Timing and Performance Analysis using Model Checking

Kim G. Larsen

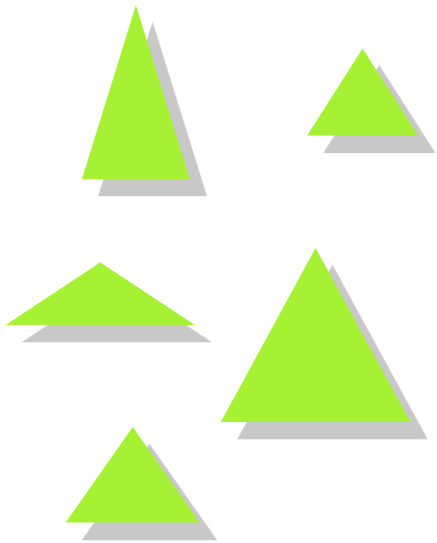
CISS – Aalborg University

DENMARK

Embedded Systems

Tasks:
Computation times
Deadlines
Dependencies
Arrival patterns
uncertainties

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



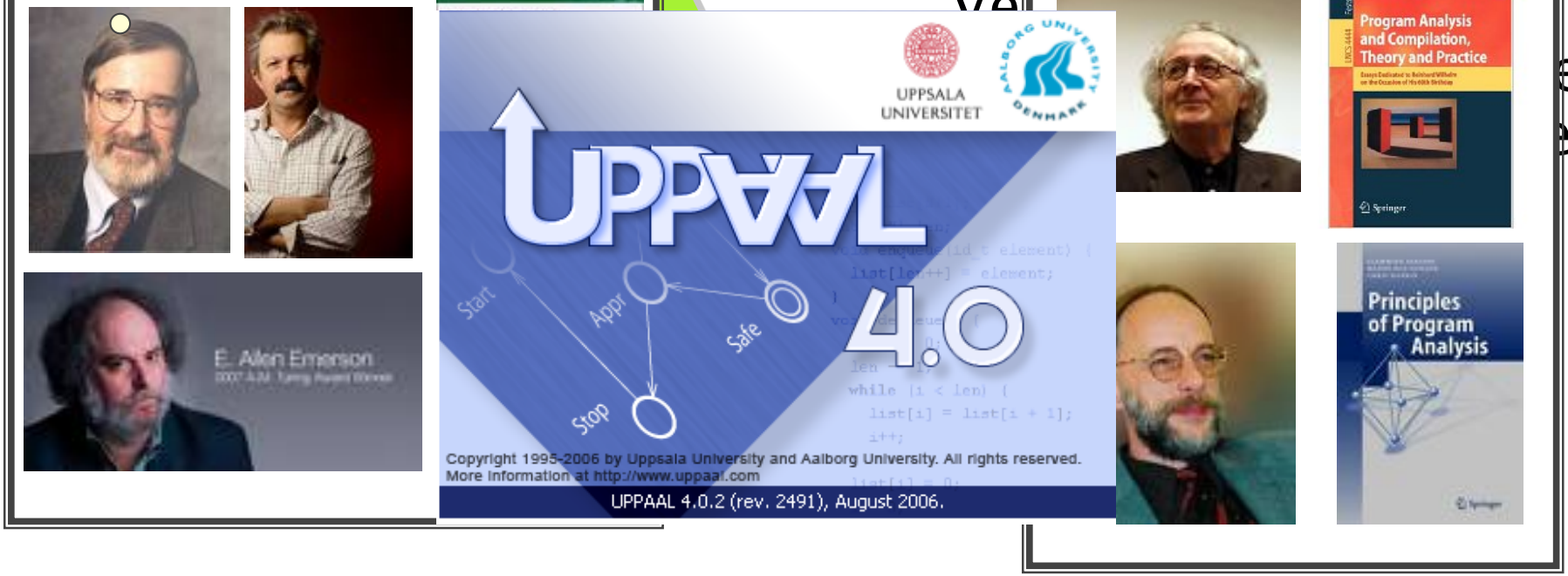
Resources
Execution platform
PE, Memory
Networks
Drivers
uncertainties

Timing Analysis

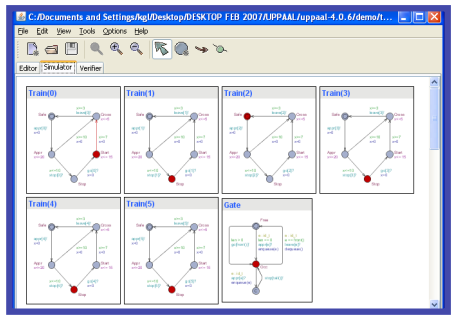
Model checking *is fixpoint iteration without dynamic abstraction and using set union to collect states.*

Abstract Interpretation *is fixpoint iteration with dynamic abstraction using lattice join to combine abstract states.*

- Sch… …s: Ve… …el …ed.



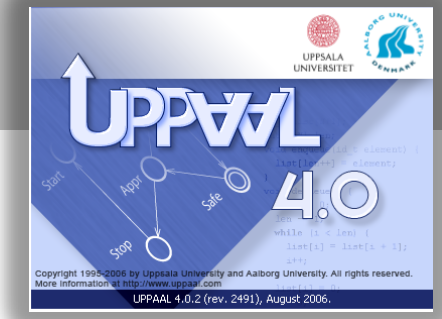
QUANTITATIVE Model Checking



Time

Cost

Probability



System Description

Requirement

TOOL

No!
Debugging Information

Yes
Prototypes
Executable Code
Test sequences

$A\square(\ req\ )\ A\}\ grant)$

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

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

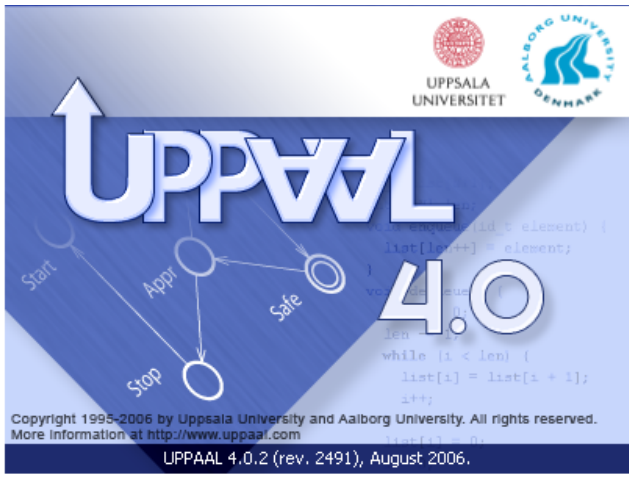
$A\square(\ req\ )\ A\}_{t<30s\ ,\ p>0.90}\ grant)$

Overview

- Timed Automata Model Checking

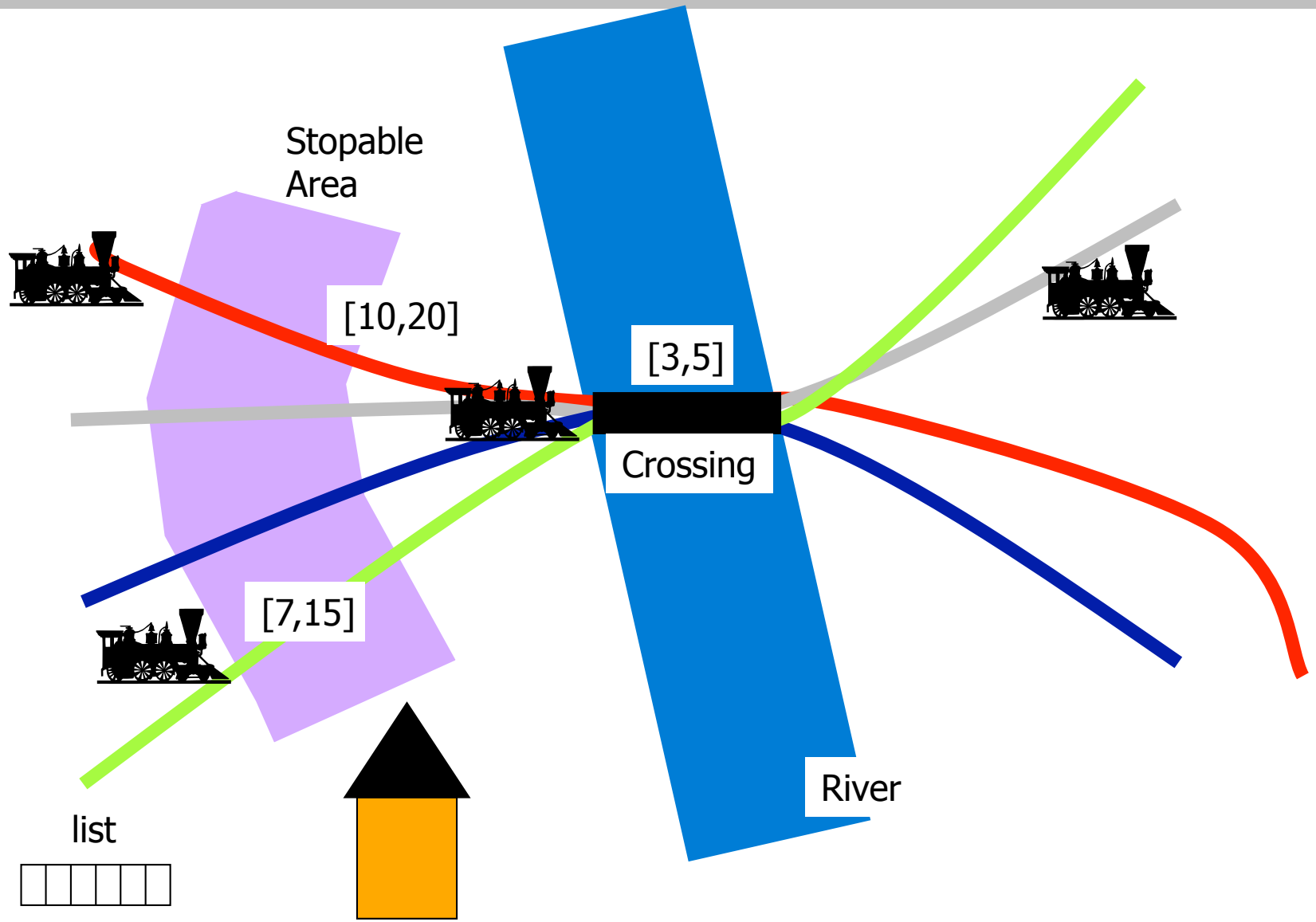
- Scheduling
  - Task Graph Scheduling
- Schedulability Analysis
  - Single Processor
  - Multi Processor

- WCET Analysis
- Performance Analysis
  - Statistical Model Checking



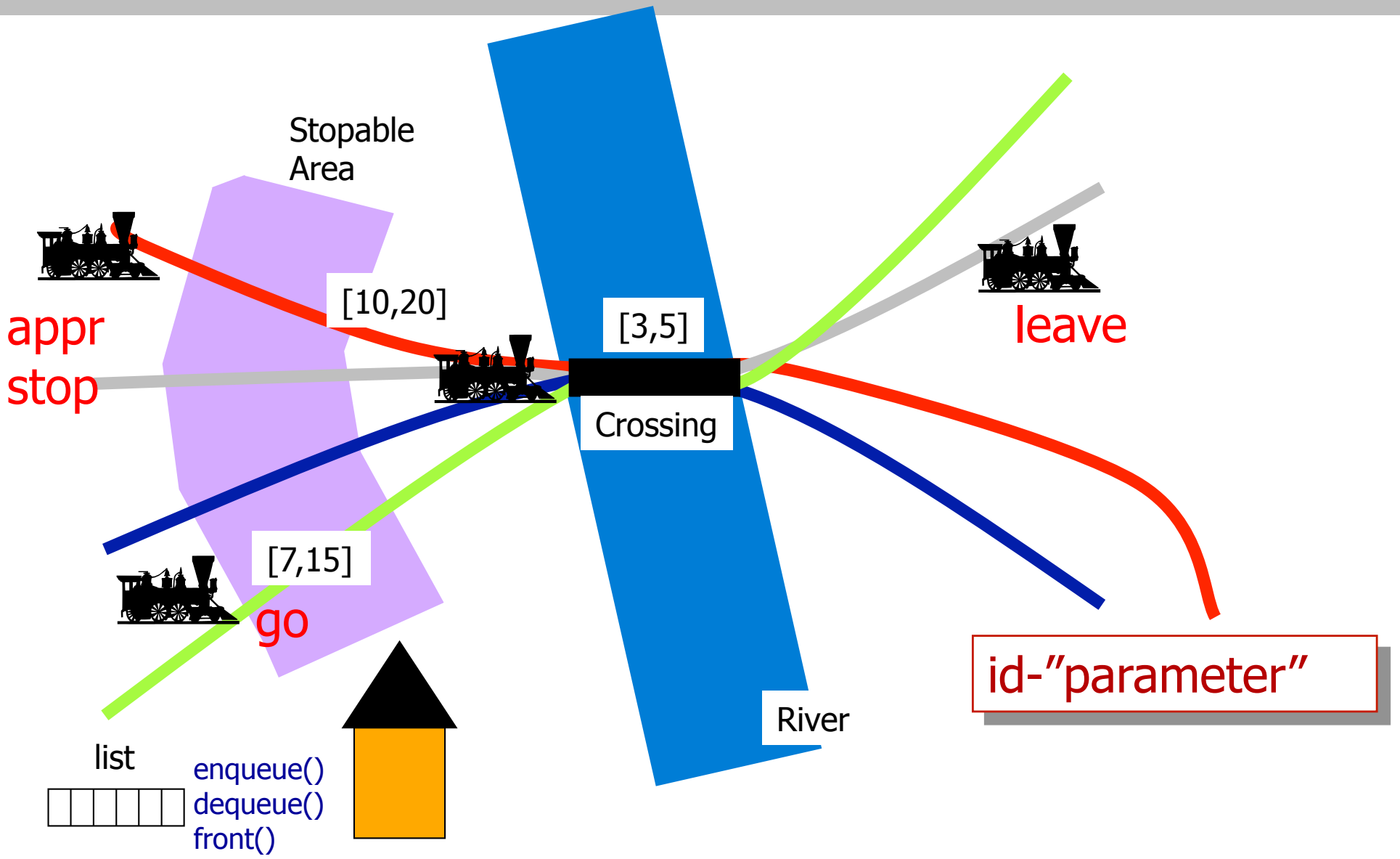
Timed Automata

Train Crossing



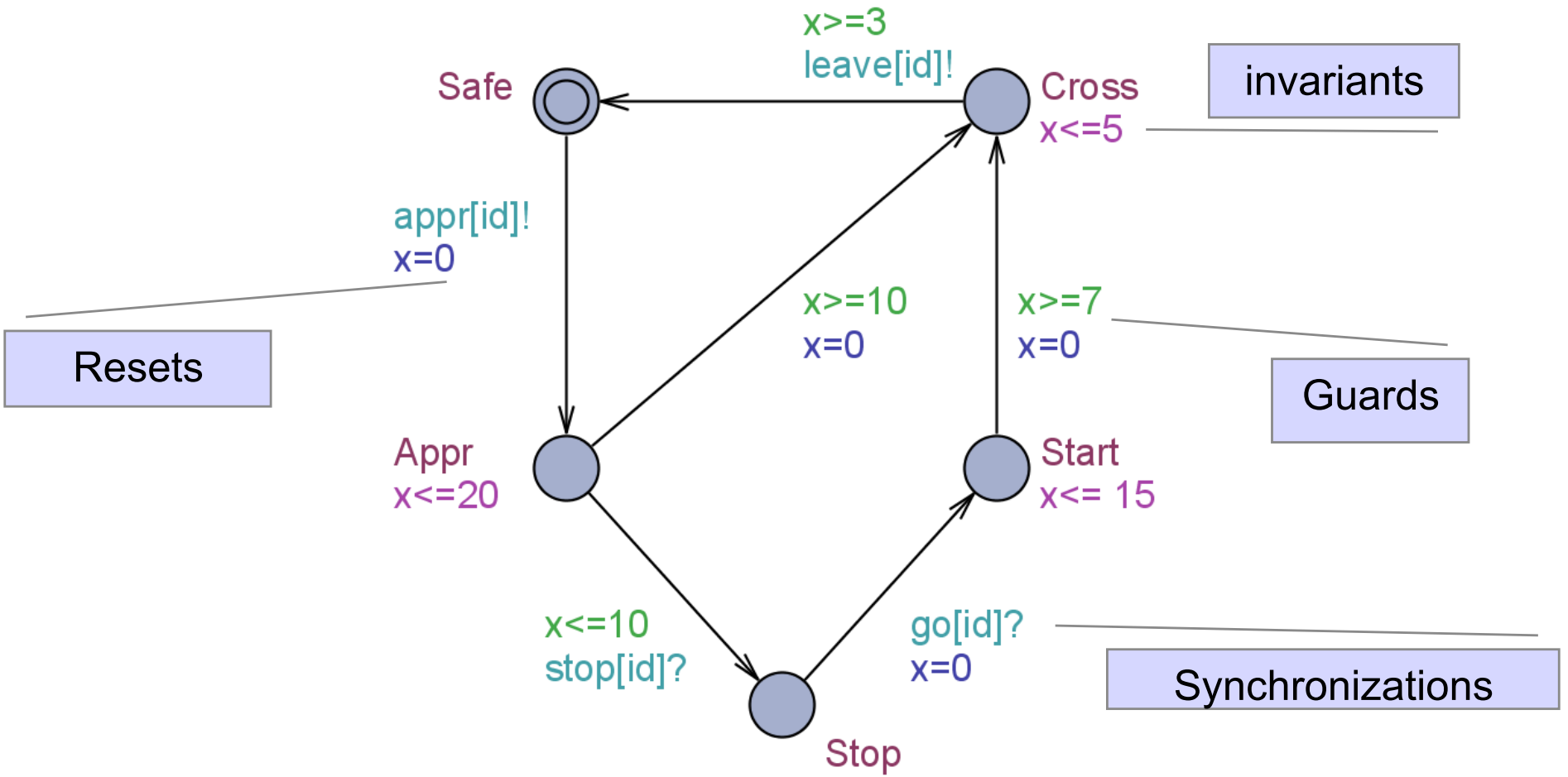
Train Crossing

Communication via channels!



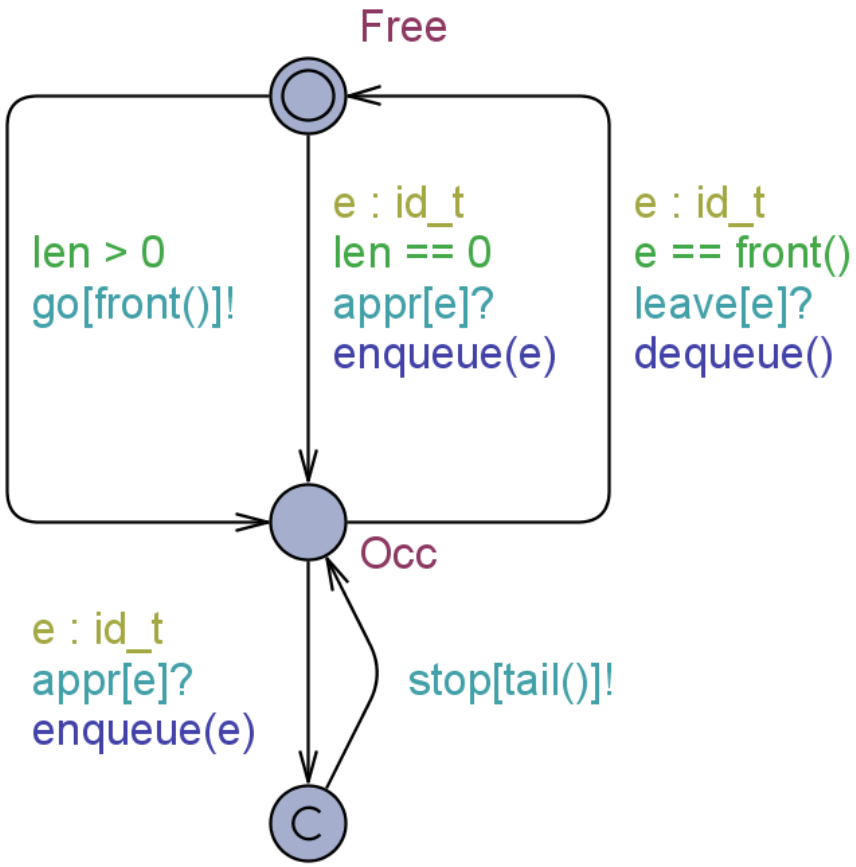
Timed Automata [Train]

= Finite State Control
+ Real Valued Clocks



Timed Automata [Gate]

= Finite State Control
+ Real Valued Clocks
+ Discrete Variables




```
id_t list[N+1];
int[0,N] len;

// Put an element at the end of the queue
void enqueue(id_t element)
{
        list[len++] = element;
}

// Remove the front element of the queue
void dequeue()
{
        int i = 0;
        len -= 1;
        while (i < len)
        {
                list[i] = list[i + 1];
                i++;
        }
        list[i] = 0;
}
```

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: $P \rightarrow_{\leq t} Q$

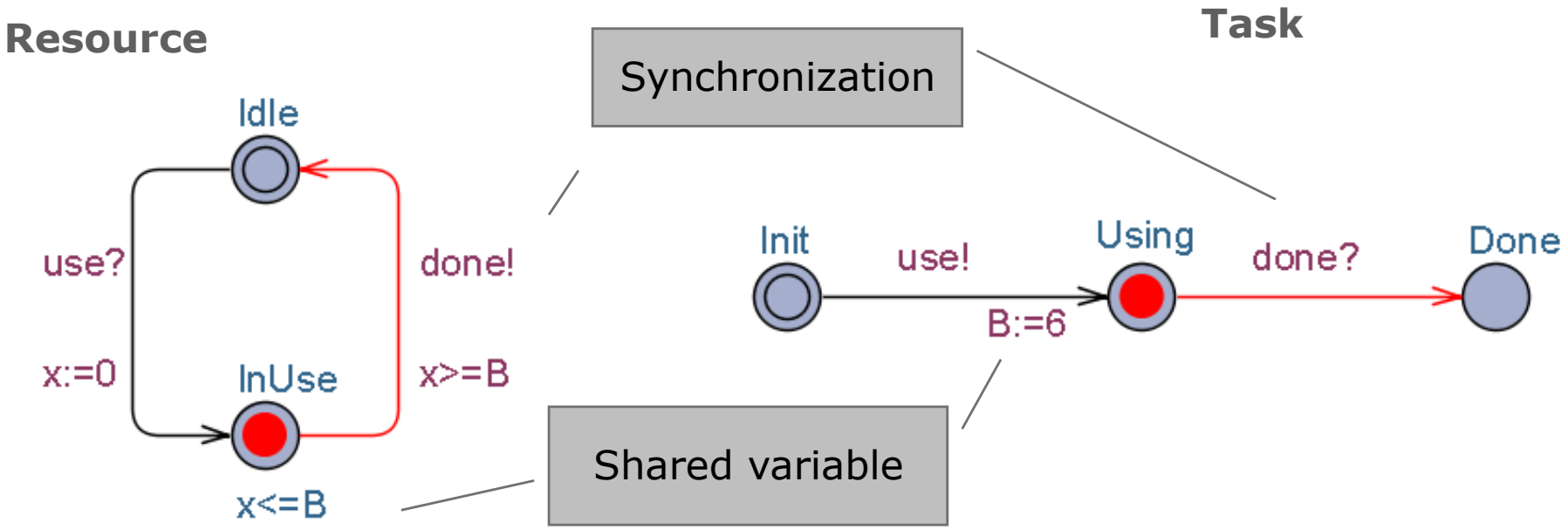
The expressions *P* and *Q* must be type safe, side effect free, and evaluate to a boolean.

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

Scheduling


Mon
Tue
Wed
Thu
Fri
8-10
10-12
12-14
14-16
16-18
18-20
20-22
22-24
0-2
2-4
4-6
6-8
Halo 3 (match)
Half-life
TF2 training
TEAM FORTRESS!!!...
Halo Halo Halo!
sweet sleep
Buy Jericho
Jericho
TF2
Portal
Halo
sleep
sleep
MORE HALO

Resources and Tasks



Semantics:

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

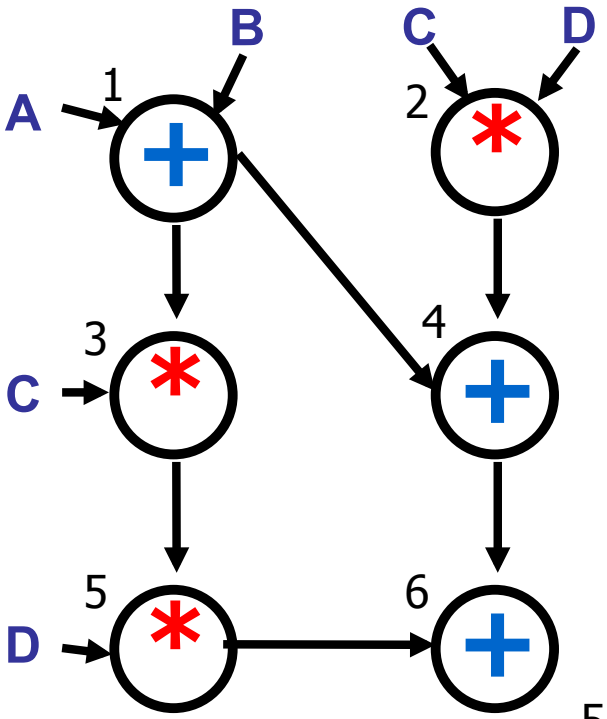
d(3.1415) → (Idle , Init , B=0 , x=3.1415)

use → (InUse , Using , B=6, x=0)

d(6) → (InUse , Using , B=6, x=6)

done → (Idle , Done , B=6 , x=6)

Optimal Scheduling – TIME



Compute :

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

4

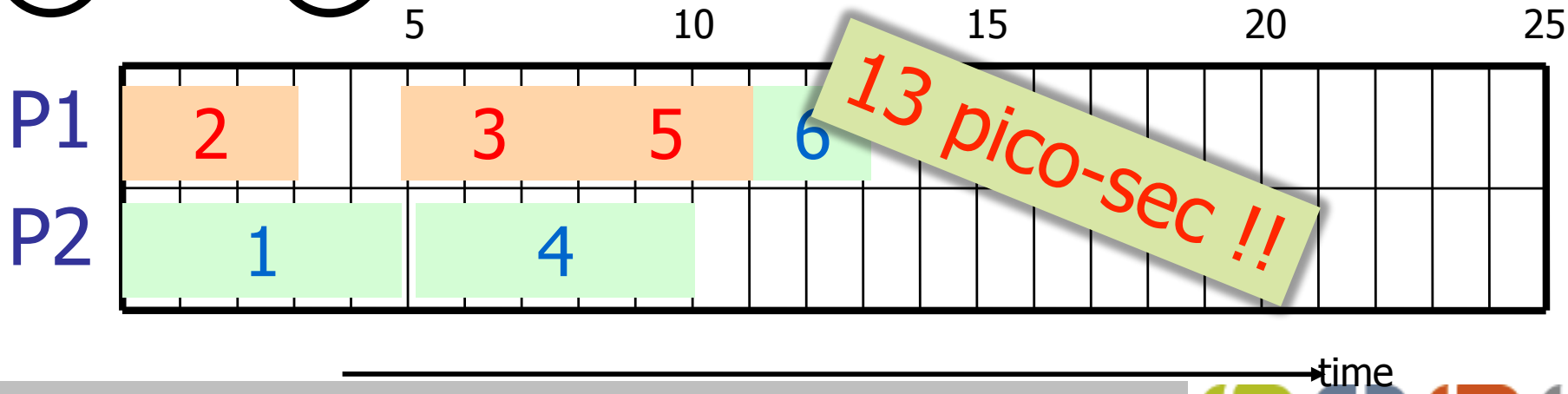
using 2 processors

P1 (fast)

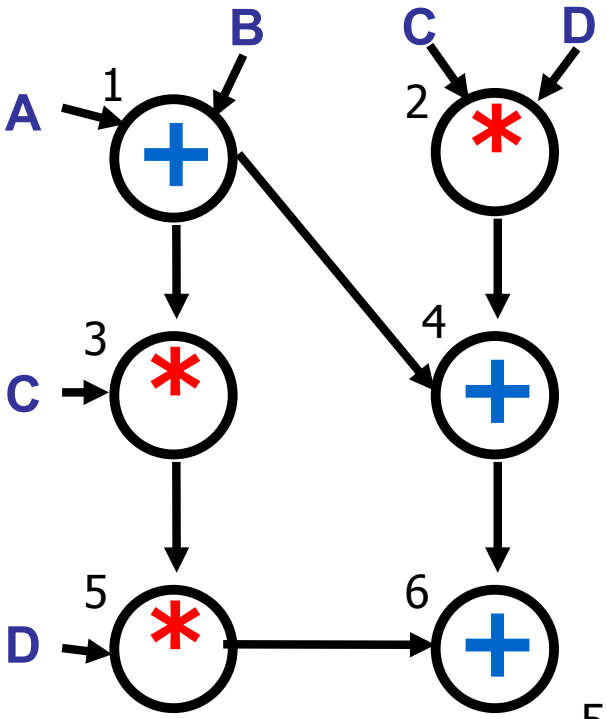
+	2ps
*	3ps

P2 (slow)

+	5ps
*	7ps



Optimal Scheduling – TIME



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

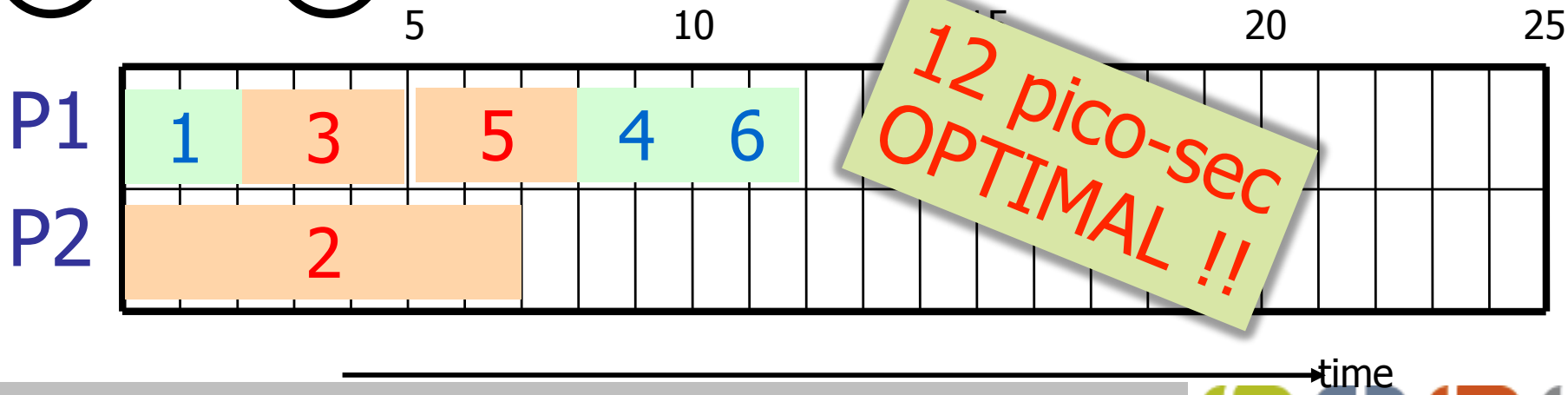
using 2 processors

P1 (fast)

