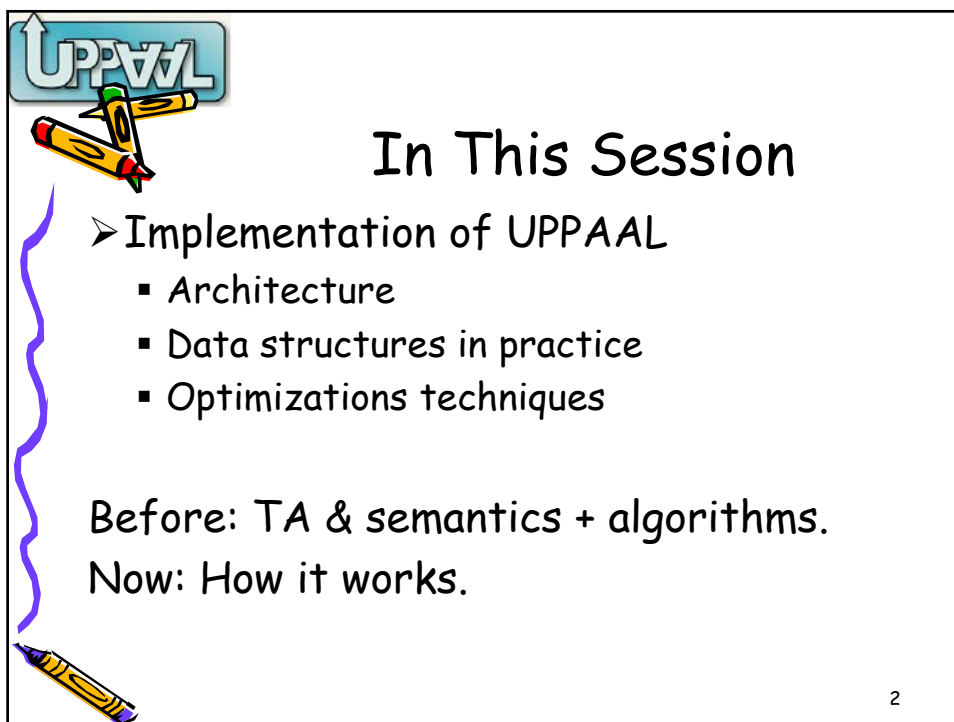


UPPAAL Tutorial

Inside UPPAAL
Advanced

Alexandre David
Paul Pettersson
RTSS'05

The slide features a yellow diamond shape in the center. At the top left, a red and yellow UPPAAL marker is shown with a red squiggle. At the bottom right, a blue and yellow marker is shown with a blue squiggle. The text 'UPPAAL Tutorial' is written in red, and 'Inside UPPAAL Advanced' is written in black. The authors' names and the conference name are listed at the bottom right.



UPPAAL

In This Session

- Implementation of UPPAAL
 - Architecture
 - Data structures in practice
 - Optimizations techniques

Before: TA & semantics + algorithms.
Now: How it works.

2

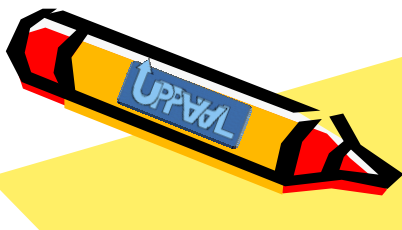
The slide has a blue wavy line on the left side. At the top left, there is a logo for UPPAAL with two markers. At the bottom left, there is a single marker. The text 'In This Session' is in a large font, followed by a list of topics. Below the list, there is a comparison of the current session's focus versus previous work. The number '2' is in the bottom right corner.



Outline

- Architecture of UPPAAL
 - Filters
 - Reachability + liveness + leadsto pipelines
- Memory optimizations
 - Sharing
 - Minimal graph
 - To store or not to store
- Other options/optimizations
 - Symmetry
 - Convex hull + Bitstate hashing

