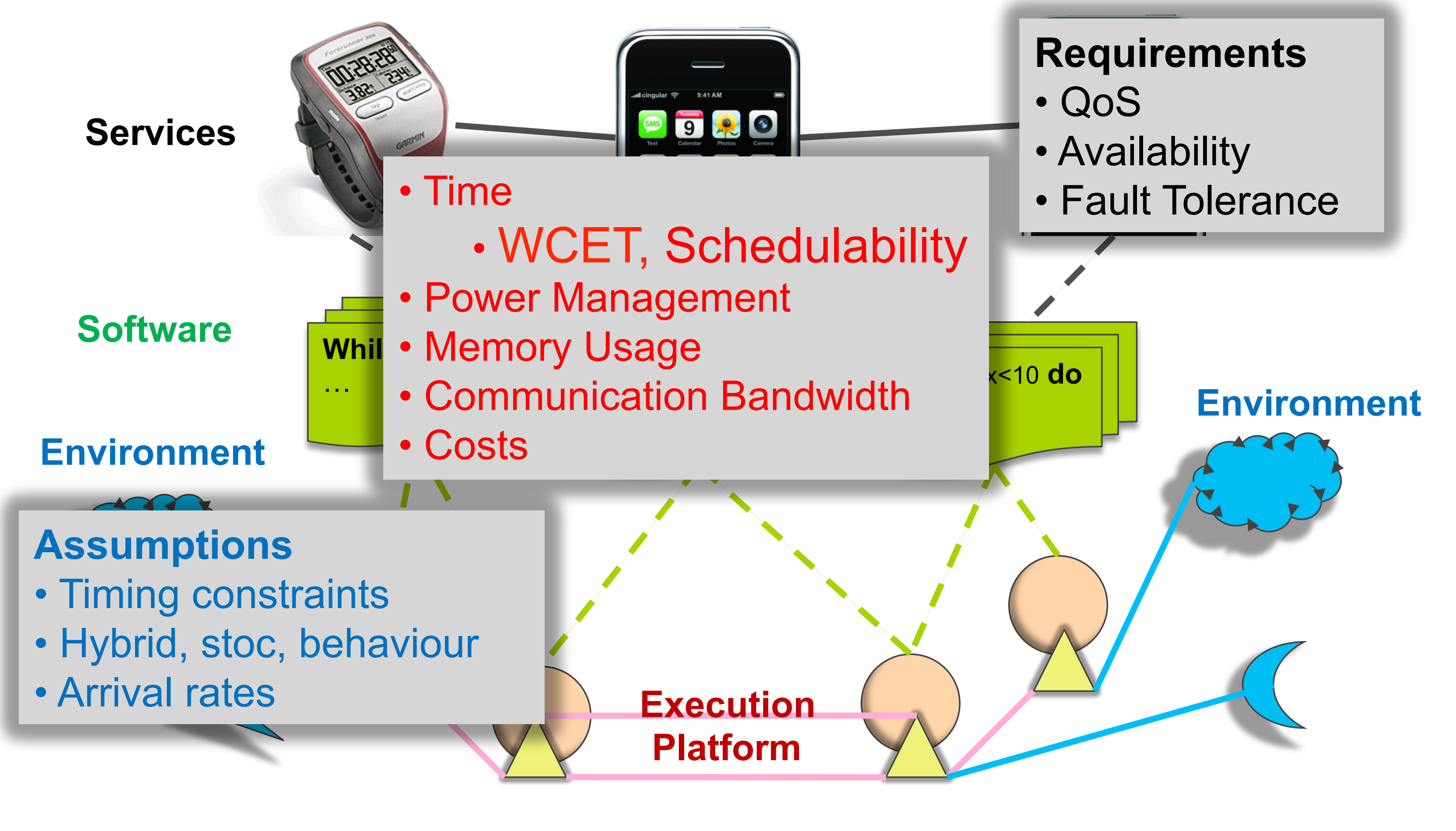
Software and Energy Aware Computation

John Gallagher, Roskilde U, DK
ENTRA
Kim G Larsen, Aalborg U, DK
SENSATION

Software & Resource Usage

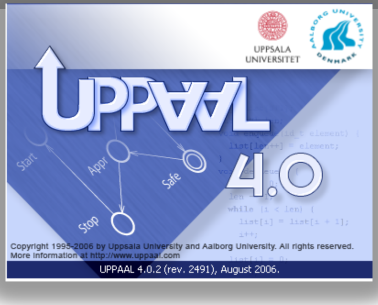


Requirements
- QoS
- Availability
- Fault Tolerance

- Time
  - WCET, Schedulability
- Power Management
- Memory Usage
- Communication Bandwidth
- Costs

Assumptions
- Timing constraints
- Hybrid, stoc, behaviour
- Arrival rates

Model-Based Analysis, Synthesis and Optimization of Energy-Aware Systems



Copyright 1995-2006 by Uppsala University and Aalborg University. All rights reserved.
More information at http://www.uppaal.com

UPPAAL 4.0.2 (rev. 2491), August 2006.

Kim G. Larsen
Aalborg University, DENMARK

SENSATION

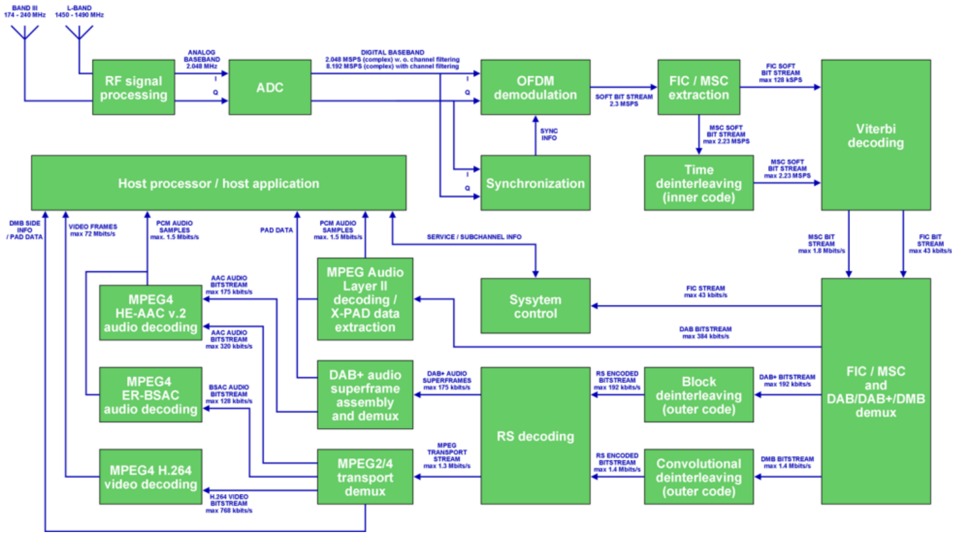
Self Energy-Supporting Autonomous Computation

FET Proactive: Minimising Energy Consumption of Computing to the Limit (MINECC)

SEVENTH FRAMEWORK PROGRAMME

Partners

- AAU: Aalborg University, Denmark
- RWTH: RWTH Aachen University, Germany
- ESI: Embedded Systems Institute, The Netherlands
- INRIA: Institut National de Recherche en Informatique et Automatiqe, France
- SAU: Saarbrücken University, Germany
- UT: University of Twente, The Netherlands
- GOM: GOMSpace, Denmark
- RS: Recore Systems, The Netherlands
- STM: STMicroelectronics, France


BAND III
174 - 240 MHz
L-BAND
1450 - 1490 MHz
RF signal processing
ANALOG BASEBAND 2.048 MHz
I
Q
ADC
DIGITAL BASEBAND
2.048 MSPS (complex) w. o. channel filtering
8.192 MSPS (complex) with channel filtering
OFDM demodulation
SOFT BIT STREAM 2.3 MSPS
FIC / MSC extraction
FIC SOFT BIT STREAM max 128 kSPS
Viterbi decoding
SYNC INFO
Synchronization
MSC SOFT BIT STREAM max 2.23 MSPS
Time deinterleaving (inner code)
MSC SOFT BIT STREAM max 2.23 MSPS
Host processor / host application
DMB SIDE INFO / PAD DATA
VIDEO FRAMES max 72 Mbits/s
PCM AUDIO SAMPLES max. 1.5 Mbits/s
PAD DATA
PCM AUDIO SAMPLES max. 1.5 Mbits/s
SERVICE / SUBCHANNEL INFO
MSC BIT STREAM max 1.8 Mbits/s
FIC BIT STREAM max 43 kbits/s
MPEG4 HE-AAC v.2 audio decoding
AAC AUDIO BITSTREAM max 175 kbits/s
AAC AUDIO BITSTREAM max 320 kbits/s
MPEG Audio Layer II decoding / X-PAD data extraction
Sysytem control
FIC STREAM max 43 kbits/s
DAB BITSTREAM max 384 kbits/s
MPEG4 ER-BSAC audio decoding
BSAC AUDIO BITSTREAM max 128 kbits/s
DAB+ audio superframe assembly and demux
DAB+ AUDIO SUPERFRAMES max 175 kbits/s
RS decoding
RS ENCODED BITSTREAM max 192 kbits/s
Block deinterleaving (outer code)
DAB+ BITSTREAM max 192 kbits/s
FIC / MSC and DAB/DAB+/DMB demux
MPEG4 H.264 video decoding
H.264 VIDEO BITSTREAM max 768 kbits/s
MPEG2/4 transport demux
MPEG TRANSPORT STREAM max 1.3 Mbits/s
RS ENCODED BITSTREAM max 1.4 Mbits/s
Convolutional deinterleaving (outer code)
DMB BITSTREAM max 1.4 Mbits/s

Main Objectives

1. To develop adequate **automata based modeling formalisms** to describe a wide range of energy-related systems, and tailored towards power-aware optimization.
2. To advance **quantitative model-checking** techniques and tools to allow for scalable model-based quantitative analysis of energy-aware models.

Main Objectives

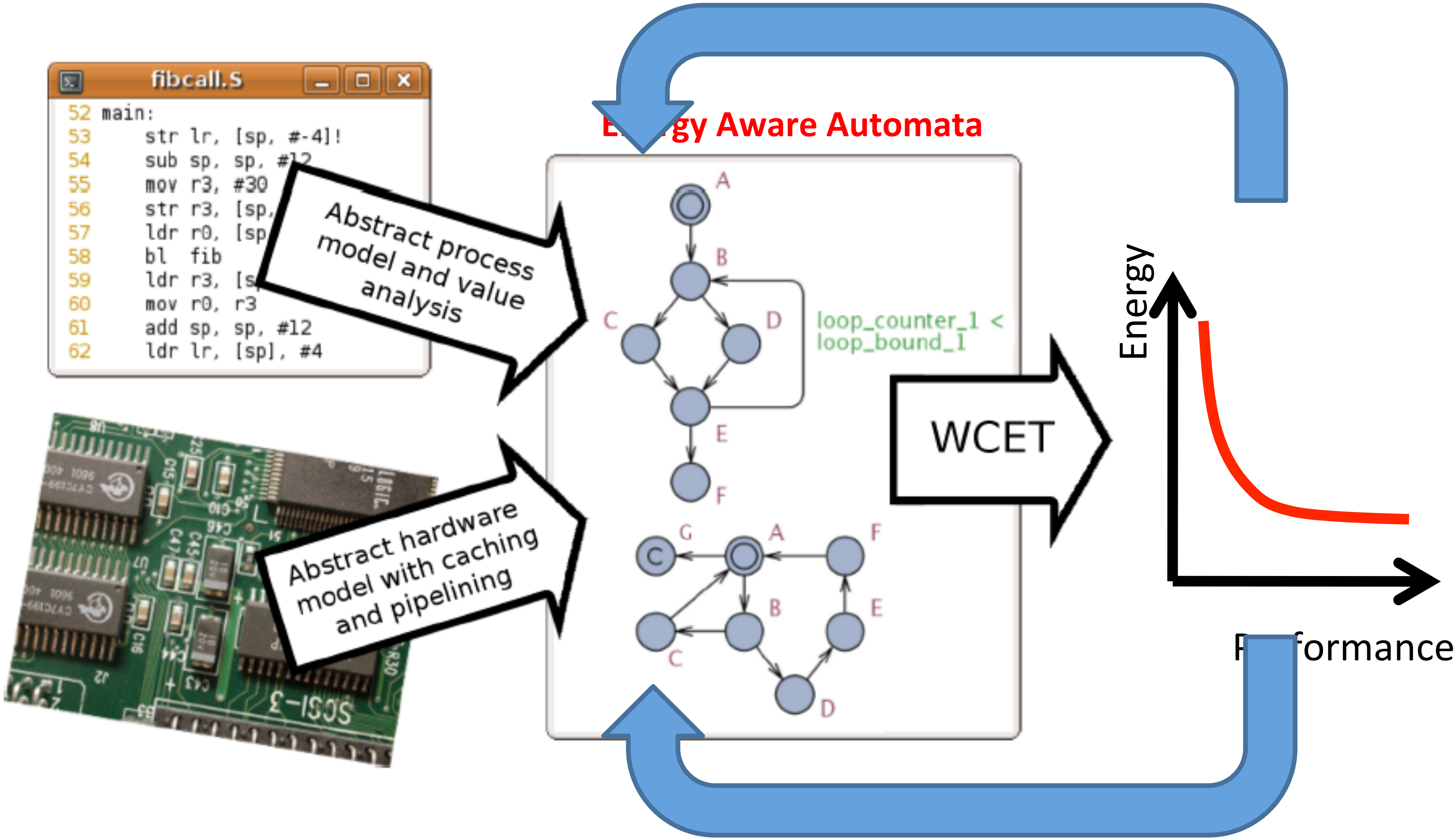
3. To provide algorithmic and tool support for **automatic synthesis of energy-optimal adaptive** and dynamic **energy management strategies**.
4. To provide a **design exploration method** allowing to analyse the effect of design choices in terms of a trade-off between energy, performance and reliability.
5. To **experimentally demonstrate** the radically increased scale of systems being energy-wise selfsupporting ranging based on cases arising from **space missions**, **streaming applications** and **software-defined radios**.

Vision

- Increase by orders of magnitude the scope of computing systems and applications which are self-supporting from an energy perspective.

- To arrive at this we will build upon **sound modeling and composition concepts** and innovative **quantitative verification technologies**, allowing to optimize energy-efficiency with a trade-off of other resources (timing, memory).

Resource Analysis & Optimization

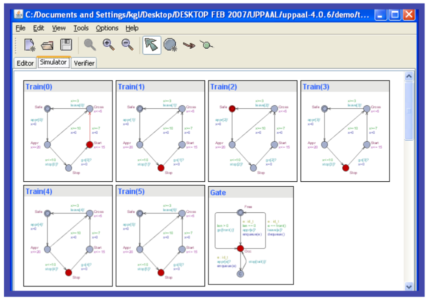


fibcall.S

```
main:
    str lr, [sp, #-4]!
    sub sp, sp, #12
    mov r3, #30
    str r3, [sp,
    ldr r0, [sp
    bl  fib
    ldr r3, [s
    mov r0, r3
    add sp, sp, #12
    ldr lr, [sp], #4
```

Abstract process model and value analysis
Abstract hardware model with caching and pipelining
Energy Aware Automata
A
B
C
D
E
F
loop_counter_1 < loop_bound_1
G
C
A
F
B
E
C
D
WCET
Energy
Performance

QUANTITATIVE Model Checking

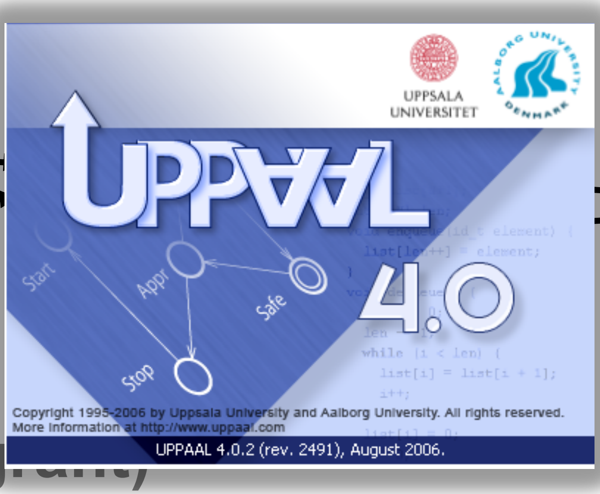

C:/Documents and Settings/kgl/Desktop/DESKTOP FEB 2007/UPPAAL/uppaal-4.0.6/demo/t...
File Edit View Tools Options Help
Editor Simulator Verifier
Train(0)
Train(1)
Train(2)
Train(3)
Train(4)
Train(5)
Gate

System Description

Time

Cost

Probability



Copyright 1995-2006 by Uppsala University and Aalborg University. All rights reserved.
More information at http://www.uppaal.com

UPPAAL 4.0.2 (rev. 2491), August 2006.

No!

Debugging Information

Yes

Prototypes

Executable Code

Test sequences

Requirement

A(req) A} grant)

A(req) A}$_{t<30s}$ grant)

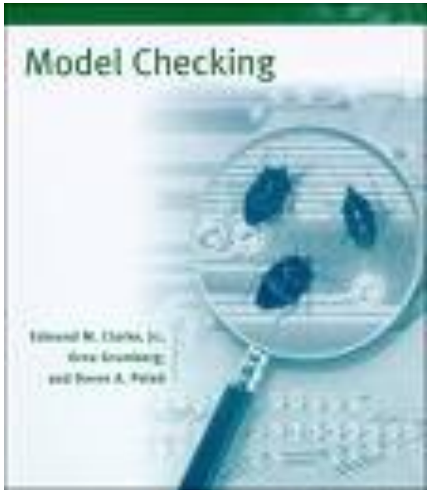
A(req) A}$_{t<30s,c<5J}$ grant)

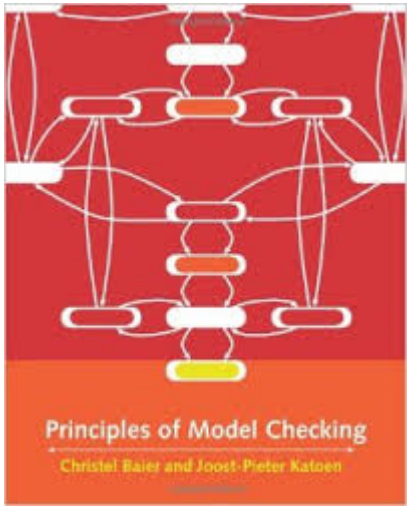
A(req) A}$_{t<30s\ ,\ p>0.90}$ grant)

Model Checking

E Allen Emerson

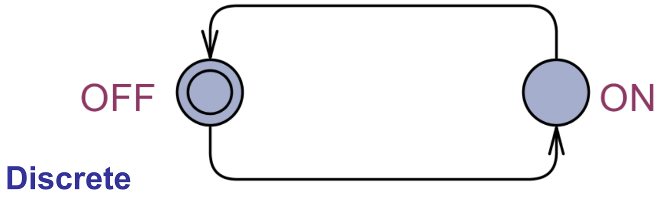
Ed Clarke

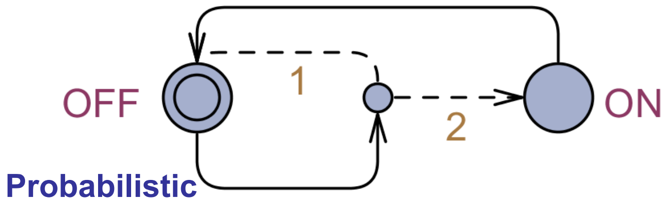


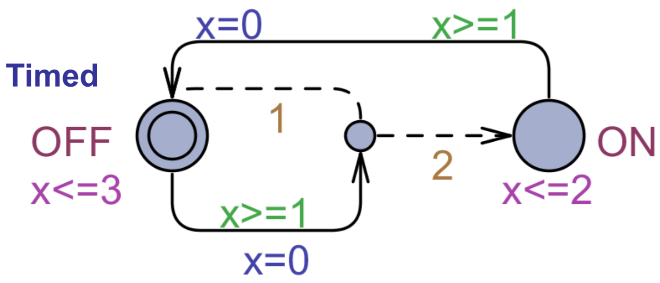

Principles of Model Checking
Christel Baier and Joost-Pieter Katoen

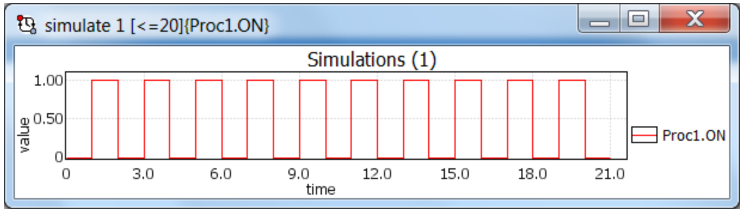
Joseph Sifakis

Models

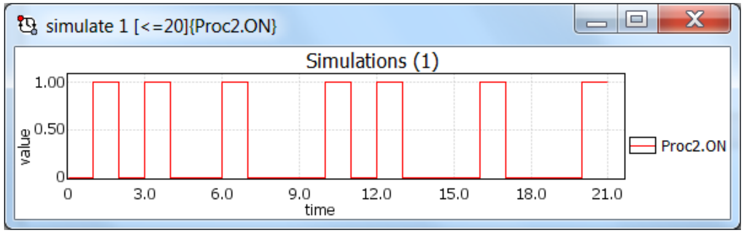




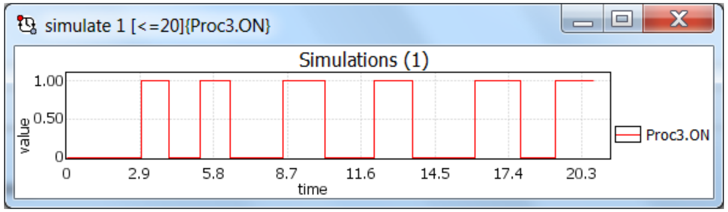




ALW(OFF) EVEN ON)



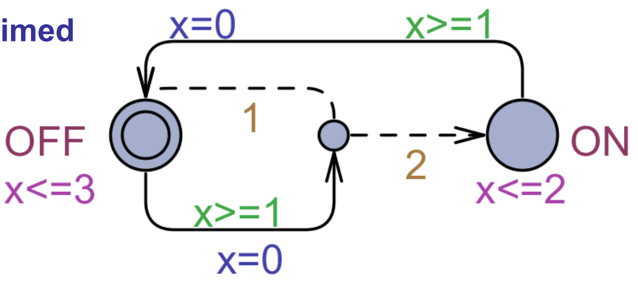
ALW(OFF) EVEN$_{s<3, p>0.90}$ ON)

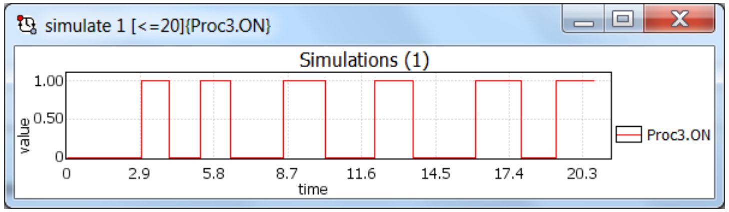


ALW(OFF) EVEN$_{t<5, p>0.90}$ ON)

Models

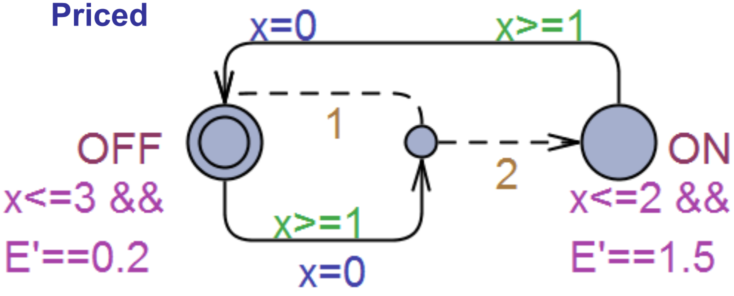
Timed

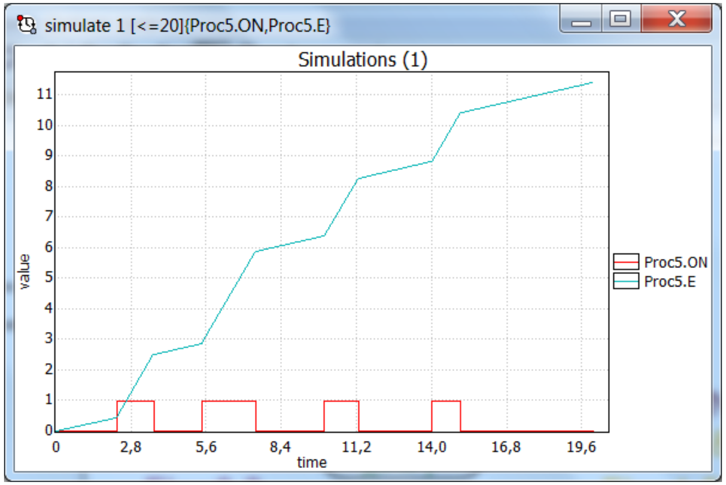



simulate 1 [<=20]{Proc3.ON}

ALW(OFF) EVEN$_{t<5,\ p>0.90}$ ON)

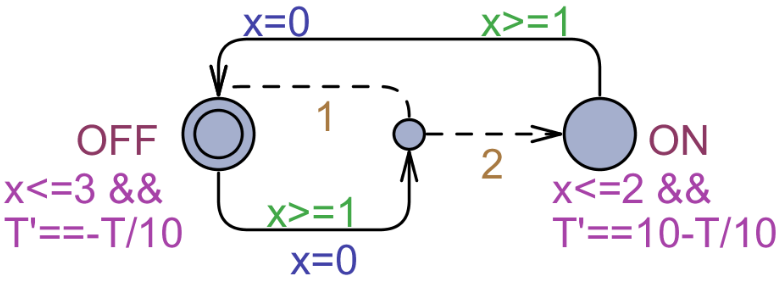
Priced

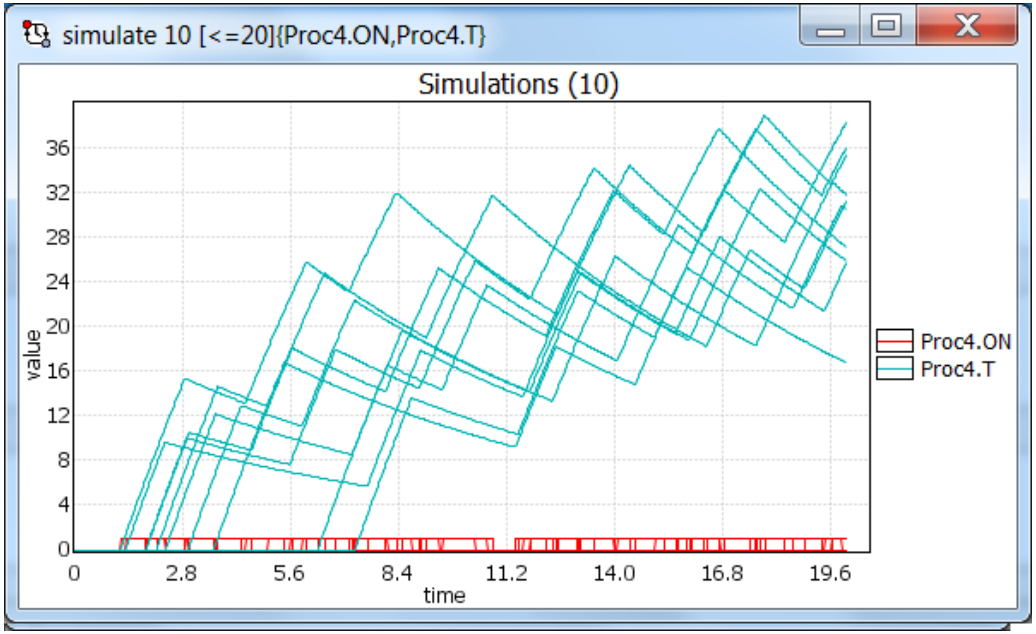



simulate 1 [<=20]{Proc5.ON,Proc5.E}

ALW(OFF) EVEN$_{E<0.9,\ p>0.90}$ ON)

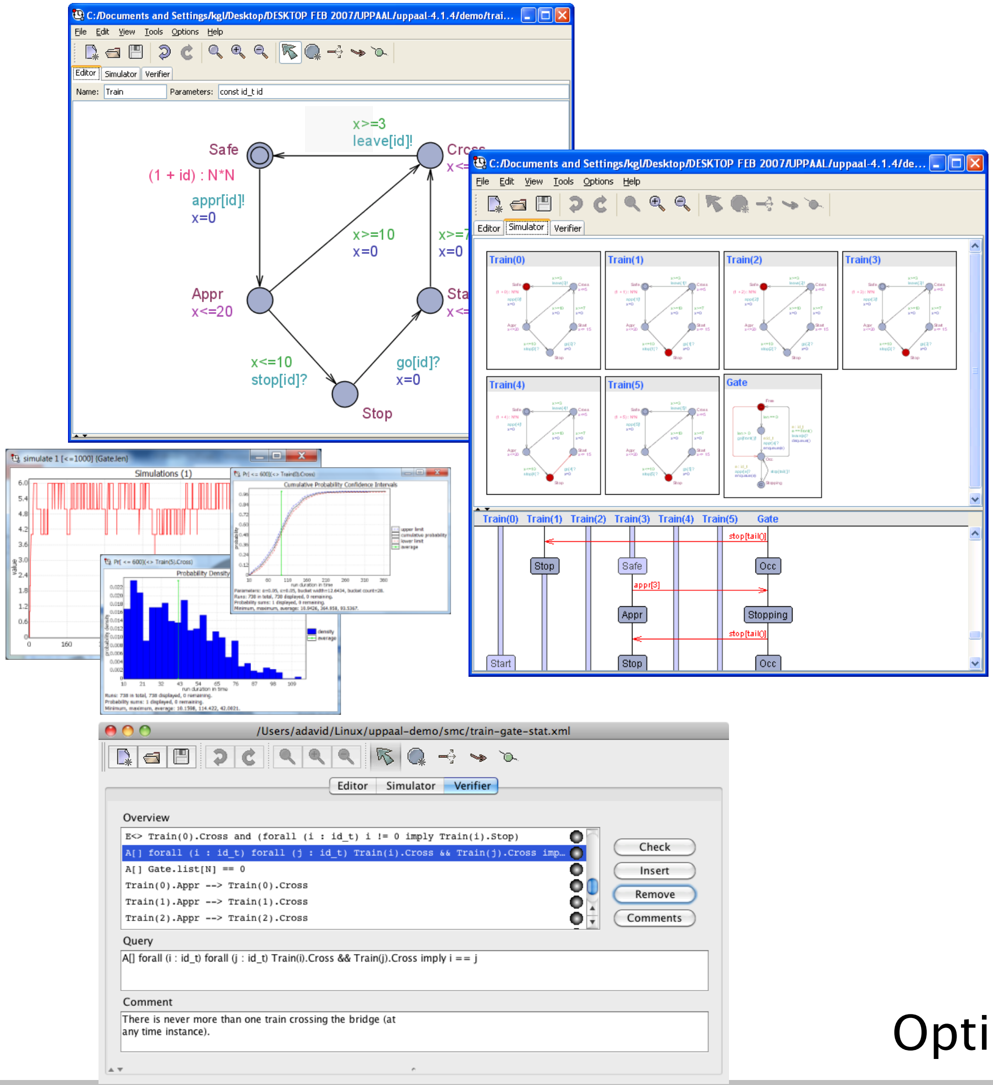
Models





Hybrid

UPPAAL Tool Suit


C:/Documents and Settings/kgl/Desktop/DESKTOP FEB 2007/UPPAAL/uppaal-4.1.4/demo/trai...
File Edit View Tools Options Help
Editor Simulator Verifier
Name: Train Parameters: const id_t id
Safe
(1 + id) : N*N
appr[id]!
x=0
x>=3
leave[id]!
Cross
x>=10
x=0
Appr
x<=20
x<=10
stop[id]?
go[id]?
x=0
Stop
Start
C:/Documents and Settings/kgl/Desktop/DESKTOP FEB 2007/UPPAAL/uppaal-4.1.4/de...
Train(0)
Train(1)
Train(2)
Train(3)
Train(4)
Train(5)
Gate
stop[tail()]
appr[3]
Safe
Appr
Occ
Stopping
Simulations (1)
Cumulative Probability Confidence Intervals
Probability Density
/Users/adavid/Linux/uppaal-demo/smc/train-gate-stat.xml
Editor Simulator Verifier
Check
Insert
Remove
Comments

Overview

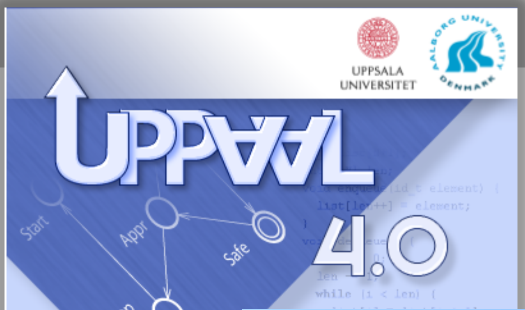
```
E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)
A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imp…
A[] Gate.list[N] == 0
Train(0).Appr --> Train(0).Cross
Train(1).Appr --> Train(1).Cross
Train(2).Appr --> Train(2).Cross
```

Query

A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j

Comment

There is never more than one train crossing the bridge (at any time instance).