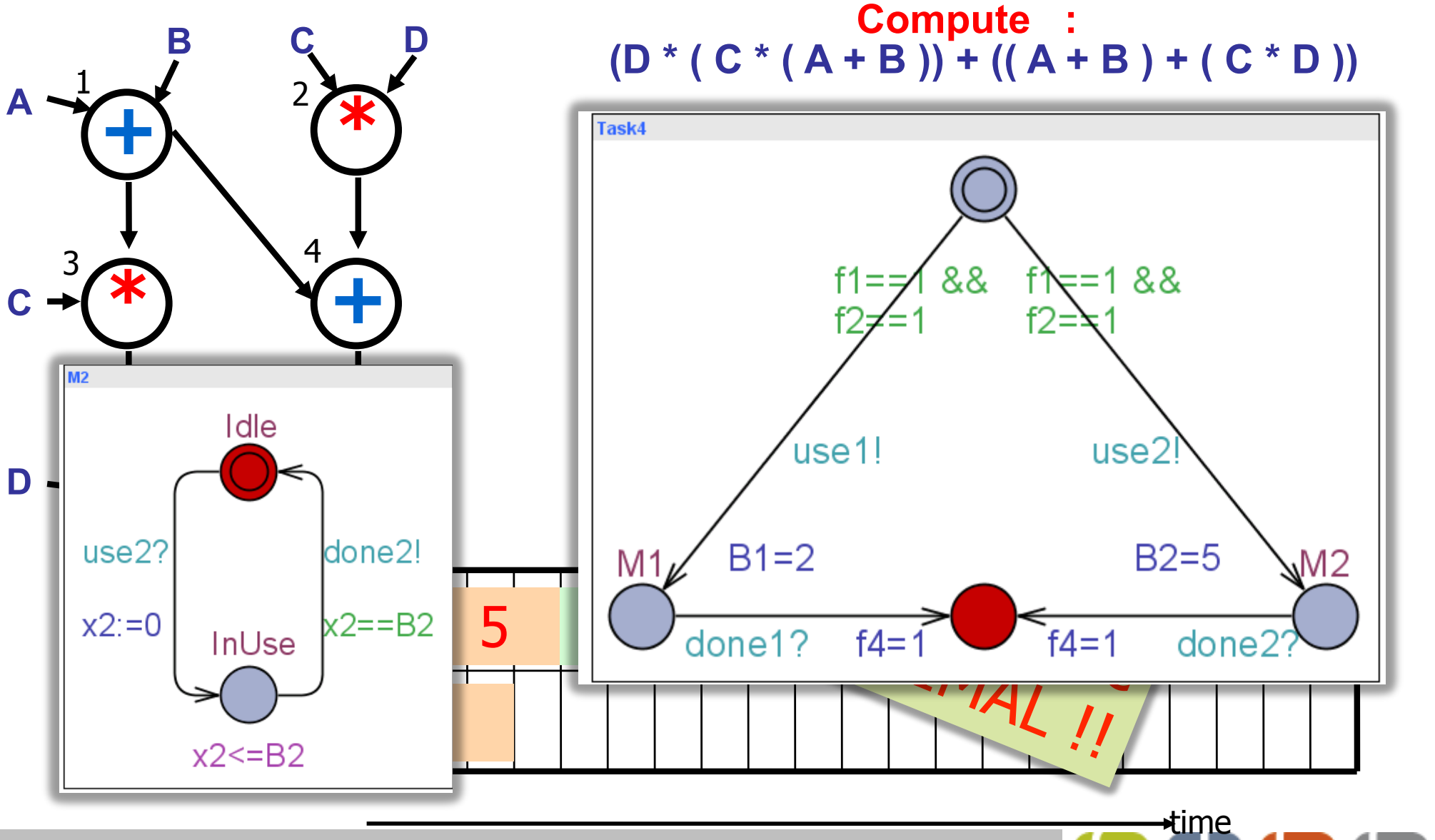
+	2ps
*	3ps

P2 (slow)

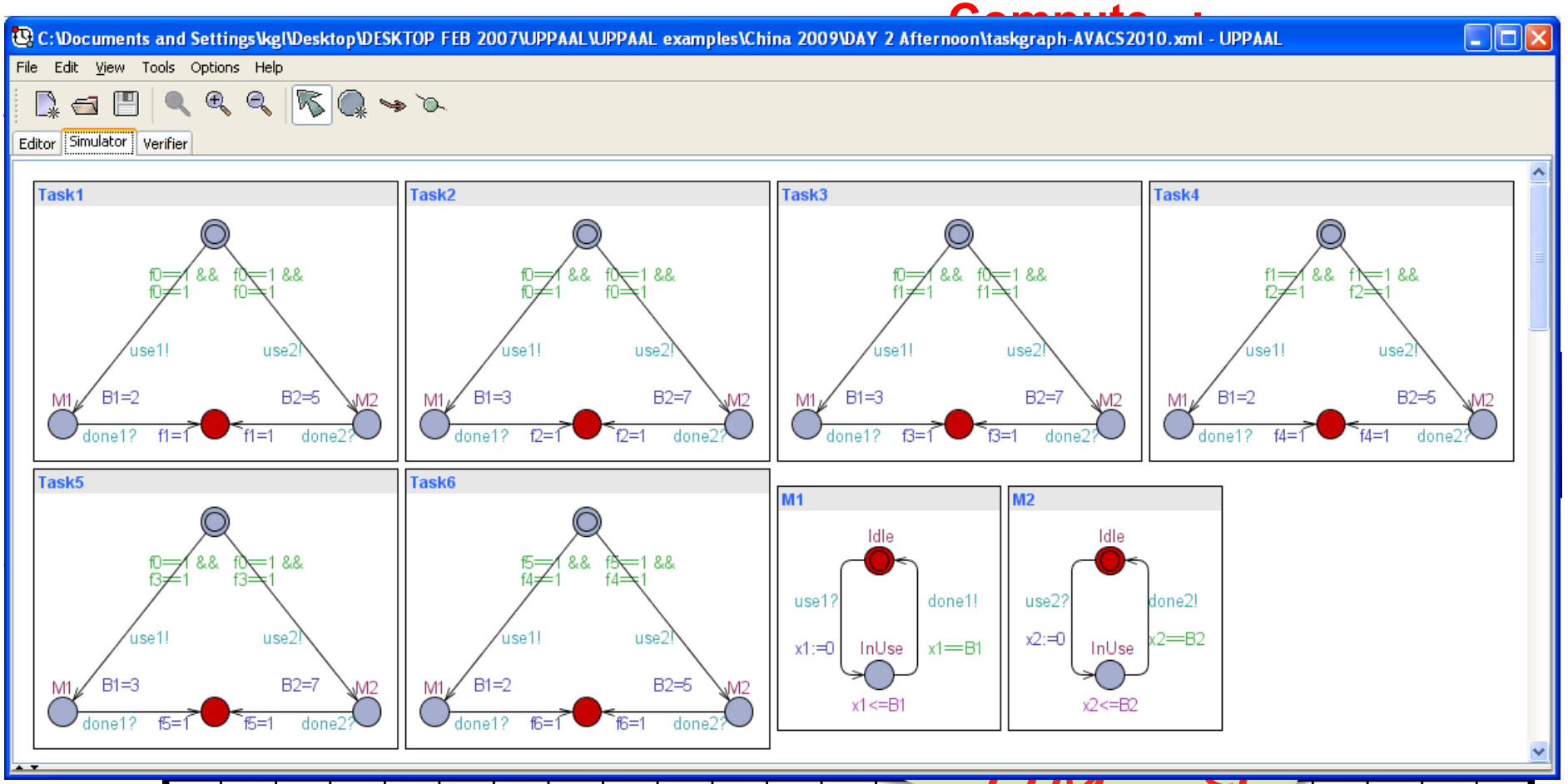
+	5ps
*	7ps



Optimal Scheduling – TIME



Optimal Scheduling – TIME



A

C

D

P2

E<> (Task1.End and … and Task6.End)

time

Experimental Results

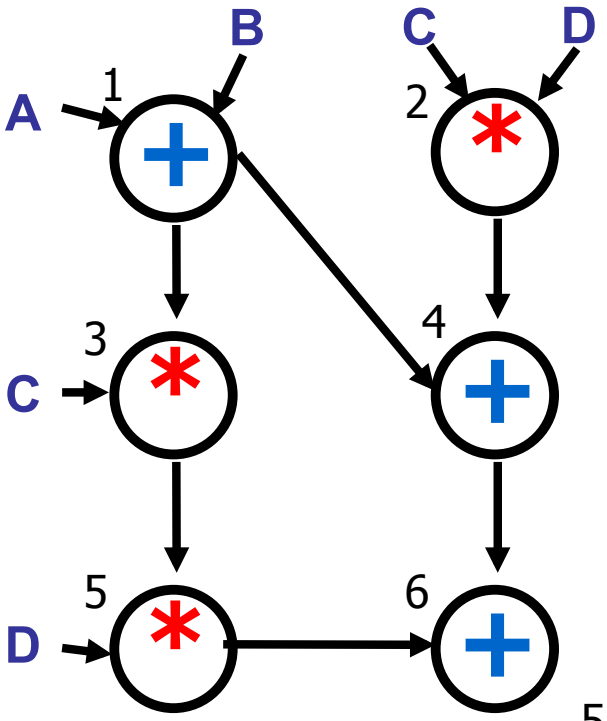
name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.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

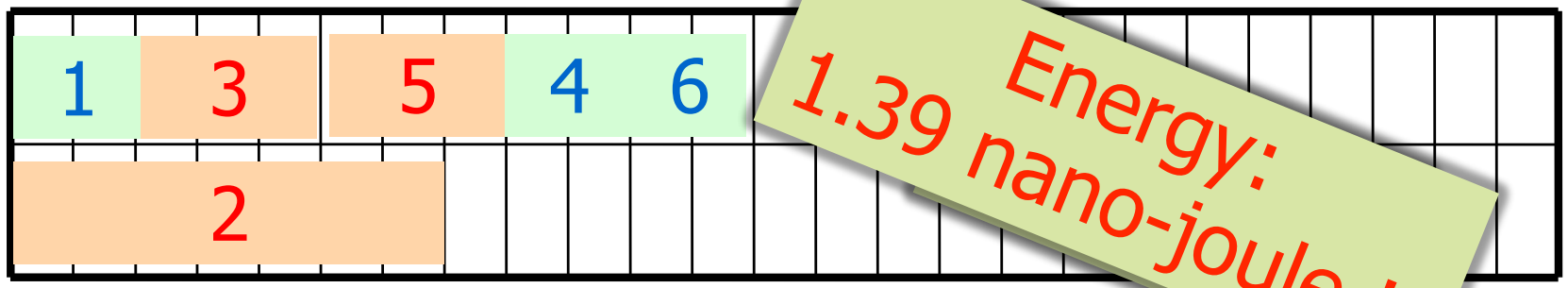
using 2 processors

P1 (fast) P2 (slow)

+	2ps
*	3ps
Idle	1oW
use	90W

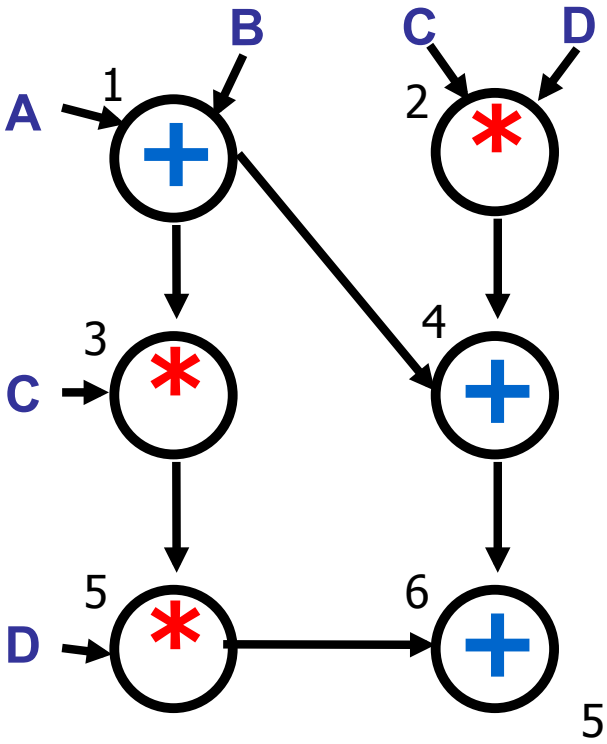
+	5ps
*	7ps
Idle	20W
In use	30W

ENERGY:



Energy:
1.39 nano-joule !!

Task Graph Scheduling – Revisited



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

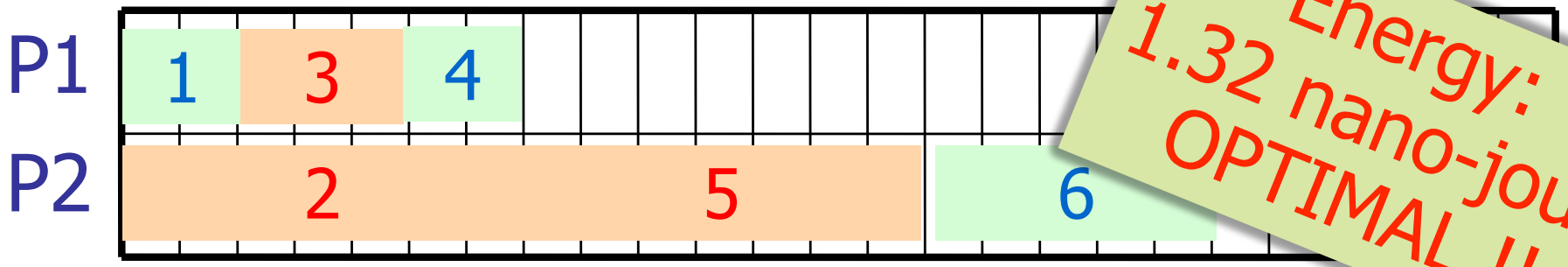
using 2 processors

P1 (fast) **P2** (slow)

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

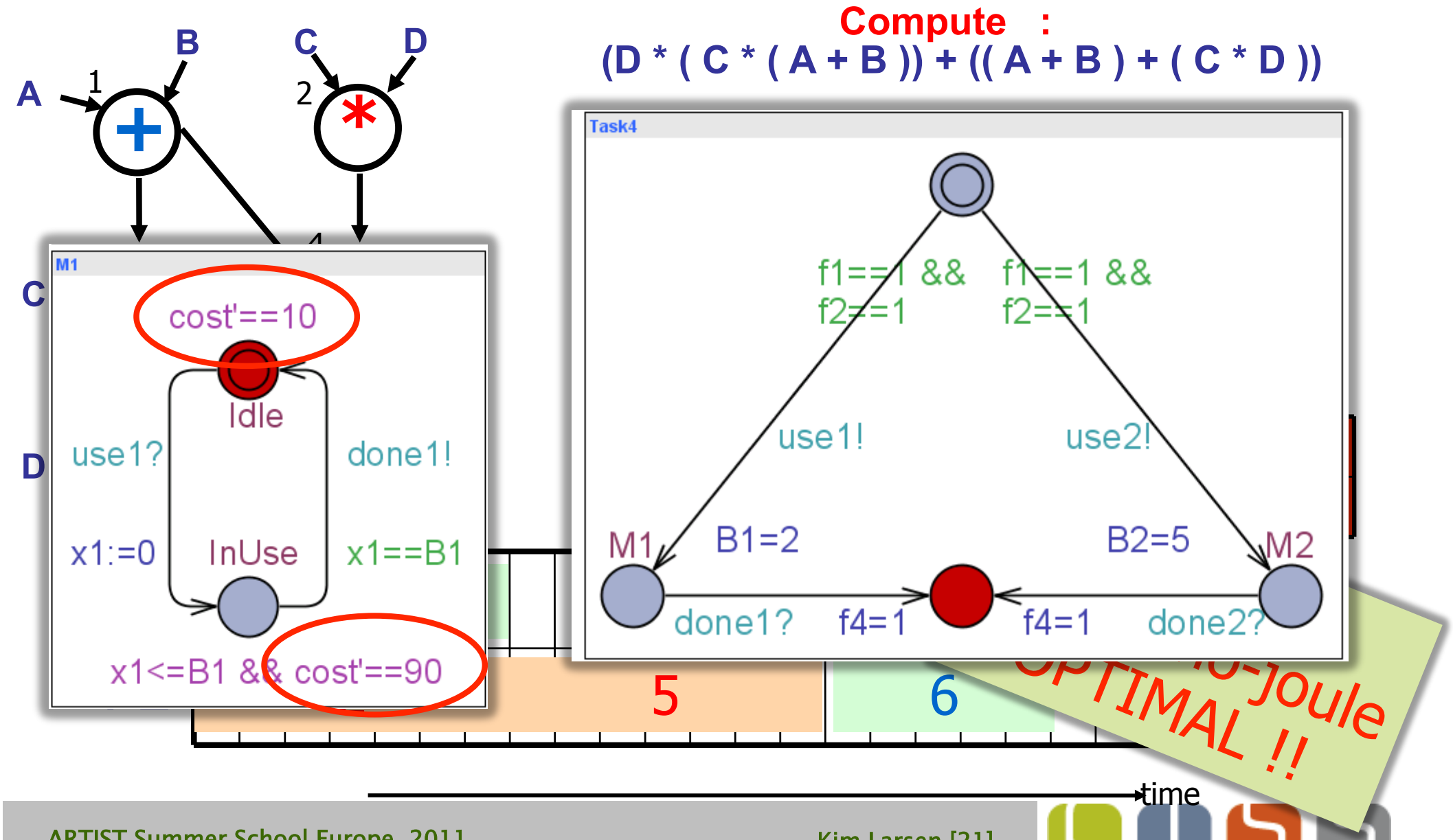
+	5ps
*	7ps
Idle	20W
In use	30W

ENERGY:
10

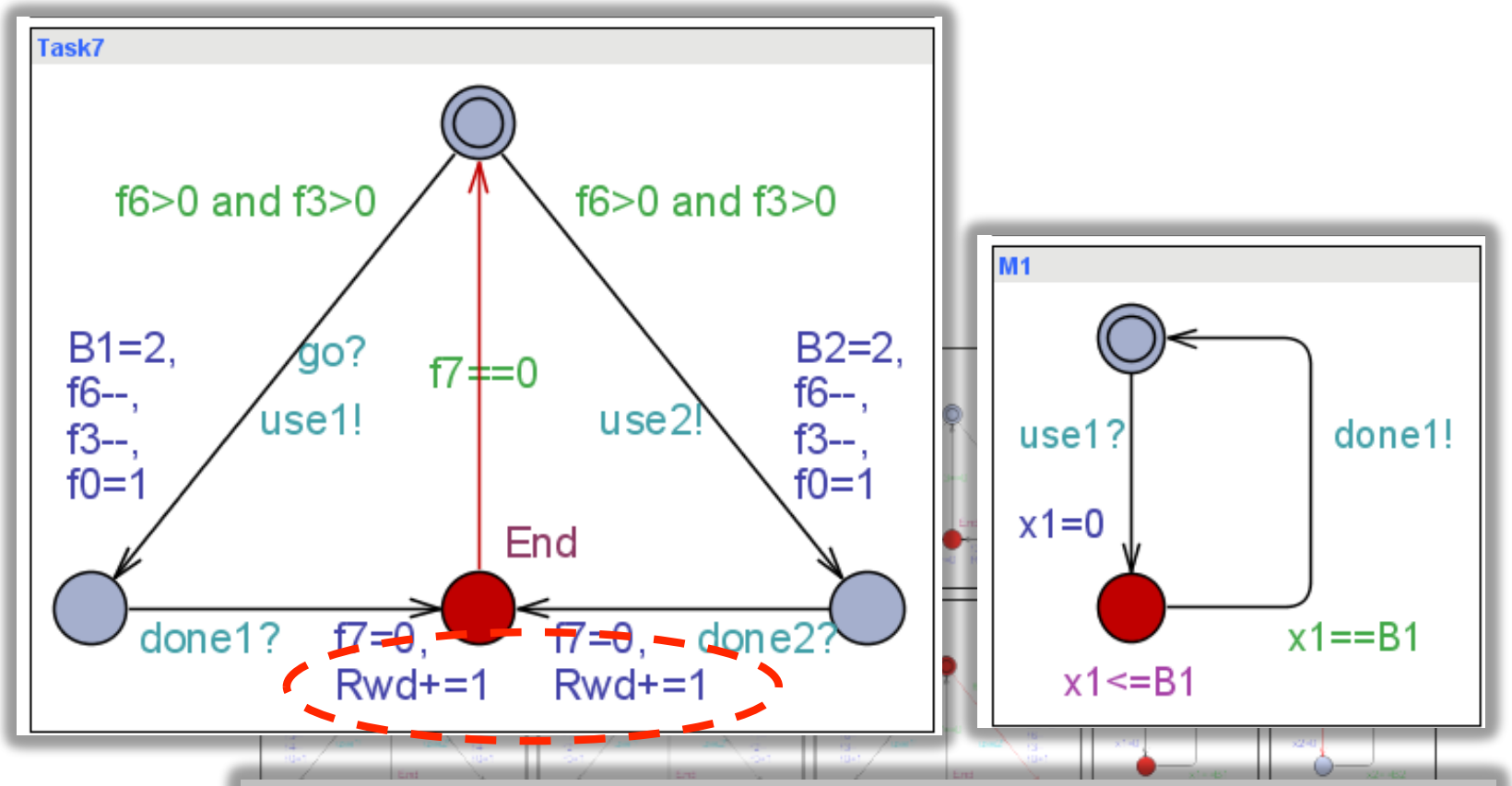


Energy:
1.32 nano-joule
OPTIMAL !!

Task Graph Scheduling – Revisited

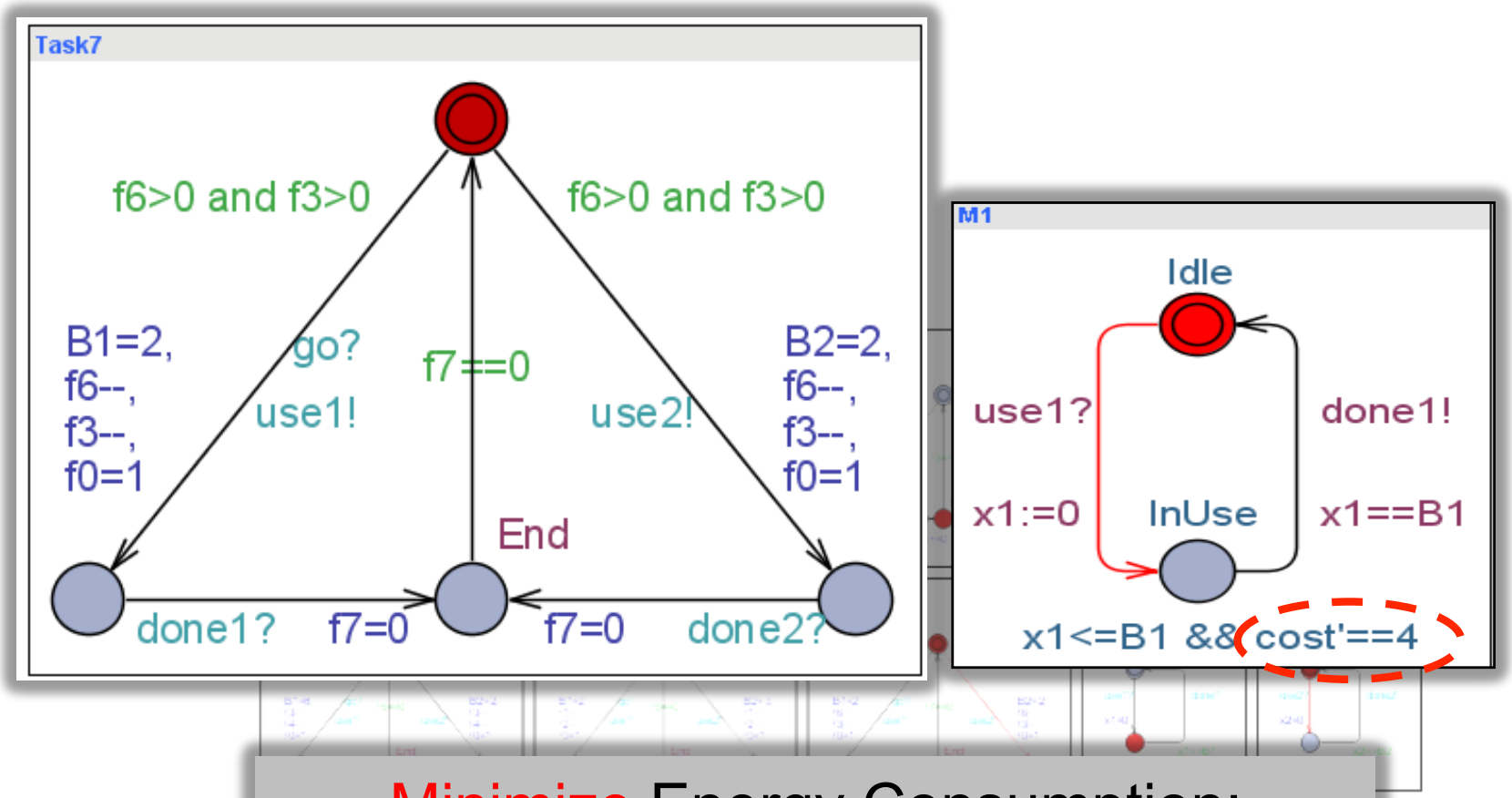


Optimal Infinite Scheduling



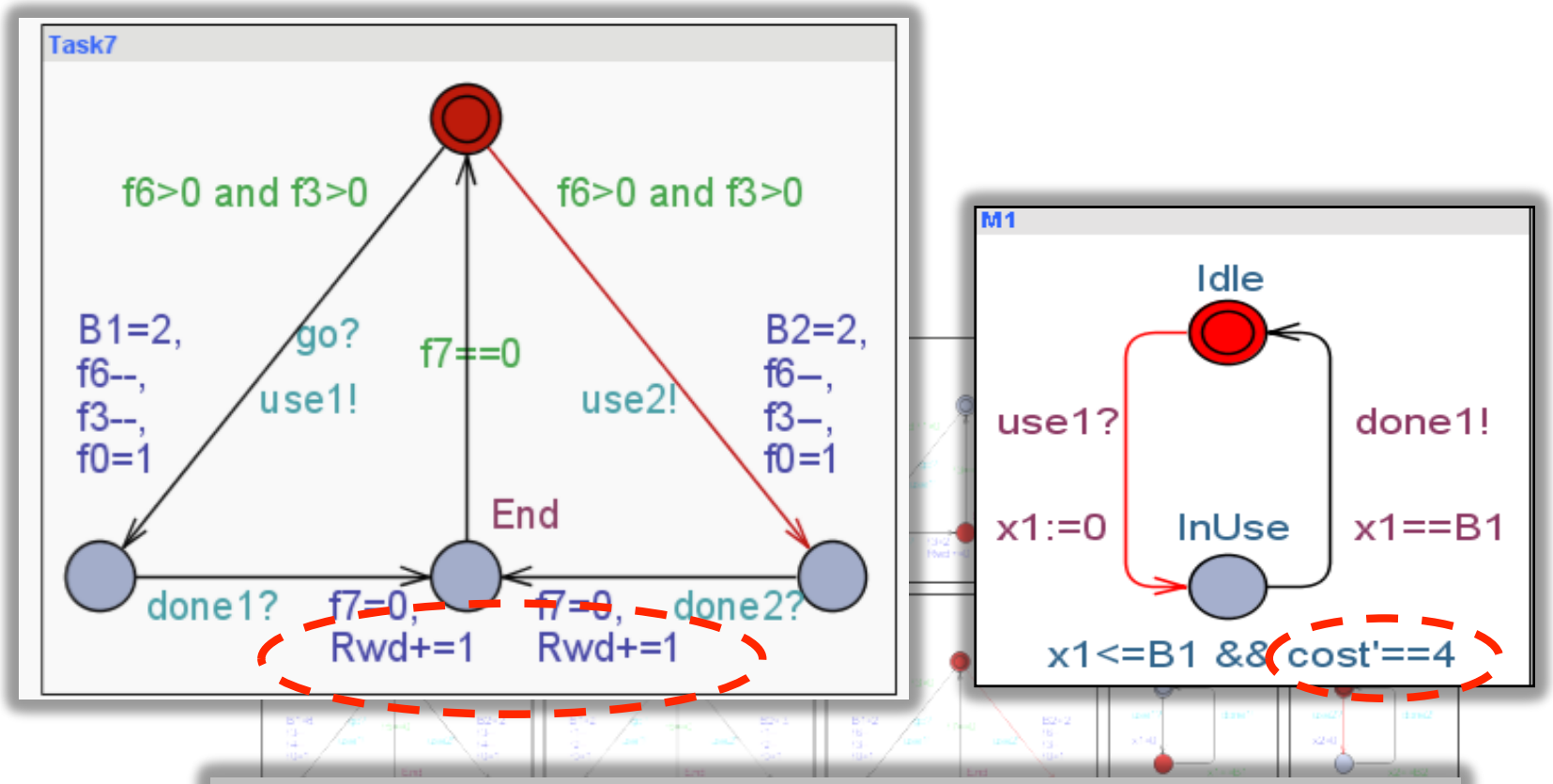
Maximize throughput:
i.e. maximize Reward / Time in the long run!