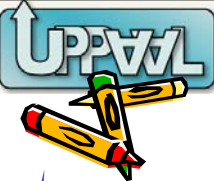
3



Architecture


Pipelines



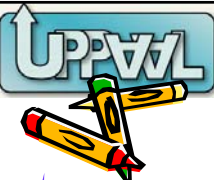


Architecture of UPPAAL

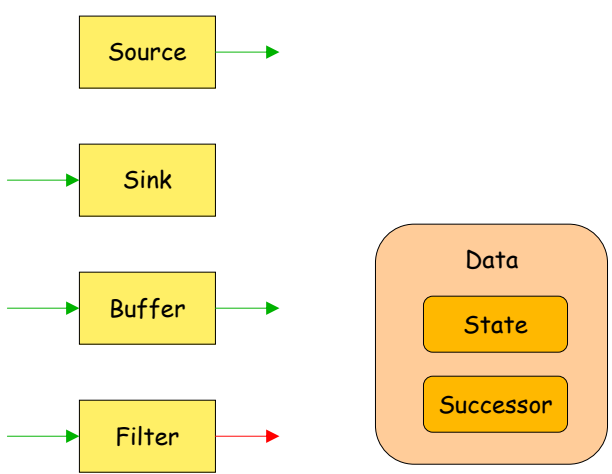
- Pipeline architecture
 - In terms of components and flow of data
 - Not with parallel processing units
- Basic components
 - Sink
 - Source
 - Buffer
 - Filter



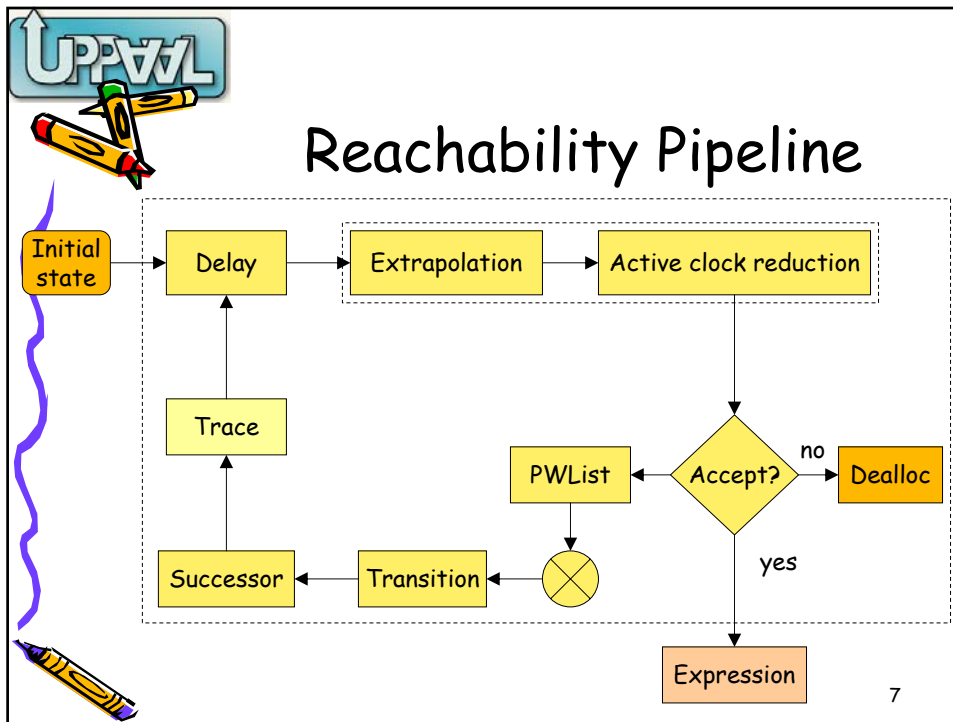
5



Pipeline Components



6



7

-
- Features**
- Reusable/exchangeable components
 - Flexible architecture
 - PWList = passed & waiting list
 - Unified structure
 - Early termination
 - Check property after successor computation, not when taking states from waiting list

8

UPPAAL

Delay

- Initial state pushed here
- Future operation + invariant

9

UPPAAL

Extrapolation

- Different algorithms (choice automatic)
 - Correctness depends on which kind of constraints are used
 - Basic extrapolation:
 - + active clock reduction: if bound = $-\infty$ then free clock

10

UPPAAL

PWList

- PWList = unified passed and waiting list
- Accept = add state if not included in passed + waiting states
- IN: add state to passed + waiting list
- OUT: remove from waiting list

