

UPPAAL now, next, and future

meet a family of model checkers

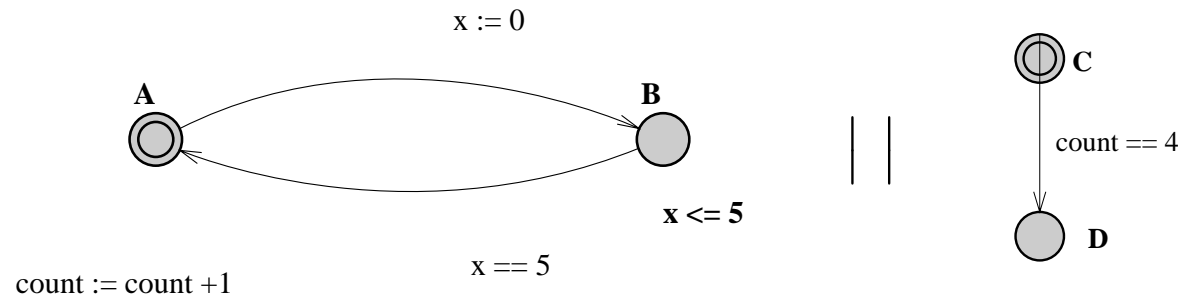
Tobias Amnell¹, Gerd Behrmann², Johan Bengtsson¹, Pedro R. D'Argenio³,
Alexandre David¹, Ansgar Fehnker⁴, Thomas Hune⁵, Bertrand Jeannot², Kim
G. Larsen², **M. Oliver Möller**⁵, Paul Pettersson¹ and Carsten Weise⁶

¹ Uppsala University, ² Aalborg University, ³ University of Twente, ⁴ University of Nijmegen, ⁵ BRICS Aarhus, ⁶ Ericsson Eurolab Deutschland GmbH, Germany

Outline:

1. Model-checking with UPPAAL
2. Internal Optimizations
3. Extensions of the timed automata model
4. Some recent case studies

UPPAAL: Model checking Timed Automata



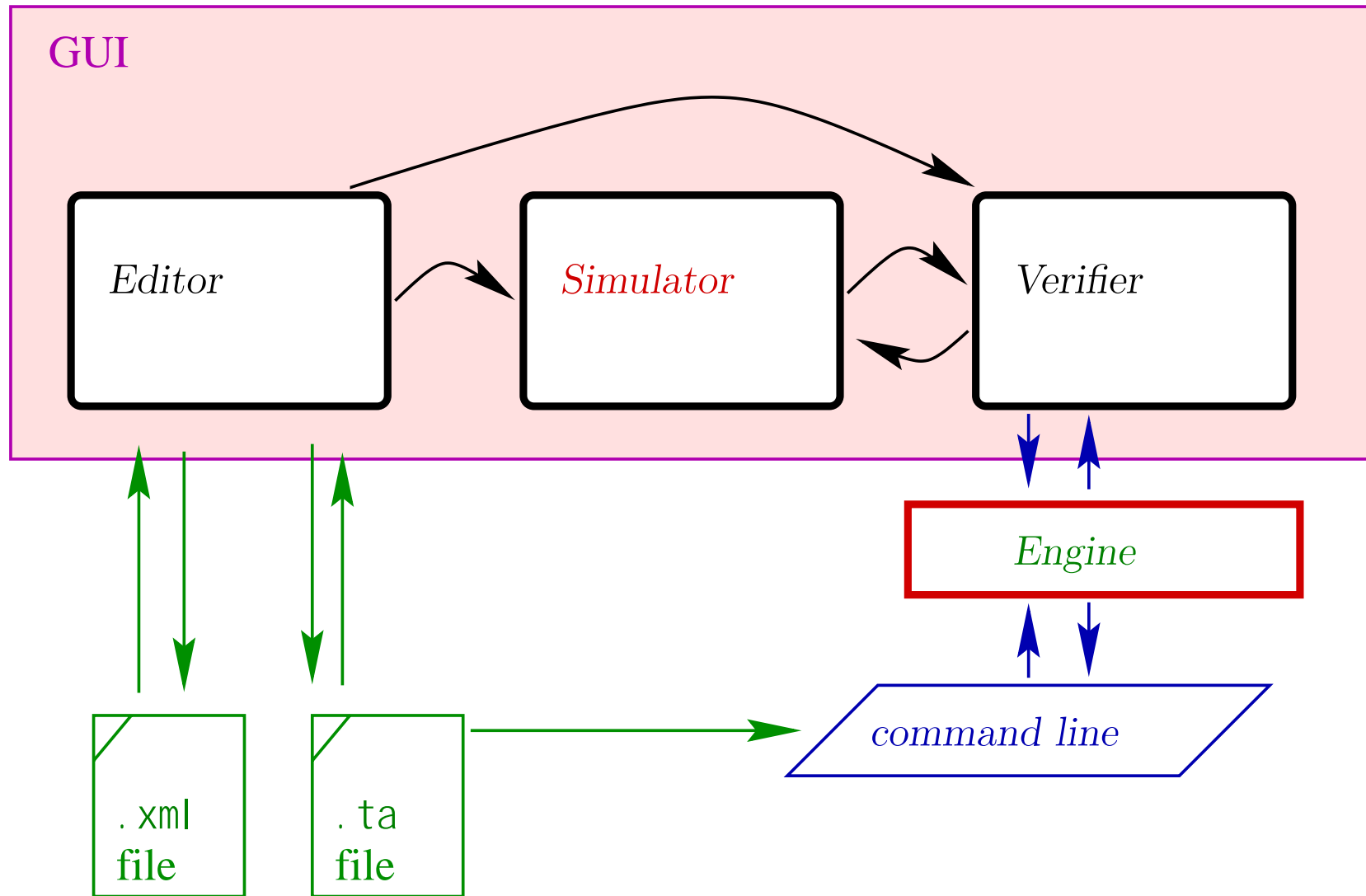
```
clock x; int count
```

Only subset of TCTL supported:

