Formal
methods
& Tools

University of Twente
*department of
computer science*

UPPAAL
Implementation Secrets

Kim Guldstrand Larsen
BRICS@Aalborg & FMT@Twente

UCb

Collaborators

@UPPsala

- Wang Yi
- Paul Pettersson
- Johan Bengtsson
- Fredrik Larsson
- Alexandre David
- Tobias Amnell
- Leonid

@AALborg

- Kim G Larsen
- Gerd Behrman
- Arne Skou
- Patricia Bouyer
- Emmanuel Fleury
- Carsten Weise
- Kåre J Kristoffersen
- Thomas Hune
- Oliver Möller

@Elsewhere

- David Griffioen, Ansgar Fehnker, Frits Vandraager, Klaus Havelund, Theo Ruys, Pedro D'Argenio, J-P Katoen, J. Tretmans, Judi Romijn, Ed Brinksma, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson...

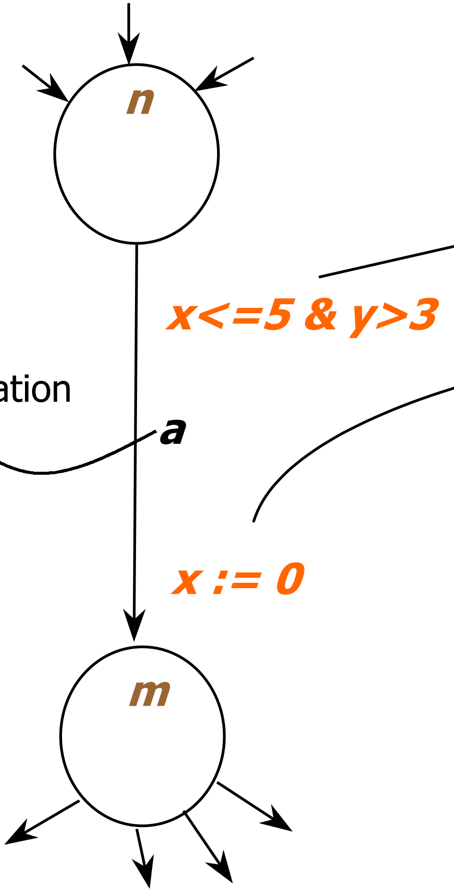
Overview

- Timed Automata (review)
- UPPAAL 3.2 (3.3.23 and 4.0)
- The Early History of UPPAAL
- The Implementation Secrets of UPPAAL
  - DBMs
  - Minimal Constraint Form
  - CDDs
  - Distributed UPPAAL
  - PW-list, Dynamic sized DBMs, Sharing ….
- Acceleration and Metatransitions
- Beyond Model Checking

Timed Automata *review*

Alur & Dill 1990

Clocks: *x, y*



Action used for synchronization

Guard
Boolean combination of integer bounds on **clocks** and **clock-differences**.

Reset
Action perfomed on clocks

State

(***location*** , ***x***=v , ***y***=u) where v,u are in $\mathbb{R}$

Transitions

Discrete Trans

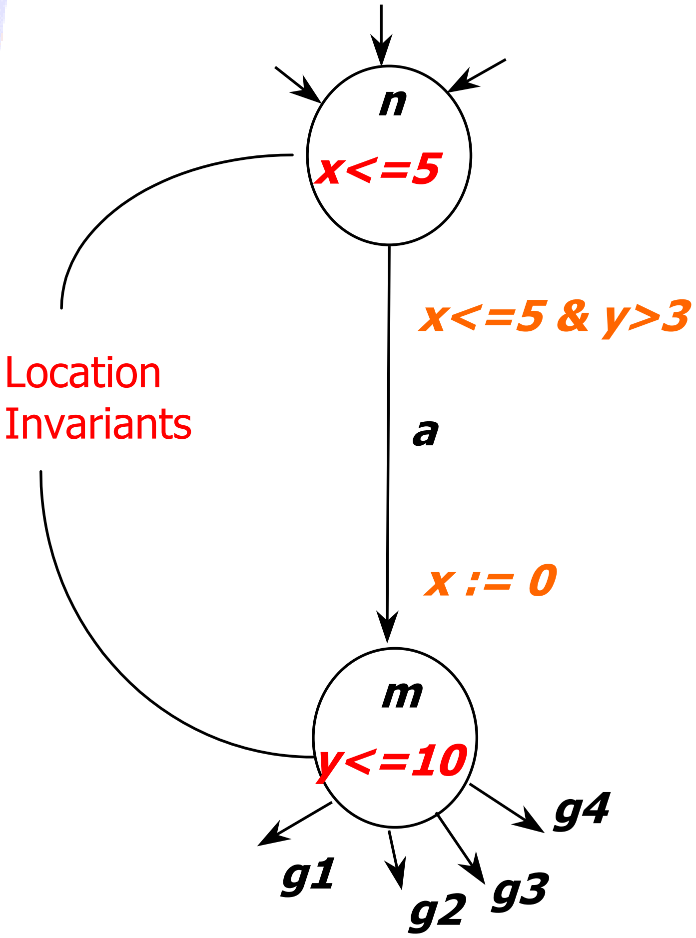
$(n , x=2.4 , y=3.1415) \xrightarrow{a} (m , x=0 , y=3.1415)$

Delay Trans

$(n , x=2.4 , y=3.1415) \xrightarrow{e(1.1)} (n , x=3.5 , y=4.2415)$

Timed Automata *review*

Invariants



Clocks: ***x, y***

Transitions

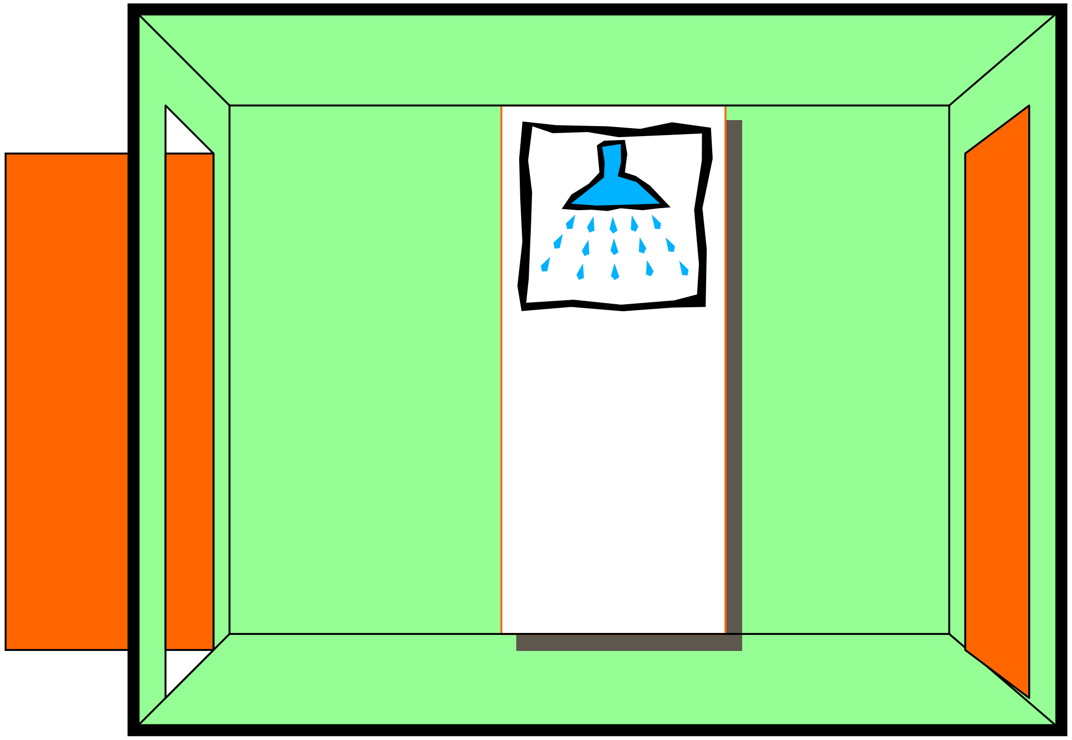
$(n , x=2.4 , y=3.1415) \xrightarrow{\cancel{e(3.2)}}$

$(n , x=2.4 , y=3.1415) \xrightarrow{e(1.1)} (n , x=3.5 , y=4.2415)$

Invariants ensure progress!!

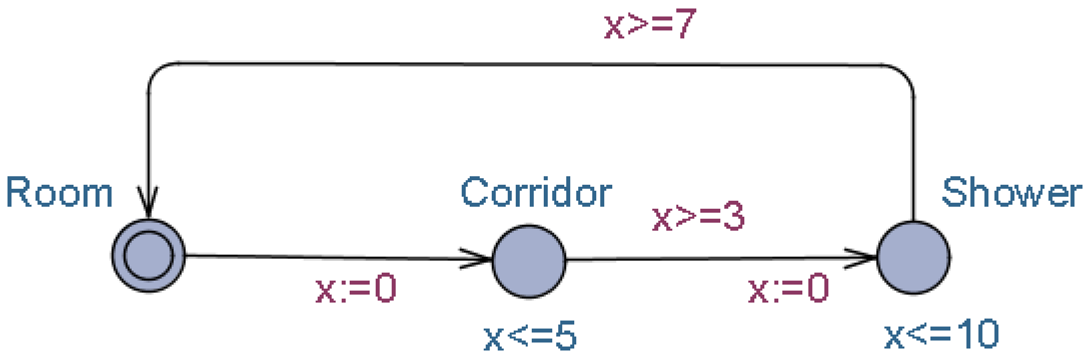
The Druzba MUTEX Problem

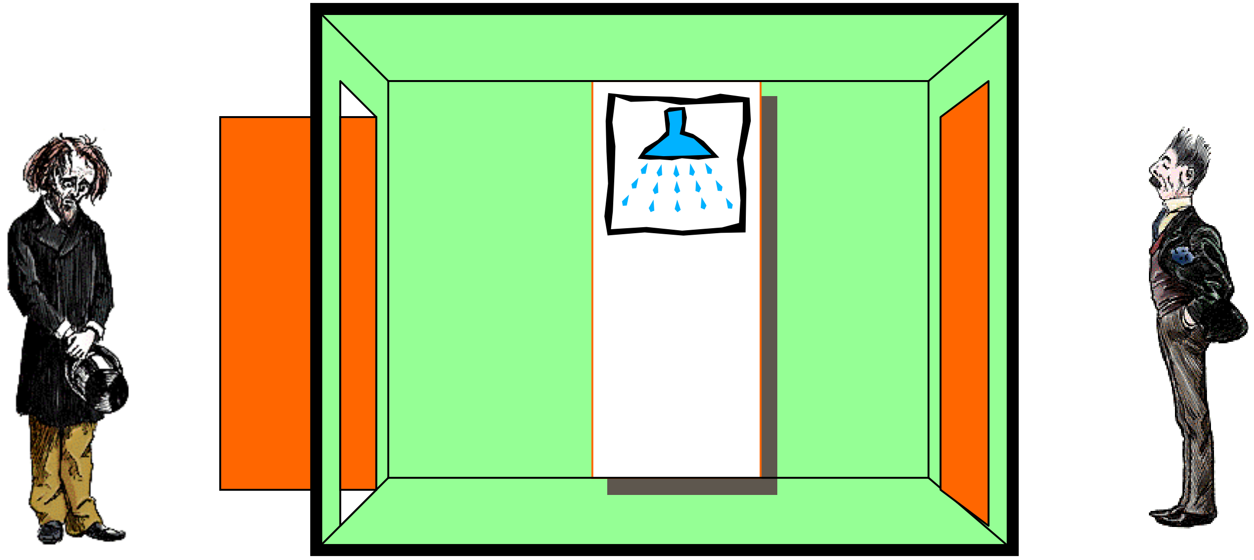
Kim



Gerd

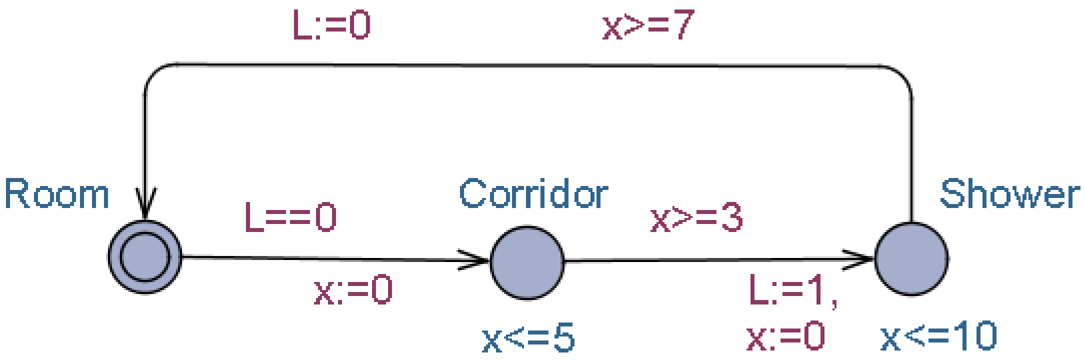
The Druzba MUTEX Problem

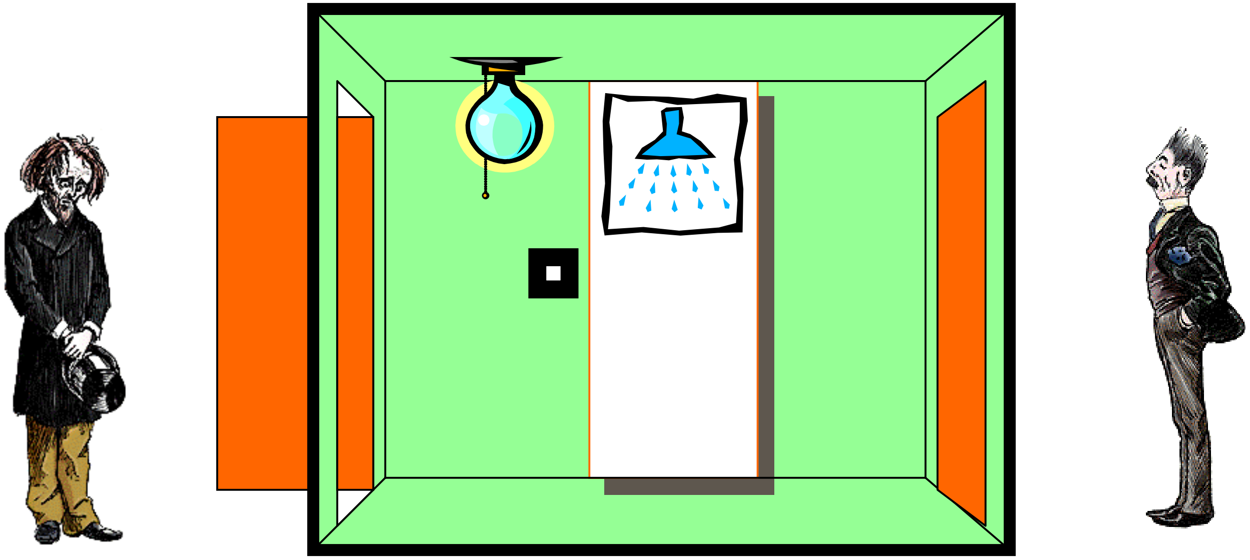




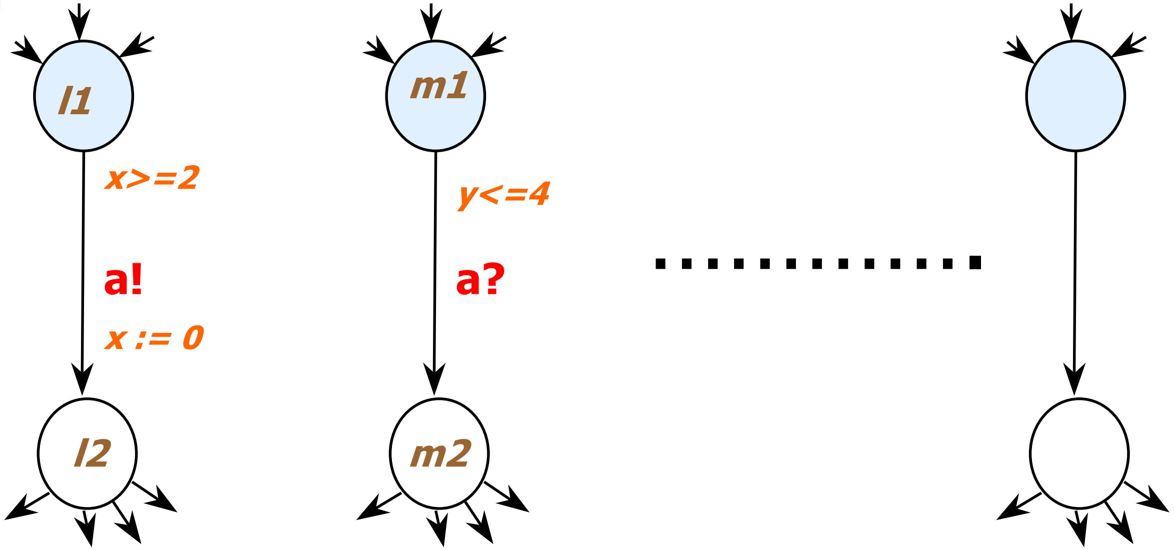
The Druzba MUTEX Problem

Using the light as semaphor





Parallel Composition (a'la CCS)



Two-way synchronization on *complementary* actions.

Closed Systems!

Example transitions

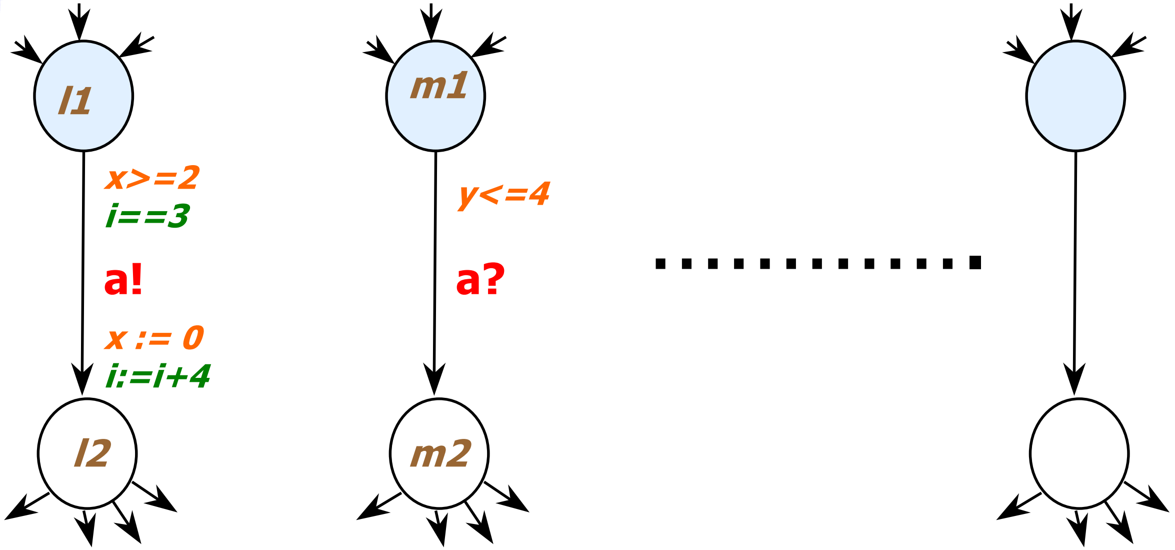
(***l1***, ***m1***,………, *x=2, y=3.5,…..)* $\xrightarrow{\text{tau}}$ *(**l2**,**m2**,……..,x=0, y=3.5, …..)*

$\cancel{\xrightarrow{0.2}}$ *(**l1**,**m1**,………,x=2.2, y=3.7, …..)*

If **a** URGENT CHANNEL

The UPPAAL Model

= Networks of Timed Automata + Integer Var + Array Var + ….



Two-way synchronization on *complementary* actions.

Closed Systems!

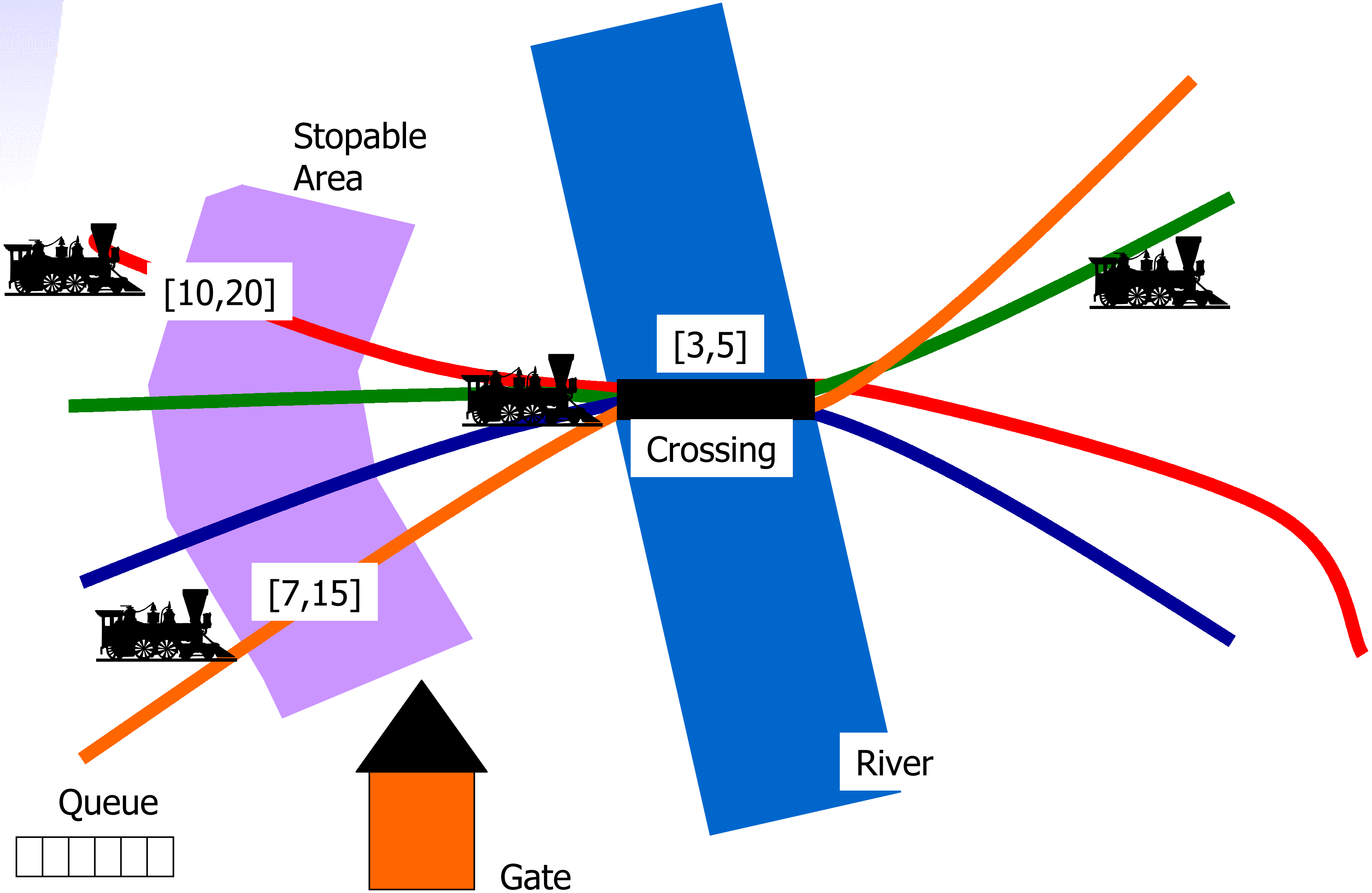
Example transitions

(*l1*, *m1*,………, x=2, y=3.5, i=3,…..) $\xrightarrow{\text{tau}}$ (*l2*,*m2*,……..,x=0, y=3.5, i=7,…..)

$\cancel{\xrightarrow{0.2}}$ (*l1*,*m1*,………,x=2.2, y=3.7, I=3,…..)

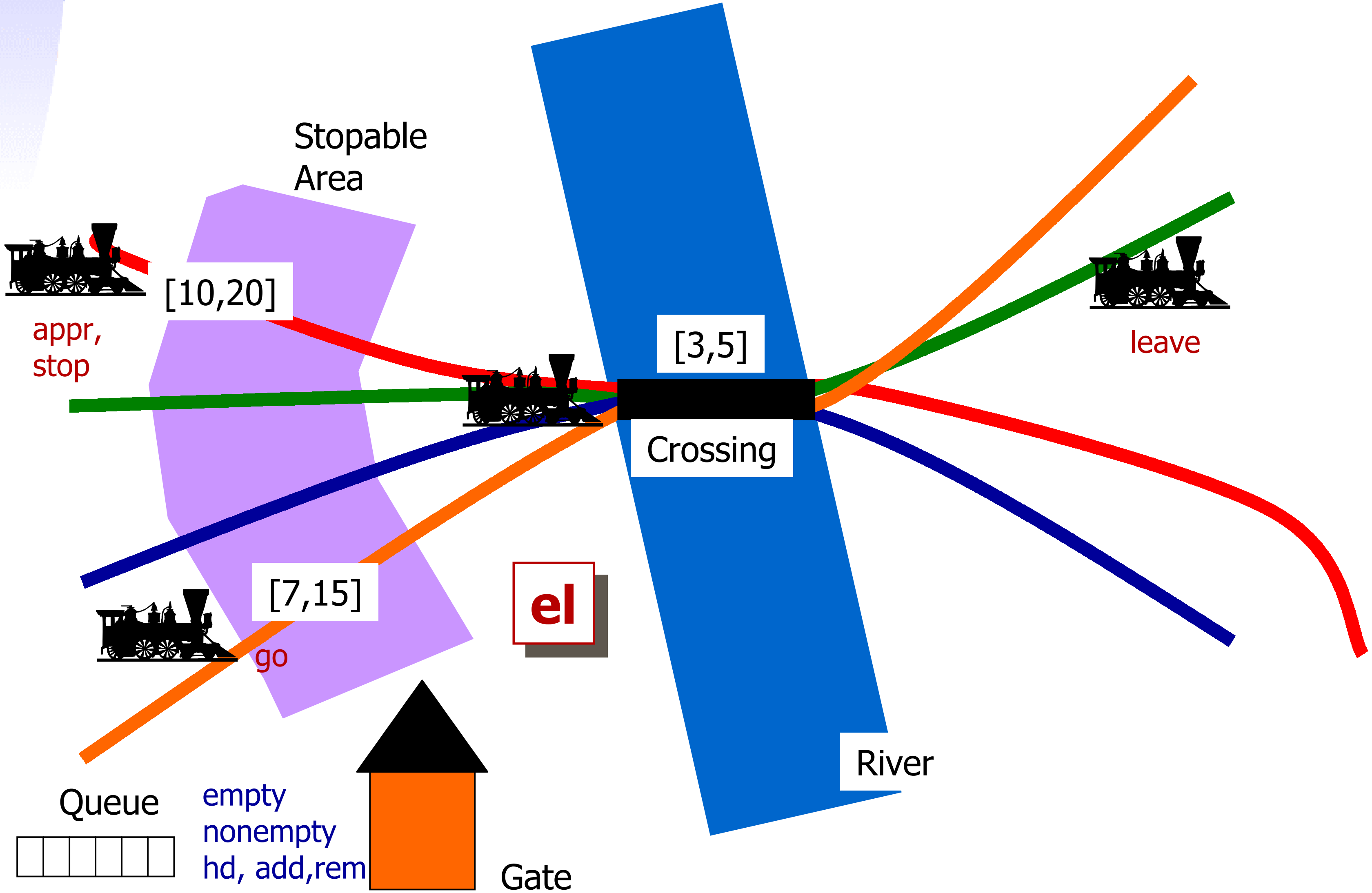
If **a** URGENT CHANNEL

Train Crossing



Train Crossing

Communication via channels and shared variable.



