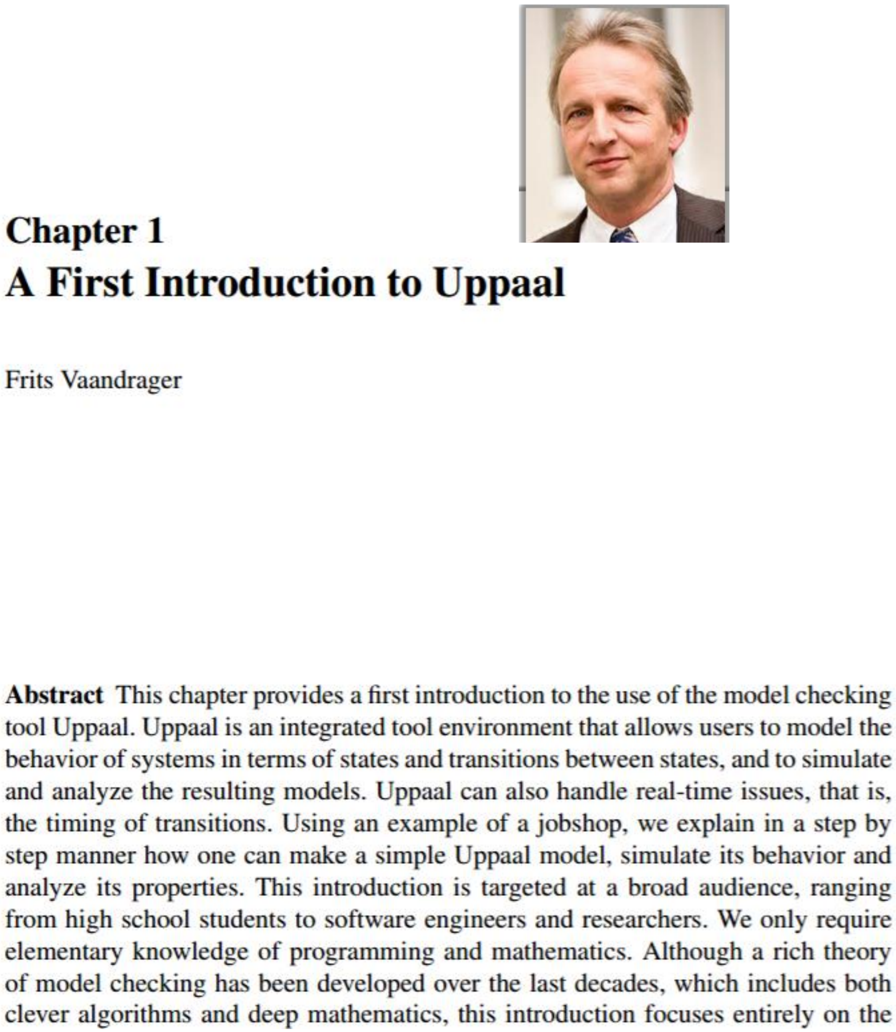
Modeling & Verification 2021

Lecture 11

- UPPAAL
  - History, Demos
- Train Gate Example
  - Modelling Formalism
  - Specification Formalism
- Engine & Options

This lecture will be recorded and afterwards be made available on Moodle
PLEASE INFORM THE LECTURER IF YOU DO NOT WANT RECORDING TO TAKE PLACE

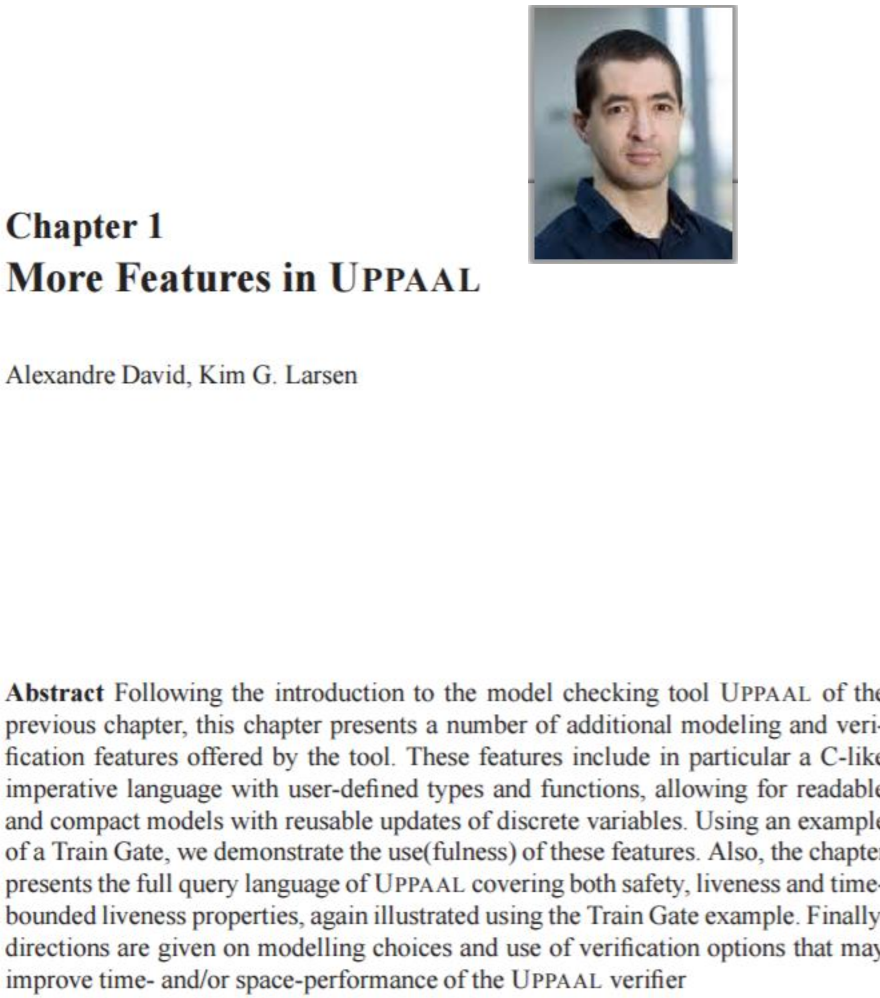
Main Readings (+RS Book)



Chapter 1
A First Introduction to Uppaal

Frits Vaandrager

Abstract This chapter provides a first introduction to the use of the model checking tool Uppaal. Uppaal is an integrated tool environment that allows users to model the behavior of systems in terms of states and transitions between states, and to simulate and analyze the resulting models. Uppaal can also handle real-time issues, that is, the timing of transitions. Using an example of a jobshop, we explain in a step by step manner how one can make a simple Uppaal model, simulate its behavior and analyze its properties. This introduction is targeted at a broad audience, ranging from high school students to software engineers and researchers. We only require elementary knowledge of programming and mathematics. Although a rich theory of model checking has been developed over the last decades, which includes both clever algorithms and deep mathematics, this introduction focuses entirely on the



Chapter 1
More Features in UPPAAL

Alexandre David, Kim G. Larsen

Abstract Following the introduction to the model checking tool UPPAAL of th previous chapter, this chapter presents a number of additional modeling and verification features offered by the tool. These features include in particular a C-lik imperative language with user-defined types and functions, allowing for readabl and compact models with reusable updates of discrete variables. Using an exampl of a Train Gate, we demonstrate the use(fulness) of these features. Also, the chapte presents the full query language of UPPAAL covering both safety, liveness and time bounded liveness properties, again illustrated using the Train Gate example. Finally directions are given on modelling choices and use of verification options that may improve time- and/or space-performance of the UPPAAL verifier

Origin of UPPAAL

TAU

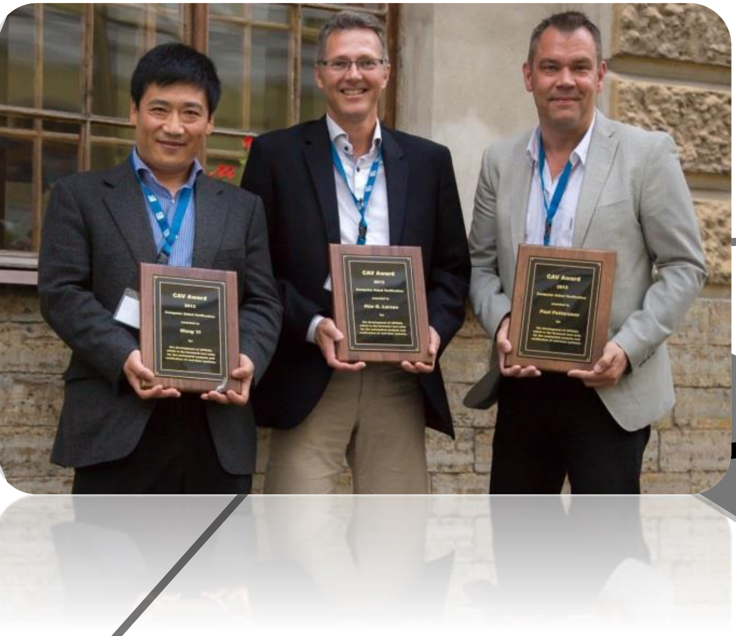
CCS & Modal Transition Systems

Refinements

Modal Mu-Calculus

Explicit State Representation

Prolog



1995

UPPAAL

Timed Automata

TCTL

Zones

C++ & Java

2007

EPSILON

TCCS

Timed Refinements

Timed Mu-Calculus

Regions

Prolog<

UP4ALL

2013

2016

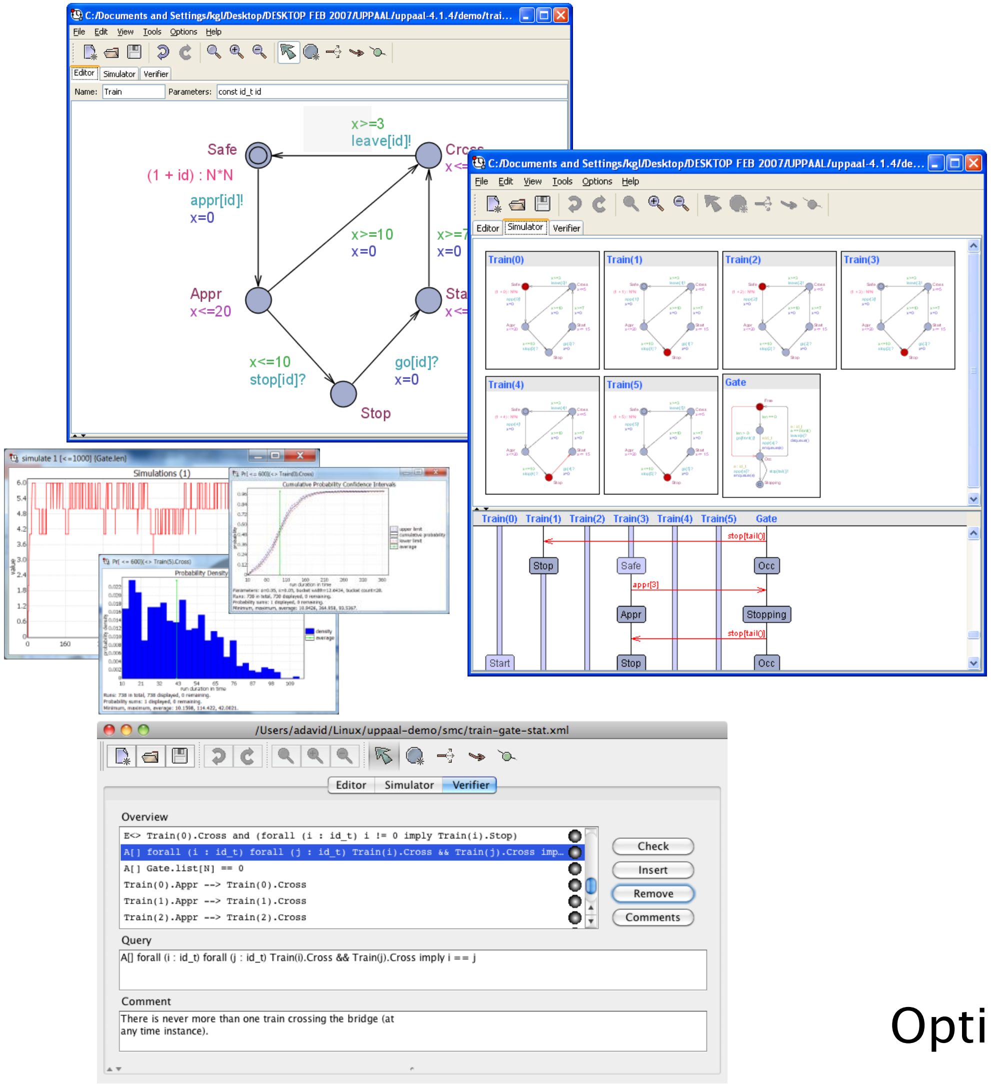
2018

CAV Award

Grundfos Prize

ATS

UPPAAL Tool Suit



C:/Documents and Settings/kgl/Desktop/DESKTOP FEB 2007/UPPAAL/uppaal-4.1.4/demo/trai...
File Edit View Tools Options Help
Editor Simulator Verifier
Name: Train Parameters: const id_t id
Safe
(1 + id) : N*N
appr[id]!
x=0
x>=3
leave[id]!
Cross
x>=10
x=0
x>=7
x=0
Appr
x<=20
Start
x<=10
stop[id]?
go[id]?
x=0
Stop
C:/Documents and Settings/kgl/Desktop/DESKTOP FEB 2007/UPPAAL/uppaal-4.1.4/de...
Train(0) Train(1) Train(2) Train(3) Train(4) Train(5) Gate
stop[tail()]
Stop
Safe
Occ
appr[3]
Appr
Stopping
Start
simulate 1 [<=1000] {Gate.len}
Simulations (1)
Pr[<= 600](<> Train(5).Cross)
Cumulative Probability Confidence Intervals
Probability Density
/Users/adavid/Linux/uppaal-demo/smc/train-gate-stat.xml
Editor Simulator Verifier
Check
Insert
Remove
Comments

Overview

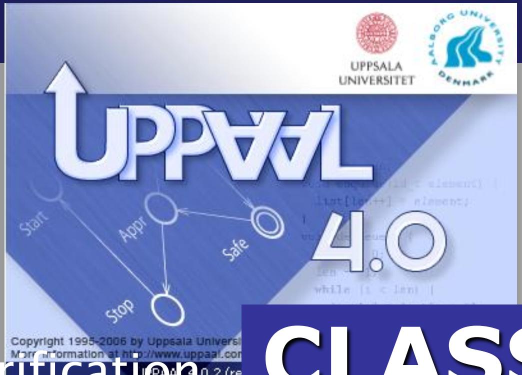
```
E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)
A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imp…
A[] Gate.list[N] == 0
Train(0).Appr --> Train(0).Cross
Train(1).Appr --> Train(1).Cross
Train(2).Appr --> Train(2).Cross
```

Query

```
A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j
```

Comment

There is never more than one train crossing the bridge (at any time instance).



UPPSALA UNIVERSITET
AALBORG UNIVERSITY DENMARK
UPPAAL
4.0
Start
Appr
Safe
Stop
Copyright 1995-2006 by Uppsala Universi
More information at http://www.uppaal.com
UPPAAL 4.0.2 (re

Verification	CLASSIC	1995
Optimization	CORA	2001
Synthesis	TIGA	2005
Component	ECDAR	2010
Testing	TRON	2004
Performance Analysis	SMC	2011
Optimal Synthesis	STRATEGO	2014

Contributors

@UPPsala

UPPSALA UNIVERSITET

- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@AALborg

AALBORG UNIVERSITY DENMARK

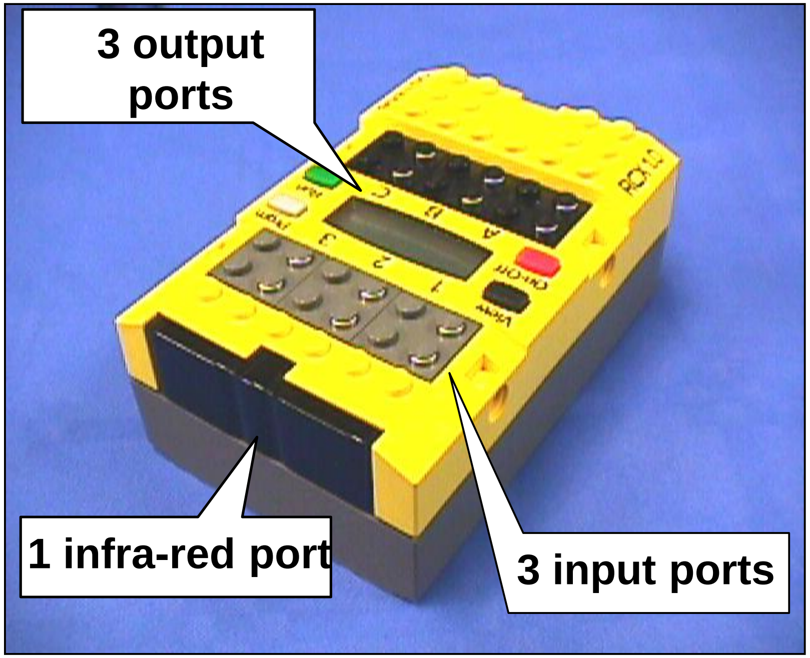
- Kim G Larsen
- Alexandre David
- Gerd Behrman
- Arne Skou
- Brian Nielsen
- Jacob I. Rasmussen
- Marius Mikucionis
- Thomas Chatain

@Elsewhere

- Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen, Jan Tretmans, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson...

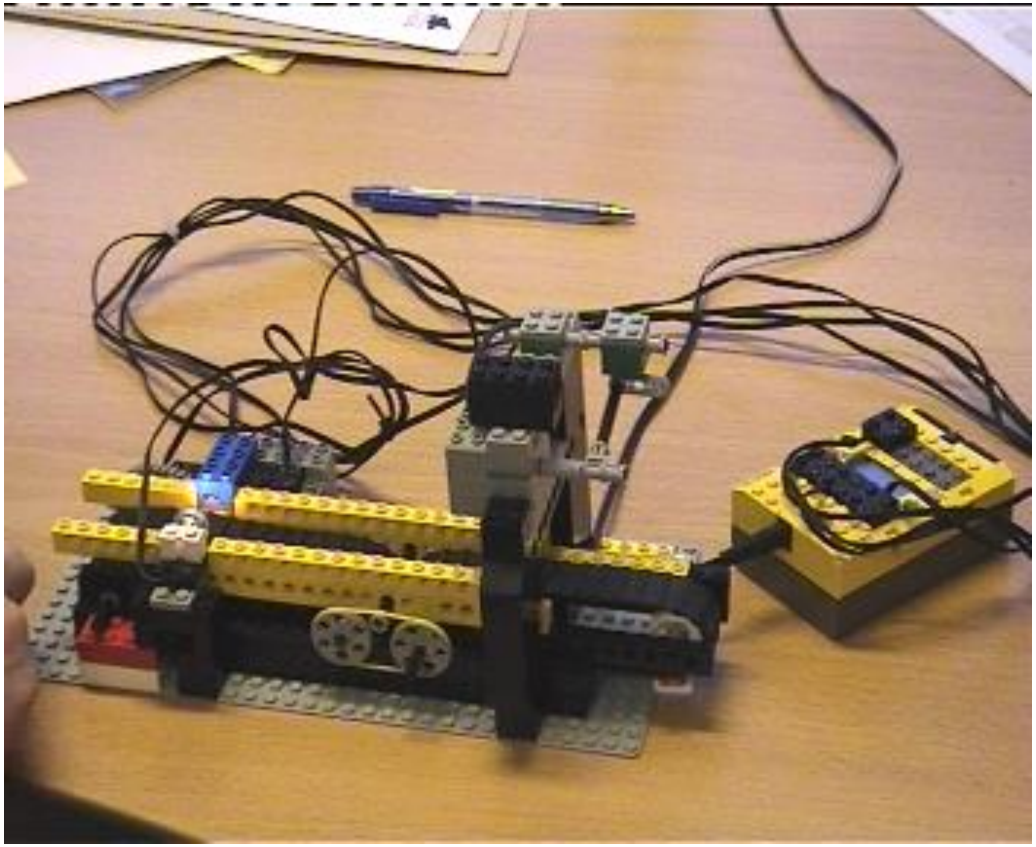
LEGO Mindstorms/RCX

- Sensors: temperature, light, rotation, pressure.
- Actuators: motors, lamps,
- Virtual machine:
  - 10 tasks, 4 timers, 16 integers.
- Several Programming Languages:
  - NotQuiteC, Mindstorm, Robotics, legOS, etc.