- $E \langle \rangle \varphi$ reachability
- $A [] \varphi$ safety (invariantly φ)
- $E [] \varphi$ possibly always φ
- $A \langle \rangle \varphi$ inevitably φ
- $A [] \varphi \Rightarrow A \langle \rangle \psi$ unbounded response

φ, ψ : propositional formula over locations and (existing) clocks

Architecture of UPPAAL



Forward State Space Exploration

Algorithm: *Reachability*

input: *Goal* : $(\vec{l}_g; v_g)$

Passed := $\{\}$; *Waiting* := $\{(\vec{l}_0; v_0^{\uparrow \vec{l}_0})\}$

REPEAT

FORALL $(\vec{l}; v) \in \textit{Waiting}$

IF $\forall (\vec{l}; v') \in \textit{Passed}. v \not\subseteq v'$ THEN

Passed := *Passed* \cup $(\vec{l}; v)$

FORALL $(\vec{l}'; v')$ with $\vec{l} \xrightarrow{g,r} \vec{l}'$

$v' := r(v \cap g)$

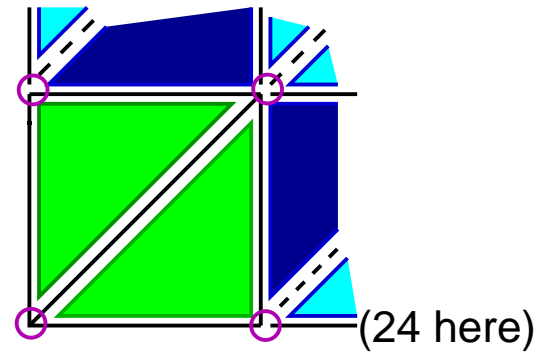
$v' \neq \emptyset$

Waiting := *Waiting* \cup $\{(\vec{l}'; v'^{\uparrow \vec{l}'})\}$

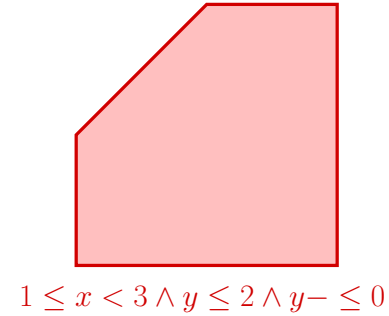
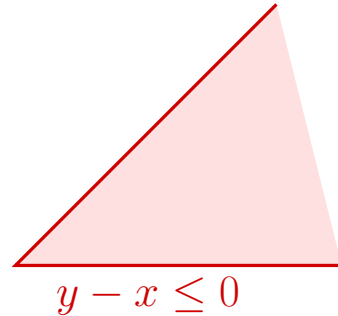
UNTIL *Waiting* = $\emptyset \vee \exists (\vec{l}, v) \in \textit{Passed}. \vec{l}_g \subseteq \vec{l} \wedge v_g \cap v \neq \emptyset$

Sets of Clock-Evaluations

- regions: smallest distinguishable sets



- zones: unions of regions:



For zones (or unions of them), there exist various data structures
DBMs, CDDs, DDDs, ...

