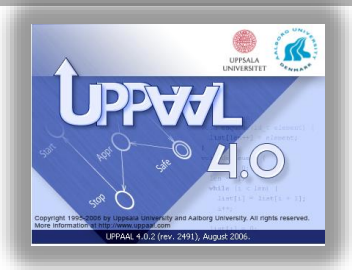
From Timed Automata to Stochastic Hybrid Games

Model Checking, Performance Analysis, Optimization, Synthesis, and Machine Learning



Kim G. Larsen

Aalborg University, DENMARK

CISS –
Center For Embedded Software Systems

Regional ICT Center (2002-)

- 3 research groups
  - Computer Science
  - Control Theory
  - Hardware
  - Wireless Communication

- 20 Employed
- 25 Associated
- 20 PhD Students
- 70 Industrial projects
- 10 Elite-students

- ARTIST Design
- ARTEMIS / ECSEL
- … …

From ES to CPS

New Foundation

Discrete Models
(Boolean correctness)
→
Quantitive Models
(time, resources, probabilistic, stochastic, continuous,..)
(Quantitative correctness)

Embedded Systems

Networked ES

IoT
Cloud Computin
Big Data

Cyber-Physical

Discrete

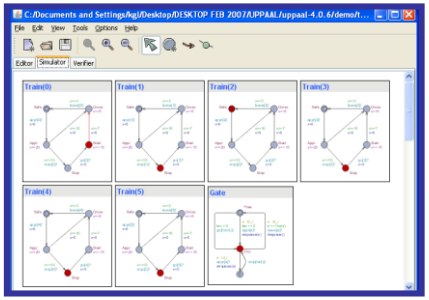
Real Time

Resources

Stochasticity

Hybrid

QUANTITATIVE Model Checking

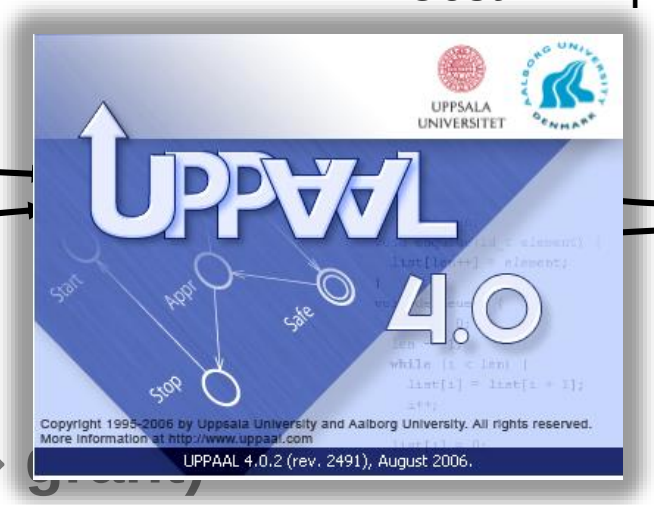


System Description

Time

Cost

Probability



No!
Debugging Information

Yes
Prototypes
Executable Code
Test sequences

Requirement

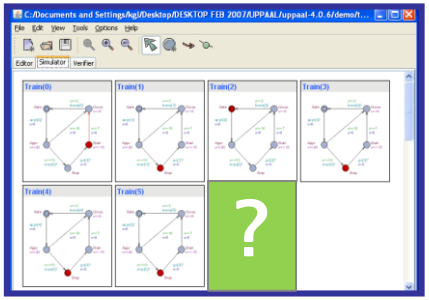
$A\Box(\ req \Rightarrow A\Diamond$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s}\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s,c<5\$}\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s\ ,\ p>0.90}\ grant)$

Synthesis

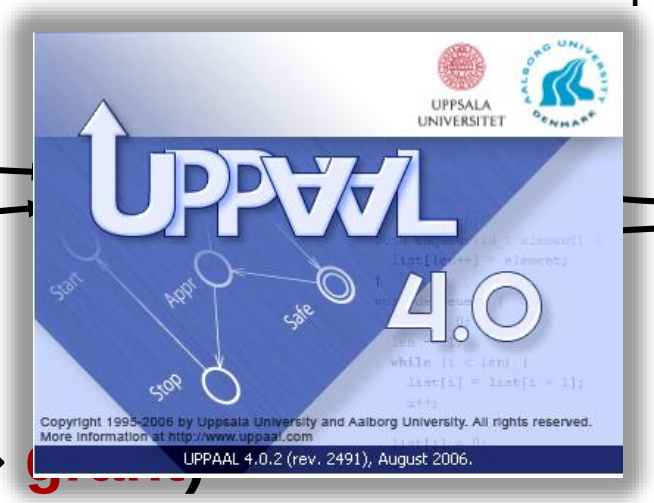


System Description

Time

Cost

Probability



No!
Debugging Information

Yes
Control Strategy

Requirement

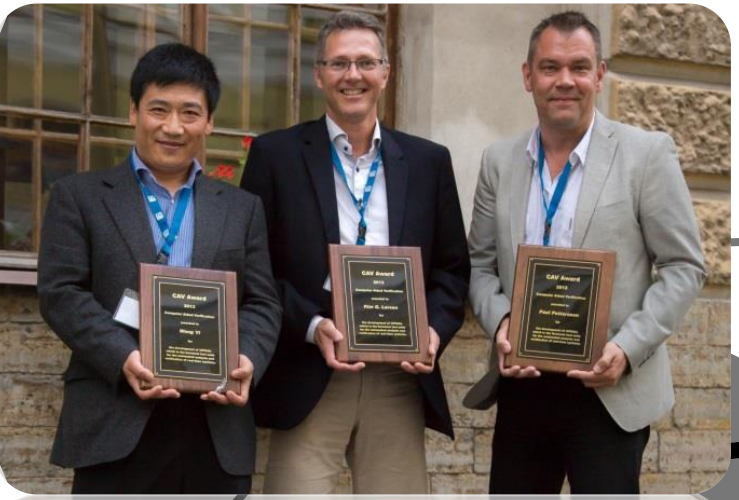
$A\Box(\ req \Rightarrow A\Diamond\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s}\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s,c<5\$}\ grant)$

$A\Box(\ req \Rightarrow A\Diamond_{t<30s\ ,\ p>0.90}\ grant)$

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

2013

2016

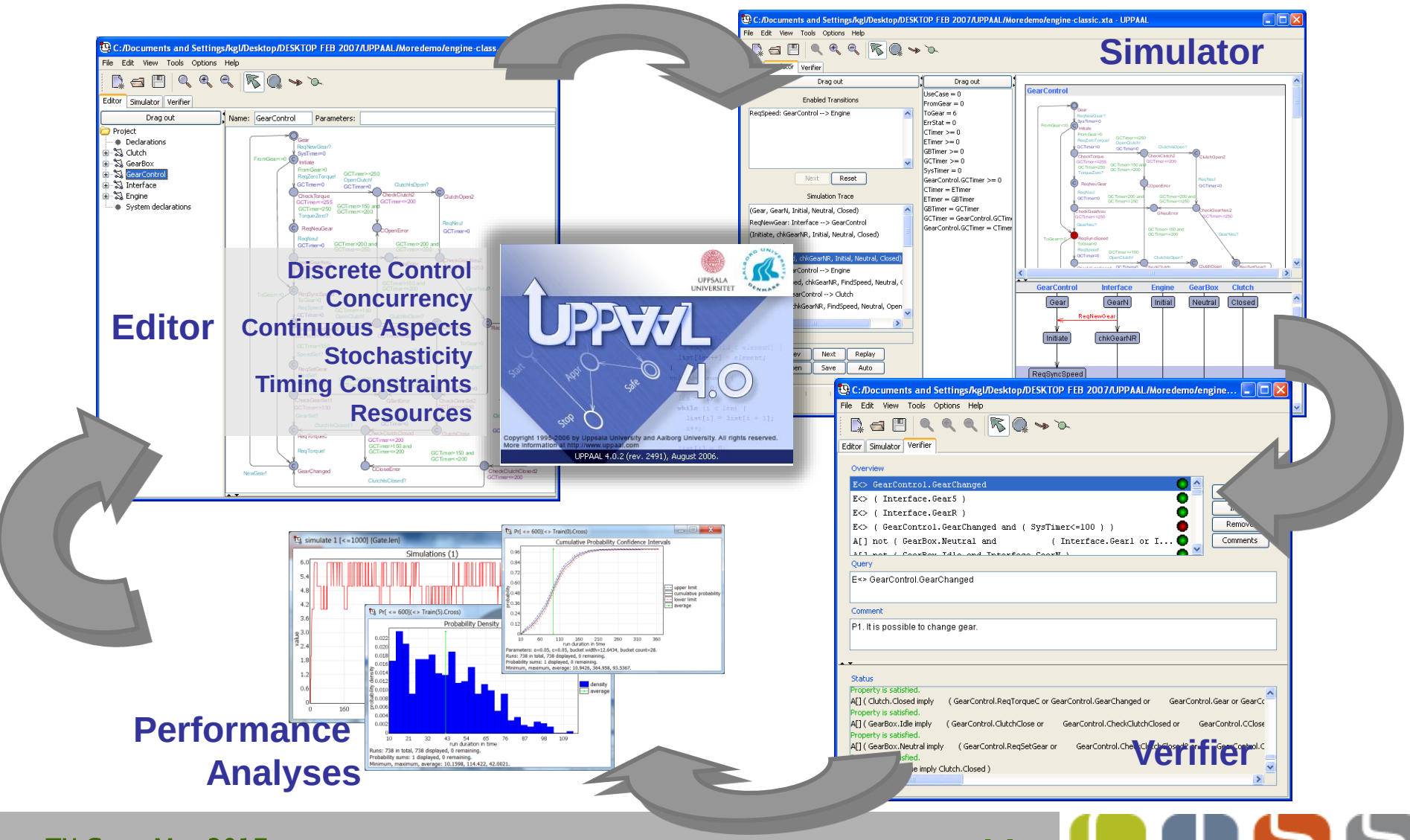
EPSILON
TCCS
Timed Refinements
Timed Mu-Calculus
Regions
Prolog<

UP4ALL

Grundfos Prize

CAV Award

UPPAAL Model Checker



UPPAAL (1995–)

@UPPsala

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

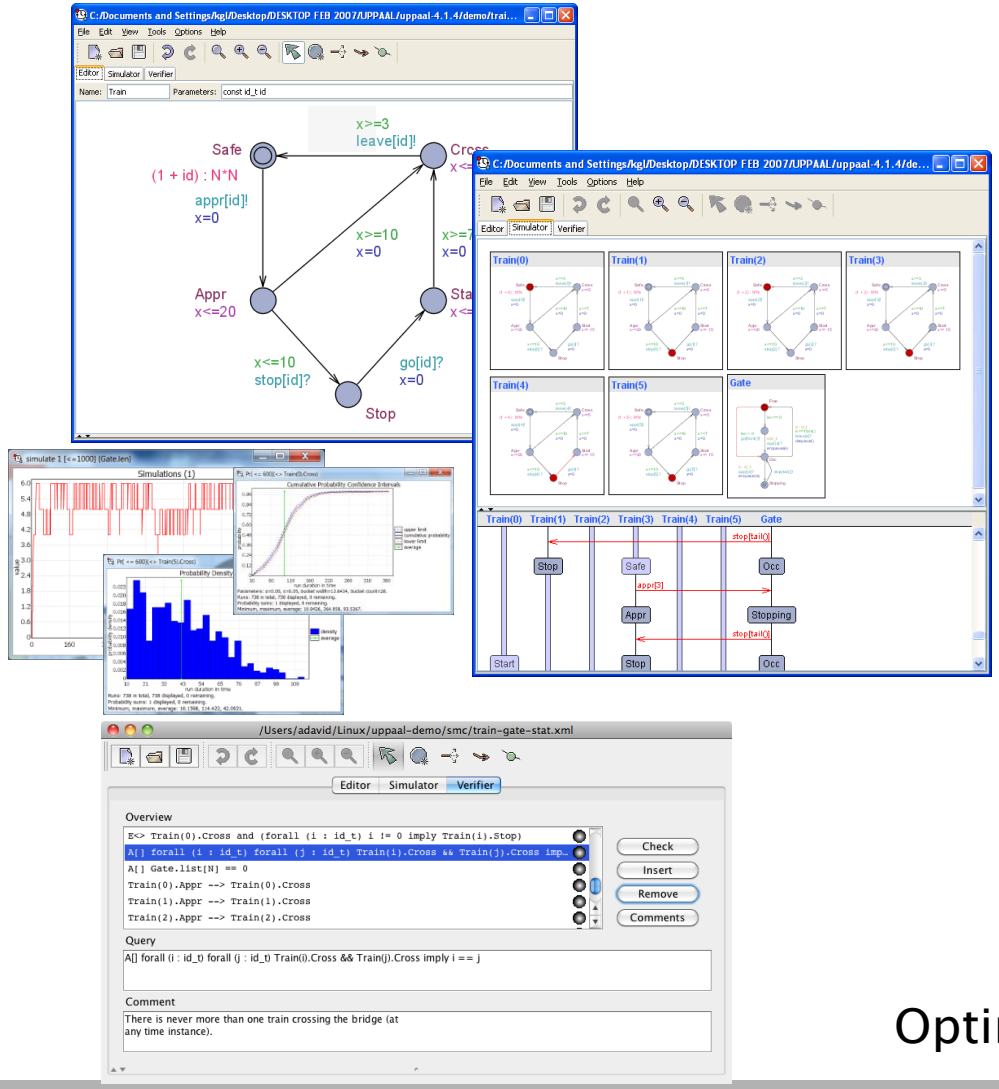
@AALborg

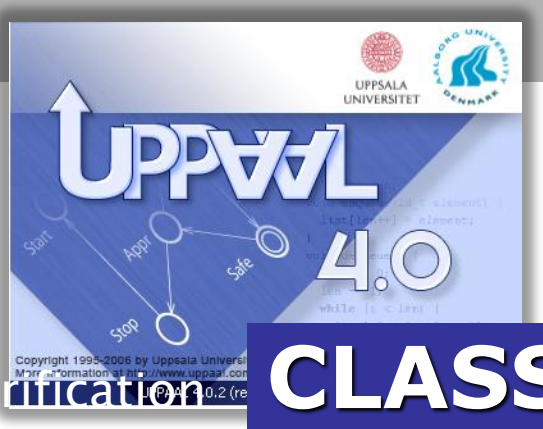
- Kim G Larsen
- Alexandre David
- Marius Mikucionis
- Gerd Behrman
- Arne Skou
- Brian Nielsen
- Jacob I. Rasmussen
- Thomas Chatain

@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...

UPPAAL Tool Suit

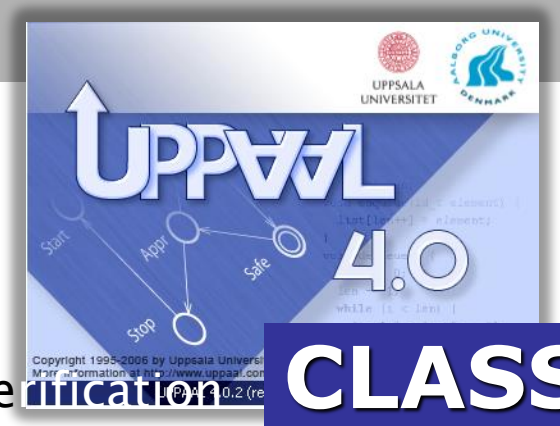




Verification	CLASSIC	1995
Optimization	CORA	2001
Synthesis	TIGA	2005
Component	ECDAR	2010
Testing	TRON	2004
Performance Analysis	SMC	2011
Optimal Synthesis	STRATEGO	2014

Topics

- **Timed Automata**
  - Decidability (regions)
  - Symbolic Verification (zones)
- **Priced Timed Automata**
  - Decidability (priced regions)
  - Symbolic Verification (priced zones)
- **Stochastic Timed Automata**
  - Stochastic Semantics
  - Statistical Model Checking
  - Stochastic Hybrid Automata
- **Timed Games & Interfaces**
  - Strategies, Symbolic Synthesis
  - Refinement
- **Stochastic Priced Timed Games**
  - Strategies
  - Symbolic Synthesis (zones)
  - Stochastic Strategies
  - Reinforcement Learning



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

people.cs.aau.dk/~kgl/GRAZ17/

people.cs.aau.dk/~kgl/GRAZ17/

From Timed Automata to Stochastic Hybrid Games

Model Checking, Optimization, Performance Evaluation, Synthesis and Machine Learning

using UPPAAL

Technische Universität Graz, May 2017

Kim Guldstrand Larsen

CISS, Aalborg University, DENMARK

Slides

1. Timed Automata, UPPAAL
2. Priced Timed Automata, Energy Automata, UPPAAL CORA
3. Stochastic Timed Automata, UPPAAL SMC
4. Stochastic Timed Games, UPPAAL TIGA and UPPAAL STRATEGO

Primary Reading

Frits Vaandrager: A first introduction to UPPAAL

Alexandre David, Kim G Larsen: More features in UPPAAL

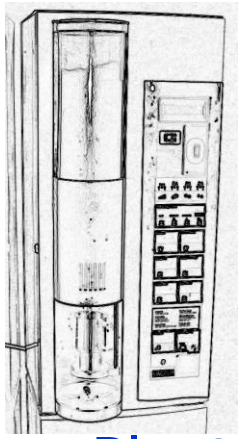
Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, Joel Ouaknine, James Worrel: Model Checking Real Time Systems. To appear in Handbook of Model Checking.

Uli Fahrenberg, Kim G Larsen, Axel Legay: Model-based Verification, Optimization, Synthesis and Performance Analysis of Real-Time Systems.

Johan Bengtsson and Wang Yi: Timed Automata: Semantics, Algorithms and Tools. Concurrency and Petri Nets 2004, LNCS 3098

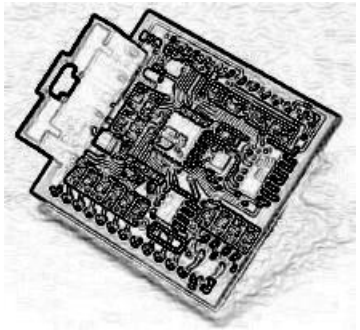
Timed Automata

Real Time Systems



sensors →

← actuators



Plant

Continuous

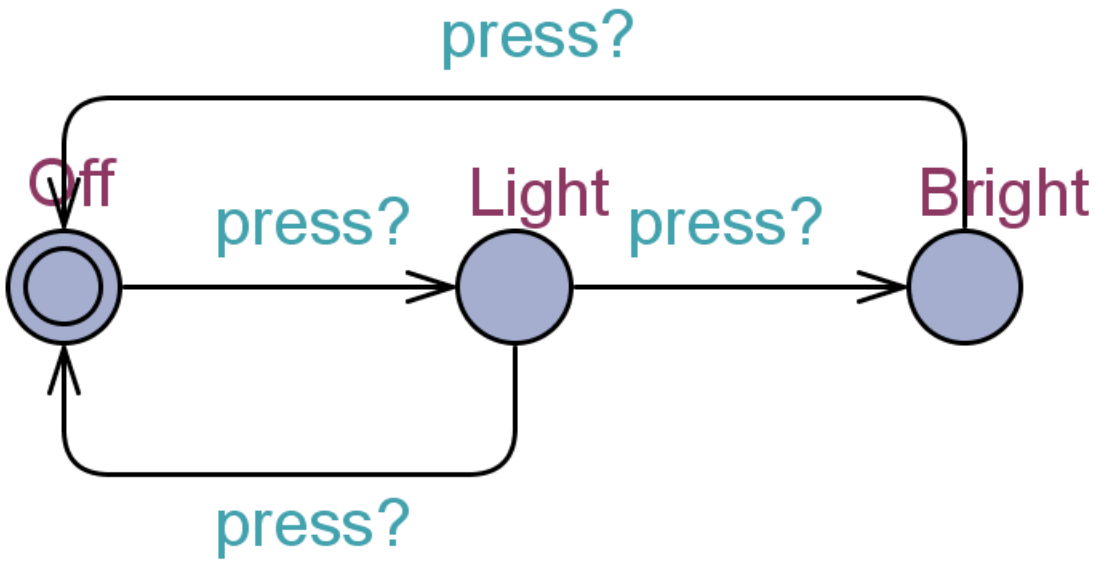
Controller Program

Discrete

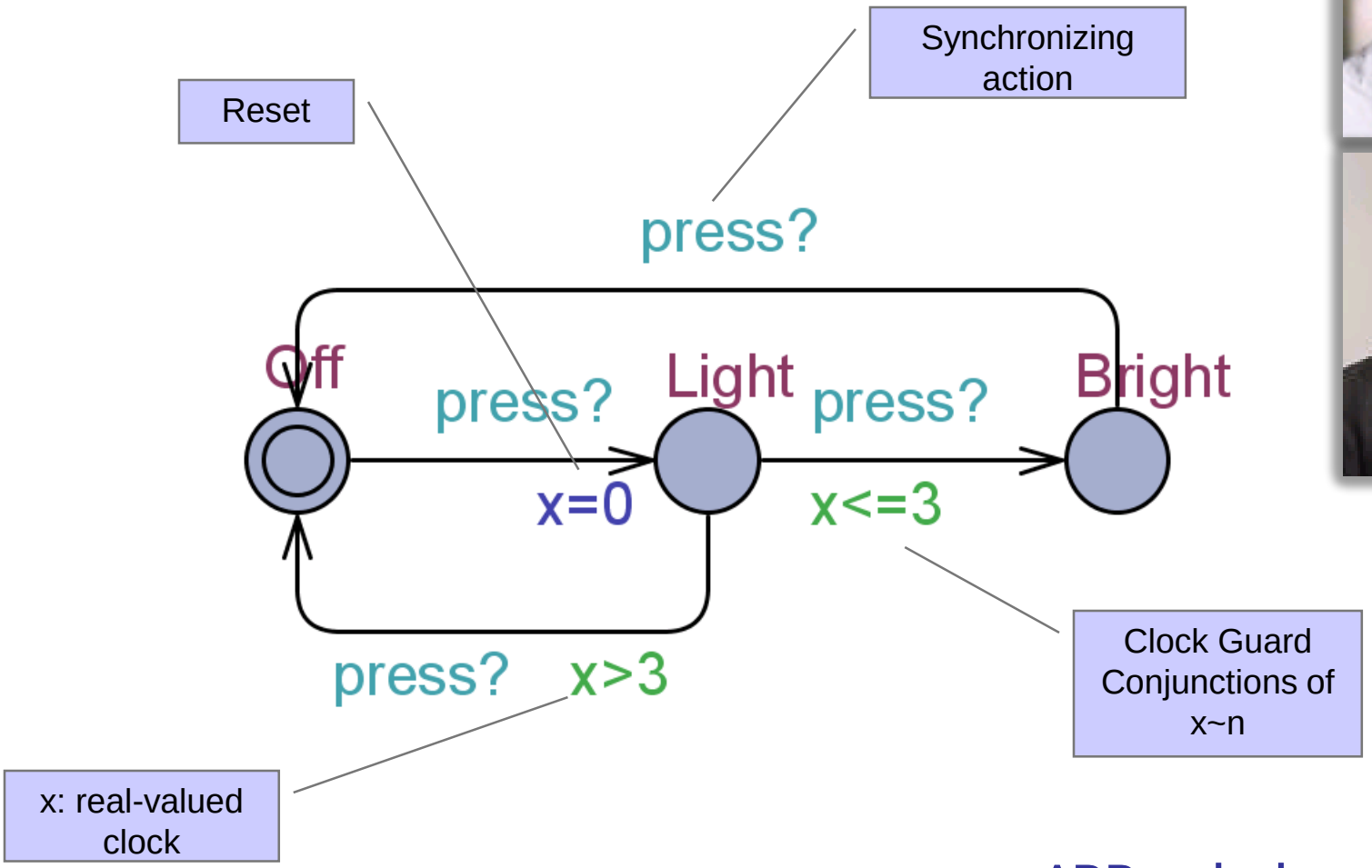
Eg.: Realtime Protocols
Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