UPPSALA UNIVERSITET
AALBORG UNIVERSITY DENMARK
UPPAAL
4.0
Start
Appr
Safe
Stop
Copyright 1995-2006 by Uppsala Universi
More information at http://www.uppaal.com

Verification	CLASSIC
Cost Optimization	CORA
Synthesis	TIGA
Component	ECDAR
Testing	TRON
Performance Analysis	SMC
Optimal Synthesis	STRATEGO

Contributors

@UPPsala

UPPSALA UNIVERSITET

- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@AALborg

AALBORG UNIVERSITY DENMARK

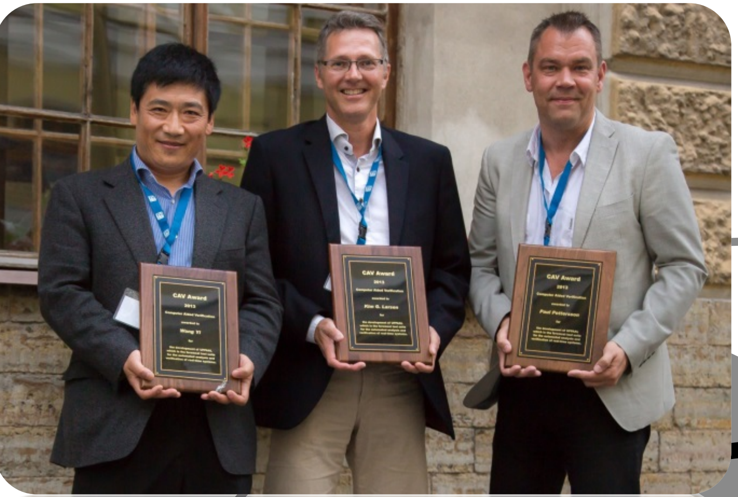
- Kim G Larsen
- Alexandre David
- Marius Mikucionis
- Gerd Behrman
- Arne Skou
- Brian Nielsen
- Jacob I. Rasmussen
- Thomas Chatain

@Elsewhere

- Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen, Jan Tretmans, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson…

Origin of UPPAAL

TAU
CCS & Modal Transition Systems
Refinements
Modal Mu-Calculus
Explicit State Representation
Prolog



1995

UPPAAL
Timed Automata
TCTL
Zones
C++ & Java

2007

2013

EPSILON
TCCS
Timed Refinements
Timed Mu-Calculus
Regions
Prolog<

UP4ALL

CAV Award

Overview

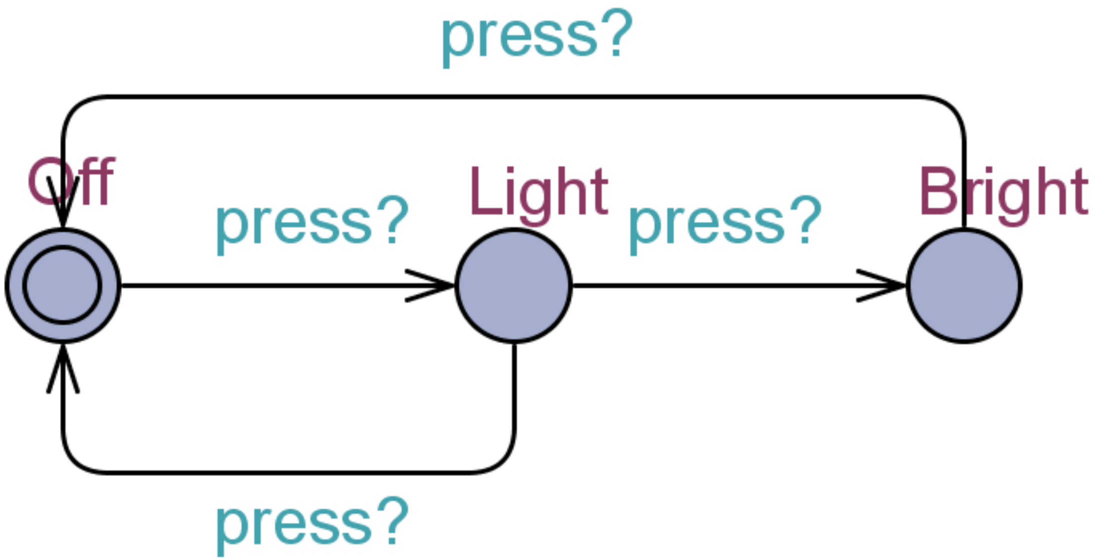
- Timed Automata / UPPAAL
    - Verification
- Priced Timed Automata / UPPAAL CORA
    - Optimal Scheduling (multicore applications)
    - Optimal Infinite Scheduling
    - Multi objective optimization
- Schedulability Analysis & Scheduling
    - Single Core, Multi Core
    - Dynamic voltage Scheduling
    - Energy Automata
- Stochastic Priced Timed Automata / UPPAAL SMC
    - Statistical Model Checking
    - Low Power Medium Access Protocol
    - Stochastic Hybrid Automata
    - Energy-Aware Buildings
    - Battery-Aware Scheduling
- Stochastic Priced Timed Games / UPPAAL STRATEGO
    - Optimal & Safe Synteses
    - Energy-Aware and Optimal Satelitte Scheduling
- Conclusion

Overview

- Timed Automata / UPPAAL
  - Verification
- Priced Timed Automata / UPPAAL CORA
  - Optimal Scheduling (multicore applications)
  - Optimal Infinite Scheduling
  - Multi objective optimization
- Schedulability Analysis & Scheduling
  - Single Core, Multi Core
  - Dynamic voltage Scheduling
  - Energy Automata
- Stochastic Priced Timed Automata / UPPAAL SMC
  - Statistical Model Checking
  - Low Power Medium Access Protocol
  - Stochastic Hybrid Automata
  - Energy-Aware Buildings
  - Battery-Aware Scheduling
- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synteses
  - Energy-Aware and Optimal Satelitte Scheduling
- Conclusion

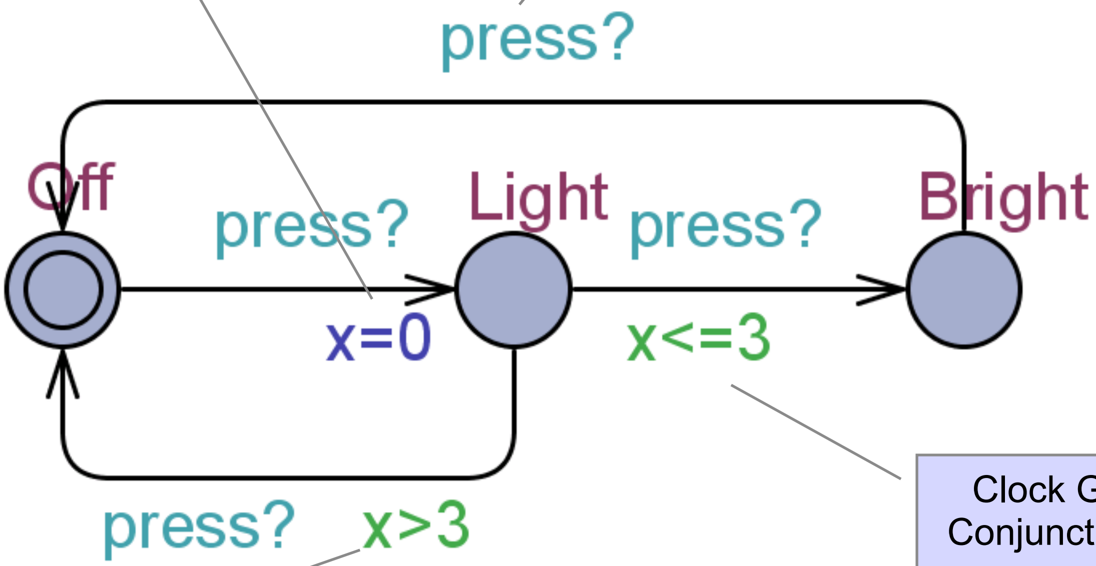
Timed Automata

A Dumb Light Controller



Timed Automata

[Alur & Dill'89]



Reset

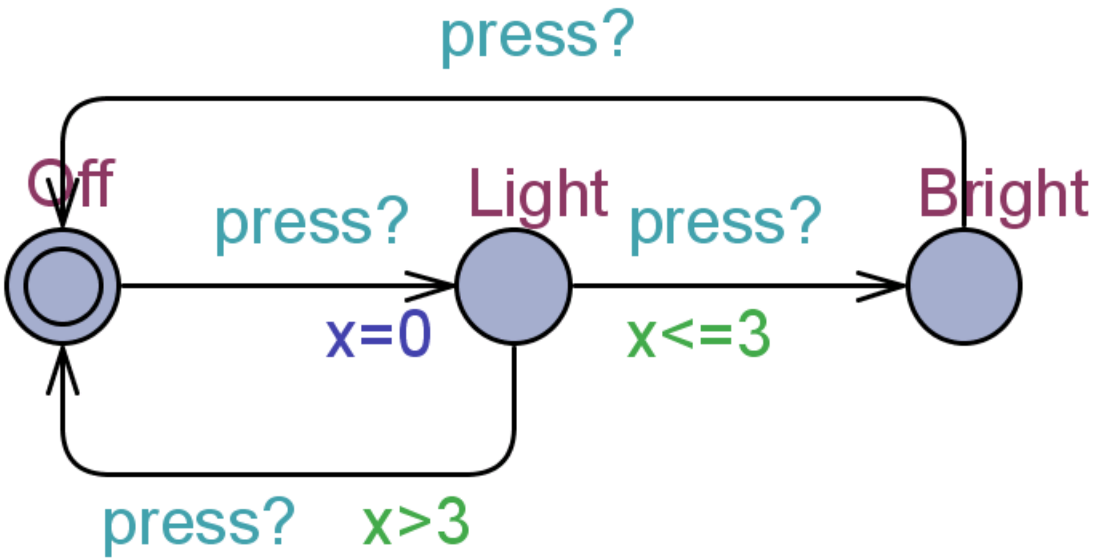
Synchronizing action

Clock Guard Conjunctions of x~n

x: real-valued clock

ADD a clock x

A Timed Automata (Semantics)



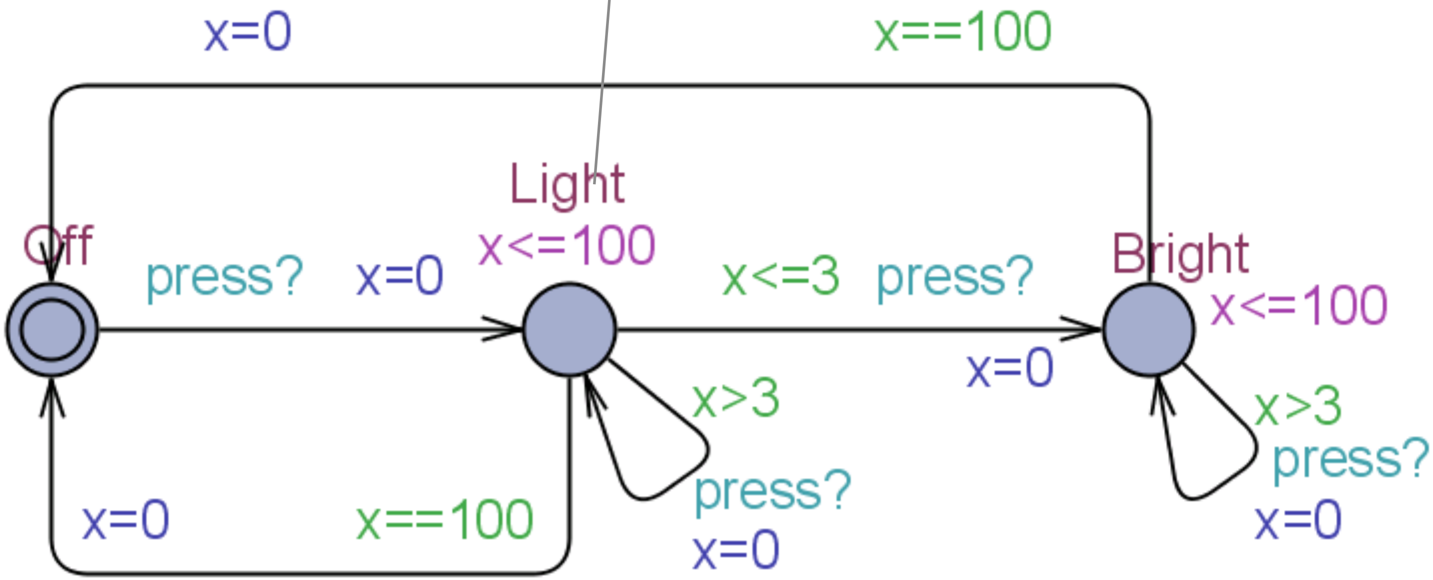
States:
(location , x=v) where v2**R**

Transitions:

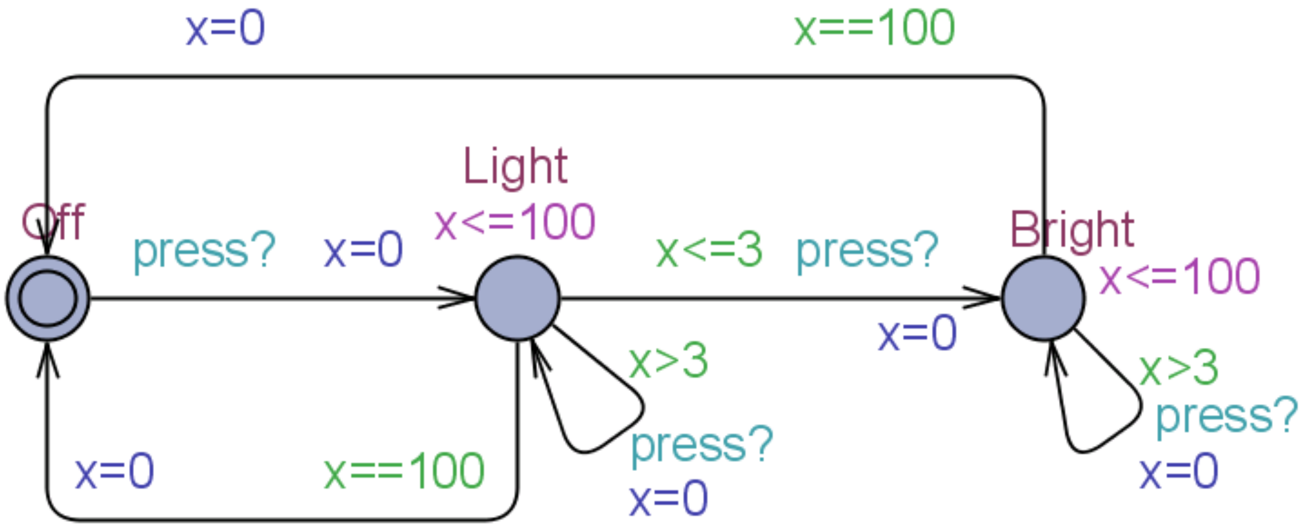
	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 2.51	→ (Light , x=2.51)
press?	→ (Bright , x=2.51)

Intelligent Light Controller

Invariant (Henzinger)



Intelligent Light Controller



Transitions:

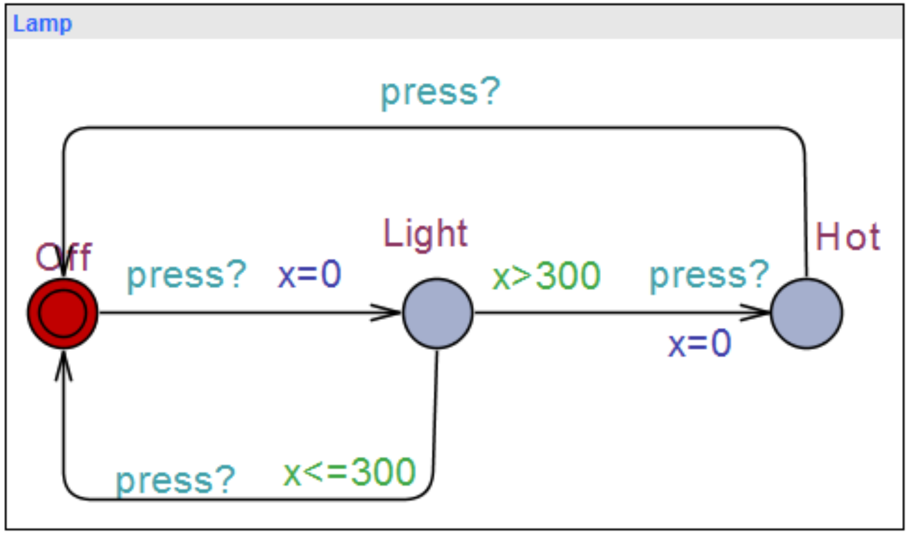
	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 4.51	→ (Light , x=4.51)
press?	→ (Light , x=0)
delay 100	→ (Light , x=100)
τ	→ (Off , x=0)

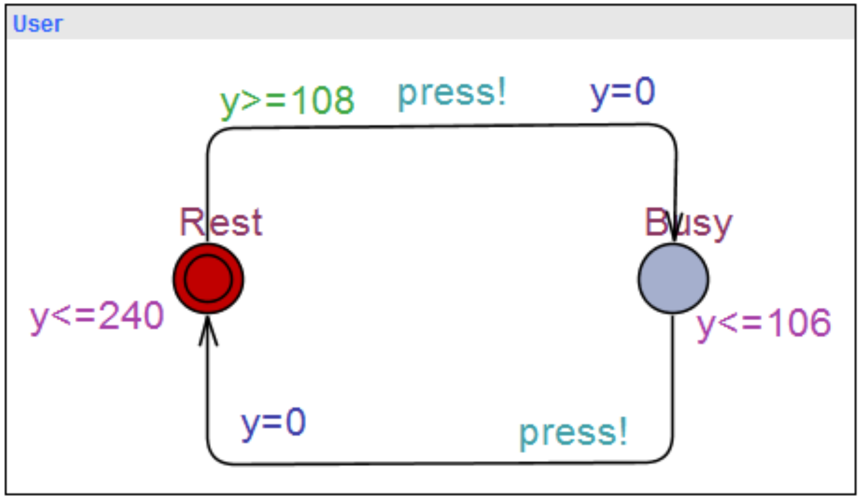
Note: ✗

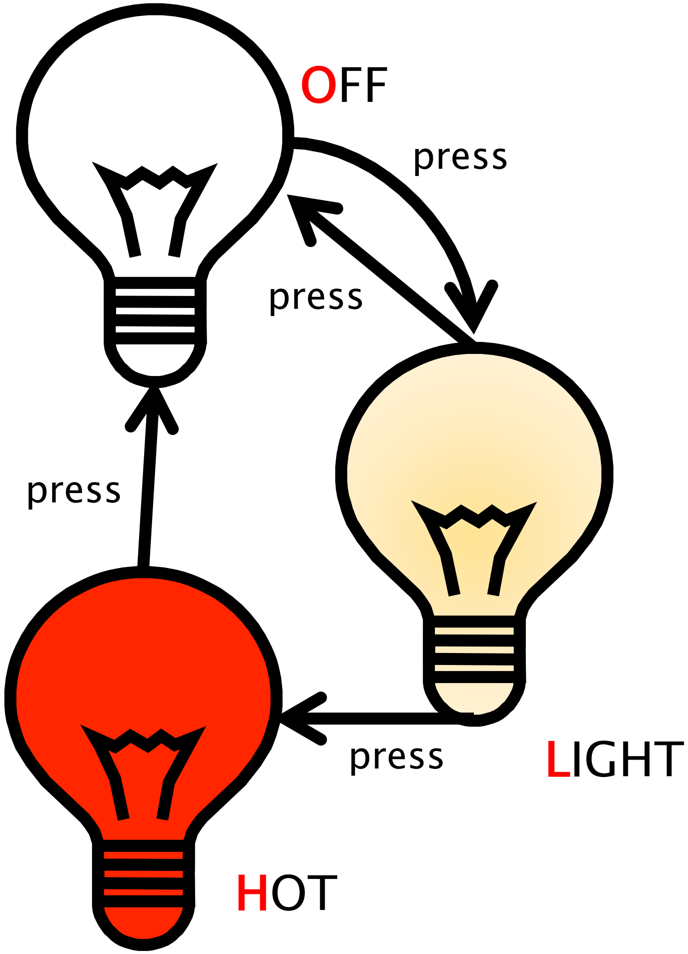
(Light , x=0) delay 103 →

Invariants ensures progress

Intellingent Light Control







OFF

press

press

press

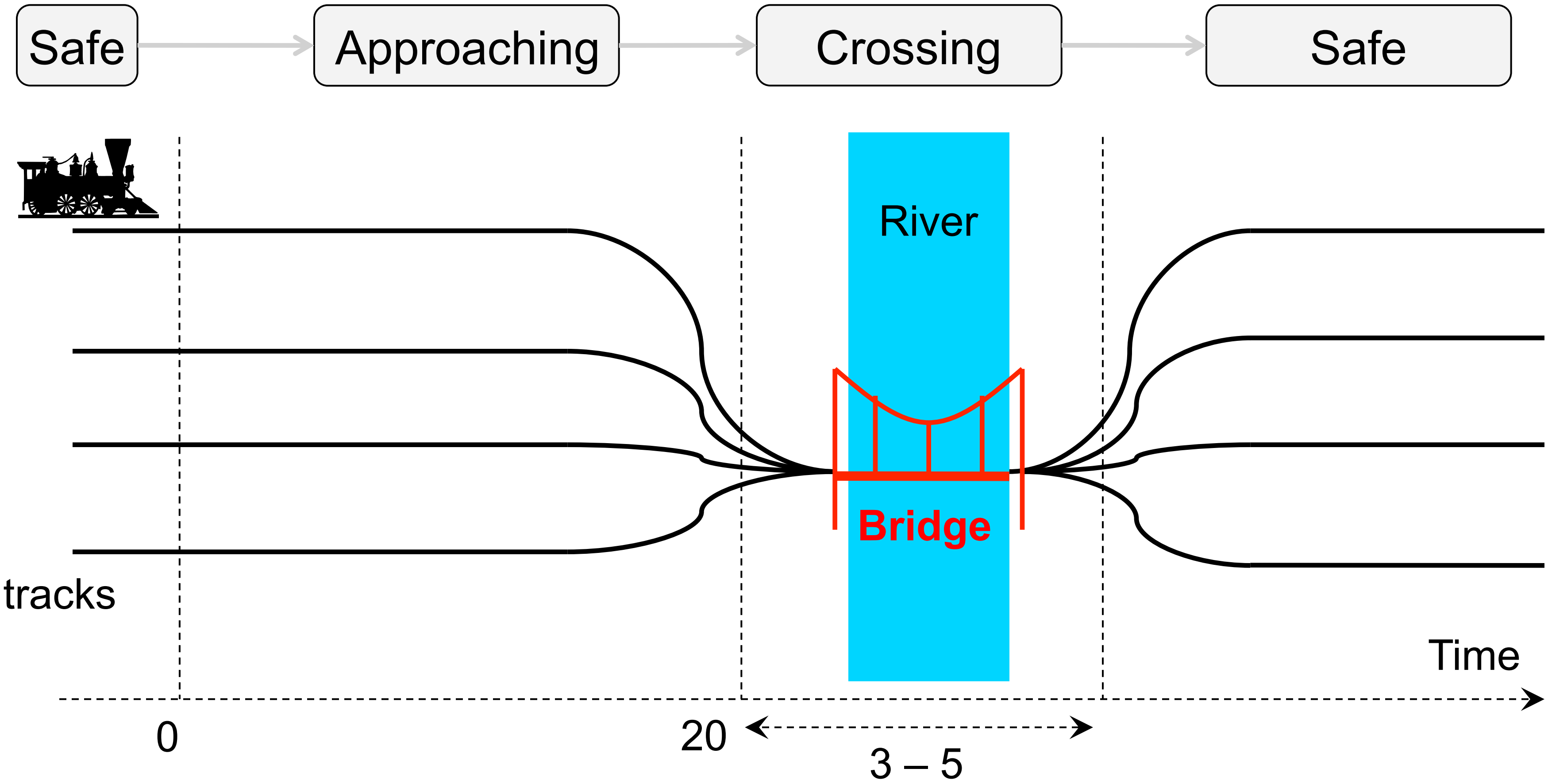
press

LIGHT

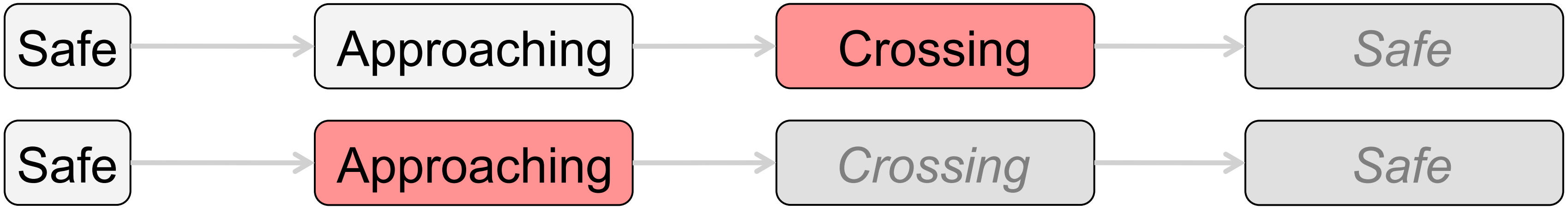
HOT

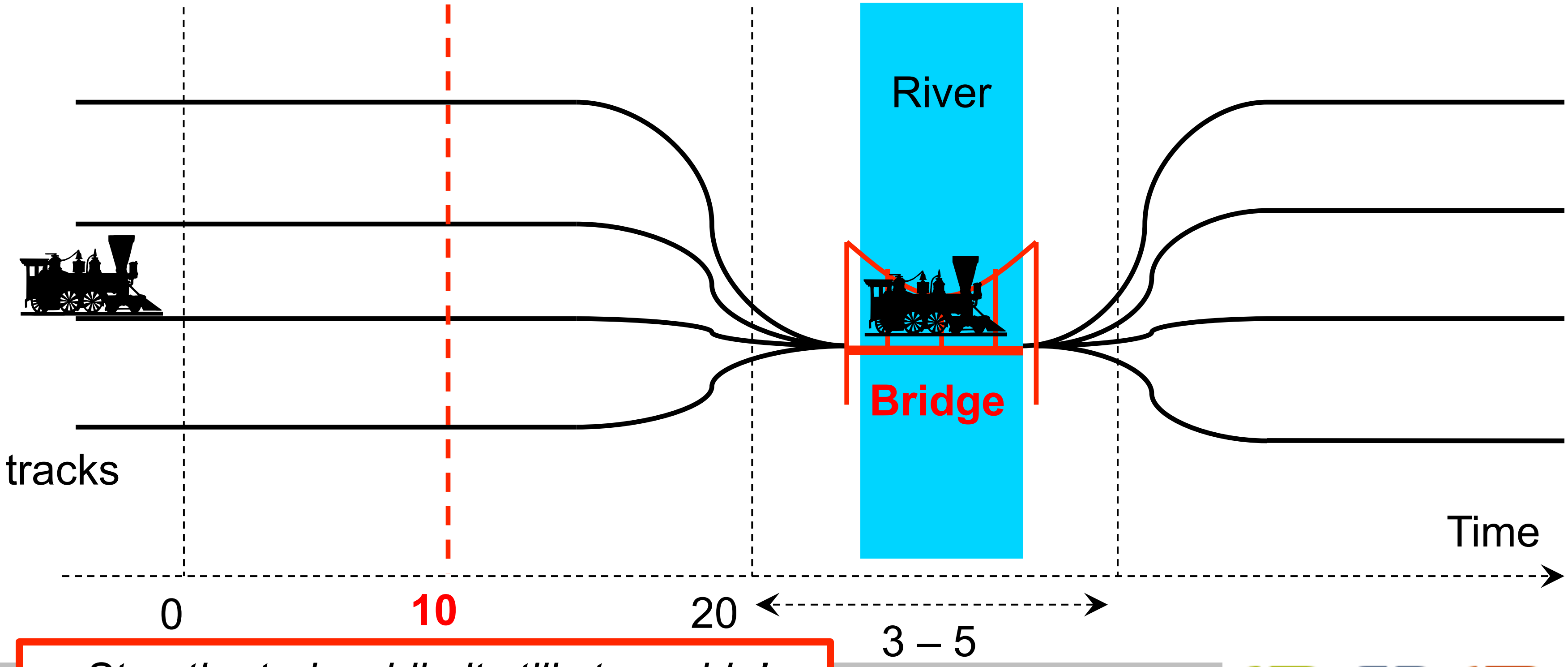
DEMO

Train Crossing



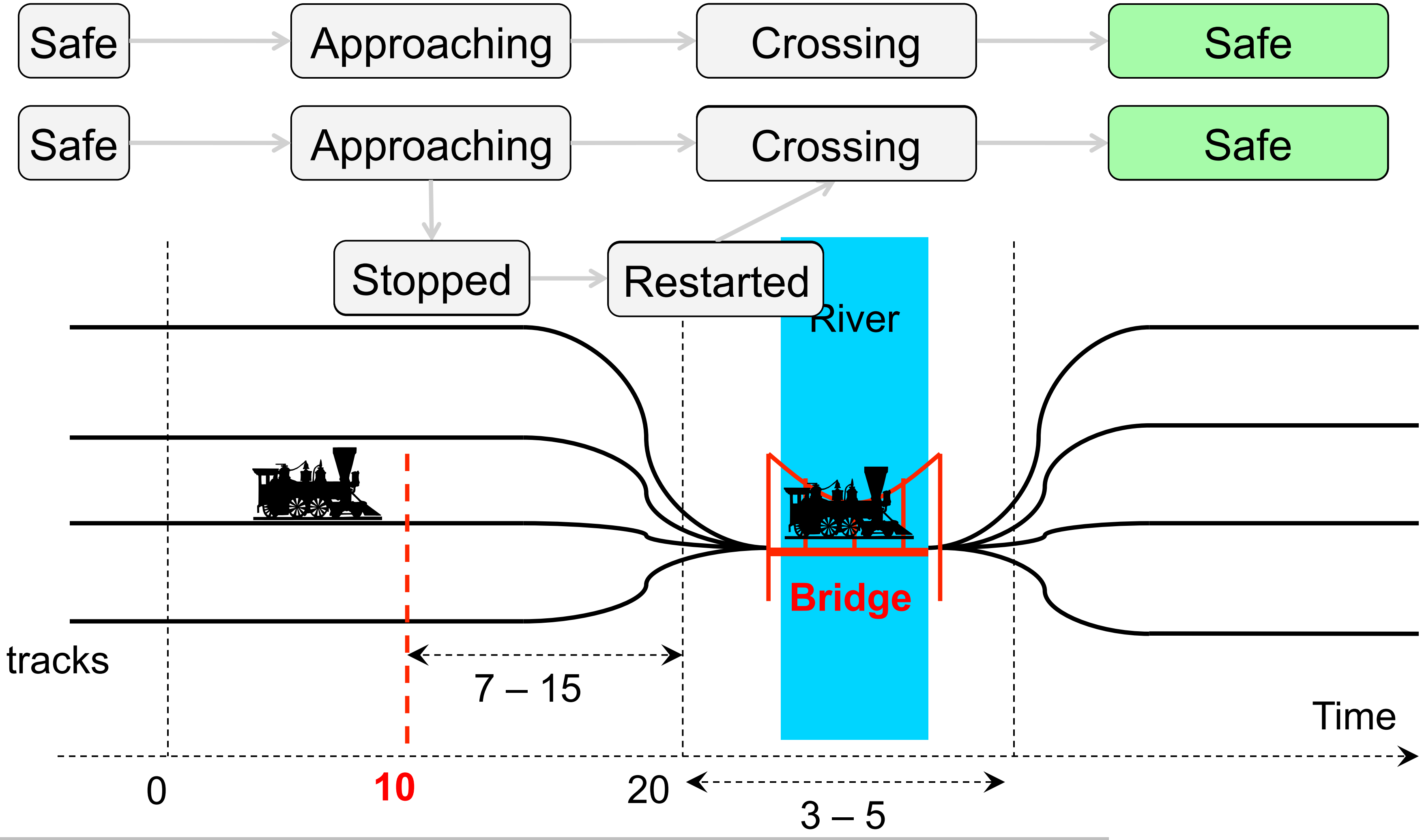
Train Crossing



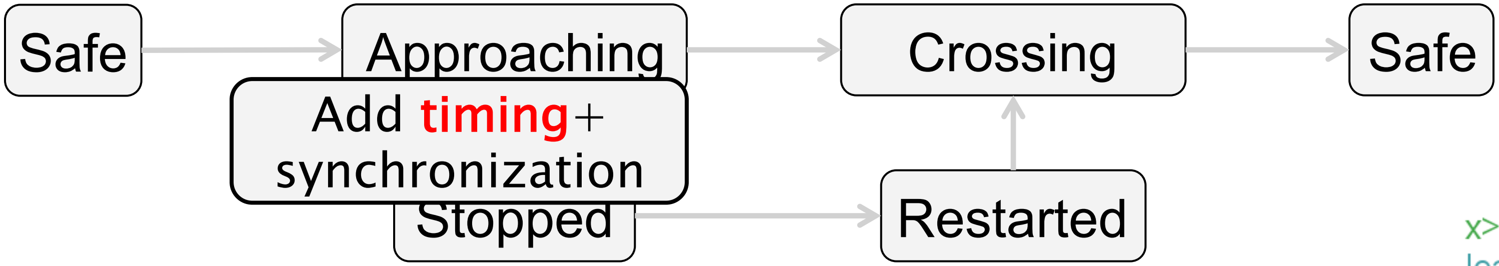


Stop the train while it still stoppable!

Train Crossing



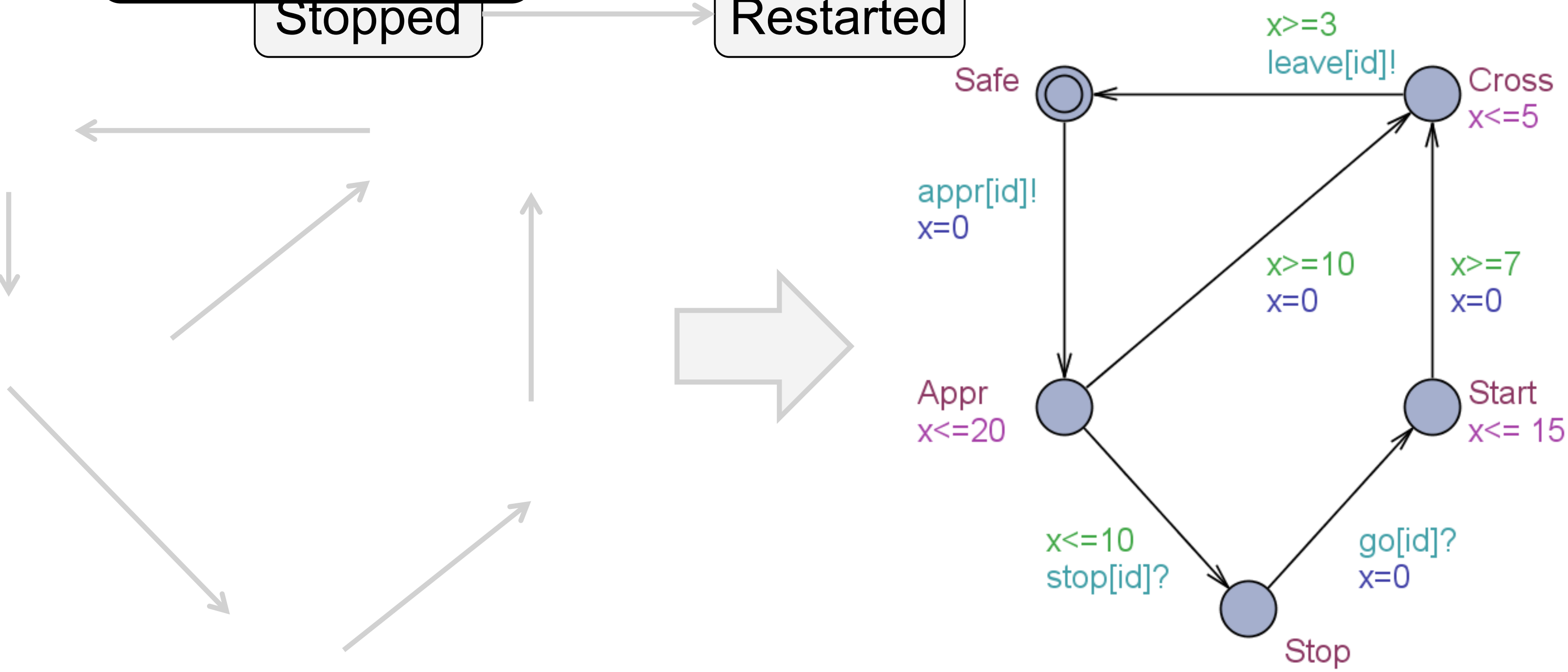
Train Crossing



Safe → Approaching → Crossing → Safe

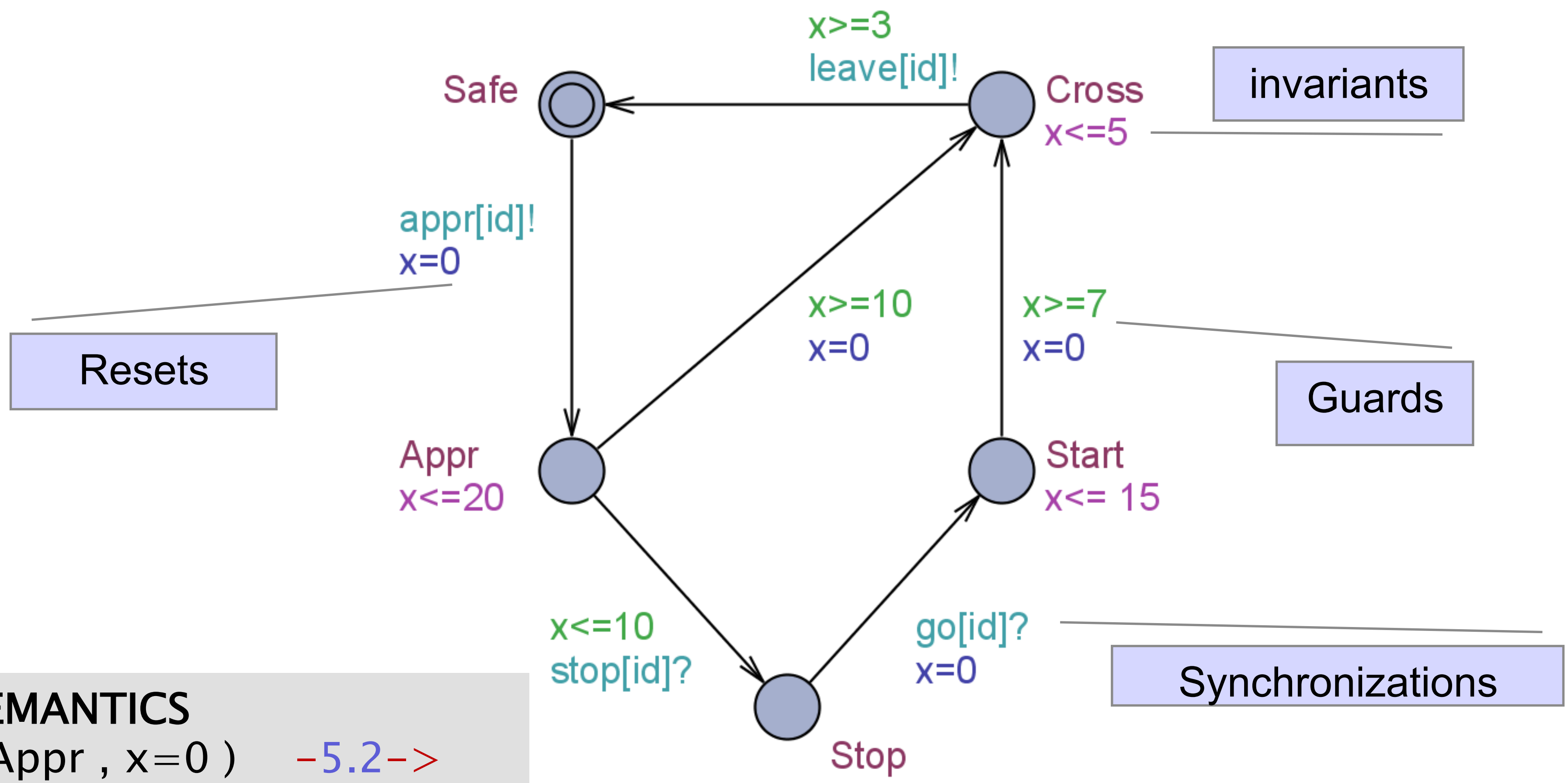
Stopped → Restarted → Crossing

Add timing+ synchronization



Timed Automata [Train]

= Finite State Control
+ Real Valued Clocks



SEMANTICS

(Appr , x=0) -5.2->
(Appr , x=5.2) -stop? ->
(Stop , x=5.2)

DEMO

AALBORG UNIVERSITY
DENMARK

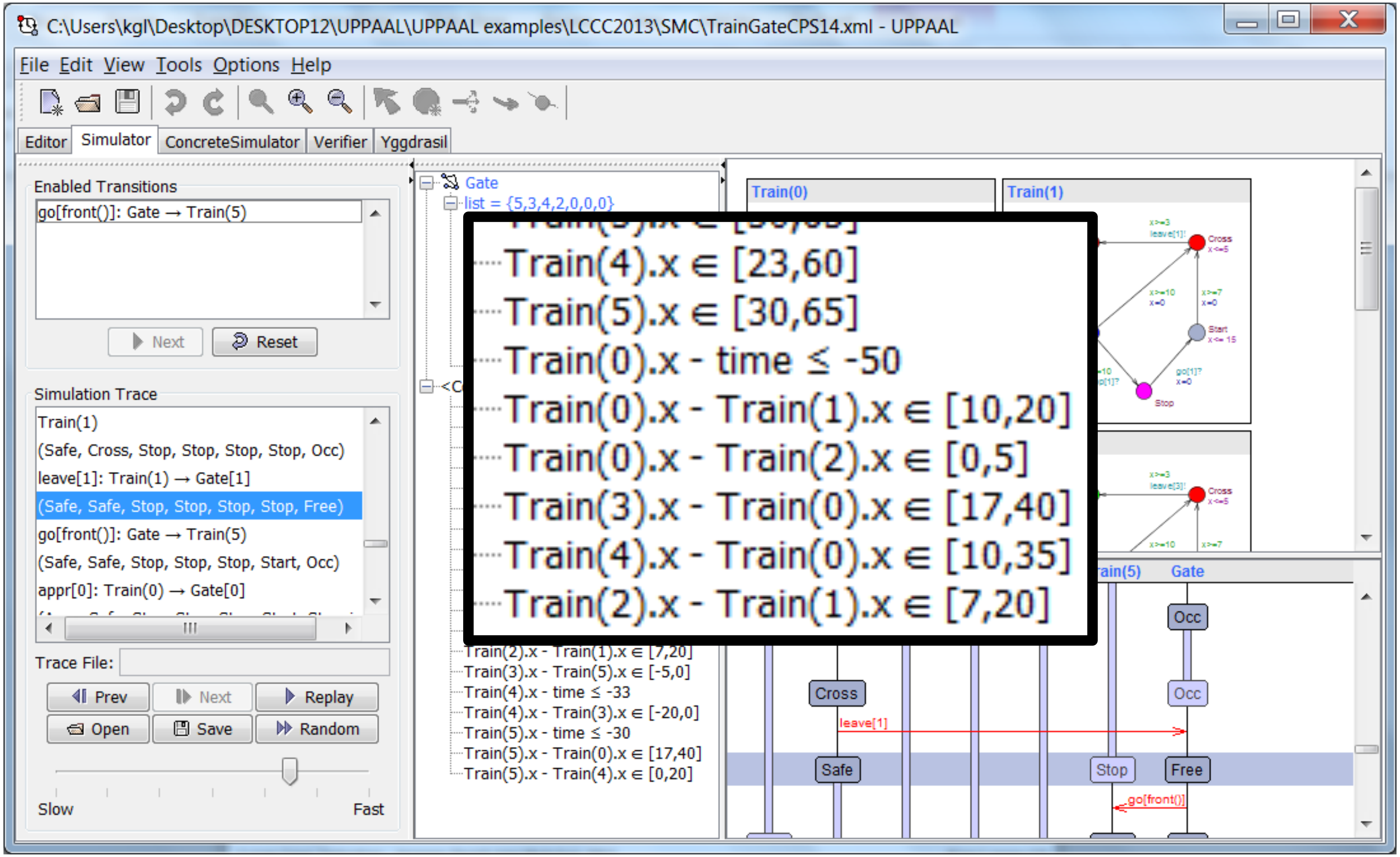
Logical Specifications

- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{.\ t} Q$

The expressions *P* and *Q* must be type safe, side effect free, and evaluate to a boolean.

