Timed and Hybrid Systems

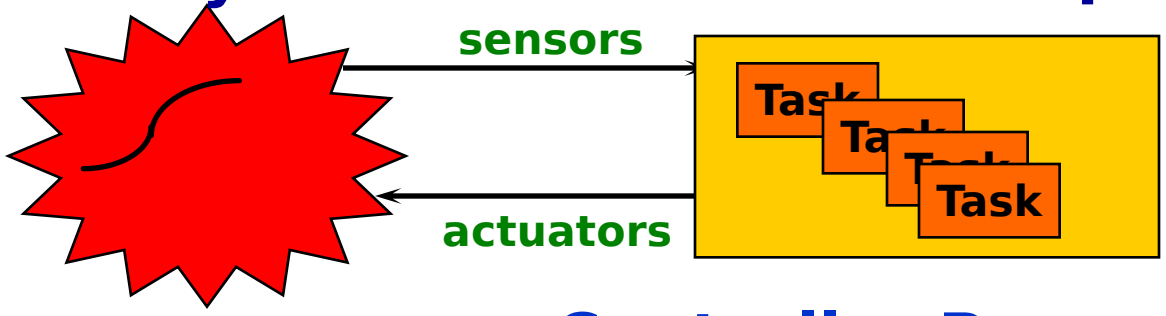
in UPPAAL2k

Kim Guldstrand Larsen
BRICS@Aalborg

Paul Pettersson
BRICS@Aalborg & DoCS@Uppsala

Hybrid & Real Time Systems

Control Theory **Computer Science**



Plant
Continuous

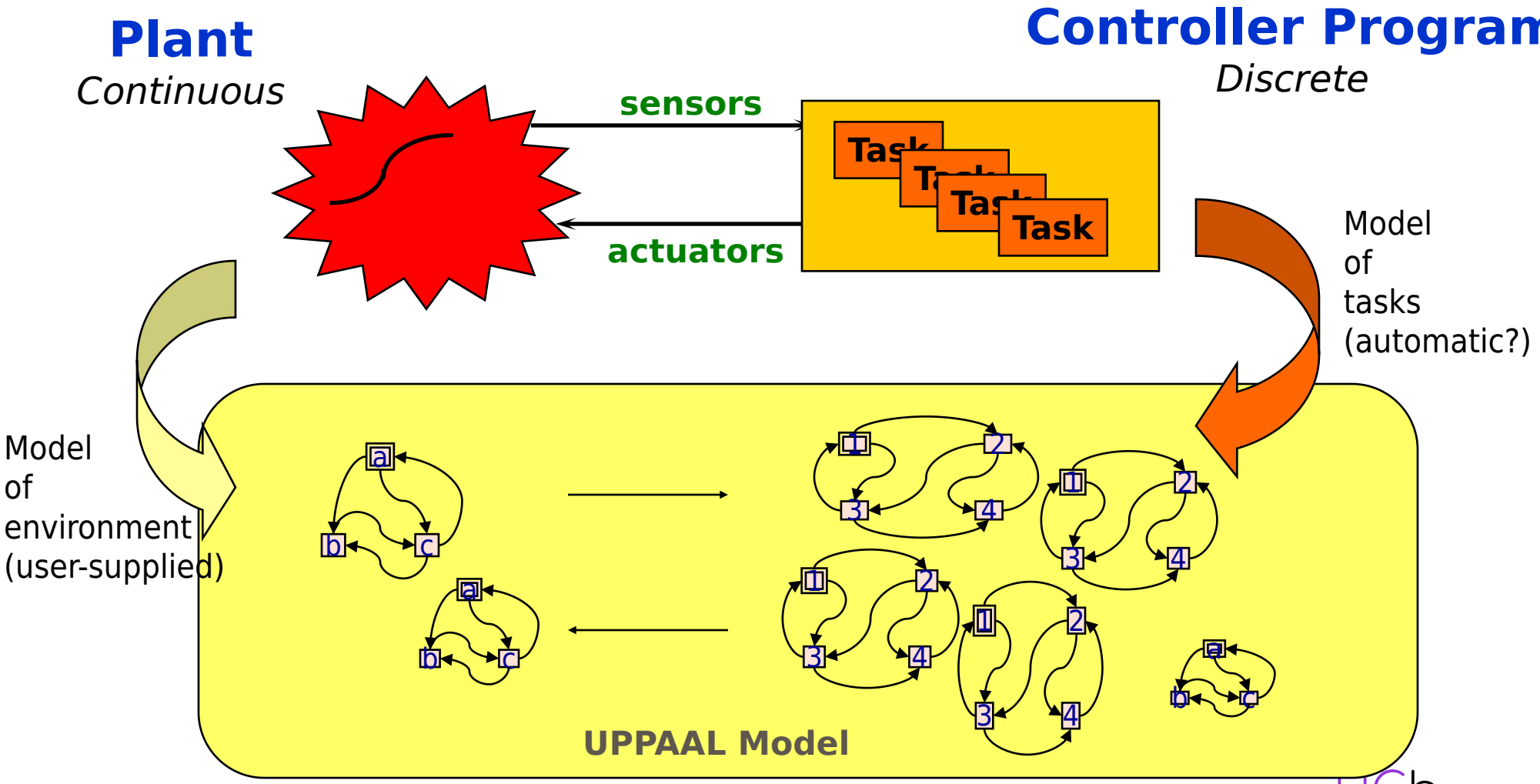
Controller Program
Discrete

Eg.: Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

Real Time System
A system where correctness not only depends on the logical order of events but also on their timing!!

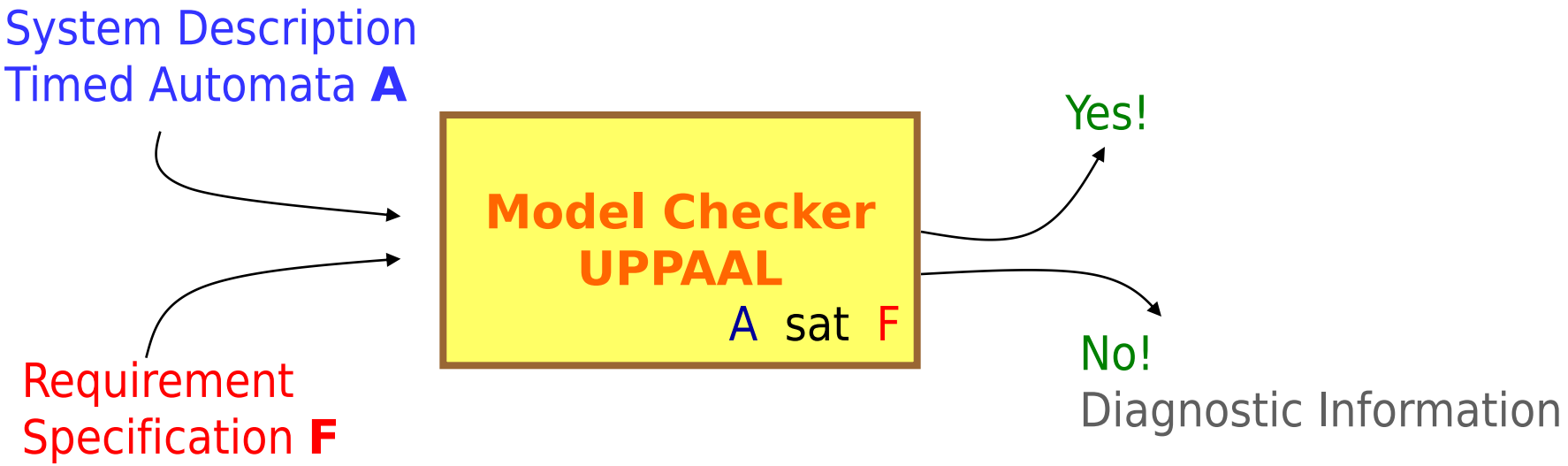
Validation & Verification

Construction of UPPAAL models



Real Timed Model Checking

History



89 90 93 94 95 97 98 99

Timed Automata
Decidability

Epsilon
TAB
Regions

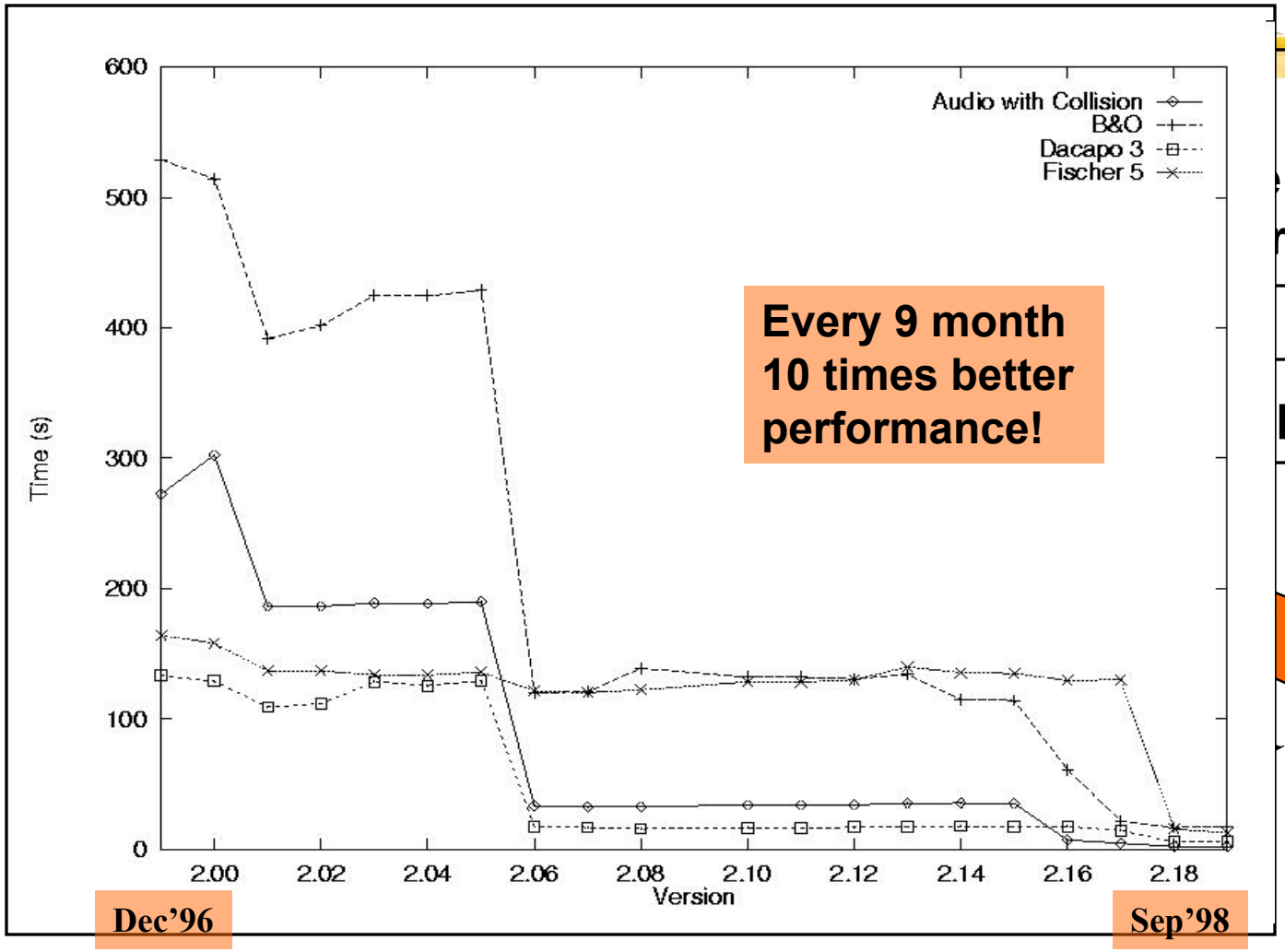
HyTech
Polyhedra

UPPAAL
Kronos
Zones, DBM

Minimal
Constraints

UPPAAL2k
CDDs

UPPAAL 1995 - 2000



with
r/Validator

L 2.17

Collaborators

@UPPsala

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

@AALborg

- Kim G Larsen
- Arne Skou
- Paul Pettersson
- Carsten Weise
- Kåre J Kristoffersen
- Gerd Behrman
- Thomas Hune

@Elsewhere

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

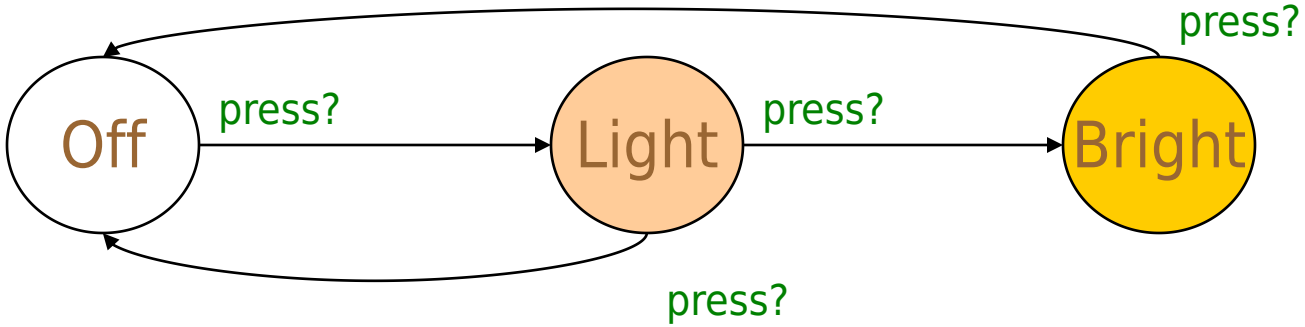
Overview of Tutorial

- Timed Automata
- UPPAAL and LEGO MINDSTORM
- The UPPAAL Engine
  - Symbolic Reachability
  - Verification Options
- Applications/Case Studies
  - SIDMAR Steel Production Plant (VHS project)

Alur, Dill 1990

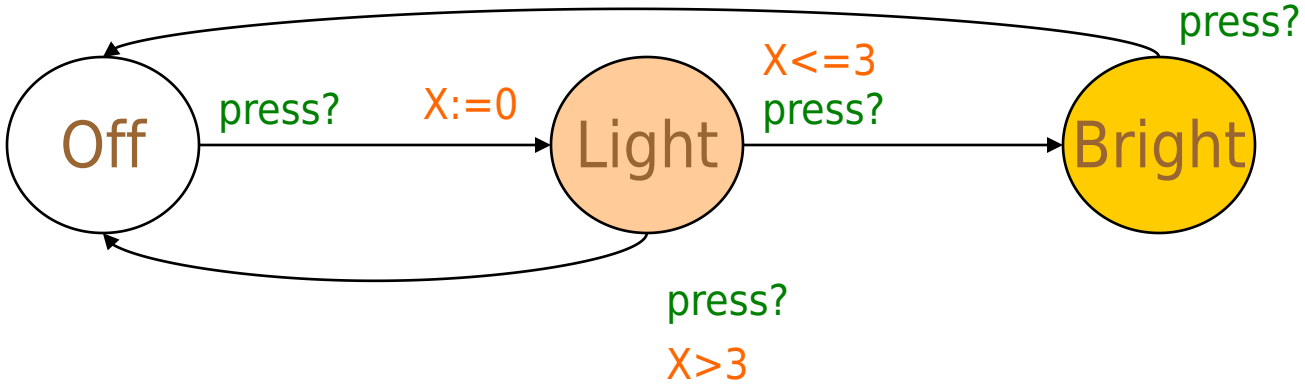
TIMED AUTOMATA

Intelligent Light Control



WANT: if press is issued twice quickly then the light will get brighter; otherwise the light is turned off.

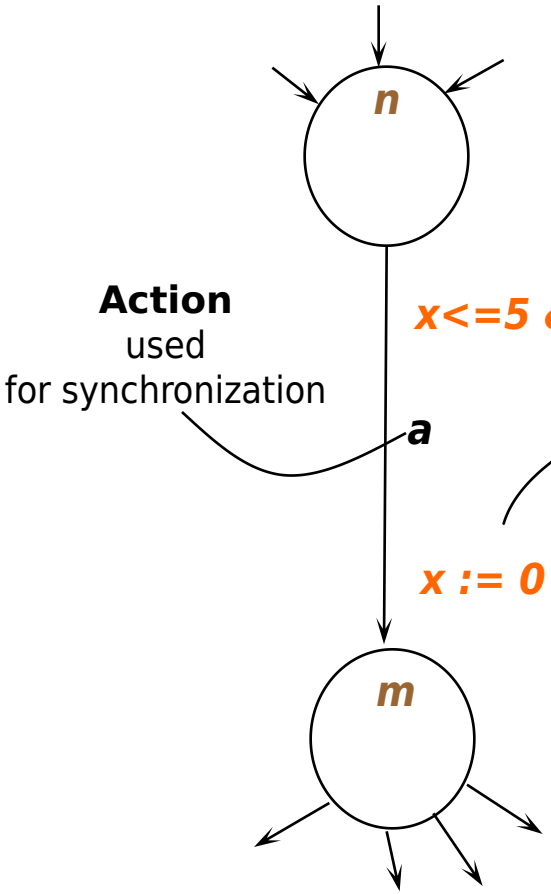
Intelligent Light Control



Solution: Add real-valued clock x

Timed Automata

Alur & Dill 1990



Clocks: *x, y*

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 R

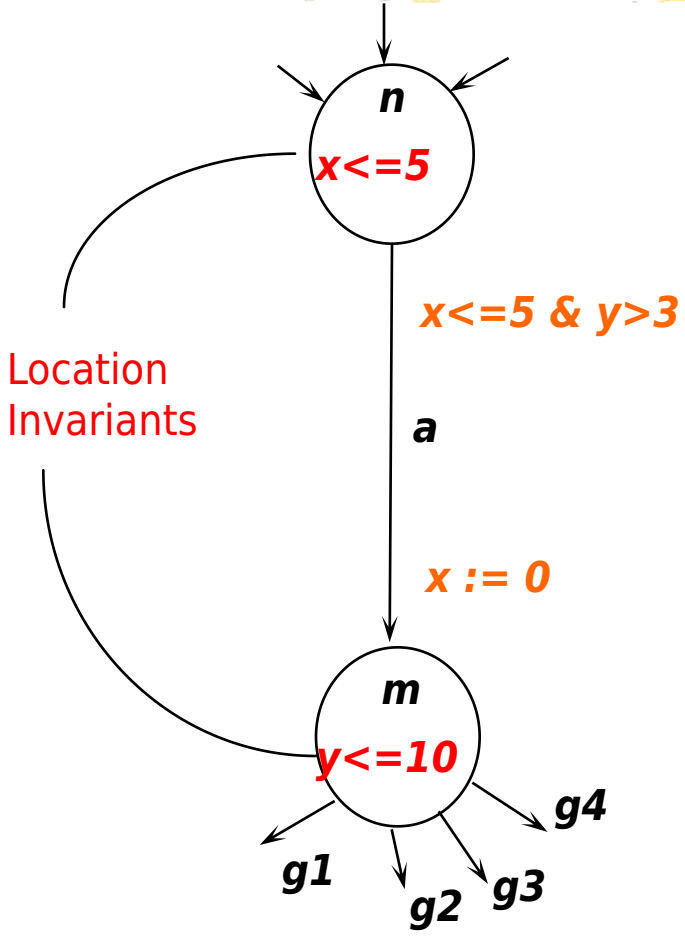
Transitions

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

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

Timed Automata -

Invariants



Clocks: *x, y*

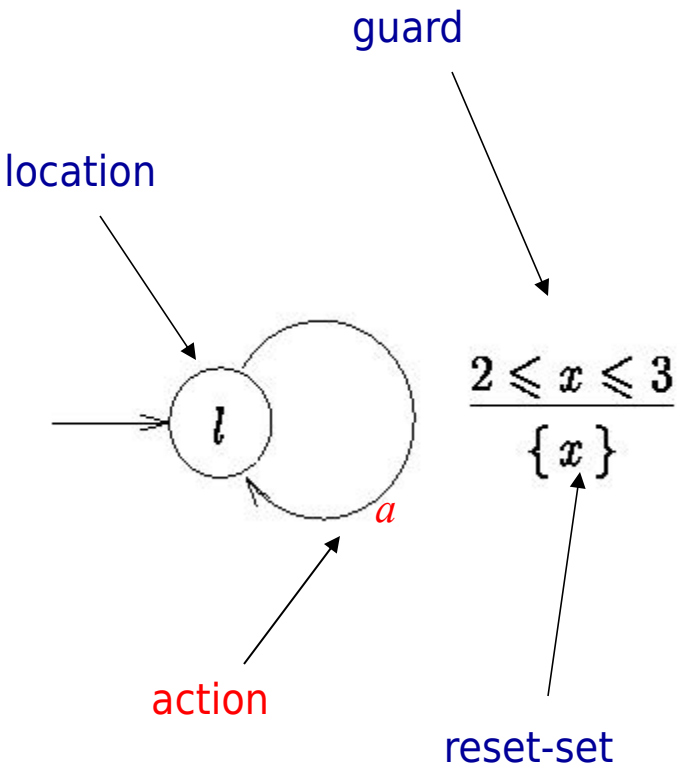
Transitions

(***n*** , ***x***=2.4 , ***y***=3.1415) —*e(3.2)*→ ✗

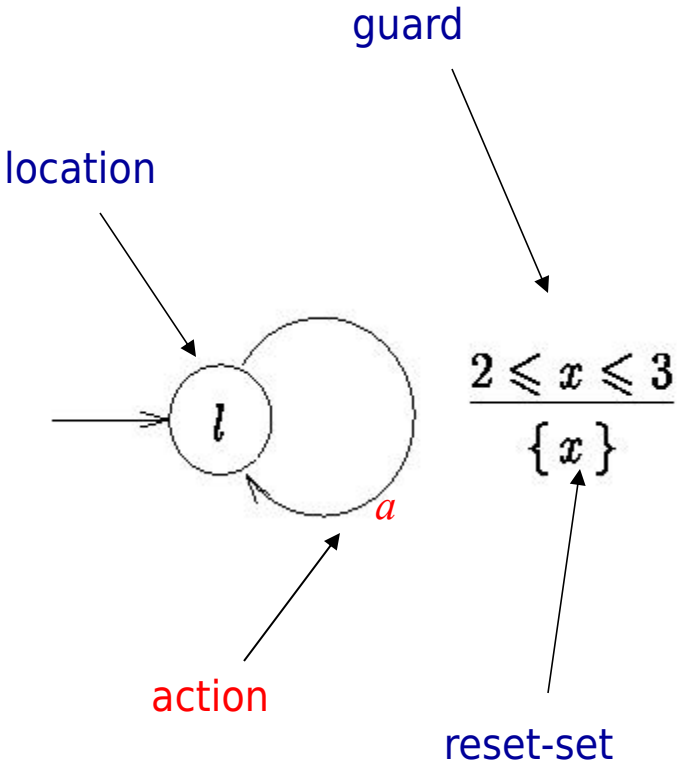
(***n*** , ***x***=2.4 , ***y***=3.1415) —*e(1.1)*→ (***n*** , ***x***=3.5 , ***y***=4.2415)

