UPPAAL Tutorial

Beyond UPPAAL

Alexandre David

Paul Pettersson

RTSS'05

UPPAAL Family

- "Classic": real-time verification
- Cora: real-time scheduling
- Tron: online real-time testing
- Tiga: timed game
- Times: schedulability analysis
- CoVer: test case generation

UPPAAL "Classic"

- Real-time verification
  - Presented today

http://www.uppaal.com

UPPAAL Cora

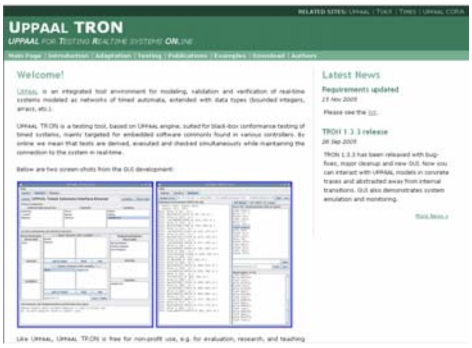
- **Real Time Scheduling**
  - Optimality
  - Reachability
  - Safety

Cost Optimal
Reachability Analysis

http://www.cs.aau.dk/~behrmann/cora/

UPPAAL Tron

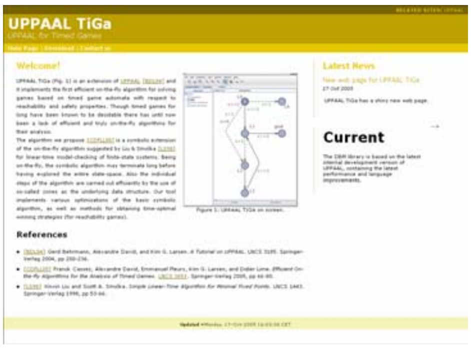
- **Real Time Testing**
  - Off-line Test Generation
  - On-line Test Generation and Execution



http://www.cs.aau.dk/~marius/tron/

UPPAAL Tiga

- Timed Games
  - Optimal winning strategies
  - Controller synthesis



http://www.cs.aau.dk/~adavid/tiga/

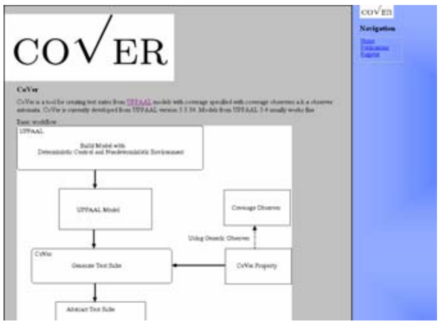
Times

- Schedulability Analysis
  - Schedule synthesis
  - Code synthesis

http://www.timestool.com

UPPAAL CoVer

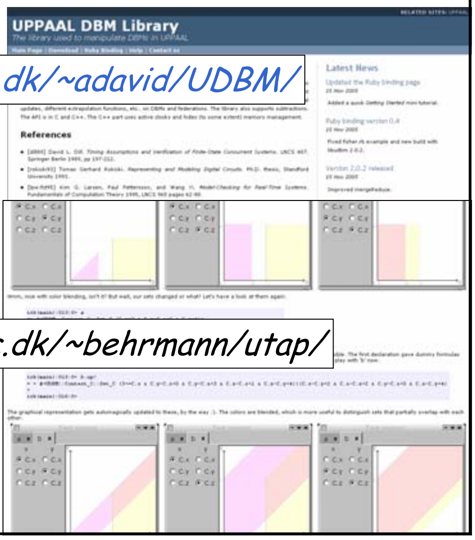
- Conformance Testing
  - Test suite generation
  - Coverage observer