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

A Dumb Light Controller

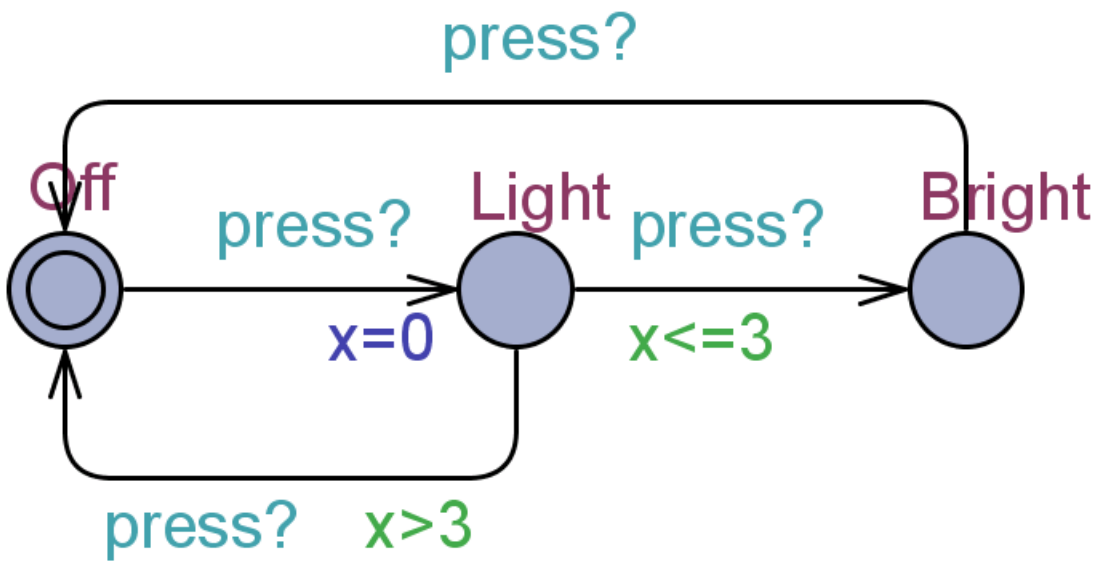


Timed Automata [Alur & Dill'89]



ADD a clock x

A Timed Automata (Semantics)

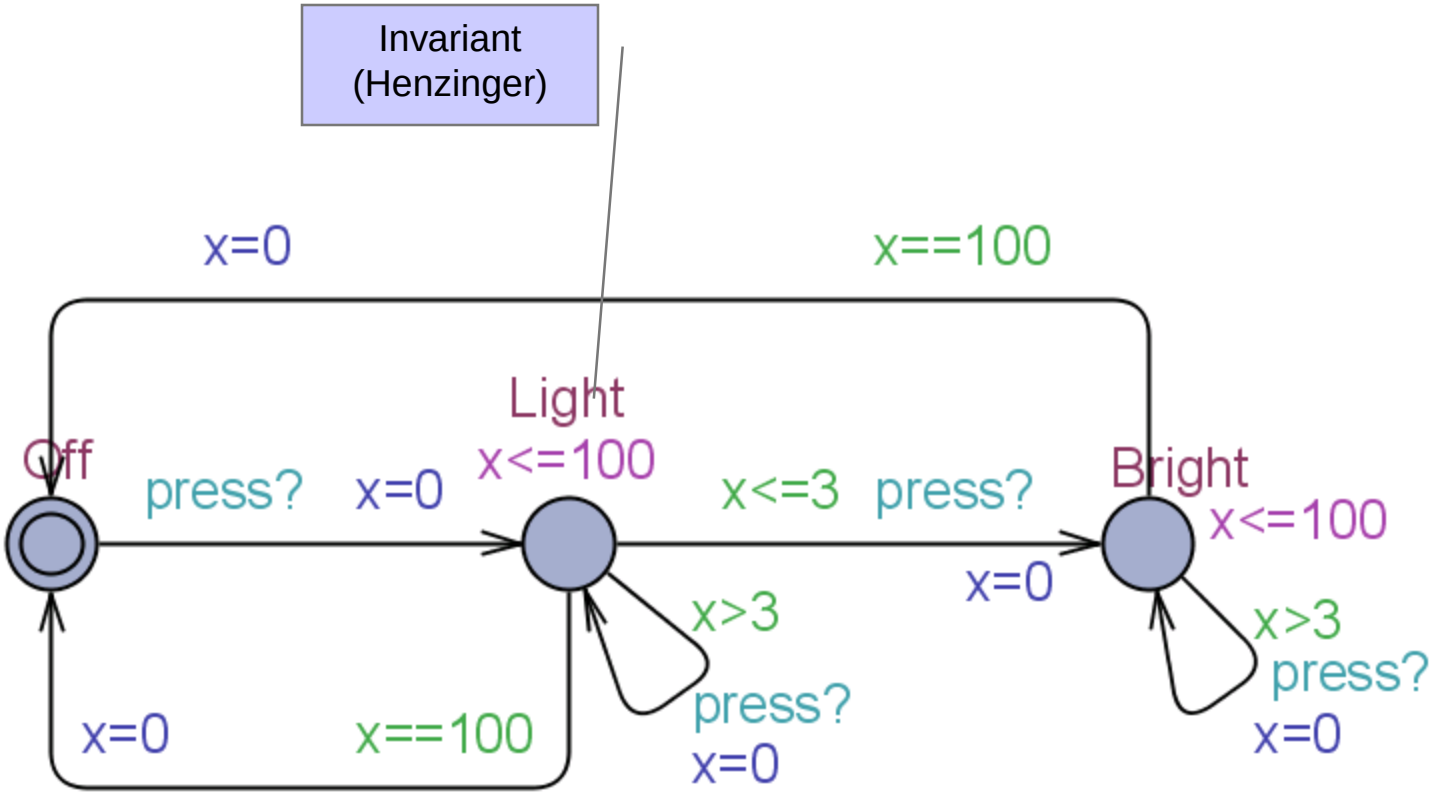


States:
(location , x=v) where v∈**R**

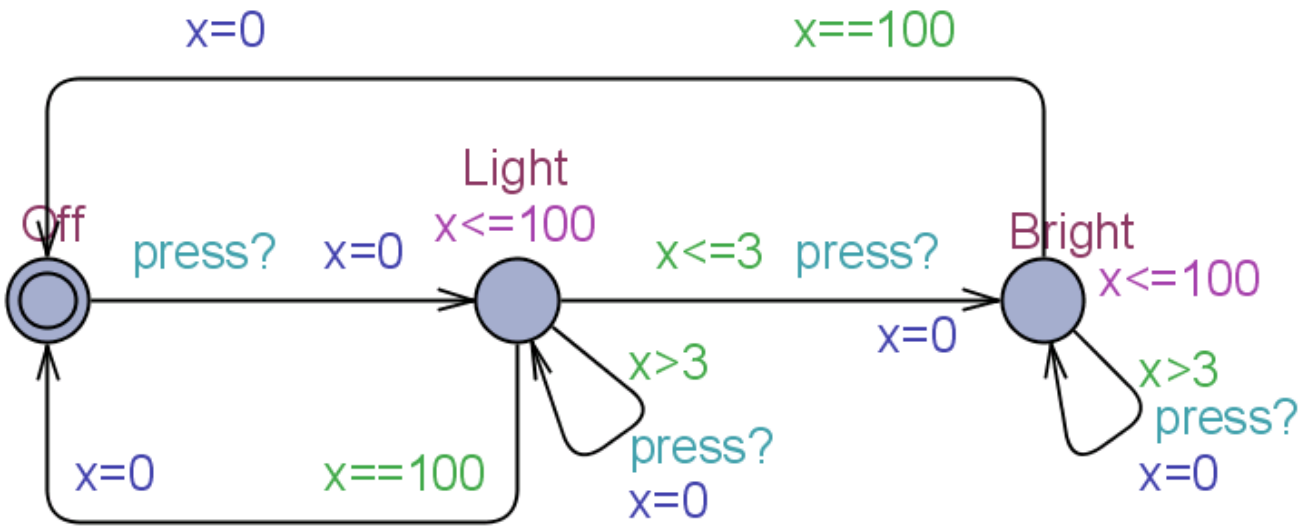
Transitions:

	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 2.51	→ (Light , x=2.51)
press?	→ (Bright , x=2.51)

Intelligent Light Controller



Intelligent Light Controller



Transitions:

(Off , x=0)

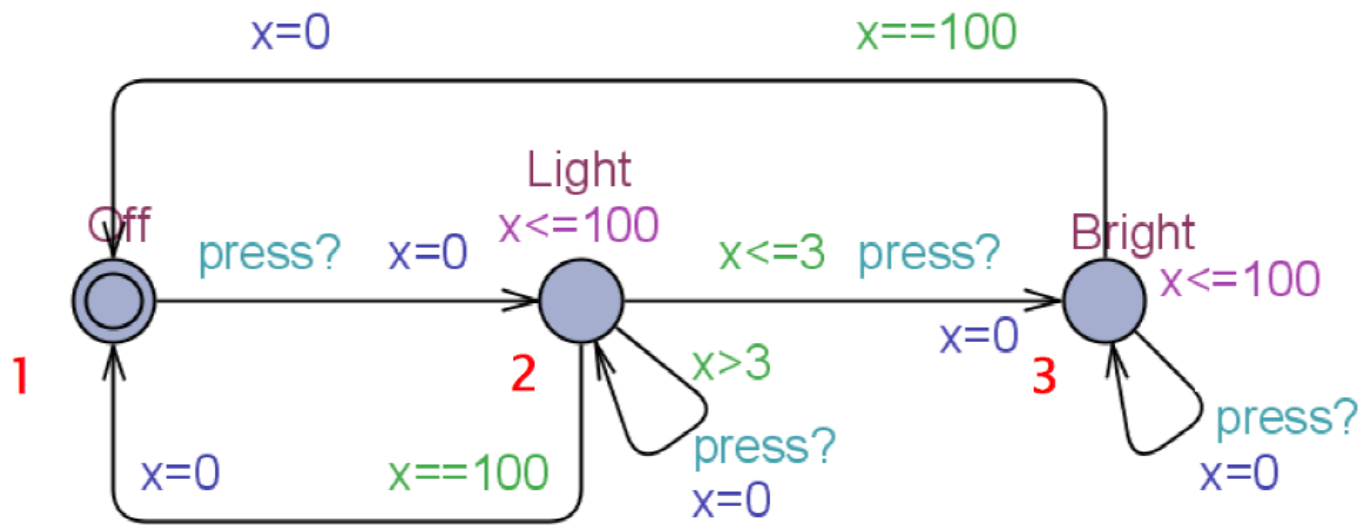
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 4.51	→ (Light , x=4.51)
press?	→ (Light , x=0)
delay 100	→ (Light , x=100)
τ	→ (Off , x=0)

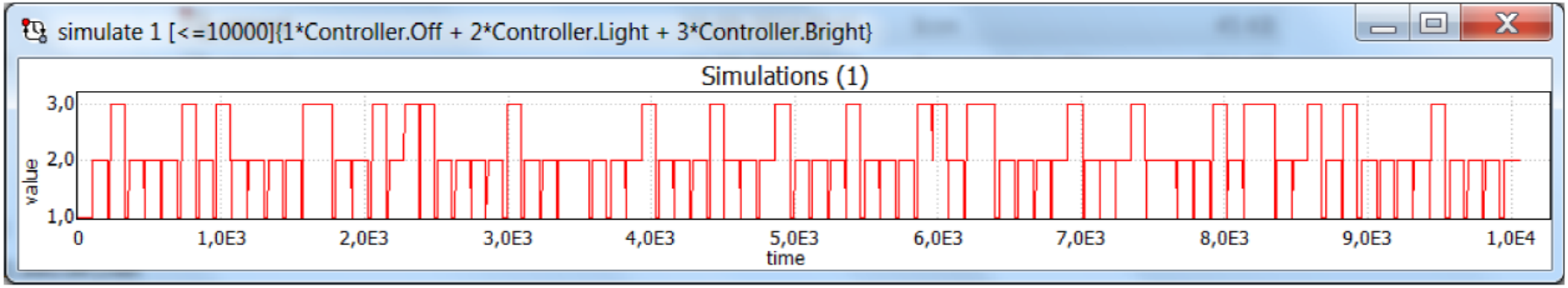
Note:

(Light , x=0) delay 103 →

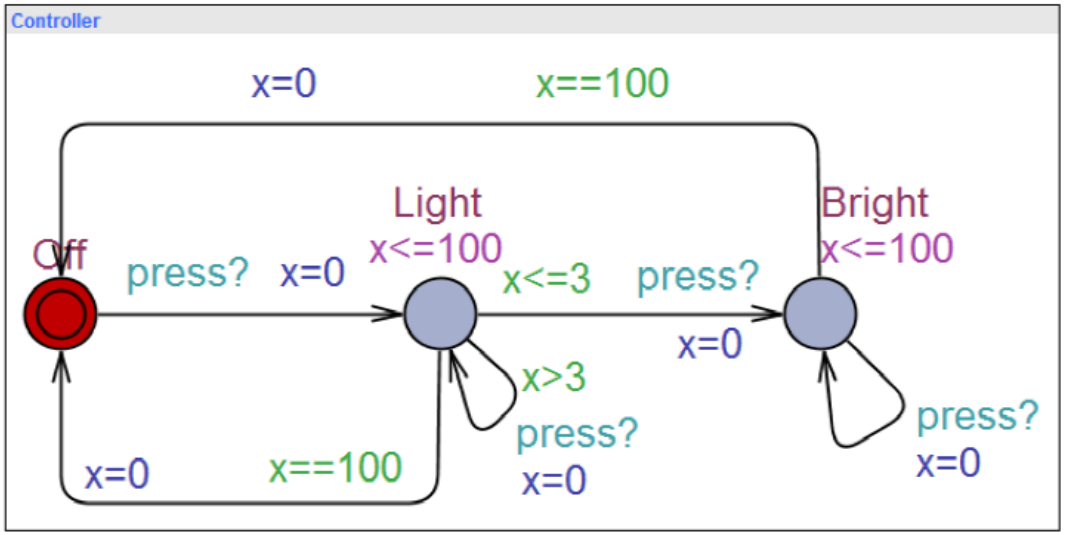
Invariants ensures progress

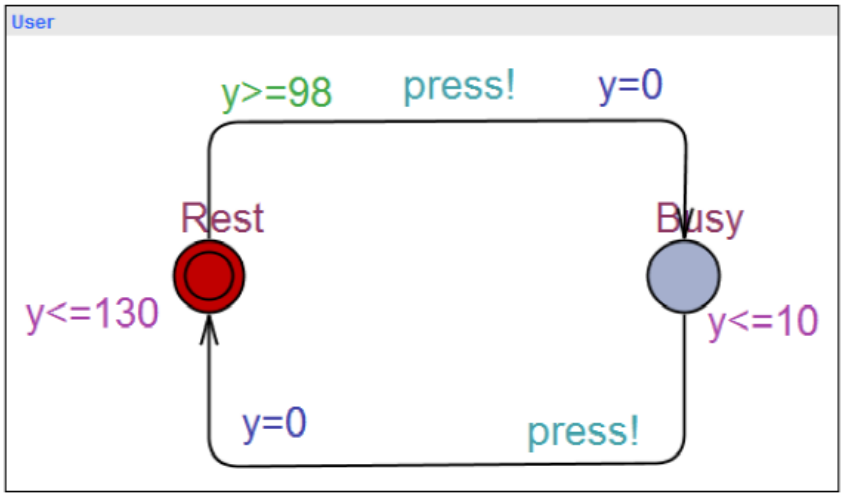
Intelligent Light Controller





UPPAAl Demo


Controller
x=0
x==100
Light
x<=100
Bright
x<=100
Off
press?
x=0
x<=3
press?
x=0
x>3
press?
x=0
press?
x=0
x=0
x==100


User
y>=98
press!
y=0
Rest
Busy
y<=130
y<=10
y=0
press!

Clock Valuations

Let $C = \{x, y, \ldots\}$ be a finite set of clocks.

Set $\mathcal{B}(C)$ of clock constraints over C

$\mathcal{B}(C)$ is defined by the following abstract syntax

$$g, g_1, g_2 \ ::= x \sim n \ \mid \ x - y \sim n \ \mid \ g_1 \wedge g_2$$

where $x, y \in C$ are clocks, $n \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$.

Example:

Clock Valuations – Operations

Clock valuation

Clock valuation v is a function $v : C \to \mathbb{R}^{\geq 0}$.

Let v be a clock valuation. Then

- $v + d$ is a clock valuation for any $d \in \mathbb{R}^{\geq 0}$ and it is defined by

$$(v + d)(x) = v(x) + d \text{ for all } x \in C$$

- $v[r]$ is a clock valuation for any $r \subseteq C$ and it is defined by

$$v[r](x) \begin{cases} 0 & \text{if } x \in r \\ v(x) & \text{otherwise.} \end{cases}$$

Clock Valuations – Evaluation

Evaluation of clock constraints ($v \models g$)

$$
\begin{array}{ll}
v \models x < n & \text{iff } v(x) < n \\
v \models x \leq n & \text{iff } v(x) \leq n \\
v \models x = n & \text{iff } v(x) = n \\
\vdots & \\
v \models x - y < n & \text{iff } v(x) - v(y) < n \\
v \models x - y \leq n & \text{iff } v(x) - v(y) \leq n \\
\vdots & \\
v \models g_1 \wedge g_2 & \text{iff } v \models g_1 \text{ and } v \models g_2
\end{array}
$$

Timed Automata – Syntax

Definition

A timed automaton over a set of clocks C and a set of labels N is a tuple

$$(L, \ell_0, E, I)$$

where

- L is a finite set of locations
- $\ell_0 \in L$ is the initial location
- $E \subseteq L \times \mathcal{B}(C) \times N \times 2^C \times L$ is the set of edges
- $I : L \to \mathcal{B}(C)$ assigns invariants to locations.

We usually write $\ell \xrightarrow{g,a,r} \ell'$ whenever $(\ell, g, a, r, \ell') \in E$.

Timed Automata – Semantics

Let $A = (L, \ell_0, E, I)$ be a timed automaton.

Timed transition system generated by A

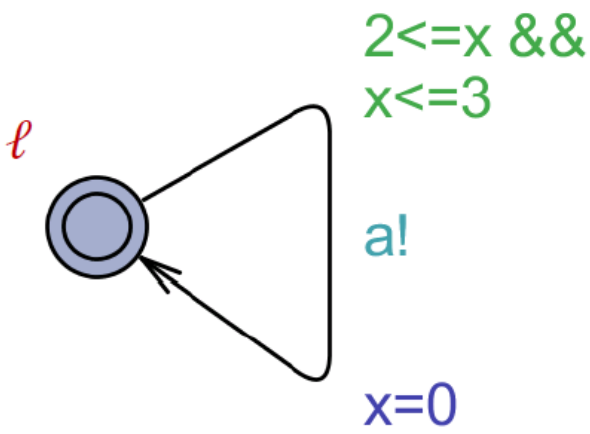
$T(A) = (Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ where

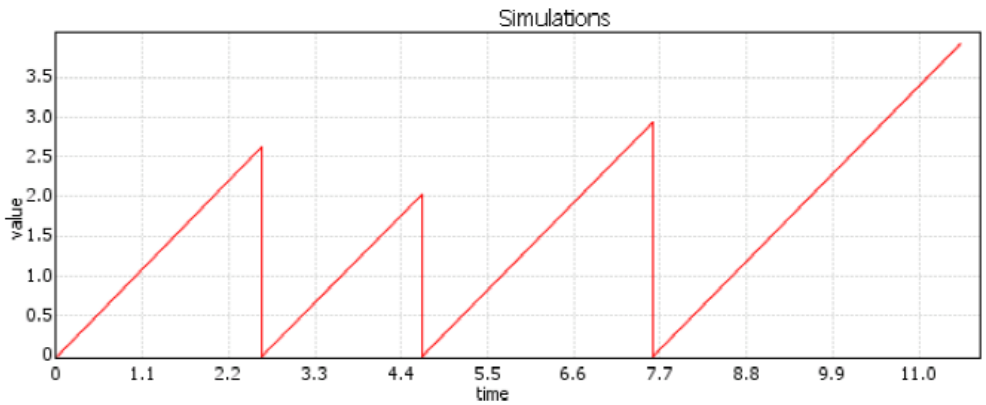
- $Proc = L \times (C \to \mathbb{R}^{\geq 0})$, i.e. states are of the form (ℓ, v) where ℓ is a location and v a valuation
- $Act = N \cup \mathbb{R}^{\geq 0}$
- $\longrightarrow$ is defined as follows:

$(\ell, v) \xrightarrow{a} (\ell', v')$ if there is $(\ell \xrightarrow{g,a,r} \ell') \in E$ s.t. $v \models g$ and $v' = v[r]$

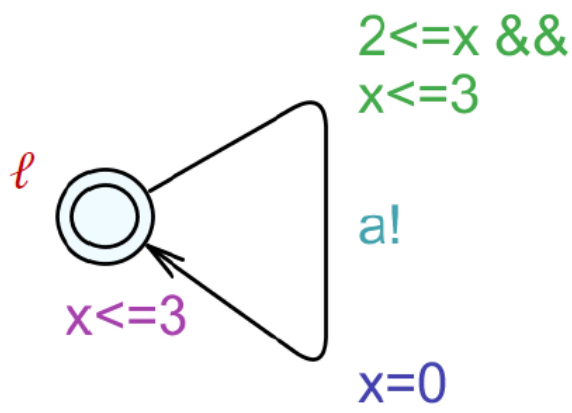
$(\ell, v) \xrightarrow{d} (\ell, v + d)$ for all $d \in \mathbb{R}^{\geq 0}$ s.t. $v \models I(\ell)$ and $v + d \models I(\ell)$