Optimal Infinite Scheduling



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

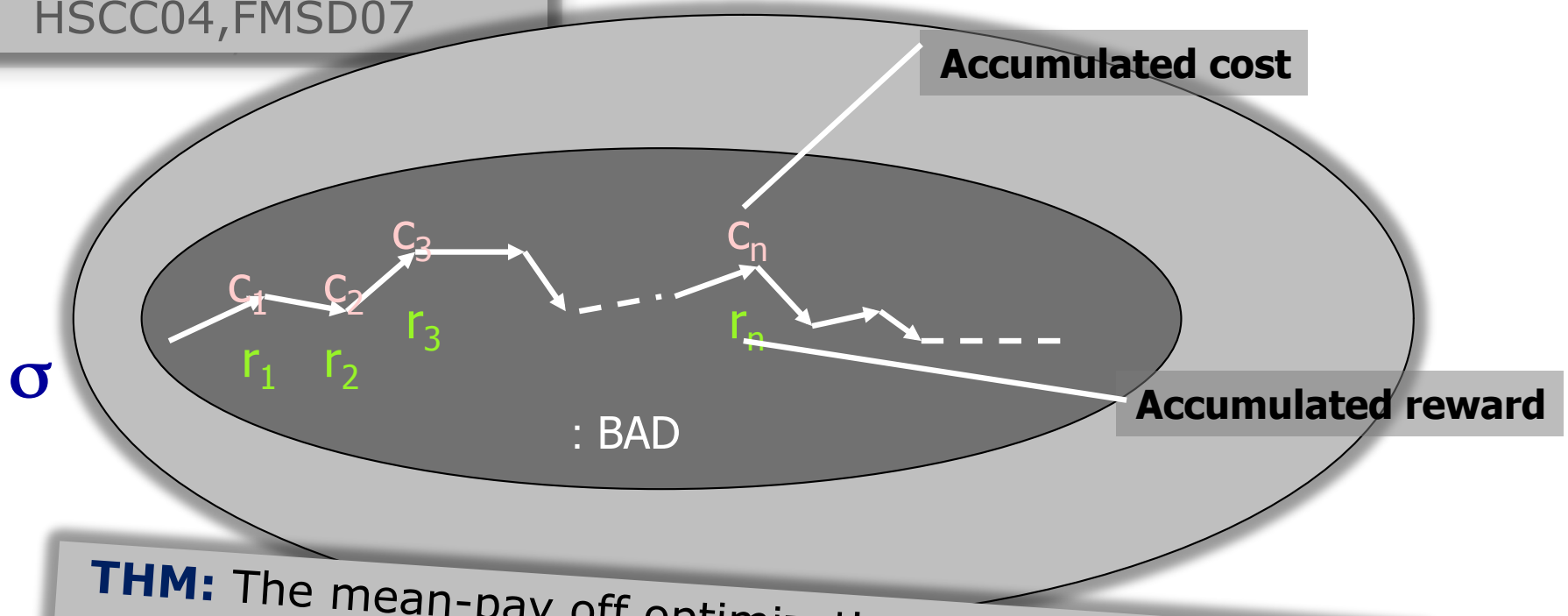
Optimal Infinite Scheduling



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

Mean Pay-Off Optimality

Bouyer, Brinksma, Larsen:
HSCC04,FMSD07



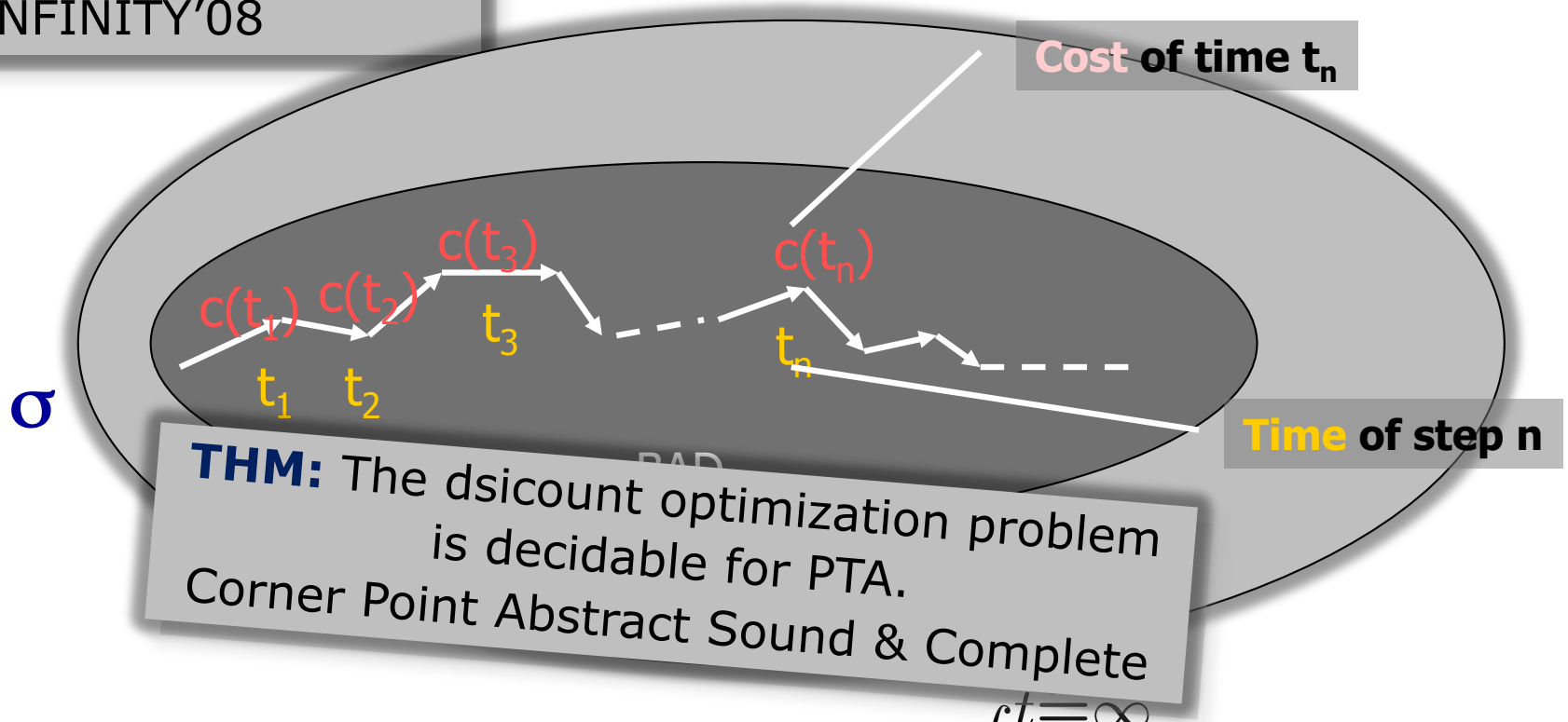
THM: The mean-pay off optimization problem is decidable (and PSPACE-complete) for PTA.
Corner Point Abstract Sound & Complete

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

Discount Optimality

$\lambda < 1$: discounting factor

Larsen, Fahrenberg: INFINITY'08



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

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

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:

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

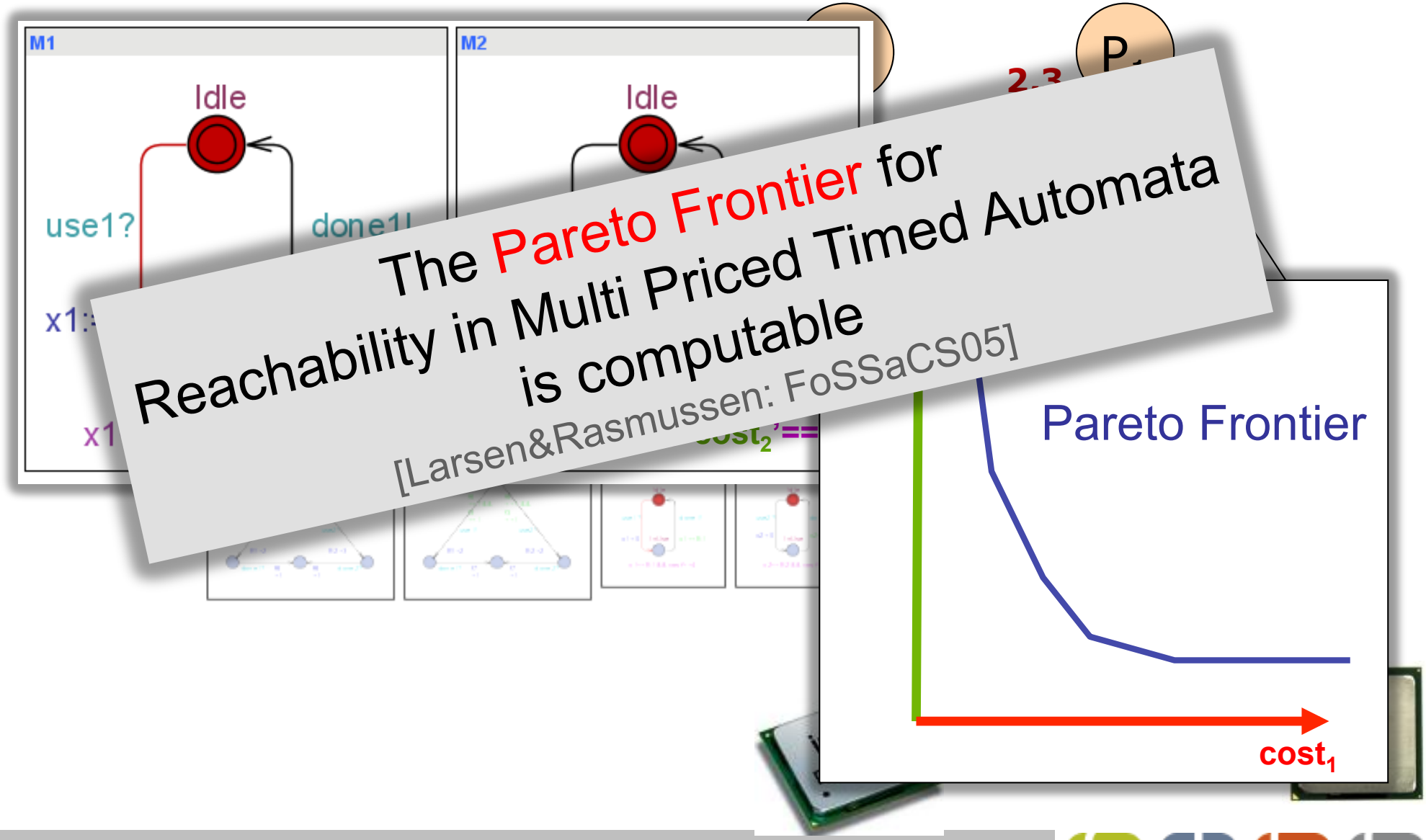
then $\inf_Z f$ is obtained at a corner-point of Z (with integer coefficients).

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

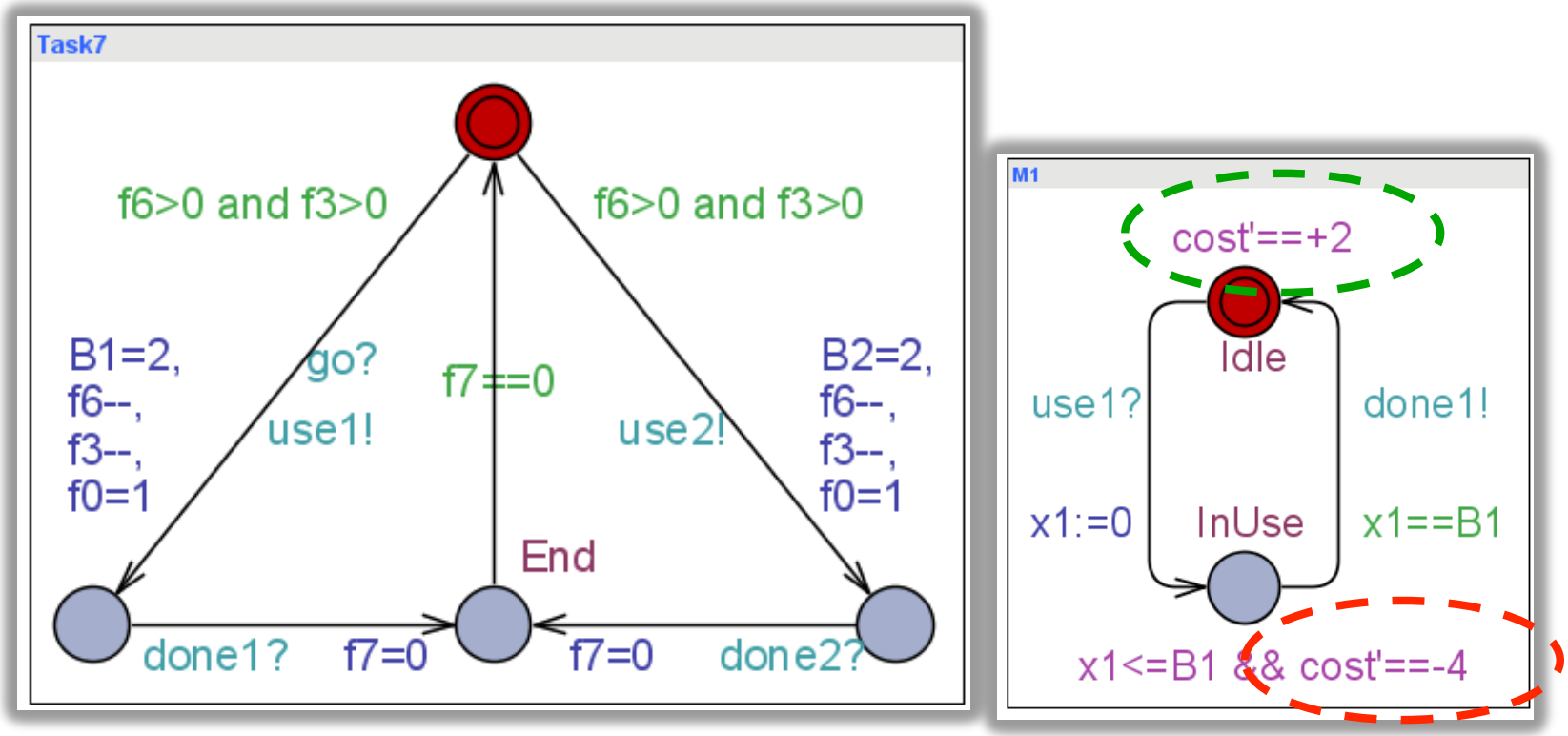
$$f : (t_1, \ldots, t_n) \mapsto a_1 \lambda^{t_1} + \cdots a_n \lambda^{t_n} + a$$

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

Multiple Objective Scheduling



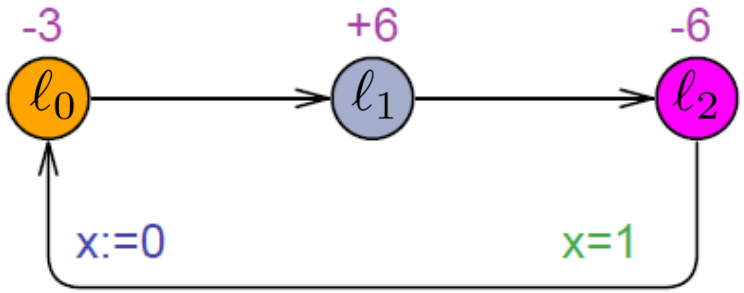
Consuming & Harvesting Energy

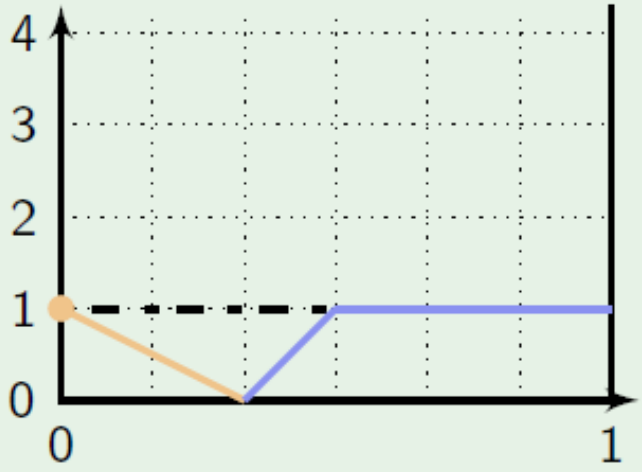


Maximize throughput
while respecting: 0 · E · MAX

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 (early)

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

Untimed

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

1 Clock

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

Integer Point Abstraction Suffice

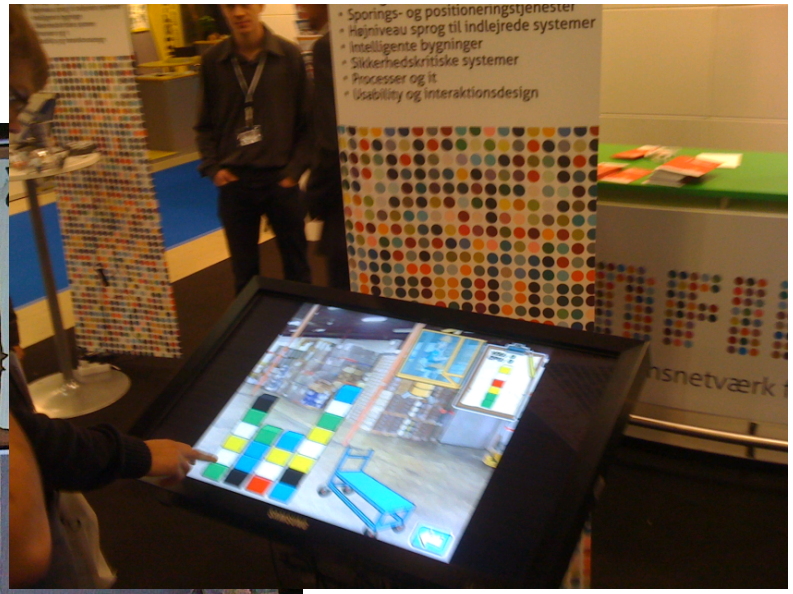
Results (new)

Fahrenberg, Juhl, Larsen, Legay, Srba: **ICTAC 2011**

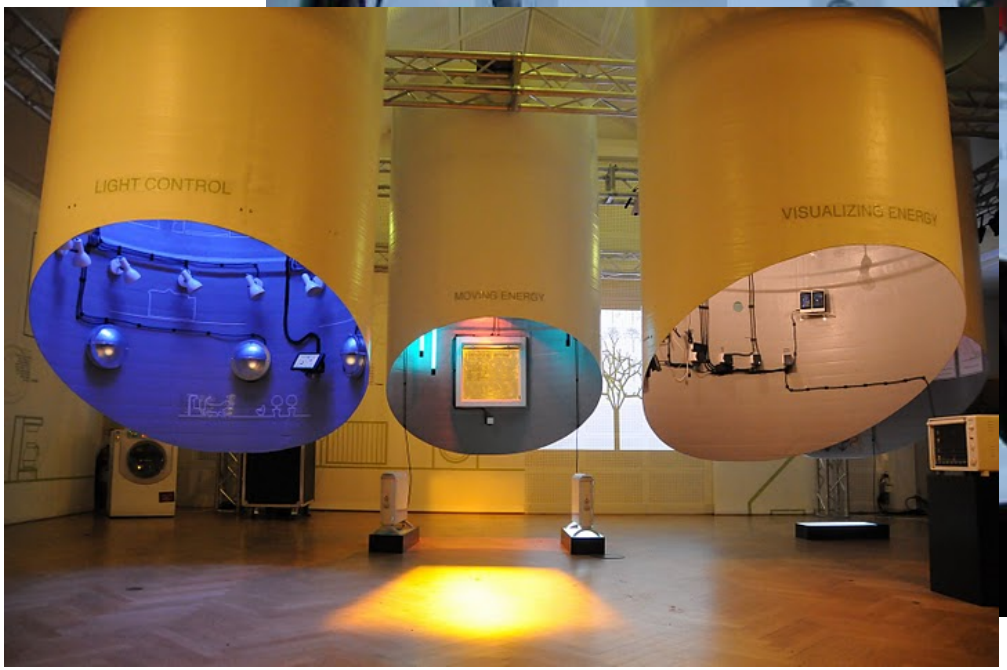
Untimed, Multiple Weights

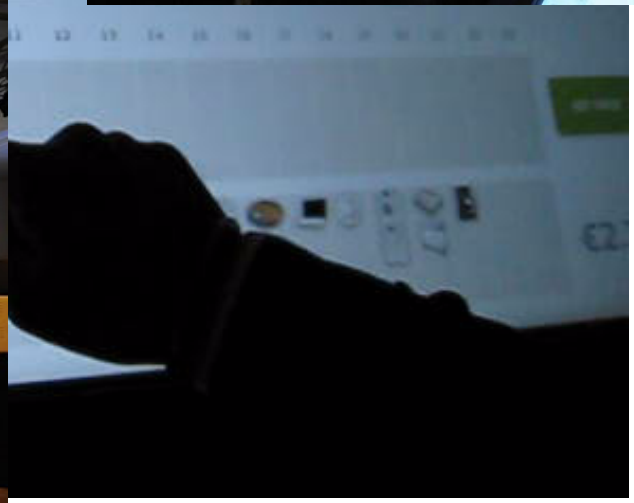
# weights	Bound	Existential	Universal	Game
One	L	$\in$ P [4]	$\in$ P [4]	$\in$ UP $\cap$ coUP [4]
	LW	$\in$ P [4]	$\in$ P [4]	$\in$ NP $\cap$ coNP [4]
	LU	NP-hard [4], $\in$ PSPACE [4]	$\in$ P [4]	EXPTIME-complete [4]
Fixed ($k > 1$)	L	**NP-hard**, $\in$ k-EXPTIME [3] (Remark 17)	$\in$ P (Remark 18)	**EXPTIME-hard**, $\in$ k-EXPTIME [3] (Remark 19)
	LW	**NP-hard**, $\in$ **PSPACE** **PSPACE-complete** for $k \geq 4$ (Remark 20)	$\in$ P (Remark 18)	**EXPTIME-complete** (Remark 21)
	LU	**PSPACE-complete** (Remark 20)	$\in$ P (Remark 18)	**EXPTIME-complete** (Remark 21)
Arbitrary	L	**EXPSPACE-complete** (Theorem 9)	$\in$ P (Remark 18)	**EXPSPACE-hard** (from EL) decidable [3]
	LW	**PSPACE-complete** (Theorem 9)	$\in$ P (Remark 18)	**EXPTIME-complete** (Remark 21)
	LU	**PSPACE-complete** (Theorem 9)	$\in$ P (Remark 18)	**EXPTIME-complete** (Remark 21)

”Experimental” Results

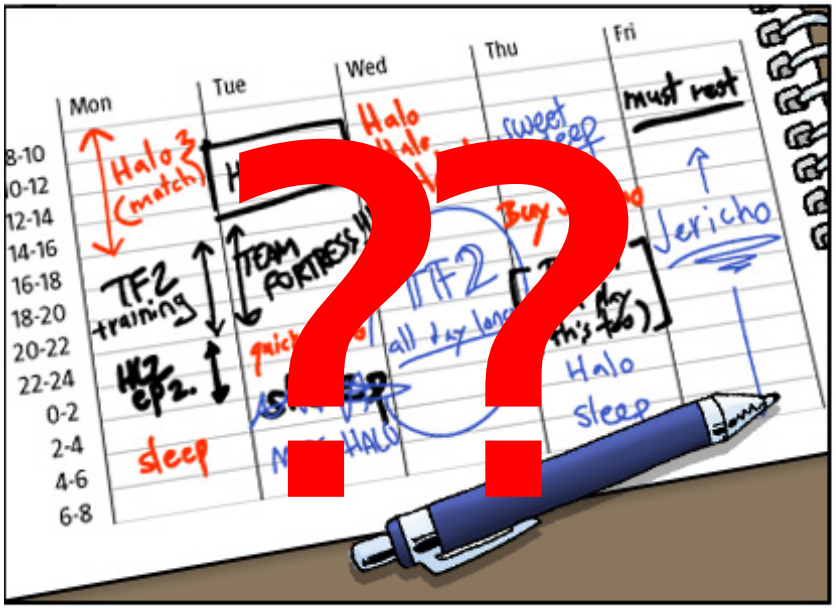


”Experimental” Results



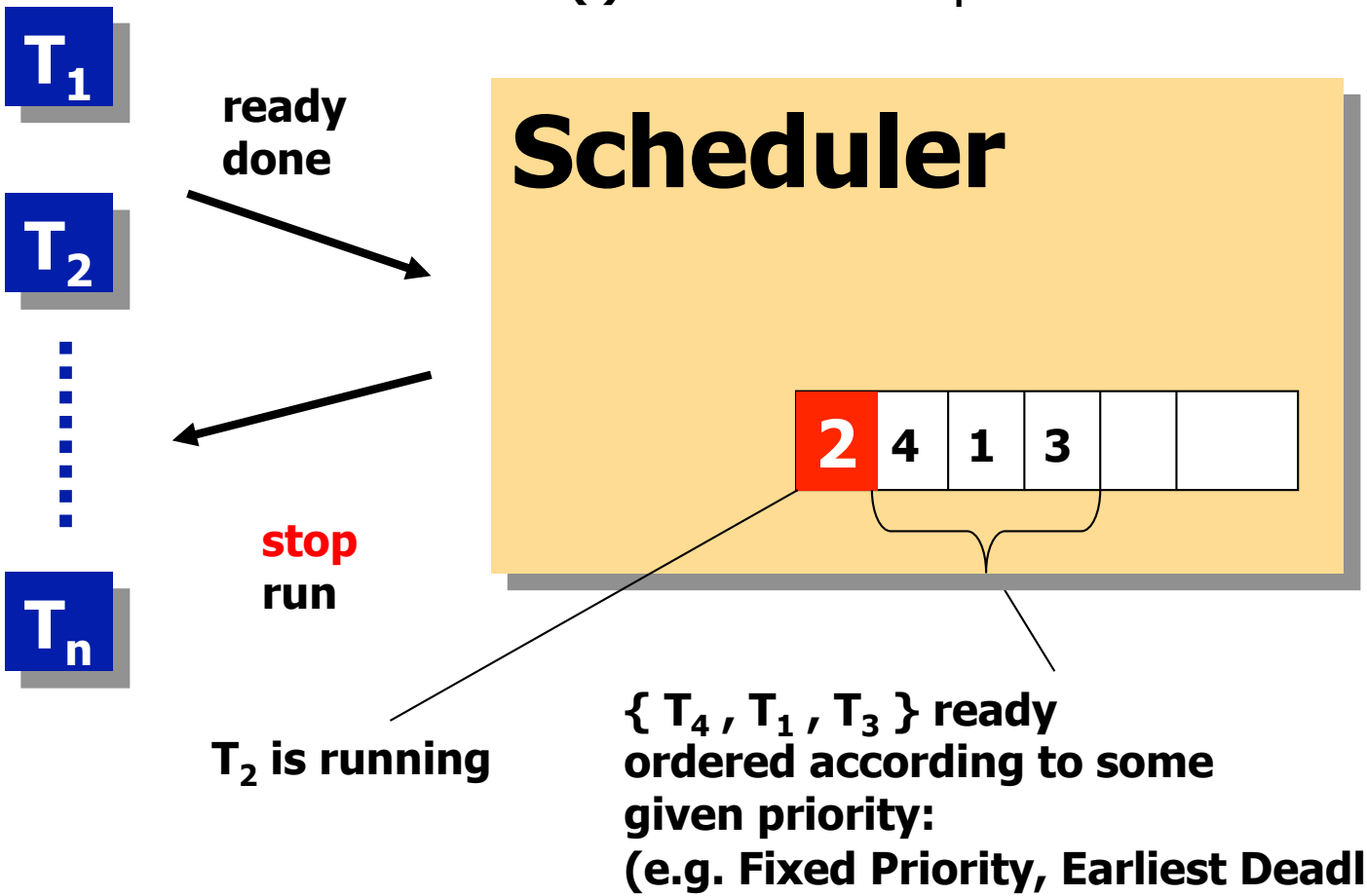


Schedulability Analysis



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



Classical Scheduling Theory

Utilisation-Based Analysis

- A simple sufficient but not necessary schedulability test exists

$$U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N(2^{1/N} - 1)$$

$$U \leq 0.69 \text{ as } N \to \infty$$

Where C is WCET and T is period

41

Response Time Equation

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Where $hp(i)$ is the set of tasks with priority higher than task i

Solve by forming a recurrence relationship:

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

The set of values $w_i^0, w_i^1, w_i^2, \ldots, w_i^n, \ldots$ is monotonically non decreasing
When $w_i^n = w_i^{n+1}$ the solution to the equation has been found, w_i^0 must not be greater that R_i (e.g. 0 or C_i)

42

Classical WCRT Analysis

Quasimodo

- "Classical" scheduling analysis technique
- *For all tasks i: $WCRT_i \leq Deadline_i$*

$$R_i = B_i + C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Blocking times for priority inheritance protocol (BSW):

$$Blocking(i) = \sum_{r=1}^{R} usage(r,i) WCET_{CriticalSection}(r)$$

Blocking times for priority ceiling protocol (ASW):

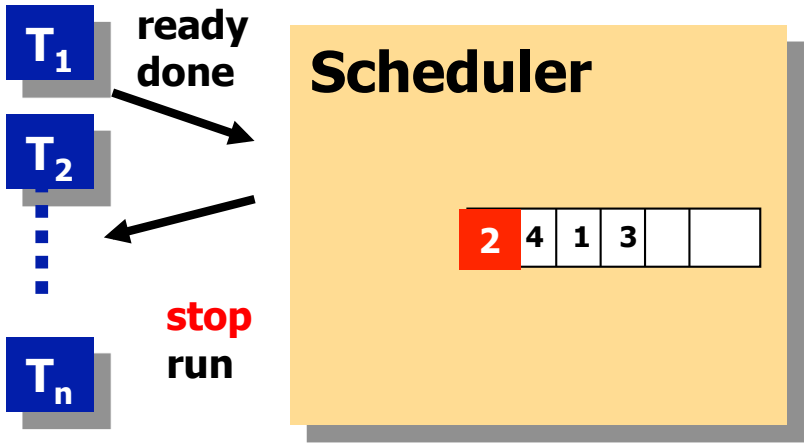
$$Blocking(i) = \max_{r=1}^{R} usage(r,i) WCET_{CriticalSection}(r)$$

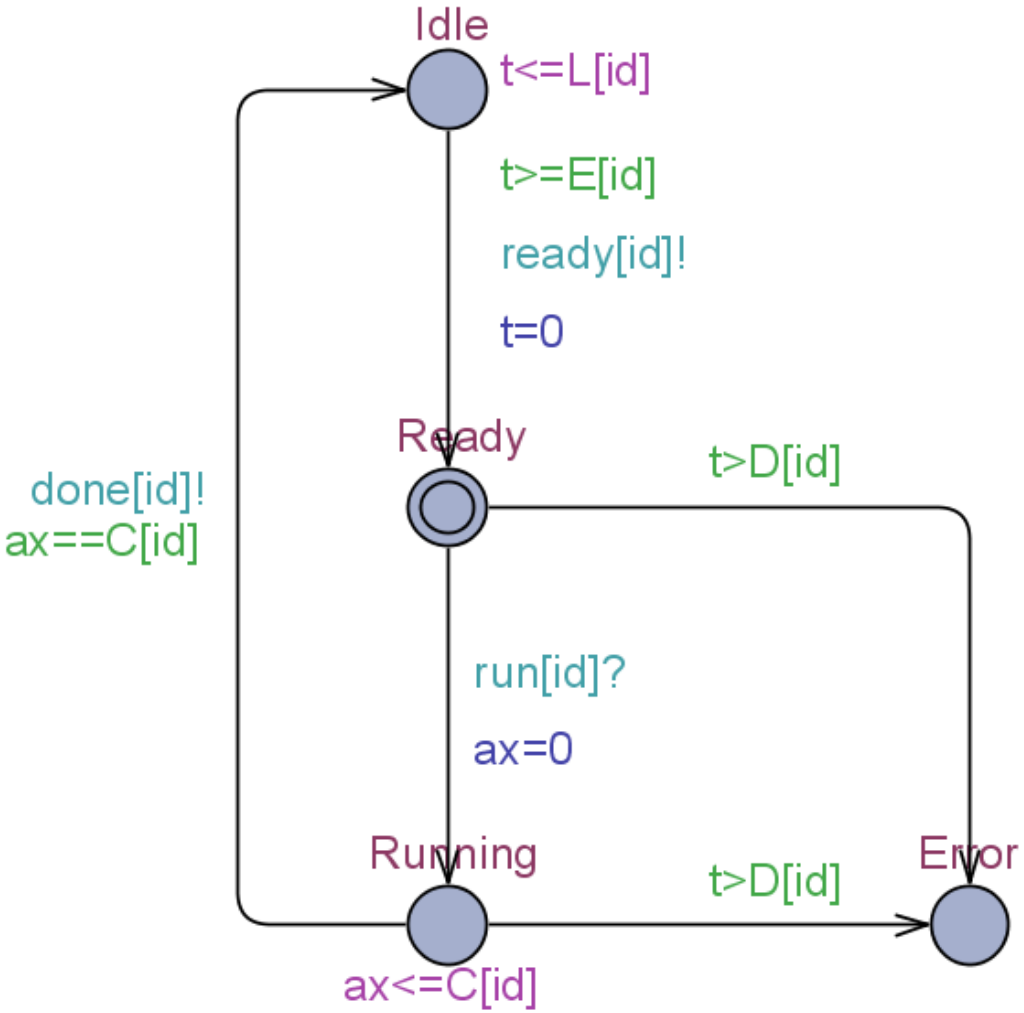
Quasimodo Workshop, Eindhoven, Nov 6, 2009 Page 21

✓ Simple to perform

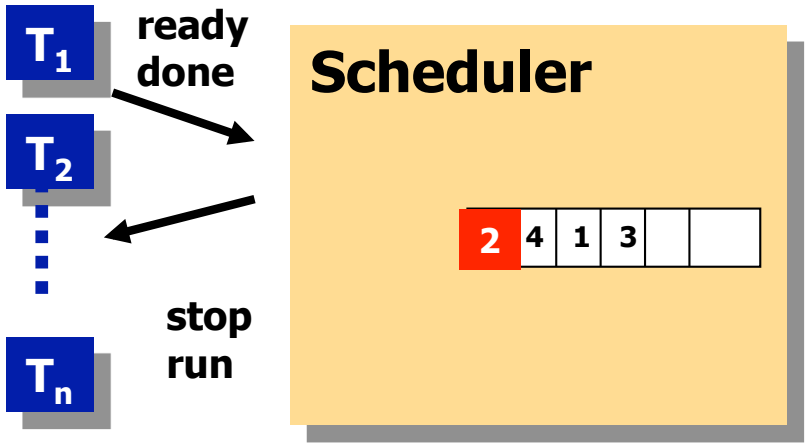
- Overly conservative
- Limited settings
- Single-processor