Only references to integer variables, constants, clocks, and locations are allowed (and arrays of these).

THE ”secret” of UPPAAL



Enabled Transitions:

```
go[front()]: Gate → Train(5)
```

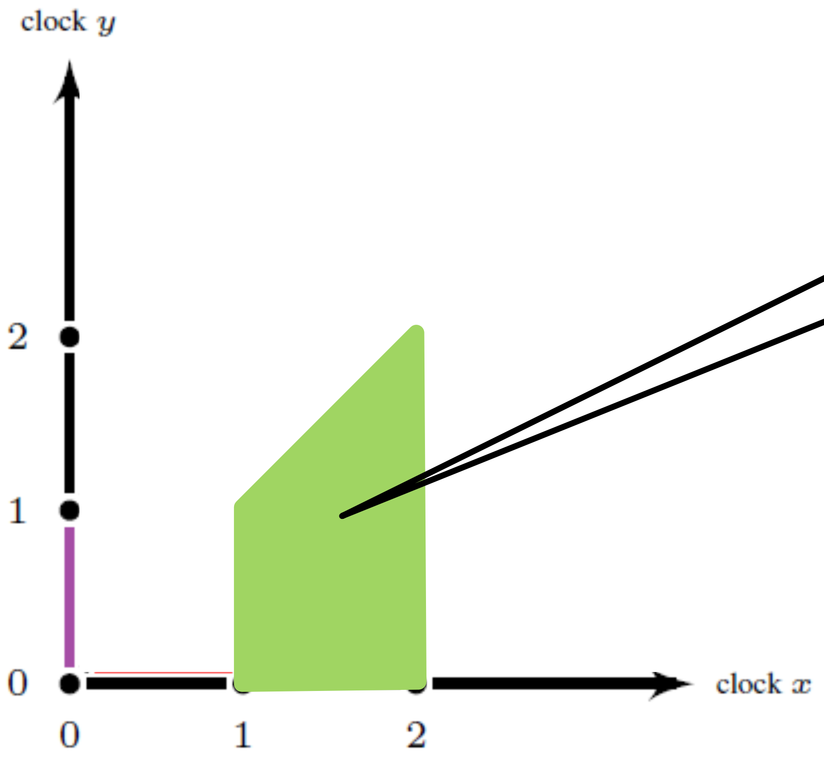
Simulation Trace:

```
Train(1)
(Safe, Cross, Stop, Stop, Stop, Stop, Occ)
leave[1]: Train(1) → Gate[1]
(Safe, Safe, Stop, Stop, Stop, Stop, Free)
go[front()]: Gate → Train(5)
(Safe, Safe, Stop, Stop, Stop, Start, Occ)
appr[0]: Train(0) → Gate[0]
```

```
Gate
list = {5,3,4,2,0,0,0}
```

```
Train(4).x ∈ [23,60]
Train(5).x ∈ [30,65]
Train(0).x - time ≤ -50
Train(0).x - Train(1).x ∈ [10,20]
Train(0).x - Train(2).x ∈ [0,5]
Train(3).x - Train(0).x ∈ [17,40]
Train(4).x - Train(0).x ∈ [10,35]
Train(2).x - Train(1).x ∈ [7,20]
Train(3).x - Train(5).x ∈ [-5,0]
Train(4).x - time ≤ -33
Train(4).x - Train(3).x ∈ [-20,0]
Train(5).x - time ≤ -30
Train(5).x - Train(0).x ∈ [17,40]
Train(5).x - Train(4).x ∈ [0,20]
```

Zones – From Finite to Efficiency



A zone **Z**:

1· x · 2 Æ

0· y · 2 Æ

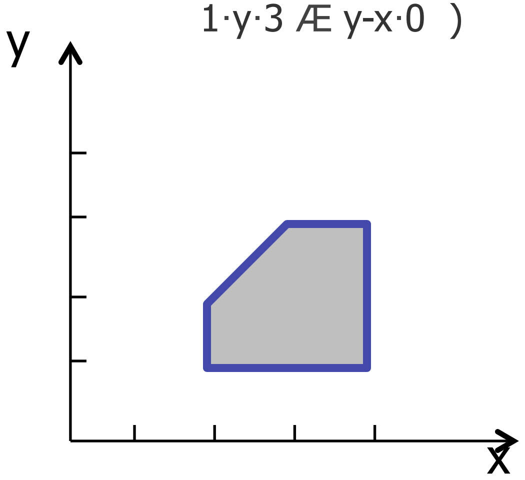
x - y ¸ 0

Theorem

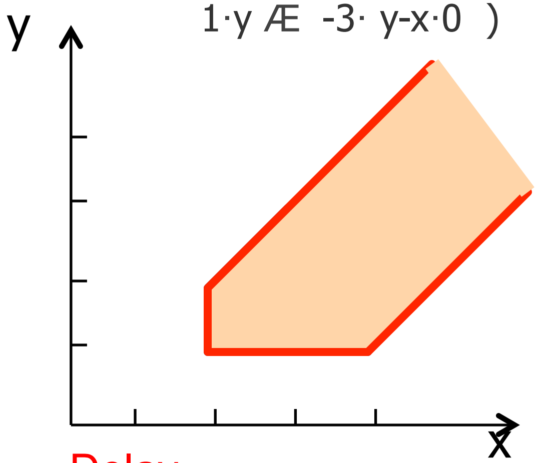
The number of regions is $n! \cdot 2^n \cdot \prod_{x \in C}(2c_x + 2)$.

Zones – Operations

(n, 2·x·4 Æ
1·y·3 Æ y-x·0)

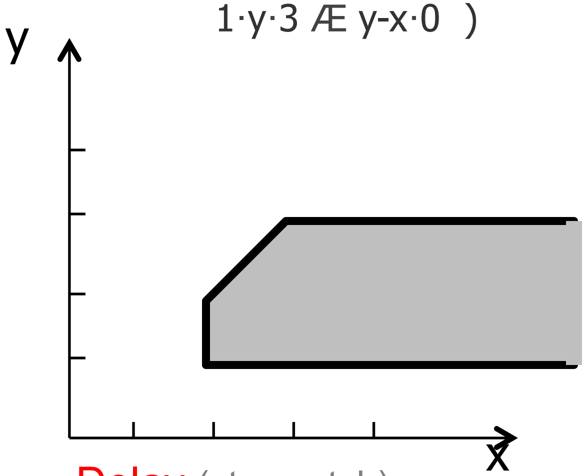


(n, 2·x Æ
1·y Æ -3· y-x·0)



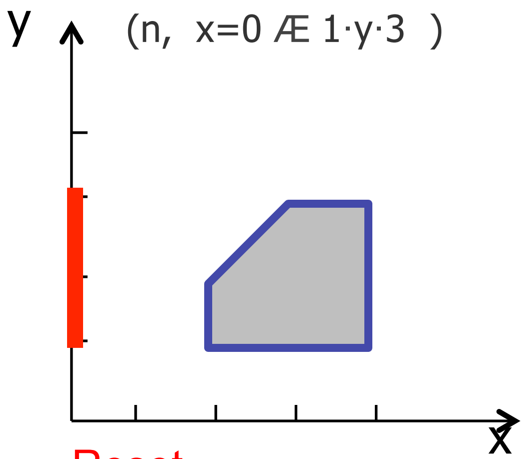
Delay

(n, 2·x Æ
1·y·3 Æ y-x·0)



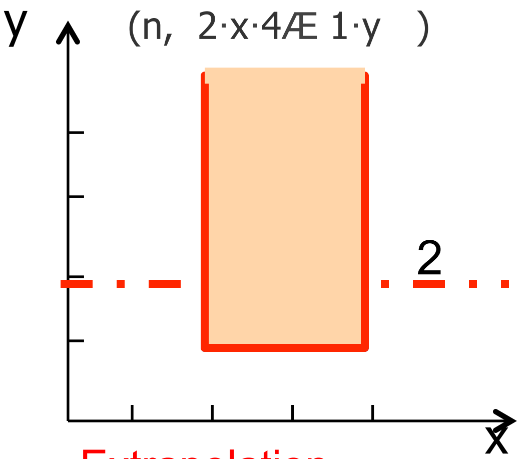
Delay (stopwatch)

(n, x=0 Æ 1·y·3)

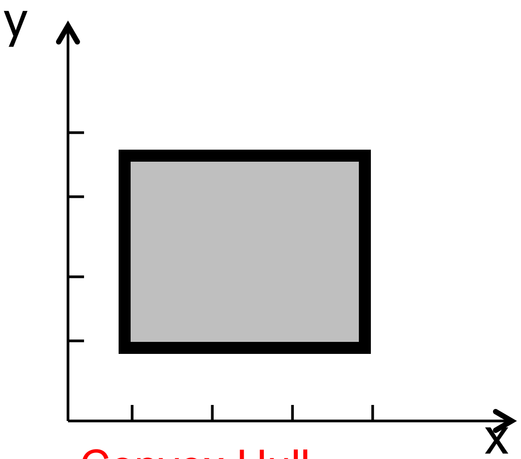


Reset

(n, 2·x·4Æ 1·y)



Extrapolation



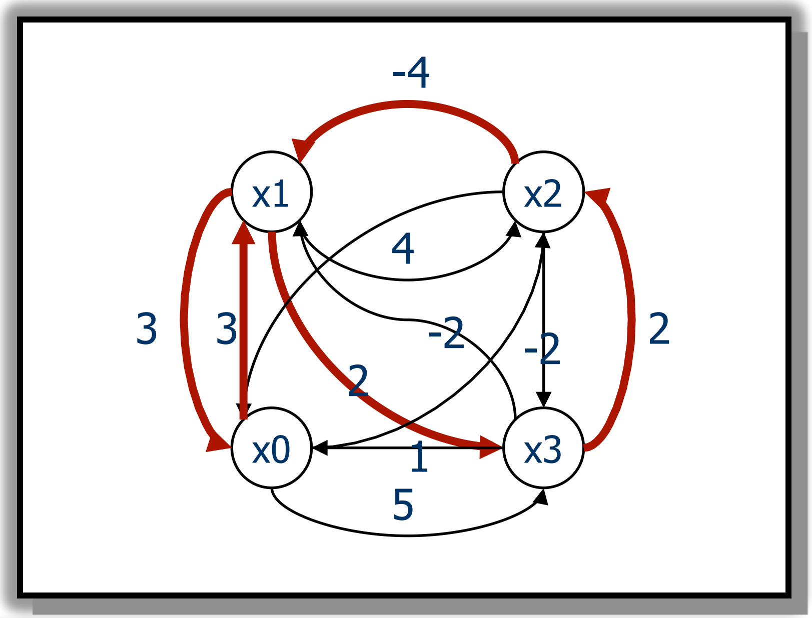
Convex Hull

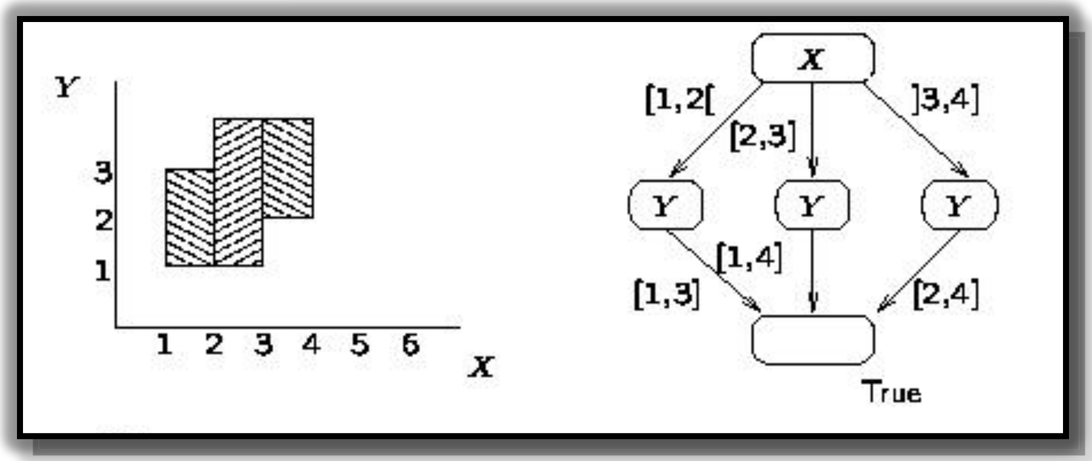
Datastructures for Zones

- Difference Bounded Matrices (DBMs)
- Minimal Constraint Form

  [RTSS97]
- Clock Difference Diagrams

  [CAV99]



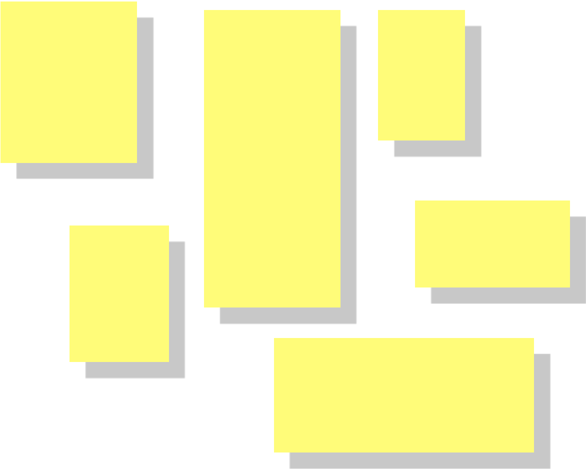


Overview

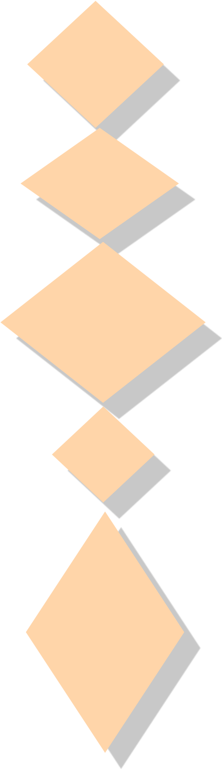
- Timed Automata / UPPAAL
  - Verification
- Priced Timed Automata / UPPAAL CORA
  - Optimal Scheduling (multicore applications)
  - Optimal Infinite Scheduling
  - Multi objective optimization
- Schedulability Analysis & Scheduling
  - Single Core, Multi Core
  - Dynamic voltage Scheduling
  - Energy Automata
- Stochastic Priced Timed Automata / UPPAAL SMC
  - Statistical Model Checking
  - Low Power Medium Access Protocol
  - Stochastic Hybrid Automata
  - Energy-Aware Buildings
  - Battery-Aware Scheduling
- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synteses
  - Energy-Aware and Optimal Satelitte Scheduling
- Conclusion

Priced Timed Automata

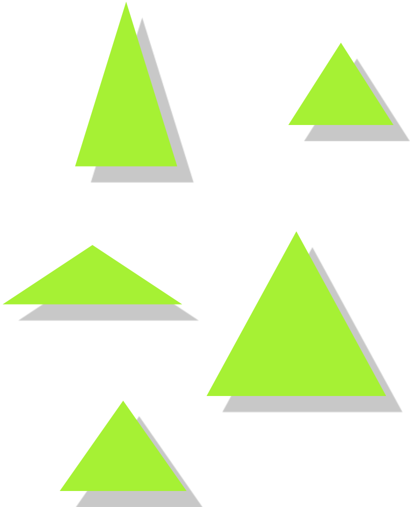
Embedded Systems



Tasks:
Computation times
Deadlines
Dependencies
Arrival patterns
uncertainties

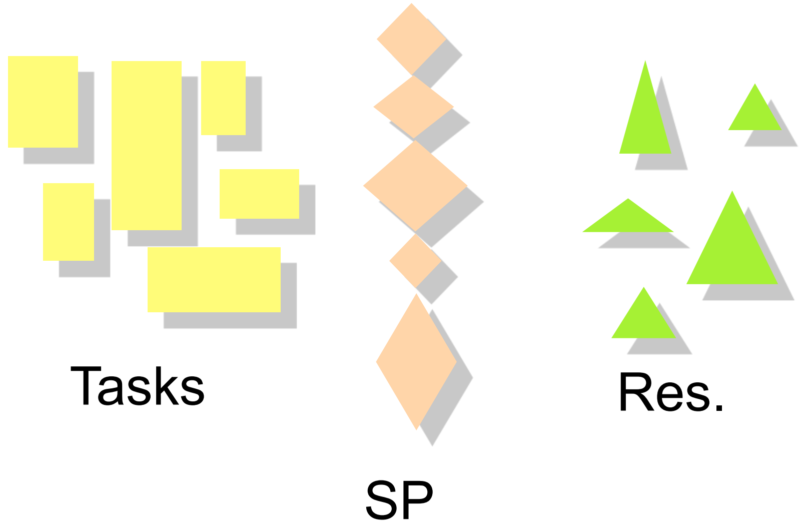


Scheduling Principles (OS)
EDF, FPS, RMS, DVS, ..



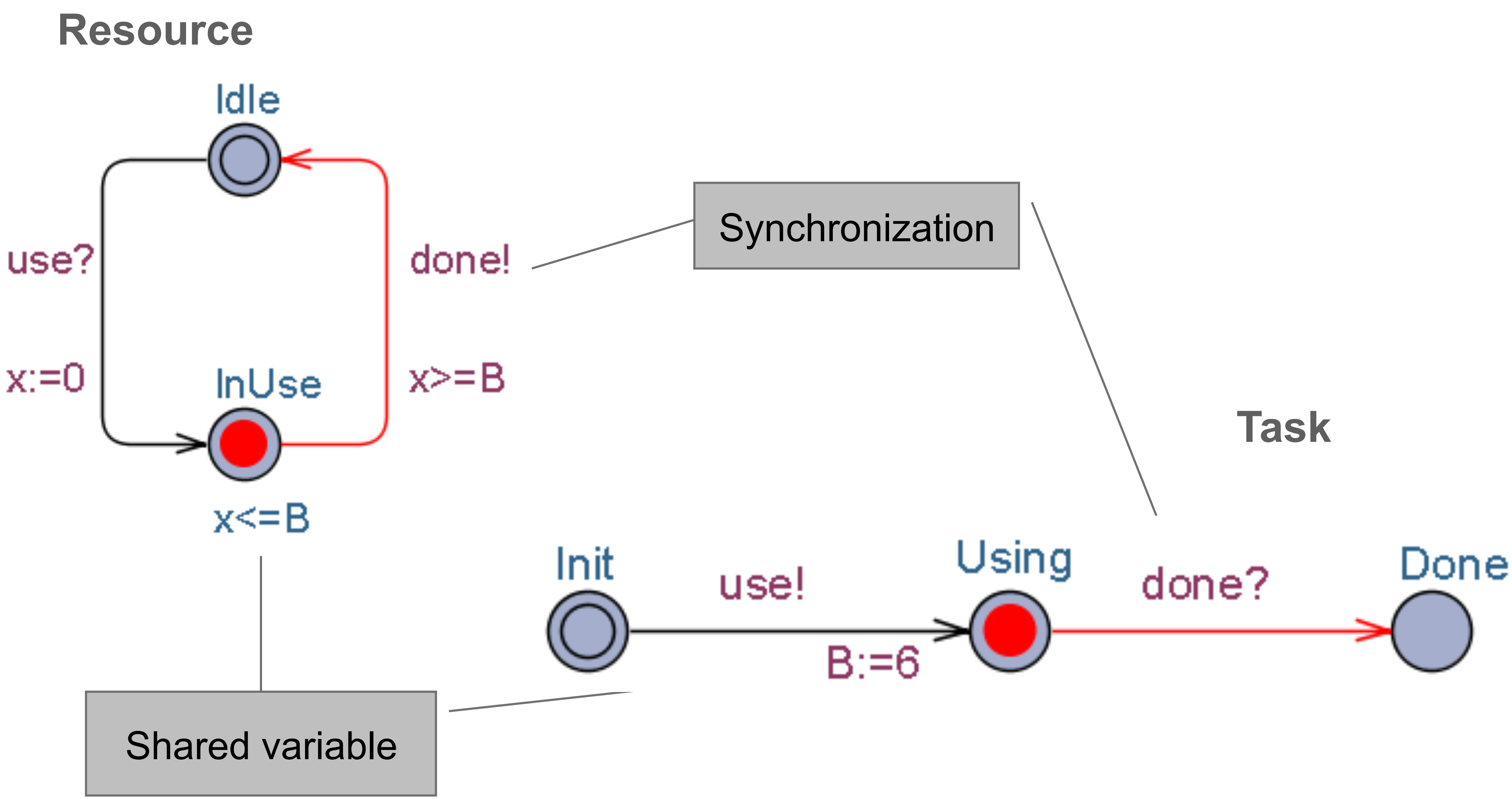
Resources
Execution platform
Energy, Memory
Networks
Drivers
uncertainties

Timing Analyses



- **Worst Case Analysis**: of execution time, energy, memory, etc. of an isolated task.
- **Schedulability analysis**: Verify no deadlines are violated in higher level system for given sched. principnple
- **Scheduling**: Assign resources to tasks

Resources & Tasks

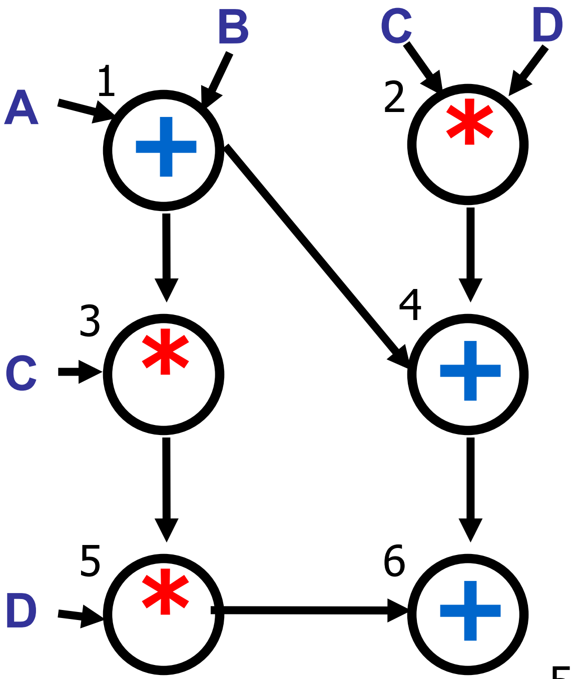

Resource

Task

Synchronization

Shared variable

Task Graph Scheduling – Example



Compute :

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

4

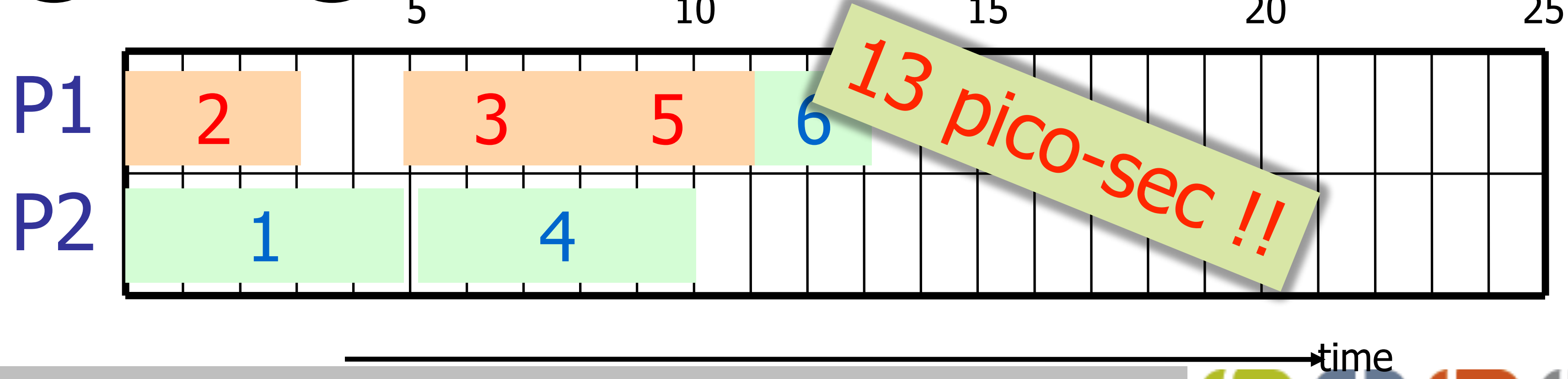
using 2 processors

P1 (fast)

+	2ps
*	3ps

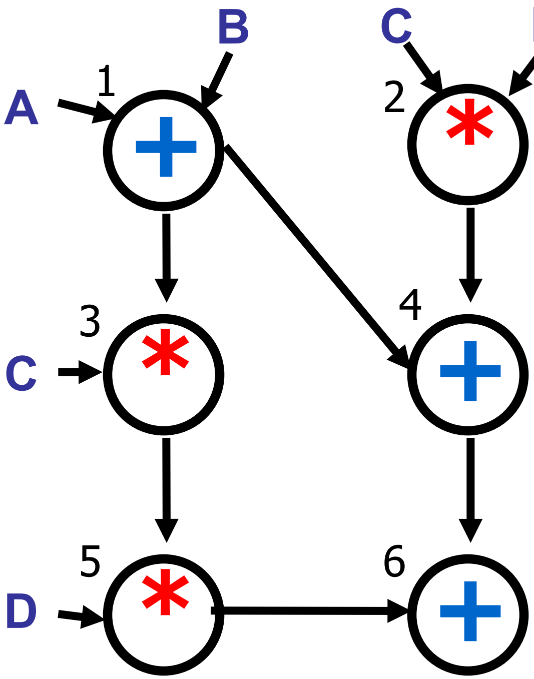
P2 (slow)

+	5ps
*	7ps



13 pico-sec !!

Task Graph Scheduling – Example



Compute :

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

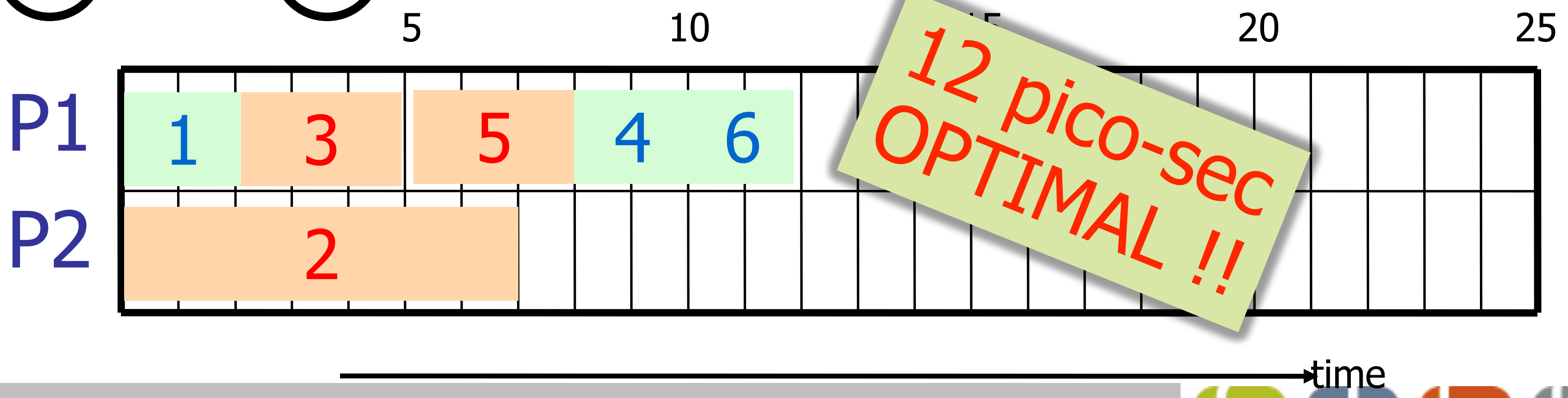
using 2 processors

P1 (fast)

+	2ps
*	3ps

P2 (slow)

+	5ps
*	7ps

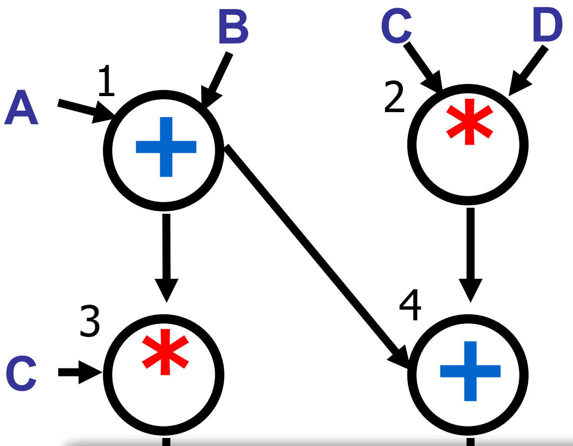


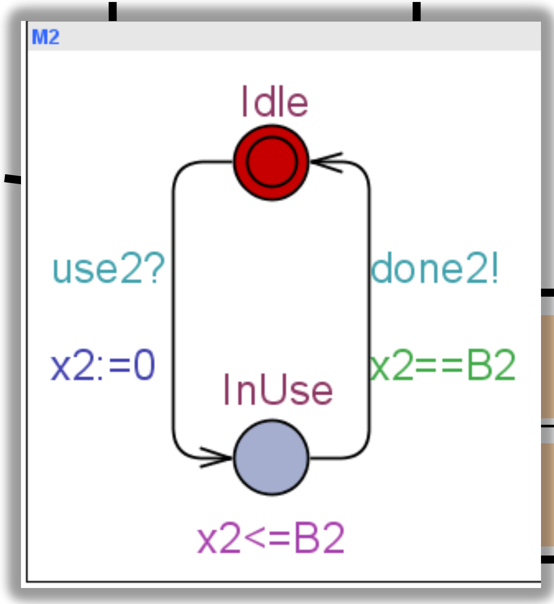
12 pico-sec OPTIMAL !!

