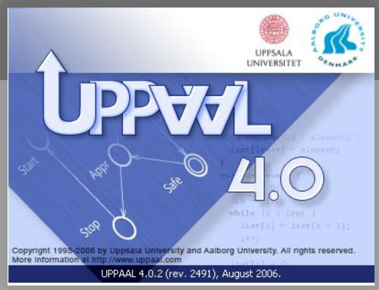
From Timed Automata to Stochastic Hybrid Games

Model Checking, Performance Evaluation and Synthesis



Kim G. Larsen

Aalborg University, DENMARK

Aalborg

Jomfru Ane Gade

Aalborg University lea
publ

CISS – Center For Embedded Software Systems

Regional ICT Center (2003-)

- 3 research groups
  - Computer Science
  - Control Theory
  - HW/SW- codesign

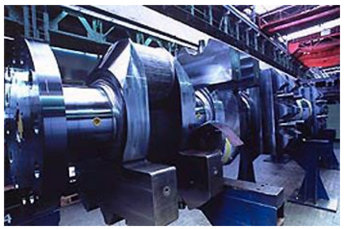
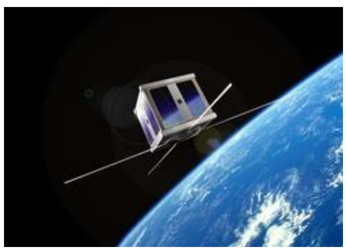
- 20 Employed
- 25 Associated
- 20 PhD Students
- 50 Industrial projects
- 10 Elite-students
- 65 MDKK

- ARTIST Design
- ARTEMIS

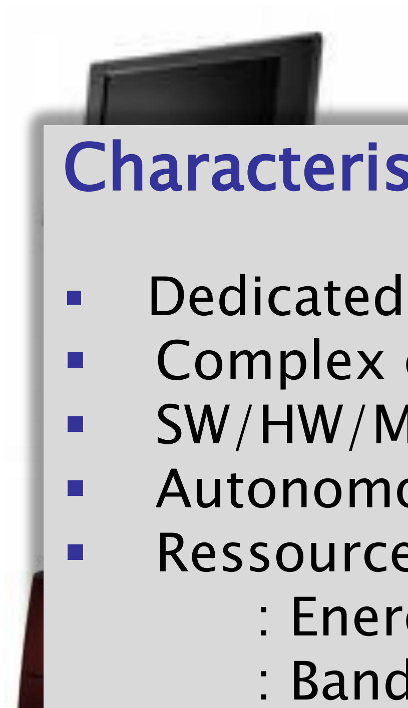
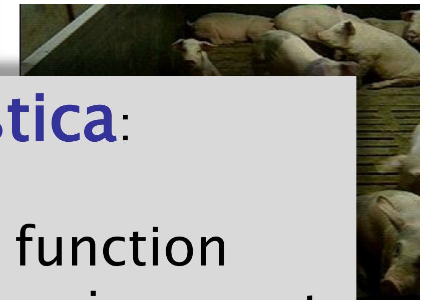
Information Society Technologies

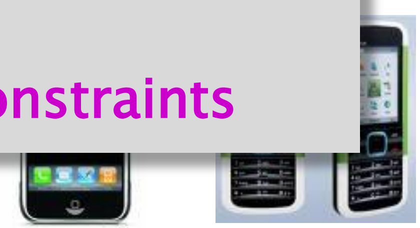
ARTEMIS

ES are Pervasive





Characteristica:

- Dedicated function
- Complex environment
- SW/HW/Mechanics
- Autonomous
- Ressource constrained
  - : Energy
  - : Bandwidth
  - : Memory
  - : …
- Timing constraints

ES are often Safety Critical

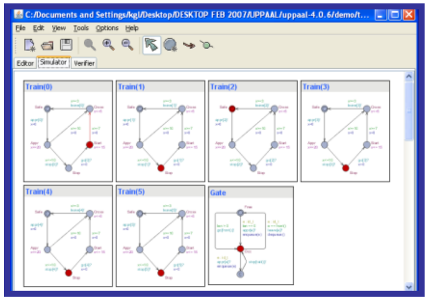
300 horse power
100 processors

How to achieve ES that are:

- correct
- predicable
- dependable
- fault tolerant
- ressource minial
- cheap

Model-Based Development

QUANTITATIVE Model Checking

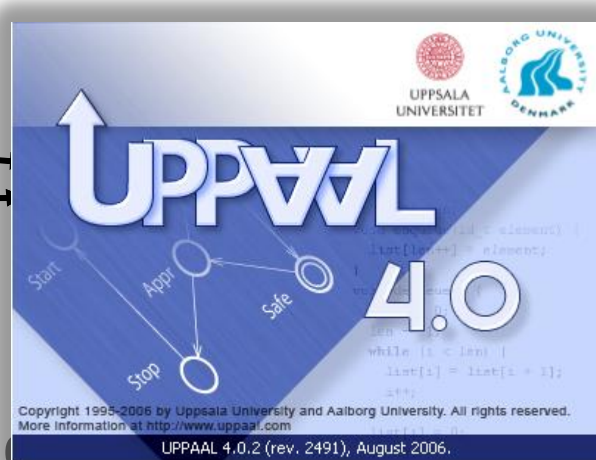


System Description

Time

Cost

Probability



No!
Debugging Information

Yes
Prototypes
Executable Code
Test sequences

Requirement

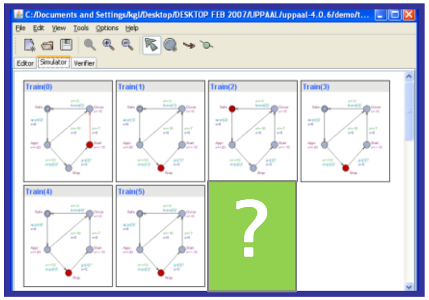
$A\Box(\ \mathrm{req} \Rightarrow A\Diamond\ \mathrm{grant})$

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

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

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

Synthesis

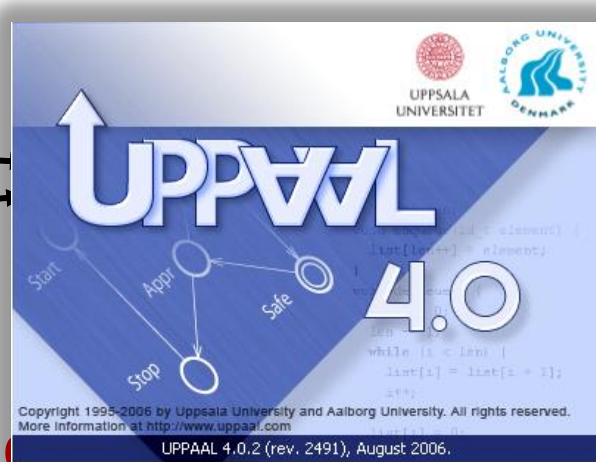


System Description

Time

Cost

Probability



No!
Debugging Information

Yes
Control Strategy

Requirement

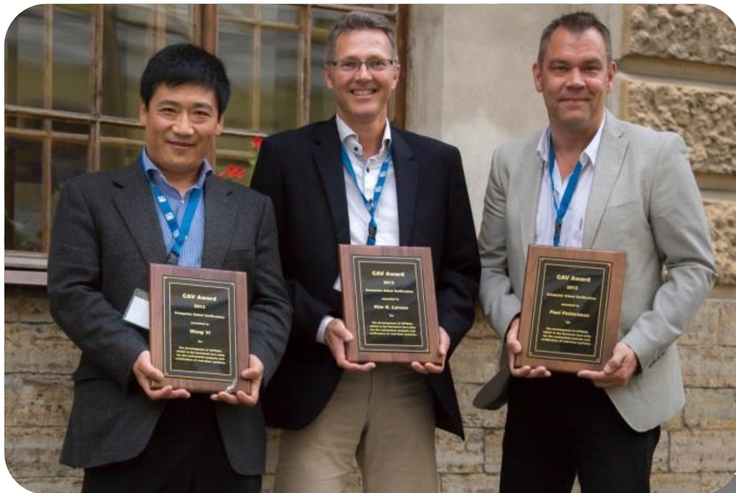
$A\Box(\text{req} \Rightarrow A\Diamond\, \text{grant})$

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

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

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

Origin of UPPAAL



TAU
CCS & Modal Transition Systems
Refinements
Modal Mu-Calculus
Explicit State Representation
Prolog

1995

UPPAAL
Timed Automata
TCTL
Zones
C++ & Java

EPSILON
TCCS
Timed Refinements
Timed Mu-Calculus
Regions
Prolog

2007

UP4ALL

2013

CAV Award

Contributors

@UPPsala

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

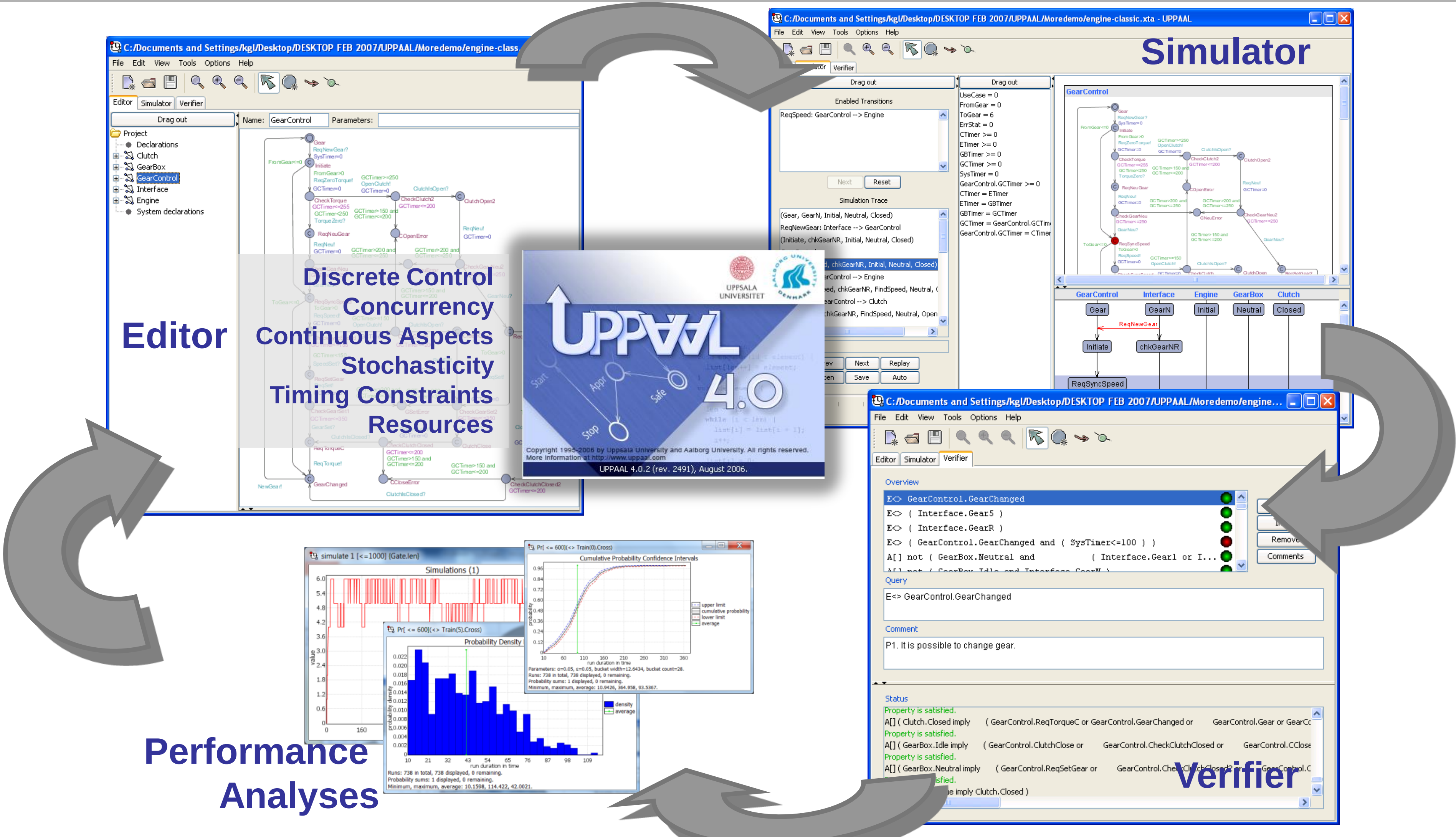
@AALborg

- 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...

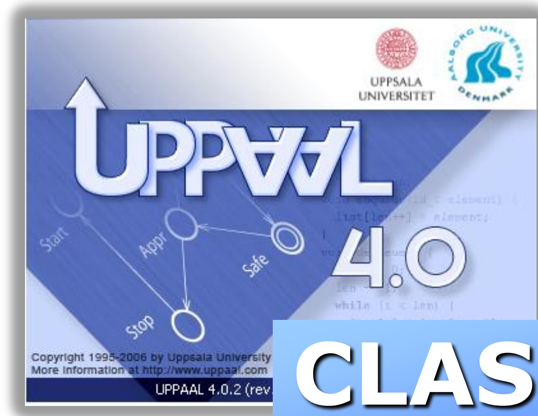
UPPAAL Model Checker



Overview

- Timed Automata & UPPAAL
- Symbolic Verification &
  UPPAAL Engine, Options
- Priced Timed Automata
  and Timed Games
- Stochastic Timed Automata
  Statistical Model Checking
  Optimal Synthesis

(Lecture + Exercise)^4



CLASSIC
CORA
TIGA
SMC
STRATEGO
ECDAR
TRON

www.cs.aau.dk/~kgl/SSFT205

From Timed Automata to Stochastic Hybrid Games
Model Checking, Performance Evaluation and Synthesis
using UPPAAL

Kim Guldstrand Larsen
CISS, Aalborg University, DENMARK

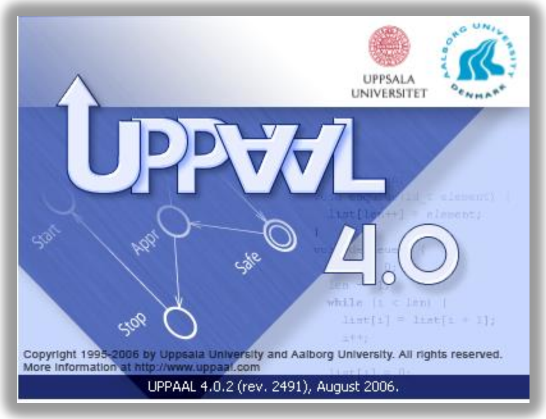
Fifth Summer School on Formal Techniques
May 17 - May 22, 2015
Menlo College, Atherton, CA

Slides (preliminary – will be updated)

1. Timed Automata and UPPAAL
2. Symbolic Verification and UPPAAL Engine
3. Priced Timed Automata and Timed Games
4. Stochastic Timed Automata and Statistical Model Checking

Material available here!

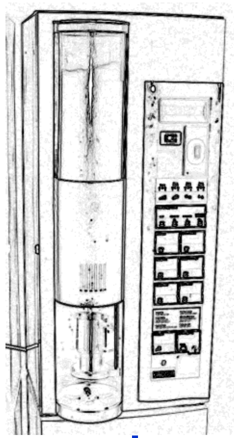
Exercises available here!



www.uppaal.org

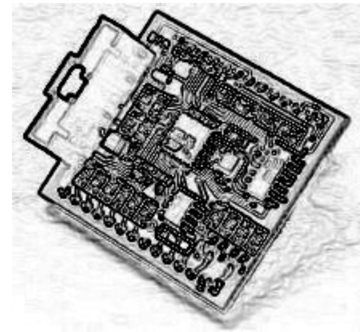
Timed Automata

Real Time Systems



sensors

actuators



Plant

Continuous

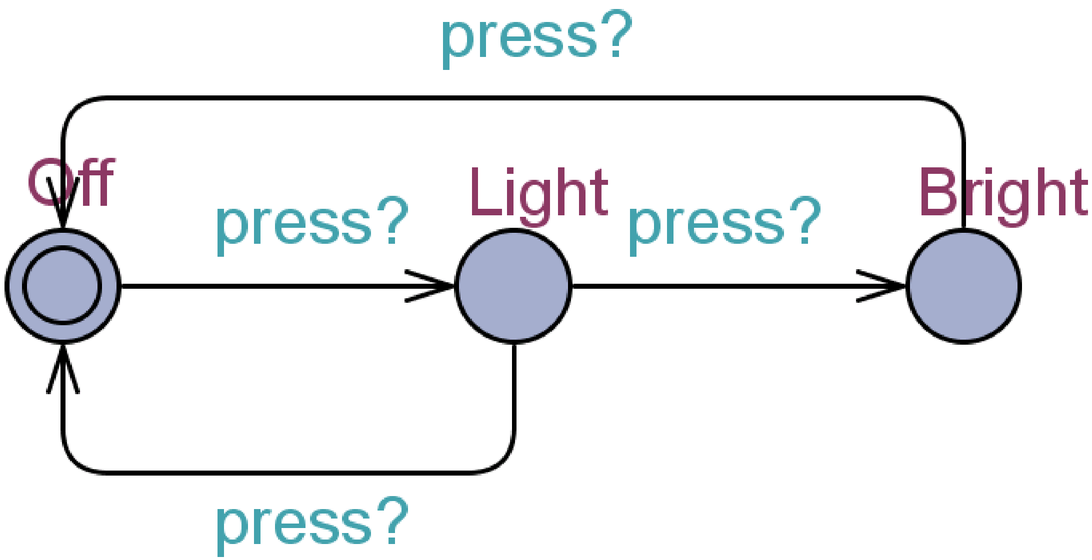
Controller Program

Discrete

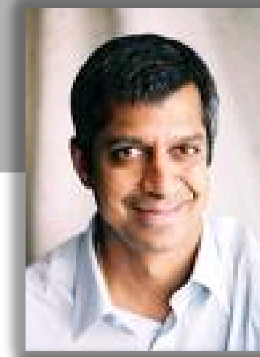
Eg.:
- Realtime Protocols
- Pump Control
- Air Bags
- Robots
- Cruise Control
- ABS
- CD Players
- Production Lines

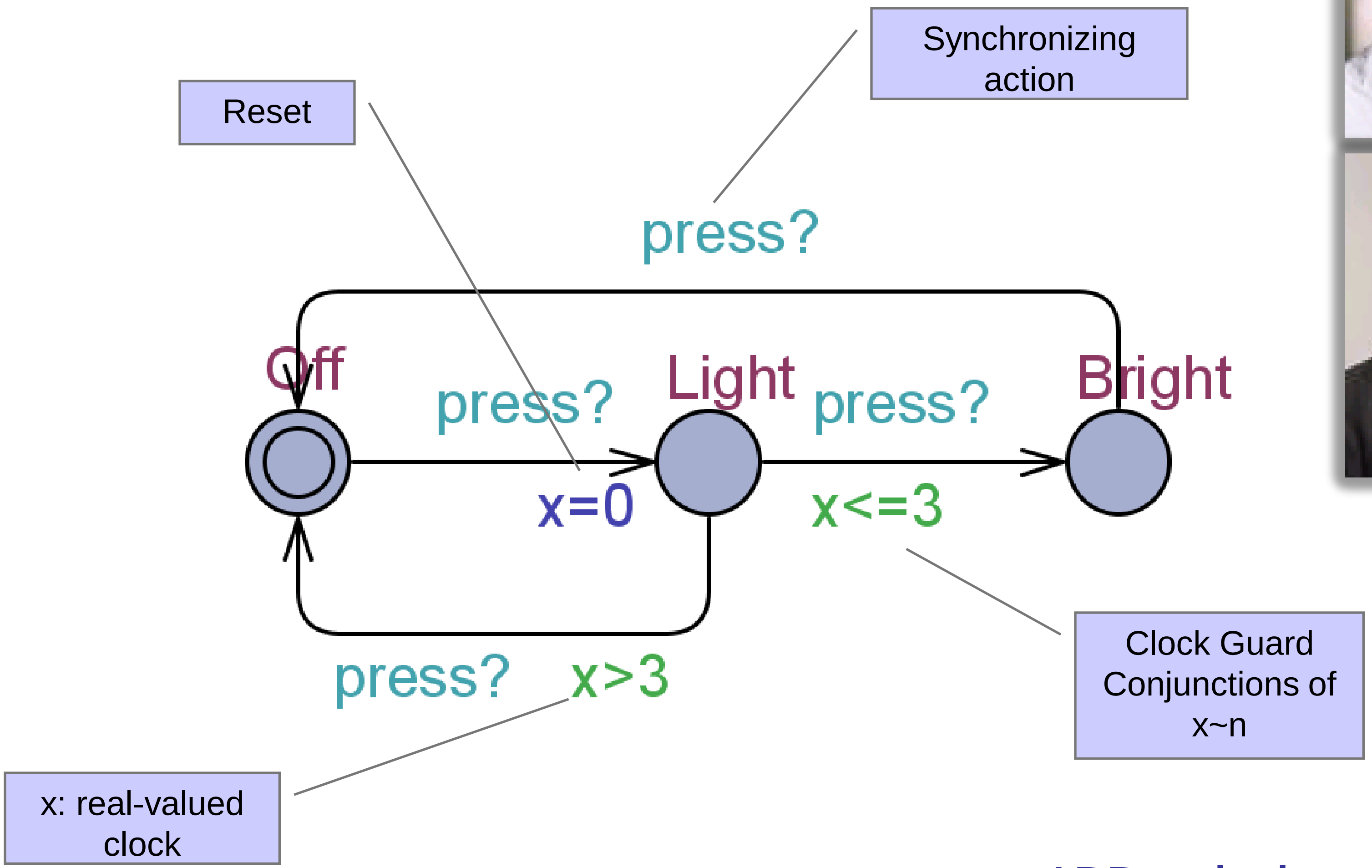
> **Real Time System**
> A system where correctness not only depends on the logical order of events but also on their timing!!

