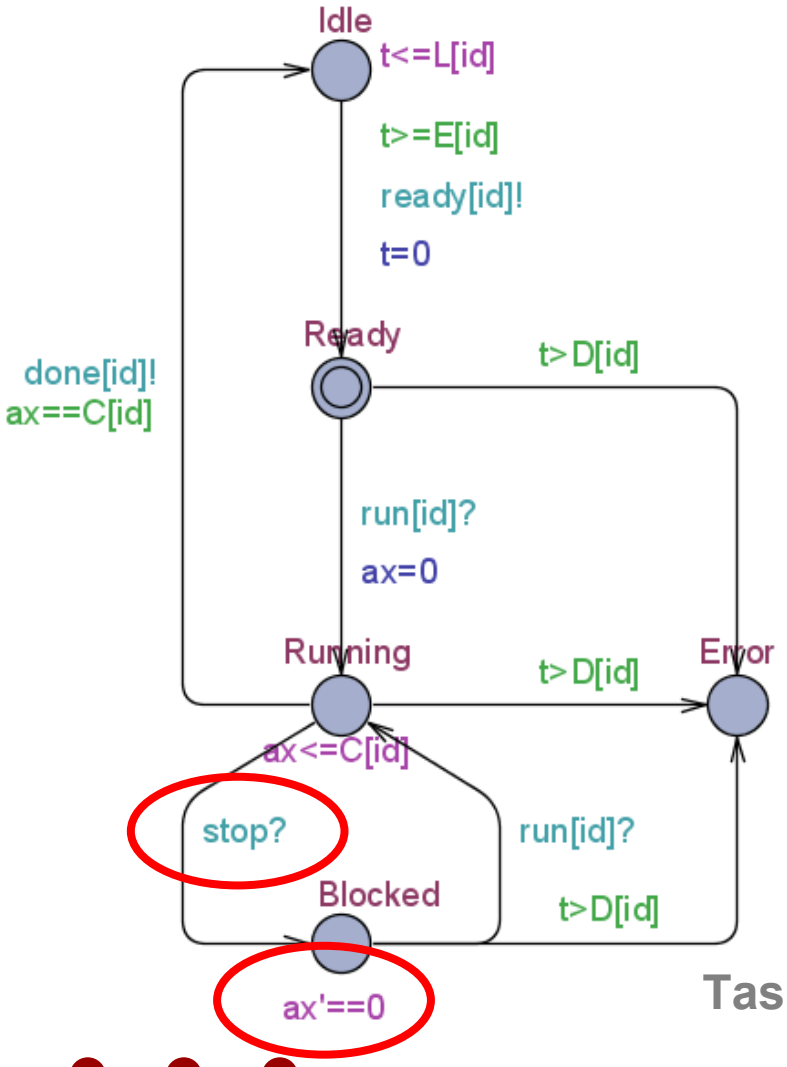

```
// Put an element at the end of the queue
void enqueue(id_t element)
{
int tmp=0;
list[len++] = element;
if (len>0)
{
        int i=len-1;
        while (i>1 && P[list[i]]>P[list[i-1]])
        {
                tmp = list[i-1];
                list[i-1] = list[i];
                list[i] = tmp;
                i--;
        }
}
}

// Remove the front element of the queue
void dequeue()
{
  ......
```