UCb

UPPAAL 3.2 (and 3.3 & 4.0)

Released October 01

www.uppaal.com

- Graphical User Interface
  - XML based file format
  - Better syntax-error indicataion
  - Drop-and-drag for transitions
  - Changed menu

27254 lines of JAVA code

- Verification Engine
  - Restructured (increased flexibility)
  - Normalization-bug fixed
  - More freedom in combing optimization options
  - Deadlock checking
  - Support for more general properties (E[]p, A<>p, p→q)

22081 lines of C++ code

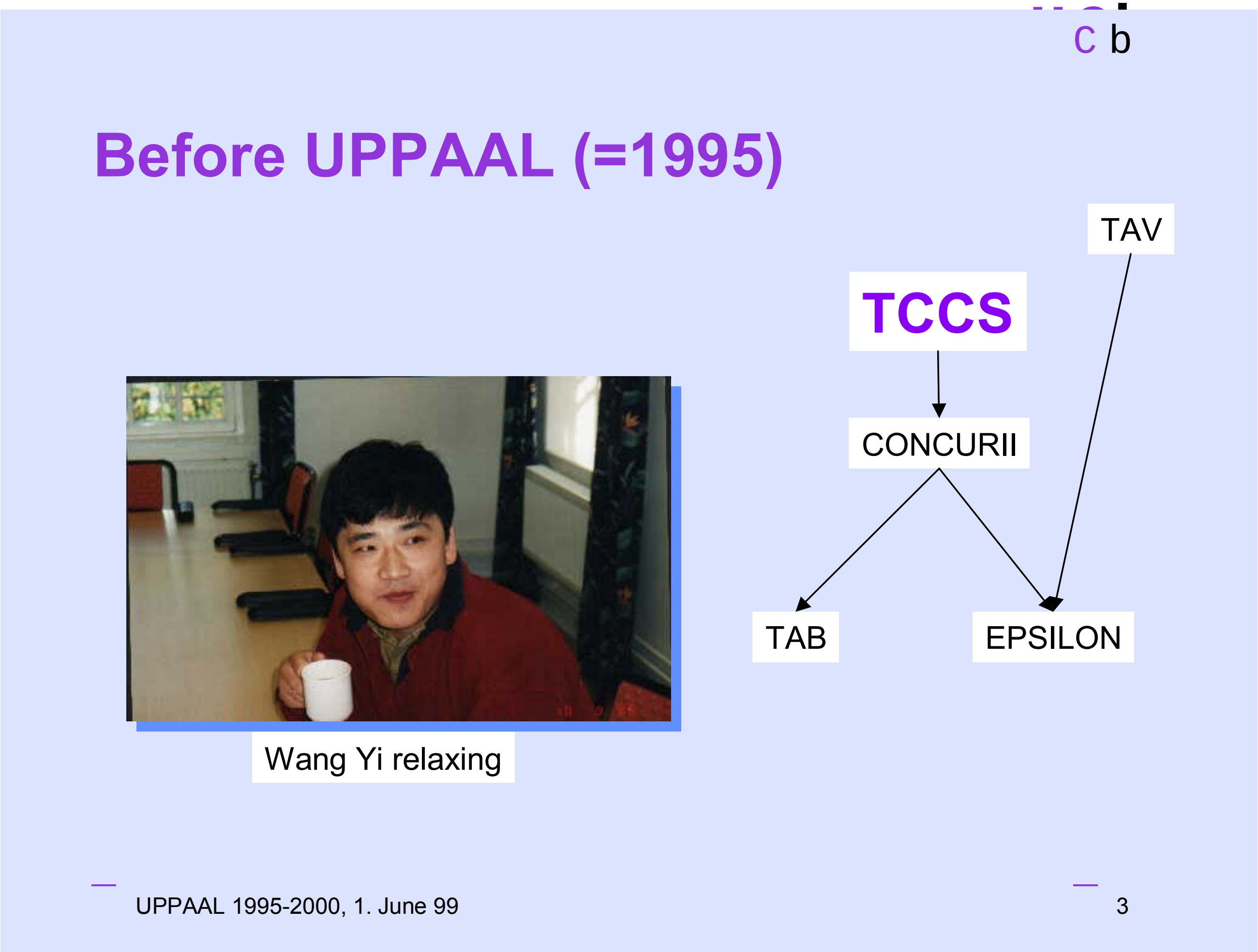
Case Studies: Protocols

- Philips Audio Protocol [HS'95, CAV'95, RTSS'95, CAV'96]
- Collision-Avoidance Protocol [SPIN'95]
- Bounded Retransmission Protocol [TACAS'97]
- Bang & Olufsen Audio/Video Protocol [RTSS'97]
- TDMA Protocol [PRFTS'97]
- Lip-Synchronization Protocol [FMICS'97]
- Multimedia Streams [DSVIS'98]
- ATM ABR Protocol [CAV'99]
- ABB Fieldbus Protocol [ECRTS'2k]
- IEEE 1394 Firewire Root Contention (2000)

Case-Studies: Controllers

- Gearbox Controller [TACAS'98]
- Bang & Olufsen Power Controller [RTPS'99,FTRTFT'2k]
- SIDMAR Steel Production Plant [RTCSA'99, DSVV'2k]
- Real-Time RCX Control-Programs [ECRTS'2k]
- Experimental Batch Plant (2000)
- RCX Production Cell (2000)

The Early History of UPPAAL



Before UPPAAL (=1995)

TAV

TCCS

CONCURII

TAB

EPSILON

Wang Yi relaxing

UPPAAL 1995-2000, 1. June 99

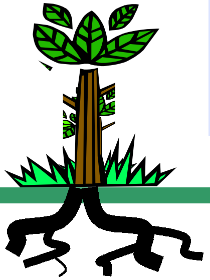
3

Sävja spring 1995

Mia at Dalaresan 20

UPPAAL 1995-2000, 1. June 99

4



Are we still in Denmark?

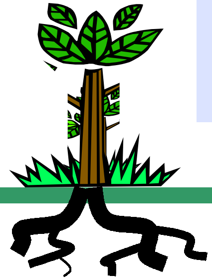
Mia in the local football club

No !!!!

The Larsen family searching for beer

UPPAAL 1995-2000, 1. June 99

6

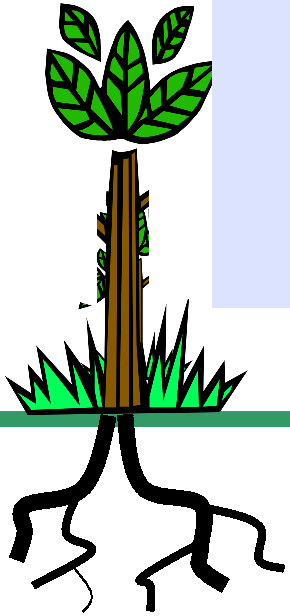


First official UPPAAL presentation

Wang Yi, TACAS, Aarhus, April 1995

UPPAAL 1995-2000, 1. June 99

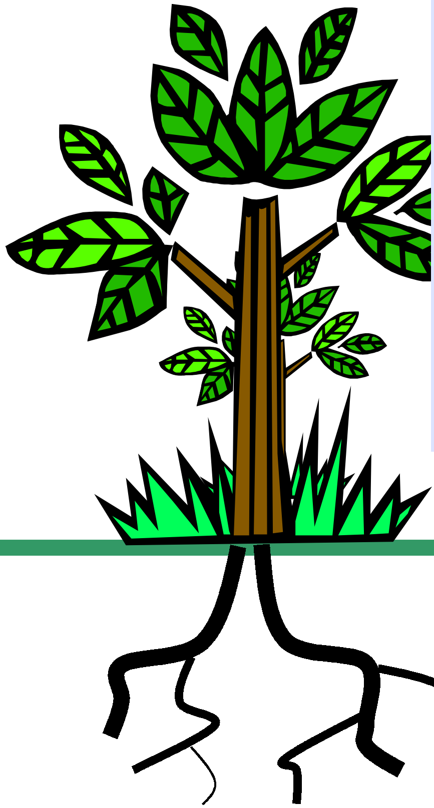
7

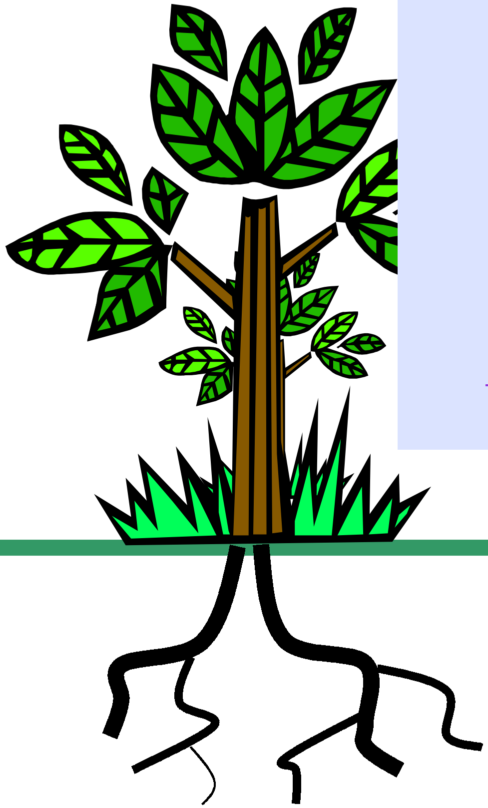


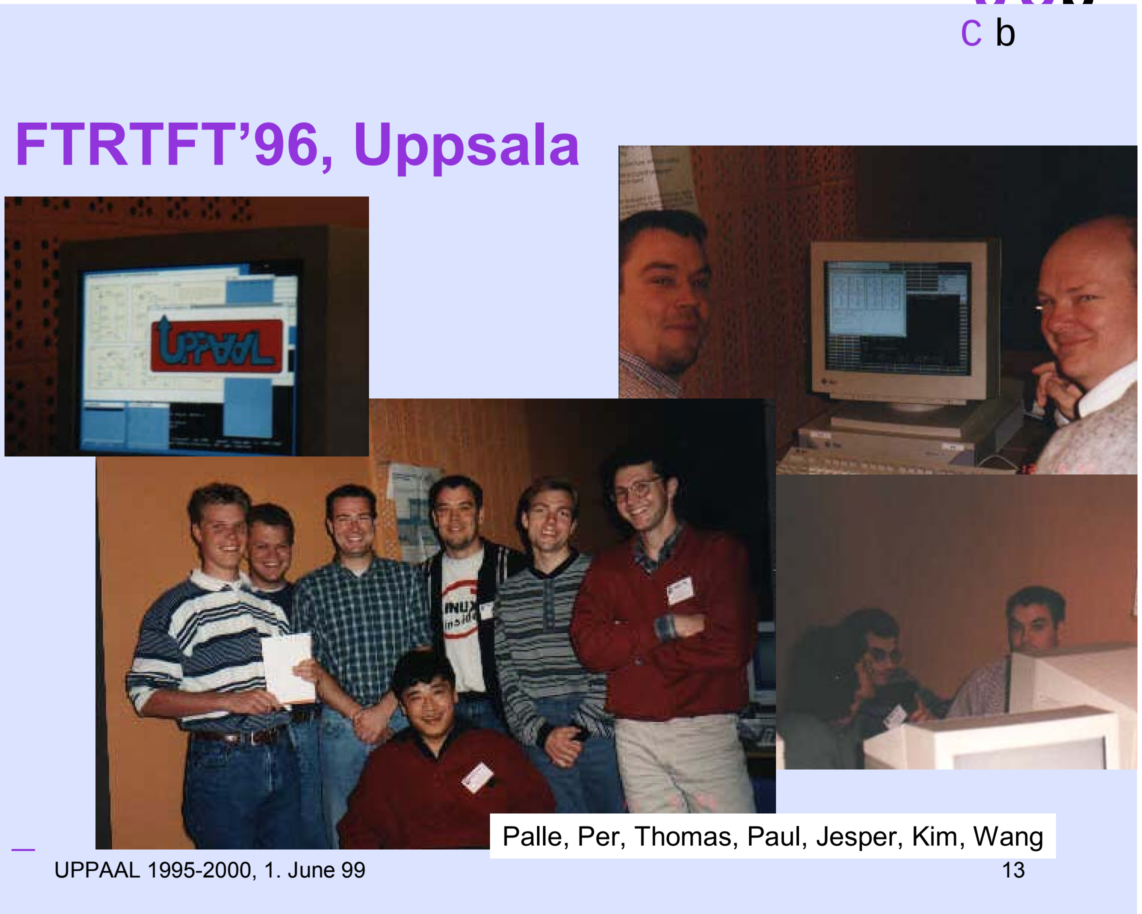
CAV’96, New York

David Griffeon and some Scandinavian friends.

UPPAAL 1995-2000, 1. June 99



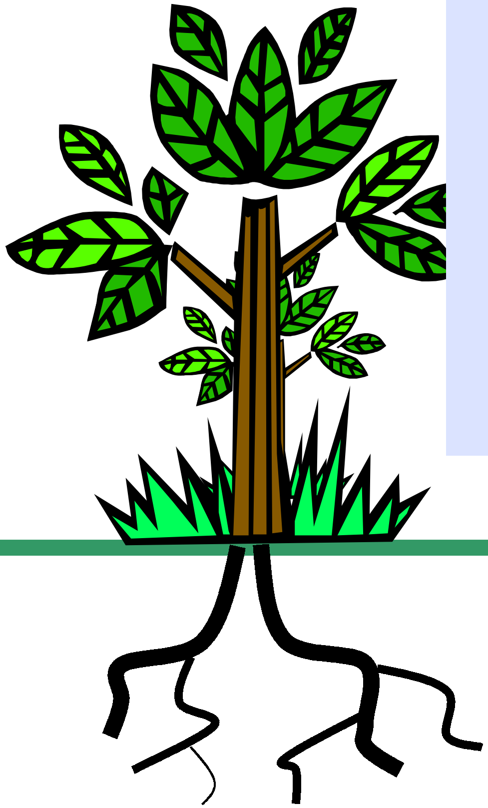


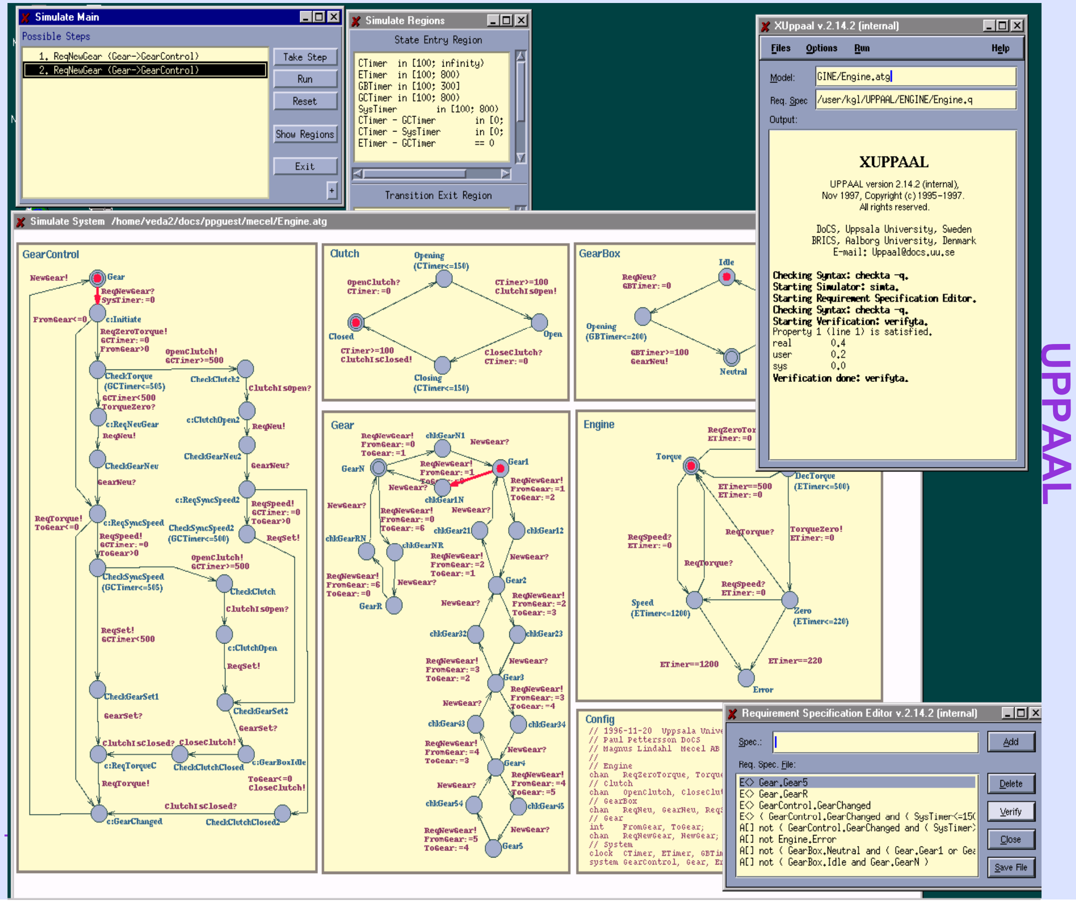


FTRTFT'96, Uppsala

Palle, Per, Thomas, Paul, Jesper, Kim, Wang

UPPAAL 1995-2000, 1. June 99 13





```
1. ReqNewGear (Gear->GearControl)
2. ReqNewGear (Gear->GearControl)
```

```
CTimer  in [100; infinity)
ETimer  in [100; 800)
GBTimer in [100; 300]
GCTimer in [100; 800)
SysTimer        in [100; 800)
CTimer - GCTimer        in [0;
CTimer - SysTimer       in [0;
ETimer - GCTimer        == 0
```

```
// 1996-11-20  Uppsala Unive
// Paul Pettersson DoCS
// Magnus Lindahl  Mecel AB
//
// Engine
chan   ReqZeroTorque, Torque
// Clutch
chan   OpenClutch, CloseClut
// GearBox
chan   ReqNeu, GearNeu, ReqS
// Gear
int    FromGear, ToGear;
chan   ReqNewGear, NewGear;
// System
clock  CTimer, ETimer, GBTim
system GearControl, Gear, Er
```

Model: GINE/Engine.atg

Req. Spec /user/kgl/UPPAAL/ENGINE/Engine.q

Output:

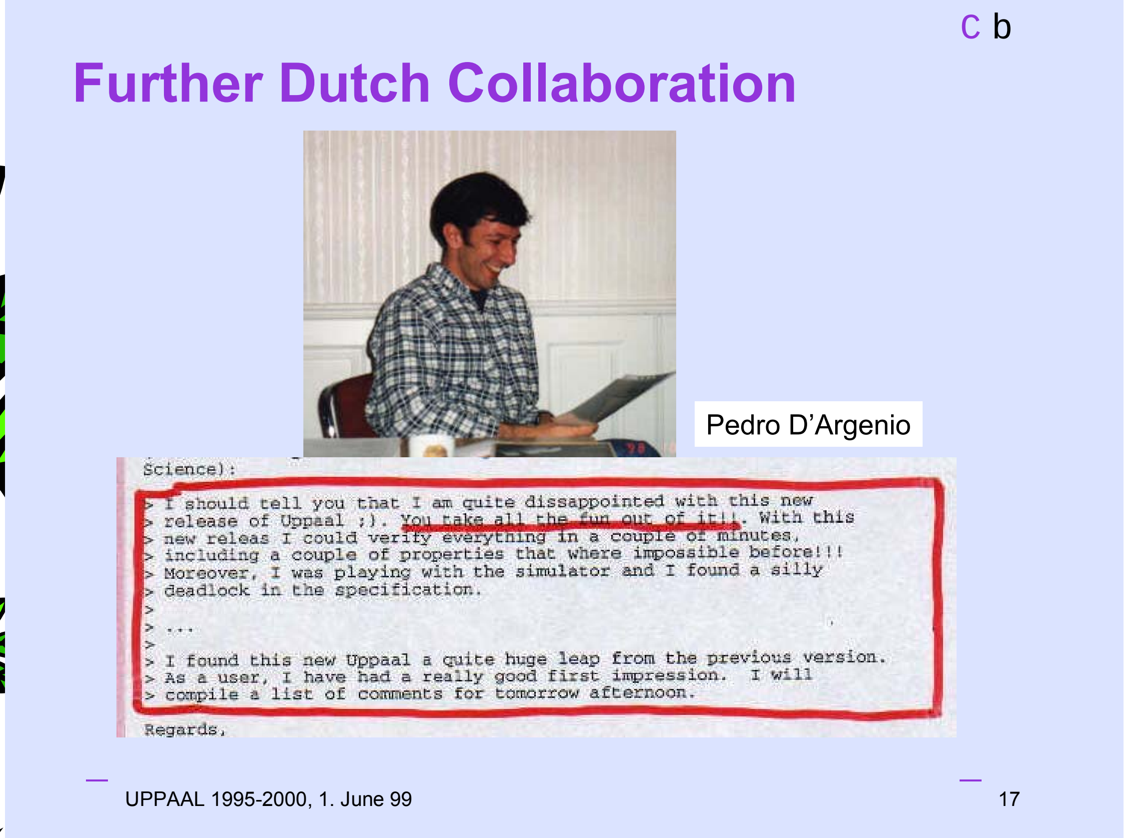
```
XUPPAAL

UPPAAL version 2.14.2 (internal),
Nov 1997, Copyright (c) 1995-1997.
All rights reserved.

DoCS, Uppsala University, Sweden
BRICS, Aalborg University, Denmark
E-mail: Uppaal@docs.uu.se

Checking Syntax: checkta -q.
Starting Simulator: simta.
Starting Requirement Specification Editor.
Checking Syntax: checkta -q.
Starting Verification: verifyta.
Property 1 (line 1) is satisfied.
real      0.4
user      0.2
sys       0.0
Verification done: verifyta.
```

Spec.:

Req. Spec. File:

```
E<> Gear.Gear5
E<> Gear.GearR
E<> GearControl.GearChanged
E<> ( GearControl.GearChanged and ( SysTimer<=15(
A[] not ( GearControl.GearChanged and ( SysTimer>
A[] not Engine.Error
A[] not ( GearBox.Neutral and ( Gear.Gear1 or Gea
A[] not ( GearBox.Idle and Gear.GearN )
```

UPPAAL



C b

Further Dutch Collaboration

Pedro D'Argenio

```
Science):

> I should tell you that I am quite dissappointed with this new
> release of Uppaal ;). You take all the fun out of it!!. With this
> new releas I could verify everything in a couple of minutes,
> including a couple of properties that where impossible before!!!
> Moreover, I was playing with the simulator and I found a silly
> deadlock in the specification.
>
> ...
>
>
> I found this new Uppaal a quite huge leap from the previous version.
> As a user, I have had a really good first impression.  I will
> compile a list of comments for tomorrow afternoon.

Regards,
```

UPPAAL 1995-2000, 1. June 99

17

RTSS'97

Fairmont Hotel, San Francisco

1st RTSS'97 talk, Kluas Havelund

Breakfast

UPPAAL 1995-2000, 1. June 99

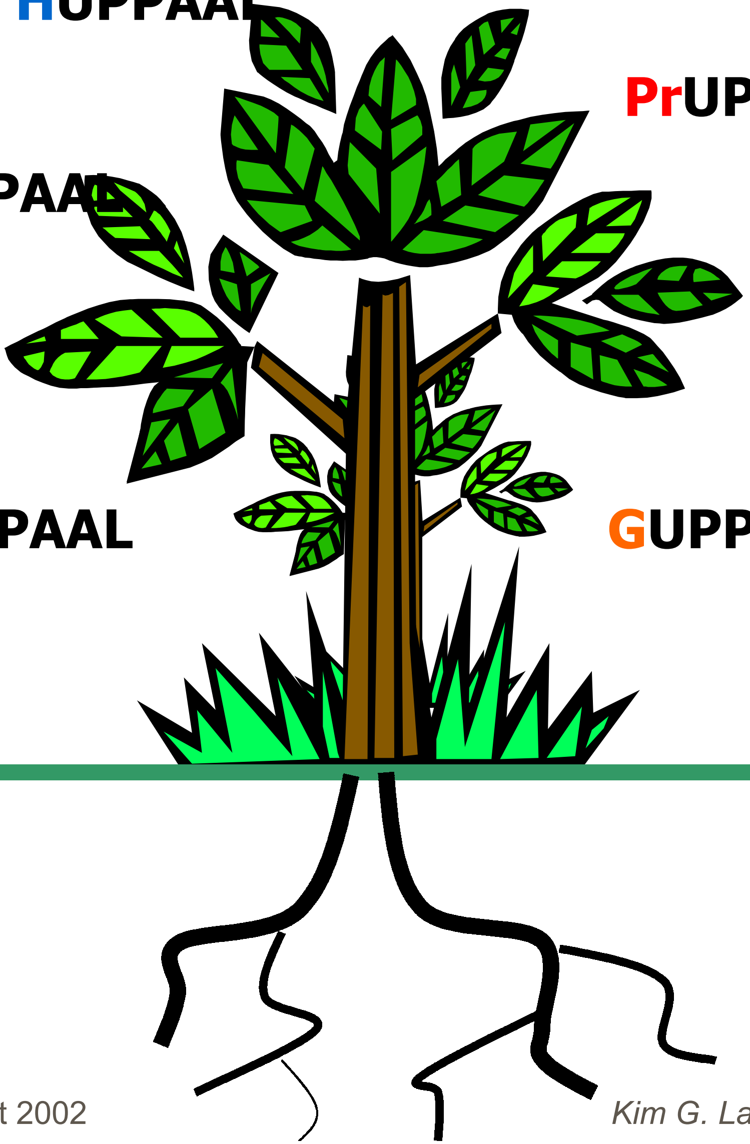
21

C b

1st UPPAAL workshop,1998

UPPAAL 1995-2000, 1. June 99

24



PUPPAAL

HUPPAAL

PrUPPAAL

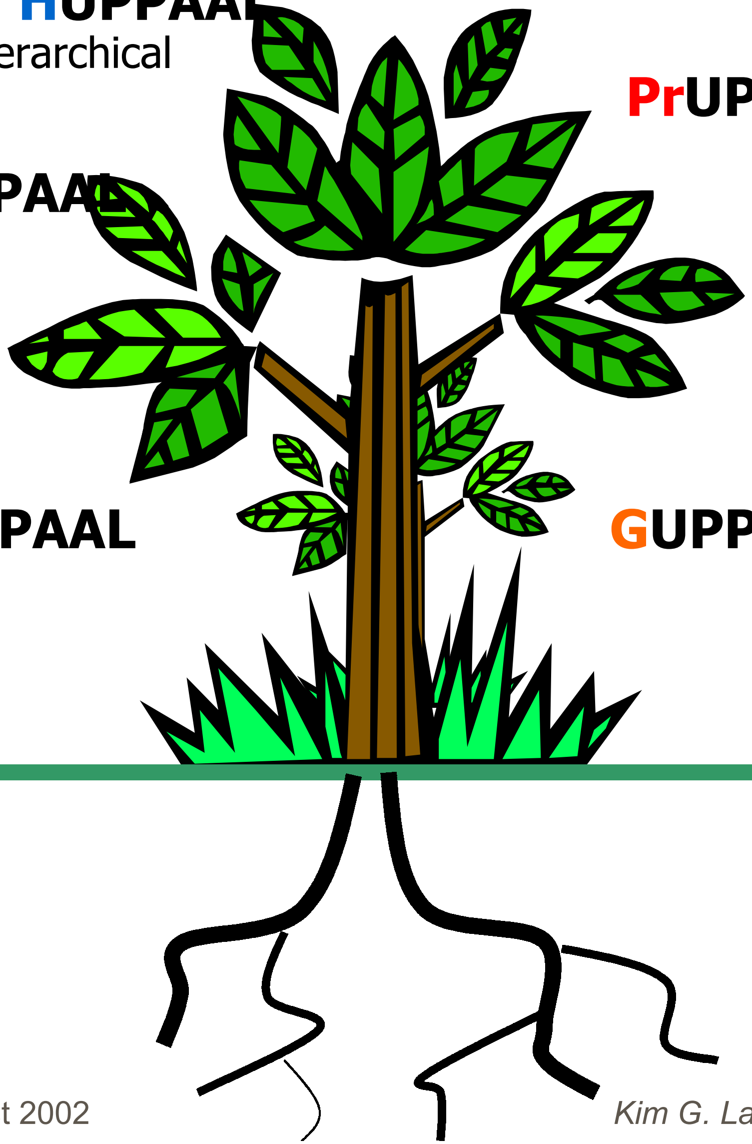
HyUPPAAL

DUPAAL

CUPPAAL

TUPPAAL

GUPPAAL



PUPPAAL
Parameterized

HUPPAAL
Hierarchical

PrUPPAAL
Probabilistic

HyUPPAAL
Hybrid

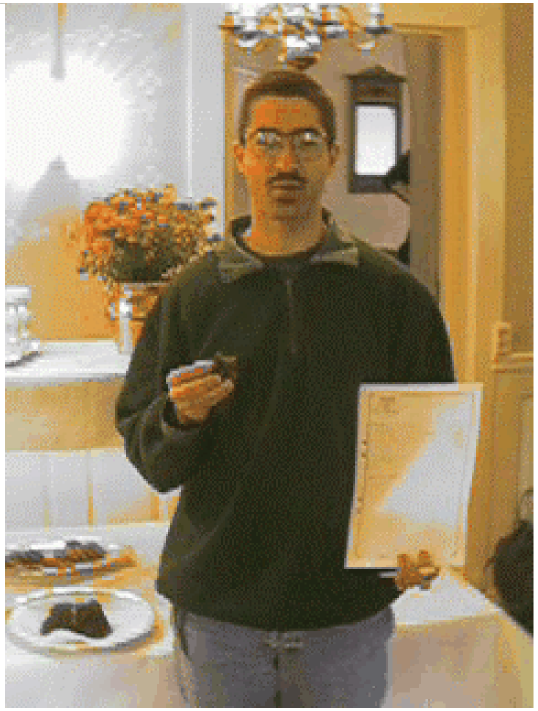
DUPAAL
Distributed

CUPPAAL
Cost-optimizing

TUPPAAL
Testing

GUPPAAL
Guiding

Gerd, Alexandre, Johan



The Main Implementers

Summary & Lessons

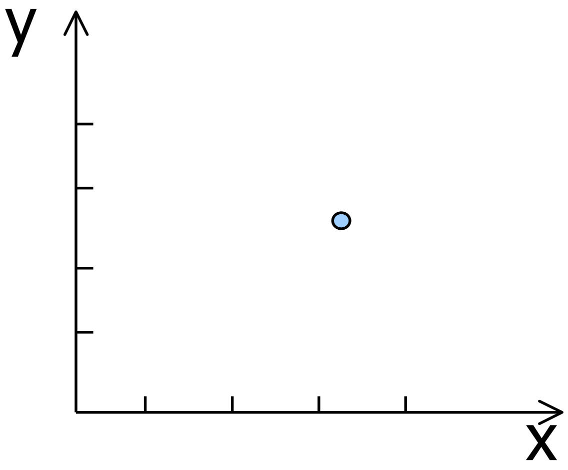
- Driven by case-studies
    - Modelling capabilities
    - Improvement of performance
- Complete rewriting of GUI & Verifier 2-3 times (involving several teams of student programmers).
- Large involvement of external researchers on case-studies & branches
- Kernel UPPAAL staff *very stable.*
- Common CVS archieve for all variants
- Frequent “exchange” visits.