A Dumb Light Controller



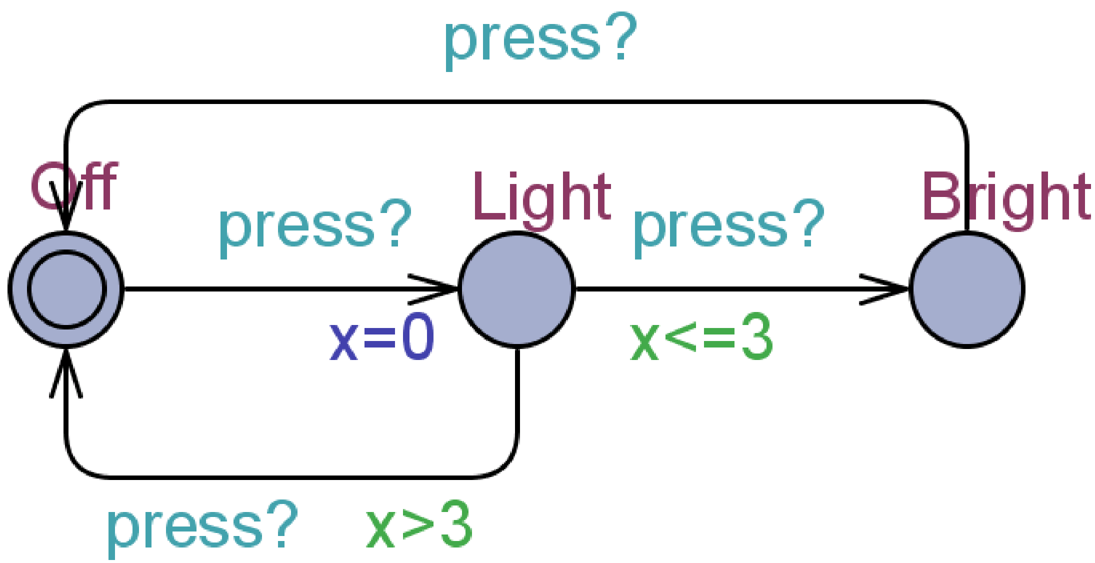
Timed Automata [Alur & Dill'89]





ADD a clock x

A Timed Automata (Semantics)

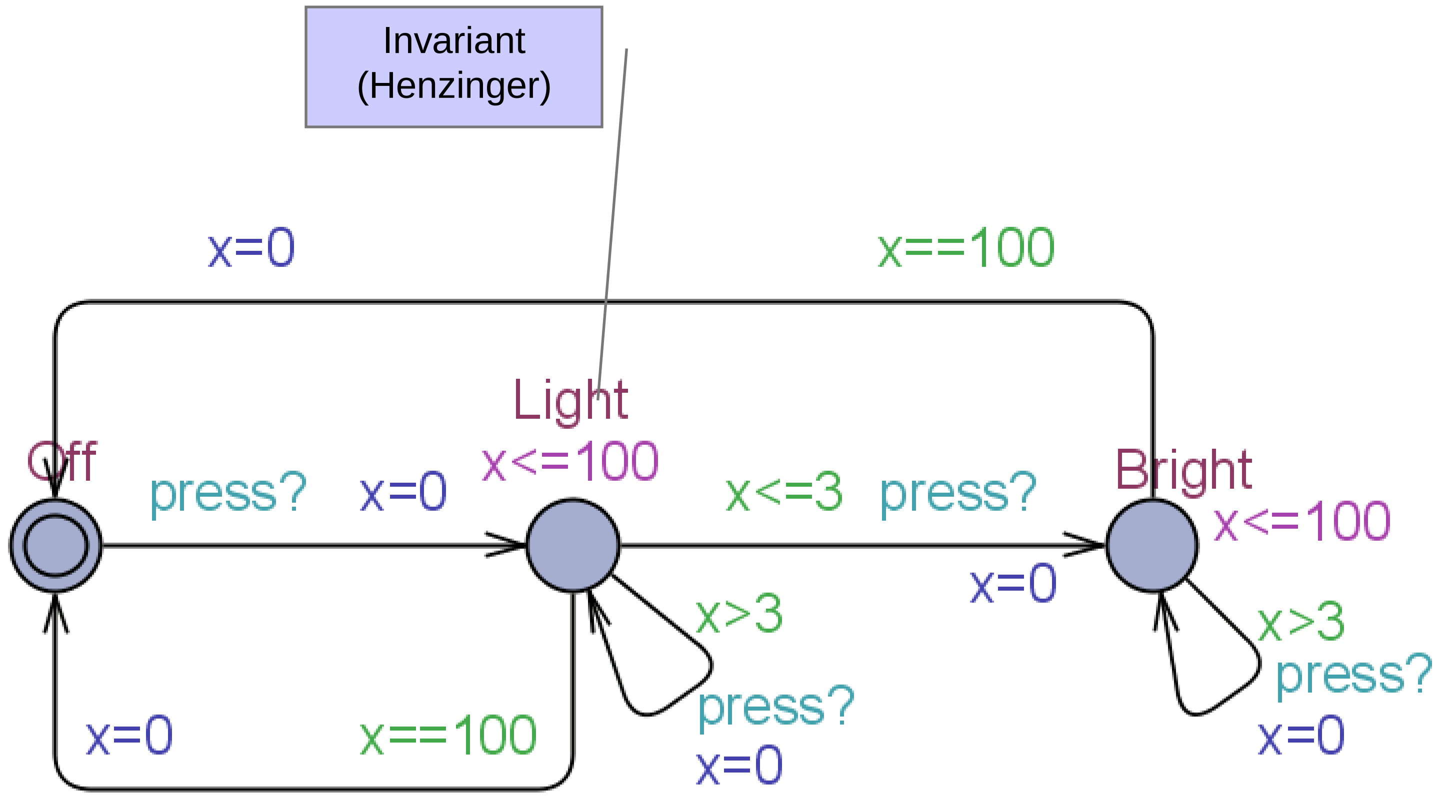


States:
(location , x=v) where v∈**R**

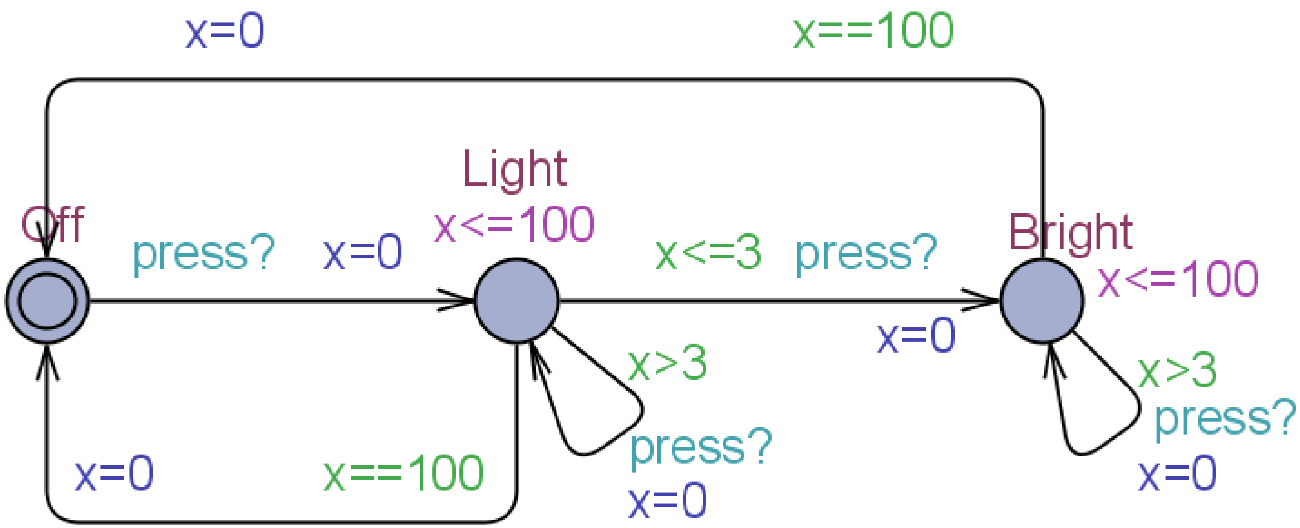
Transitions:

	(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



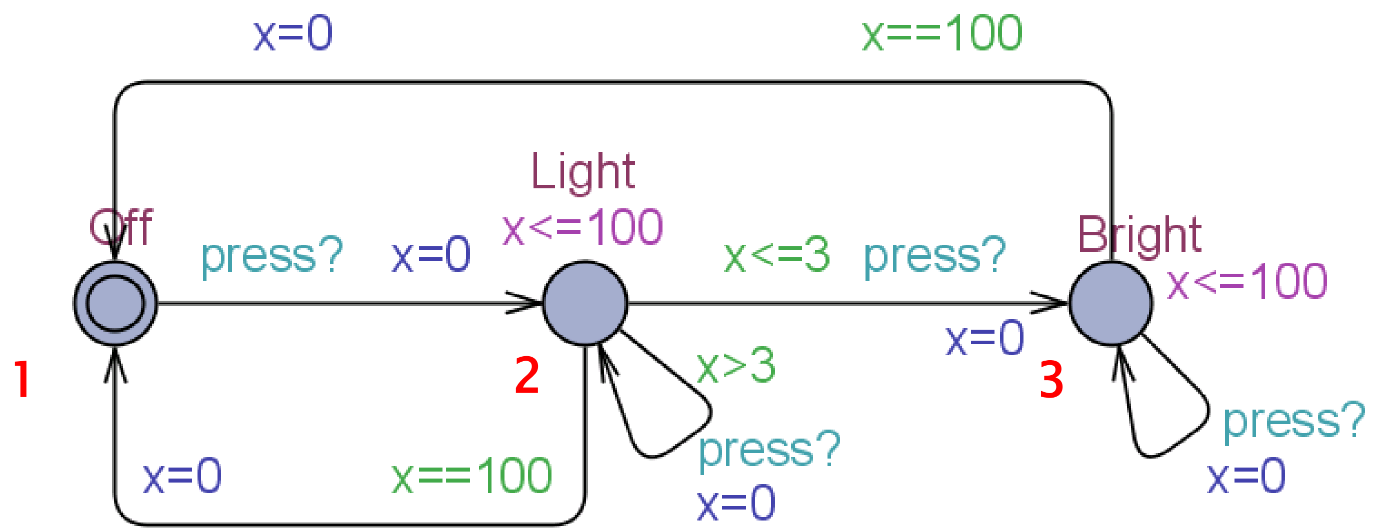
Transitions:

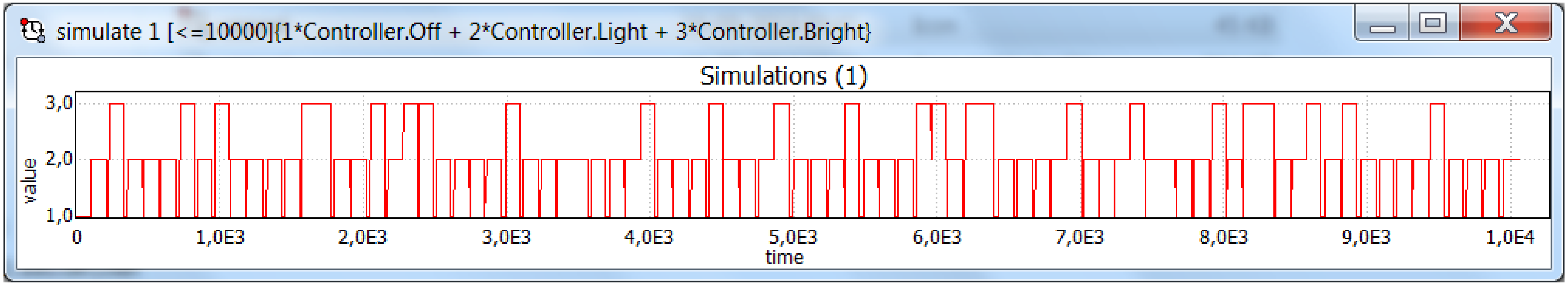
	(Off , x=0)
delay 4.32	→ (Off , x=4.32)
press?	→ (Light , x=0)
delay 4.51	→ (Light , x=4.51)
press?	→ (Light , x=0)
delay 100	→ (Light , x=100)
τ	→ (Off , x=0)

Note:
(Light , x=0) delay 103 → **X**

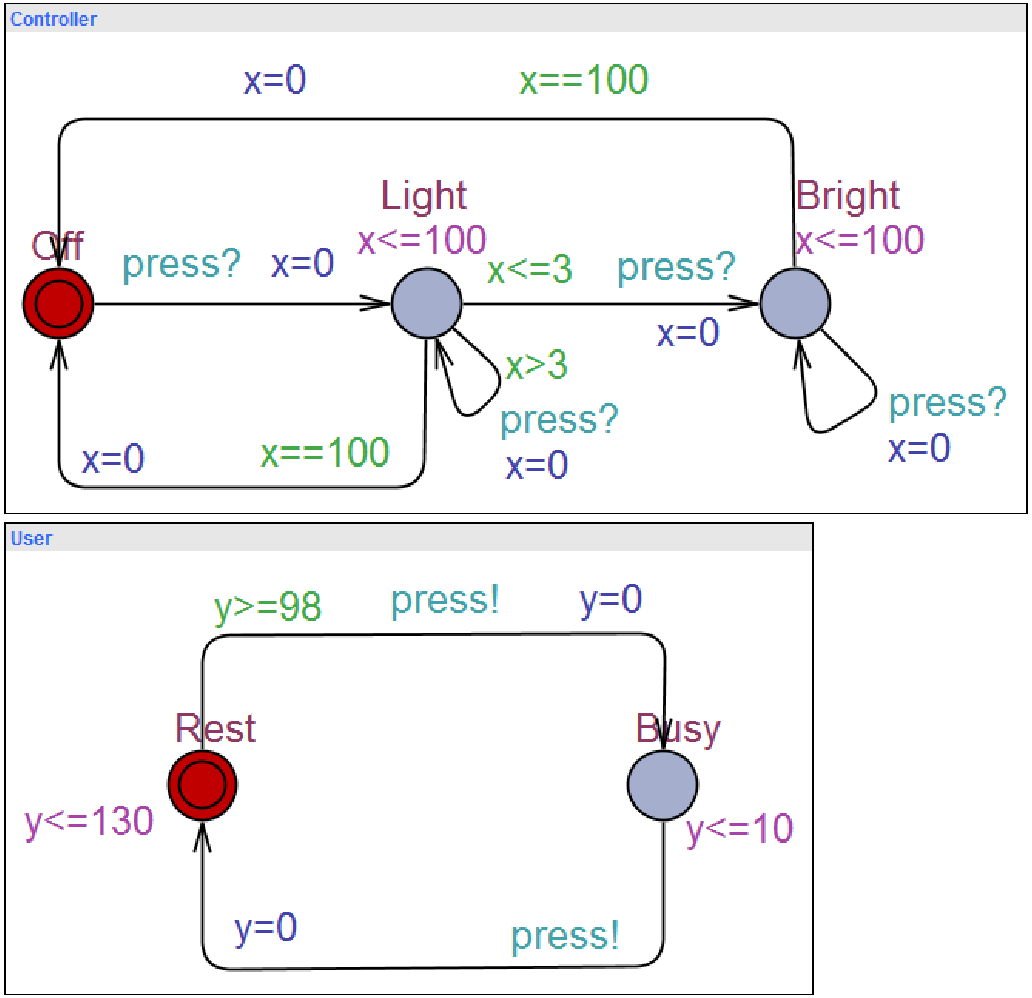
Invariants ensures progress

Intellingent 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: $x \leq 3 \ \wedge \ y > 0 \ \wedge \ y - x = 2$

Clock Valuation – 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 Valuation – Evaluation

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

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

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
- $I : L \rightarrow \mathcal{B}(C)$ 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

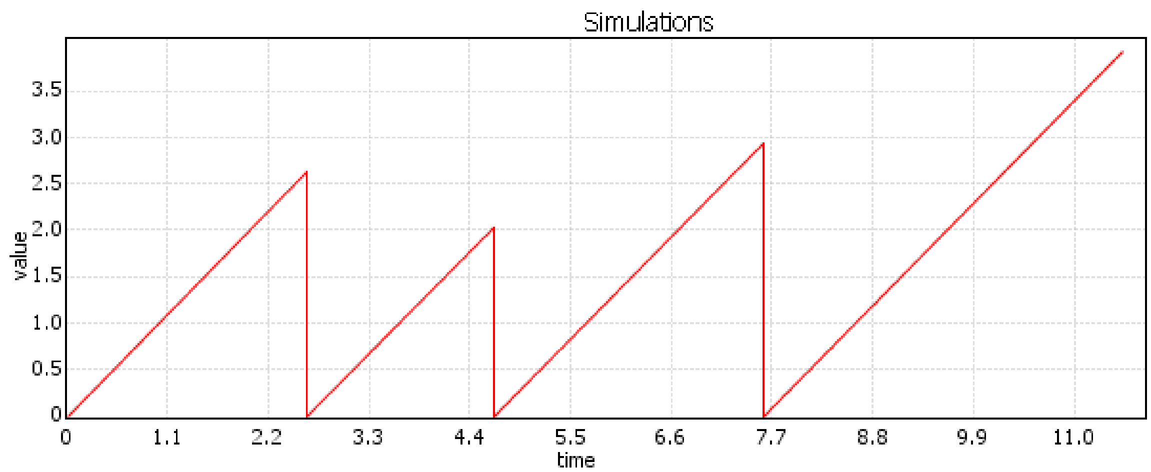
$T(A) = (Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ where