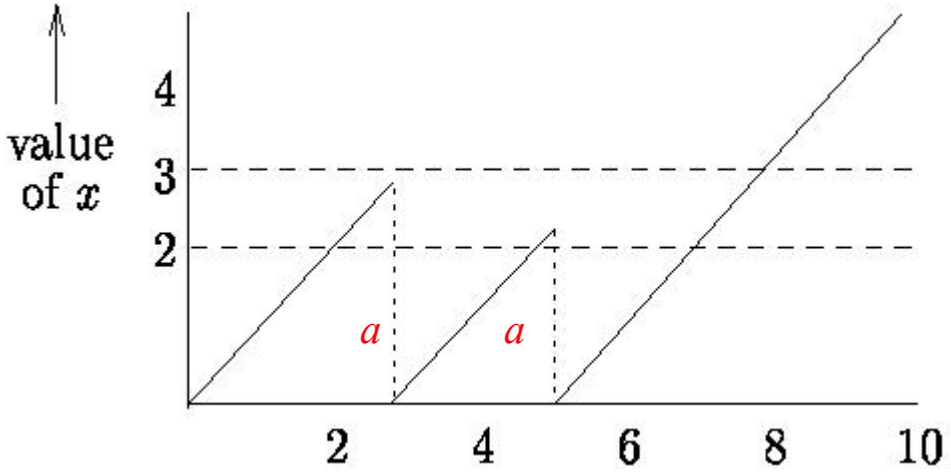
Invariants insure progress!!

Timed Automata: Example

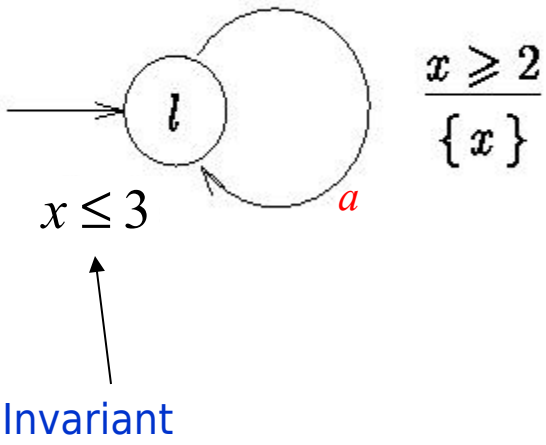


Timed Automata: Example

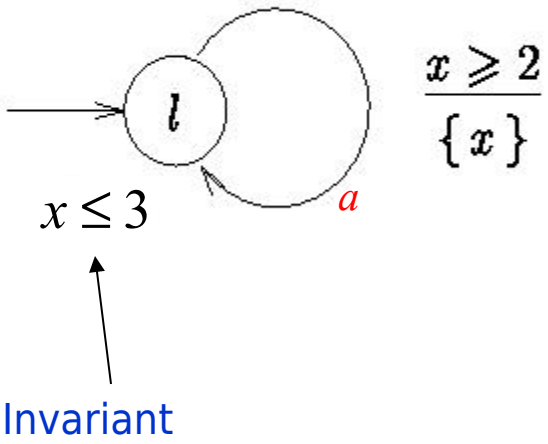


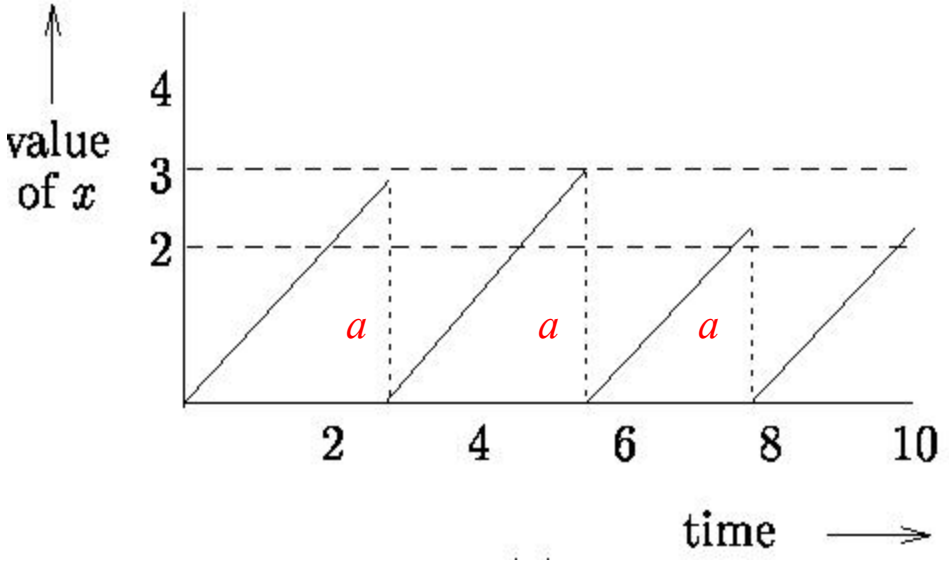


Timed Automata: Example



Timed Automata: Example





Fundamental Results

PSPACE-c

- Reachability **Alur, Dill**
- Trace-inclusion **Alur, Dill**
  - Timed ; Untimed
- Bisimulation
  - Timed **Cerans** ; Untimed
- Model-checking
  - TCTL, T_{mu}, L_{nu},...

PSPACE-c / EXPTIME-c

Updatable Timed Automata

Patricia Bouyer, Catherine Dufourd,
Emmanuel Fleury, Antoine Petit

	Diagonal-free	General
$x := c,\ x := y$	Pspace complete	Pspace complete
$x := x + 1$		Undecidable
$x := y + c$		
$x := x - 1$	Undecidable	
$x :< c,\ x :\leq c$	Pspace complete	Pspace complete
$x :> c,\ x :\geq c$		Undecidable
$x :\sim y + c$		
$(y+)c <: x :< (y+)d$		
$y + c <: x :< z + d$	Undecidable	

With $\sim \in \{<, \leq, \geq, >\}$ and $c, d \in \mathbb{Q}_+$

	Diagonal-free	General
$:= c,\ x := y$	TA-bisimilar	TA-bisimilar
$x := x + 1$		Turing
$x := y + c$		
$:< c,\ x :\leq c$	TA_ϵ	TA_ϵ
$x :> c,\ x :\geq c$		Turing
$x :\sim y + c$		
$(y+)c <: x :< (y+)d$		

With $\sim \in \{<, \leq, \geq, >\}$ and $c, d \in \mathbb{Q}_+$

Other Extensions

- Ordinary clocks ..... ***x* rate** 1
- Integer variables .... ***x* rate** 0

- Stopwatches ..... ***x* rate** 0 or ***x* rate** 1 (*loc.dep.*)

Cassez, Larsen

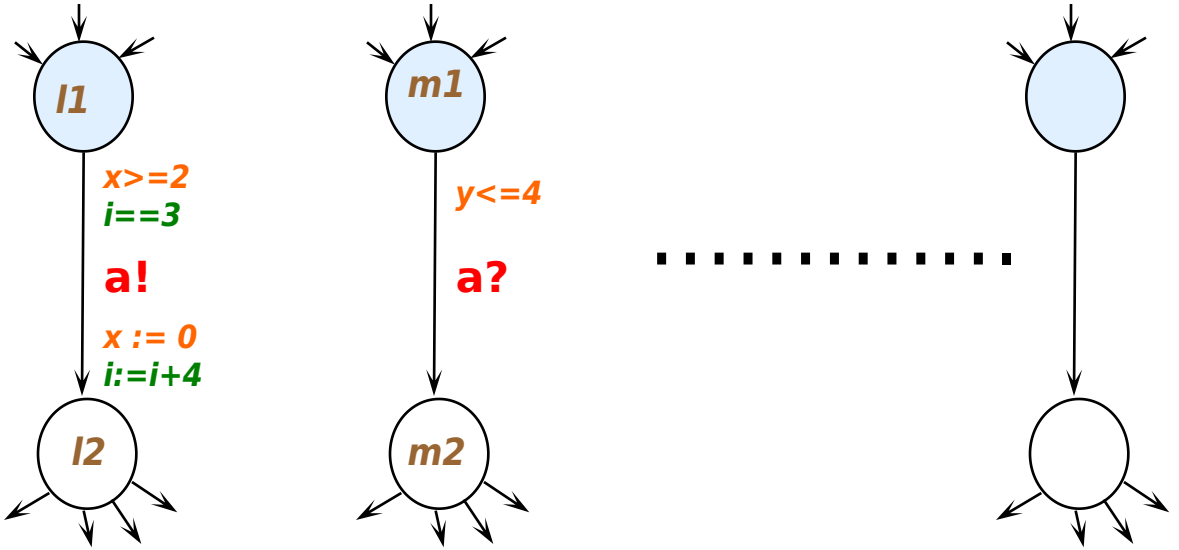
- Const. slope clocks .. ***x* rate** n where n is in Nat
- Parameters .... ***x* rate** 0 (and NOT assignable)
- Multirate clocks

  Lin. Hyb. Aut. ..... ***x* rate** [l,u] where l,u is in Nat
  linear guards & linear asgn.

HyTech

The UPPAAL Model

= Networks of Timed Automata + Integer Variables +....



Two-way synchronization on *complementary* actions.

Closed Systems!

Example transitions

(*l1*, *m1*,....., x=2, y=3.5, i=3,.....) —tau→ (*l2*,*m2*,.....,x=0, y=3.5, i=7,.....)

—0.2→ (*l1*,*m1*,.....,x=2.2, y=3.7, I=3,.....)

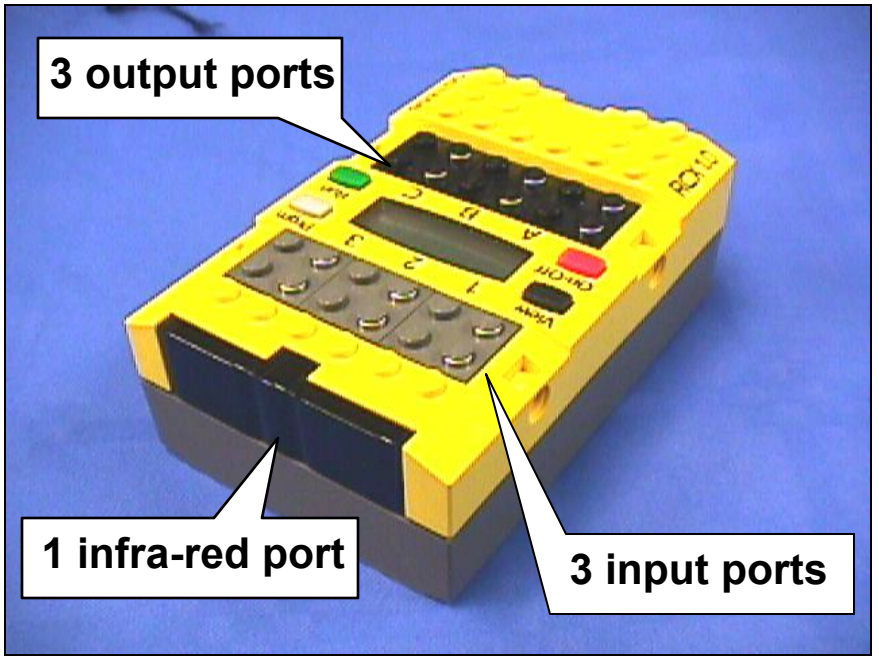
If **a** URGENT CHANNEL

UPPAAL &
LEGO MINDSTORM

DEMO

LEGO Mindstorms/RCX

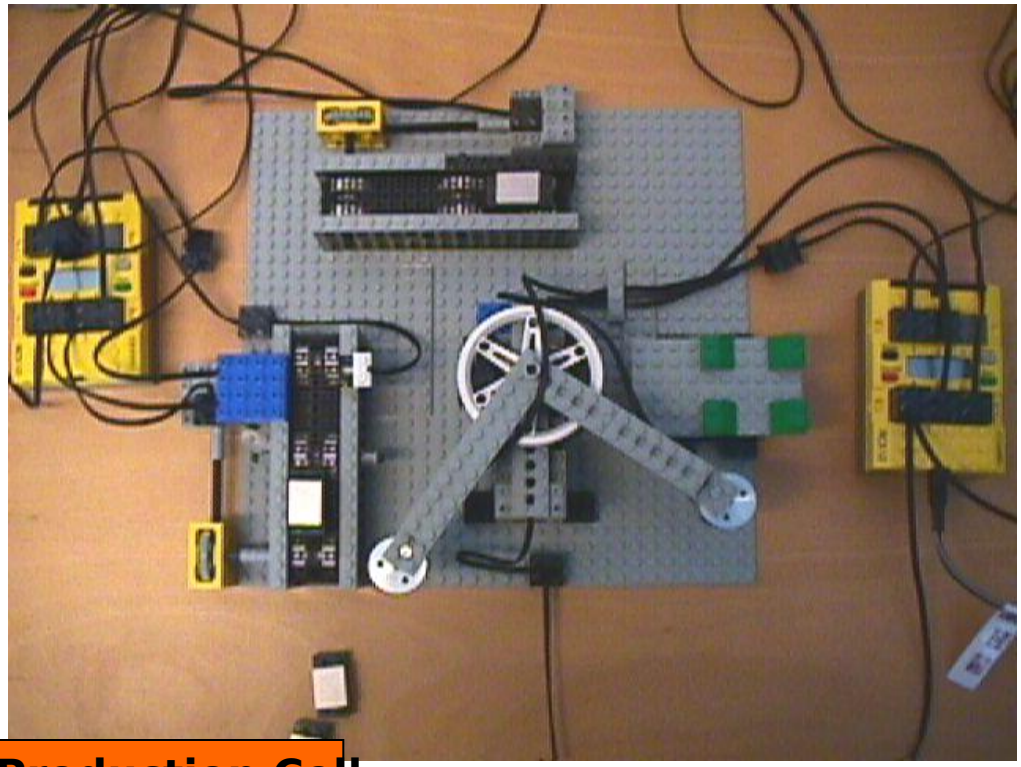
- Sensors: temperature, light, rotation, pressure.
- Actuators: motors, lamps,
- Virtual machine:
  - 10 tasks, 4 timers, 16 integers.



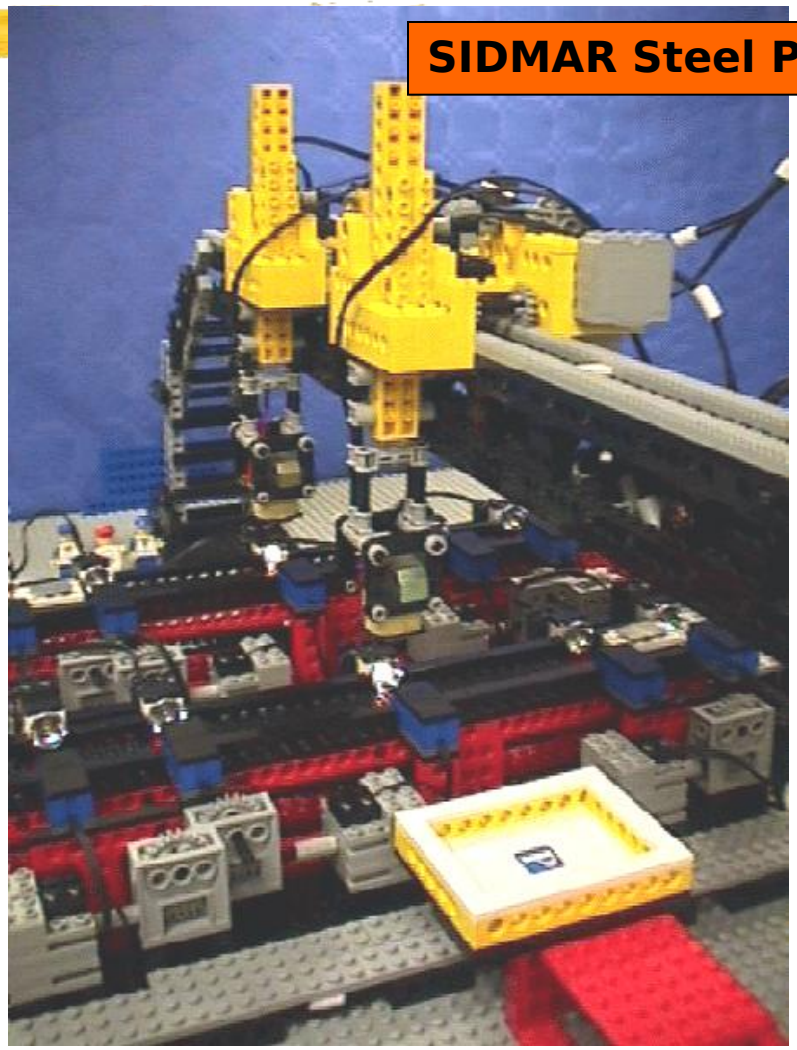
- Several Programming Languages:
  - NotQuiteC, Mindstorm, Robotics, legOS, etc.

LEGO Mindstorms/RCX

- LEGO bricks!!!



