Formal Verification and Performance Analysis of Embedded Systems

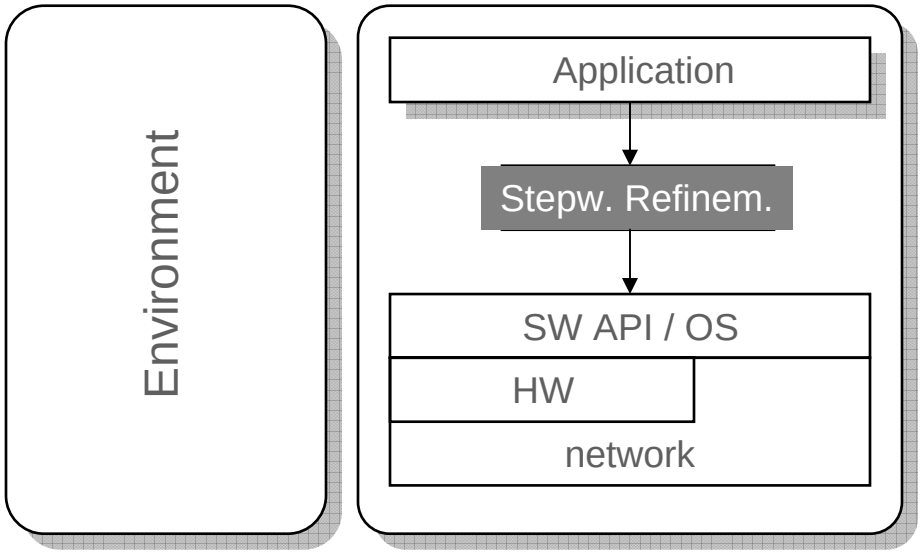
Kim G. Larsen

CISS, Aalborg U., DK

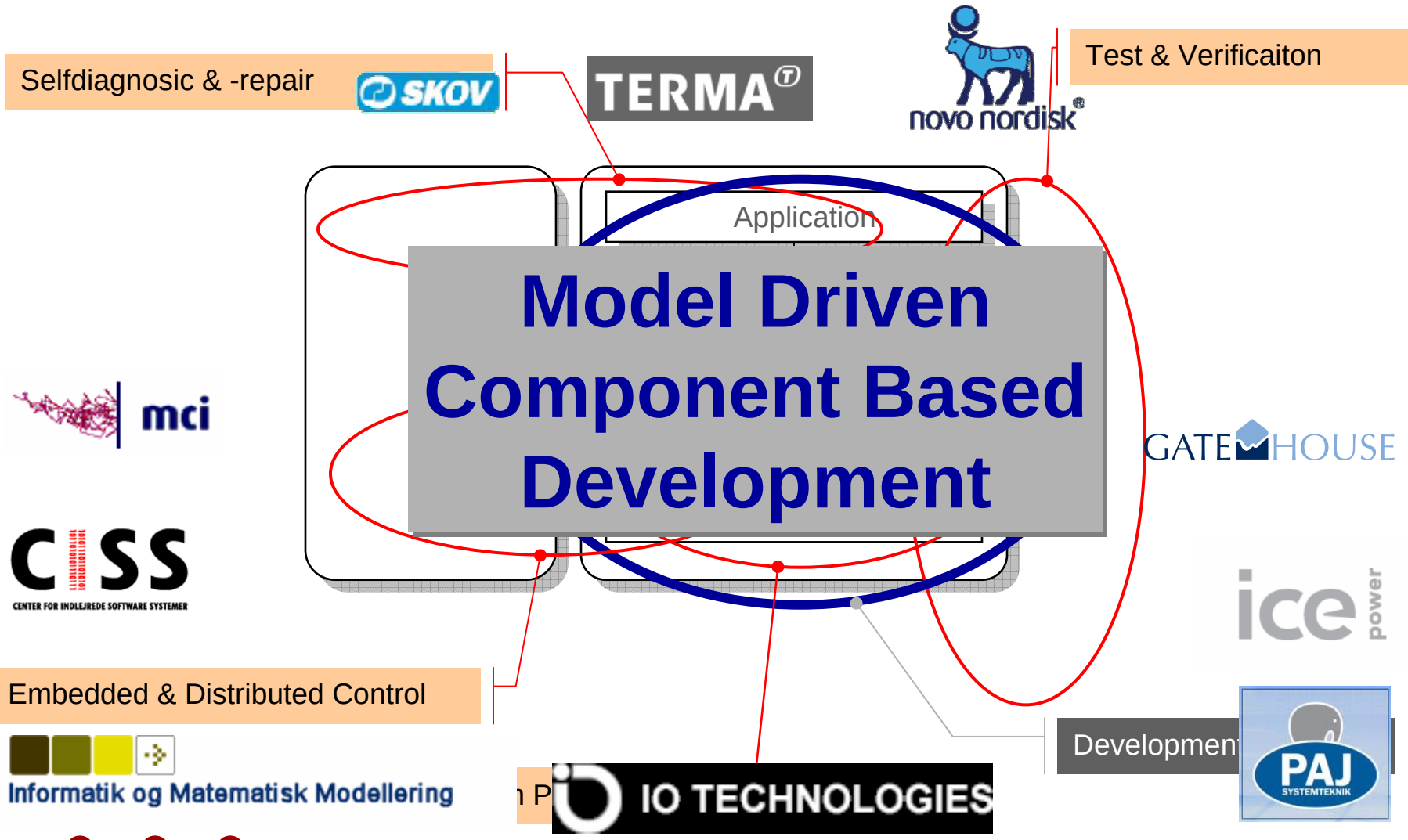
Michael R. Hansen

IMM, Technical U. of Denmark, DK

DaNES Challenges



DaNES Challenges



Outline

- Validation and Performance Analysis in UPPAAL / Kim G. Larsen
- Formalising the ARTS MPSoC Model in UPPAAL / Michael R. Hansen
- Discussion Points / KGL & MRH

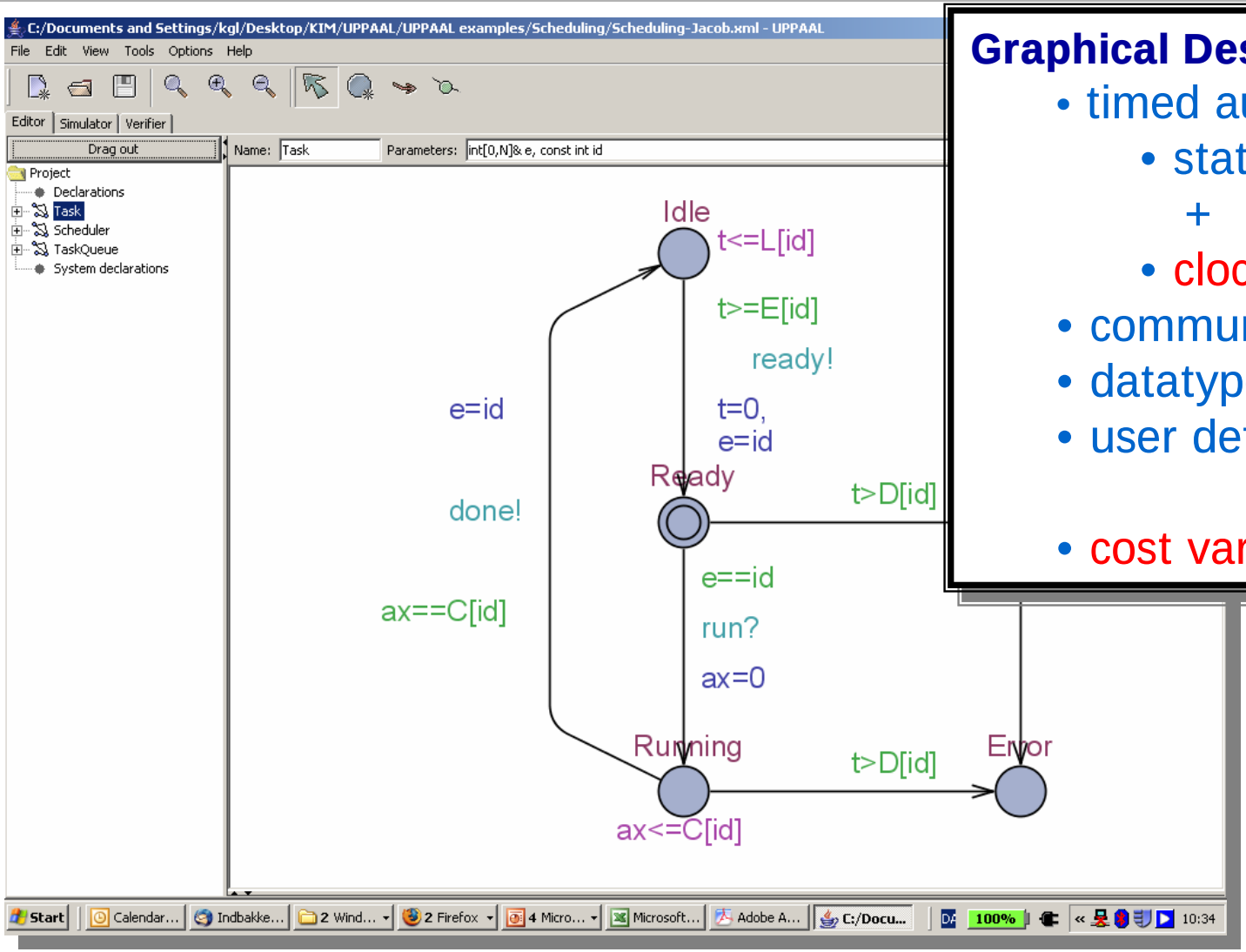
Validation and Performance Analysis in UPPAAL

DaNES

AALBORG UNIVERSITY DENMARK

DTU

UPPAAL

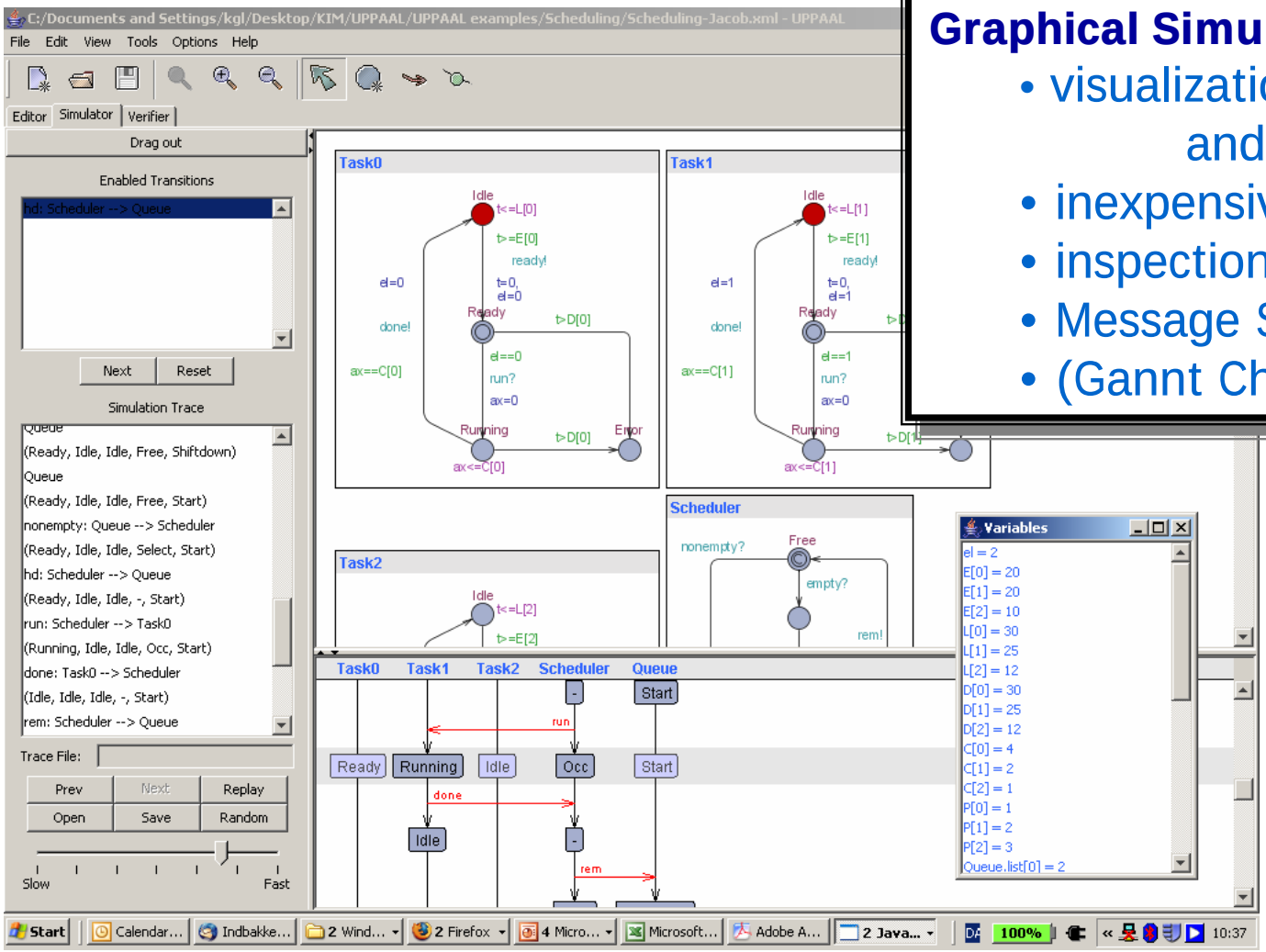


Graphical Design Tool

- timed automata =
  - state machines
    +
  - clocks
- communication
- datatypes
- user defined functions

- cost variable

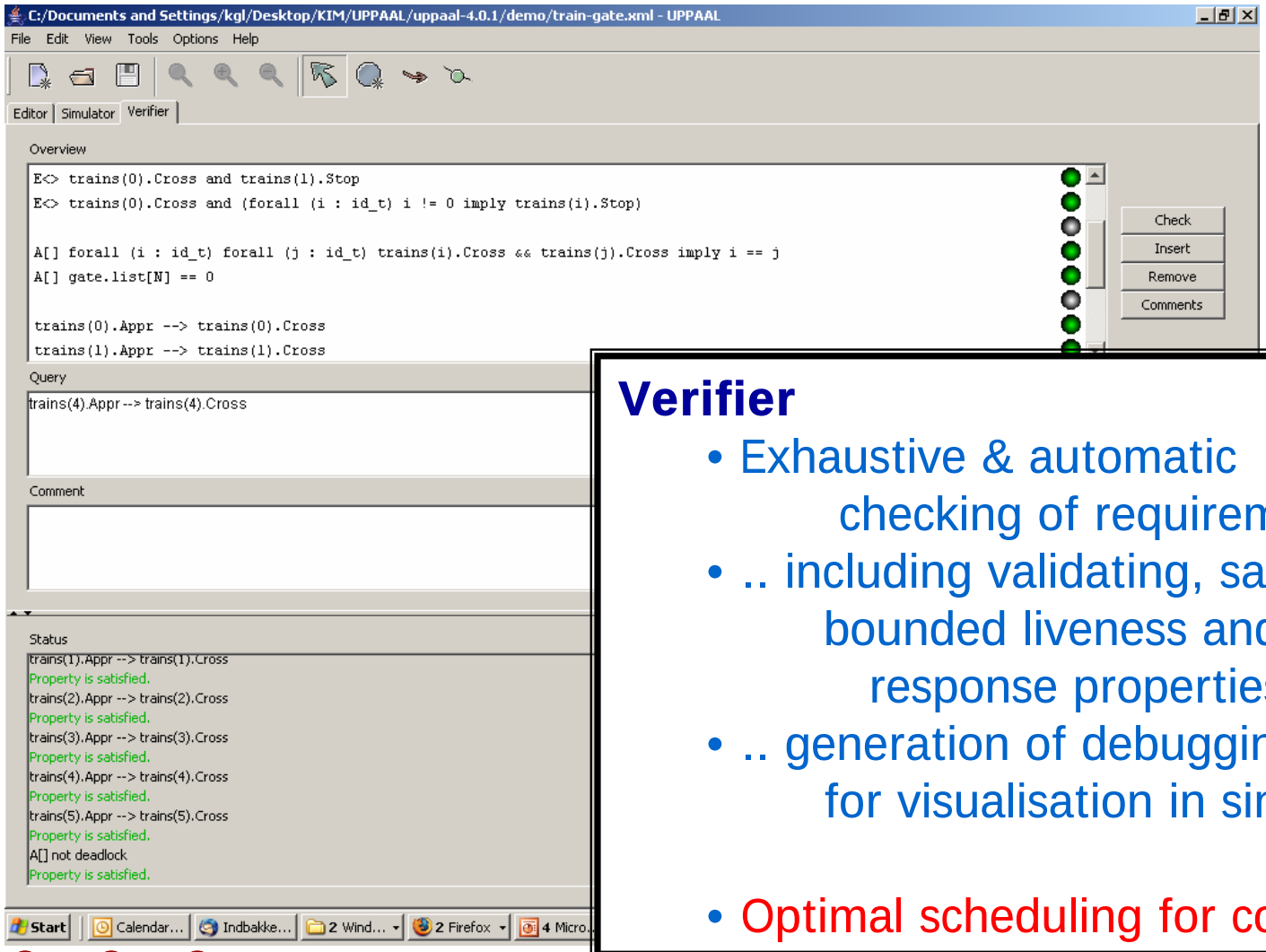
UPPAAL



Graphical Simulator

- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- (Gannt Charts)