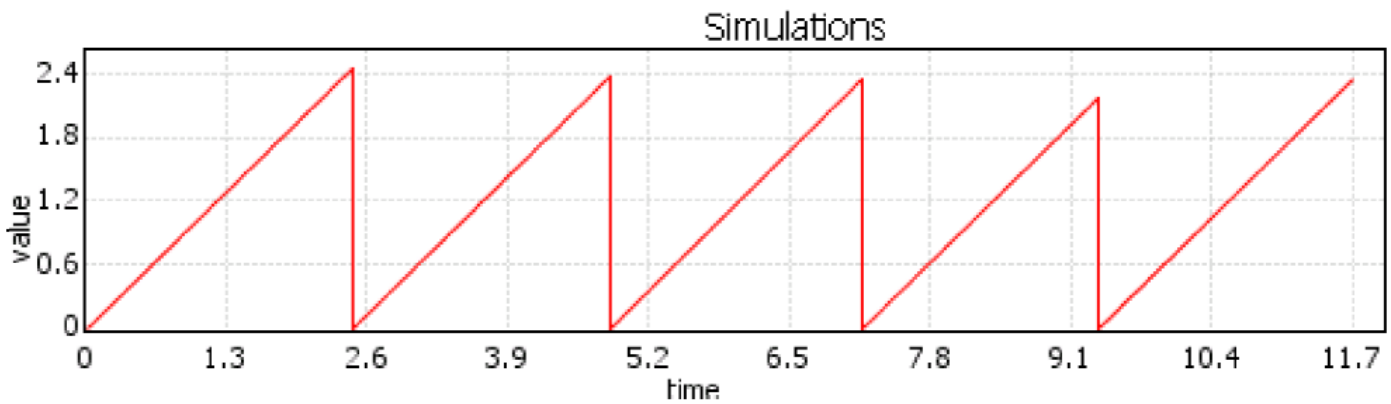
Example


ℓ
2<=x &&
x<=3
a!
x=0

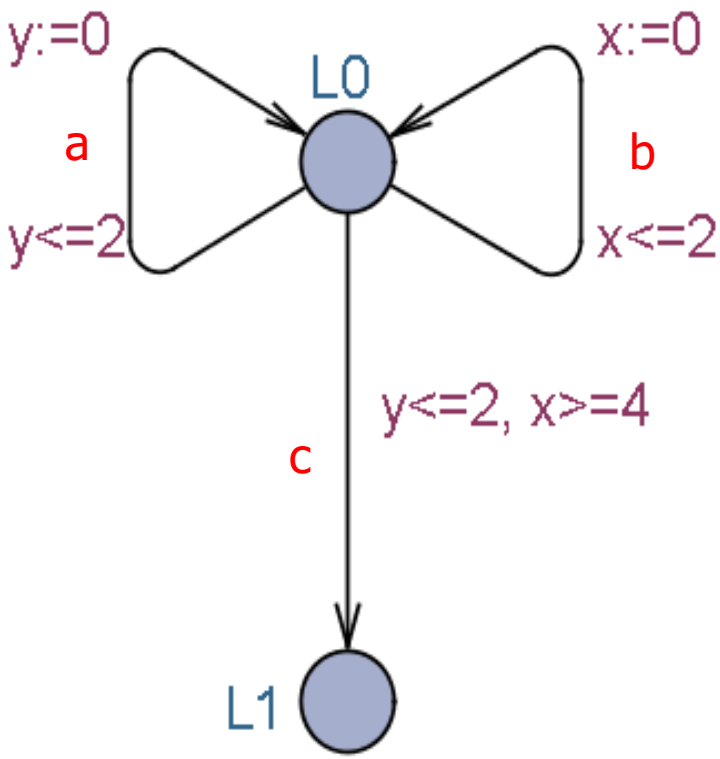

Simulations
value
time
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1.1
2.2
3.3
4.4
5.5
6.6
7.7
8.8
9.9
11.0

Example


2<=x &&
x<=3
ℓ
a!
x<=3
x=0

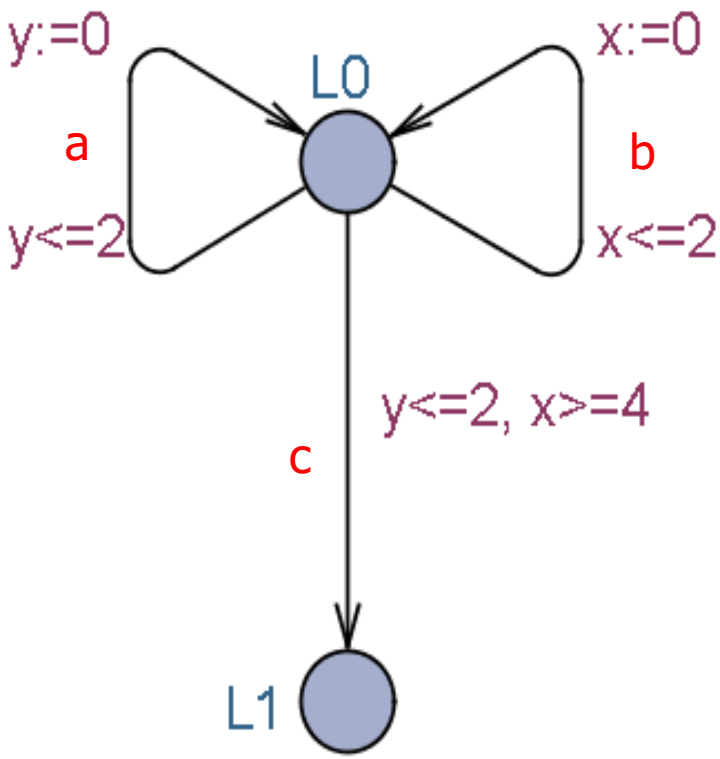

Simulations
value
2.4
1.8
1.2
0.6
0
0
1.3
2.6
3.9
5.2
6.5
7.8
9.1
10.4
11.7
time

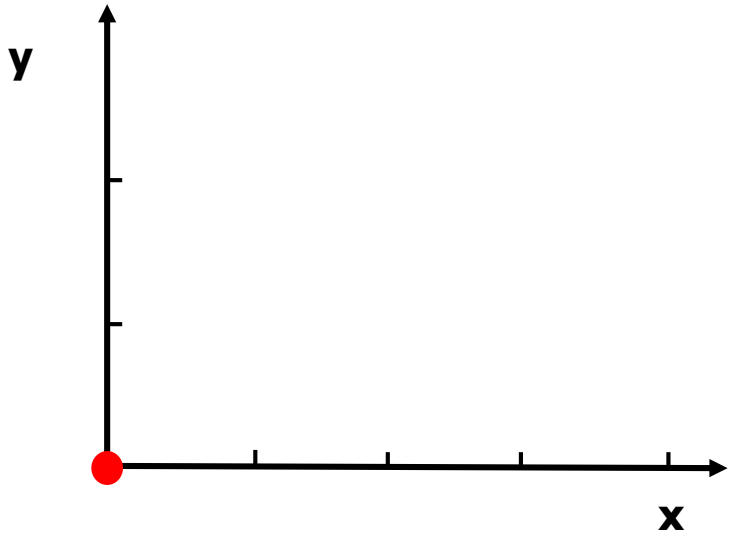
Example



Is L1 reachable ?

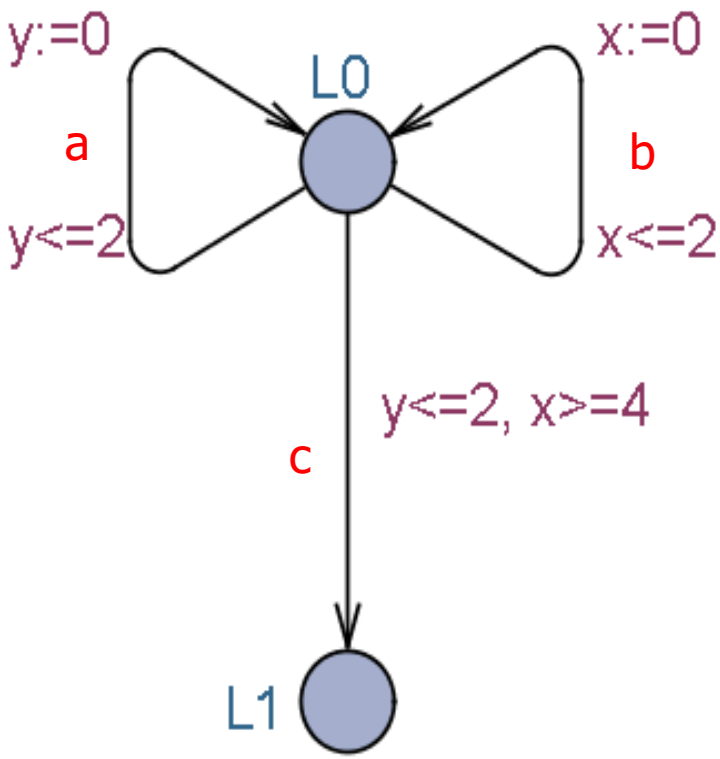
Example

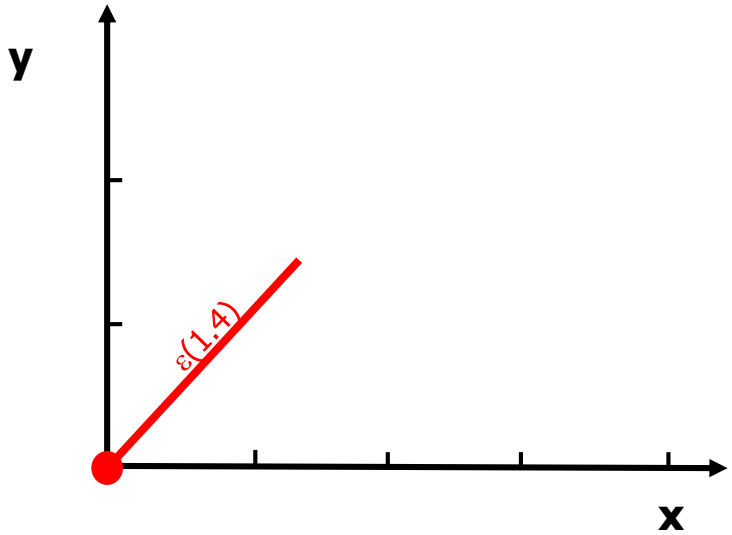




$(\ell_0, x = 0, y = 0)$

Example

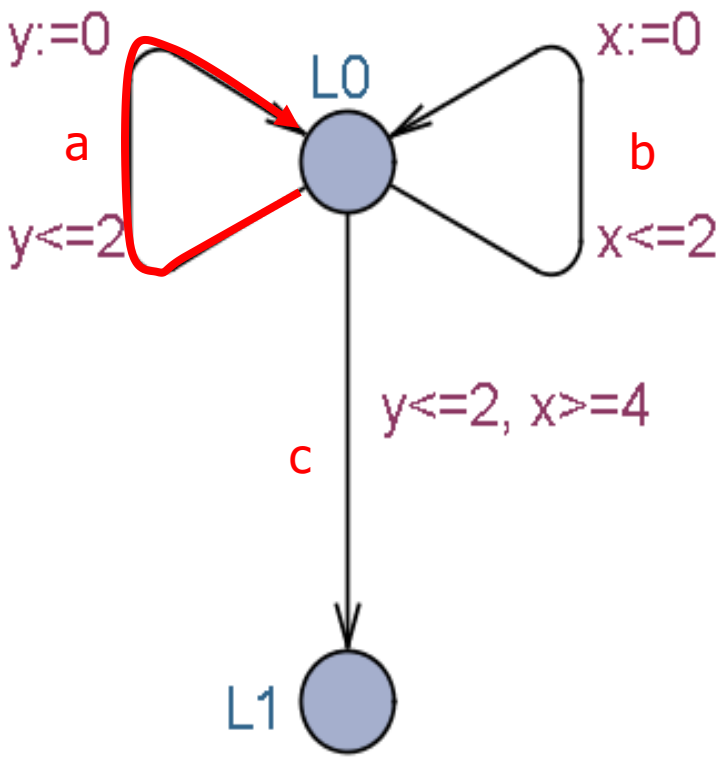


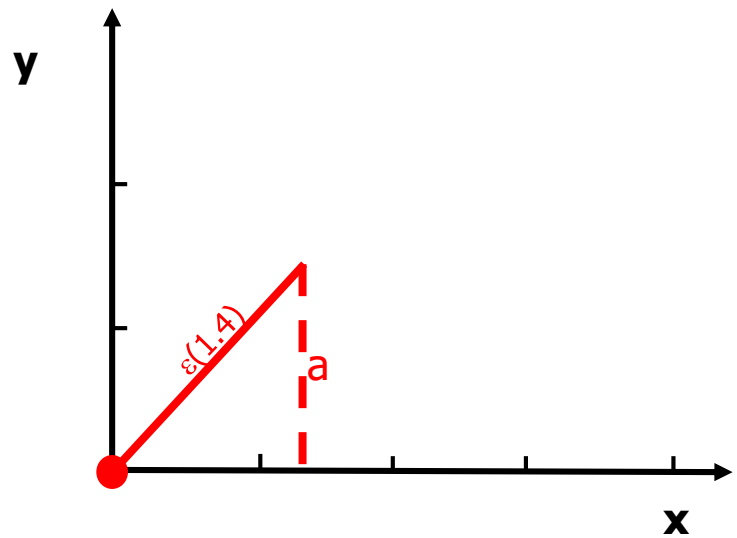


$$(\ell_0, x = 0, y = 0)$$

$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$

Example



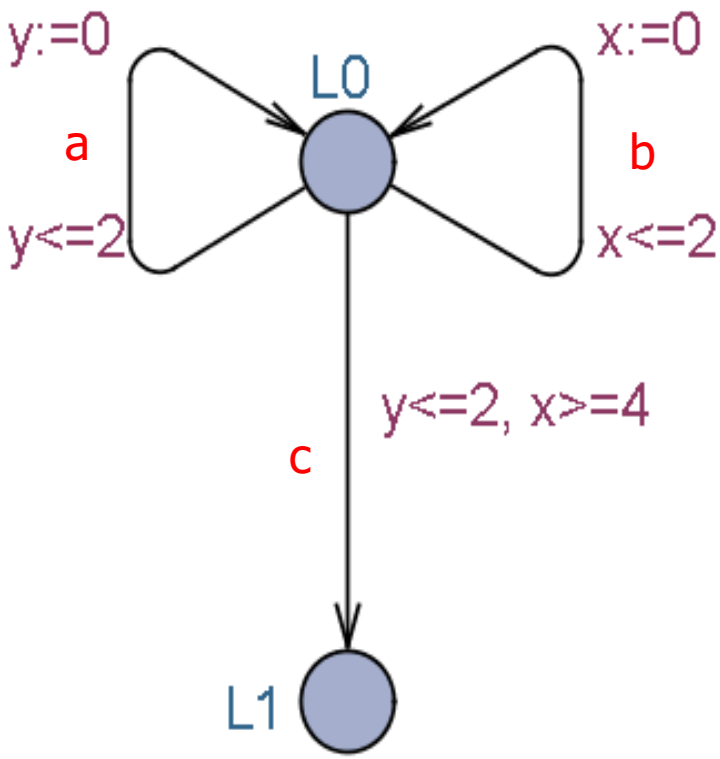


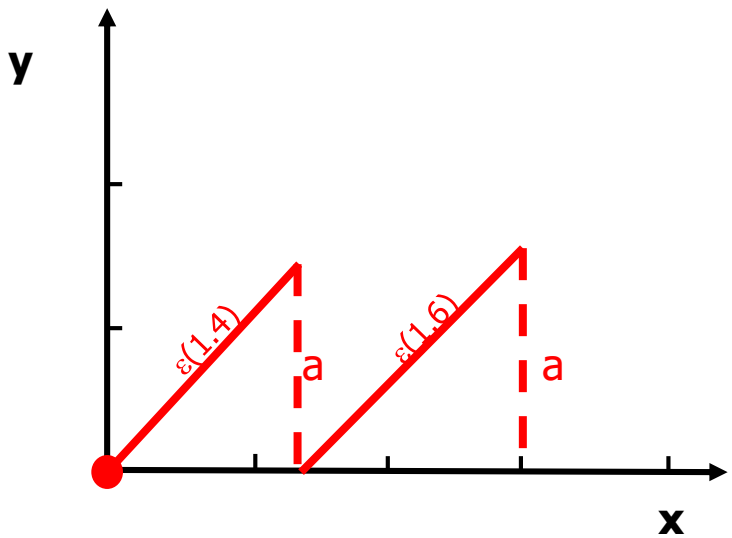
$$(\ell_0, x = 0, y = 0)$$

$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$

$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

Example





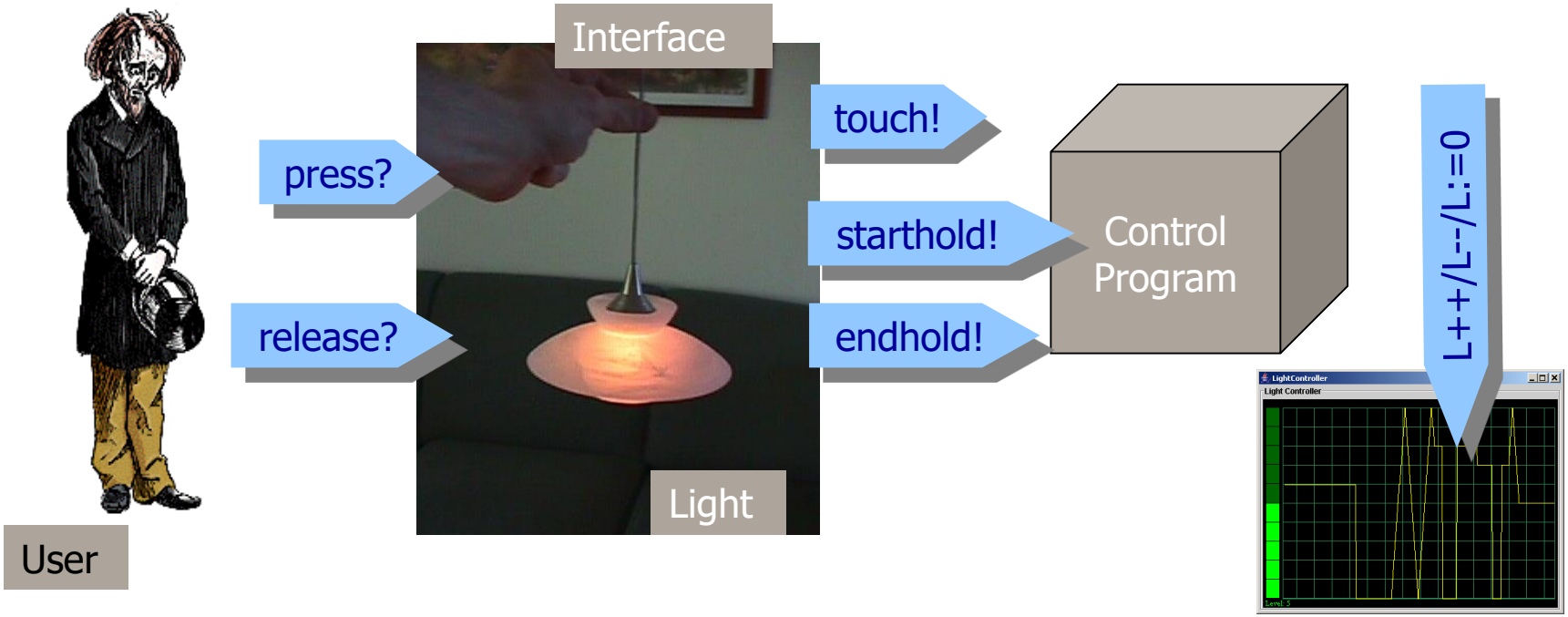
$$(\ell_0, x = 0, y = 0)$$
$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$
$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$
$$\xrightarrow{1.6} (\ell_0, x = 3.0, y = 1.6)$$
$$\xrightarrow{a} (\ell_0, x = 3.0, y = 0)$$

UPPAAL

First Introduction

Light Control Interface

press? d release?	→ touch!	$0.5 \leq d \leq 1$
press? 1	→ starthold!	
press? d release?	→ endhold!	$d > 1$

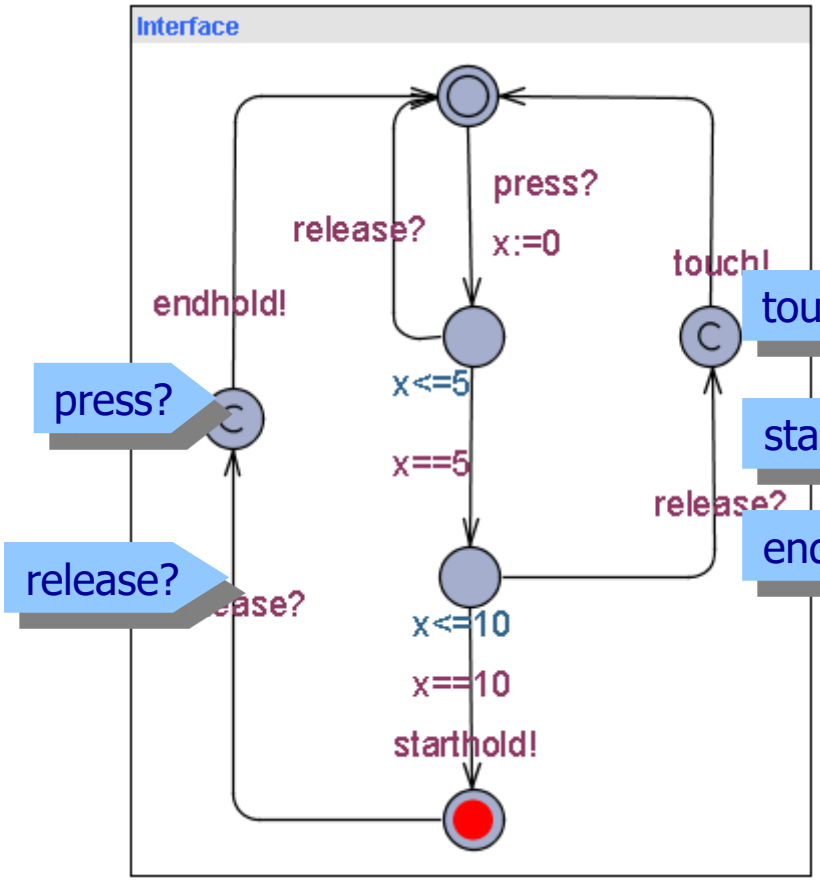


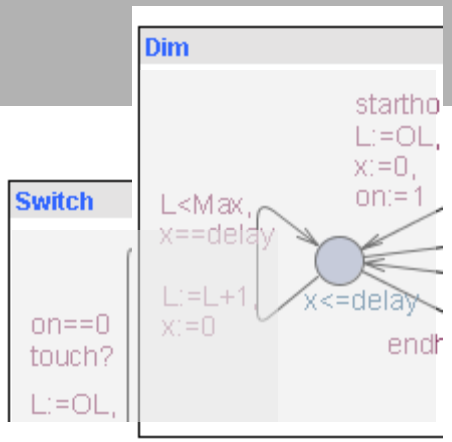
press? 0.2 release? … press? 0.7 release? … press? 1.0 2.4 release? …

↓ ∅ ↓ touch! ↓ starthold! ↓ endhold!