http://www.hessel.nu/CoVer/

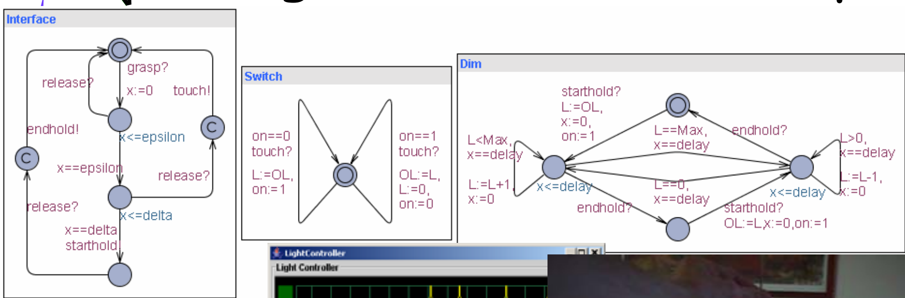
Open Source Initiatives

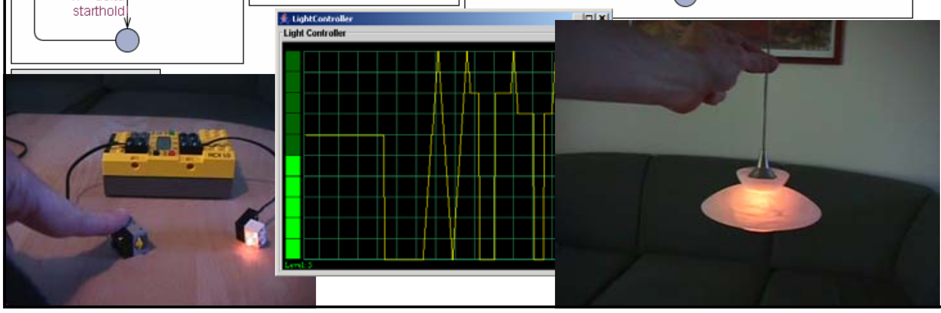
- DBM Library (GPL)
  - Eff… DB…
  - *http://www.cs.aau.dk/~adavid/UDBM/*
  - Subtractions & reduction techniques
  - Ruby binding (with graphical viewer)
  - Used in UPPAAL
- UTAP (UPPAAL TA parser library, LGPL)
  - Pars… repr…
  - *http://www.cs.auc.dk/~behrmann/utap/*
  - Support for full syntax of UPPAAL TA (xta + xml)
- Soon GUI XML components



UPPAAL Tron Light Controller Example

TRON

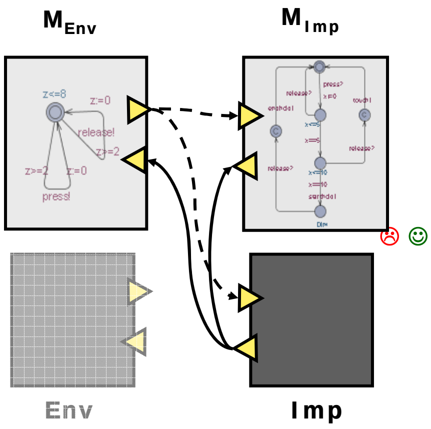




UPPAAL Tron Online Testing

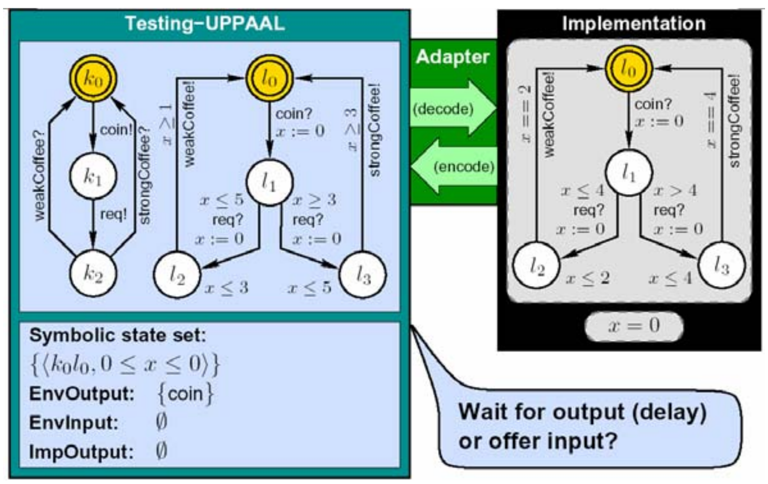
TRON

- Released on May 16, 2004 [Fates'04]
- Black-box conformance testing of real-time systems.
- Online generation and execution of timed test traces from given TA model.
- Explicit modelling of environment
  - allowing for more relevant testing
  - Allowing for more efficient testing (guiding)