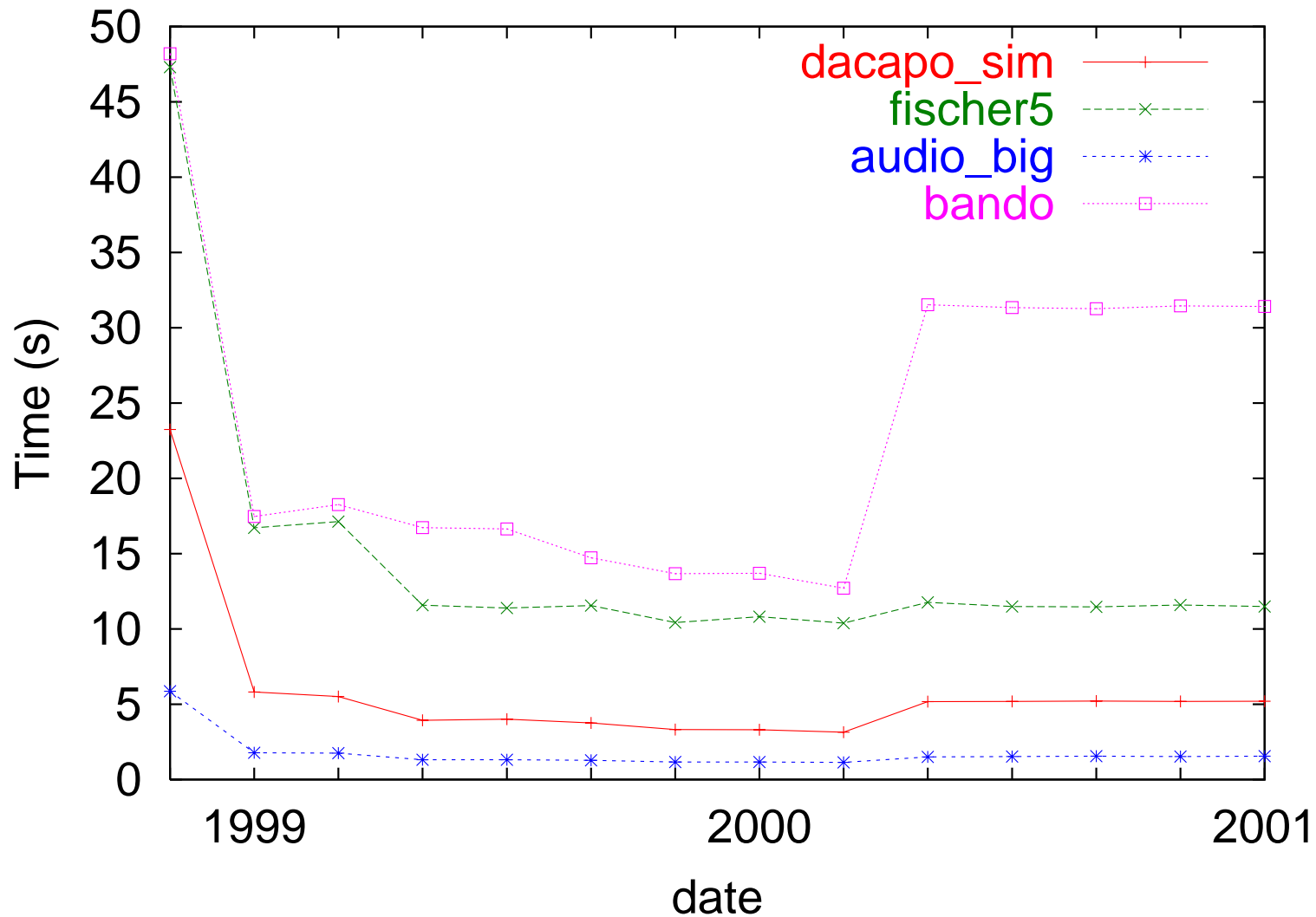
- basic operations:

$\cdot \cap \cdot : R \times R \rightarrow R$	intersection
$\cdot \uparrow : R \rightarrow R$	delay
$\cdot \subseteq \cdot : R \times R \rightarrow bool$	containment