UPPAAL

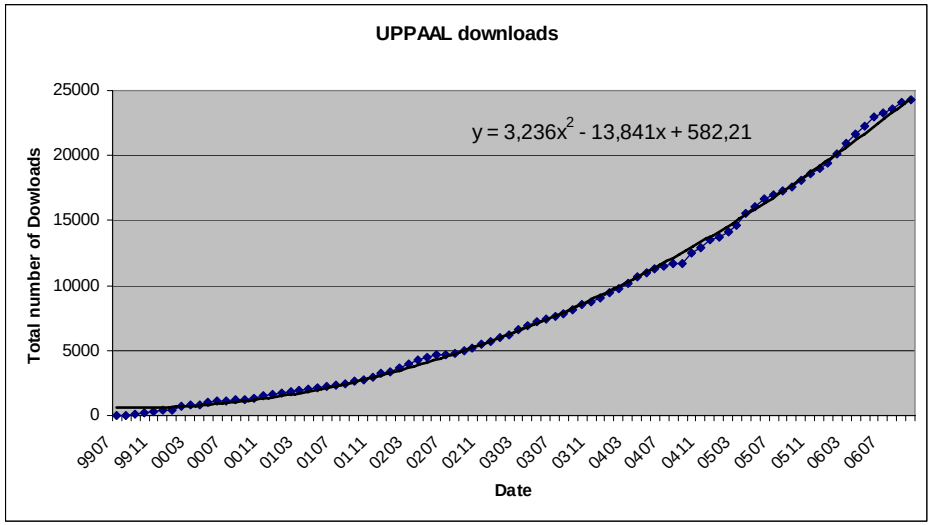


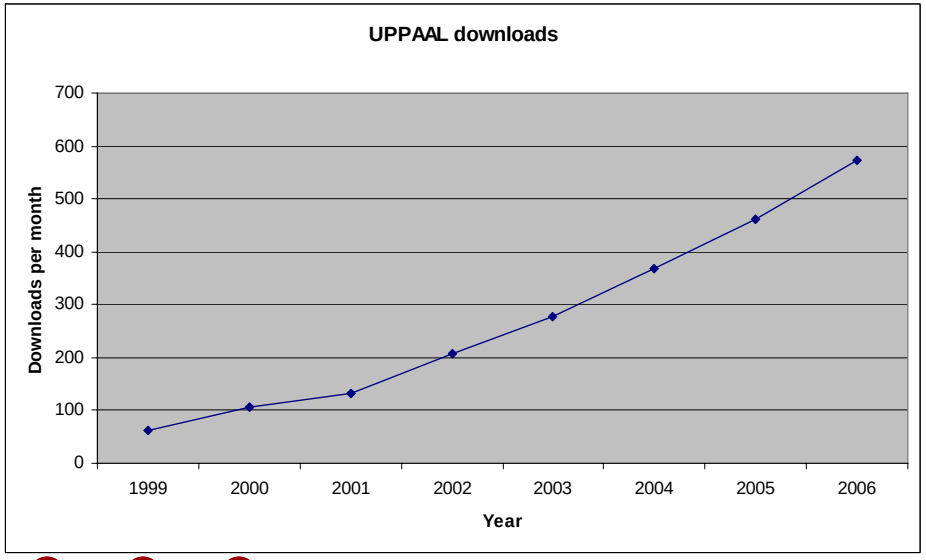
Verifier

- Exhaustive & automatic checking of requirements
- .. including validating, safety, liveness, bounded liveness and response properties
- .. generation of debugging information for visualisation in simulator.

- Optimal scheduling for cost models

“Impact”





Google:

UPPAAL:	134.000
SPIN Verifier:	242.000
nuSMV:	57.700

> 1.500
Google Scholar Citations
(Rhapsody/Esterel < 3.500)

Impact

Academic Courses @

DTU, MCI, IT-U (DK)

Chalmers, Linköping,Lund, Chalmers,

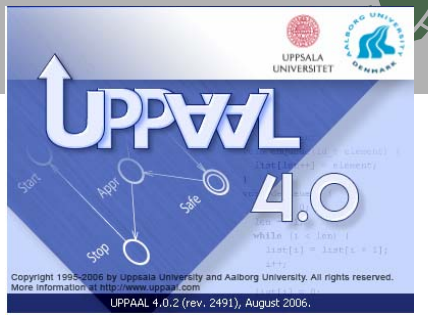
Mälardalarn (S)

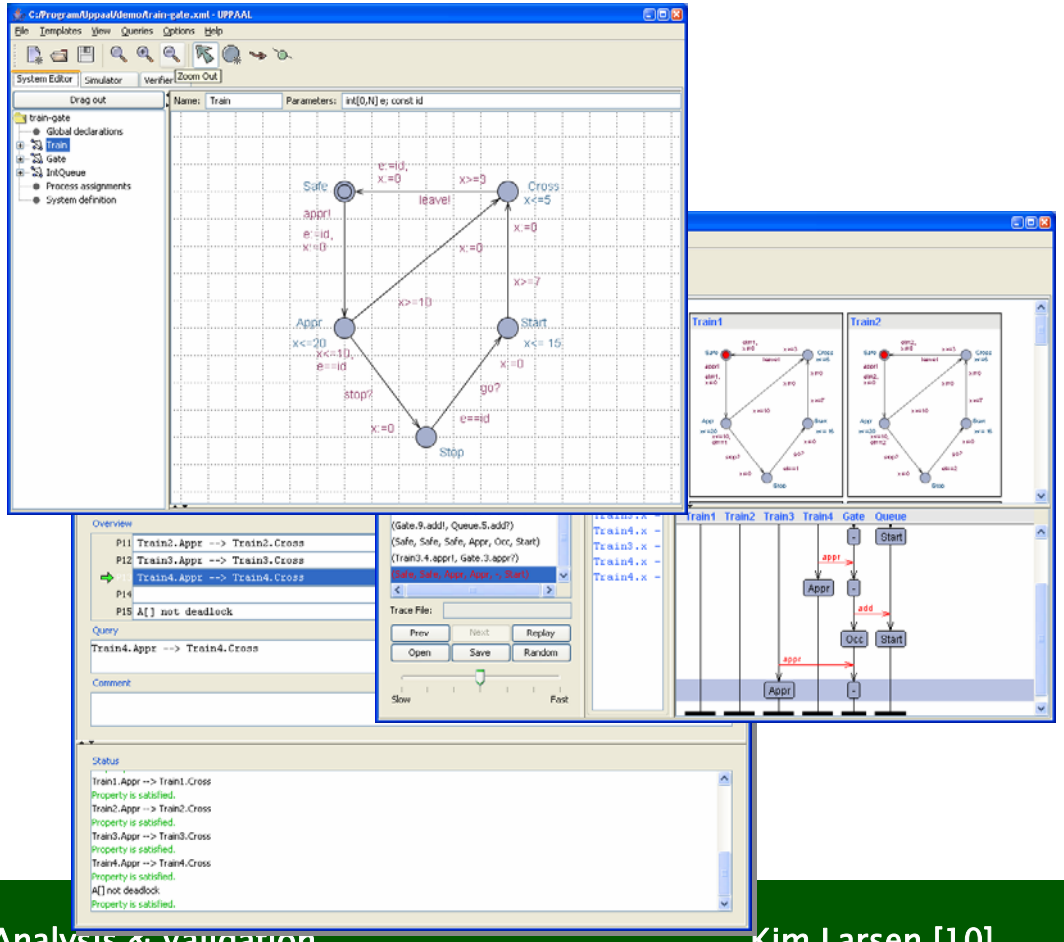
Nijmegen, Twente, CWI (NL)

Upenn, Northumbria(US)

Braunschweig, Oldenborg, Marktoberdorf (D)

Tsinghua, Shanghai, ISS, NUS (Asia)

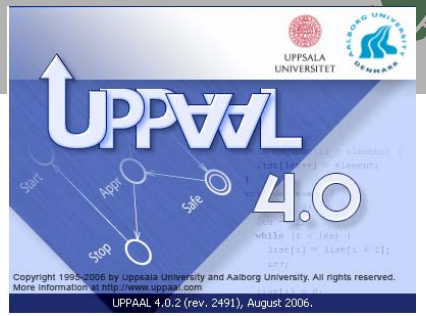


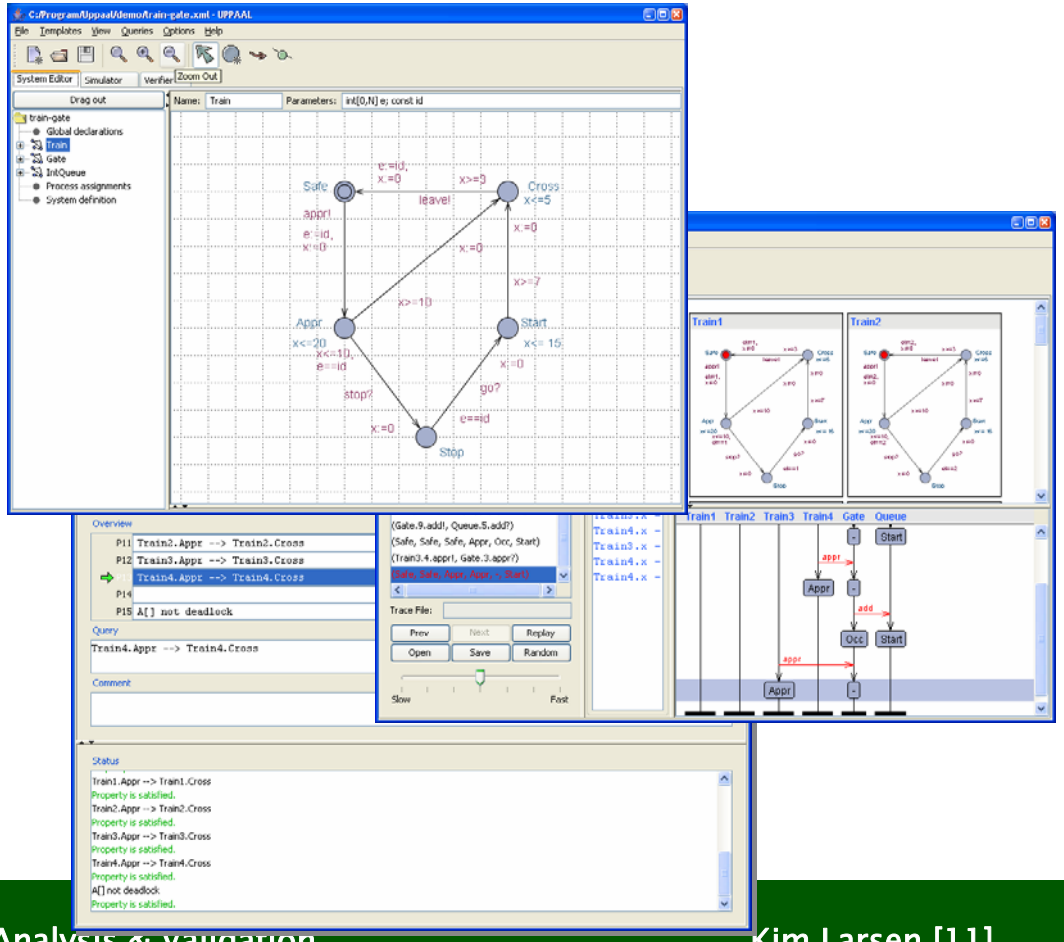


Impact

Company Downloads

Mecel
Jet
Symantec
SRI
Relogic
Realwork
NASA
Verified Systems
Microsoft
ABB
Airbus
PSA
Saab
Siemens
Volvo
Lucent Technologies
Systematic
…

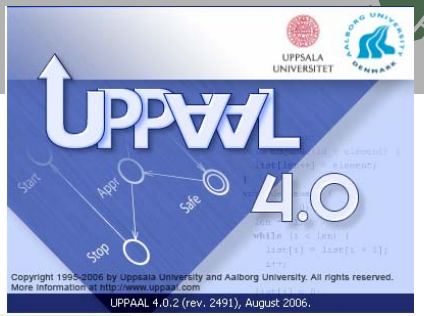


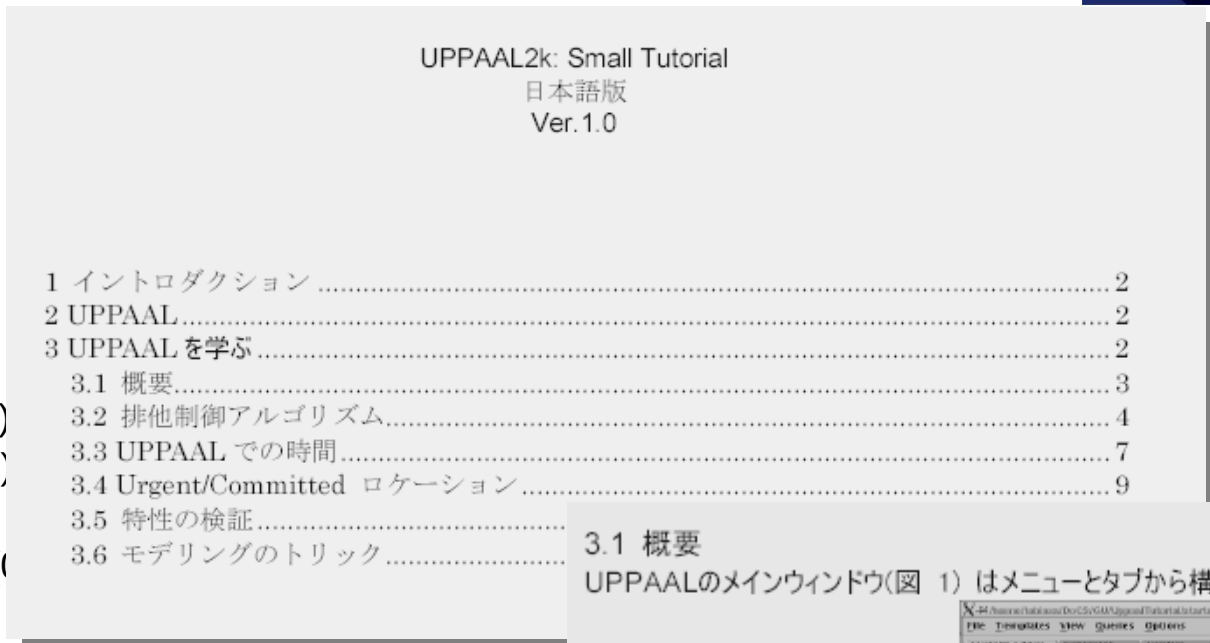


Impact

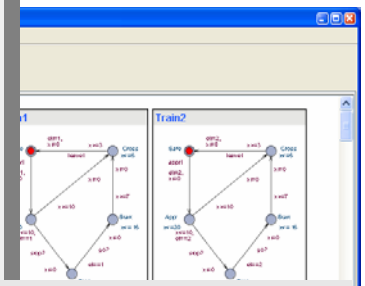
Tutorials Given @

Estonian School (01)
IPA Fall Days (01)
FTRTFT (02)
CPN (02)
SFM (02)
MOVEP (02)
DISC School
MOVEP (04)
PRISE (04)
PDMC (05)
ARTIST2 (05)
EMSOFT (05)
RTSS (05)
TECS week (
TAROT (06)
ARTS (06)
GLOBAN (06)
ARTIST ASIAN SCH (07)
…….