UPPAAL Tron Online Testing

TRON

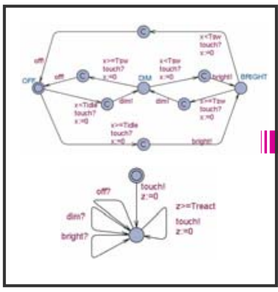


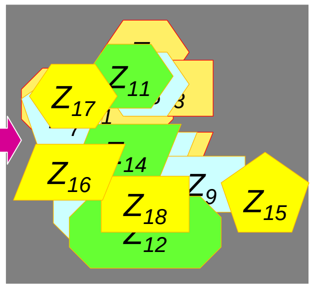
TRON

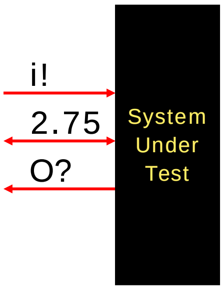
Online State Estimation

Timed Automata Specification

State-set explorer:
maintain and analyse a *set* of *symbolic* states in real time!



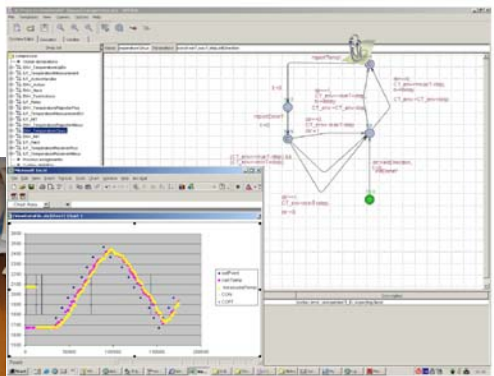




TRON

Industrial Application

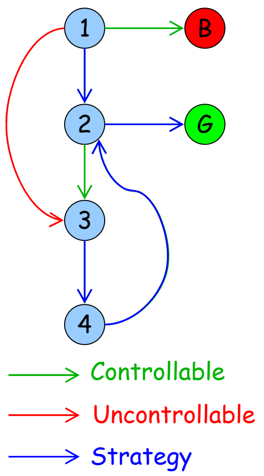
Danfoss Electronic Cooling Controller



TIGA

Untimed Games

- Find a memoryless winning strategy
  - taking controllable edges to reach the Goal
  - that is memoryless
- Rule: 2-player game, controller can choose only controllable transitions
- Winning run:
  - reachability $states \cap G \neq \emptyset$
  - safety $states \cap B = \emptyset$

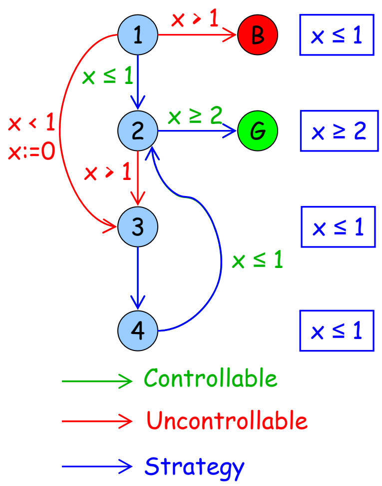


TIGA

Timed Games

- Similar with timed constraints
  - Choose controllable transitions with time constraints!
  - Find memoryless winning strategy
- Algorithm:
  - Timed version of Liu & Smolka 98
  - Forward reachability +
  - Backward fixed-point computation