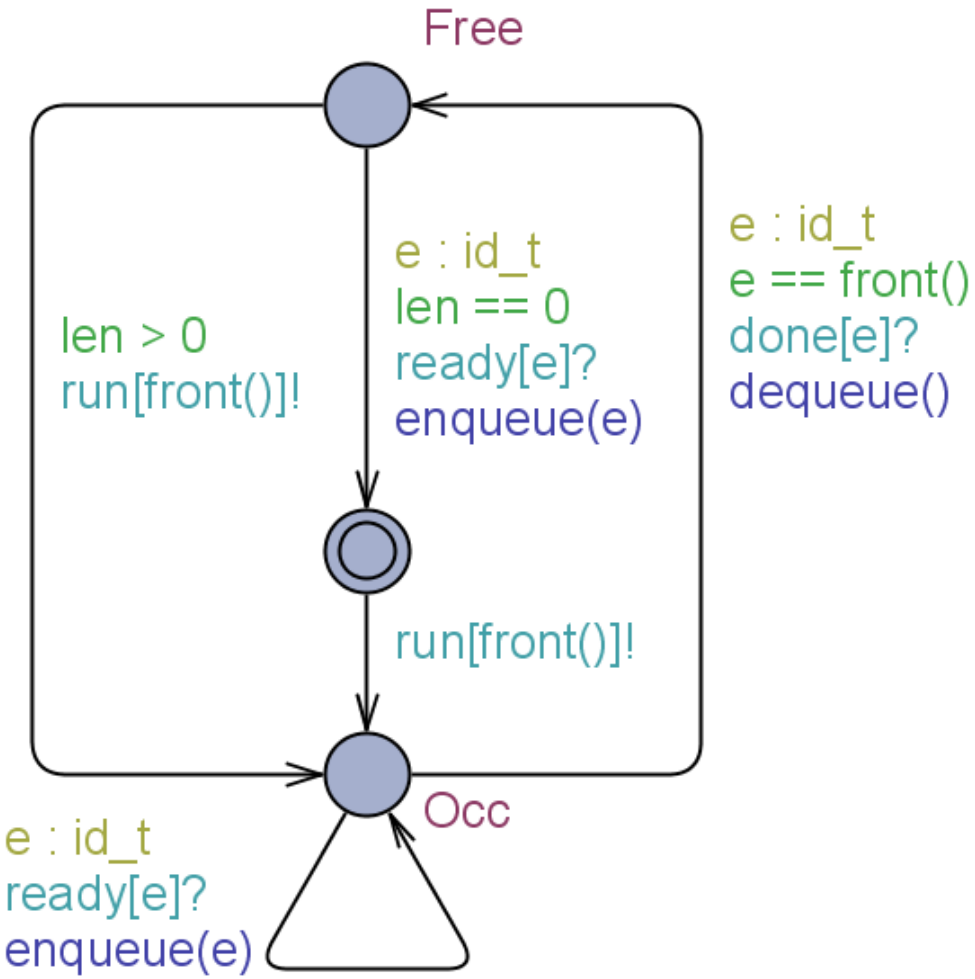
Modeling Task





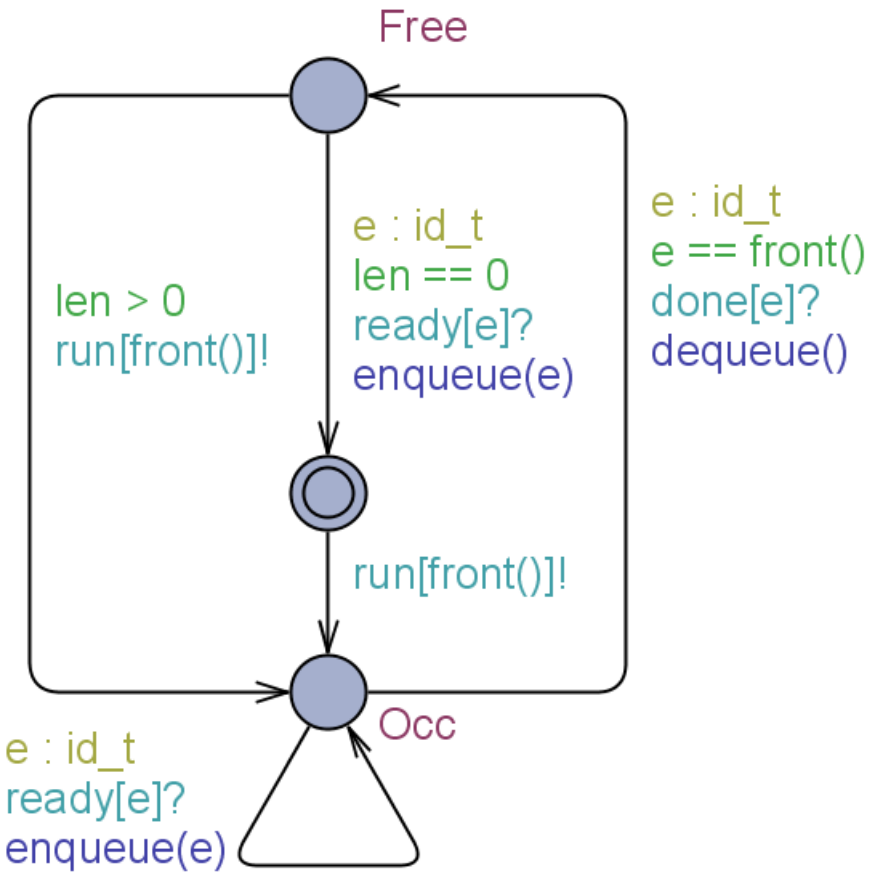
Modeling Scheduler





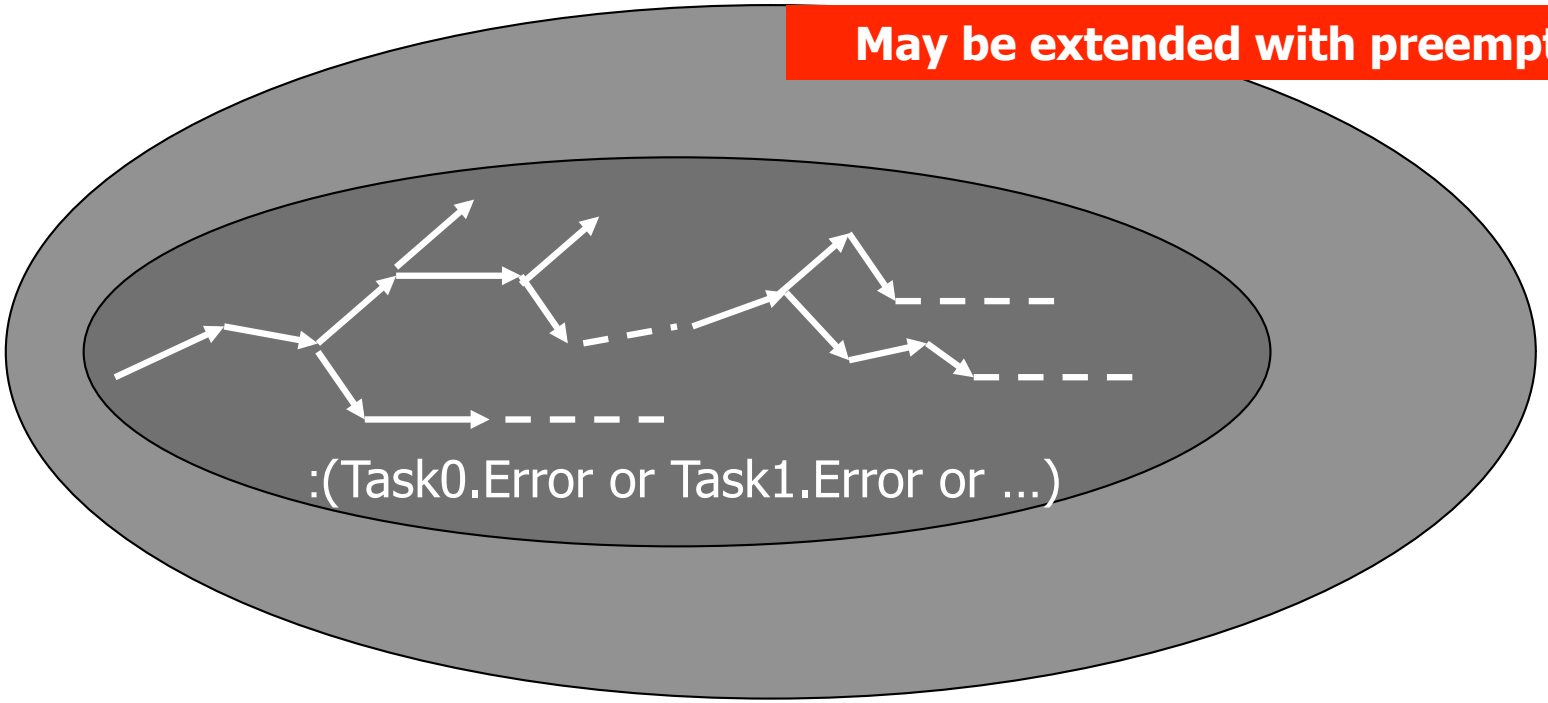
Modeling Queue

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()
{
  ......
```

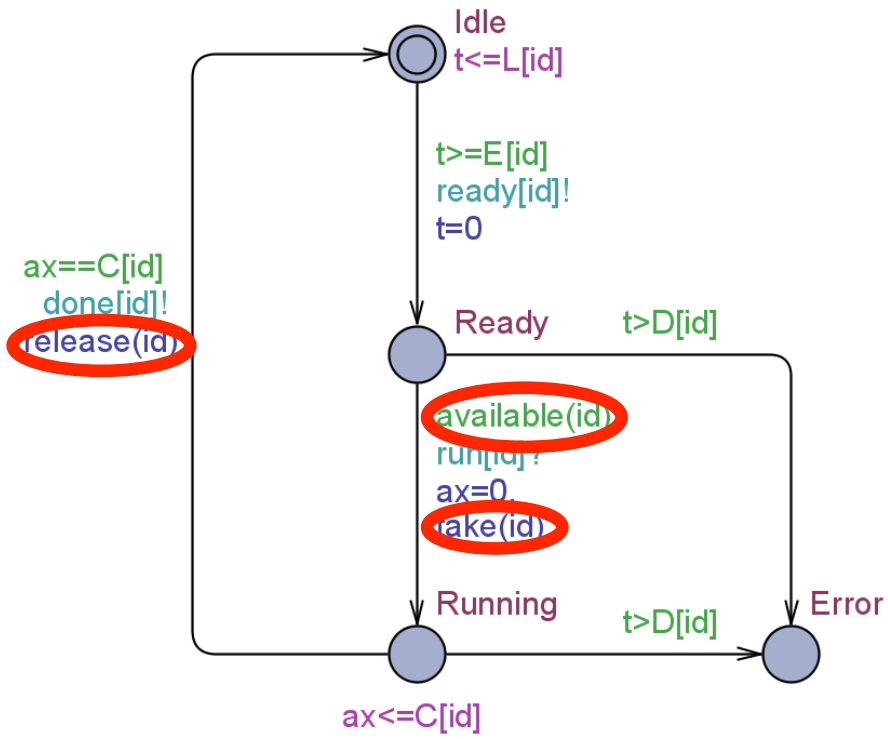
Schedulability = Safety Property



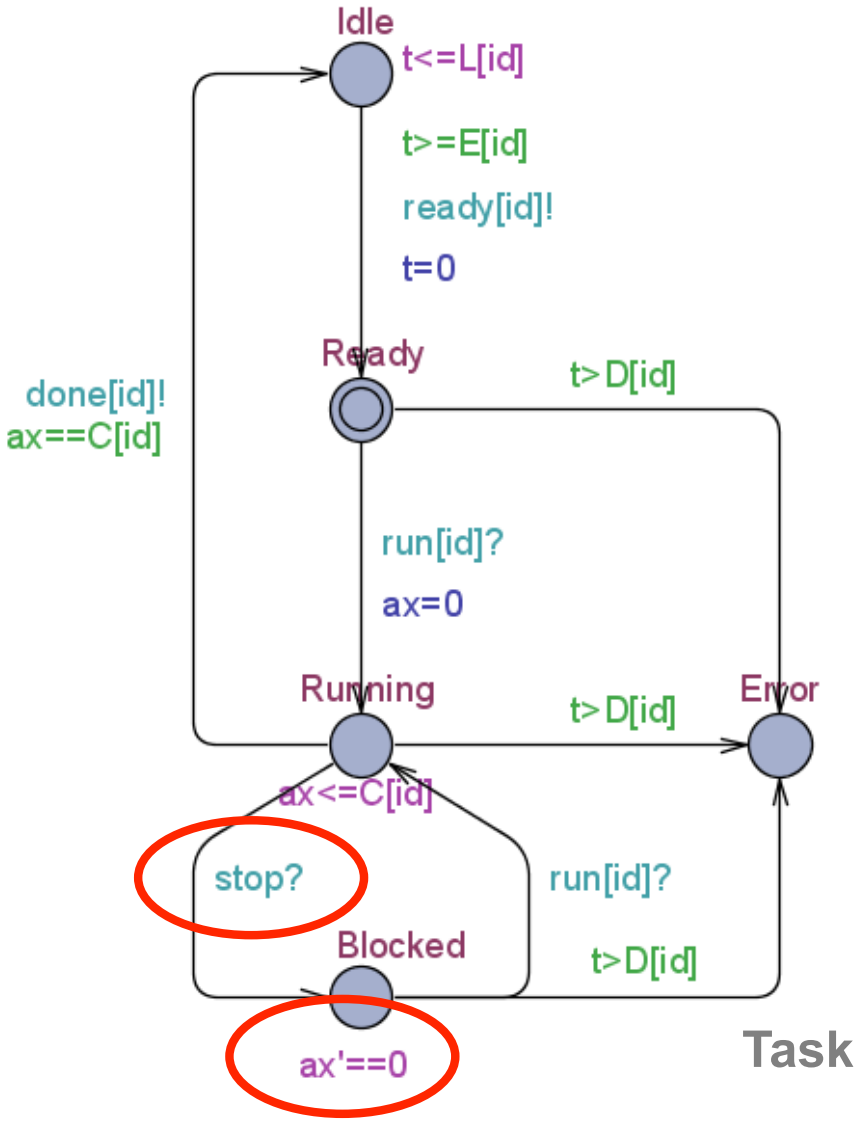
A :(Task0.Error or Task1.Error or ...)

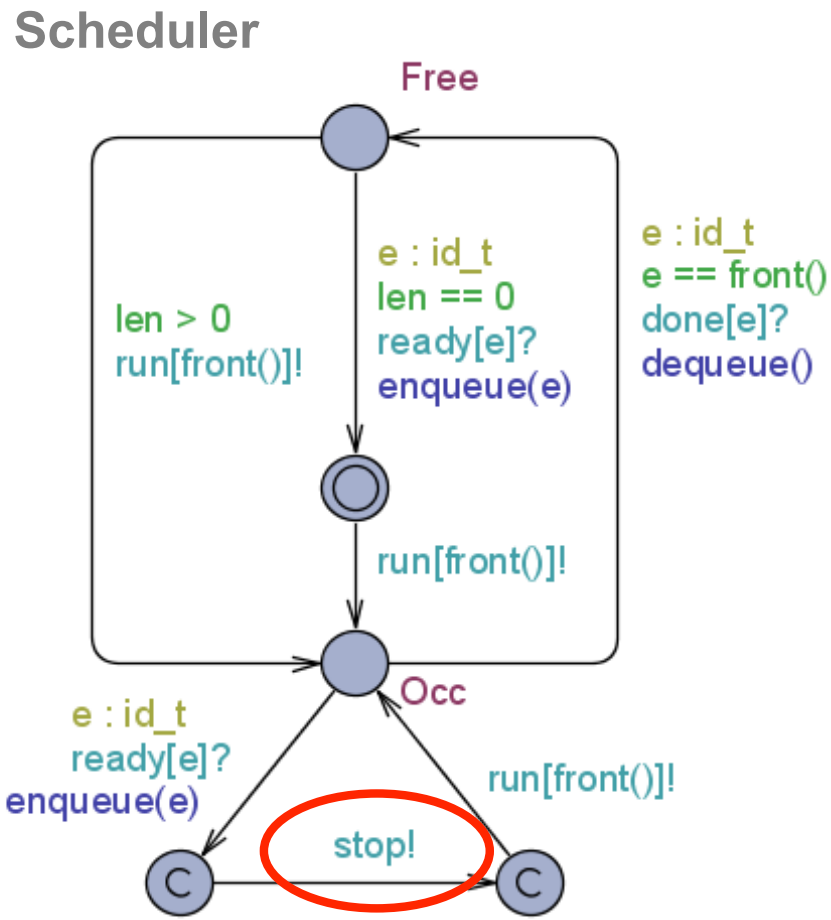
Dealing with Resources

```
bool resource[N];

bool available(id_t id)
{
    return !resource[need[id]];
}

void take(id_t id)
{
    assert(!resource[need[id]]);
    resource[need[id]] = true;
}