Light Control Interface

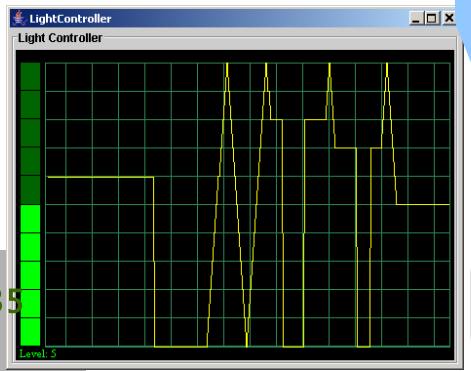
User



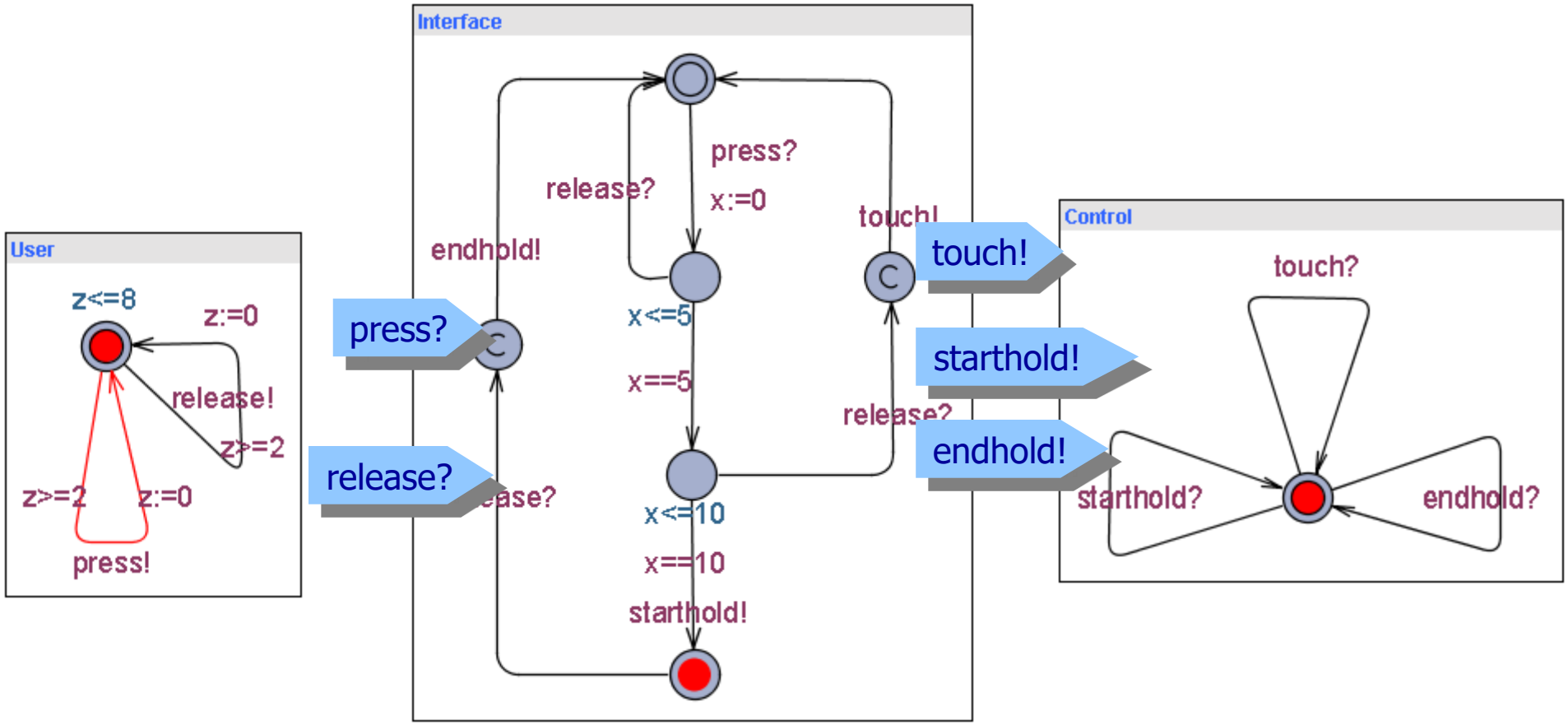


Control Program

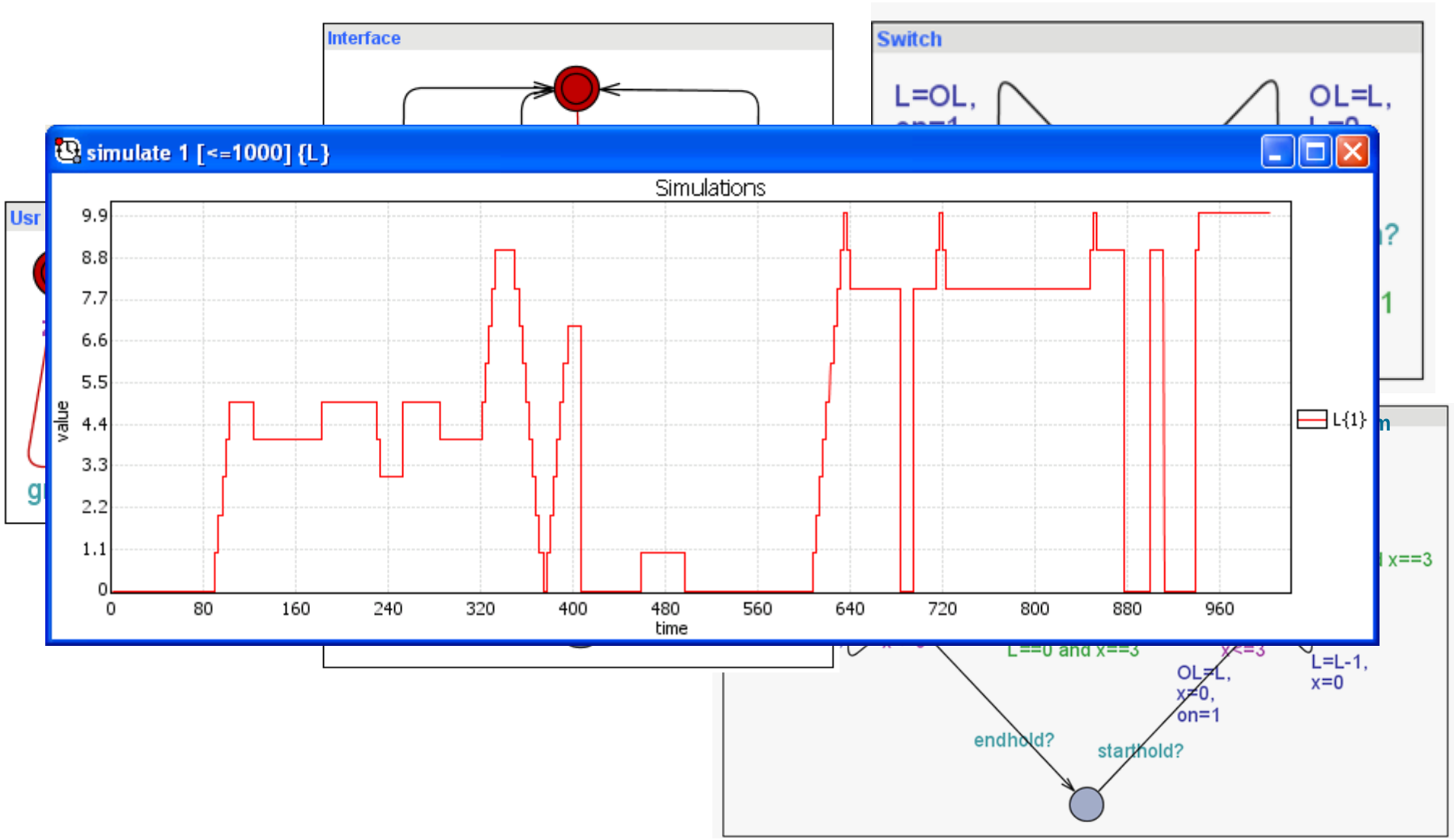
L++/L--/L:=0



Light Control Network

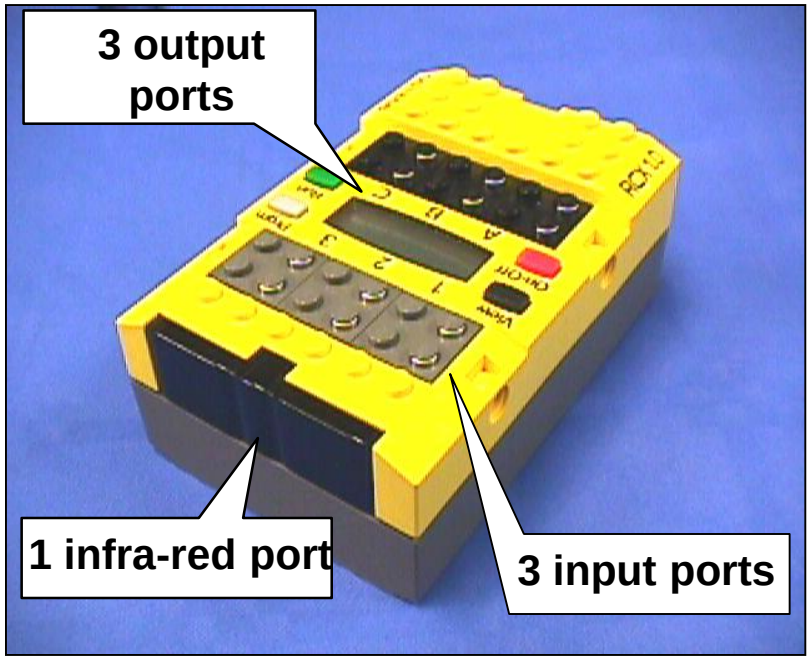


Full Light Controller



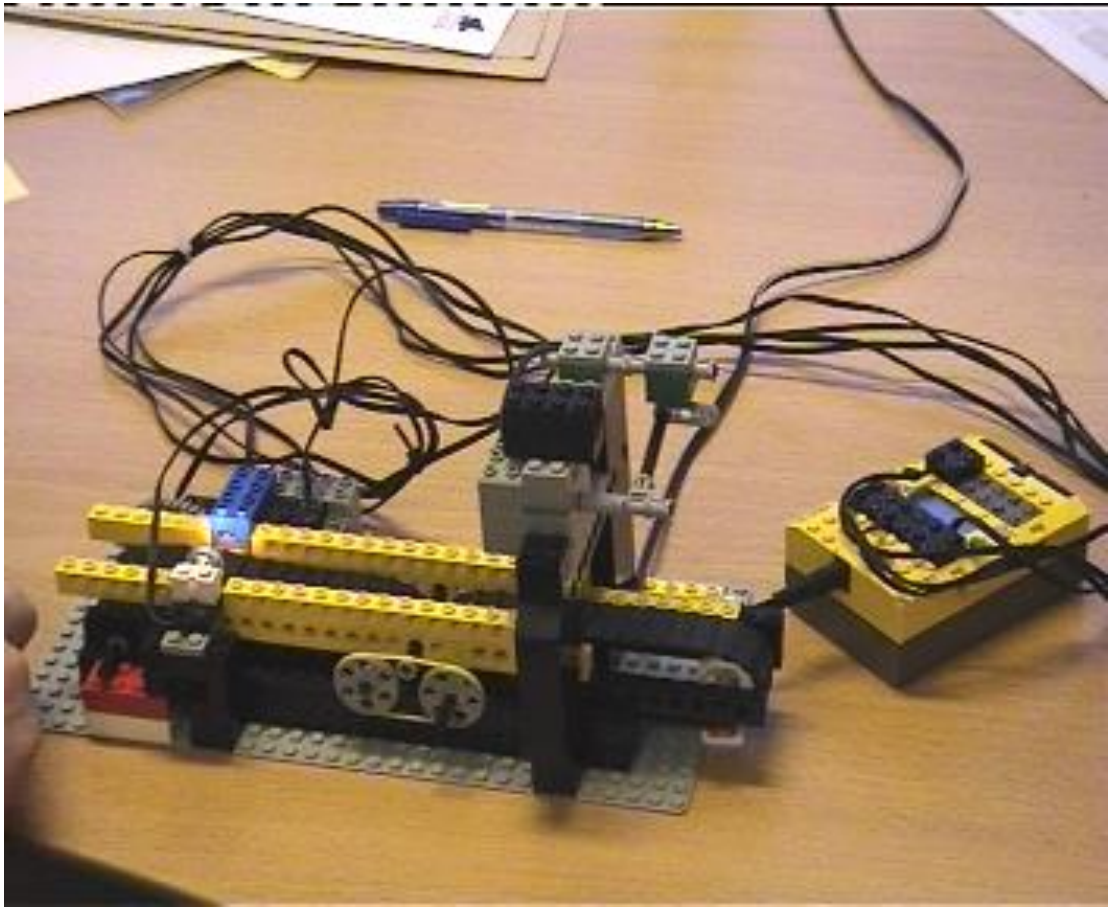
LEGO Mindstorms/RCX

- 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

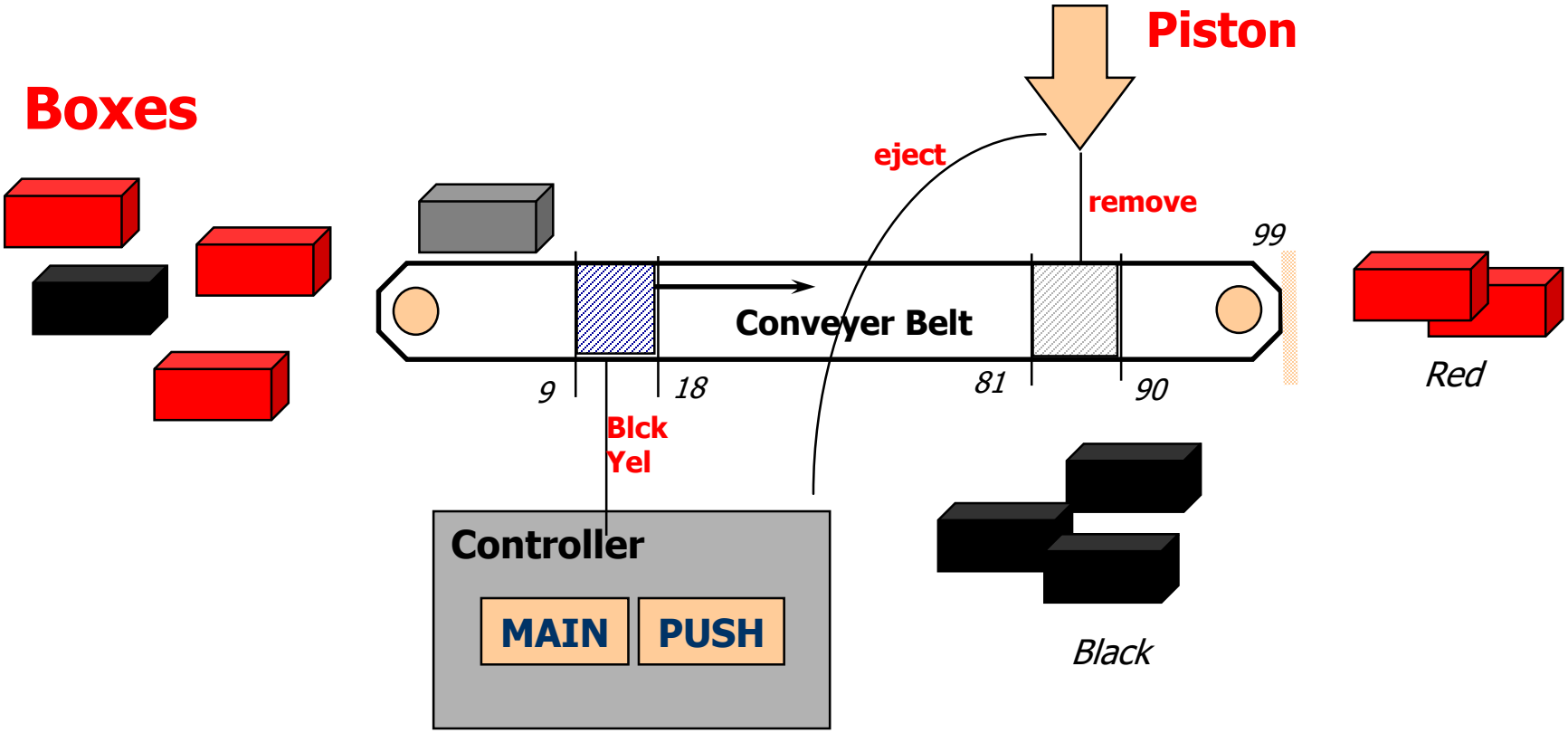


Controller Program
LEGO MINDSTORM

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



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

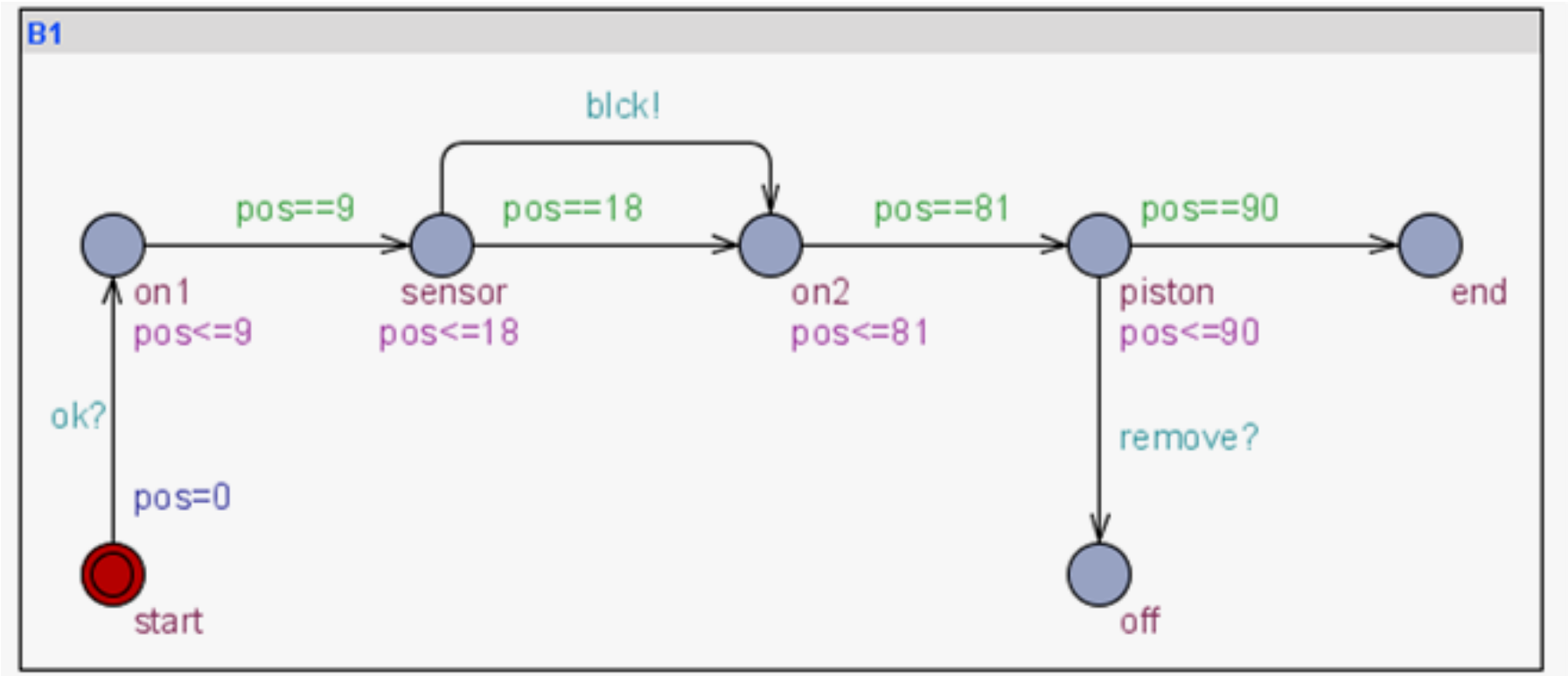
NQC programs

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

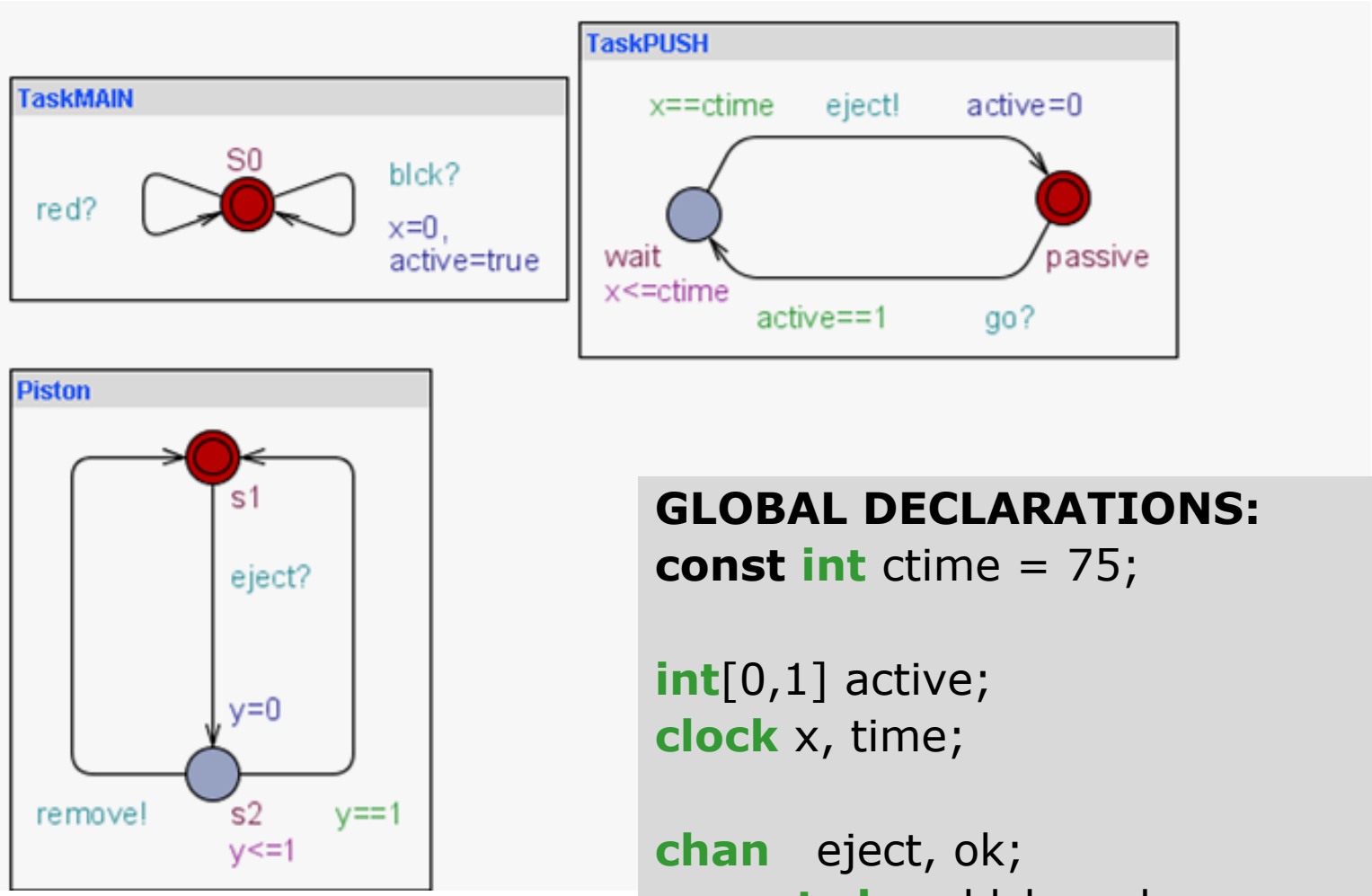
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 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