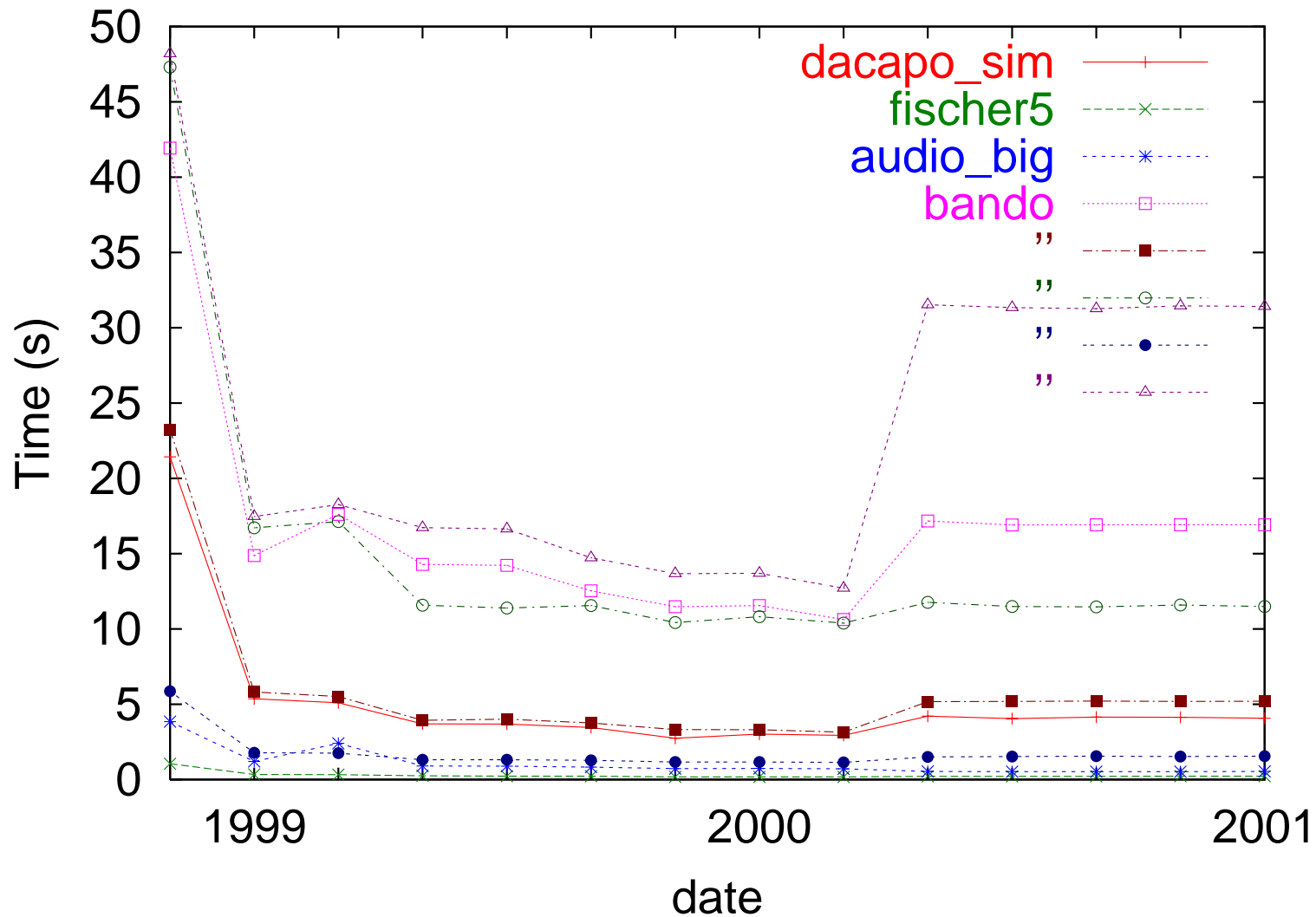
Internal Optimizations

- × committed locations (to reduce interleavings)
 - × active clock reduction
 - × variation of search order
 - × local reduction (compact DBM representation)
 - × global reduction (remove covered states from *Passed*)
 - ≈ convex hull over-approximation [safe]
 - ≈ bit-state hashing [sound]
- ... and of course: *a lot of software engineering!*

Benchmarks (without optimizations)



Benchmarks (with optimizations)



Compact Data Structures

Each state encoded as a **number**, according to a multiply-and-add scheme:

(1) encode *both* location and DBM with this scheme

large space savings

expensive inclusion check

(2) encode DBMs as bit-strings for bounds

only small space savings

cheap inclusion check

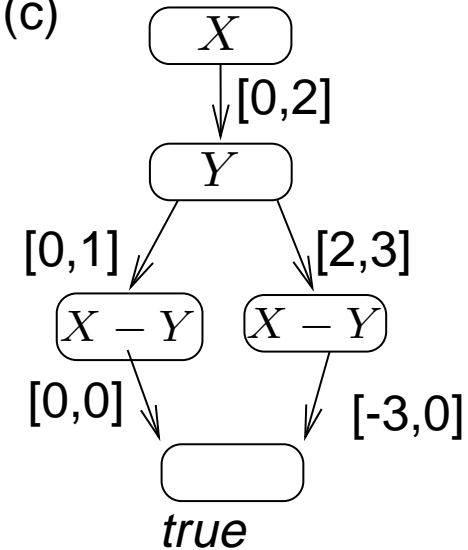
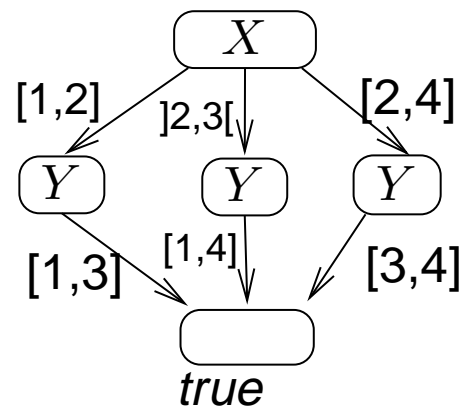
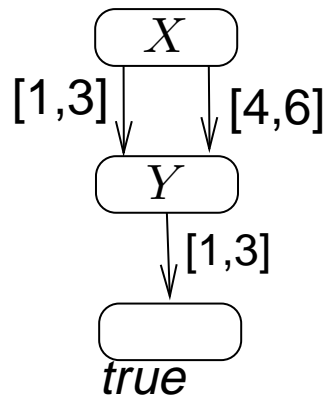
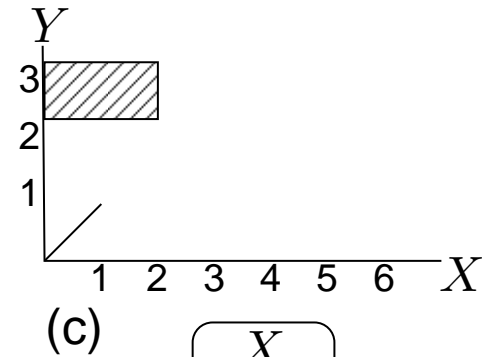
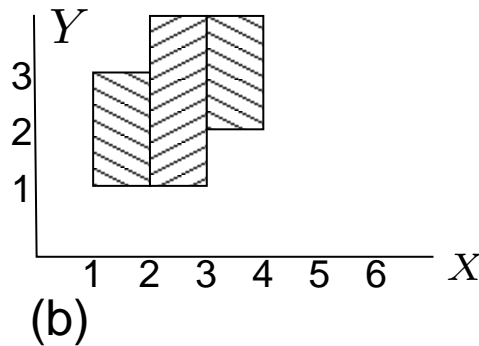
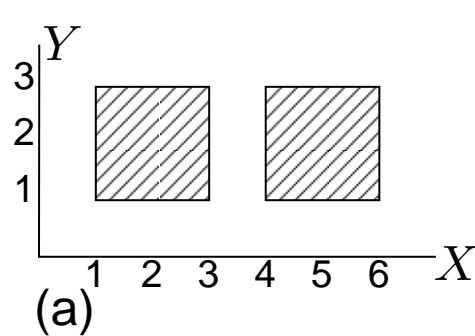
Clock Difference Diagrams (CDDs)

Data structure to express disjunction of zones

- similar to BDDs
- rooted, directed, acyclic graph
- every node labeled x or $x - y$
- every edge labeled with an interval
- order of labels fixed
- one terminal node: true
- missing edges lead to false

 *not canonical*

Clock Difference Diagrams (CDDs) (2)



Distributed Execution of UPPAAL

Forward state-space exploration: spends most of the time browsing *Passed*
no strong interaction of steps