A Real Real Timed System

The Plant
Conveyor Belt
&
Bricks

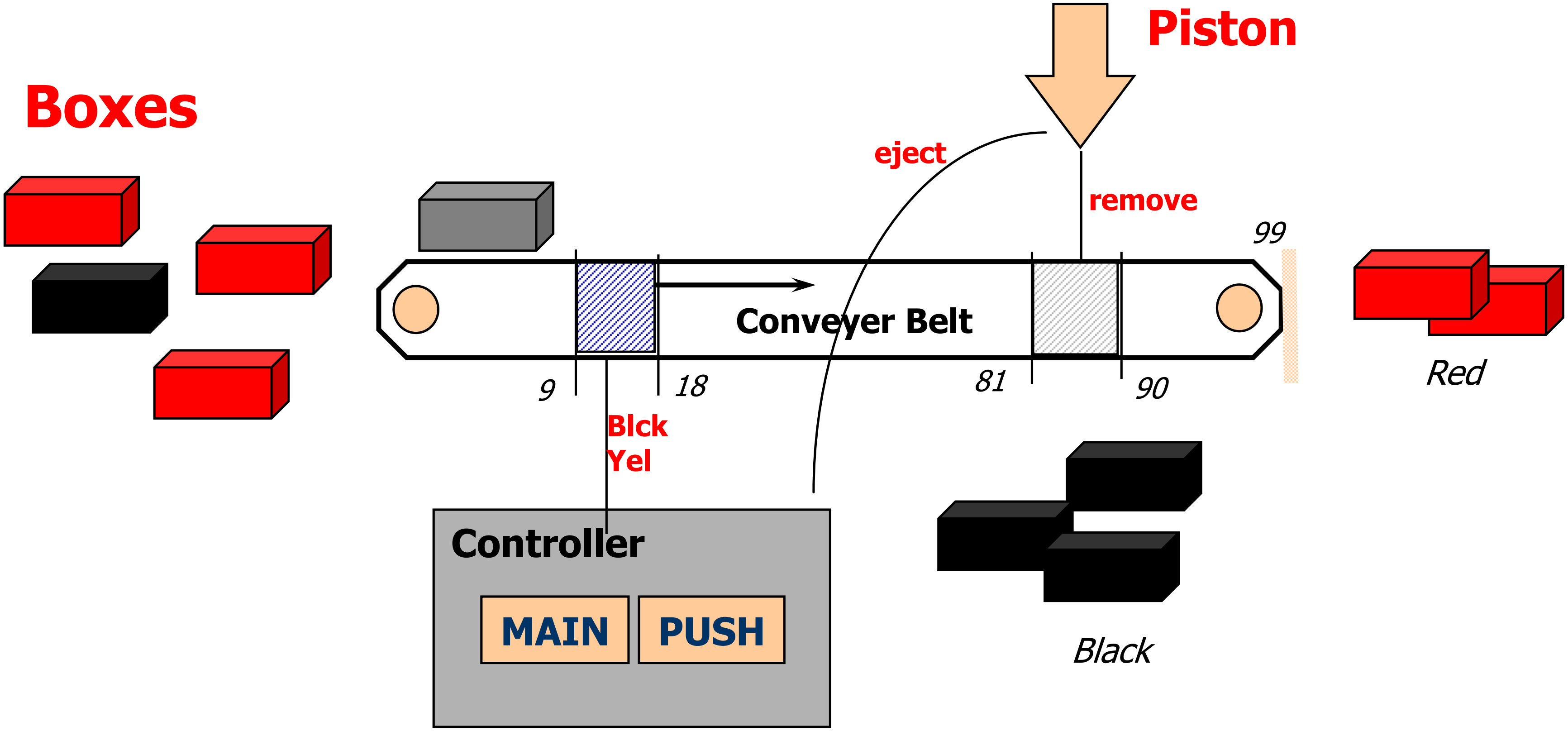


Controller Program
LEGO MINDSTORM

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



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

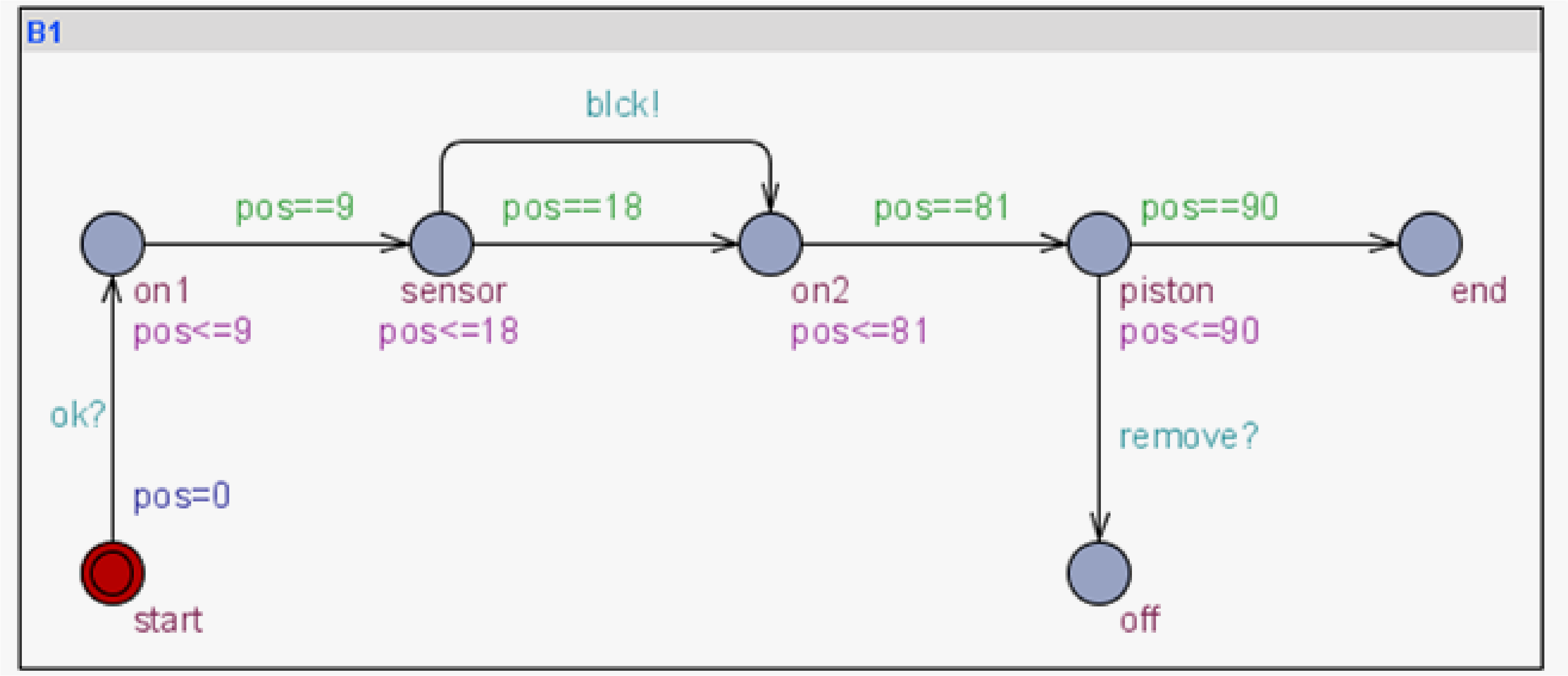
NQC programs

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

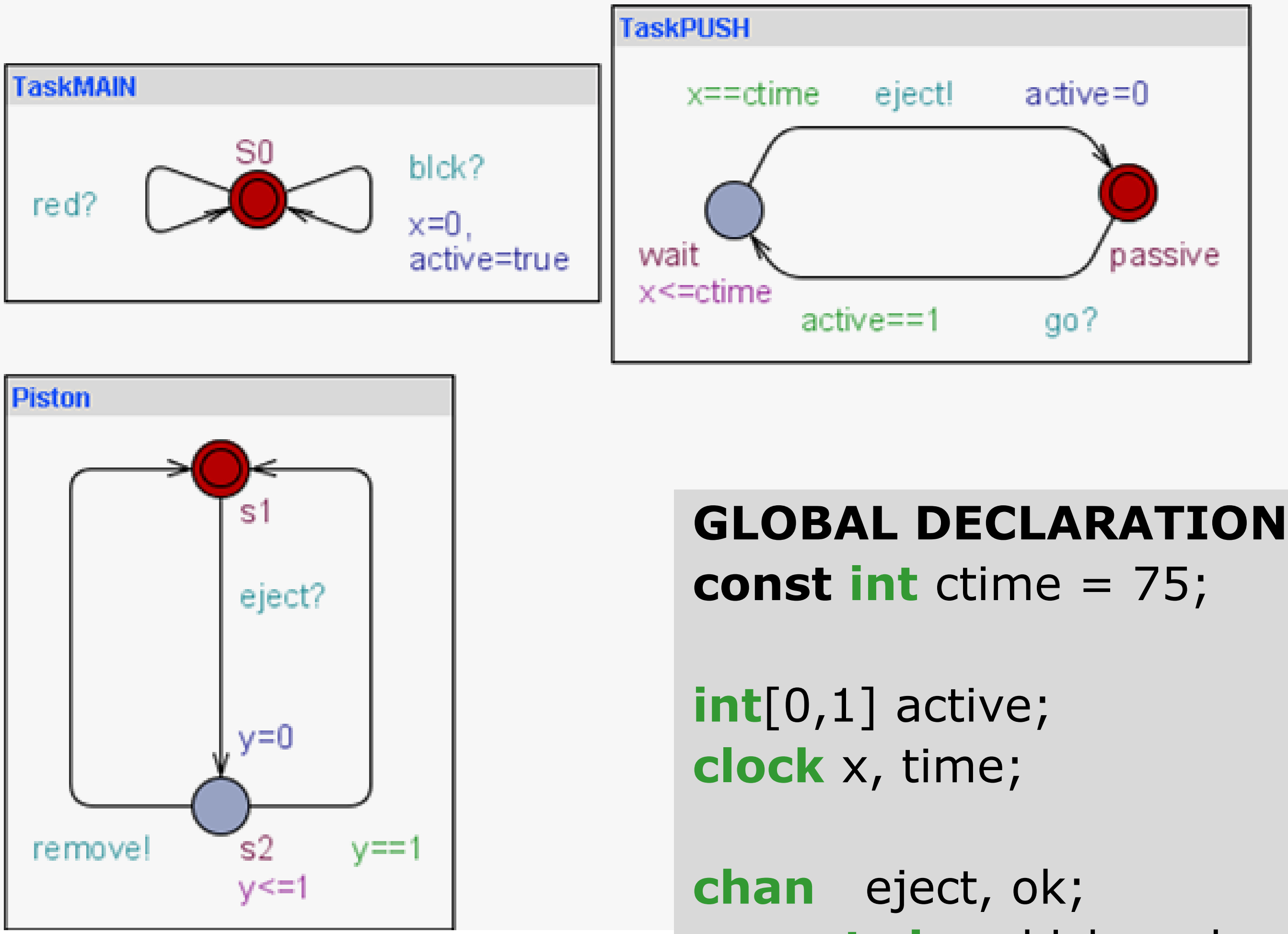
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 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);
  }
}
```

A Black Brick



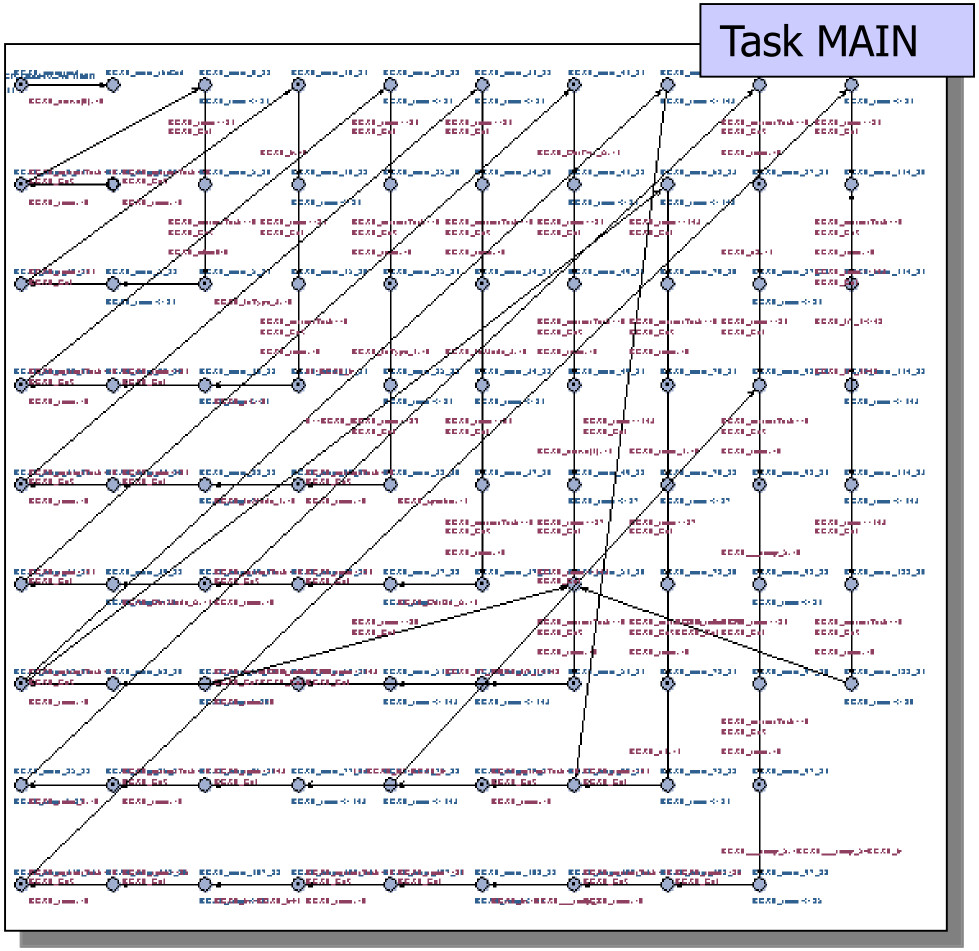
Control Tasks & Piston



```
GLOBAL DECLARATIONS:
const int ctime = 75;

int[0,1] active;
clock x, time;