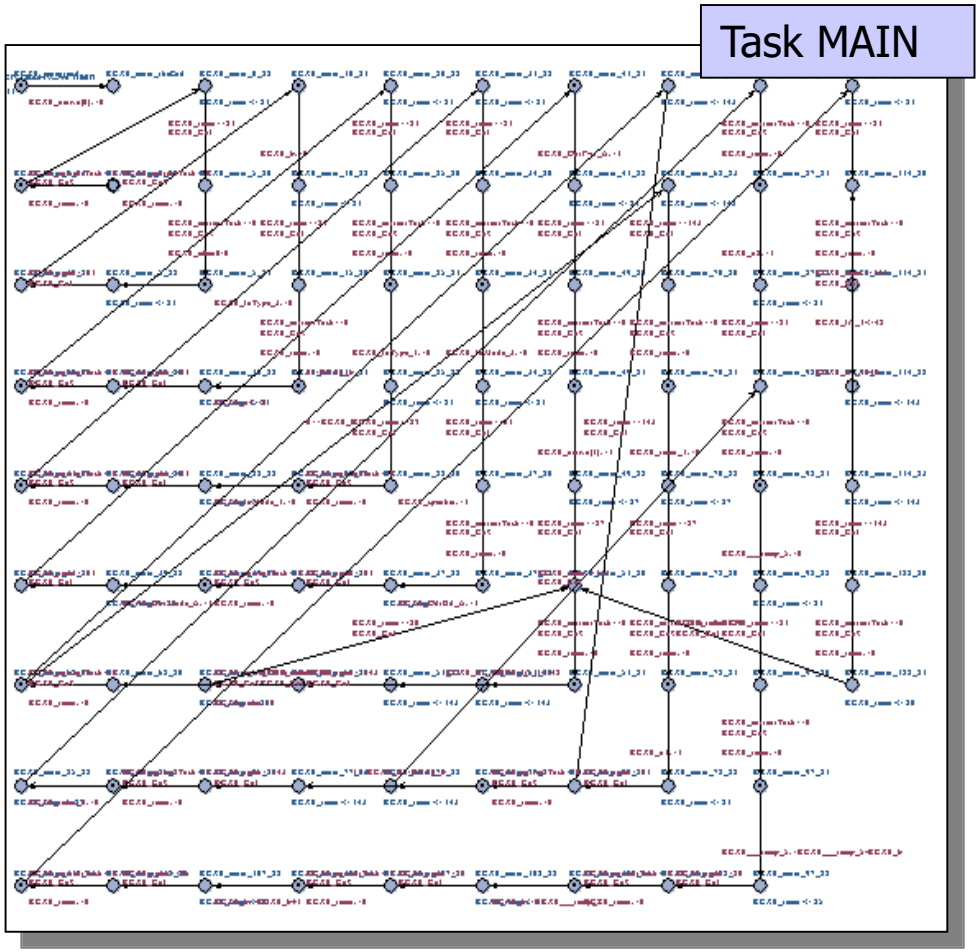
GLOBAL DECLARATIONS:

```
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

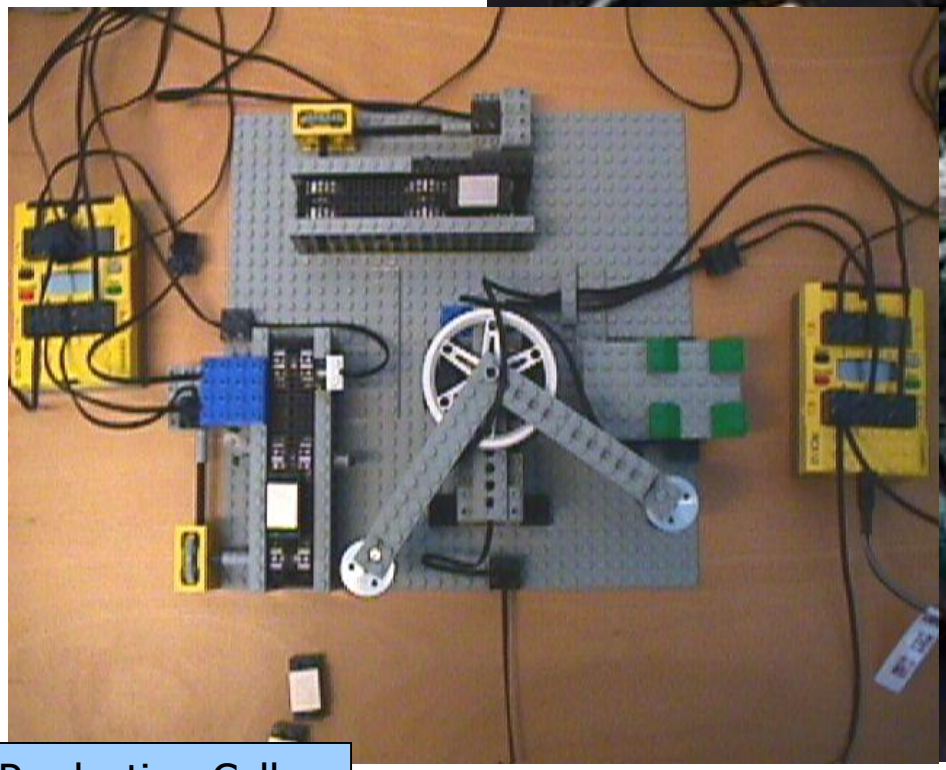
- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000 in Stockholm.

- From UPPAAL to RCX: Martijn Hendriks.

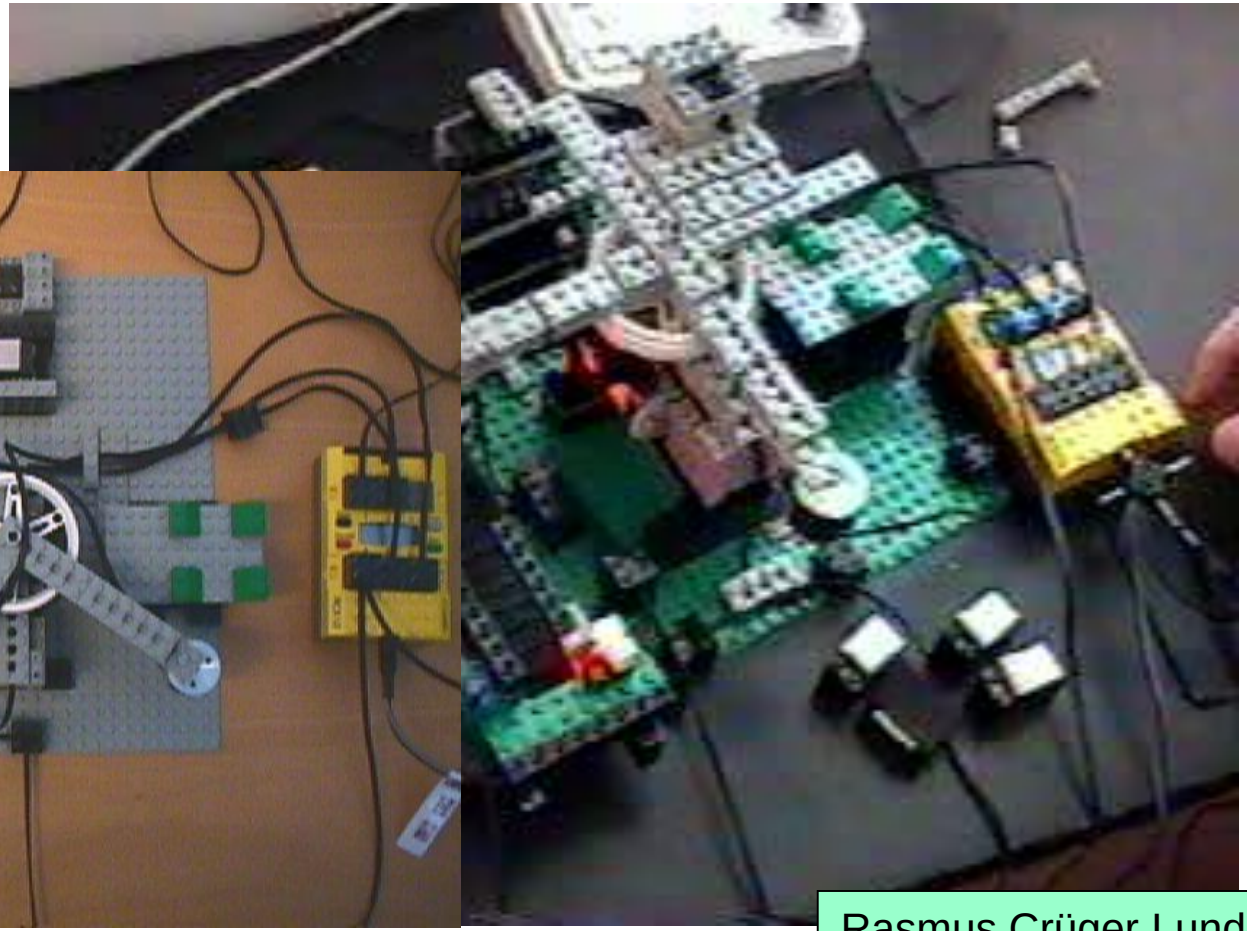


The Production Cell in LEGO

Course at DTU, Copenhagen



Production Cell

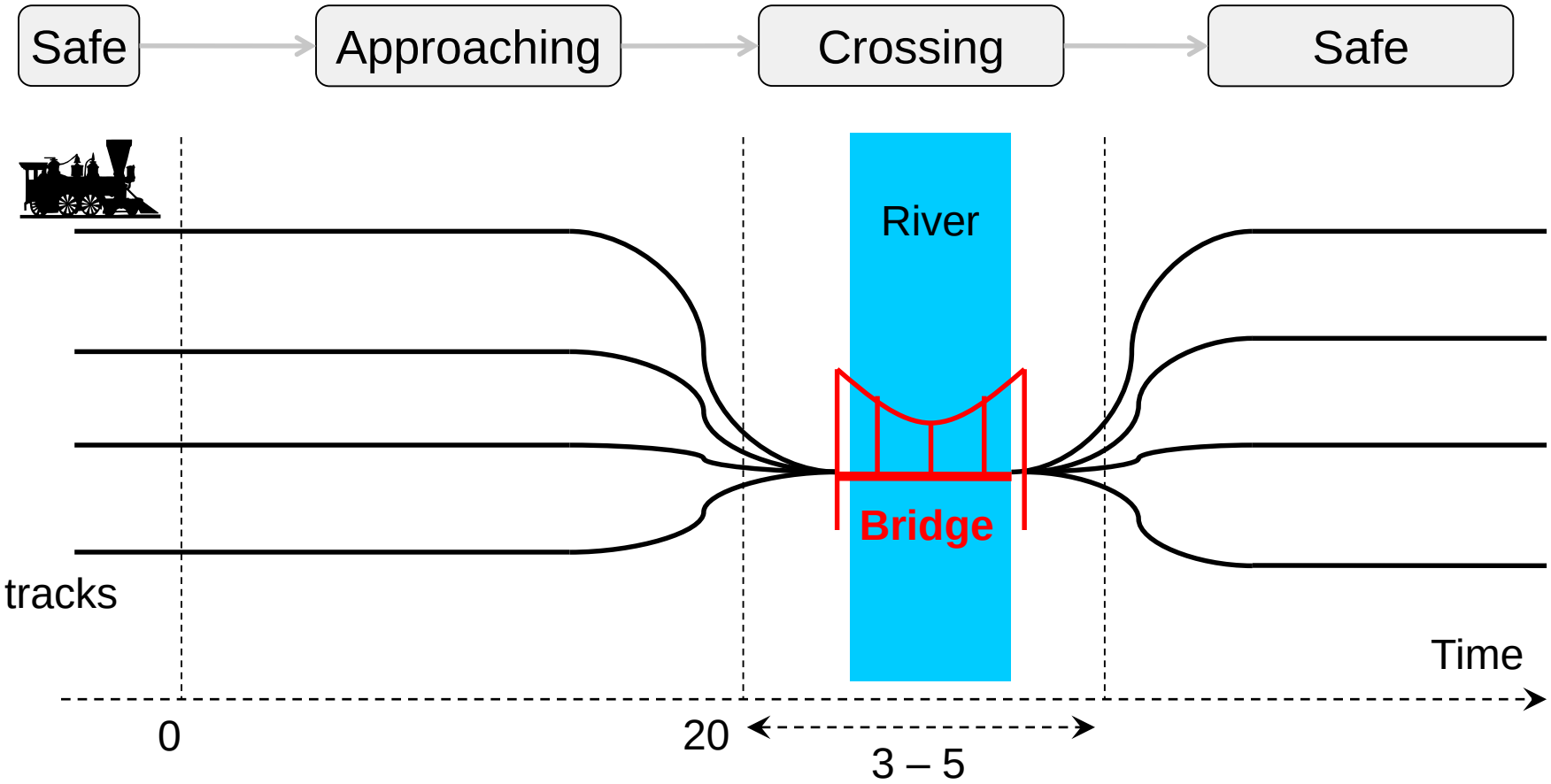


Rasmus Crüger Lund
Simon Tune Riemanni

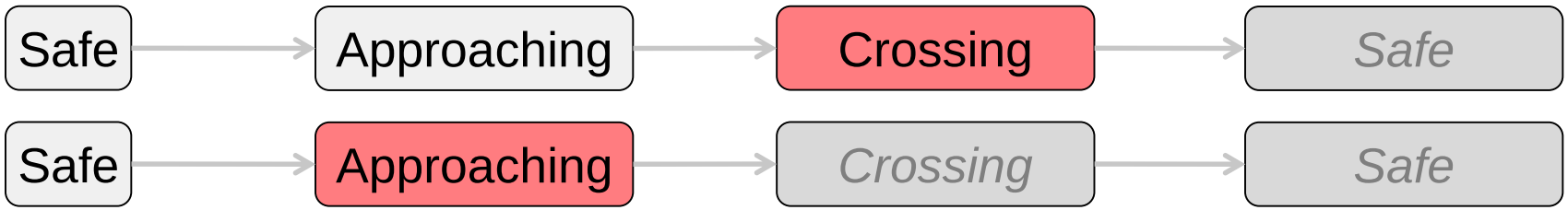
UPPAAL

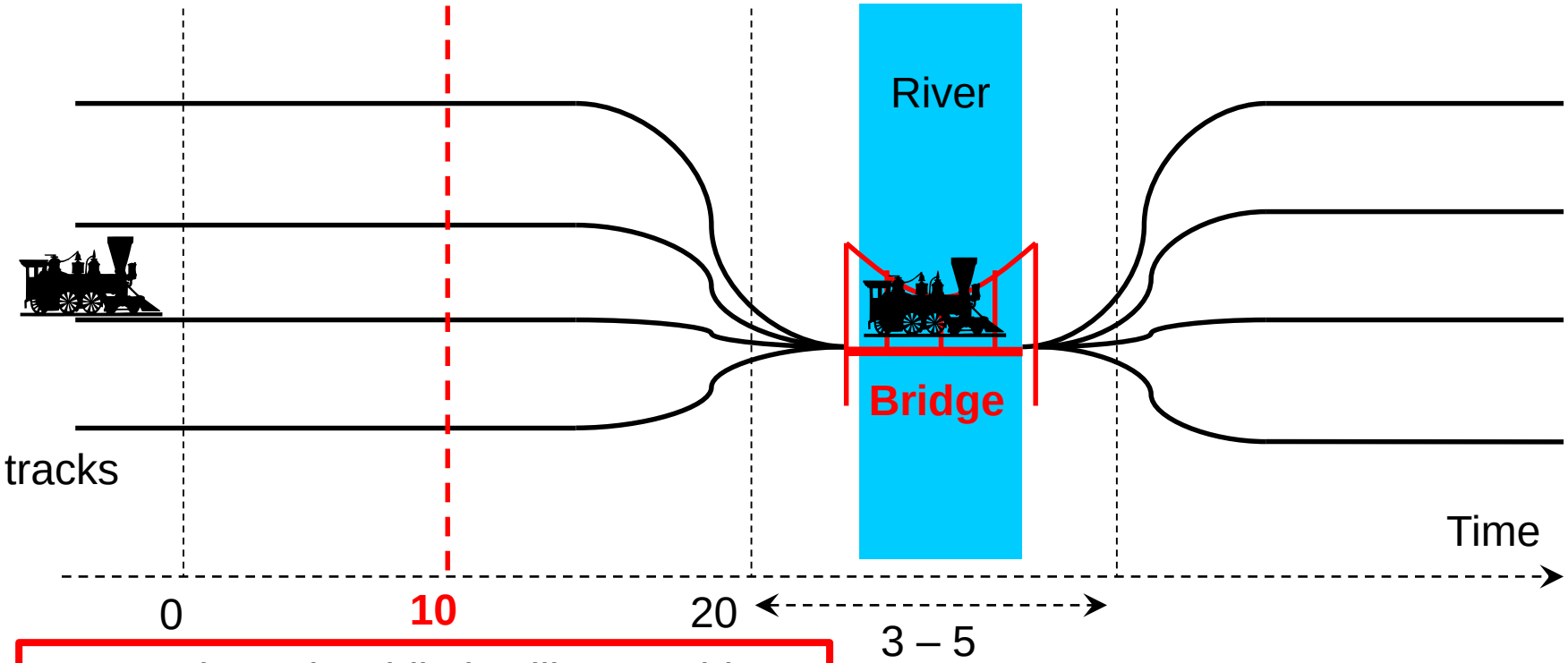
Modeling & Specification

Train Crossing



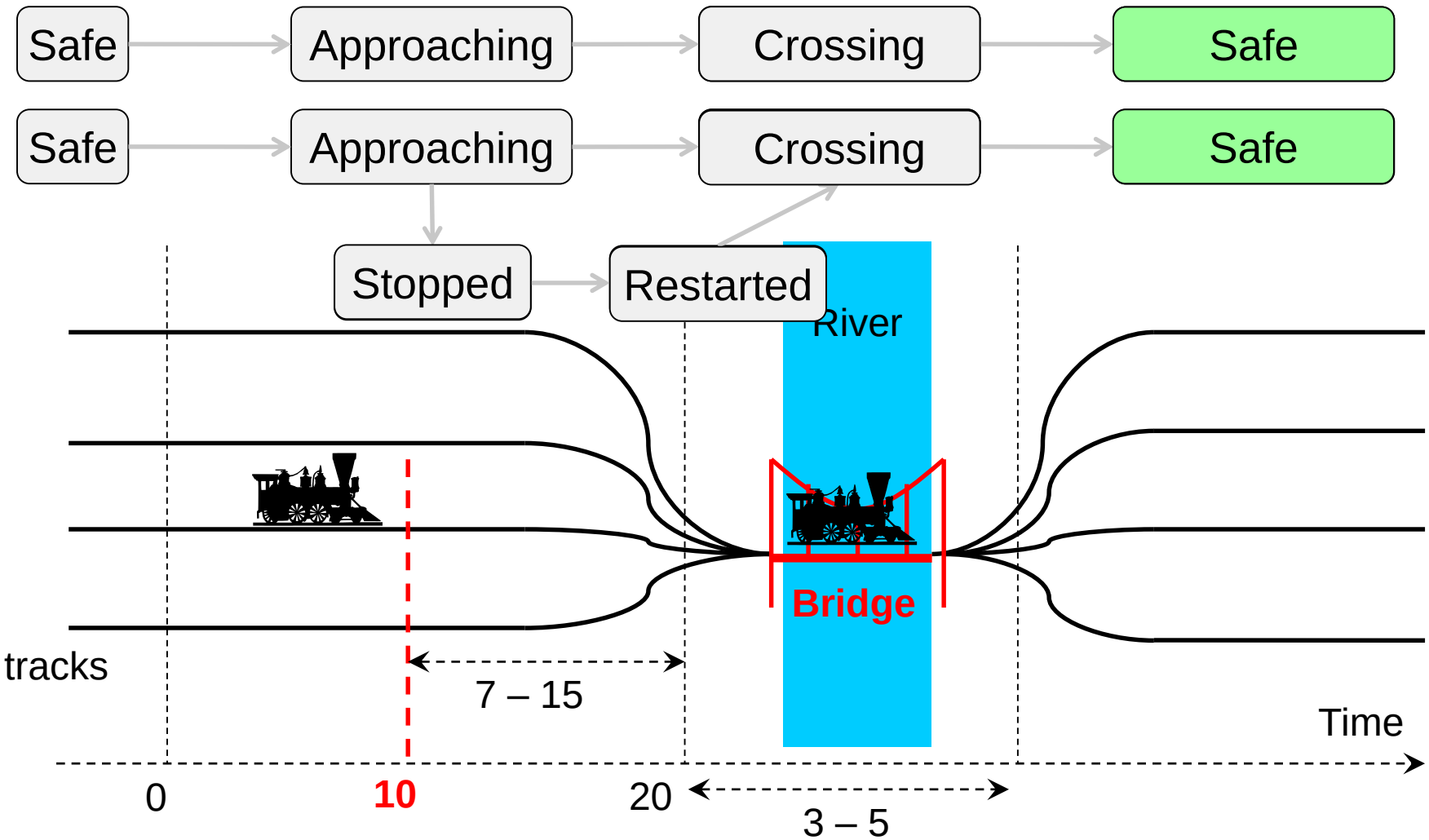
Train Crossing



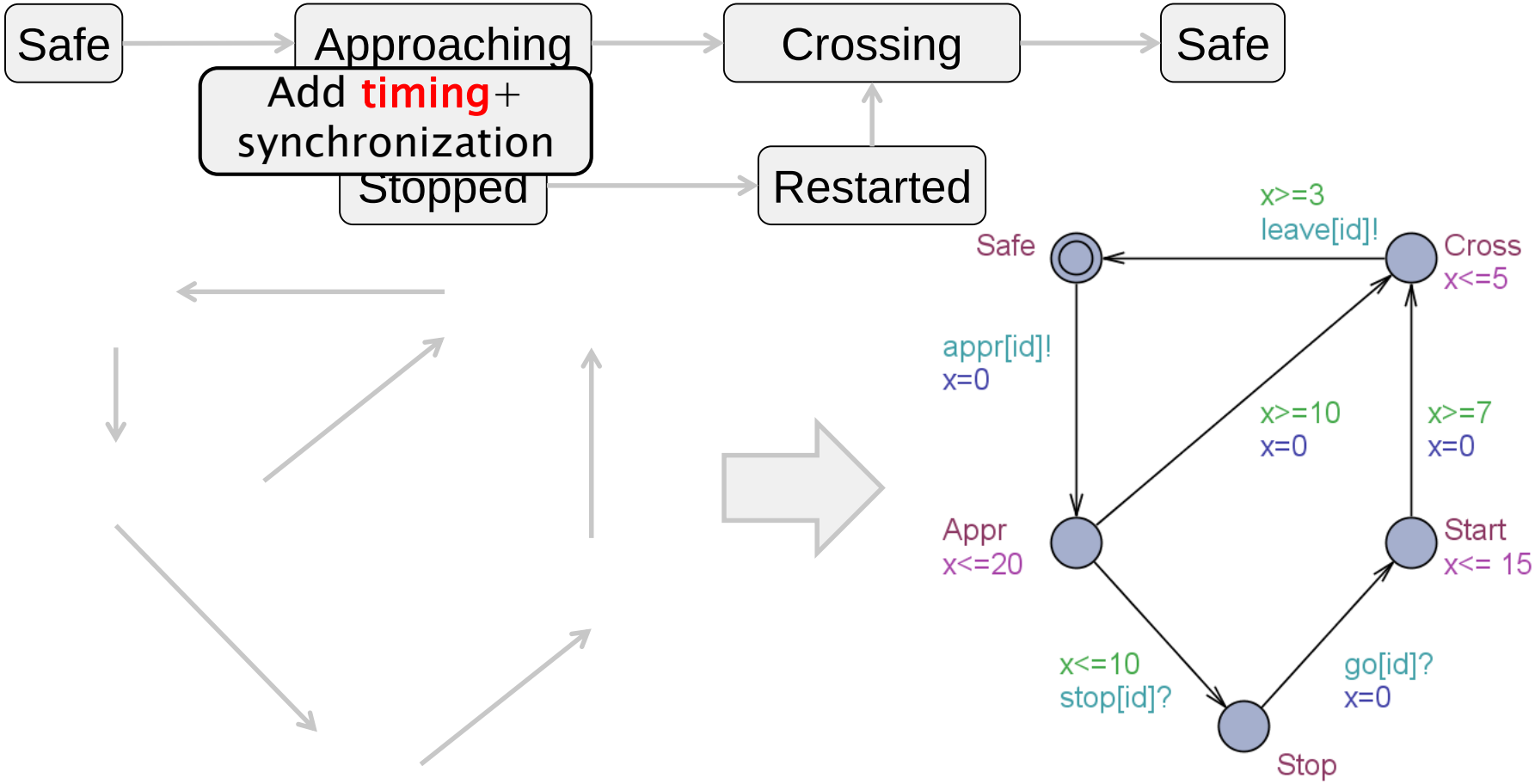


Stop the train while it still stoppable!

Train Crossing

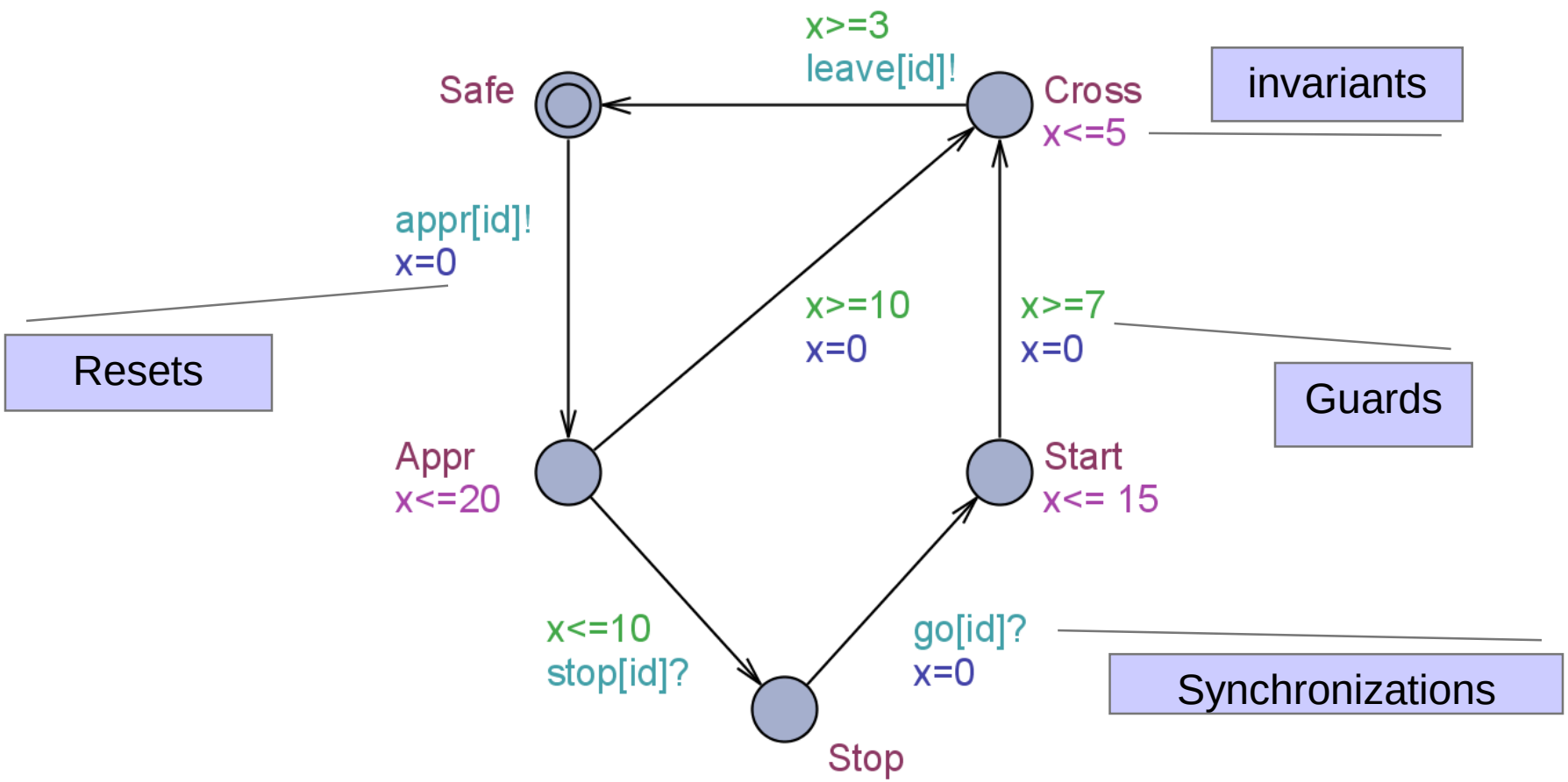


Train Crossing



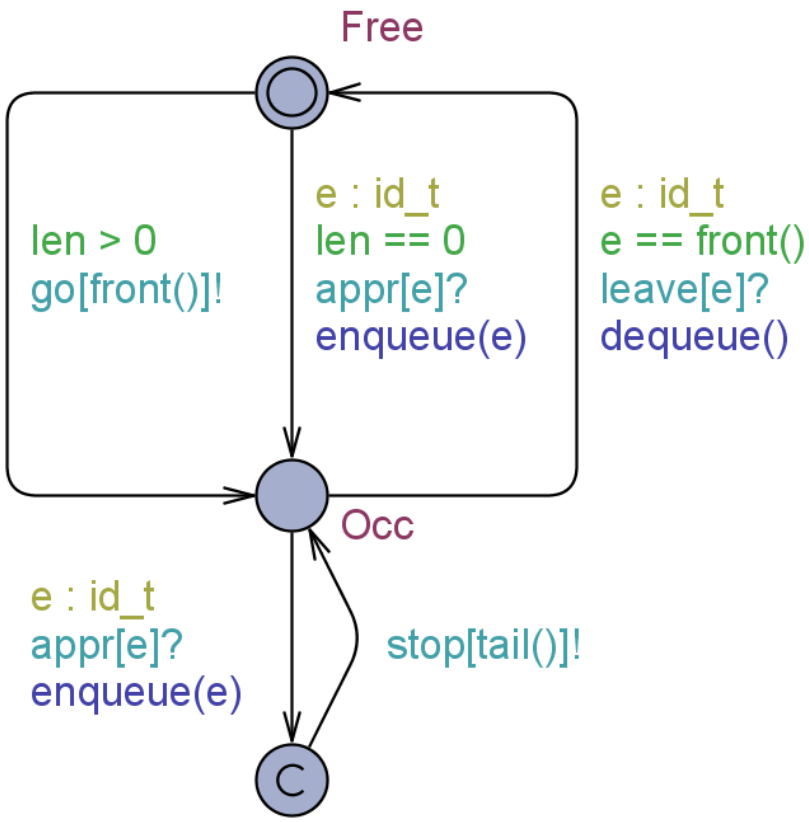
Timed Automata [Train]

= Finite State Control
+ Real Valued Clocks



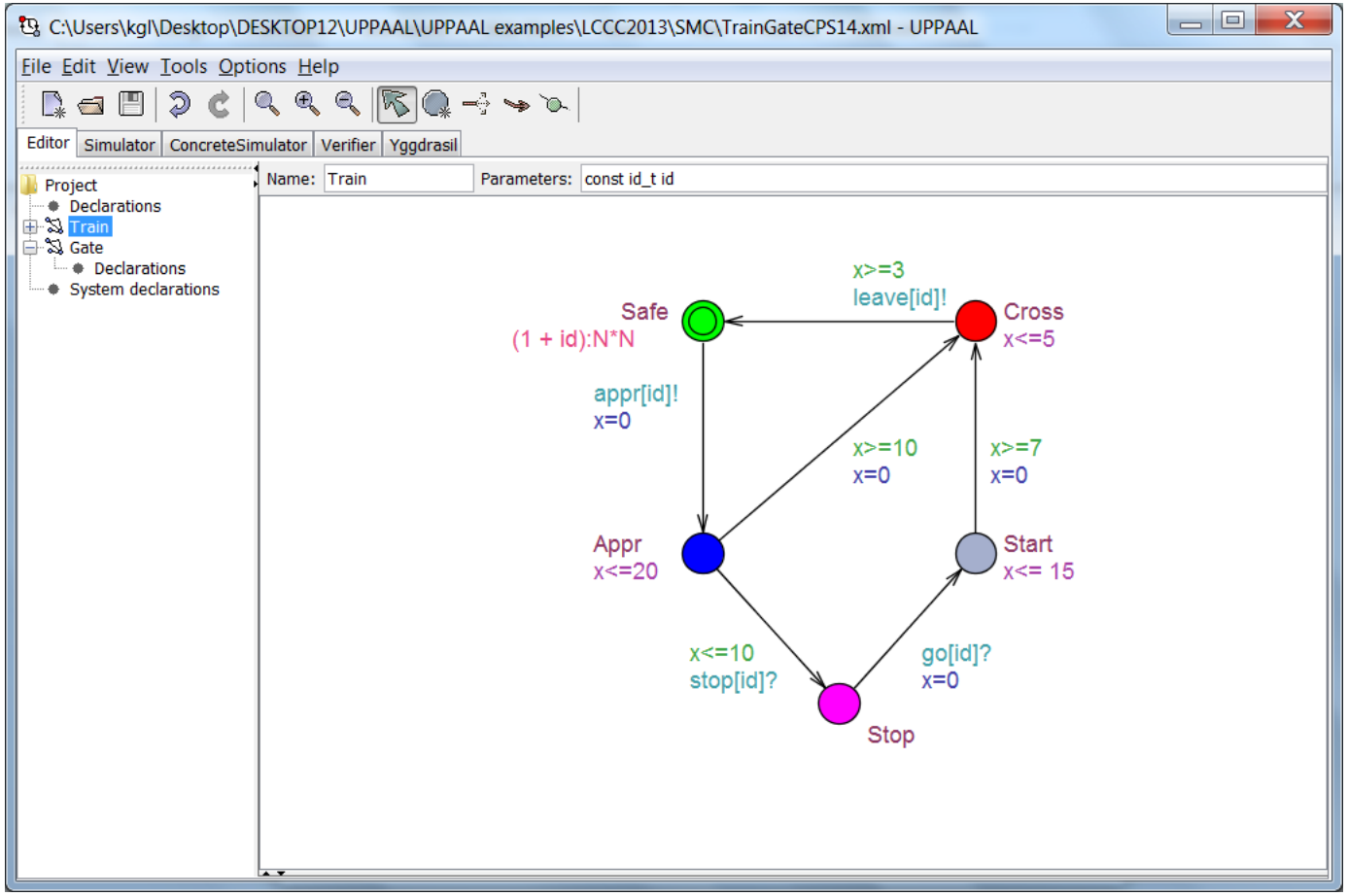
Timed Automata [Gate]

= Finite State Control
+ Real Valued Clocks
+ Discrete Variables

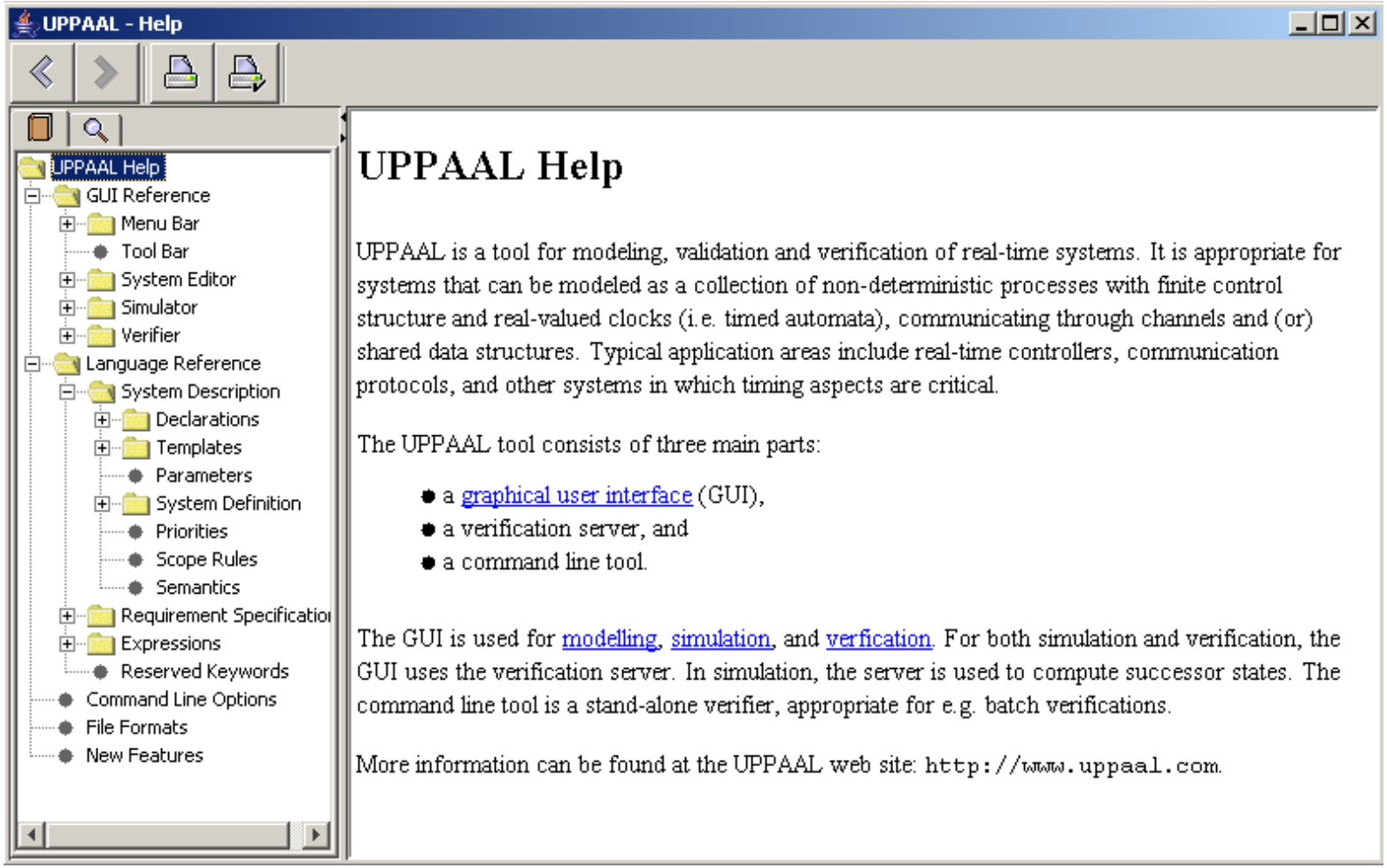