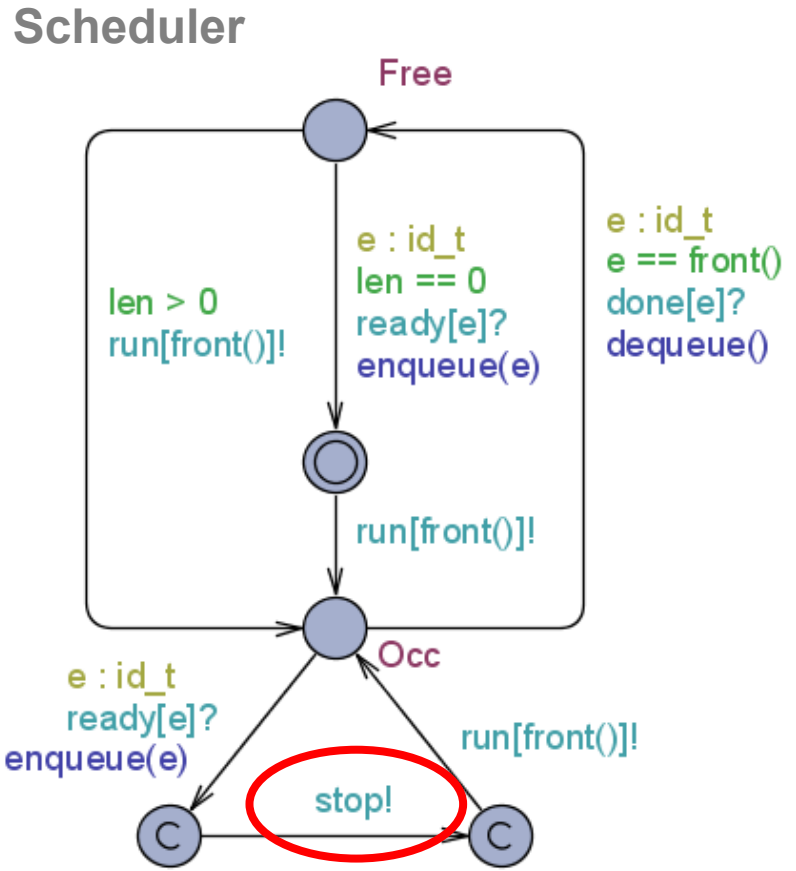
Schedulability = Safety Property



A□ :(Task0.Error or Task1.Error or ...)

Preemption – Stopwatches!



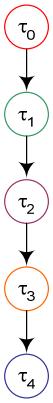


Defeating undecidability ☺

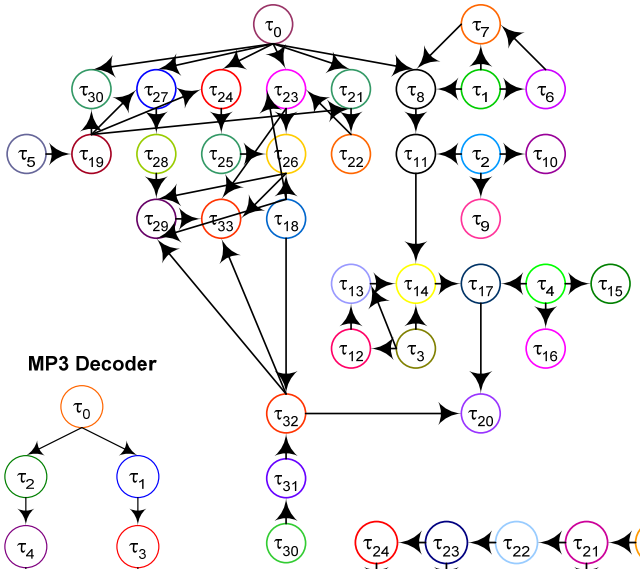
Handling realistic applications?