11

UPPAAL

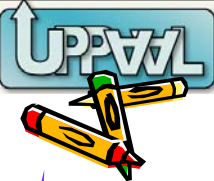
Transition & Successor

- Transition computes possible transitions, not states

- Successor computes successor state

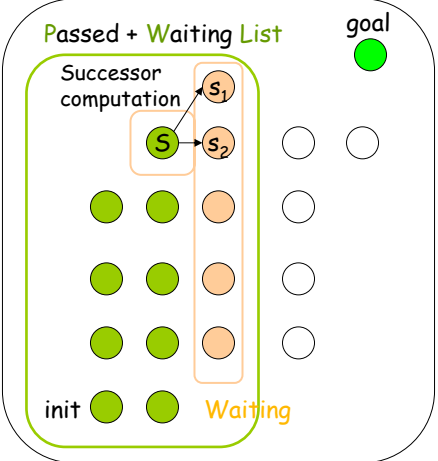
12

Implemented Reachability Algorithm



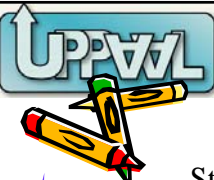
- Passed=Waiting={init}
 - If init == goal then stop
- While Waiting not empty
 - S=pick state from waiting
 - for all successors s_i
 - PWList += s_i
 - if s_i == goal then stop
- Inclusion check in '+='
- One inclusion check (instead of two)
- Earlier termination than classical reachability

Passed + Waiting List goal



13

Classical Passed + Waiting Lists



States

Hash table

Passed list

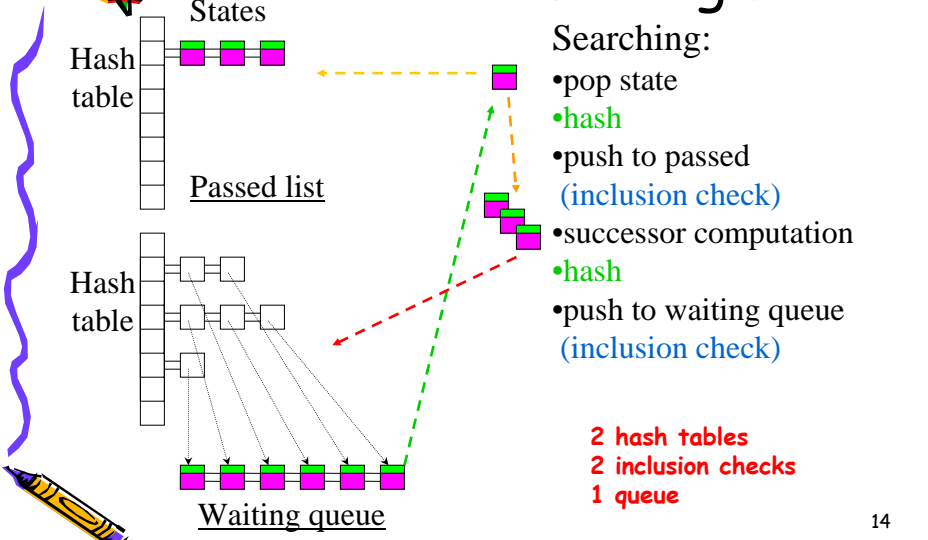
Hash table

Waiting queue

Searching:

- pop state
- hash
- push to passed (inclusion check)
- successor computation
- hash
- push to waiting queue (inclusion check)

2 hash tables
2 inclusion checks
1 queue



14

UPPAAL

PWList

Hash table

States

Unified list

Searching:

- pop state reference
- successor computation
- hash
- push to unified list (inclusion check) and append state reference

Waiting queue

1 hash table
1 inclusion check
1 queue

15

UPPAAL

Active Clock Reduction

x is only active in location S_1

Definition

Clock x is *inactive* at S if on all paths from S , x is always reset before being tested.

16

UPPAAL

Active Clock Reduction

Definition

Clock x is *inactive* at S if on all paths from S , x is always reset before being tested.

$Act(S) =$

$$\bigcup_i Clocks(g_i)$$

$$\bigcup_i (Act(S_i) / Clocks(r_i))$$

Only save constraints on active clocks.

17

UPPAAL

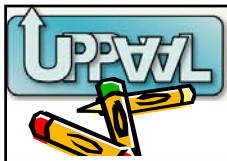
Shortest Path Closure

- Always maintained after operations to avoid $O(n^3)$ whenever possible.
- Vital for inclusion checking between DBMs.