UPPAAL2k: Small Tutorial
日本語版
Ver.1.0

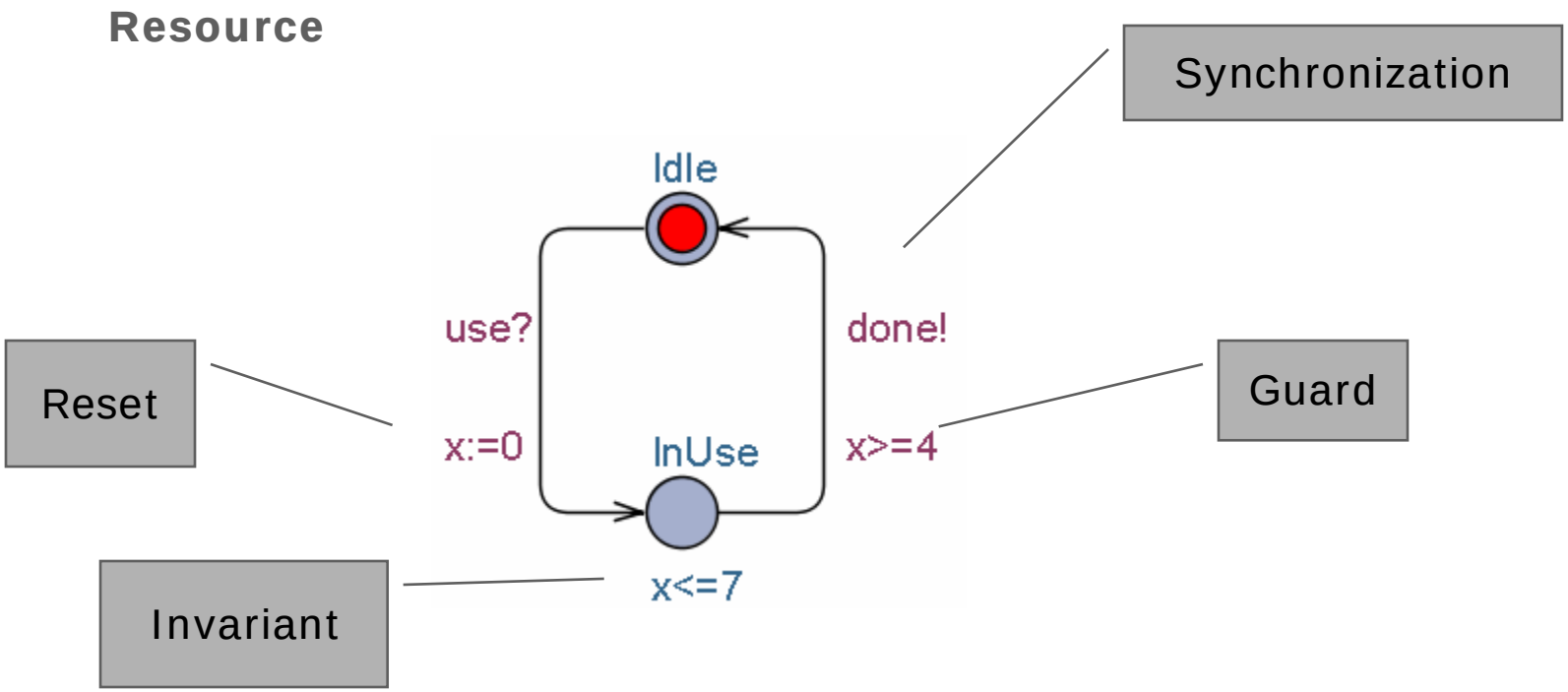
1 イントロダクション 2
2 UPPAAL 2
3 UPPAAL を学ぶ 2
3.1 概要 3
3.2 排他制御アルゴリズム 4
3.3 UPPAAL での時間 7
3.4 Urgent/Committed ロケーション 9
3.5 特性の検証 .....
3.6 モデリングのトリック .....




3.1 概要
UPPAALのメインウィンドウ(図 1) はメニューとタブから構成されています。

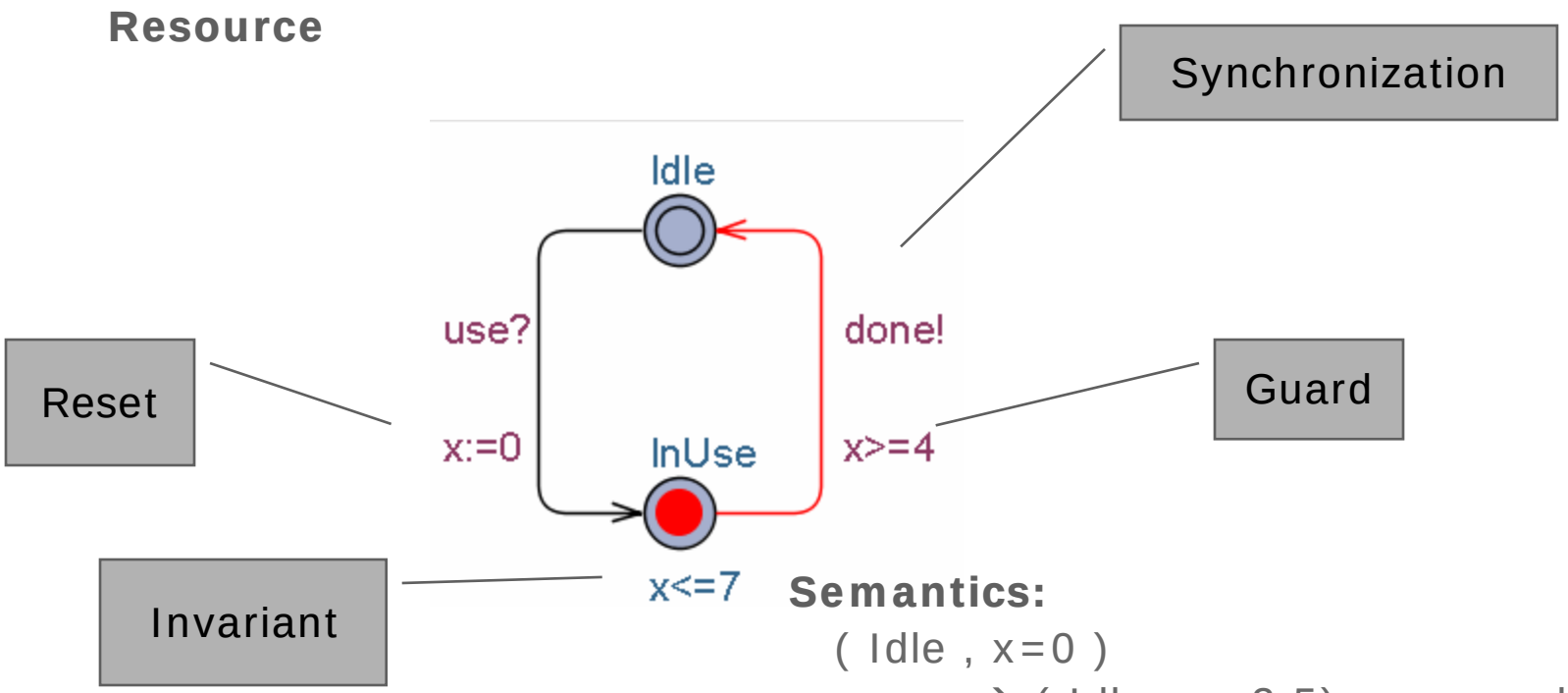
Timed Automata

[Alur & Dill'89]



Timed Automata

[Alur & Dill'89]

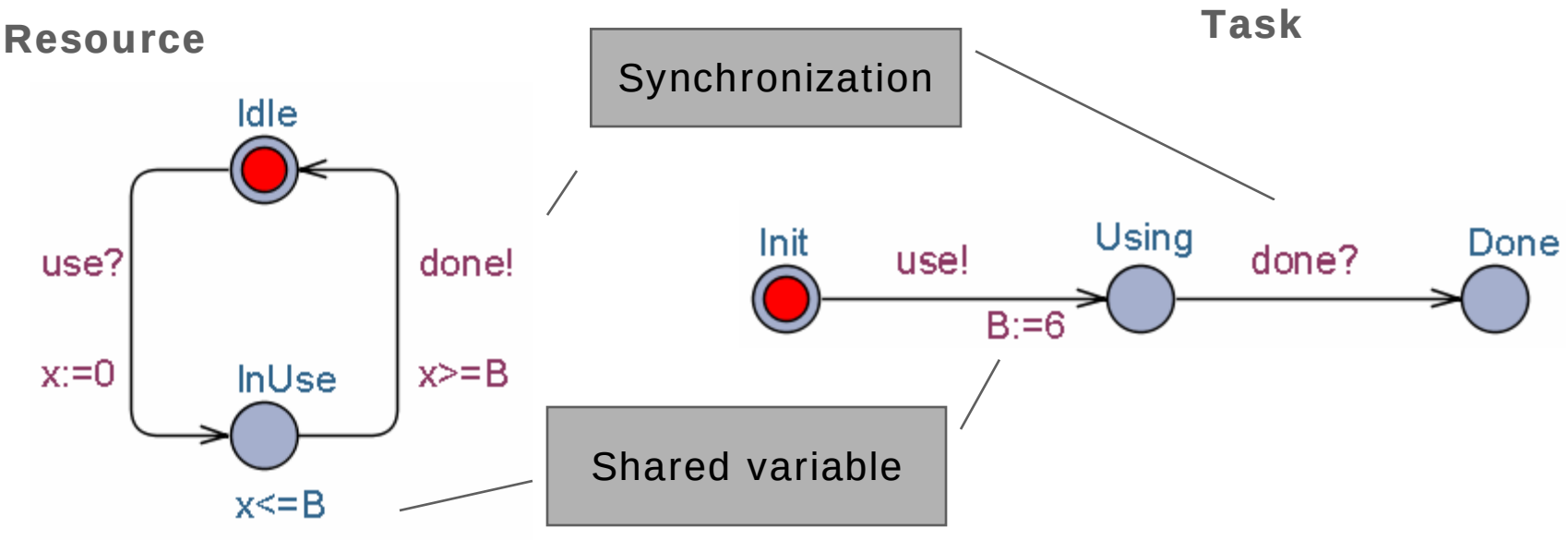


Semantics:

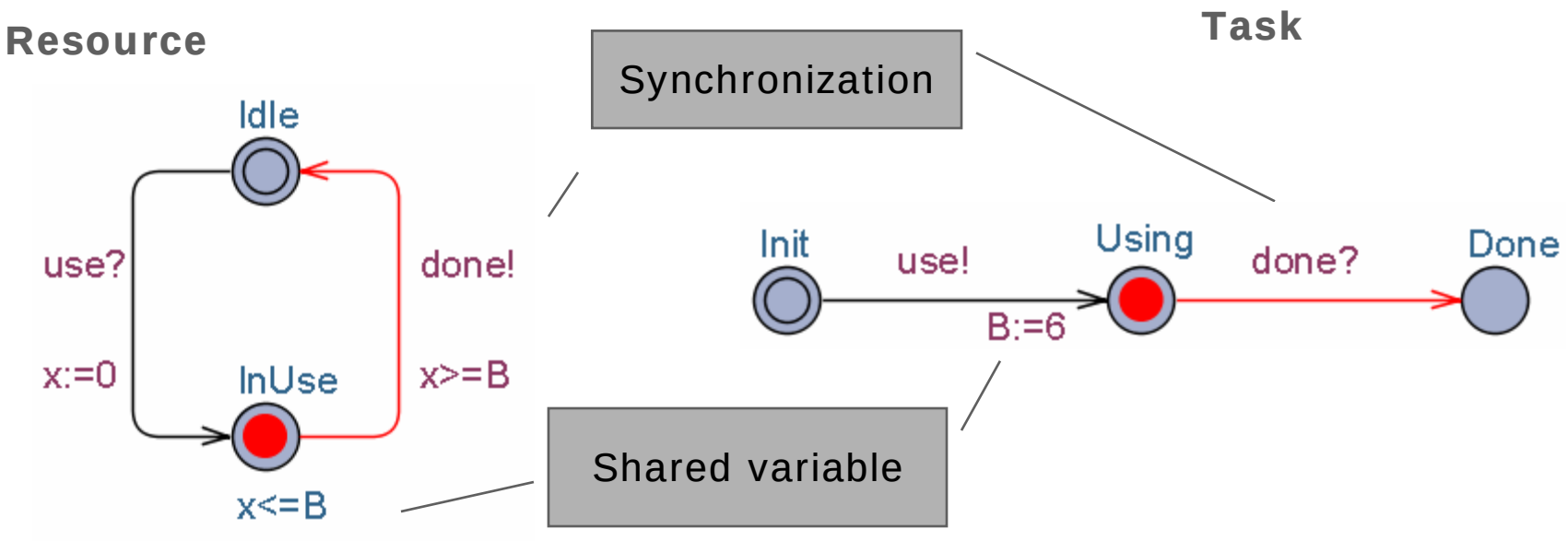
(Idle , x=0)

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

Composition



Composition



Semantics:

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

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

→ (InUse , Using , B=6, x=0) use

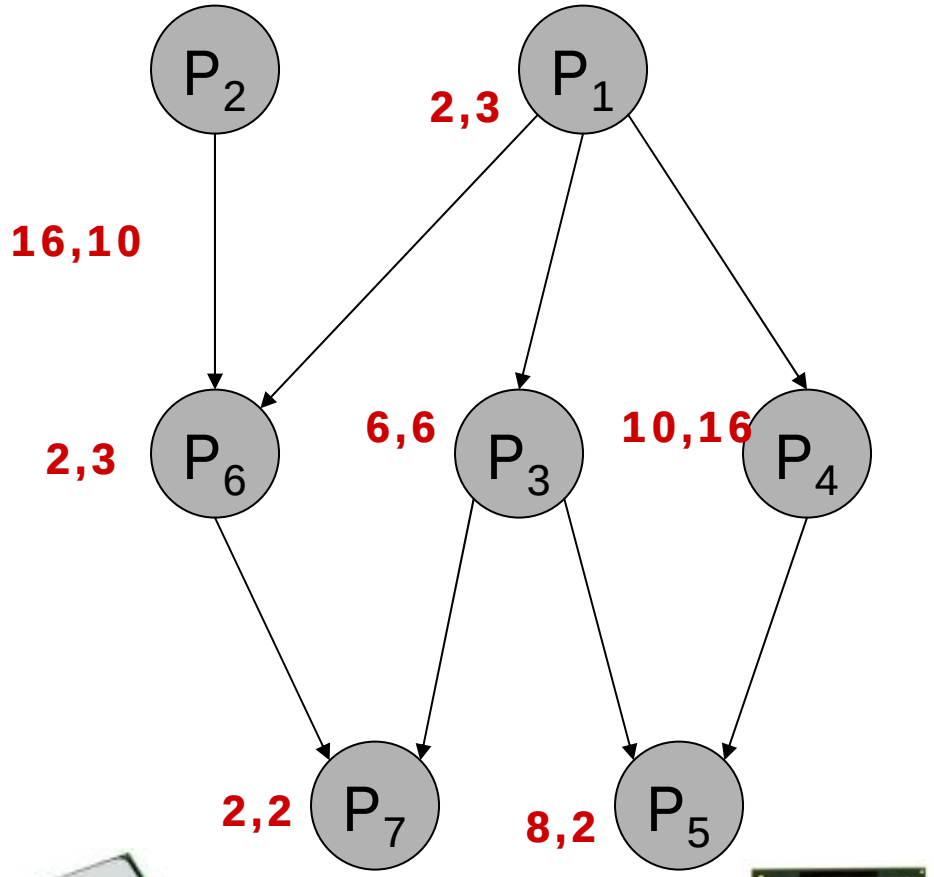
→ (InUse , Using , B=6, x=6) d(6)

→ (Idle , Done , B=6 , x=6) done

Task Graph Scheduling

Optimal Static Task Scheduling

- Task **P**={P_1,.., P_m}
- Machines **M**={M_1,..,M_n}
- Duration Δ : **(P**×**M)** → $\mathbf{N}_\infty$
- < : p.o. on **P** (pred.)