chan   eject, ok;
urgent chan blck, red, remove, go;
```

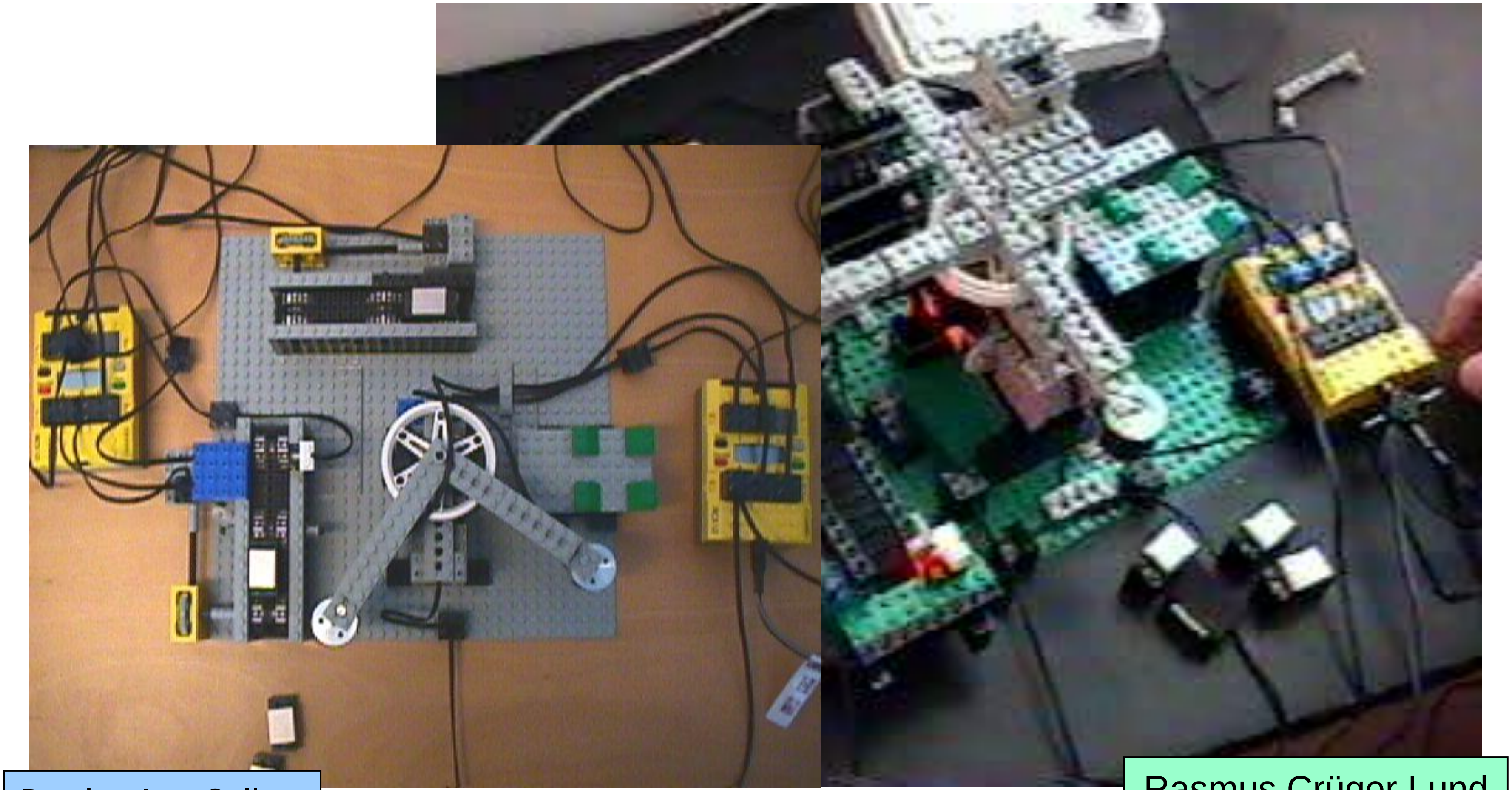
From RCX to UPPAAL – and back

- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000 in Stockholm.
- From UPPAAL to RCX: Martijn Hendriks.



The Production Cell in LEGO

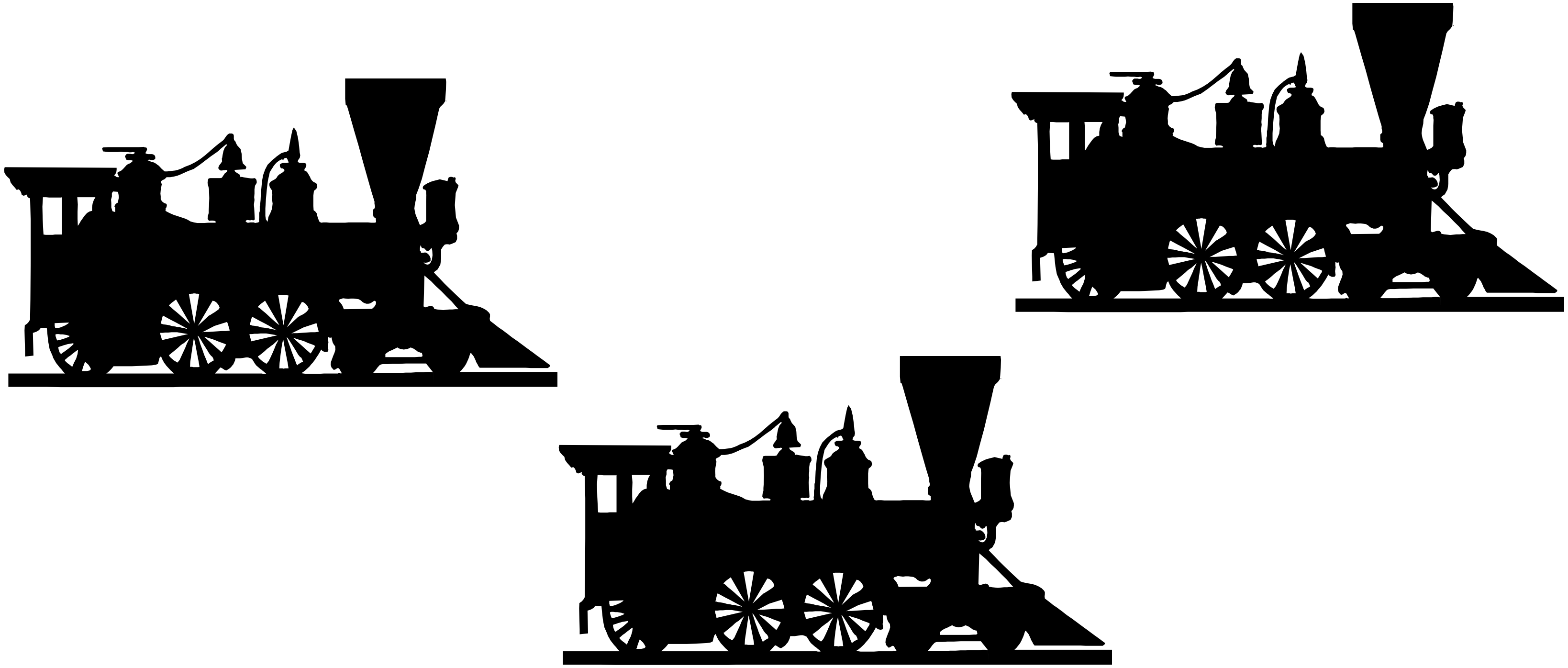
Course at DTU, Copenhagen



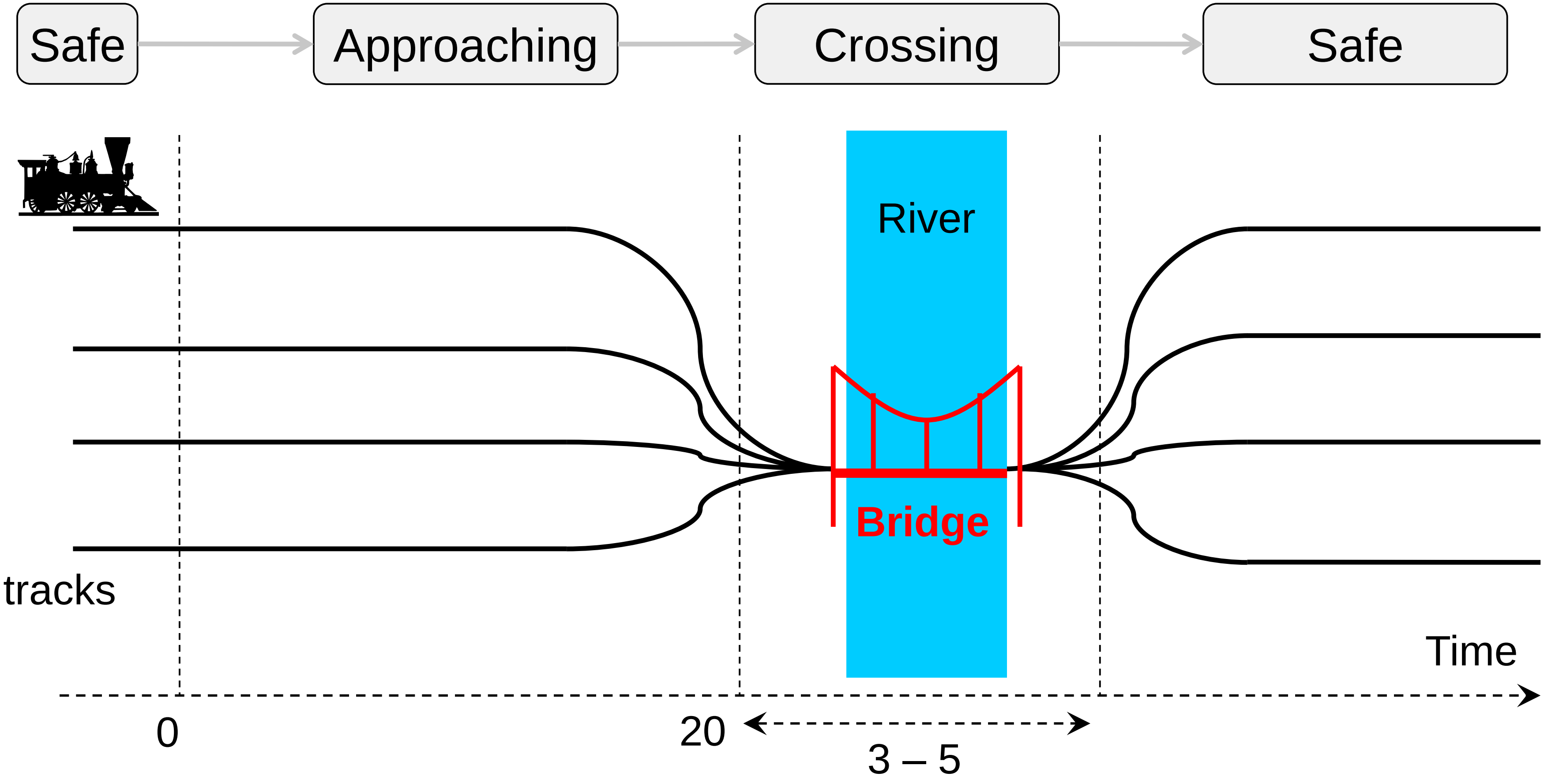
Production Cell

Rasmus Crüger Lund
Simon Tune Riemanni

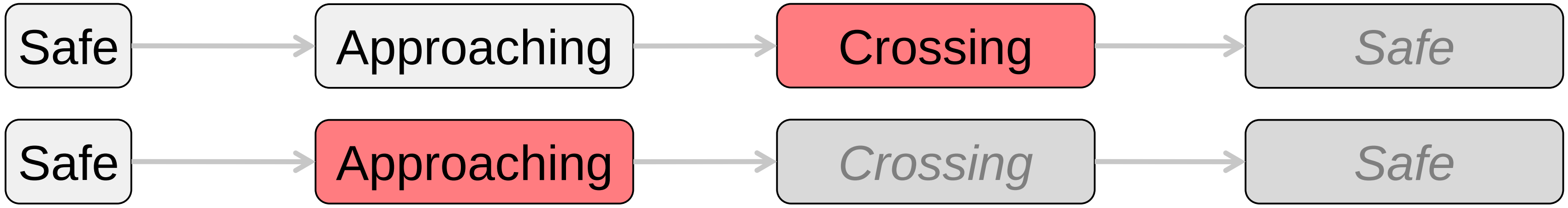
Train Crossing



Train Crossing

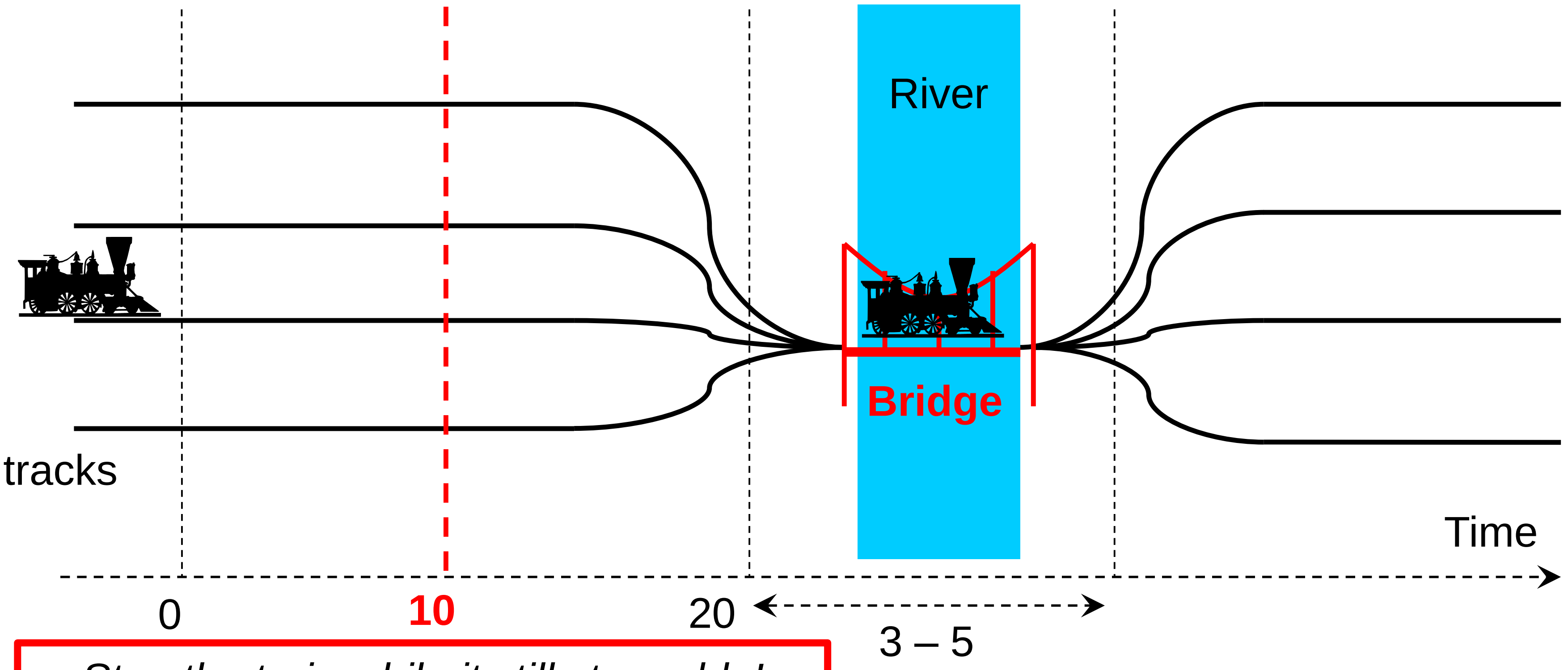


Train Crossing



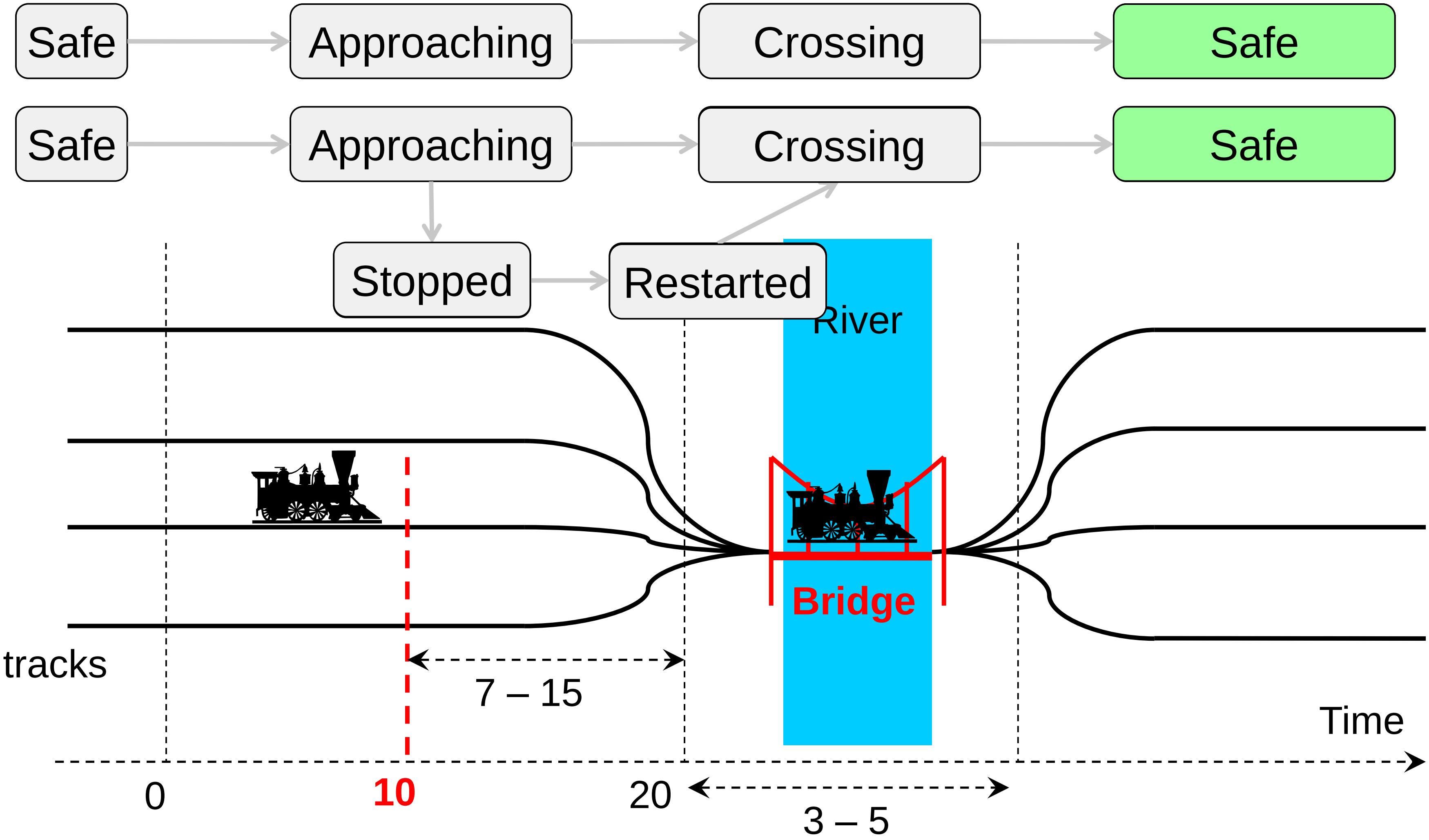
Safe → Approaching → Crossing → *Safe*

Safe → Approaching → *Crossing* → *Safe*

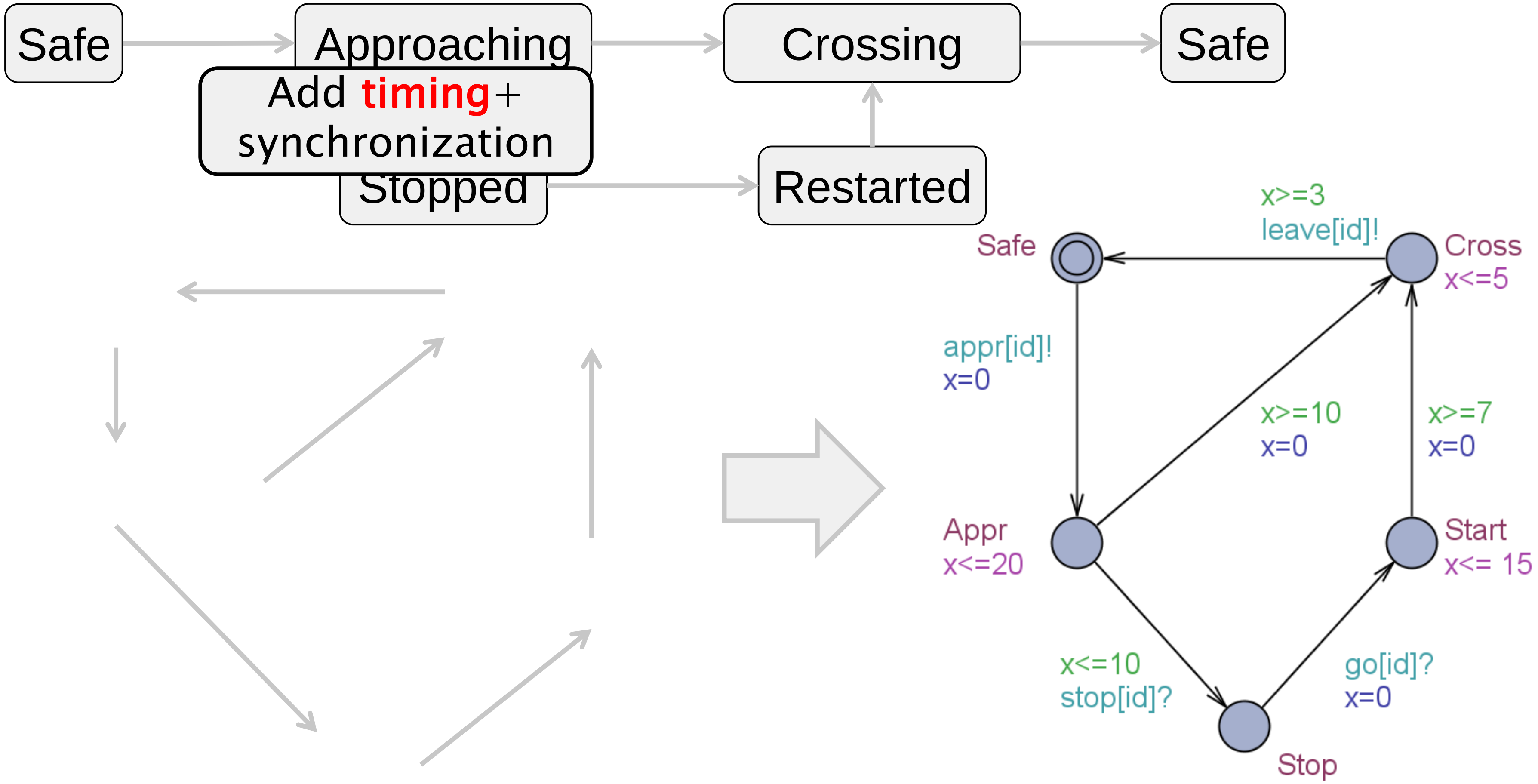


Stop the train while it still stoppable!

Train Crossing

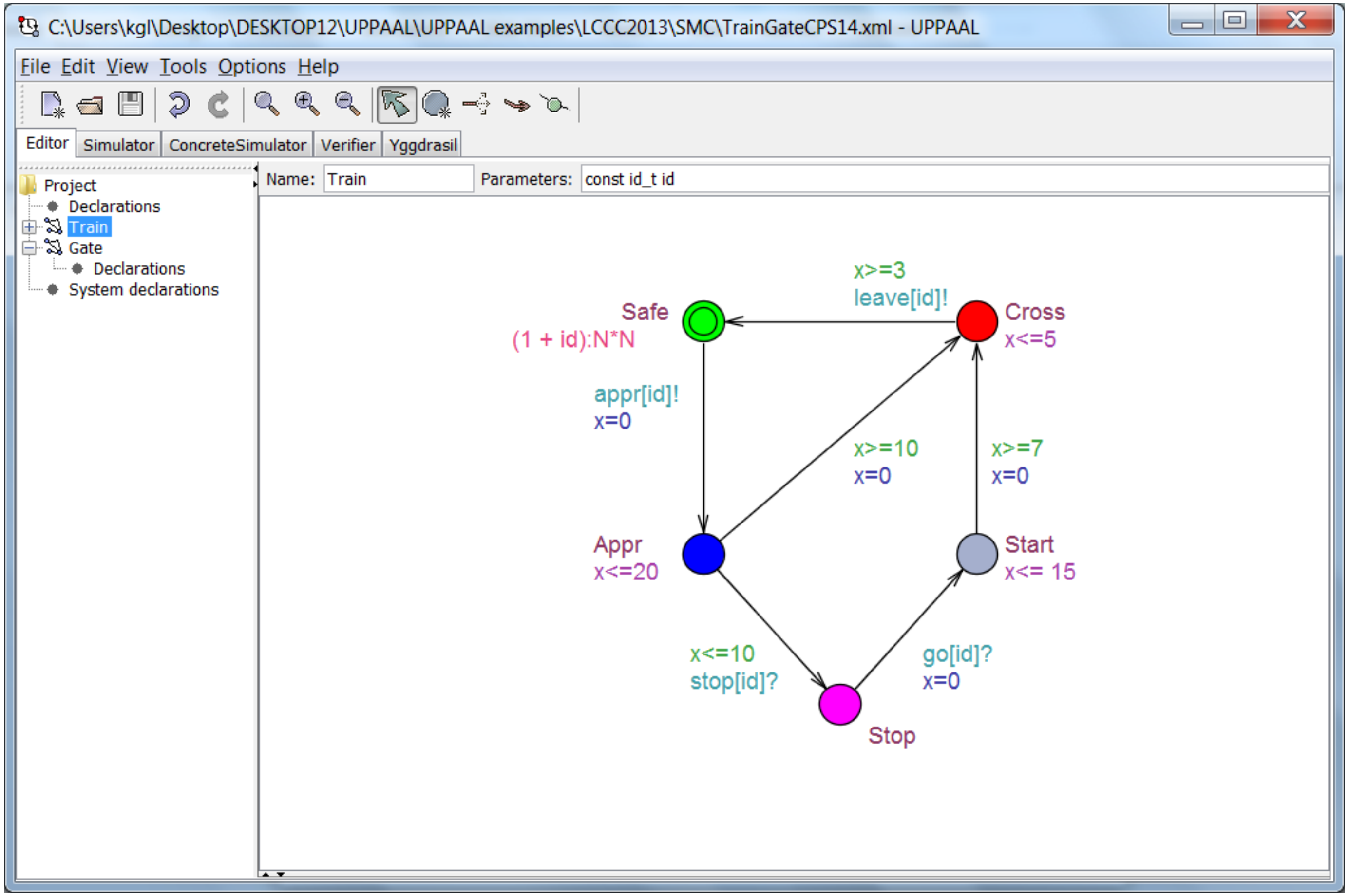


Train Crossing



Add **timing**+ synchronization

Demo



C:\Users\kgl\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS14.xml - UPPAAL

Name: Train Parameters: const id_t id

Logical Specifications

- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{.\ t} Q$

The expressions *P* and *Q* must be type safe, side effect free, and evaluate to a boolean.

Only references to integer variables, constants, clocks, are allowed (and arrays of these).

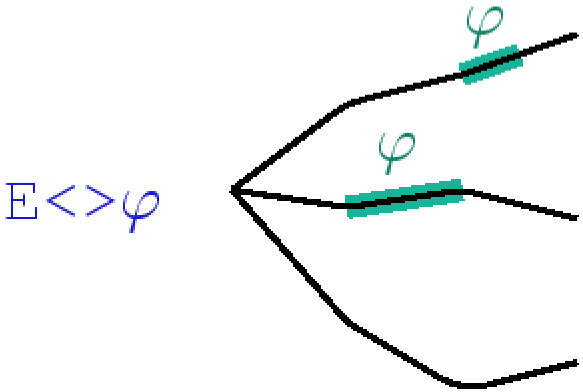
Logical Specifications

- Validation Properties
  - Possibly:
    E<> *P*

- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*

- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P
    → *Q*

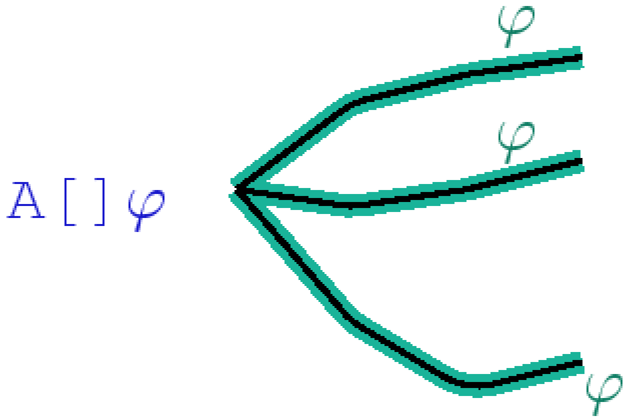
- Bounded Liveness
  - Leads to within: *P*$\rightarrow_{. t}$ *Q*

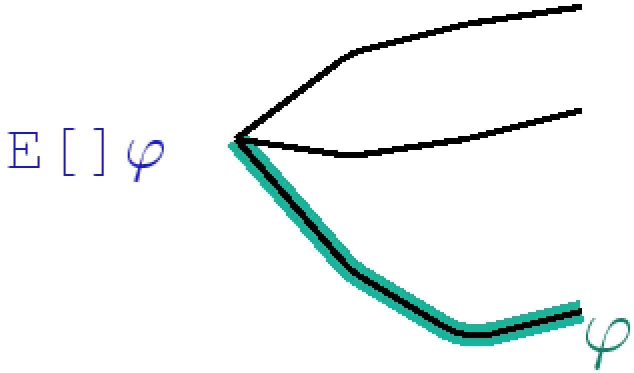


E A path
<> [] state

Logical Specifications

- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{.\ t} Q$





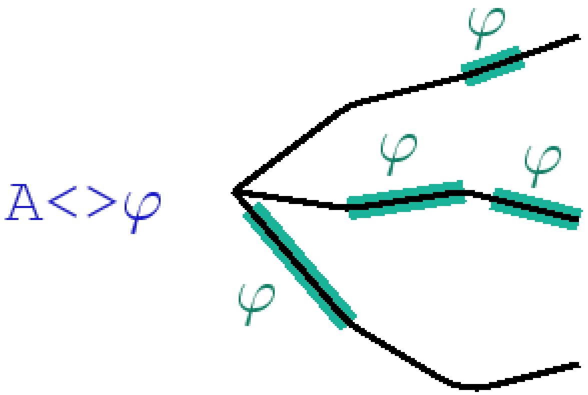
Logical Specifications

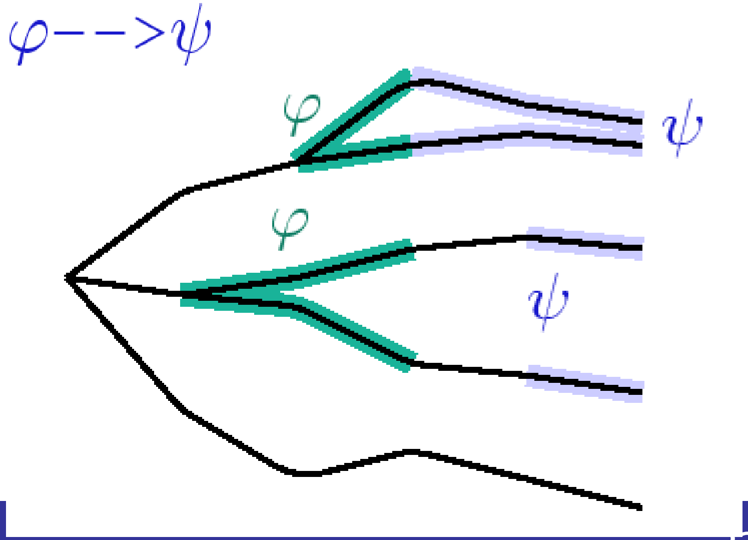
- Validation Properties
  - Possibly:
    E<> *P*

- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*

- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*

- Bounded Liveness
  - Leads to within: *P*→$_{.\ t}$ *Q*





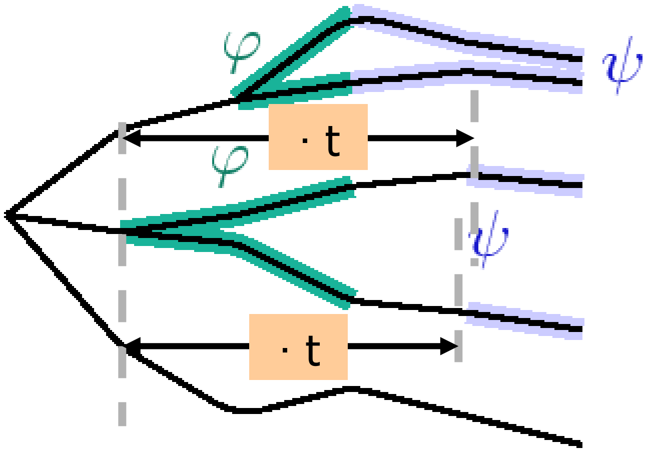
Logical Specifications

- Validation Properties
  - Possibly:
    E<> *P*

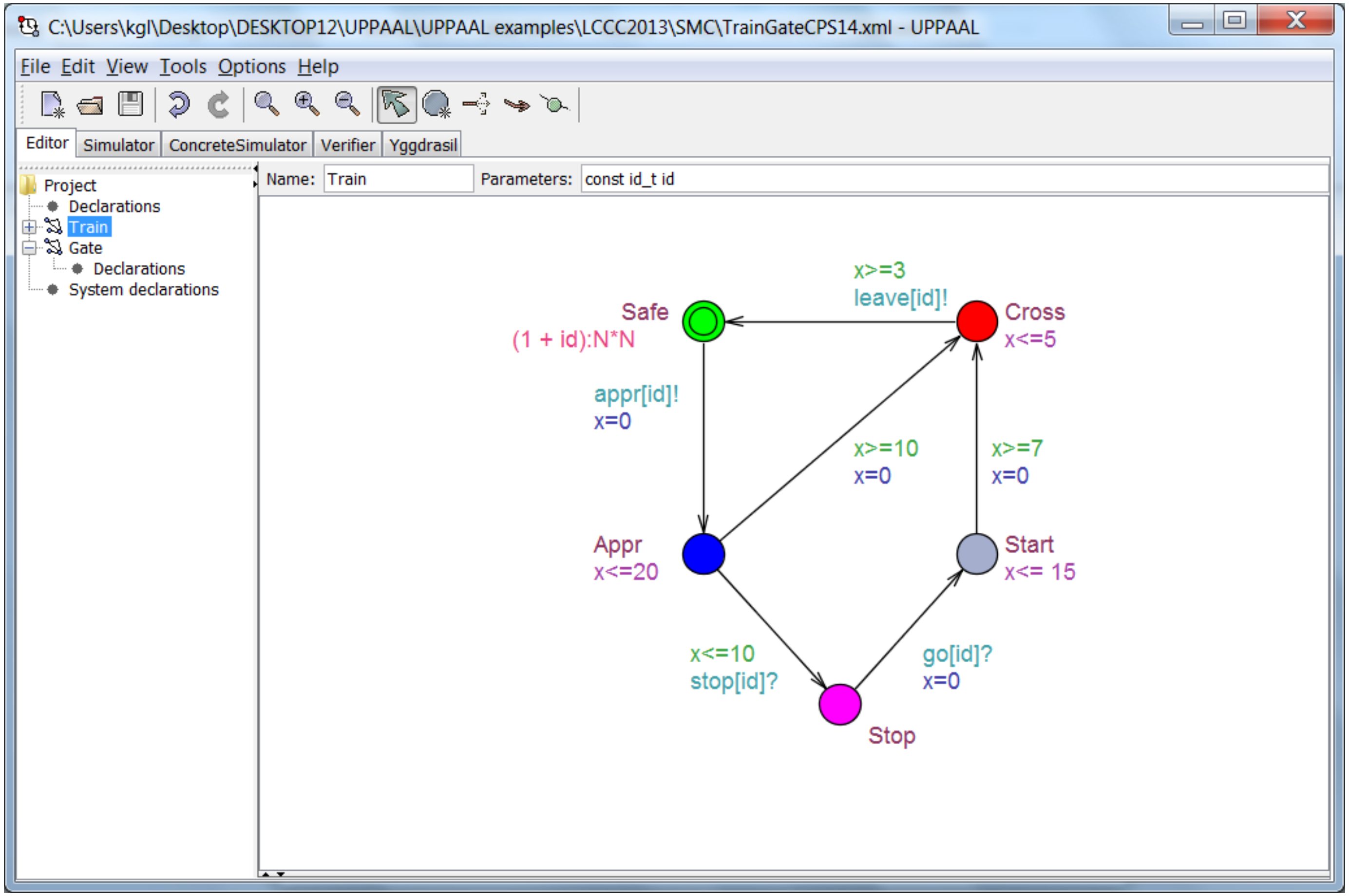
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*

- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*

- Bounded Liveness
  - Leads to within: $P \rightarrow_{\cdot\ t} Q$



Demo



C:\Users\kgl\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS14.xml - UPPAAL