$x_1 - x_2 \leq -4$
 $x_2 - x_1 \leq 10$
 $x_3 - x_1 \leq 2$
 $x_2 - x_3 \leq 2$
 $x_0 - x_1 \leq 3$
 $x_3 - x_0 \leq 5$

Shortest Path Closure
 $O(n^3)$

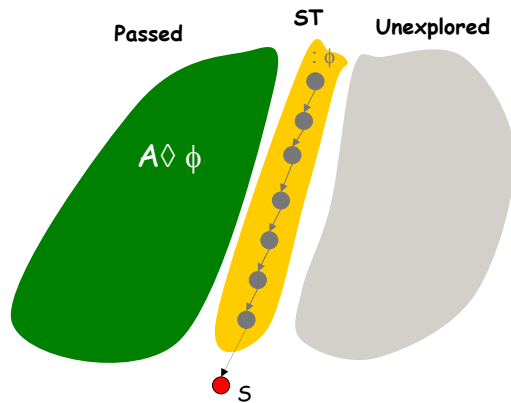
18



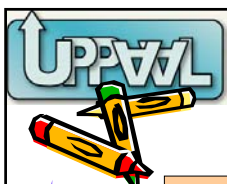
Liveness Algorithm

```

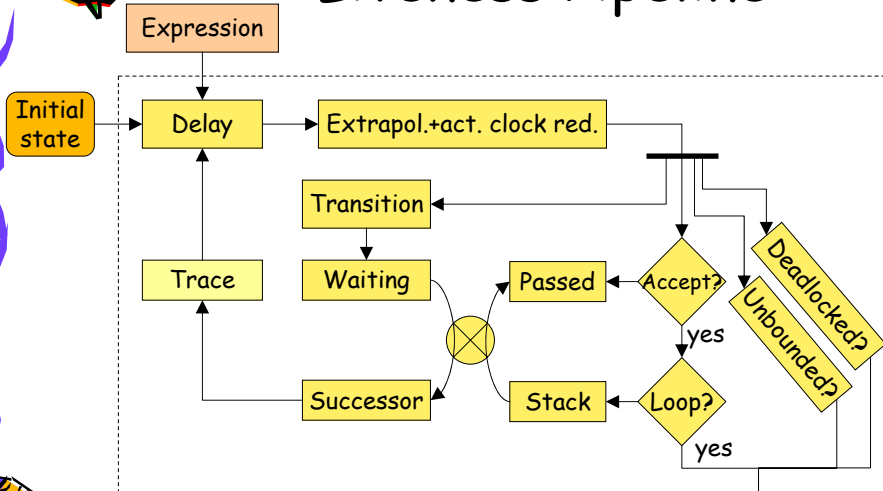
proc Eventually( $S_0, \varphi$ )  $\equiv$ 
   $ST := \emptyset$ 
   $Passed := \emptyset$ 
  Search(delay( $S_0, \neg\varphi$ ))
  exit(true)
end
proc Search( $S$ )  $\equiv$ 
  if loop( $S, ST$ ) then exit(false) fi
   $\bar{S} := S \wedge \neg\varphi$ 
  push( $ST, S$ )
  if unbounded( $S$ )  $\vee$  deadlocked( $S$ ) then
    exit(false) fi
  if  $\forall S' \in Passed : S \not\sqsubseteq S'$ 
    then foreach  $S' : S \xrightarrow{a} S'$  do
      Search(delay( $S', \neg\varphi$ ))
    od
  fi
   $Passed := Passed \cup \{pop(ST)\}$ 
end
  