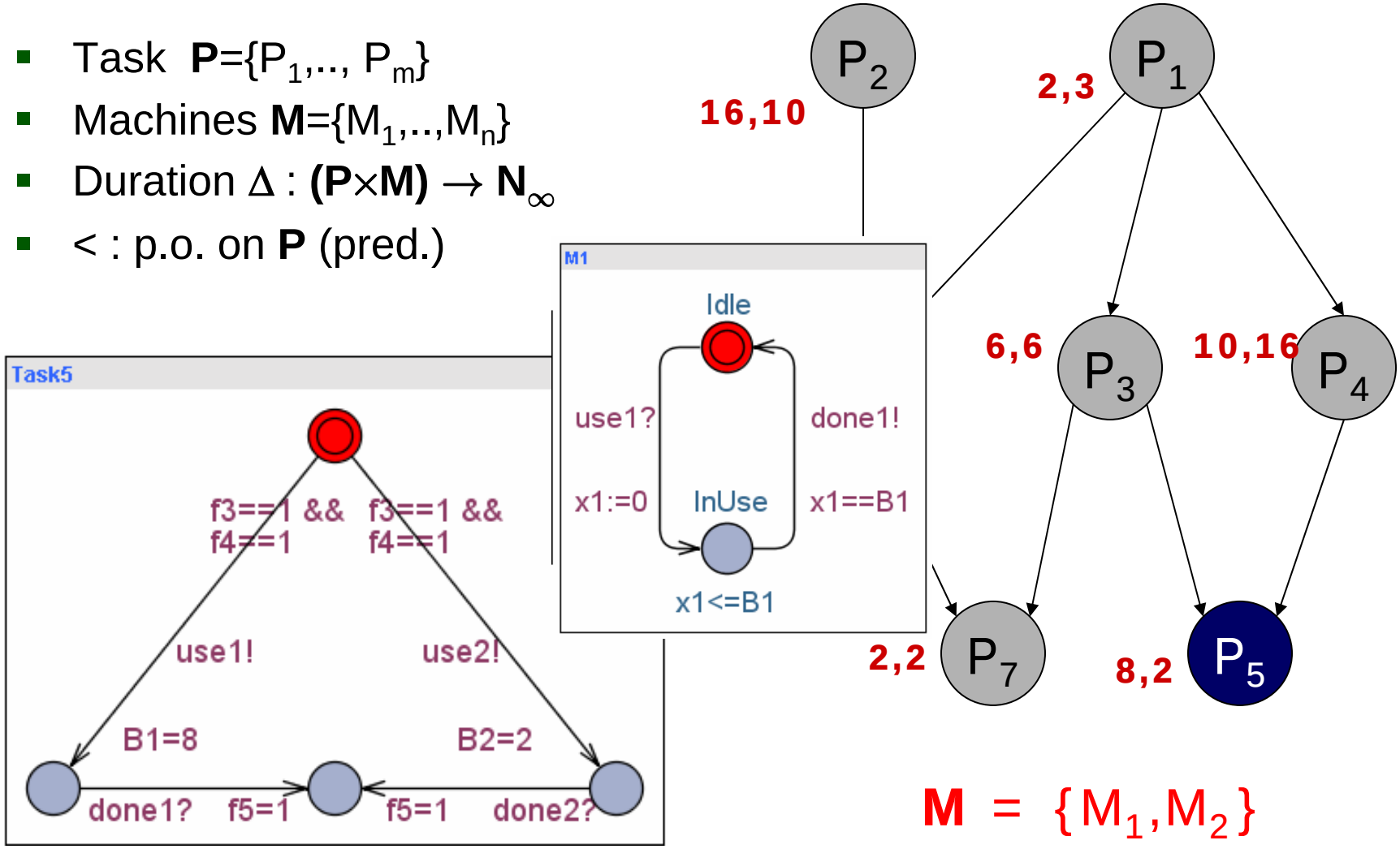
- A task can be executed only if all predecessors have completed
- Each machine can process at most one task at a time
- Task cannot be preempted.

M = {M_1,M_2}

Task Graph Scheduling

Optimal Static Task Scheduling

- Task **P**={P_1,.., P_m}
- Machines **M**={M_1,..,M_n}
- Duration Δ : **(P**×**M)** → $\mathbf{N}_\infty$
- < : p.o. on **P** (pred.)



M = {M_1,M_2}

Experimental Results

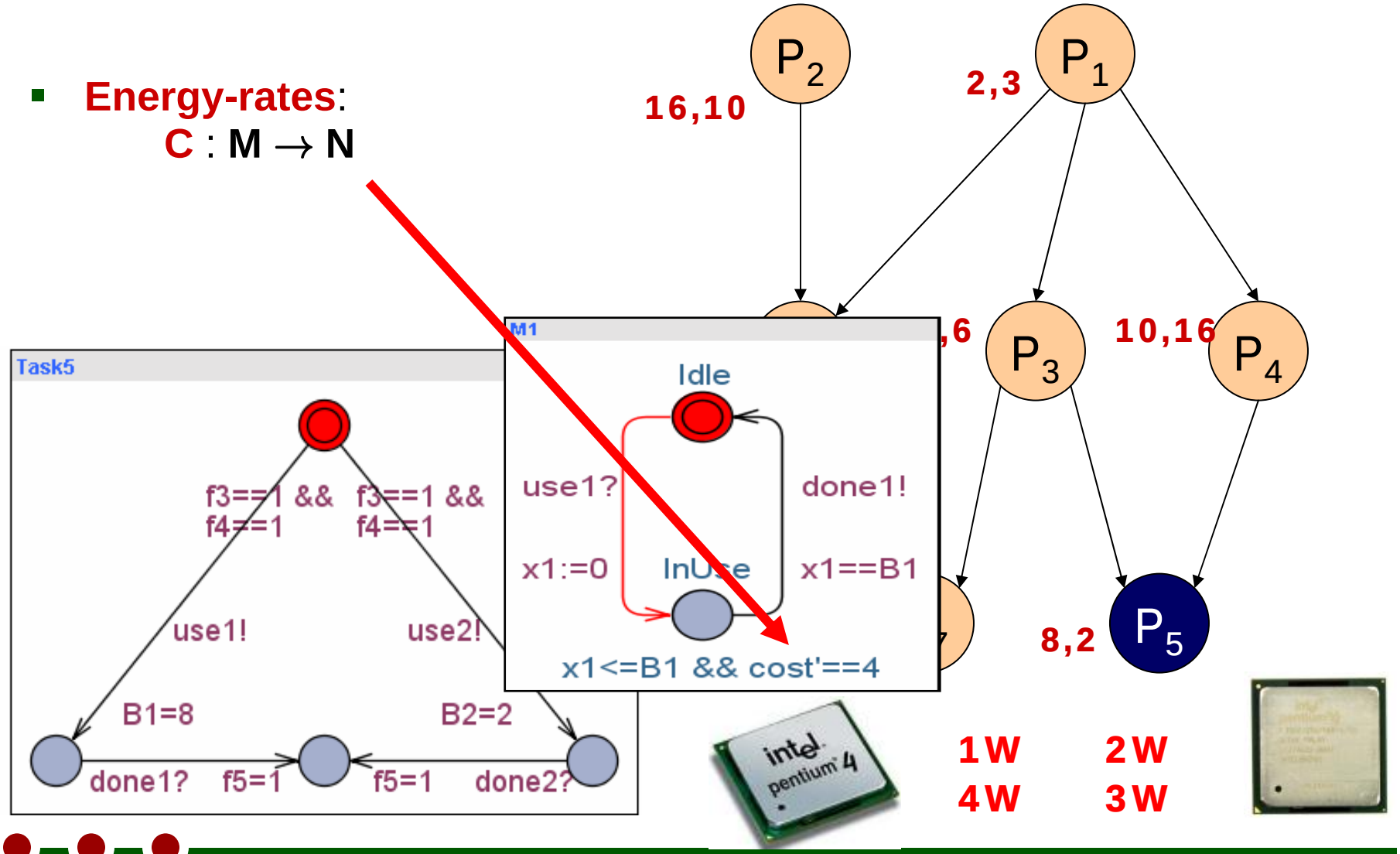
name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.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