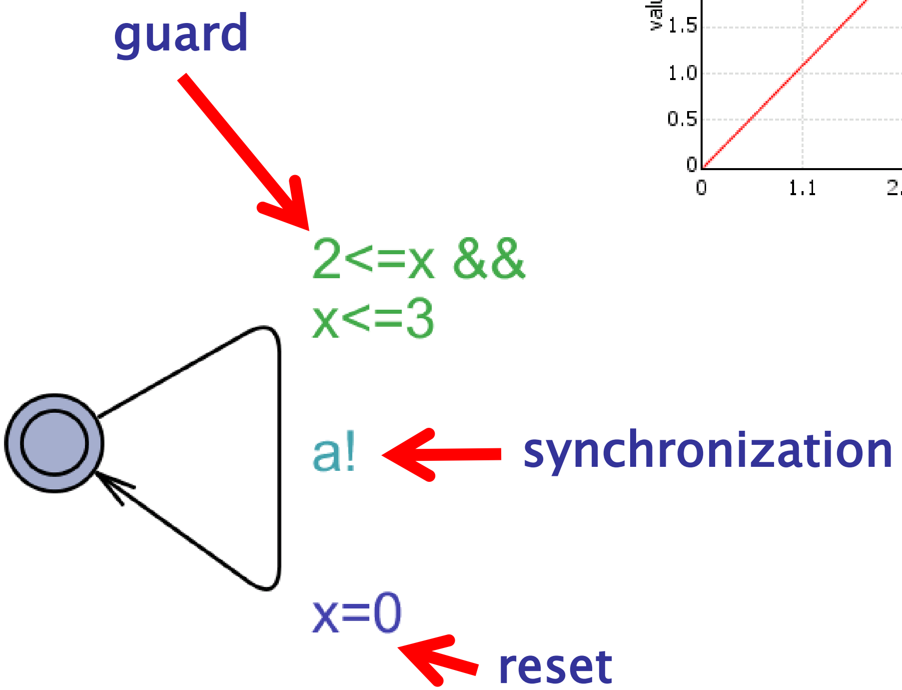
- $Proc = L \times (C \to \mathbb{R}^{\geq 0})$, i.e. states are of the form (ℓ, v) where ℓ is a location and v a valuation
- $Act = N \cup \mathbb{R}^{\geq 0}$
- $\longrightarrow$ is defined as follows:

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

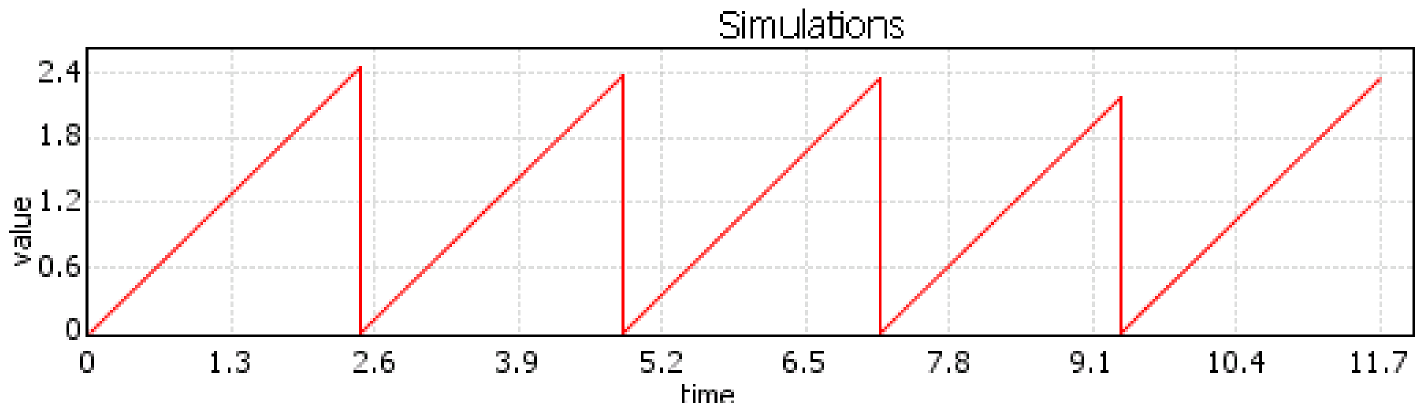
$(\ell, v) \xrightarrow{d} (\ell, v + d)$ for all $d \in \mathbb{R}^{\geq 0}$ s.t. $v \models I(\ell)$ and $v + d \models I(\ell)$

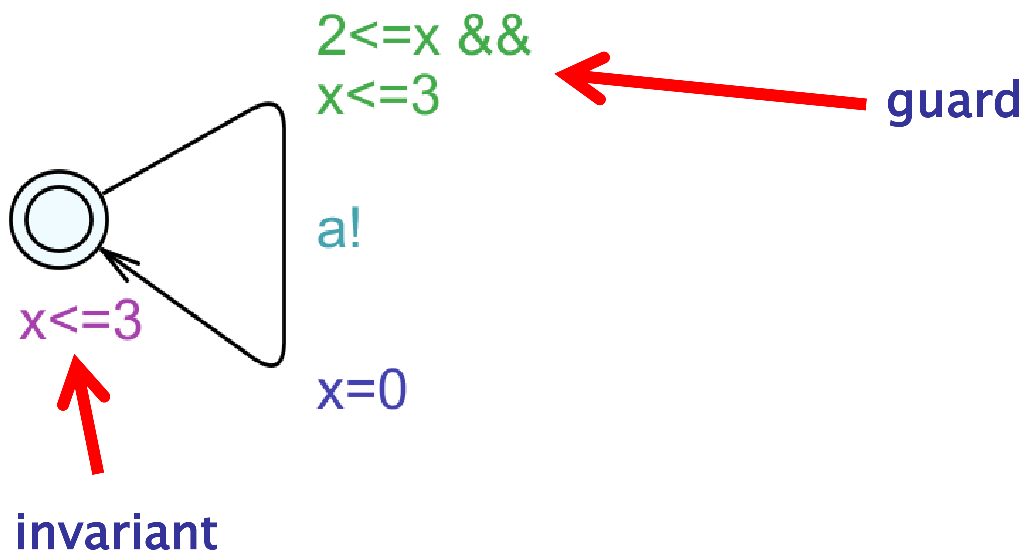
Timed Automata: Example



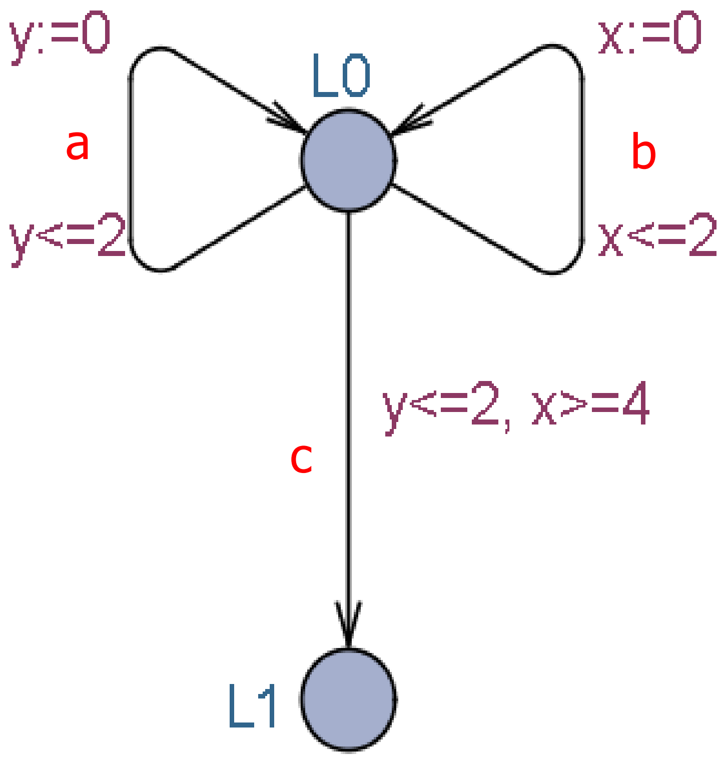


Timed Automata: Example



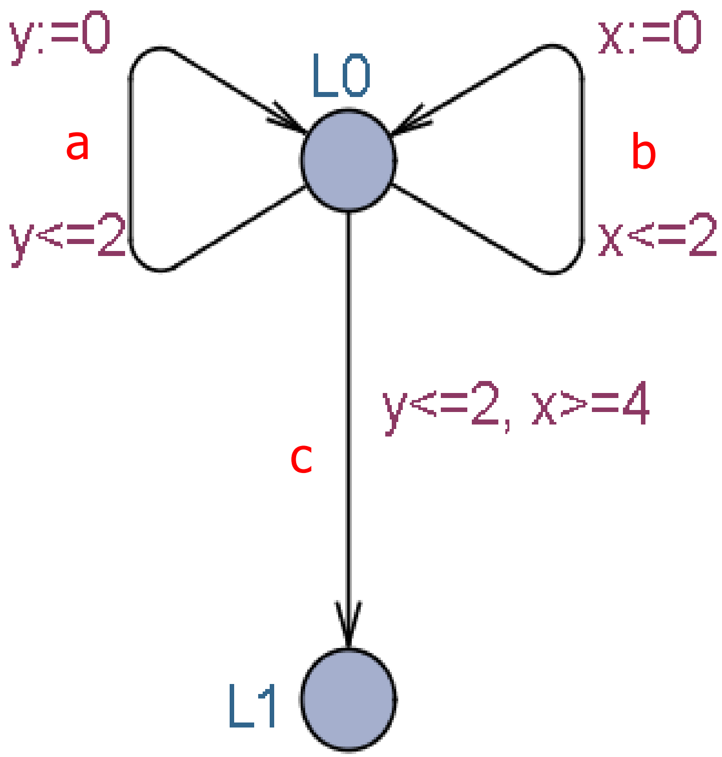


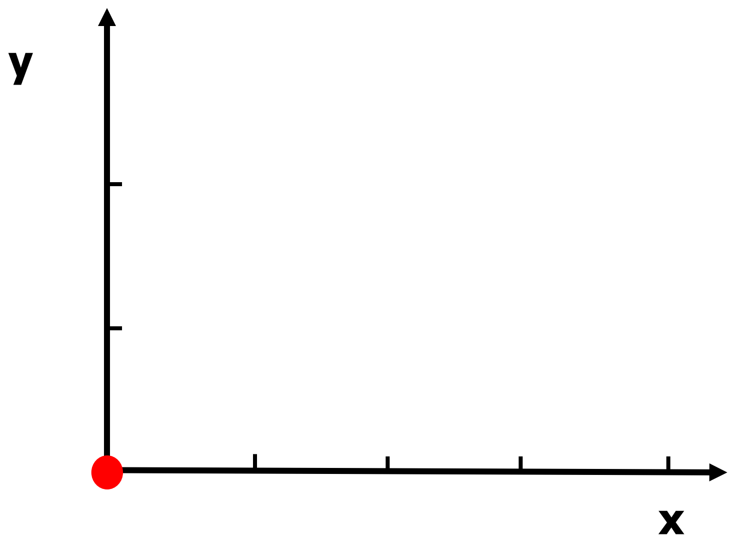
Example



Is L1 reachable ?