Name: Train Parameters: const id_t id

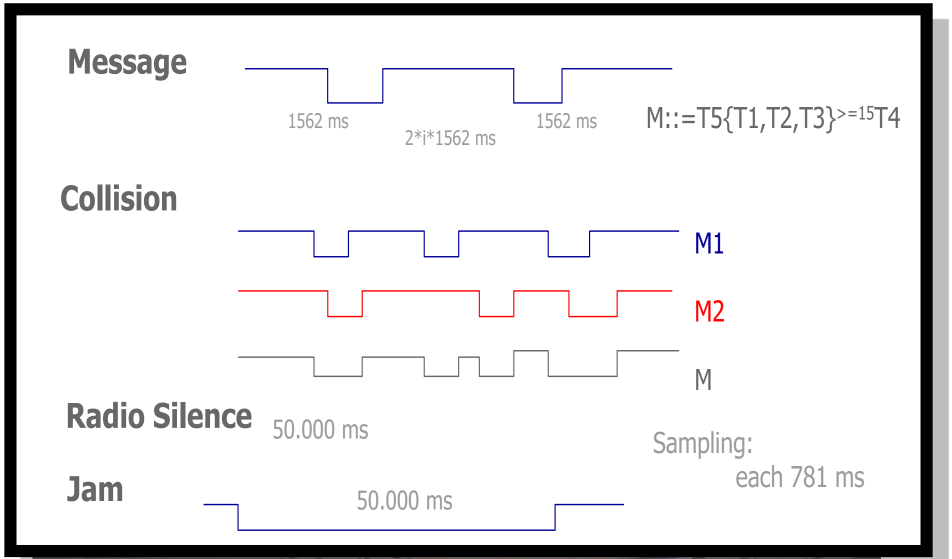
Bang & Olufsen IR-Link

- Bug known to exist for 10 years
- Ill-described:
  2.800 lines of assembler code + 3 flowchart + 1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.

Arne Skou, Klaus Havelund

Bang & Olufsen IR–Link

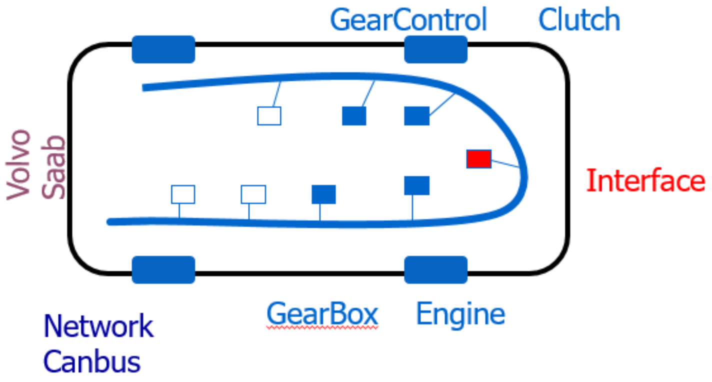
- Bug known to exist for 10 years
- Ill–described:
  2.800 lines of assembler code + 3 flowchart + 1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.



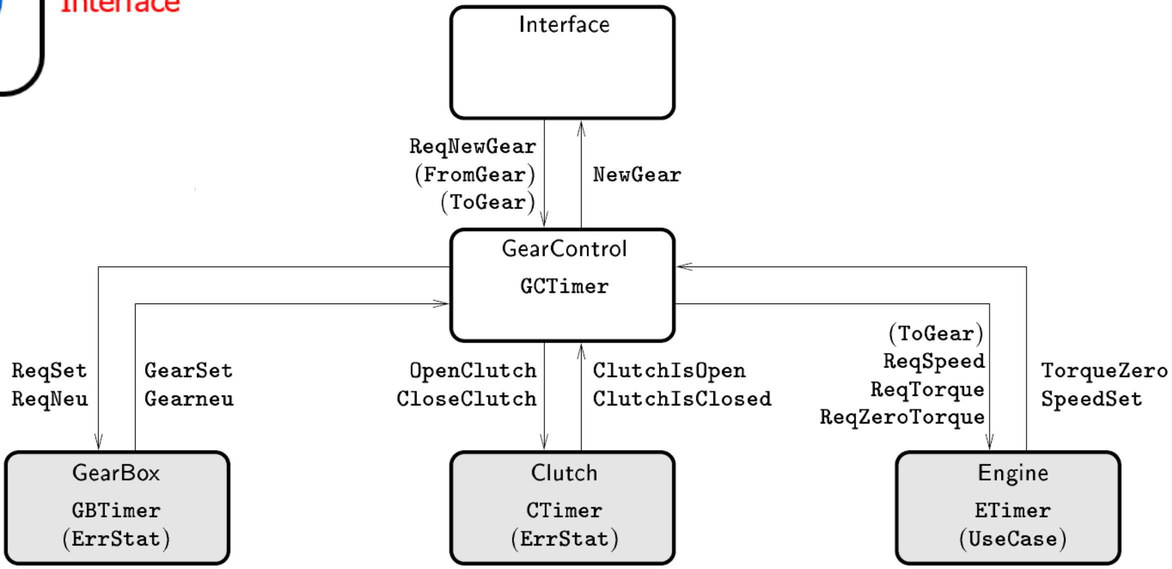
1st RTSS'97 talk, Klaus Havelund

Gear Controller

with MECEL AB



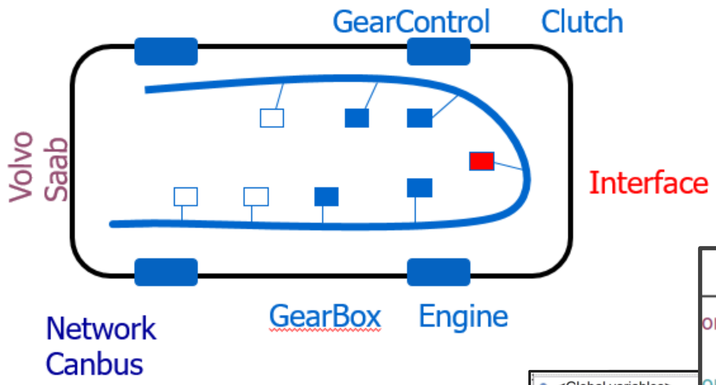
Flowgraph



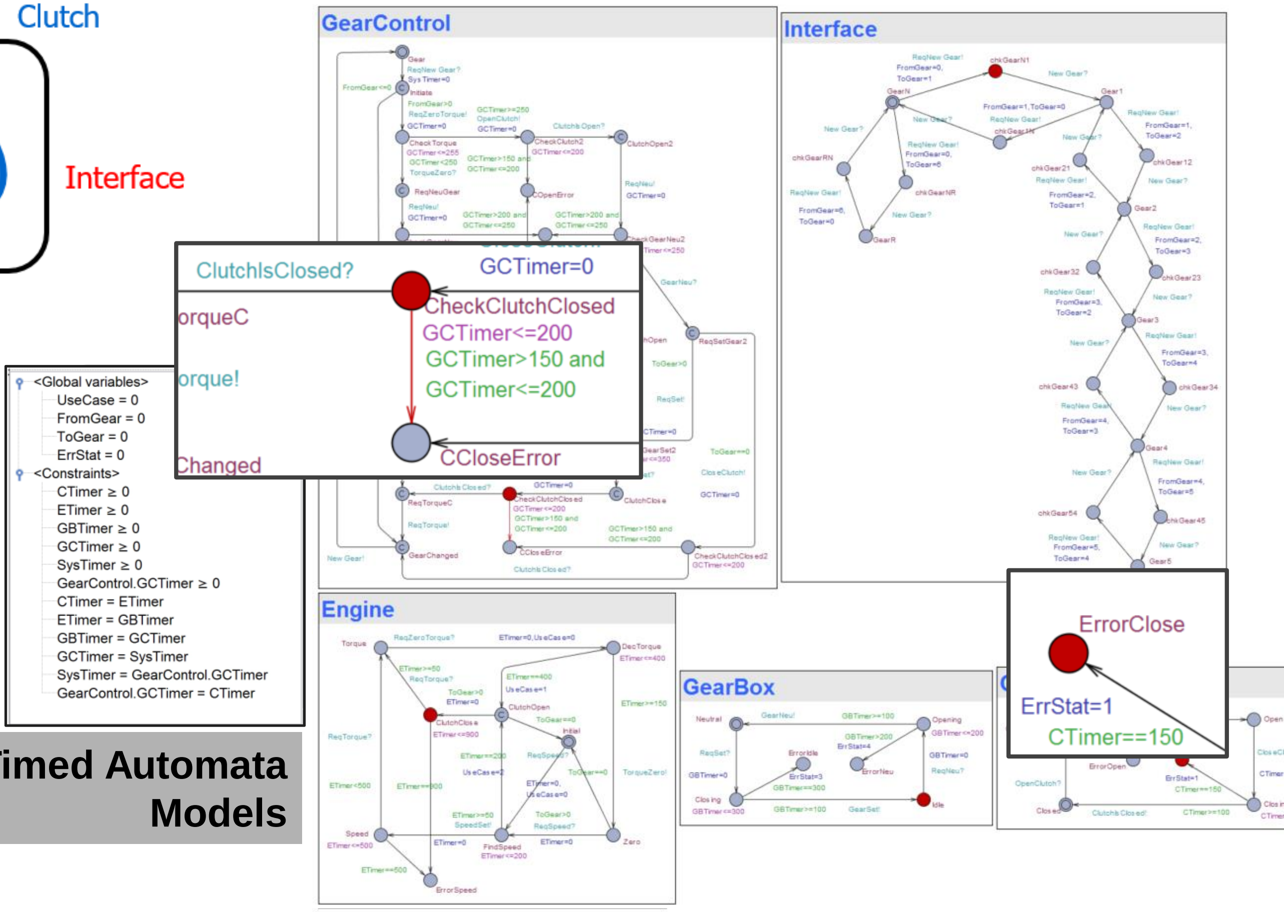
Magnus Lindahl
Paul Pettersson
Wang Yi
2001

Gear Controller

with MECEL AB



Magnus Lindahl
Paul Pettersson
Wang Yi
2001

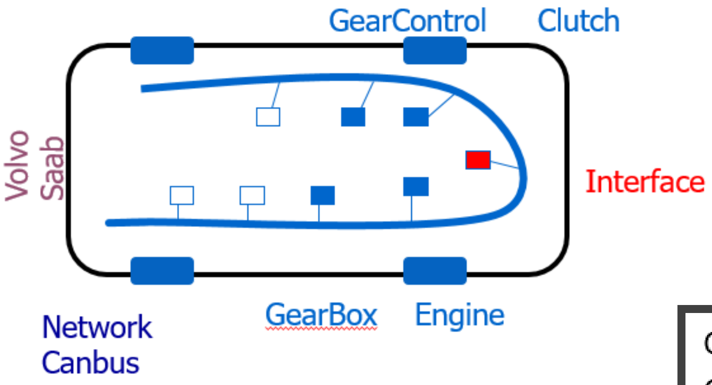


<Global variables>
UseCase = 0
FromGear = 0
ToGear = 0
ErrStat = 0
<Constraints>
CTimer ≥ 0
ETimer ≥ 0
GBTimer ≥ 0
GCTimer ≥ 0
SysTimer ≥ 0
GearControl.GCTimer ≥ 0
CTimer = ETimer
ETimer = GBTimer
GBTimer = GCTimer
GCTimer = SysTimer
SysTimer = GearControl.GCTimer
GearControl.GCTimer = CTimer

Timed Automata Models

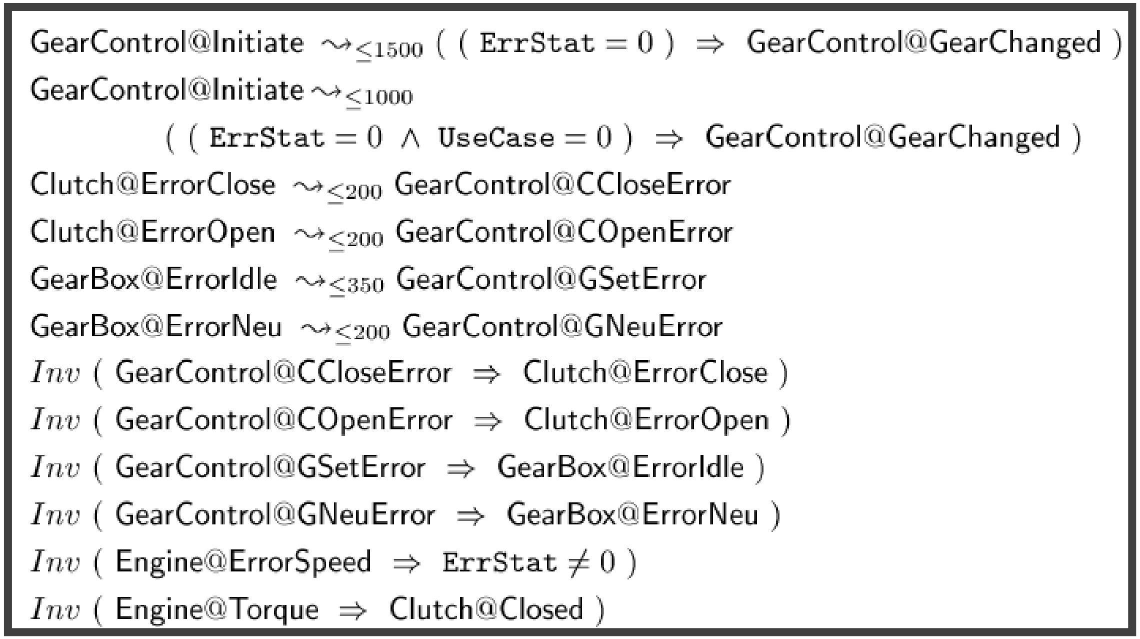
Gear Controller

with MECEL AB



Magnus Lindahl
Paul Pettersson
Wang Yi
2001

Requirements



$\text{GearControl@Initiate} \leadsto_{\leq 1500} ((\texttt{ErrStat} = 0) \Rightarrow \text{GearControl@GearChanged})$

$\text{GearControl@Initiate} \leadsto_{\leq 1000}$
$\quad ((\texttt{ErrStat} = 0 \wedge \texttt{UseCase} = 0) \Rightarrow \text{GearControl@GearChanged})$

$\text{Clutch@ErrorClose} \leadsto_{\leq 200} \text{GearControl@CCloseError}$

$\text{Clutch@ErrorOpen} \leadsto_{\leq 200} \text{GearControl@COpenError}$

$\text{GearBox@ErrorIdle} \leadsto_{\leq 350} \text{GearControl@GSetError}$

$\text{GearBox@ErrorNeu} \leadsto_{\leq 200} \text{GearControl@GNeuError}$

$Inv\ (\text{GearControl@CCloseError} \Rightarrow \text{Clutch@ErrorClose})$

$Inv\ (\text{GearControl@COpenError} \Rightarrow \text{Clutch@ErrorOpen})$

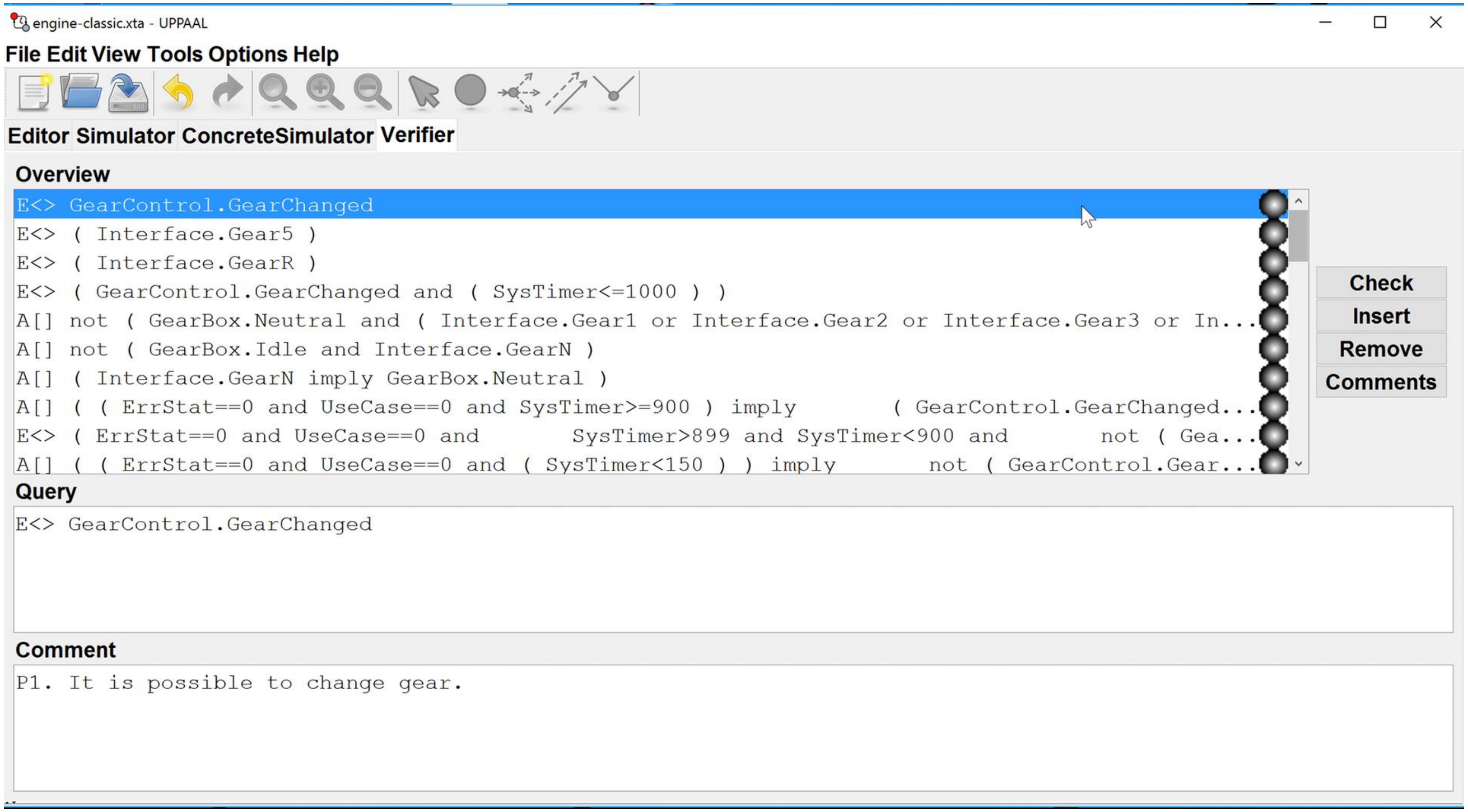
$Inv\ (\text{GearControl@GSetError} \Rightarrow \text{GearBox@ErrorIdle})$

$Inv\ (\text{GearControl@GNeuError} \Rightarrow \text{GearBox@ErrorNeu})$

$Inv\ (\text{Engine@ErrorSpeed} \Rightarrow \texttt{ErrStat} \neq 0)$

$Inv\ (\text{Engine@Torque} \Rightarrow \text{Clutch@Closed})$

UPPAAL Model Checking – Demo



engine-classic.xta - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier
Check
Insert
Remove
Comments

Overview