Production Cell

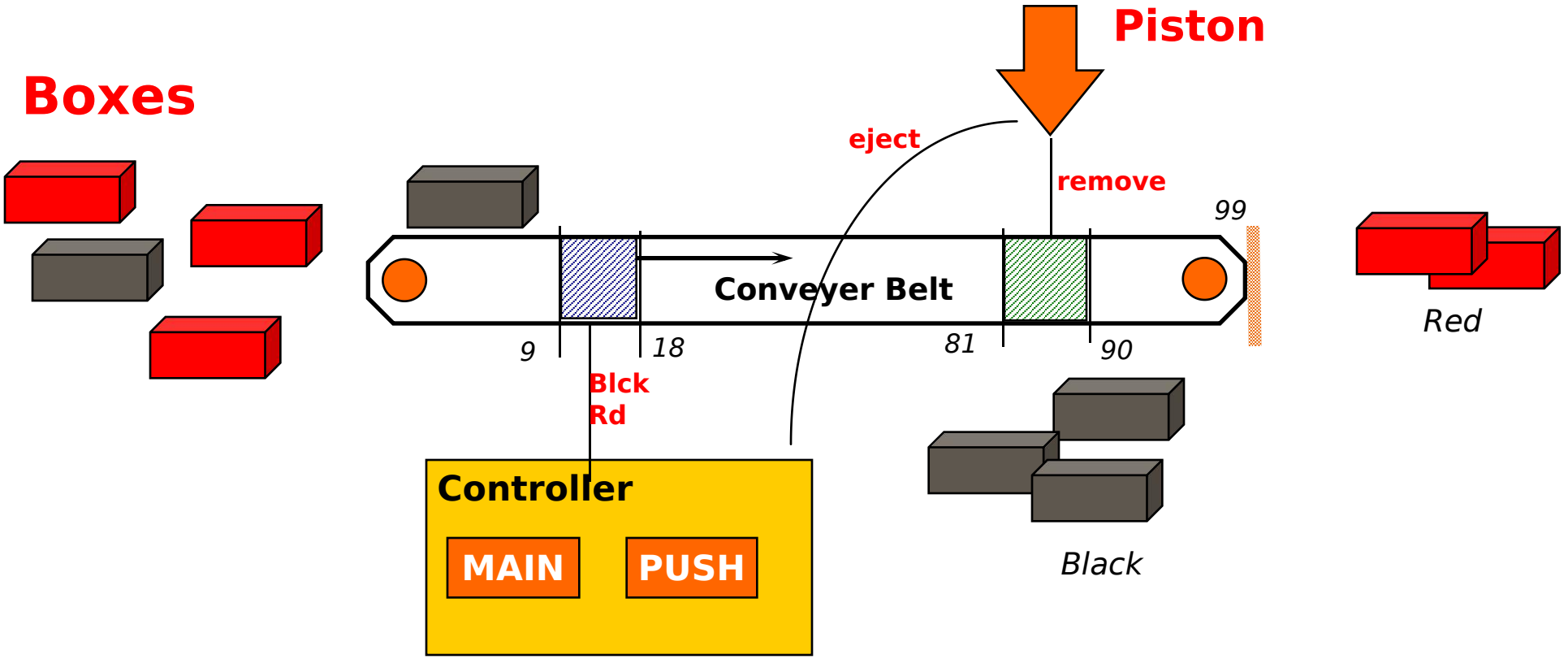


SIDMAR Steel Plant

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



Exercise: Design **Controller** so that only black boxes are being pushed out

NQC programs

```
int active;
int DELAY;
int LIGHT_LEVEL;
```

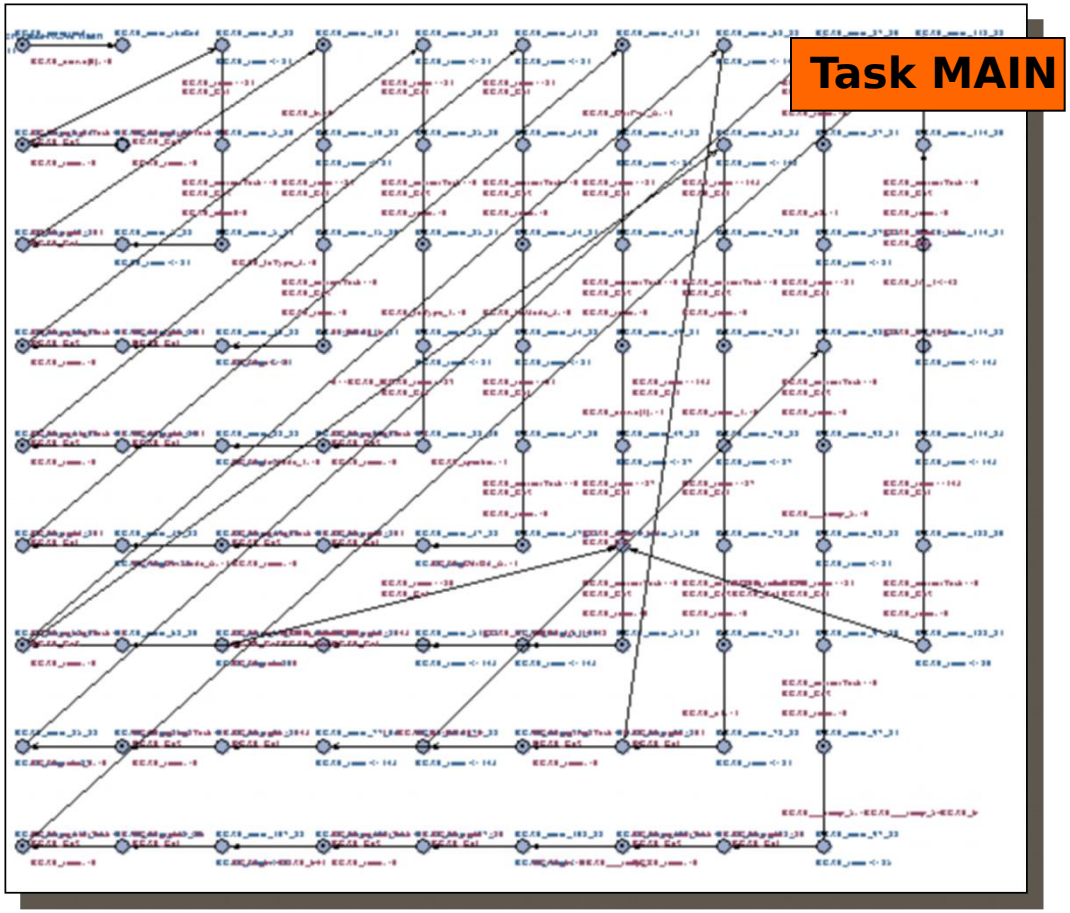
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 start PUSH;

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

```
task PUSH{
  while(true){
    wait(Timer(1)>DELAY && active==1);
    active=0;
    Rev(OUT_C,1);
    Sleep(8);
    Fwd(OUT_C,1);
    Sleep(12);
    Off(OUT_C);
  }
}
```

UPPAAL Demo

From RCX to UPPAAL

- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000 (yesterday) in Stockholm.

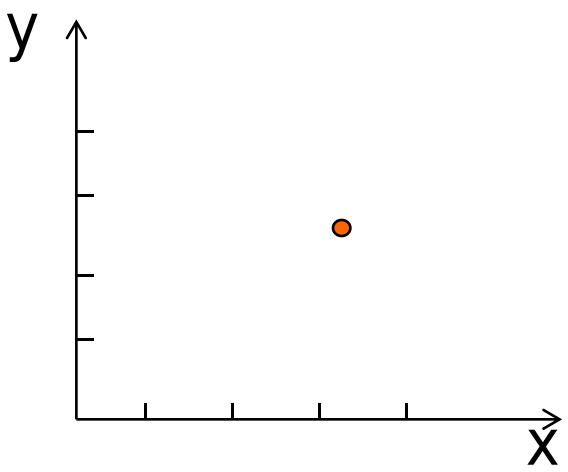


THE UPPAAL ENGINE

Symbolic Reachability Checking

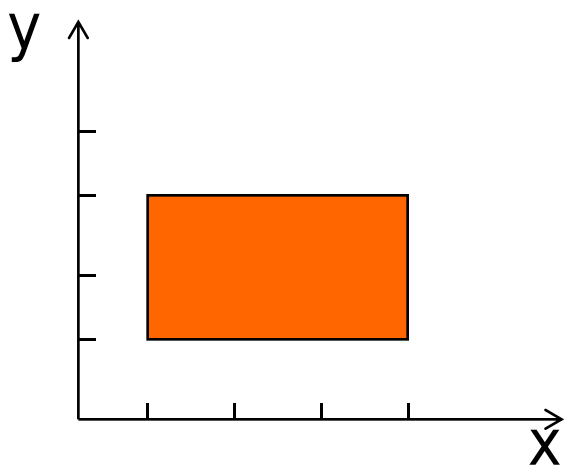
Zones
From infinite to finite

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

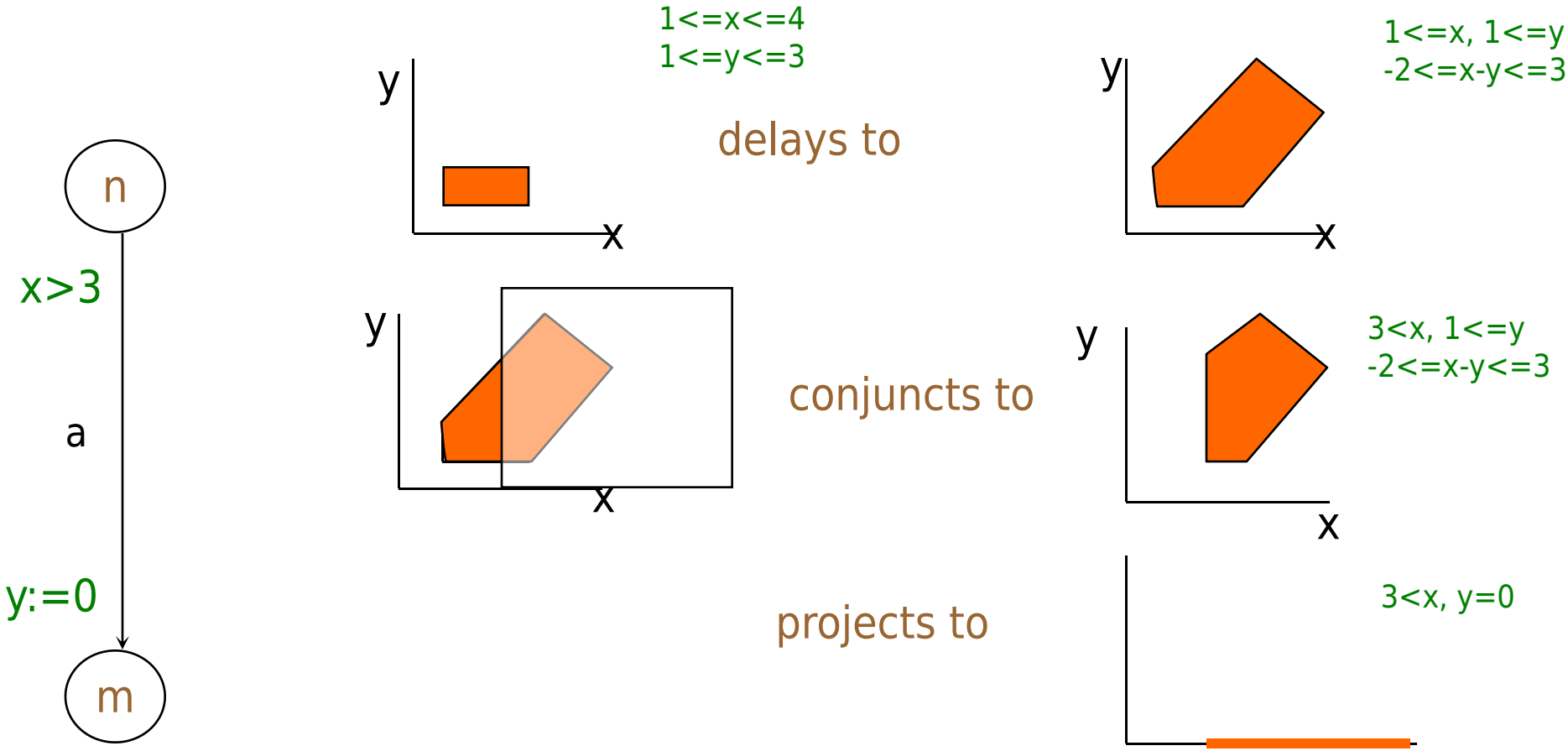


Symbolic state (set)
(n, 1≤x≤4,1≤y≤3)

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



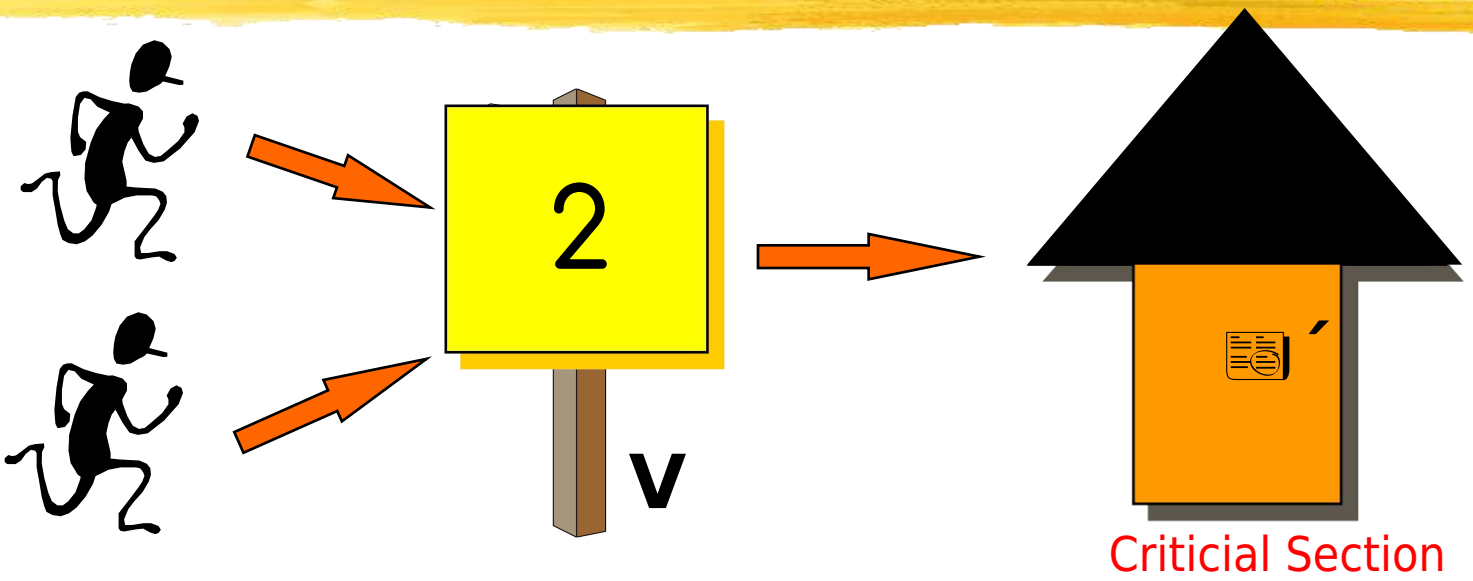
Symbolic Transitions



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

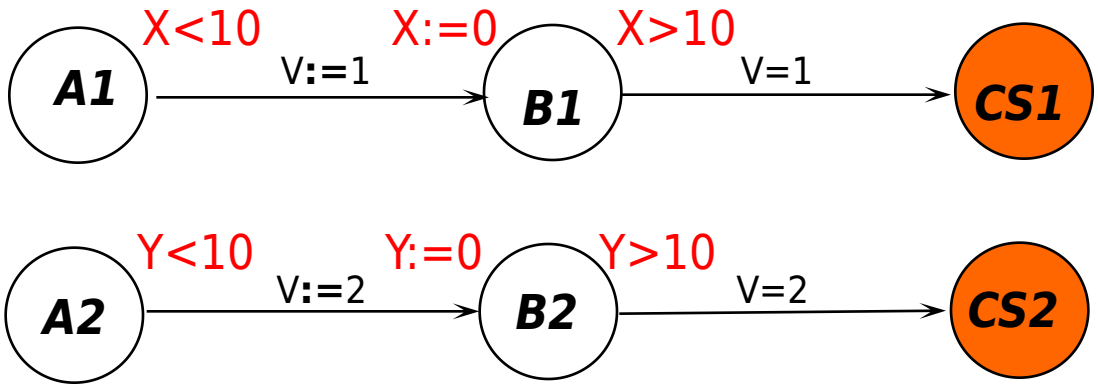
Fischer's Protocol

analysis using zones

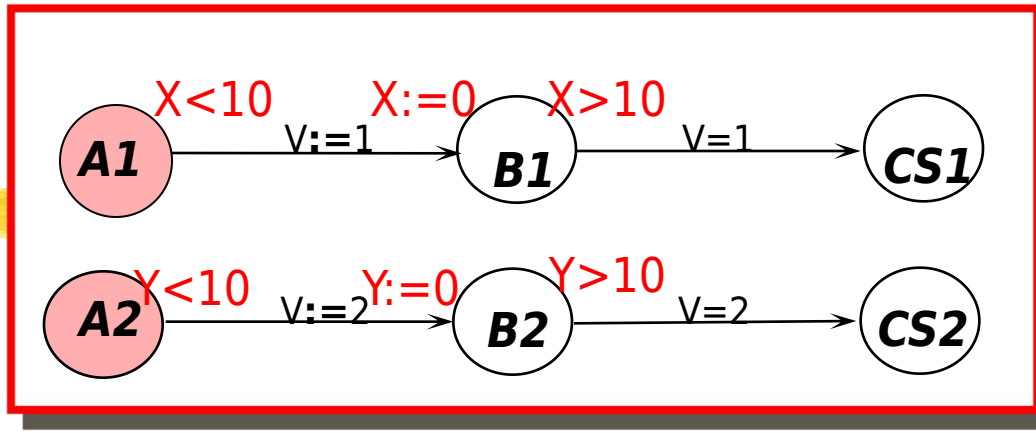


Init

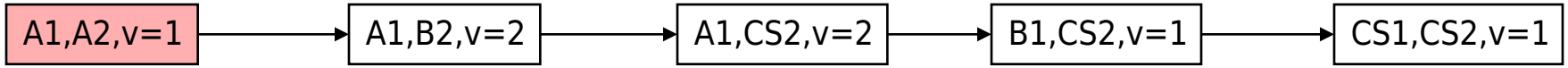
V=1



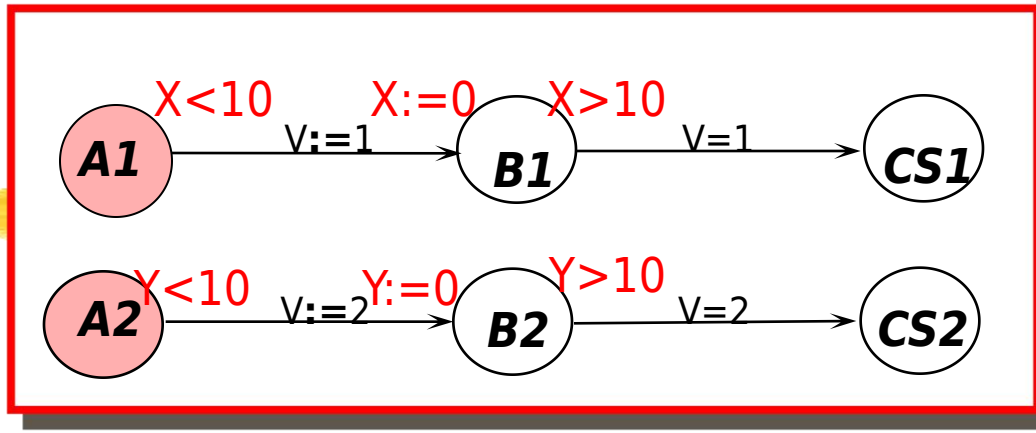
Fischers cont.