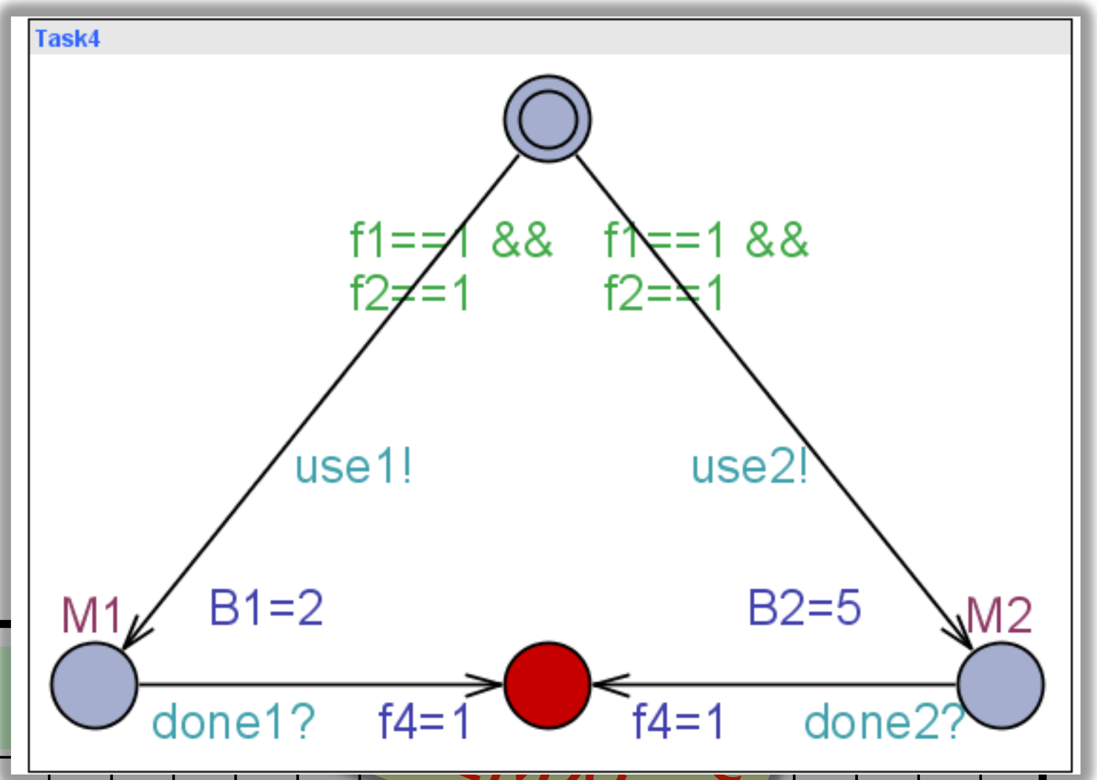
Task Graph Scheduling – Example

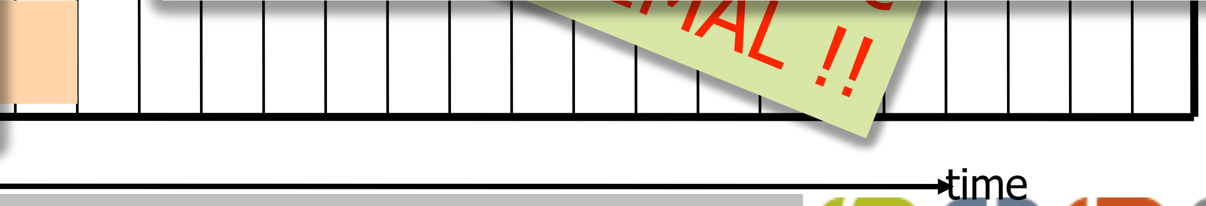
Compute :

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

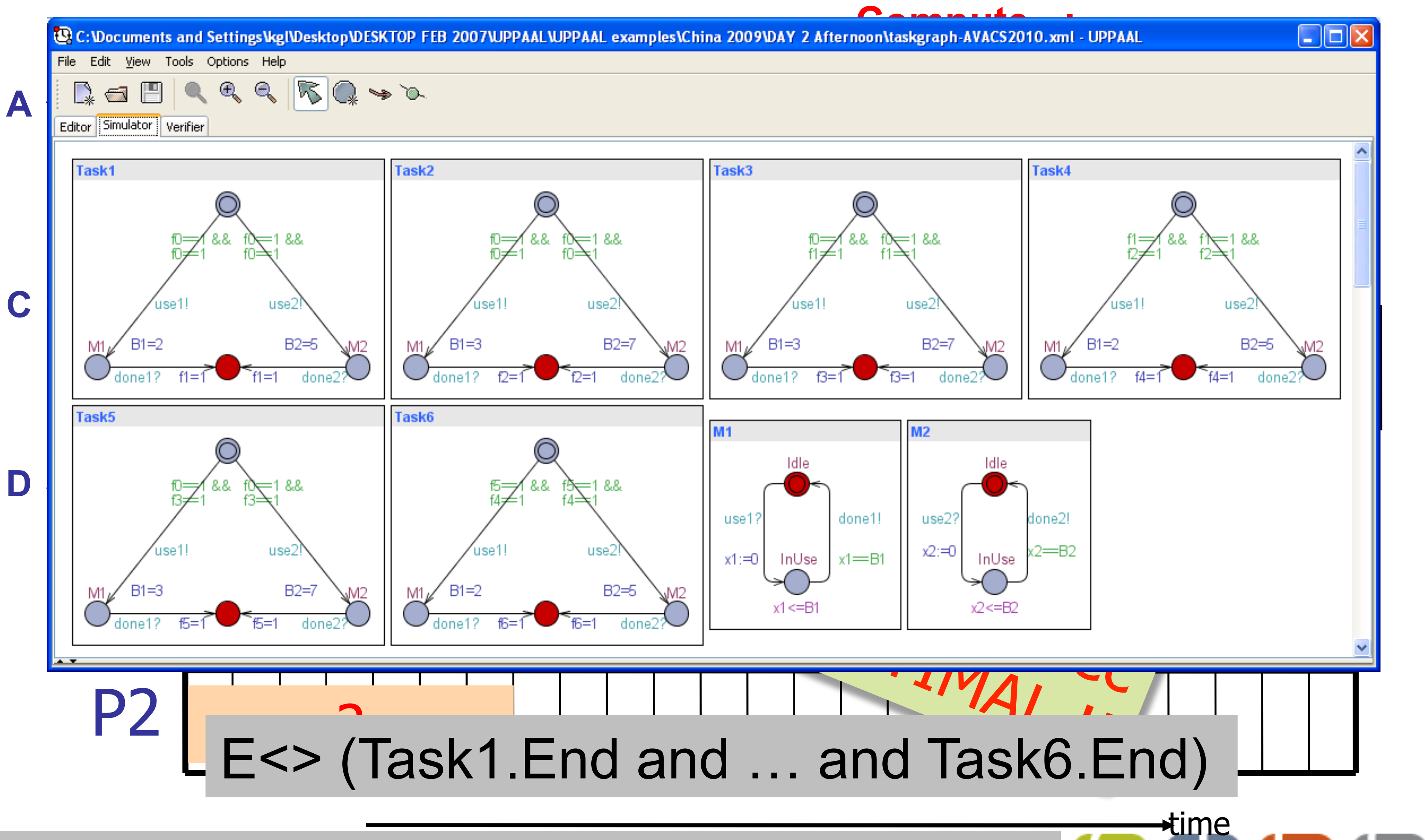






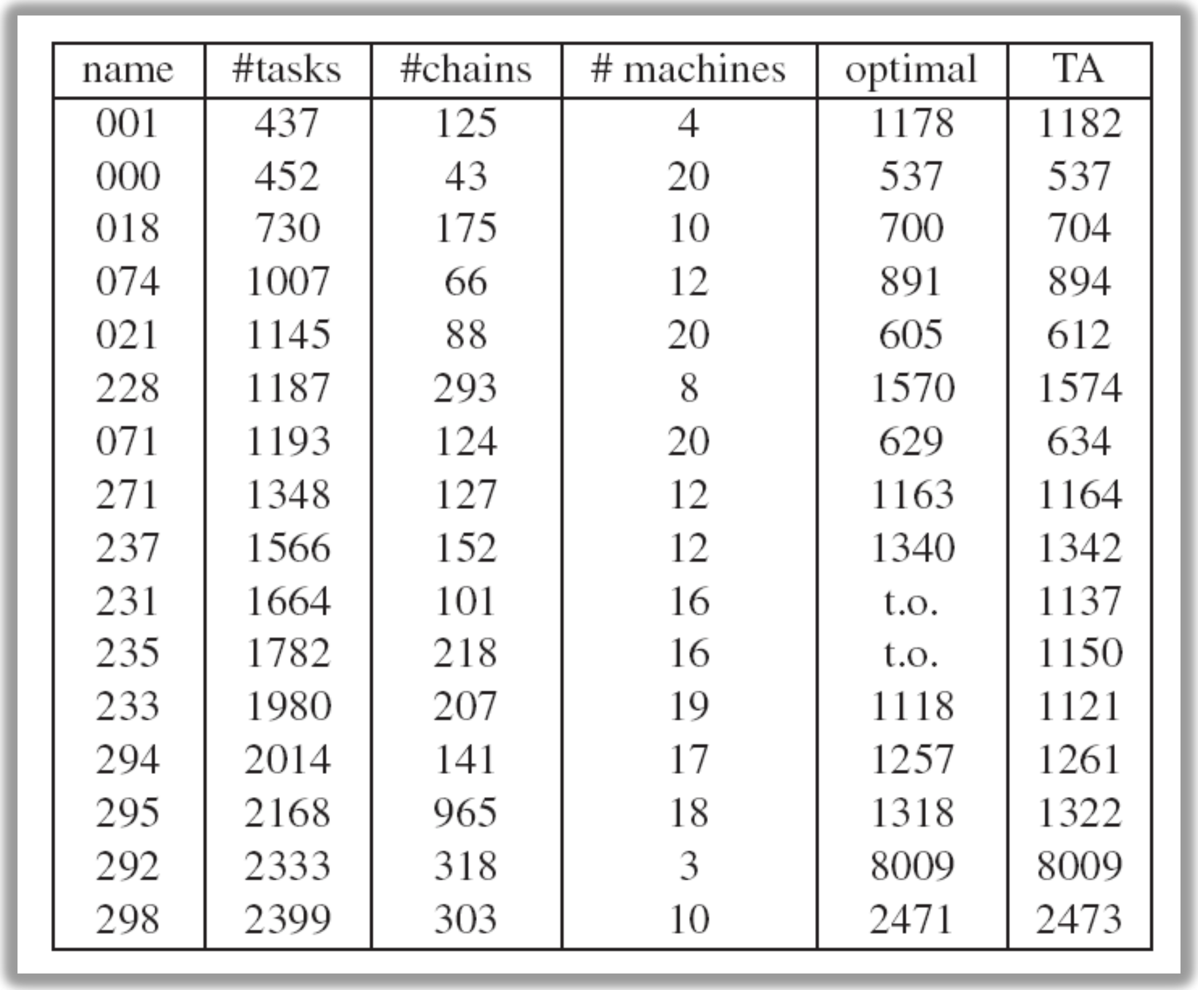


Task Graph Scheduling – Example



E<> (Task1.End and … and Task6.End)

Experimental Results



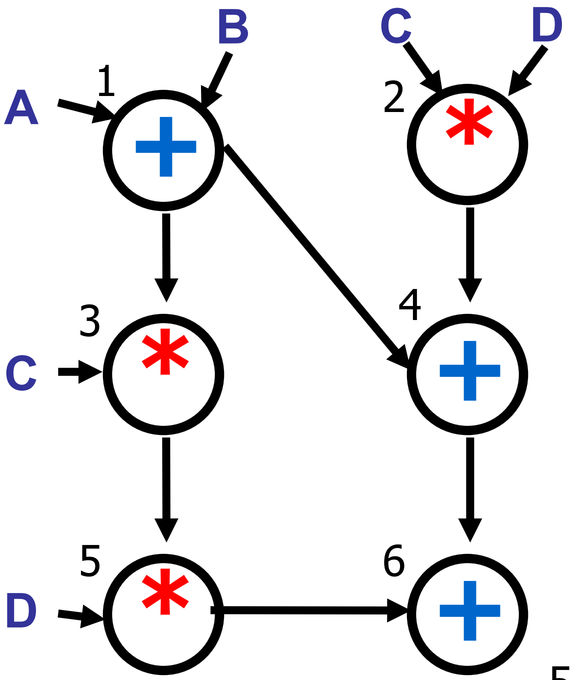
name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.	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

Task Graph Scheduling – Revisited



Compute :

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

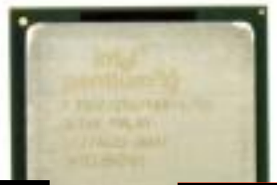
using 2 processors

P1 (fast)

+	2ps
*	3ps

Idle	10W
In use	90W

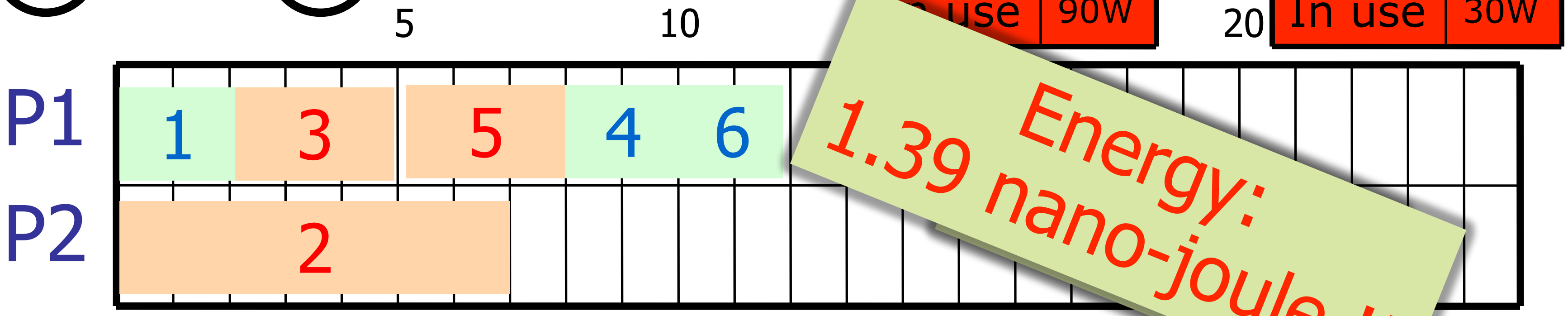
P2 (slow)



+	5ps
*	7ps

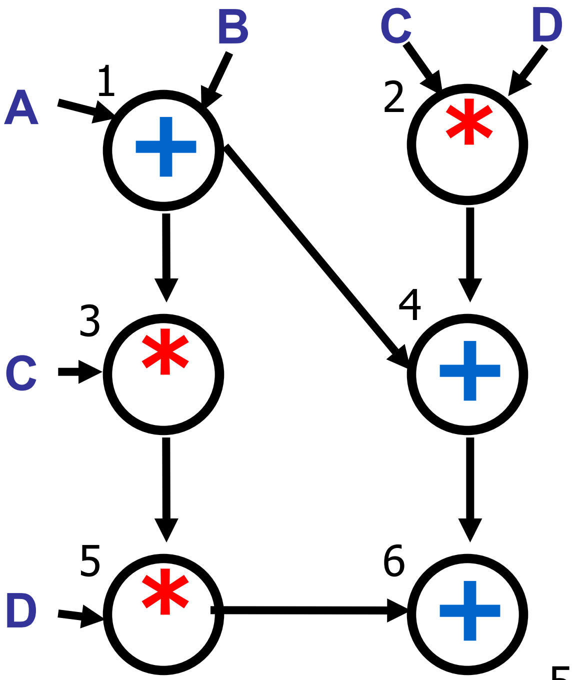
Idle	20W
In use	30W

ENERGY:



Energy:
1.39 nano-joule !!

Task Graph Scheduling – Revisited



Compute :

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

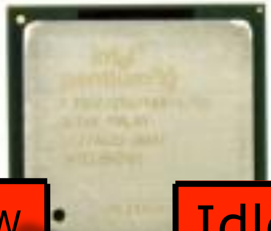
using 2 processors

P1 (fast)

+	2ps
*	3ps

Idle	10W
In use	90

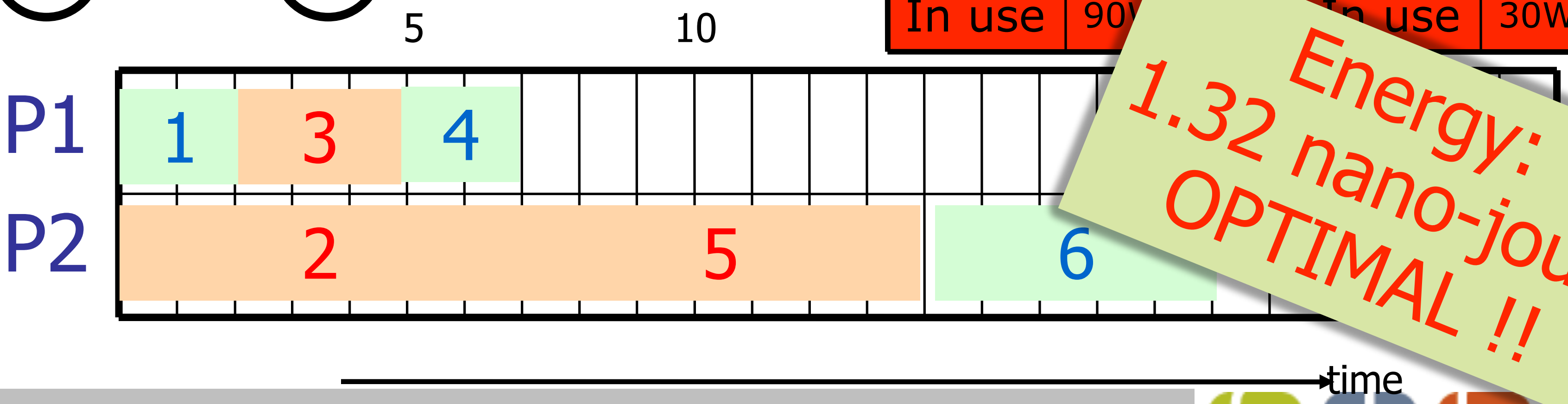
P2 (slow)



+	5ps
*	7ps

Idle	20W
In use	30W

ENERGY:

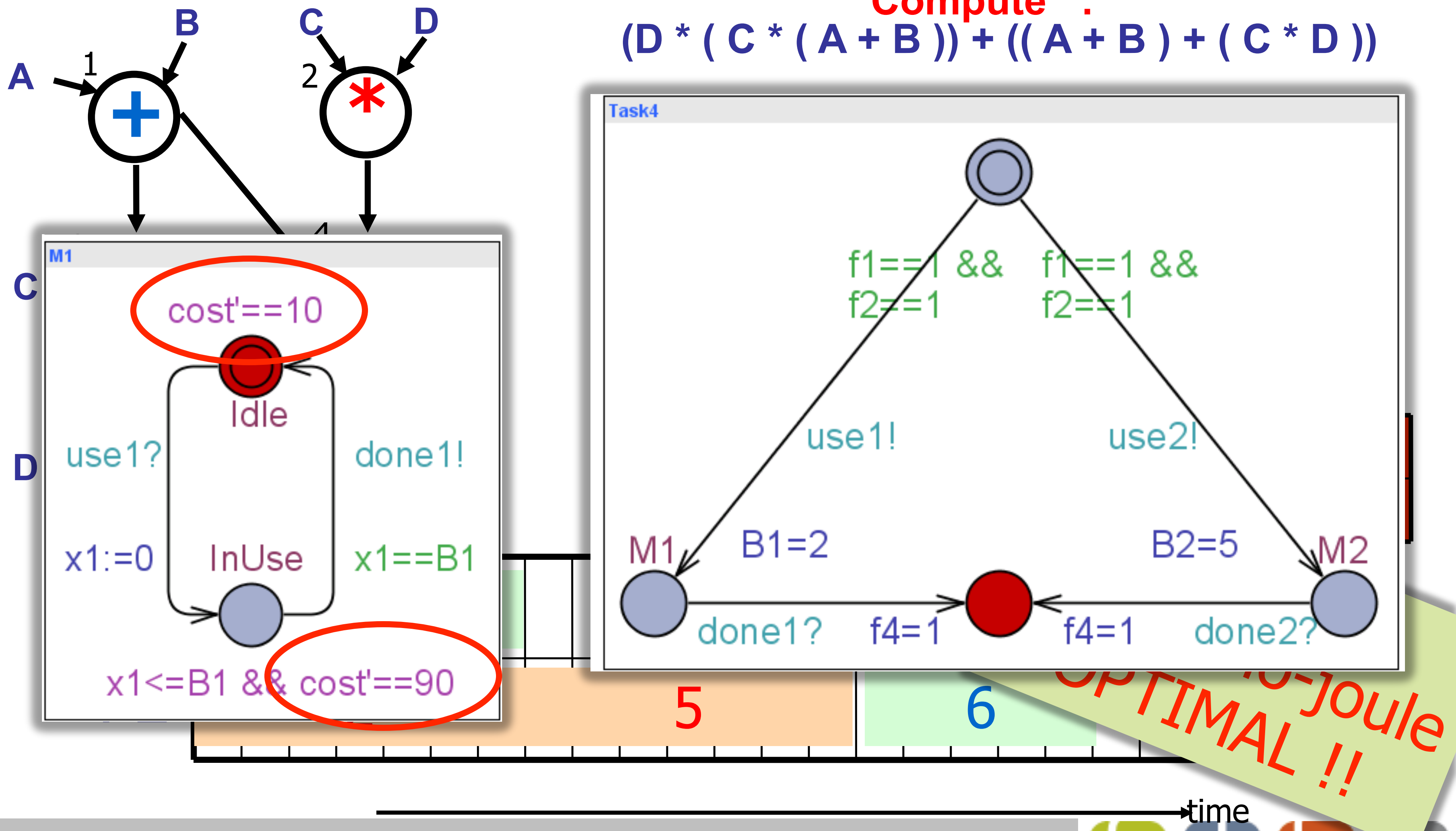


Energy:
1.32 nano-joule
OPTIMAL !!

Task Graph Scheduling – Revisited

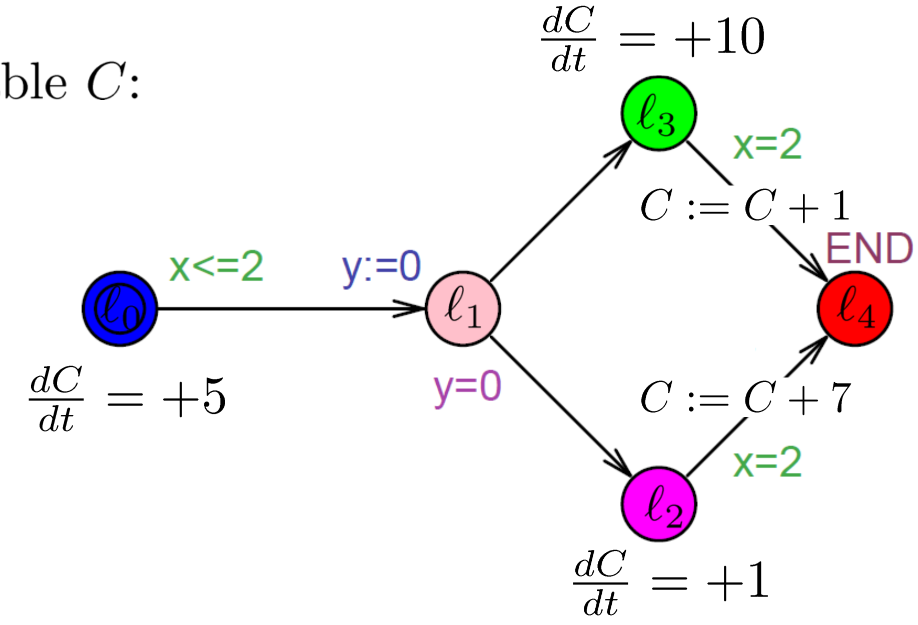
Compute :

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



A simple example

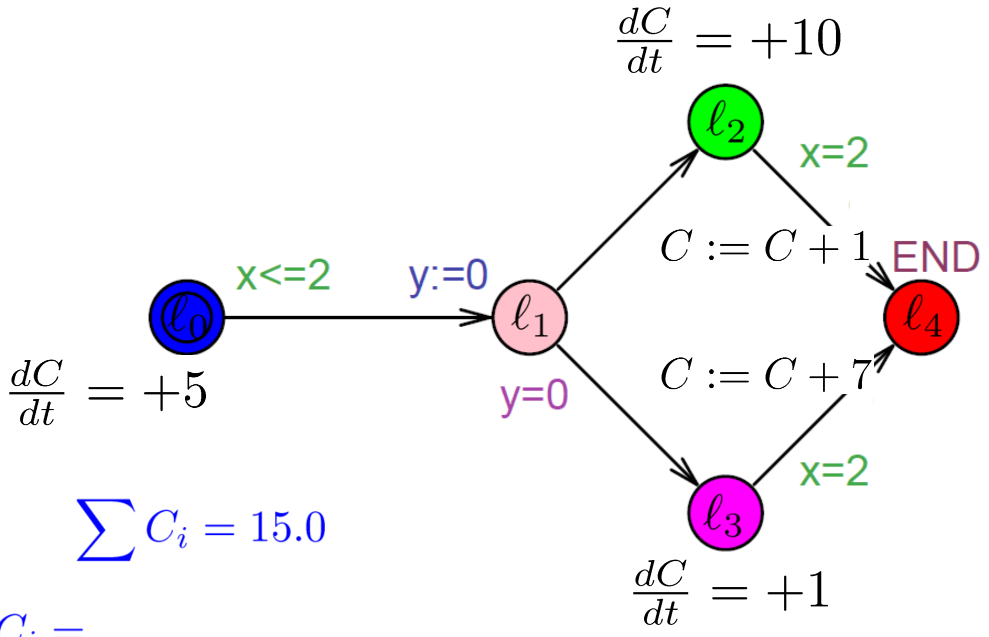
Observer variable C:



$$(\ell_0, [0,0]) \xrightarrow{1.9}_{9.5} (\ell_0, [1.9, 1.9]) \rightarrow_0 (\ell_1, [1.9, 0]) \rightarrow_0 (\ell_2, [1.9, 0]) \xrightarrow{0.1}_{0.1} (\ell_2, [2, 0.1]) \rightarrow_7 (\ell_4, [2, 0.1]) \qquad \sum C_i = 16.6$$

$$(\ell_0, [0,0]) \xrightarrow{1.2}_{6.0} (\ell_0, [1.2, 1.2]) \rightarrow_0 (\ell_1, [1.2, 0]) \rightarrow_0 (\ell_3, [1.2, 0]) \xrightarrow{0.8}_{8.0} (\ell_3, [2, 0.8]) \rightarrow_1 (\ell_4, [2, 0.8]) \qquad \sum C_i = 15.0$$

A simple example



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

$\sum C_i = ..$

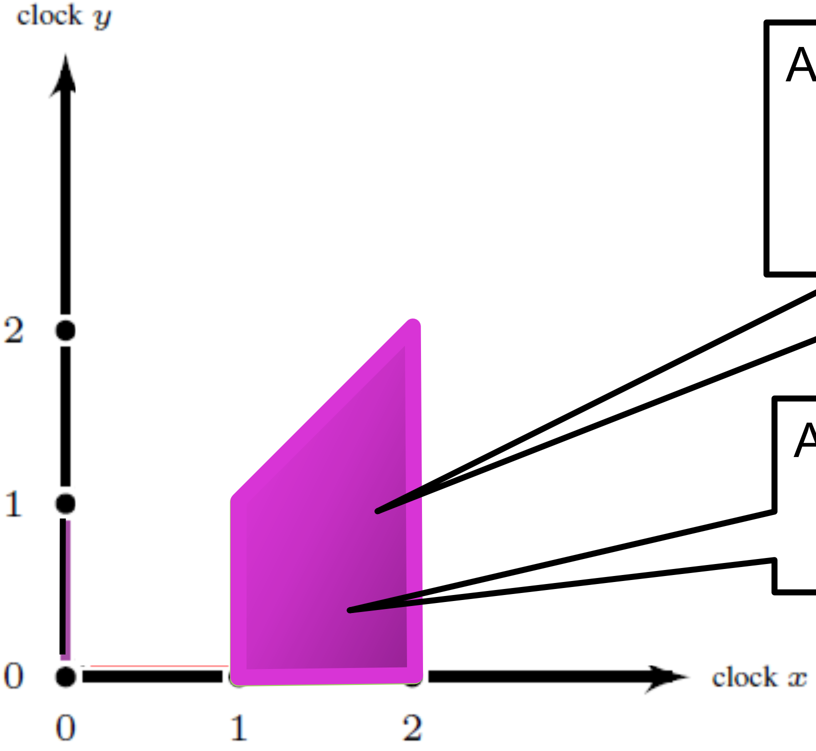
Q: What is cheapest cost for reaching ℓ_4 ?

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

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

Priced Zones

[CAV01]



A zone **Z**:

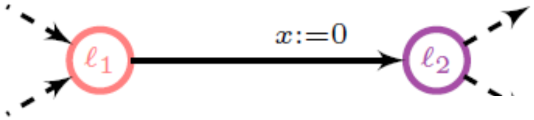
1· x · 2 Æ
0· y · 2 Æ
x - y ¸ 0

A cost function **C**

C(x,y)=
2¢x - 1¢y + 3

Priced Zones – Reset

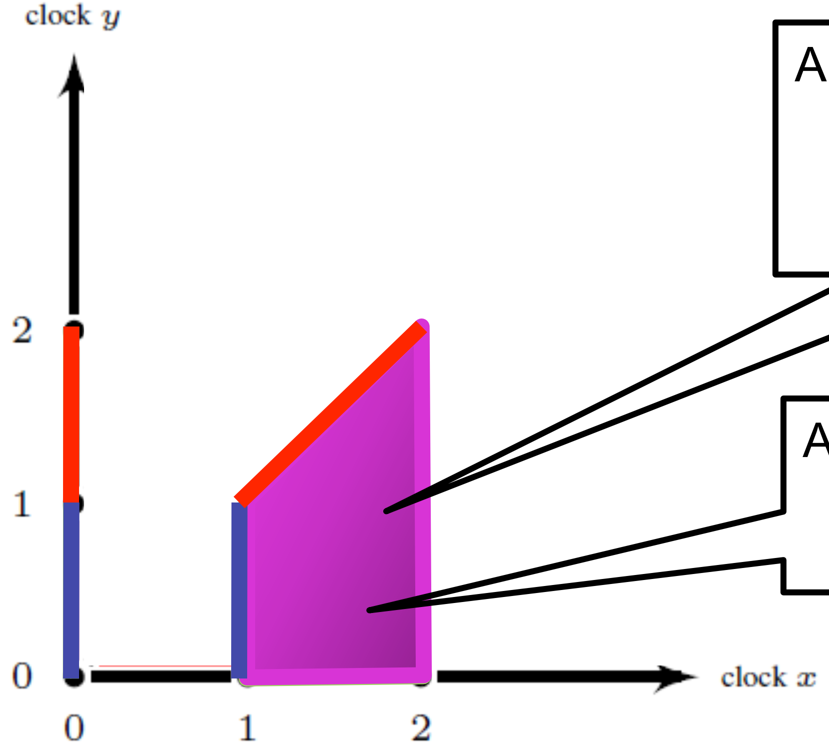
[CAV01]



Z[x=0]:
x=0 Æ
0· y · 2

C = 1¢y + 3

C= -1¢y + 5



A zone **Z**:
1· x · 2   Æ
0· y · 2   Æ
x - y ¸ 0

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

Symbolic Branch & Bound Algorithm

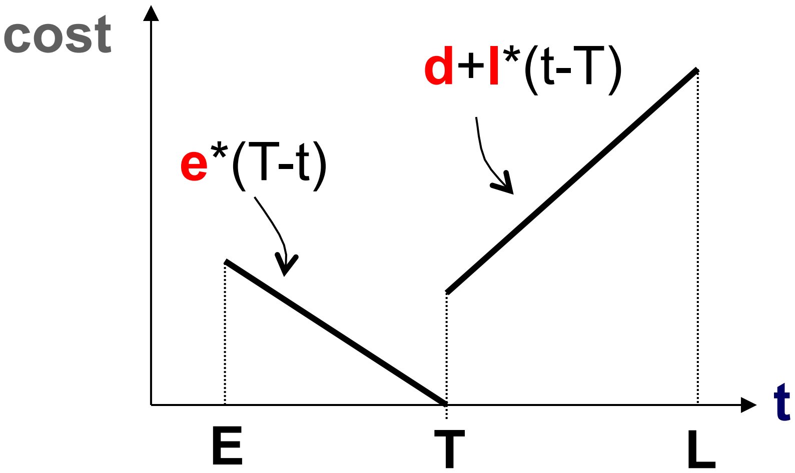
```
Cost := $\infty$
Passed := $\emptyset$
Waiting := $\{(l_0, Z_0)\}$
**while** Waiting $\neq \emptyset$ **do**
    **select** $(l, Z)$ from Waiting
    **if** $l = l_g$ and minCost$(Z)$ < Cost **then**
        Cost := minCost$(Z)$
    **if** minCost$(Z)$ + Rem$_{(l,Z)} \geq$ Co the
    **if** for all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

$Z' \leq Z$

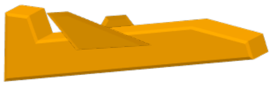
Z′* is bigger & cheaper than *Z

· is a well-quasi ordering which guarantees termination!

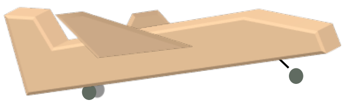
Example: Aircraft Landing

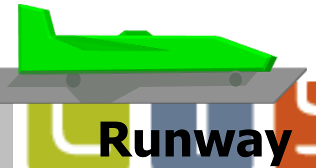


E	earliest landing time
T	target time
L	latest time
e	cost rate for being early
l	cost rate for being late
d	fixed cost for being late

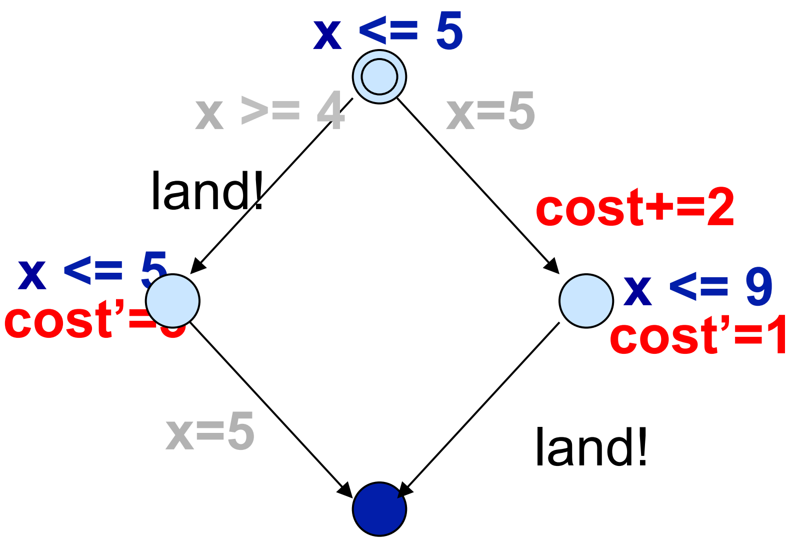


Planes have to keep separation distance to avoid turbulences caused by preceding planes

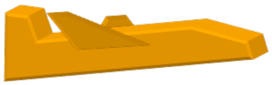




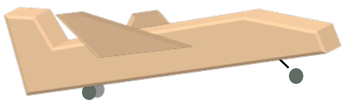
Example: Aircraft Landing

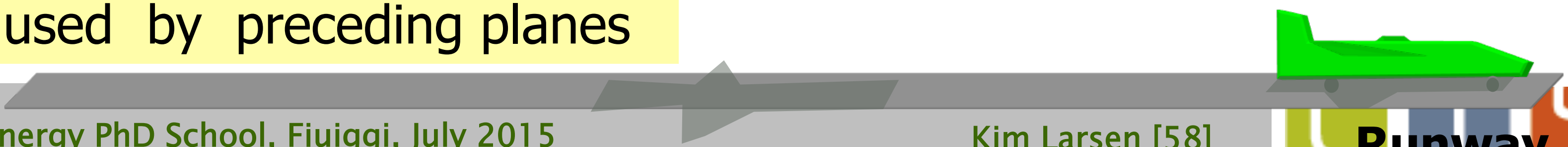


4 earliest landing time
5 target time
9 latest time
3 cost rate for being early
1 cost rate for being late
2 fixed cost for being late



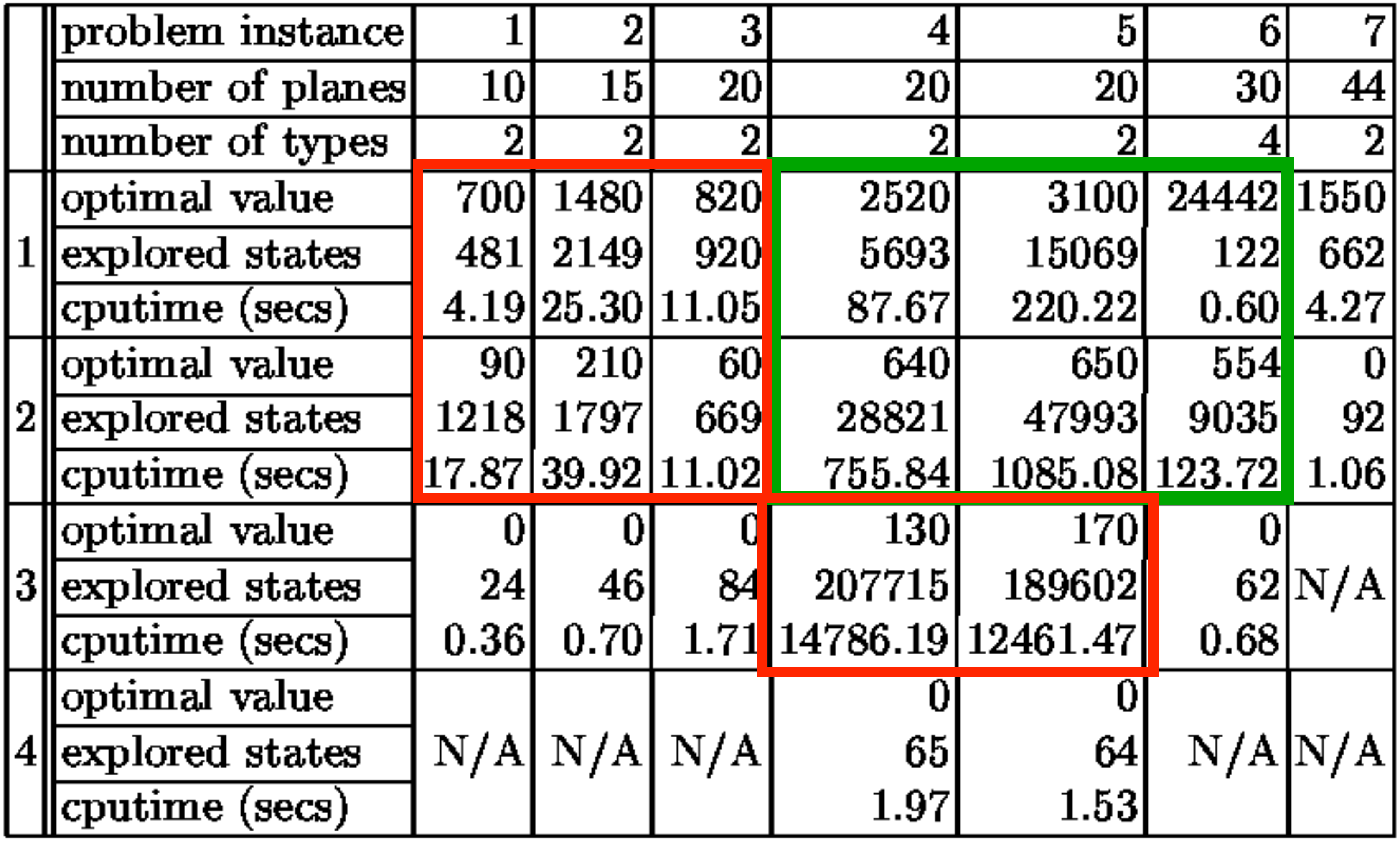
Planes have to keep separation distance to avoid turbulences caused by preceding planes





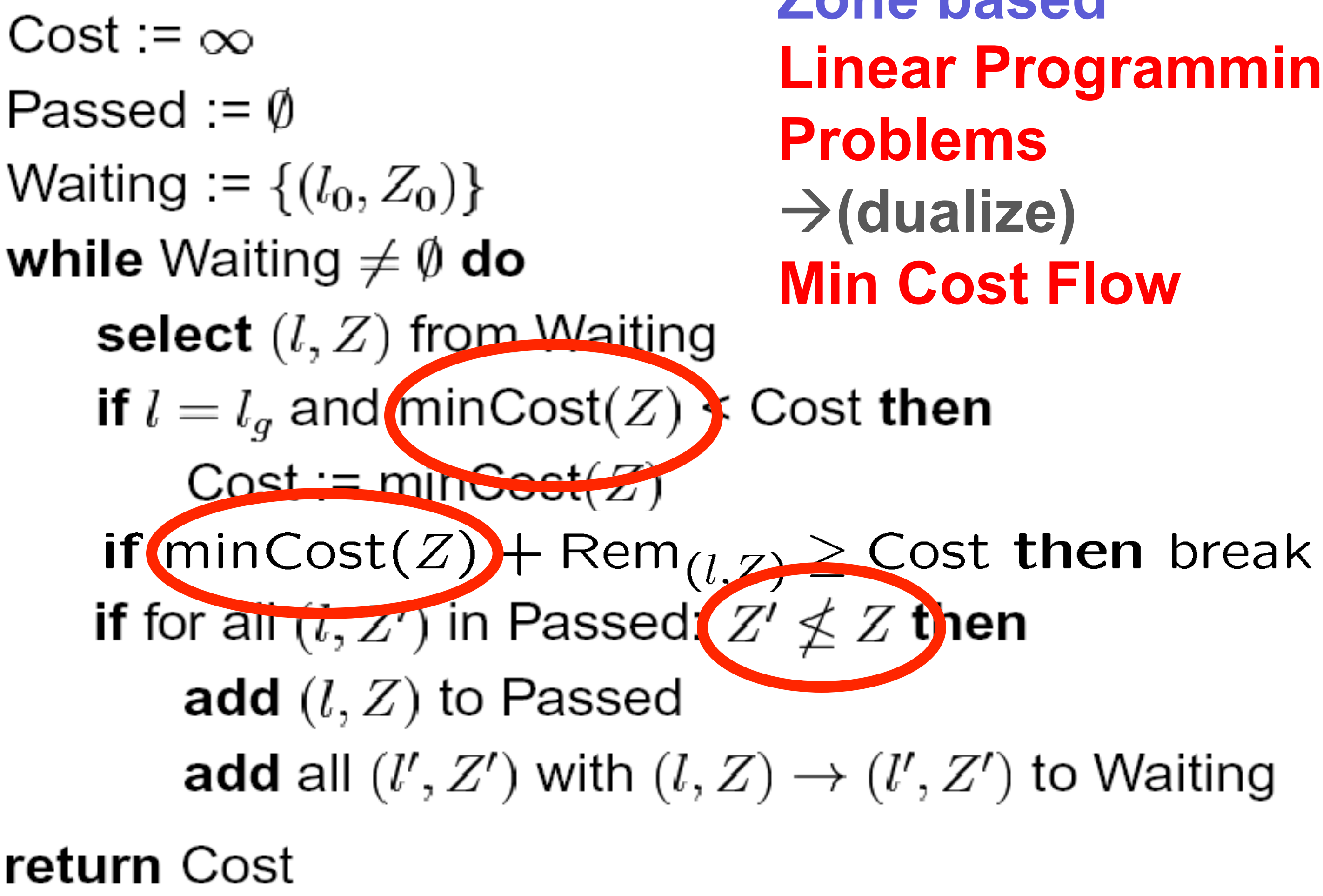
Aircraft Landing

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		

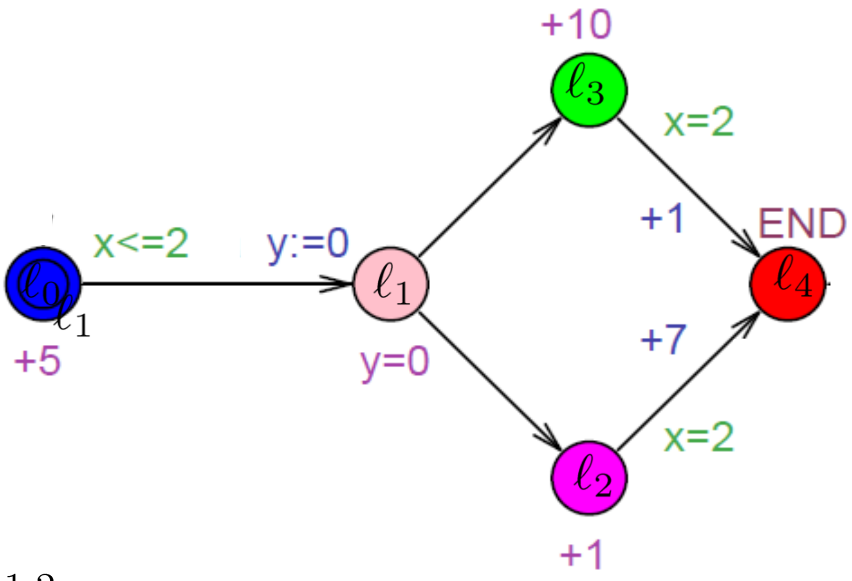
Symbolic Branch & Bound Algorithm