Implementation Secrets

Zones

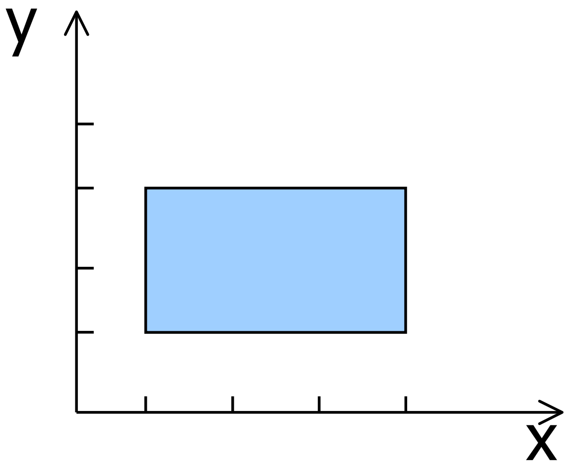
From infinite to finite

State
(n, x=3.2, y=2.5)

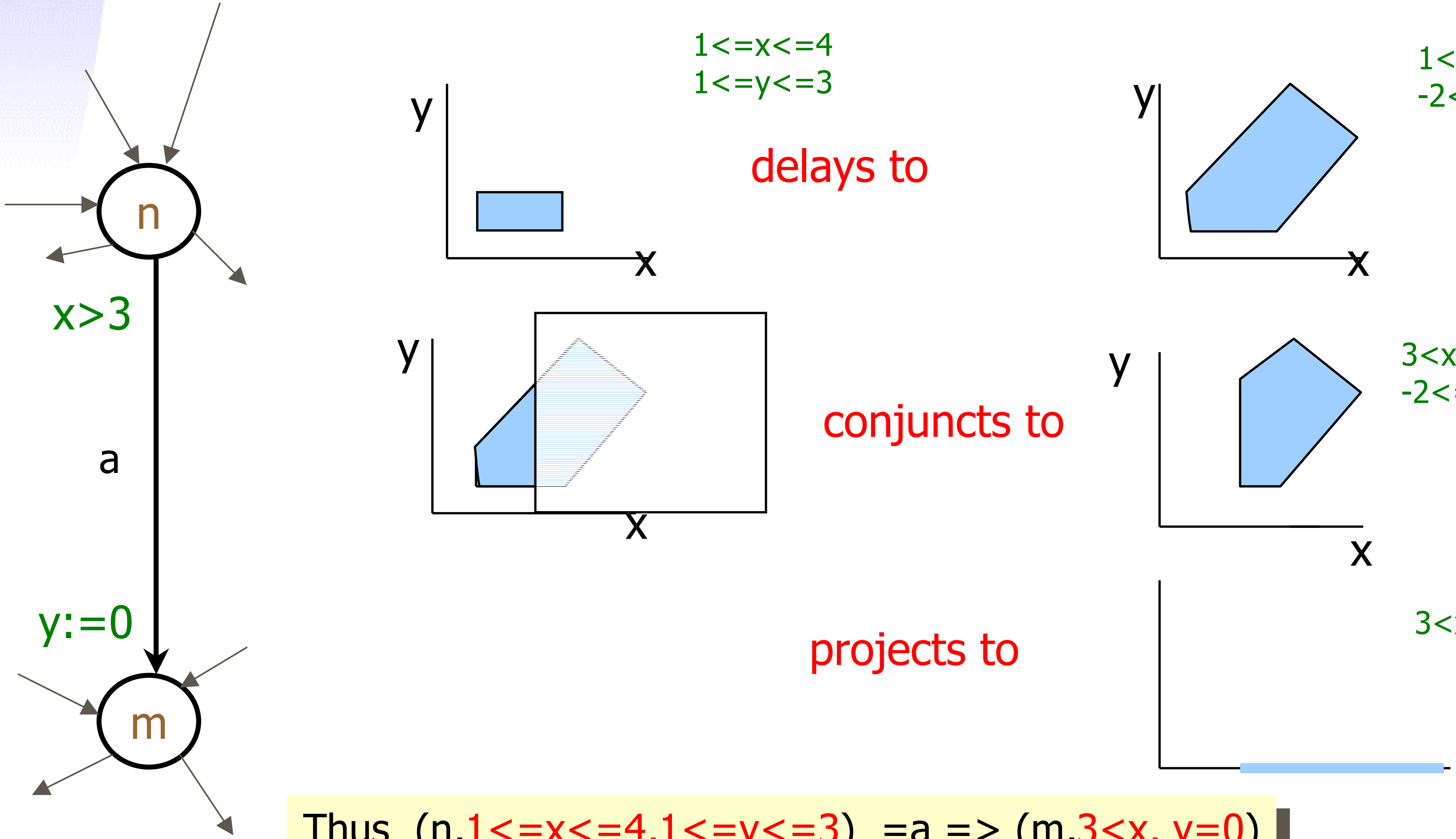


Symbolic state (set)
(n, $1 \leq x \leq 4, 1 \leq y \leq 3$)

Zone:
conjunction of
x-y<=n, x<=>n



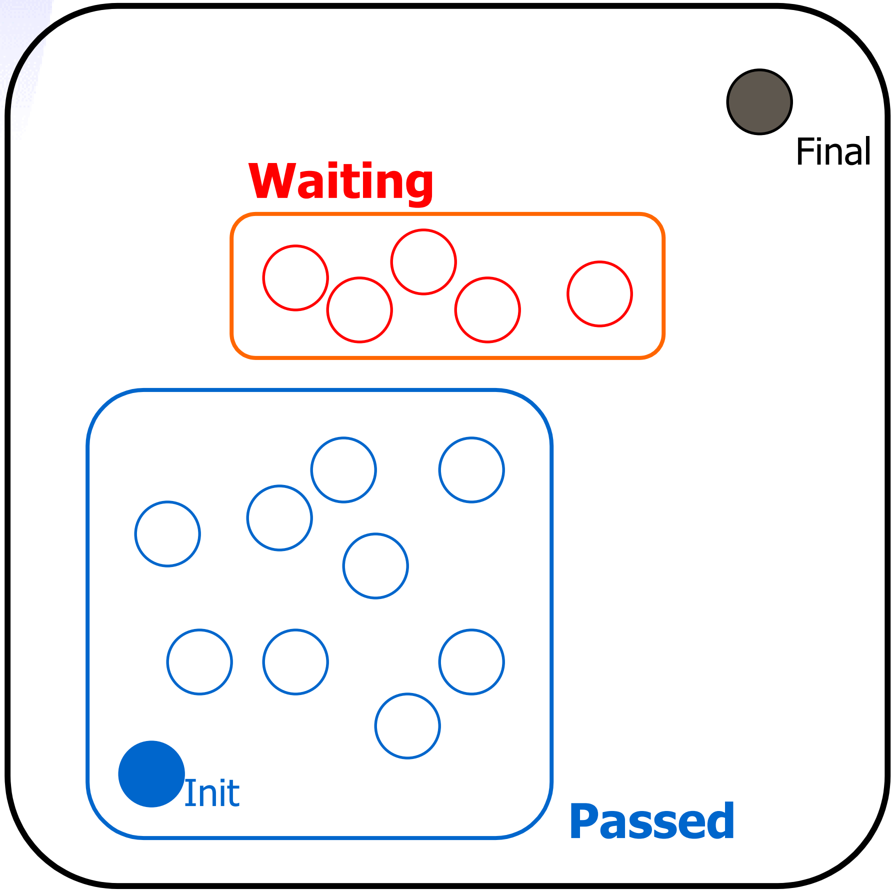
Symbolic Transitions



Thus (n,1<=x<=4,1<=y<=3) =a => (m,3<x, y=0)

Forward Rechability

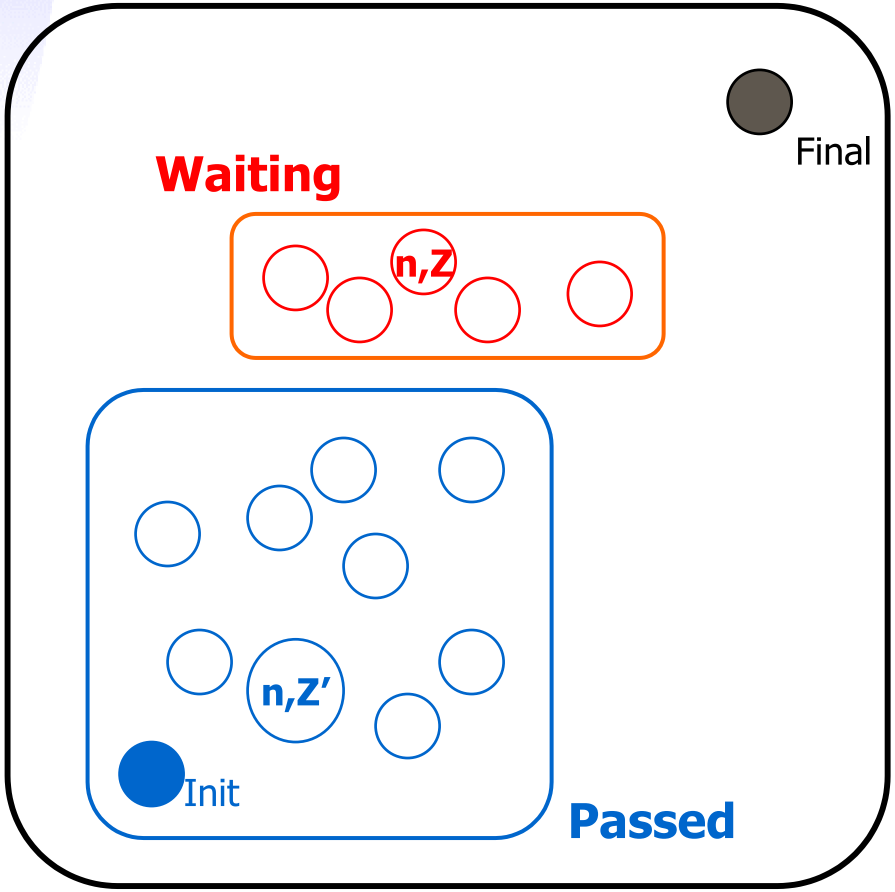
Init -> Final ?



```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT





UNTIL  Waiting = Ø
         or
         Final is in Waiting
```

Forward Rechability

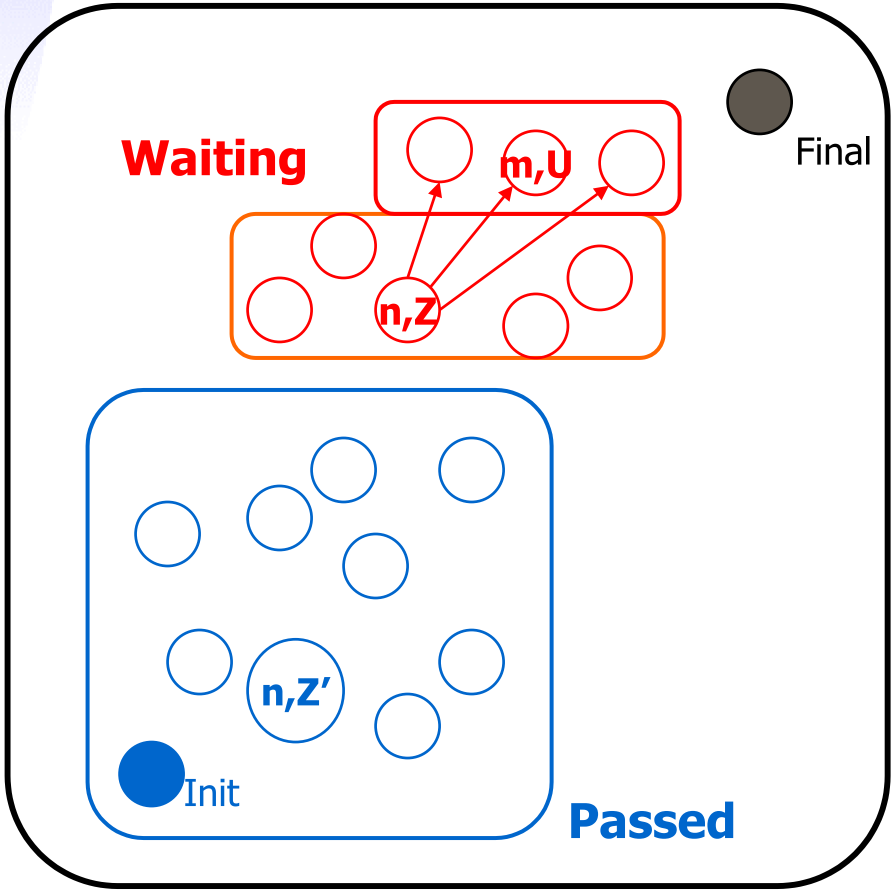
Init -> Final ?



```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
   - pick  (n,Z) in Waiting
   - if for some Z′ ⊇ Z
       (n,Z’) in Passed then  STOP

UNTIL  Waiting = Ø
         or
         Final is in Waiting
```

Forward Rechability

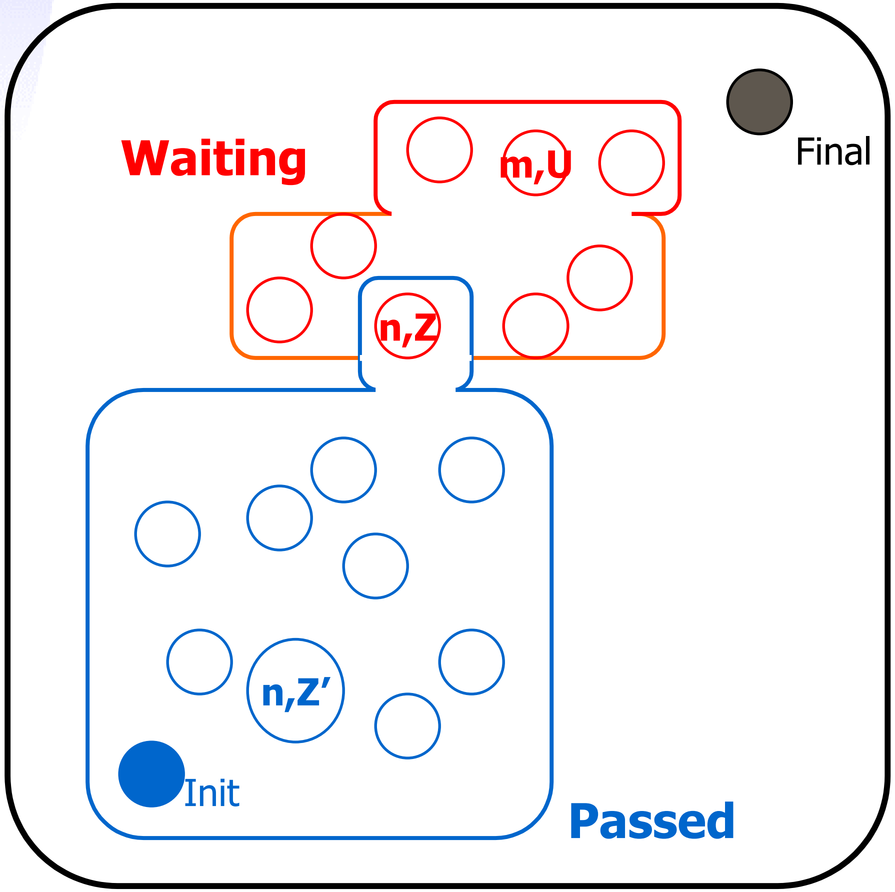
Init -> Final ?



```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
   - pick  (n,Z) in Waiting
   - if for some Z′ ⊇ Z
       (n,Z’) in Passed then  STOP
   - else /explore/ add
           { (m,U) : (n,Z) => (m,U) }
           to Waiting;

UNTIL  Waiting = Ø
           or
           Final is in Waiting
```

Forward Rechability

Init -> Final ?



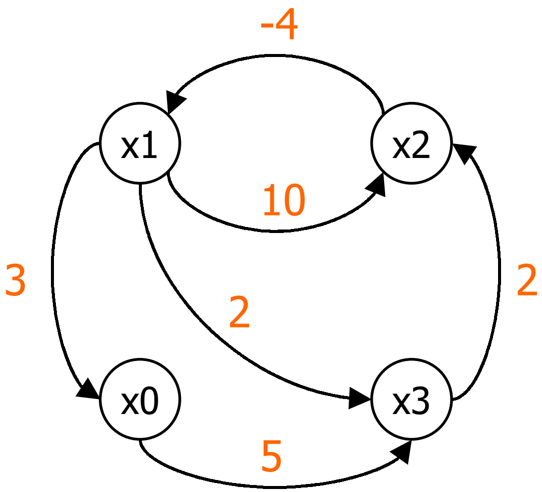
```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick (n,Z) in Waiting
  - if for some Z′ ⊇ Z
      (n,Z′) in Passed then STOP
  - else /explore/ add
        { (m,U) : (n,Z) => (m,U) }
        to Waiting;
        Add (n,Z) to Passed

UNTIL  Waiting = Ø
        or
        Final is in Waiting
```

Canonical Datastructure for Zones

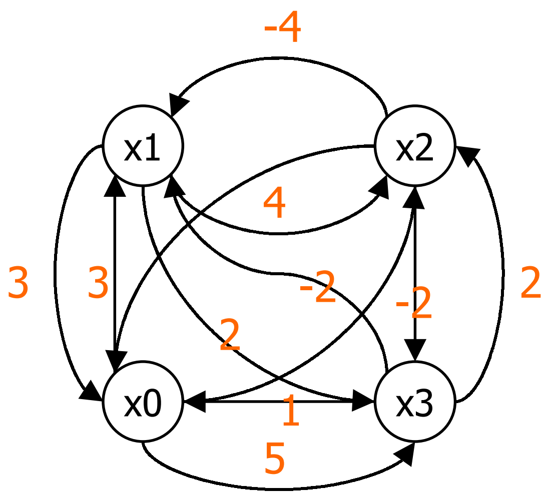
Difference Bounded Matrices

Bellman'58, Dill'89

x1-x2<=4
x2-x1<=10
x3-x1<=2
x2-x3<=2
x0-x1<=3
x3-x0<=5



Shortest Path Closure O(n^3)



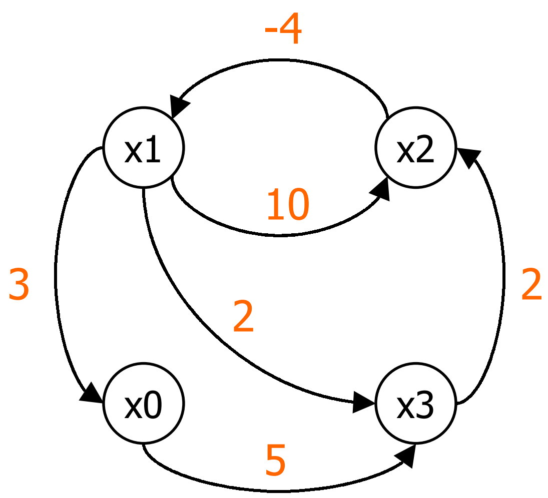
New Canonical Datastructure

UCb

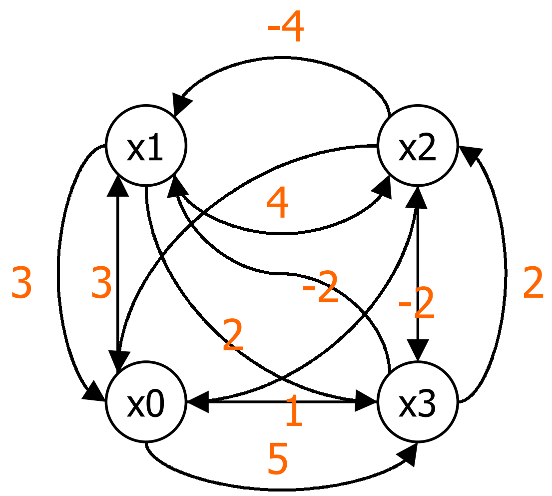
Minimal collection of constraints

RTSS 1997

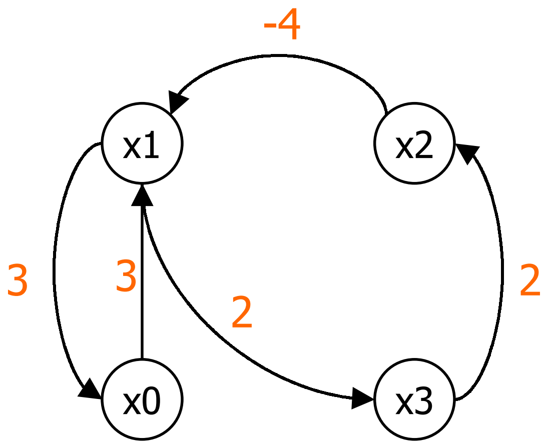
x1-x2<=4
x2-x1<=10
x3-x1<=2
x2-x3<=2
x0-x1<=3
x3-x0<=5



Shortest Path Closure O(n^3)

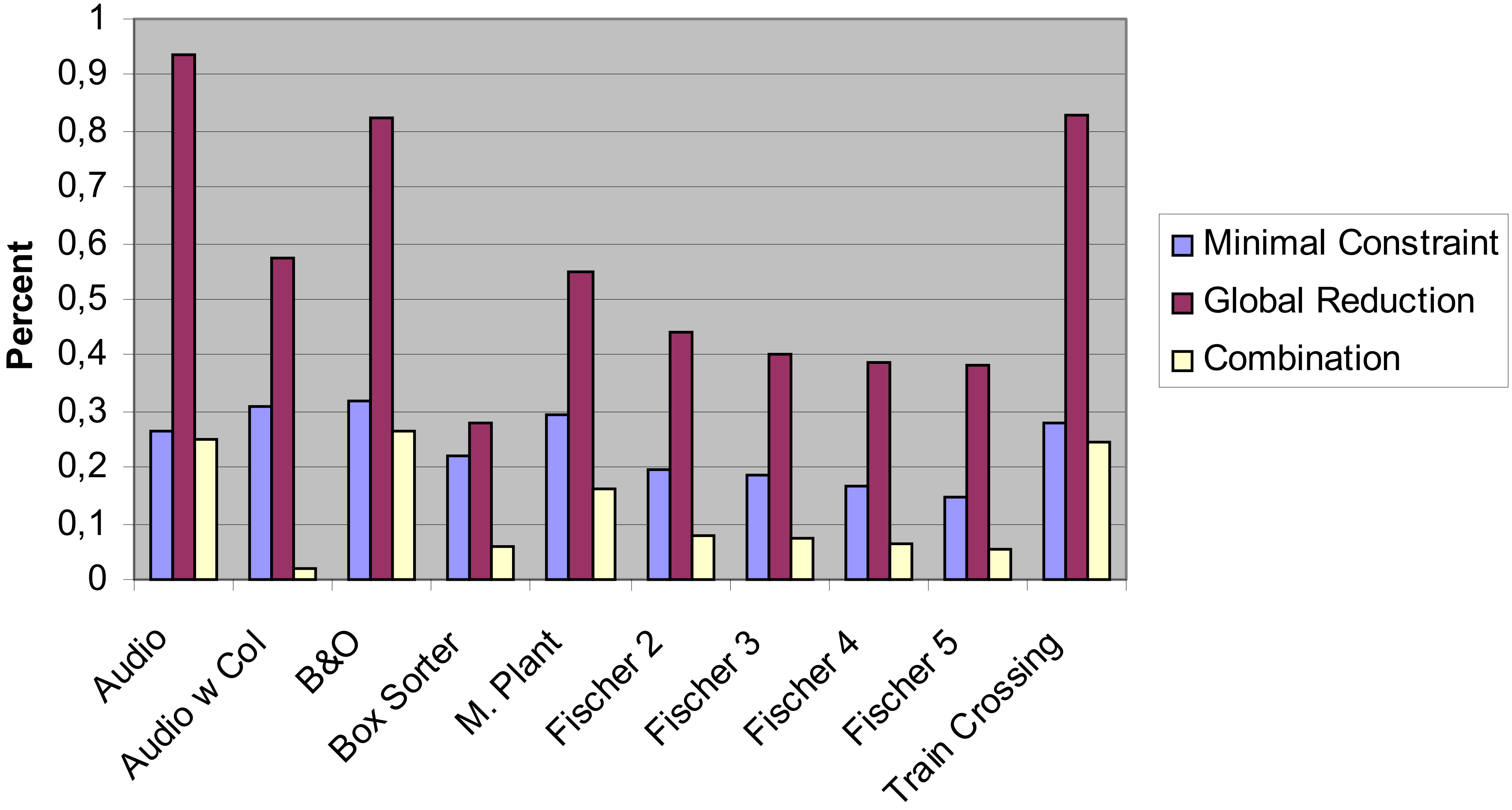


Shortest Path Reduction O(n^3)

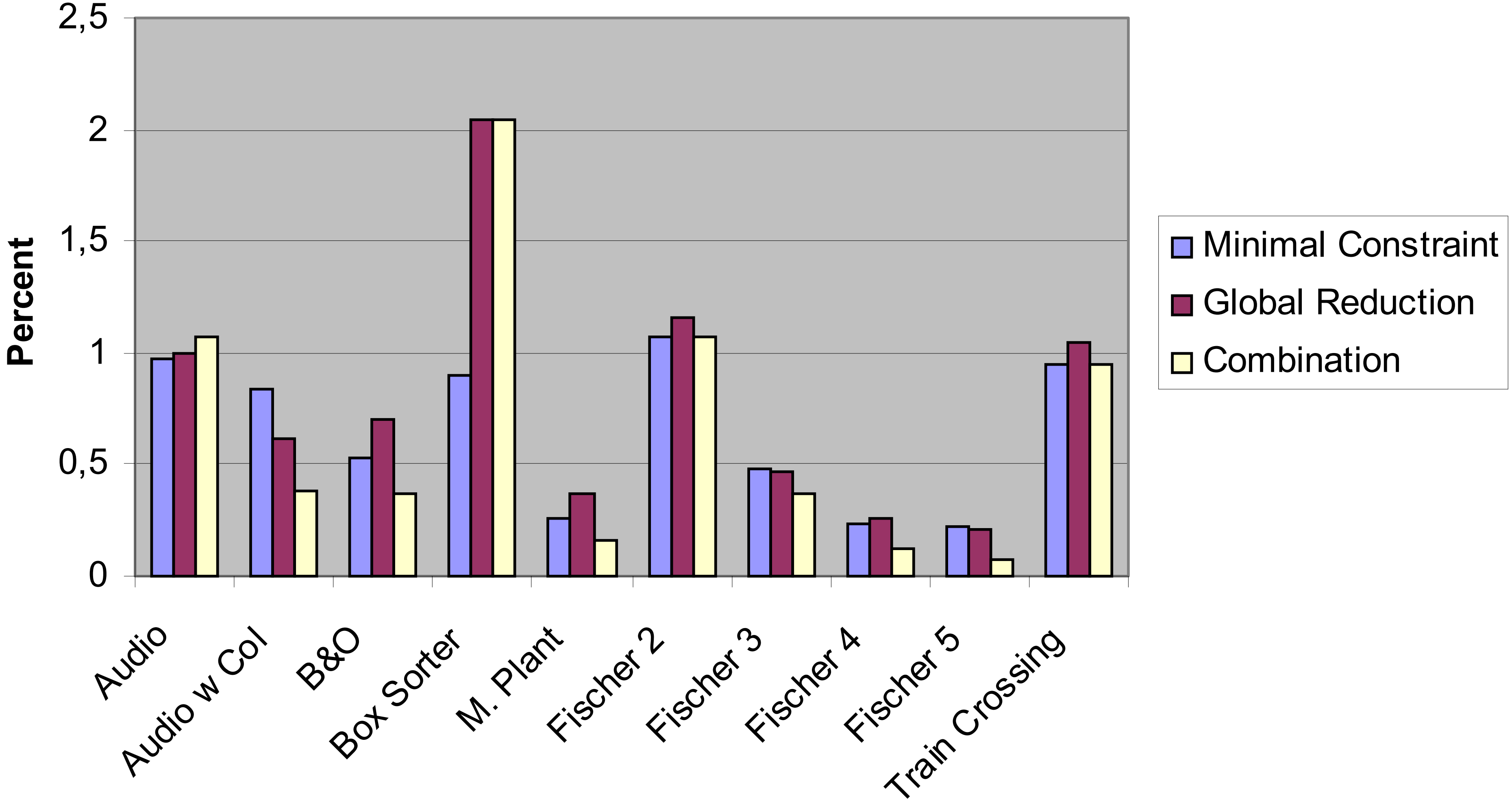


Space worst O(n^2)
practice O(n)

SPACE PERFORMANCE



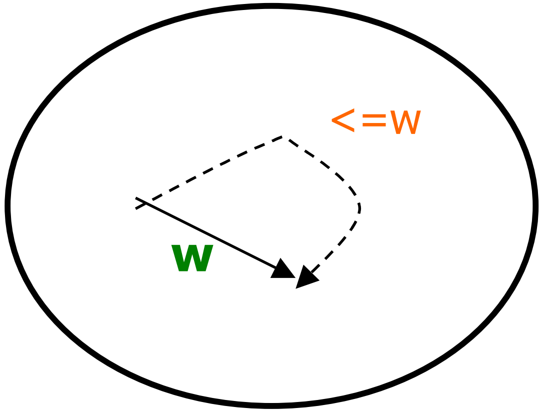
TIME PERFORMANCE



Shortest Path Reduction

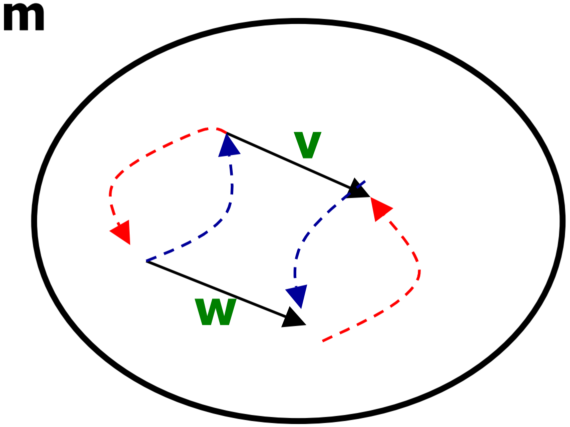
1st attempt

Idea



An edge is REDUNDANT if there exists an alternative path of no greater weight
THUS Remove all redundant edges!