```
E<> GearControl.GearChanged
E<> ( Interface.Gear5 )
E<> ( Interface.GearR )
E<> ( GearControl.GearChanged and ( SysTimer<=1000 ) )
A[] not ( GearBox.Neutral and ( Interface.Gear1 or Interface.Gear2 or Interface.Gear3 or In...
A[] not ( GearBox.Idle and Interface.GearN )
A[] ( Interface.GearN imply GearBox.Neutral )
A[] ( ( ErrStat==0 and UseCase==0 and SysTimer>=900 ) imply       ( GearControl.GearChanged...
E<> ( ErrStat==0 and UseCase==0 and       SysTimer>899 and SysTimer<900 and       not ( Gea...
A[] ( ( ErrStat==0 and UseCase==0 and ( SysTimer<150 ) ) imply       not ( GearControl.Gear...
```

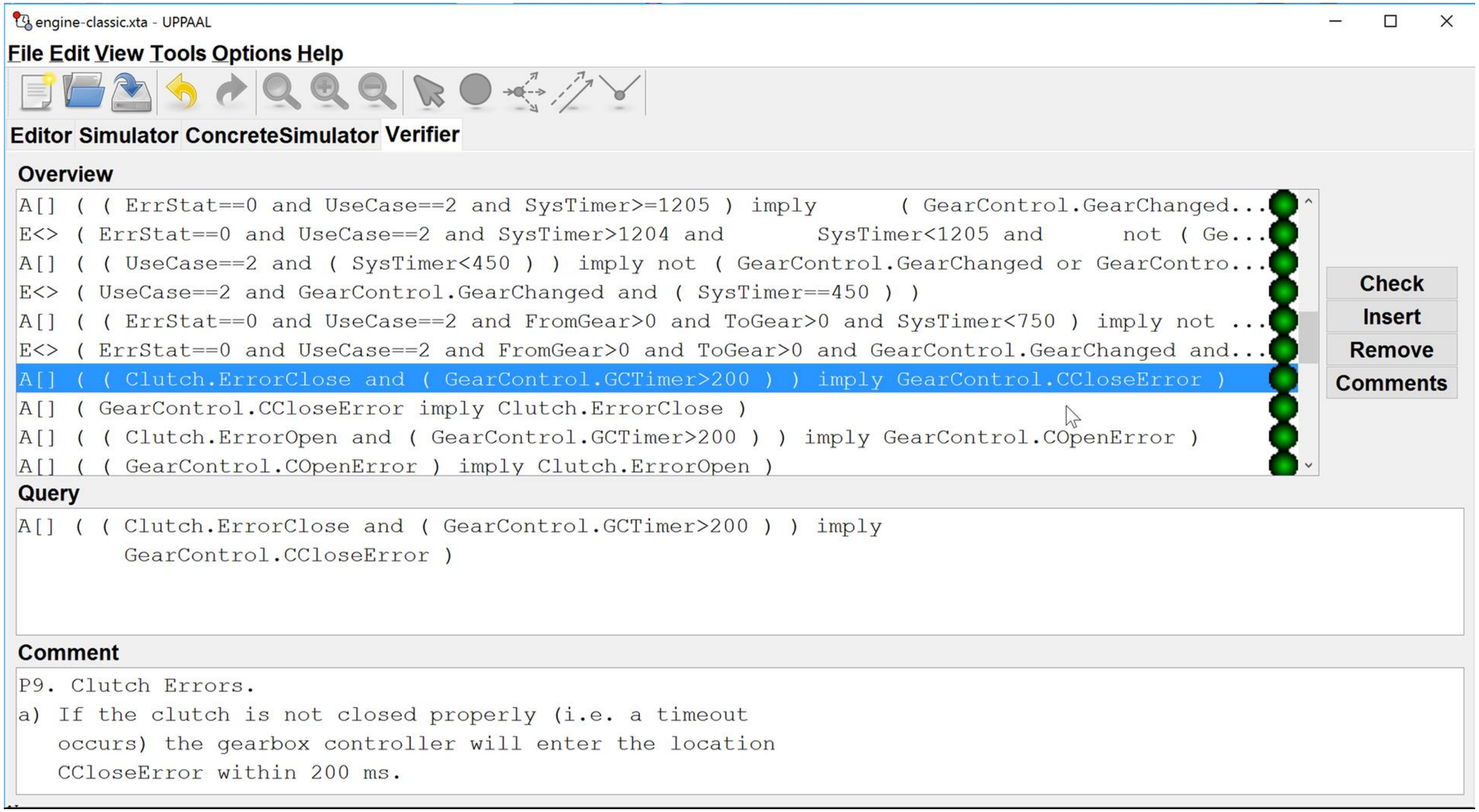
Query