void release(id_t id)
{
    assert(resource[need[id]]);
    resource[need[id]] = false;
}
```



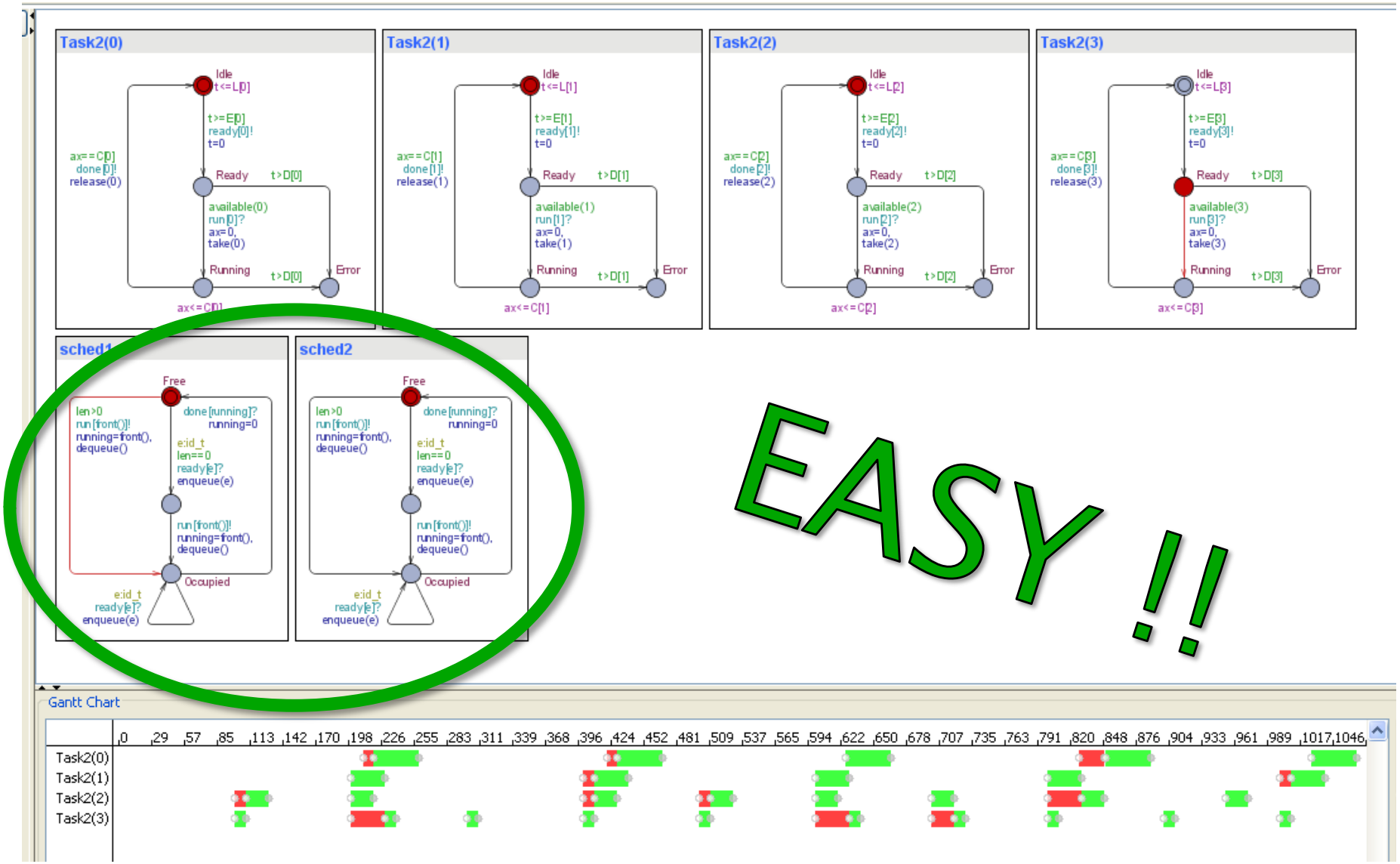
Preemption – Stopwatches!





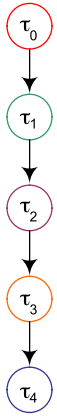
Defeating undecidability ☺

Multi-Processor

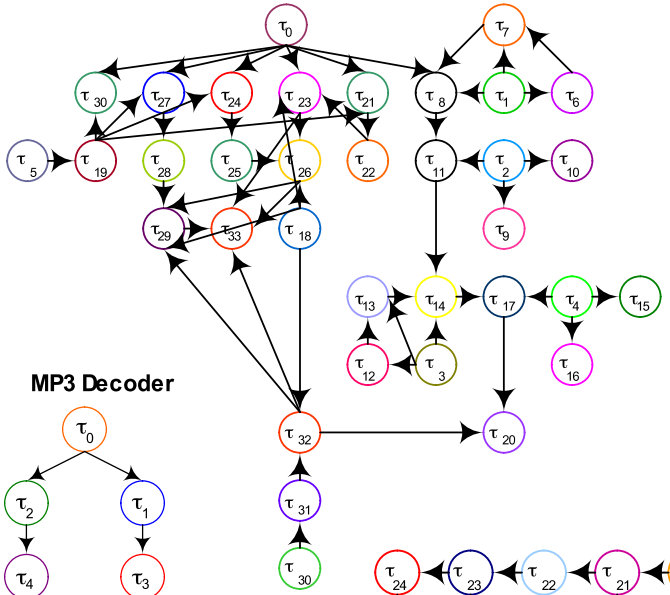


Handling realistic applications?

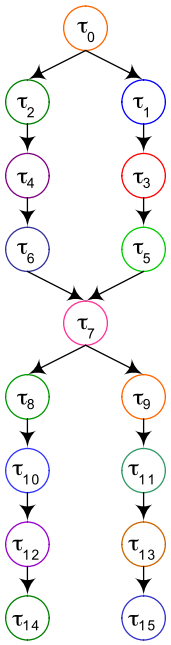
JPEG Encoder



GSM Decoder

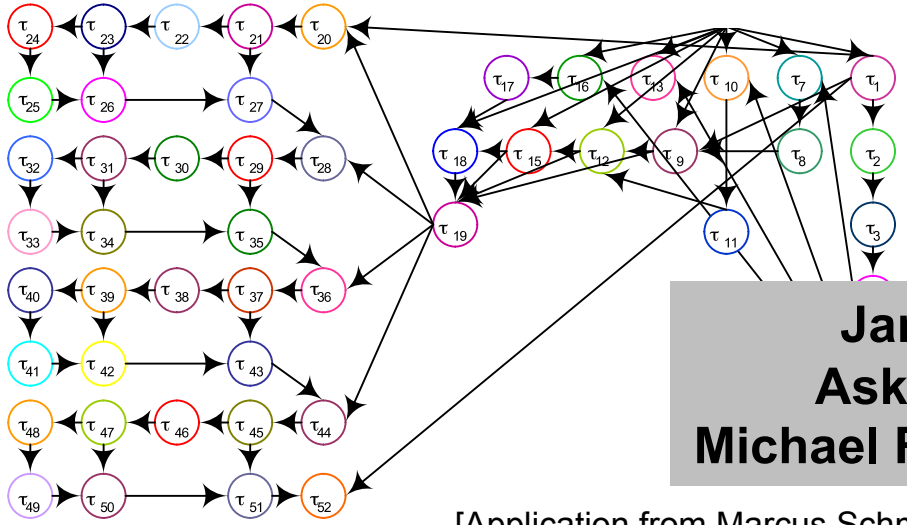


MP3 Decoder



JPEG Decoder

GSM Encoder

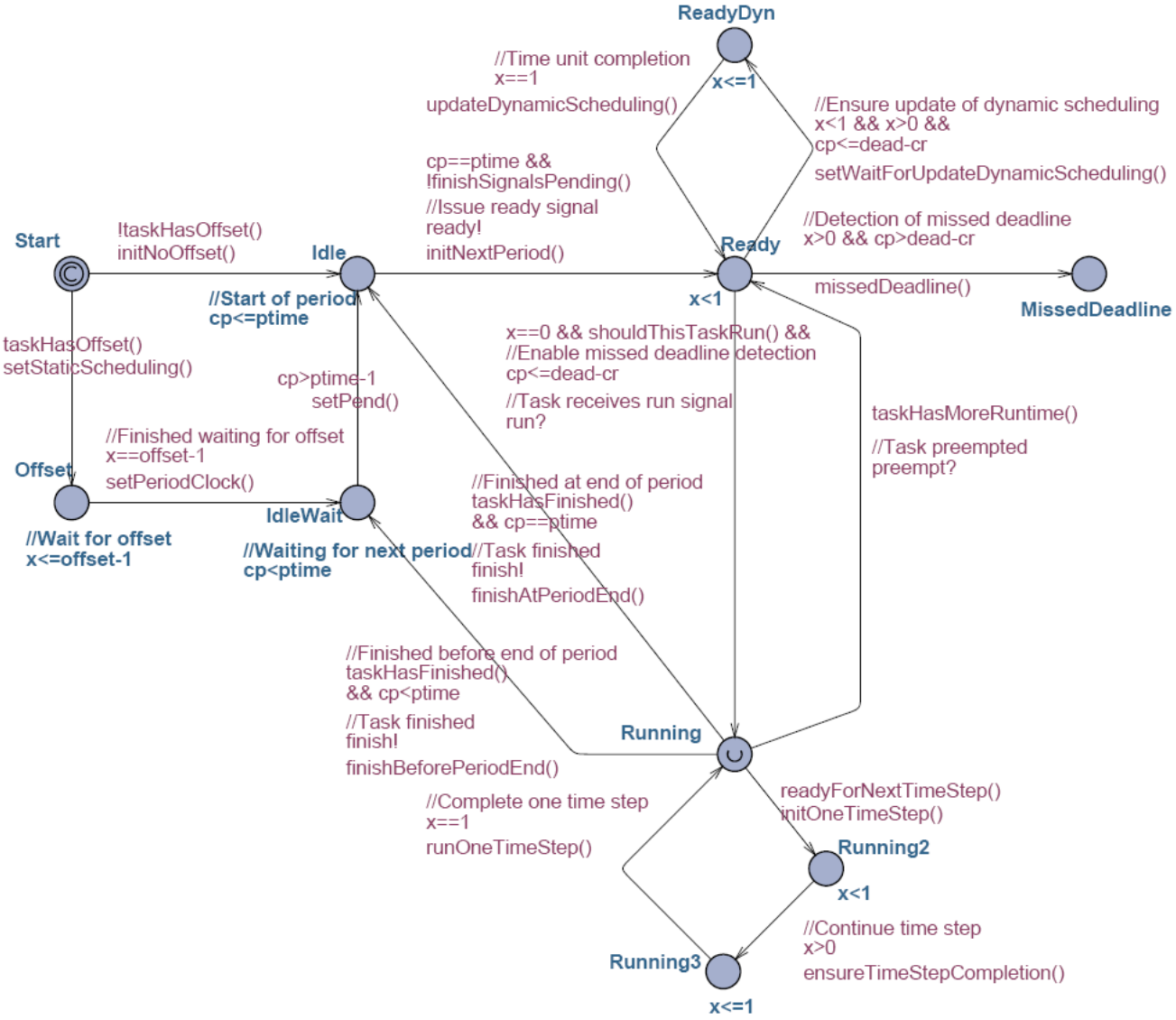


Smart phone:

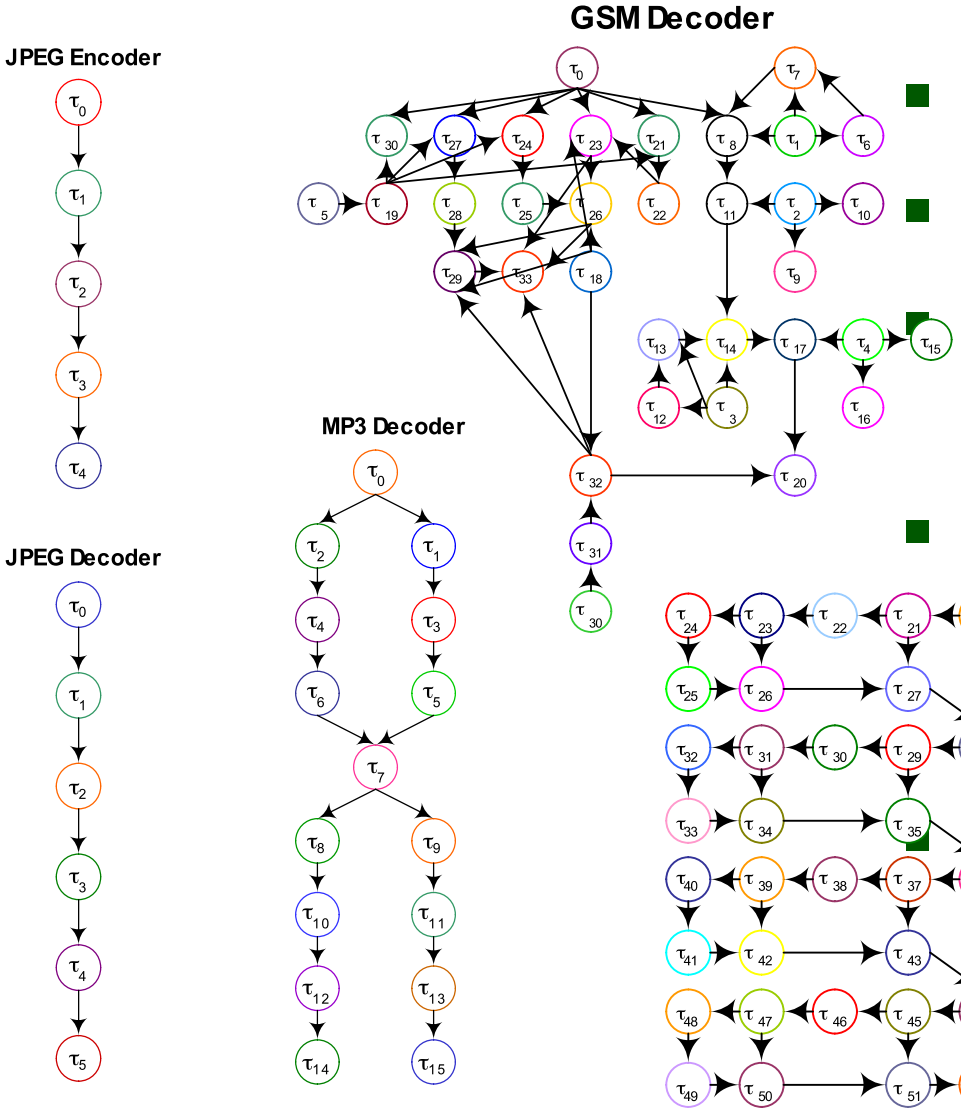
Jan Madsen
Aske Brekling
Michael R. Hansen/ DTU

[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

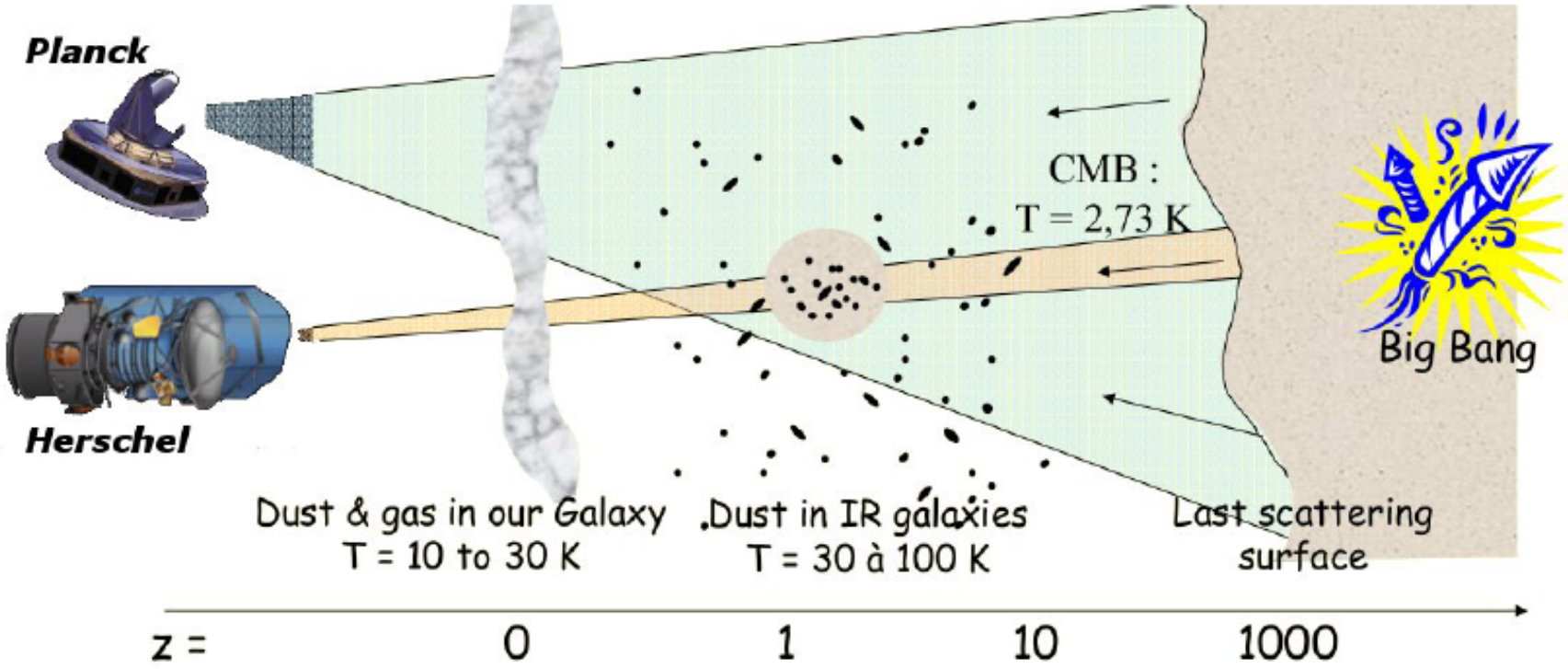
Verified in 1.5 hours!

ESA Missions

TERMA

Quasiomodo

- Solar System, cold dust clouds and cores, star and galaxy formations, cataloging galaxies, gravitational lensing, cosmic microwave background, topology of the universe...

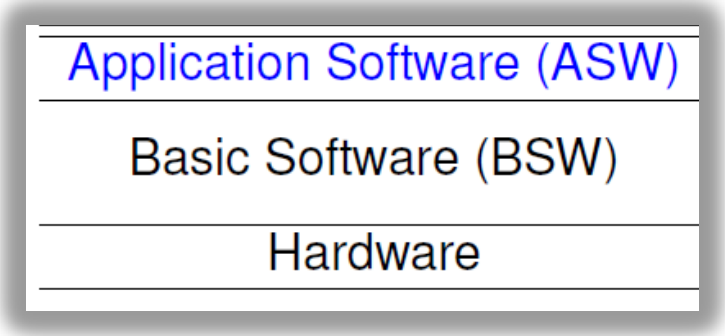


- Terma: Develop software for Attitude and Orbit Control System

Herschel & Planck Satelites

TERMA

- 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

UPPAAL Model

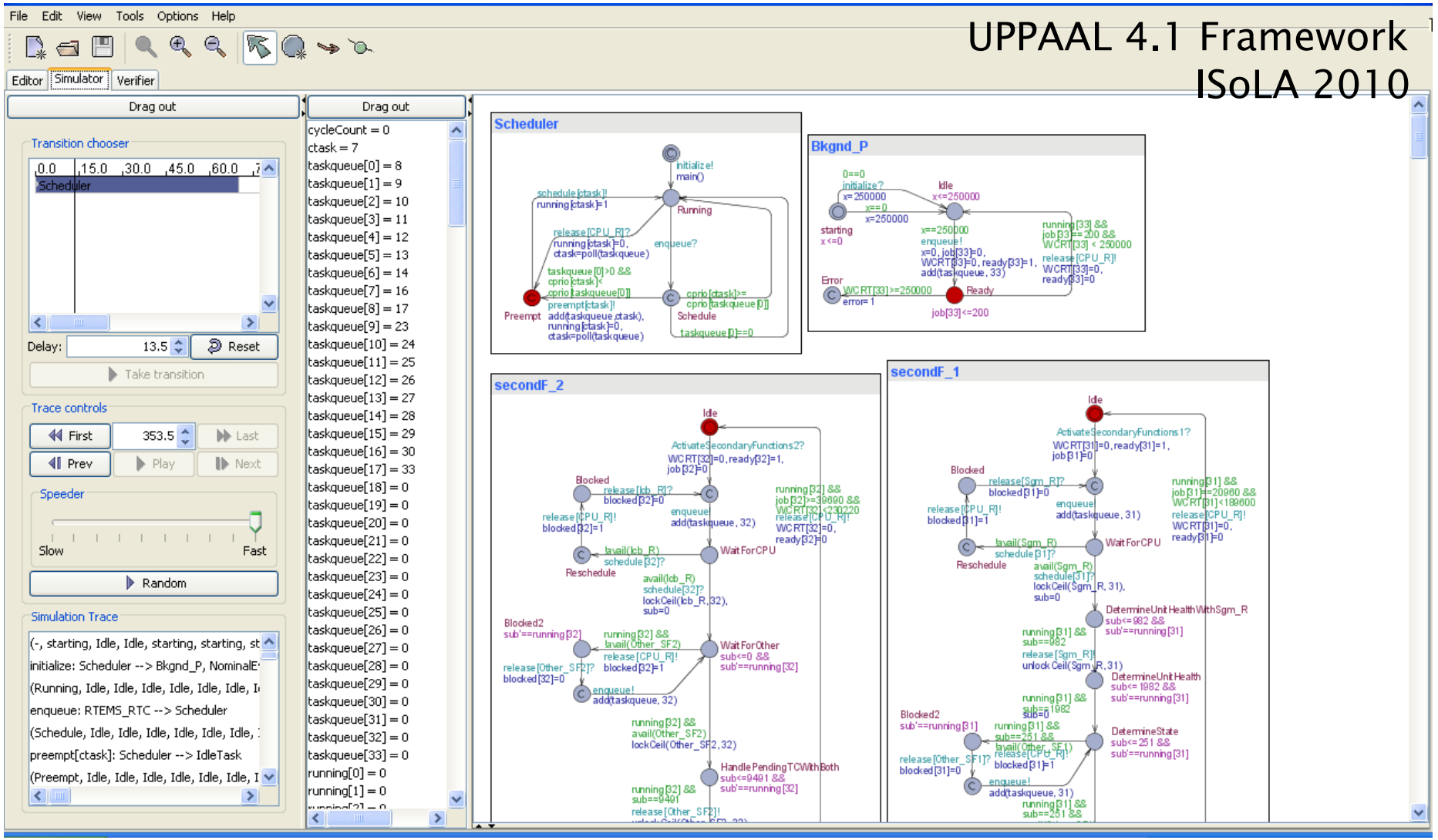
TERMA

- One template for CPU scheduler:
  - maintains a queue of ready tasks
  - schedules tasks with highest priority in the queue
  - reschedule if higher priority task arrives to the queue
- One template per **each** ASW task.
- Two templates for BSW tasks: 1 plain, 1 using resource.
- One template for "idle" task to count CPU utilization.
- WCET is modeled by stopwatches and lower bound.
- WCRT is modeled by stopwatches.
- Deadline is enforced as guard on WCRT.
- System is schedulable if no deadline violated.
- CPU load is $\frac{\text{used time}}{\text{total time}}$.

Modeling in UPPAAL

TERMA

UPPAAL 4.1 Framework
ISoLA 2010



Gantt Chart 1. cycle

TERMA

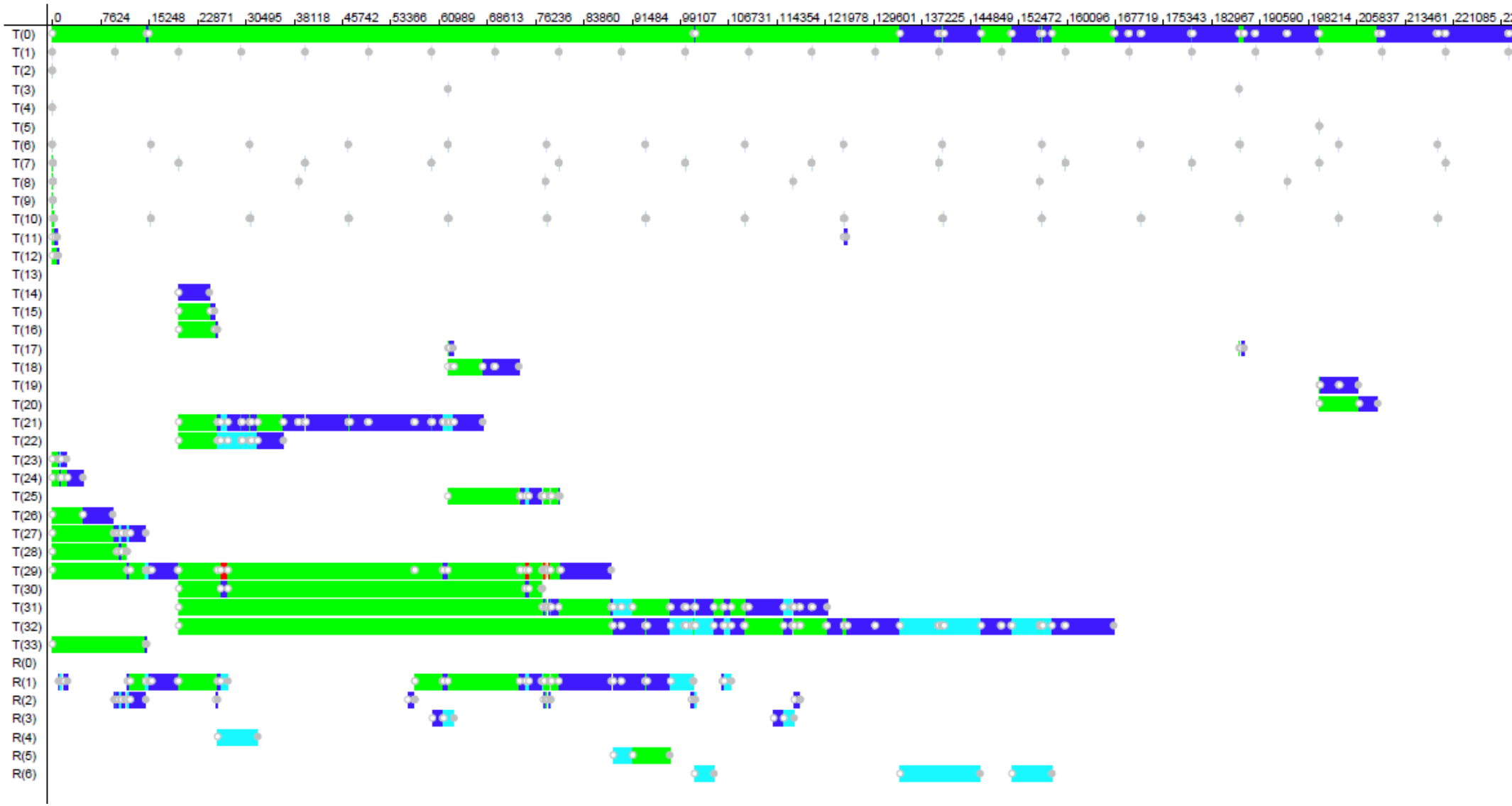


Fig. 11. Gantt chart of a schedule from the first cycle: green means ready, blue means running, cyan means suspended, red means blocked. R stand for resources: CPU_R=0, Icb_R=1, Sgm_R=2, PmReq_R=3, Other_RCS=4, Other_SF1=5, Other_SF2=6.

Blocking & WCRT

TERMA

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

Effort and Utilization

TERMA

cycle limit	Uppaal resources			Herschel CPU utilization				
	CPU, s	Mem, KB	States, #	Idle, μs	Used, μs	Global, μs	Sum, μs	Used, %
1	465.2	60288	173456	91225	160015	250000	251240	0.640060
2	470.1	59536	174234	182380	318790	500000	501170	0.637580
3	461.0	58656	175228	273535	477705	750000	751240	0.636940
4	474.5	58792	176266	363590	636480	1000000	1000070	0.636480
6	474.6	58796	178432	545900	955270	1500000	1501170	0.636847
8	912.3	58856	352365	727110	1272960	2000000	2000070	0.636480
13	507.7	58796	186091	1181855	2069385	3250000	3251240	0.636734
16	1759.0	58728	704551	1454220	2545850	4000000	4000070	0.636463
26	541.9	58112	200364	2363640	4137530	6500000	6501170	0.636543
32	3484.0	75520	1408943	2908370	5091700	8000000	8000070	0.636463
39	583.5	74568	214657	3545425	6205745	9750000	9751170	0.636487
64	7030.0	91776	2817704	5816740	10183330	16000000	16000070	0.636458
78	652.2	74768	257582	7089680	12411420	19500000	19501100	0.636483
128	14149.4	141448	5635227	11633480	20366590	32000000	32000070	0.636456
156	789.4	91204	343402	14178260	24821740	39000000	39000000	0.636455
256	23219.4	224440	11270279	23266890	40733180	64000000	64000070	0.636456
312	1824.6	124892	686788	28356520	49643480	78000000	78000000	0.636455
512	49202.2	390428	22540388	46533780	81466290	128000000	128000070	0.636455
624	3734.7	207728	1373560	56713040	99286960	156000000	156000000	0.636455

Marius Micusionis

TERMA Case Conclusion

- Schedulability analysis using UPPAAL:
  - Reusable and customizable task templates.
  - *Blocking* times and WCRTs can be derived from the model.
  - WCRTs of all tasks are more optimistic than in RTA.
  - There are very few blocking times and they are short.
  - PrimaryF meets deadline (59.6ms) with WCRT=54.1ms (65.5ms in RTA).
  - Herschel event mode is **schedulable**.
- UPPAAL verification for schedulability:
  - can be **scaled** using sweep-line method,
  - takes up to **2min** to verify schedulability of 32 task system,
  - takes up to **8min** to find all WCRTs and CPU utilization.
- In addition, it is possible to:
  - simulate the system model and examine details,
  - render a Gantt chart, validate and inspect visually.

TERMA Case Follow-Up