Problem



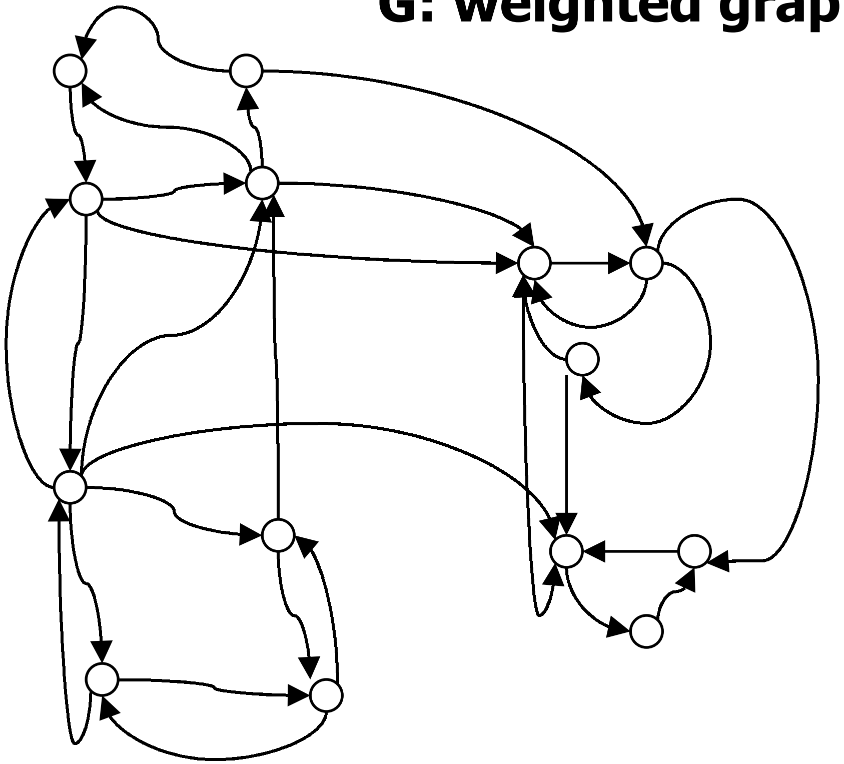
v and **w** are both redundant
Removal of one depends on presence of other.

Observation: If no zero- or negative cycles then SAFE to remove all redundancies.

Shortest Path Reduction

Solution

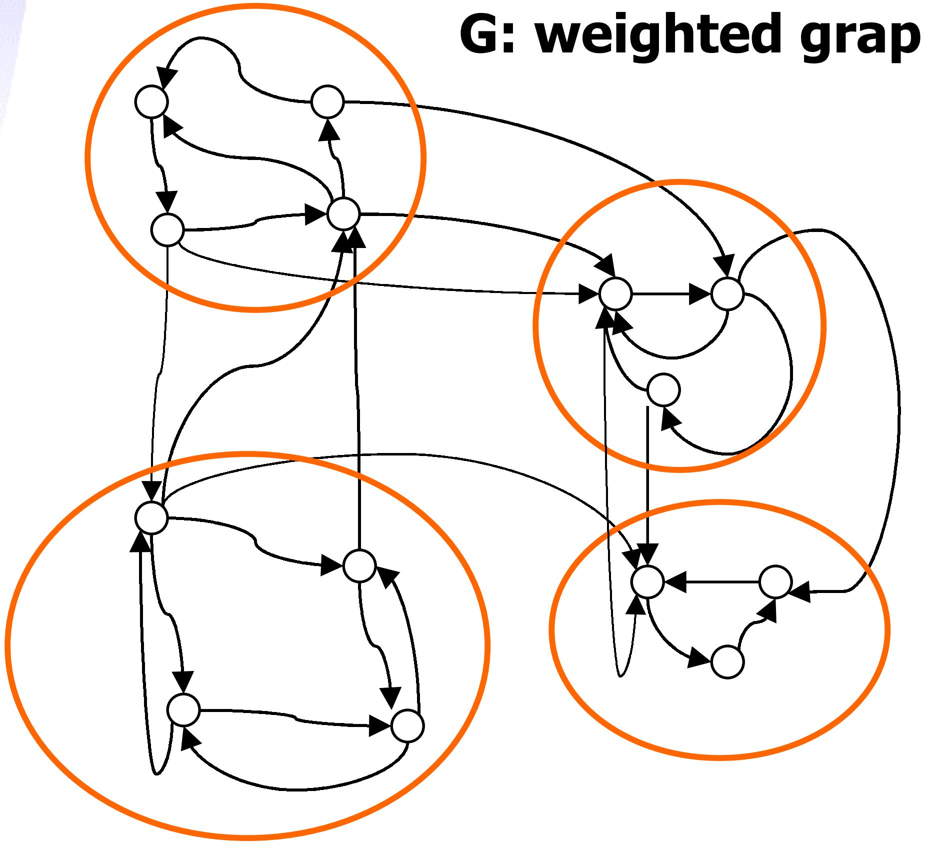
G: weighted graph



Shortest Path Reduction

Solution

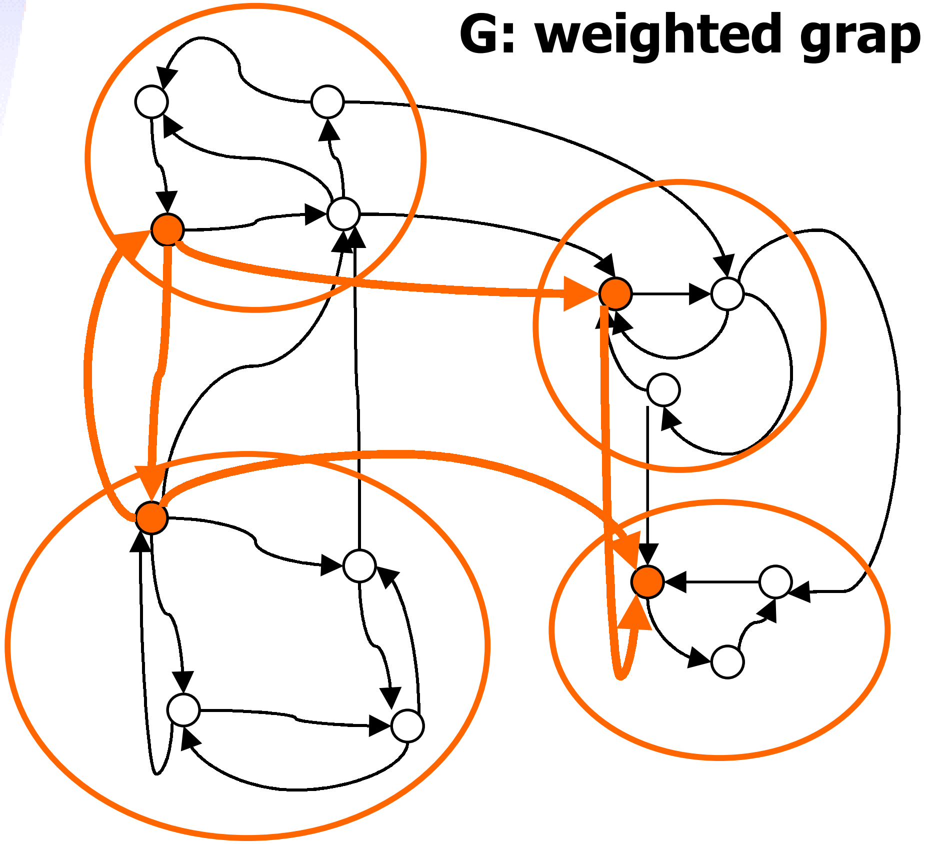
G: weighted graph



1. Equivalence classes based on 0-cycles.

Shortest Path Reduction

Solution



G: weighted graph

1. Equivalence classes based on 0-cycles.

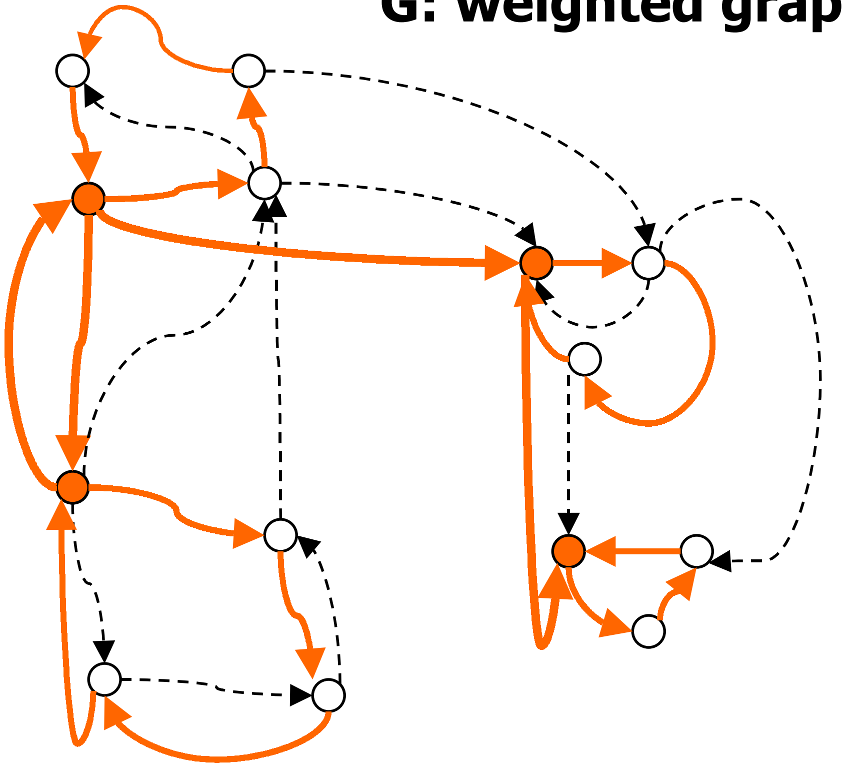
2. Graph based on representatives. Safe to remove redundant edges

UCb

Shortest Path Reduction

Solution

G: weighted graph



1. Equivalence classes based on 0-cycles.

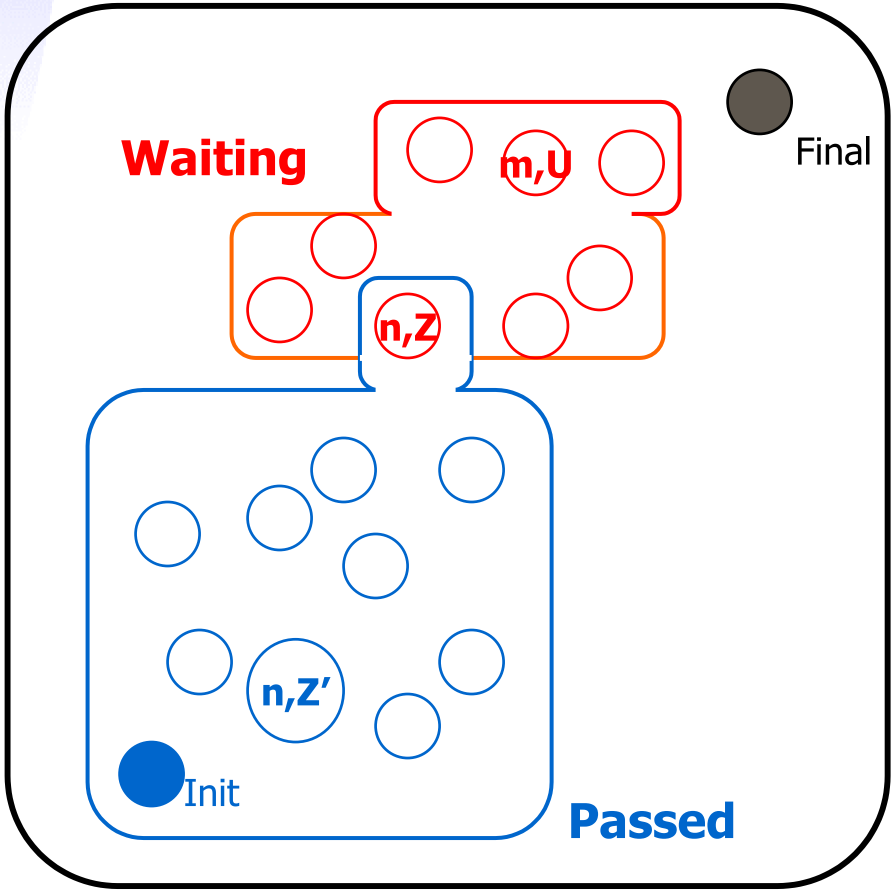
2. Graph based on representatives.
   Safe to remove redundant edges

3. **Shortest Path Reduction**
   =
   One cycle pr. class
   +
   Removal of redundant edges between classes

Canonical given order of clocks

Earlier Termination

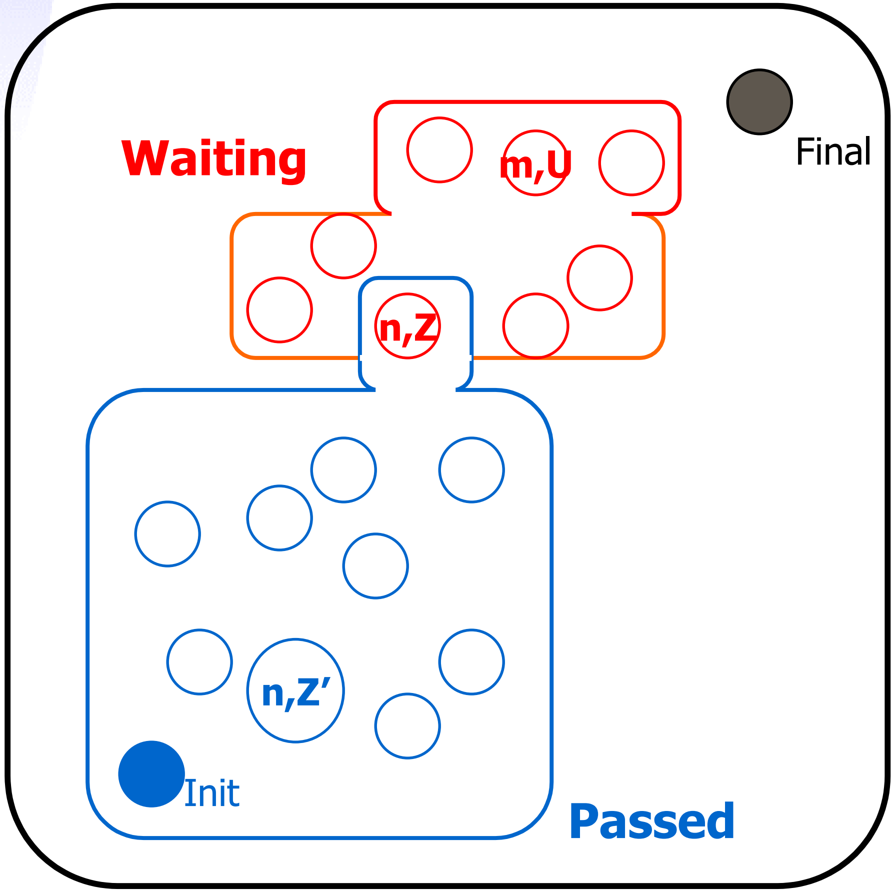
Init -> Final ?



```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some Z′ ⊇ Z
     (n,Z′) in Passed then  STOP
  - else /explore/ add
          { (m,U) : (n,Z) => (m,U) }
          to Waiting;
          Add  (n,Z)  to Passed

UNTIL  Waiting = Ø
          or
          Final is in Waiting
```

Earlier Termination

Init -> Final ?

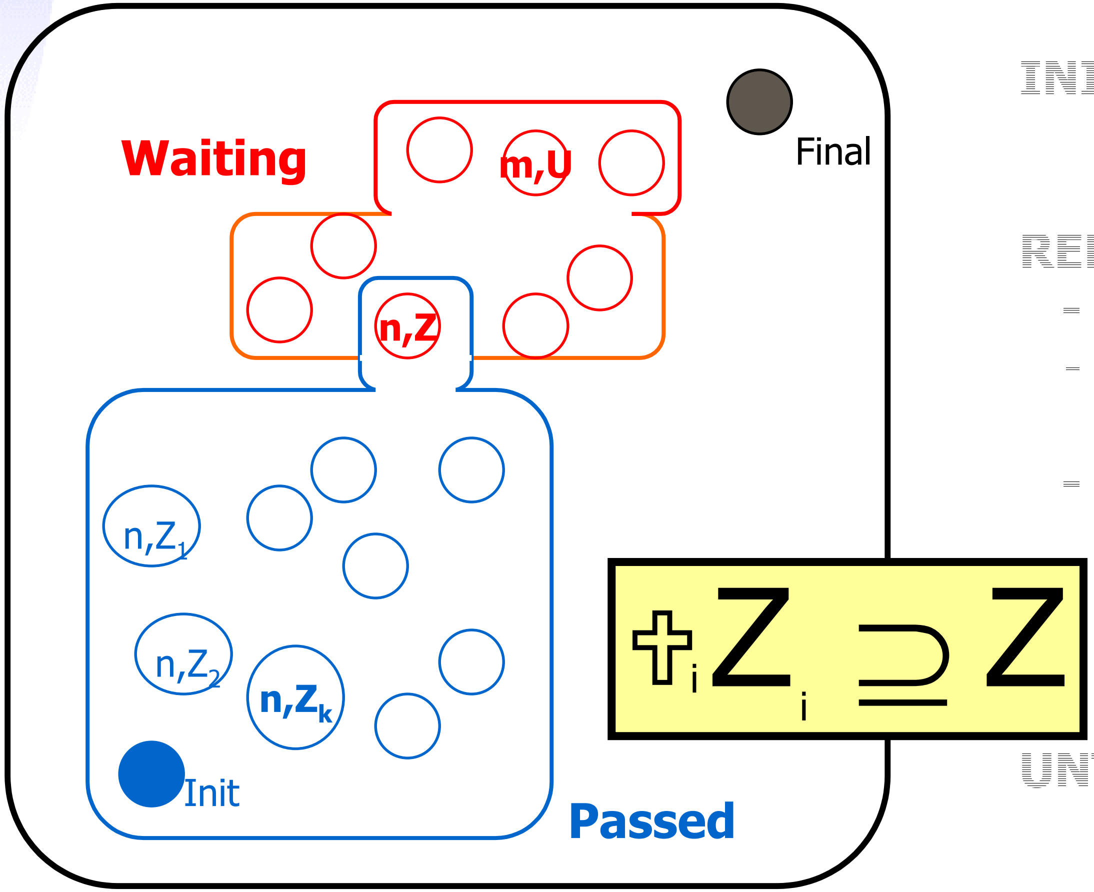


```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some Z' ⊇ Z
      (n,Z') in Passed then  STOP
  - else /explore/ add
           { (m,U) : (n,Z) => (m,U) }
           to Waiting;
           Add  (n,Z)  to Passed

UNTIL  Waiting = Ø
           or
           Final is in Waiting
```

UCb

Earlier Termination

Init -> Final ?



$$\dagger_i Z_i \supseteq Z$$

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some  $Z' \supseteq Z$
     (n,Z') i  Passed then  STOP
  - el     xplore/ add
         { (m,U) : (n,Z) => (m,U) }
         to Waiting;
         Add  (n,Z)  to Passed