```
E<> GearControl.GearChanged
```

Comment

P1. It is possible to change gear.

UPPAAL Model Checking – Demo


engine-classic.xta - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier
Check
Insert
Remove
Comments

Overview

```
A[] ( ( ErrStat==0 and UseCase==2 and SysTimer>=1205 ) imply      ( GearControl.GearChanged...
E<> ( ErrStat==0 and UseCase==2 and SysTimer>1204 and       SysTimer<1205 and       not ( Ge...
A[] ( ( UseCase==2 and ( SysTimer<450 ) ) imply not ( GearControl.GearChanged or GearContro...
E<> ( UseCase==2 and GearControl.GearChanged and ( SysTimer==450 ) )
A[] ( ( ErrStat==0 and UseCase==2 and FromGear>0 and ToGear>0 and SysTimer<750 ) imply not ...
E<> ( ErrStat==0 and UseCase==2 and FromGear>0 and ToGear>0 and GearControl.GearChanged and...
A[] ( ( Clutch.ErrorClose and ( GearControl.GCTimer>200 ) ) imply GearControl.CCloseError )
A[] ( GearControl.CCloseError imply Clutch.ErrorClose )
A[] ( ( Clutch.ErrorOpen and ( GearControl.GCTimer>200 ) ) imply GearControl.COpenError )
A[] ( ( GearControl.COpenError ) imply Clutch.ErrorOpen )
```

Query

```
A[] ( ( Clutch.ErrorClose and ( GearControl.GCTimer>200 ) ) imply
       GearControl.CCloseError )
```

Comment

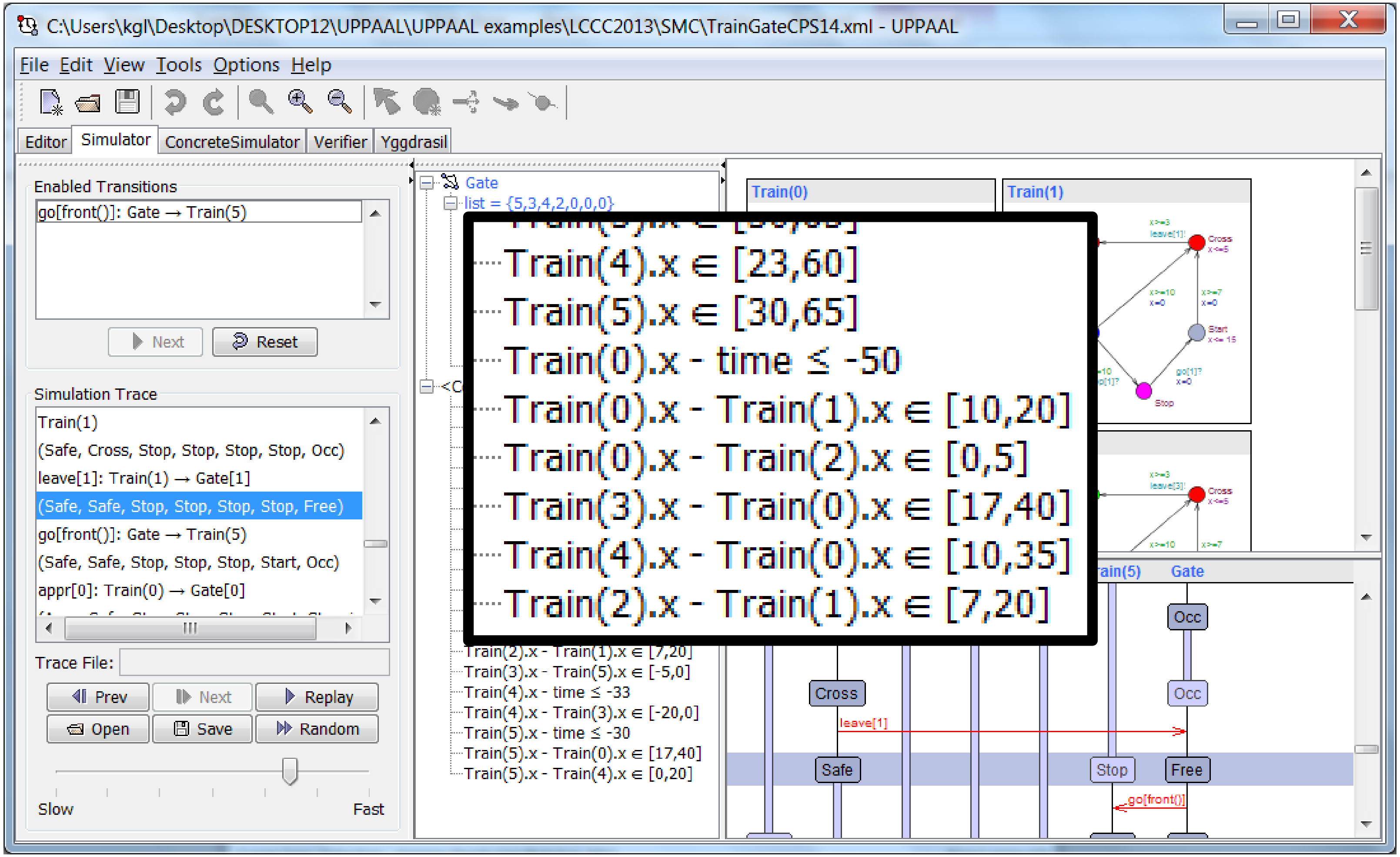
```
P9. Clutch Errors.
a) If the clutch is not closed properly (i.e. a timeout
   occurs) the gearbox controller will enter the location
   CCloseError within 200 ms.
```

(Wireless) Protocols in UPPAAL

- Bang & Olufsen IR Link
- Philips Audio Protocol
- Collision-Avoidance Protocol
- Bounded Retransmission Protocol
- TDMA Protocol
- Multimedia Streams
- ATM ABR Protocol
- Lamport's Leader Election Protocol
- ABB Fieldbus Protocol
- IEEE 1394 Firewire Root Contention
- Bluetooth Protocol
- Distributed Agreement Protocol
- FlexRay
- CHESS MAC Protocol
- Proprietary WSN, Other Big Danish Company
- MESH Protocol (MAC & Routing), NEOCORTEC

Engine & Options

The ”secret” of UPPAAL



Enabled Transitions:

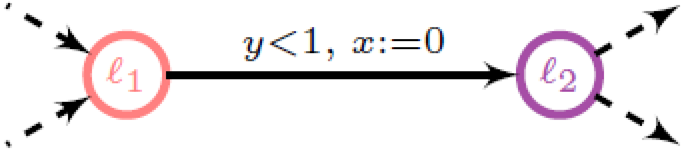
go[front()]: Gate → Train(5)

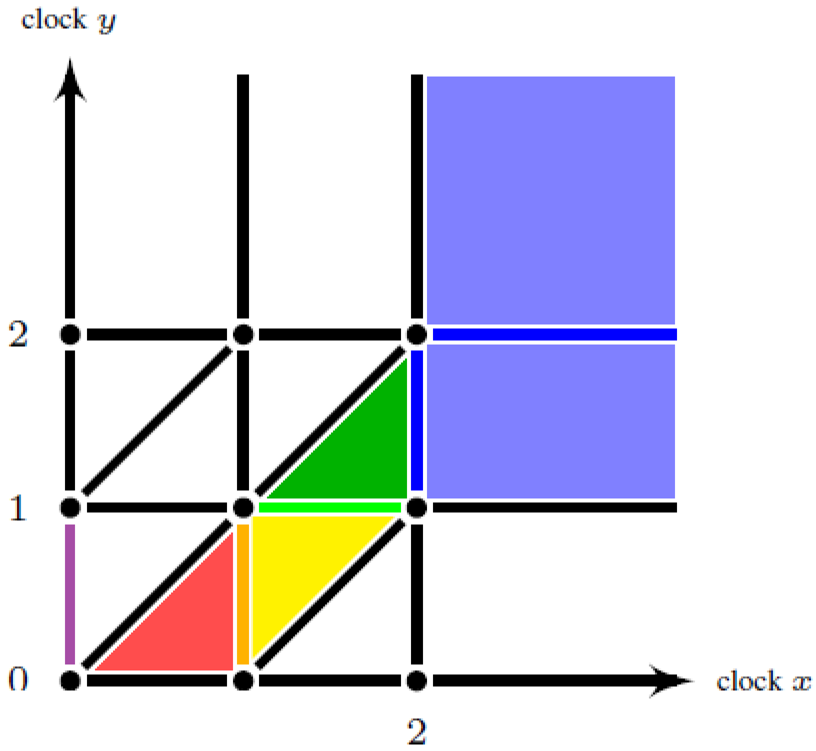
Simulation Trace:

```
Train(1)
(Safe, Cross, Stop, Stop, Stop, Stop, Occ)
leave[1]: Train(1) → Gate[1]
(Safe, Safe, Stop, Stop, Stop, Stop, Free)
go[front()]: Gate → Train(5)
(Safe, Safe, Stop, Stop, Stop, Start, Occ)
appr[0]: Train(0) → Gate[0]
```