Jan Madsen / DTU

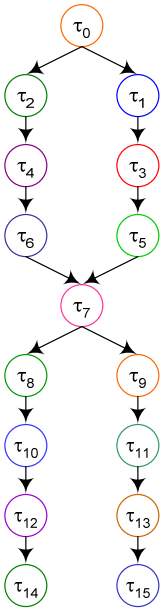
JPEG Encoder



GSM Decoder

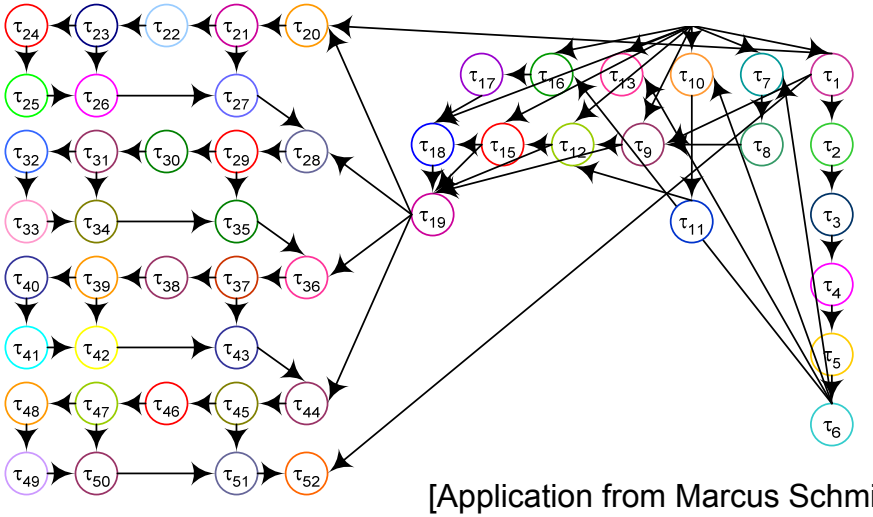


MP3 Decoder



JPEG Decoder

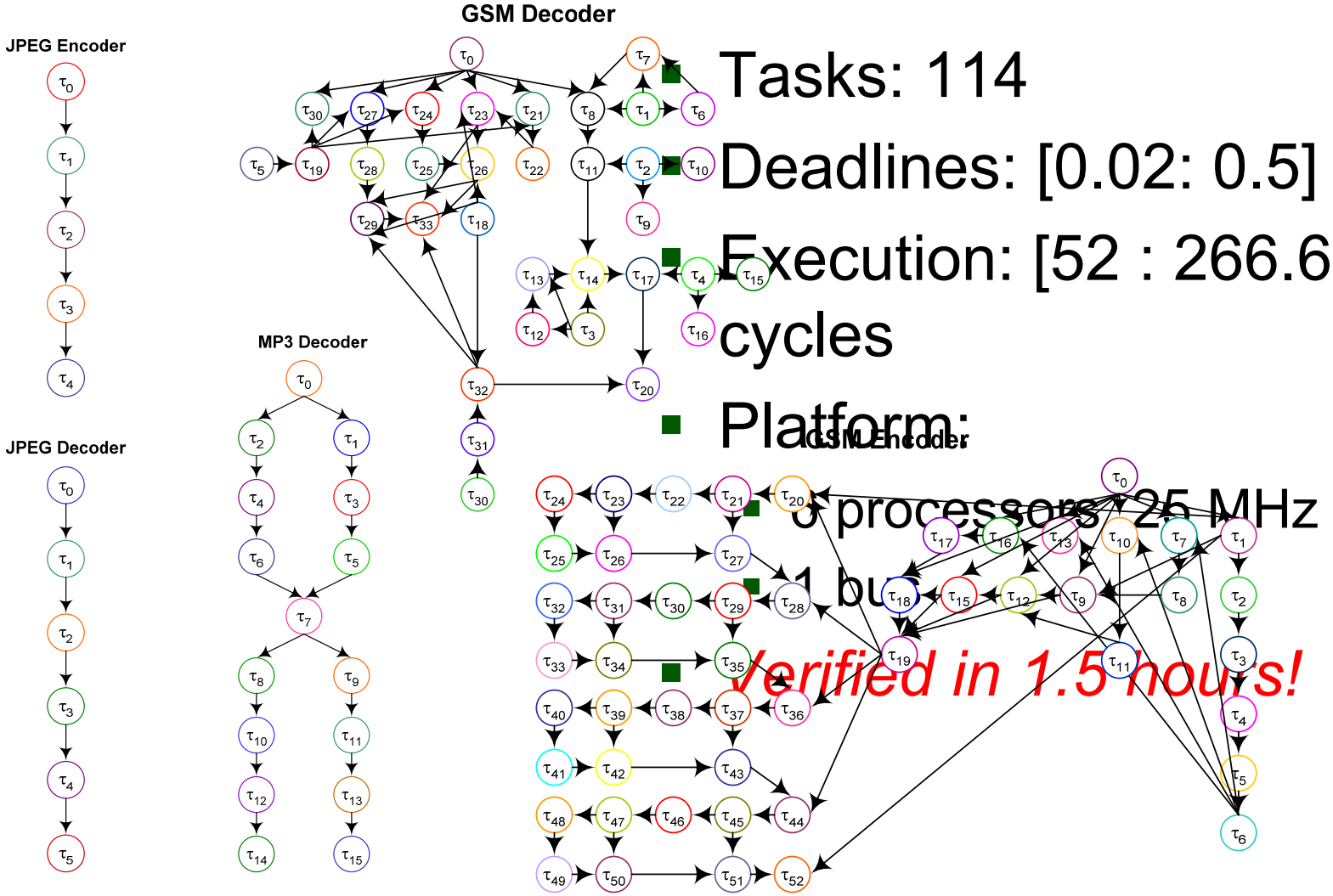
GSM Encoder



Smart phone:

[Application from Marcus Schmitz, TU Linkoping]