```
Cost := $\infty$
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 $(l, Z) \rightarrow (l', Z')$ to Waiting
return Cost
```

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

Optimal Schedule



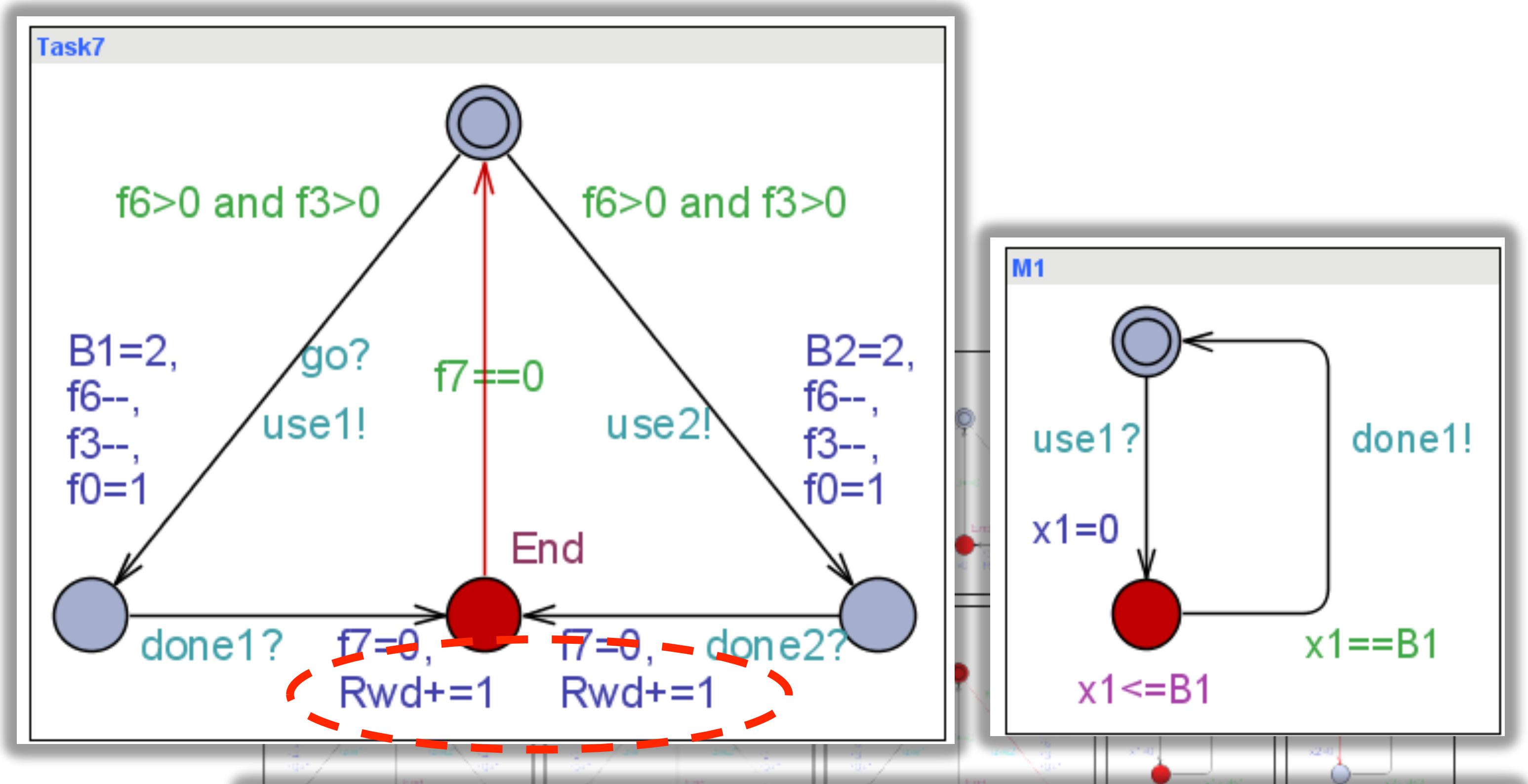
$$(\ell_0,[0,0]) \xrightarrow{1.2}_{6.0} (\ell_0,[1.2,1.2]) \rightarrow_{0} (\ell_1,[1.2,0]) \rightarrow_{0}$$

$$(\ell_3,[1.2,0]) \xrightarrow{0.8}_{8.0} (\ell_3,[2,0.8]) \rightarrow_{1} (\ell_4,[2,0.8])$$

$$\rightarrow_{2.0} (\ell_0,[0,0])$$

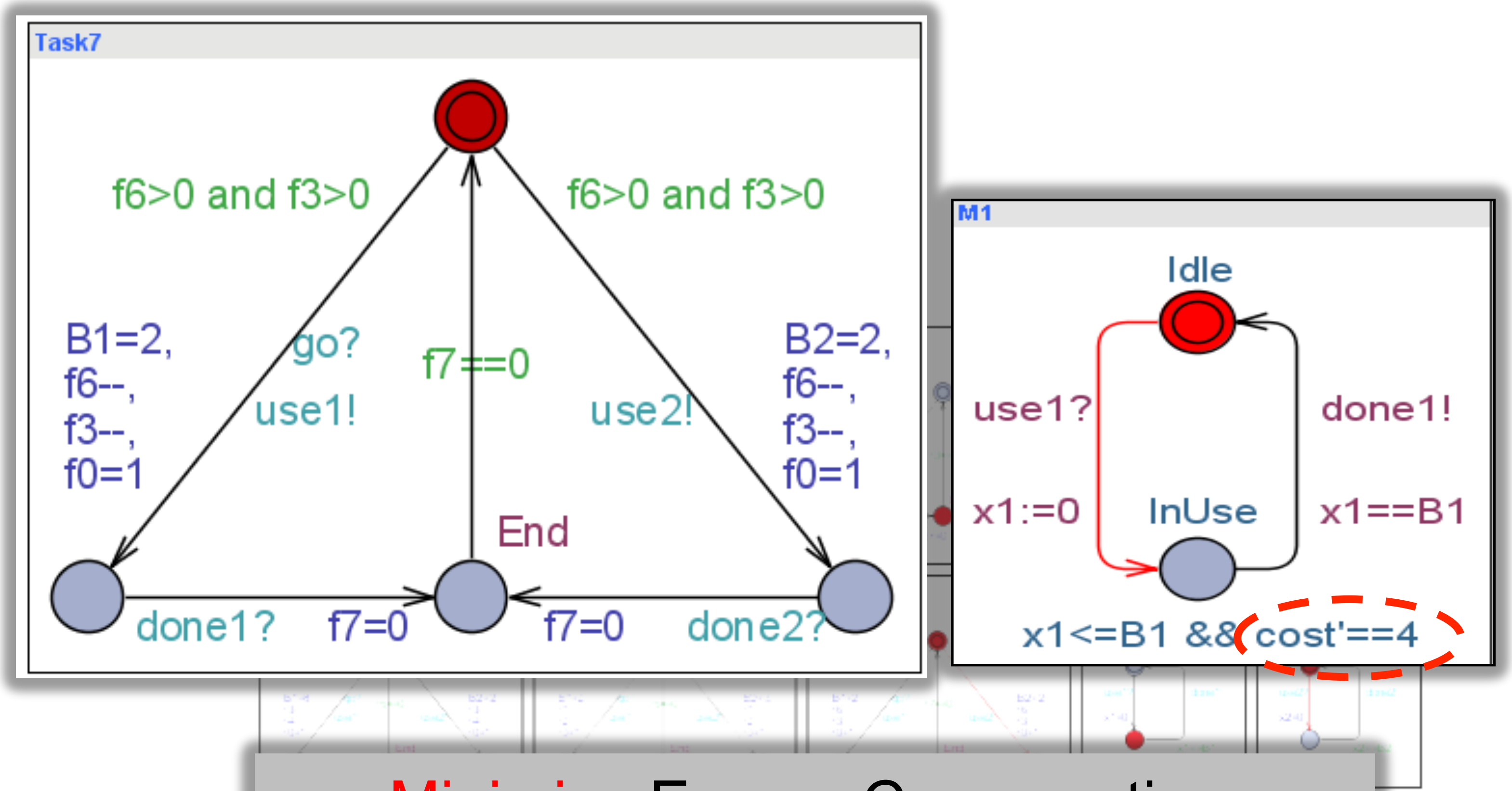
$$\boxed{\sum_i C_i / \sum_i t_i = 17/2 = 8.5}$$

Optimal Infinite Scheduling



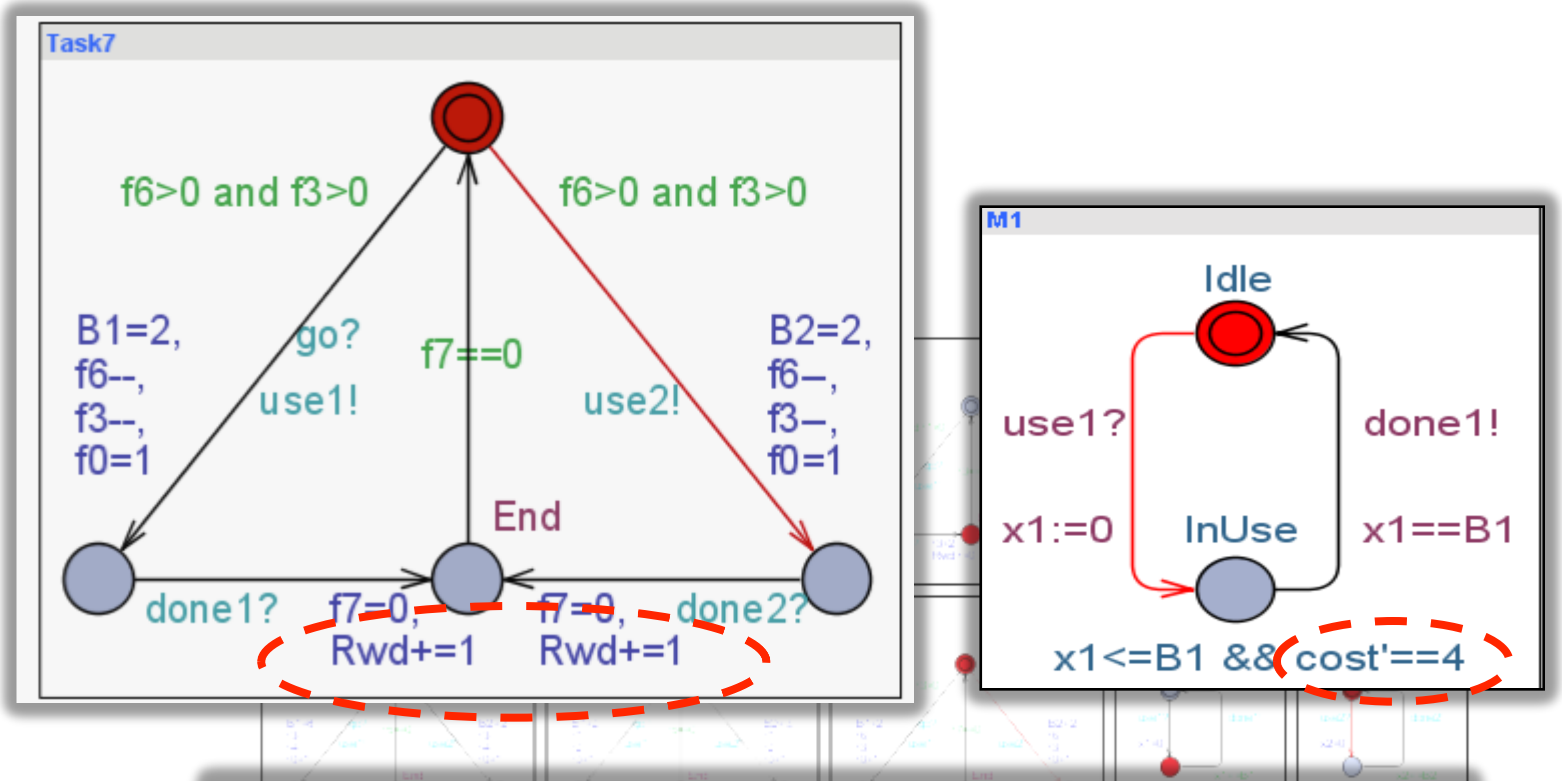
Maximize throughput:
i.e. maximize Reward / Time in the long run!

Optimal Infinite Scheduling



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

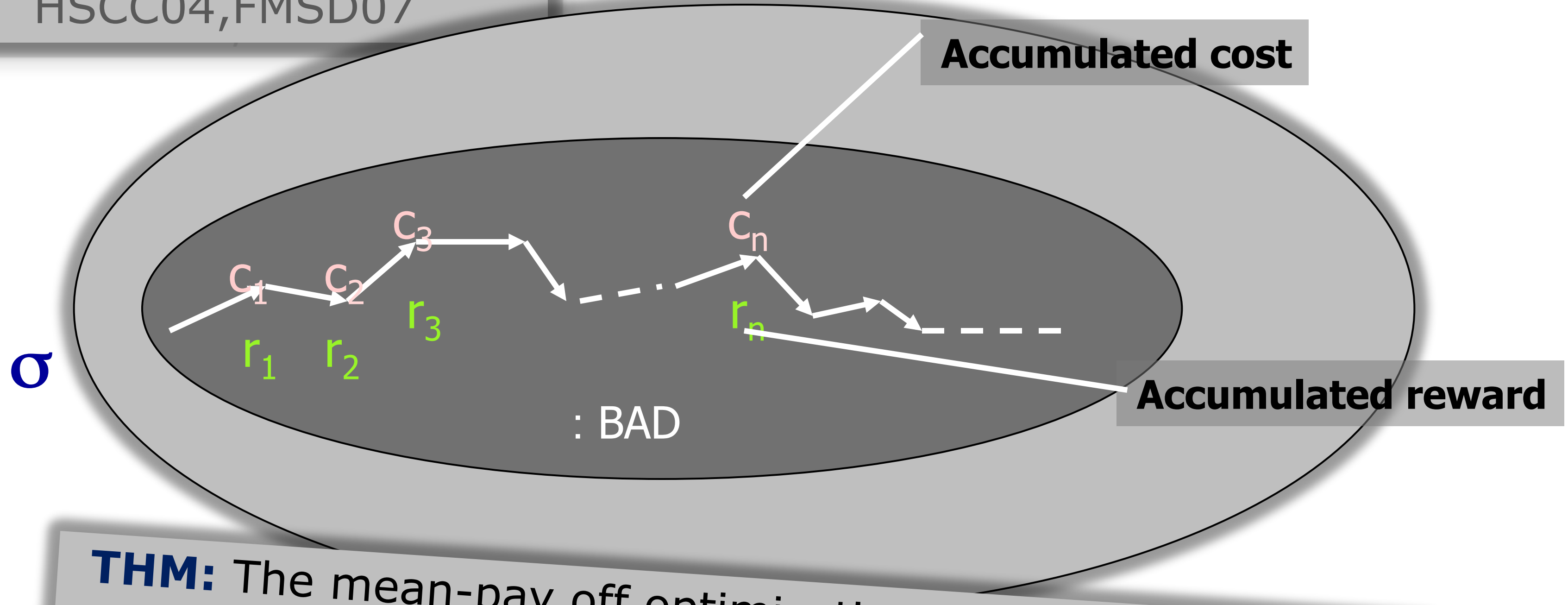
Optimal Infinite Scheduling



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

Mean Pay-Off Optimality

Bouyer, Brinksma, Larsen:
HSCC04,FMSD07



σ

c_1 c_2 c_3 c_n

r_1 r_2 r_3 r_n

: BAD

Accumulated cost

Accumulated reward

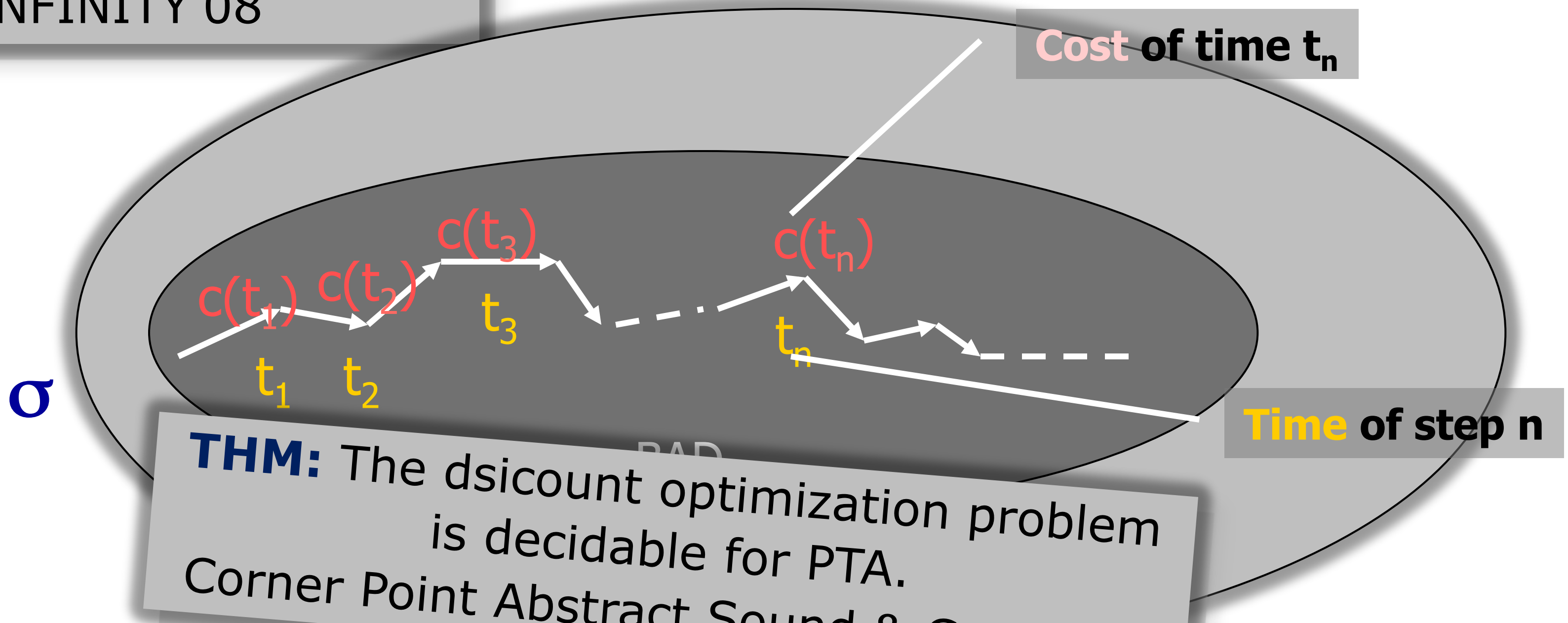
THM: The mean-pay off optimization problem is decidable (and PSPACE-complete) for PTA.
Corner Point Abstract Sound & Complete

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

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: $\mathrm{val}(\sigma) = \int_{t=0}^{t=\infty} c(t)\lambda^t dt$

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

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:

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

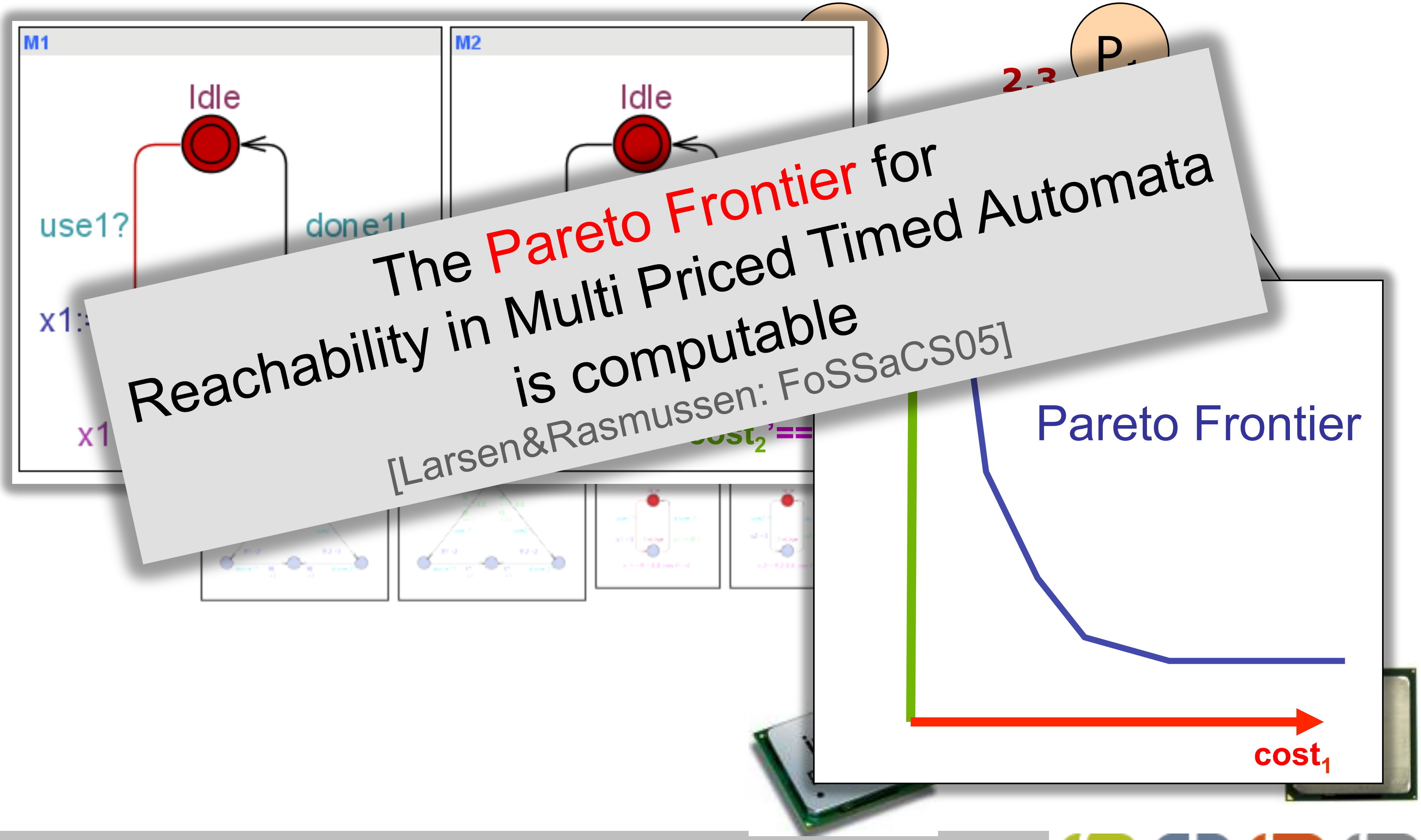
then $\inf_Z f$ is obtained at a corner-point of Z (with integer coefficients).

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).

Multiple Objective Scheduling



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

[Larsen&Rasmussen: FoSSaCS05]

Overview

- Timed Automata / UPPAAL
  - Verification
- Priced Timed Automata / UPPAAL CORA
  - Optimal Scheduling (multicore applications)
  - Optimal Infinite Scheduling
  - Multi objective optimization
- Schedulability Analysis & Scheduling
  - Single Core, Multi Core
  - Dynamic voltage Scheduling
  - Energy Automata
- Stochastic Priced Timed Automata / UPPAAL SMC
  - Statistical Model Checking
  - Low Power Medium Access Protocol
  - Stochastic Hybrid Automata
  - Energy-Aware Buildings
  - Battery-Aware Scheduling
- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synteses
  - Energy-Aware and Optimal Satelitte Scheduling
- Conclusion

Schedulability & Performance Analysis

AALBORG UNIVERSITY DENMARK

LISS

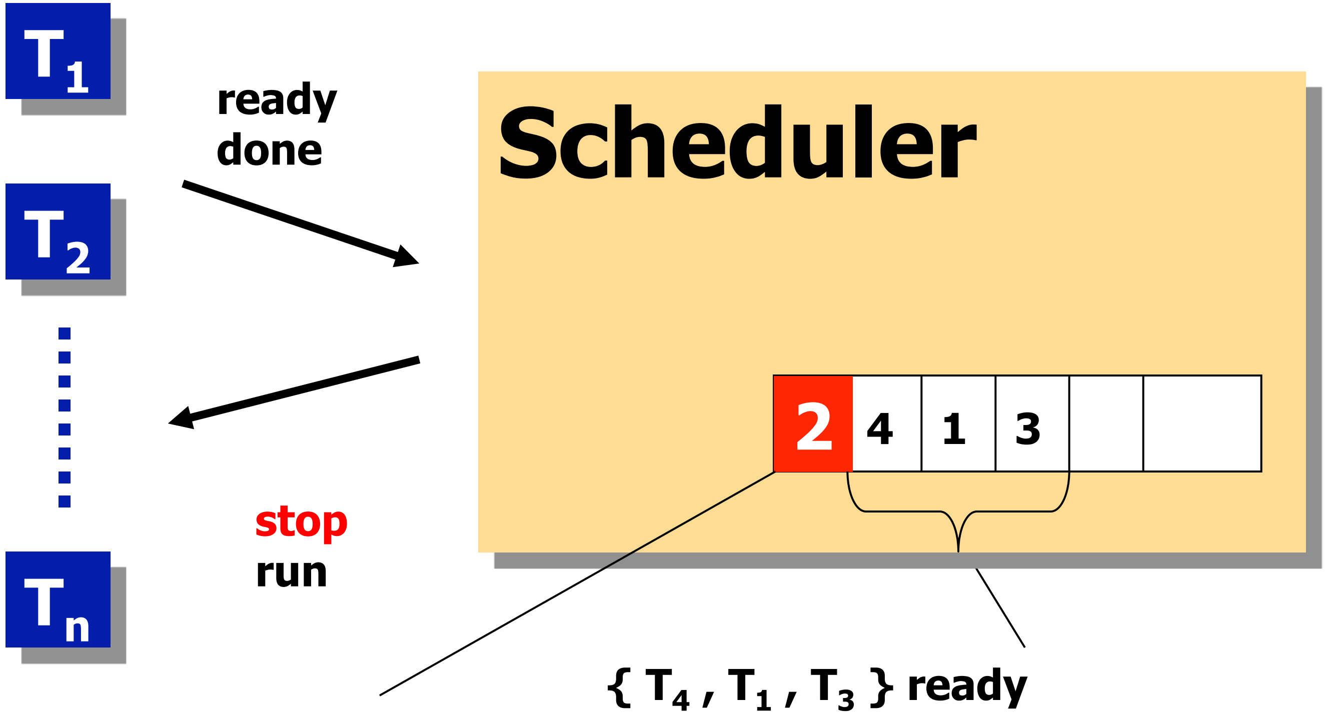
Task Scheduling

utilization of CPU

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

C(i), UNI[BC(i),WC(i)] : execution time for T_i

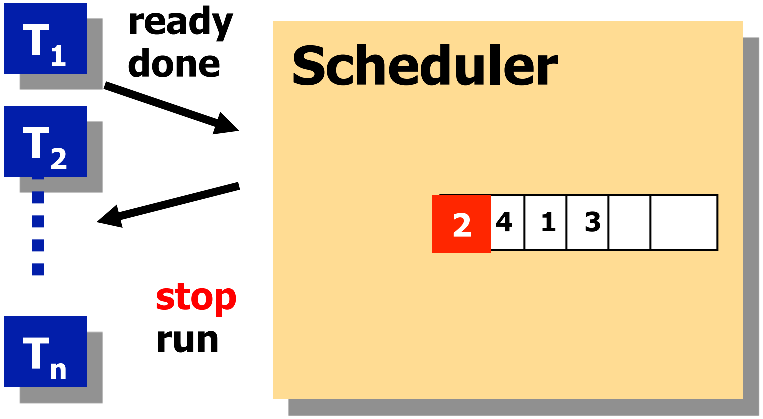
D(i): deadline for T_i

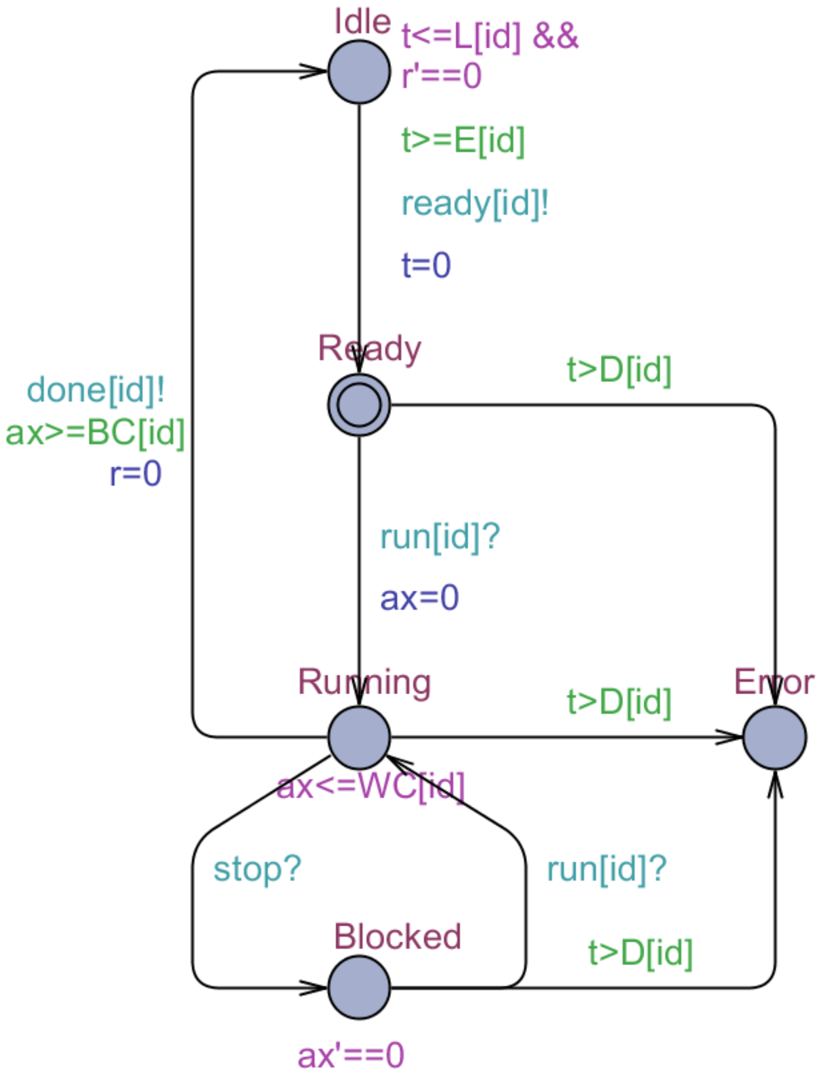


T_2 is running

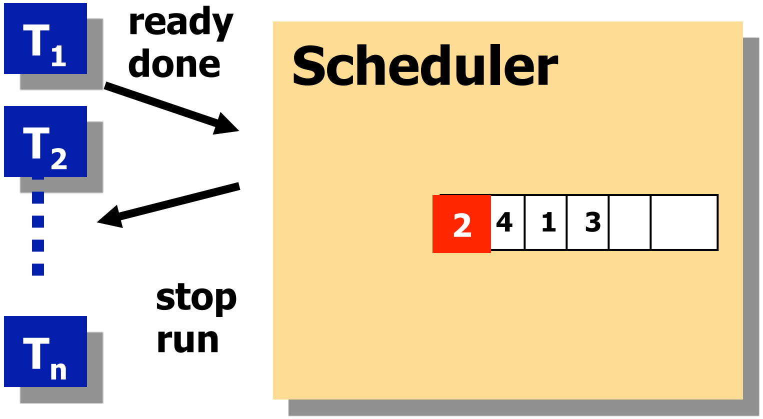
{ T_4 , T_1 , T_3 } ready
ordered according to some
given priority:
(e.g. Fixed Priority, Earliest Deadline,..)

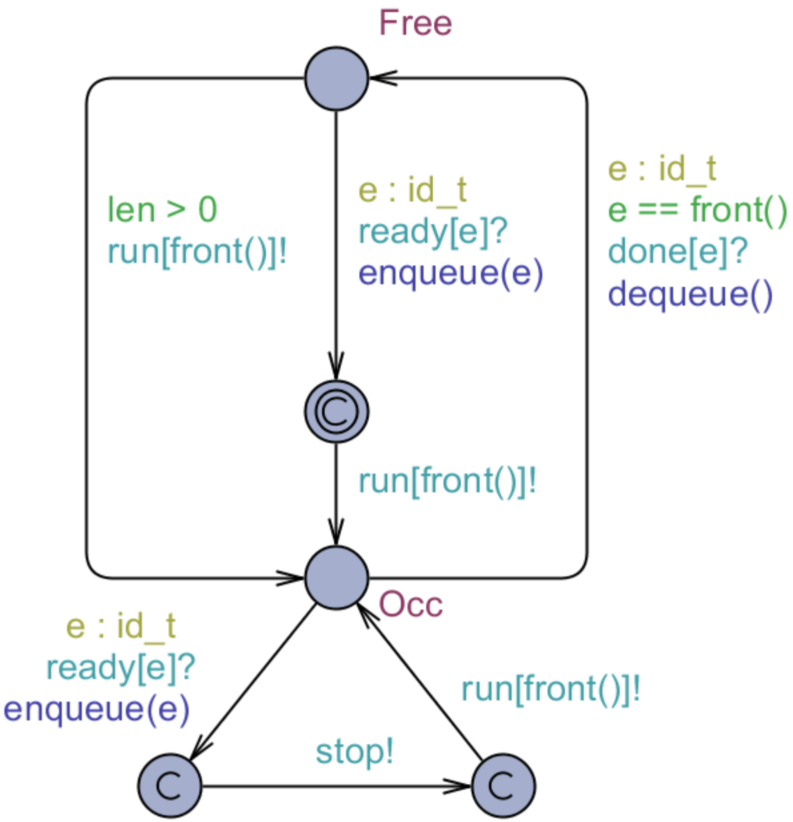
Modeling Task



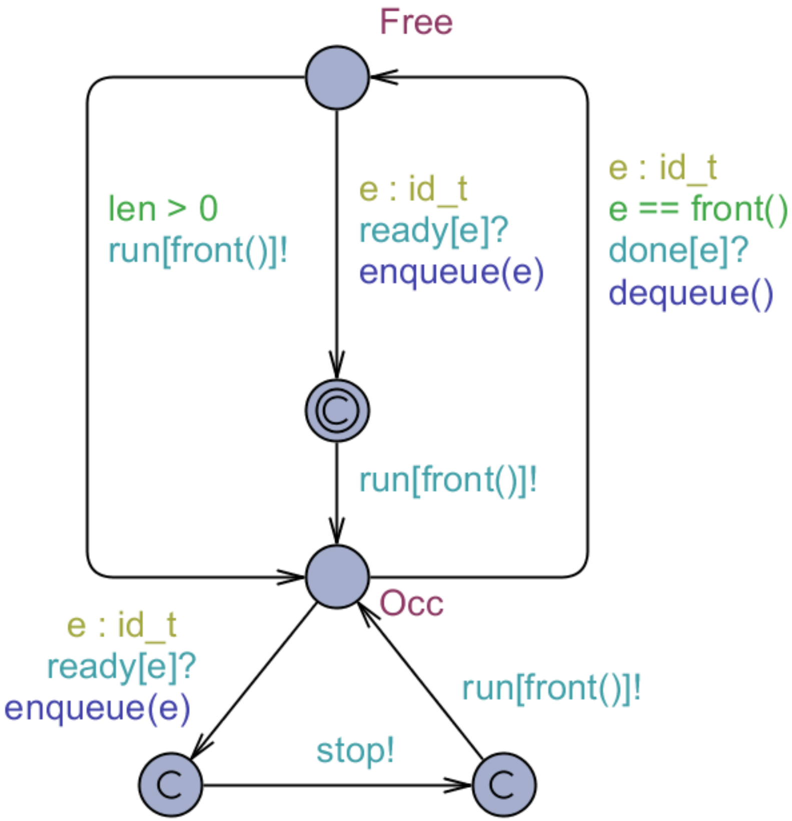


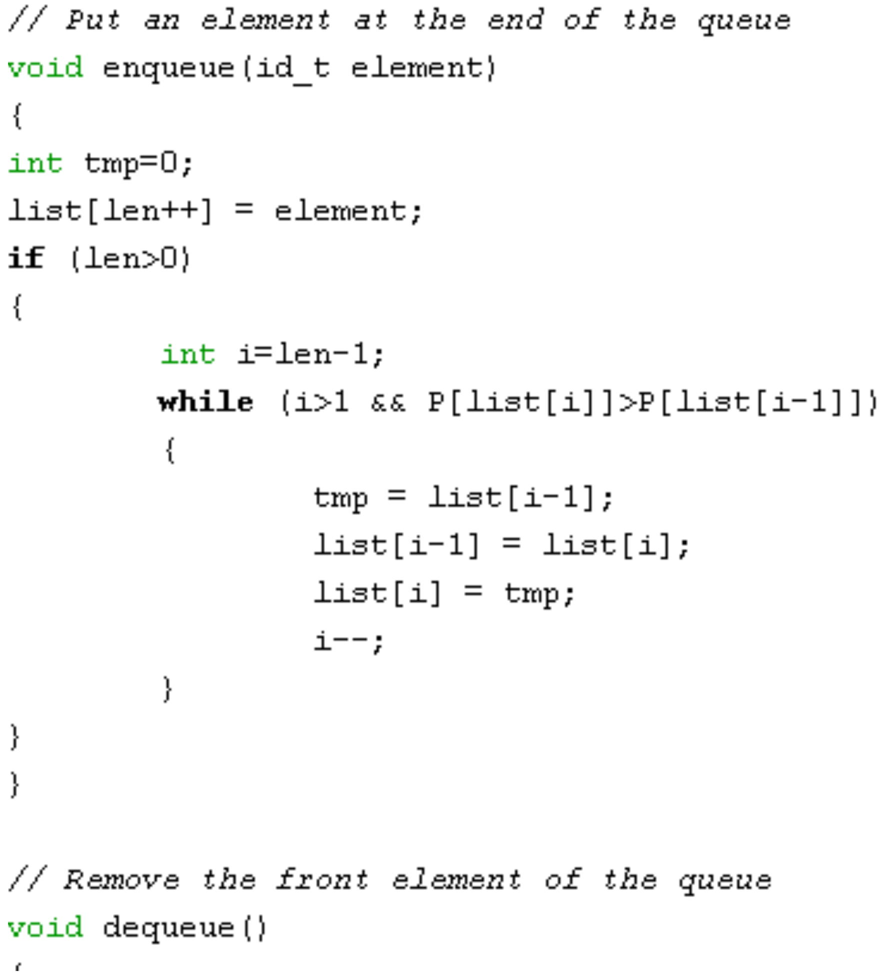
Modeling Scheduler





Modeling Queue



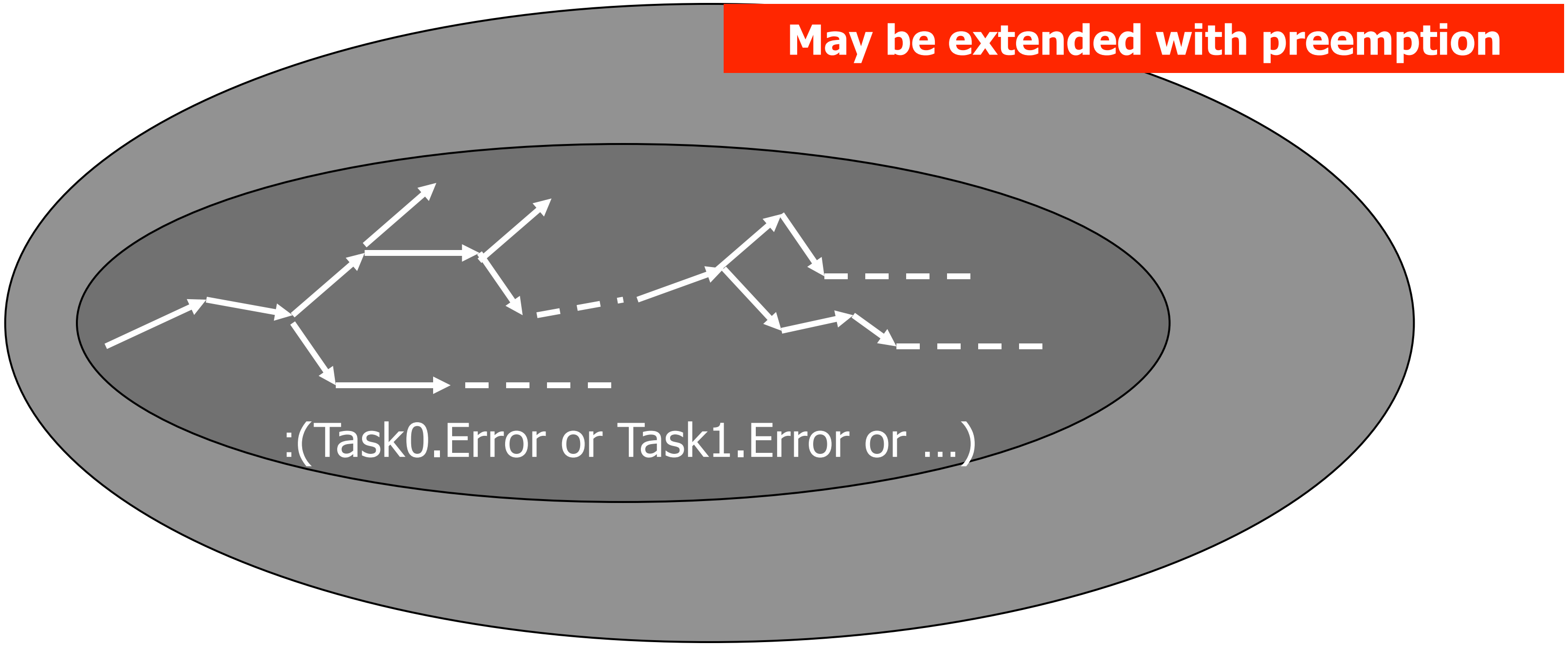