Idea: run state-space exploration on a cluster of machines

Difficulties: distribute work evenly
keep communication between nodes small

Approach: hash according to discrete part of state
order states locally according to smallest distance

Surprising: speed up usually close to linear
- and sometimes even *better*

Approximation by Dynamic Partitioning

Convex-hull approximation: over-approximates clock regions

Discrete part of the state: untouched

Idea: Group together *similar* control locations

Major Difficulty: Adjust the *level* of approximation

Approach: based on abstract interpretation
abstract lattice combines boolean and numerical properties
start with coarse abstraction & refine

Related: techniques used in tool NB_{AC}

Extensions of the Modeling Language

- Stopwatch extension
- Probabilistic timed automata
- Hierarchical timed automata
- Parameters on clock constraints
- Cost-Optimal timed automata
- Executable timed automata

Stopwatch UPPAAL

timed automaton + stopwatches = SWA

Fact: Any *timed language* accepted by a *linear hybrid automaton* can also be accepted by a *stopwatch automaton*

linear hybrid automaton *—translate→* **SWA**

Problem: reachability analysis of SWA is undecidable

Observation: often it suffices to *over-approximate* reachability

Approach: run DBM-based SWA, with *approximative future*
(only differences of *two* stop-watches considered)

Notes: way to *translate* effects accuracy
more sophisticated translations could preserve *termination*

Probabilistic UPPAAL

Example Problem:

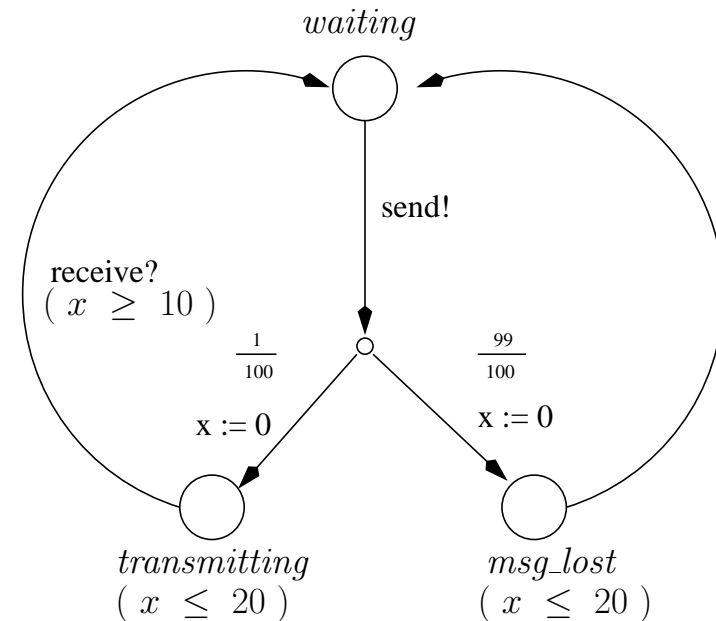
Lossy channel with known probabilities

Cannot prove:

in time X , message will arrive

But:

$P_{\geq 95\%}(\forall \square \leq 1000 \text{ received})$



existing Approaches: Jensen 96, Kwiatkowska et al. 99

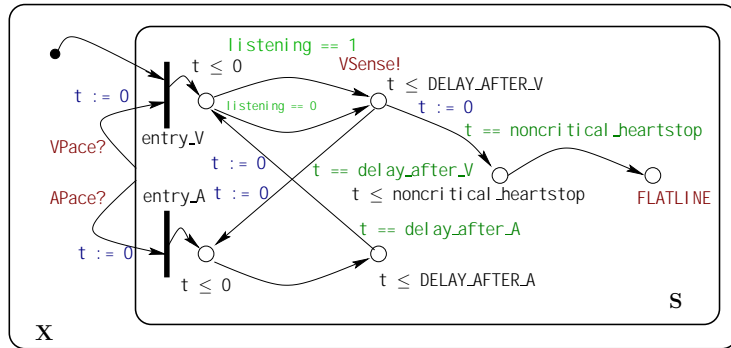
Problem: based on region graph construction

new Approach: use minimization techniques to obtain
stable probabilistic zone graphs