```
Gate
list = {5,3,4,2,0,0,0}
Train(4).x ∈ [23,60]
Train(5).x ∈ [30,65]
Train(0).x - time ≤ -50
Train(0).x - Train(1).x ∈ [10,20]
Train(0).x - Train(2).x ∈ [0,5]
Train(3).x - Train(0).x ∈ [17,40]
Train(4).x - Train(0).x ∈ [10,35]
Train(2).x - Train(1).x ∈ [7,20]
Train(3).x - Train(5).x ∈ [-5,0]
Train(4).x - time ≤ -33
Train(4).x - Train(3).x ∈ [-20,0]
Train(5).x - time ≤ -30
Train(5).x - Train(0).x ∈ [17,40]
Train(5).x - Train(4).x ∈ [0,20]
```

Regions – From Infinite to Finite

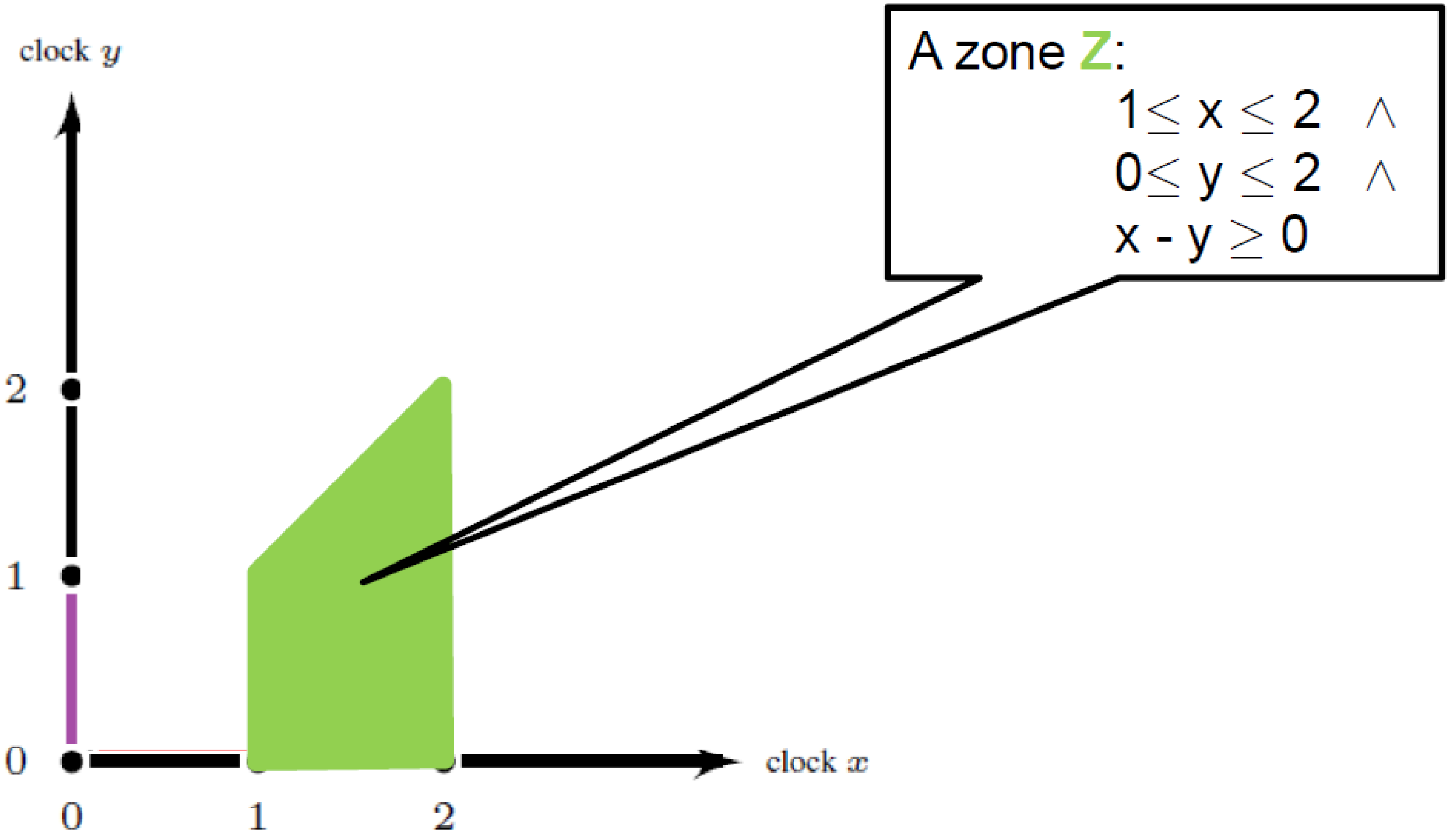




Theorem

The number of regions is $n! \cdot 2^n \cdot \prod_{x \in C}(2c_x + 2)$.

Zones – From Finite to Efficiency

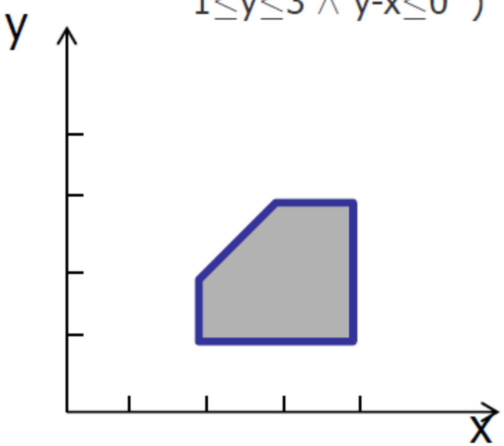


A zone **Z**:

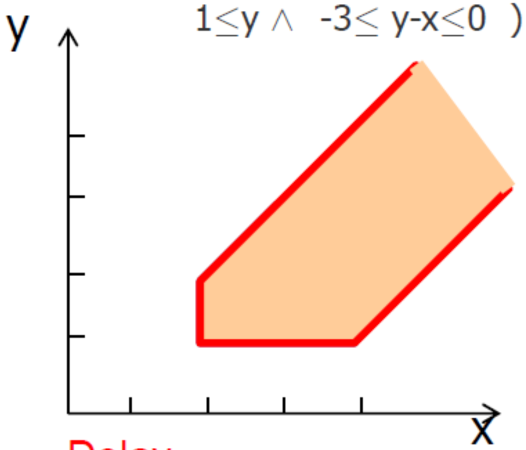
$$1 \leq x \leq 2 \quad \wedge$$
$$0 \leq y \leq 2 \quad \wedge$$
$$x - y \geq 0$$

Zones – Operations

$(n,\ 2 \leq x \leq 4 \wedge 1 \leq y \leq 3 \wedge y - x \leq 0\ )$

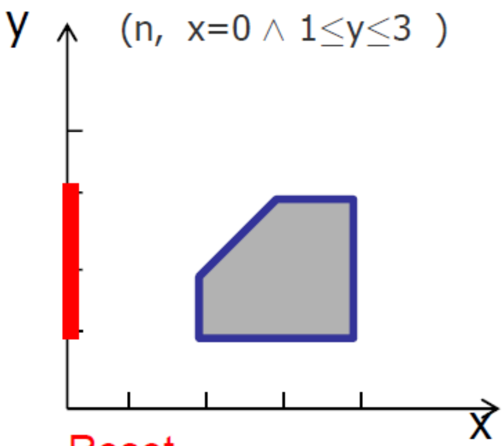


$(n,\ 2 \leq x \wedge 1 \leq y \wedge -3 \leq y - x \leq 0\ )$



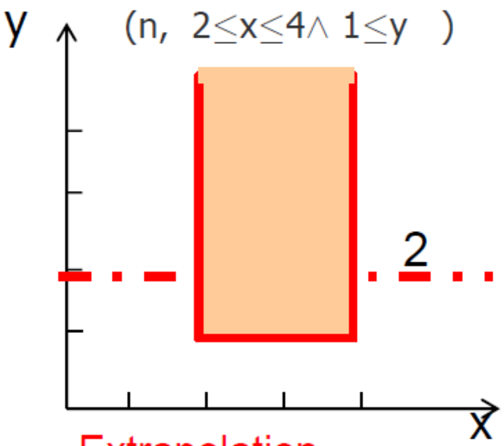
Delay

$(n,\ x = 0 \wedge 1 \leq y \leq 3\ )$



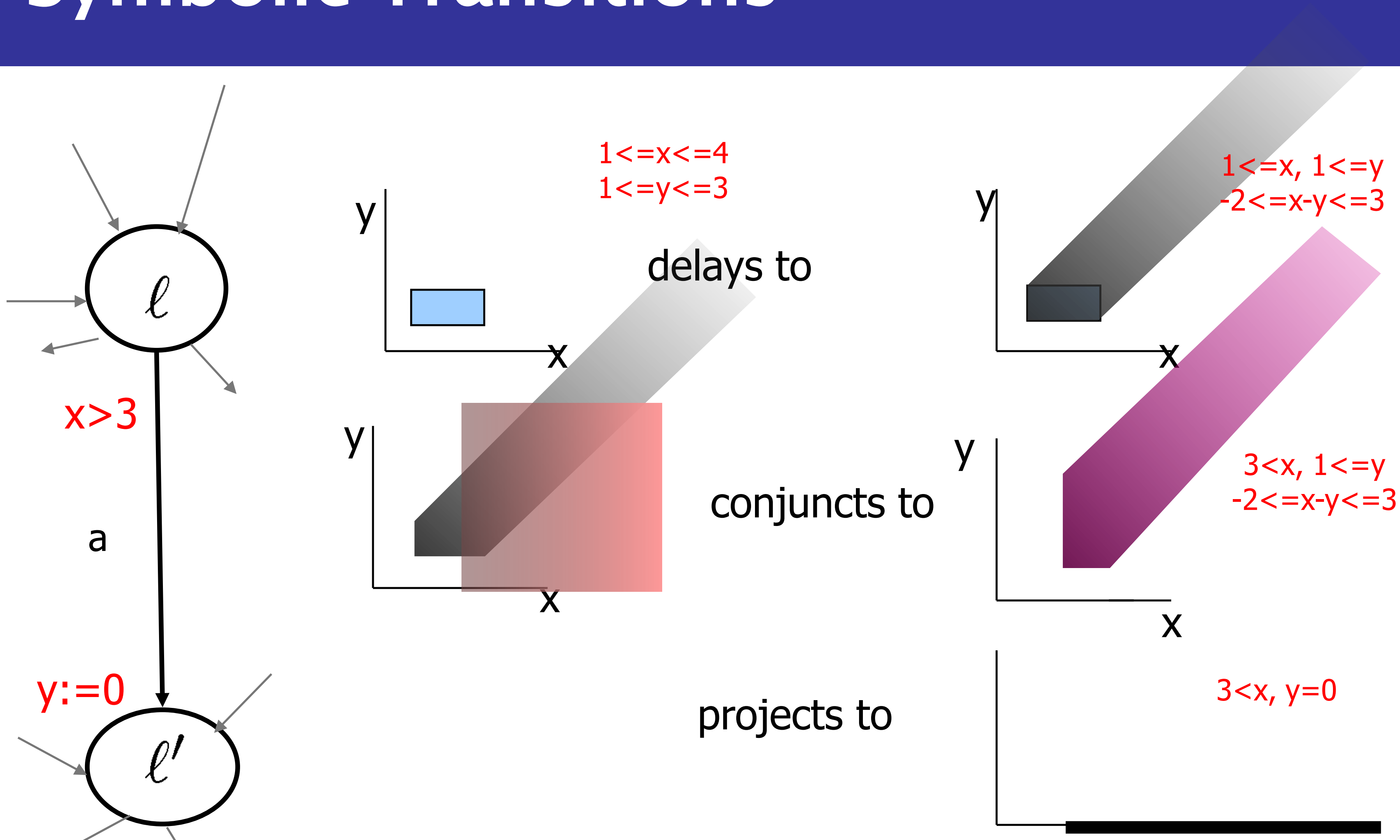
Reset

$(n,\ 2 \leq x \leq 4 \wedge 1 \leq y\ )$



Extrapolation

Symbolic Transitions



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

delays to

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

conjuncts to

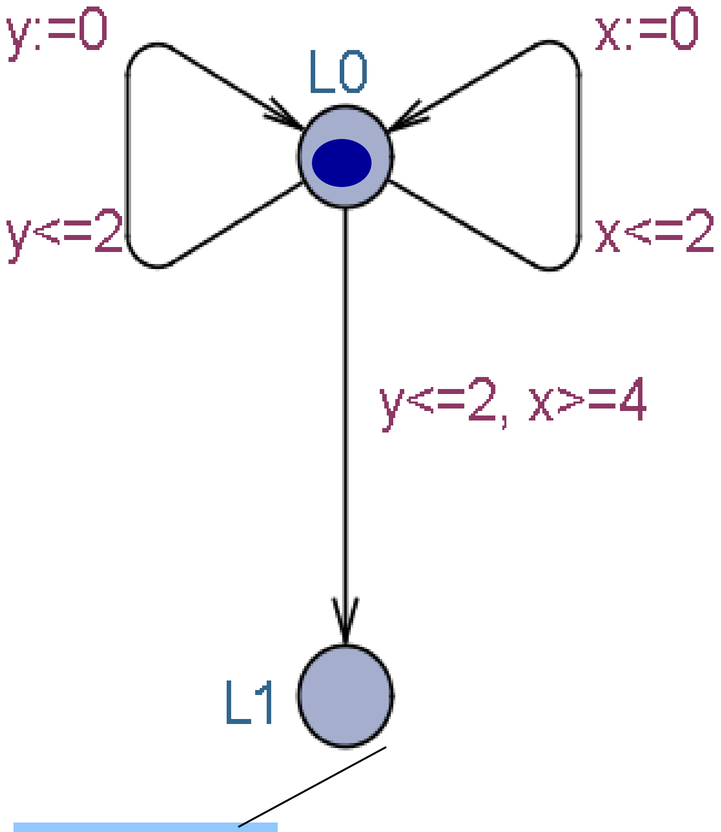
3<x, 1<=y
-2<=x-y<=3

projects to

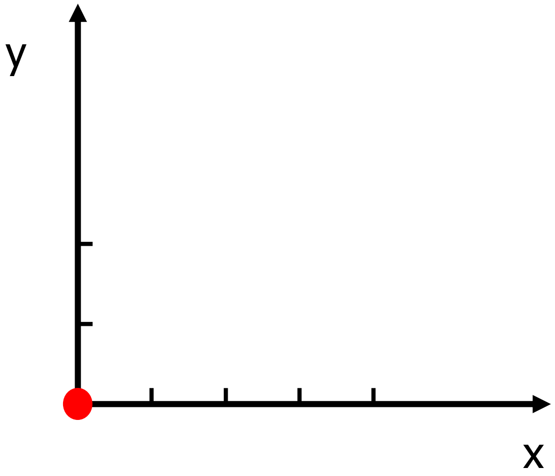
3<x, y=0

$$(\ell, 1 \leq x \leq 4, 1 \leq y \leq 3) \Rightarrow_a (\ell', 3 < x, y = 0)$$

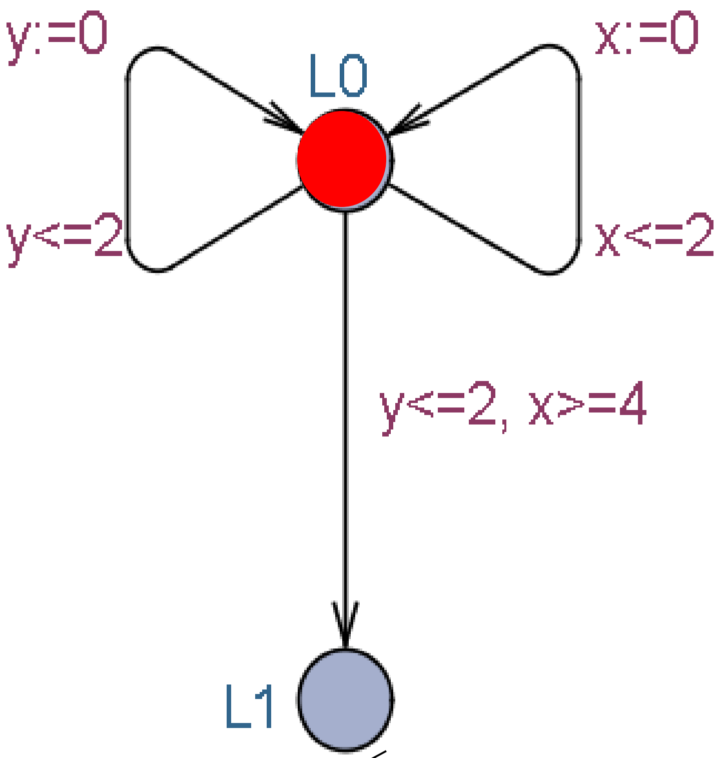
Symbolic Exploration



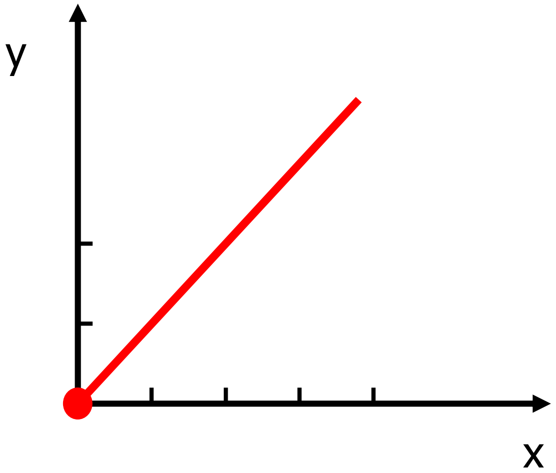
Reachable?



Symbolic Exploration

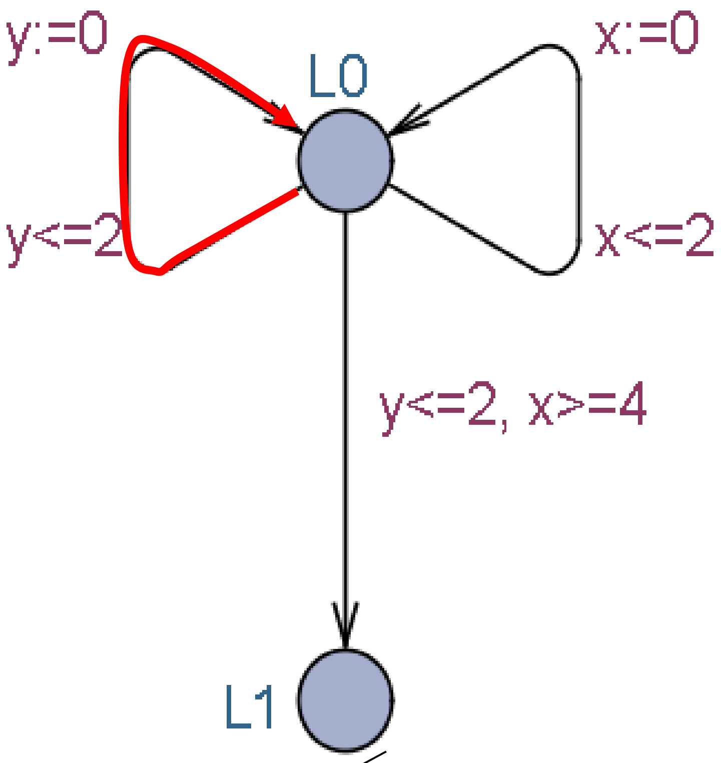


Reachable?

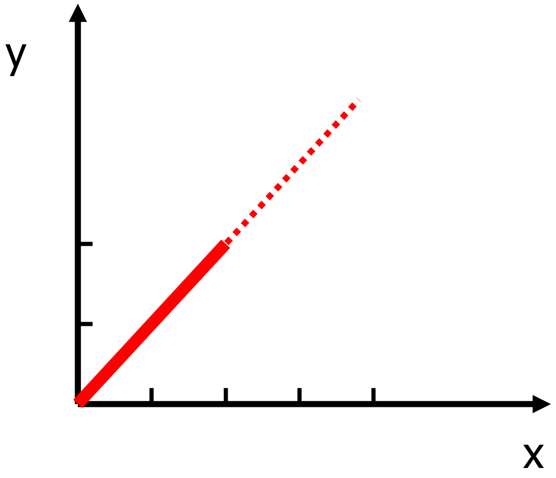


Delay

Symbolic Exploration

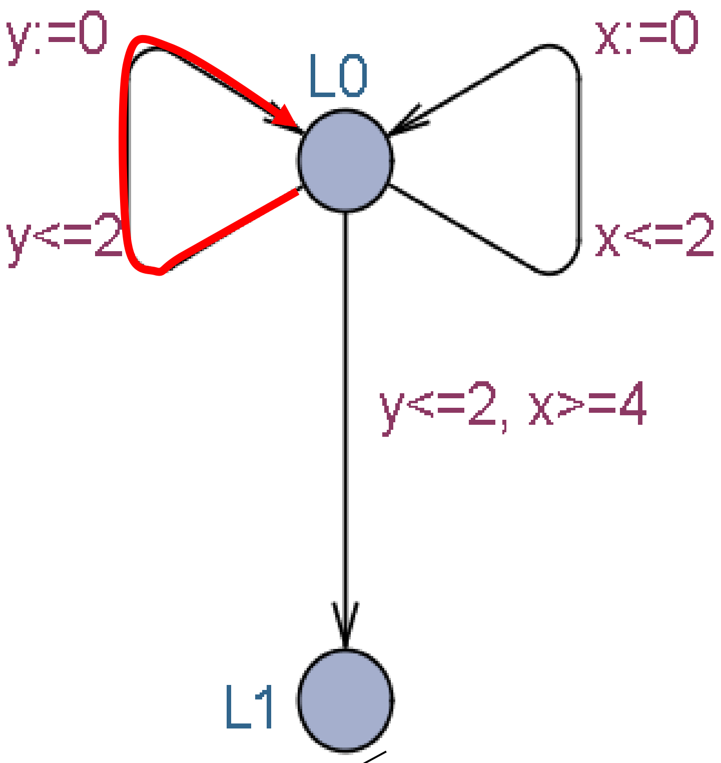


Reachable?

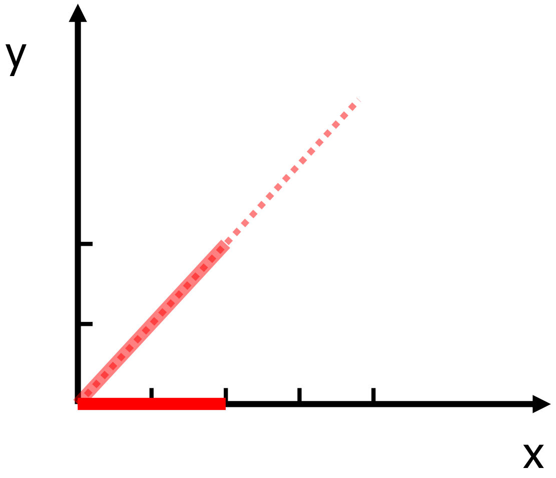


Left

Symbolic Exploration

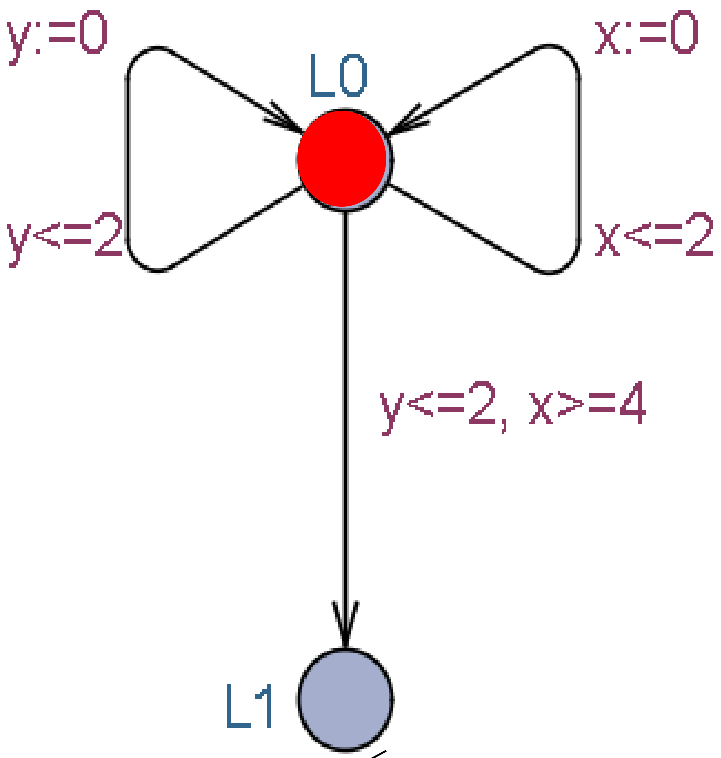


Reachable?

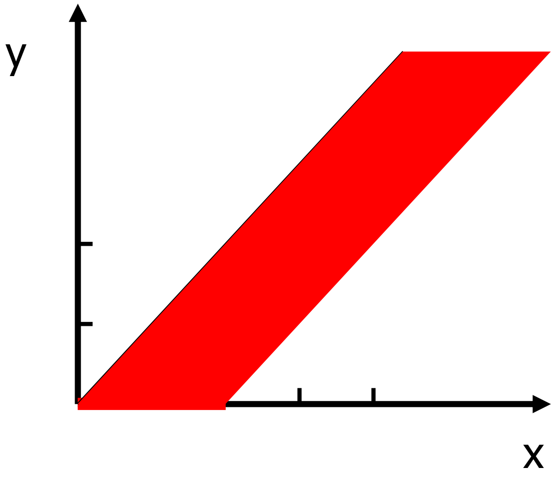


Left

Symbolic Exploration

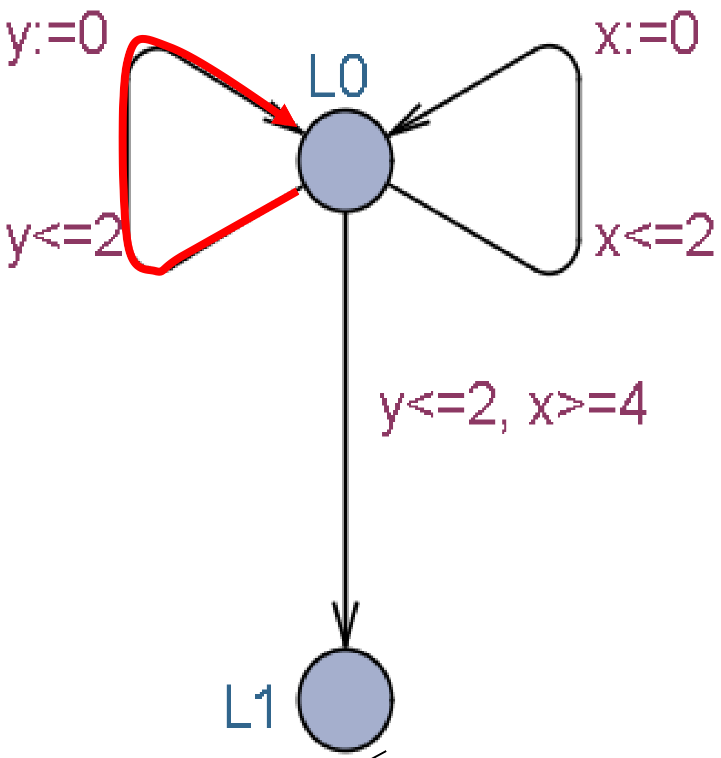


Reachable?

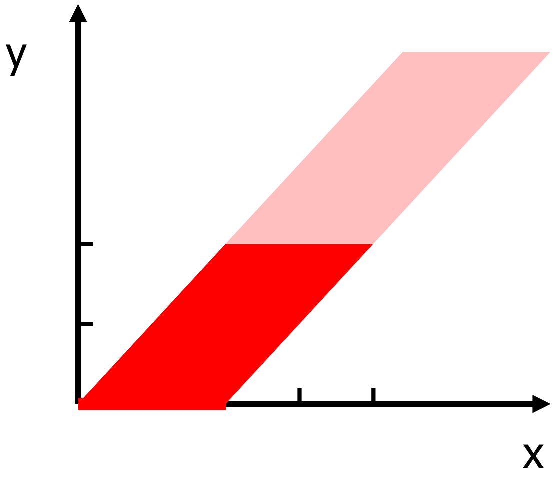


Delay

Symbolic Exploration

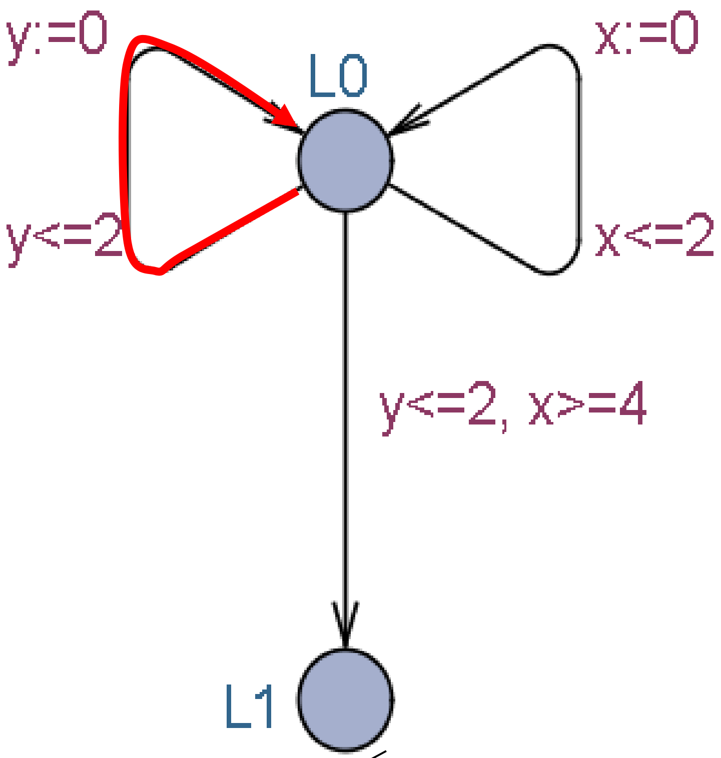


Reachable?

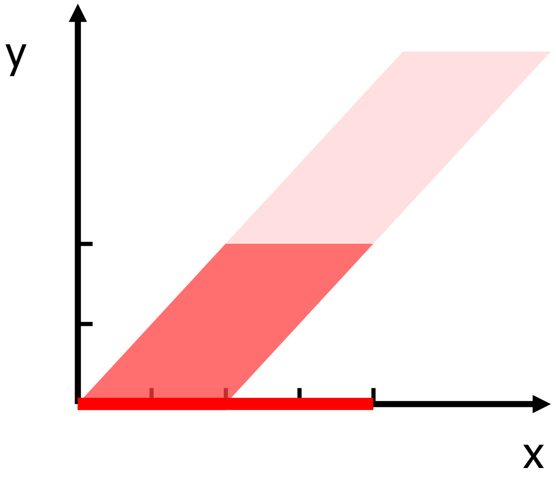


Left

Symbolic Exploration

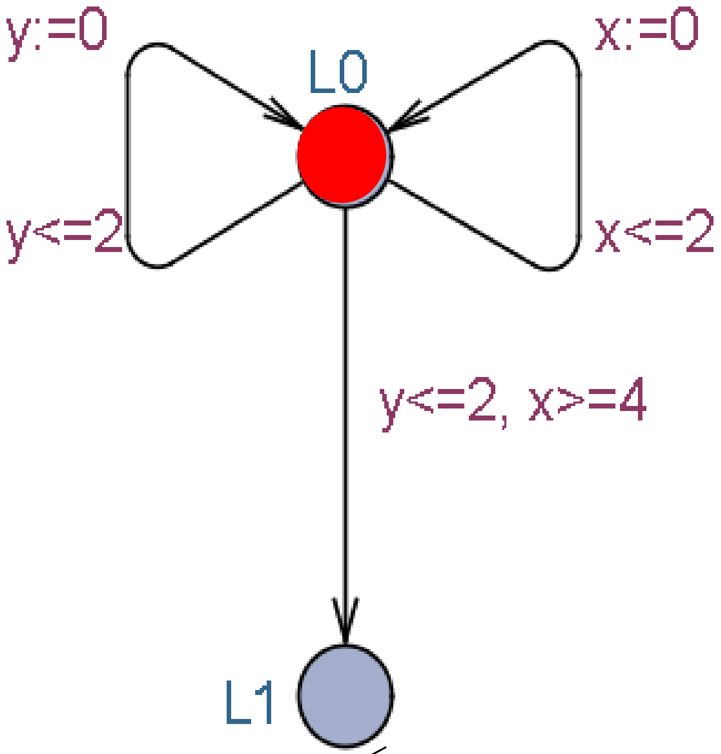


Reachable?

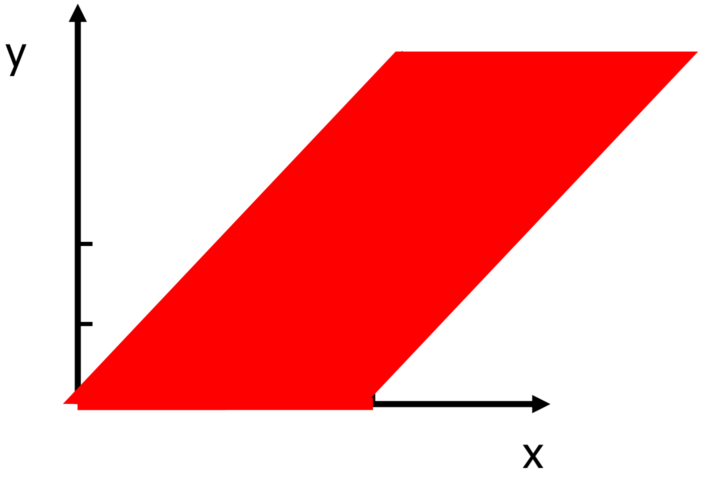


Left

Symbolic Exploration

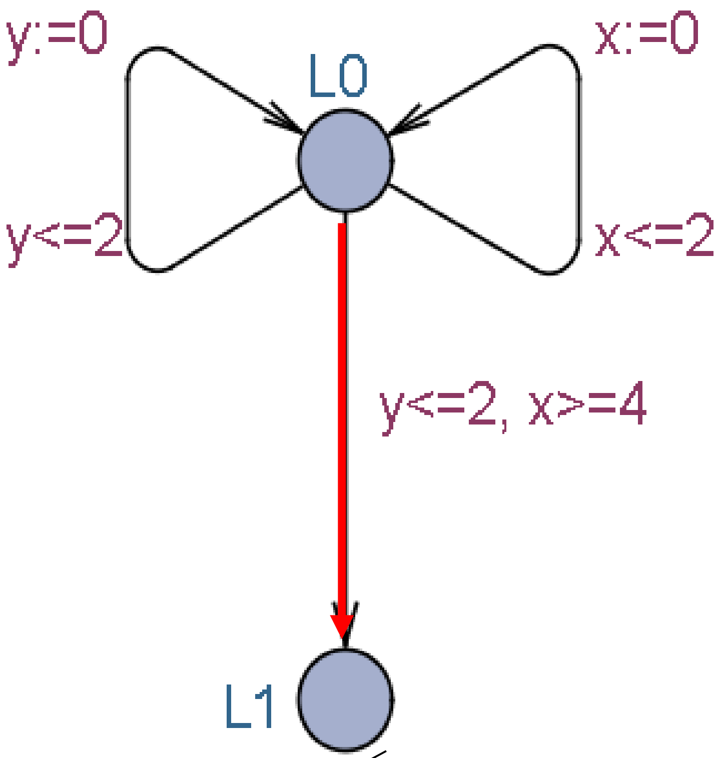


Reachable?

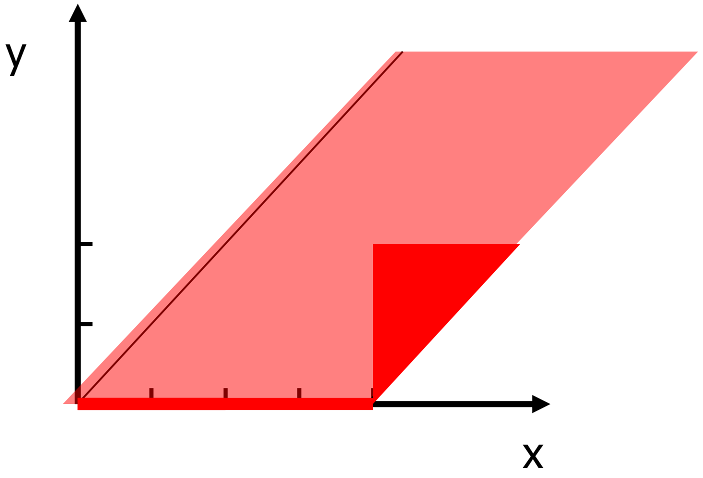


Delay

Symbolic Exploration



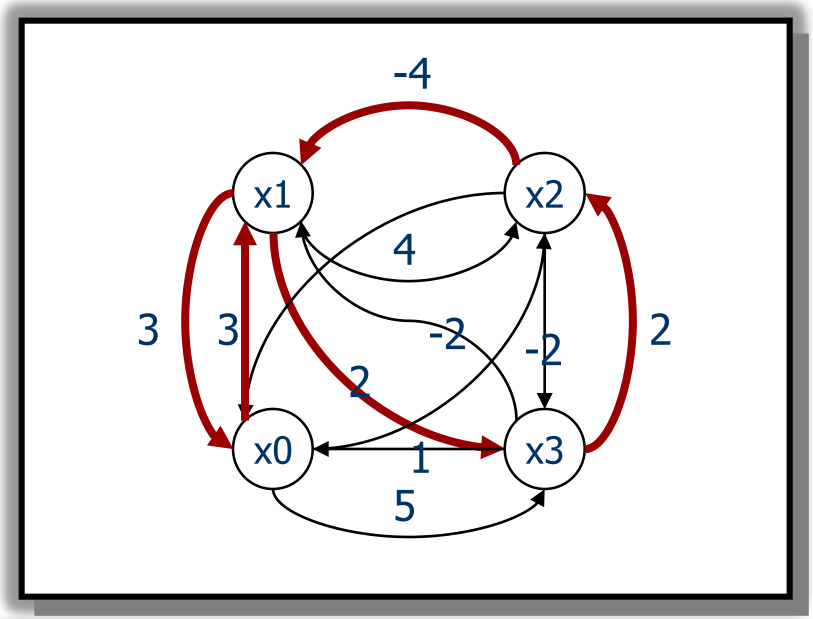
Reachable?



Down

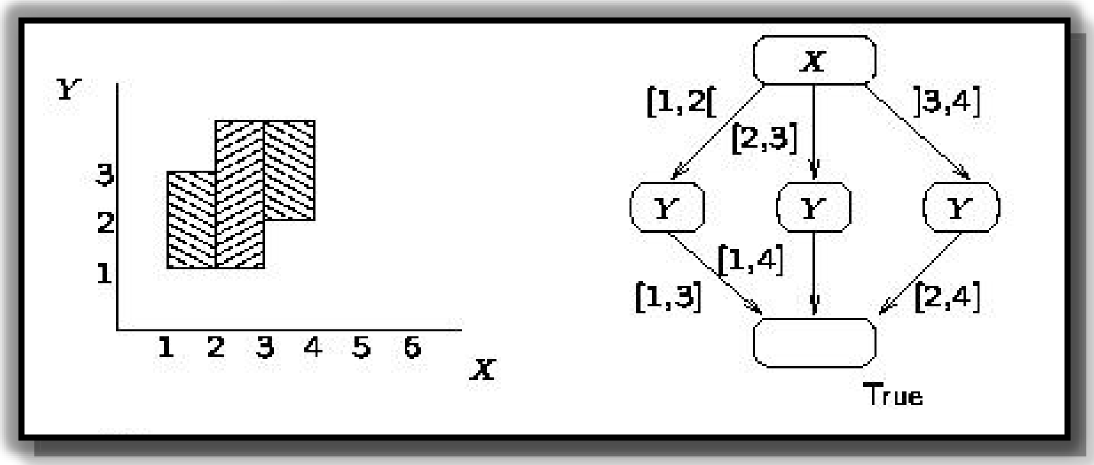
Datastructures for Zones

- Difference Bounded Matrices (DBMs)



- Minimal Constraint Form
  [RTSS97]

- Clock Difference Diagrams
  [CAV99]



Inclusion Checking (DBMs)

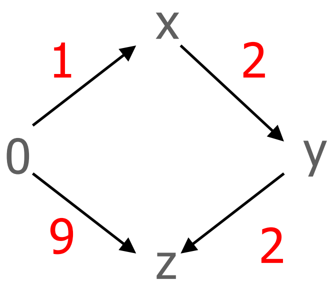
Bellman 1958, Dill 1989

Inclusion

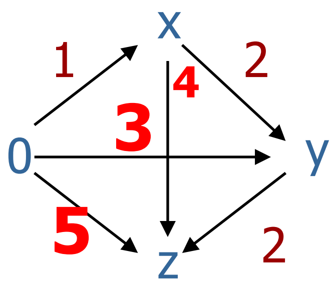
D1

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

Graph



Shortest Path Closure

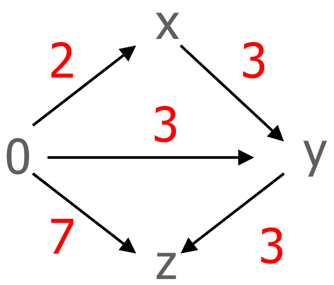


? ⊆ ?

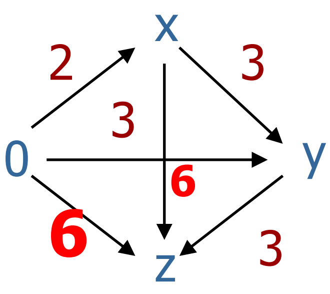
D2

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

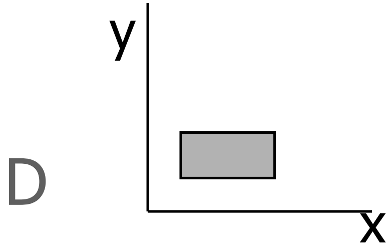
Graph



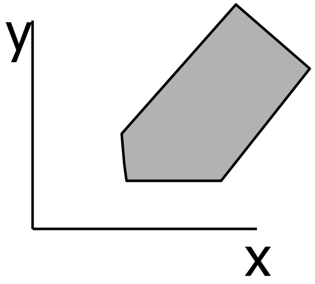
Shortest Path Closure



Future (DBMs)

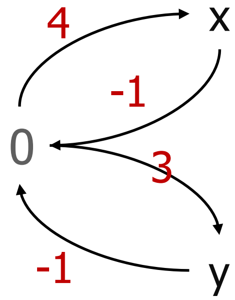


1<= x <=4
1<= y <=3

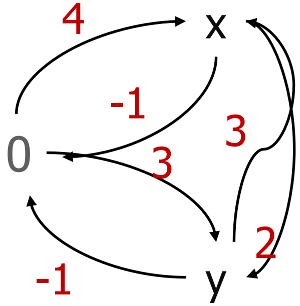


Future D

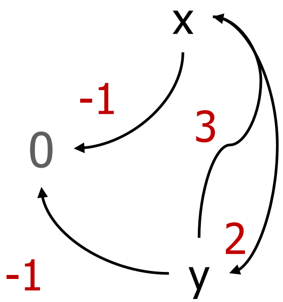
1<=x, 1<=y
-2<=x-y<=3



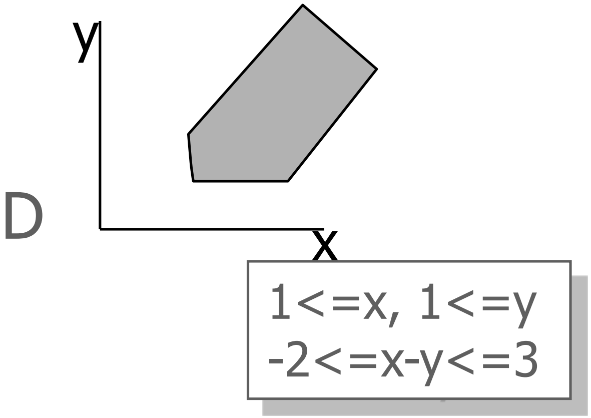
Shortest
Path
Closure



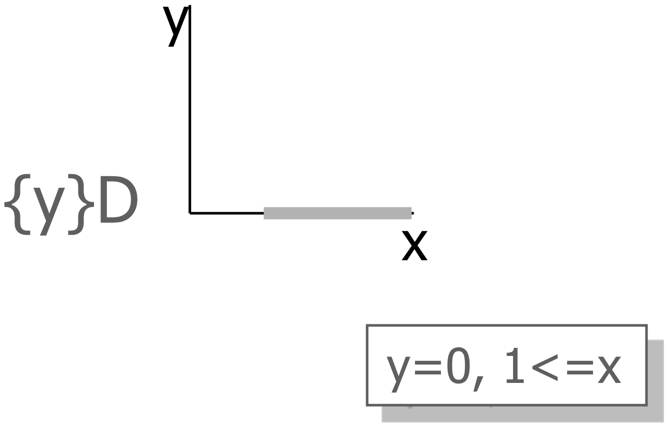
Remove
upper
bounds
on clocks



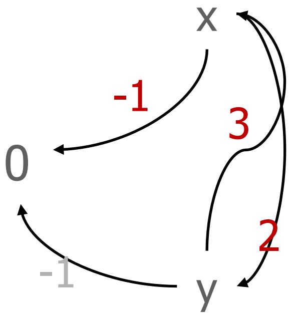
Reset (DBMs)



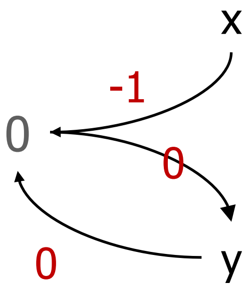
1<=x, 1<=y
-2<=x-y<=3



y=0, 1<=x

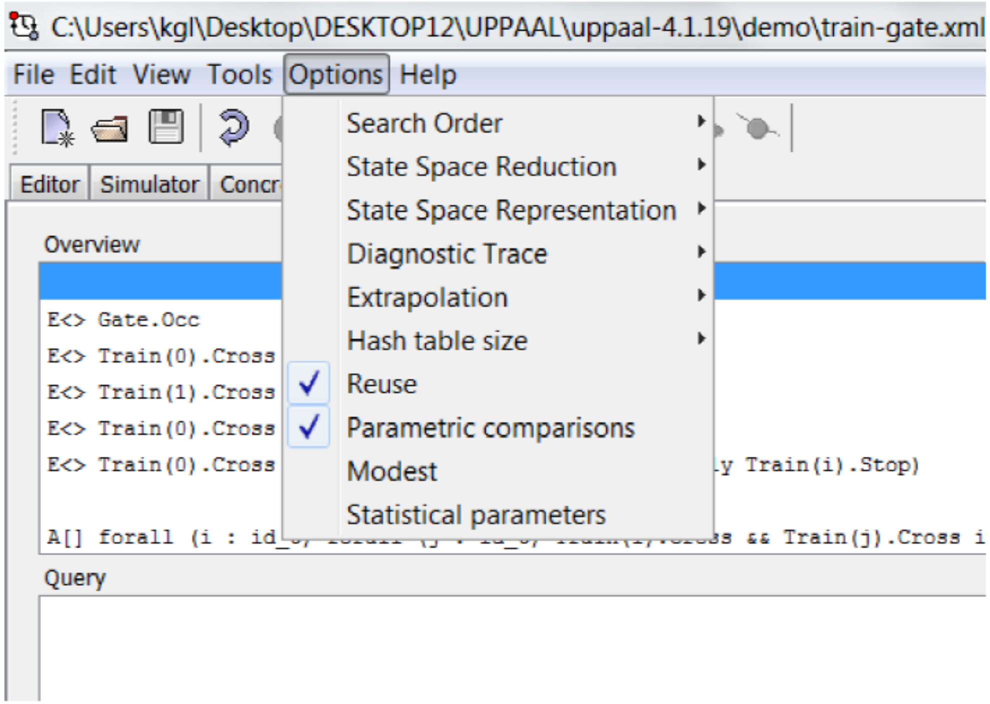


Remove all bounds involving y and set y to 0



Verification Options

Verification Options