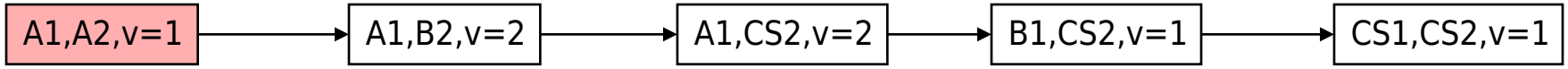
Untimed case



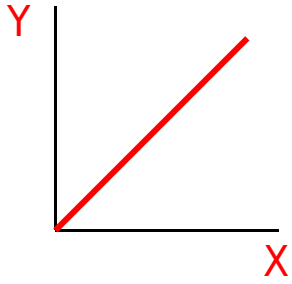
Fischers cont.



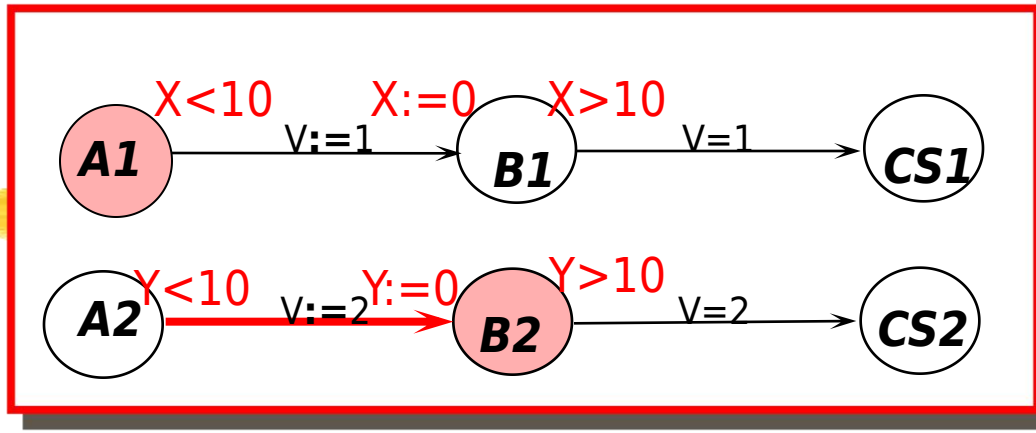
Untimed case



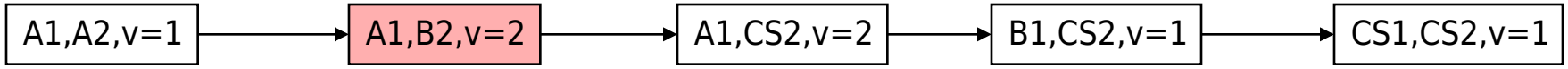
Taking time into account



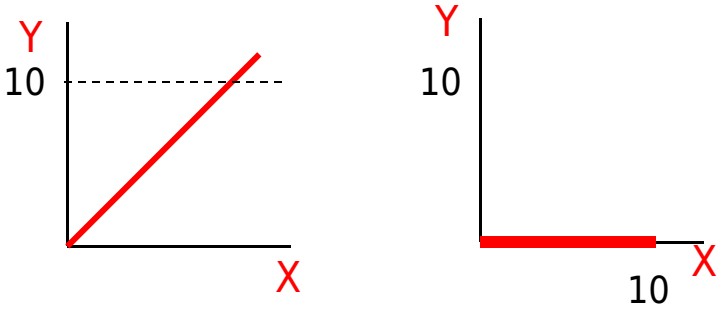
Fischers cont.



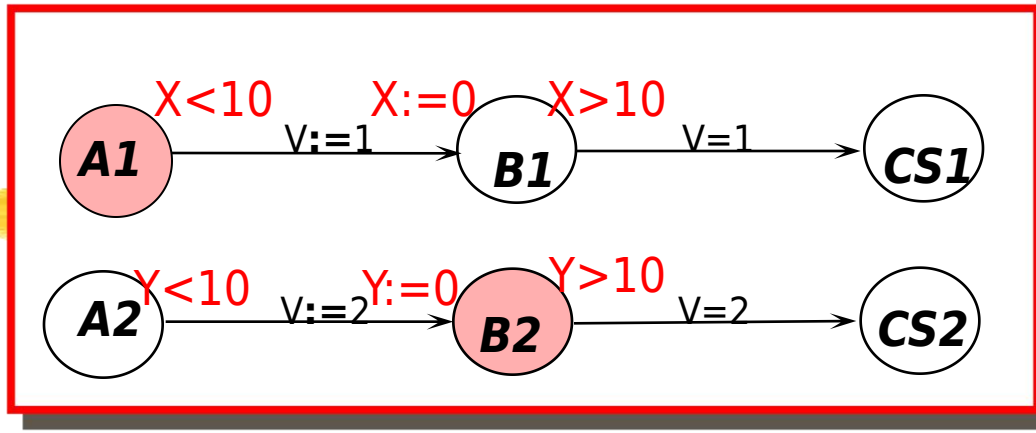
Untimed case



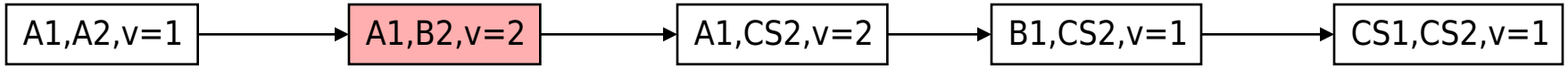
Taking time into account



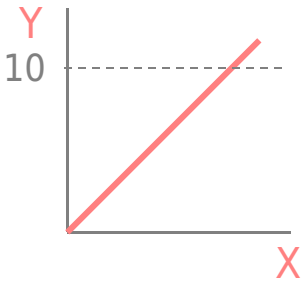
Fischers cont.

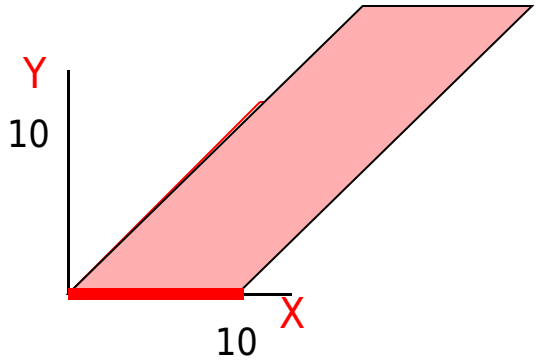


Untimed case

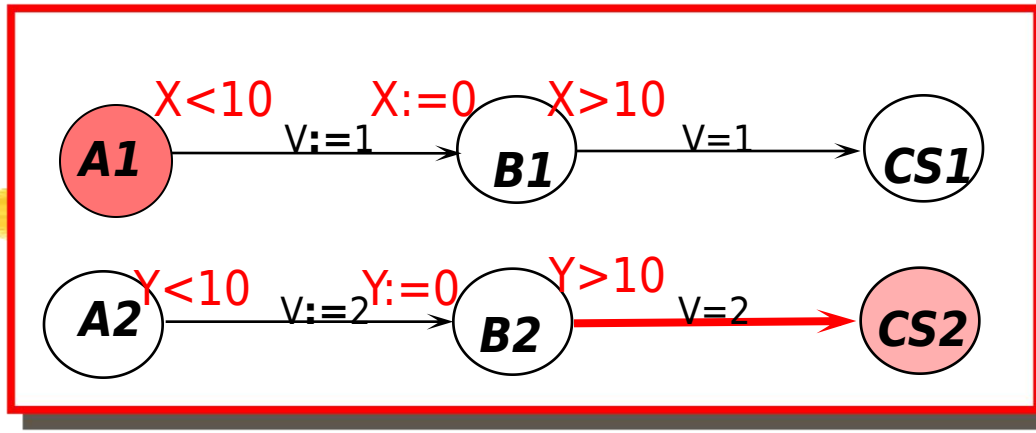


Taking time into account

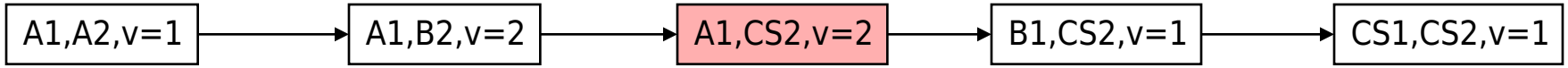




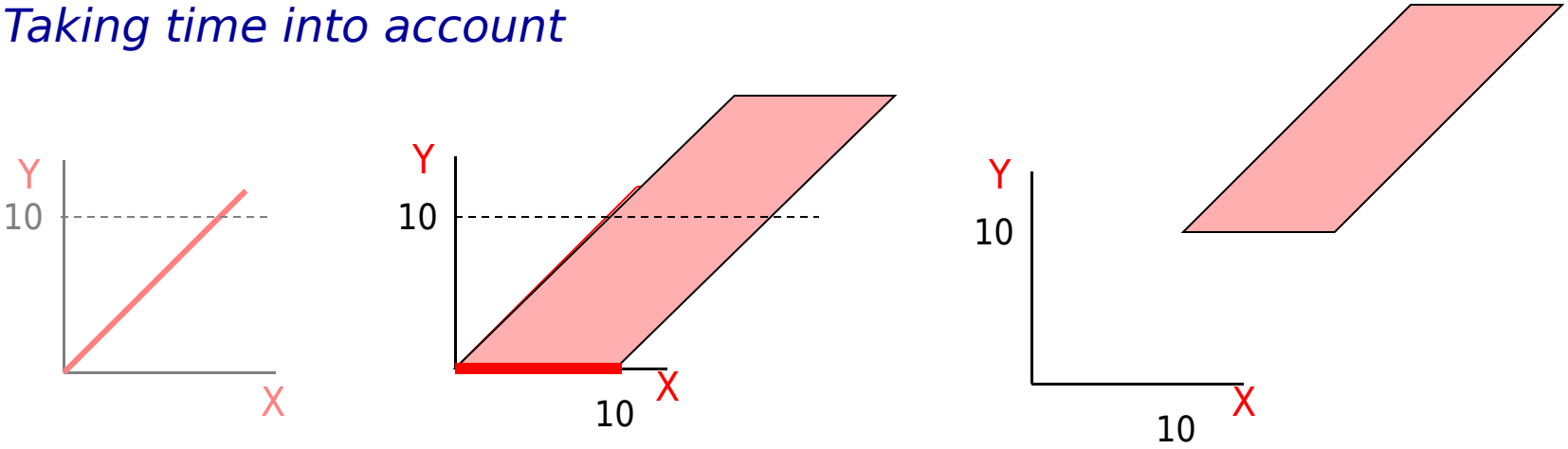
Fischers cont.



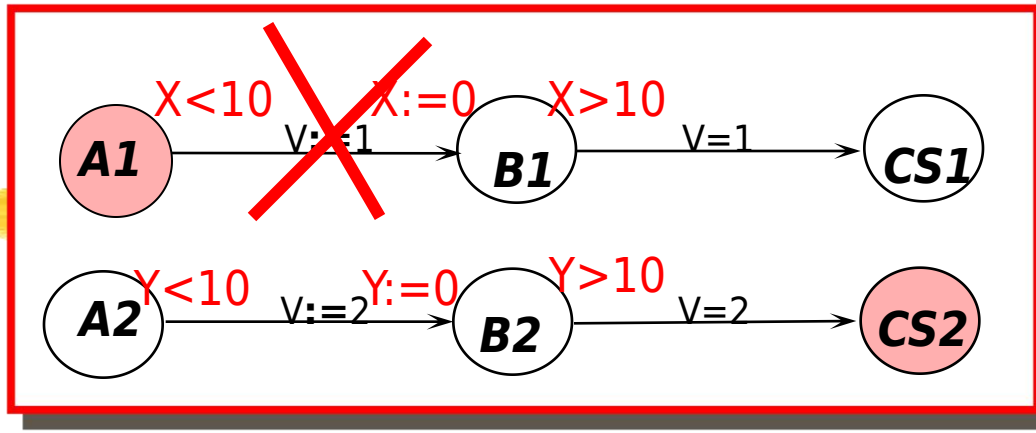
Untimed case



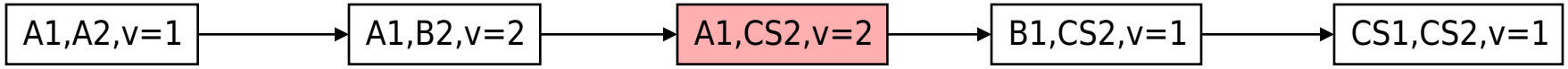
Taking time into account



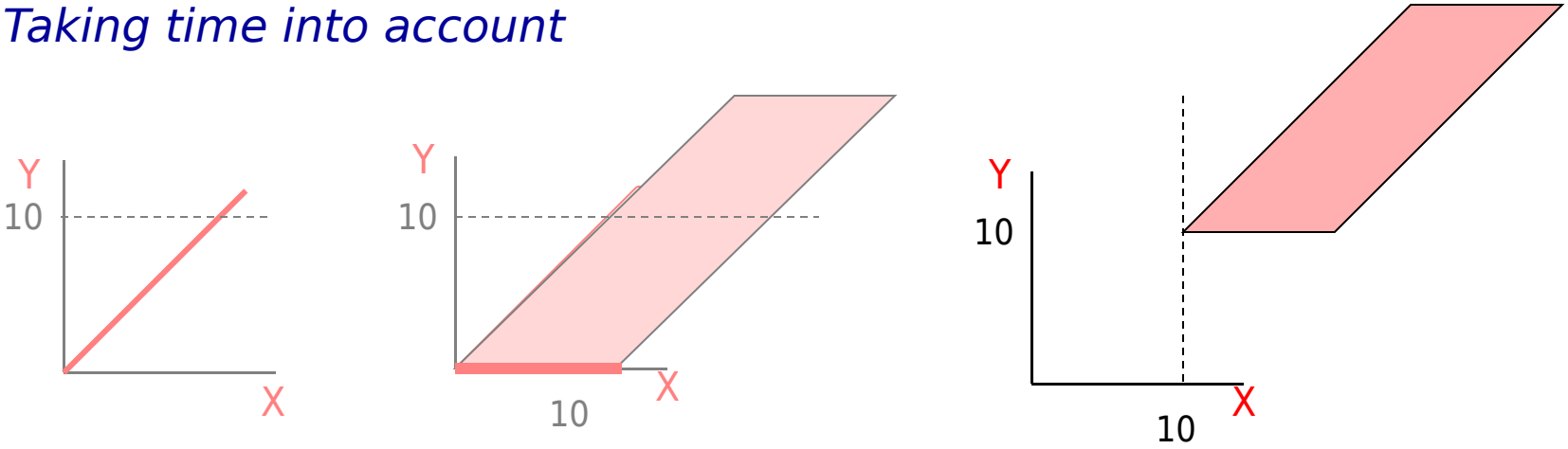
Fischers cont.



Untimed case



Taking time into account



Canonical Dastructures for Zones

Difference Bounded Matrices

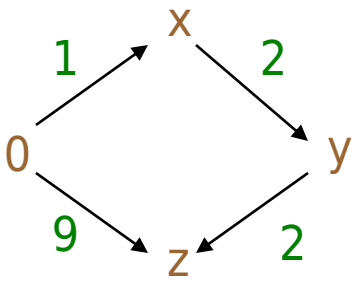
Bellman 1958, Dill 1989

Inclusion

D1

x<=1
y-x<=2
z-y<=2
z<=9

Graph

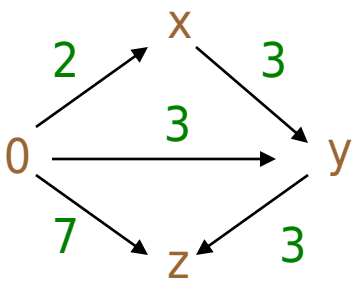


? ⊆ ?

D2

x<=2
y-x<=3
y<=3
z-y<=3
z<=7

Graph



Canonical Dastructures for Zones

Difference Bounded Matrices

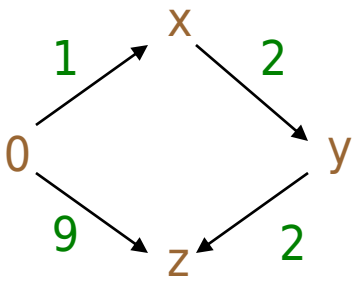
Bellman 1958, Dill 1989

Inclusion

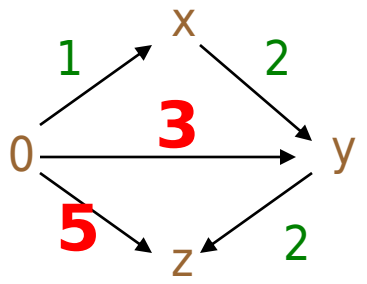
D1

x<=1
y-x<=2
z-y<=2
z<=9

Graph



Shortest Path Closure

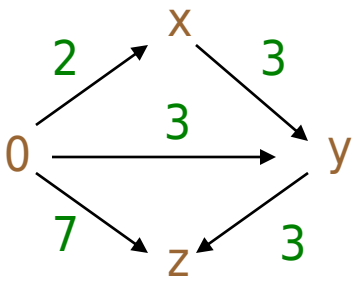


? ⊆ ?