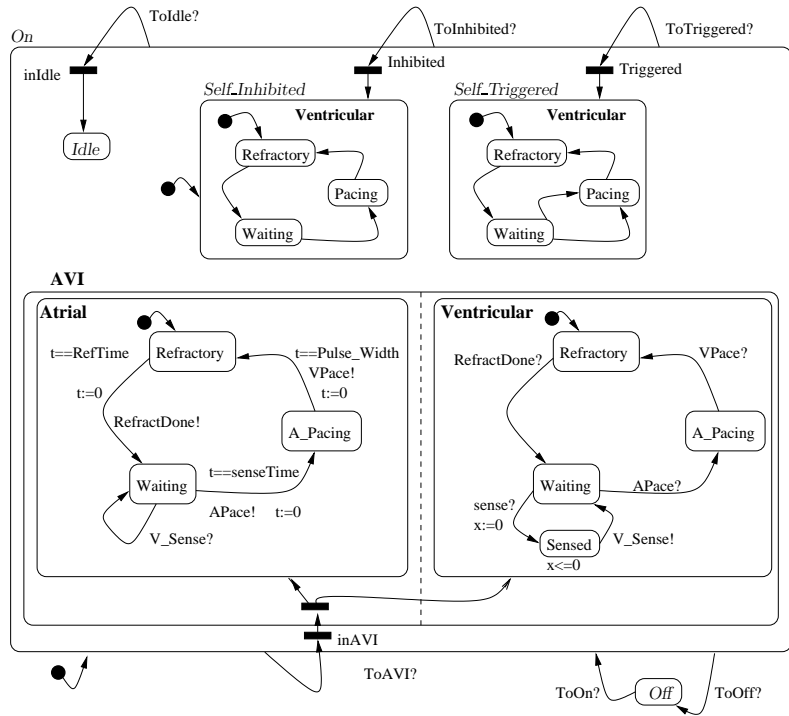
use matching data structure

Hierarchical UPPAAL

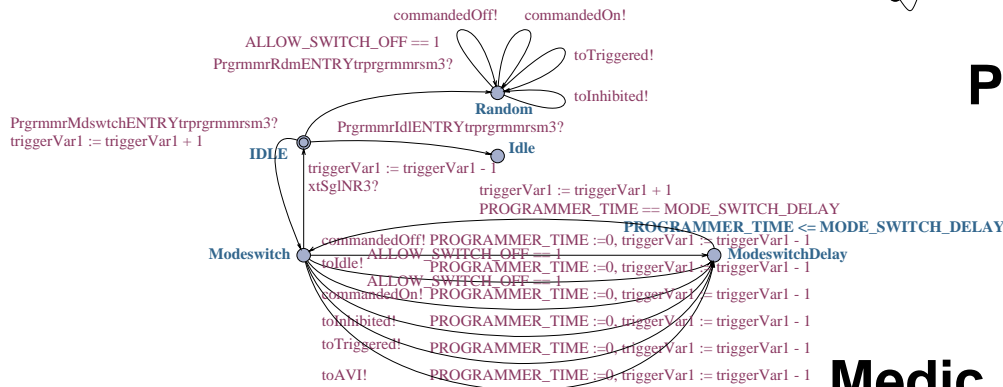
Use **hierarchical** timed automata:



Human Heart



Pacemaker



Medic

Flattened Version of the Pacemaker

HTA model	# XML tags	564	→	1191	UPPAAL model
	# proper control locations	35	→	45	

- SAFETY:

$A[] \neg \text{heart stops}$

- LIVENESS:

$A[] V\text{contract} \Rightarrow A\langle\rangle A\text{contract}$

Parameters:

REFRACTORY_TIME = 50

SENSE_TIMEOUT = 15

DELAY_AFTER_V = 50

DELAY_AFTER_A = 5

MODE_SWITCH_DELAY = 66

E.g. for $\text{MODE_SWITCH_DELAY} = 65$, $A[] \neg \text{heart stops}$ is violated

Determining Parameters: *Parametric-Uppaal*

Parameters: in clock guards $x \bowtie p, x - y \bowtie p$
 $\bowtie \in \{<, \leq, =, \geq, >\}, p$ a linear expression

Fact: parameterized timed reachability undecidable for systems with ≥ 3 clocks [AHV93]

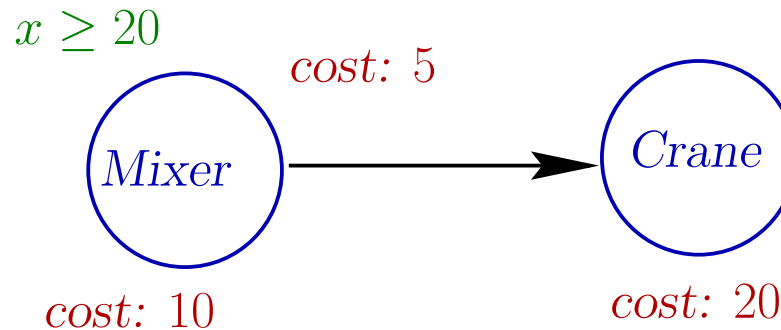
UPPAAL + LP solver (from PMC tool) = *semi-algorithm*

data-structure: parametric DBMs

modified algorithm: split, if the outcome of a comparison is dependent
on parameter values

not guaranteed to terminate \Rightarrow output partial solutions

Cost-Optimality



Idea: Add *cost* to locations and actions

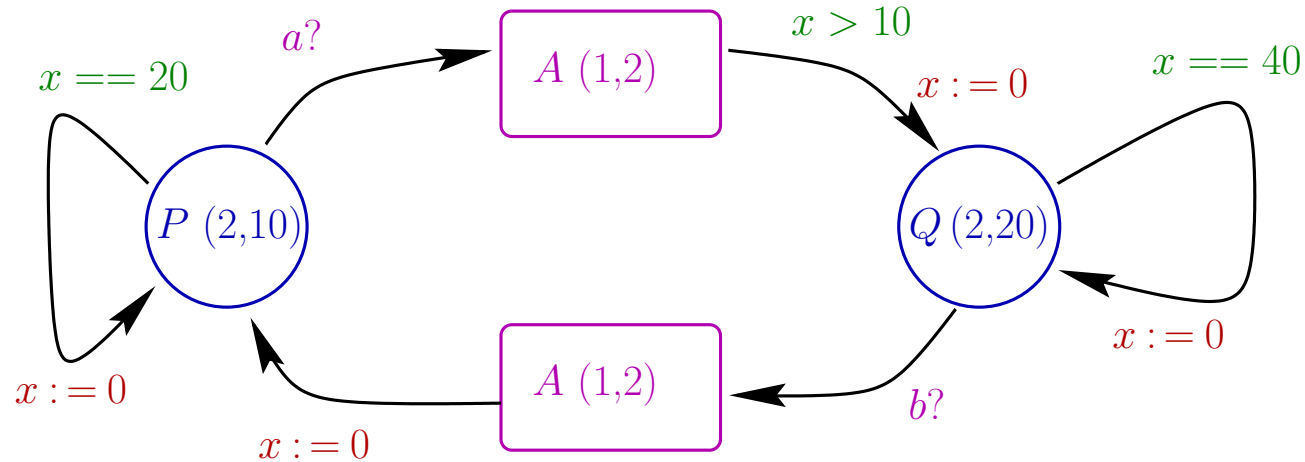
Starting Point: 'cost' not necessarily uniform