UNTIL  Waiting = Ø
         or
         Final is in Waiting
```

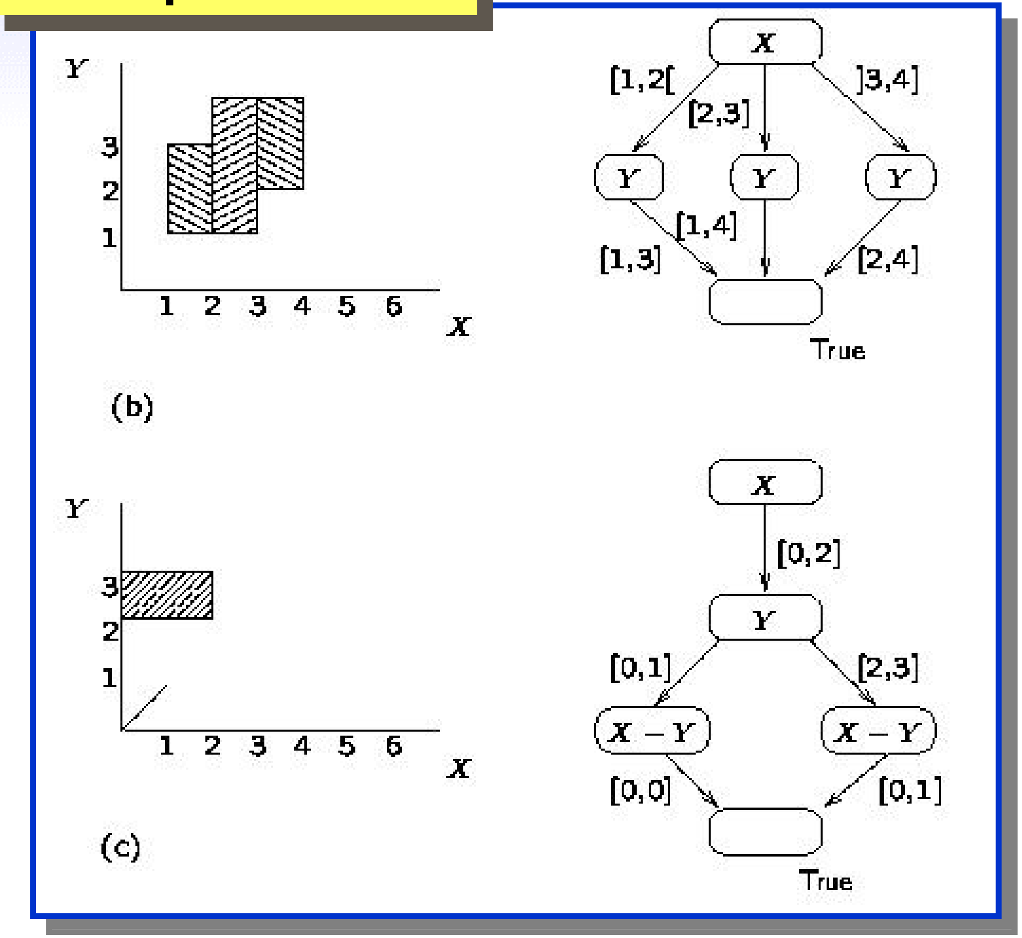
Clock Difference Diagrams

UCb

CAV99

= Binary Decision Diagrams + Difference Bounded Matrices

CDD-representations

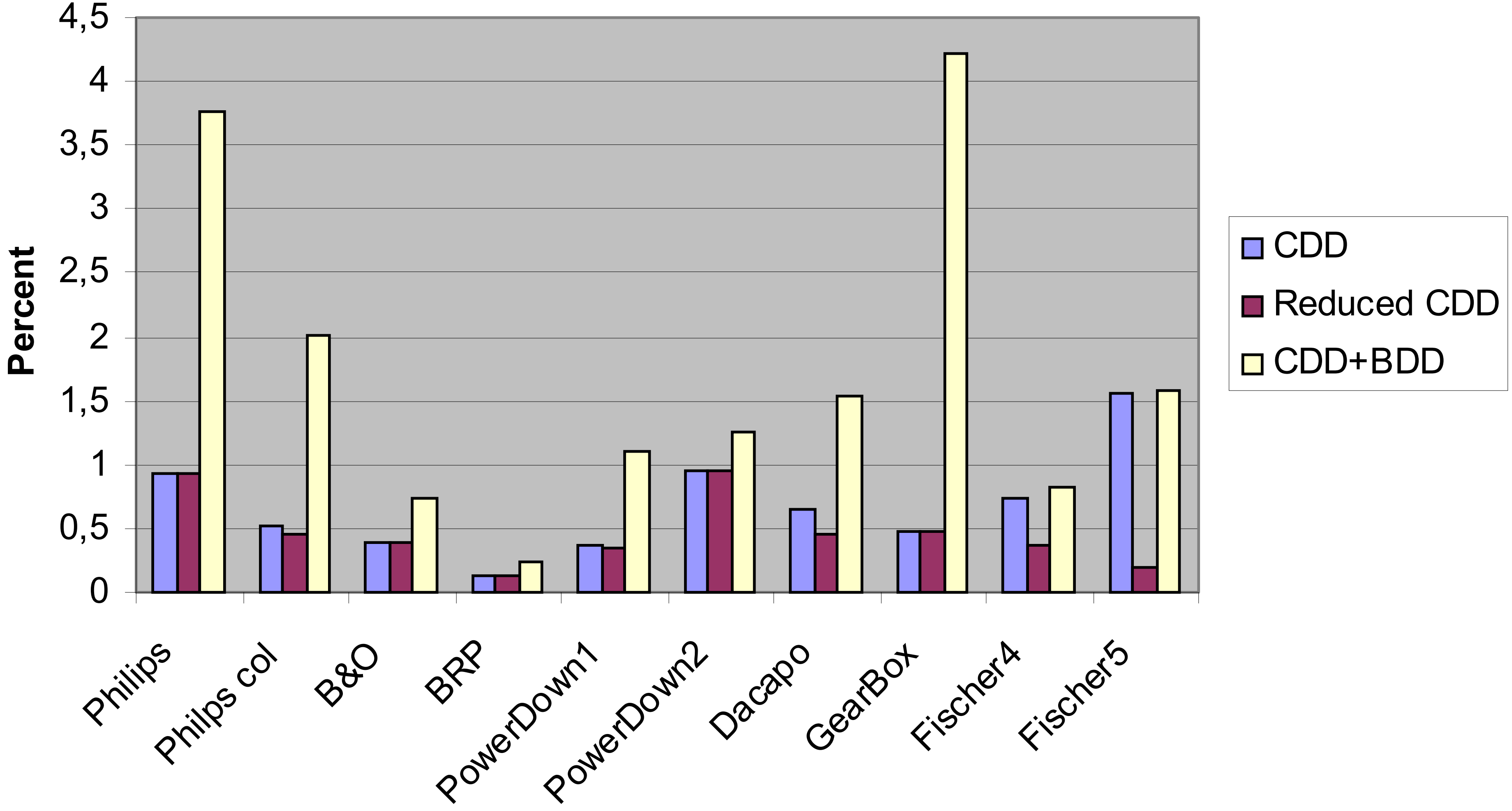


- Nodes labeled with differences
- Maximal sharing of substructures (also across different CDDs)
- Maximal intervals
- Linear-time algorithms for set-theoretic operations.

- NDD's Maler et. al
- DDD's Møller, Lichtenberg

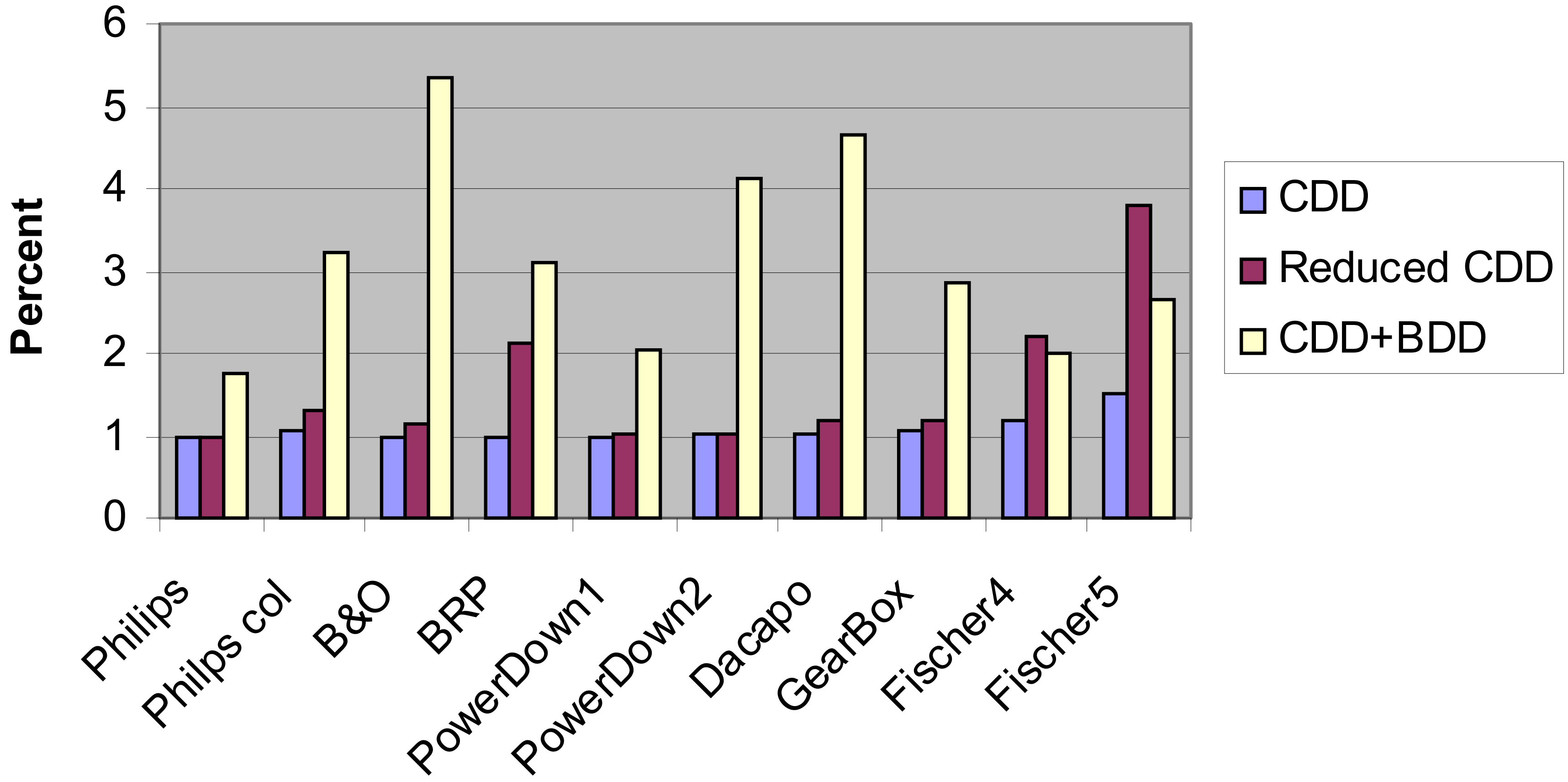
UCb

SPACE PERFORMANCE



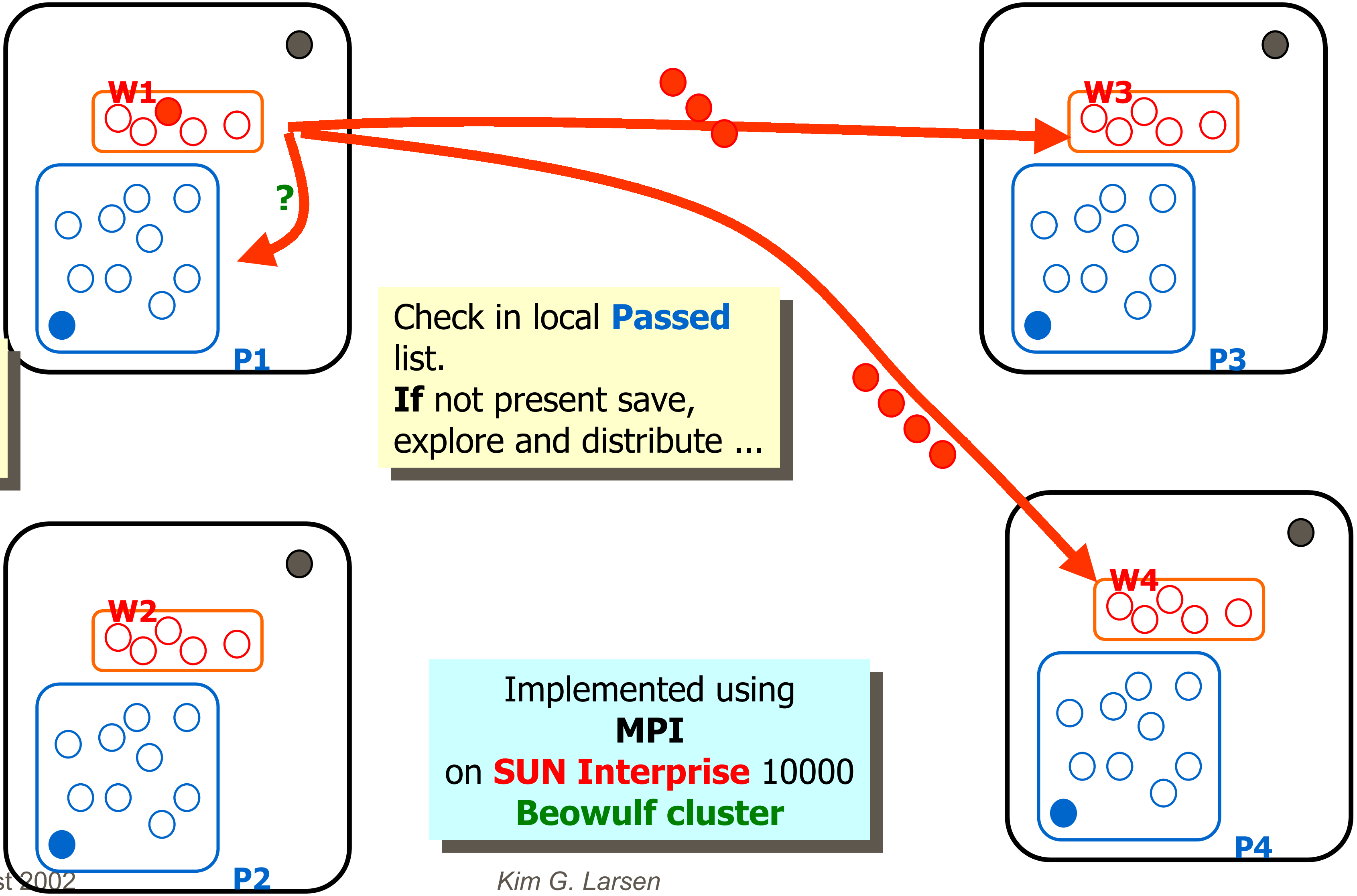
UCb

TIME PERFORMANCE



Distributing **UPPAAL**

Gerd Behrmann, Thomas Hune,
Frits Vandraager **CAV2k**



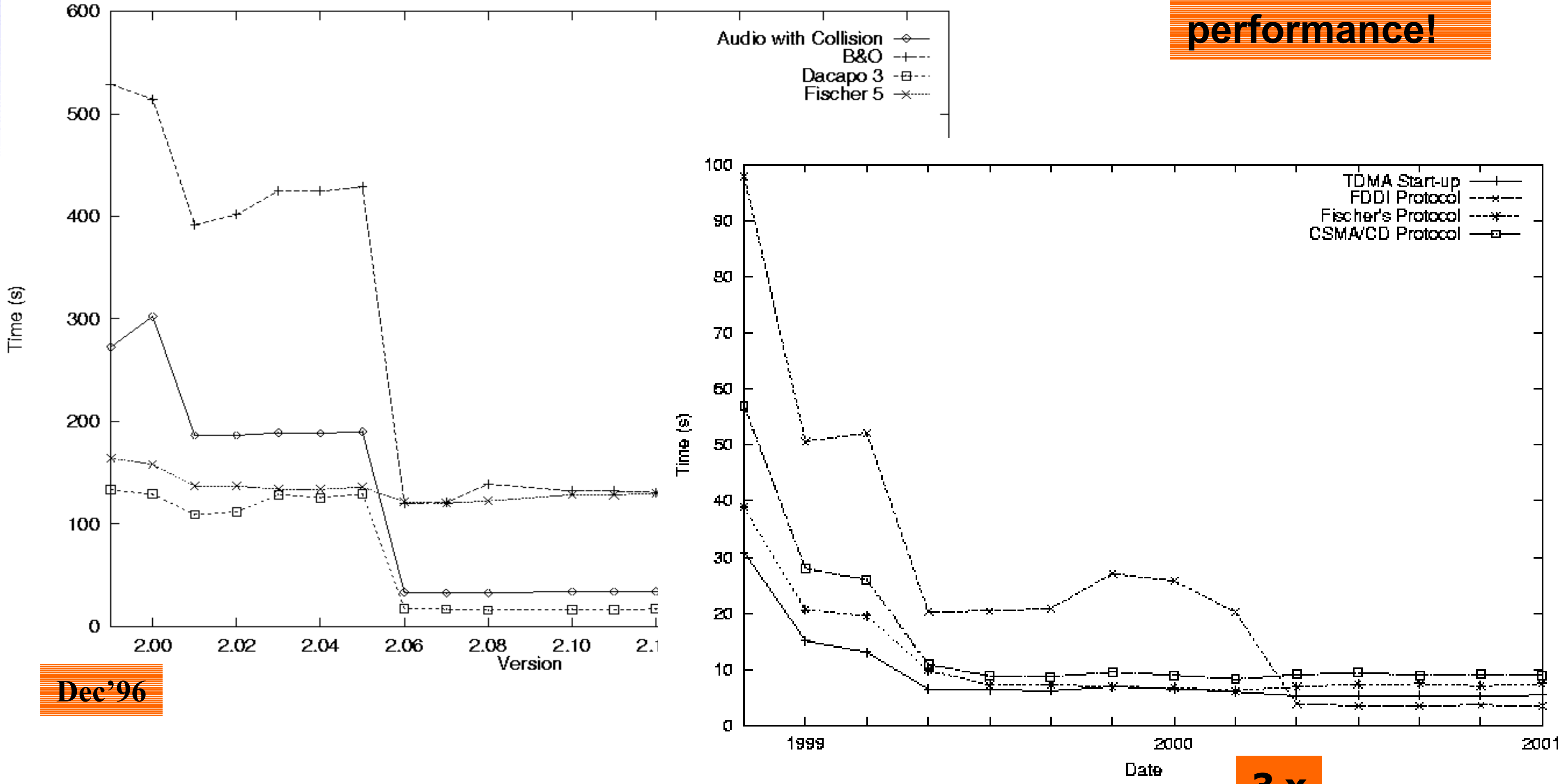
Passed **structure distributed**

Check in local **Passed** list.
If not present save, explore and distribute ...

Implemented using
MPI
on **SUN Interprise** 10000
Beowulf cluster

UPPAAL 1995 - 2001

Every 9 month 10 times better performance!

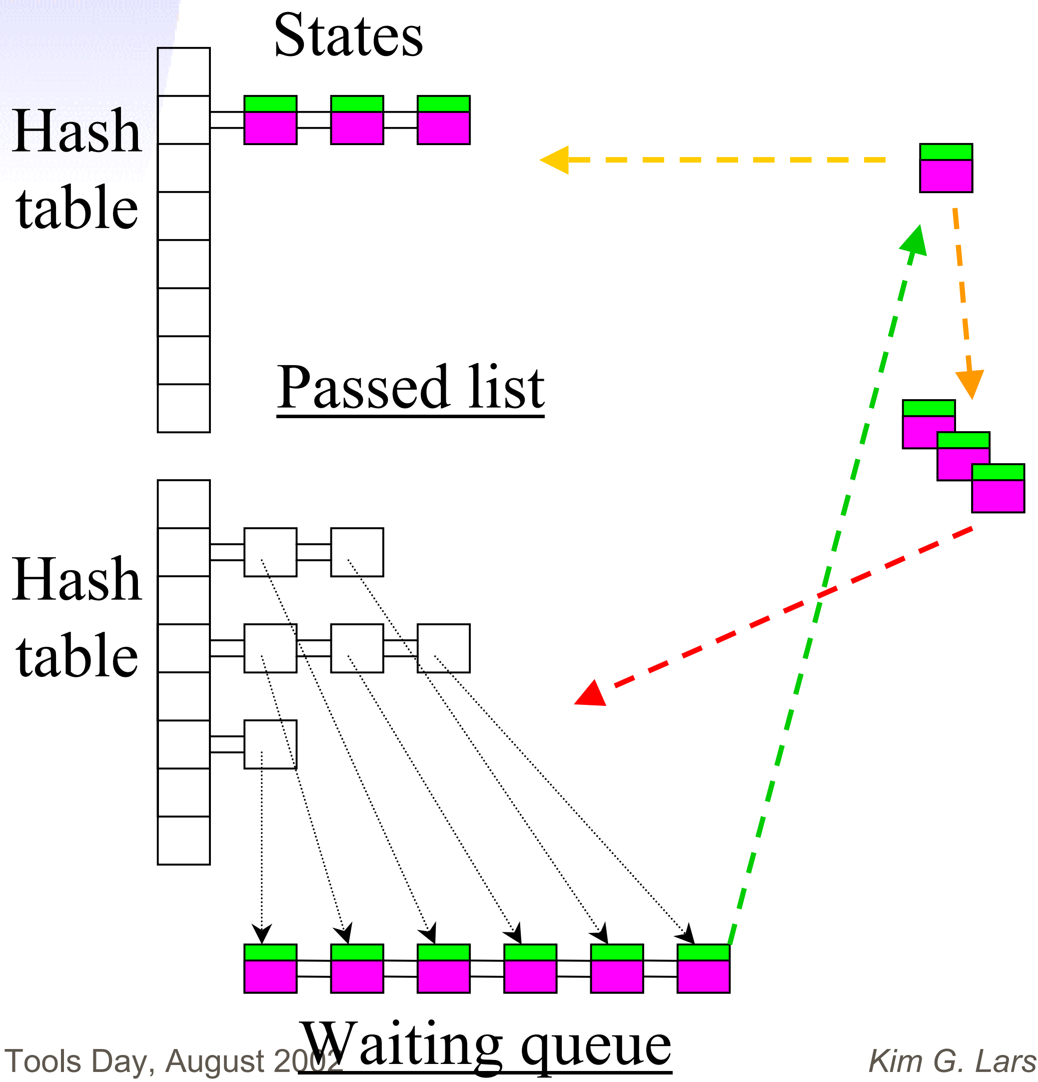


Dec'96

3.x

P+W Reachability

Alexandre & Gerd with Wang & Kim RT-TOOLS'02



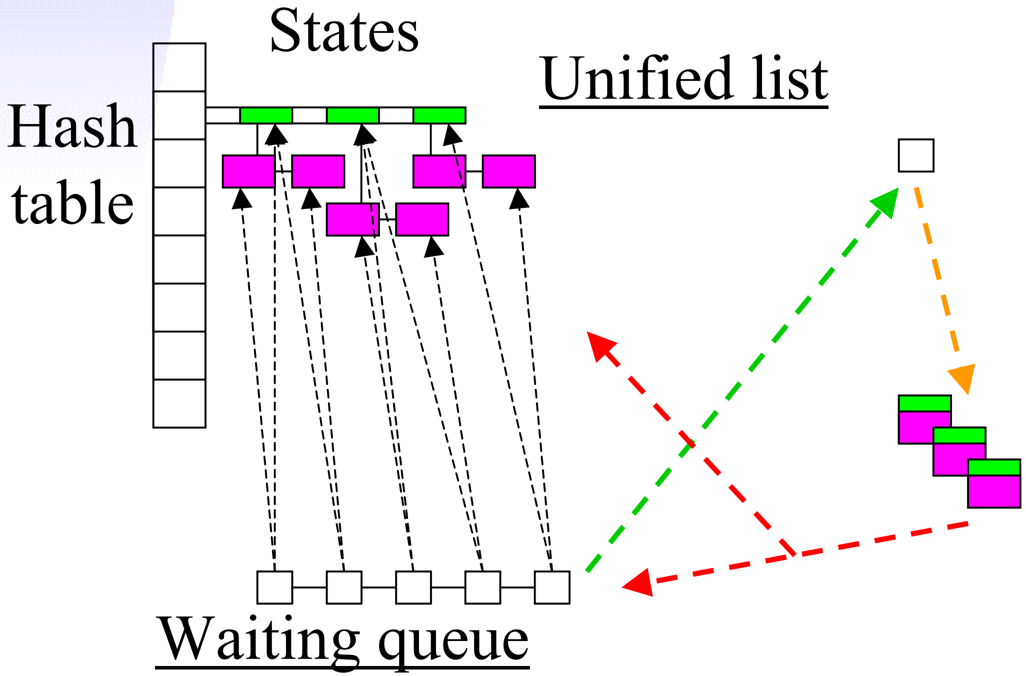
Searching:

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

UCb

PW Reachability

Alexandre & Gerd
with Wang & Kim
RT-TOOLS'02

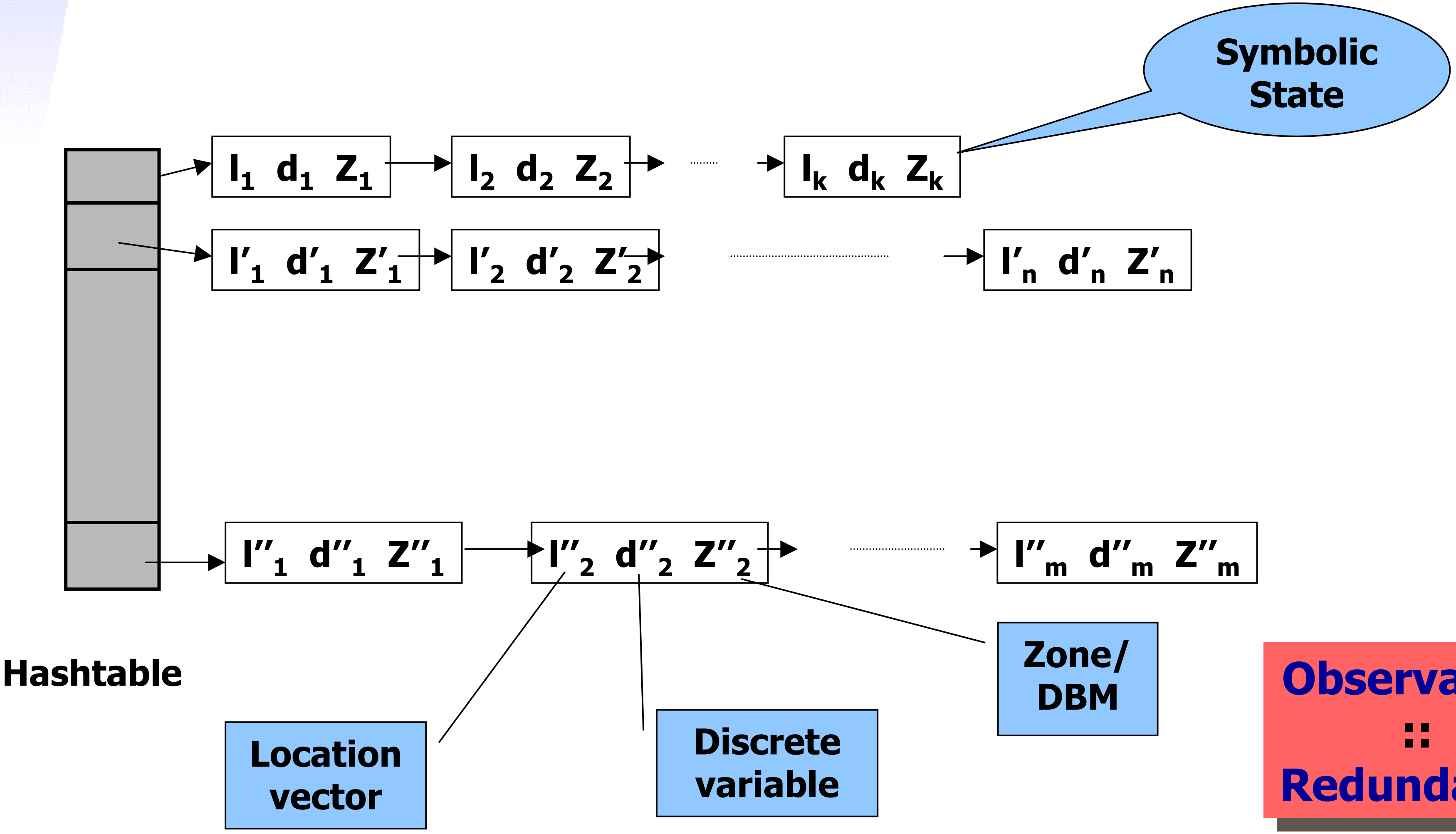


Searching:

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

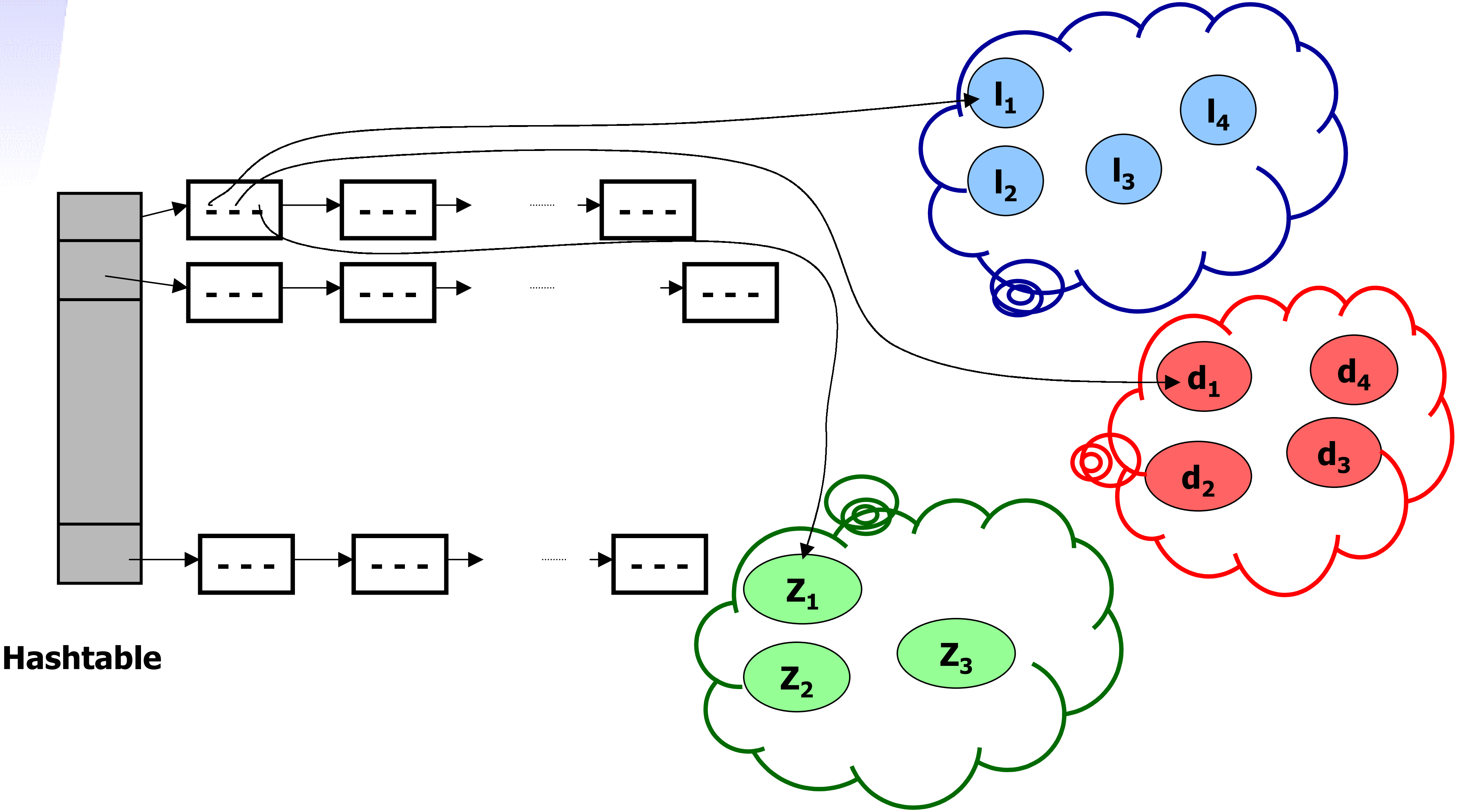
Dynamic DBMs & Sharing

Alexandre & Gerd with Wang & Kim RT-TOOLS'02

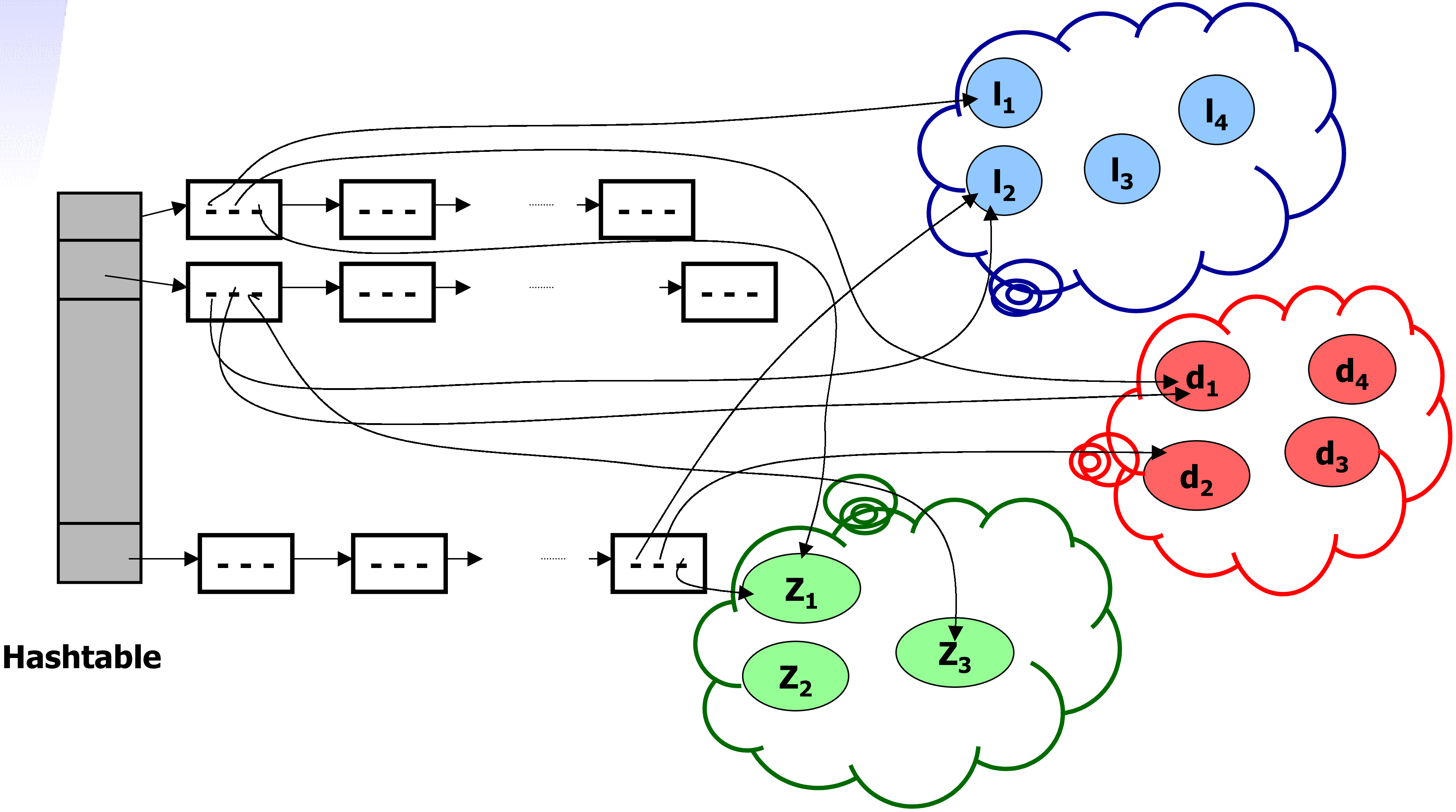


Observation :: Redundancy

Dynamic DBMs & Sharing

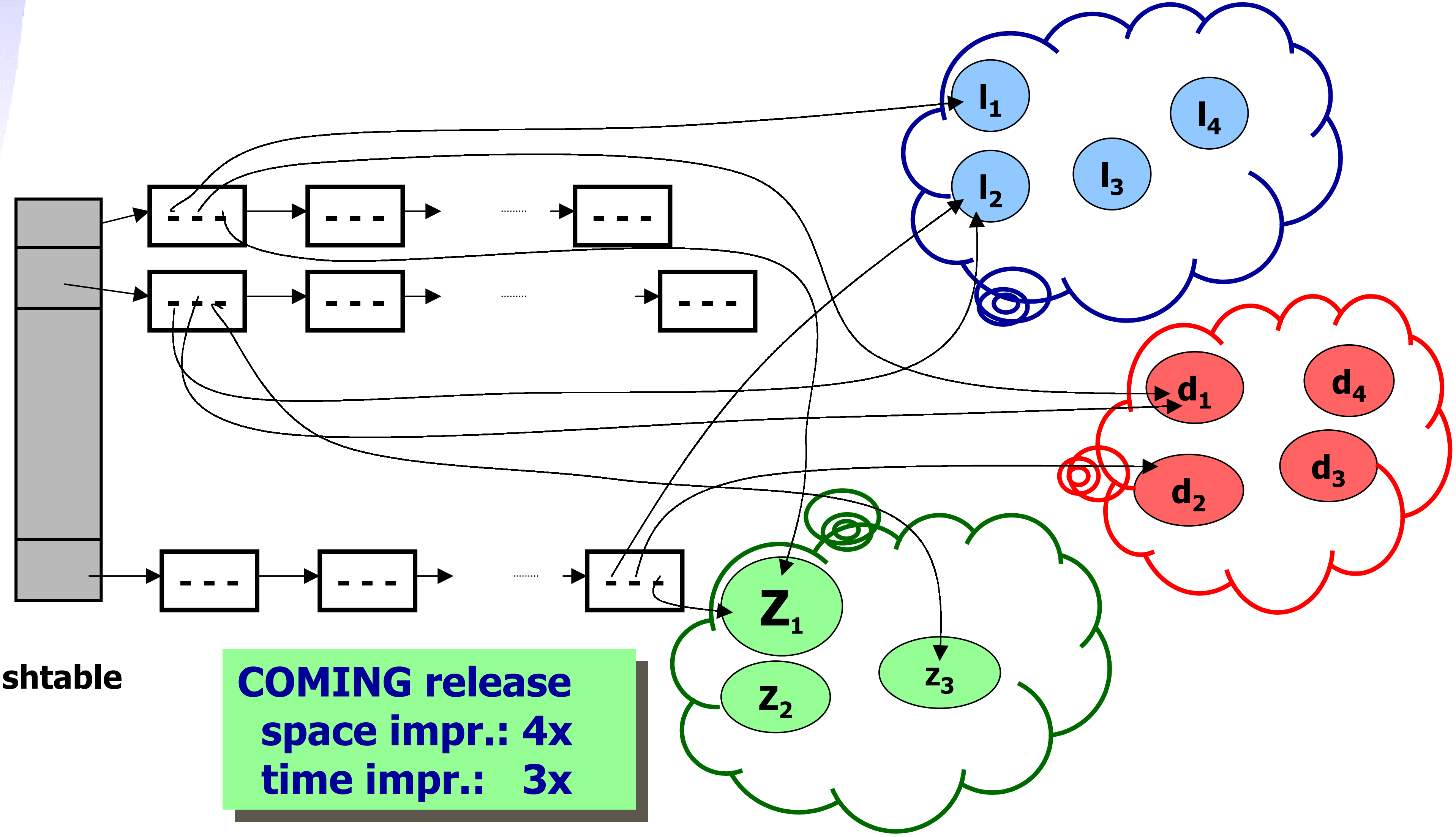


Dynamic DBMs & Sharing



Dynamic DBMs & Sharing

RT-TOOLS'02



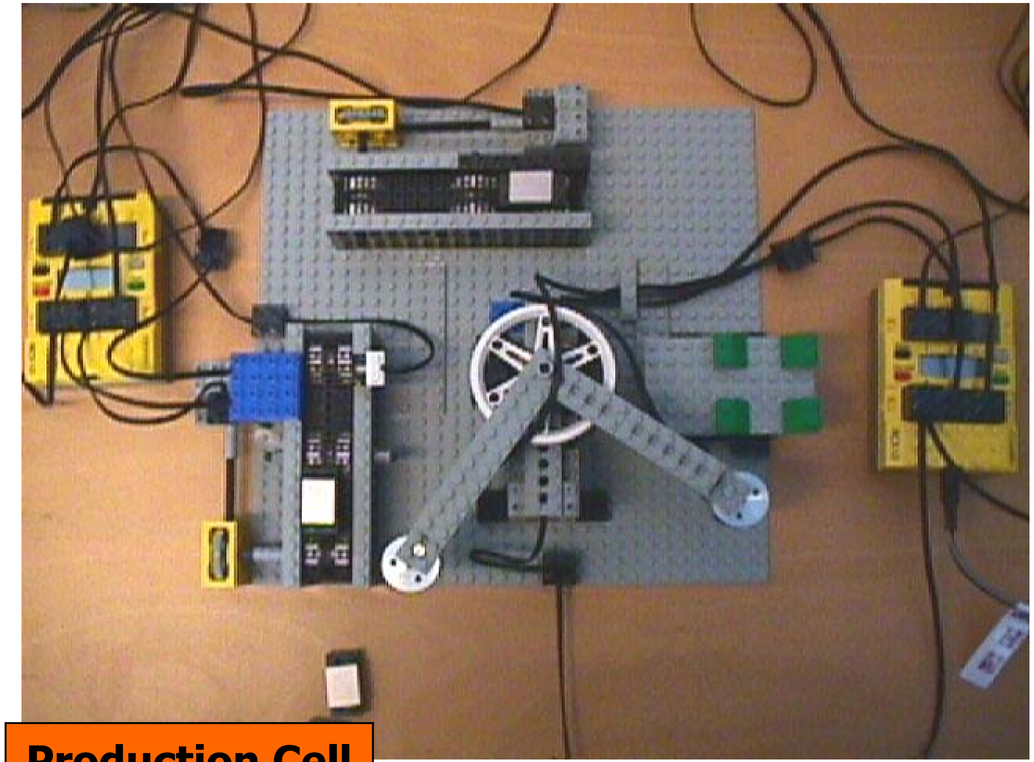
COMING release
space impr.: 4x
time impr.: 3x

Accelleration & Metatransitions

Different Time Scales Models

Typical: Models of scheduled and polling control programs (microseconds) operating in an environment (seconds); e.g. PLC programs or LEGO Mindstorms