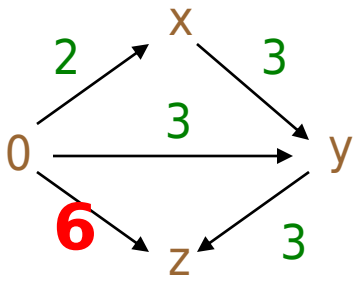
D2

x<=2
y-x<=3
y<=3
z-y<=3
z<=7

Graph



Shortest Path Closure



Canonical Dastructures for Zones

Difference Bounded Matrices

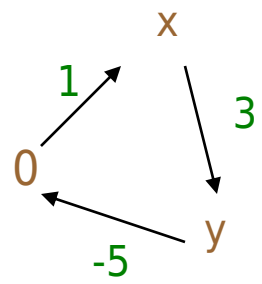
Bellman 1958, Dill 1989

Emptiness

D

x<=1
y>=5
y-x<=3

Graph



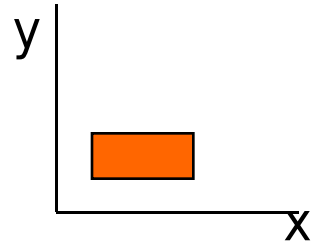
Negative Cycle
iff
empty solution set

Compact

Canonical Dastructures for Zones

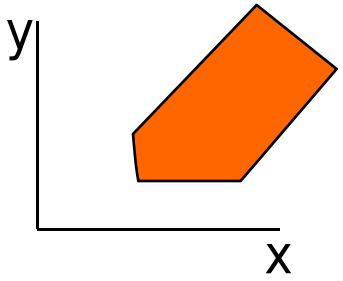
Difference Bounded Matrices

Future



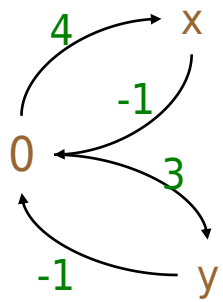
D

1<= x <=4
1<= y <=3

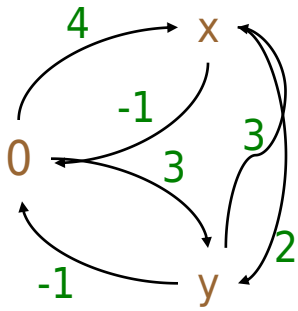


Future D

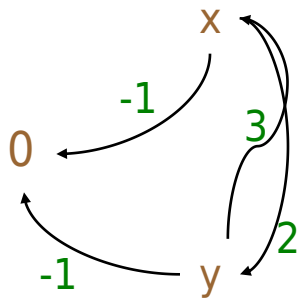
1<=x, 1<=y
-2<=x-y<=3



Shortest Path Closure



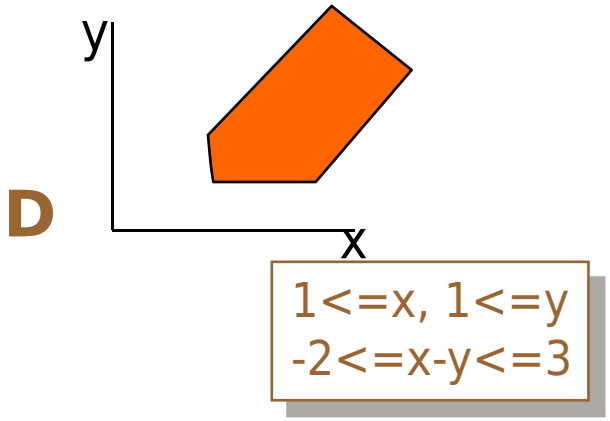
Remove upper bounds on clocks

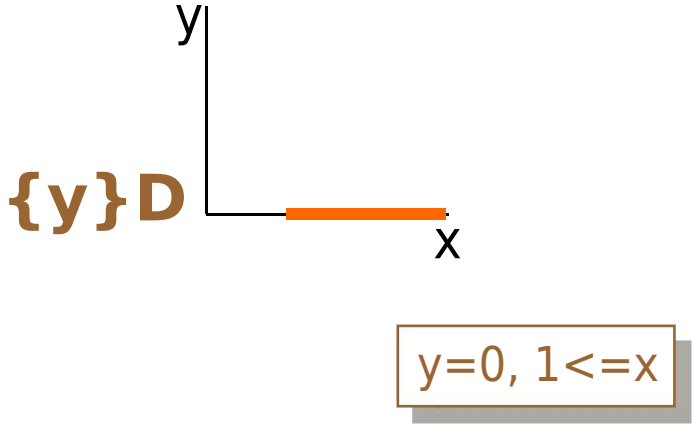


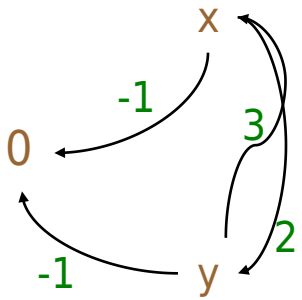
Canonical Dastructures for Zones

Difference Bounded Matrices

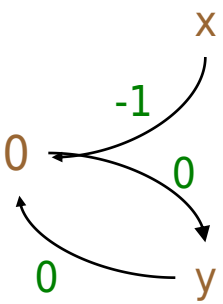
Reset







Remove all bounds involving y and set y to 0

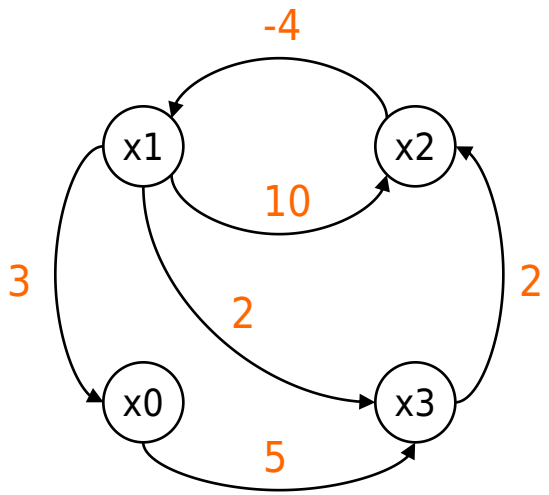


Improved Datastructures

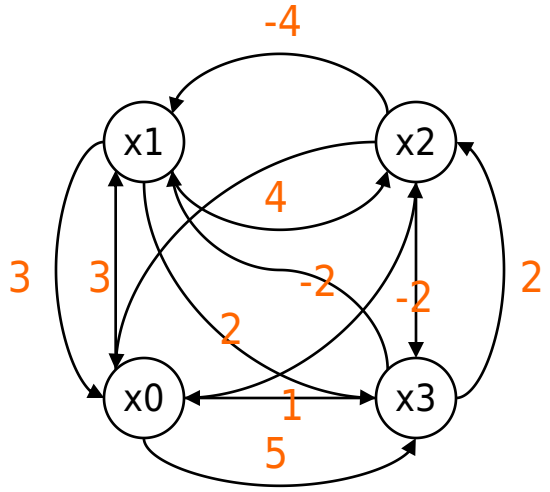
Compact Datastructure for Zones

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)

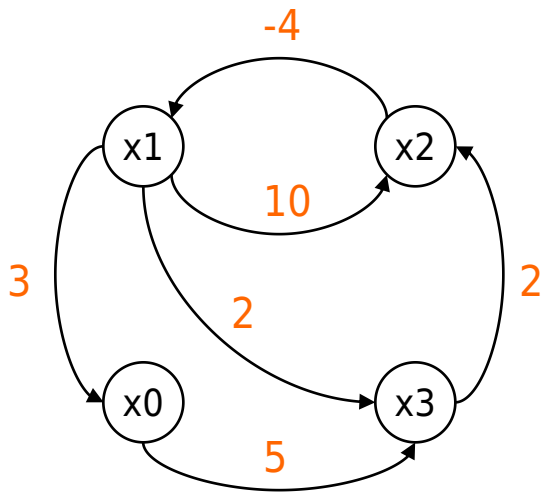


Improved Datastructures

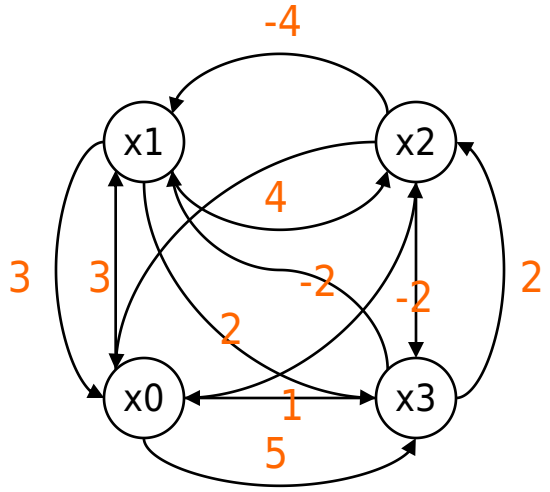
Compact Datastructure for Zones

RTSS 1997

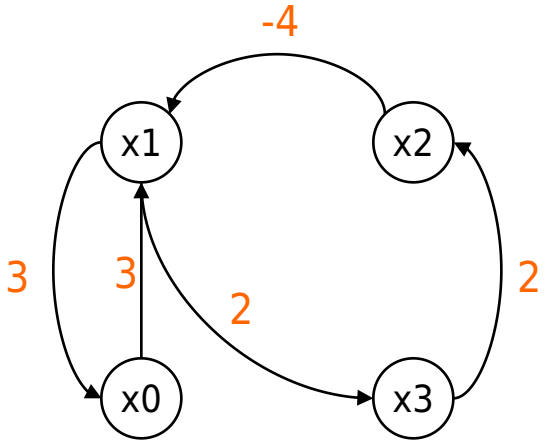
x1-x2<=4
x2-x1<=10
x3-x1<=2
x2-x3<=2
x0-x1<=3
x3-x0<=5



Shortest Path Closure O(n^3)



Shortest Path Reduction O(n^3)

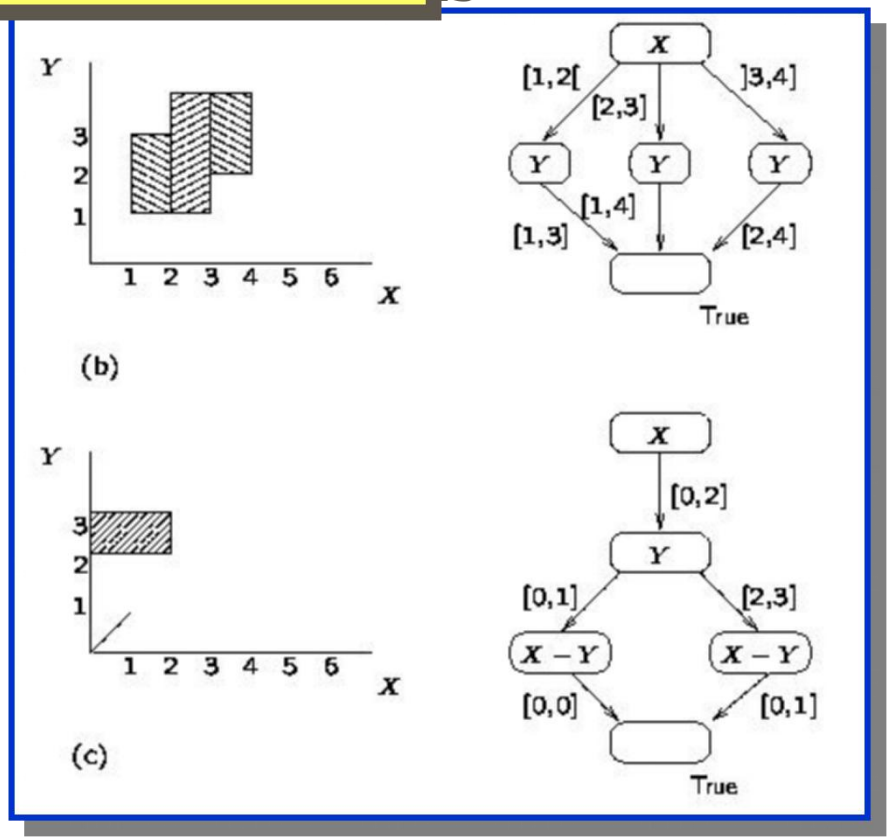


Canonical wrt =
Space worst O(n^2)
practice O(n)

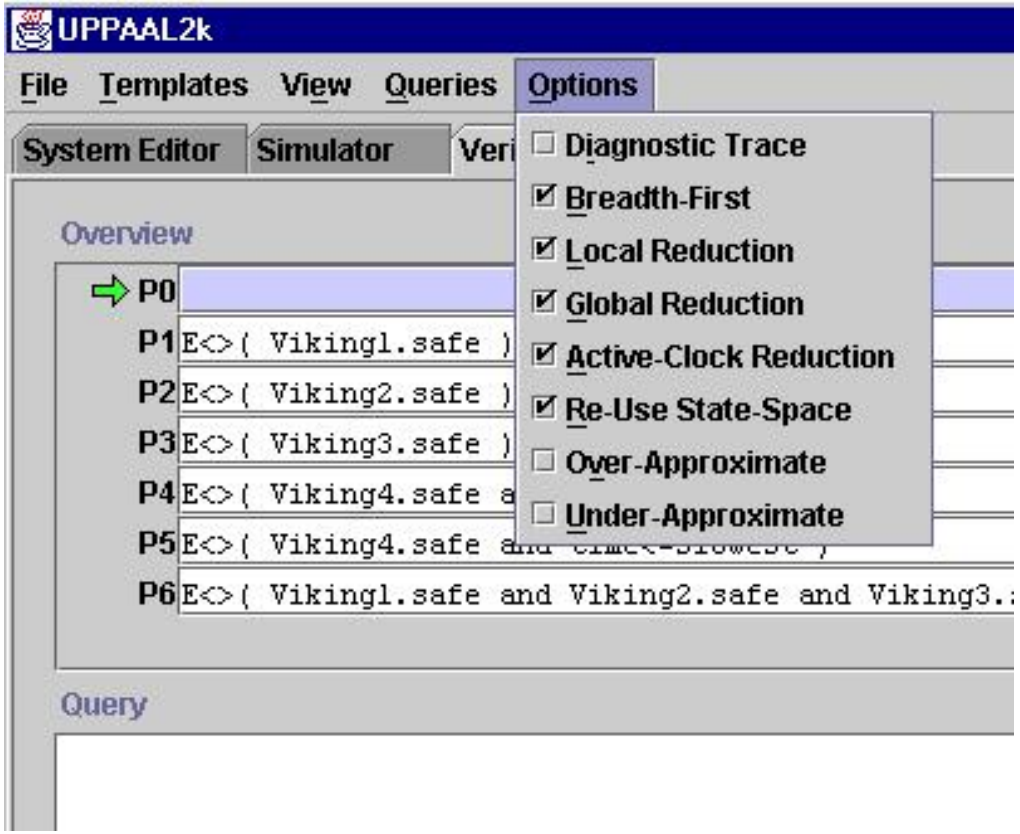
Other Symbolic Datastructures

- Regions Alur, Dill
- NDD's Maler et. al.
- CDD's UPPAAL/CAV99
- DDD's Møller, Lichtenberg
- Polyhedra HyTech
- .....

CDD-representations



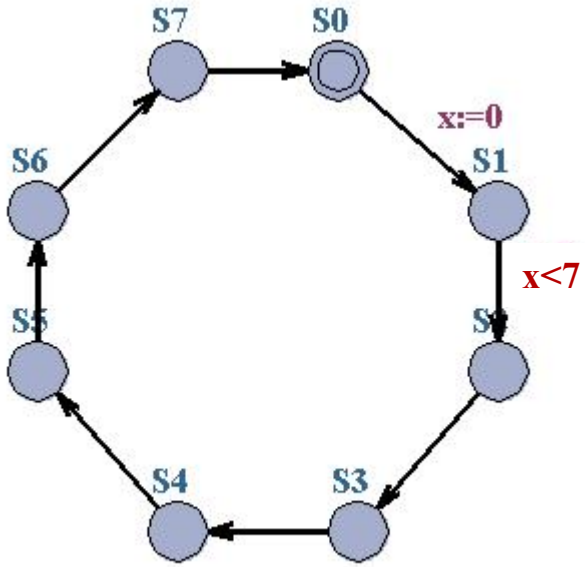
Verification Options



- Diagnostic Trace
- Breadth-First
- Depth-First
- Local Reduction
- Active-Clock Reduction
- Global Reduction
- Re-Use State-Space
- Over-Approximation
- Under-Approximation

Case Studies

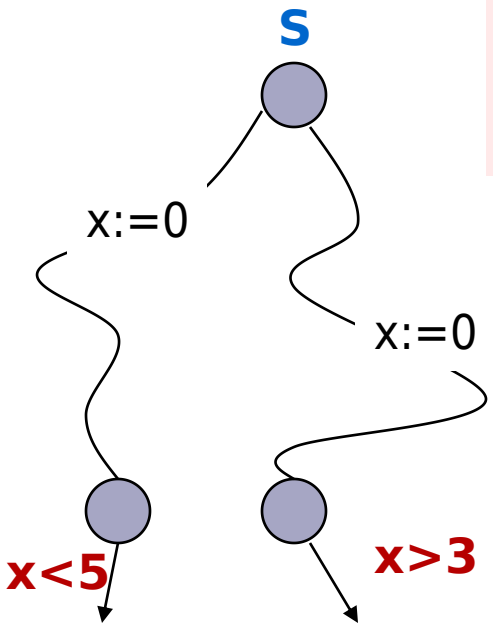
Representation of symbolic states *(In)Active Clock Reduction*



x is only ***active*** in location **S1**

Definition

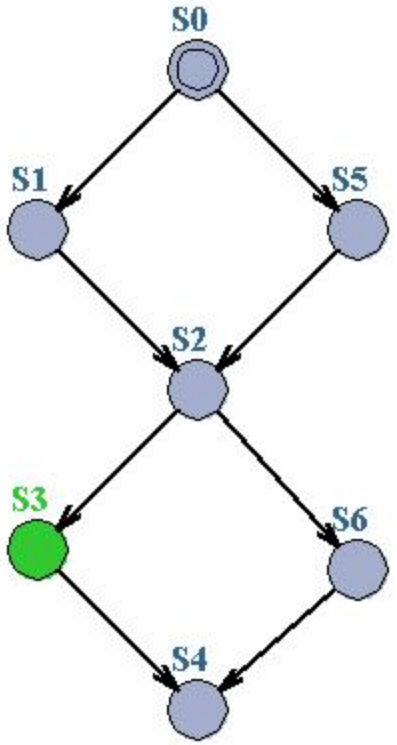
x is ***inactive*** at **S** if on all path from **S**, x is always reset before being tested.



Case Studies

When to store symbolic state

Global Reduction



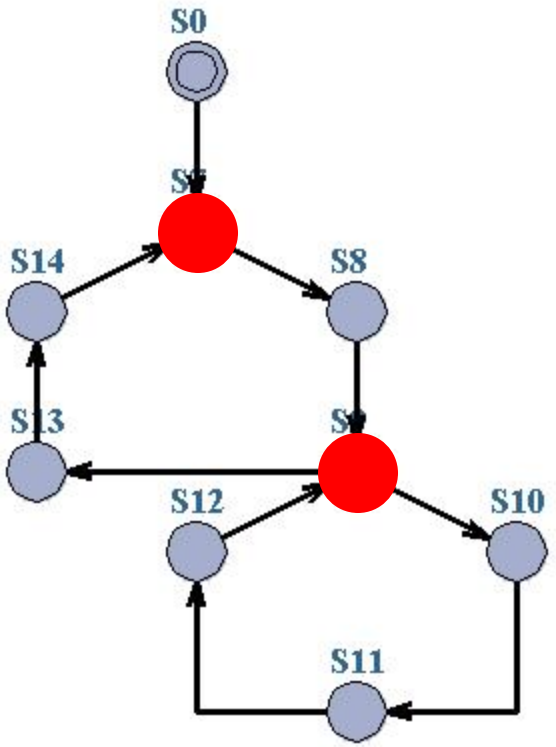
However,
Passed list useful for efficiency

Case Studies

No Cycles: **Passed** list not needed for *termination*

When to store symbolic state

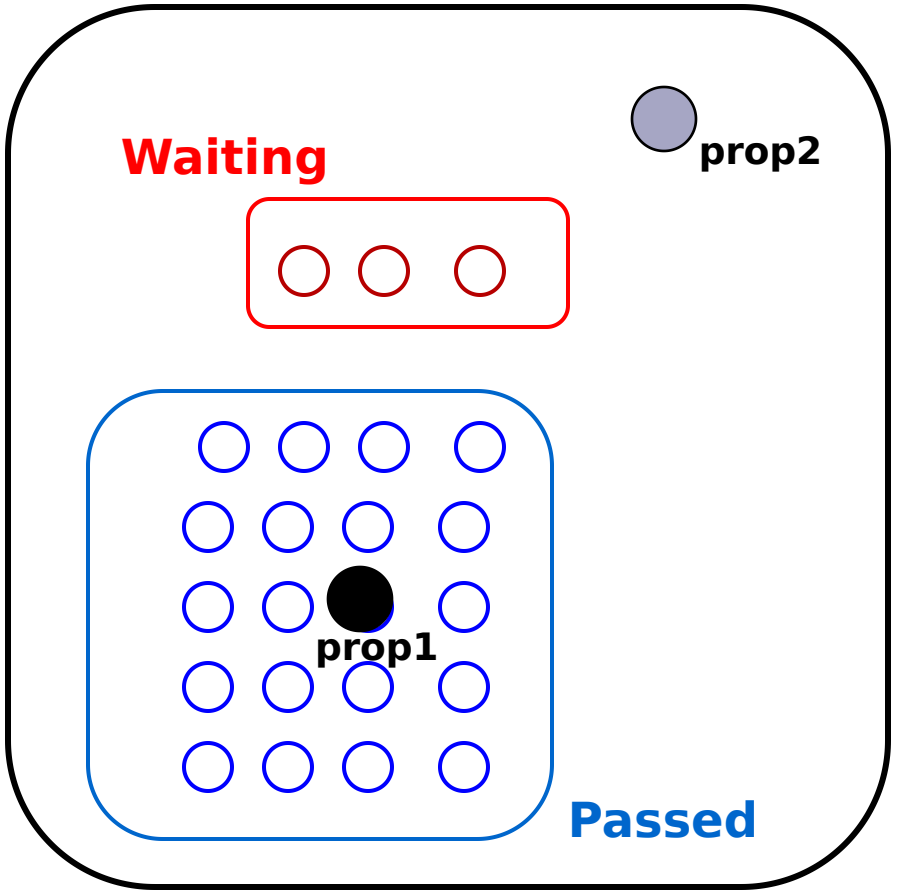
Global Reduction



Cycles:
Only symbolic states involving loop-entry points need to be saved on **Passed** list

Case Studies

Reuse State Space



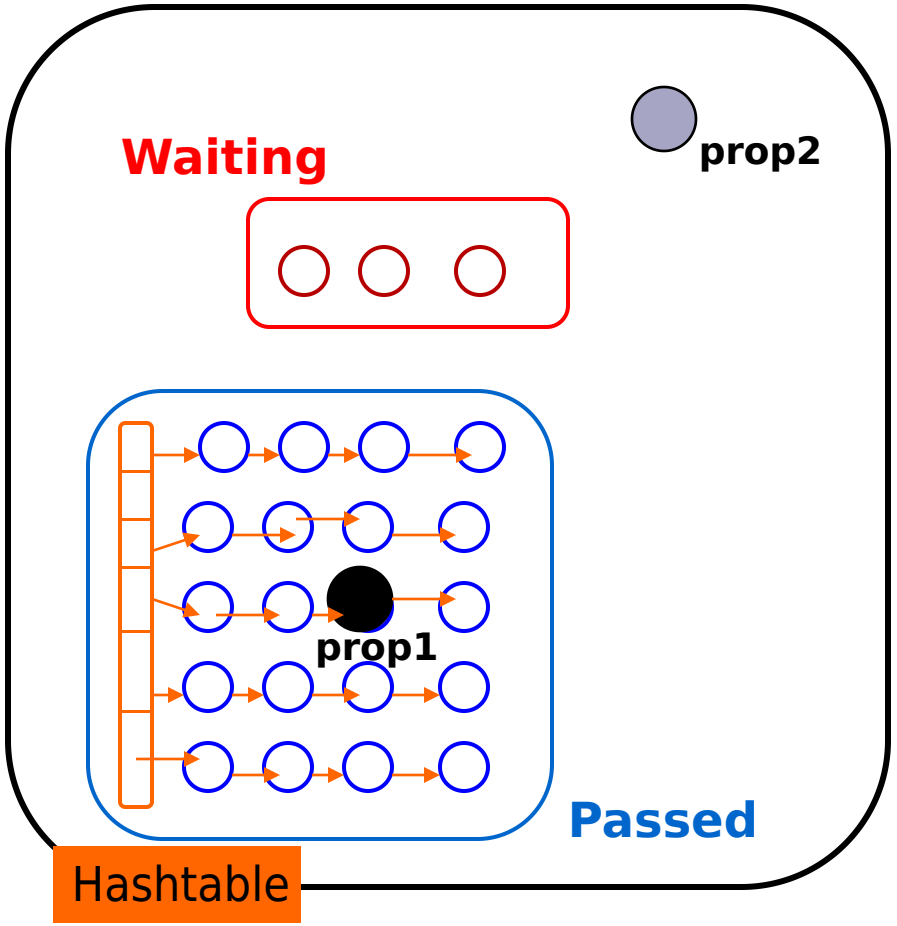

```
A[]  prop1

A[]  prop2
A[]  prop3
A[]  prop4
A[]  prop5
.
.
.
A[]  propn
```

Search in existing **Passed** list before continuing search

Which order to search?