```
// Put an element at the end of the queue
void enqueue(id_t element)
{
int tmp=0;
list[len++] = element;
if (len>0)
{
        int i=len-1;
        while (i>1 && P[list[i]]>P[list[i-1]])
        {
                tmp = list[i-1];
                list[i-1] = list[i];
                list[i] = tmp;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

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

......

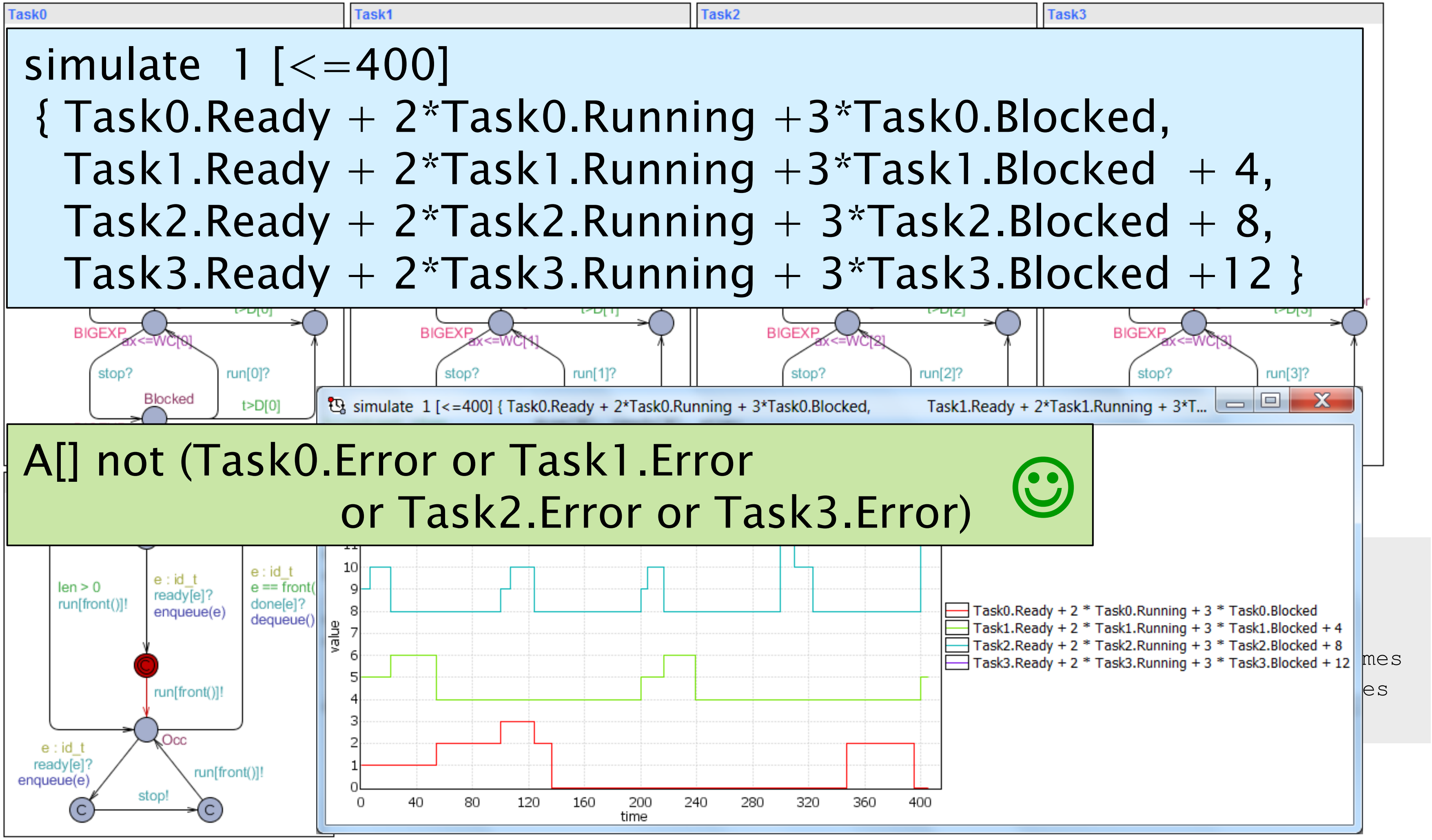
Schedulability = Safety Property



May be extended with preemption

A :(Task0.Error or Task1.Error or …)

Schedulability Analysis

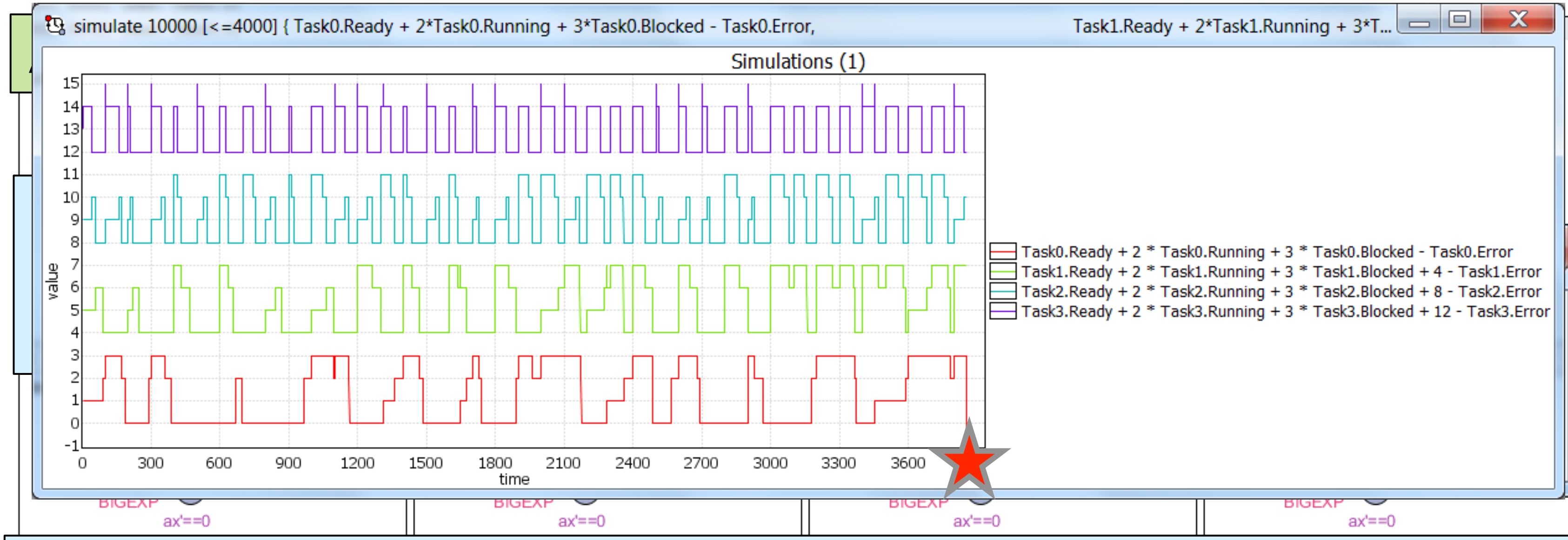


simulate 1 [<=400] { Task0.Ready + 2*Task0.Running + 3*Task0.Blocked, Task1.Ready + 2*Task1.Running + 3*T...

simulate 1 [<=400]
{ Task0.Ready + 2*Task0.Running +3*Task0.Blocked,
Task1.Ready + 2*Task1.Running +3*Task1.Blocked + 4,
Task2.Ready + 2*Task2.Running + 3*Task2.Blocked + 8,
Task3.Ready + 2*Task3.Running + 3*Task3.Blocked +12 }

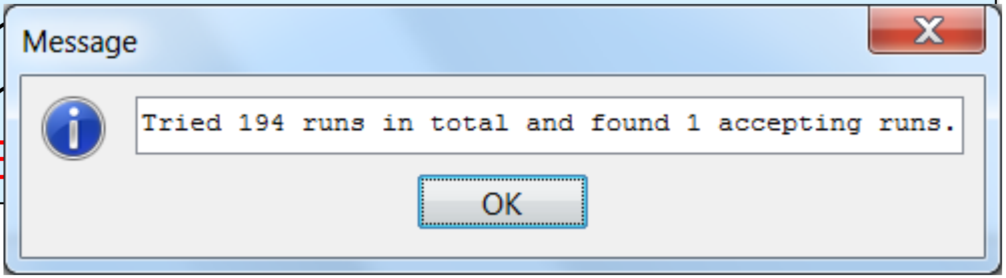
A[] not (Task0.Error or Task1.Error
or Task2.Error or Task3.Error) ☺

Schedulability Analysis



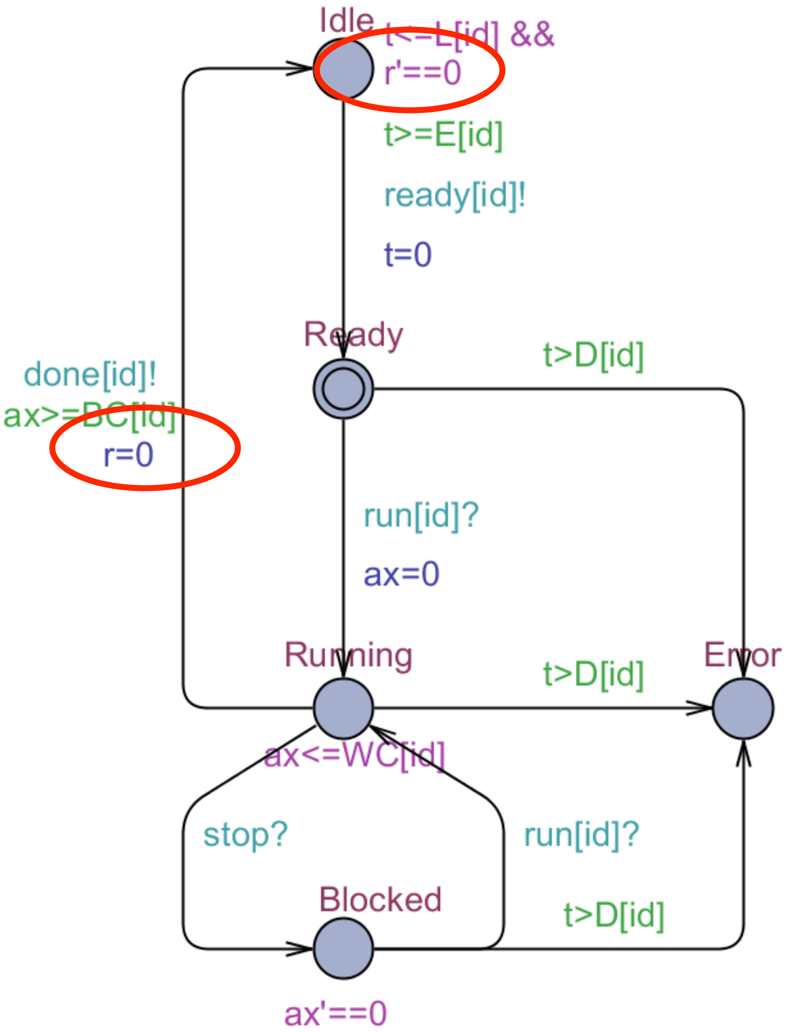
simulate 10000 [<=4000] { Task0.Ready + 2*Task0.Running + 3*Task0.Blocked - Task0.Error, Task1.Ready + 2*Task1.Running + 3*T...

```
simulate  10000 [<=400]
 { Task0.Ready + 2*Task0.Running +3*Task0.Blocked,
   Task1.Ready + 2*Task1.Running +3*Task1.Blocked  + 4,
   Task2.Ready + 2*Task2.Rur
   Task3.Ready + 2*Task3.Rur
   : 1 : (Task0.Error or Task1.E
```

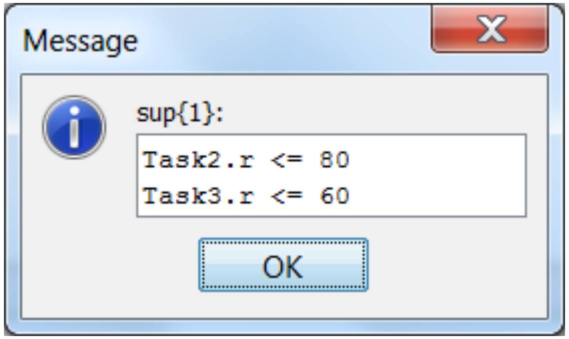


Tried 194 runs in total and found 1 accepting runs.

Performance Analysis



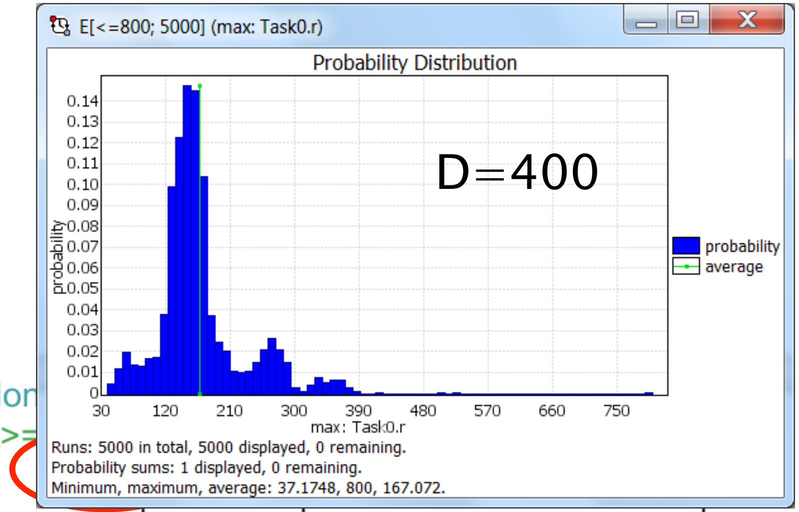
sup : Task2.r, Task3.r



```
sup{1}:
Task2.r <= 80
Task3.r <= 60
```

Performance Analysis

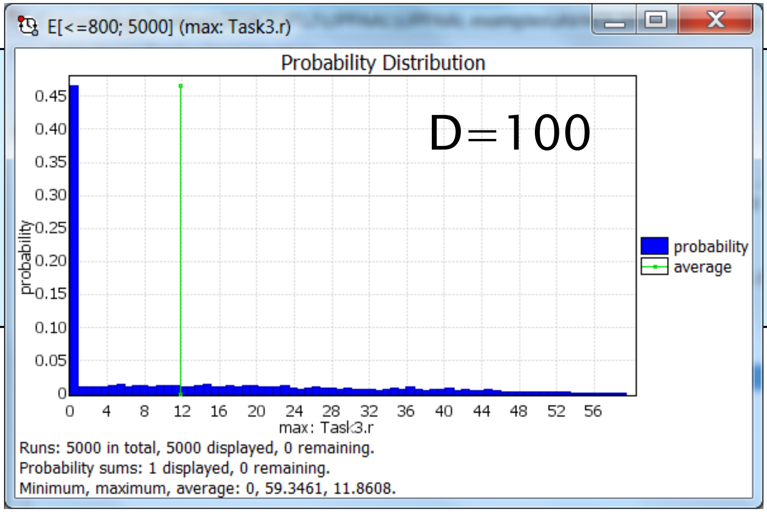
don
ax>=



D=400

```
Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 37.1748, 800, 167.072.
```

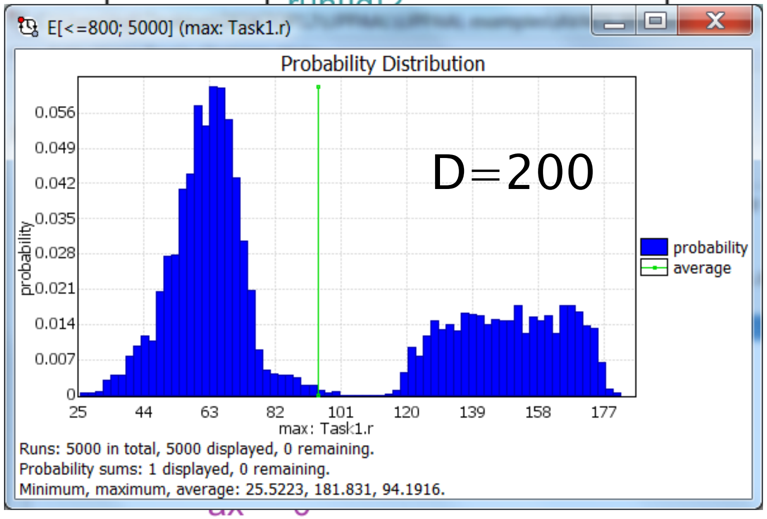
E[
E[
E[
E[



D=100

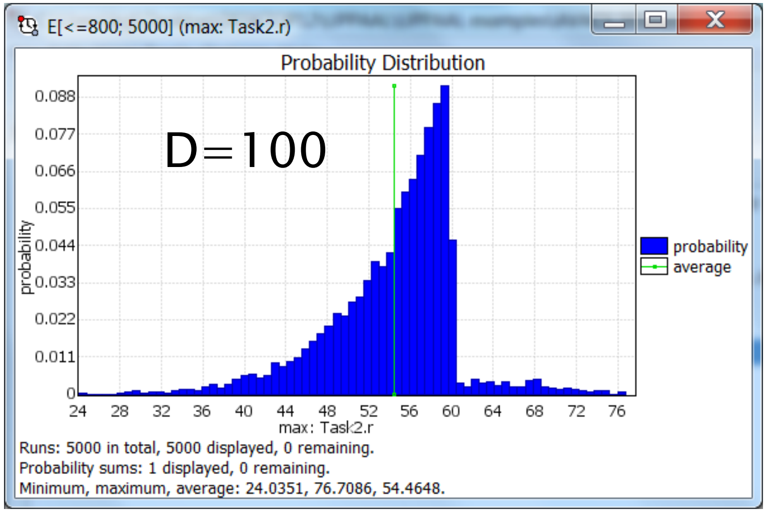
```
Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 0, 59.3461, 11.8608.
```

run[id]?



D=200

```
Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 25.5223, 181.831, 94.1916.
```



D=100

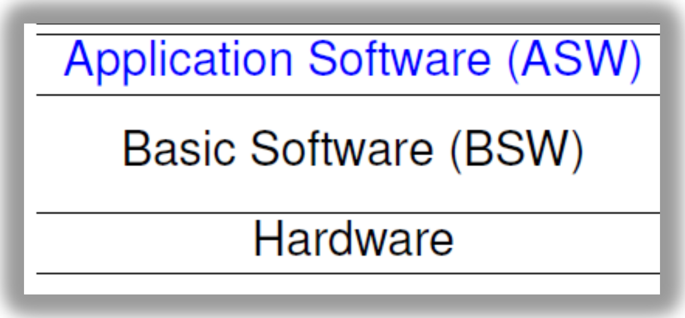
```
Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 24.0351, 76.7086, 54.4648.
```

Herschel-Planck Scientific Mission at ESA

Attitude and Orbit Control Software
TERMA A/S Steen Ulrik Palm, Jan Storbank Pedersen, Poul Hougaard

Herschel & Planck Satelites

- Application software (ASW)
  - built and tested by Terma:
  - does attitude and orbit control, tele-commanding, fault detection isolation and recovery.
- Basic software (BSW)
  - low level communication and scheduling periodic events.
- Real-time operating system (RTEMS)
  - Priority Ceiling for ASW,
  - Priority Inheritance for BSW
- Hardware
  - single processor, a few buses, sensors and act



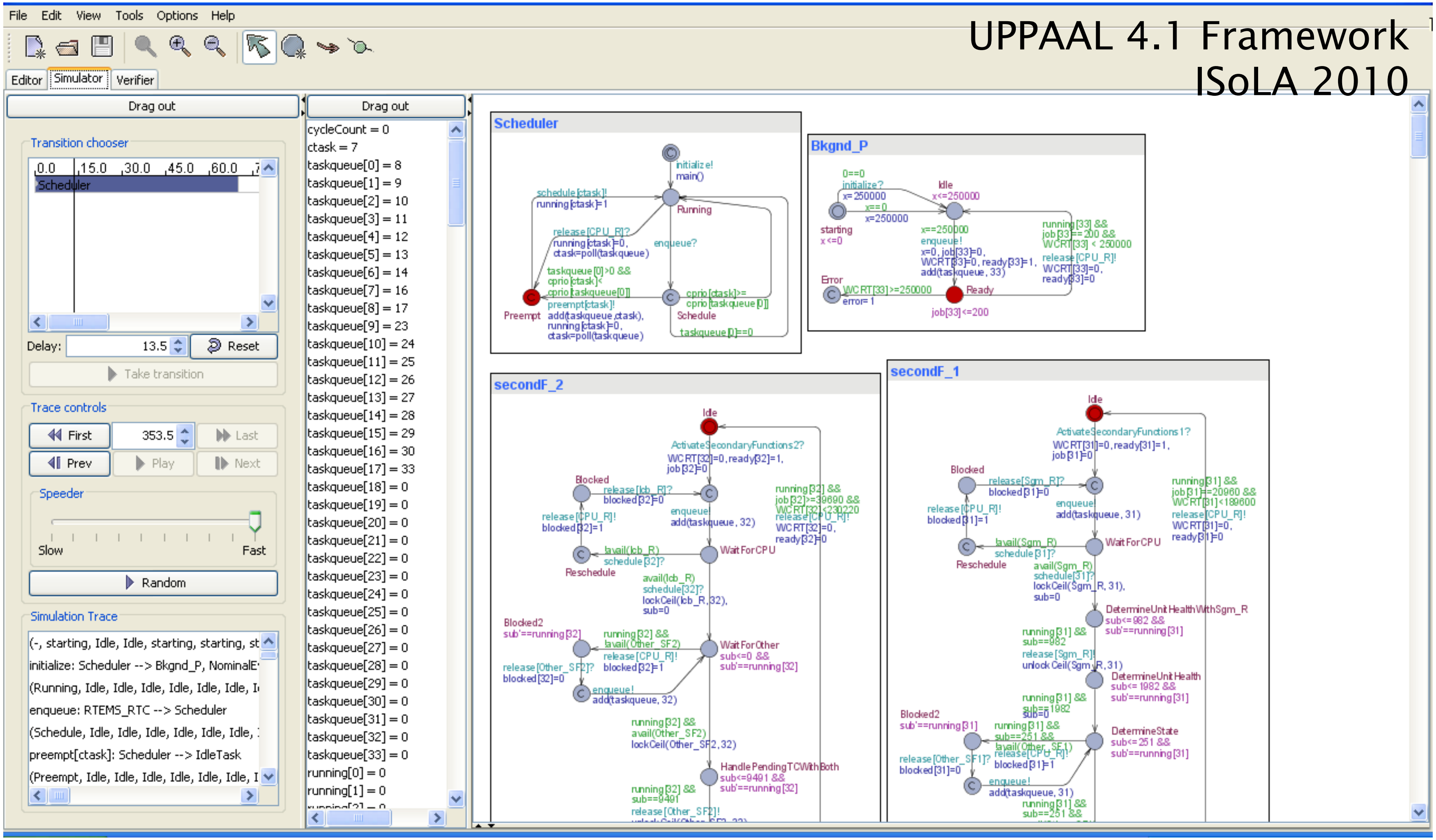
Requirements:

Software tasks should be schedulable.

CPU utilization should not exceed 50% load

Modeling in UPPAAL

UPPAAL 4.1 Framework
ISoLA 2010



```
cycleCount = 0
ctask = 7
taskqueue[0] = 8
taskqueue[1] = 9
taskqueue[2] = 10
taskqueue[3] = 11
taskqueue[4] = 12
taskqueue[5] = 13
taskqueue[6] = 14
taskqueue[7] = 16
taskqueue[8] = 17
taskqueue[9] = 23
taskqueue[10] = 24
taskqueue[11] = 25
taskqueue[12] = 26
taskqueue[13] = 27
taskqueue[14] = 28
taskqueue[15] = 29
taskqueue[16] = 30
taskqueue[17] = 33
taskqueue[18] = 0
taskqueue[19] = 0
taskqueue[20] = 0
taskqueue[21] = 0
taskqueue[22] = 0
taskqueue[23] = 0
taskqueue[24] = 0
taskqueue[25] = 0
taskqueue[26] = 0
taskqueue[27] = 0
taskqueue[28] = 0
taskqueue[29] = 0
taskqueue[30] = 0
taskqueue[31] = 0
taskqueue[32] = 0
taskqueue[33] = 0
running[0] = 0
running[1] = 0
```

```
(-, starting, Idle, Idle, starting, starting, st
initialize: Scheduler --> Bkgnd_P, NominalE
(Running, Idle, Idle, Idle, Idle, Idle, Idle, I
enqueue: RTEMS_RTC --> Scheduler
(Schedule, Idle, Idle, Idle, Idle, Idle, Idle,
preempt[ctask]: Scheduler --> IdleTask
(Preempt, Idle, Idle, Idle, Idle, Idle, Idle, I
```

Gantt Chart 1. cycle

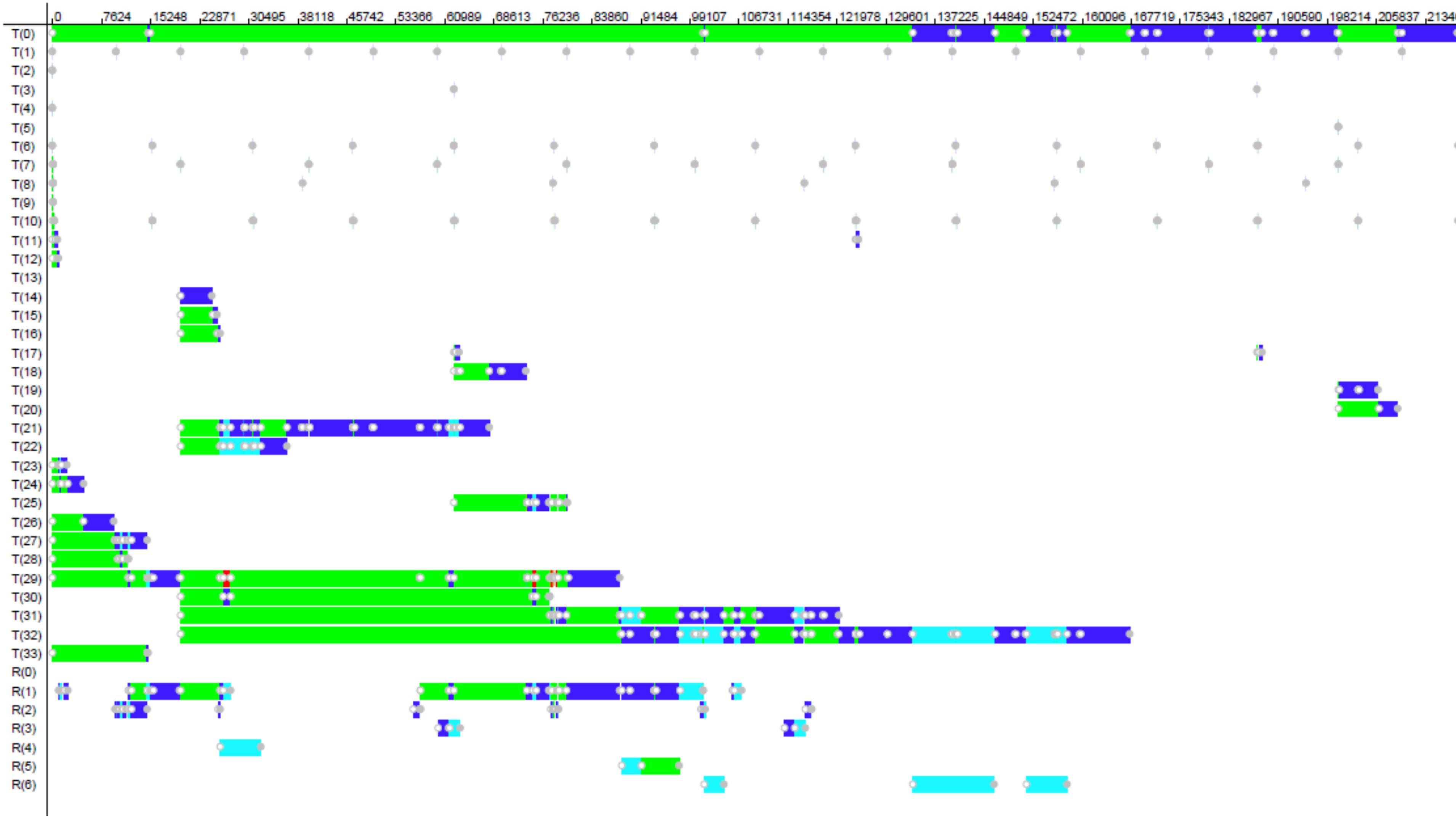
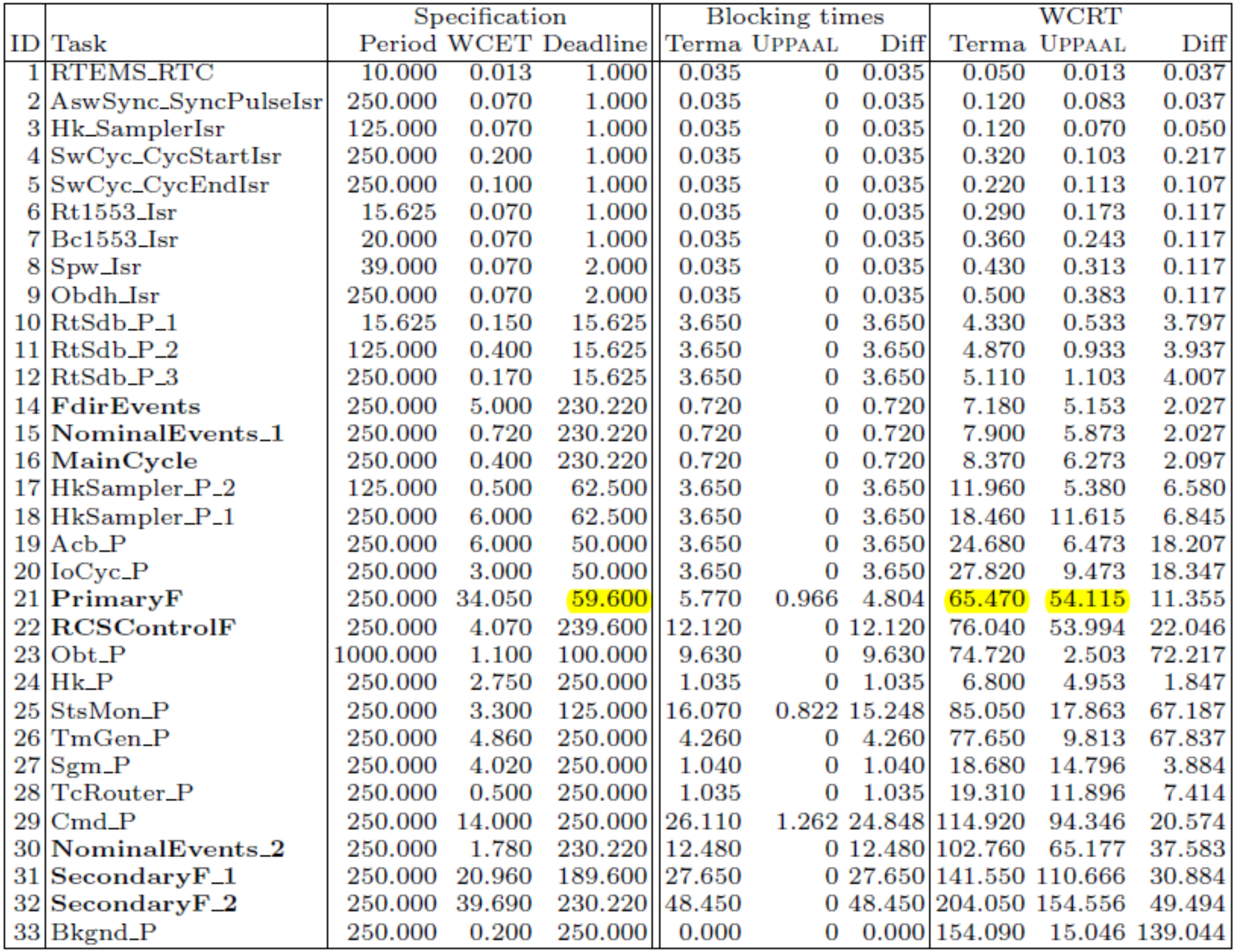


Fig. 11. Gantt chart of a schedule from the first cycle: green means ready, blue means running, cyan means suspended, red means blocked. R stand for resources: CPU_R=0, Icb_R=1, Sgm_R=2, PmReq_R=3, Other_RCS=4, Other_SF1=5, Other_SF2=6.