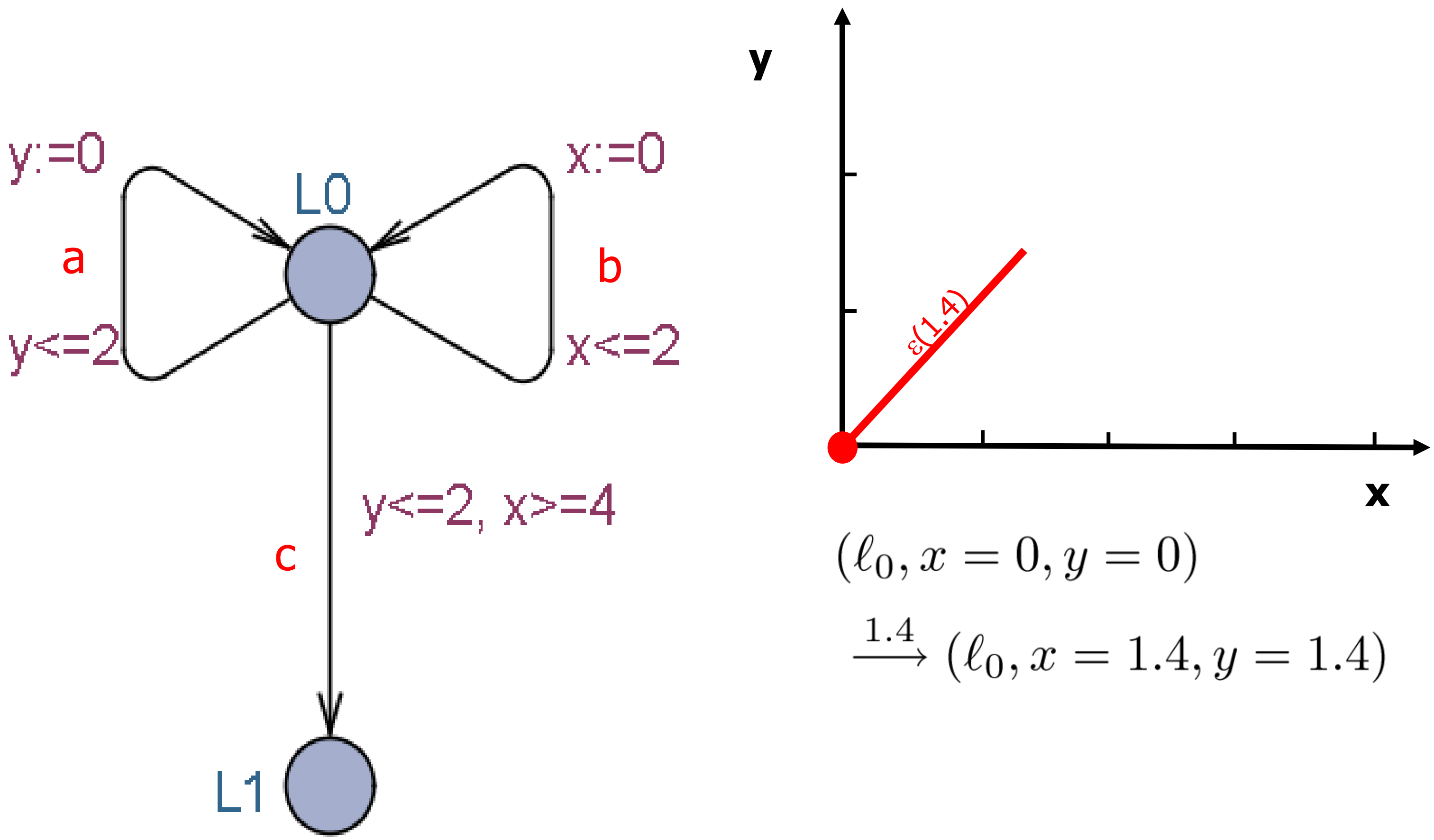
Example





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

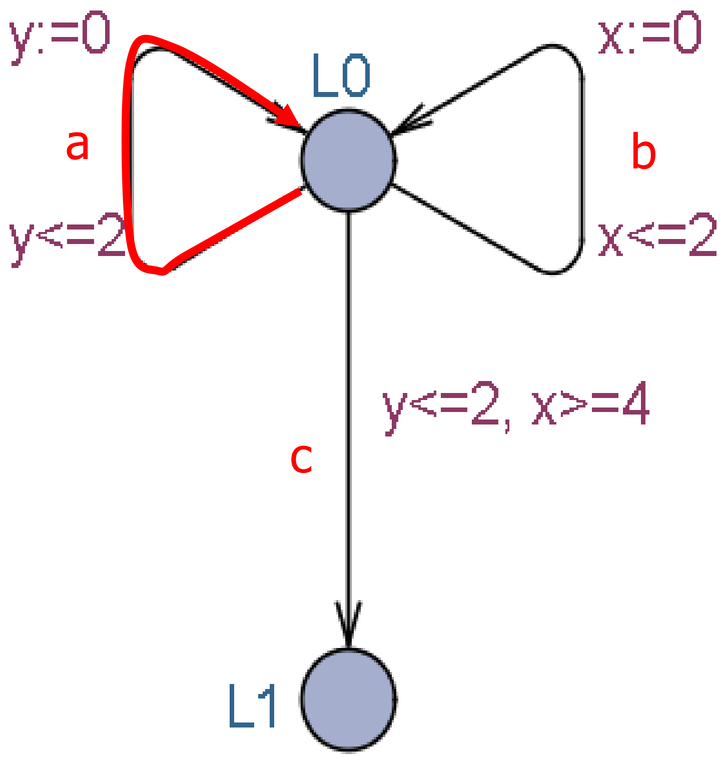
Example

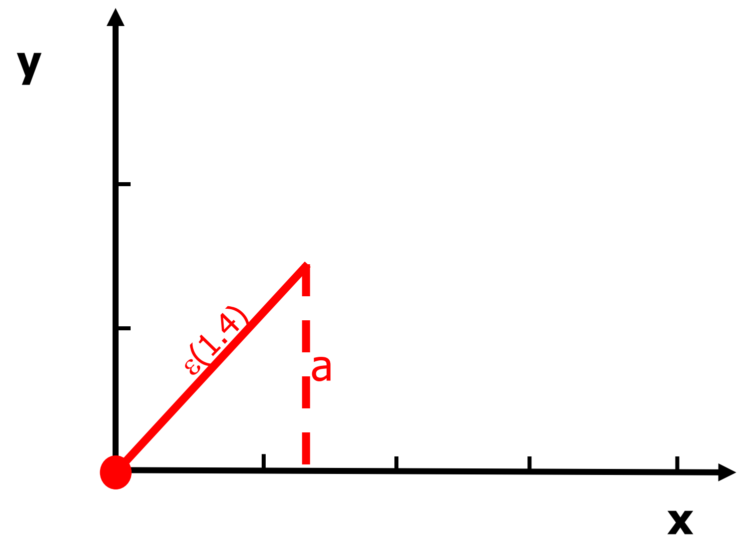


$$(\ell_0, x = 0, y = 0)$$

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

Example



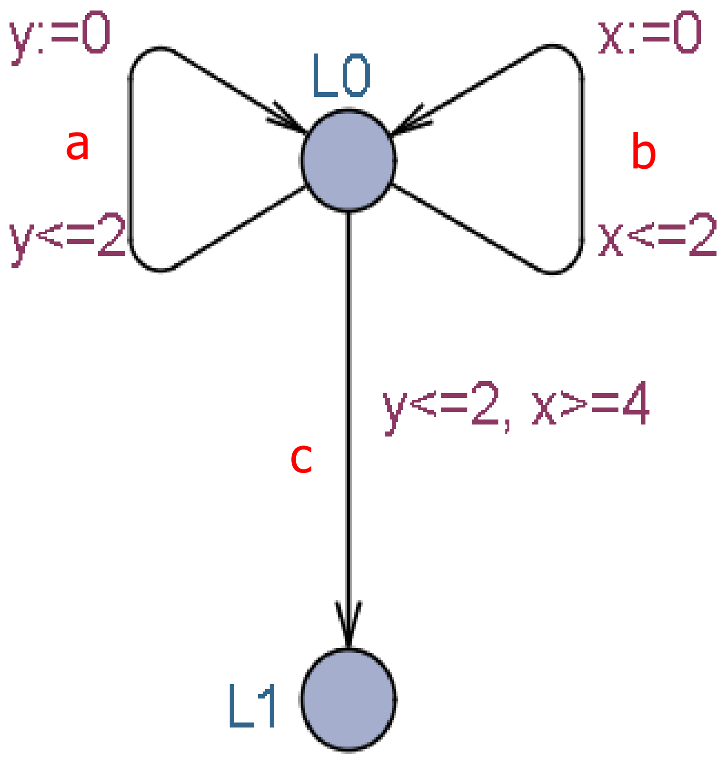


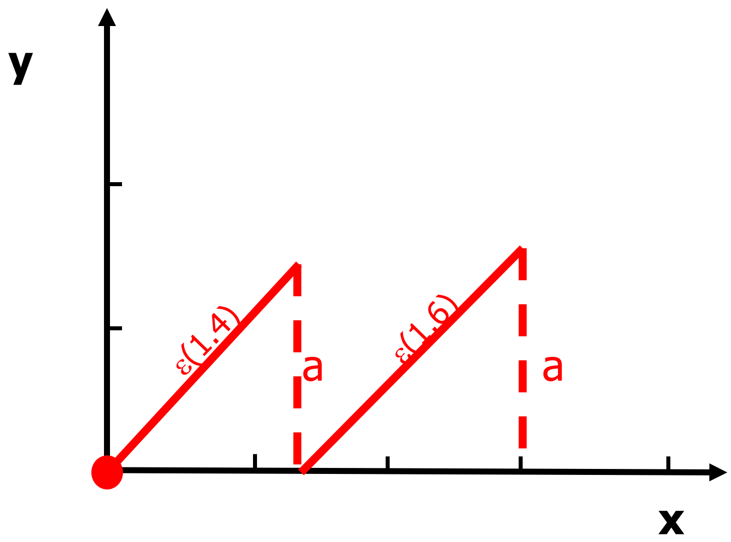
$$(\ell_0, x = 0, y = 0)$$

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

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

Example





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

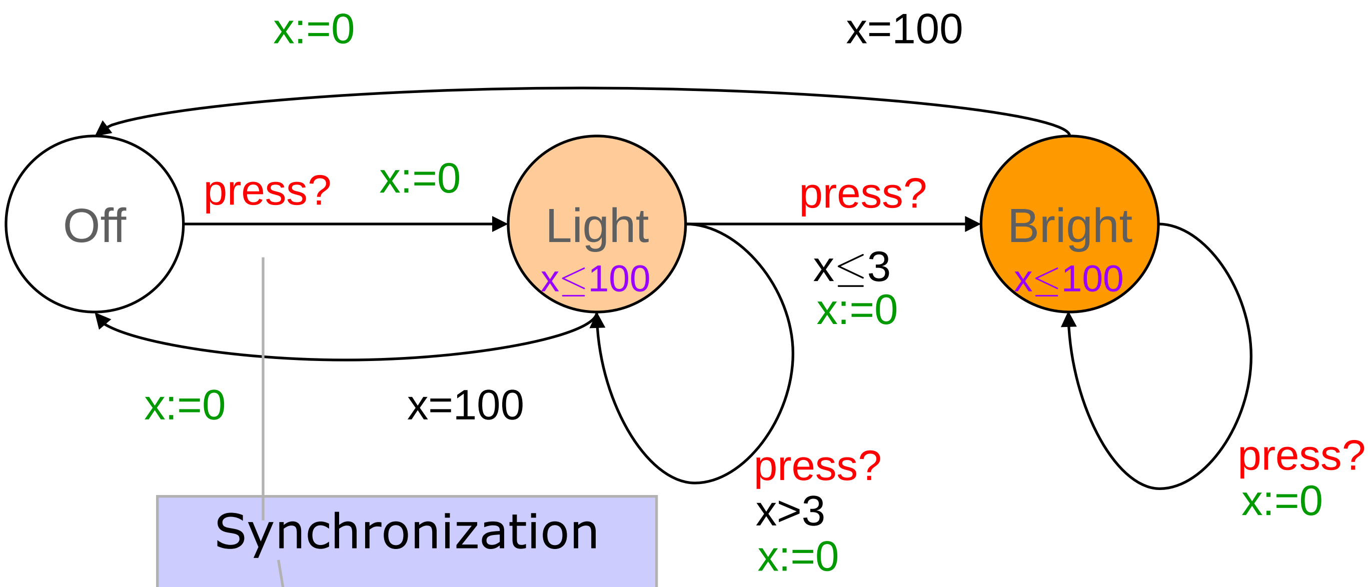
$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$

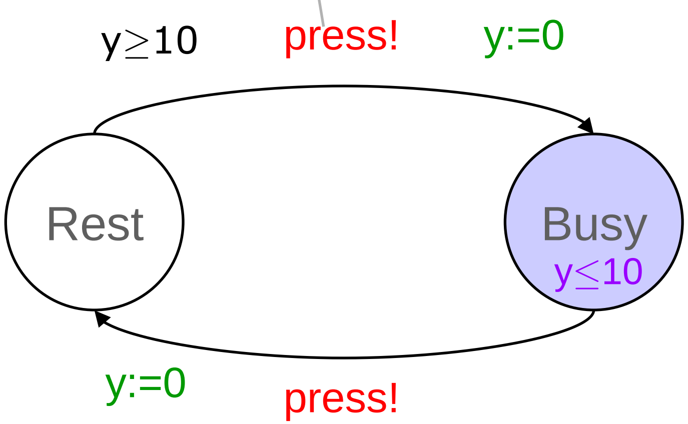
$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

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

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

Networks Light Controller & User





Transition

	(Off, Rest, x=0, y=0)
delay 20	→ (Off, Rest, x=20, y=20)
press?!	→ (Light, Busy, x=0, y=0)
delay 2	→ (Light, Busy, x=2, y=2)
press?!	→ (Bright, Rest, x=0, y=0)

Network Semantics

$$T_1 \|_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \|_X s_0^2)$$ where

$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1' \|_X s_2}$$

$$\frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1 \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{a!}_1 s_1' \qquad s_2 \xrightarrow{a?}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\tau} s_1' \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{e(d)}_1 s_1' \qquad s_2 \xrightarrow{e(d)}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{e(d)} s_1' \|_X s_2'}$$

Network Semantics

(URGENT synchronization)

+ Urgent synchronization

$$T_1 \|_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \|_X s_0^2)$$ where

$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1' \|_X s_2} \qquad \frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1 \|_X s_2'}$$

$$\frac{s_1 \xrightarrow{a!}_1 s_1' \qquad s_2 \xrightarrow{a?}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\tau} s_1' \|_X s_2'}$$

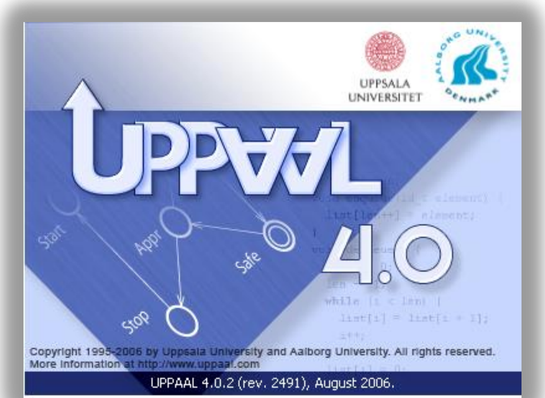
$$\frac{s_1 \xrightarrow{e(d)}_1 s_1' \qquad s_2 \xrightarrow{e(d)}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{e(d)} s_1' \|_X s_2'}$$

$\forall d' < d,\ \forall u \in UAct:$

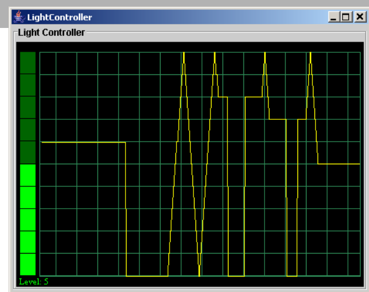
$\neg (s_1 \xrightarrow{e(d')} \xrightarrow{u?} \wedge\ s_2 \xrightarrow{e(d')} \xrightarrow{u!})$

UPPAAL

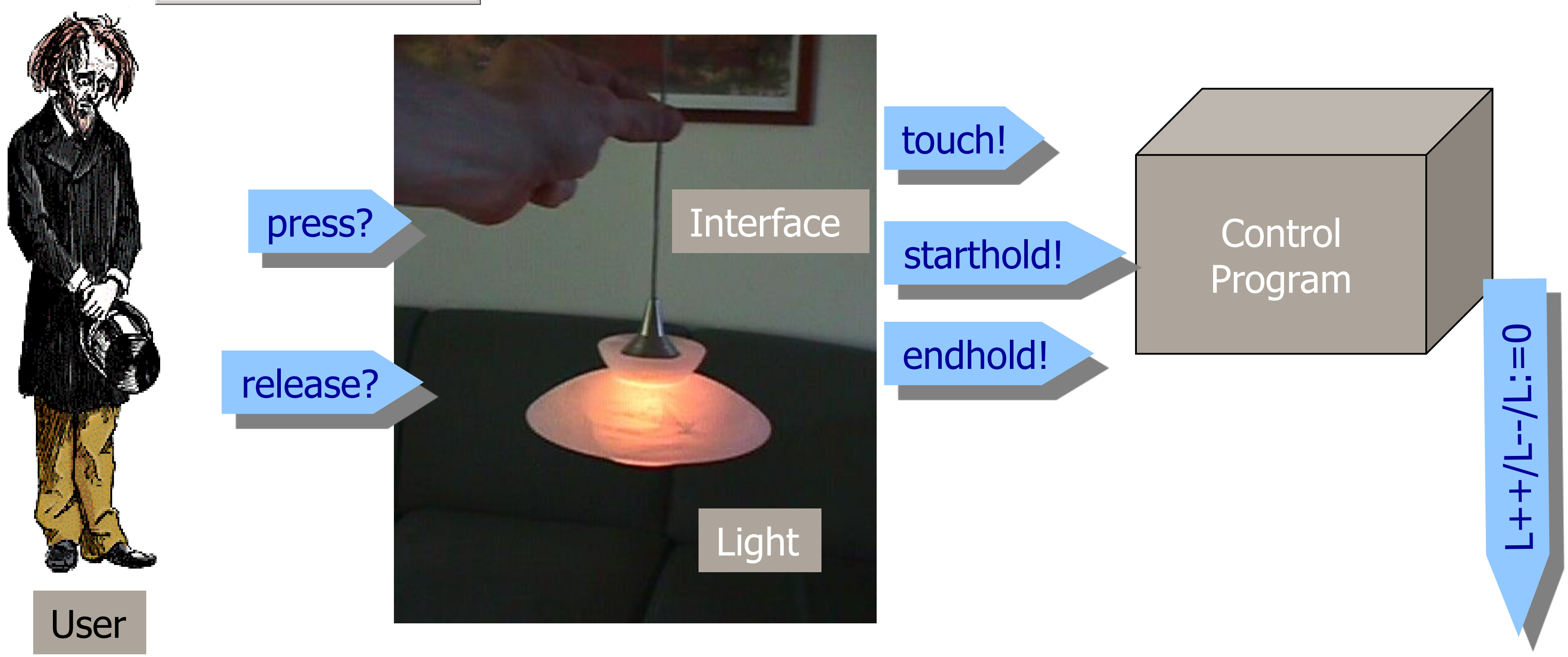
First Introduction



Light Control Interface



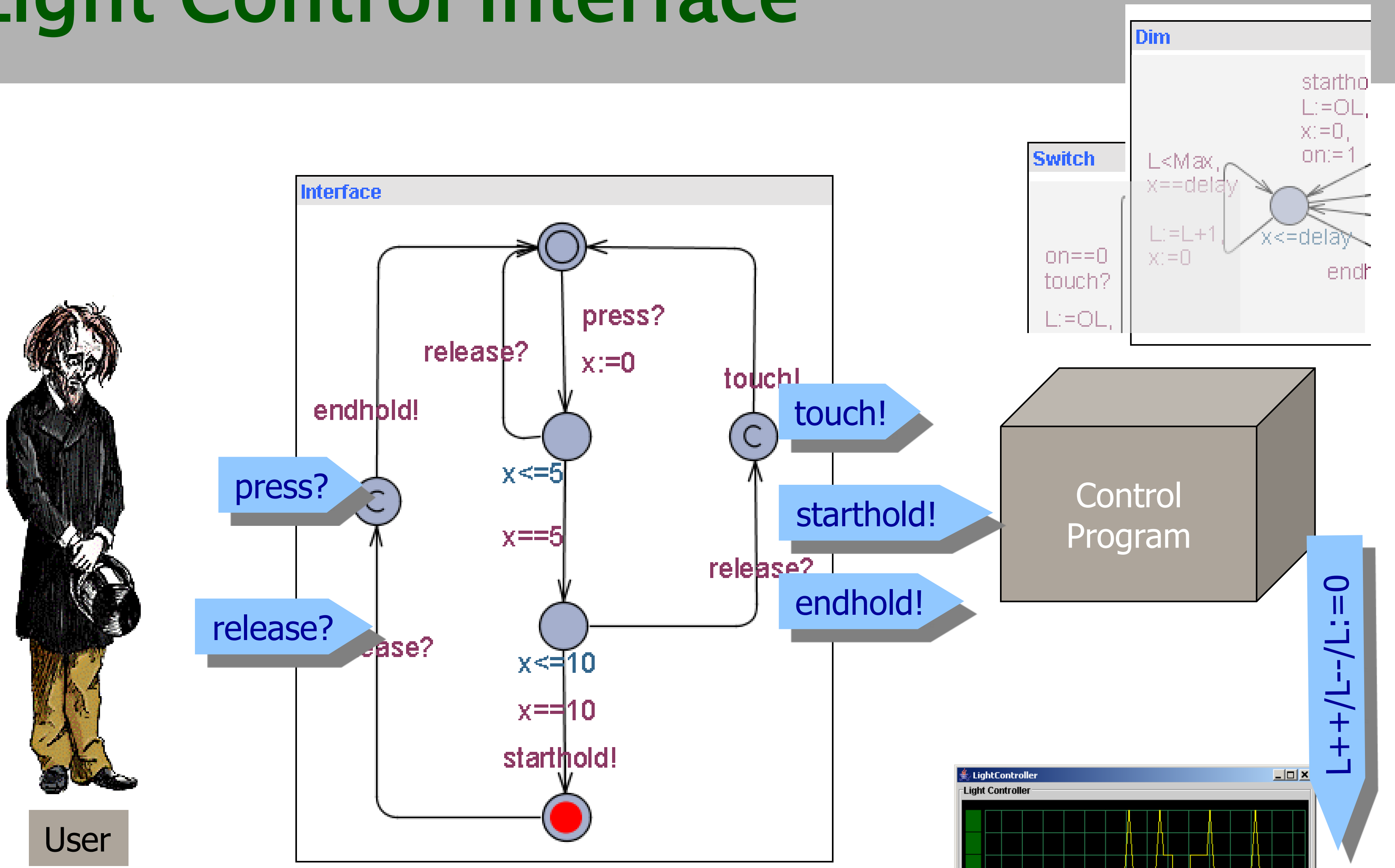
press? d release? → touch! $0.5 \leq d \leq 1$
press? 1 → startH!
press? d release? → endH! $d > 1$

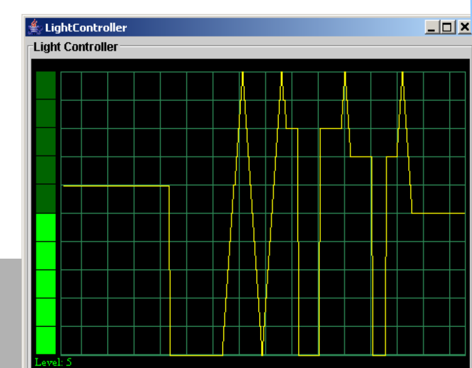


press? 0.2 release? … press? 0.7 release? … press? 1.0 2.4 release? …

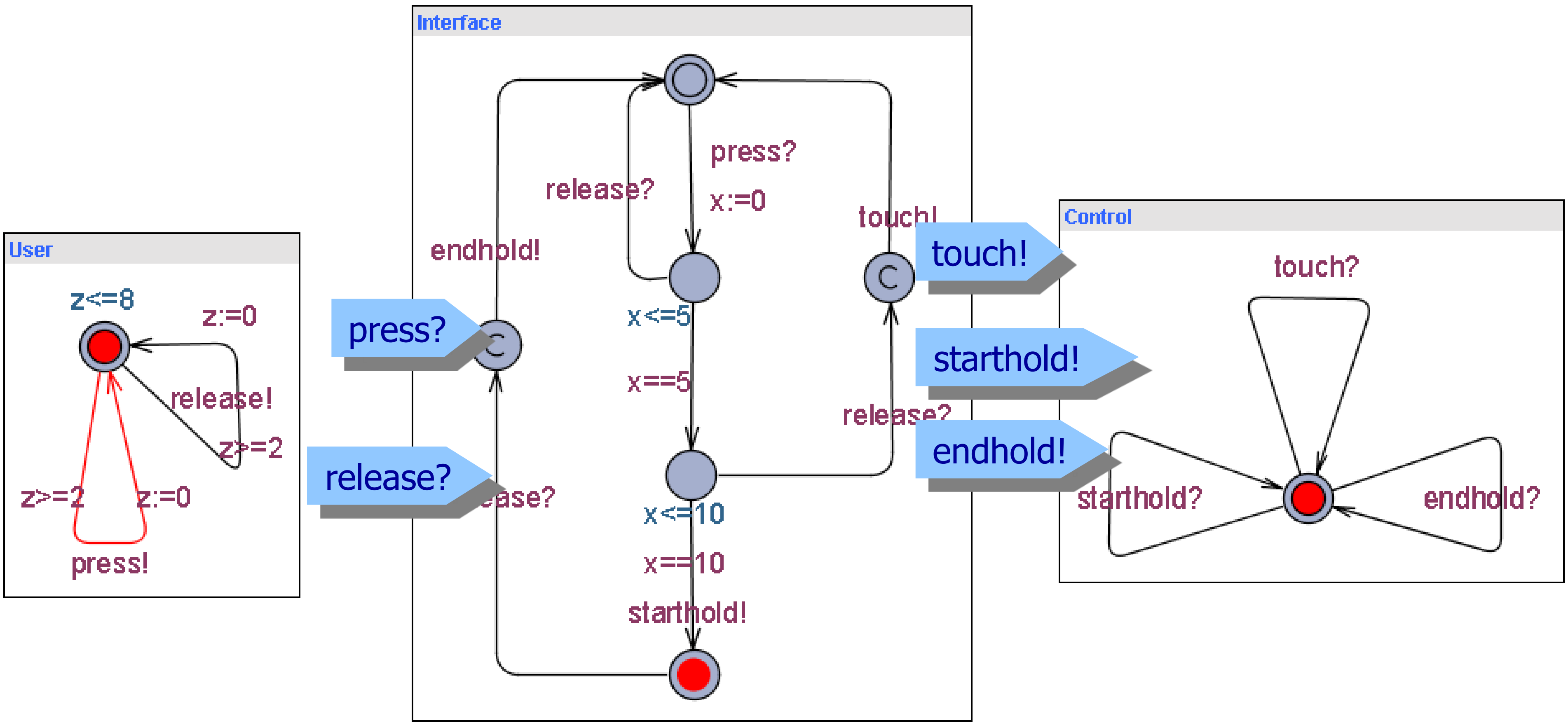
Ø thouch! startH! endH!