```
while(true){
   wait(IN_1<=LIGHT_LEVEL);
   ClearTimer(1);
   active=1;
   PlaySound(1);
   wait(IN_1>LIGHT_LEVEL);
```



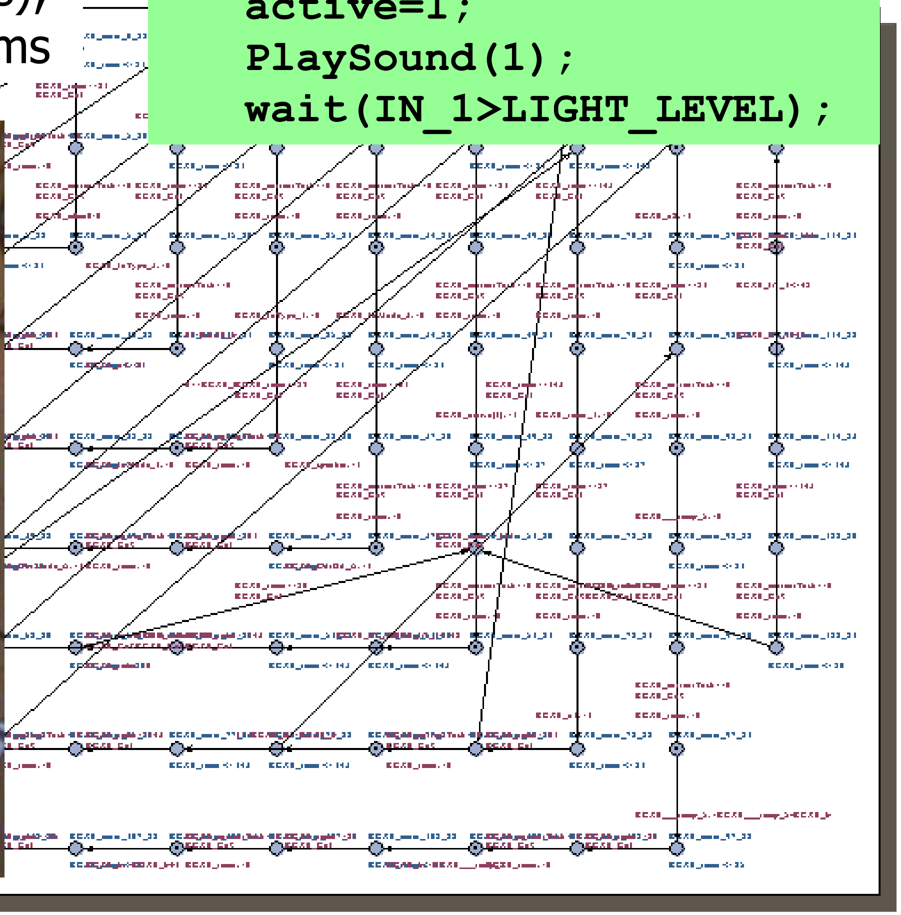
Production Cell

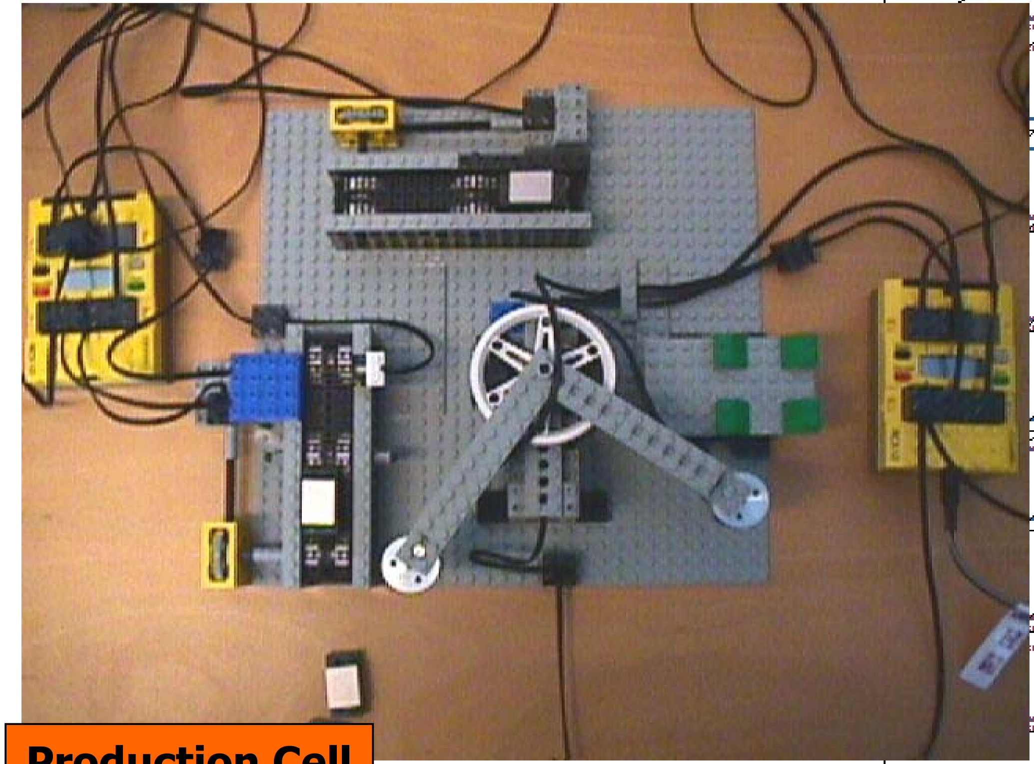
w Martijn Hendriks, TPTS 2002

Different Time Scales Models

Typical: Models of scheduled and polling control programs (microseconds) operating in an environment (seconds); e.g. PLC programs or LEGO Mindstorms