Abdeddaïm, Kerbaa, Maler

Optimal Task Graph Scheduling

Power-Optimality

- **Energy-rates**:
  C : **M** → **N**



Dynamic Voltage Scaling

ANALOG DEVICES

DaNES

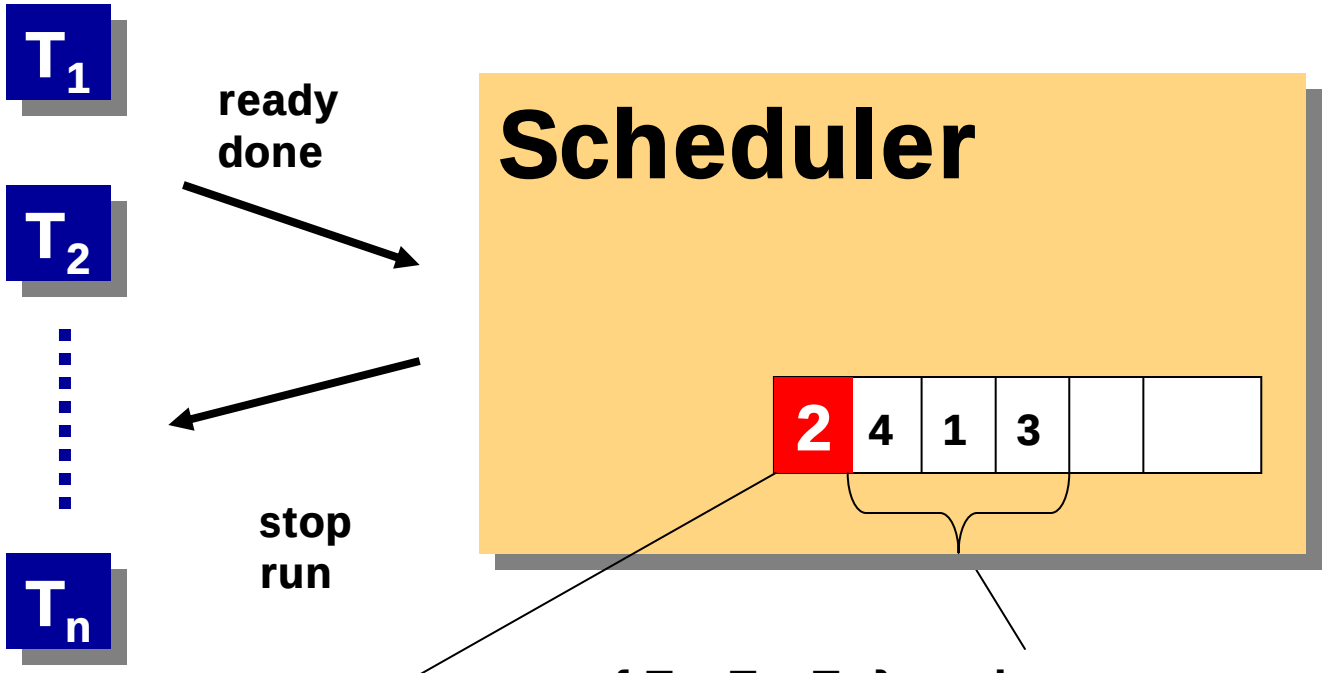
CISS

CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

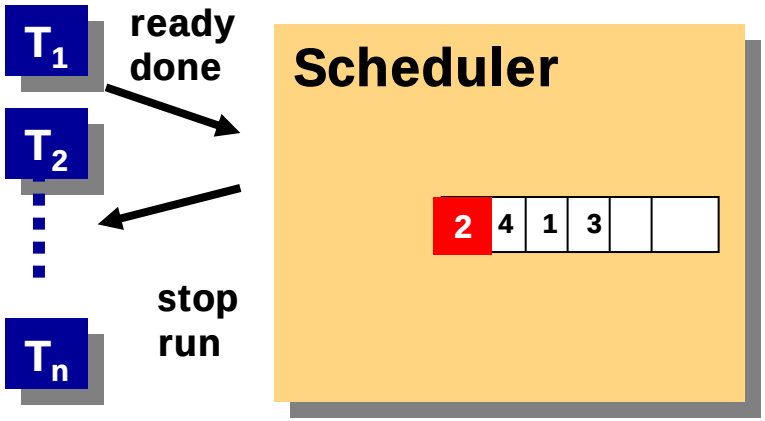
Task Scheduling

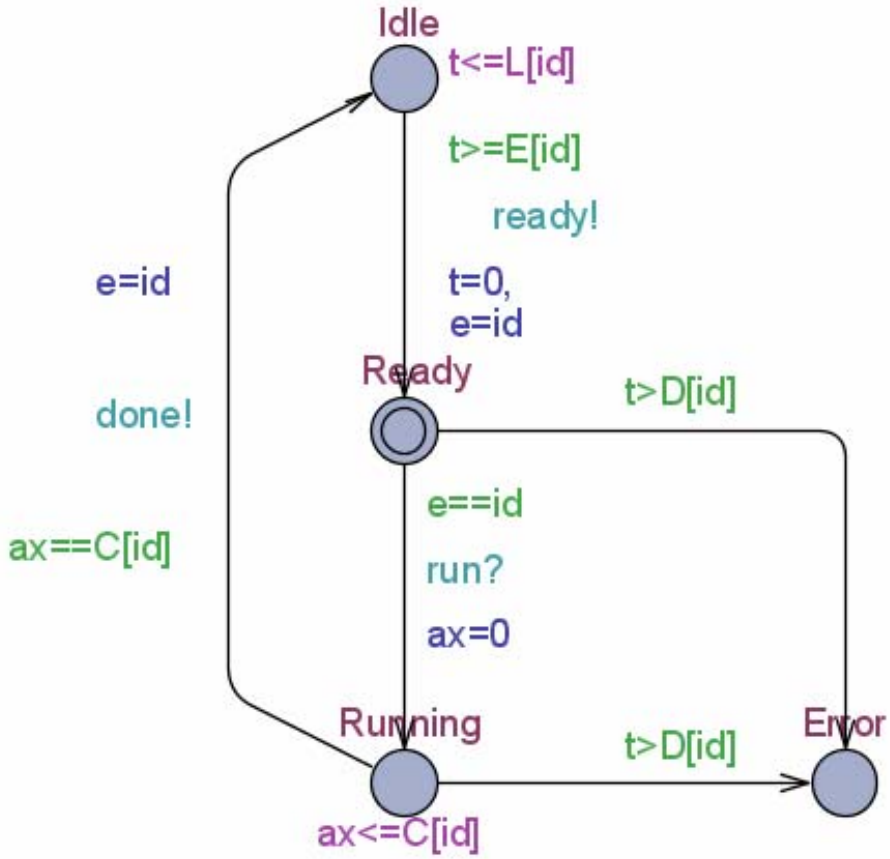
utilization of CPU

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

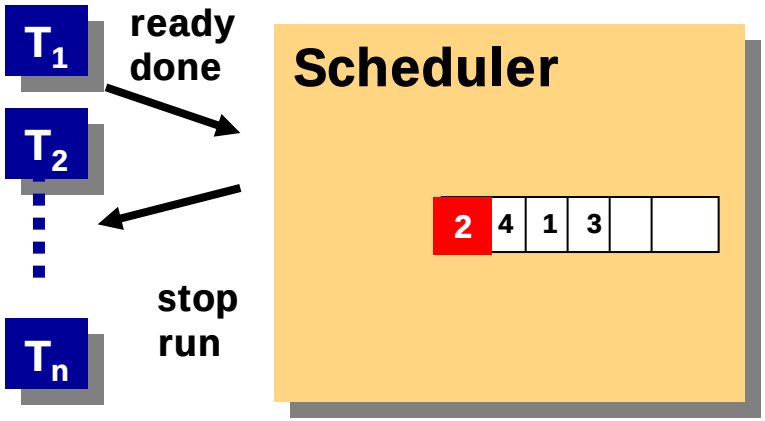


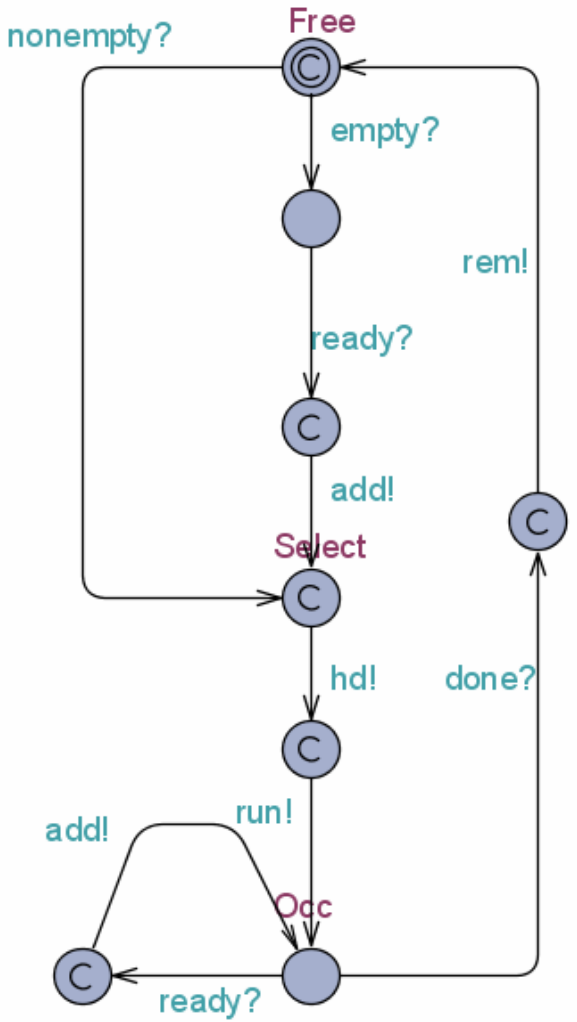
Modeling Task





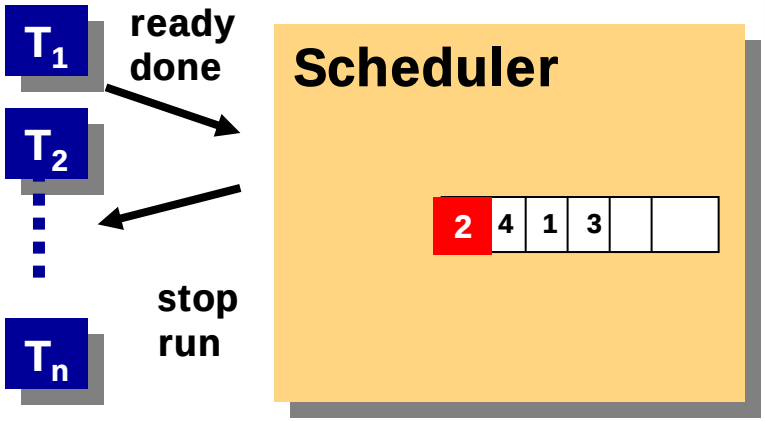
Modeling Scheduler





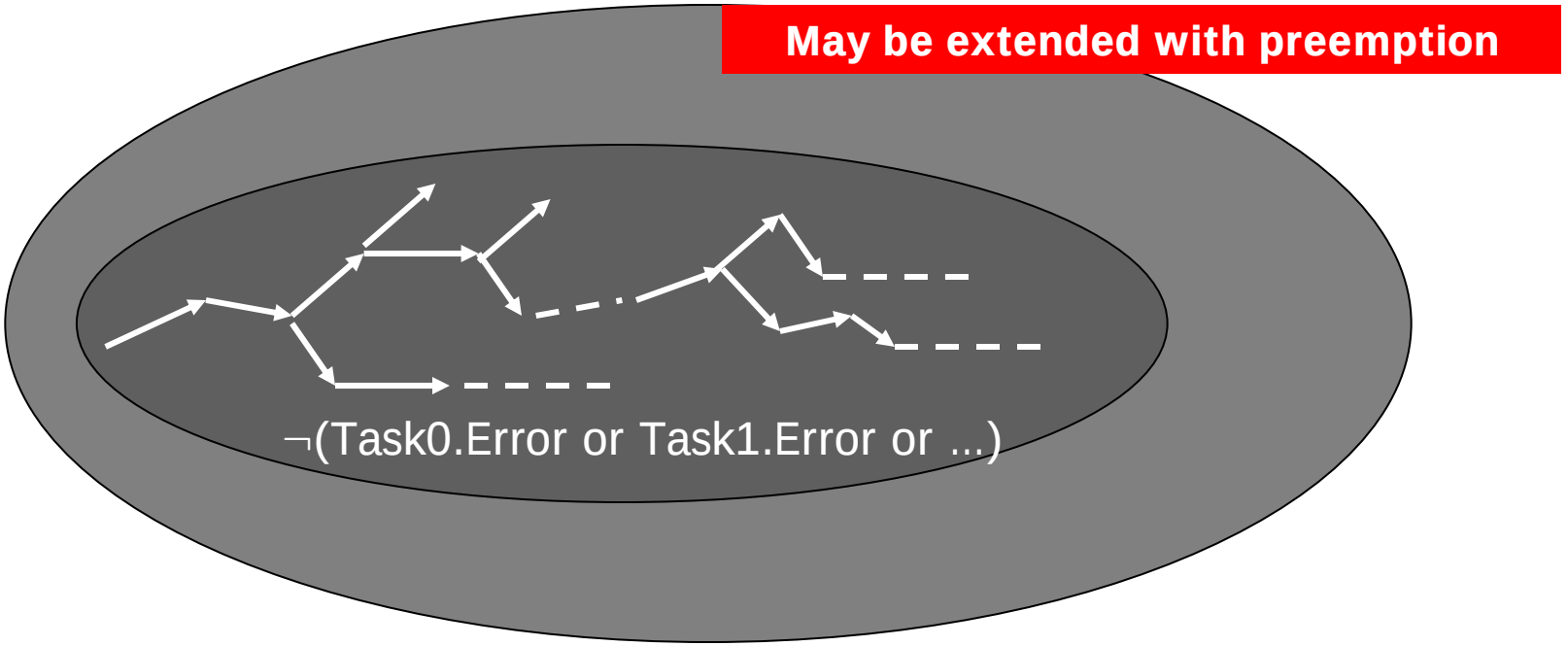
Modeling Queue

In UPPAAL 4.0
User Defined Function




```
void add(id_t element)
{
int i=len;
int temp=0;
if(len==0)
{       list[len]=e;
len=1;
}
   else
{       list[len]=e;
        i=len;
        len=len+1;
        while (i>1 && P[list[i]]> P[list[i-1]])
        {       temp=list[i-1];
                list[i-1]=list[i];
                list[i]=temp;
                i=i-1;
        }
}
}
```

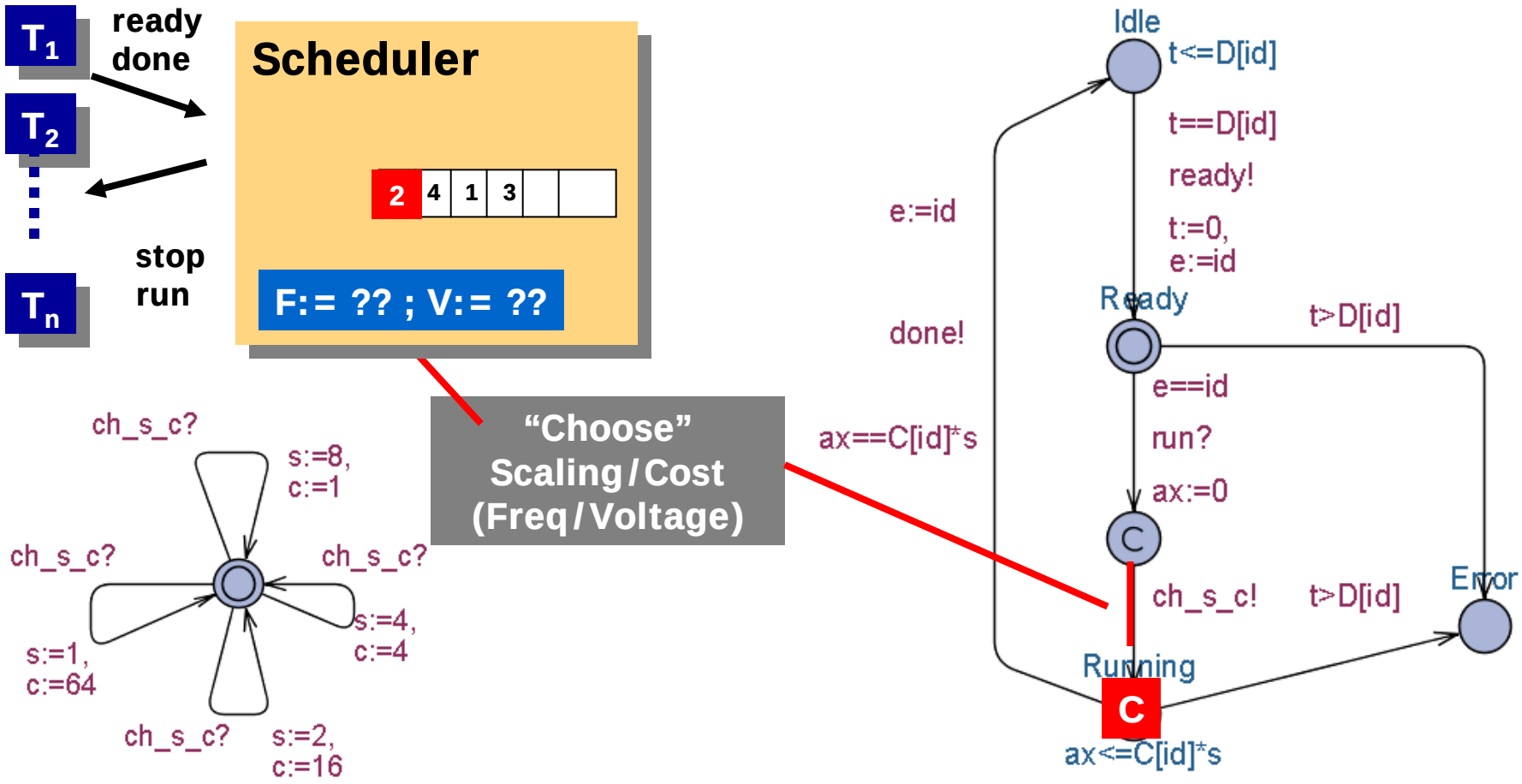
Schedulability = Safety Property



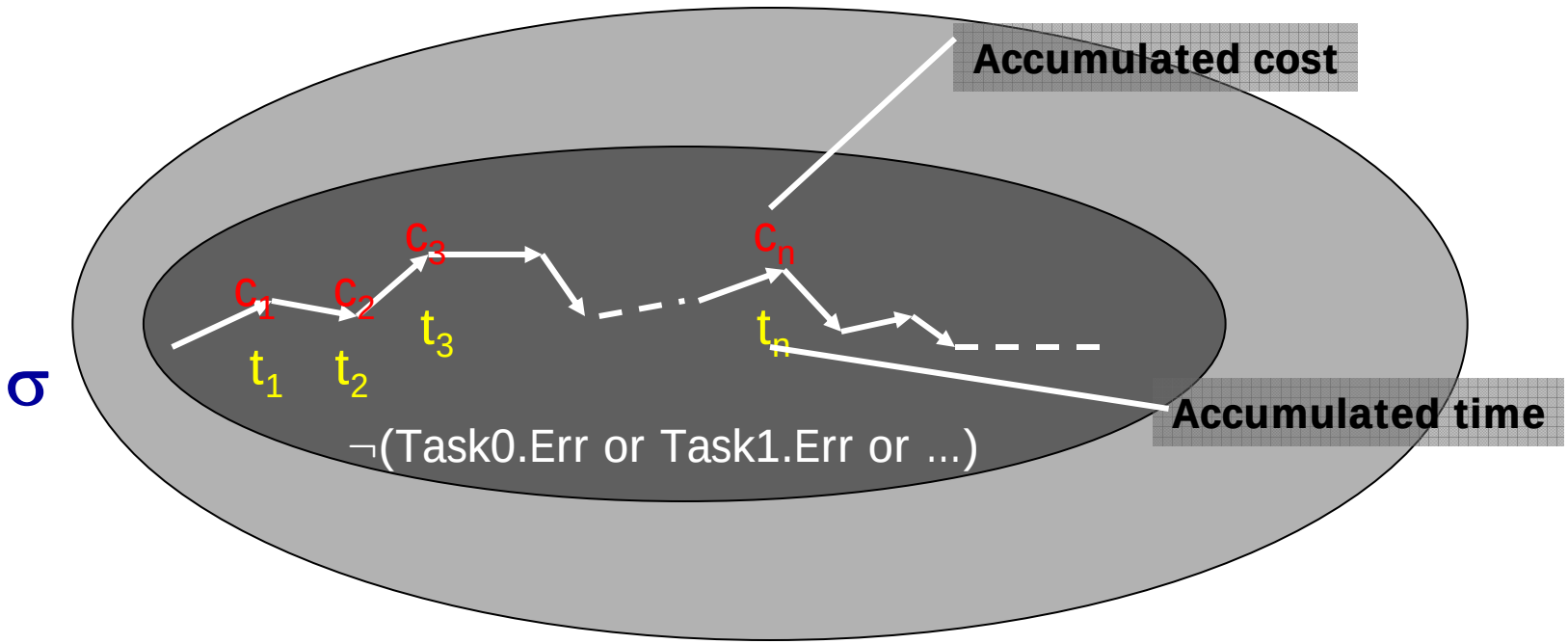
A□ ¬(Task0.Error or Task1.Error or ...)

Energy Optimal Scheduling

Using Priced Timed Automata



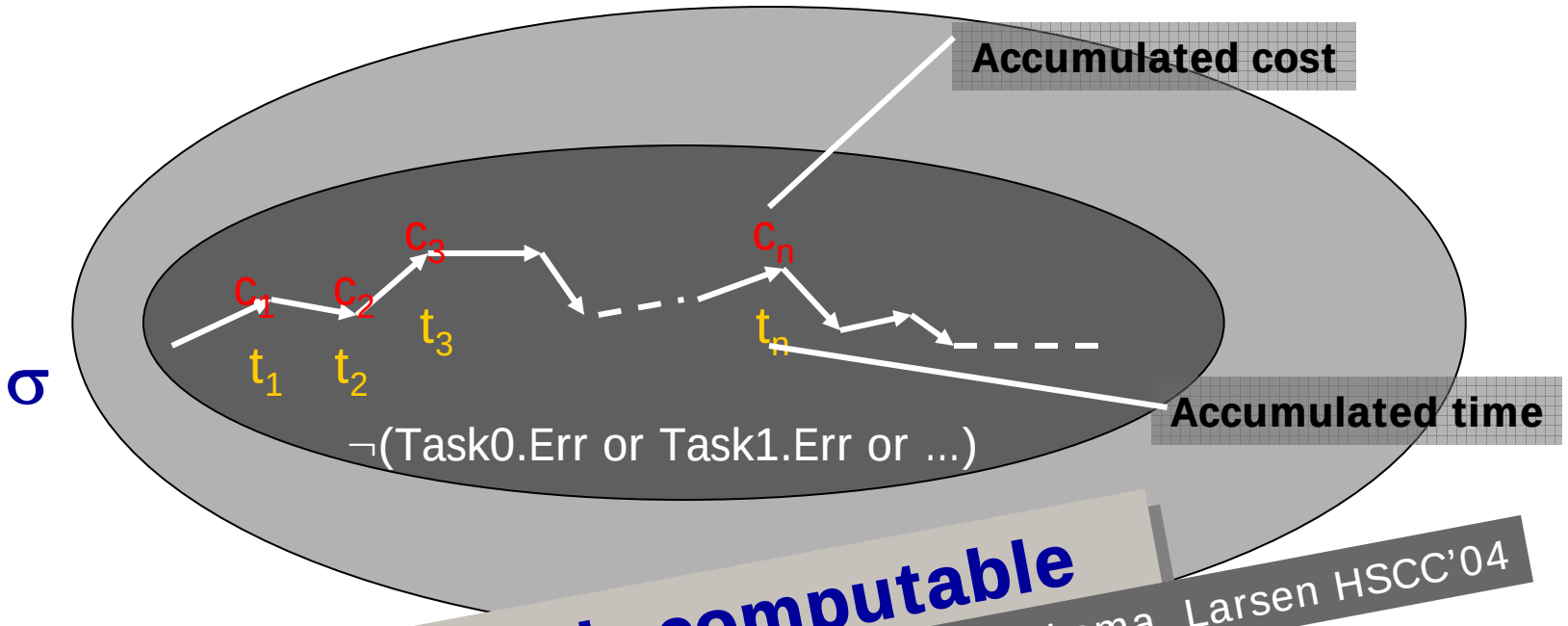
Cost Optimal Scheduling = *Optimal Infinite Path*