Light Control Interface

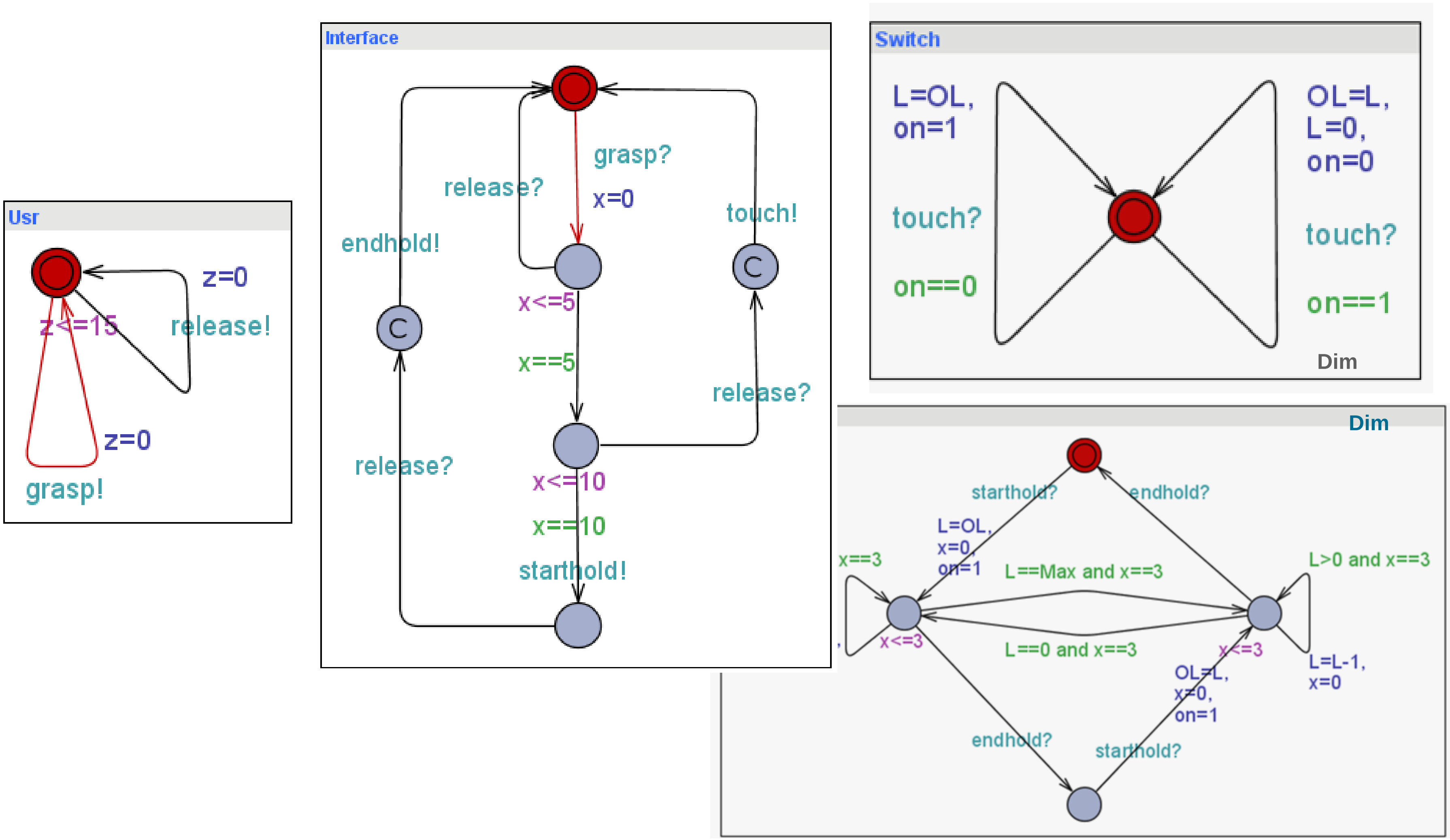




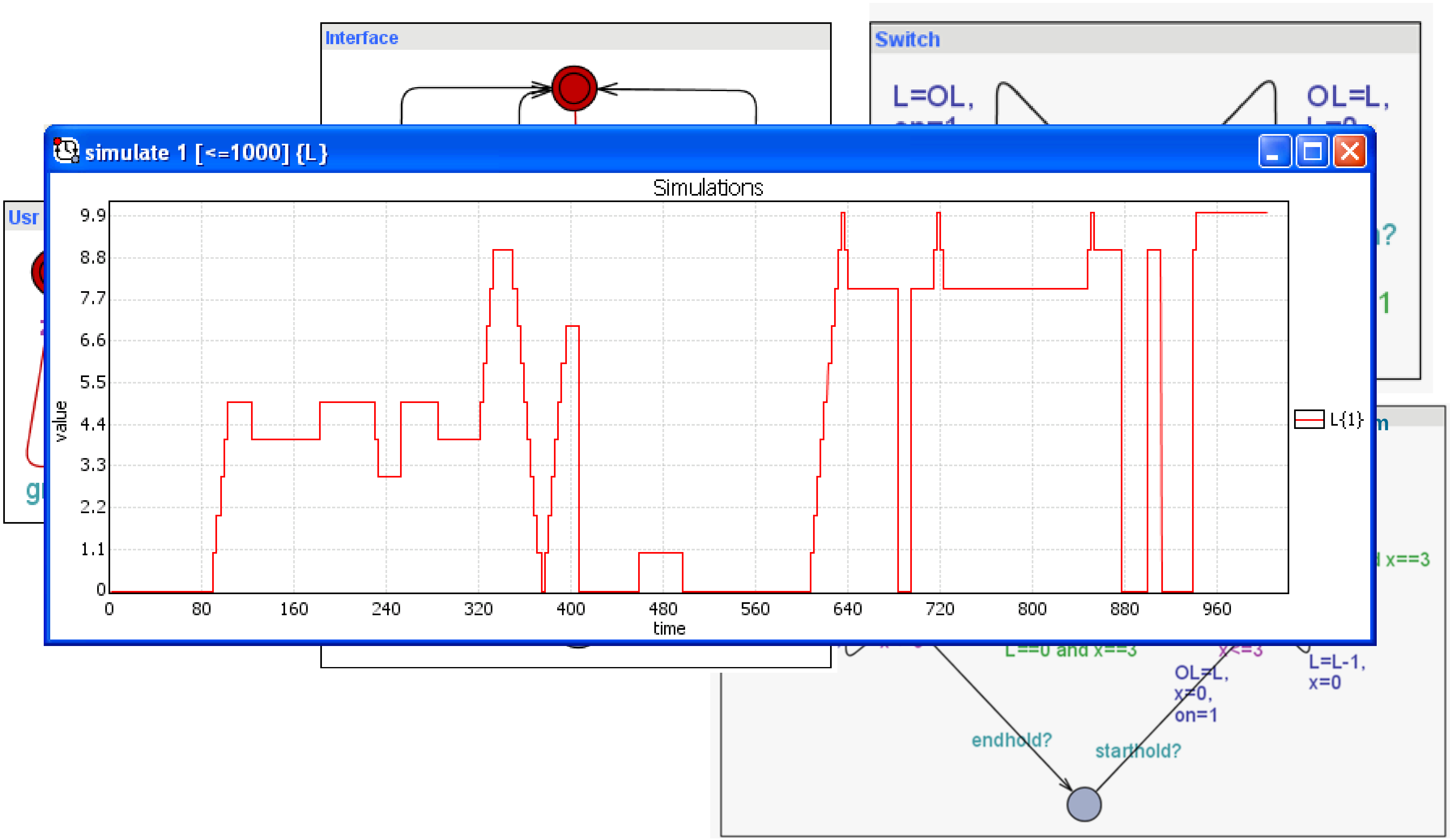
Light Control Network



Full Light Controller

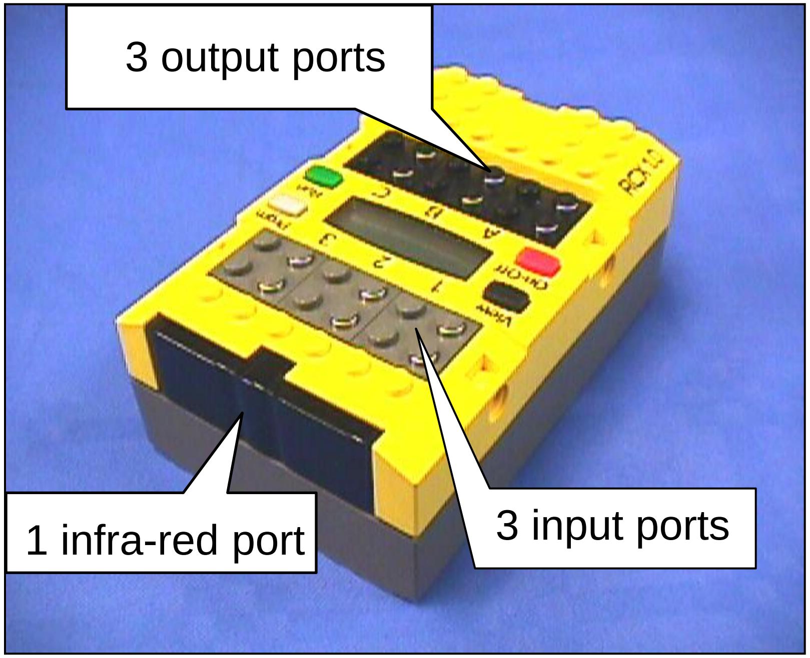


Full Light Controller



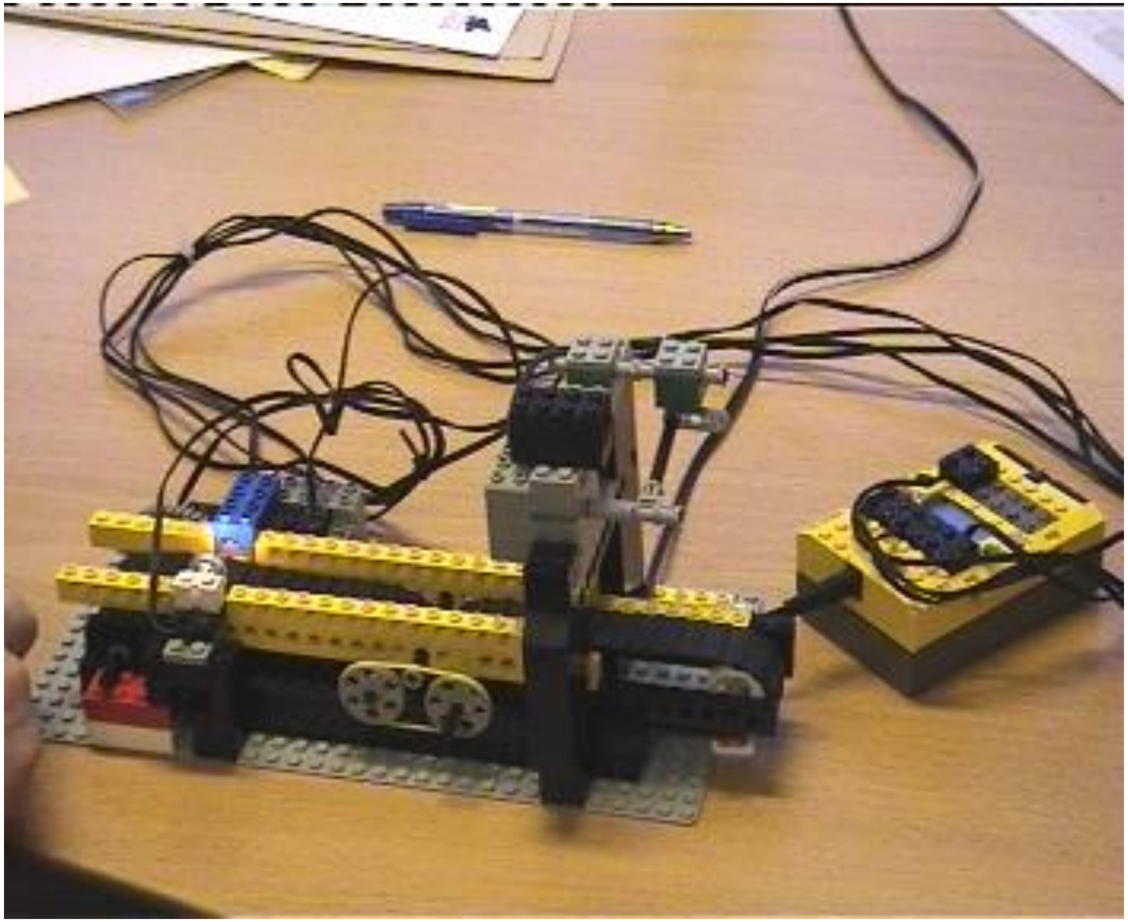
LEGO Mindstorms/RCX

- 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 Timed System

The Plant
Conveyor Belt
&
Bricks

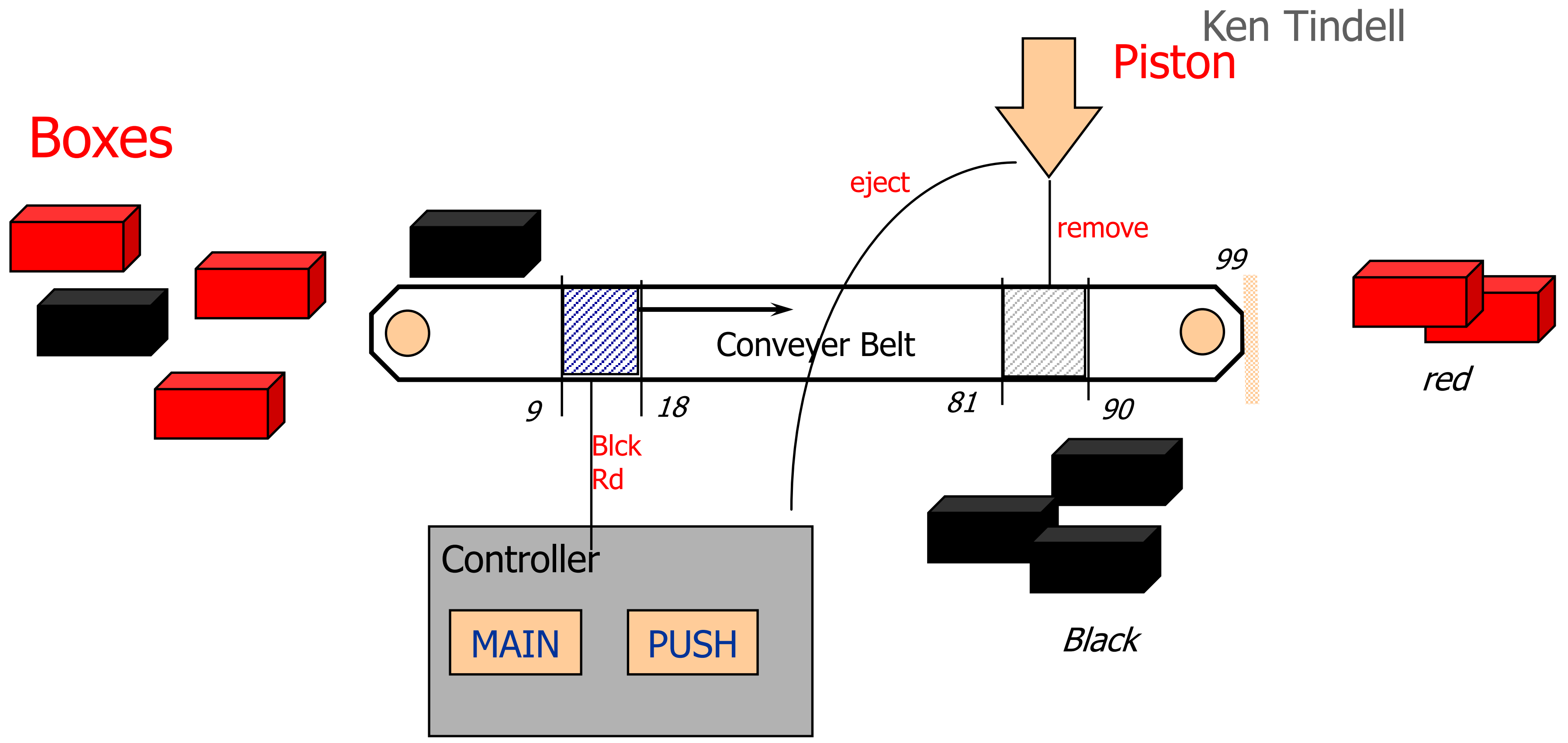


Controller Program
LEGO MINDSTORM

What is suppose to happen?