```
while(true){
   wait(IN_1<=LIGHT_LEVEL);
   ClearTimer(1);
   active=1;
   PlaySound(1);
   wait(IN_1>LIGHT_LEVEL);
```

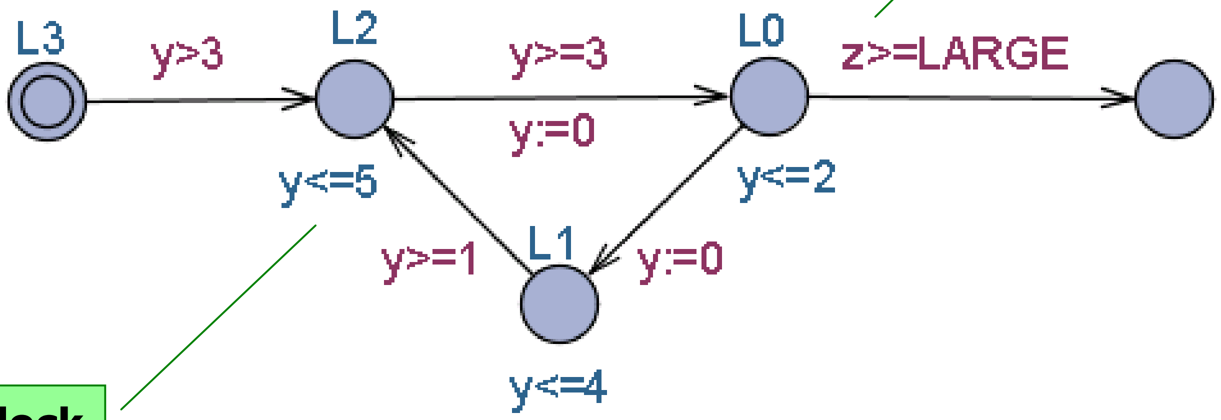




Production Cell

w Martijn Hendriks, TPTS 2002

Fragmentation



z: clock of the Environment

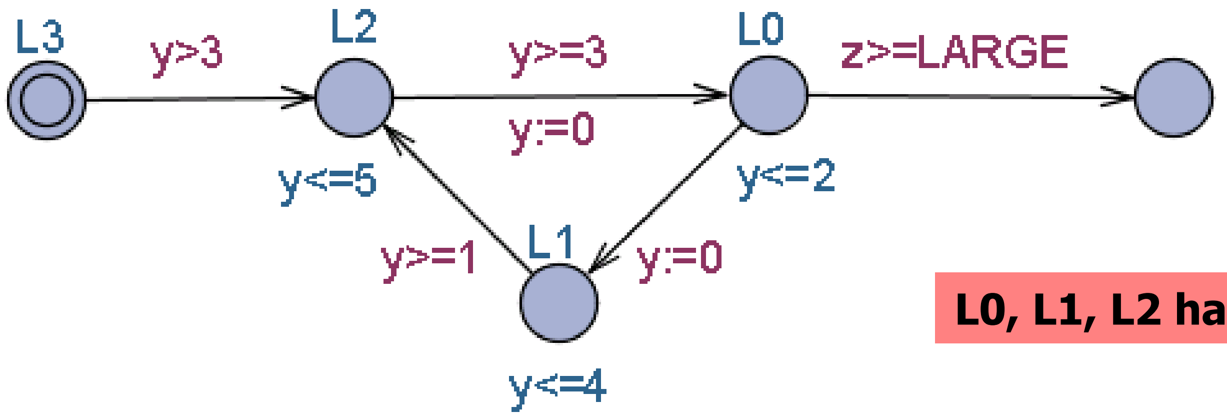
y: the clock used by Control Program

FRAGMENTATION PROBLEM of Symbolic State Space
→
Idea: accelerate cycles

w Martijn Hendriks, TPTS 2002

UCb

Window



L0, L1, L2 has window [3,7]

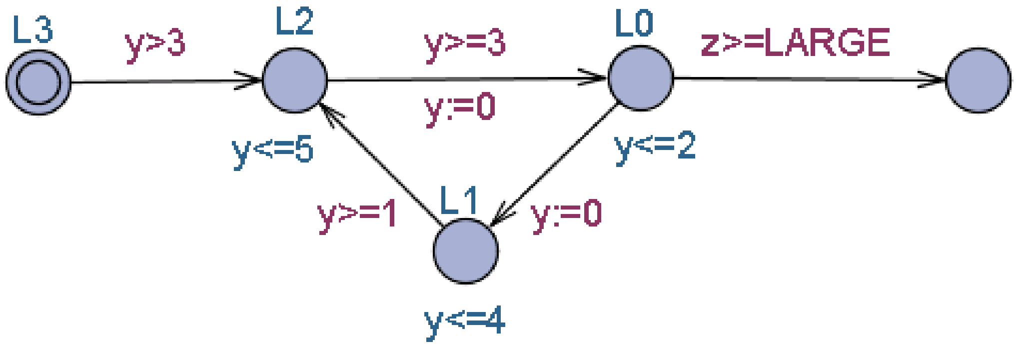
DEFN

WINDOW of a cycle C =(e_1, e_2, … , e_k) is [a,b] iff

1. Every execution of C has accumulated delay between a and b.

2. For any delay d between a and b there there exists an execution of C with accumulated delay d.

Acceleratable Cycles



DEFN

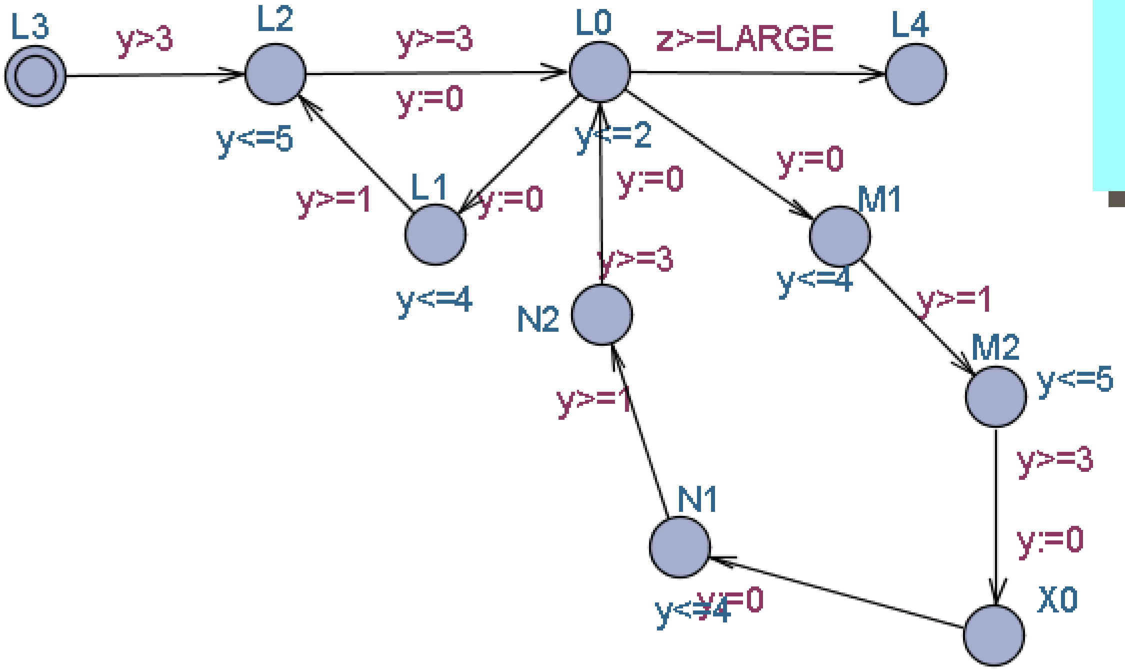
Let $C = (e_1, e_2, \dots, e_k)$ be a cycle and let y be a clock.
Then (C,y) is an accelaratable cycle if:

1. Every invariant of C is of the form $y<=n$ (or true)
2. Every guard of C is of the form $y>=m$ (or true)
3. y is reset on all ingoing edges to src(e_1)

THM

Every acceleratable cycle has a window

Exact Acceleration

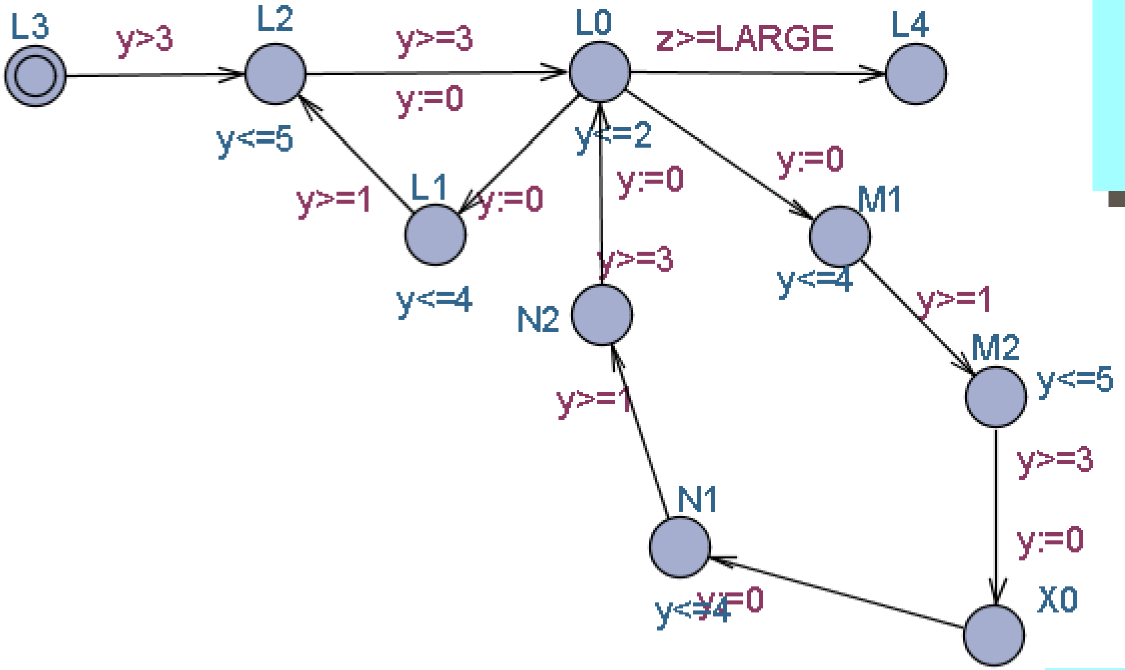


Acceleration of cycle
=
"Unfolding of cycle"

THM

$$3a \leq 2b \Rightarrow (M \models \phi \Leftrightarrow Acc(M,A) \models \phi)$$

Exact & Efficient Acceleration



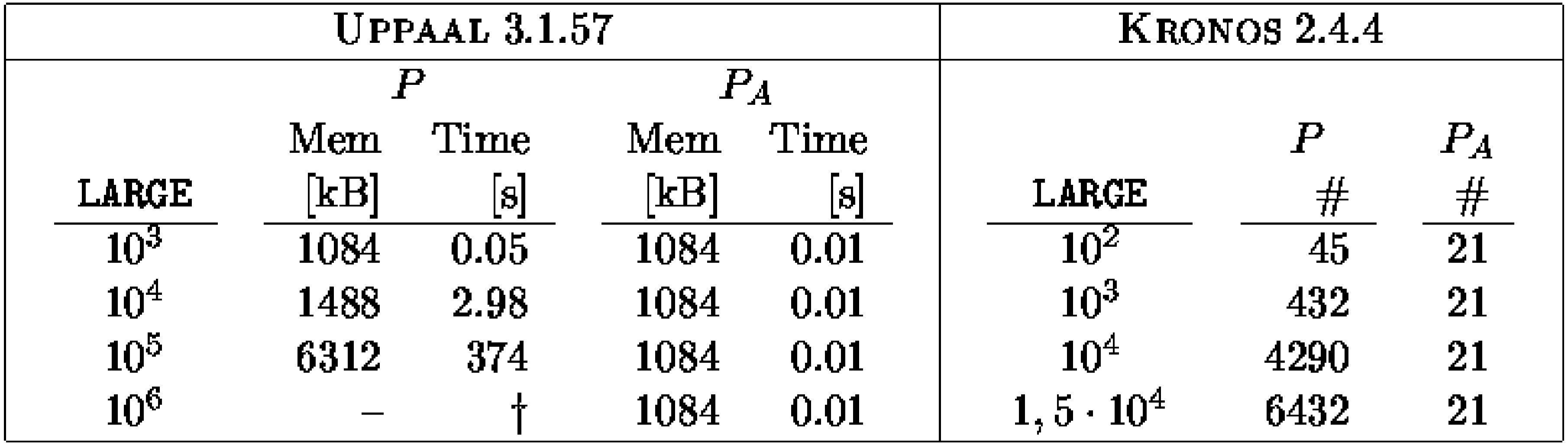
Acceleration of cycle
=
"Unfolding of cycle"

THM

$$3a \leq 2b \Rightarrow (M \models \phi \Leftrightarrow Acc(M,A) \models \phi)$$

If y is reset on the first edge in the accelerated cycle C, then one execution of the appended cycle suffices!!

Experimental Results



UPPAAL 3.1.57					KRONOS 2.4.4		
	P		P_A				
	Mem	Time	Mem	Time		P	P_A
LARGE	[kB]	[s]	[kB]	[s]	LARGE	#	#
10^3	1084	0.05	1084	0.01	10^2	45	21
10^4	1488	2.98	1084	0.01	10^3	432	21
10^5	6312	374	1084	0.01	10^4	4290	21
10^6	–	†	1084	0.01	$1,5 \cdot 10^4$	6432	21

On sample LEGO Mindstorm Models:

Speed-up-factor:	2-5
With convex-hull:	200

MAIN Questions

Which cycles to ”accelerate”

Generalization to full model

Beyond Model Checking

Formal
methods
& Tools

University of Twente
department of
computer science

CUPPAAL

optimal

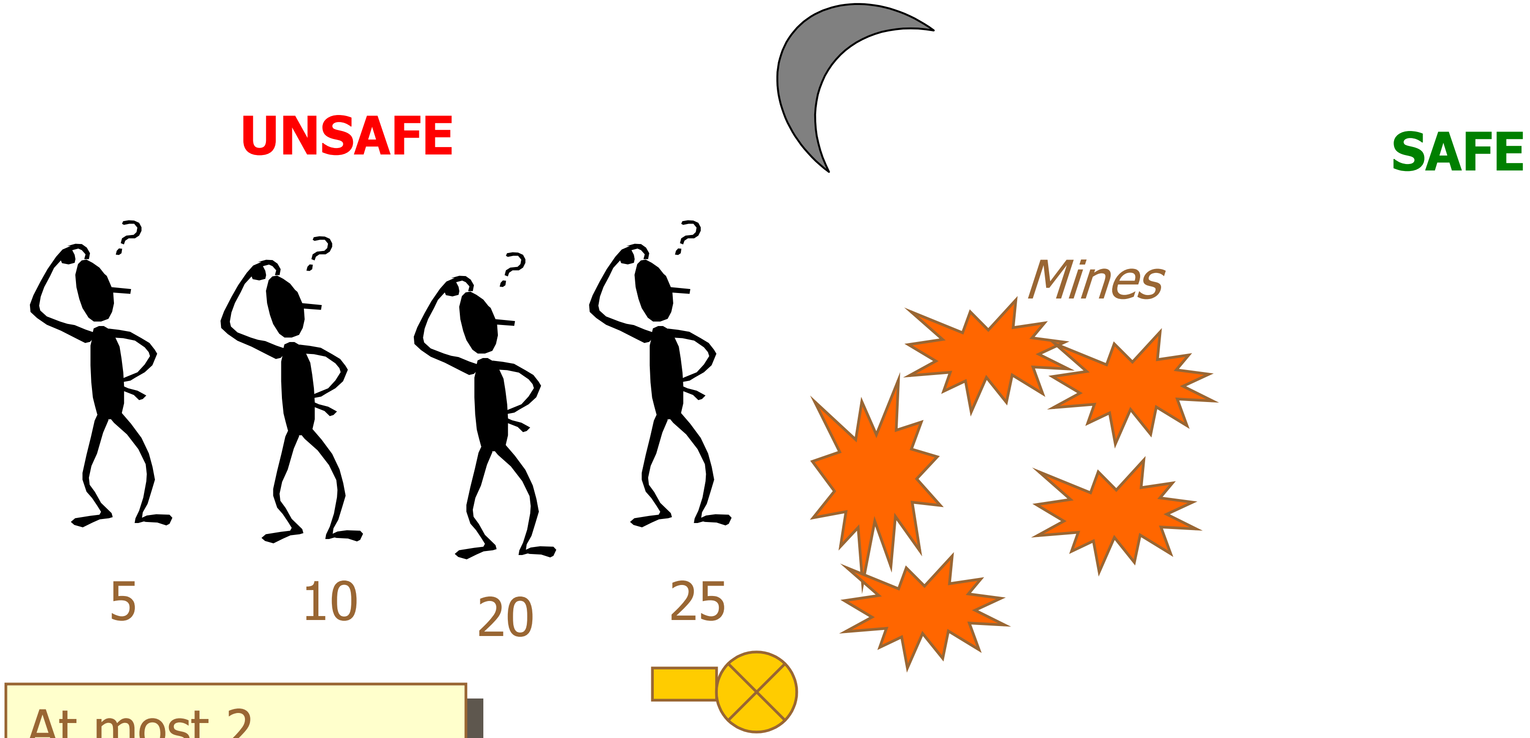
Scheduling &
Synthesis of Control Programs

w Gerd Behrman, Ed Brinksma, Ansgar Fehnker,
Thomas Hune, Paul Pettersson,
Judi Romijn, Frits Vaandrager, Henning Dierks

…, HSCC'01, TACAS'01, CAV'01, AIPS'02,…. **AMETIST**

UCb

A real-time scheduling problem

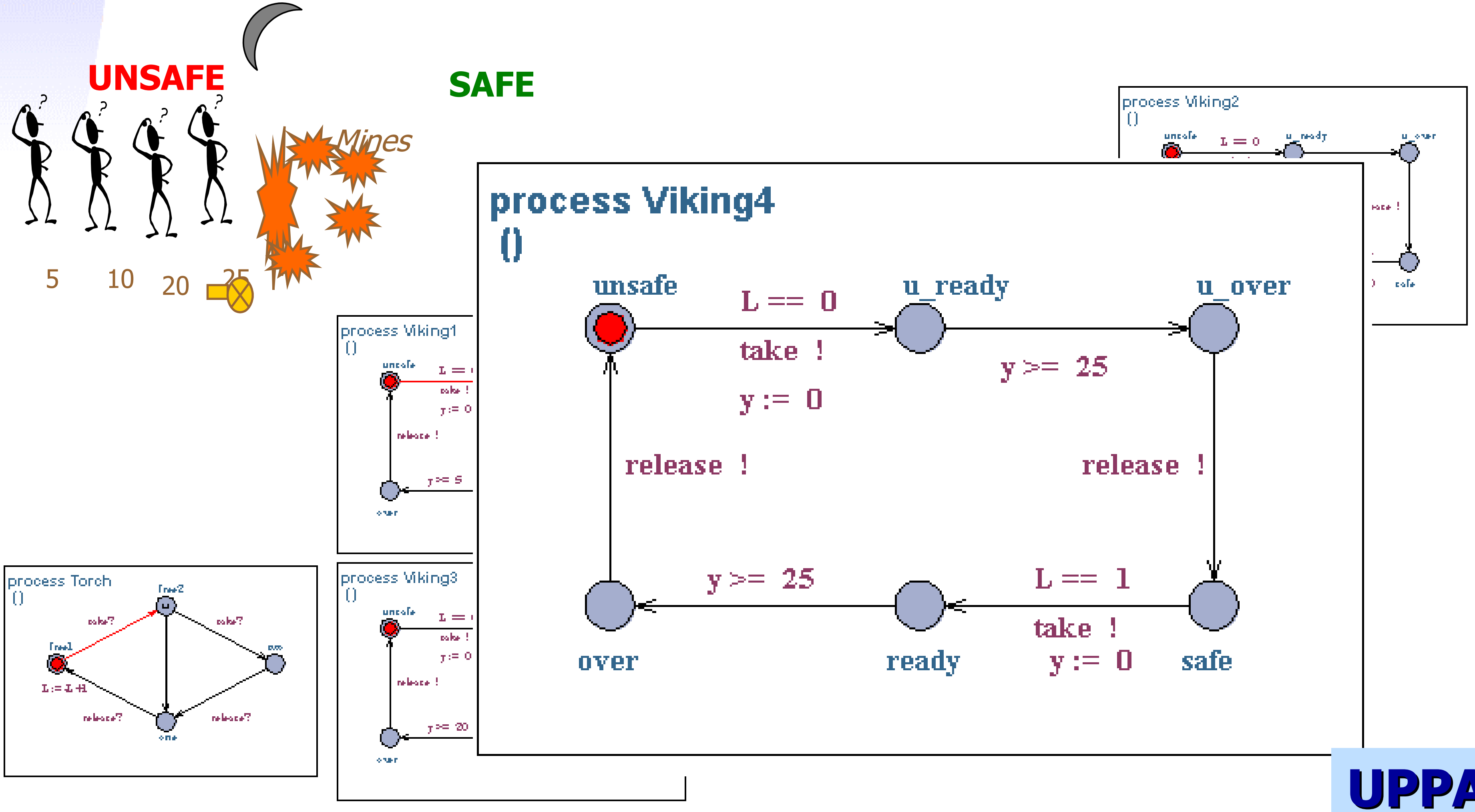


At most 2
crossing at a time
Need torch

Can they make
it within 60 minutes ?

Observation

Many scheduling problems can be phrased naturally as reachability problems for timed automata!

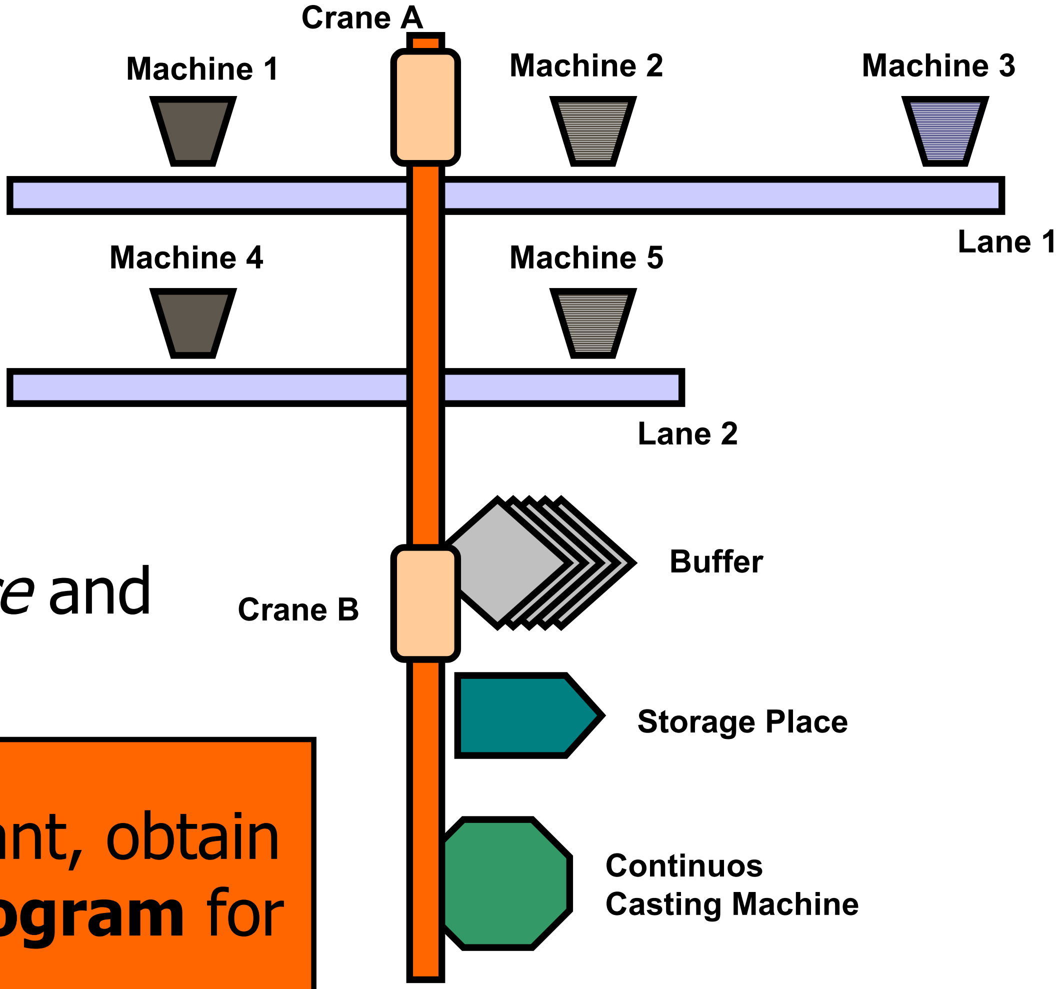


UPPAAL

Steel Production Plant

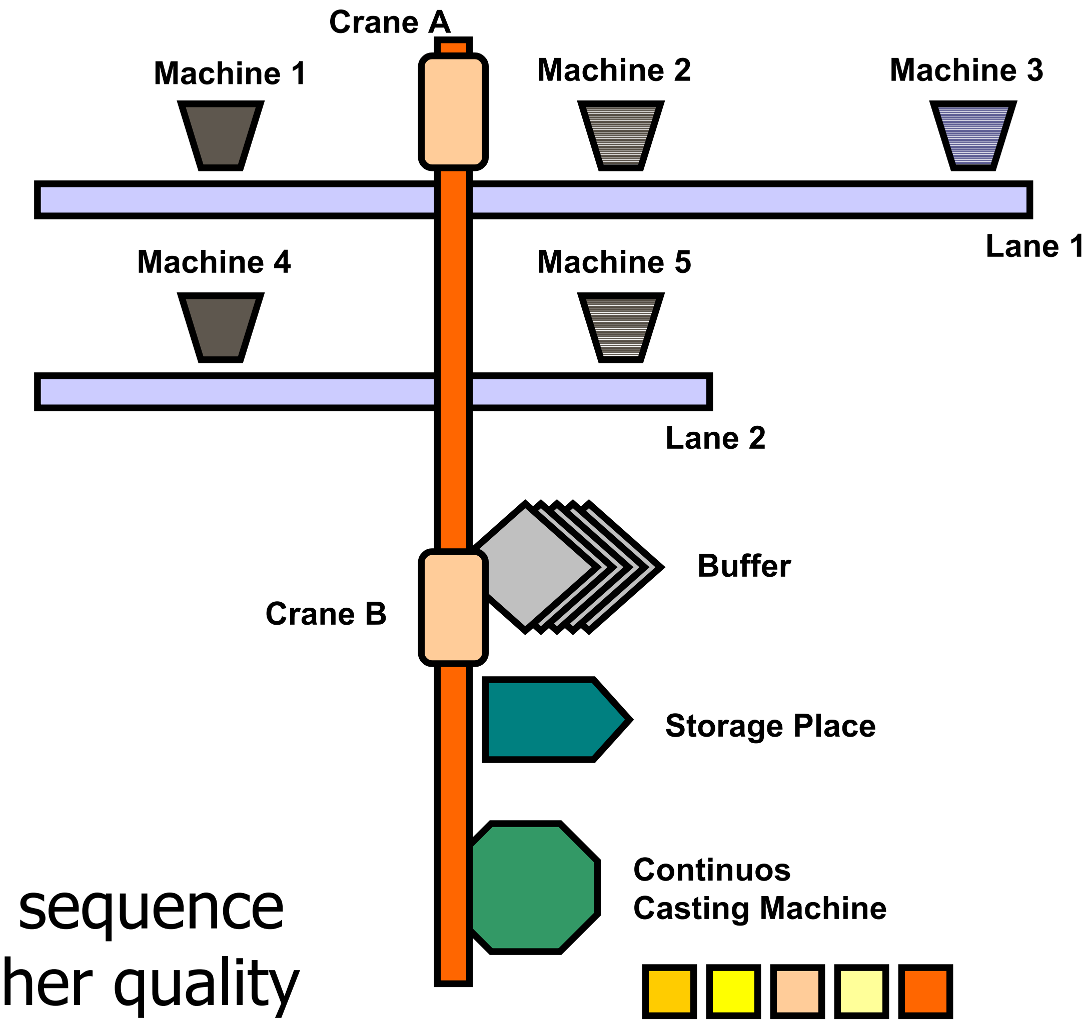
- A. Fehnker
- Hune, Larsen, Pettersson
- Case study of Esprit-LTR project 26270 VHS
- Physical plant of SIDMAR located in Gent, Belgium.
- Part between *blast furnace* and *hot rolling mill*.

Objective: model the plant, obtain **schedule** and control **program** for plant.



Steel Production Plant

Input: sequence of steel loads ("pigs").



Load follows **Recipe** to become certain quality, e.g:

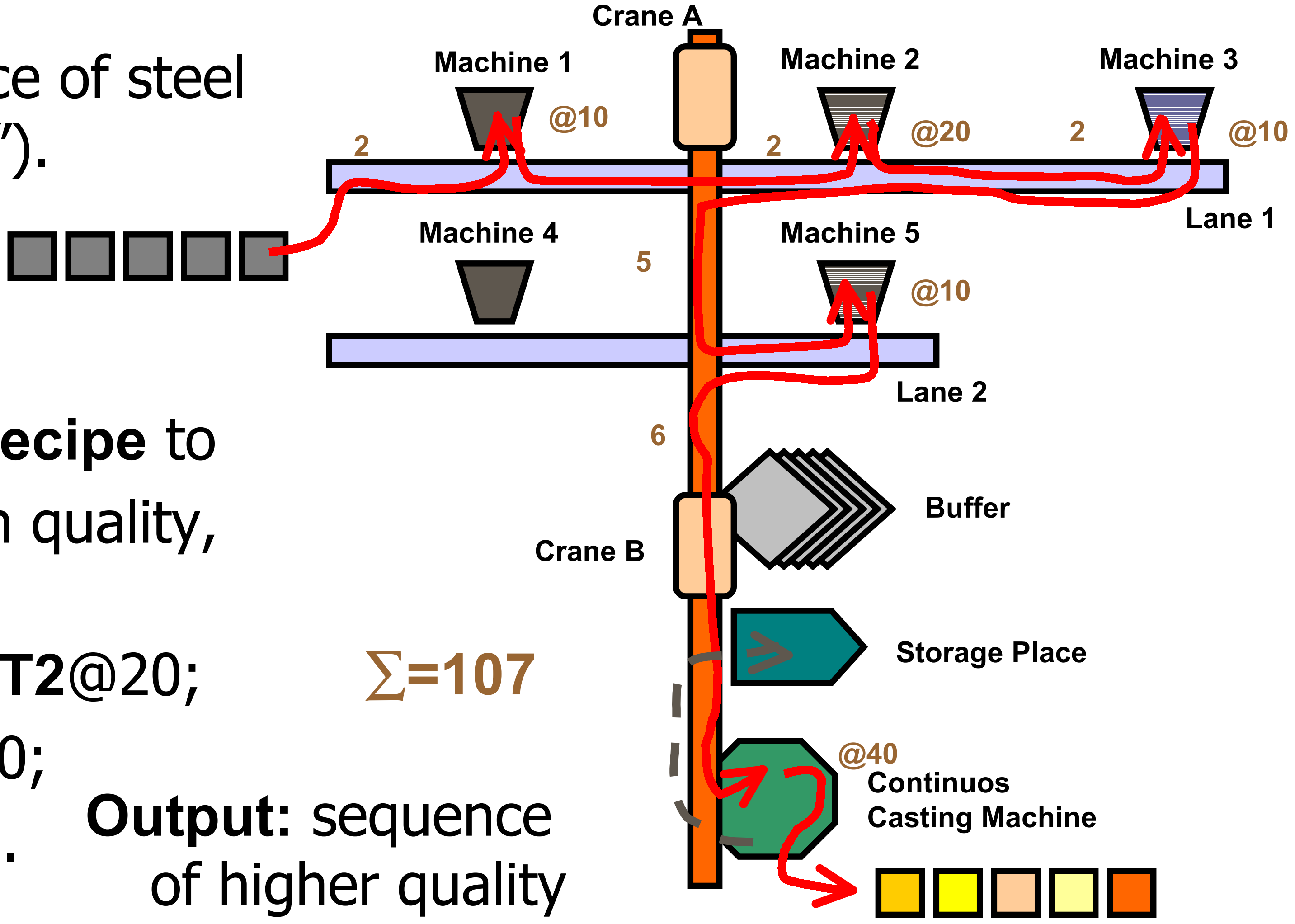
start; **T1**@10; **T2**@20; **T3**@10; **T2**@10;

end within 120.

Output: sequence of higher quality steel.

Steel Production Plant

Input: sequence of steel loads ("pigs").



Load follows **Recipe** to become certain quality, e.g:

start; **T1**@10; **T2**@20; **T3**@10; **T2**@10;

end within 120.

∑=107

Output: sequence of higher quality steel.

Steel Production Plant

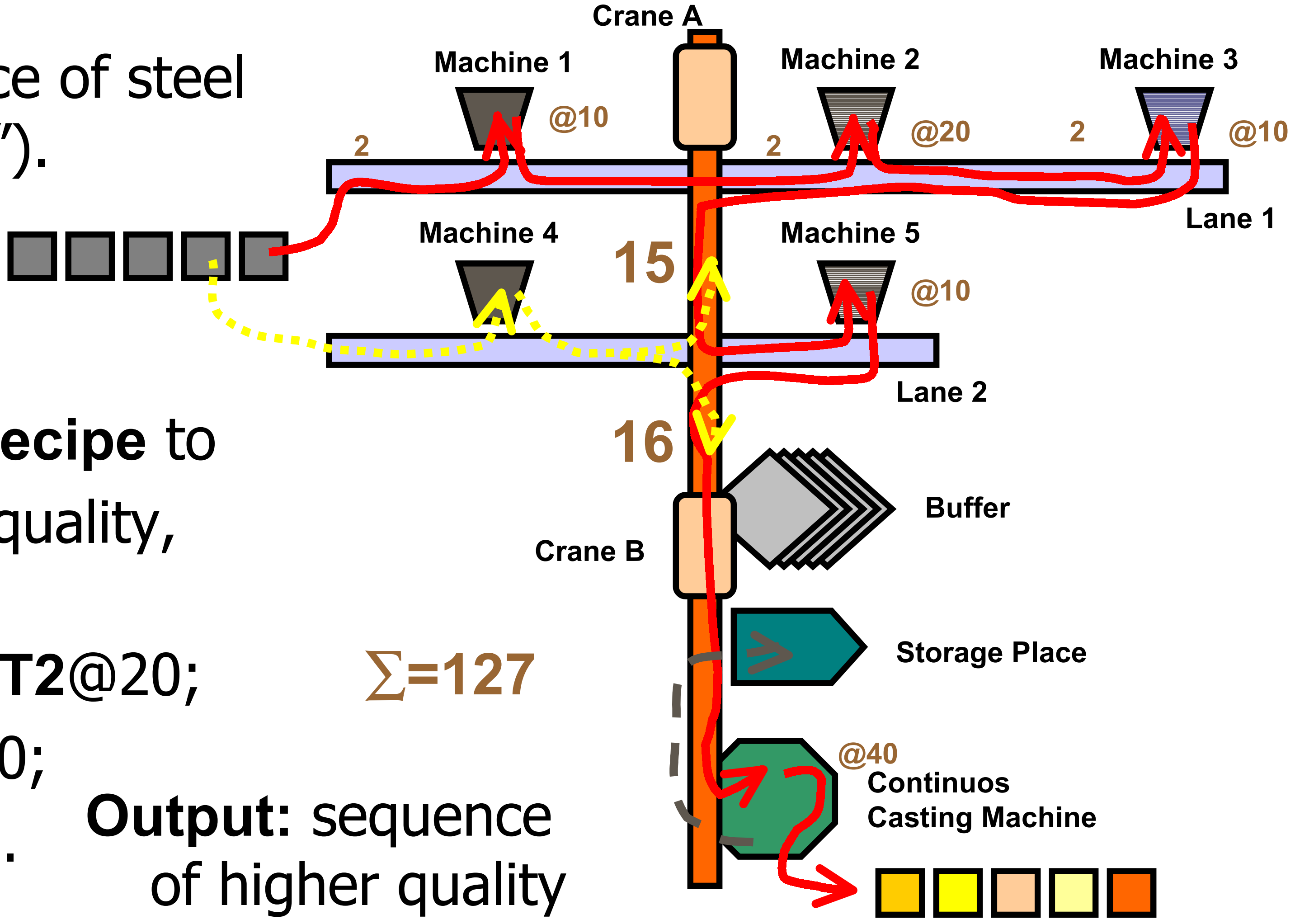
Input: sequence of steel loads (“pigs”).

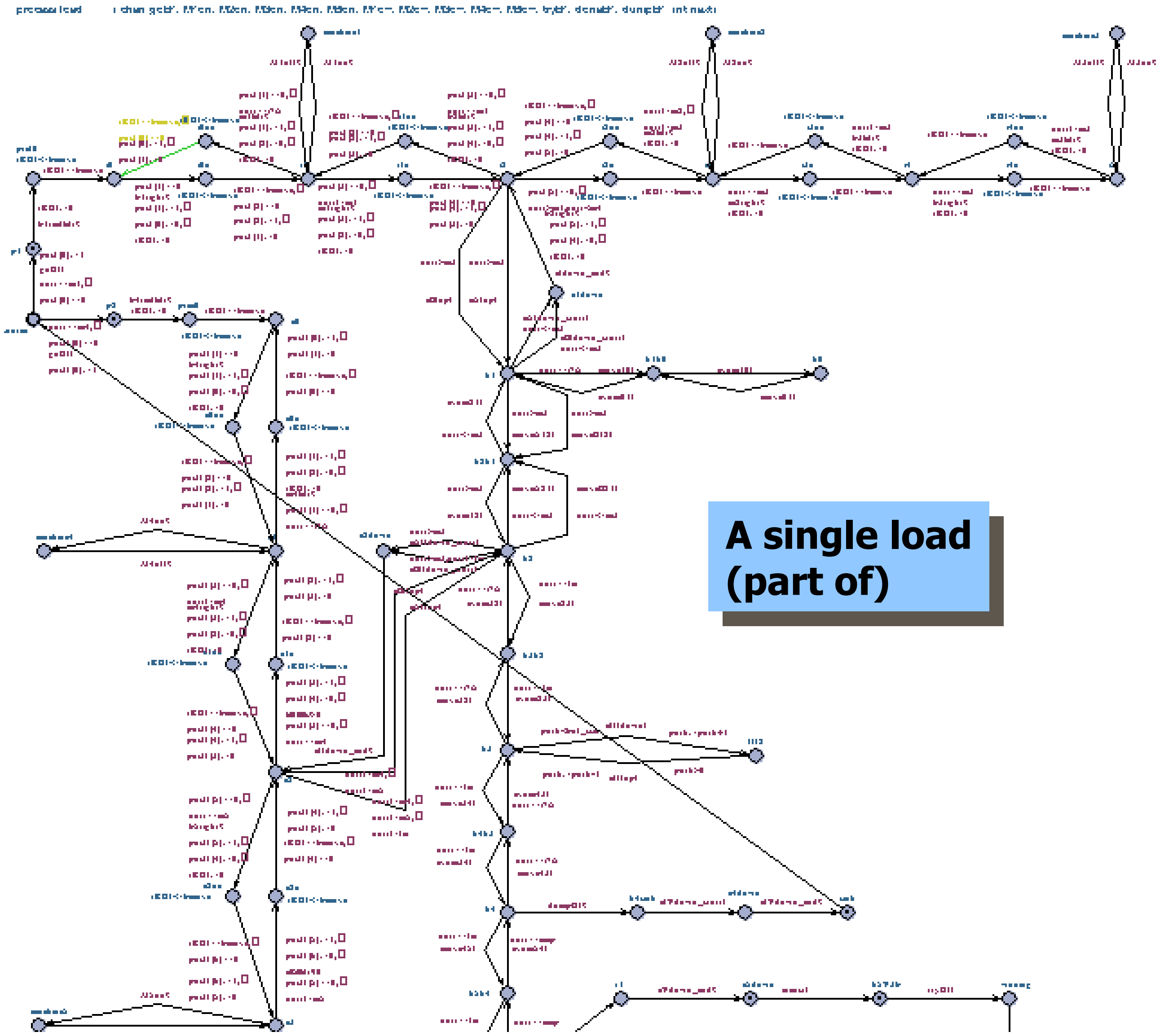
Load follows **Recipe** to obtain certain quality, e.g:

start; **T1**@10; **T2**@20; **T3**@10; **T2**@10;

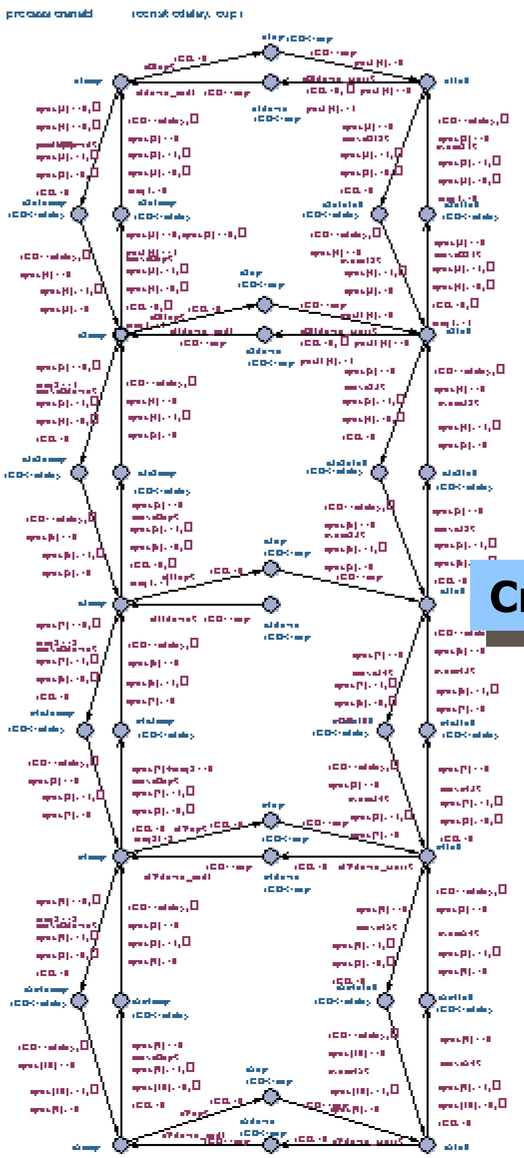
end within 120.

Output: sequence of higher quality steel.





A single load (part of)



Crane B

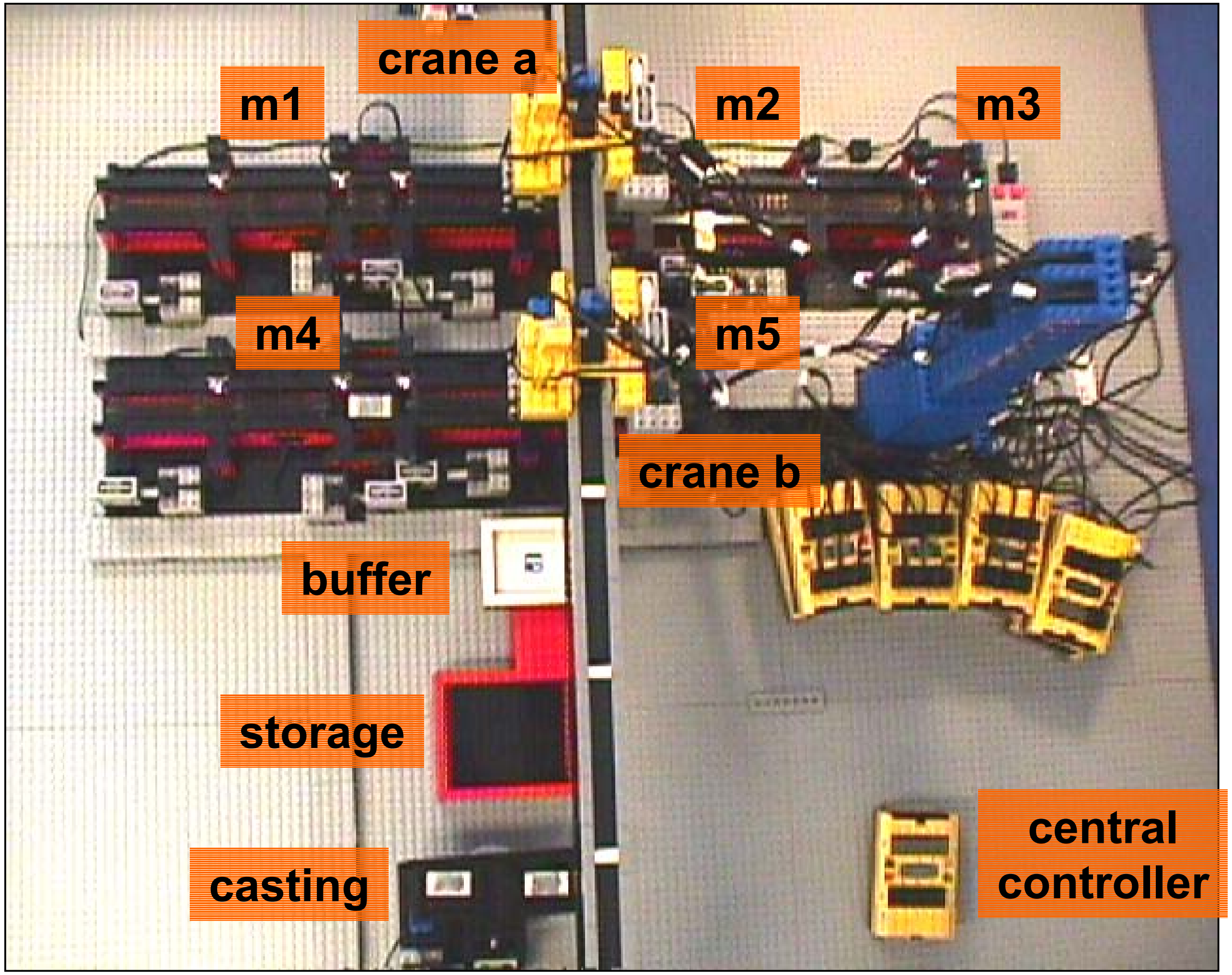
Experiment

n	All Guides						Some Guides						No Guides					
	BFS		DFS		BSH		BFS		DFS		BSH		BFS		DFS		BSH	
	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB
1	0,1	0,9	0,1	0,9	0,1	0,9	0,1	0,9	0,1	0,9	0,1	0,9	3,2	6,1	0,8	2,2	3.9	3.3
2	18,4	36,4	0,1	1	0,1	1,1	-	-	4,4	7,8	7,8	1,2	-	-	19,5	36,1	-	-
3	-	-	3,2	6,5	3,4	1,4	-	-	72,4	92,1	901	3,4	-	-	-	-	-	-
4	-	-	4	8,2	4,6	1,8	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	5	10,2	5,5	2,2	-	-	-	-	-	-	-	-	-	-	-	-
10	-	-	13,3	25,3	16,1	9,3	-	-	-	-	-	-	-	-	-	-	-	-
15	-	-	31,6	51,2	48,1	22,2	-	-	-	-	-	-	-	-	-	-	-	-
20	-	-	61,8	89,6	332	46,1	-	-	-	-	-	-	-	-	-	-	-	-