Smart phone



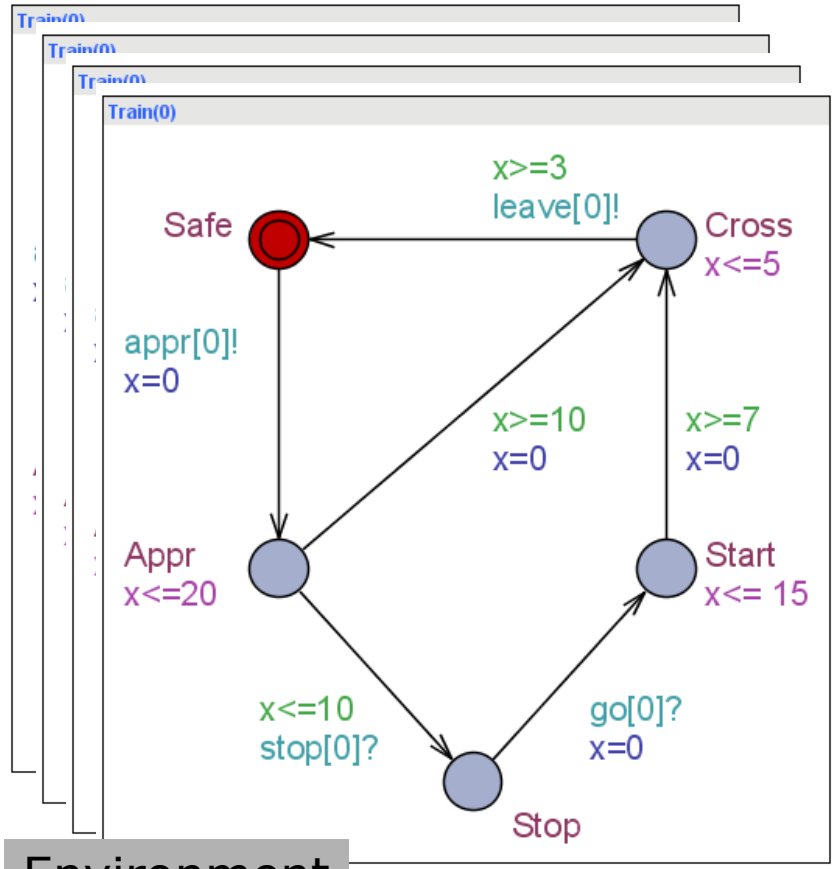
- Tasks: 114
- Deadlines: [0.02: 0.5] sec
- Execution: [52 : 266.687] cycles
- Platform:
  - 3 processors, 25 MHz
  - 1 bus

Verified in 1.5 hours!

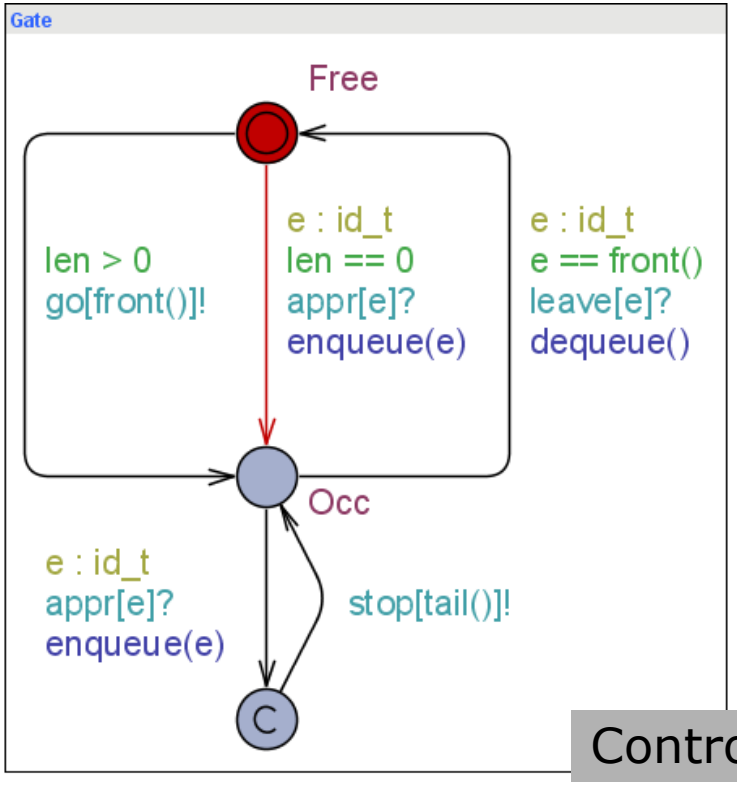
Synthesis

Timed Game Automata

Model Checking (ex Train Gate)