First UPPAAL model

Sorting of Lego Boxes



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

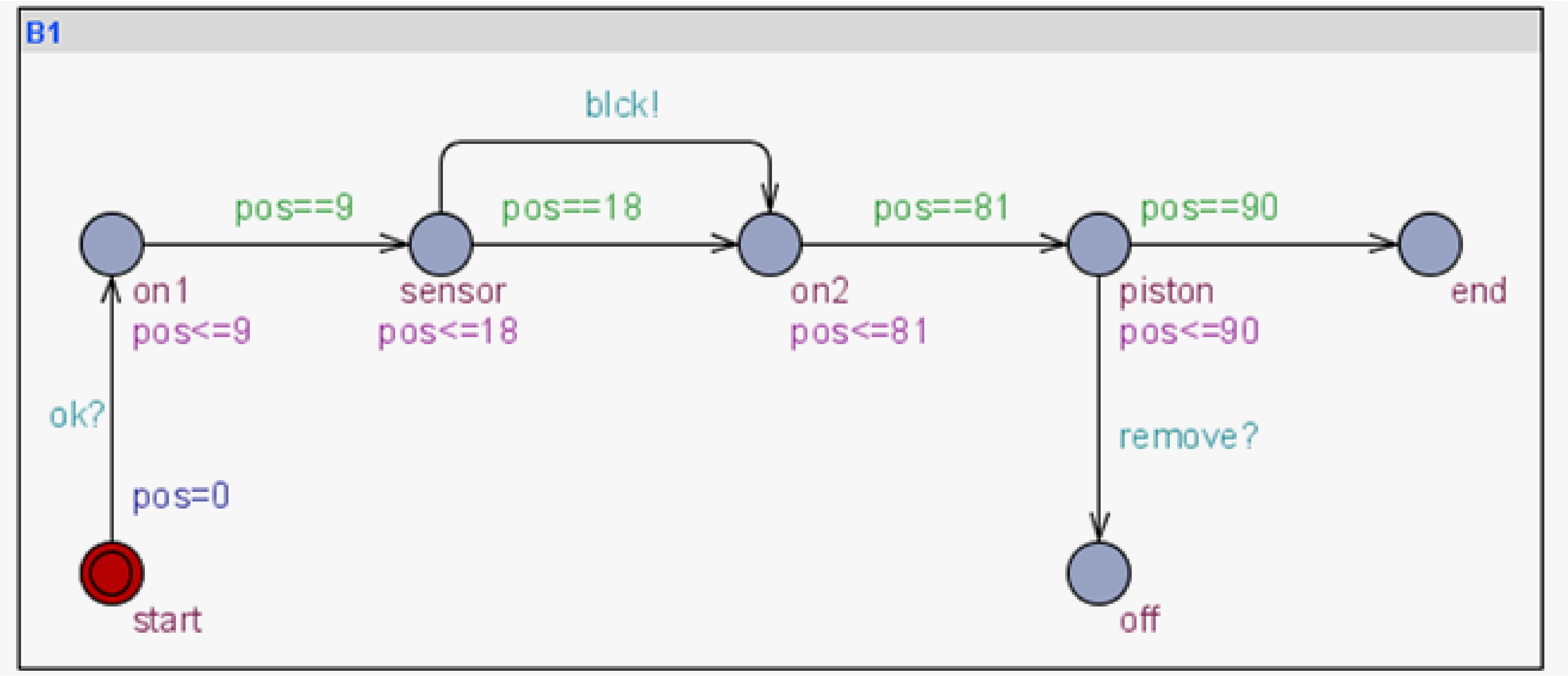
NQC programs

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

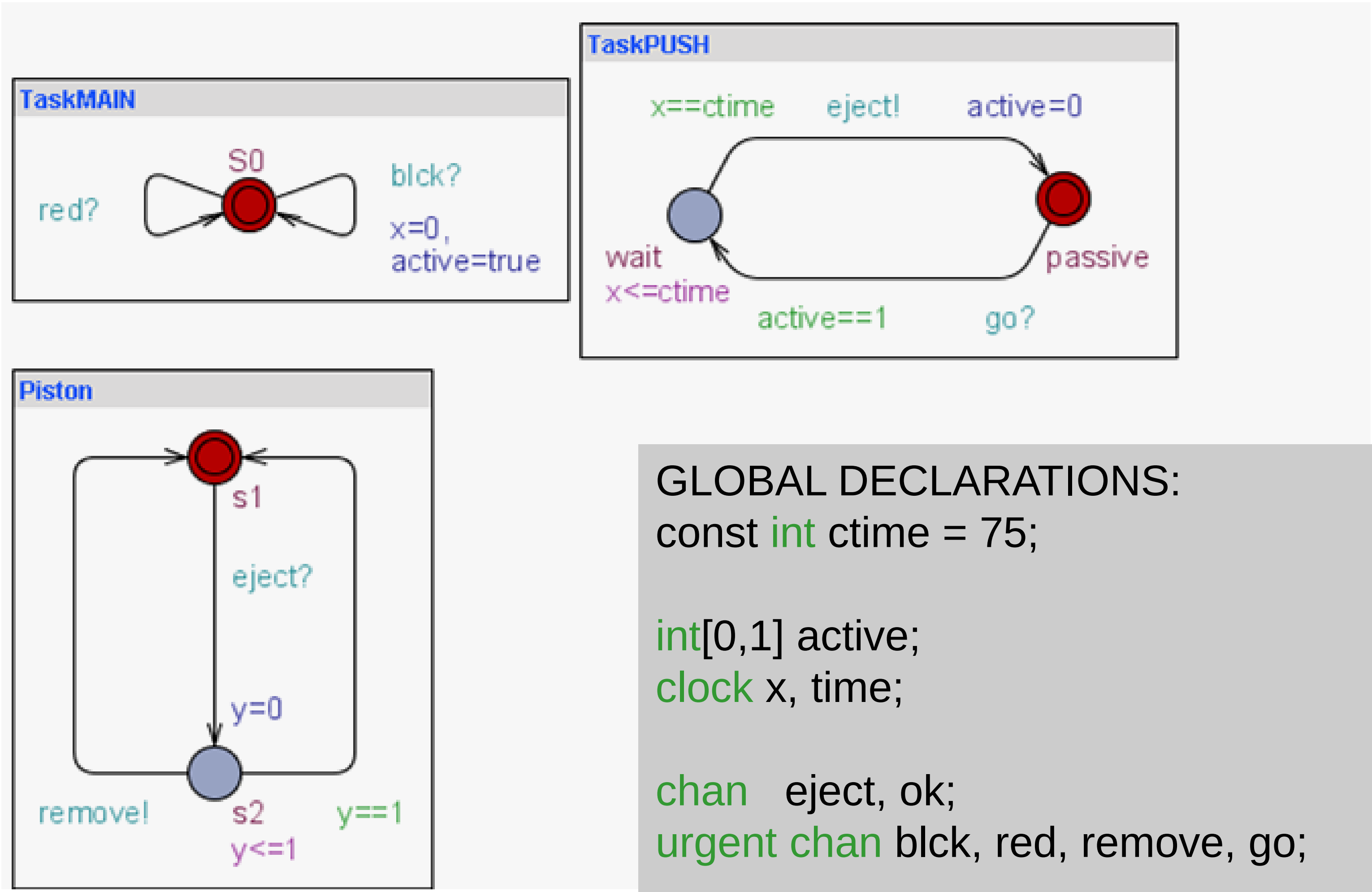
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 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);
  }
}
```

A Black Brick



Control Tasks & Piston



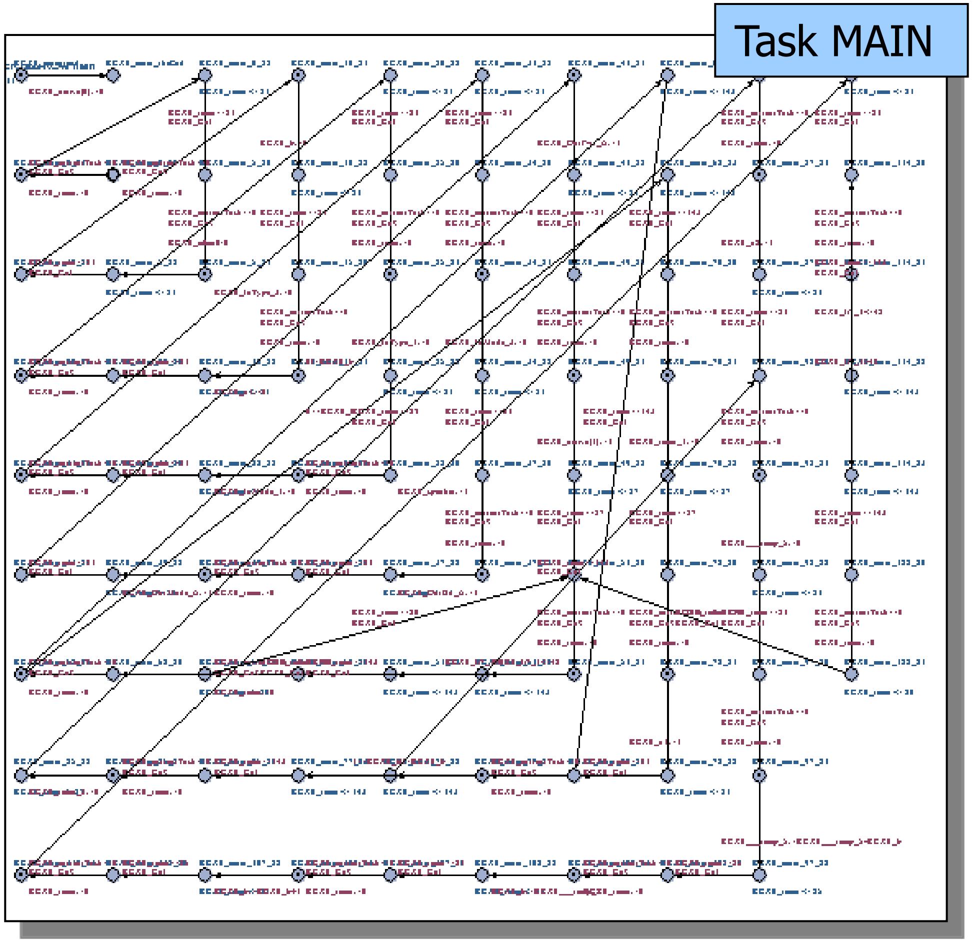
GLOBAL DECLARATIONS:
const int ctime = 75;

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

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

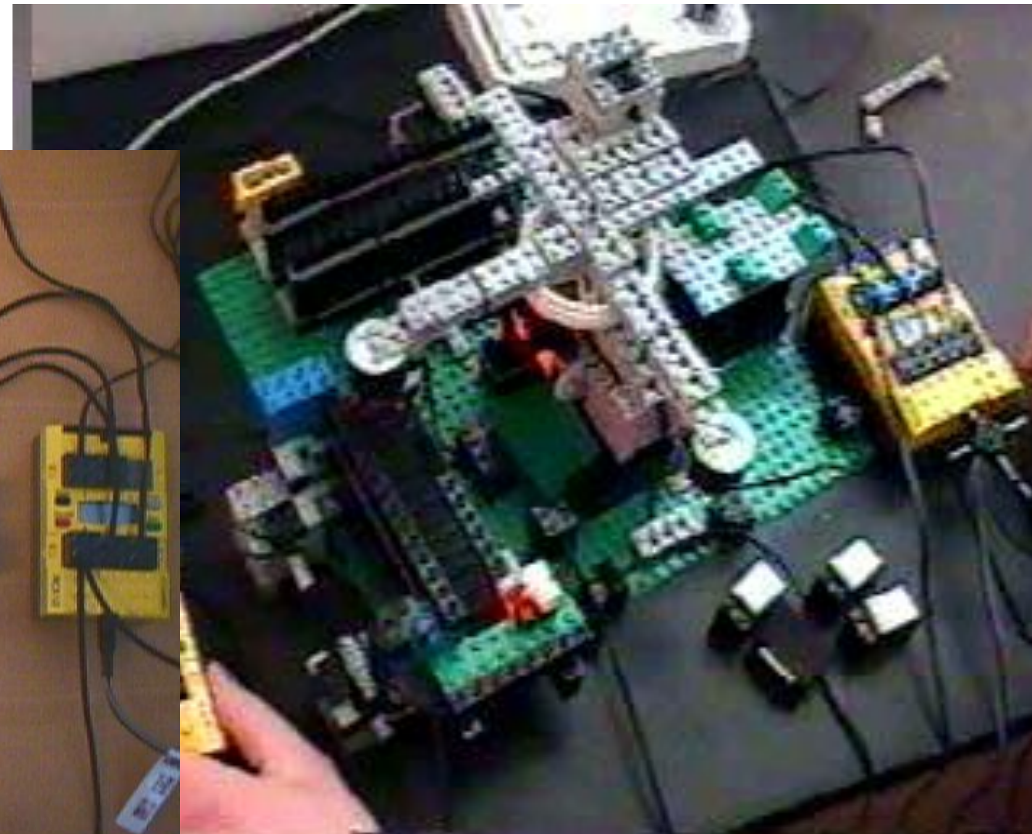
From RCX to UPPAAL

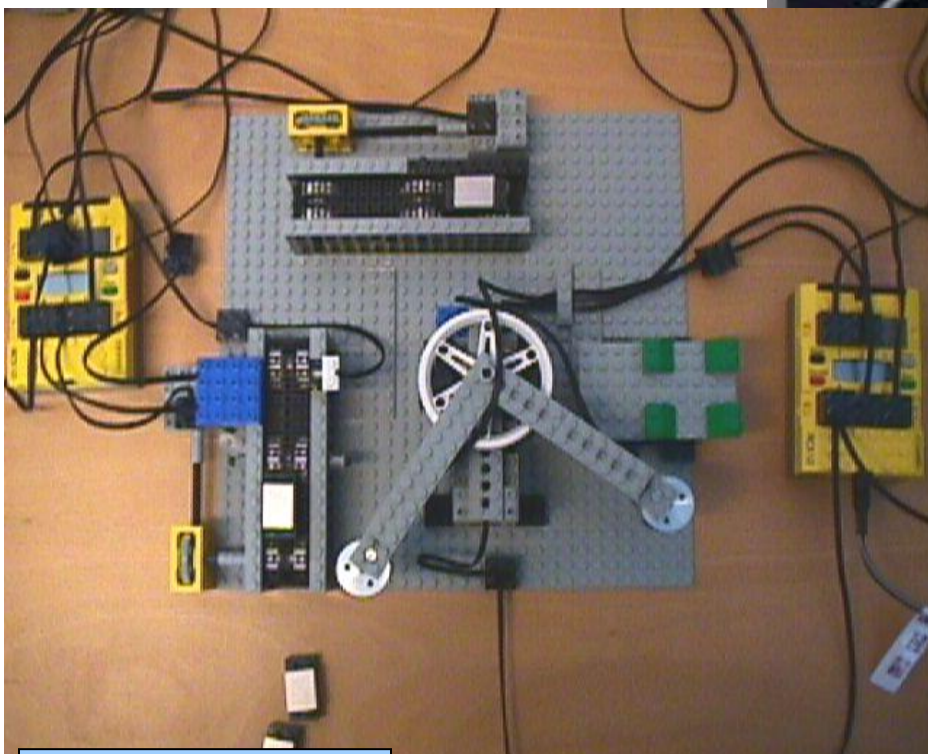
- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000



The Production Cell

Course at DTU, Copenhagen



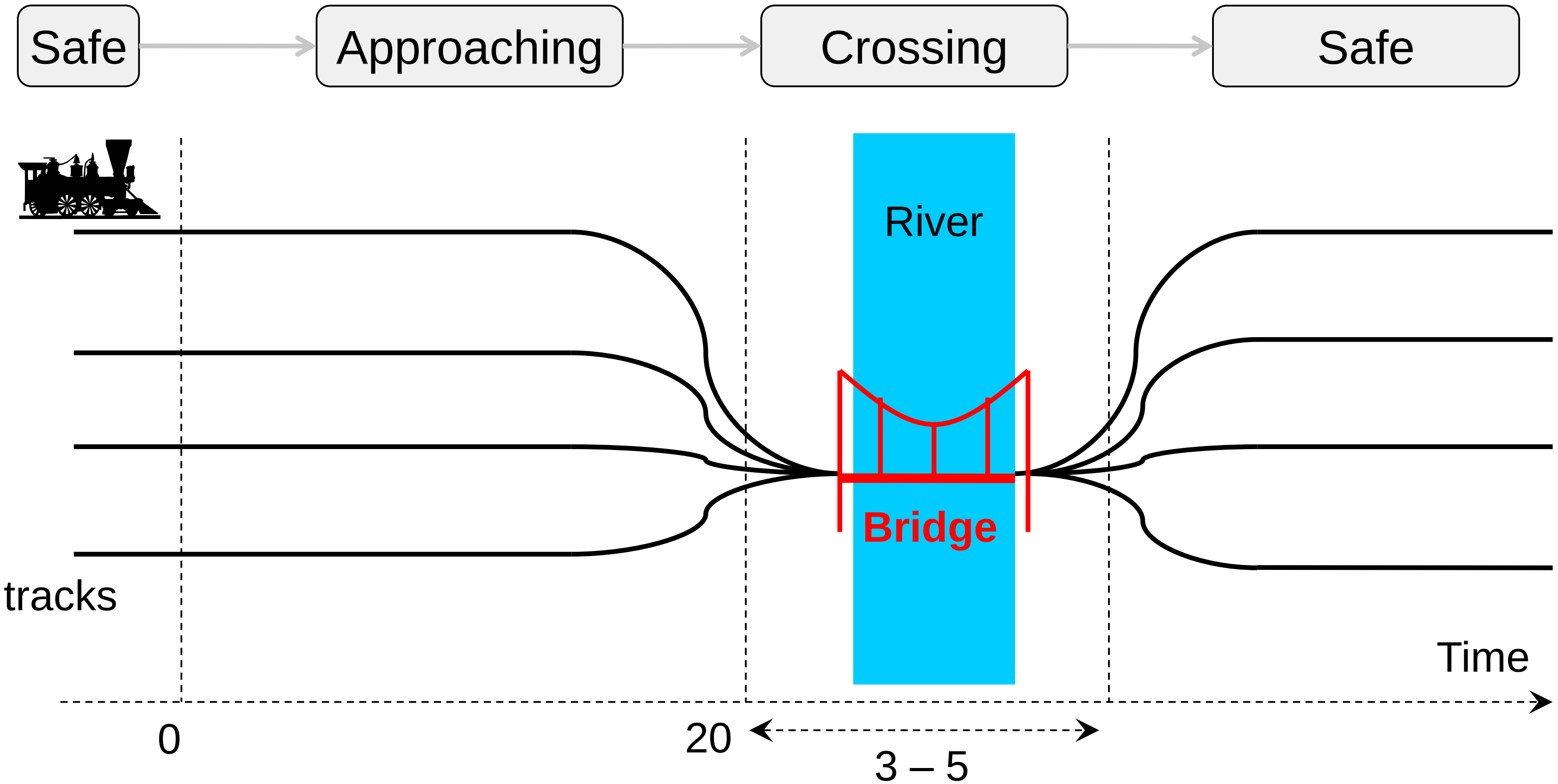