Case Studies

Reuse State Space



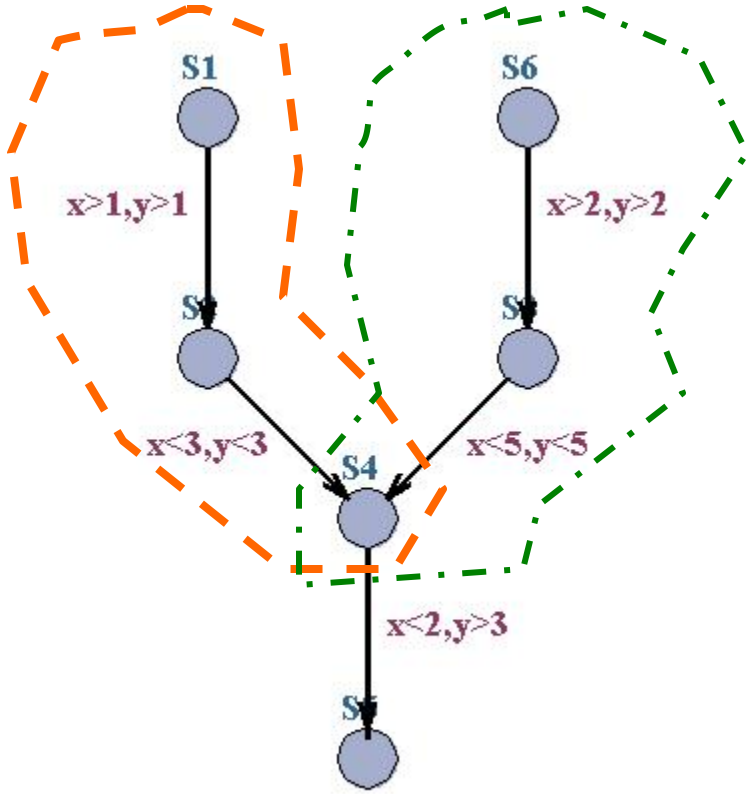

```
A[]   prop1

A[]   prop2
A[]   prop3
A[]   prop4
A[]   prop5
.
.
.
A[]   propn
```

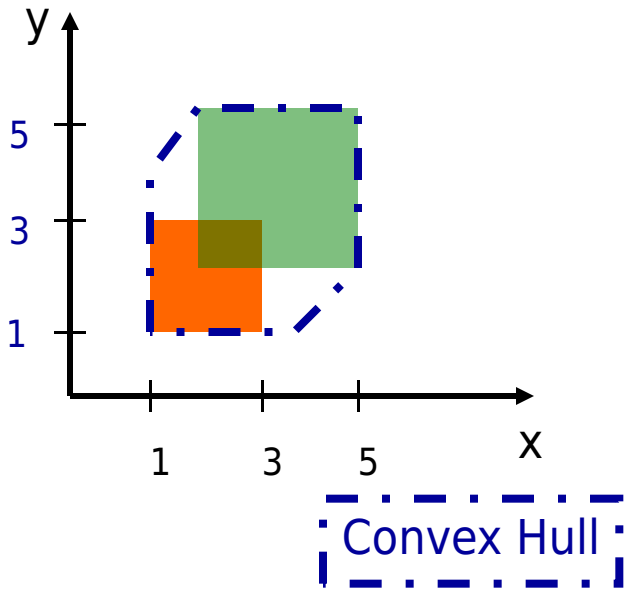
Search in existing **Passed** list before continuing search

Which order to search?

Over-approximation

Convex Hull





Case Studies

Case Studies

Case Studies: Protocols

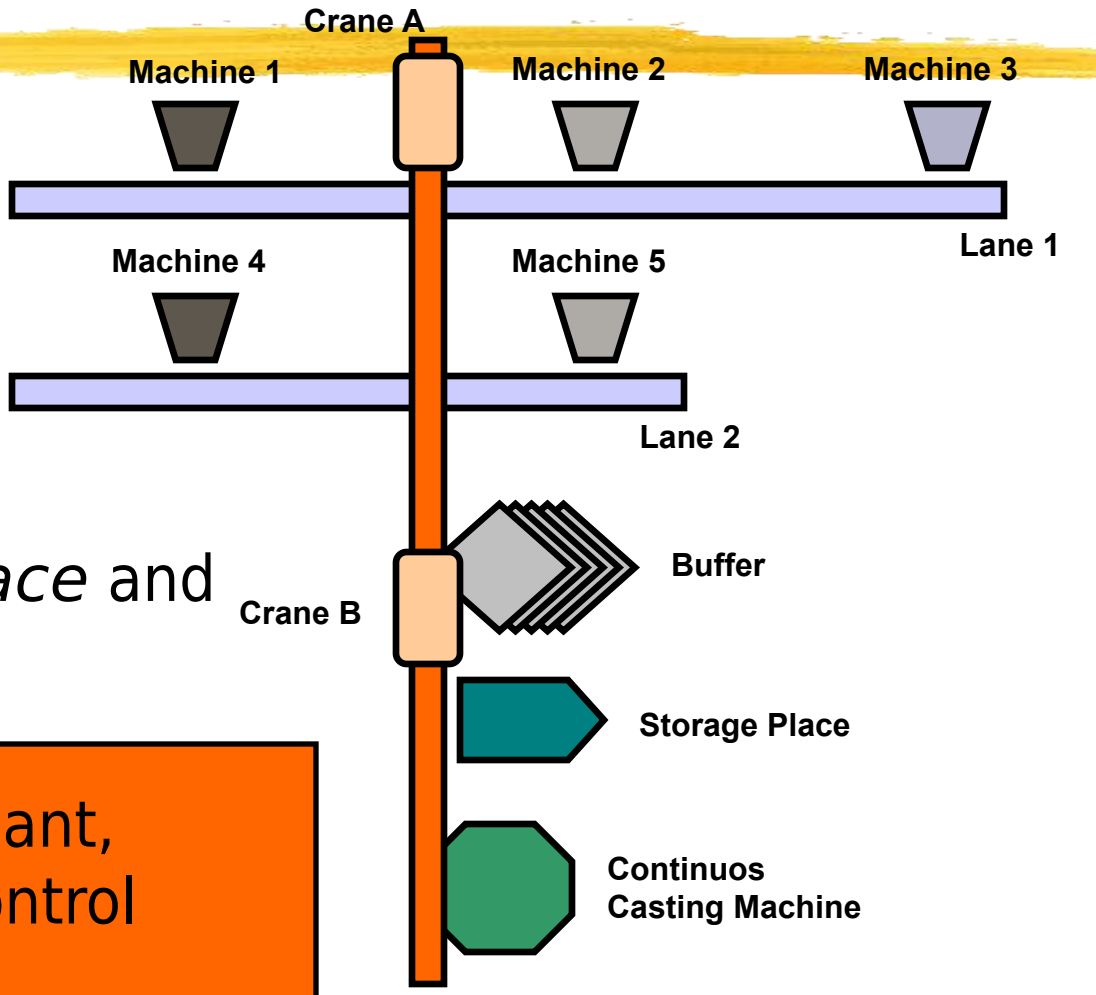
- Philips Audio Protocol [HS'95, CAV'95, RTSS'95, CAV'96]
- Collision-Avoidance Protocol [SPIN'95]
- Bounded Retransmission Protocol [TACAS'97]
- Bang & Olufsen Audio/Video Protocol [RTSS'97]
- TDMA Protocol [PRFTS'97]
- Lip-Synchronization Protocol [FMICS'97]
- Multimedia Streams [DSVIS'98]
- ATM ABR 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)

Steel Production Plant

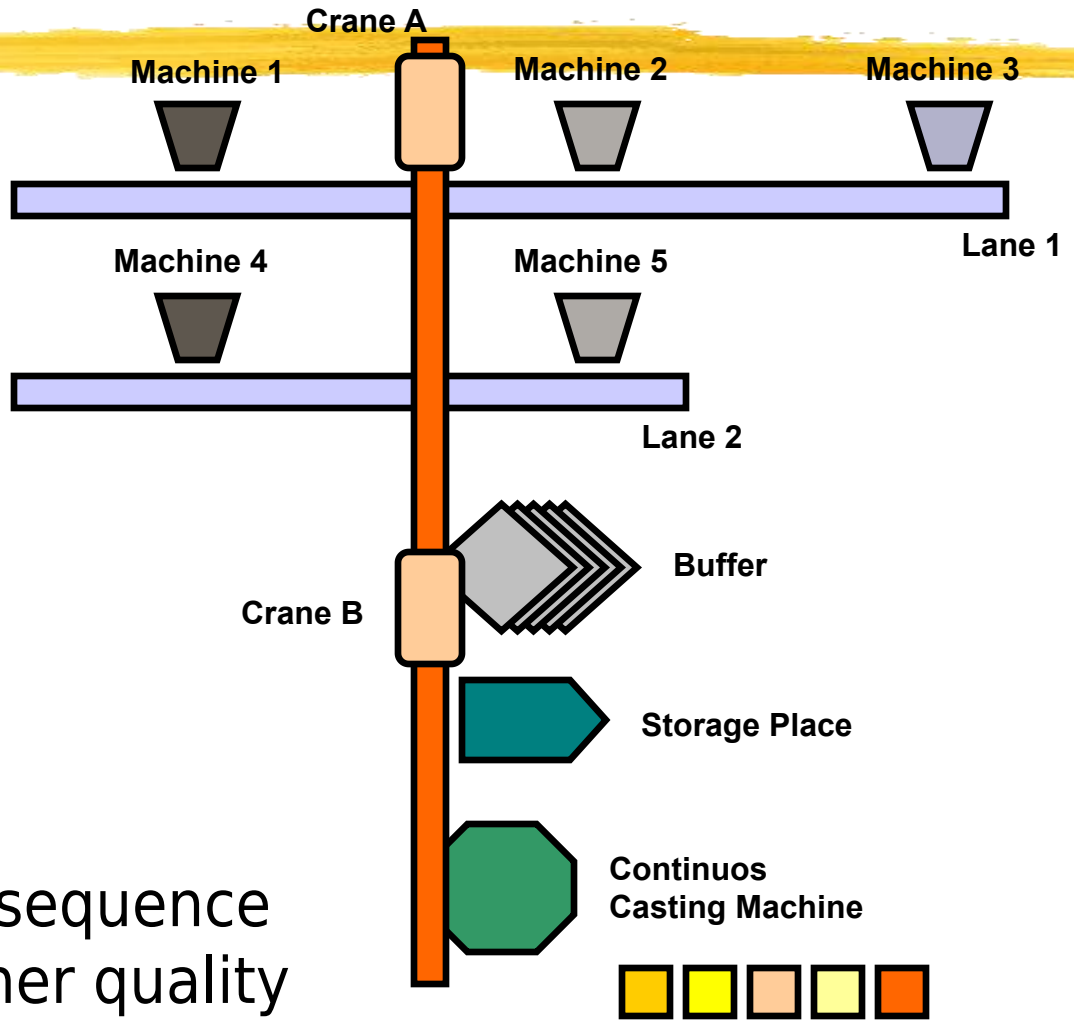
- A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson
- Case study of Esprit-LTR project 26270 VHS
- Physical plant of SIDMAR located in Gent, Belgium.
- Part between *blast furnace* and *hot rolling mill*.



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

Steel Production Plant

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



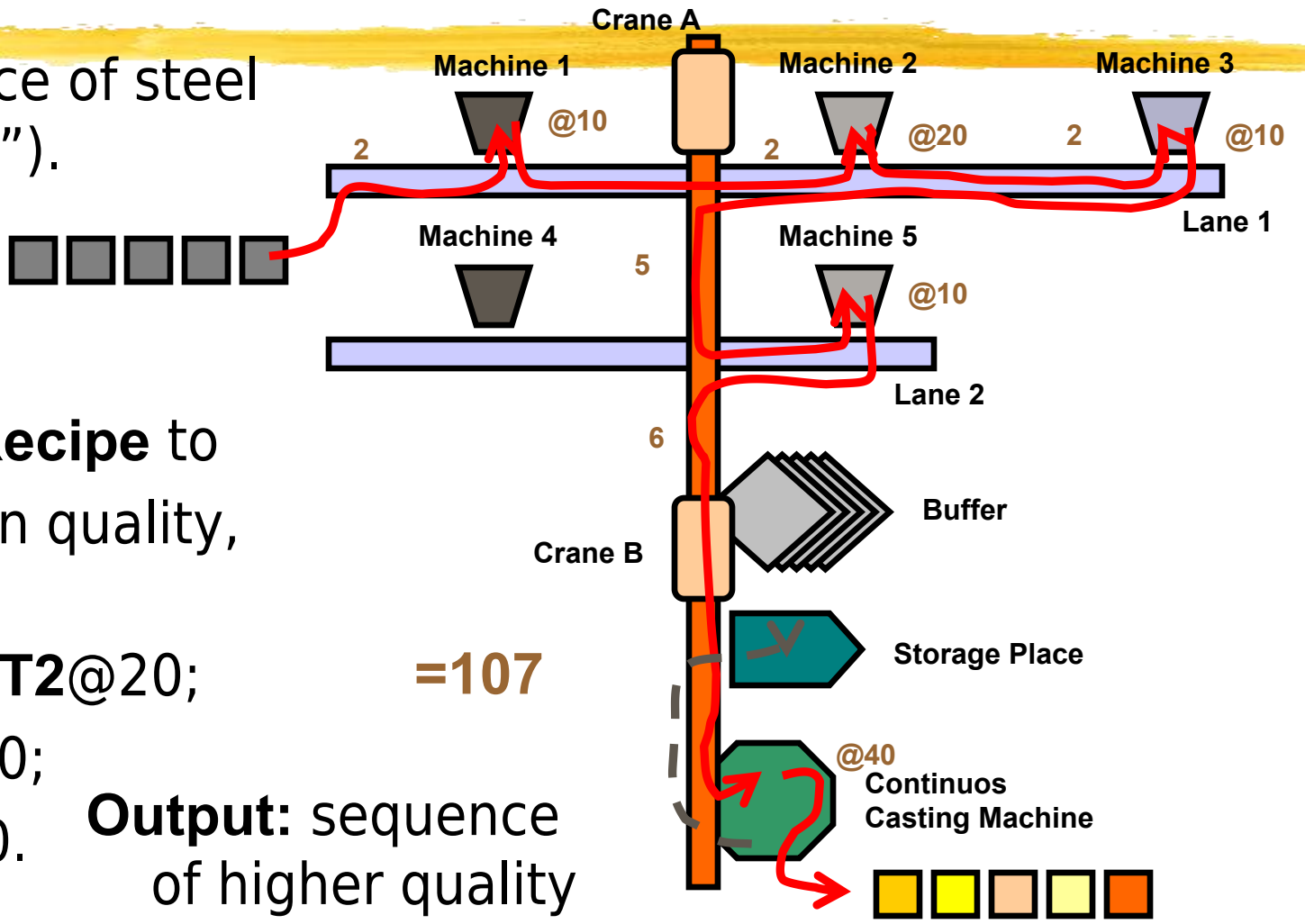
Load follows **Recipe** to become certain quality, e.g:

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:

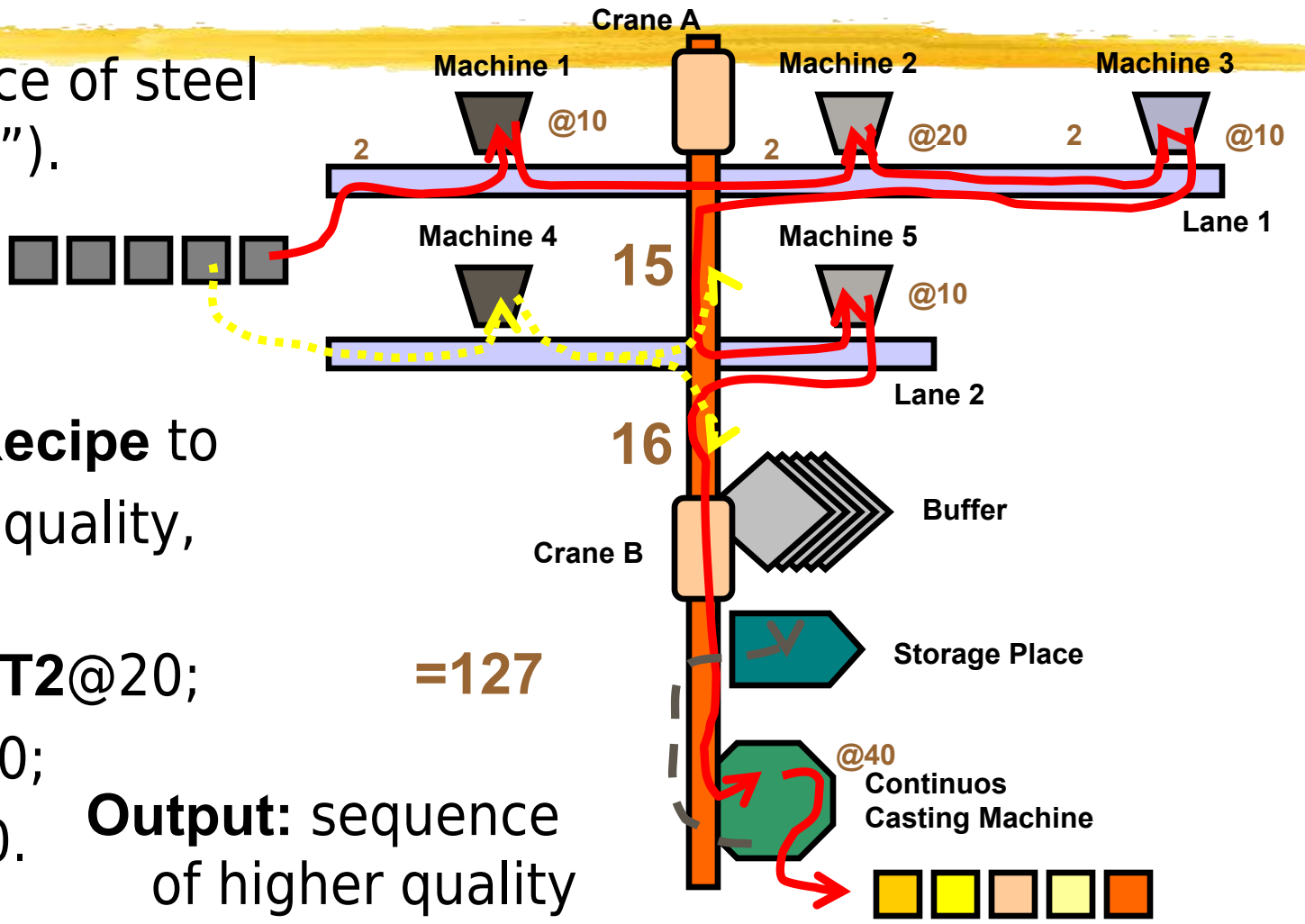
start; **T1**@10; **T2**@20; **T3**@10; **T2**@10; end within 120.