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

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

Cost Optimal Scheduling = *Optimal Infinite Path*



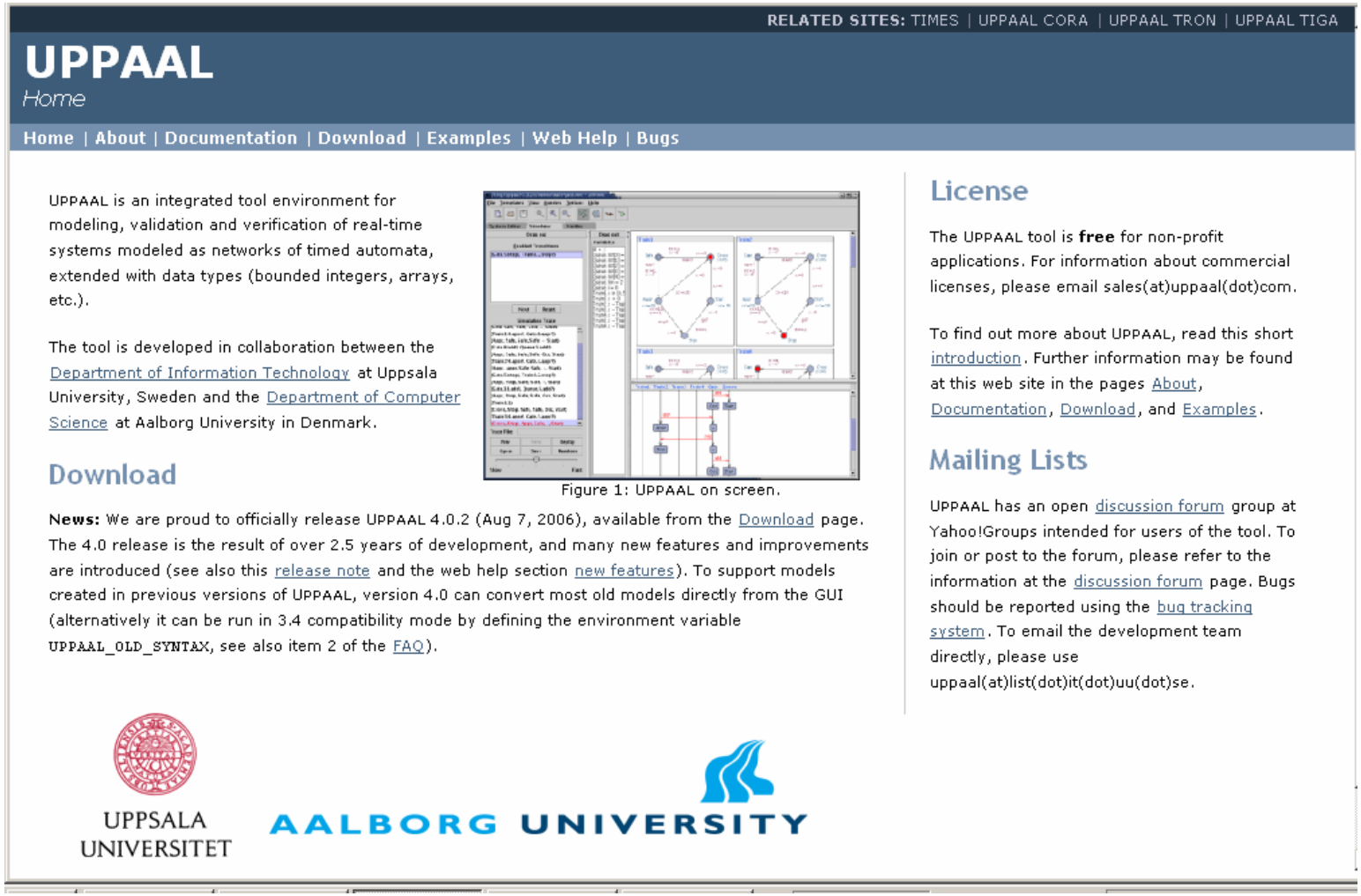
THEOREM: σ^* is computable

Bouyer, Brinksma, Larsen HSCC'04

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

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

Contact: www.uppaal.com



RELATED SITES: TIMES | UPPAAL CORA | UPPAAL TRON | UPPAAL TIGA

UPPAAL

Home

Home | About | Documentation | Download | Examples | Web Help | Bugs

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

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

Figure 1: UPPAAL on screen.

Download

News: We are proud to officially release UPPAAL 4.0.2 (Aug 7, 2006), available from the Download page. The 4.0 release is the result of over 2.5 years of development, and many new features and improvements are introduced (see also this release note and the web help section new features). To support models created in previous versions of UPPAAL, version 4.0 can convert most old models directly from the GUI (alternatively it can be run in 3.4 compatibility mode by defining the environment variable `UPPAAL_OLD_SYNTAX`, see also item 2 of the FAQ).

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

License

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

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

Mailing Lists

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

Formalizing the ARTS MPSoC Model in UPPAAL

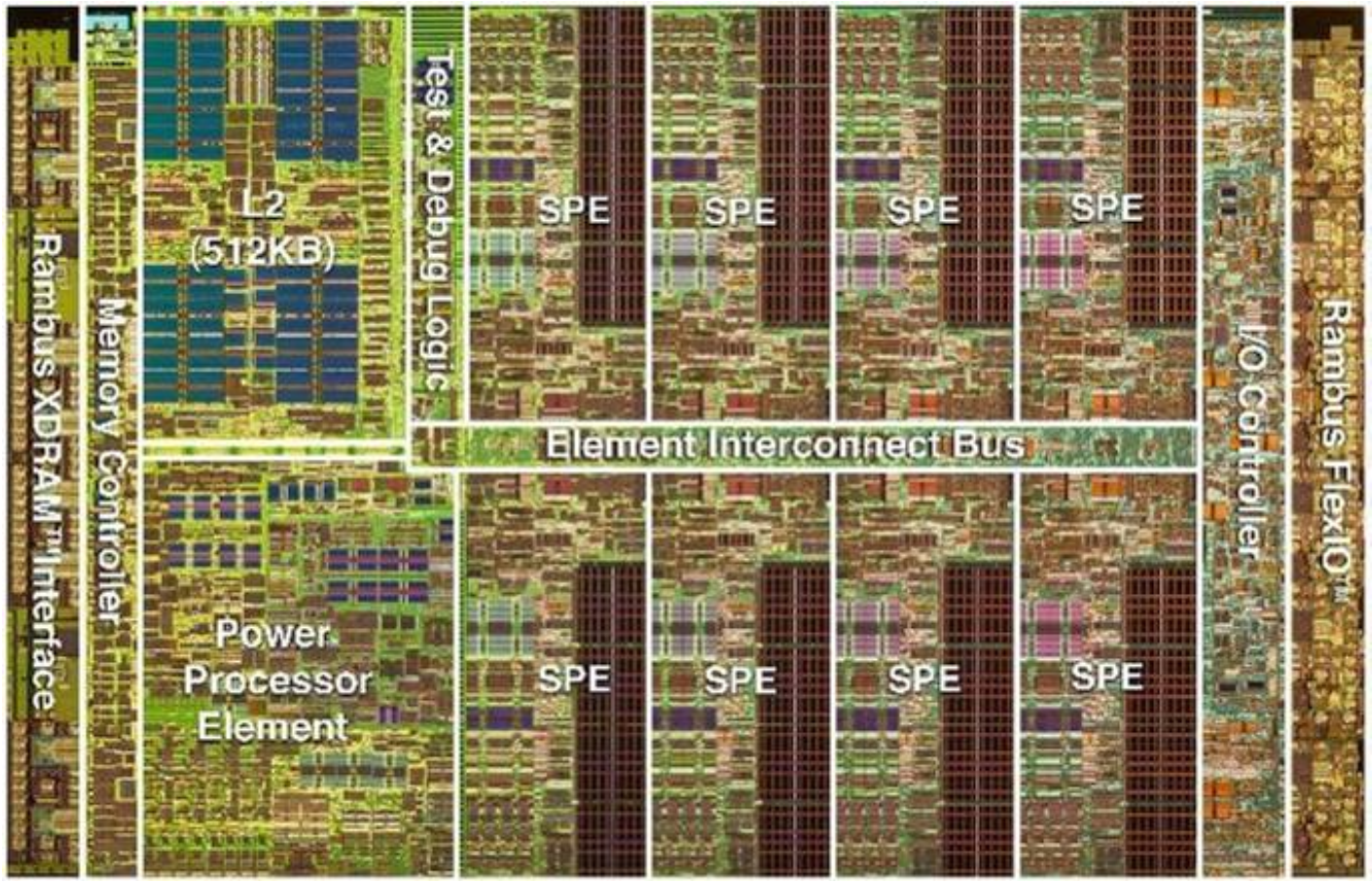
Jan Madsen and Michael R. Hansen

Embedded Systems Engineering Group

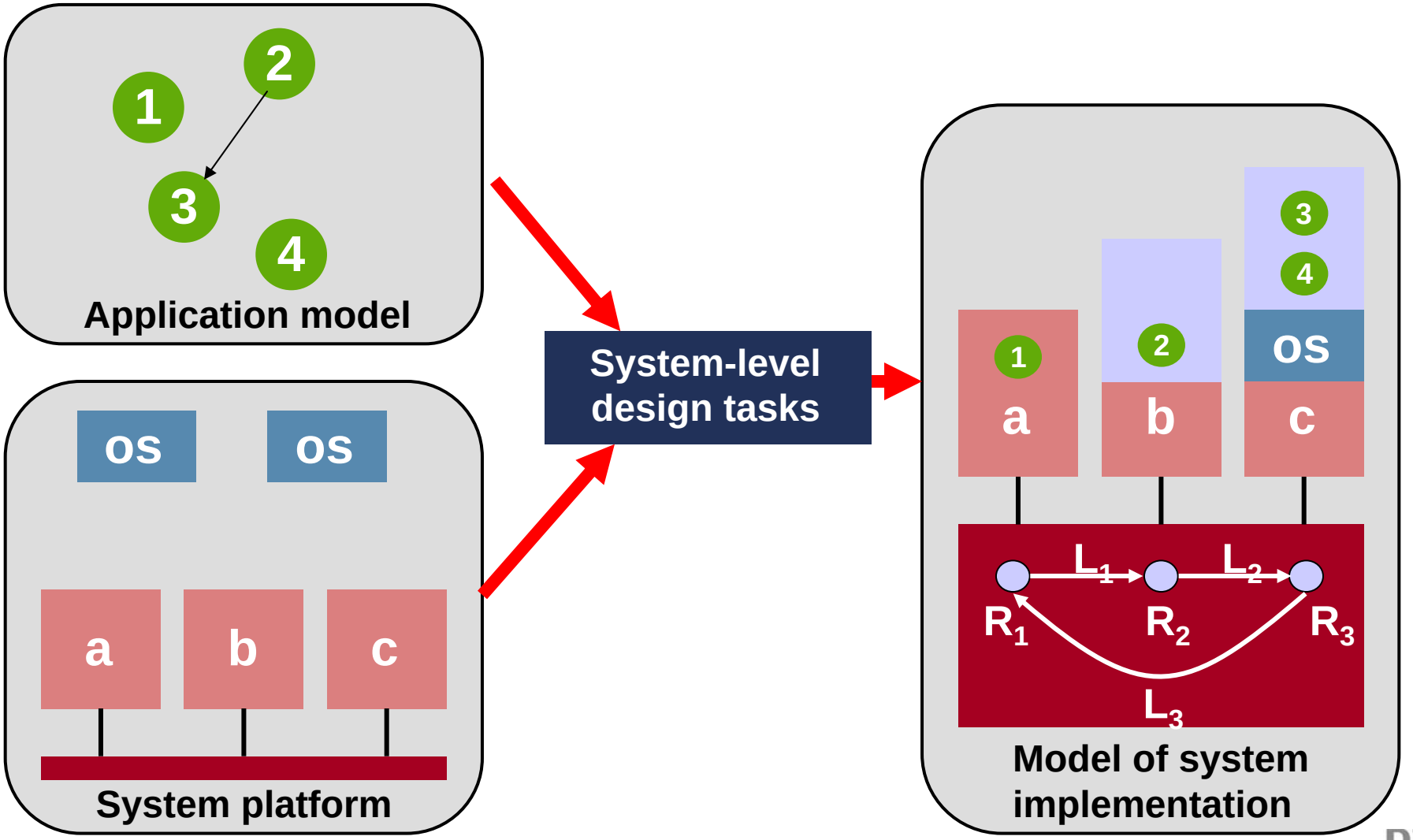
Informatics and Mathematical Modeling
Technical University of Denmark

Motivation

CELL processor

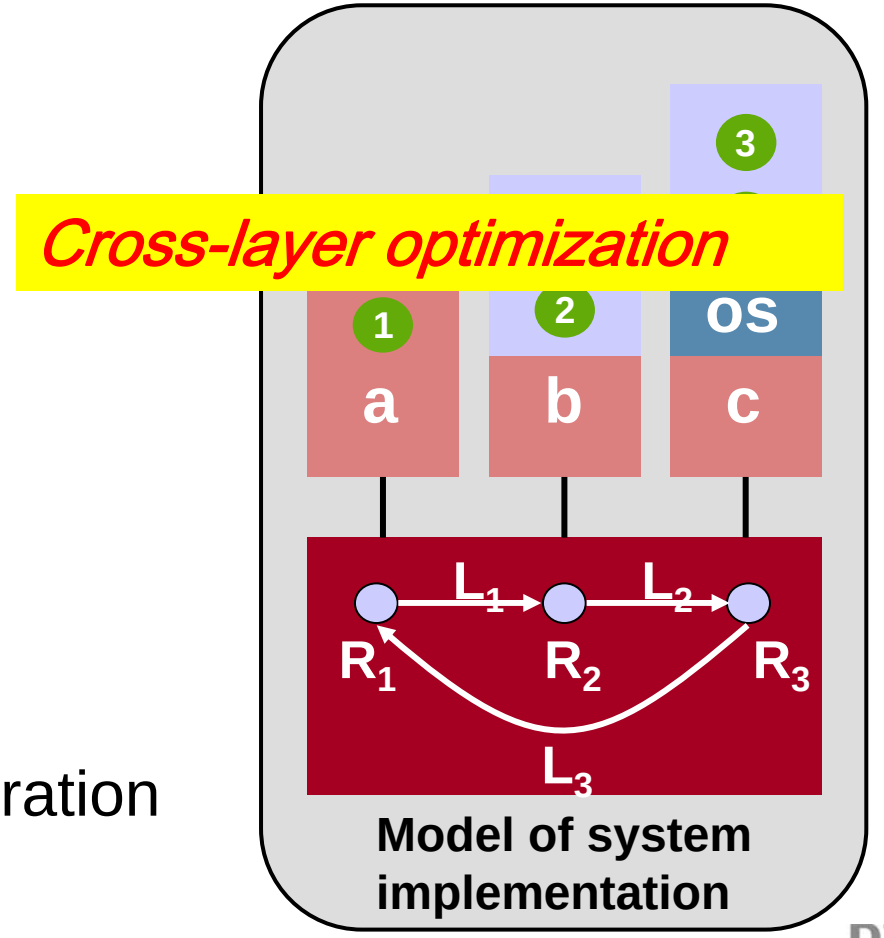


Motivation

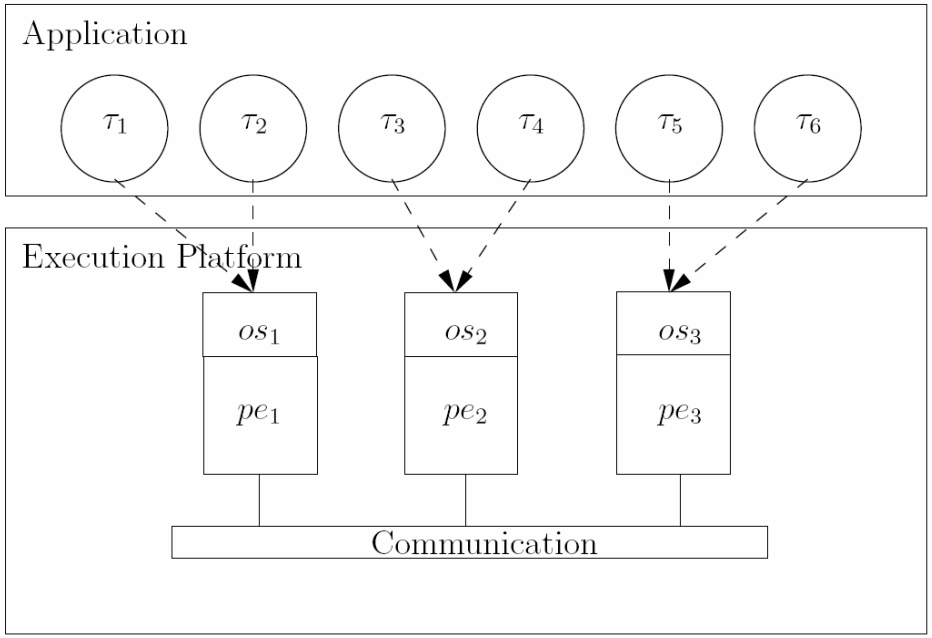

1
2
3
4
Application model
OS
OS
a
b
c
System platform
System-level design tasks
1
2
3
4
OS
a
b
c
L_1
L_2
L_3
R_1
R_2
R_3
Model of system implementation