```



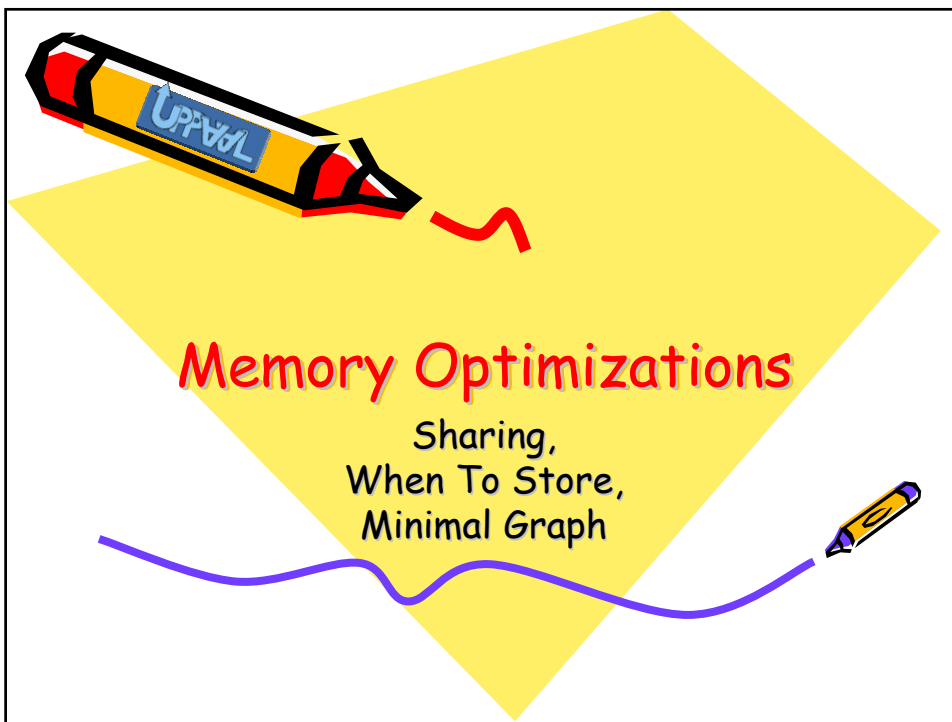
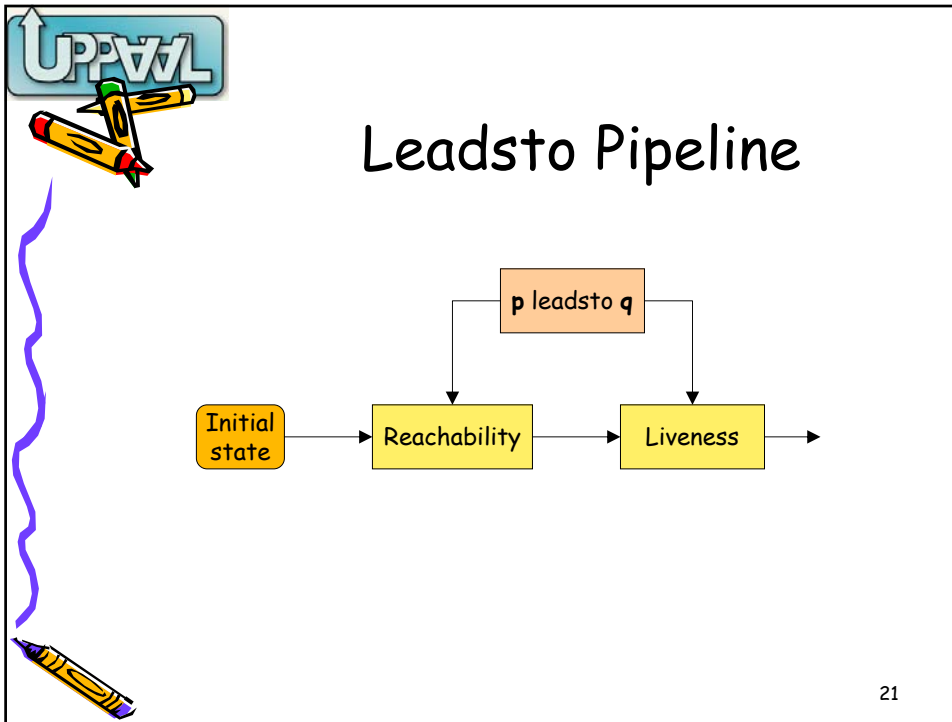
19

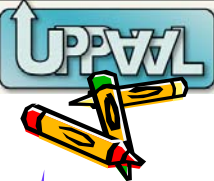


Liveness Pipeline



20

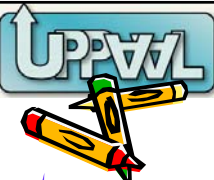




Data Sharing

- Key idea: Working states different from stored states
 - **Working states** optimized for **computation**
Symbolic state = discrete part (location+variables) + symbolic part (DBM).
 - **Stored states** optimized for **memory**
Stored state = $\langle \text{lockey}, \text{varkey}, \text{dbmkey} \rangle$.