[CONCUR'05]

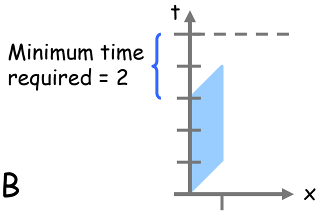


TIGA

Time Optimality Winning Strategy

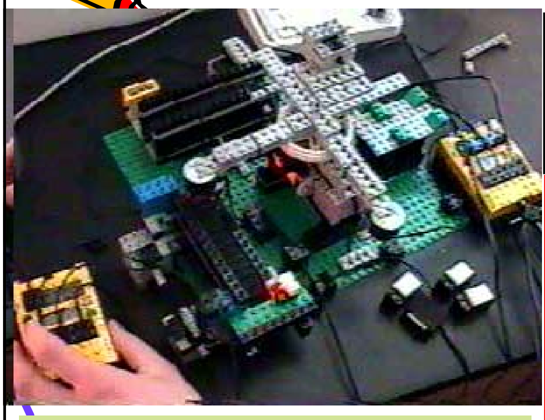
- Assume
  - The game is winning
  - We know an upper bound B for the minimal time needed to reach the goal
- Modification
  - Add a clock t (initially unconstrained)
  - Add the global invariant t ≤ B

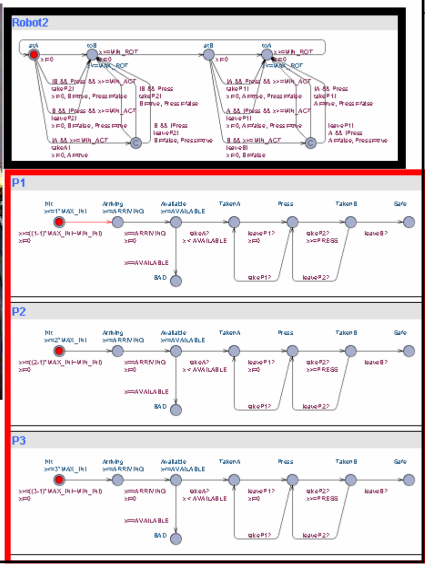
Result:



TIGA

Case Study: Production Cell



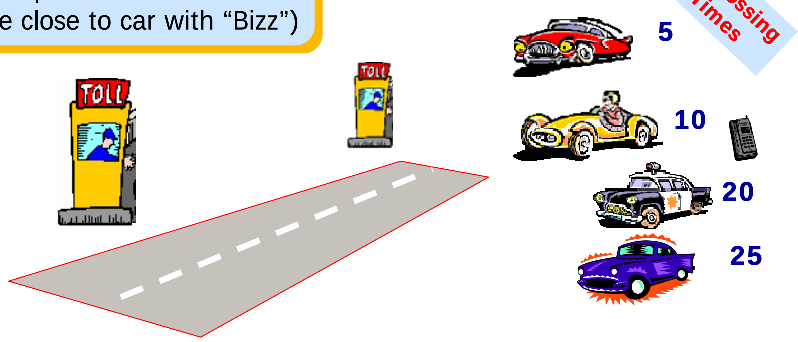


GIVEN System moves **S**, Controller moves **C**, and property ϕ **FIND** strategy s_C such that $s_C||S$ satisfies ϕ

Real-time Scheduling

- Only 1 "BroBizz"
- Cheat is possible
  (drive close to car with "Bizz")