Blocking & WCRT

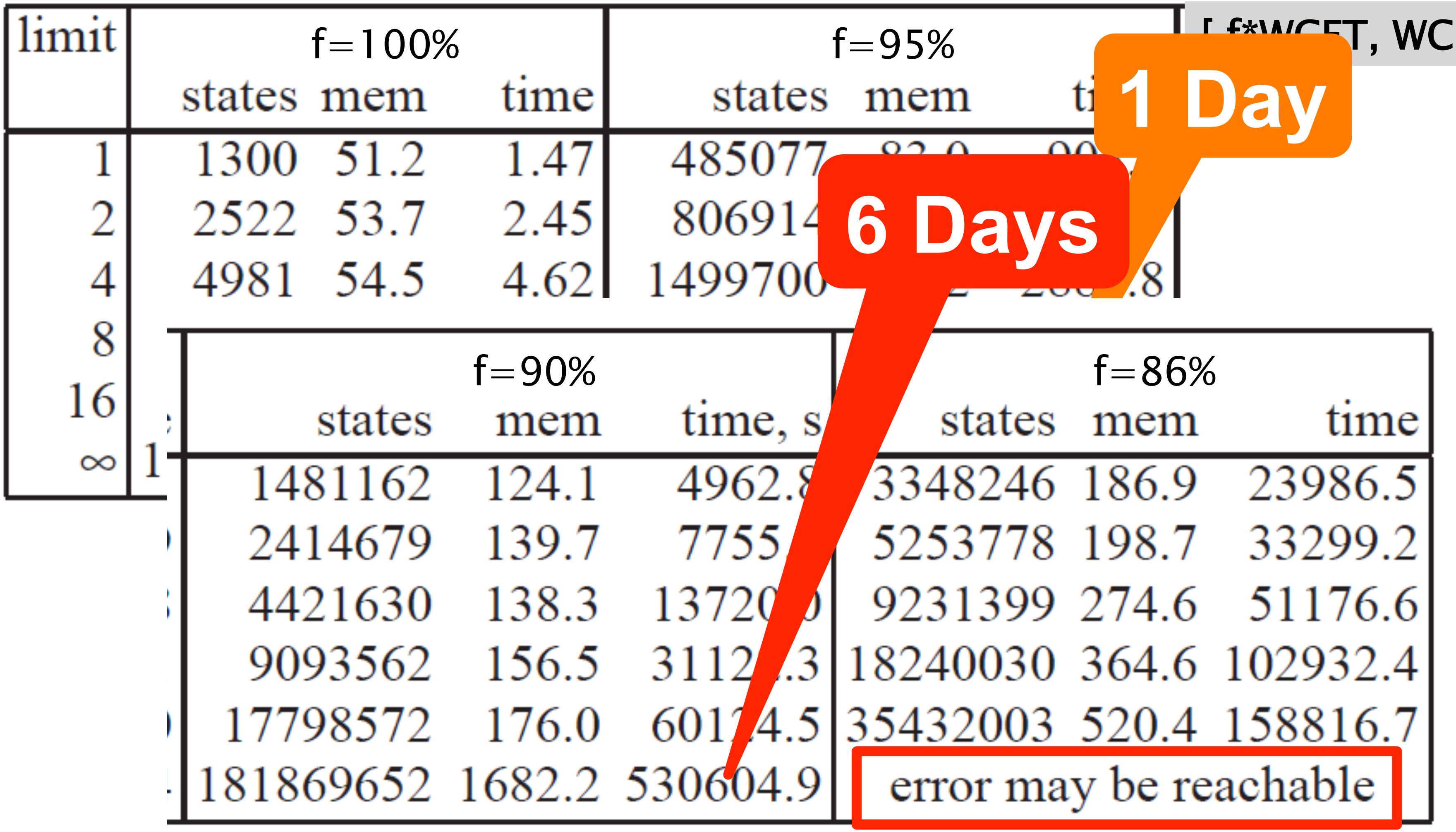


ID	Task	Specification			Blocking times			WCRT		
		Period	WCET	Deadline	Terma	UPPAAL	Diff	Terma	UPPAAL	Diff
1	RTEMS_RTC	10.000	0.013	1.000	0.035	0	0.035	0.050	0.013	0.037
2	AswSync_SyncPulseIsr	250.000	0.070	1.000	0.035	0	0.035	0.120	0.083	0.037
3	Hk_SamplerIsr	125.000	0.070	1.000	0.035	0	0.035	0.120	0.070	0.050
4	SwCyc_CycStartIsr	250.000	0.200	1.000	0.035	0	0.035	0.320	0.103	0.217
5	SwCyc_CycEndIsr	250.000	0.100	1.000	0.035	0	0.035	0.220	0.113	0.107
6	Rt1553_Isr	15.625	0.070	1.000	0.035	0	0.035	0.290	0.173	0.117
7	Bc1553_Isr	20.000	0.070	1.000	0.035	0	0.035	0.360	0.243	0.117
8	Spw_Isr	39.000	0.070	2.000	0.035	0	0.035	0.430	0.313	0.117
9	Obdh_Isr	250.000	0.070	2.000	0.035	0	0.035	0.500	0.383	0.117
10	RtSdb_P_1	15.625	0.150	15.625	3.650	0	3.650	4.330	0.533	3.797
11	RtSdb_P_2	125.000	0.400	15.625	3.650	0	3.650	4.870	0.933	3.937
12	RtSdb_P_3	250.000	0.170	15.625	3.650	0	3.650	5.110	1.103	4.007
14	**FdirEvents**	250.000	5.000	230.220	0.720	0	0.720	7.180	5.153	2.027
15	**NominalEvents_1**	250.000	0.720	230.220	0.720	0	0.720	7.900	5.873	2.027
16	**MainCycle**	250.000	0.400	230.220	0.720	0	0.720	8.370	6.273	2.097
17	HkSampler_P_2	125.000	0.500	62.500	3.650	0	3.650	11.960	5.380	6.580
18	HkSampler_P_1	250.000	6.000	62.500	3.650	0	3.650	18.460	11.615	6.845
19	Acb_P	250.000	6.000	50.000	3.650	0	3.650	24.680	6.473	18.207
20	IoCyc_P	250.000	3.000	50.000	3.650	0	3.650	27.820	9.473	18.347
21	**PrimaryF**	250.000	34.050	59.600	5.770	0.966	4.804	65.470	54.115	11.355
22	**RCSControlF**	250.000	4.070	239.600	12.120	0	12.120	76.040	53.994	22.046
23	Obt_P	1000.000	1.100	100.000	9.630	0	9.630	74.720	2.503	72.217
24	Hk_P	250.000	2.750	250.000	1.035	0	1.035	6.800	4.953	1.847
25	StsMon_P	250.000	3.300	125.000	16.070	0.822	15.248	85.050	17.863	67.187
26	TmGen_P	250.000	4.860	250.000	4.260	0	4.260	77.650	9.813	67.837
27	Sgm_P	250.000	4.020	250.000	1.040	0	1.040	18.680	14.796	3.884
28	TcRouter_P	250.000	0.500	250.000	1.035	0	1.035	19.310	11.896	7.414
29	Cmd_P	250.000	14.000	250.000	26.110	1.262	24.848	114.920	94.346	20.574
30	**NominalEvents_2**	250.000	1.780	230.220	12.480	0	12.480	102.760	65.177	37.583
31	**SecondaryF_1**	250.000	20.960	189.600	27.650	0	27.650	141.550	110.666	30.884
32	**SecondaryF_2**	250.000	39.690	230.220	48.450	0	48.450	204.050	154.556	49.494
33	Bkgnd_P	250.000	0.200	250.000	0.000	0	0.000	154.090	15.046	139.044

Marius Micusionis

TERMA Case Follow-Up

ISOLA 2012



[f*WCET, WCET]

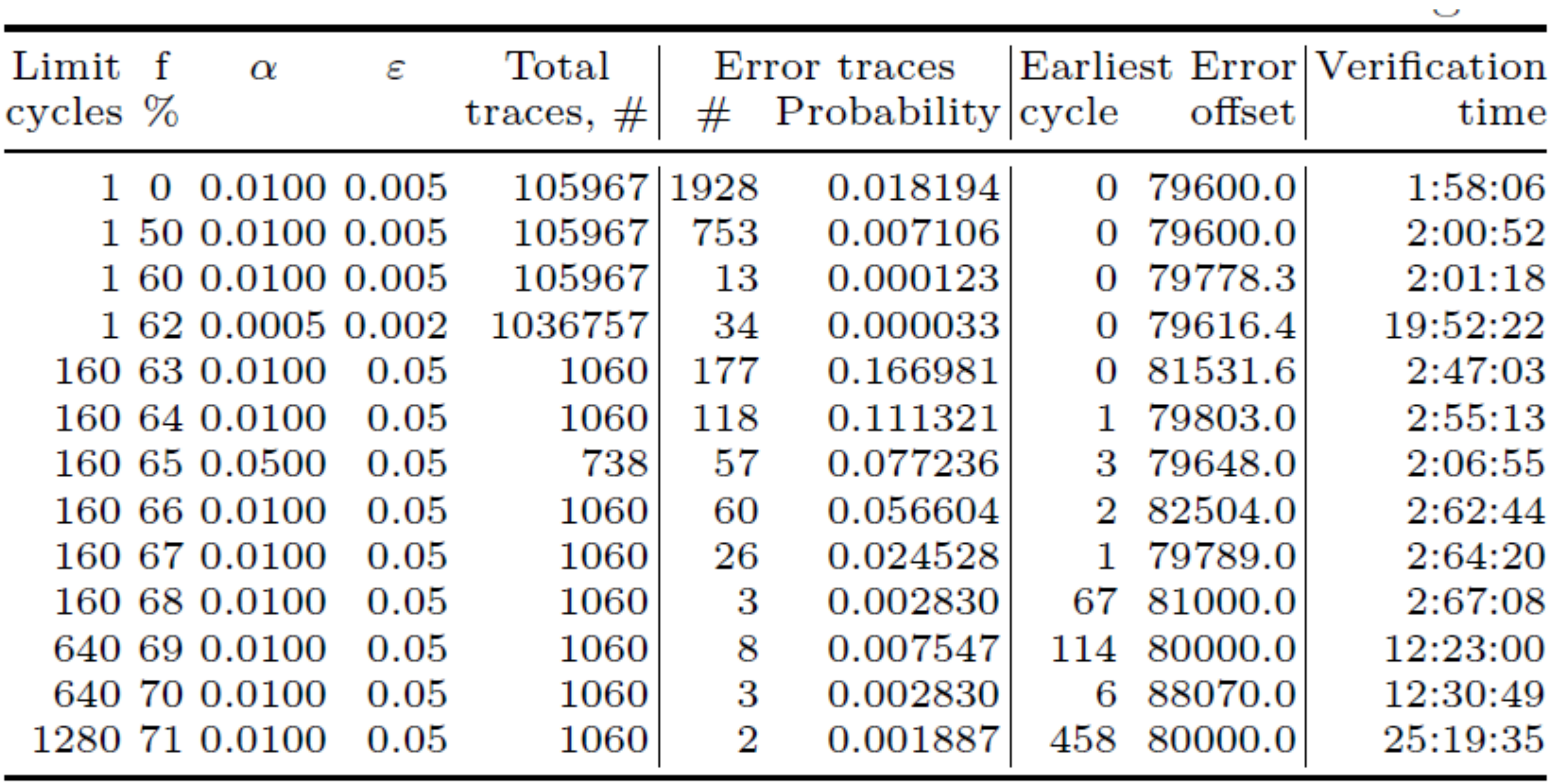
limit	f=100%			f=95%		
	states	mem	time	states	mem	ti[illegible]
1	1300	51.2	1.47	485077	[illegible]	[illegible]
2	2522	53.7	2.45	806914	[illegible]	[illegible]
4	4981	54.5	4.62	1499700	[illegible]	[illegible].8
8						
16						
∞	1[illegible]					

f=90%			f=86%		
states	mem	time, s	states	mem	time
1481162	124.1	4962.8	3348246	186.9	23986.5
2414679	139.7	7755.[illegible]	5253778	198.7	33299.2
4421630	138.3	13720[illegible]0	9231399	274.6	51176.6
9093562	156.5	3112[illegible].3	18240030	364.6	102932.4
17798572	176.0	601[illegible]4.5	35432003	520.4	158816.7
181869652	1682.2	530604.9	error may be reachable		

1 Day

6 Days

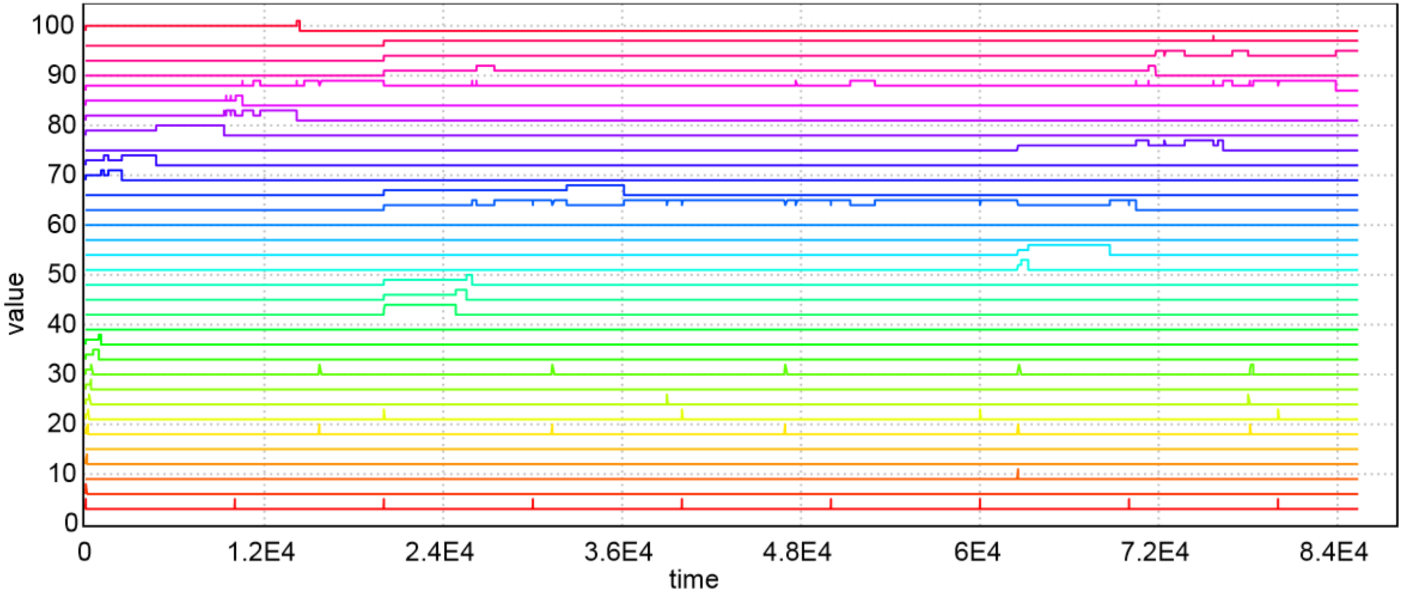
TERMA Case – Statistical MC



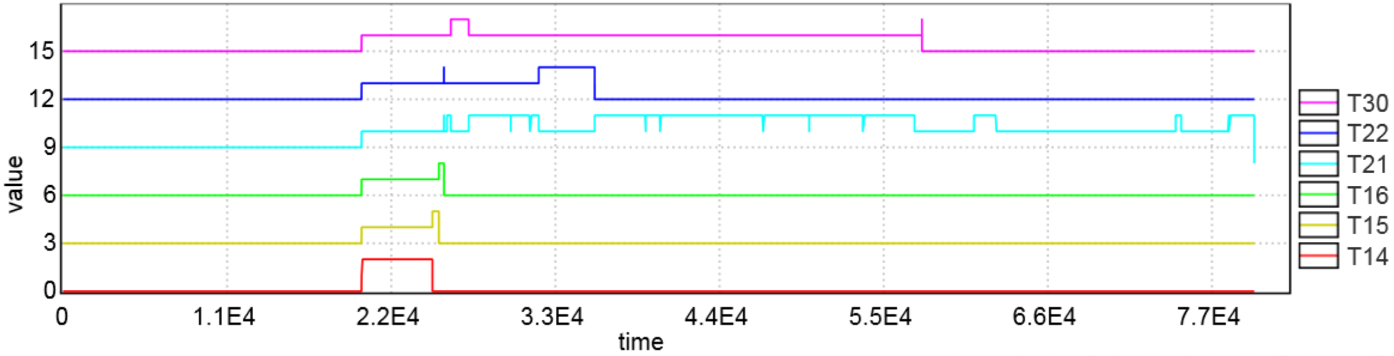
Limit cycles	f %	α	ε	Total traces, #	Error traces #	 Probability	Earliest cycle	Error offset	Verification time
1	0	0.0100	0.005	105967	1928	0.018194	0	79600.0	1:58:06
1	50	0.0100	0.005	105967	753	0.007106	0	79600.0	2:00:52
1	60	0.0100	0.005	105967	13	0.000123	0	79778.3	2:01:18
1	62	0.0005	0.002	1036757	34	0.000033	0	79616.4	19:52:22
160	63	0.0100	0.05	1060	177	0.166981	0	81531.6	2:47:03
160	64	0.0100	0.05	1060	118	0.111321	1	79803.0	2:55:13
160	65	0.0500	0.05	738	57	0.077236	3	79648.0	2:06:55
160	66	0.0100	0.05	1060	60	0.056604	2	82504.0	2:62:44
160	67	0.0100	0.05	1060	26	0.024528	1	79789.0	2:64:20
160	68	0.0100	0.05	1060	3	0.002830	67	81000.0	2:67:08
640	69	0.0100	0.05	1060	8	0.007547	114	80000.0	12:23:00
640	70	0.0100	0.05	1060	3	0.002830	6	88070.0	12:30:49
1280	71	0.0100	0.05	1060	2	0.001887	458	80000.0	25:19:35

TERMA Case – Conclusion

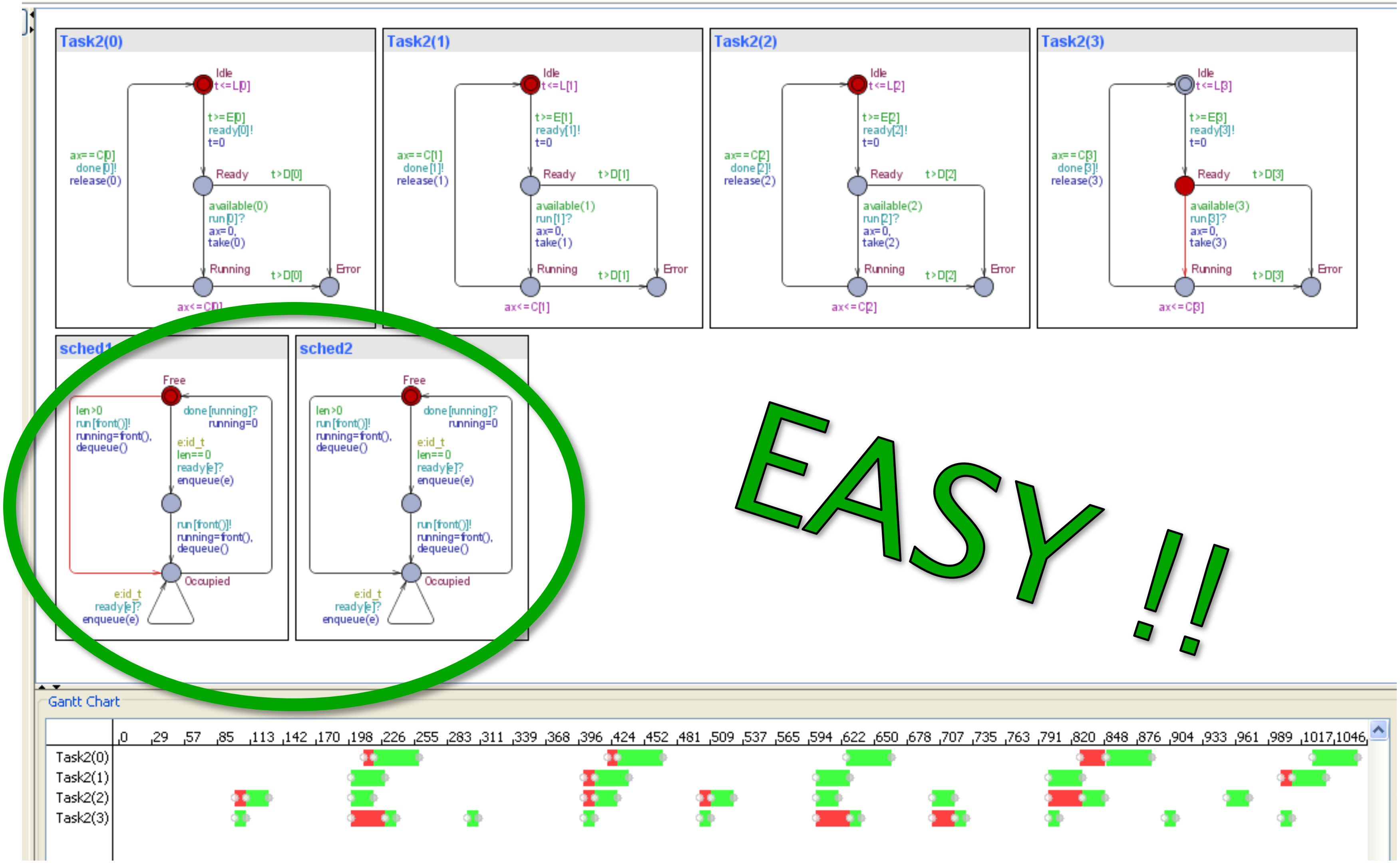
Herschel simulation run with $f = 90\%$:



Herschel deadline violation with $f = 50\%$:

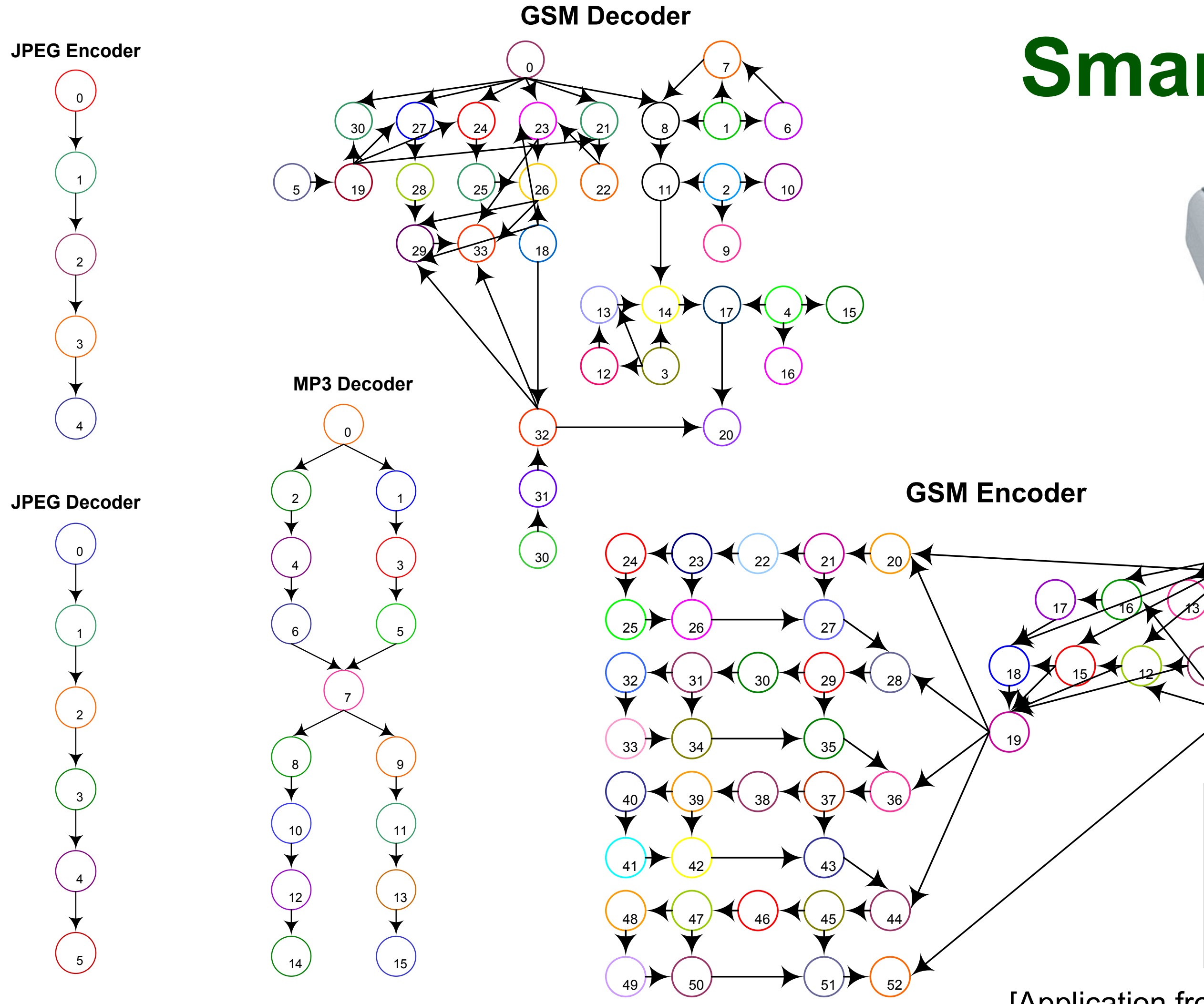


Multi–Processor



EASY !!

Handling realistic applications?

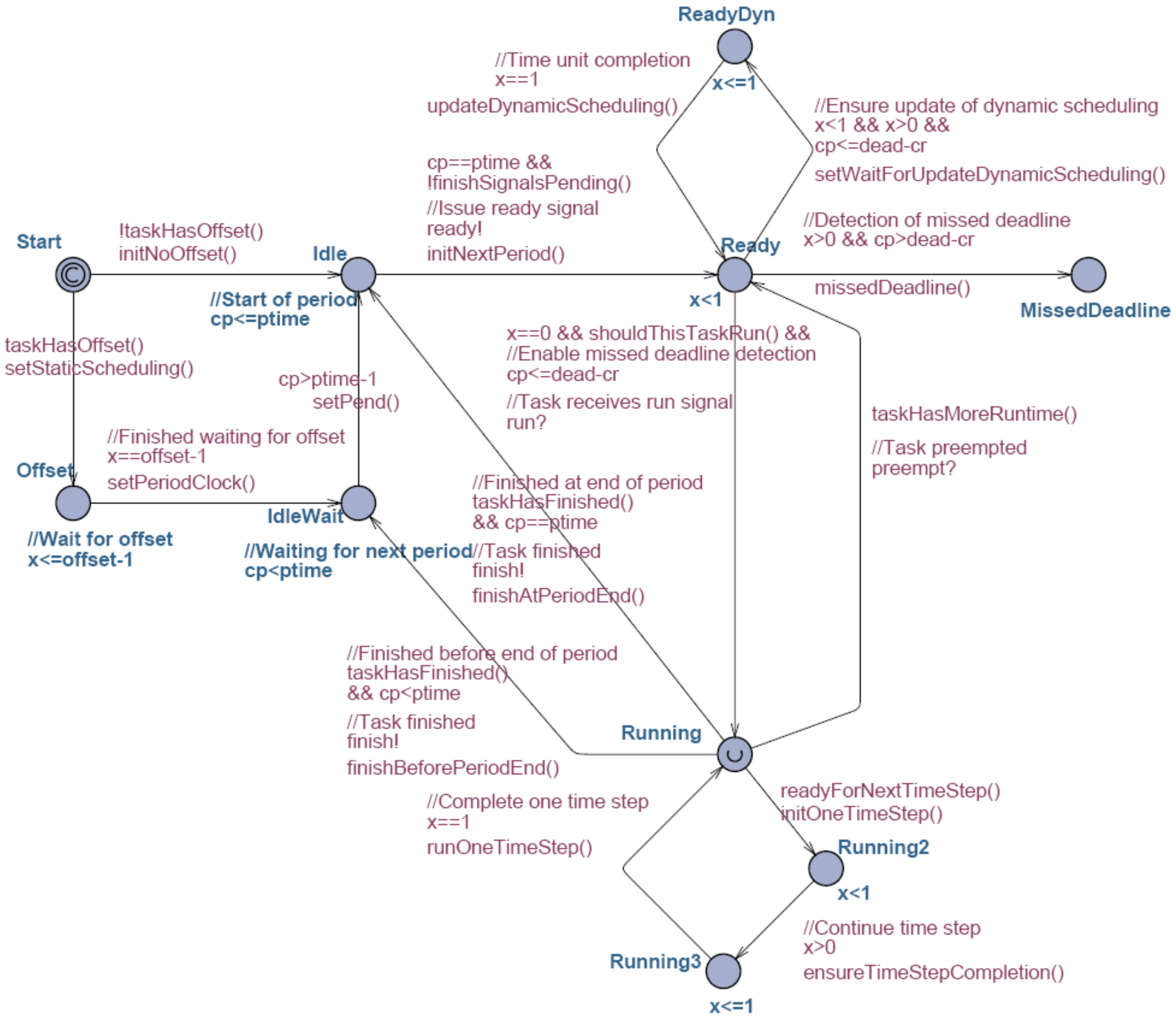


Smart phone:

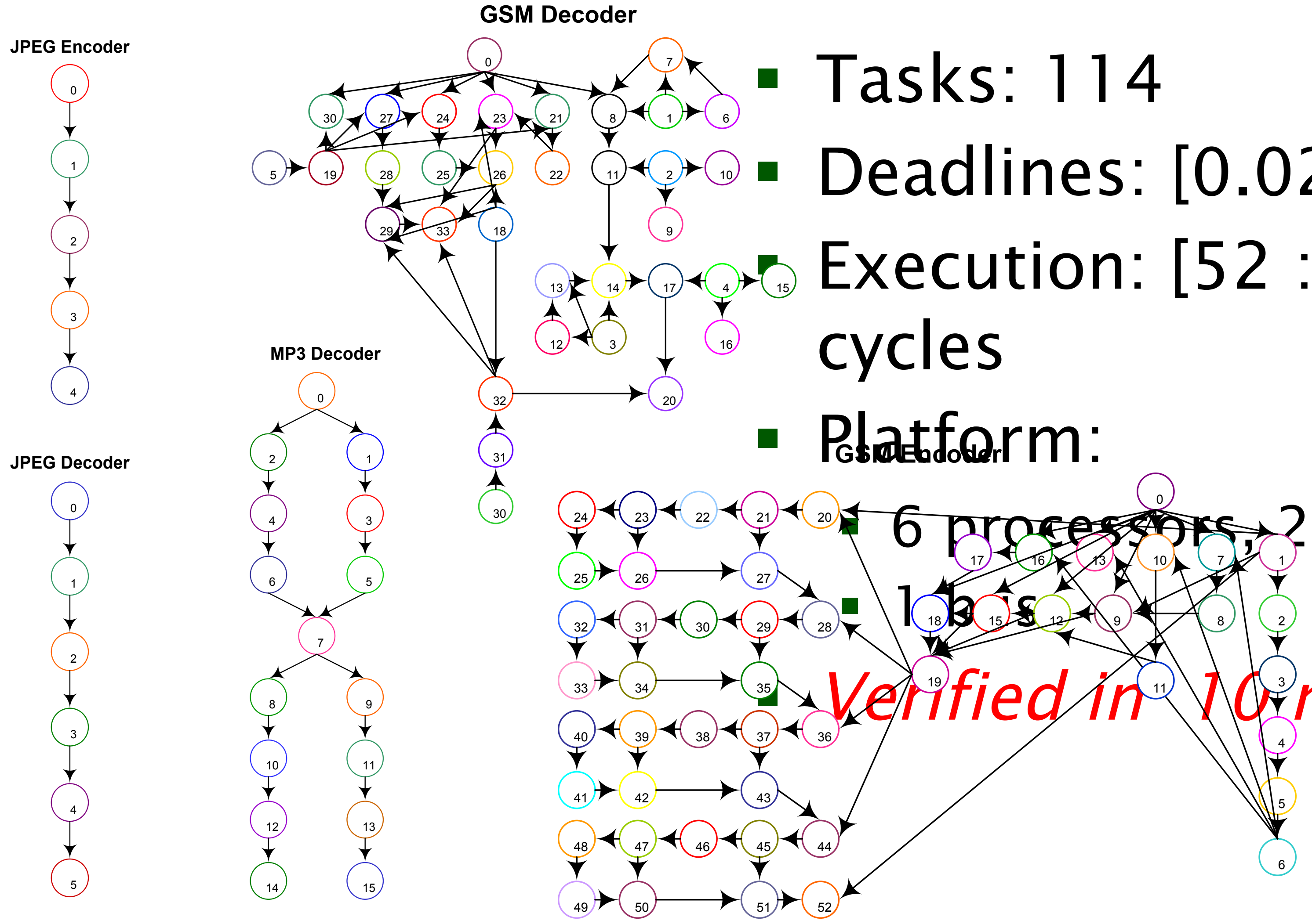
Jan Madsen
Aske Brekling
Michael R. Hansen/ DTU

[Application from Marcus Schmitz, TU Linkoping]

Timed Automata for a task



Smart phone



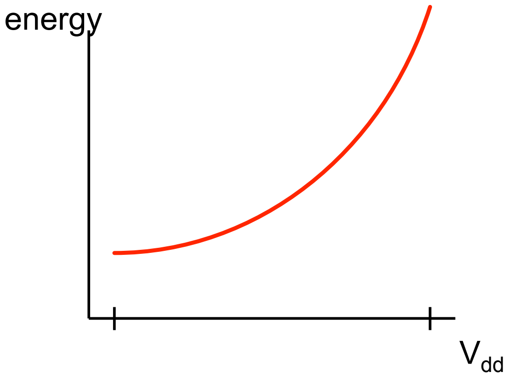
- Tasks: 114
- Deadlines: [0.02: 0.5] sec
- Execution: [52 : 266.687] cycles
- Platform:
  - 6 processors, 25 MHz
  - 1 bus

Verified in 10 min!

Energy and Scheduling ?