23



Data Sharing

Symbolic state for computation

Location vector
Variables
DBM

Symbolic state for storage (PWList)

lockey
varkey
dbmkey

Sharing of data

Discrete storage

Symbolic storage

save → ← load

inclusion? ↔

24



Data Sharing

- In practice: 80% reduction.
- Easy to change storage implementation to favor speed or memory.
 - Compression of integer paired with minimal graph
 - Convex hull is a special storage

25



PWList & Sharing In Figures

Model	Before		Unification		Unication & Sharing	
Audio	≤ 0.5s	2M	≤ 0.5s	2M	≤ 0.5s	2M
Engine	≤ 0.5s	3M	≤ 0.5s	4M	≤ 0.5s	5M
Dacapo	3s	7M	3s	5M	3s	5M
Cups	43s	116M	37s	107M	36s	26M
BC	428s	681M	359s	641M	345s	165M
Master	306s	616M	277s	558M	267s	153M
Slave	440s	735M	377s	645M	359s	151M
Plant	19688s	> 4G	9207s	2771M	8513s	1084M

[SPIN03]

26

UPPAAL

State-space Reduction When to Store?

- No loop => The **passed** list is not needed for termination.
- Loops => Only symbolic states involving loop-entry points need to be saved in the **passed** list.

UPPAAL

To Store Or Not To Store?

117 states_{total}
 ↓
 81 states_{entrypoint}
 ↓
 9 states

Time OH
 less than 10%
 (need to re-explore some states)

[CAV03]
 Audio Protocol

UPPAAL

Minimal Graph

$$\begin{aligned}
 &x1-x2 \leq -4 \\
 &x2-x1 \leq 10 \\
 &x3-x1 \leq 2 \\
 &x2-x3 \leq 2 \\
 &x0-x1 \leq 3 \\
 &x3-x0 \leq 5
 \end{aligned}$$

Shortest Path Closure
 $O(n^3)$

(DBM)

Shortest Path Reduction
 $O(n^3)$

**Space worst $O(n^2)$
practice $O(n)$**

(Minimal graph, a.k.a. compact data structure)

29

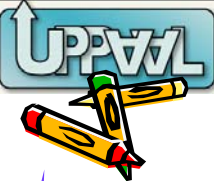
UPPAAL

Graph Reduction Algorithm

G: weighted graph

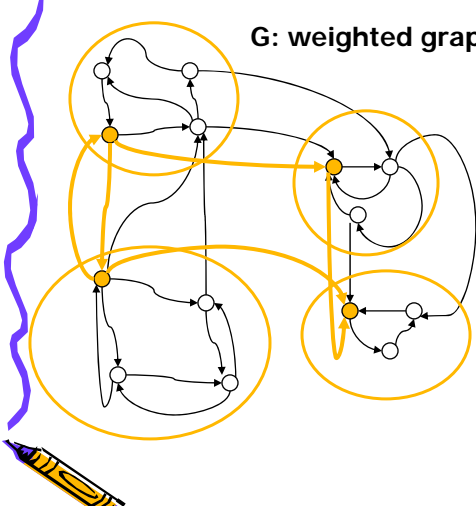
1. Equivalence classes based on 0-cycles.

30



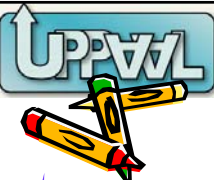
Graph Reduction Algorithm

G: weighted graph



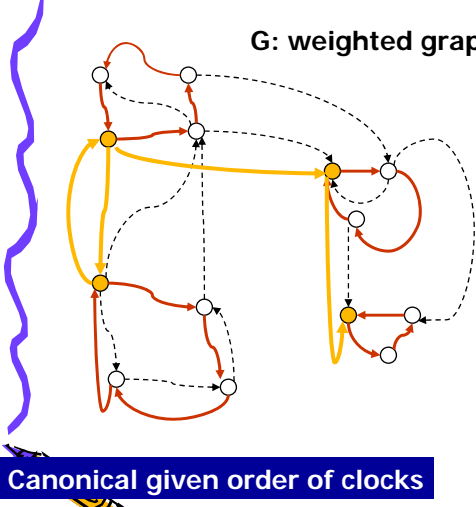
1. Equivalence classes based on 0-cycles.
2. Graph based on **representatives**.
Safe to remove redundant edges