```
E<> Gate.Occ
E<> Train(0).Cross
E<> Train(1).Cross
E<> Train(0).Cross
E<> Train(0).Cross                        y Train(i).Stop)

A[] forall (i : id_                         s && Train(j).Cross i
```

Search Order
- Depth First
- Breadth First
- Random Depth First

State Space Reduction
- None
- Conservative
- Aggressive
- Extreme

State Space Representation
- DBM
- Compact Form
- Under Approximation
- Over Approximation

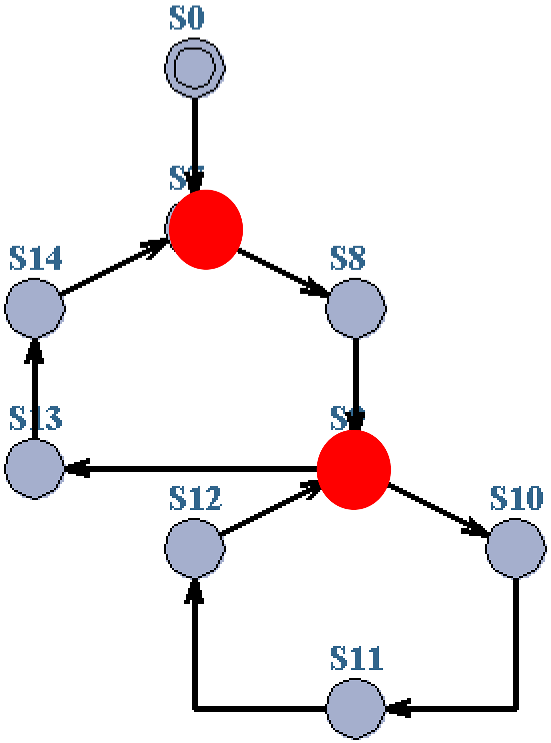
Diagnostic Trace
- Some
- Shortest
- Fastest

Extrapolation

Hash Table size

Reuse

State Space Reduction



Cycles:

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

To Store or Not To Store

Behrmann, Larsen, Pelanek 2003

117 $\text{states}_{\text{total}}$

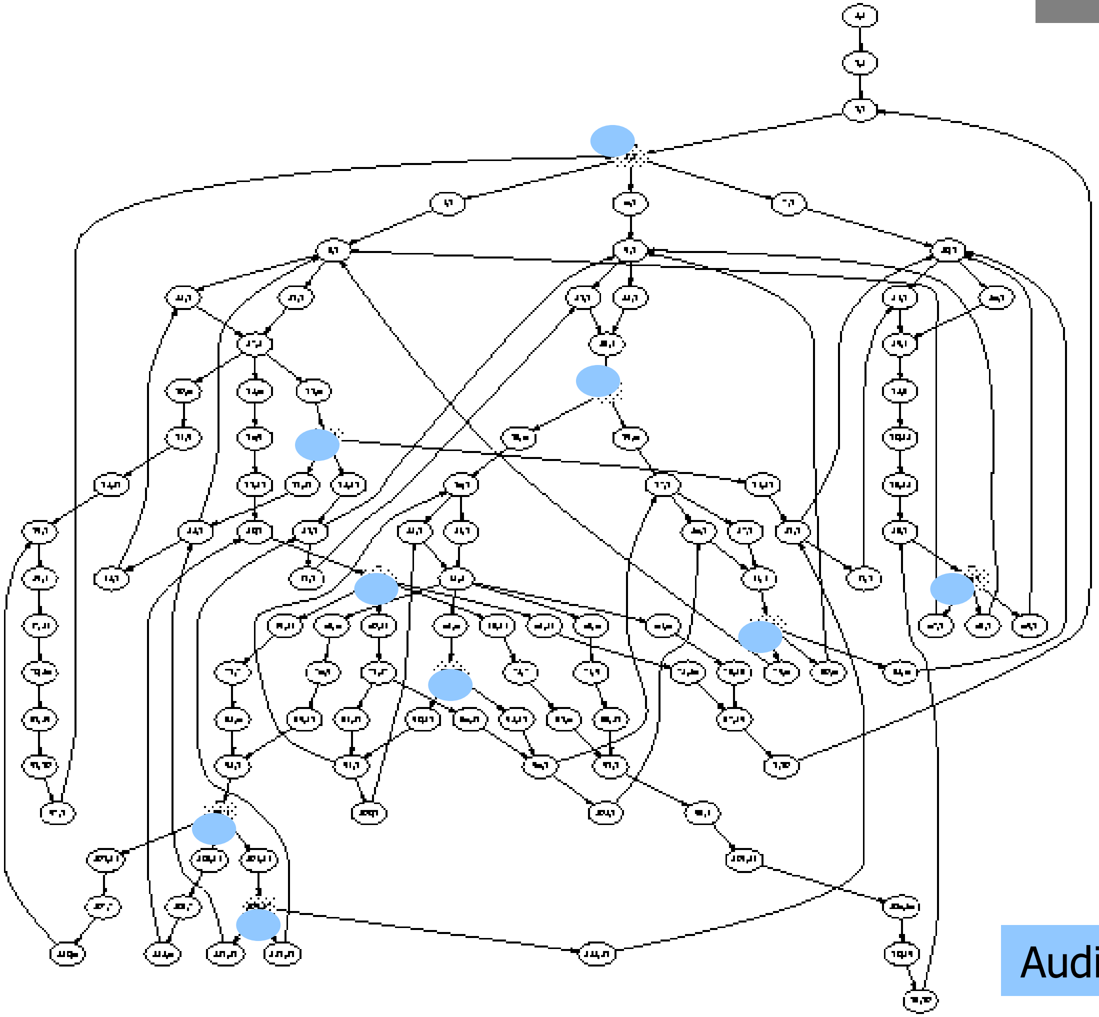
!

81 $\text{states}_{\text{entrypoint}}$

!

9 states

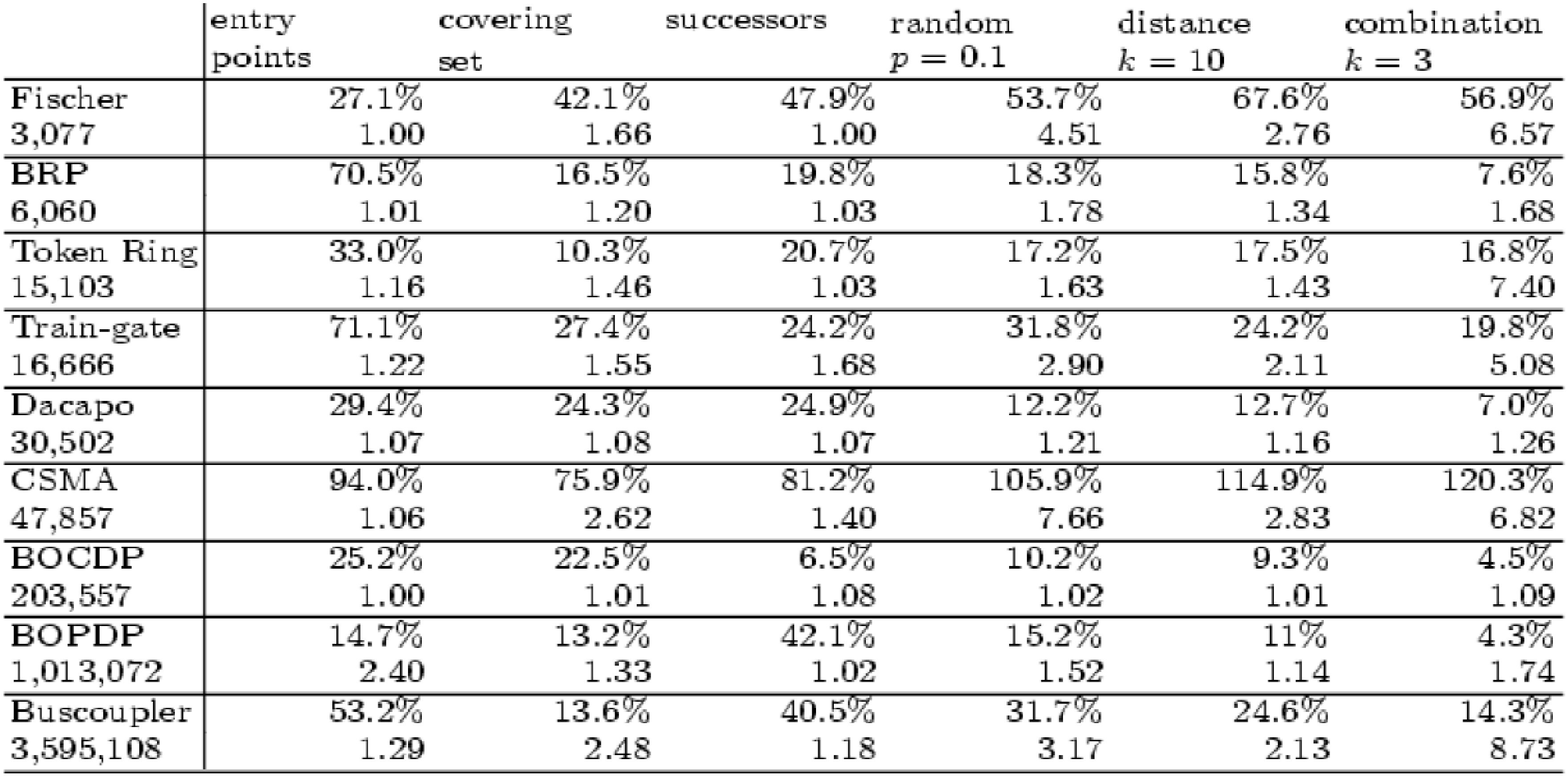
Time OH
less than 10%



Audio Protocol

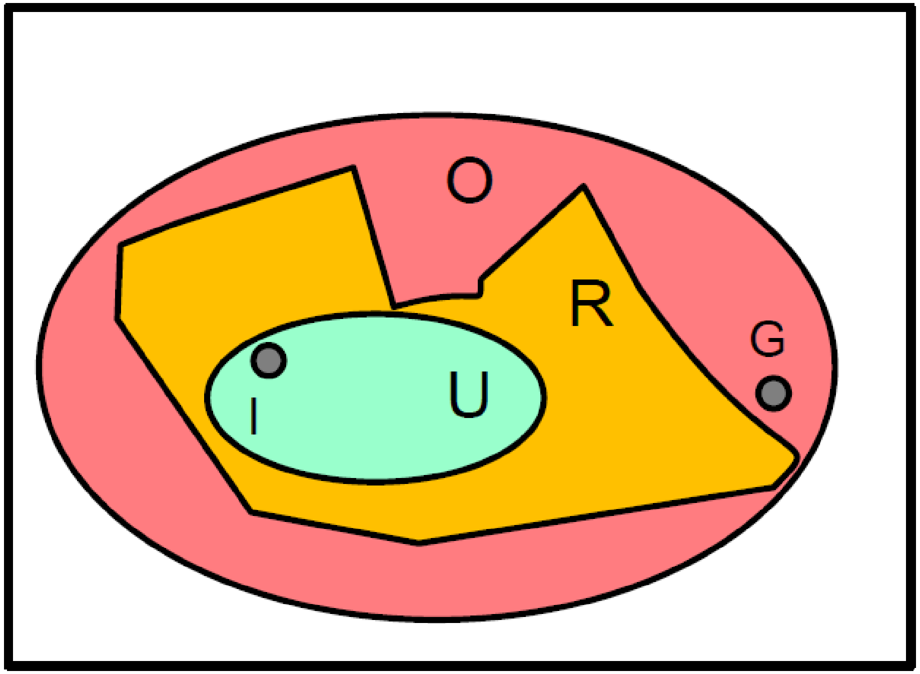
To Store or Not To Store

Behrmann, Larsen, Pelanek 2003



	entry	covering	successors	random	distance	combination
	points	set		$p = 0.1$	$k = 10$	$k = 3$
Fischer	27.1%	42.1%	47.9%	53.7%	67.6%	56.9%
3,077	1.00	1.66	1.00	4.51	2.76	6.57
BRP	70.5%	16.5%	19.8%	18.3%	15.8%	7.6%
6,060	1.01	1.20	1.03	1.78	1.34	1.68
Token Ring	33.0%	10.3%	20.7%	17.2%	17.5%	16.8%
15,103	1.16	1.46	1.03	1.63	1.43	7.40
Train-gate	71.1%	27.4%	24.2%	31.8%	24.2%	19.8%
16,666	1.22	1.55	1.68	2.90	2.11	5.08
Dacapo	29.4%	24.3%	24.9%	12.2%	12.7%	7.0%
30,502	1.07	1.08	1.07	1.21	1.16	1.26
CSMA	94.0%	75.9%	81.2%	105.9%	114.9%	120.3%
47,857	1.06	2.62	1.40	7.66	2.83	6.82
BOCDP	25.2%	22.5%	6.5%	10.2%	9.3%	4.5%
203,557	1.00	1.01	1.08	1.02	1.01	1.09
BOPDP	14.7%	13.2%	42.1%	15.2%	11%	4.3%
1,013,072	2.40	1.33	1.02	1.52	1.14	1.74
Buscoupler	53.2%	13.6%	40.5%	31.7%	24.6%	14.3%
3,595,108	1.29	2.48	1.18	3.17	2.13	8.73

Over/Under Approximation



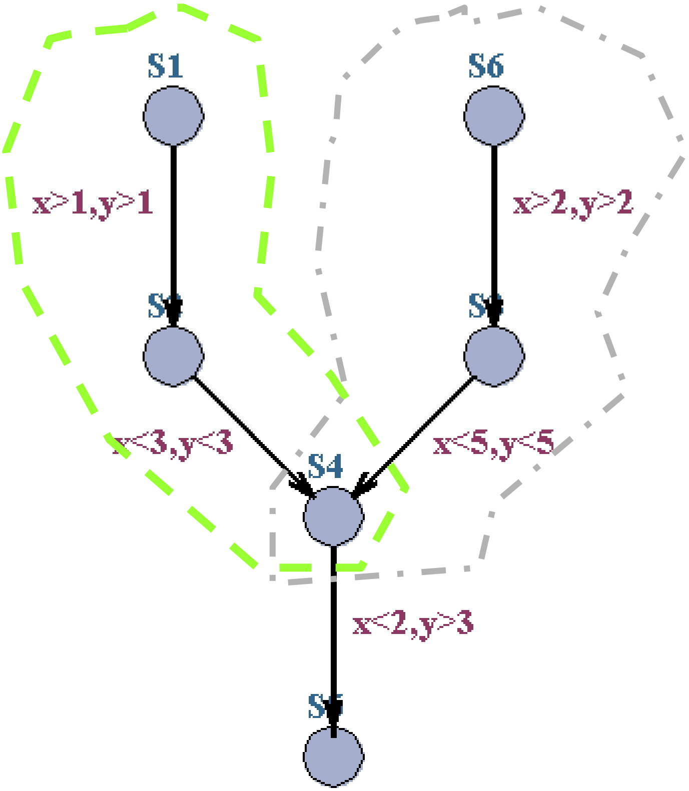
Declared State Space

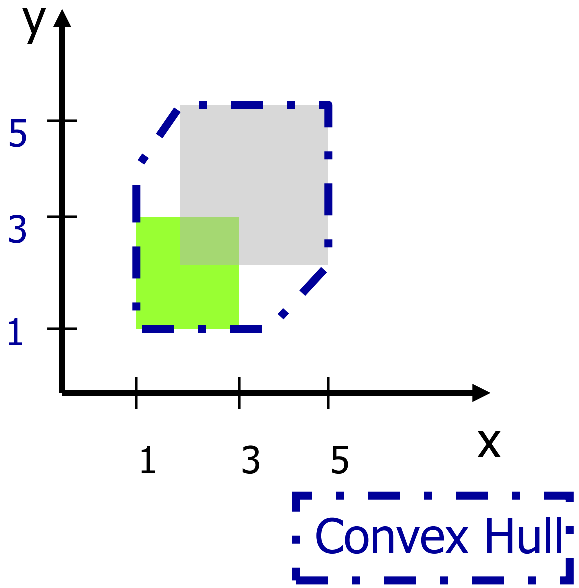
Question:

$G \in R$?

Over−approximation

Convex Hull

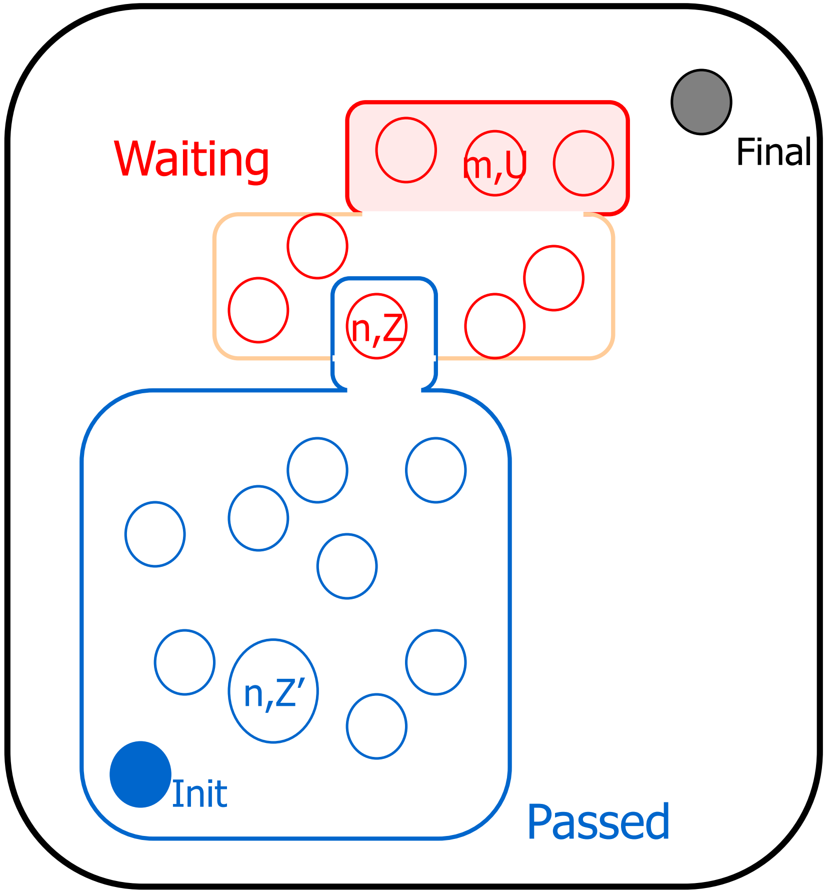




TACAS04: An EXACT method performing as well as Convex Hull has been developed based on abstractions taking max constants into account distinguishing between clocks, locations and · & ¸

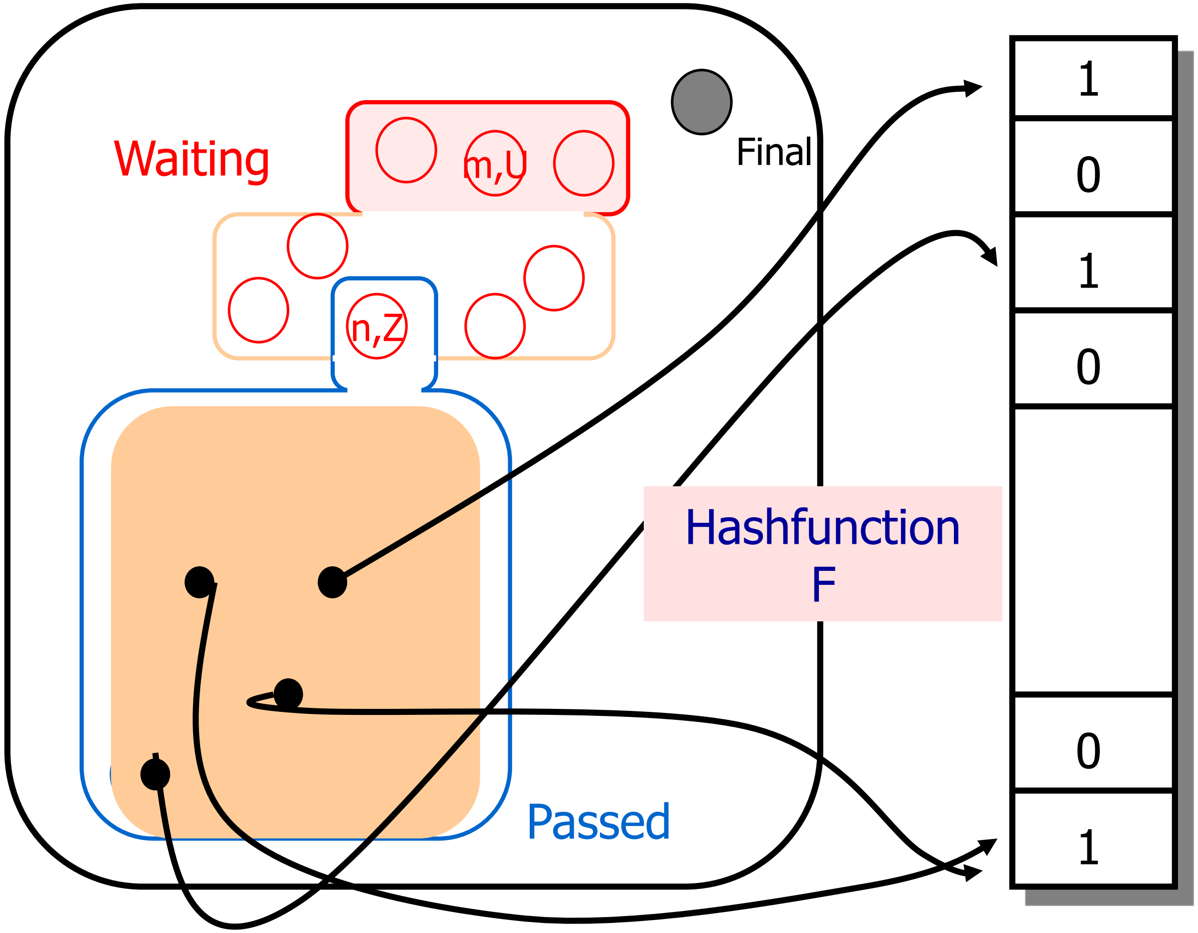
Under–approximation

Bitstate Hashing



Under–approximation

Bitstate Hashing



Passed=
Bitarray

UPPAAL
4 - 512 Mbits

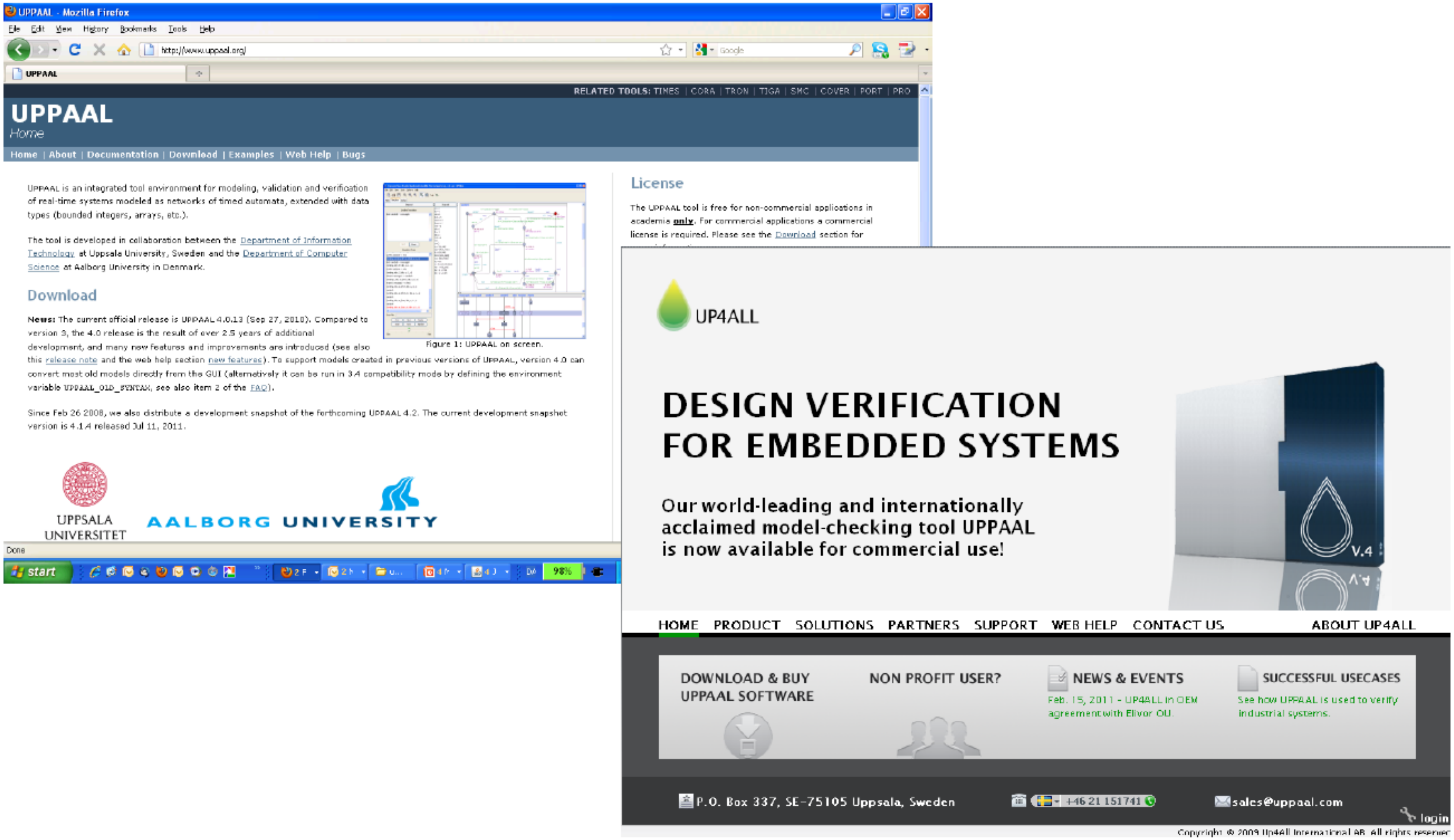
Mini Project 2 – Gossing Persons

Six persons all have a gossip of their own.

They call each other over the phone. Whenever two persons talk they exchange all gossips they know.

How many call are needed before all persons knows every gossip.

www.uppaal.{org,com}



UPPAAL - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.uppaal.org/
Google
UPPAAL
RELATED TOOLS: TIMES | CORA | TRON | TIGA | SMC | COVER | PORT | PRO
Home | About | Documentation | Download | Examples | Web Help | Bugs
Done
start
98%

UPPAAL

Home

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

Figure 1: UPPAAL on screen.

Download

News: The current official release is UPPAAL 4.0.13 (Sep 27, 2010). Compared to version 3, the 4.0 release is the result of over 2.5 years of additional development, and many new features and improvements are introduced (see also this release note and the web help section new features). To support models created in previous versions of UPPAAL, version 4.0 can convert most old models directly from the GUI (alternatively it can be run in 3.4 compatibility mode by defining the environment variable UPPAAL_OLD_SYNTAX, see also item 2 of the FAQ).

Since Feb 26 2008, we also distribute a development snapshot of the forthcoming UPPAAL 4.2. The current development snapshot version is 4.1.4 released Jul 11, 2011.

License

The UPPAAL tool is free for non-commercial applications in academia **only**. For commercial applications a commercial license is required. Please see the Download section for

UP4ALL
V.4
HOME PRODUCT SOLUTIONS PARTNERS SUPPORT WEB HELP CONTACT US ABOUT UP4ALL
login

DESIGN VERIFICATION FOR EMBEDDED SYSTEMS

Our world-leading and internationally acclaimed model-checking tool UPPAAL is now available for commercial use!

DOWNLOAD & BUY UPPAAL SOFTWARE

NON PROFIT USER?

NEWS & EVENTS

Feb. 15, 2011 - UP4ALL in OEM agreement with Elivor OU.

SUCCESSFUL USECASES

See how UPPAAL is used to verify industrial systems.

P.O. Box 337, SE-75105 Uppsala, Sweden

☎ +46 21 151741

sales@uppaal.com

www.uppaal.{org,com}