A non-experts understanding of CMOS

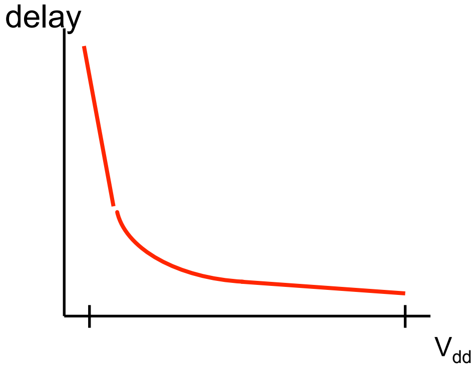
- Power consumption mainly by dynamic power



$$P_{dynamic} = C_L \cdot V_{dd}^2 \cdot f_{clk}$$

$$E_{dynamic\ pr.cycle} = C_L \cdot V_{dd}^2$$

- Supply voltage reduction => decreased frequency

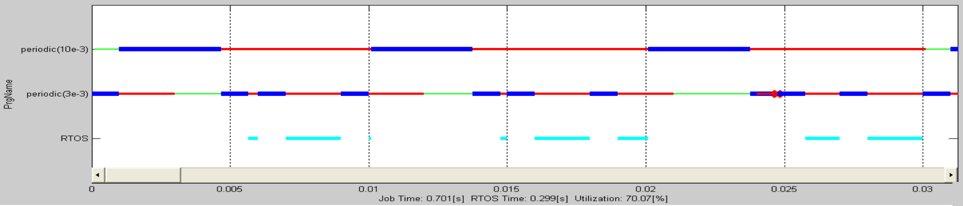


We may miss deadlines

$$f_{clk} \sim V_{dd}^{(\alpha\text{-}1)} \qquad ;\alpha > 1$$

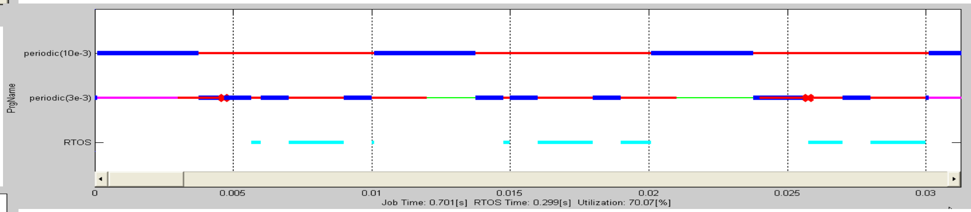
Dynamic Voltage Scaling & Task Scheduling

FCFS

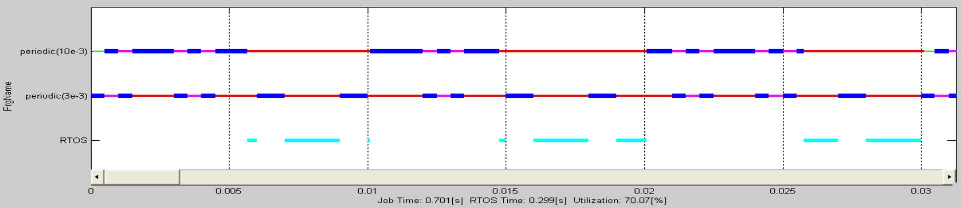


with/without preemption

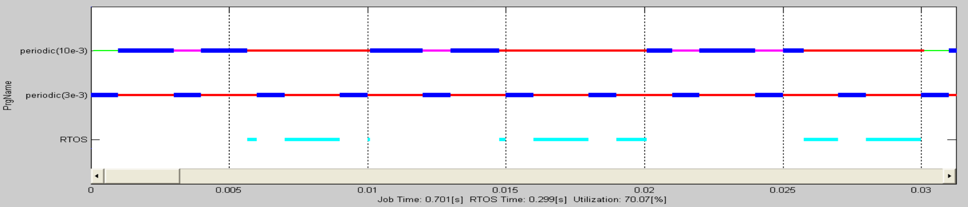
Fixed Priority



Time Slice



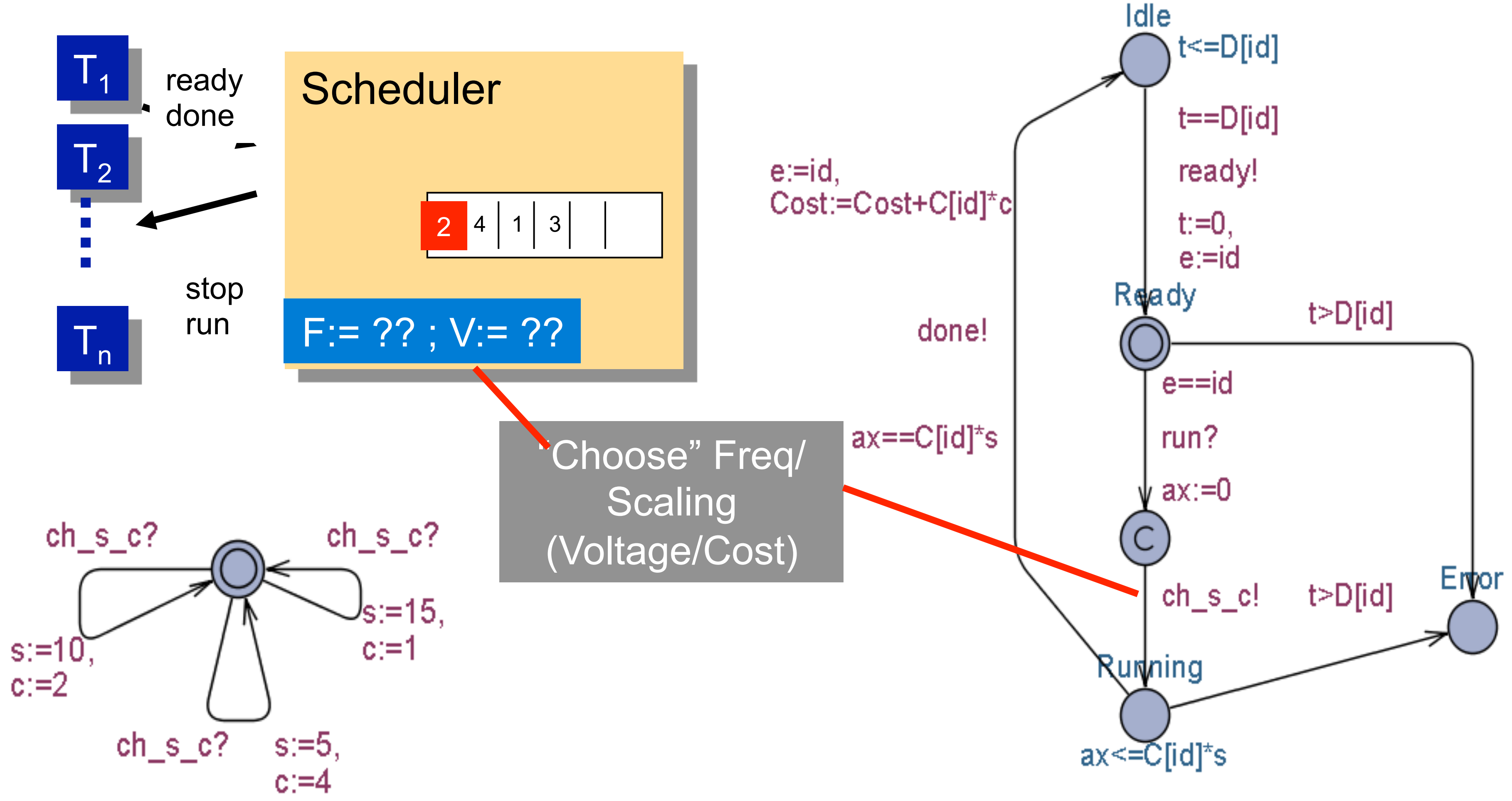
EDF



CPU not always fully utilized !
We may occasionally/dynamically lower frequency/supply voltage !
Save Energy

Energy Optimal Scheduling

Using PTA



T_1 ready done

T_2

T_n stop run

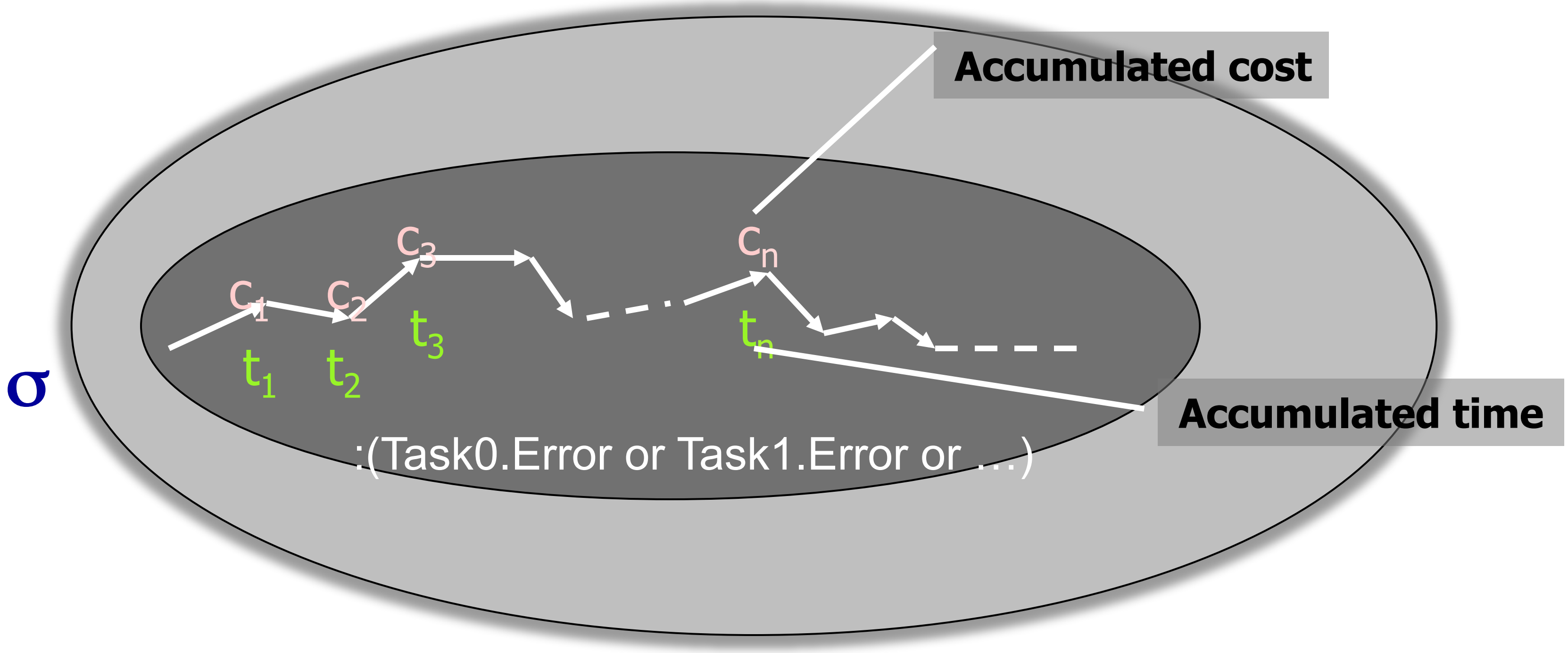
Scheduler

2 4 1 3

F:= ?? ; V:= ??

“Choose” Freq/ Scaling (Voltage/Cost)

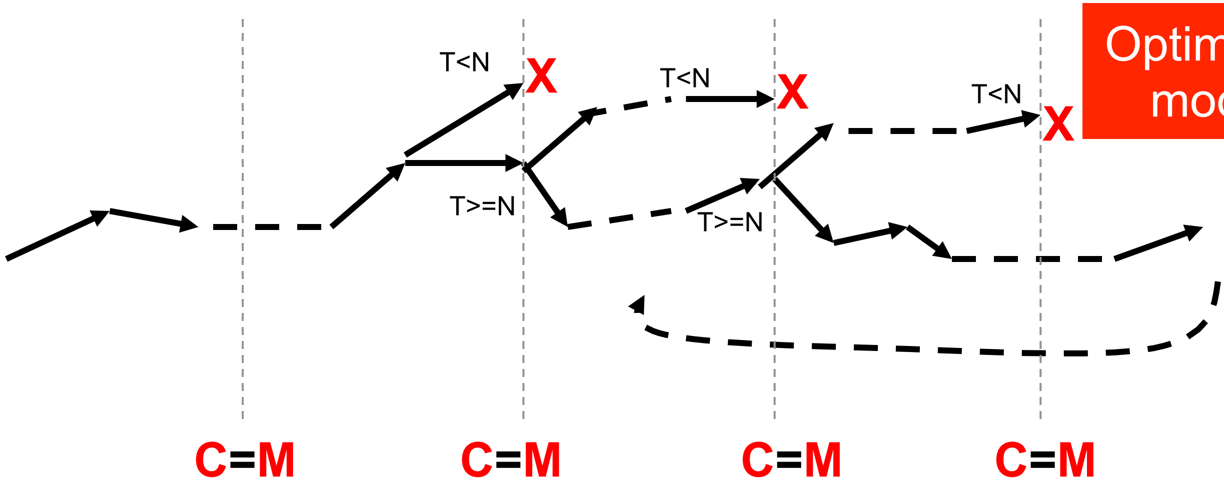
Energy Optimal Scheduling = Optimal Infinite Path



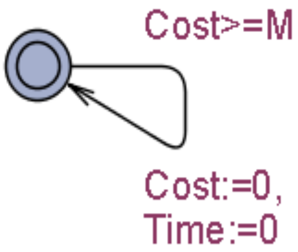
Value of path σ: $\mathrm{val}(\sigma) = \lim_{n!1} c_n/t_n$

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

Approximate Optimal Schedule



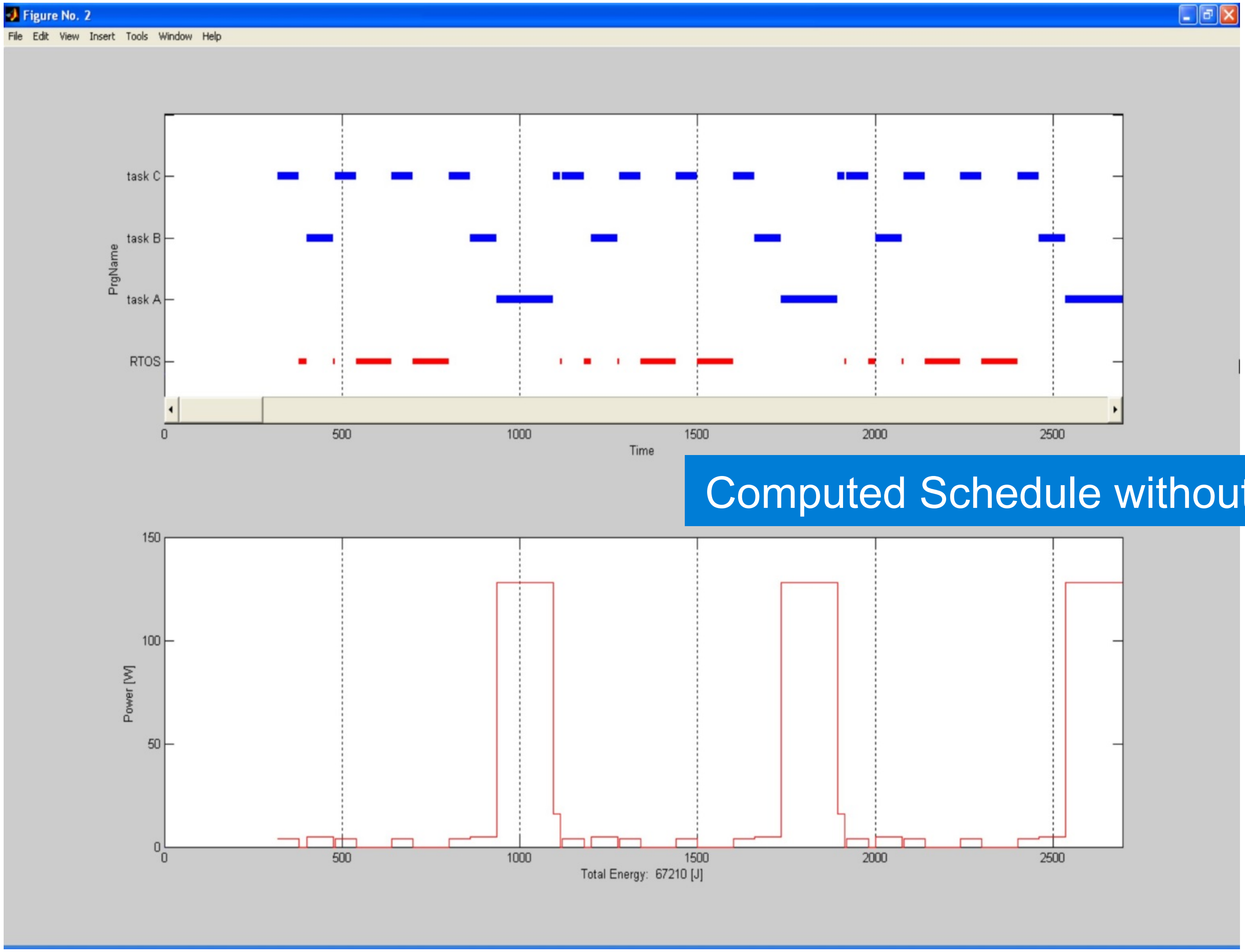
Optimal infinite schedule modulo cost-horizon



E[] (not (Task0.Error or Task1.Error or Task2.Error)
and
(cost>=M imply time >= N))
=
E[] ϕ(M,N)

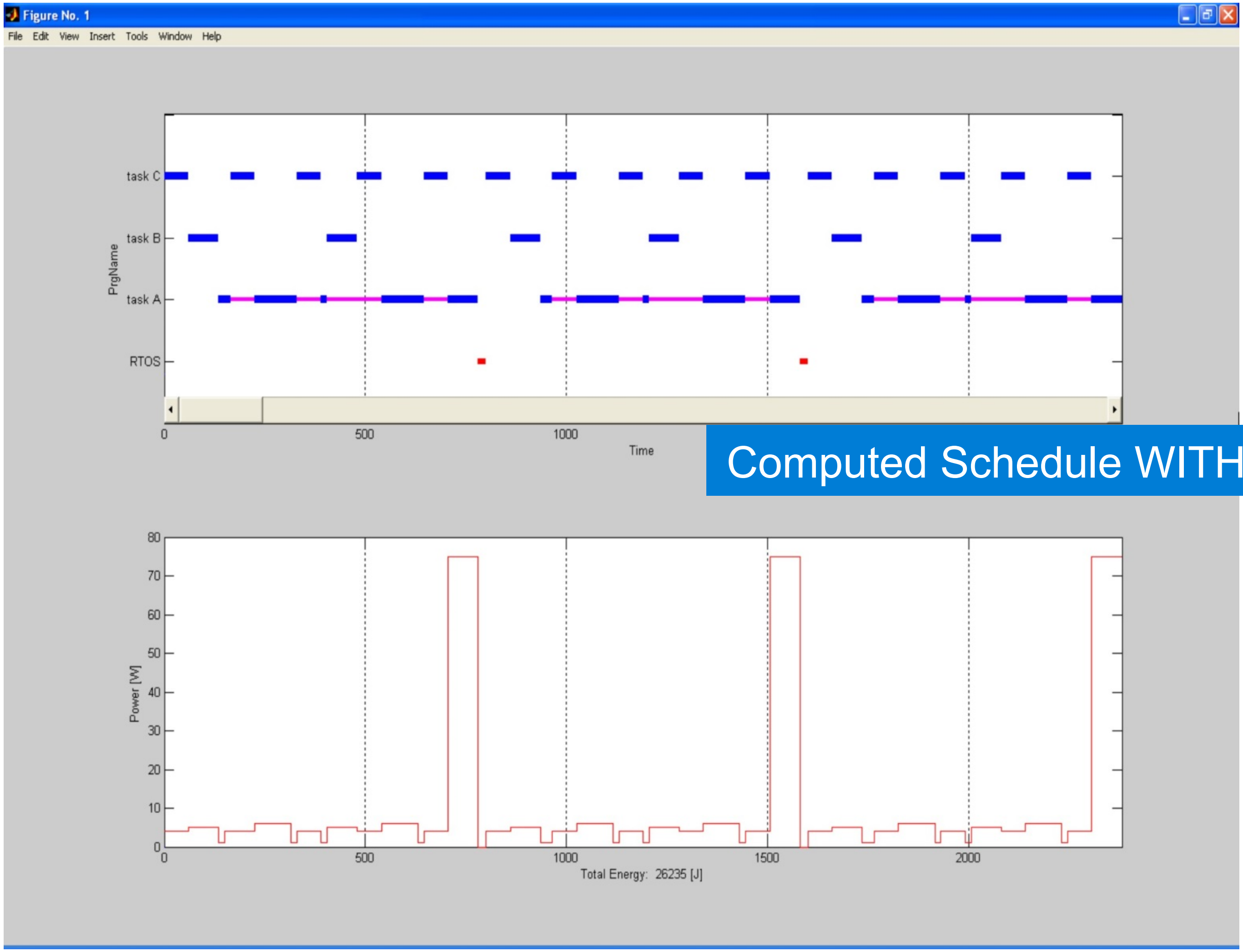
σ ² [] ϕ(M,N) imply val(σ)· M/N

Preliminary Results



Computed Schedule without DVS

Preliminary Results



Total Energy: 26235 [J]

Computed Schedule WITH DVS

Energy Automata

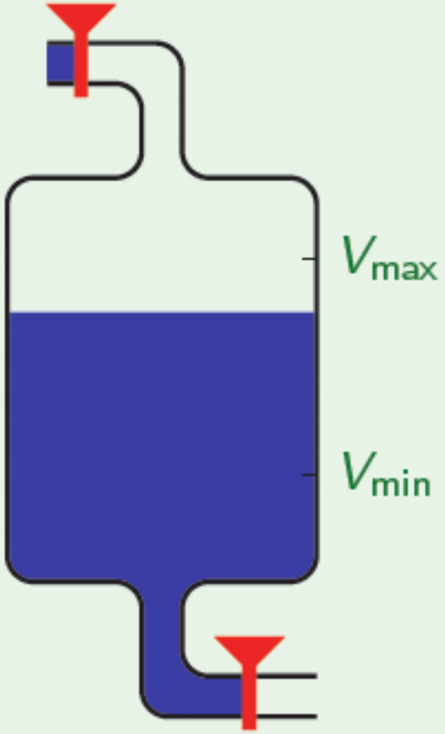
AALBORG UNIVERSITY DENMARK

Managing Resources

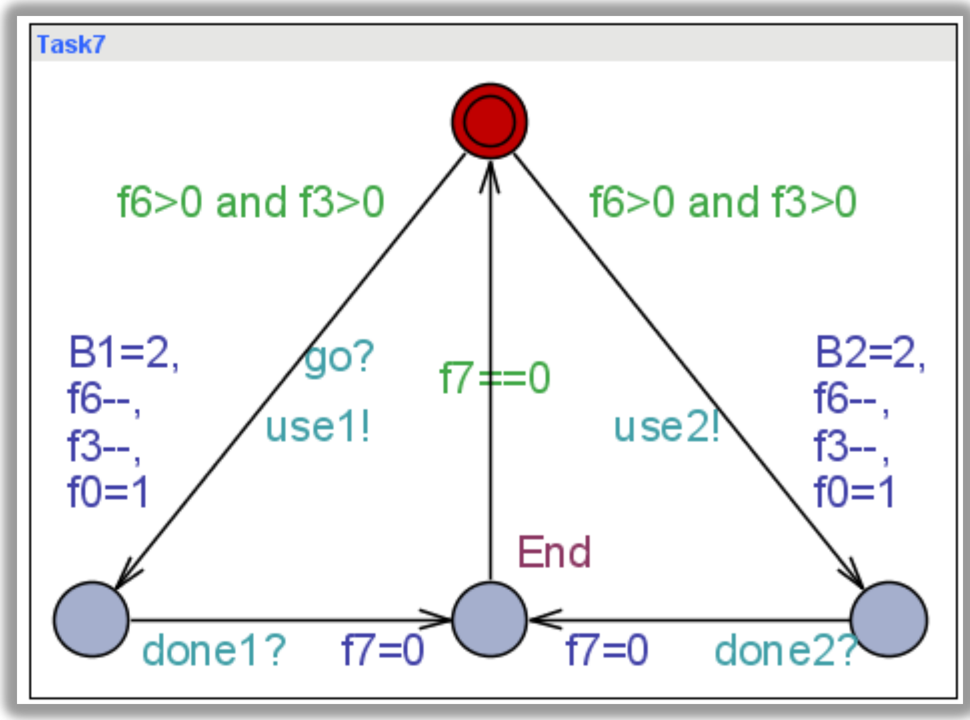
Example

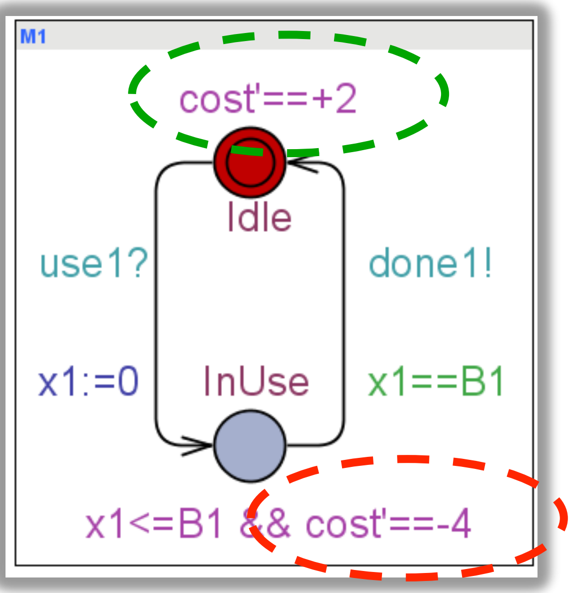
In some cases, resources can both be consumed and regained.

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



Consuming & Harvesting Energy

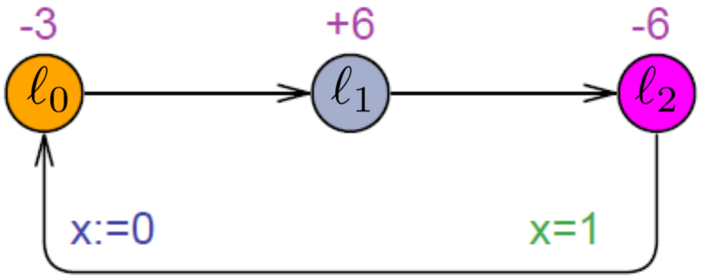


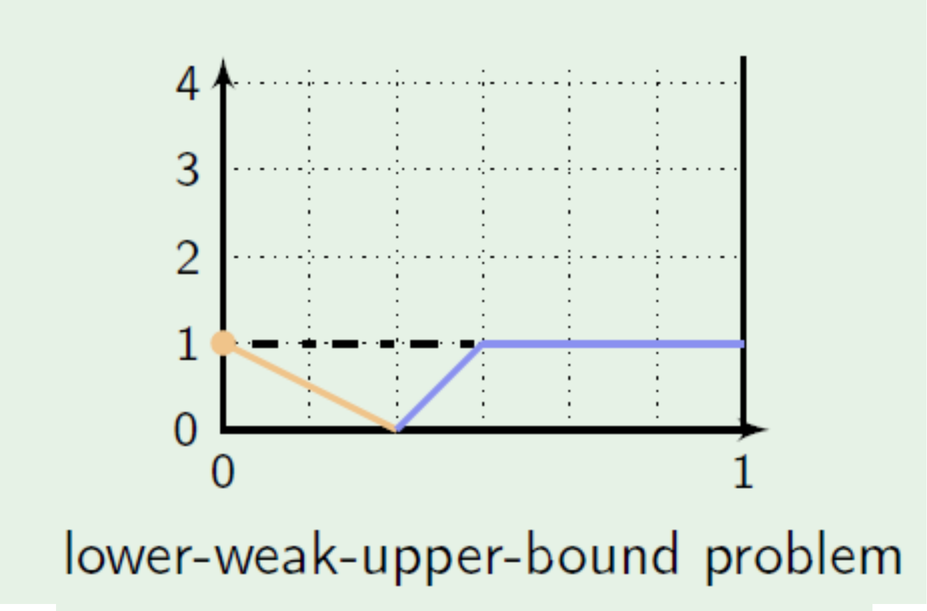


Maximize throughput
while respecting: 0 · E · MAX

Energy Constrains

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

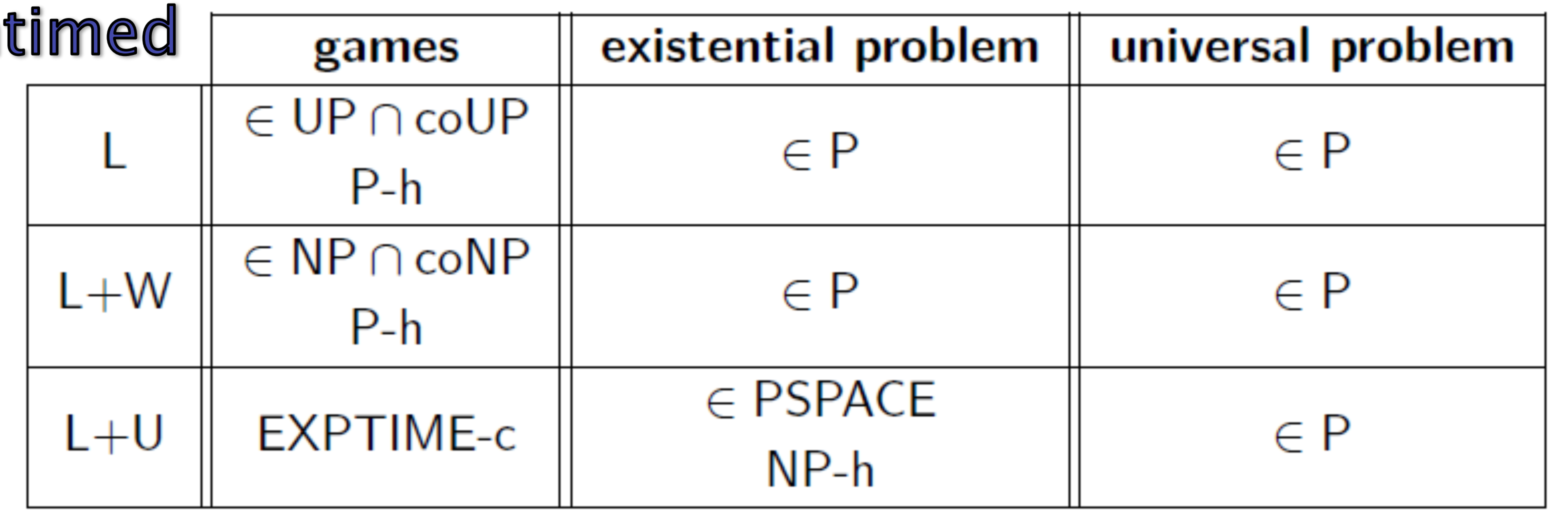




lower-weak-upper-bound problem

Results (so far)

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

P Bouyer, U Fahrenberg, K Larsen, N Markey,.. . Infinite runs in weighted timed automata with energy constraints. 2008.

One Weight Results

1 Clock	games	existential problem	universal problem
L	?	∈ P	∈ P
L+W	?	∈ P	∈ P
L+U	undecidable	decidable (flat)	?

1½ Clock	games	existential problem	universal problem
L	?	decidable	decidable
L+W	?	decidable	decidable

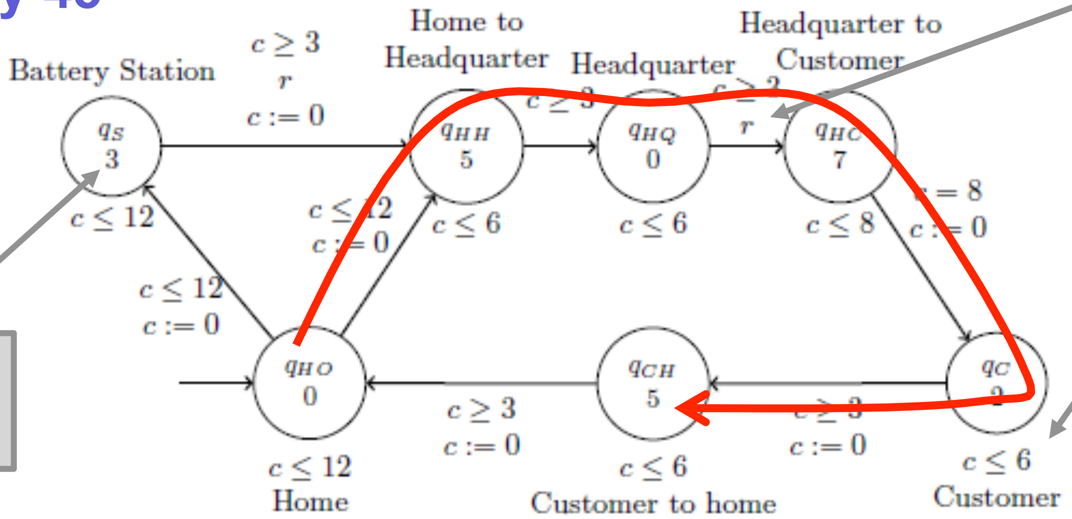
>3 Clocks	games	existential problem	universal problem
L	undecidable	undecidable	PSPACE-c
L+W	undecidable	undecidable	PSPACE-c

P Bouyer, U Fahrenberg, K Larsen, N Markey,.. . Infinite runs in weighted timed automata with energy constraints. 2008.
P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey: Timed automata with observers under energy constraints. HSCC 2010
P. Bouyer, K. G. Larsen, and N. Markey. Lower-bound constrained runs in weighted timed automata. QEST 2012

Recharge Automata

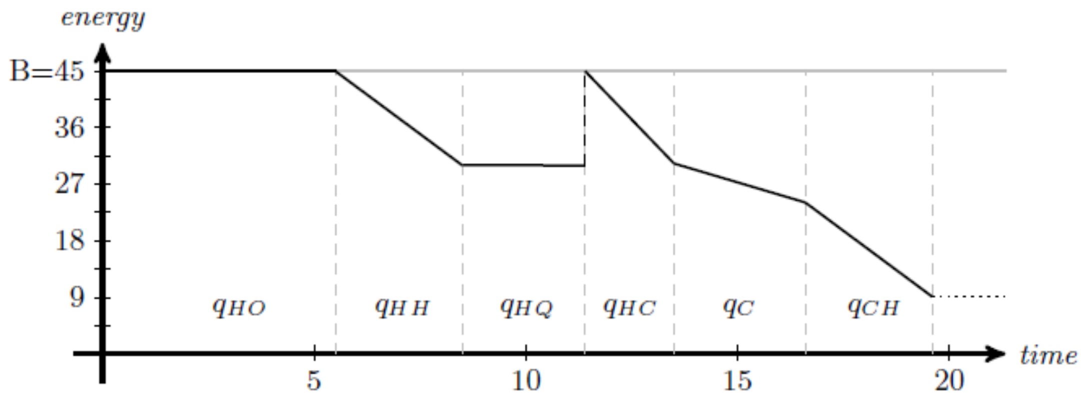
Max Capacity 45

r : recharge to **45**



c : clock

Consumption rate

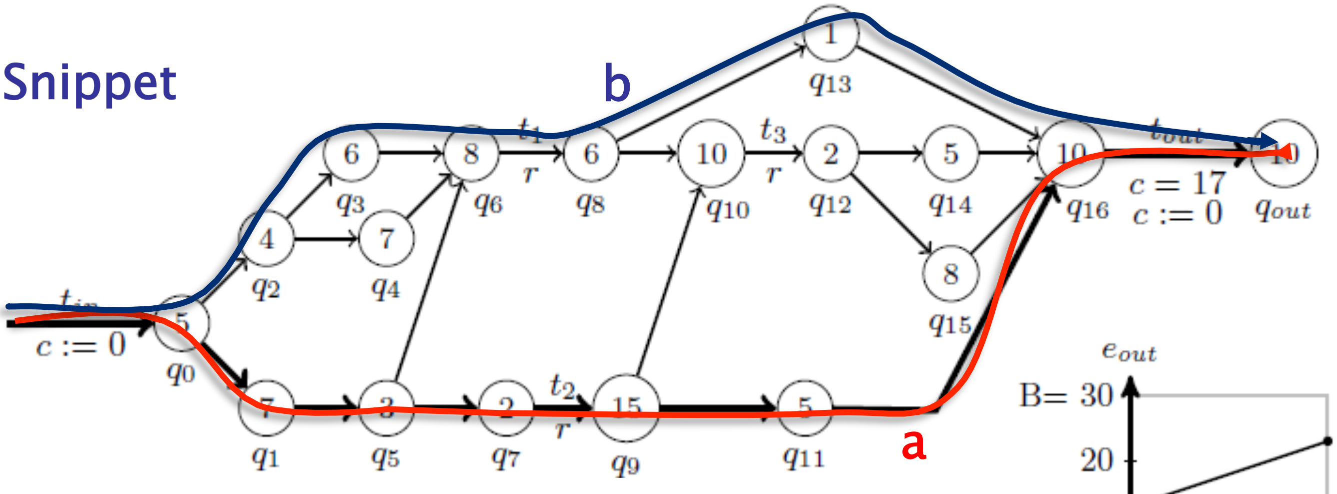


Run

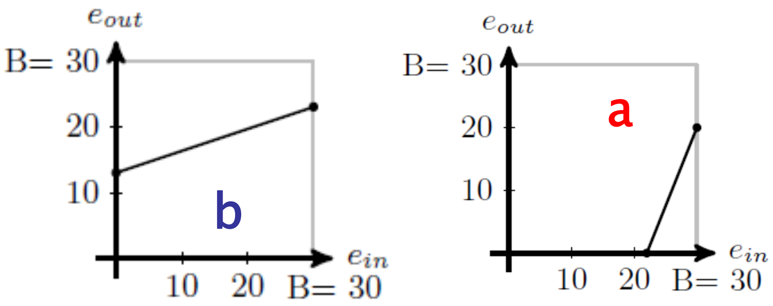
Daniel Ejsing-Duun, Lisa Fontani: Infinite Runs in Recharge Automata, MSc Thesis 2013

Recharge Automata

Snippet



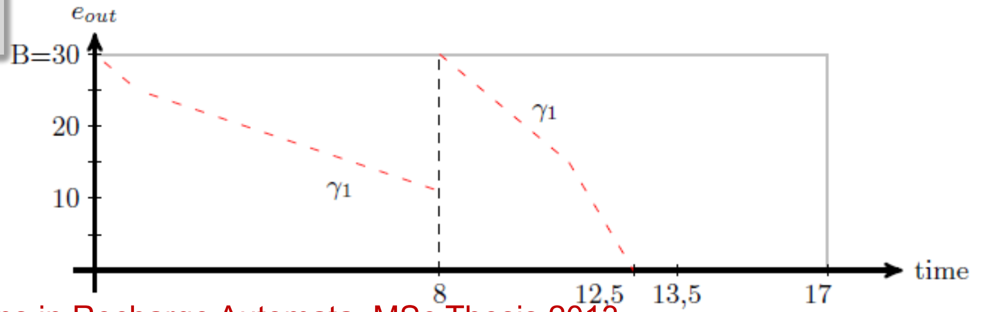
Energy Functions



THM [FED12]
Infinite run problem is in NP in general
P for "flat" RAs.

Do recharge as late as possible –
Delay in cheapest locations.

Run



Daniel Ejsing-Duun, Lisa Fontani: Infinite Runs in Recharge Automata, MSc Thesis 2013