31



Graph Reduction Algorithm

G: weighted graph



1. Equivalence classes based on 0-cycles.
2. Graph based on **representatives**.
Safe to remove redundant edges
3. **Shortest Path Reduction**
= One cycle pr. class
+ Removal of redundant edges between classes

32

Canonical given order of clocks

Other Optimizations

Symmetry,
Active Clocks,
Approximations

Symmetry Reduction

- Exploitation of full symmetry may give factorial reduction.
- Many timed systems are inherently symmetric.
- Computation of canonical state representative using swaps.

[Formats 2003]

SWAP: 1→2 ; 3→4

Variables

```

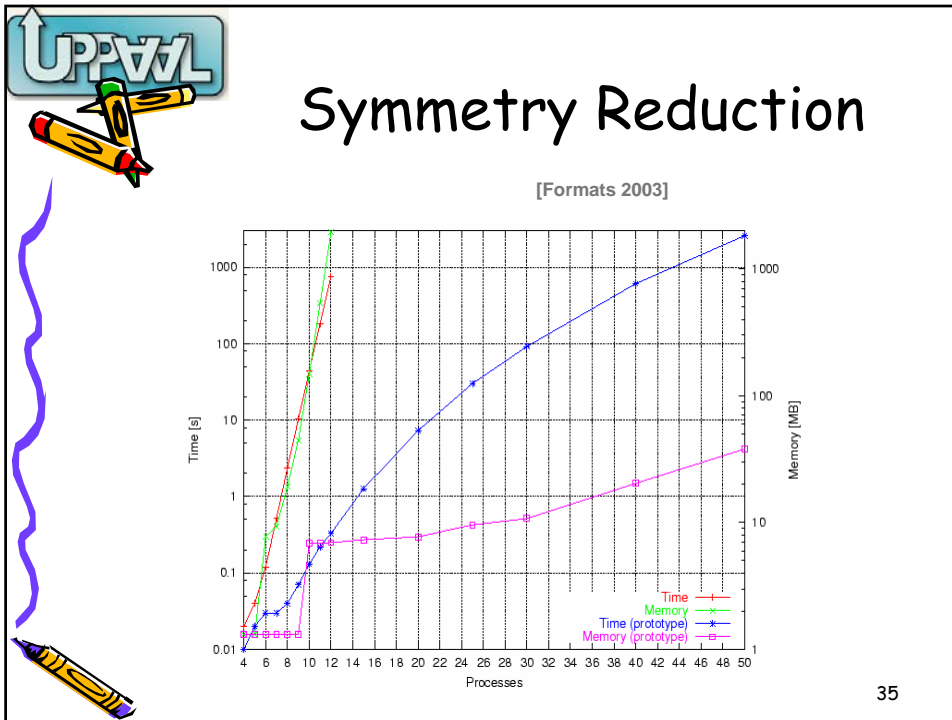
id = 2
P2.x in [0,2]
P4.x in [0,2]
P2.x - P1.x in [-inf,0]
P3.x = P1.x
P4.x - P1.x in [-inf,0]
P3.x - P2.x in [0,inf]
P4.x - P2.x in [0,2]
P4.x - P3.x in [-inf,0]
          
```

P1

P2

P3

P4



Support For Symmetry

- **Scalar set based symmetry reduction**
 - `typedef scalarset[4] pid_t;`
`scalarset[n] = {0,...,n-1}`
 - `int[0,4]` = set of integers
 - **Template sets** `process P[i:pid_t](...) {(i)}`
 - **Iterators** `for (i:pid_t) { a[ix]=0 }`
 - **Quantifiers** `forall (i:int[0,4]) a[i+1]==0`
`exists (i:int[0,4]) a[i+1]==1`
 - **Selection** `select i: int[0,4]; guard...`

Martijn Henriks, Nijmegen U

UPPAAL

Over-approximation Convex Hull

The diagram shows a state transition graph with states $s_1, s_2, s_3, s_4, s_5, s_6$ and transitions labeled with coordinates like $x > 1, y > 1$. A dashed blue line encloses the entire graph, representing the convex hull. To the right, a 2D coordinate system shows a grid with a dashed blue rectangle labeled "Convex Hull" that encompasses the points of the graph.