=107

Output: sequence of higher quality steel.

Steel Production Plant

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



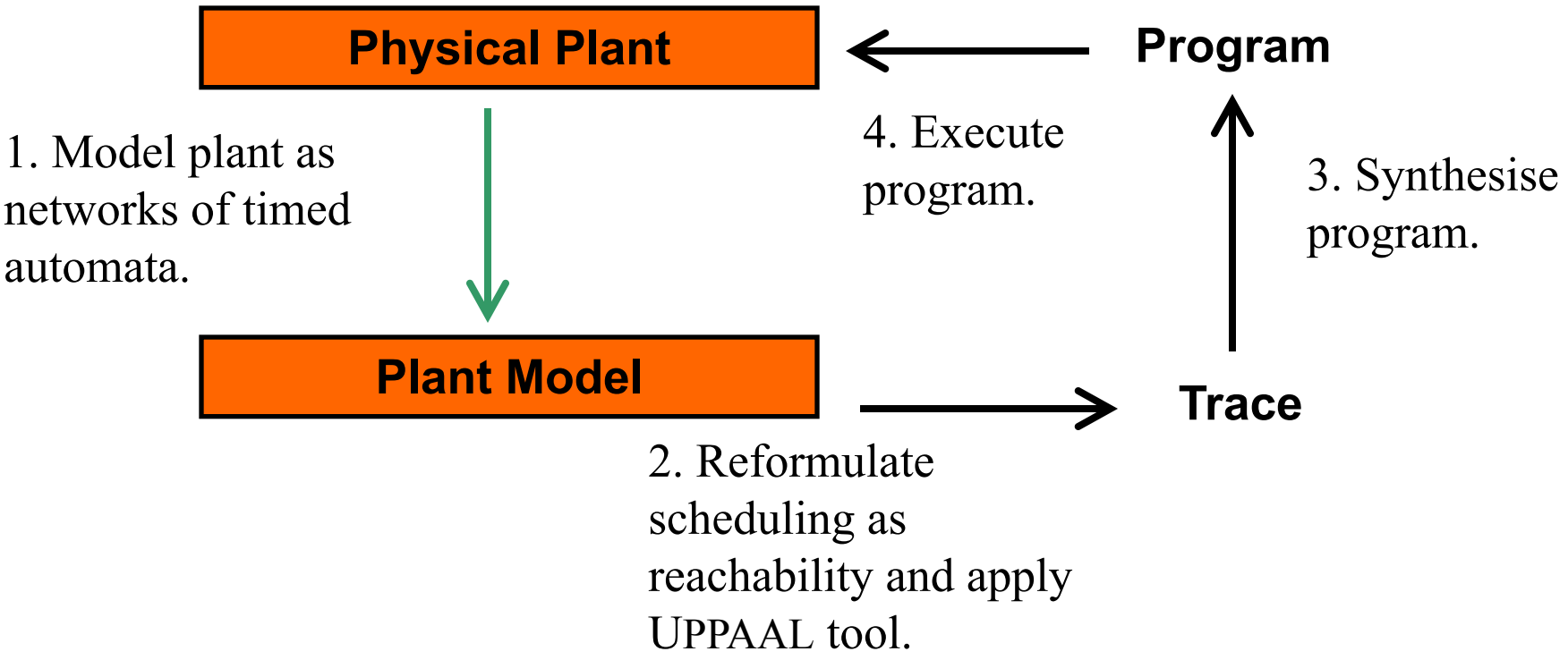
Load follows **Recipe** to obtain certain quality, e.g:

start; **T1**@10; **T2**@20; **T3**@10; **T2**@10; end within 120.

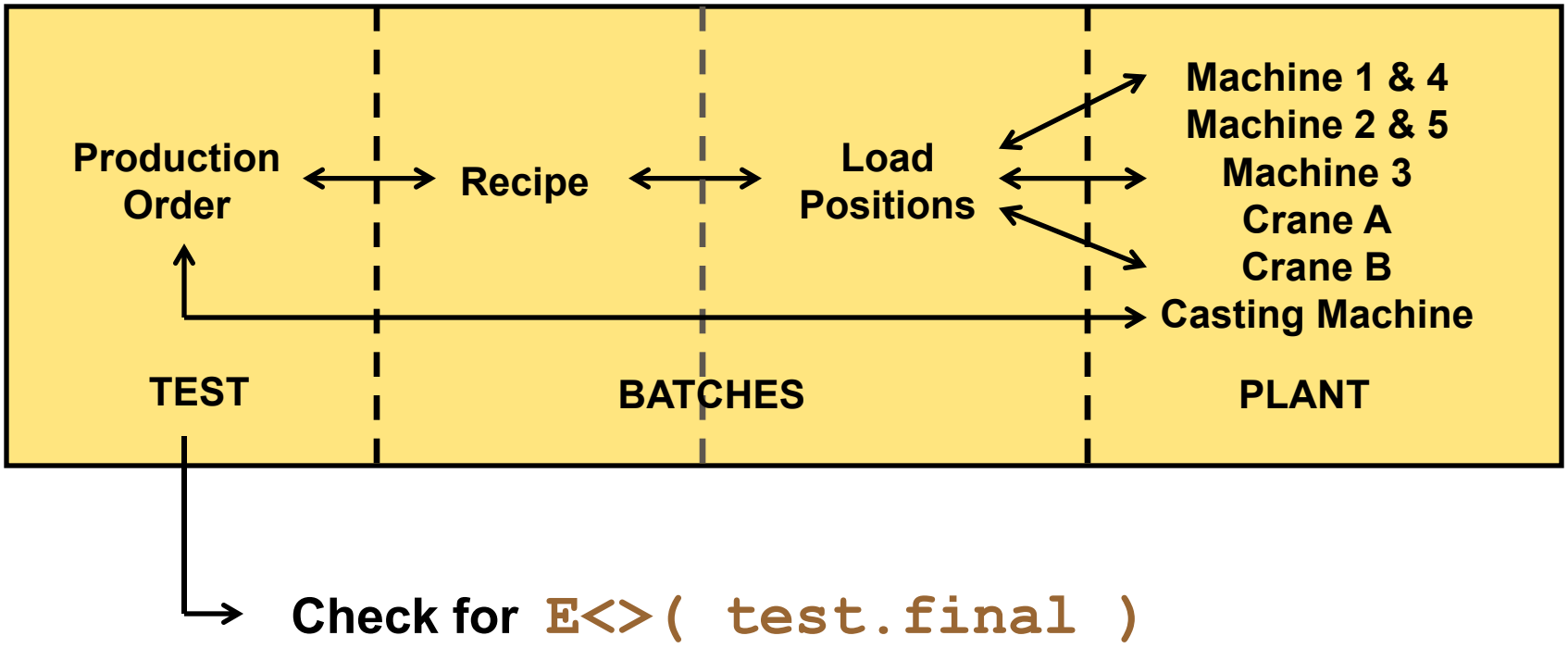
=127

Output: sequence of higher quality steel.

Modus Operandi



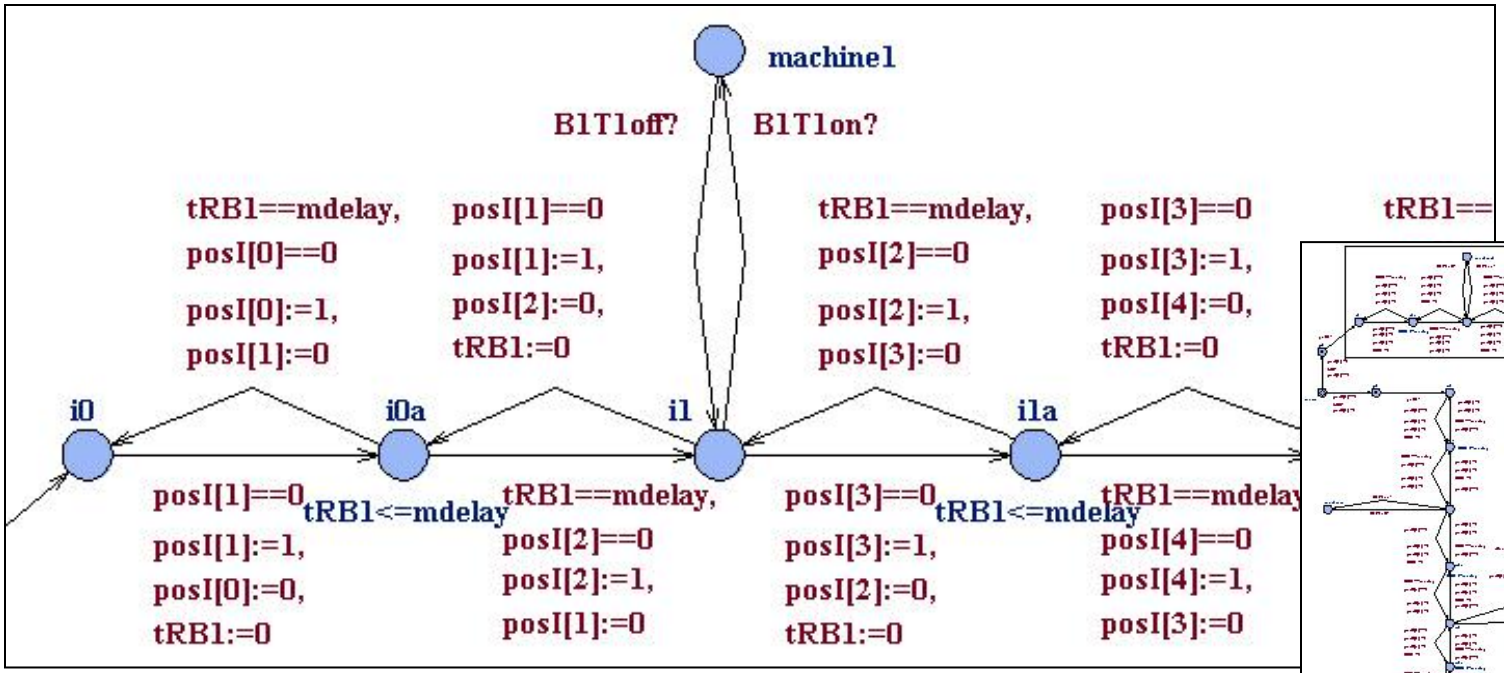
Overview of Plant Model

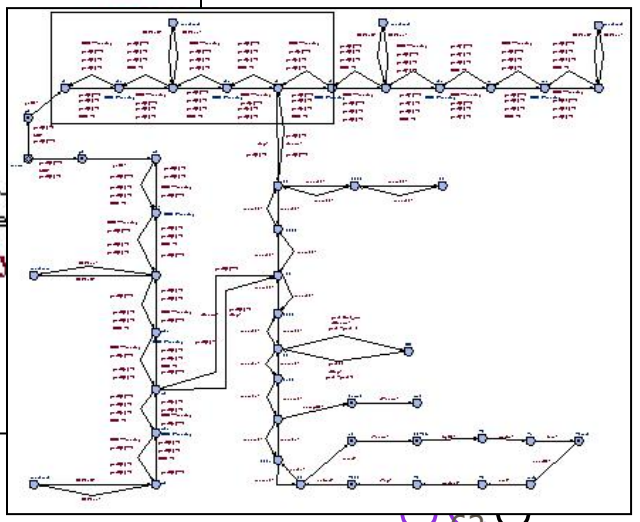


UPPAAL generates diagnostic trace if property holds.

Load Automaton

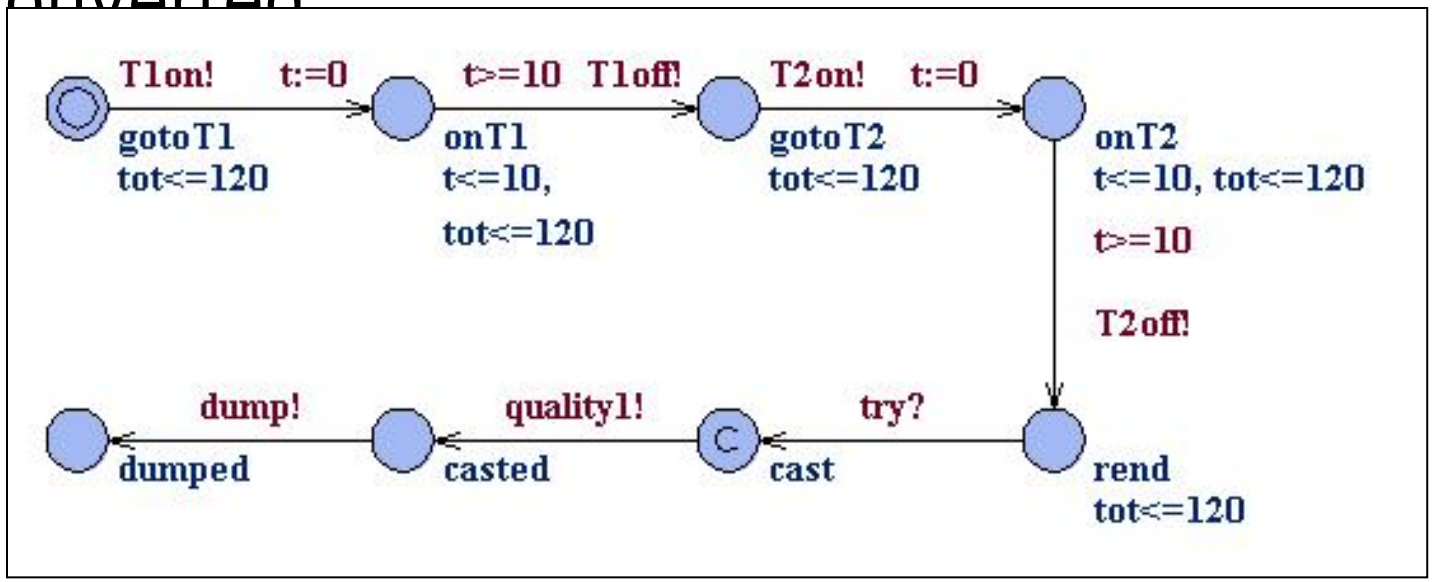
- Models all possible movements of load.
- Clock **tRB1** used to model time consumption.
- Bit vectors **posI** and **posII** models mutex.





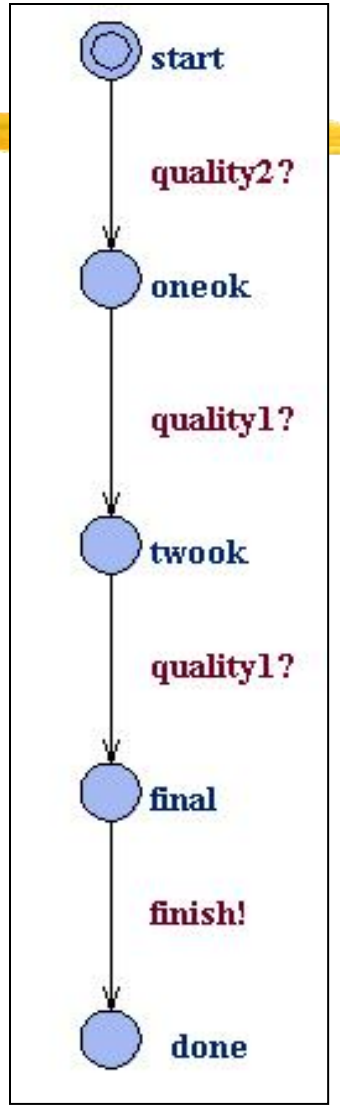
Steel Recipe

- Clock **tot** to constrain upper total time bound.
- Clock **t** to measure duration of machine treatment.
- Channel **quality1!** signaled when steel converted

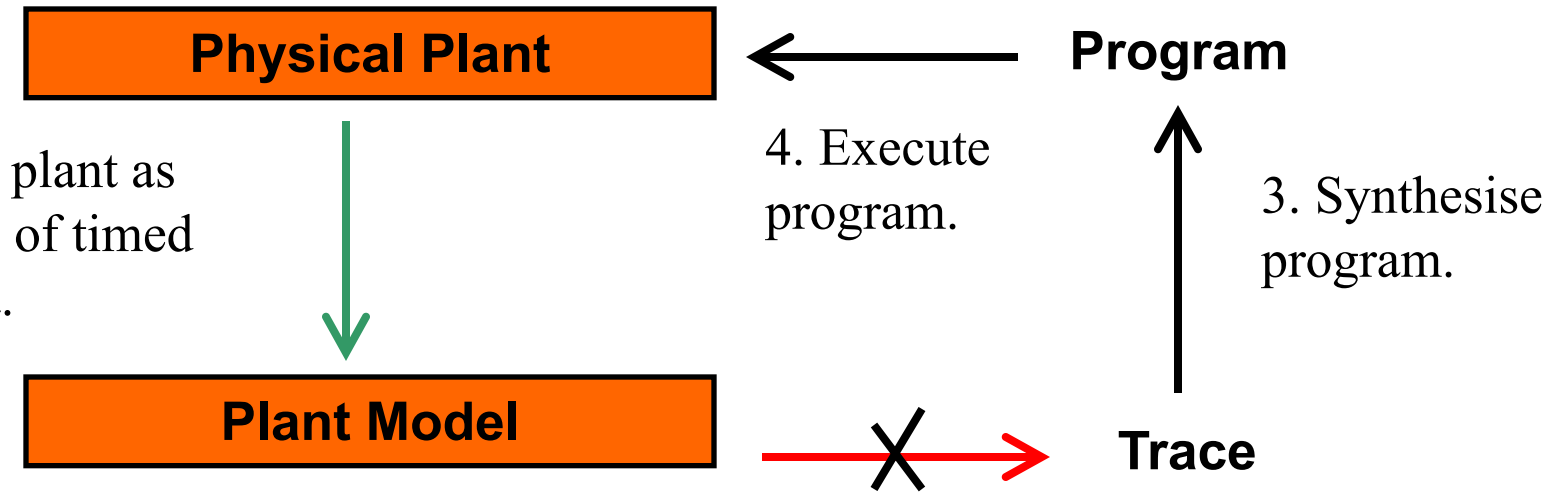


Production Order

- Specifies the steel qualities to be produced.
- Allows for scheduling to be reformulated as reachability.
- Reachable only with **feasible schedule**
- **`E<>( final )`**