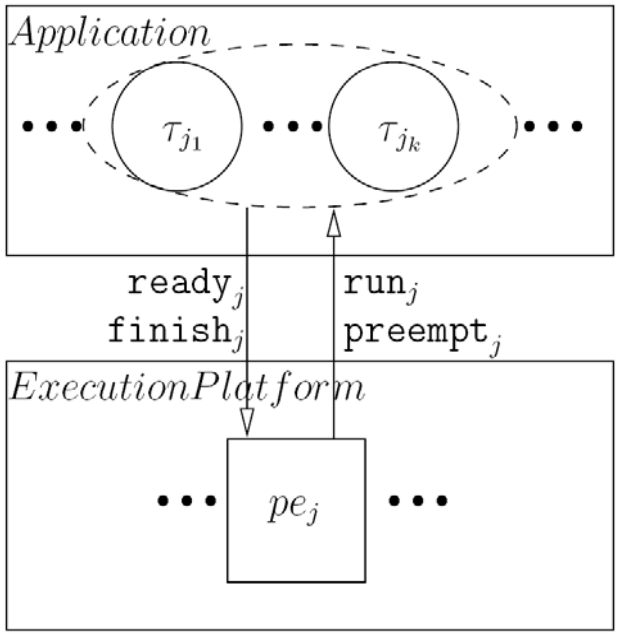
ARTS objectives

- System-level modeling framework
- Bridging,
  - Application
  - RTOS
  - Execution platform
    - Processing elements
    - NoC
- Supporting
  - System-level analysis
  - Early design space exploration



Formalizing ARTS



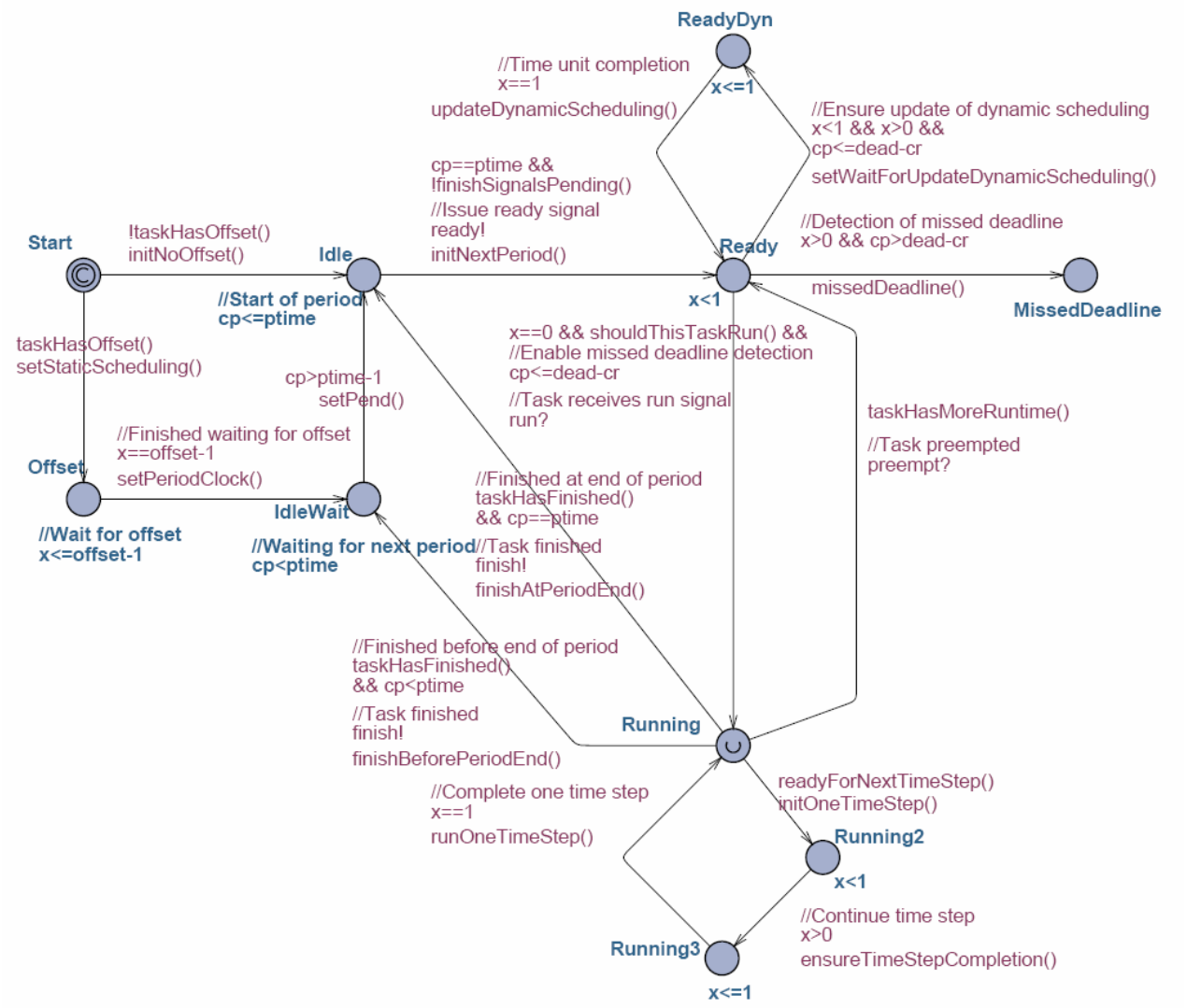


$$System = Application \parallel ExecutionPlatform$$

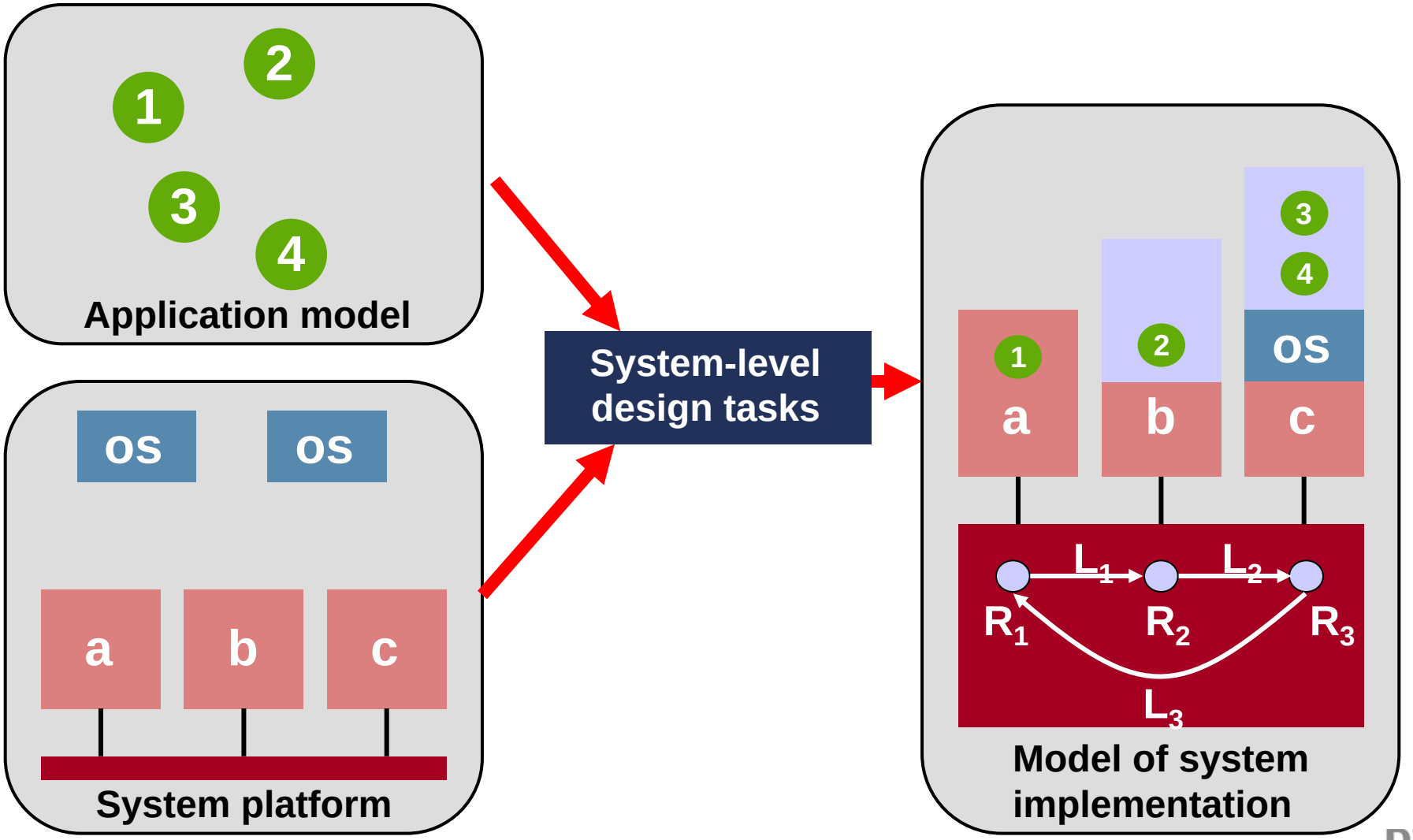
$$Application = \|_{i=1}^{n} \tau_i$$

$$ExecutionPlatform = \|_{j=1}^{m} pe_j$$

Timed Automata for a task



MOVES: Hiding UPPAAL!

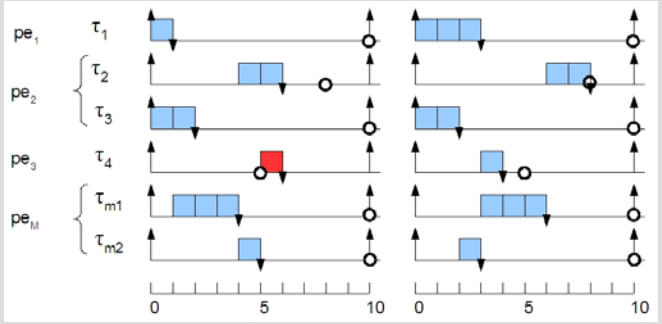


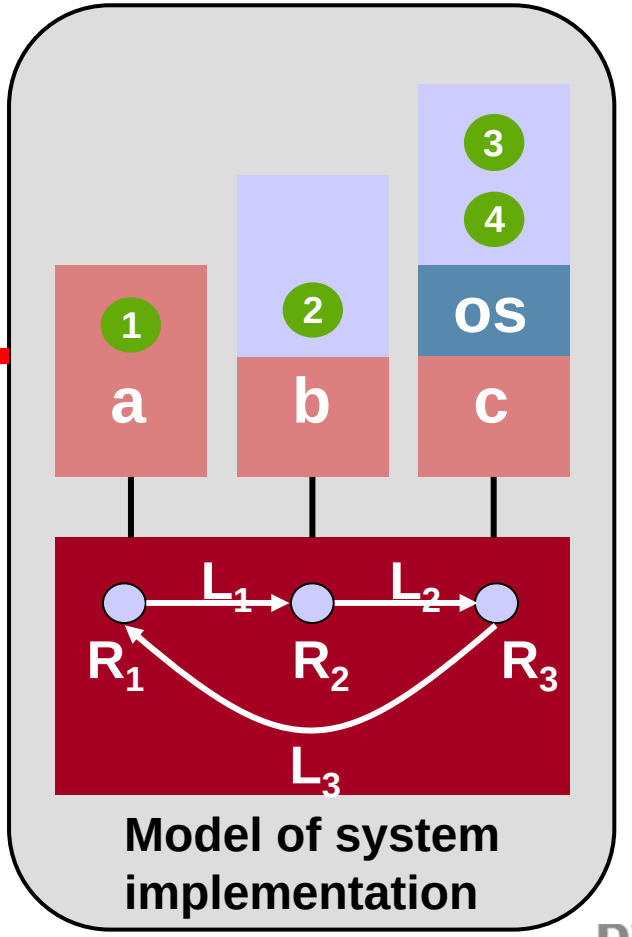
MOVES

```
E<>missedDeadline

E<>totalCostUsed(Memory)>=23

E<>totalCostUsed(Energy)>=15
```

Required specification

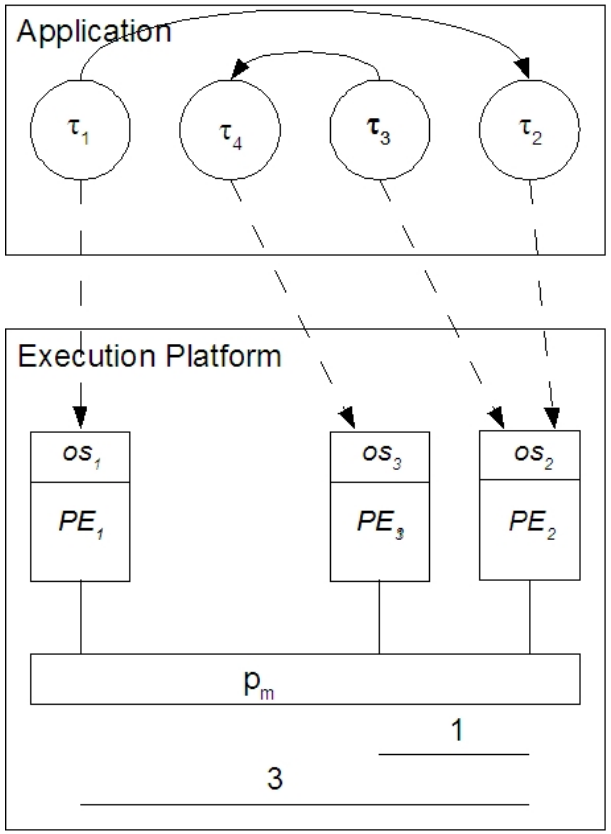
Model checking





Model of system implementation

Example: MPSoC specification



Task	π	δ	o
τ_1	10	10	0
τ_2	10	8	0
τ_3	10	10	0
τ_4	10	5	0

PE_i f = 1 MHz, os = RM

Task	*bcet*	*wcet*	*sm*	*dm*	*pw*
τ_1	1	3	1	3	5
τ_2	2	2	1	7	5
τ_3	2	2	1	6	10
τ_4	1	1	2	9	10

π : period [s]
δ : deadline [s]
ω : offset [s]
bcet : best case execution time [cycles]
wcet : worst case execution time [cycles]
sm : static memory [Byte]
dm : dynamic memory [Byte]
pw : power [*m*W]
f : frequency [Hz]
os: operating system[{*FP,RM,DM,EDF*}]