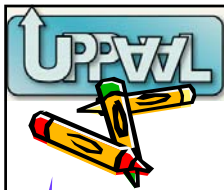
37

UPPAAL

Under-approximation Bitstate Hashing

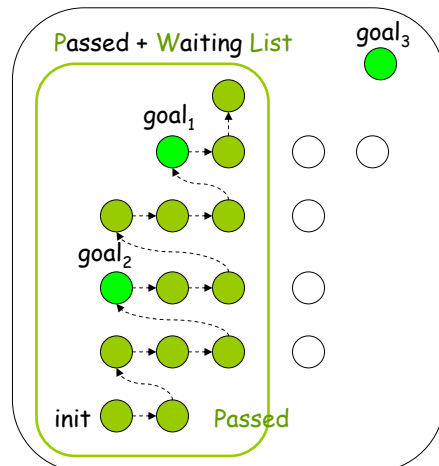
- Passed list = bit vector.
 - 1 bit per state.
 - Index = hash(state) (discrete & symbolic parts)
- Waiting list = symbolic states.
- PWList += state
 - If $\text{vector}[\text{hash}(\text{state})] == 0$
 waiting += state
 $\text{vector}[\text{hash}(\text{state})] = 1$

38

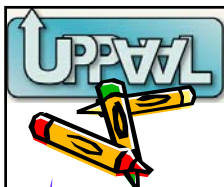


Re-using The State-space

- Several properties to check:
 $A \square \text{prop1}$
 $A \square \text{prop2}$
...
- Search in existing passed list (from previous checks) first.
- Expand missing states (not all states stored).



39



Virtual Machine

- Expressions (guards & actions) are compiled to *bytecode* and executed by a virtual machine.
- Stack machine, minimal instruction set, peep-hole optimization.
- Open the door to other optimizations or use of 3rd party VM.


Nips (Michael Weber): VM for Promela matches performance of Spin.

40

UPPAAL

Distributed UPPAAL

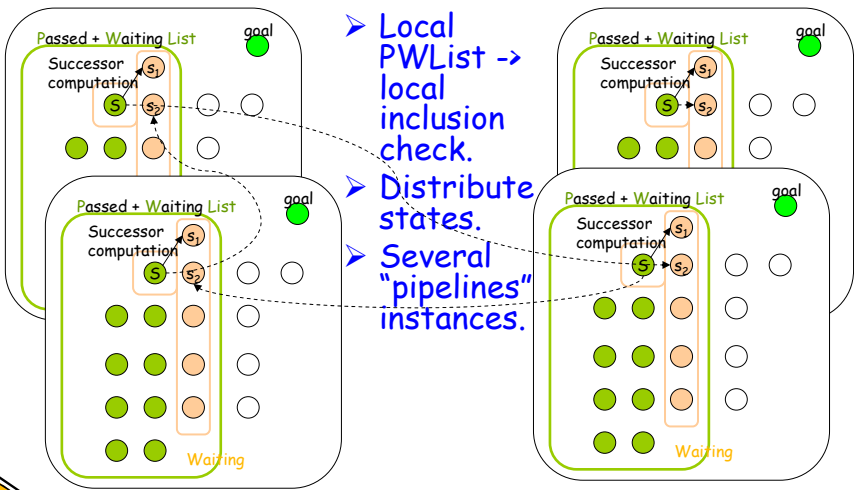
- > Distributed implementation of UPPAAL on PC-cluster [CAV'00, PDMC'02, STTT'03].
- > Applications
 - Synthesis of Dynamic Voltage Scaling strategies (CISS).
 - Real-time leader election protocol for mobile ad-hoc networks (Leslie Lamport) - 25GB in 3 min!
- > Running on NorduGrid. Local cluster: 50 CPUs and 86GB of RAM
- > To be used as inspiration for verification GRID platform within ARTIST2.



Gerd Behrmann 41

UPPAAL

Distributed UPPAAL



- > Local PWList -> local inclusion check.
- > Distribute states.
- > Several "pipelines" instances.

42