% : difference In BCET and WCET

limit: no of 250ms cycles

limit	0%			5%		
	states	mem	time	states	mem	time
1	1300	51.2	1.47	485077		1
2	2522	53.7	2.45	806914		9
4	4981	54.5	4.62	1499700	2	2881.8
8	9					
16	19					
∞	196					

10%			14%		
states	mem	time,	states	mem	time
1481162	124.1	4962	3348246	186.9	23986.5
2414679	139.7	775 2	5253778	198.7	33299.2
4421630	138.3	137 .0	9231399	274.6	51176.6
9093562	156.5	31122.3	18240030	364.6	102932.4
17798572	176.0	60124.5	35432003	520.4	158816.7
181869652	1682.2	530604.9	error may be reachable		

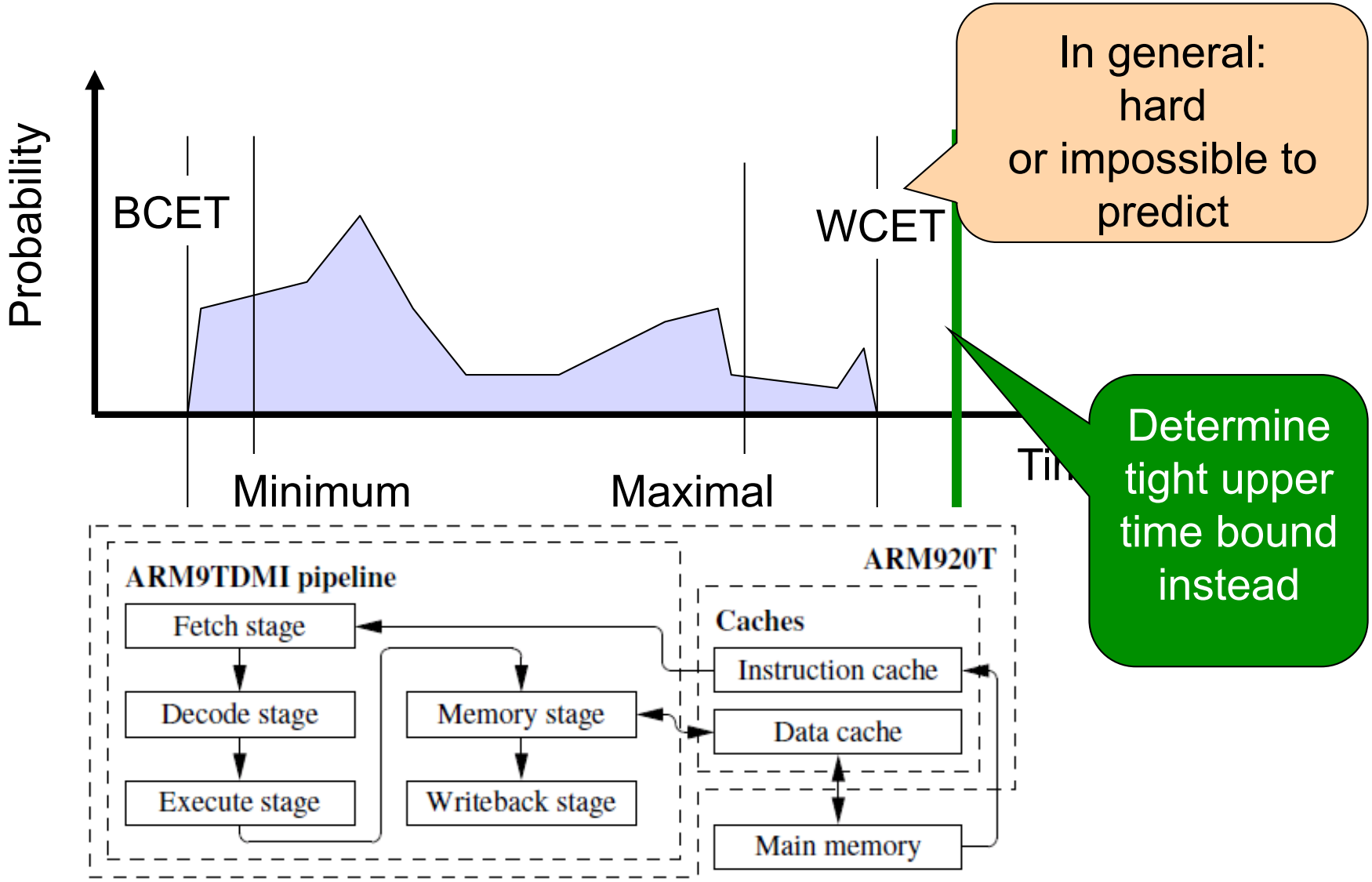
6 Days

WCET Analysis

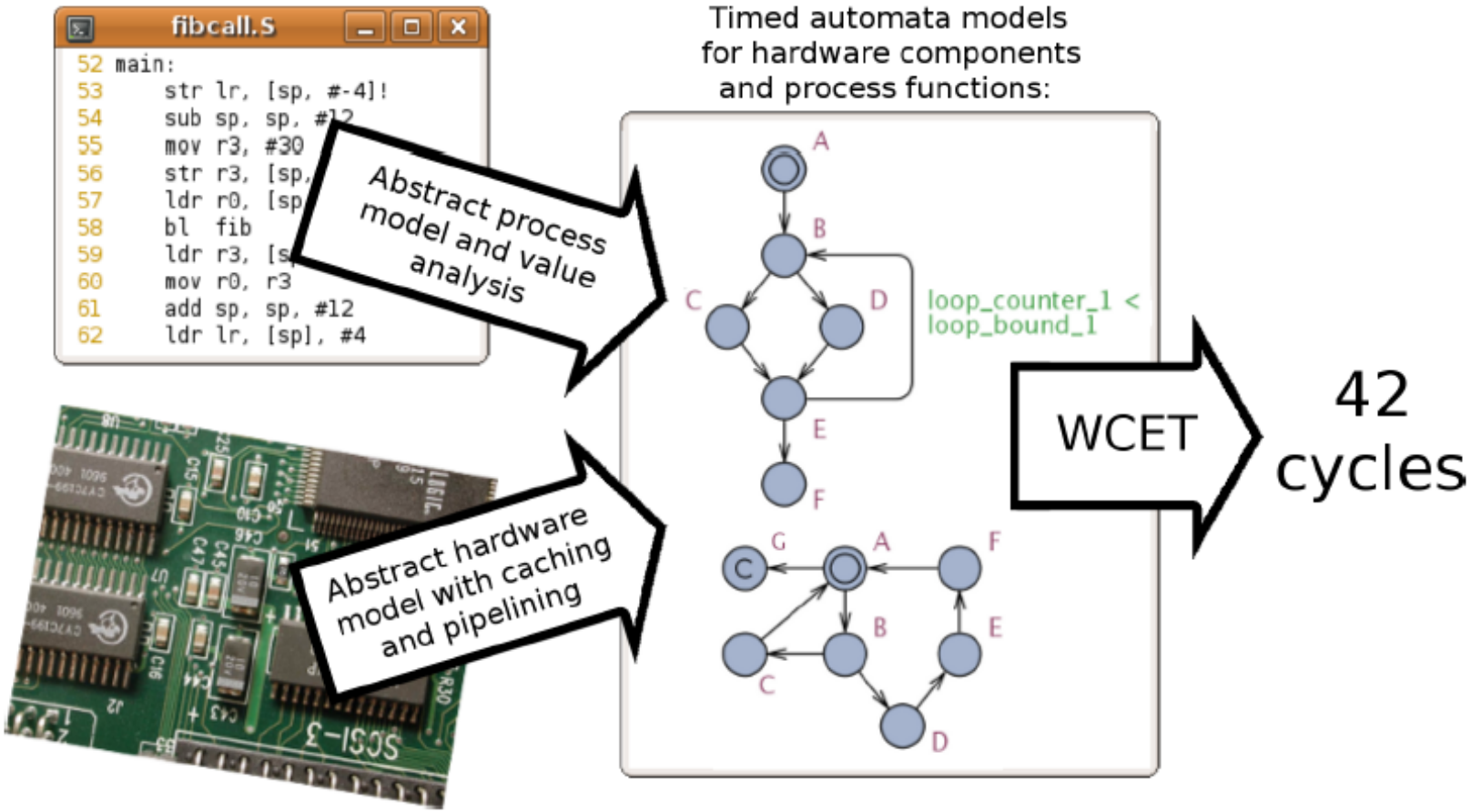
METAMOC: Modular Execution Time Analysis using MOdel Checking

with
Andreas Dalsgaard
Mads Christian Olesen
Martin Toft
René Rydhof Hansen

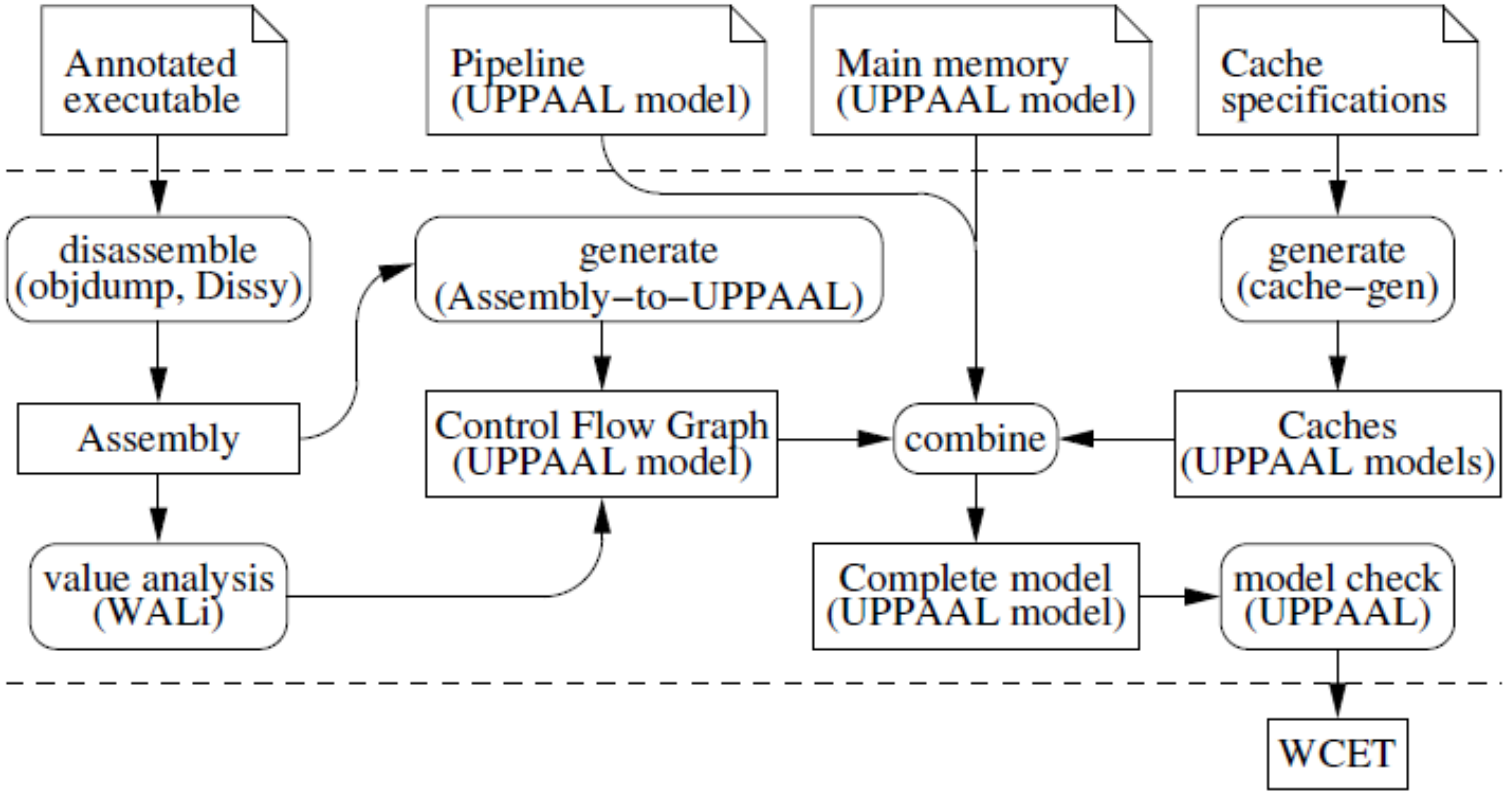
WCET: Worst Case Execution Time



METAMOC


fibcall.S
52 main:
53 str lr, [sp, #-4]!
54 sub sp, sp, #12
55 mov r3, #30
56 str r3, [sp,
57 ldr r0, [sp
58 bl fib
59 ldr r3, [s
60 mov r0, r3
61 add sp, sp, #12
62 ldr lr, [sp], #4
Abstract process model and value analysis
Abstract hardware model with caching and pipelining
Timed automata models for hardware components and process functions:
A
B
C
D
E
F
loop_counter_1 < loop_bound_1
G
WCET
42 cycles

Overview of METAMOC

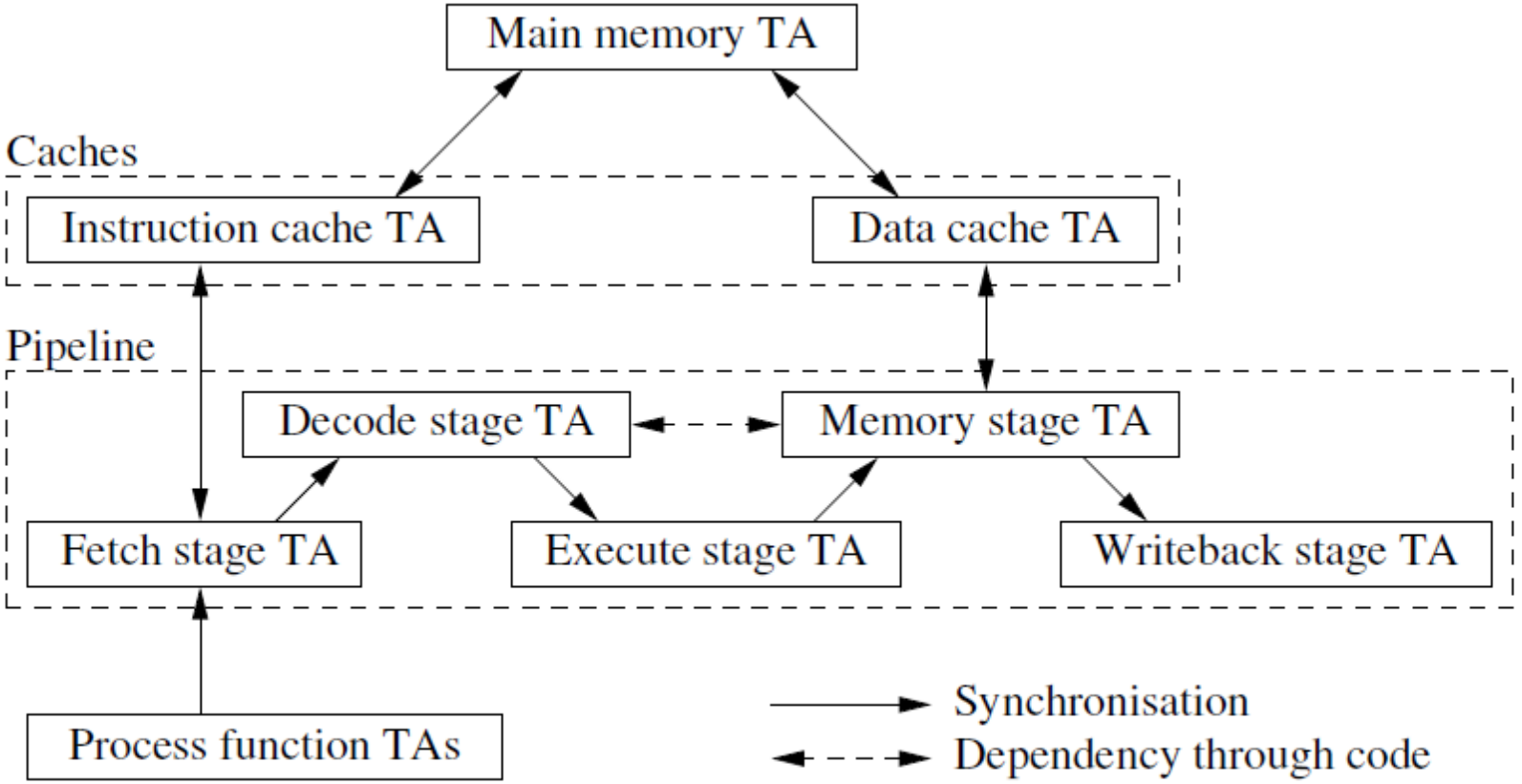


Value analysis in METAMOC

- Memory addresses needed for cache hit/miss predictions
- Registers used as base and offset for memory accesses
- Overapproximate possible register values
- METAMOC uses Weighted Push-Down Systems (WPDSs) for an inter-procedural, control-flow sensitive value analysis[1]
- Weighted Automata Library (WALi) utilised

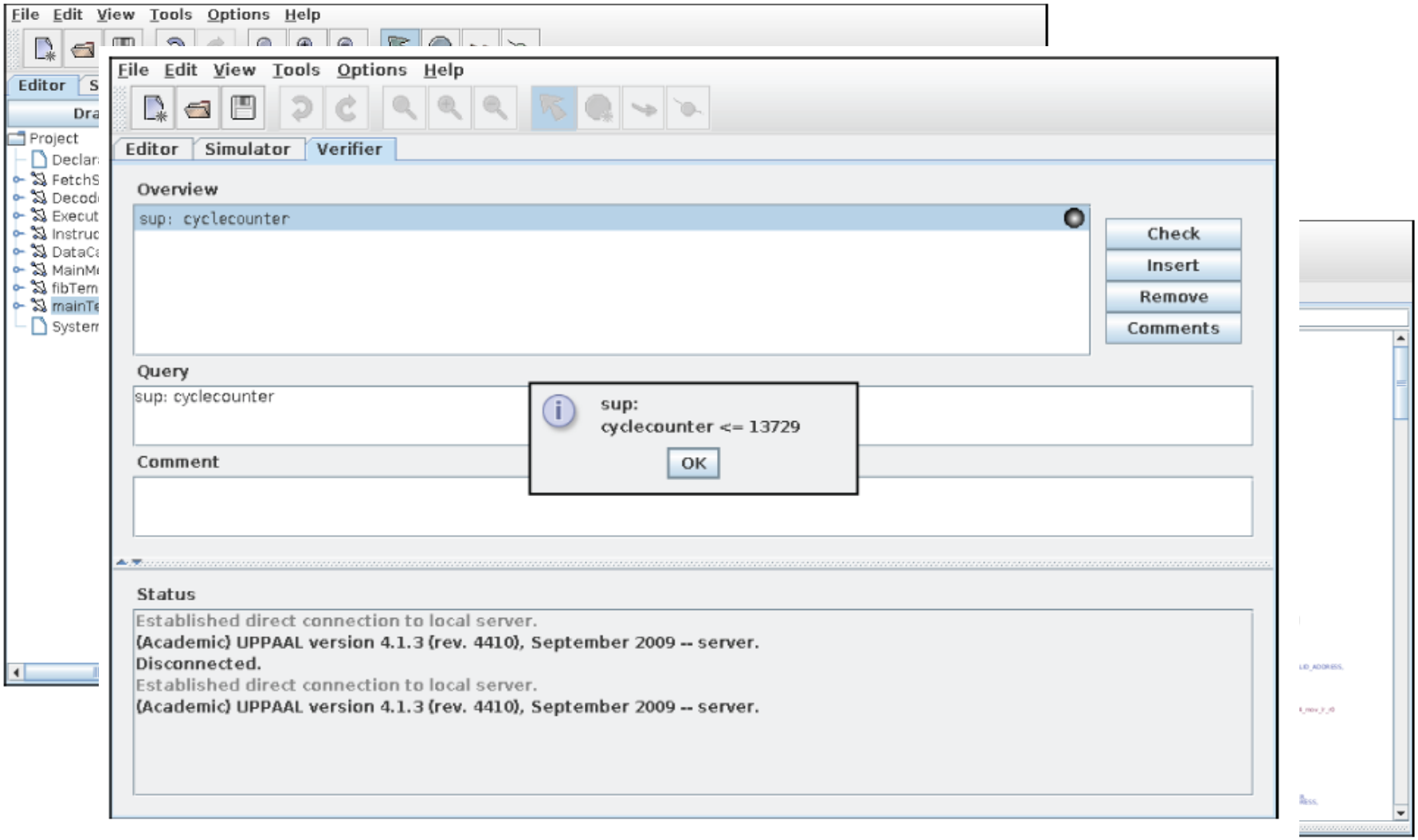
[1] T. Reps, A. Lal, N. Kidd. *Program Analysis using Weighted Push-Down Systems*. In *FSTTCS 2007*, vol. 4855 of *LNCS*, pp. 23–51.

Modeling in METAMOC

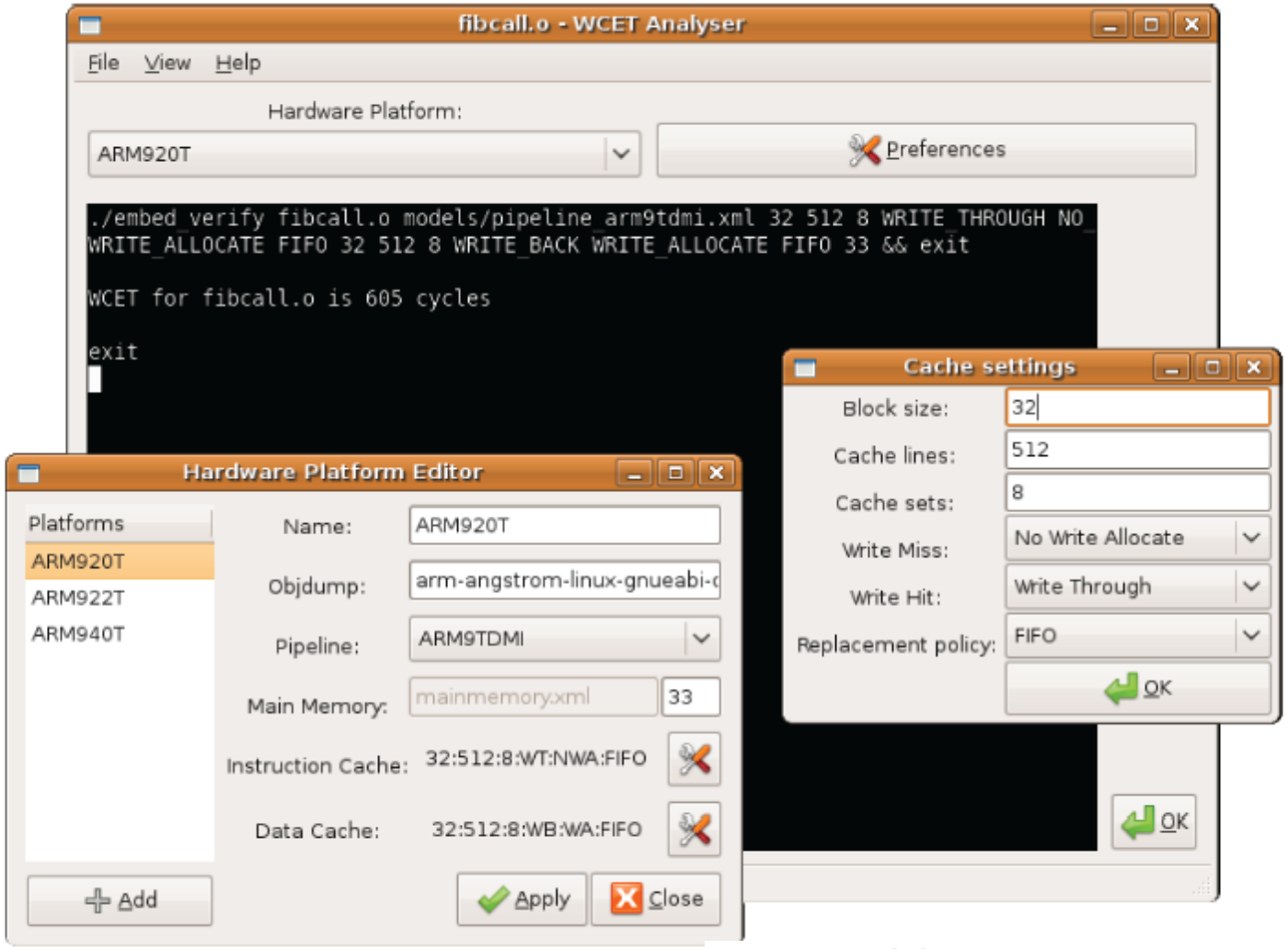


Overview of the ARM9 automata

Modeling in UPPAAL



GUI for METAMOC



http://metamoc.martintoft.dk

Status

- Started out with ARM9 support
  - Five stage pipeline, instruction cache, data caches, simple main memory
- Now
  - .. support for ARM7, ARM9 and ATMEL AVR 8-bit
  - .. with modest effort

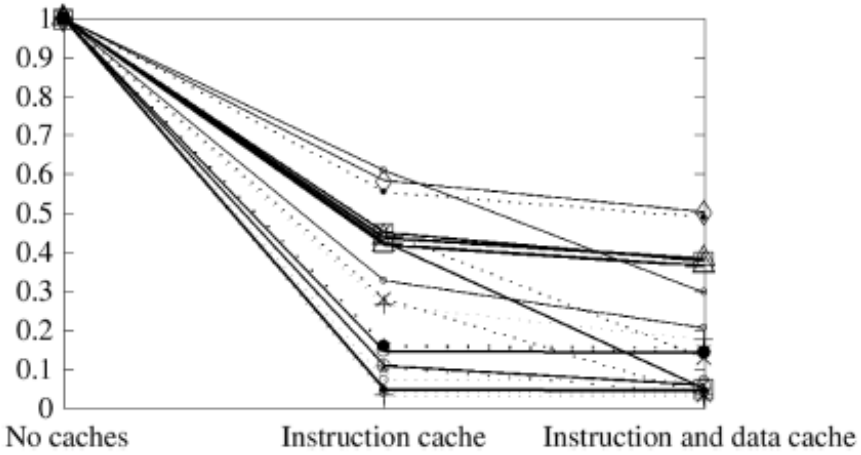
Experiments

- Evaluation using WCET benchmark programs from Mälardalen Real-Time Research Centre[2]
  - Applicability
  - Performance
- Discarded a number of programs
  - Floating point operations handled by software routines
  - Dynamic jumps
  - Some programs do not compile
- 21 programs for ARM and 19 programs for AVR
- Manually annotated loop bounds

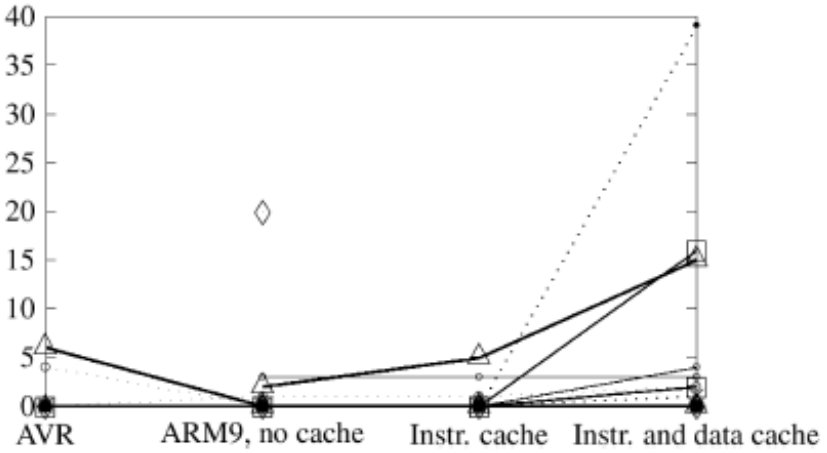
Experiments / Future

ARM9, 21 benchmarks	
Analysable without caches	21
Analysable with instruction cache	20
Unanalysable, state space explosion	1
Analysable with data and instruction cache	20
Unanalysable, state space explosion	1
Manual modification of e.g. data cache size	4

ATMEL AVR 8-bit, 19 benchmarks	
Analysable	16
Unanalysable, state space explosion	3



Relative improvement in WCET for ARM9.

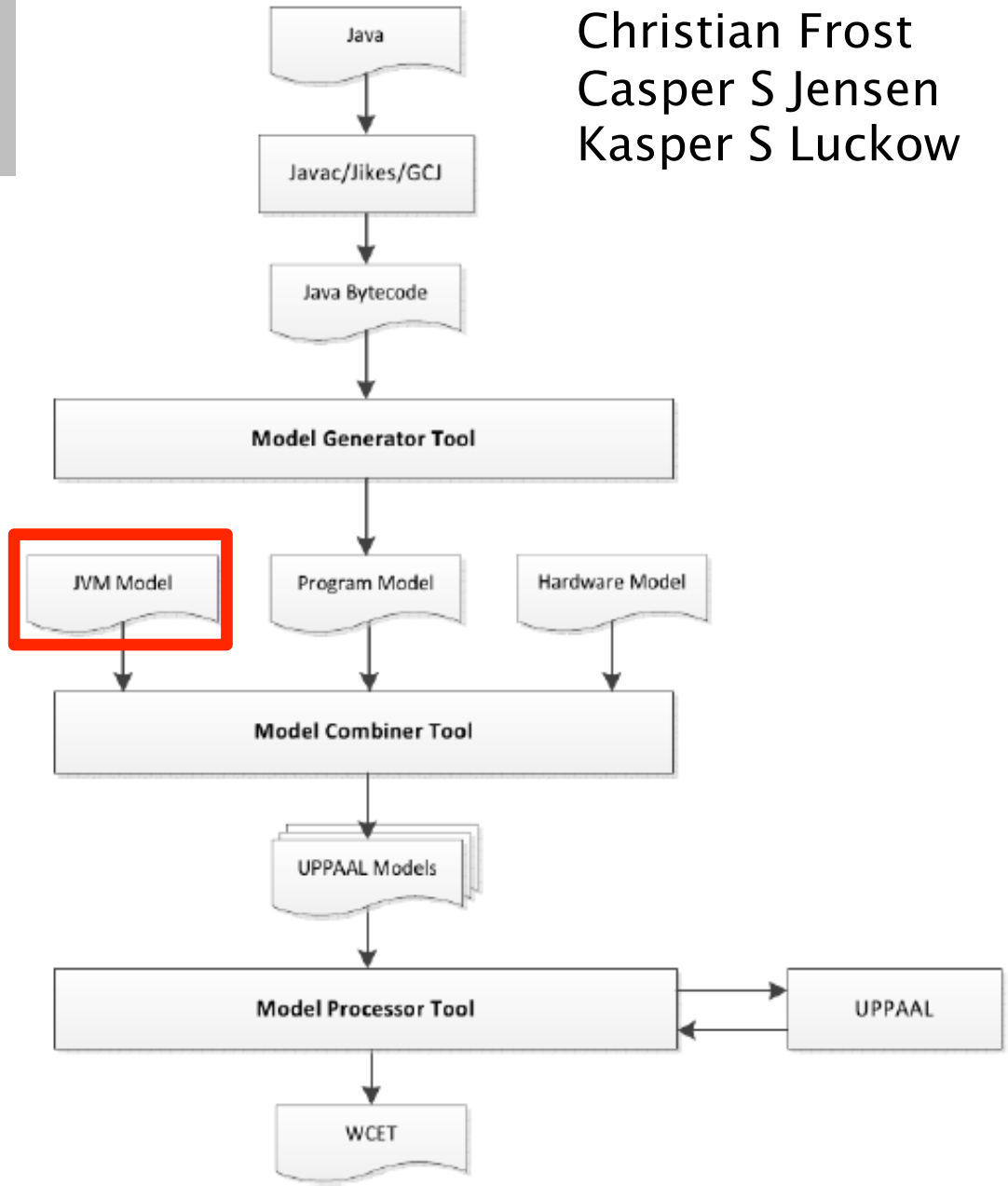


Analysis times in minutes for AVR and ARM9.

TetaJ

Christian Frost
Casper S Jensen
Kasper S Luckow

- Constructing UPPAAL models
- TetaJ Control Flow Graphs (TCFGs)
  - Forms the interface for model construction
  - Achieves reusability:
    - Java Bytecode to TCFG
    - AVR to TCFG
- CFG analyses are applied yielding annotated TCFGs
  - Loop detection
  - Condition optimisation
  - Progress measures in UPPAAL
- TCFGs are transformed to models
- The models are combined to one model
- UPPAAL conducts the WCET analysis on the final model



Evaluation of TetaJ

PERFORMANCE

Optimisation	Analysis time	States explored	Max memory usage
No optimisations	14h 51m 17s	41,854,143	3,905 MB
Only progress measures	4d 12h 7m 8s	408,223,029	589 MB
Only state space reduction	13h 33m 21s	41,854,143	2,426 MB
Only condition optimisation	1m 16s	53,732	294 MB
Only template reduction	4h 46m 41s	41,854,143	3,851 MB
All			

Algorithm	Measured WCET	TetaJ WCET	Pessimism
Iterative fibonacci	46,642 clock cycles	46,933 clock cycles	0.6%
Factorial	39,726 clock cycles	40,939 clock clycles	3.1%
Reverse ordering	64,436 clock cycles	81,919 clock cycles	27.1%
Bubble sort	907,103 clock cycles	2,270,401 clock cycles	150.3%
Binary Search	54,430 clock cycles	99,301 clock cycles	82.4%
Insertion Sort	849,353 clock cycles	3,740,769 clock cycles	440.4%