```
id_t list[N+1];
int[0,N] len;

// Put an element at the end of the queue
void enqueue(id_t element)
{
        list[len++] = element;
}

// Remove the front element of the queue
void dequeue()
{
        int i = 0;
        len -= 1;
        while (i < len)
        {
                list[i] = list[i + 1];
                i++;
        }
        list[i] = 0;
}
```

Demo 1


C:\Users\kgl\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS14.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier Yggdrasil
Project
Declarations
Train
Gate
Declarations
System declarations
Name: Train
Parameters: const id_t id
x>=3
leave[id]!
Safe
(1 + id):N*N
Cross
x<=5
appr[id]!
x=0
x>=10
x=0
x>=7
x=0
Appr
x<=20
Start
x<= 15
x<=10
stop[id]?
go[id]?
x=0
Stop

UPPAAL Help


UPPAAL - Help
UPPAAL Help
GUI Reference
Menu Bar
Tool Bar
System Editor
Simulator
Verifier
Language Reference
System Description
Declarations
Templates
Parameters
System Definition
Priorities
Scope Rules
Semantics
Requirement Specificatio
Expressions
Reserved Keywords
Command Line Options
File Formats
New Features
UPPAAL Help
UPPAAL is a tool for modeling, validation and verification of real-time systems. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks (i.e. timed automata), communicating through channels and (or) shared data structures. Typical application areas include real-time controllers, communication protocols, and other systems in which timing aspects are critical.
The UPPAAL tool consists of three main parts:
a graphical user interface (GUI),
a verification server, and
a command line tool.
The GUI is used for modelling, simulation, and verfication. For both simulation and verification, the GUI uses the verification server. In simulation, the server is used to compute successor states. The command line tool is a stand-alone verifier, appropriate for e.g. batch verifications.
More information can be found at the UPPAAL web site: http://www.uppaal.com.

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: $P \rightarrow_{\leq t} Q$

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).

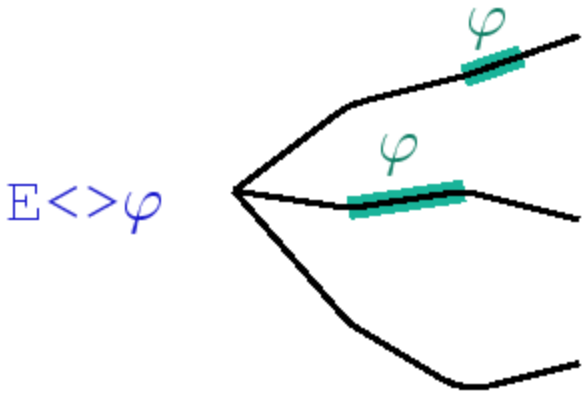
Logical Specifications

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

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

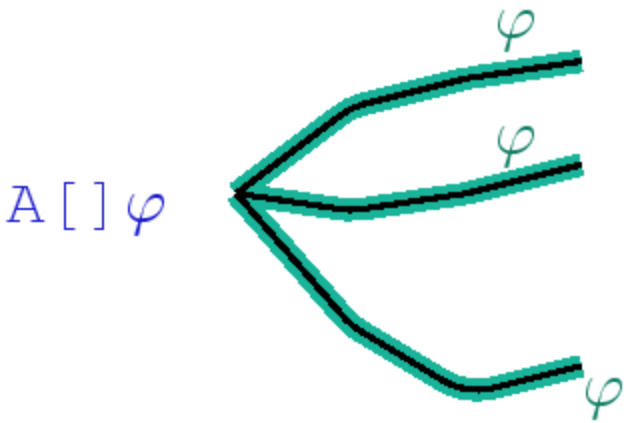
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*

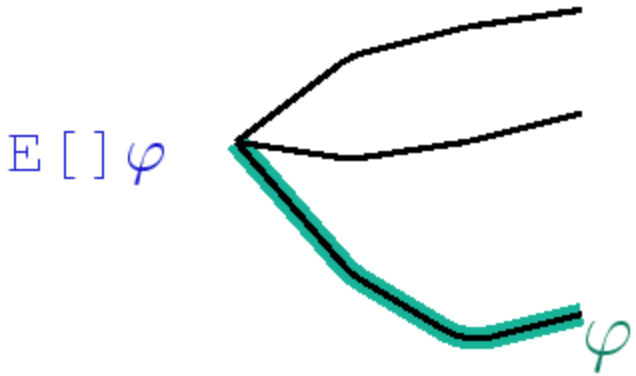
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$



Logical Specifications

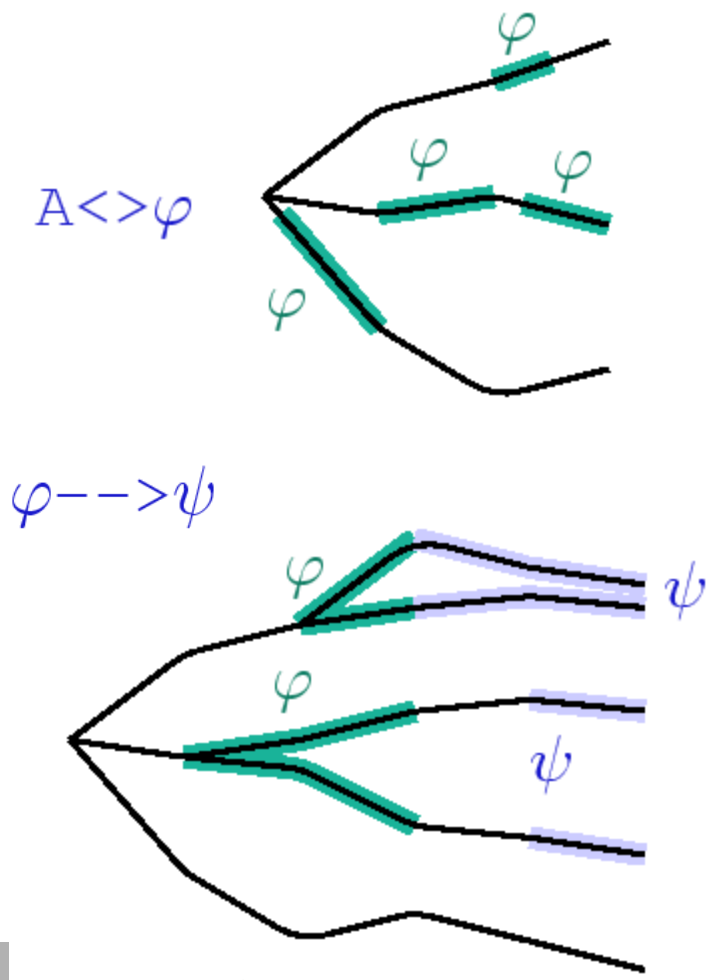
- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: P → *Q*
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$





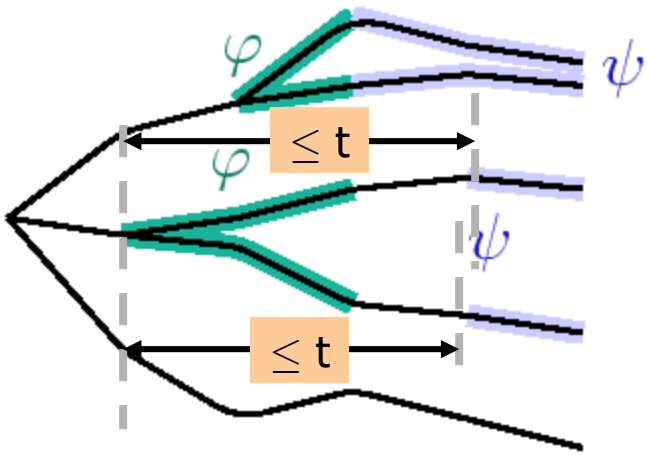
Logical Specifications

- 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: $P \rightarrow_{\leq t} Q$

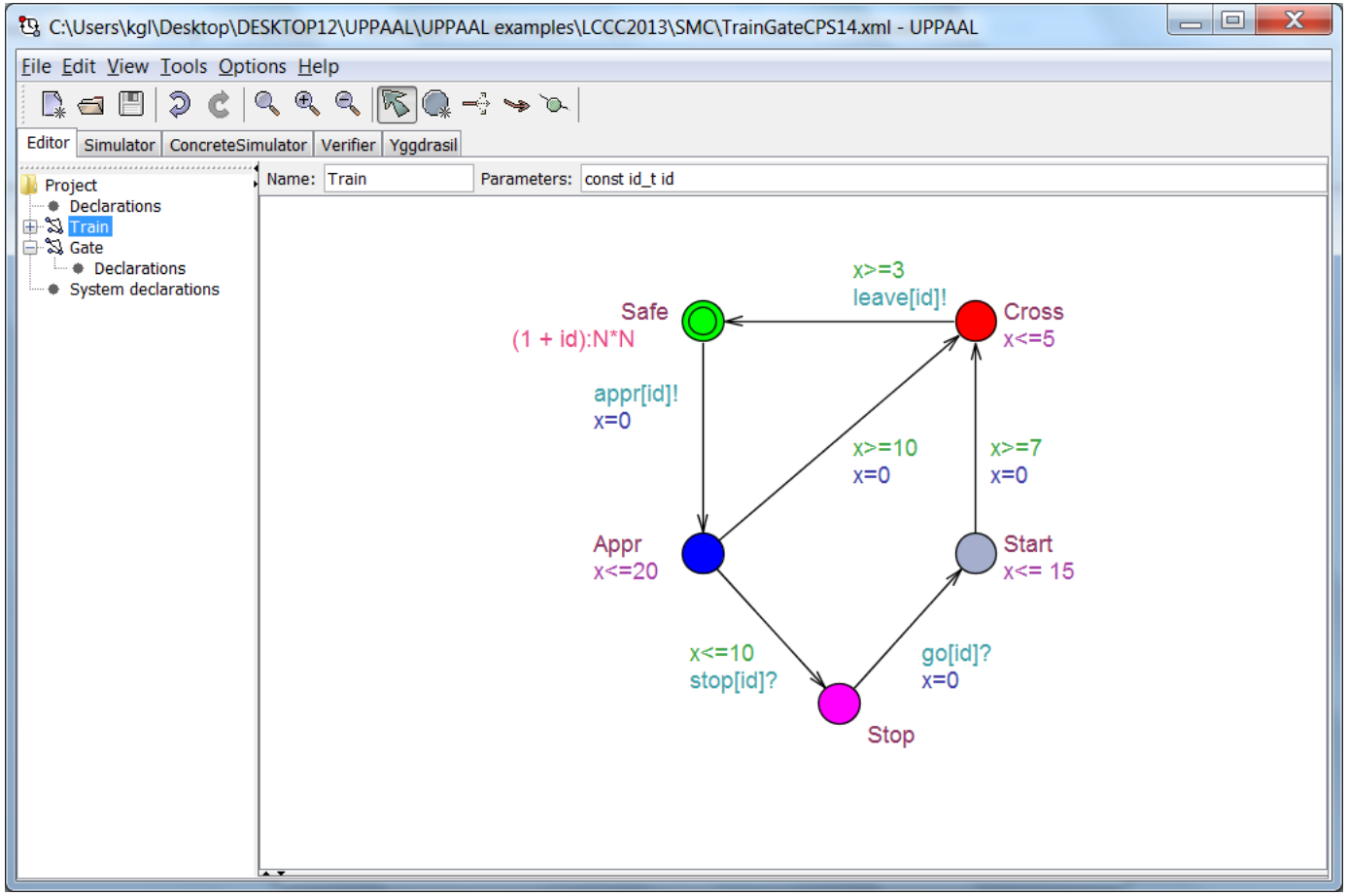


Logical Specifications

- 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: $P \rightarrow_{\leq t} Q$



Demo 2


C:\Users\kgl\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS14.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier Yggdrasil
Project
Declarations
Train
Gate
Declarations
System declarations
Name: Train
Parameters: const id_t id
x>=3
leave[id]!
Safe
(1 + id):N*N
Cross
x<=5
appr[id]!
x=0
x>=10
x=0
x>=7
x=0
Appr
x<=20
Start
x<= 15
x<=10
stop[id]?
go[id]?
x=0
Stop

Editor

C:\Users\kgl\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS1...

File Edit View Tools Options Help

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

File Edit View Tools Options Help

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

File Edit View Tools Options Help

Editor | Simulator | ConcreteSimulator | Verifier | Yggdrasil

- Project
  - Declarations
  - Train
  - Gate
    - Declarations
  - System declarations

Name: Gate Parameters:

```
id_t list[N+1];
int[0,N] len;

// Put an element at the end of the queue
void enqueue(id_t element)
{
        list[len++] = element;
}

// Remove the front element of the queue
void dequeue()
{
        int i = 0;
        len -= 1;
        while (i < len)
        {
                list[i] = list[i + 1];
                i++;
        }
        list[i] = 0;
}

// Returns the front element of the queue
id_t front()
```

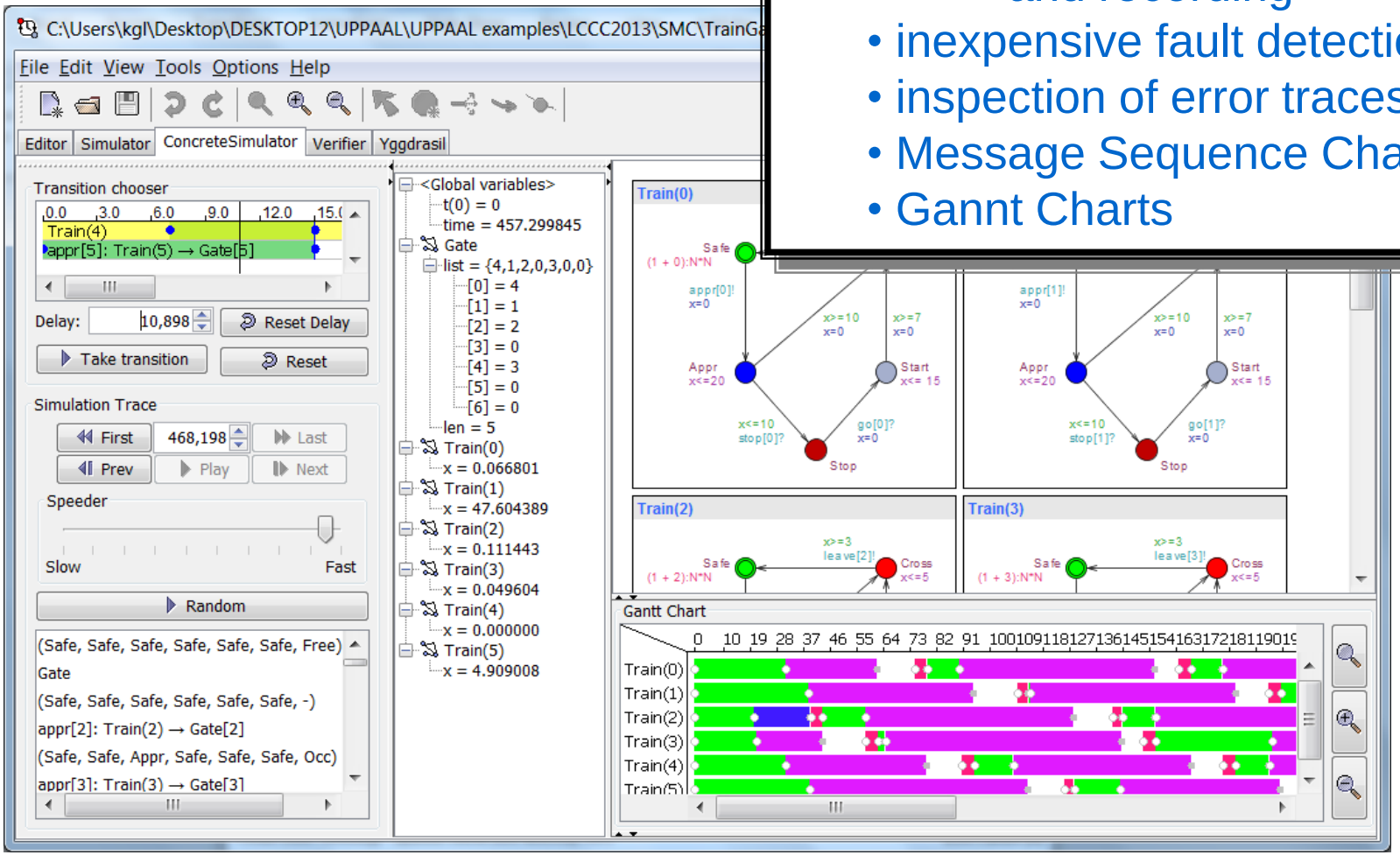
Language

- User defined functions (C-like)
- New types (records, type declarations, meta variables, scalars)
- Partial instantiation of templates
- Select clauses on edges
- Forall and exist quantifiers

Concrete Simulator

Graphical Simulator

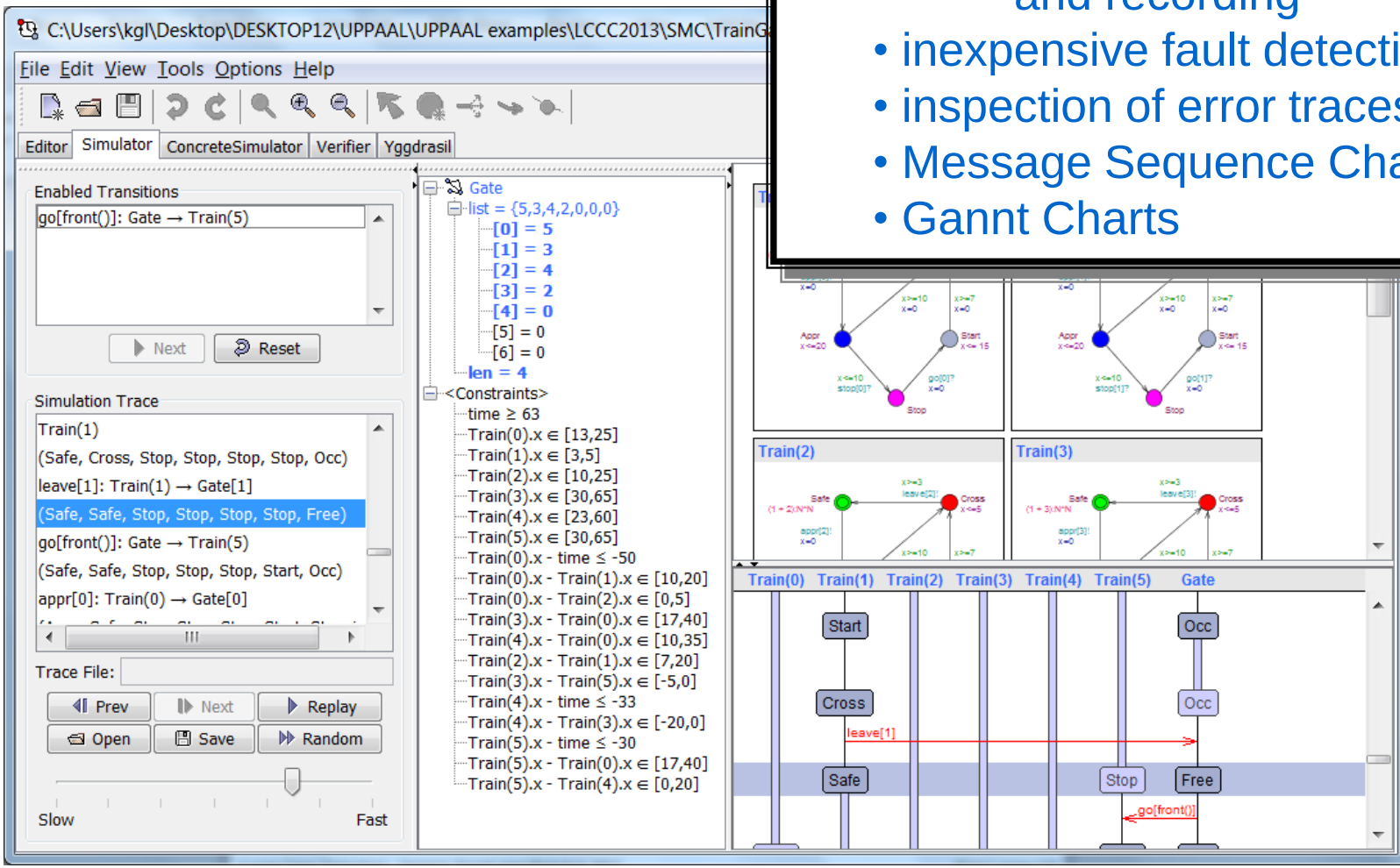
- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gannt Charts

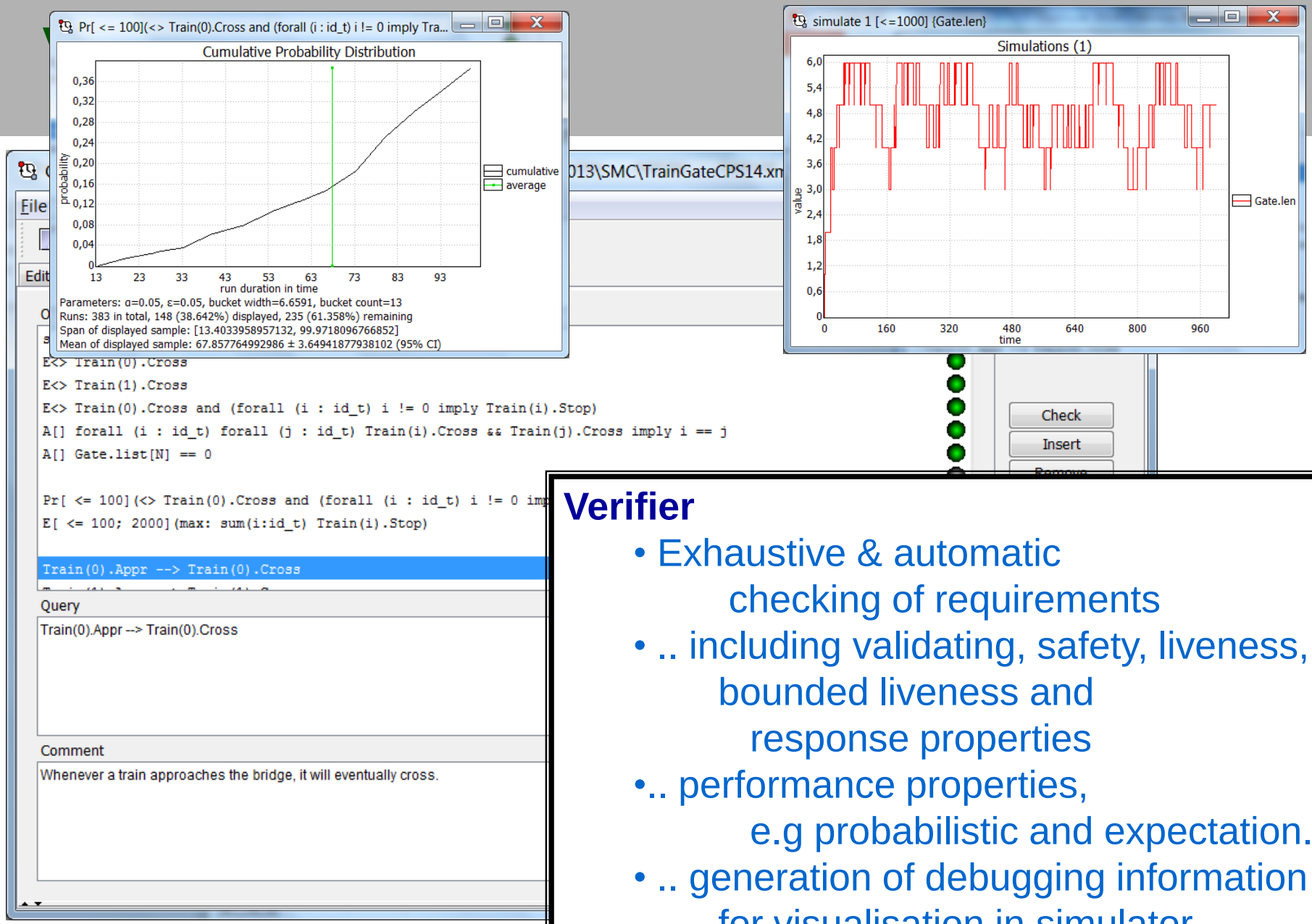


Symbolic Simulator

Graphical Simulator

- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gannt Charts





Verifier

- Exhaustive & automatic checking of requirements
- .. including validating, safety, liveness, bounded liveness and response properties
- .. performance properties, e.g probabilistic and expectation.
- .. generation of debugging information for visualisation in simulator.
- .. plot composer

Applications
(some)

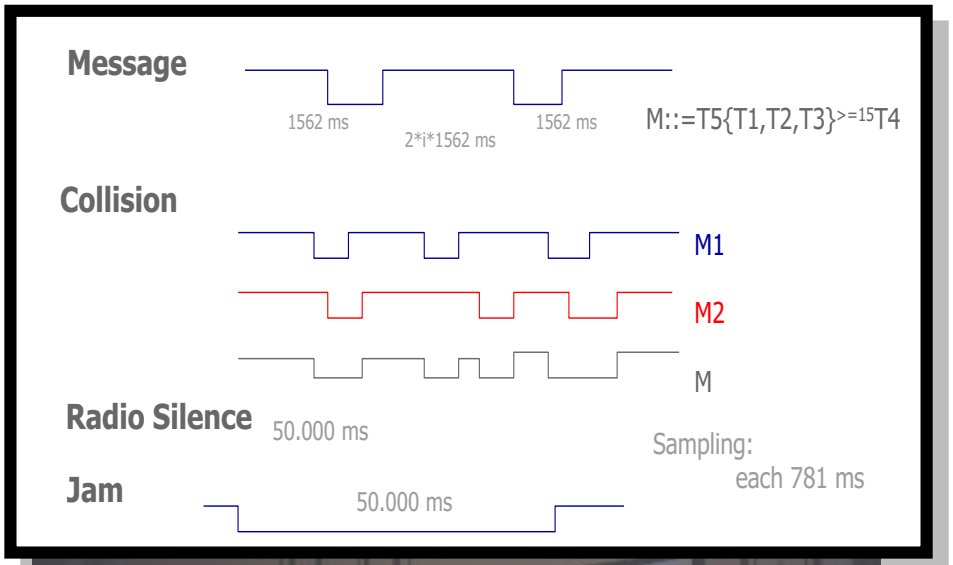
Bang & Olufsen IR-Link

- 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

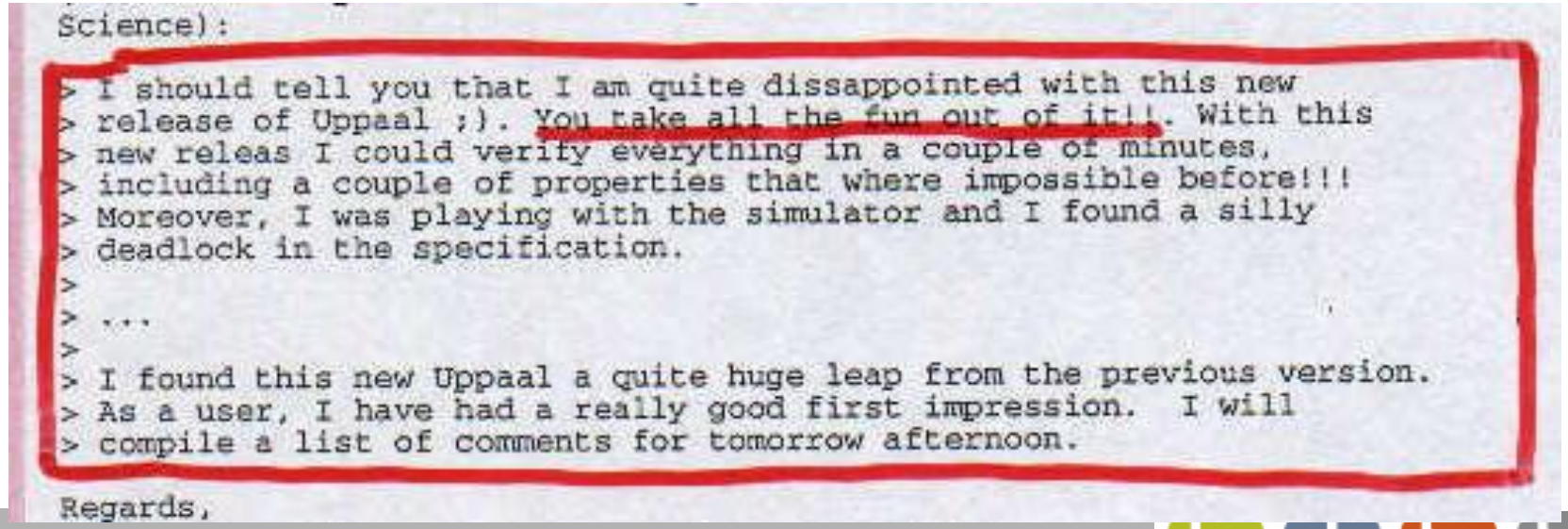
- Bug known to exist for 10 years
- Ill-described:
  2.800 lines of assembler code + 3 flowchart + 1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.



1st RTSS'97 talk, Klaus Havelund

Philips Bounded Retransmission Protocol

Pedro D'Argenio
Joost-Pieter Katoen
Theo Ruys
Jan Tretmans



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.

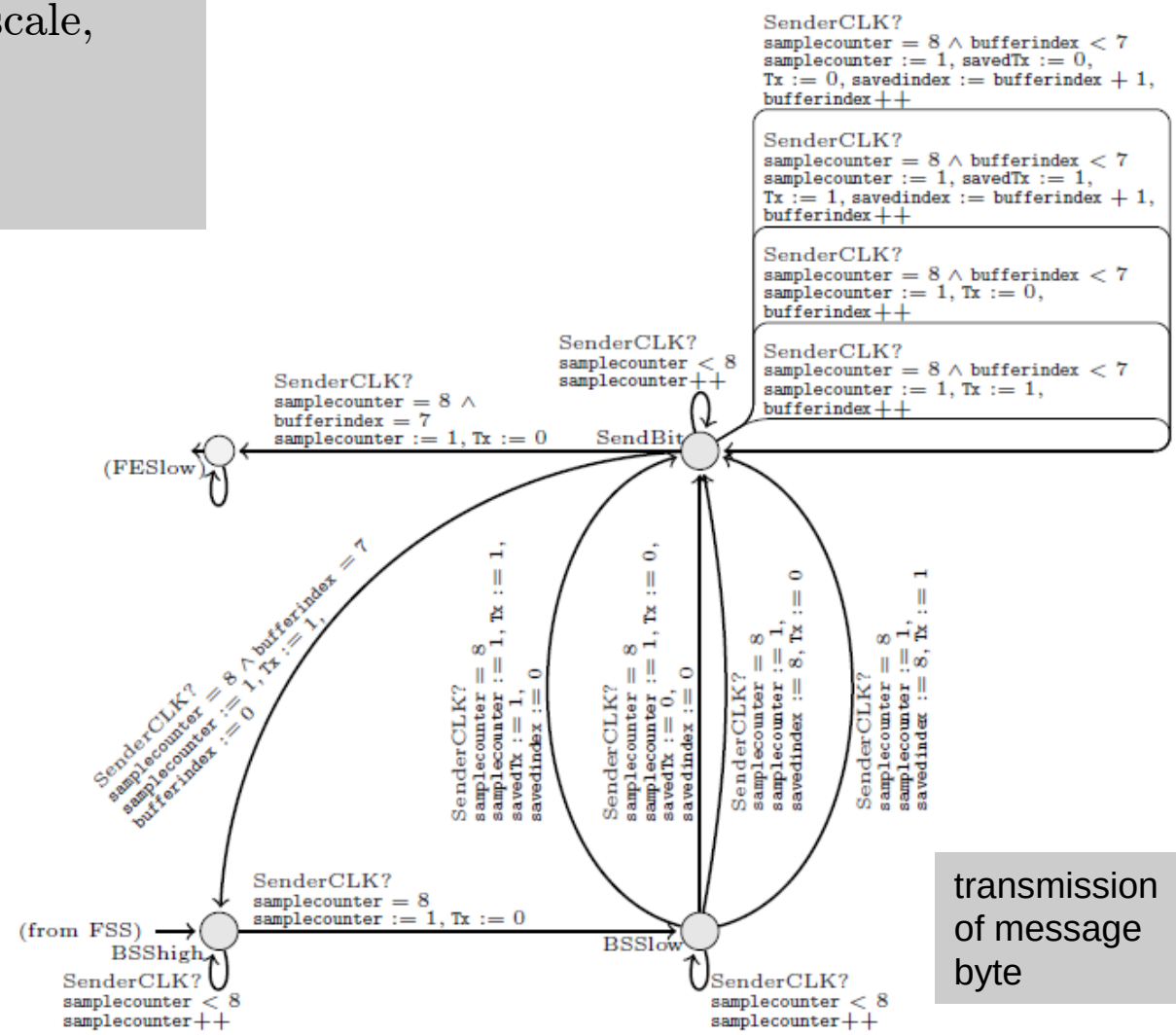
Regards,

FlexRay

[Gerke, Ehlers, Finkbeiner, Peters, 2010]

BMW, Bosch, Daimler, Freescale, General Motors, NXP Semiconductors, and Volkswagen

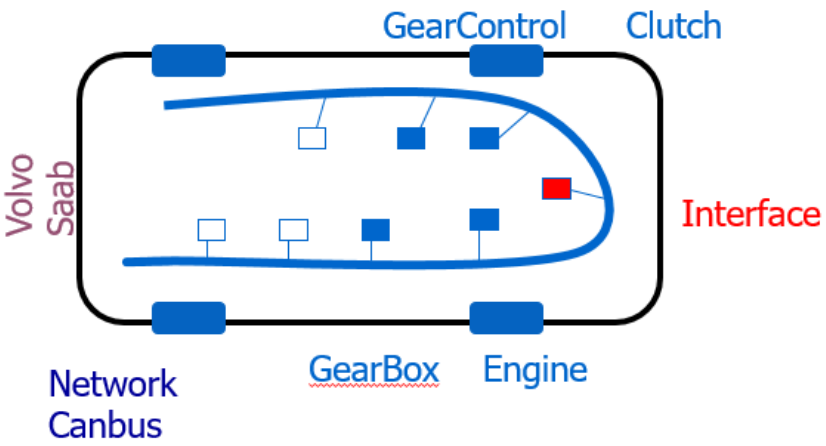
Fault-tolerance
Timed hardware model
Parameterized error models
(glitches, jitter)
Voting & bit-clock alignment



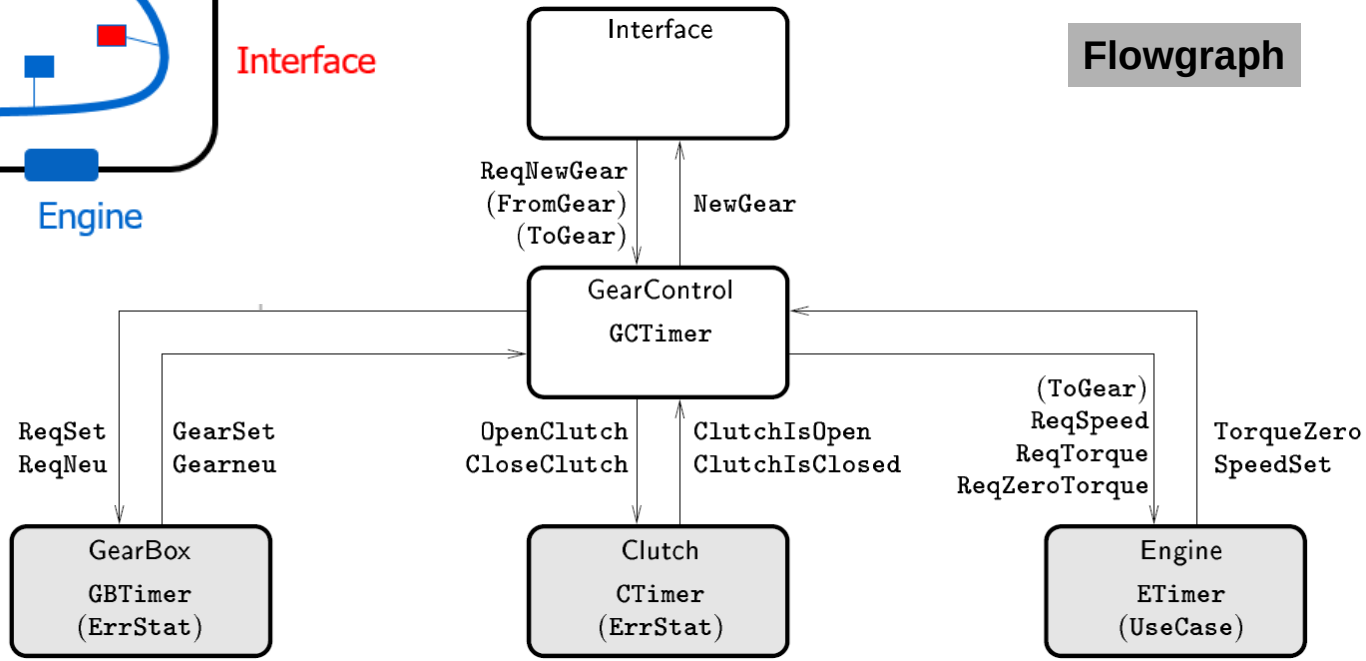
transmission of message byte

Gear Controller

with MECEL AB



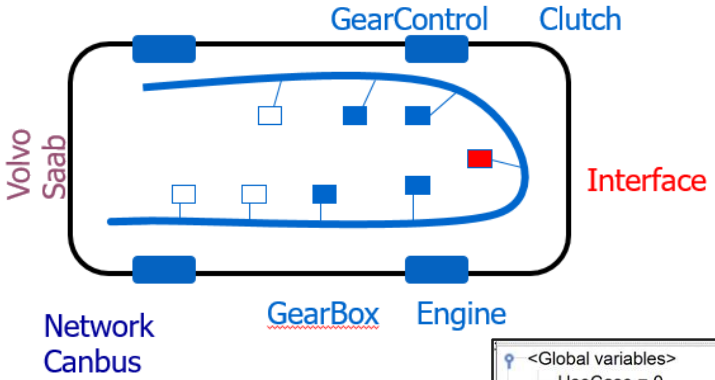
Flowgraph

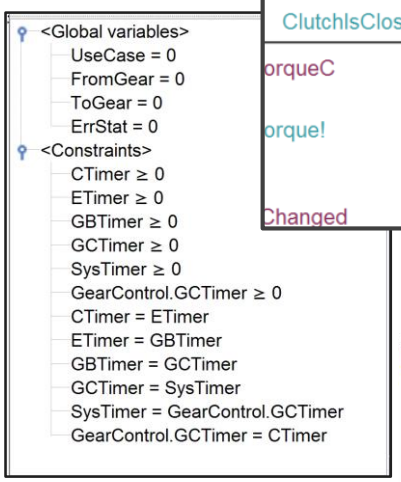


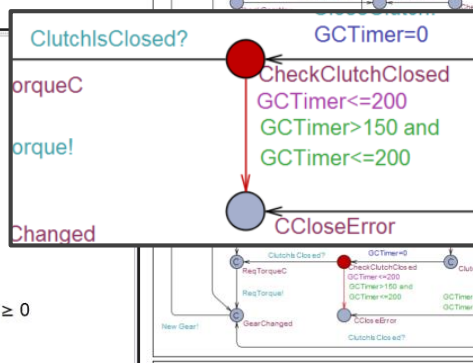
Magnus Lindahl
Paul Pettersson
Wang Yi
2001

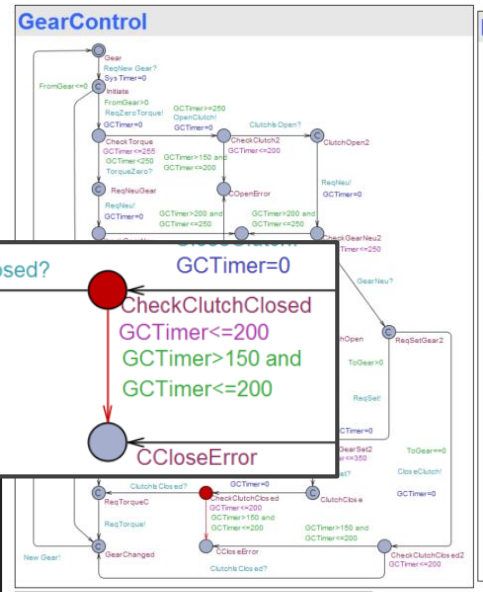
Gear Controller

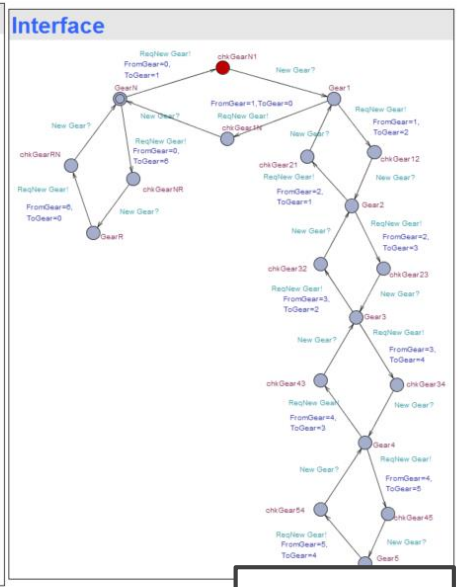
with MECEL AB

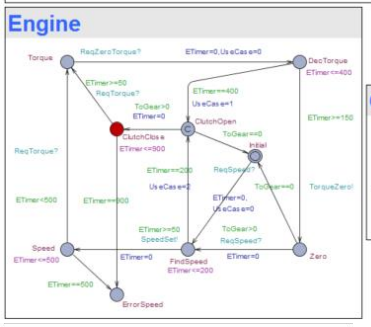


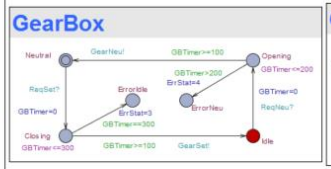


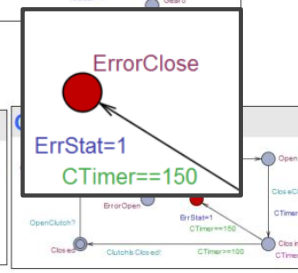










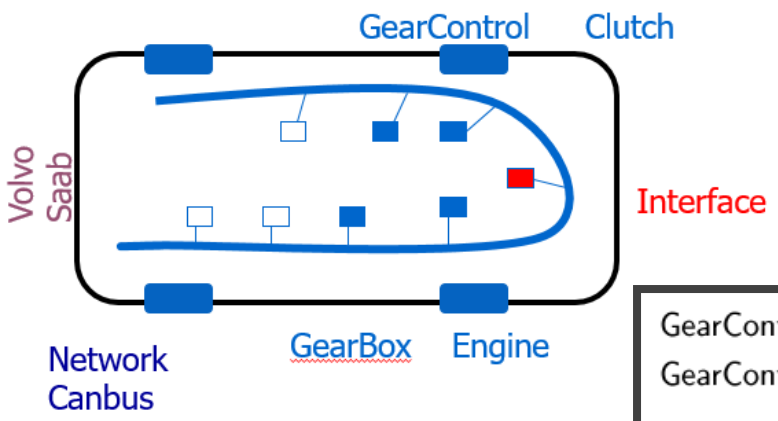


Magnus Lindahl
Paul Pettersson
Wang Yi
2001

Timed Automata Models

Gear Controller

with MECEL AB



Magnus Lindahl
Paul Pettersson
Wang Yi
2001

Requirements

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

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

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

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

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

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

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

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

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

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

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

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

UPPAAL Model Checking – Demo

engine-classic.xta - UPPAAL

File Edit View Tools Options Help

Editor Simulator ConcreteSimulator Verifier

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...
```