25	-	-	104	144	87,2	83,3	-	-	-	-	-	-	-	-	-	-	-	-
30	-	-	166	216	124,2	136	-	-	-	-	-	-	-	-	-	-	-	-
35			209	250	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **BFS** = breadth-first search, **DFS** = depth-first search, **BSH** = bit-state hashing,
- **"-"** = requires >2h (on 450MHz Pentium III), >256 MB, or suitable hash-table size was not found.
- **System size**: 2n+5 automata and 3n+3 clocks, if n=35: 75 automata and 108 clocks.
- **Schedule generated for n=60 on Sun Ultra with 2x300MHz with 1024MB in 2257s .**

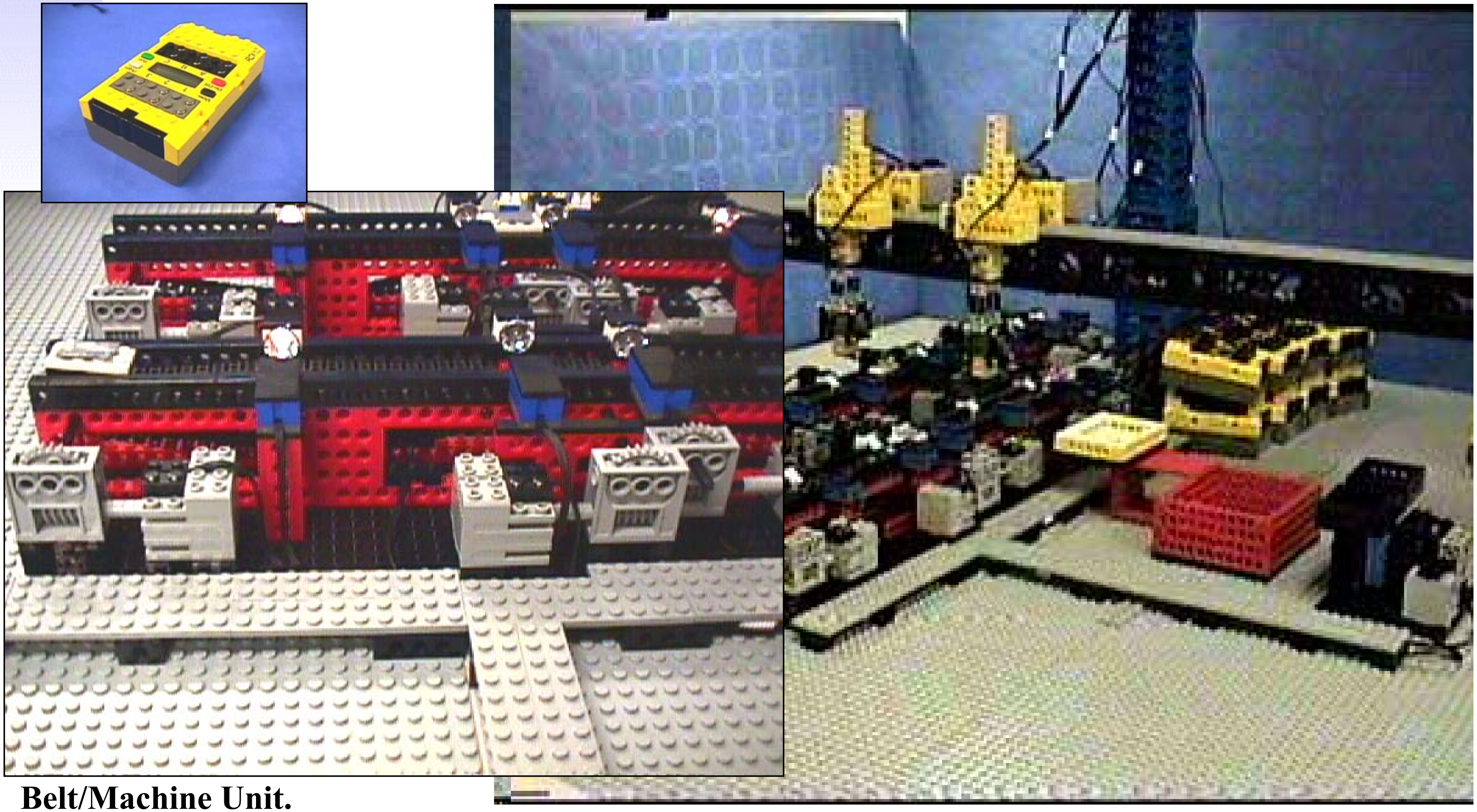
LEGO Plant Model

- LEGO RCX Mindstorms.
- Local controllers with control programs.
- IR protocol for remote invocation of programs.
- Central controller.



Synthesis

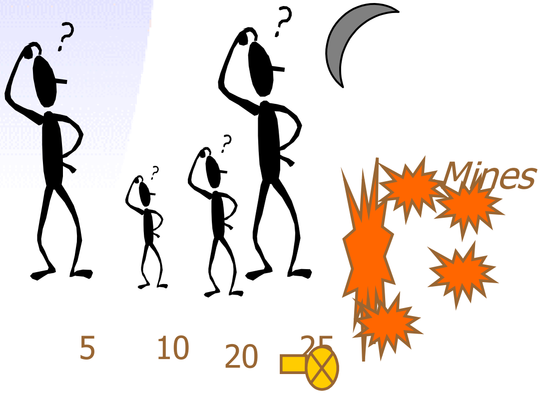
LEGO Plant Model

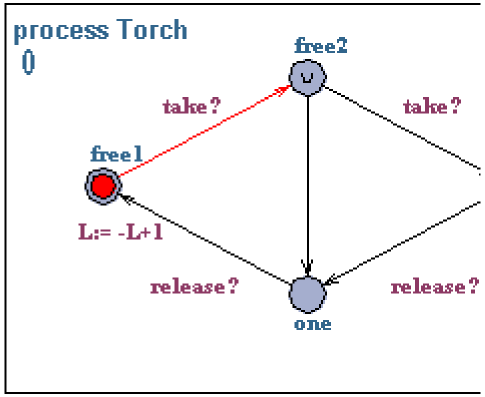


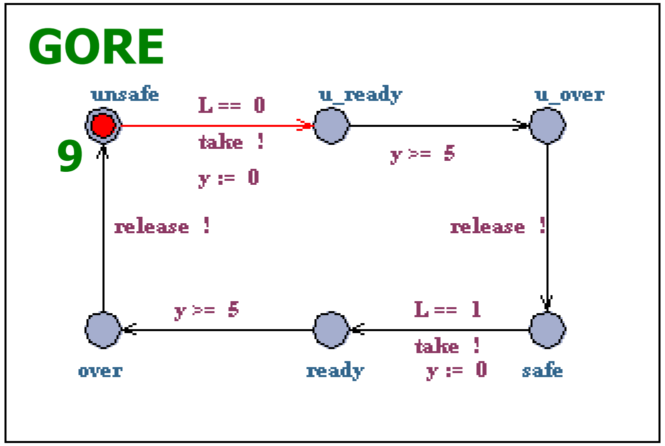
Belt/Machine Unit.

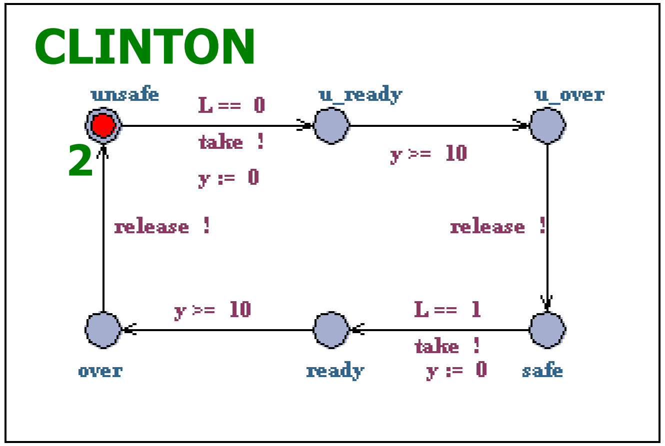
EXAMPLE: Optimal rescue plan for important persons (Presidents and Actors)

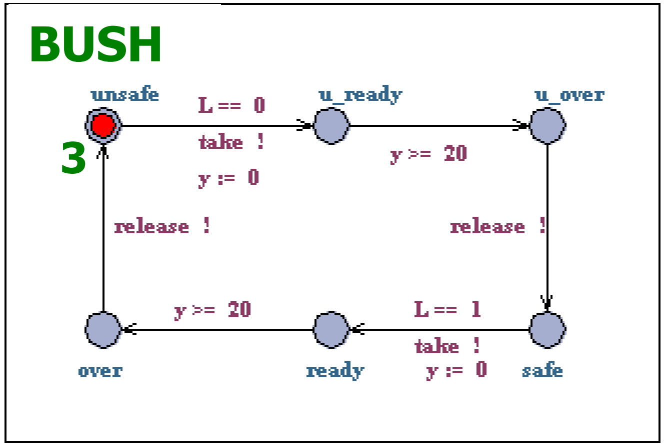
UNSAFE

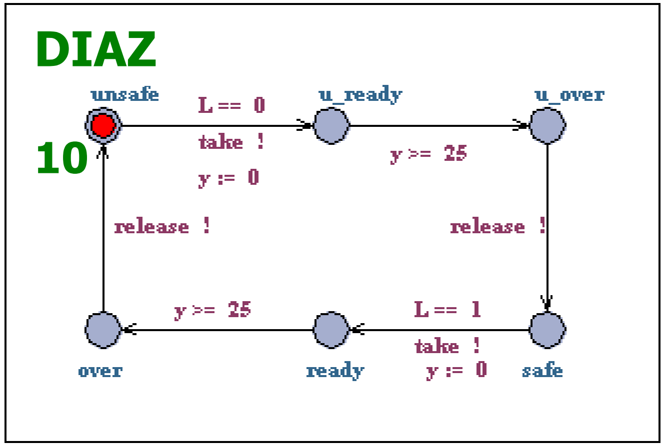










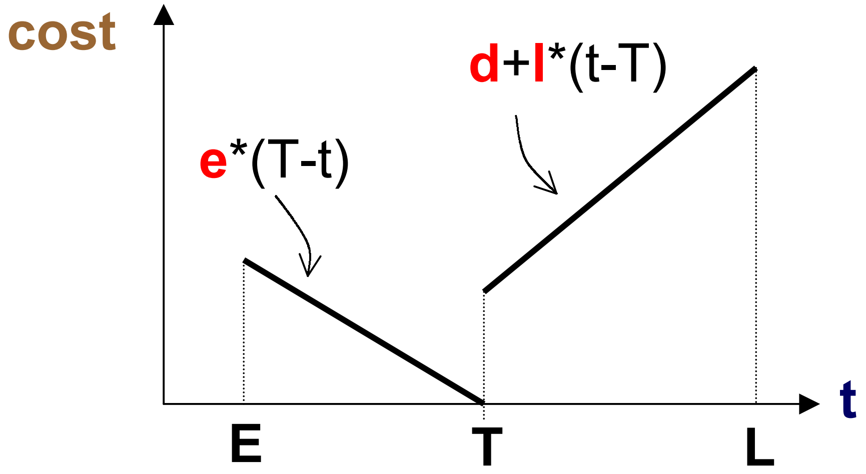


OPTIMAL PLAN HAS ACCUMULATED COST=195 and TOTAL TIME=65!

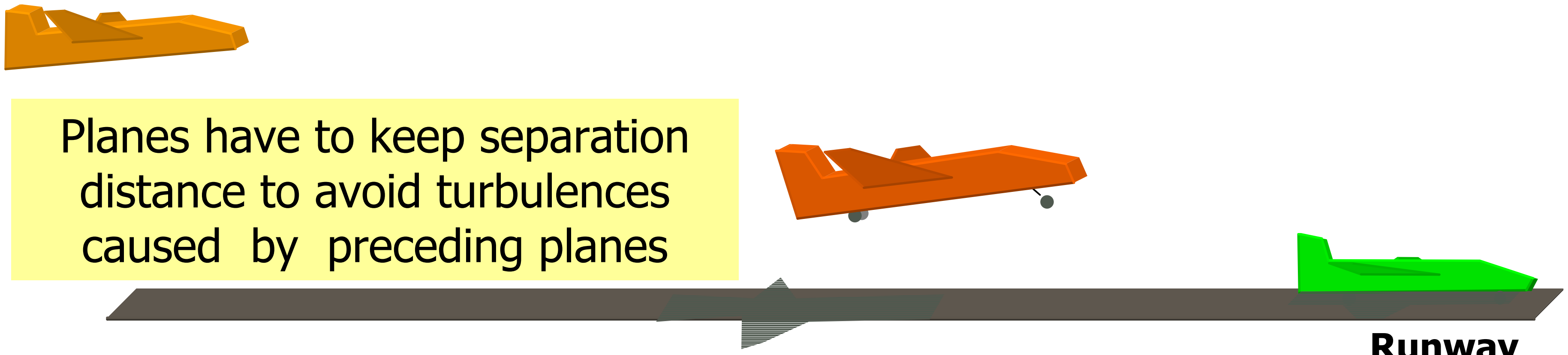
Experiments *MC Order*

COST-rates				SCHEDULE	COST	TIME	#Expl	#Pop'd
G_5	C_{10}	B_{20}	D_{25}					
Min Time				CG> G< BD> C< CG>		60	1762 1538	2638
1	1	1	1	CG> G< BG> G< GD>	55	65	252	378
9	2	3	10	GD> G< CG> G< BG>	195	65	149	233
1	2	3	4	CG> G< BD> C< CG>	140	60	232	350
1	2	3	10	CD> C< CB> C< CG>	170	65	263	408
1	20	30	40	BD> B< CB> C< CG>	975 1085	85 time<85	-	-
0	0	0	0	-	0	-	406	447

Example: Aircraft Landing

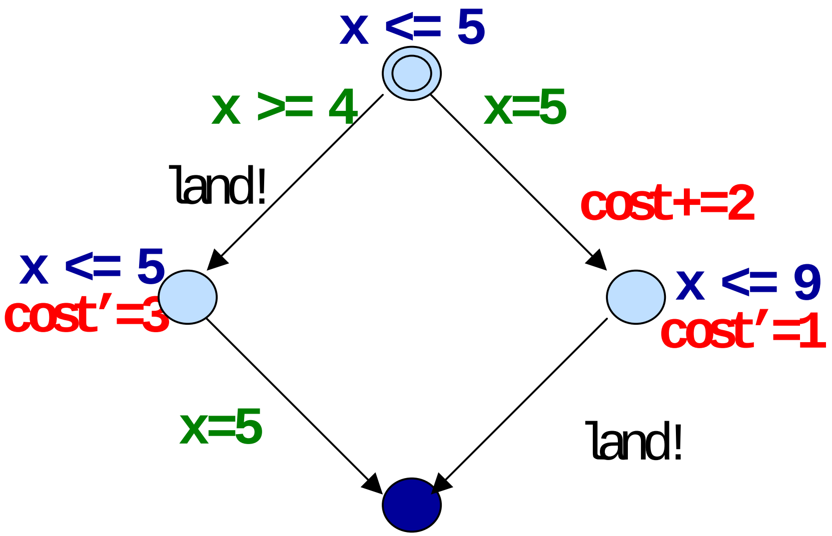


E	earliest landing time
T	target time
L	latest time
e	cost rate for being early
l	cost rate for being late
d	fixed cost for being late

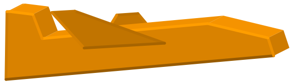


Planes have to keep separation distance to avoid turbulences caused by preceding planes

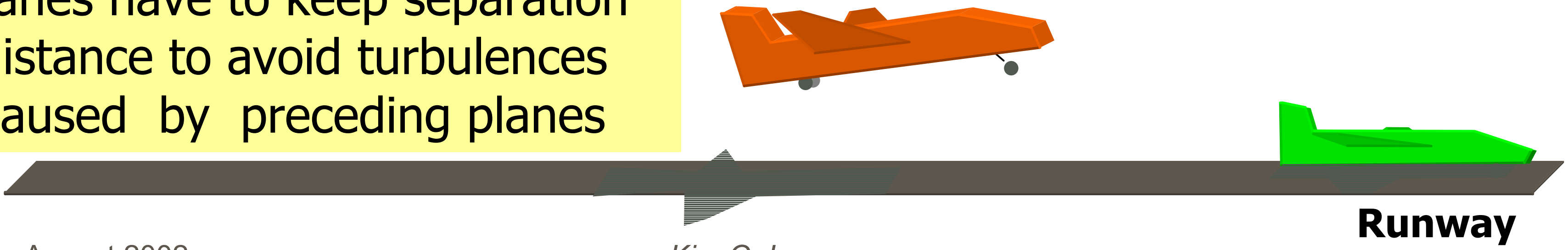
Example: Aircraft Landing



4 earliest landing time
5 target time
9 latest time
3 cost rate for being early
1 cost rate for being late
2 fixed cost for being late

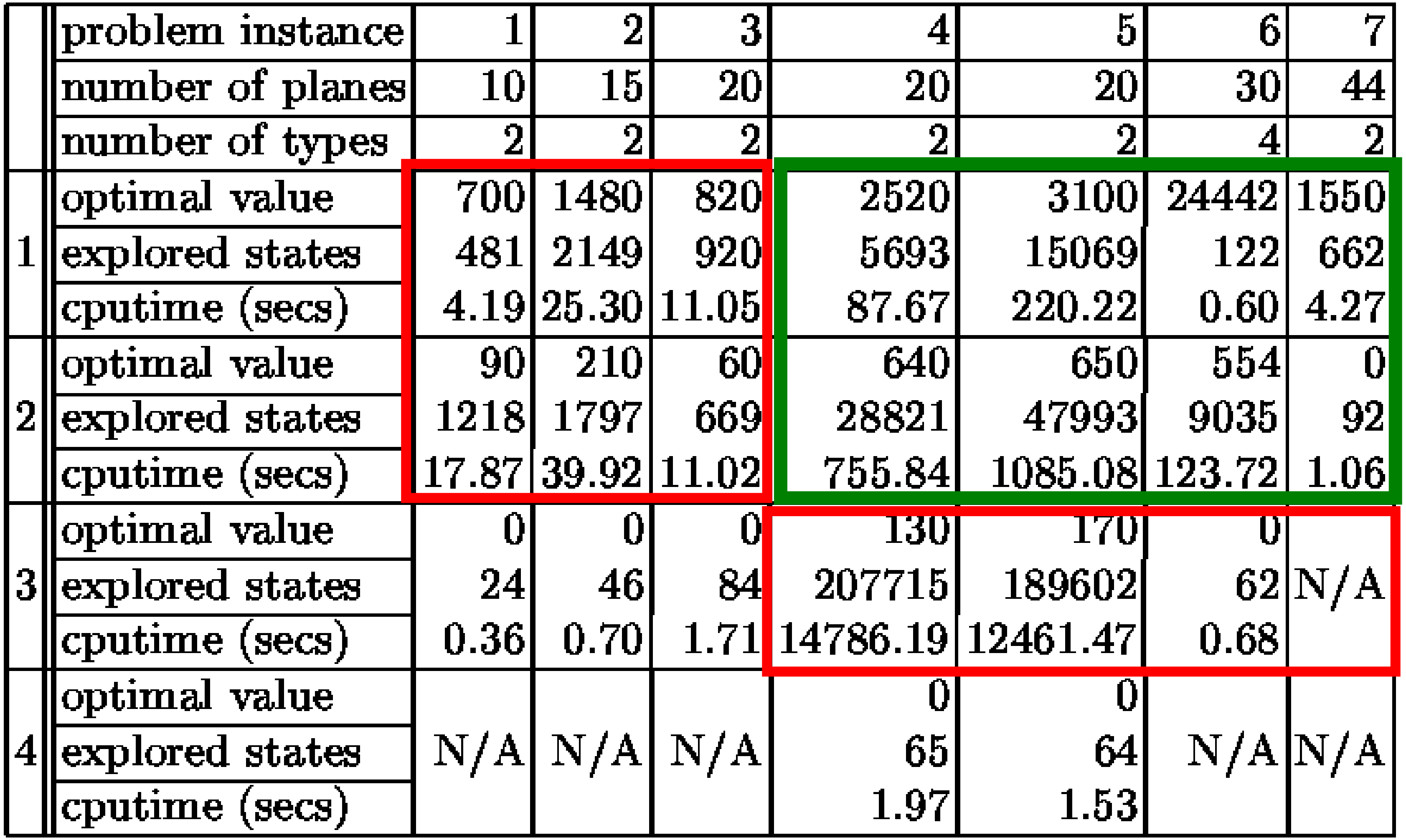


Planes have to keep separation distance to avoid turbulences caused by preceding planes



Aircraft Landing

Source of examples:
Baesley et al'2000



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

Future Work & Open Problems

- Datastructures for fully symbolic, efficient exploration
- Partial order reduction
- Exploitation of symmetries
- Distributed checking of full TCTL
- Efficient use of disk
- Extension of acceleration techniques
- Application of abstract interpretation
- ……

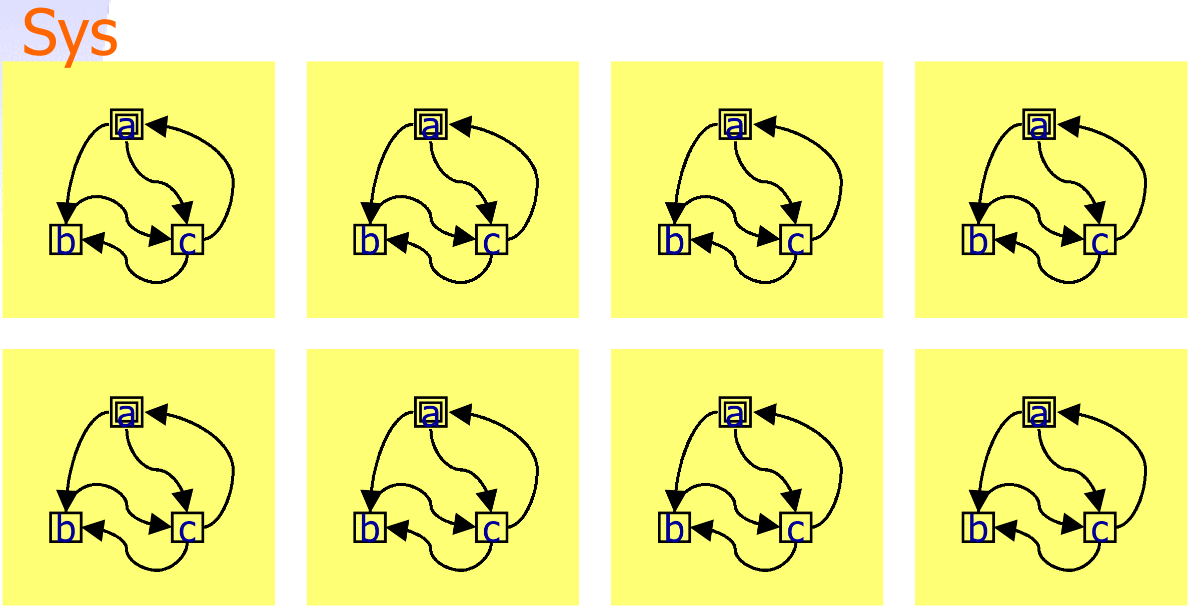
END

☺

Relevant Workshops

- RT-Tools affiliated with FLOC, July 2002
- MTCS: Models for Time-Criticial Systems, CONCUR, August 2002
- PDMC: Parallel and Distributed Model Checking, CONCUR, August 2002.

The State Explosion Problem

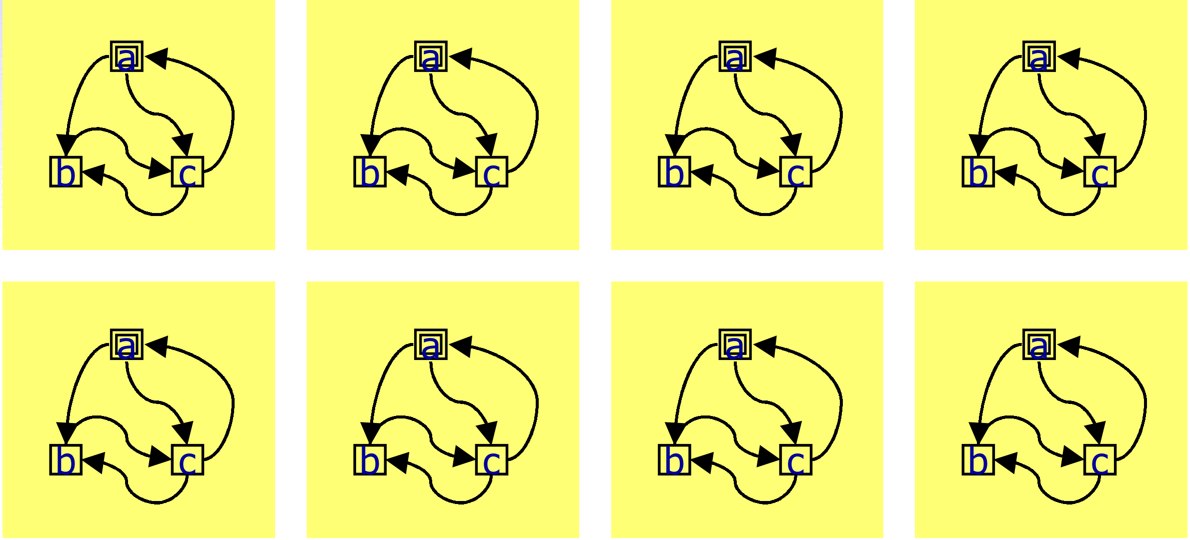


sat φ

Model-checking is either
EXPTIME-complete or PSPACE-complete
(for TA's this is true even for a single TA)

Abstraction

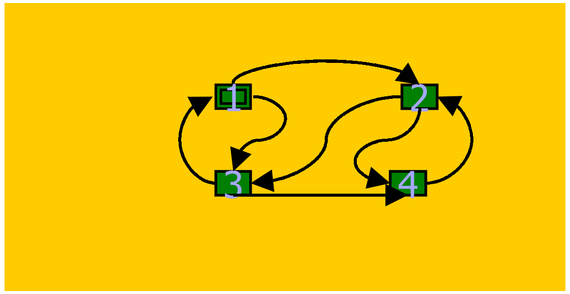
Sys



sat φ

REDUCE TO

Abs

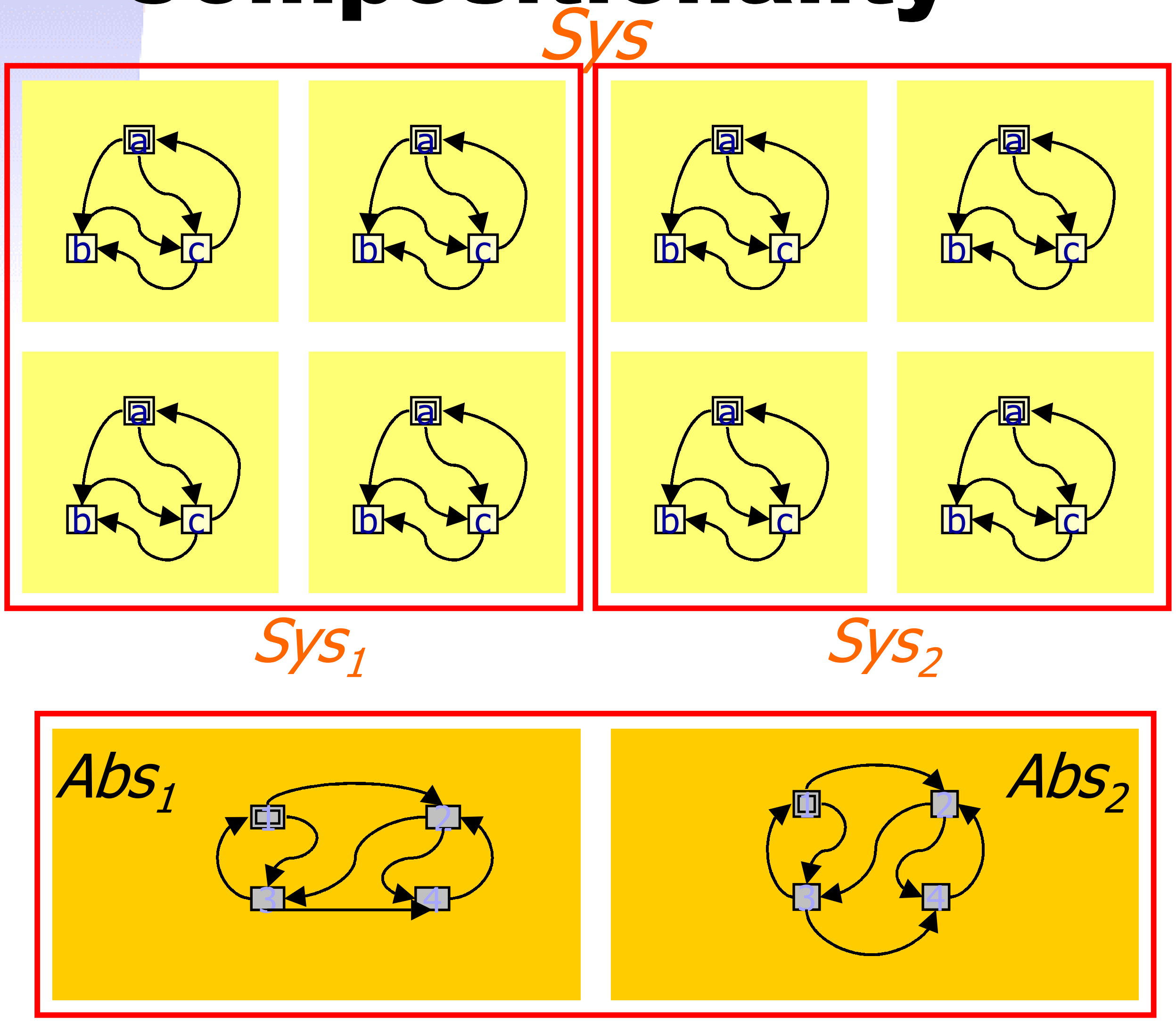


sat φ

Preserving safety properties

$$\boxed{\frac{Abs \text{ sat } \varphi \qquad Sys \leq Abs}{Sys \text{ sat } \varphi}}$$

Compositionality



$$\begin{array}{c} Sys_1 \leq Abs_1 \\ Sys_2 \leq Abs_2 \\ \hline Sys_1 \,|\, Sys_2 \leq Abs_1 \,|\, Abs_2 \end{array}$$

$$\begin{array}{c} Sys_1 \leq Abs_1 \\ Sys_2 \leq Abs_2 \\ Abs_1 \,|\, Abs_2 \leq Abs \\ \hline Sys \leq Abs \end{array}$$

Timed Simulation

$T_1 \leq T_2$ if there is a relation

$R \subseteq St_1 \times St_2$ s.t.

a) $(s_0, t_0) \in R$

b) Whenever $(s,t) \in R$ then

- if $s \rightarrow_a s'$ then
  $\exists t \Rightarrow_a t'$ st. $(s',t') \in R$
- if $s \rightarrow_{e(d)} s'$ then
  $\exists t \Rightarrow_{e(d)} t'$ st. $(s',t') \in R$

* $\leq$ preserves safety properties

* $\leq$ is preserved by parallel composition

* $\leq$ is decidable

* If T_2 is determistic then $T_1 \leq T_2$ may be reduced to a reachability question for $T_1 \,||\, Test(T_2)$

UPPAAL

UCb

Timed Simulation

$T_1 \leq T_2$ if the

$R \subseteq St_1 \times St_2$

a) $(s_0, t_0) \in R$

b) Whenever

- if $s \rightarrow_a$

$\exists t \Rightarrow$

- if $s \rightarrow_{e(}$

$\exists t \Rightarrow_{e(d)}$

rties

composition

$T_1 \leq T_2$

chability

for $T_1 \,||\, Test(T_2)$

Applied to

IEEE 1394a Root contention protocol
(Simons, Stoelinga)

B&O Power Down Protocol
(Ejersbo, Larsen, Skou, FTRTFT2k)

Modifications identified when urgency and shared integers

UPPAAL