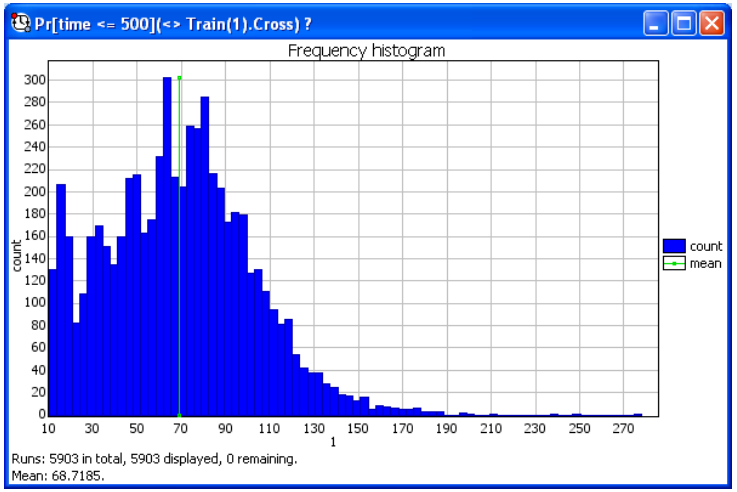
ACCURACY

Conclusion

- Schedulability Analysis
  - TIMES tool [TACAS 2002]
  - Multitasking applications under OSEK [RTS 2008]
  - CREOL Modular schedulability [FSEN09]
  - SARTS Java byte code on FPGA [JTRES08]
  - Schedulability Analysis Using UPPAAL 4.1 [Model-Based Design for ES, CRC Press 2010]
  - ARTS: MPSoC Schedulability [Model-Based Design for ES, CRC Press 2010]
- Worst Case Execution Time Analysis
  - METAMOC [WCET10,NSF11]
  - Combining AI & MC for Timing Analysis of MultiCore [Yi et al, RTSS10]
  - WCET analysis of Multicore using UPPAAL [Petterson et al]

sarts.boegholm.dk
metamoc.martintoft.dk
tetaj.dk

Performance Analysis
using
Statistical Model Checking

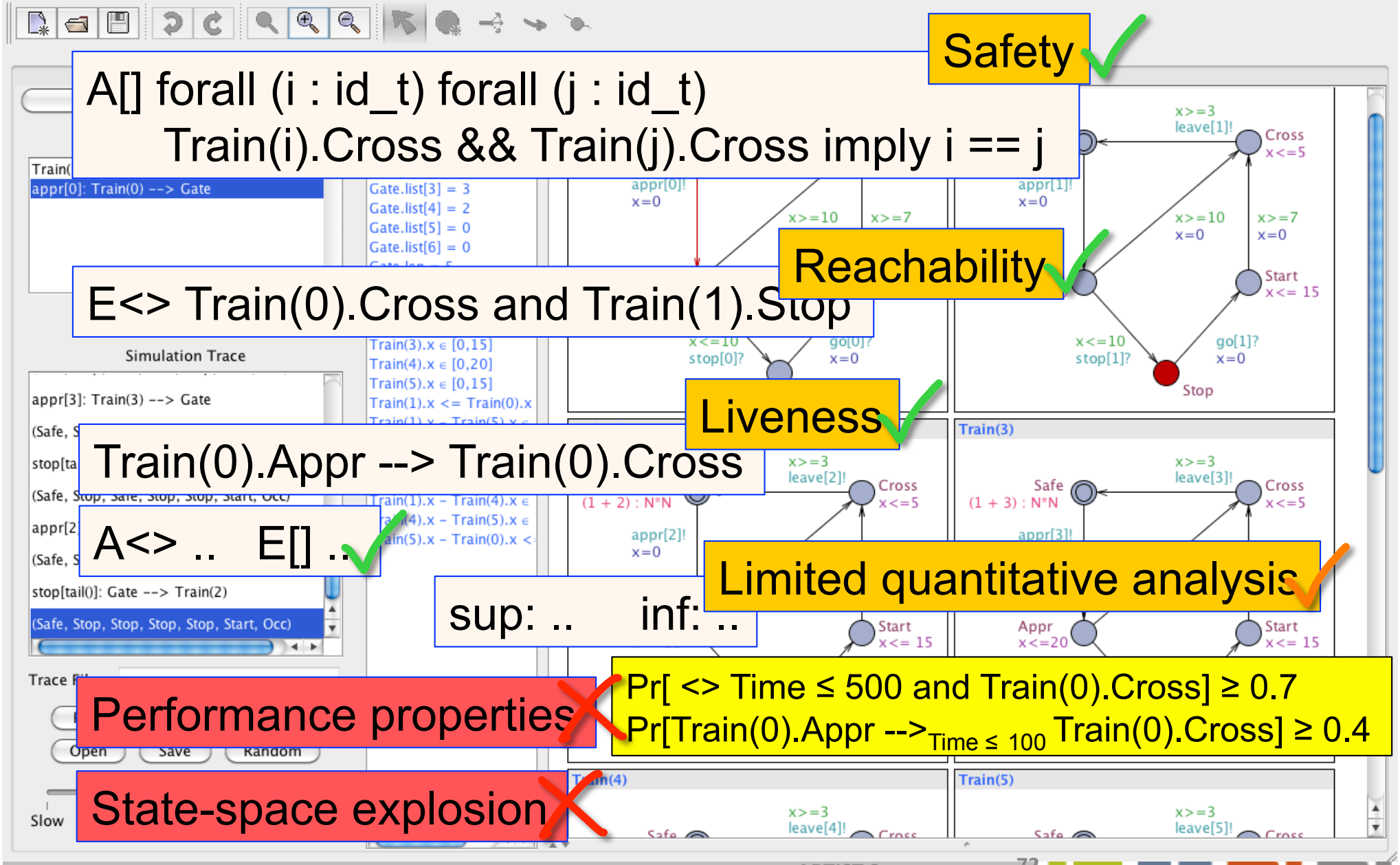


Collaborators:

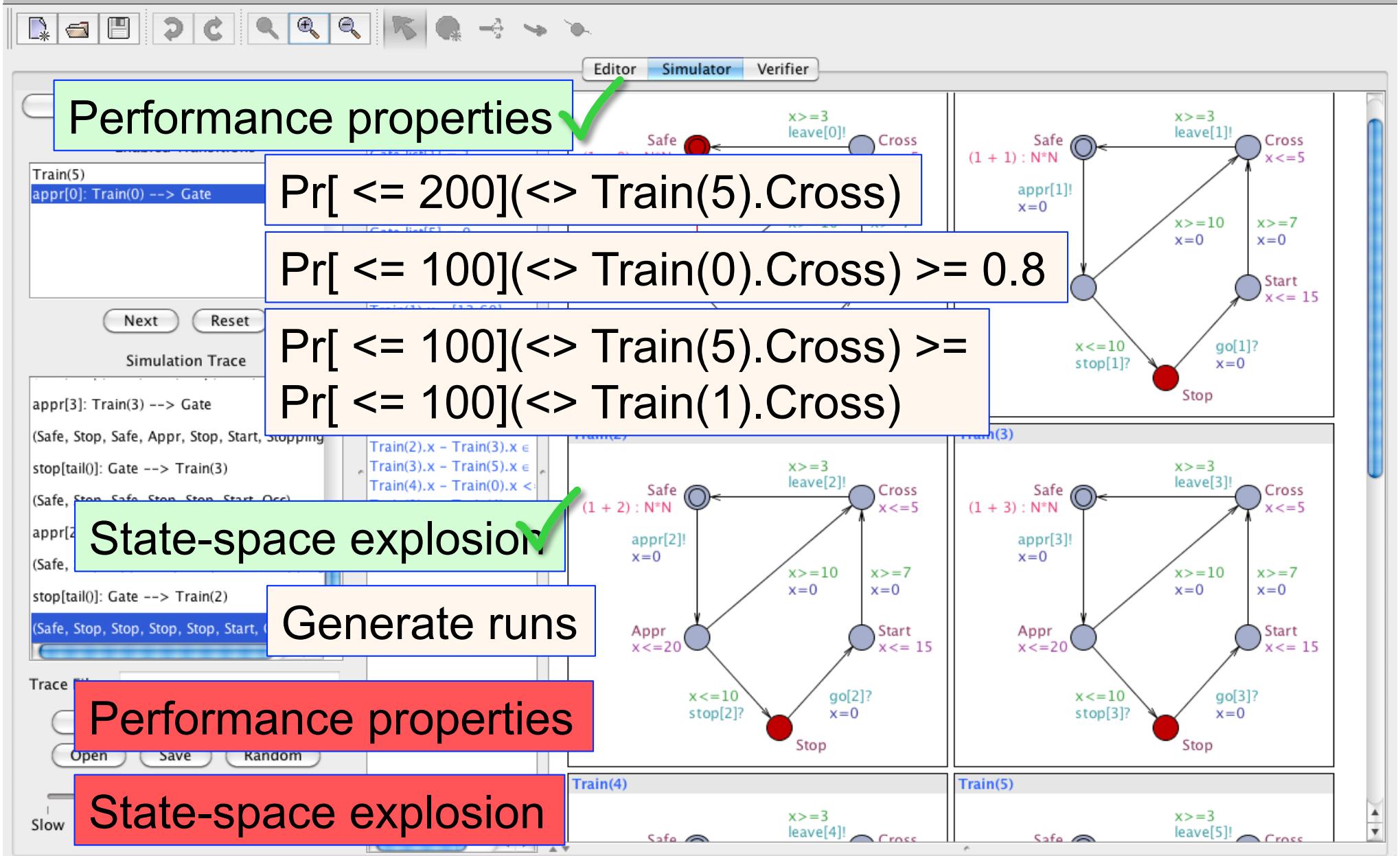
Peter Bulychev, Alexandre David
Axel Legay, Marius Mikucionis
Wang Zheng
Jonas van Vliet, Danny Poulsen

CAV 2011, PDMC 2011,
FORMATS 2011

UPPAAL


A[] forall (i : id_t) forall (j : id_t)
Train(i).Cross && Train(j).Cross imply i == j
Safety
E<> Train(0).Cross and Train(1).Stop
Reachability
Train(0).Appr --> Train(0).Cross
Liveness
A<> .. E[] ..
Limited quantitative analysis
sup: .. inf: ..
Performance properties
Pr[<> Time ≤ 500 and Train(0).Cross] ≥ 0.7
Pr[Train(0).Appr -->Time ≤ 100 Train(0).Cross] ≥ 0.4
State-space explosion
Simulation Trace
Open
Save
Random
Slow

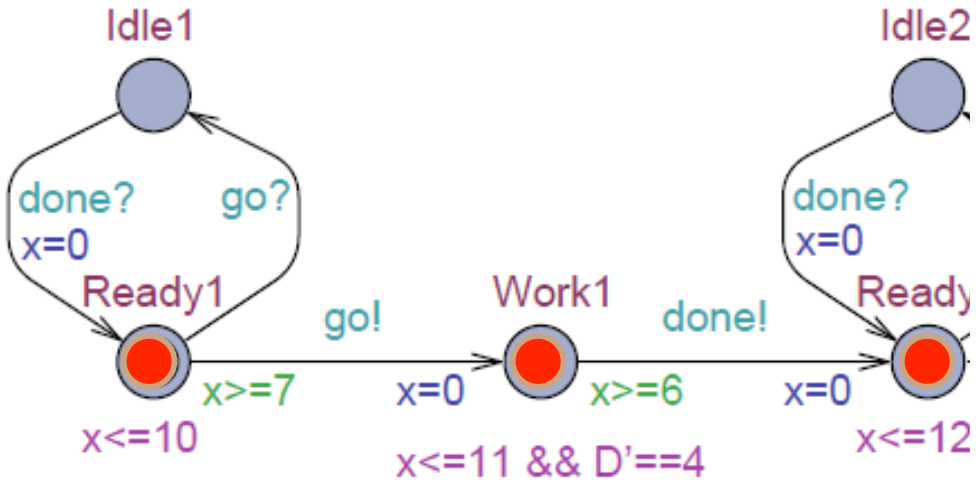
UPPAAL SMC

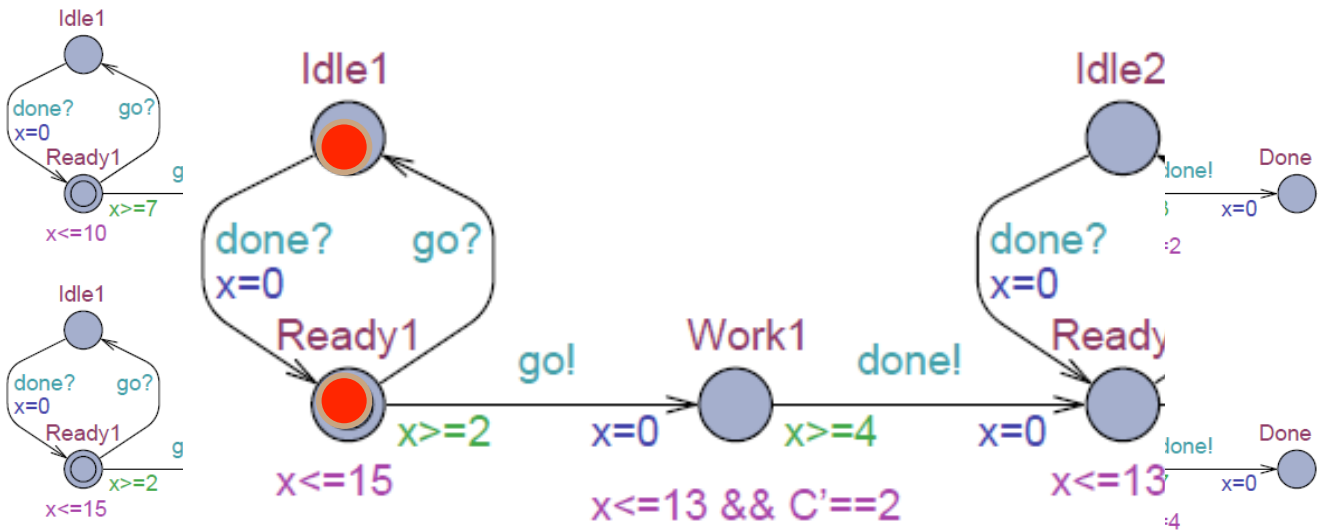

Editor
Simulator
Verifier
Performance properties
Pr[<= 200](<> Train(5).Cross)
Pr[<= 100](<> Train(0).Cross) >= 0.8
Pr[<= 100](<> Train(5).Cross) >=
Pr[<= 100](<> Train(1).Cross)
State-space explosion
Generate runs
Performance properties
State-space explosion
Train(5)
appr[0]: Train(0) --> Gate
Next
Reset
Simulation Trace
appr[3]: Train(3) --> Gate
stop[tail()]: Gate --> Train(3)
stop[tail()]: Gate --> Train(2)
Open
Save
Random
Slow
Safe
Cross
Appr
Start
Stop
x>=3
leave[0]!
leave[1]!
leave[2]!
leave[3]!
leave[4]!
leave[5]!
x<=5
x>=10
x=0
x>=7
x<= 15
x<=10
go[1]?
go[2]?
go[3]?
Train(4)

The Hammer Game

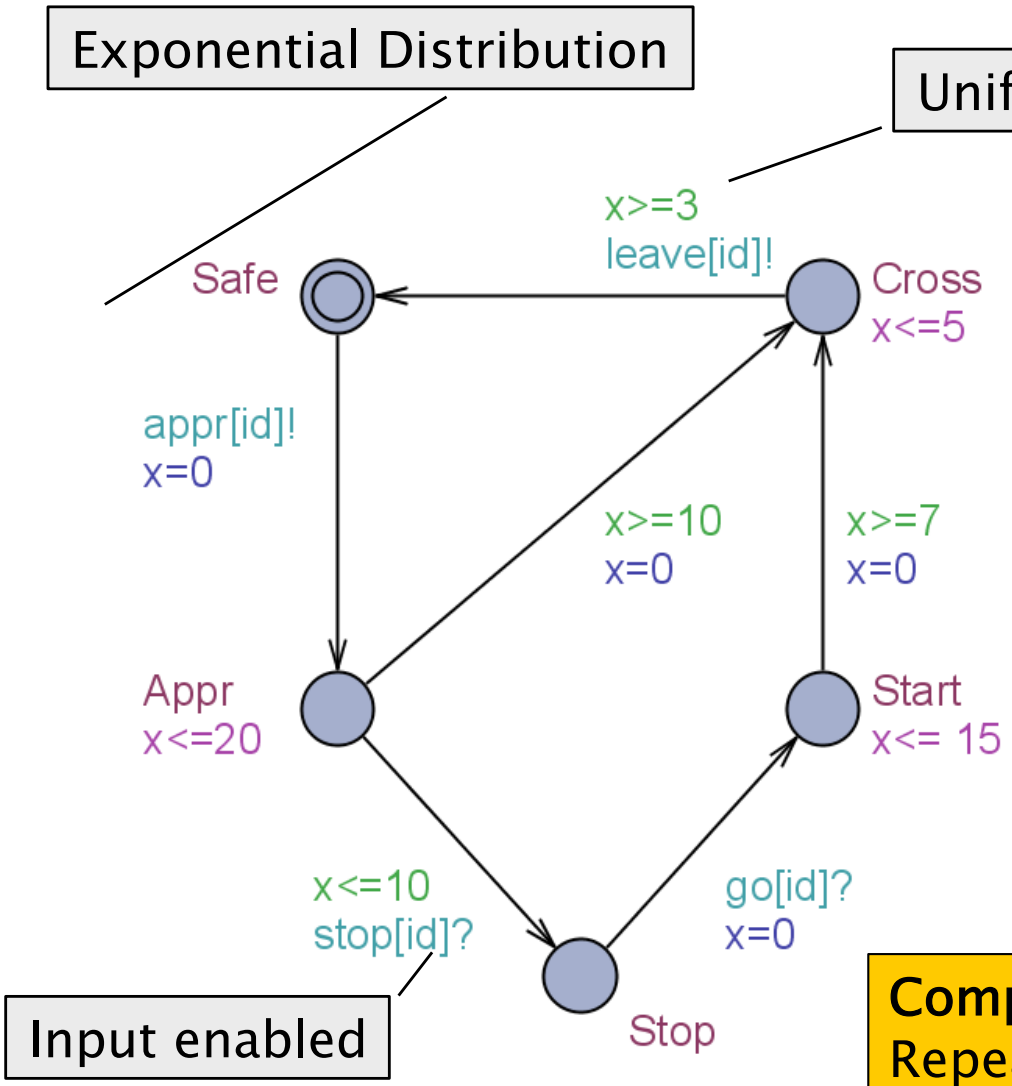
Jan

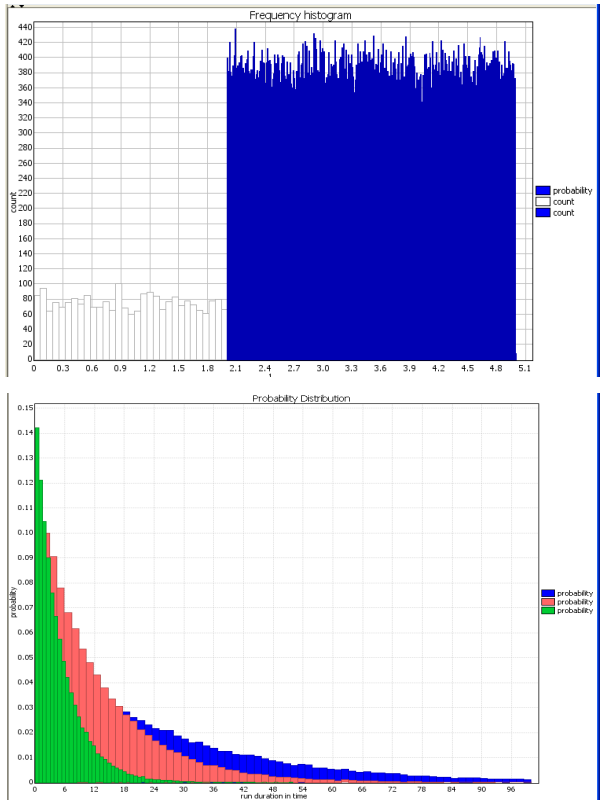
Wang





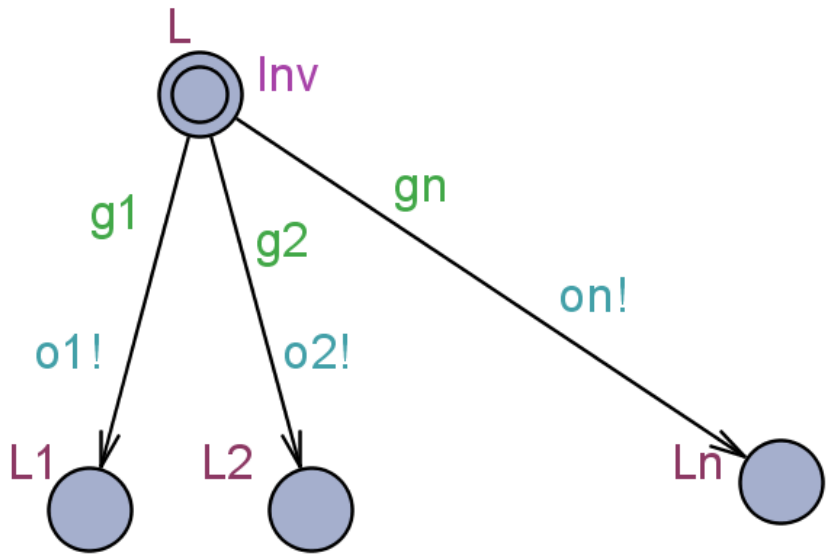
Stochastic Semantics of TA

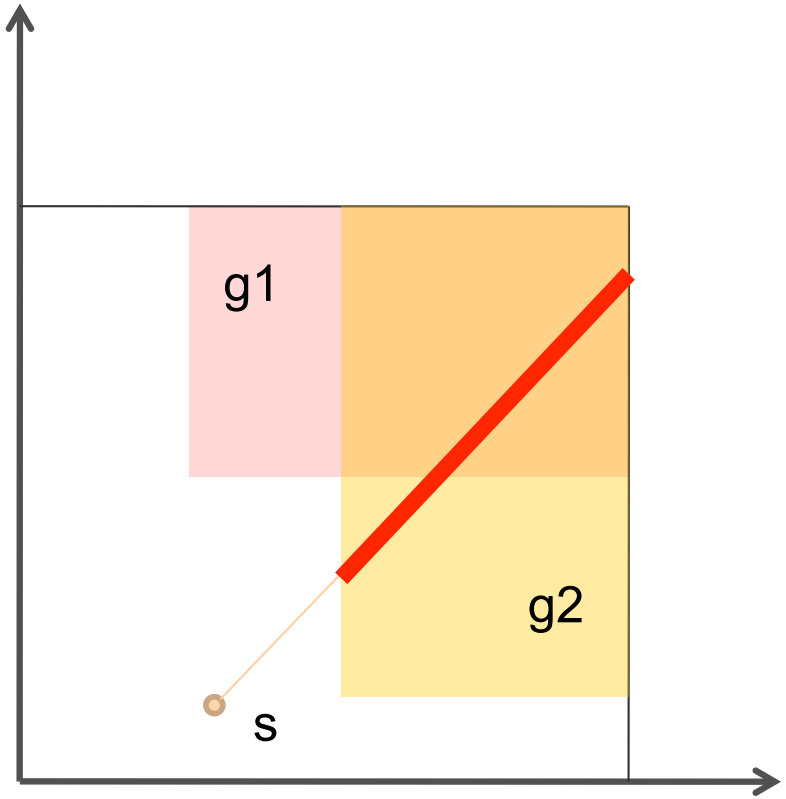




Composition =
Repeated races between components

Stochastic Semantics of Timed Automata





Delay Density Function

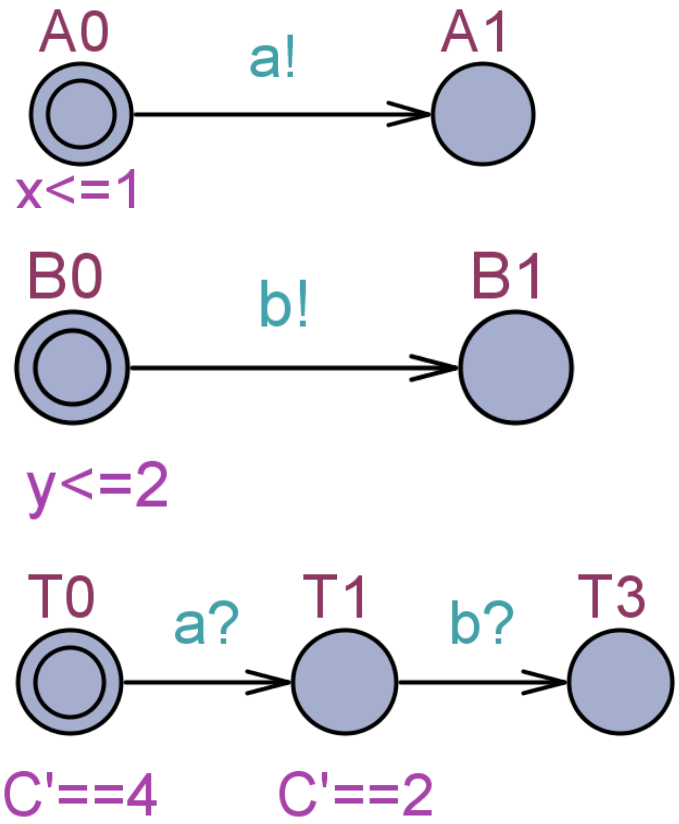
^1_s: **R! R**

Output Probability Function

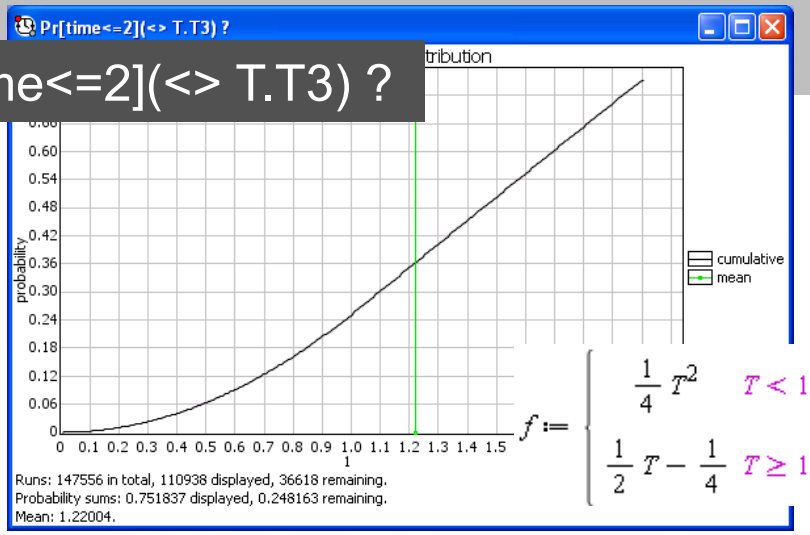
$^\circ_s$: §$_o$! [0,1]

- ^1_s uniform on [d_{min}, d_{max}]
- $^\circ_s$ uniform over enabled outputs

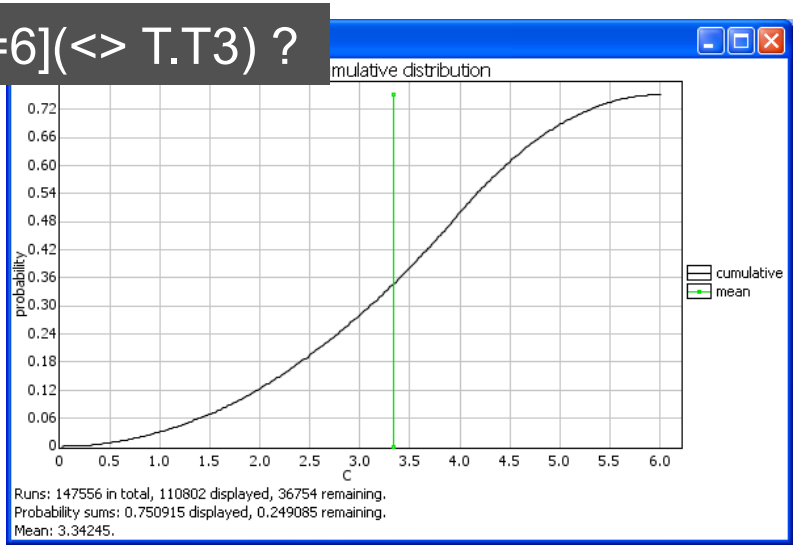
Stochastic Semantics of Timed Automata



Pr[time<=2](<> T.T3) ?

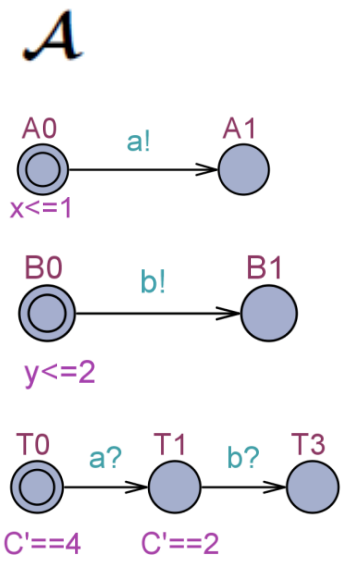


Pr[C<=6](<> T.T3) ?



Composition = Race between components for outputting

Stochastic Semantics of Timed Automata



Assumptions:
Component TAs are:
- Input enabled
- Deterministic
- Disjoint set of output actions

¼ (**s** , $a_1\ a_2\ \ldots.\ a_n$) :
the set of maximal runs from **s** with a prefix
$t_1\ a_1\ t_2\ a_2\ \ldots\ t_n\ a_k$
for some $t_1,\ldots,t_n$ 2 **R**.

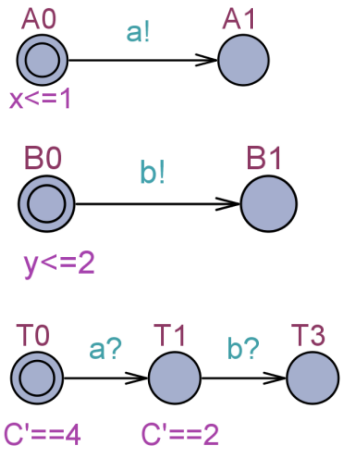
$$\mathbb{P}_{\mathcal{A}}\big(\pi(\mathbf{s},\, a_1 a_2 \ldots a_n)\big) =$$
$$\int_{t\geq 0} \mu_{s_c}(t) \cdot \Big(\prod_{j\neq c} \int_{\tau>t} \mu_{s_j}(\tau)\, d\tau\Big) \cdot \gamma_{s_c{}^t}(a_1) \cdot \mathbb{P}_{\mathcal{A}}\big(\pi(\mathbf{s}^t)^{a_1},\, a_2 \ldots a_n\big)\ dt$$

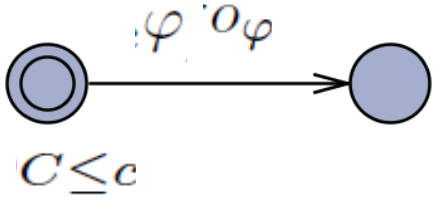
where $c = c(a_1)$, and as base case we take $P_{\mathcal{A}}(\pi(\mathbf{s}), \varepsilon) = 1$.

Logical Properties

$$\psi \ ::= \ \mathbb{P}\big(\Diamond_{C\leq c}\varphi\big) \sim p \ \mid \ \mathbb{P}\big(\Box_{C\leq c}\varphi\big) \sim p$$

$\mathcal{A}^*$





$$(A|B|T) \models \mathbb{P}\big(\Diamond_{C\leq 6}T_3\big) = 0.75$$

$$(A|B|T) \models \mathbb{P}\big(\Diamond_{t\leq 2}T_3\big) = 0.75$$

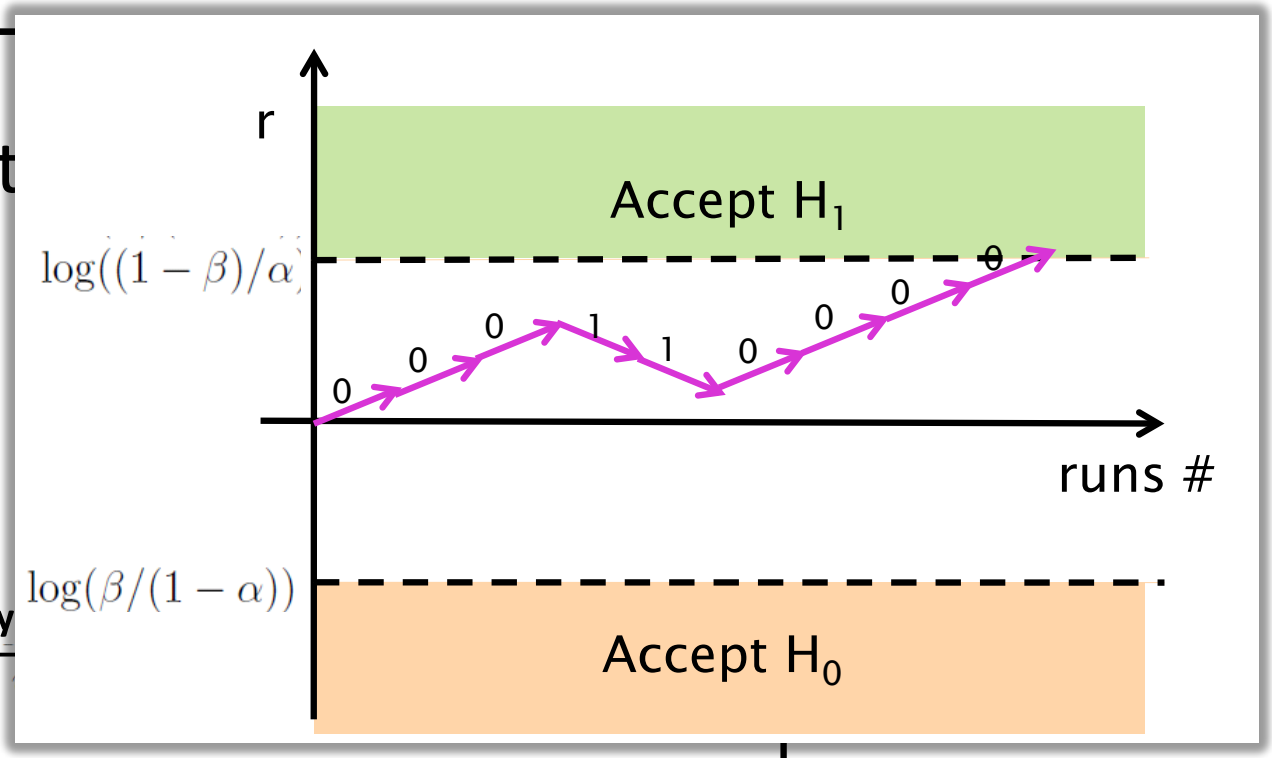
$$\mathcal{A} \models \mathbb{P}\big(\Diamond_{C\leq c}\varphi\big) \sim p \ \text{ iff } \ \mathbb{P}_{\mathcal{A}^*}\Big(\bigcup_{\sigma\in\Sigma^*} \pi(s_0, \sigma o_\varphi)\Big) \sim p$$

SMC Algorithms in UPPAAL

Qualitative (Hypot

$$p = \mathbb{P}_{\mathcal{A}}(\Diamond_{C \leq c} \varphi)$$

® prob of acc H_0 when H_1



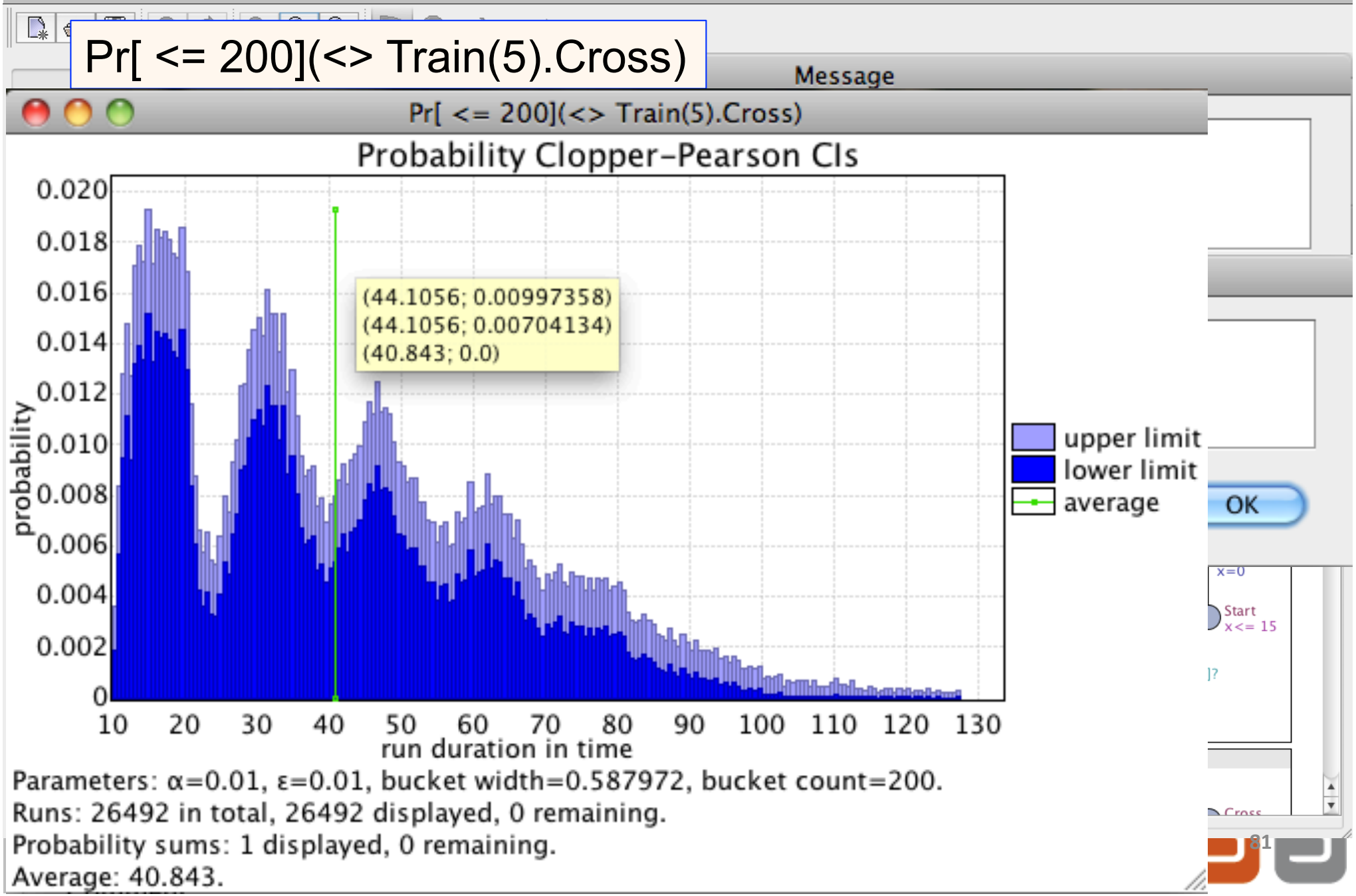
Algorithm II: Sequential Probability

function hypothesis(S:model ,