UNSAFE



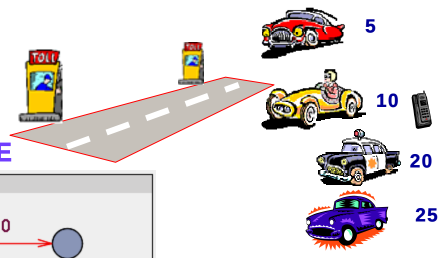
SAFE

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

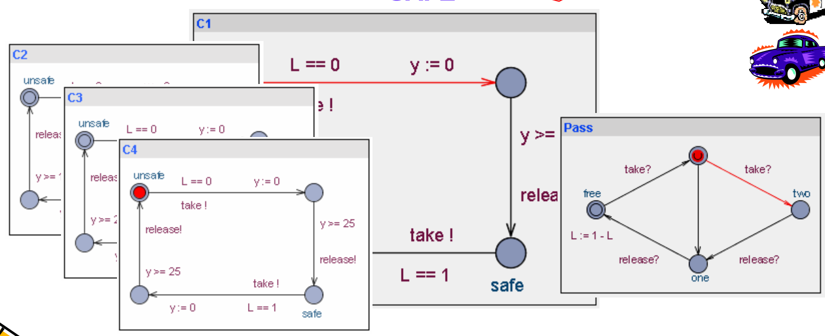
Real-time Scheduling

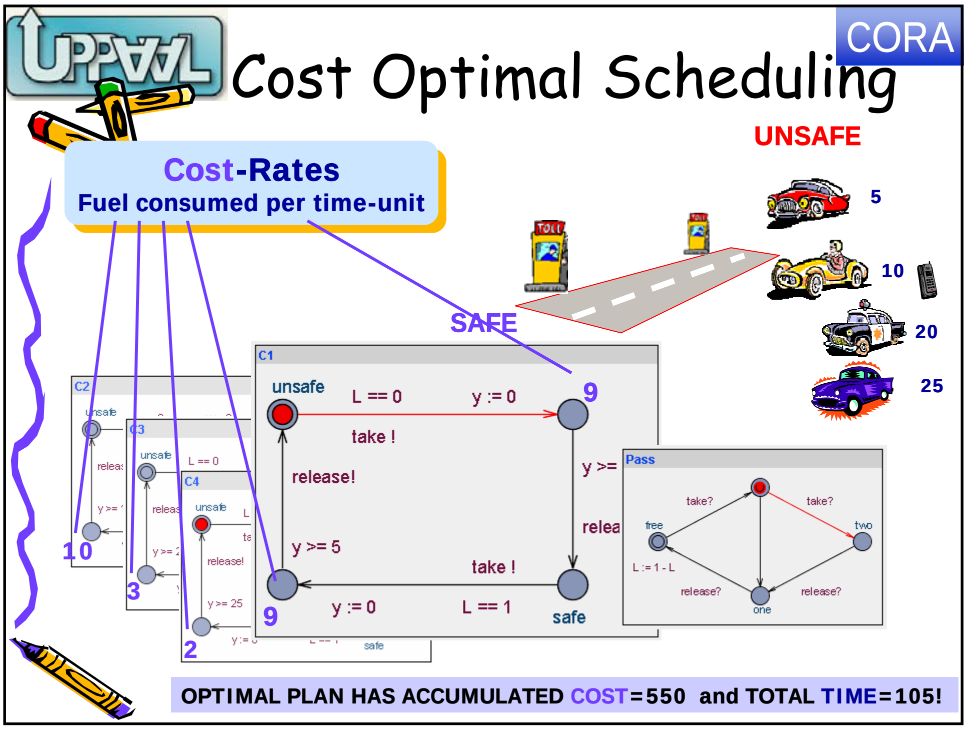
Solve Scheduling Problem using UPPAAL

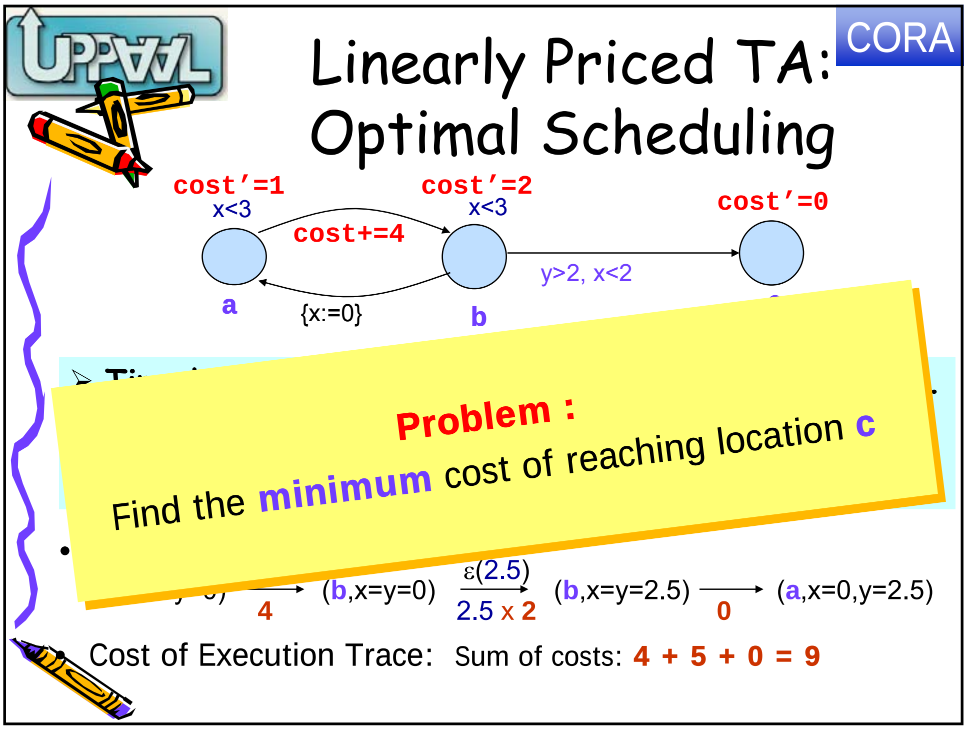
UNSAFE



SAFE

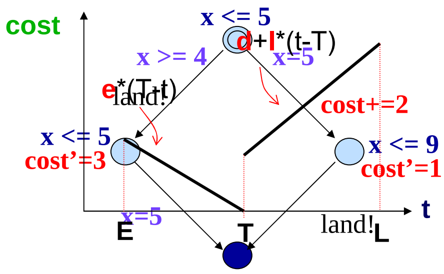



CORA
Cost Optimal Scheduling
UNSAFE
Cost-Rates
Fuel consumed per time-unit
5
10
20
25
SAFE
C1
unsafe
L == 0
y := 0
9
take !
release!
y >=
relea
y >= 5
take !
y := 0
L == 1
safe
9
10
3
2
Pass
take?
take?
free
two
L := 1 - L
release?
release?
one
OPTIMAL PLAN HAS ACCUMULATED COST=550 and TOTAL TIME=105!


CORA
Linearly Priced TA:
Optimal Scheduling
cost'=1
x<3
cost'=2
x<3
cost'=0
cost+=4
a
{x:=0}
b
y>2, x<2
Problem :
Find the minimum cost of reaching location c
(b,x=y=0)
ε(2.5)
2.5 x 2
(b,x=y=2.5)
(a,x=0,y=2.5)
4
0
Cost of Execution Trace: Sum of costs: 4 + 5 + 0 = 9

CORA

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

Runway

CORA

Example: Aircraft Landing

	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	
	explored states	24	46	84	207715	189602	62	N/A
	cputime (secs)	0.36	0.70	1.71	14786.19	12461.47	0.68	
4	optimal value				0	0		
	explored states	N/A	N/A	N/A	65	64	N/A	N/A
	cputime (secs)				1.97	1.53		

CORA

How CORA Works

- Special variables in CORA:
  - cost: the cost as mentioned
  - heur: heuristic value to guide the search
  - rem: lower bound on the remaining time to reach the goal
- Priced zones [CAV01]
- Guided search (with the heuristic variable)
- Branch & bound algorithm to prune the state-space from worse current solutions – in practice much fewer states may be explored (compared to non-cost version)