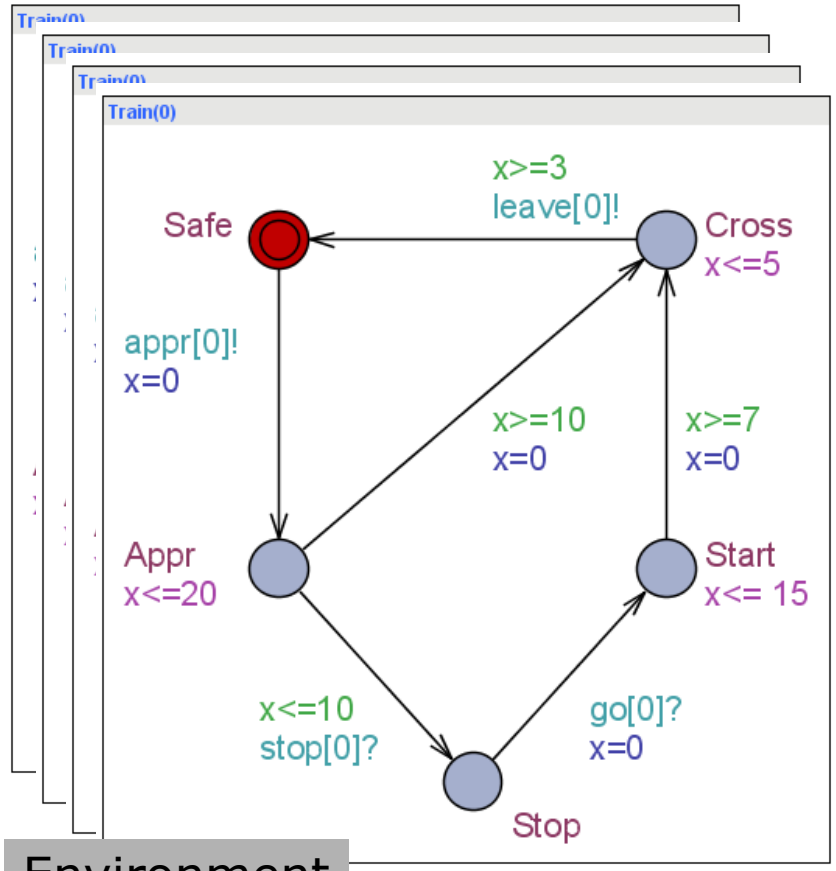
Environment



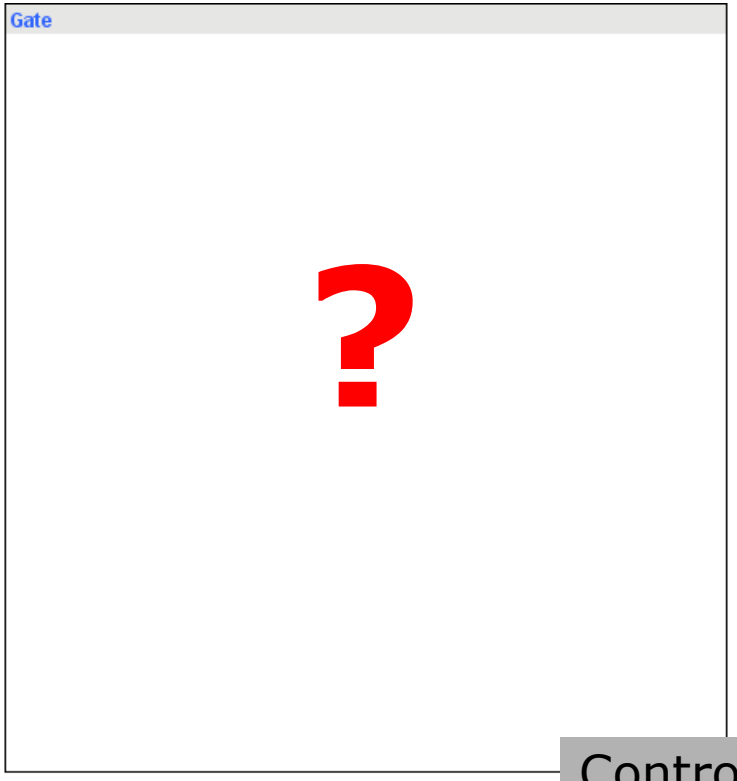
Controller

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

Synthesis (ex Train Gate)