Approach: attach different (integer) prices to locations
treat algorithmically with *priced zones*

Applied: compute schedule for a steel batch plant in Ghent
and a LEGO model of it [[Feh99](#), [HLP00](#)]

Fact: Cost-Optimal trace is computable

Executable Timed Automata



Periodic Tasks P, Q

Spontaneous Tasks A, B

Parameters: **worst-case execution time**, **deadline**

Delay transition \equiv execute task with earliest deadline

Action transition \equiv releases a new task

Automaton schedulable \Leftrightarrow every $a!, b!$ -sequence schedulable

Fact: added *Preemption* is as expressive as TAs with stop watches

Recent Case Studies

[LAM99] part of ADA run-time complex formally verified

- ★ uses Ravenscar Tasking Profile to implement concurrency part (to eliminate most implementation)
- ★ prove fourteen properties with UPPAAL
one depends directly on an upper bound on a real-time clock value

[DY00] commercial field bus communication protocol AF100

- ★ developed for safety-critical applications
- ★ been running all over the world for > 10 years
- ★ shows occasionally unexpected behaviors
- ★ flaws in protocol logic and implementation found

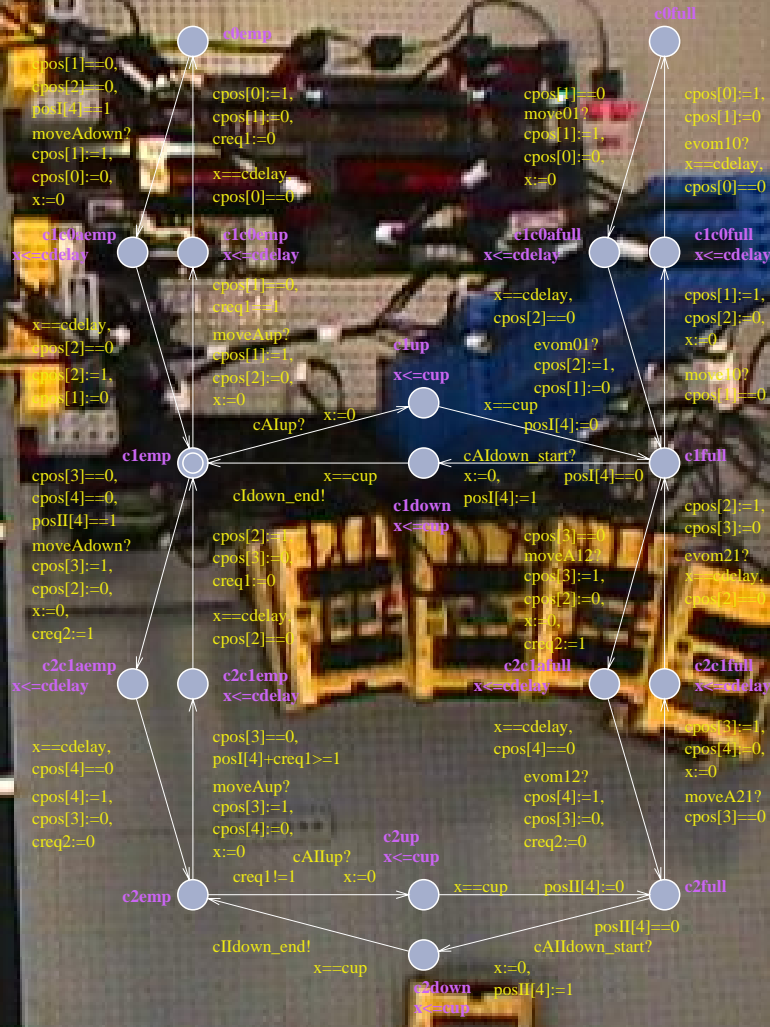
A case study with a LEGO plant

```

'''Delay 15
PB.Wait 2, 1500

'''cAIup();
'''Crane A - Pick UP
PB.PlaySystemSound 1
PB.SendPBMessage 2, 97 'Pick up, on
PB.SetVar 1, 15, 0 'Wait for ack
PB.While 0, 1, 3, 2, 97
PB.Wait 2, 20
PB.SetVar 1, 15, 0 'Read the message
PB.ClearPBMessage
PB.SumVar 2, 2, 1
PB.If 0, 2, 2, 2, 20
'If looped 20 times
PB.PlaySystemSound 1
PB.SendPBMessage 2, 97 'Then Send
'message, again same as sendig 0
PB.SetVar 2, 2, 0
PB.EndIf
PB.EndWhile

'''Delay 10
PB.Wait 2, 1000
    
```



Ongoing Case-Study

[AJ01] Saab Car Locking System

- ★ UPPAAL applied to model and analyze
- ★ derived from functional requirements
- ★ communication network attached to the physical hardware to lock or unlock doors, trunk lid, etc
- ★ distributed over several nodes
- ★ input sources: different kinds of remote controllers, speed sensors, automatic re-locking timeouts

... you will hear more about this later!