Specifying the application

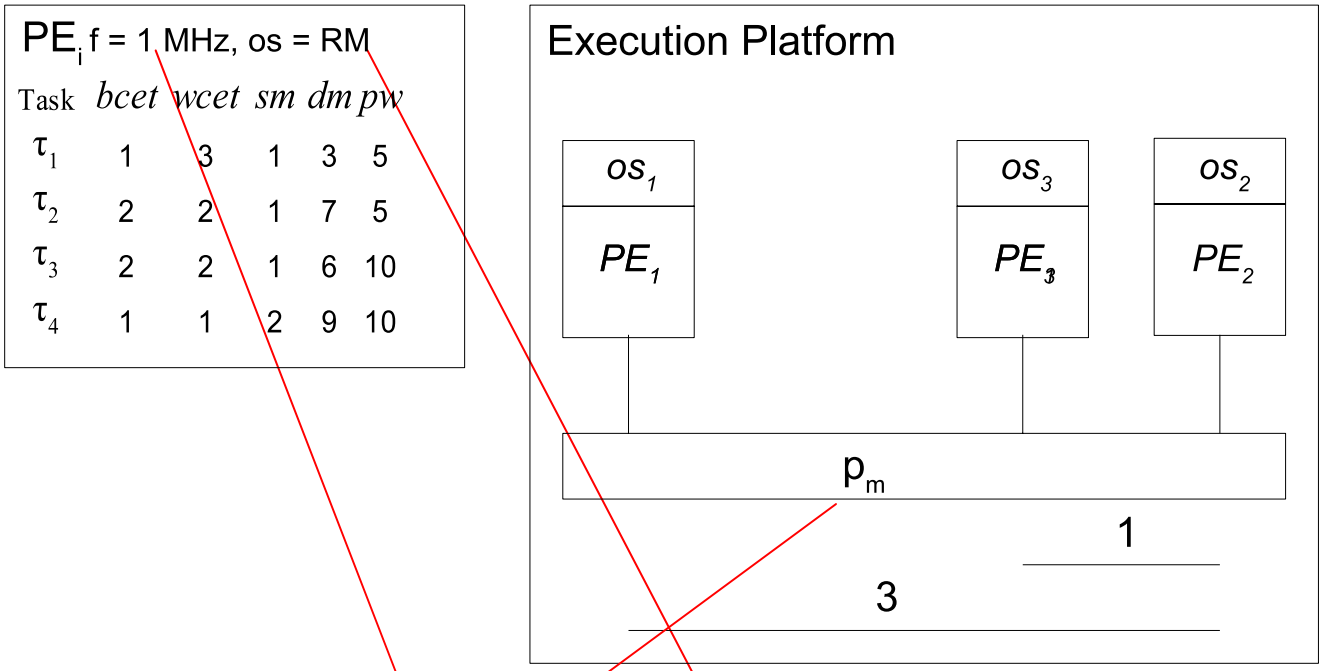
```
Task t1  = new Task(1, 3, 10, 10, 0, 1);
Task t2  = new Task(2, 2, 8, 10, 0, 2);
Task t3  = new Task(2, 2, 10, 10, 0, 3);
Task t4  = new Task(1, 1, 5, 10, 0, 4);
```

Task	π	δ	o
τ_1	10	10	0
τ_2	10	8	0
τ_3	10	10	0
τ_4	10	5	0

PE_i f = 1 MHz, os = RM

Task	*bcet*	*wcet*	*sm*	*dm*	*pw*
τ_1	1	3	1	3	5
τ_2	2	2	1	7	5
τ_3	2	2	1	6	10
τ_4	1	1	2	9	10

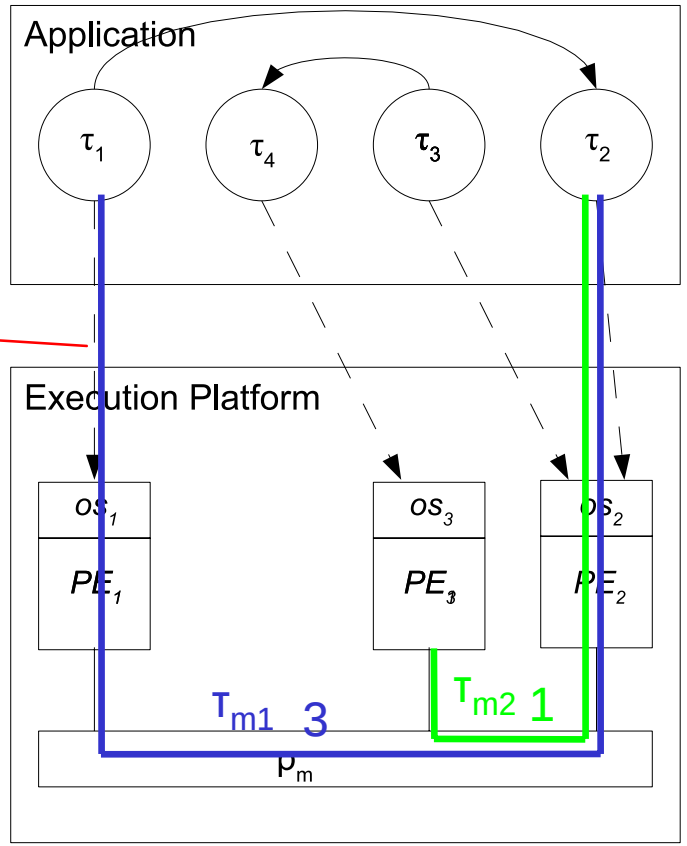
Execution platform




```
Processor p1 = new Processor(1, Processor.RM);
Processor p2 = new Processor(1, Processor.RM);
Processor p3 = new Processor(1, Processor.RM);
Processor pm = new Processor(1, Processor.RM);
Resource r1 = new Resource();
```

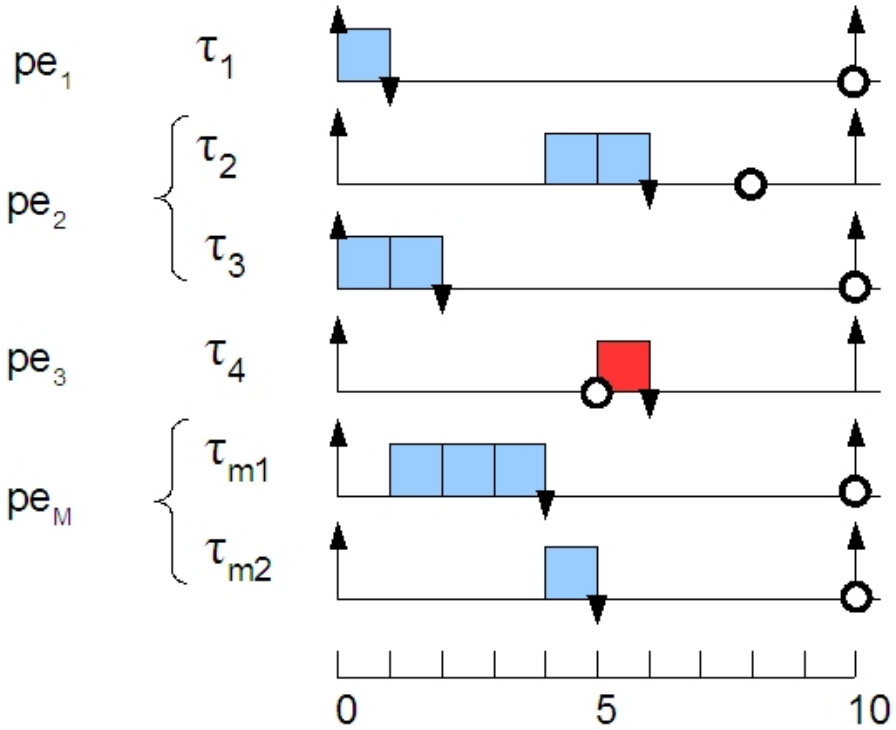
Mapping application onto platform

```
Task[][] tasks =
        {{t1},{t2,t3},{t4},{tm1,tm2}};

Task tm1 = new Task(3, 3, 10, 5);
Task tm2 = new Task(1, 1, 10, 6);

apps.useResource(tm1,r1);
apps.useResource(tm2,r1);

apps.addDep(t1,tm1);
apps.addDep(tm1,t2);
apps.addDep(t3,tm2);
apps.addDep(tm2,t4);
```

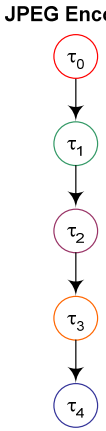


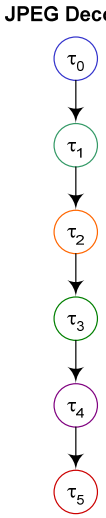
Traces

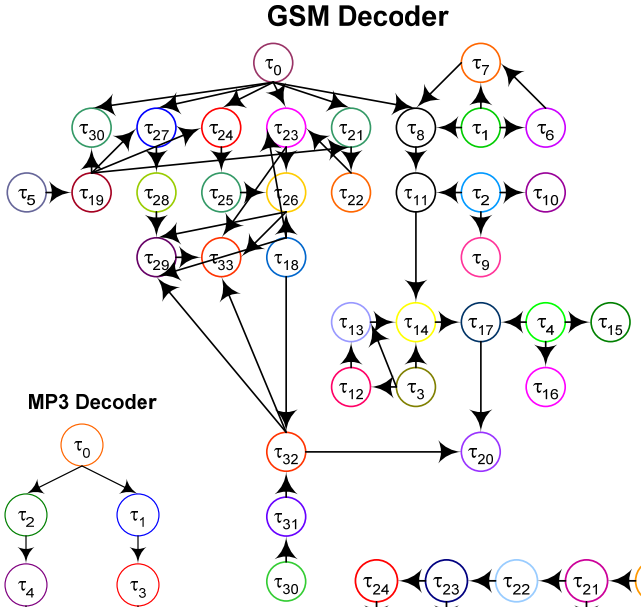
$e(\tau_1)= [1:3]$

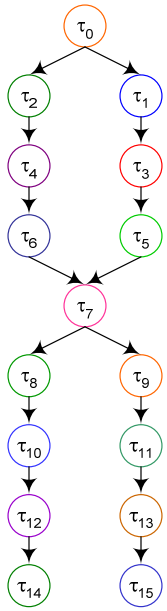


Handling realistic applications?

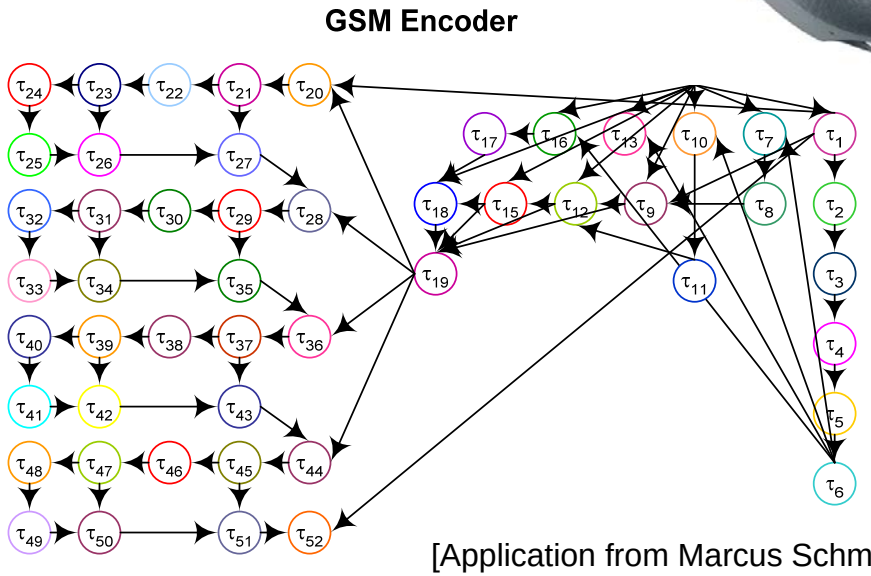






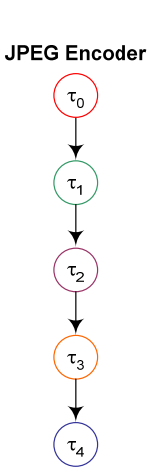


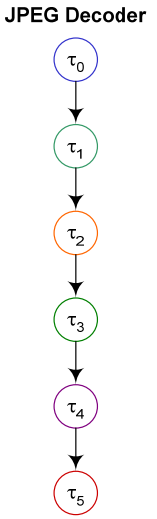
Smart phone:

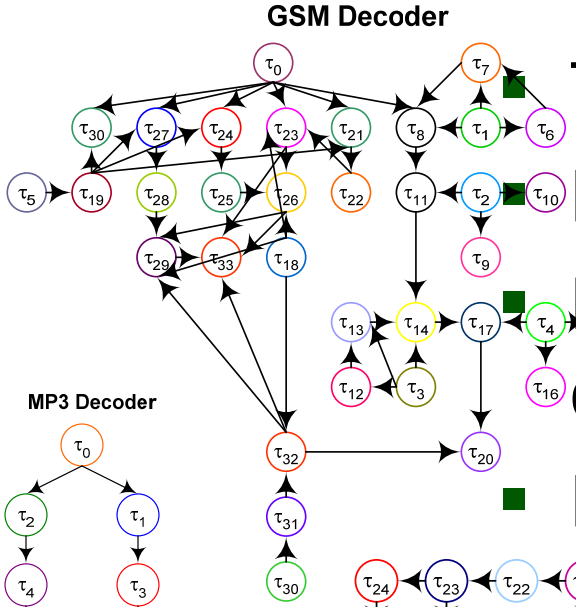


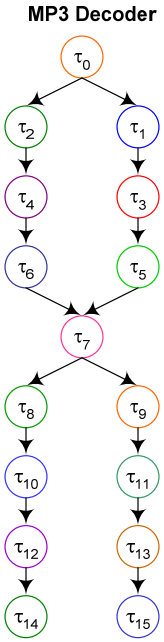
[Application from Marcus Schmitz, TU Linkoping]

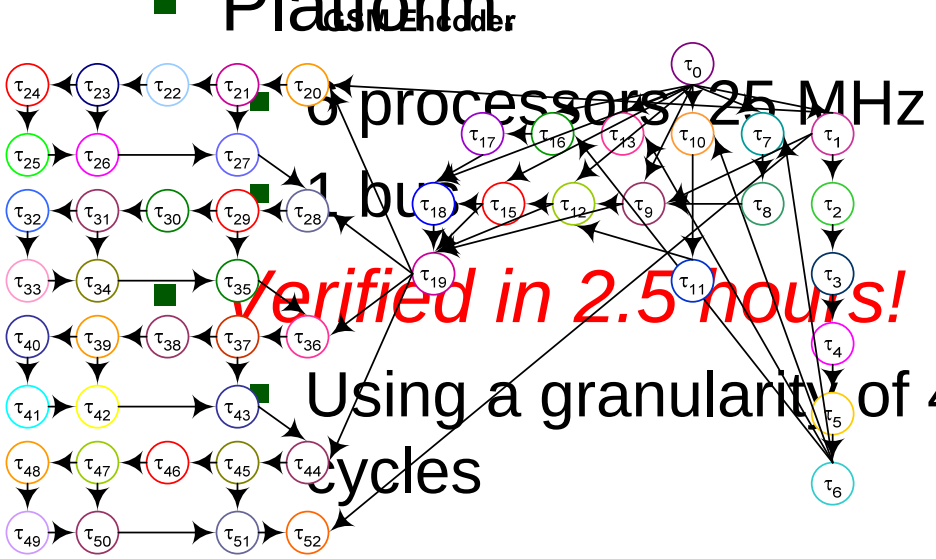
Smart phone











- Tasks: 114
- Deadlines: [0.02: 0.5] sec
- Execution: [52 : 266.687] cycles
- Platform:
  - 6 processors, 25 MHz
  - 1 bu
- *Verified in 2.5 hours!*
- Using a granularity of 400 cycles

Acknowledgements

MOVES

- Aske Brekling
- Jens Ellebæk
- Kristian S. Knudsen

ARTS

- Shankar Mahadevan
- Kehuai Wu
- Kashif Virk
- Michael Storgaard
- Mercury Gonzalez

Discussion Points

- Accuracy versus scalability
  - UPPAAL & SymTA/S (Network Calculus)
- Verification versus simulation
  - UPPAAL & ARTS (System C)
  - UPPAAL & Ptolemy
  - UPPAAL & TrueTime
- Efficient verification methods for optimal infinite scheduling for PTA.
  - Negative rates
  - Discounting

Collaborators

@UPPsala

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

@AALborg

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