Environment

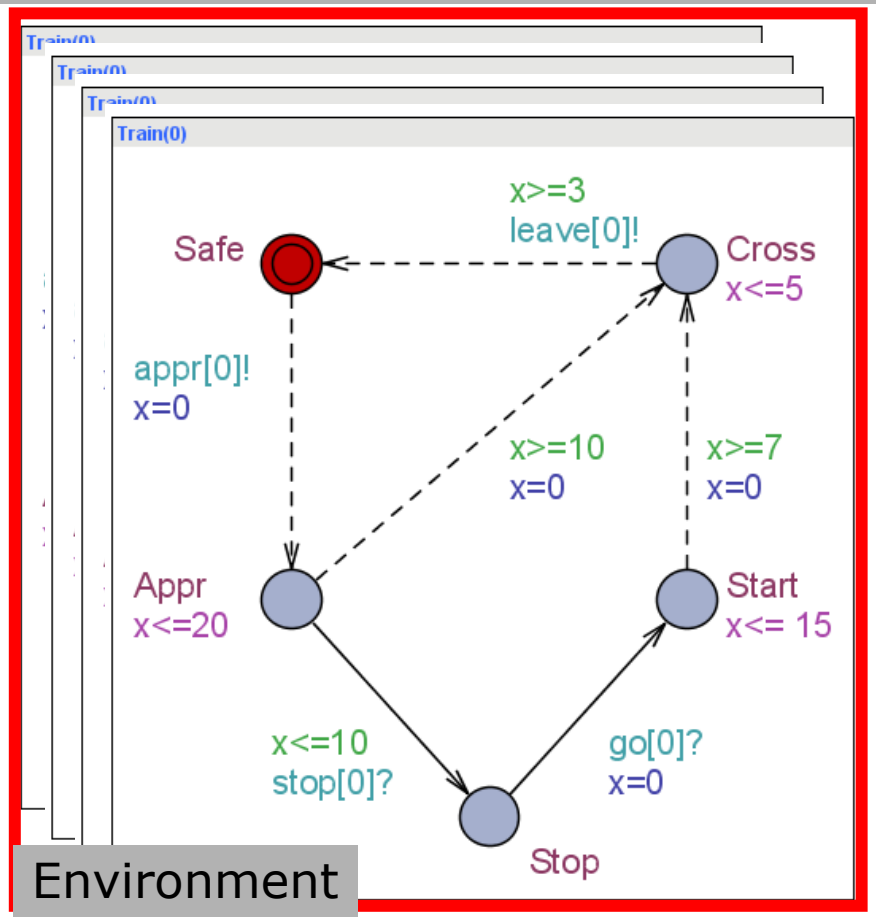


Controller

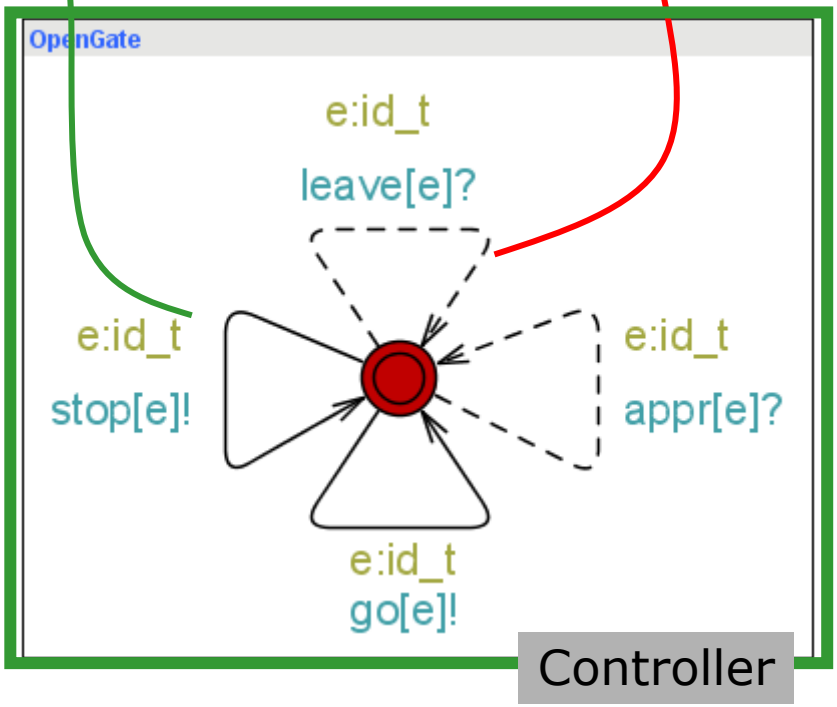
ϕ: Never two trains at the crossing at the same time