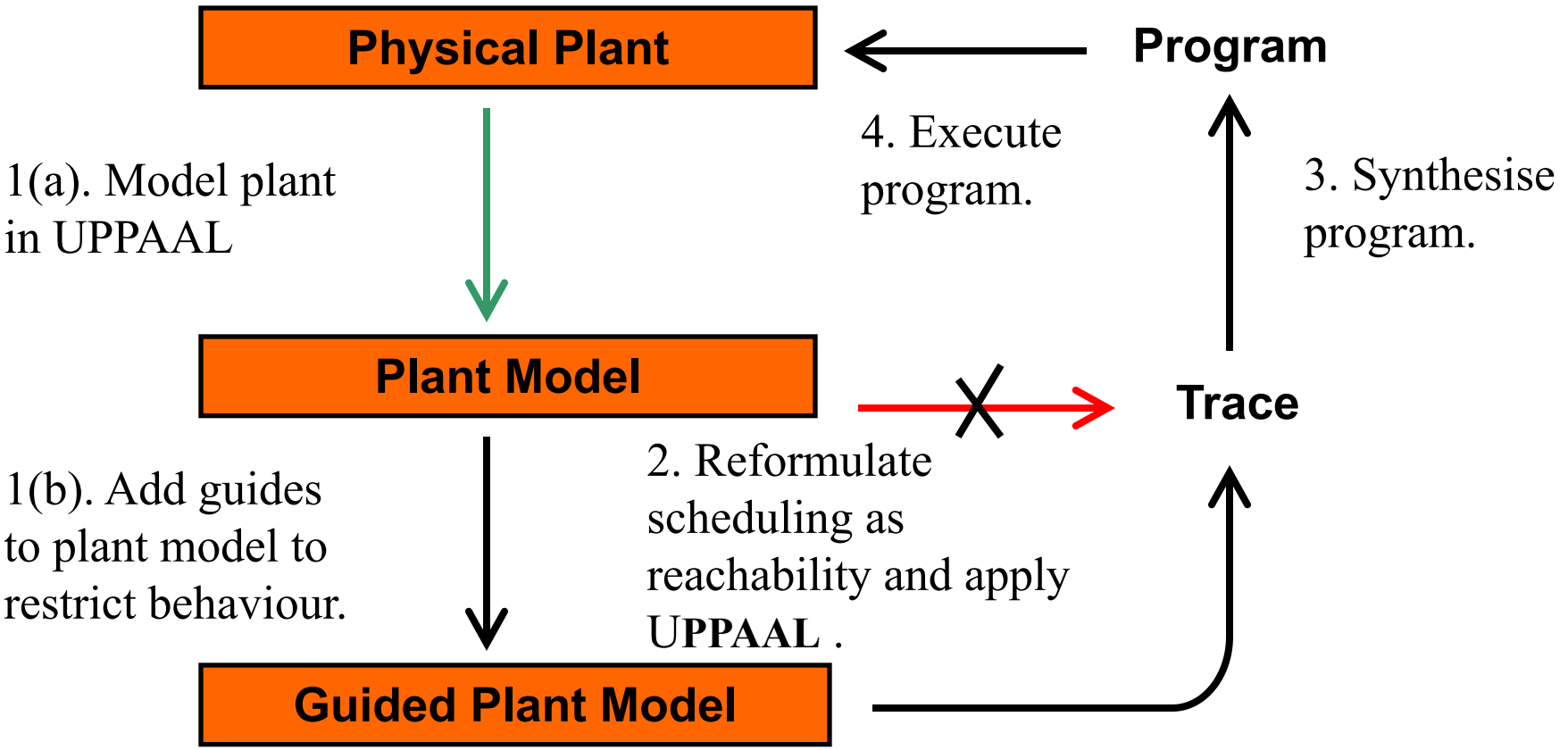
Modus Operandi



- **System with 5 steel loads: Parallel composition of:**
  - **15 timed automata (6 - 60 locations),**
  - **18 real-valued clocks,**
  - **28 bounded integer variables,**
  - **140 action channels.**

Verification:
Generating schedule for three batches **FAILS!!!**

Modus Operandi



Guiding the Model

Idea: Guide model according to chosen strategies:

- **enforce** desired behaviors,
- **restrict** undesired behaviors.

Implementation: Annotate model with:

- clock and integer variables,
- assignments (to added clock or variable), and
- guards (to any clock or variable).

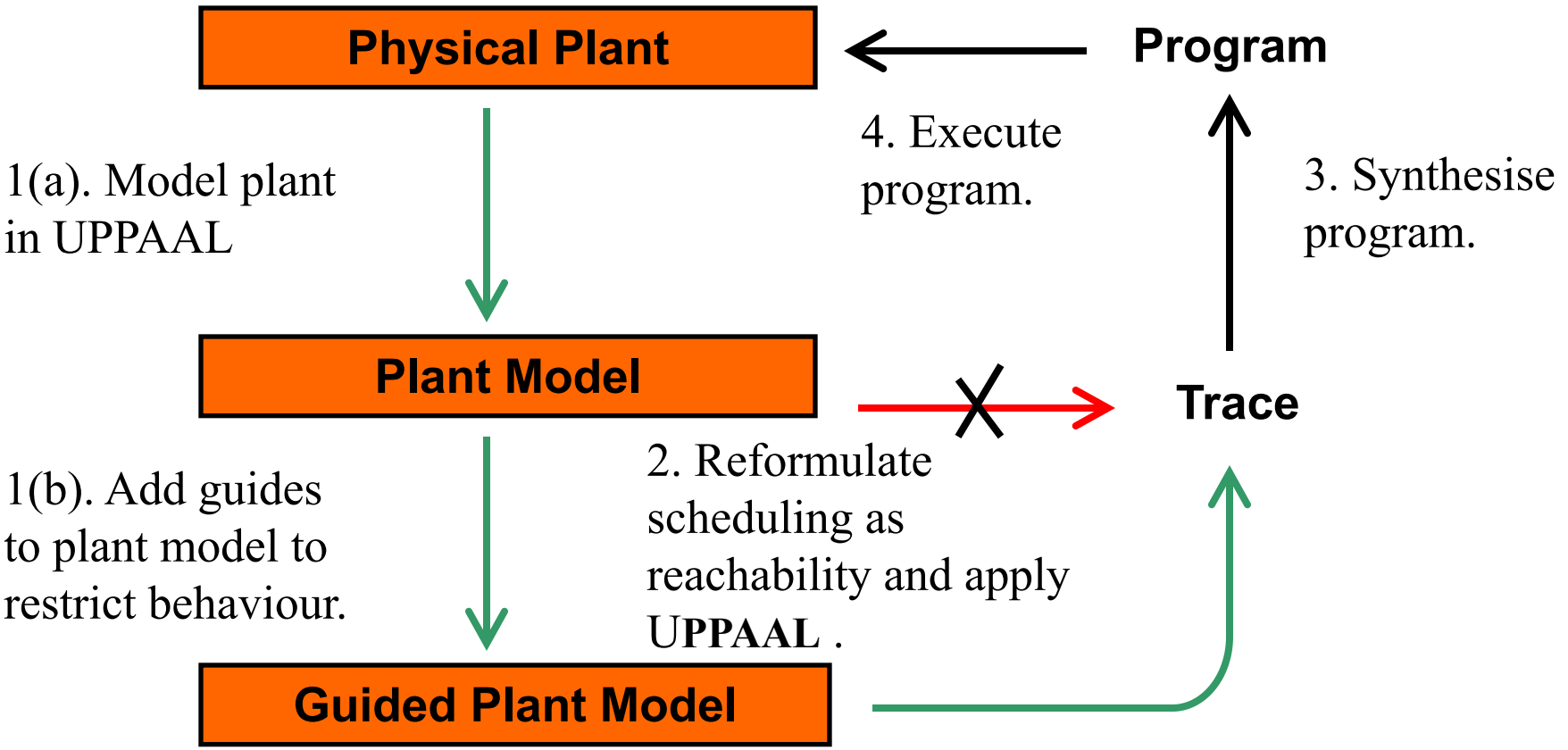
Fact: Trace of **guided** model is guaranteed to be trace of **unguided** model.

Experiment

	All Guides						Some Guides						No Guides					
n	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	-	-	-	-	-	-	-	-	-	-	-	-	-	-

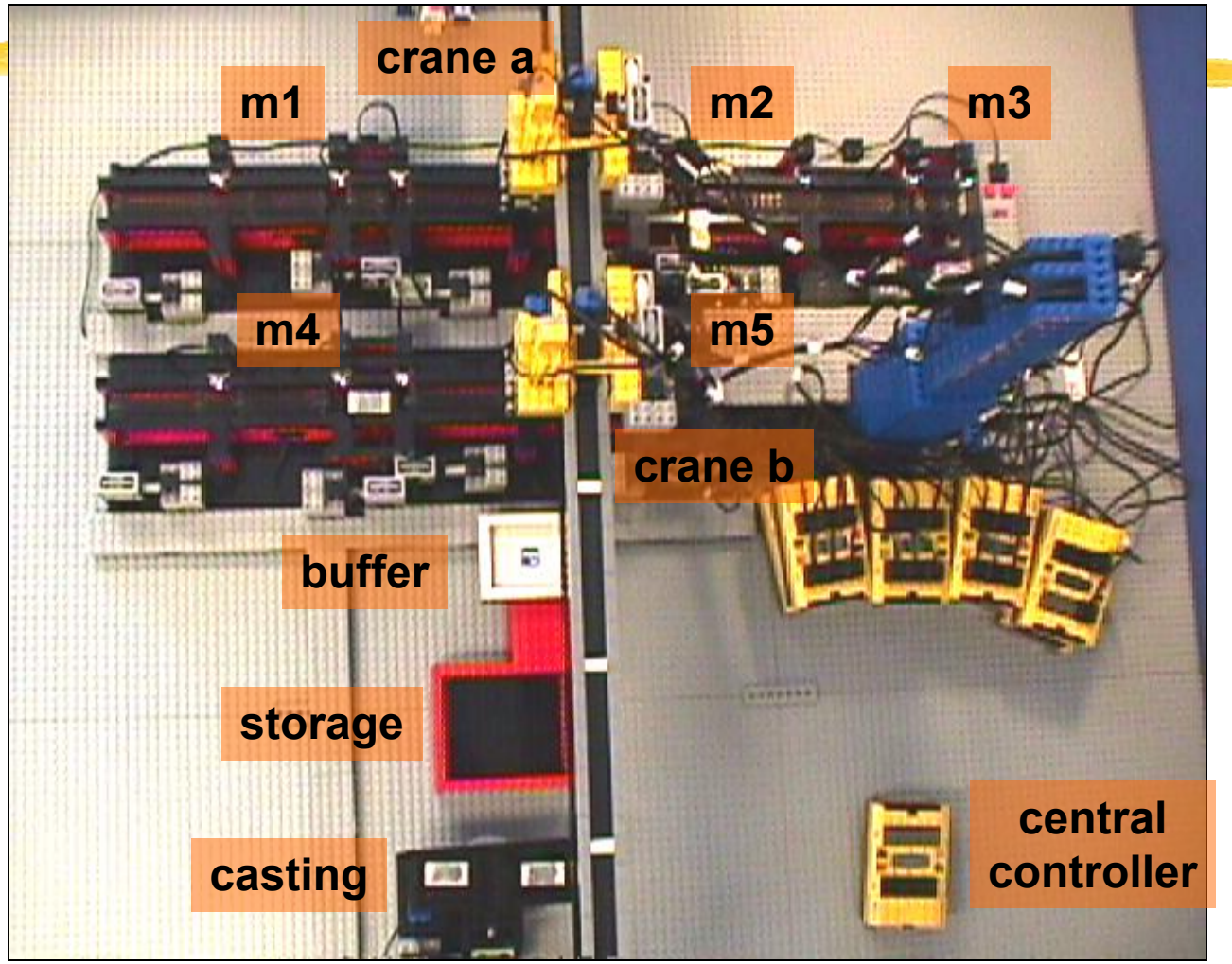
- **BFS** = breadth-first search, **DFS** = depth-first search, **BSH** = bit-state hashing,
- **"-"** = requires >2h (on 450MHz Pentium III), >256 MB, or suitable hash-table size was not found.
- **System size**: 2n+5 automata and 3n+3 clocks, if n=35: 75 automata and 108 clocks.
- **Schedule generated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

Modus Operandi



LEGO Plant Model

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



Synthesis

Local Control Programs

- Belts:
  - move left/right,
  - receive from left/right,
- Machine:
  - start,
  - stop,
  - move left/right,
  - receive from left/right,
- Cranes:
  - move up/down,
  - set down,
  - pick up,
- Casting Machine:
  - start,
  - stop,
- ...

Extracting Programs

Trace

...
(loadB1.p1 recipeB1.gotoT1 loadB2...
{ loadB1.x=5 recipeB1.tot=5 recipeB1...
Sync: b1right
(loadB1.pre recipeB1.gotoT1 loadB2...
{ loadB1.x=5 recipeB1.tot=5 recipeB1...
delay(5)
(loadB1.pre recipeB1.gotoT1 loadB2...
{ loadB1.x=10 recipeB1.tot=10 recipe...
Sync: B1M1on
(loadB1.onM1 recipeB1.onT1 loadB2...
{ loadB1.x=0 recipeB1.tot=10 recipe...
delay(10)
(loadB1.onM1 recipeB1.onT1 loadB2...
{ loadB1.x=10 recipeB1.tot=20 recipe...
Sync: B1M1off
(loadB1.pre recipeB1.gotoT2 loadB2...

Schedule

...

belt1 right

delay 5

load B1 on Machine 1

delay 10

load B1 off Machine 1

...

Program

...

// Belt Unit 1 move RIGHT
PB.SendPBMessage 2, 20

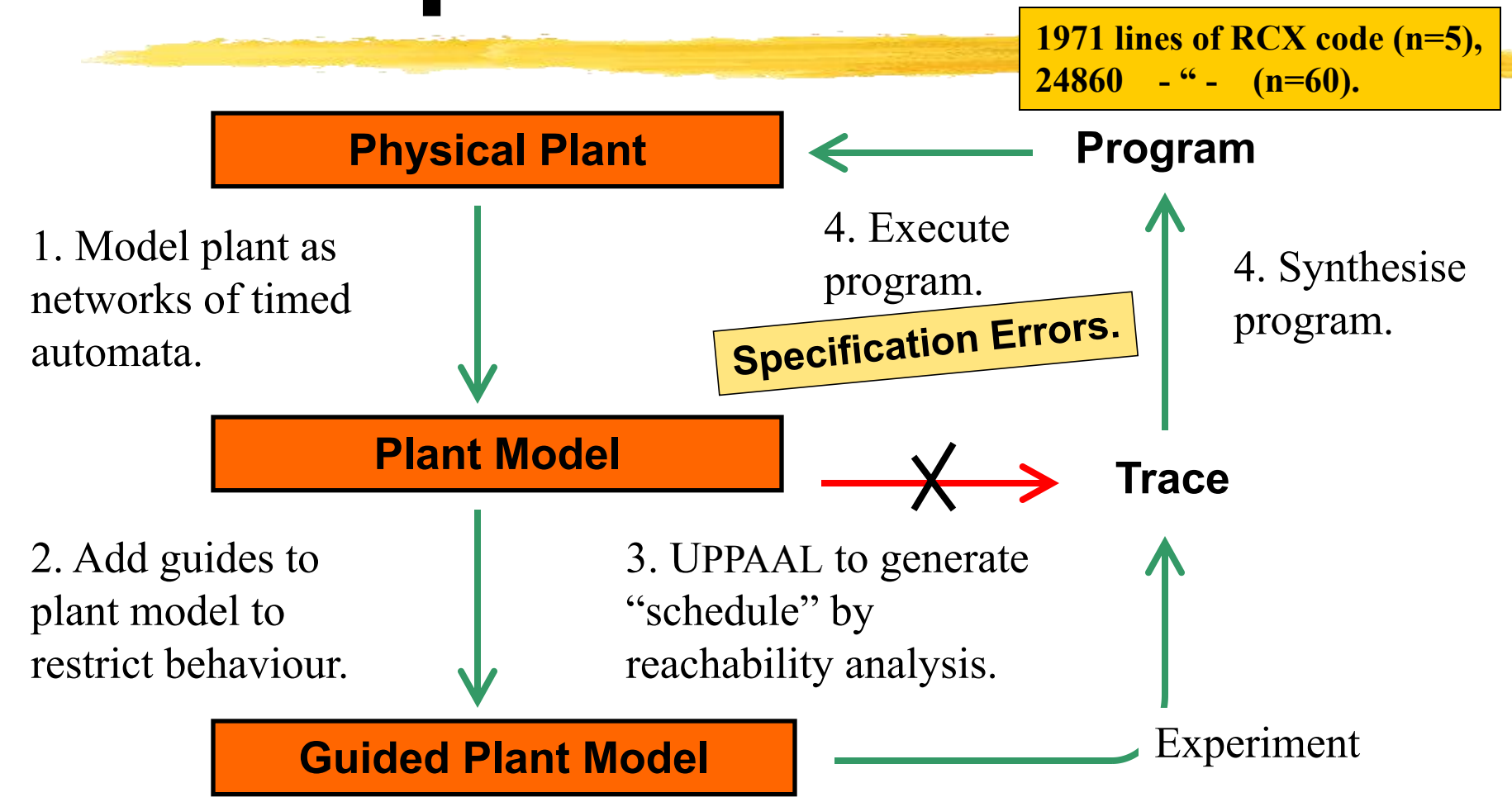
// Delay 5
PB.Wait 2, 500

// Machine 1 START
PB.SendPBMessage 2, 23

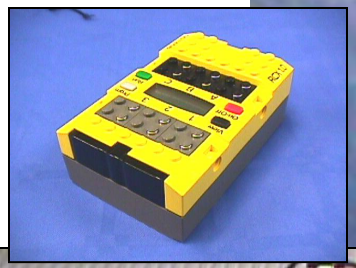
// Delay 10
PB.Wait 2, 100

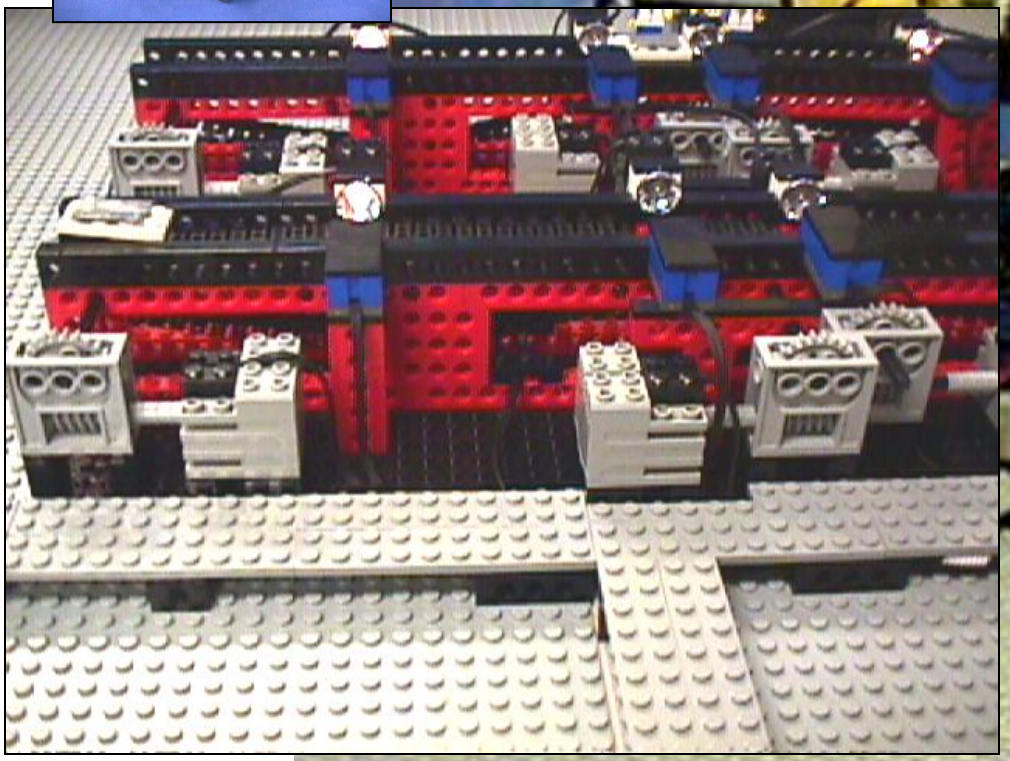
// Machine 2 STOP
PB.SendPBMessage 2,24

Modus Operandi



LEGO Plant Model





Belt/Machine U

Current & Future Research

- **D**UPPAAL
- **P**UPPAAL
- **G**UPPAAL
- **C**UPPAAL

- **H**UPPAAL
- **Pr**UPPAAL

Current & Future Research

- **D**UPPAAL Distributed [Behrmann,Hune,Vandraager, CAV2k]
- **P**UPPAAL Parameterized
- **G**UPPAAL Guided
- **C**UPPAAL Cost-Optimal

- **H**UPPAAL Hierarchical (UML)
- **Pr**UPPAAL Probabilistic

ENDE