Check
Insert
Remove
Comments

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

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 )
```

Check
Insert
Remove
Comments

Query

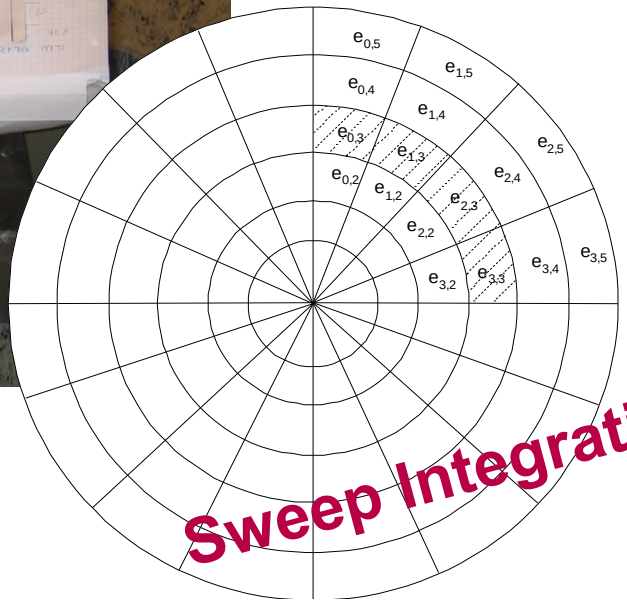
```
A[] ( ( Clutch.ErrorClose and ( GearControl.GCTimer>200 ) ) imply
        GearControl.CCloseError )
```

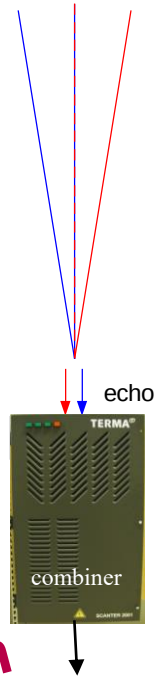
Comment

```
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.
```

TERMA A/S (2004)

Memory Management for Radars





Frequency Diversity

TERMA A/S (2011)

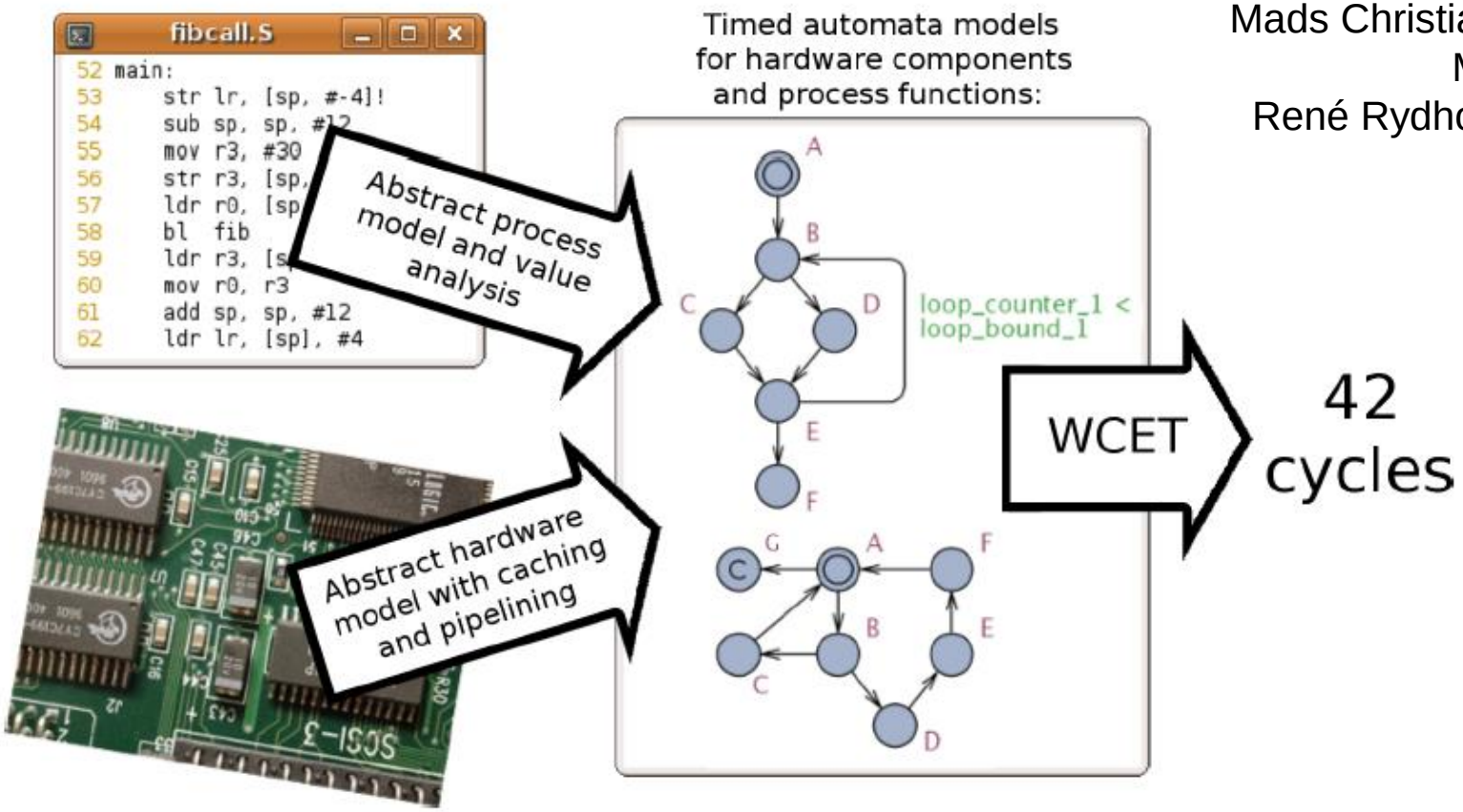
Herschel-Planck Scientific Mission at ESA

Attitude and Orbit Control Software
TERMA A/S Steen Ulrik Palm, Jan Storbank Pedersen, Poul Hougaard

METAMOC

Modular Execution Time Analysis using MOdel Checking

with
Andreas Dalsgaard
Mads Christian Olesen
Martin Toft
René Rydhof Hansen



Controllers in UPPAAL

- 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]
- Terma, Verification of Memory Management for Radar (2001)
- Scheduling Lacquer Production (2005)
- Memory Arbiter Synthesis and Verification for a Radar Memory Interface Card [NJC'05]
- Adapting the UPPAAL Model of a Distributed Lift System, 2007
- Analyzing a χ model of a turntable system using Spin, CADP and Uppaal, 2006
- Designing, Modelling and Verifying a Container Terminal System Using UPPAAL, 2008
- Model-based system analysis using Chi and Uppaal: An industrial case study, 2008
- Climate Controller for Pig Stables, 2008
- Optimal and Robust Controller for Hydralic Pump, 2009

(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

UPPAAL as a Back-End

- Vooduu: verification of object-oriented designs using Uppaal, 2004
- Moby/RT: A Tool for Specification and Verification of Real-Time Systems, 2000
- Formalising the ARTS MPSOC Model in UPPAAL, 2007

- Timed automata translator for Uppaal to PVS
- Component-Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008
- Verification of COMDES-II Systems Using UPPAAL with Model Transformation, 2008
- METAMOC: Modular WCET Analysis Using UPPAAL, 2010.

www.uppaal.org

www.uppaal.org

RELATED TOOLS: TIMES | Stratego | CORA | TRON | TIGA | SMC | COVER | PORT | PRO

UPPAAL

Home

Home | About | Documentation | Download | Examples | Web Help | Bugs

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.

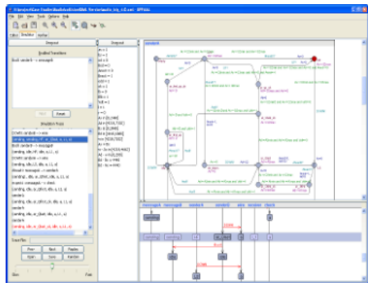


Figure 1: UPPAAL on screen.

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 development snapshots of the tool. The current version is

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 or www.uppaal.com for more information.

To find out more about UPPAAL, read this short introduction. Further information may be found at this web site in the pages About, Documentation, Download, and Examples.

Mailing Lists

UPPAAL has an open discussion forum group at Yahoo!Groups intended for users of the tool. To join or post to the forum, please refer to the information at the discussion

Excercises

http://people.cs.aau.dk/~kgl/GRAZ17/

Exercise 1 (Brick Sorter)
Excercise 19 (Train Crossing)
Exercise 2 (Coffee Machine)
Exercise 28 (Jobshop Scheduling)