```
1 r:=0
2 while true do
3     Observe the random variable x corresponding to ◇_{C≤c}φ for a run.
4     r := r + x * log(p1/p0) + (1 − x) * log((1 − p1)/(1 − p0))
5     if r ≤ log(β/(1 − α)) then accept H0
6     if r ≥ log((1 − β)/α) then accept H1
```

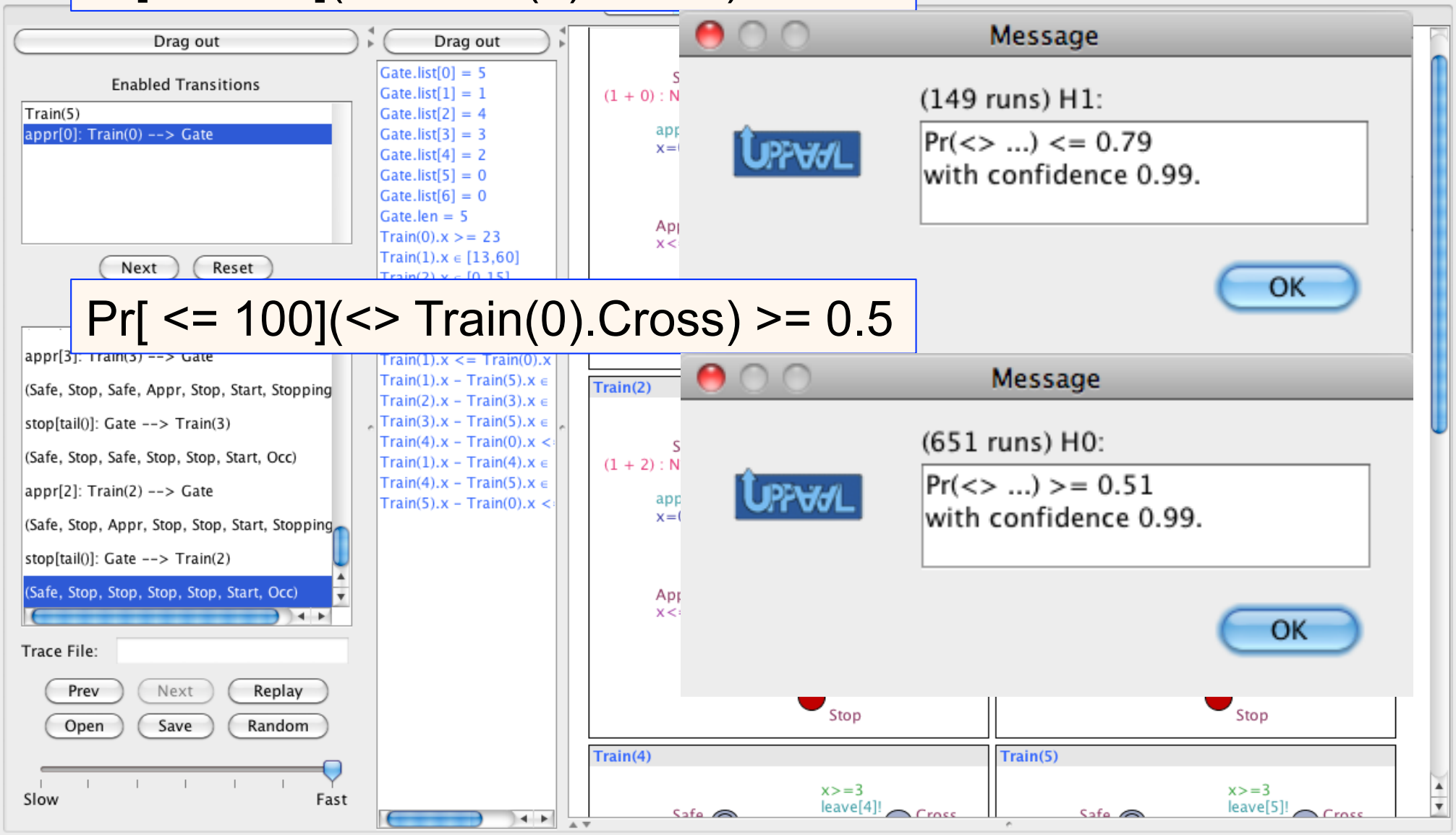
11

Queries in UPPAAL SMC



Queries in UPPAAL SMC

Pr[<= 100](<> Train(0).Cross) >= 0.8



Pr[<= 100](<> Train(0).Cross) >= 0.5

Queries in UPPAAL SMC

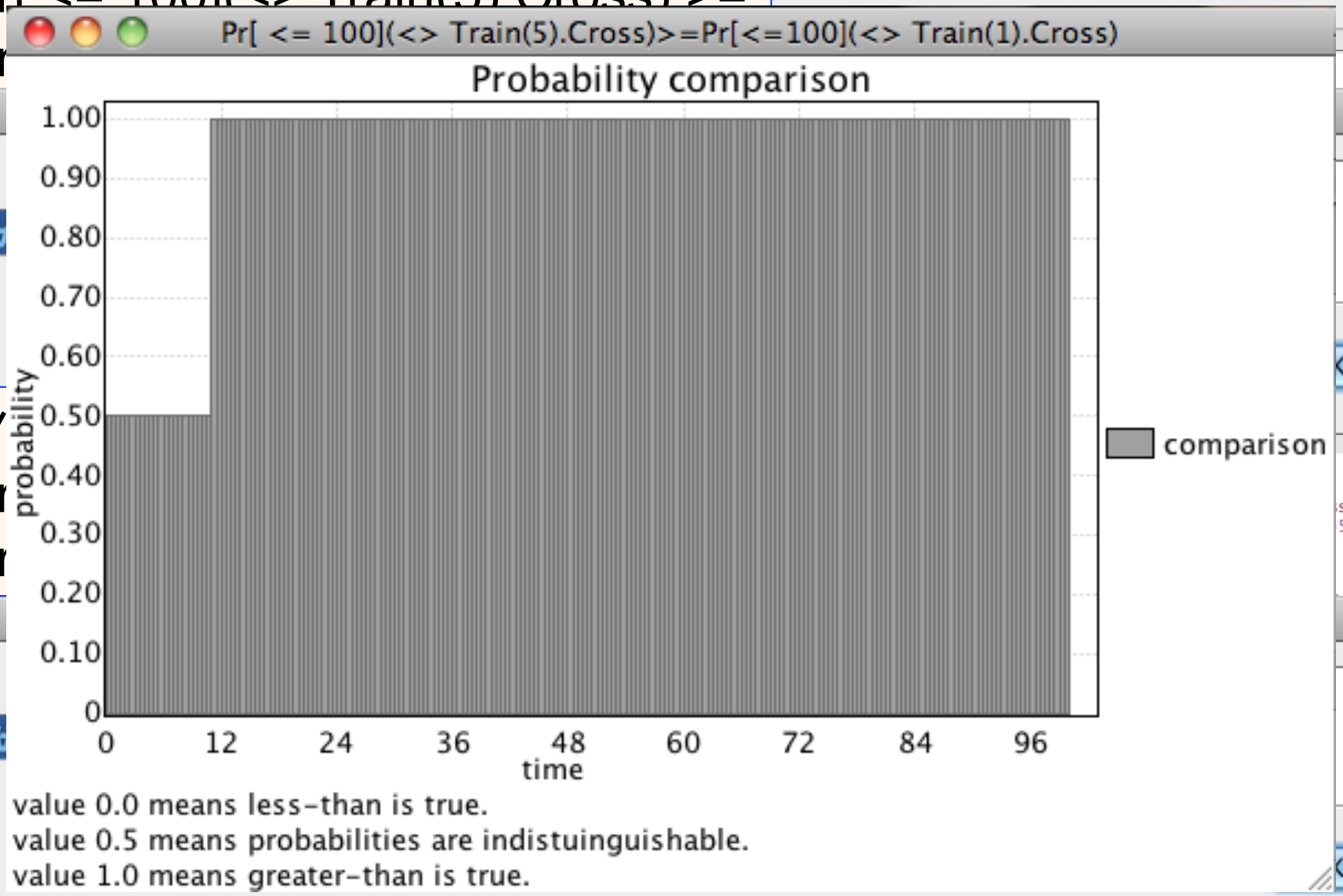
Pr[<= 100](<> Train(5).Cross) >=

Pr

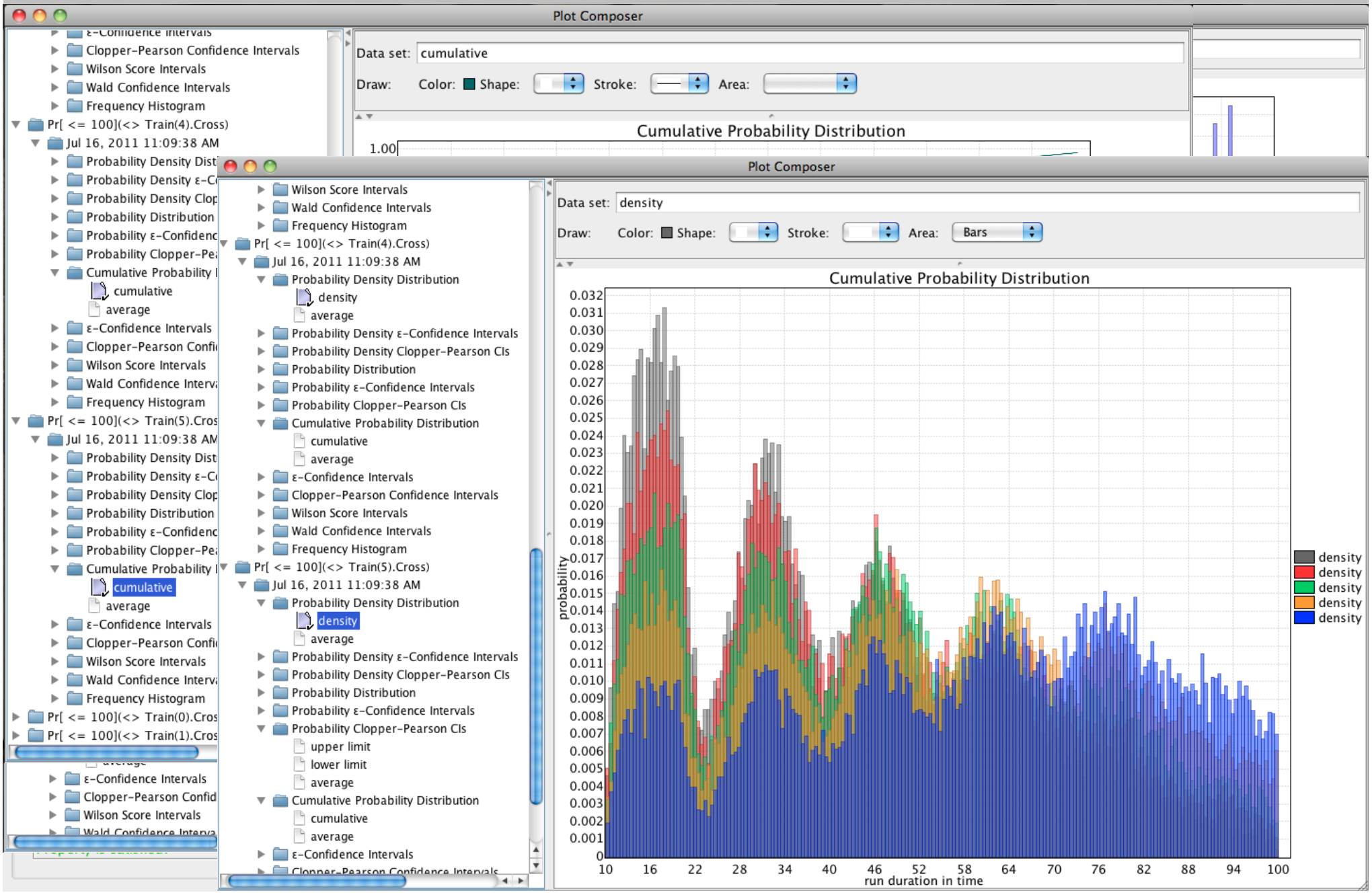
∀

Pr

Pr



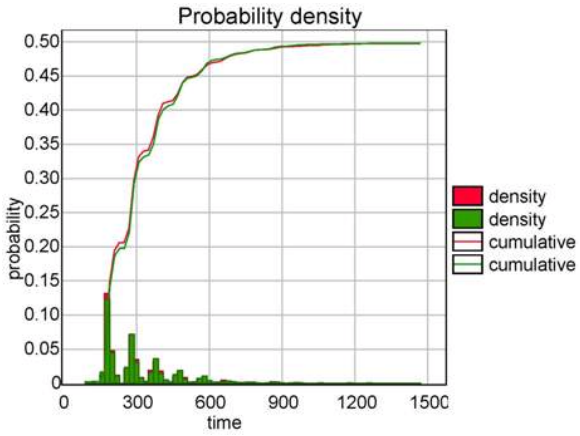
Analysis Tool: Plot Composer



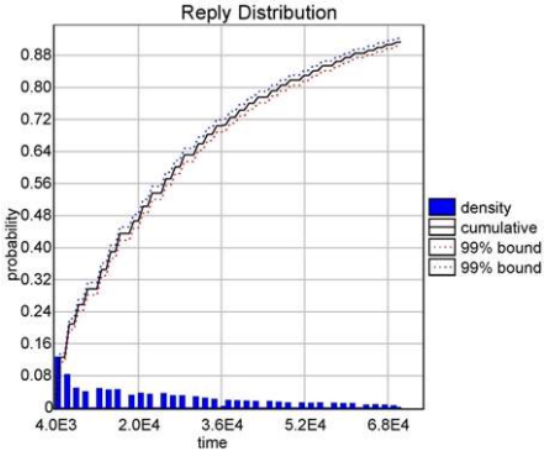
SMC in UPPAAL

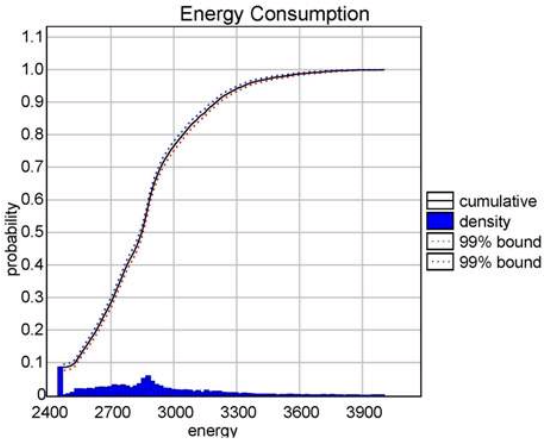
- Constant Slope Timed Automata
  - Clocks may have different (integer) slope in different locations.
  - Branching edges with discrete probabilities (weights).
  - Beyond Priced TA, Energy TA. Equal LHA in (non-stochastic) expressive power.
  - Beyond DTMC, beyond CTMC (with multiple rewards)
- All features of UPPAAL supported
  - User defined functions and types
  - Expressions in guards, invariants, clock-rates, delay-rates (rationals), and weights.
- New GUI for plot-composing and exporting.

Case Studies

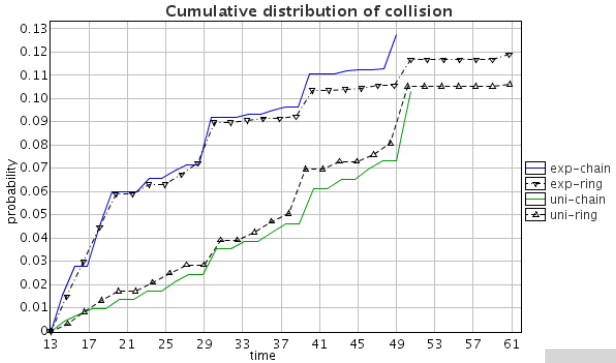


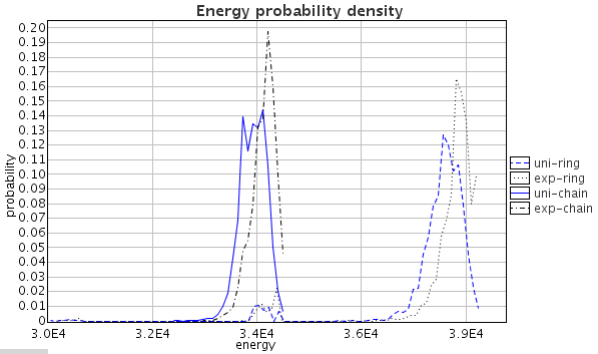
FIREWIRE





BLUETOOTH





LMAC

DPA

Benchmarking
Duration Probabilistic Automata

	Param.			Estim.				Hyp. Testing			
	n	k	m	PRISM	Up_p	Up_d	Up_c	PRISM	Up_p	Up_d	Up_c
n	4	4	3	2.7	0.3	0.2	0.2	2.0	0.1	0.1	0.1
10	6	6	3	7.7	0.6	0.5	0.4	3.9	0.2	0.2	0.3
10	8	8	3	26.5	1.2	0.9	0.7	16.4	0.5	0.4	0.3
10	20	40	20	>300				>300	35.5	26.2	20.7
10	30	40	20	>300				>300	61.2	41.8	33.2
20	40	40	20	>300				>300	92.2	56.9	59.5
20	40	20	20	>300				>300	41.1	31.2	26.5
20	40	30	20	>300				>300	68.8	46.7	46.1
	40	55	40	>300				>300			219.5

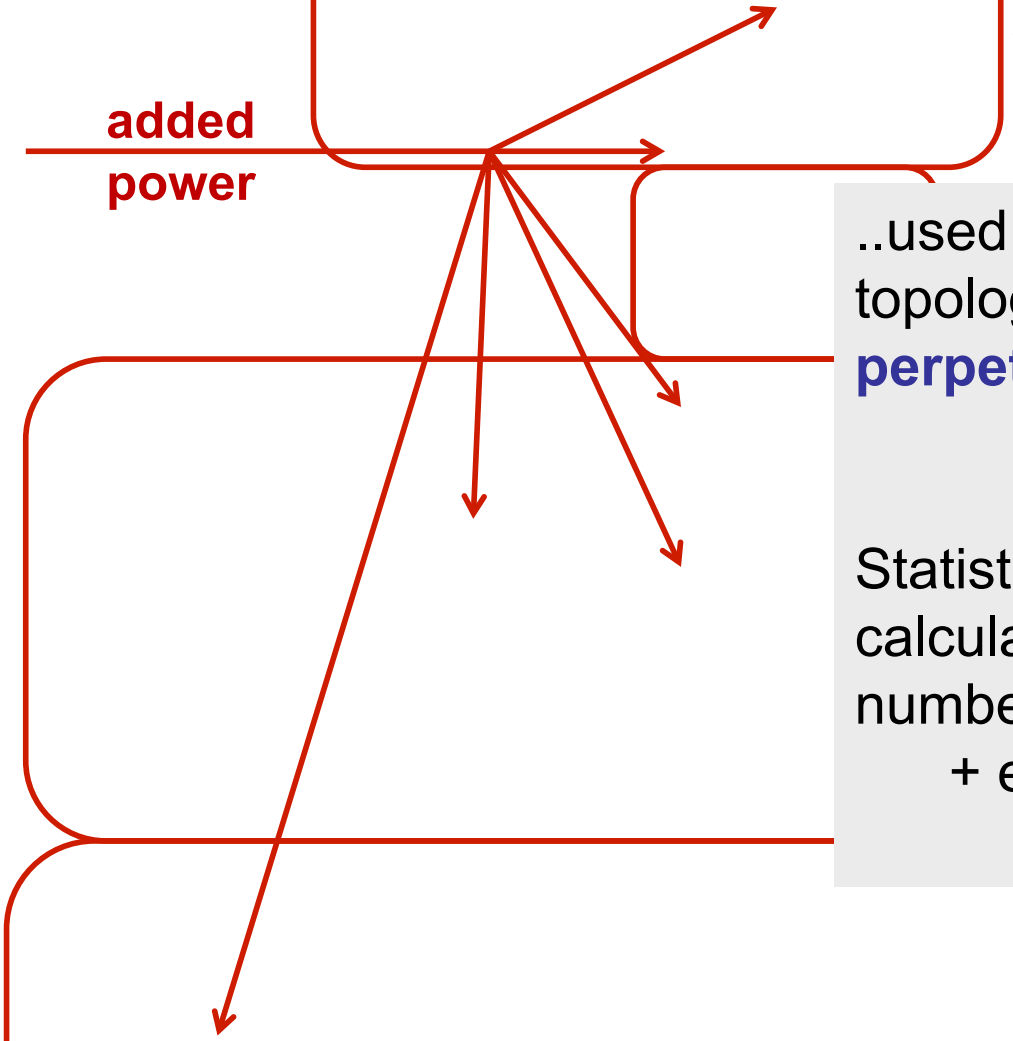
Fig
res
r_1

Lightweight Media Access Control

- Problem domain:
  - communication scheduling
- Targeted for:
  - self-configuring networks,
  - collision avoidance,
  - low power consumption
- Application domain:
  - wireless sensor networks

- Initialization (listen until a neighbor is heard)
- Waiting (delay a random amount of time frames)
- Discovery (wait for entire frame and note used slots)
- Active
  - choose free slot,
  - use it to transmit, including info about detected collisions
  - listen on other slots
  - fallback to Discovery if collision is detected
- Only neighbors can detect collision and tell the user-node that its slot is used by others

adopted from A.Fehnker, L.v.Hoesel, A.Mader



initialization

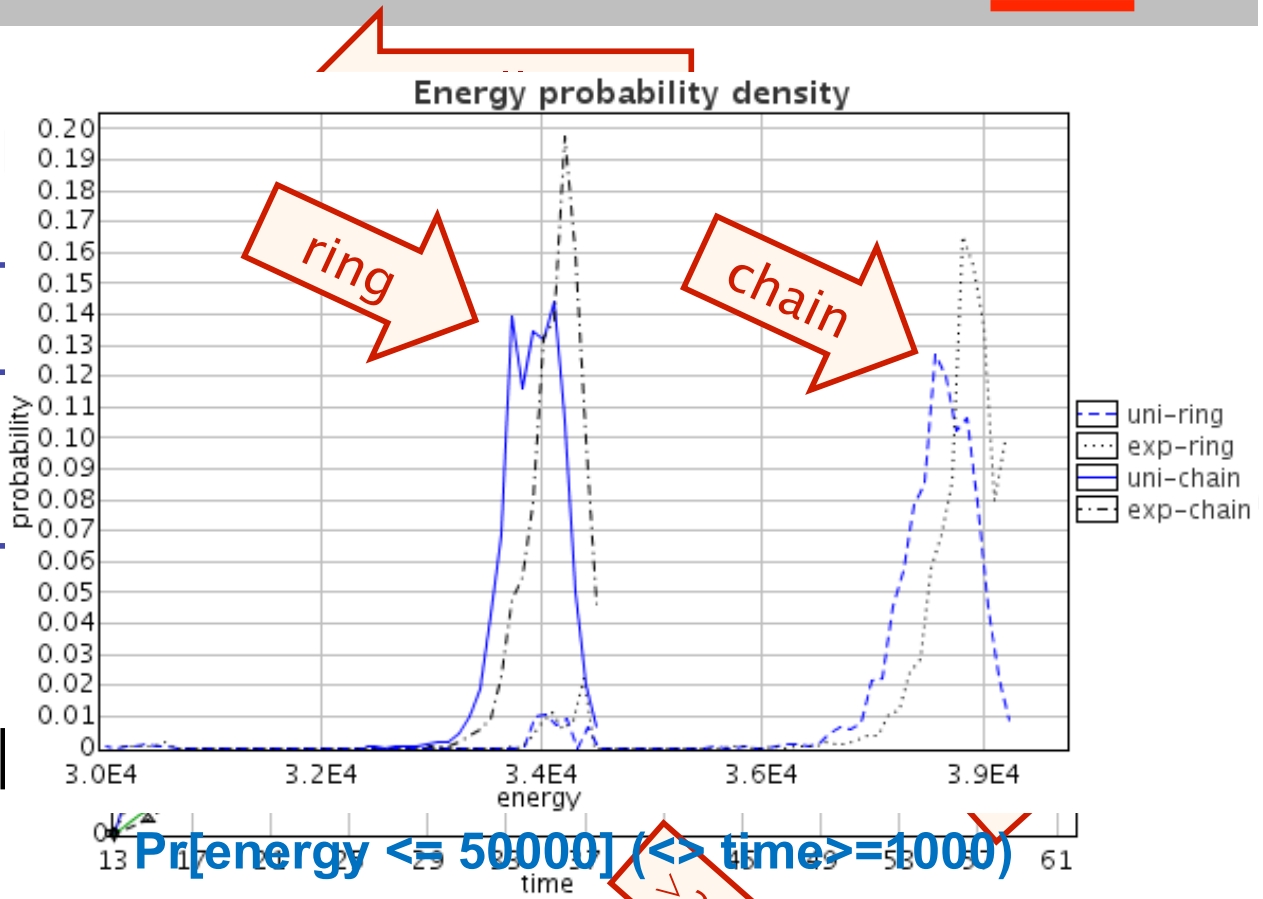
..used UPPAAL to explore 4- and 5-node topologies and found cases with **perpetual** collisions
(8.000 MC problems)

Statistical MC offers an insight by calculating the probability over the number of collisions.
+ estimated cost in terms of energy.

active usage

SMC of LMAC with 4 Nodes

- Wait distributio
  - geometric
  - uniform
- Network topolo
  - chain
  - ring
- Collision probal
- Collision count
- Power consumption

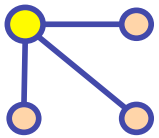
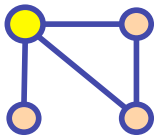
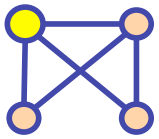
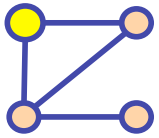


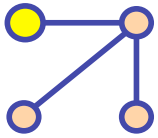
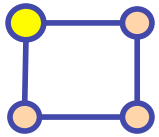
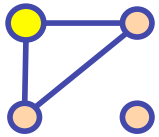
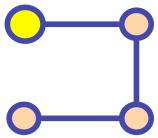
Pr[collisions<=50000] (<> time>=1000)

LMAC with Parameterized Topology
Distributed SMC

Collision probability in a 4 node network: sweep over all topologies.
32 core cluster: - 8xIntel Core2 2.66GHz CPU

Pr[time<=200] (<> col_count>0)

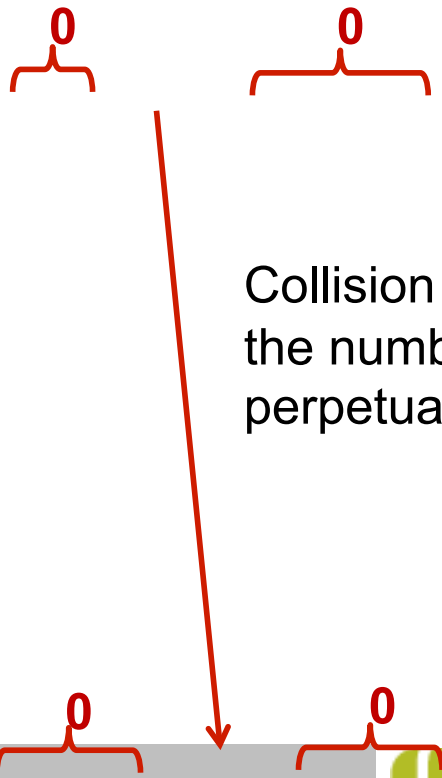
	topology	collision probability
(star)		[0.36; 0.39]
		[0.29; 0.36]
		[0.26; 0.30]
		[0.19; 0.21]

	topology	collision probability
		[0.08; 0.19]
(ring)		[0.11; 0.13]
		[0.08; 0.15]
(chain)		[0.049; 0.050]

10-Node Ring

The first collision:
happens before 1000tu

Collision counts after 1000tu



Collision counts after 2000tu:
the numbers are doubled –
perpetual collisions

- The first collisions can be as late as **920**tu.
- It is very likely (**>90%**) that
  there will be **0** collisions.
- But if they happen, they are perpetual.

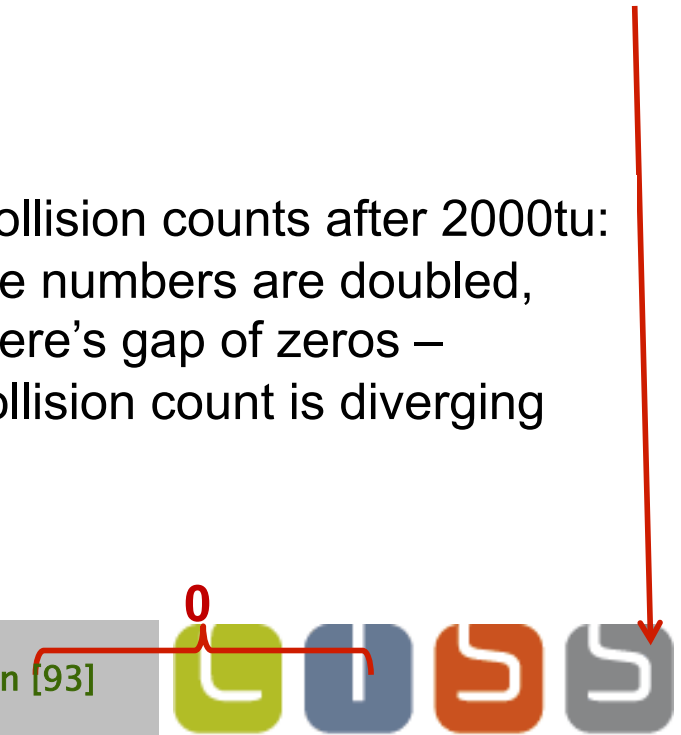
10-Node Chain

The first collision:
happens before 800tu

Collision counts after 1000tu

- The first collisions can be as late as **800**tu.
- It is very likely (**>94%**) that
  there will be **0** collisions.
- But if they happen, some are perpetual.

Collision counts after 2000tu:
the numbers are doubled,
there's gap of zeros –
collision count is diverging



10-Node Star

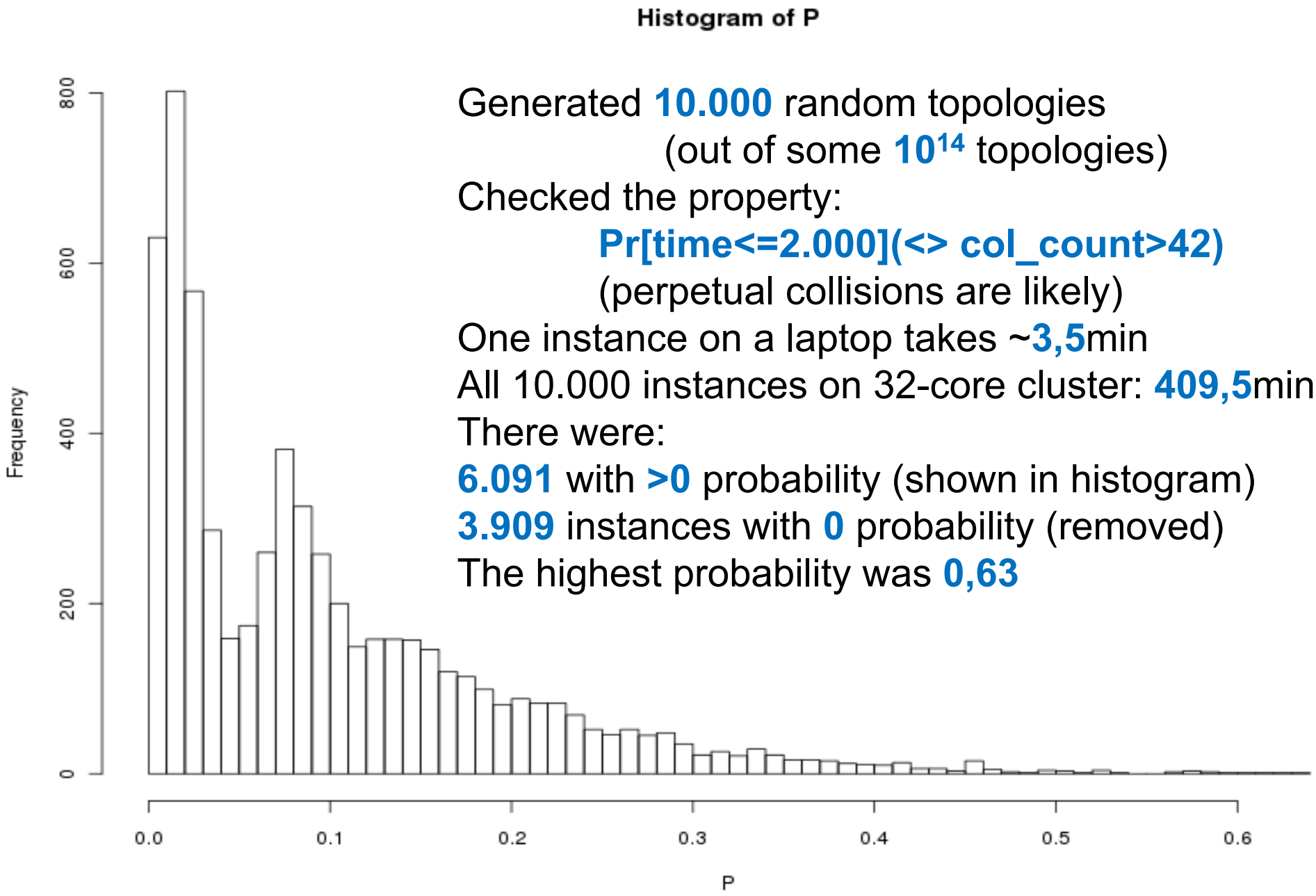
The first collision:
happens before 500tu

Collision counts after 1000tu

Collision counts after 2000tu:
the numbers are doubled –
perpetual collisions

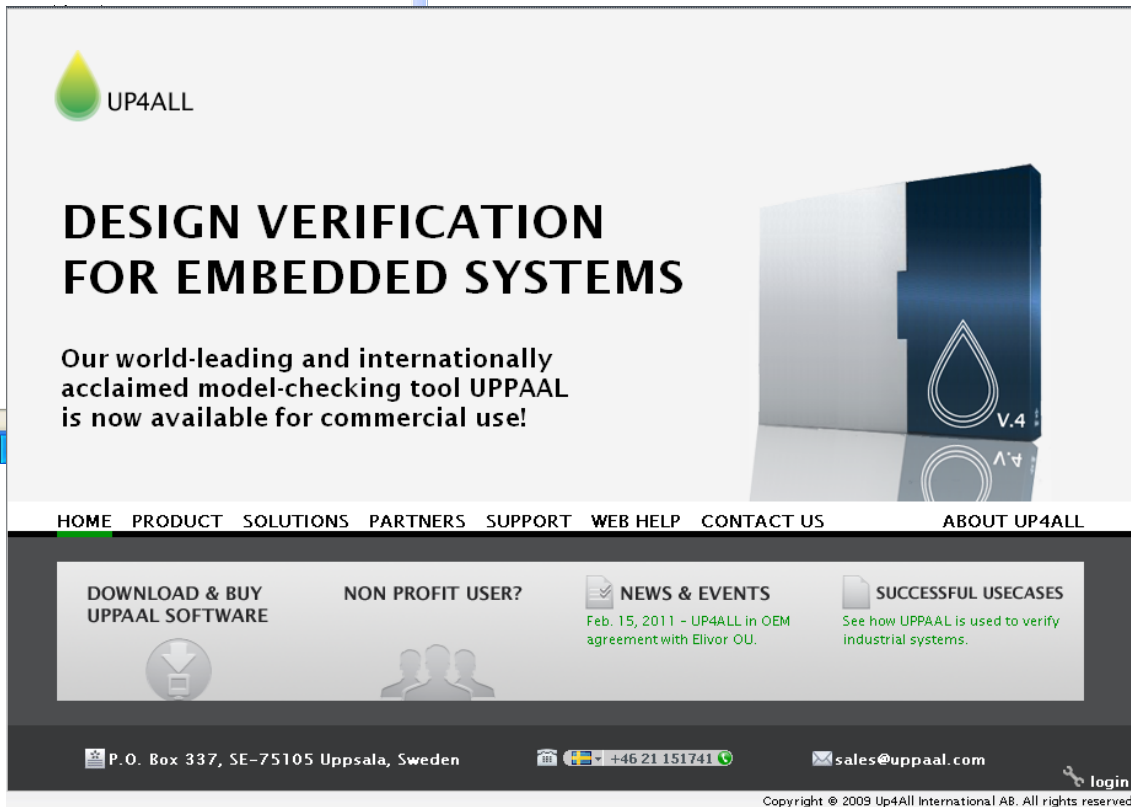
- The first collisions happen before **500**tu.
- It is unlikely (**8.2%**) that
  there will be **0** collisions.
- And if they happen, they are perpetual.

10-Node Random Topologies
Distributed SMC



Generated **10.000** random topologies
(out of some 10^{14} topologies)
Checked the property:
Pr[time<=2.000](<> col_count>42)
(perpetual collisions are likely)
One instance on a laptop takes ~**3,5**min
All 10.000 instances on 32-core cluster: **409,5**min
There were:
6.091 with **>0** probability (shown in histogram)
3.909 instances with **0** probability (removed)
The highest probability was **0,63**

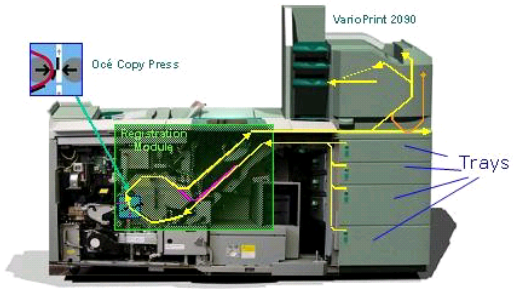
www.uppaal.{org,com}



Quasimodo

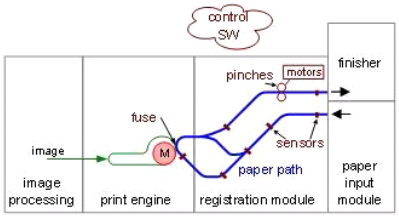
WP5 Additional Case Studies

- Self-Balancing Scooter (CHESS)
  - Highlevel control-modes modeled by engineers (UPPAAL)
  - Schedulability (UPPAAL)



- Adaptive scheduling of data paths (OCE)
  - Synthesis of optimal data path (CORA)

- Rapid Input-Output Packet Switch (ASML)
  - Simulation and verificatoin ofworst-case latencies (POOSL, UPPAAL)



Quasimodo Industrial Handbook
To be published by Springer

Thank You !

Thanks for your attention!

Please do not hesitate to contact me:

kgl@cs.aau.dk