UPPAAL in the European WOODDES project

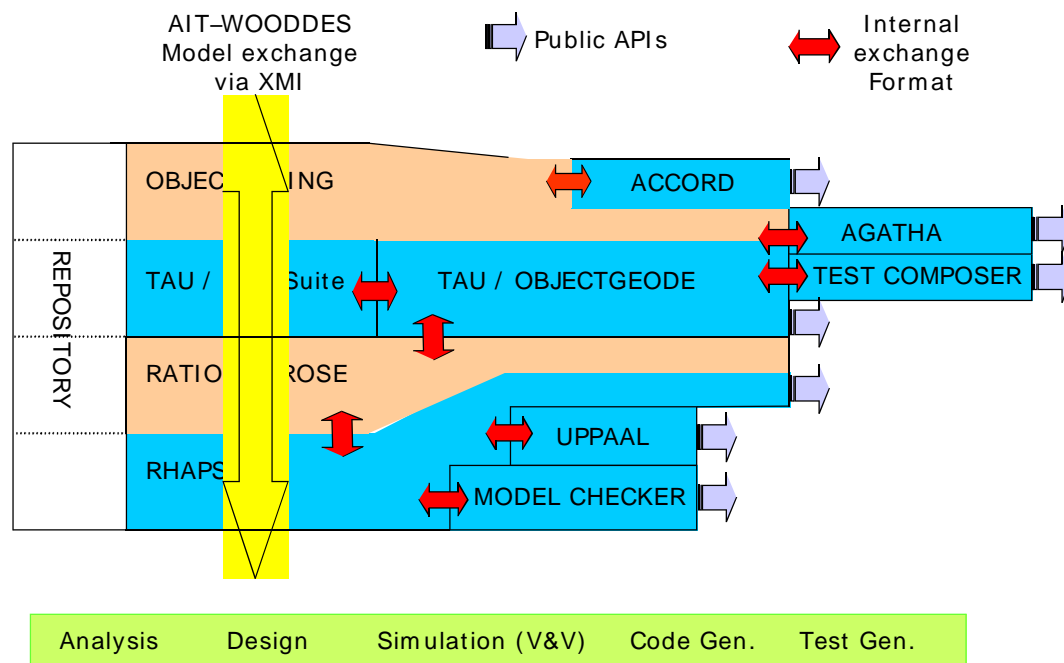
Workshop for Object-Oriented Design and Development of Embedded Systems


Partners:

-  PSA
-  Mecel
-  CEA
-  Telelogic
-  I-Logix
-  Intracom
-  Offis
-  Uppsala
-  Aalborg

Objectives:

- UML Real-Time profile
- WOODDES methodology & tool platform



In  tools owned by project partners

Completed Parts

- ✓ cost-optimal extension
- ✓ parametric extension
- ✓ distributed UPPAAL

Work in Progress

- probabilistic extension
- hierarchical extension
- partial-order techniques
- executable UPPAAL

Work Planned

- ★ dynamic partitioning
- ★ hybrid animation

Go, Get It!

UPPAAL2k (3.2beta) available for

Linux, SunOS, and MS Windows

<http://www.uppaal.com/>

Since July 1999: > 800 downloads (from different users)

> 60 countries

Open mailing list: <http://groups.yahoo.com/group/uppaal>

Bibliography

- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric Real-time Reasoning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 592–601, 1993.
- [AJ01] Tobias Amnell and Pontus Jansson. Report from astec-rt auto project — central locking system case study. In preparation, 2001.
- [BSdRT01] Giosu e Bandini, R. F. Lutje Spelberg, R. C. M. de Rooij, and W. J. Toetenel. Application of Parametric Model Checking - The Root Contention Protocol. In *Proc. of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, 2001.
- [DY00] Alexandre David and Wang Yi. Modelling and Analysis of a Commercial Field Bus Protocol. In *Proc. of 12th Euromicro Conference on Real-Time Systems*, pages 165–172. IEEE Computer Society Press, June 2000.
- [Feh99] Ansgar Fehnker. Scheduling a Steel Plant with Timed Automata. In *Proc. of the 6th International Conference on Real-Time Computing Systems and Applications (RTCSA99)*, pages 280–286. IEEE Computer Society Press, 1999.
- [HLP00] Thomas Hune, Kim G. Larsen, and Paul Pettersson. Guided Synthesis of Control Programs Using UPPAAL. In Ten H. Lai, editor, *Proc. of the IEEE ICDCS International Workshop on Distributed Systems Verification and Validation*, pages E15–E22. IEEE Computer Society Press, April 2000.

of the 18th IEEE Real-Time Systems Symposium. IEEE Computer Society Press, December 1997.

- [Hun99] Thomas Hune. Modelling a Real-time Language. In *Proceedings of FMICS*, 1999.
- [KGLP98] Wang Yi Kim G. Larsen, Carsten Weise and Justin Pearson. Clock difference diagrams. Technical Report 98/99, Department of Computer Systems, Uppsala University, P.O. Box 325, SE-751 05 Uppsala, Sweden., August 1998. Available as <http://www.docs.uu.se/docs/rtmv/papers/lwyp-sub98-1.ps.gz>.
- [LAM99] Kristina Lundqvist, Lars Asplund, and Stephen Michell. A Formal Model of the Ada Ravenscar Tasking Profile; Protected Objects. In Springer-Verlag, editor, *Proc. of the Ada Europe Conference*, pages 12–25, 1999.