Synthesis

Two Player Game



Controllable Uncontrollable



Find strategy for controllable actions st behaviour satisfies ϕ

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

Timed Games

Reachability

Memoryless strategy:

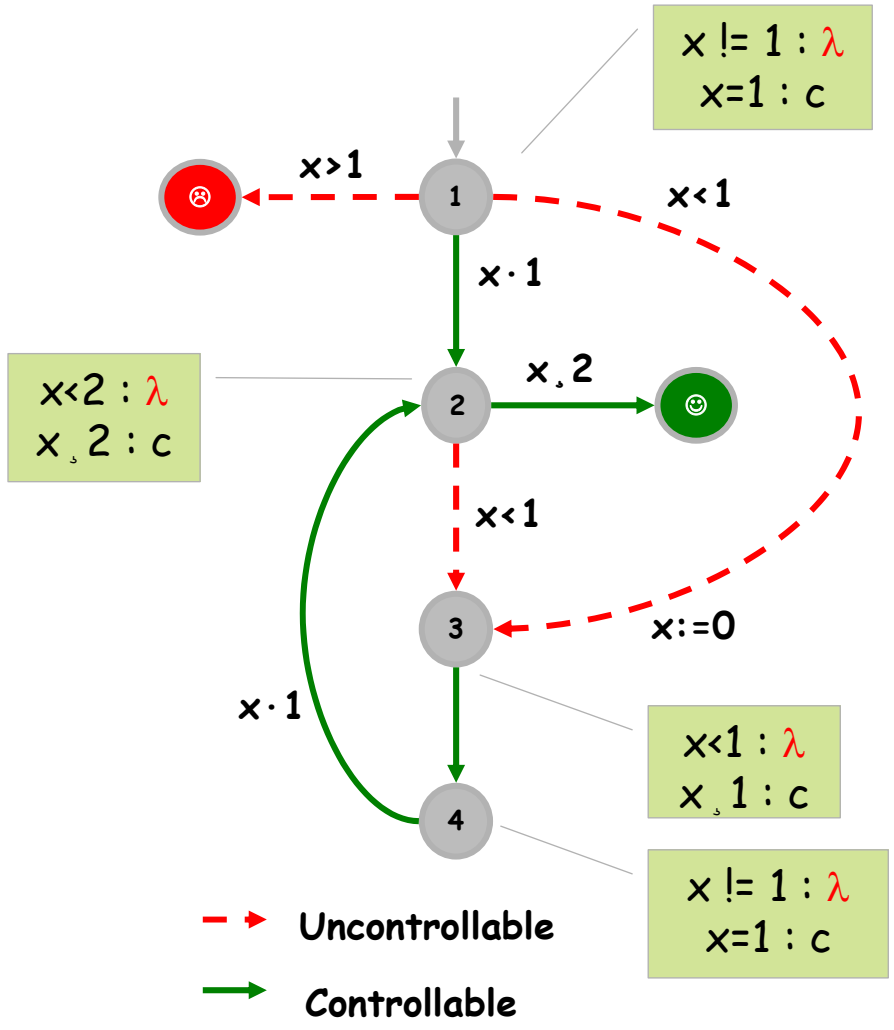
$F : Q \rightarrow E_c$ [λ

Winning Run:

States(ρ) Å $G \neq \emptyset$

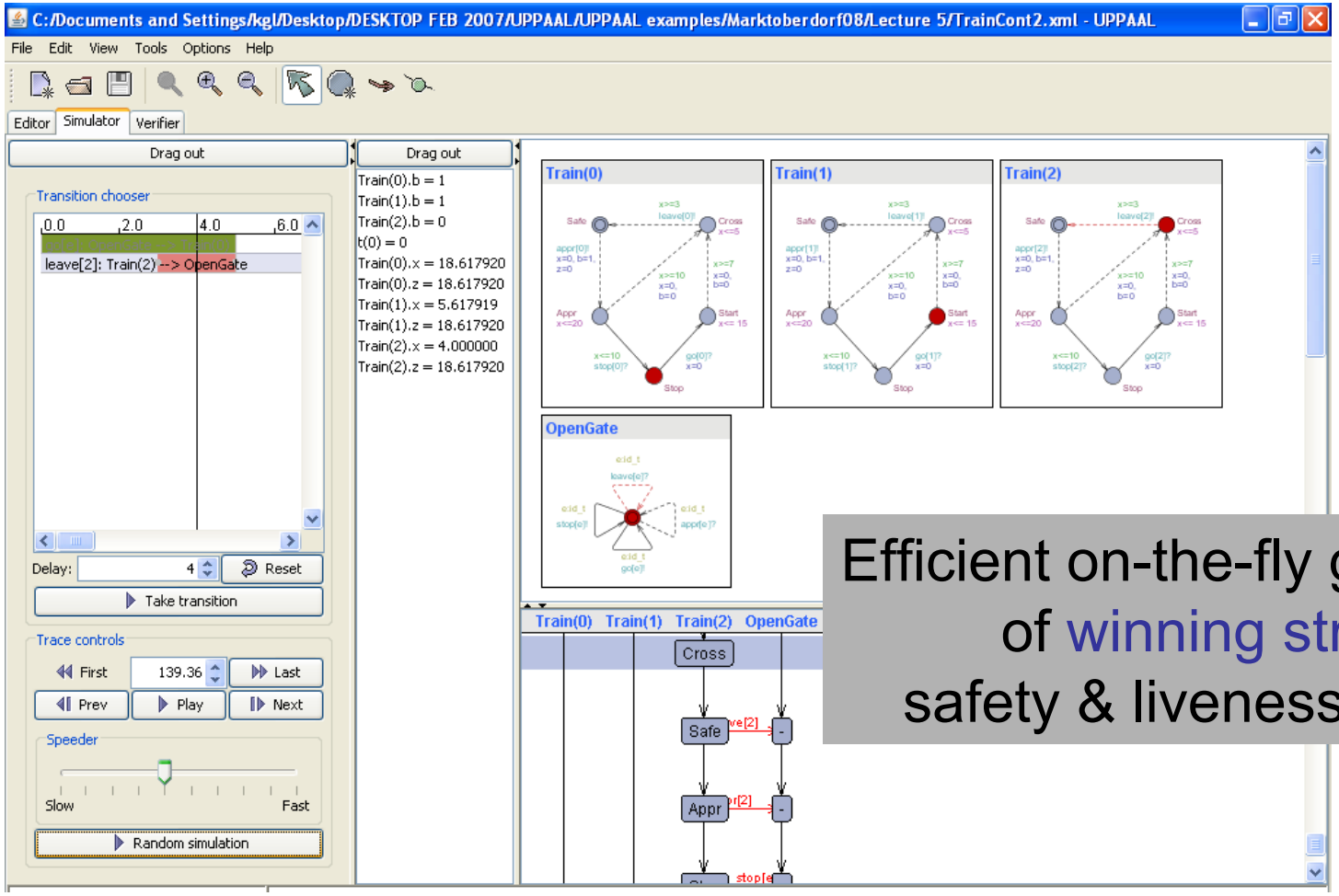
Winning Strategy:

Runs(F) μ WinRuns



UPPAAL Tiga

Synthesis of winning strategies for TIMED GAMES



CONCUR05,
CAV07,
FORMATS07

Efficient on-the-fly generation of winning strategies for safety & liveness objectives

Open Problems

Decidability

- Priced Timed Games
  - Reachability & Model Checking
  - Safety
  - Negative cost rates
- Probabilistic Priced Timed Automata
- Timed Automata
  - Fully Symbolic (CDD)
  - Static Analysis & Slicing of C-code
- Timed Games
  - Alternating Logics
  - Partial Observability
  - CEGAR

Efficiency

- Priced Timed Automata
  - Optimal Infinite Schedules for PTA (zone based)
  - Multi-Objective Optimal

Thanks for your attention!

www.uppaal.com

APPLICATIONS

Usability

- Live Sequence Charts
- Gantt Chart
- Probabilistic Timed Automata