Production Cell

UPPAAL

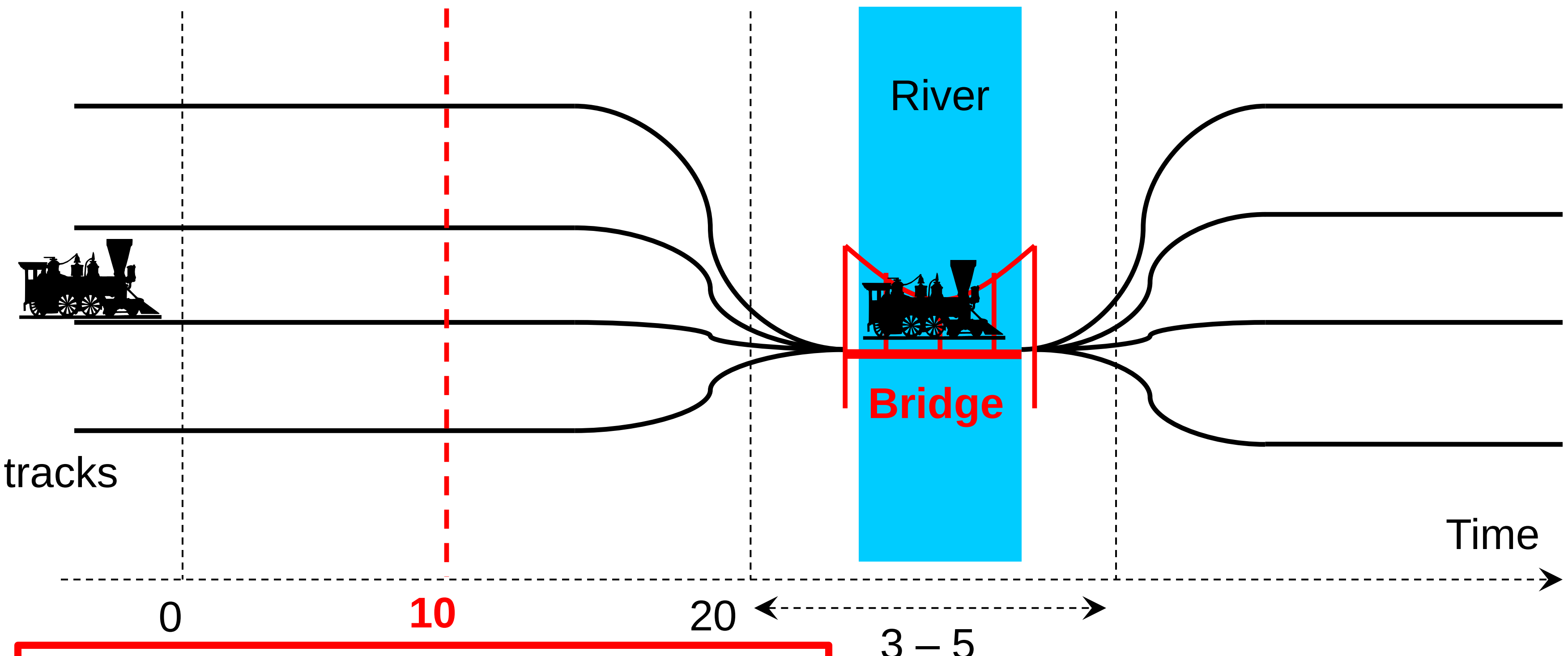
Modeling & Specification

Train Crossing



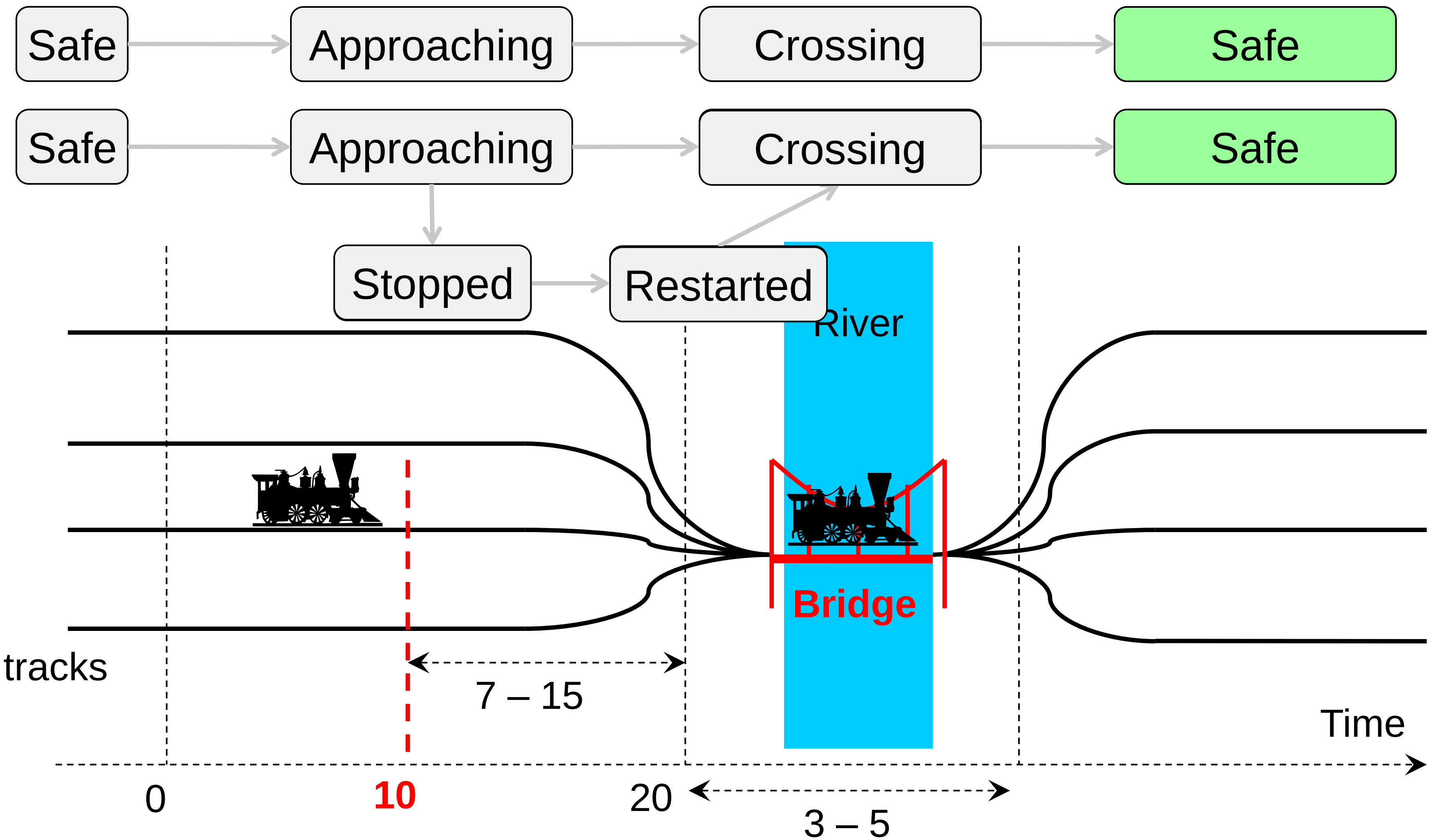
Train Crossing

Safe	Approaching	Crossing	*Safe*
Safe	Approaching	*Crossing*	*Safe*

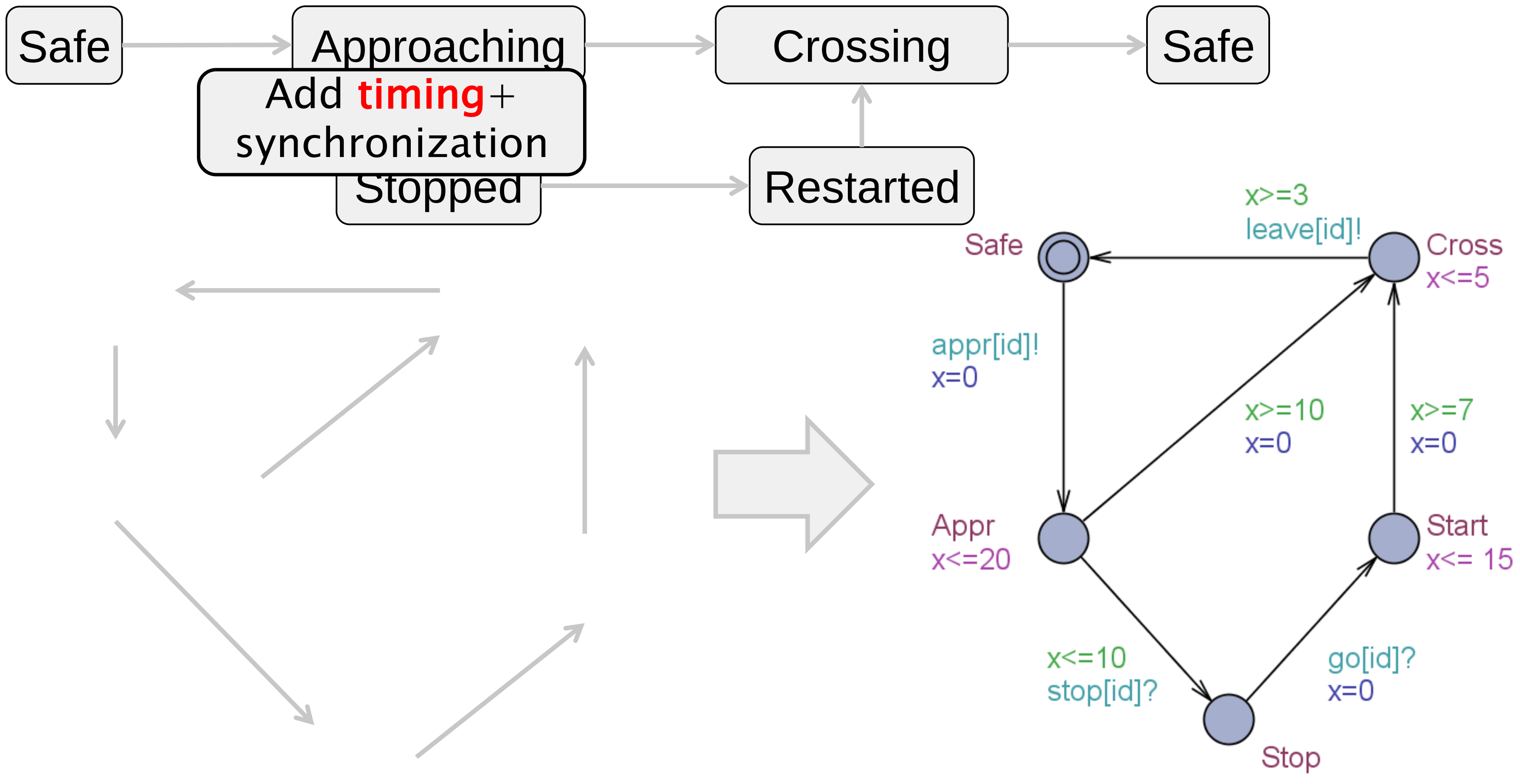


Stop the train while it still stoppable!

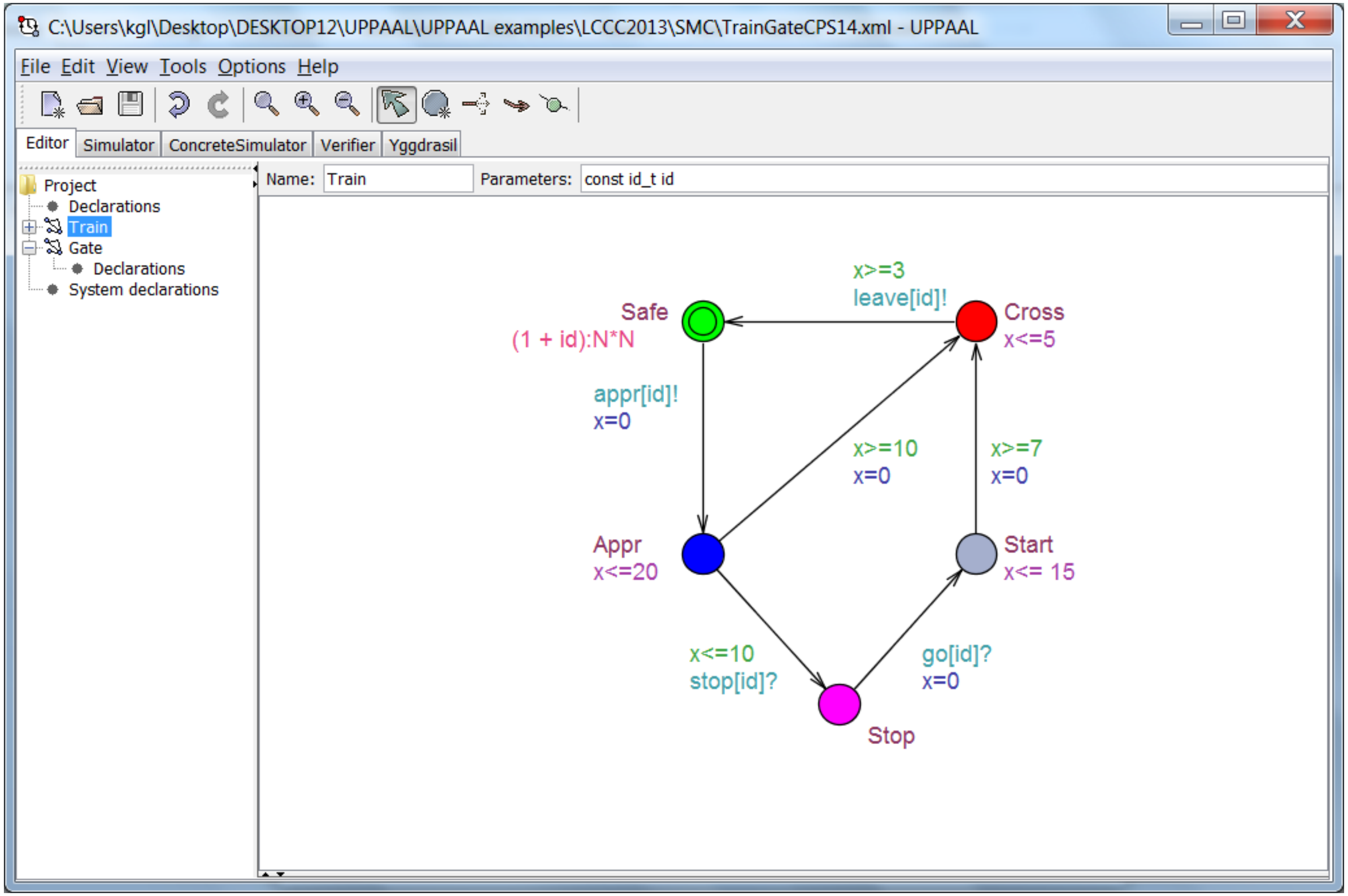
Train Crossing



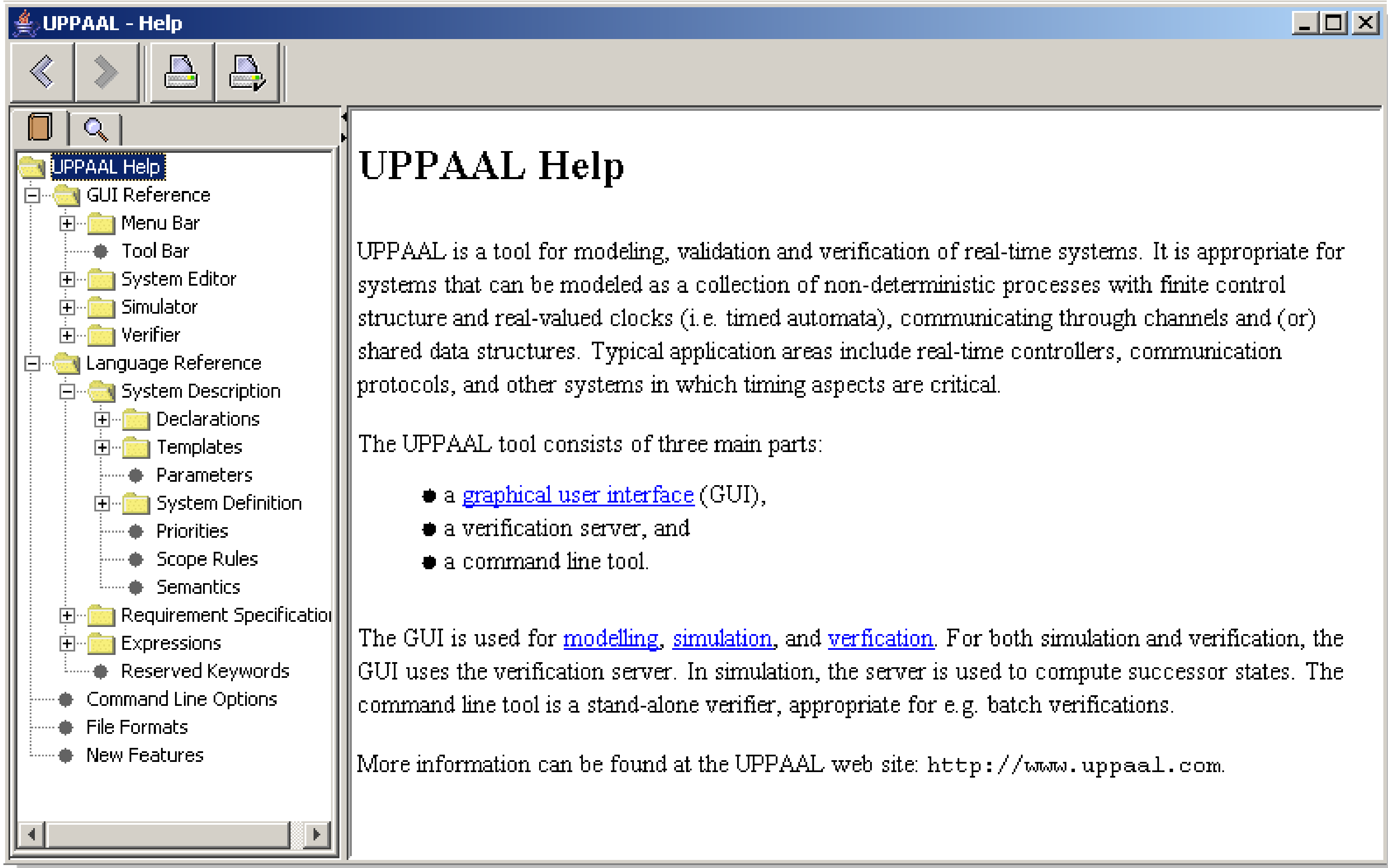
Train Crossing



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.

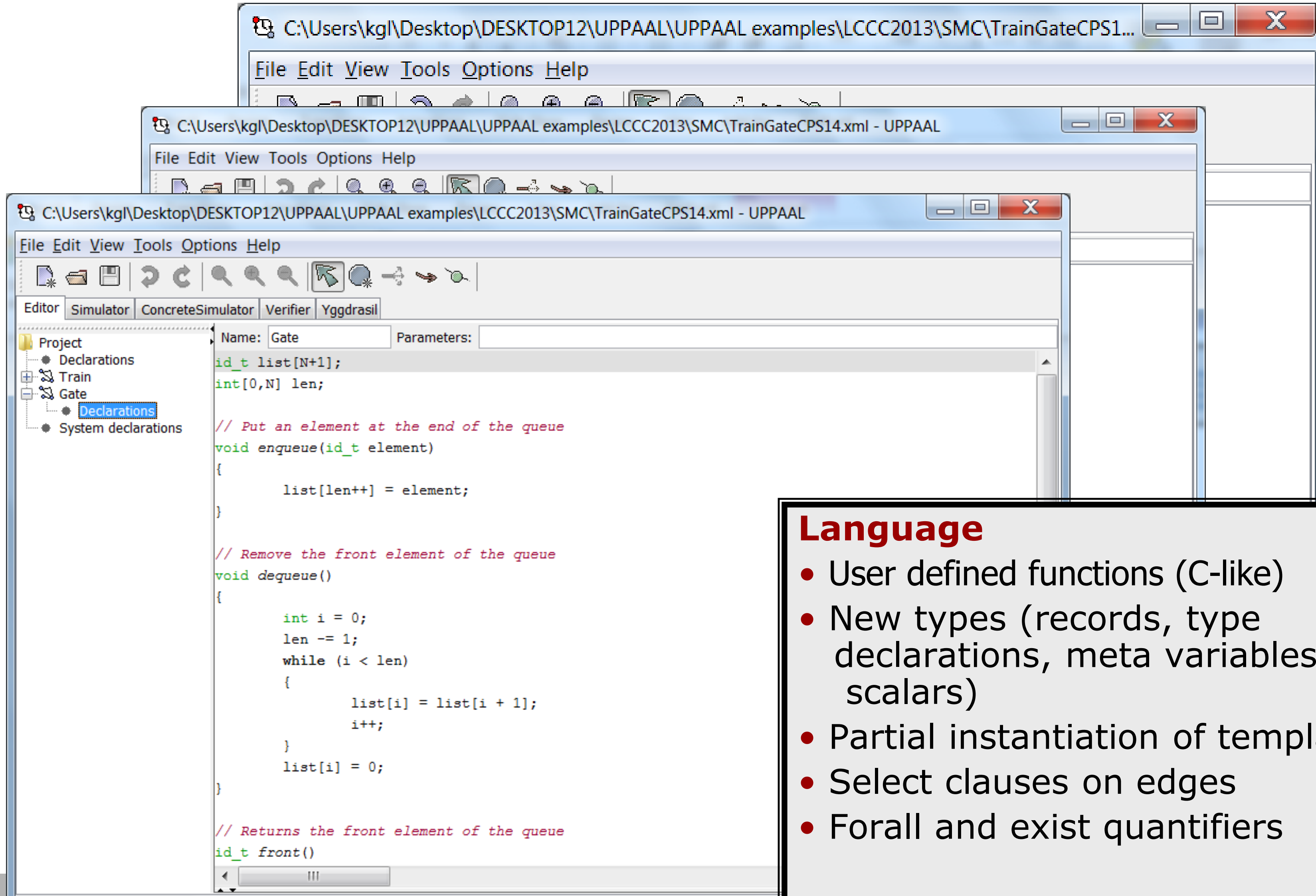
Logical Specifications

- 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, and locations are allowed (and arrays of these).

Editor



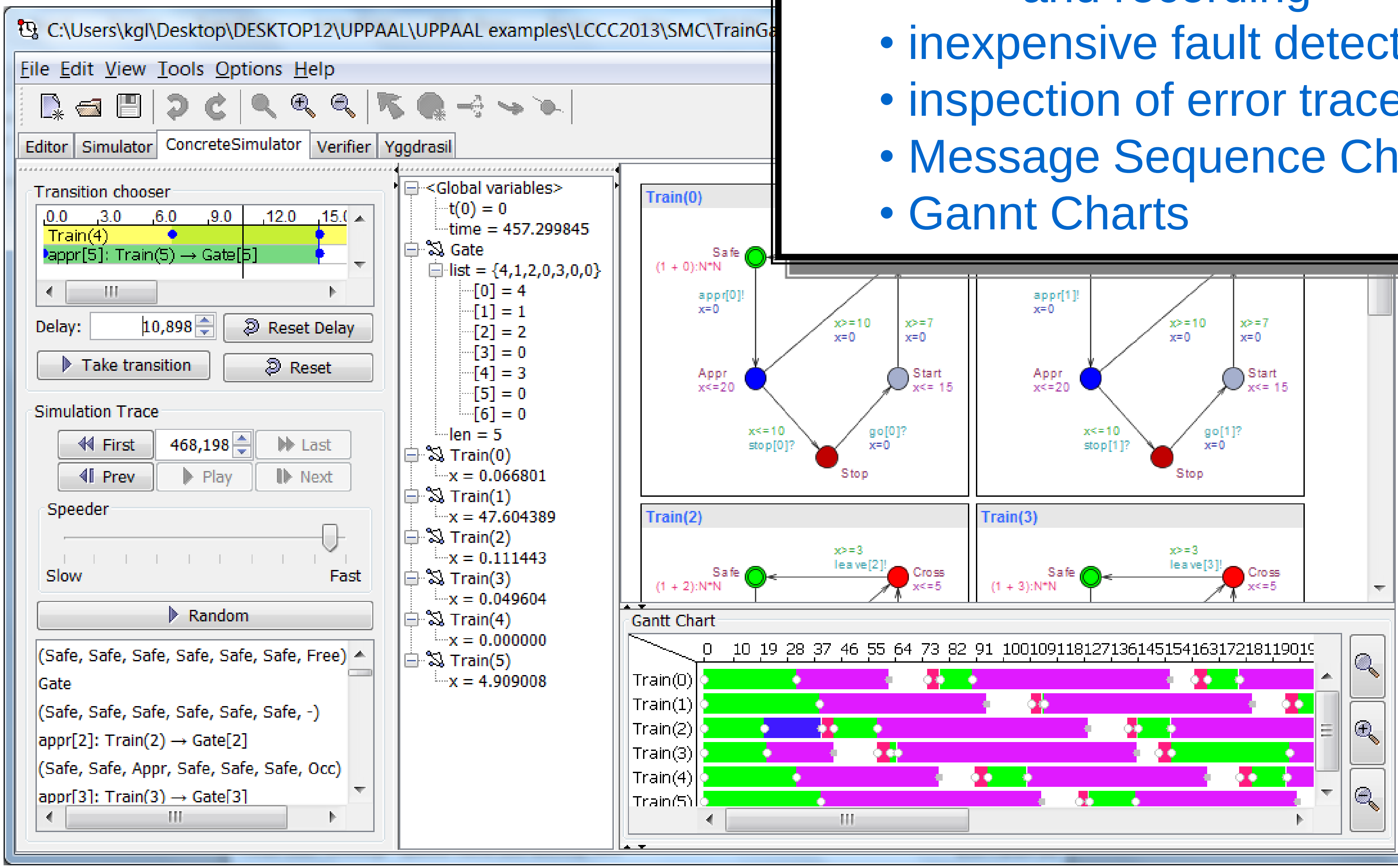
Language

- 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

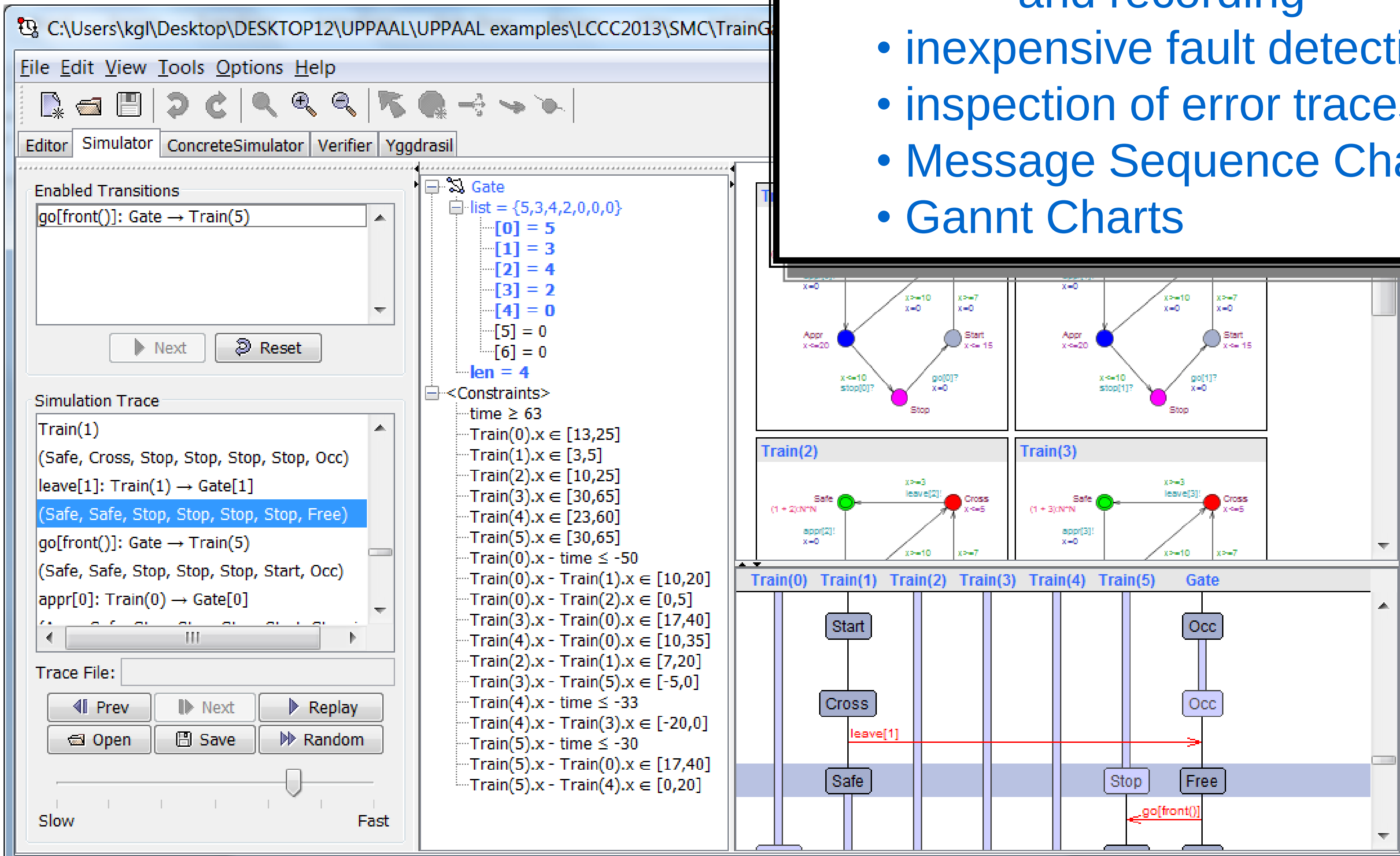
- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gannt Charts

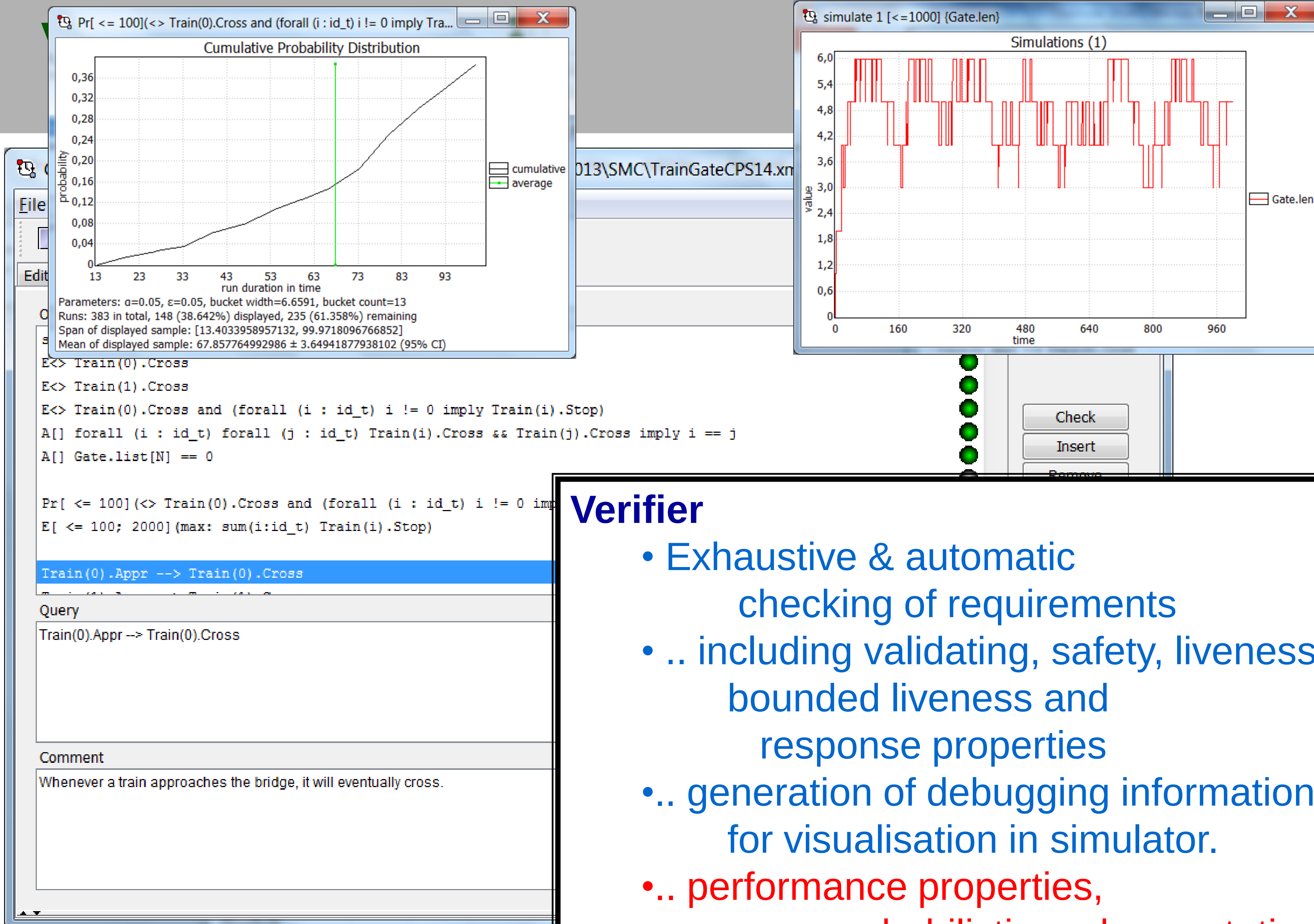


Symbolic Simulator

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
- .. generation of debugging information for visualisation in simulator.
- .. performance properties, e.g probabilistic and expectation.
- .. plot composer

Bang & Olufsen (1997)

- Bug known to exist for 10 years
- Ill-described:
  - 2.800 loc +
  - 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.
- Follow-up project.

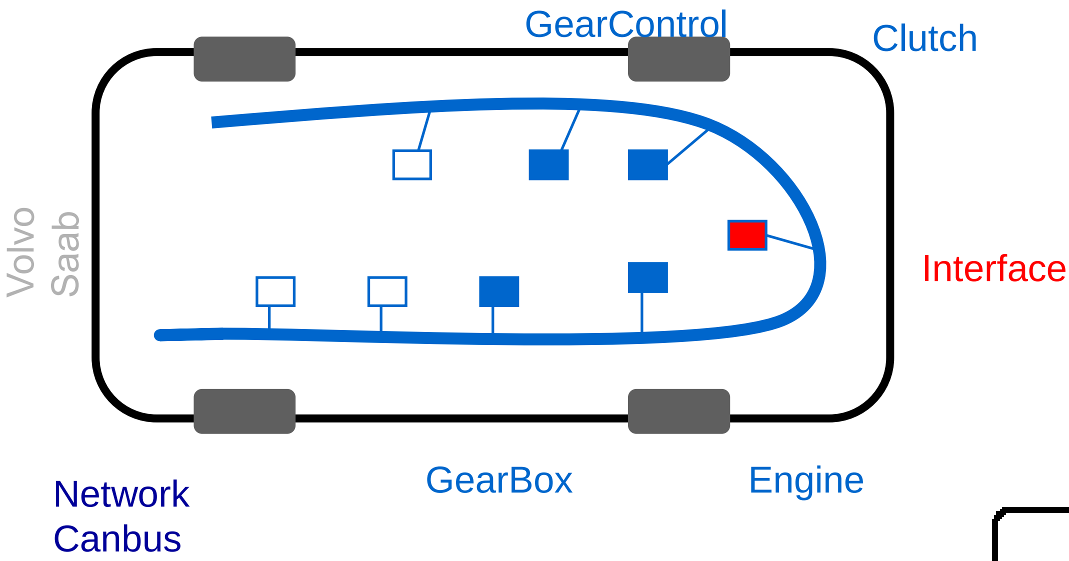
Arne Skou, Klaus Havelund

1st RTSS’97 talk, Klaus Havelund

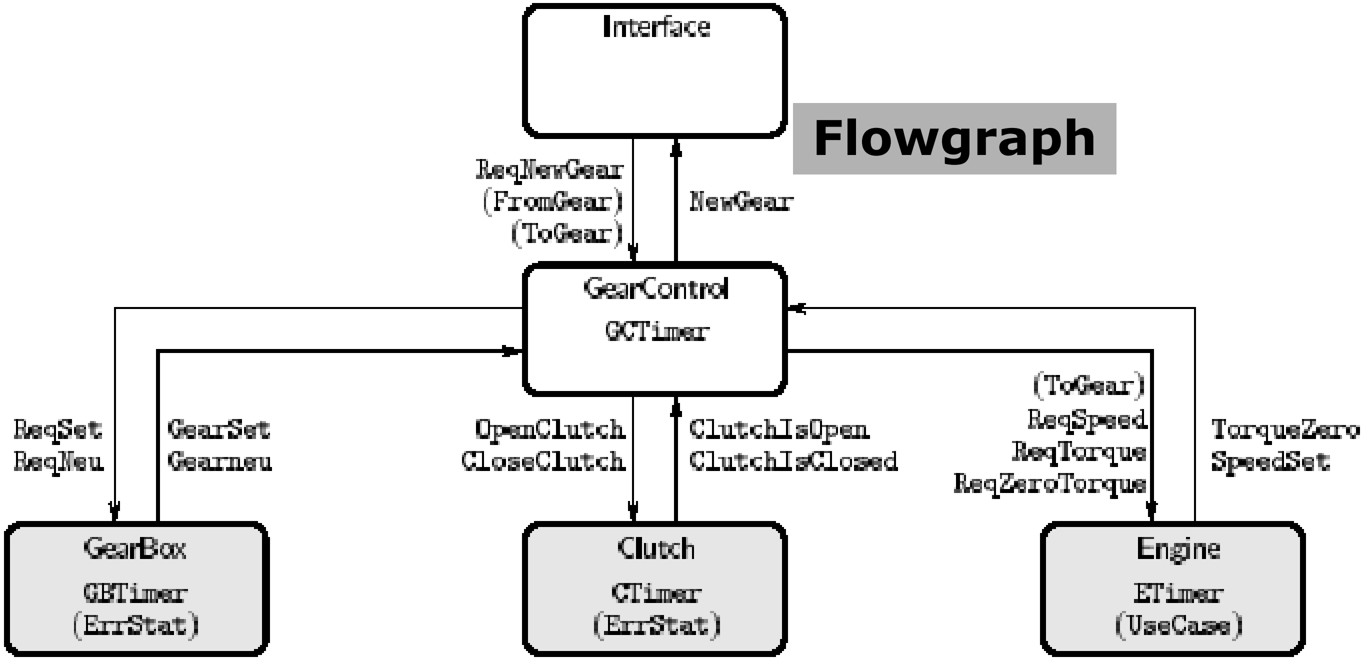
MECEL AB (1998)

Gear Controller

Lindahl, Pettersson, Yi 1998



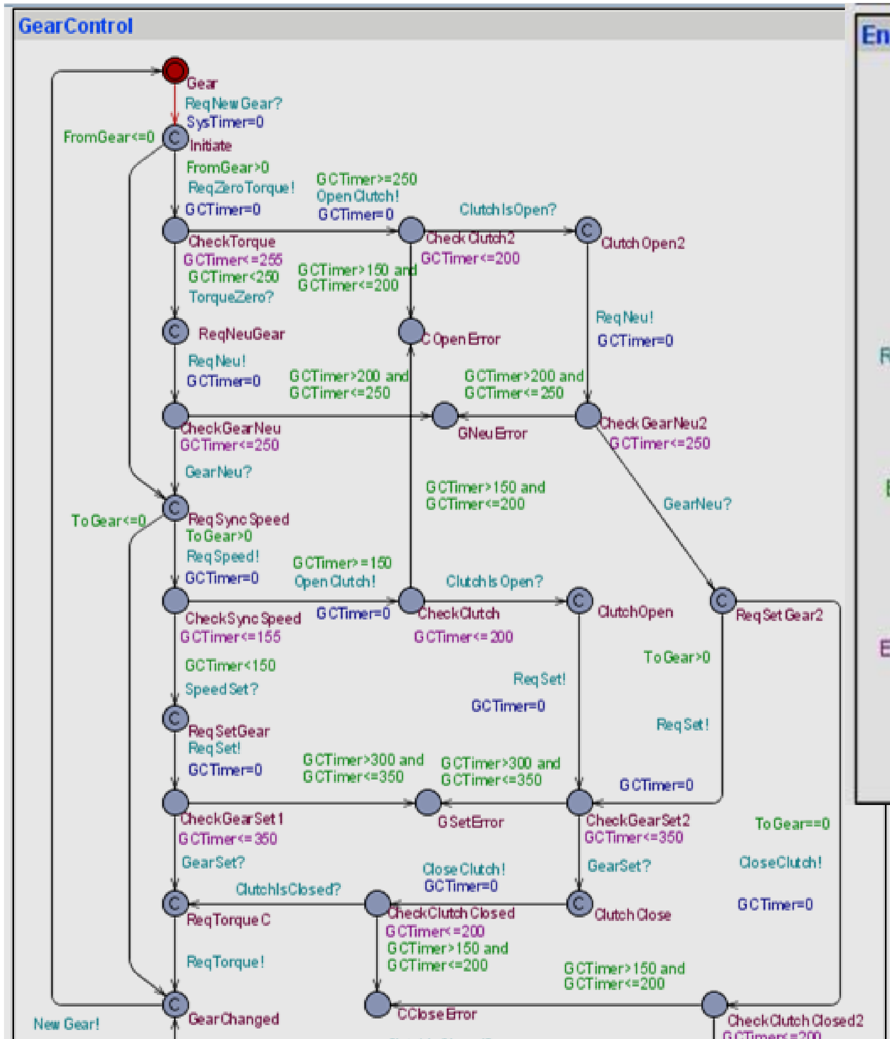
Paul Pettersson

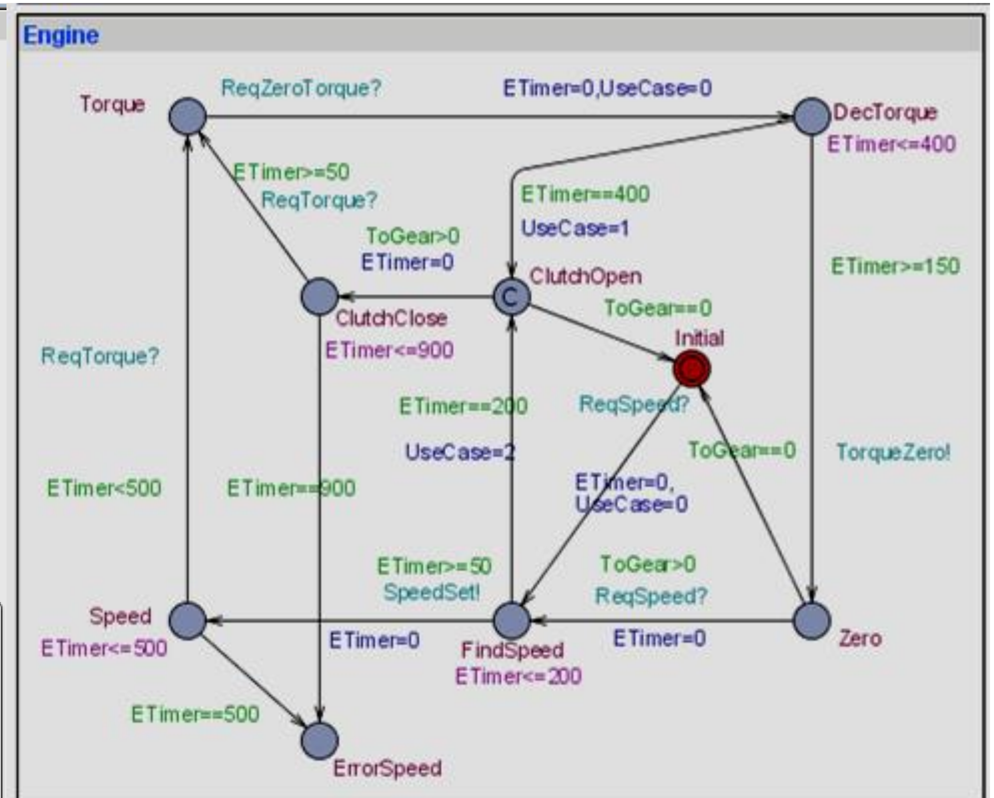


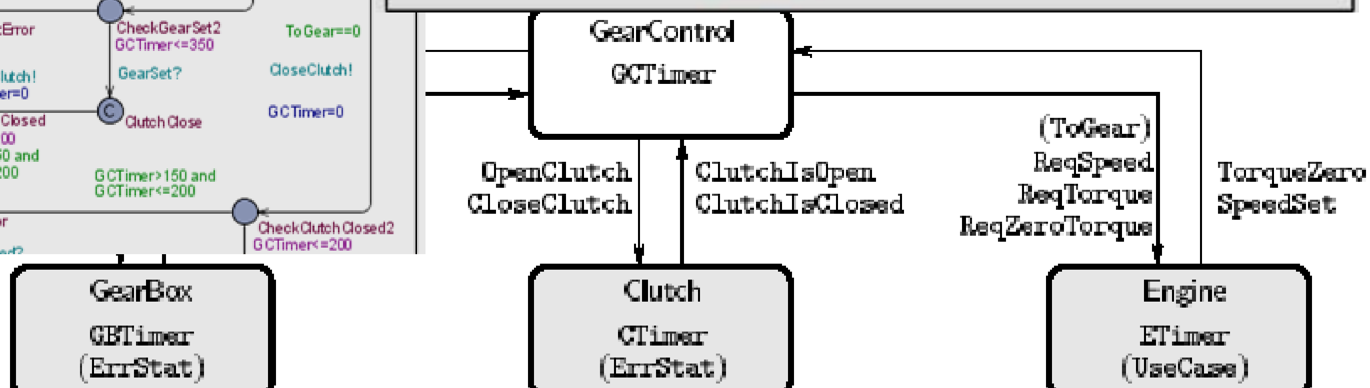
MECEL AB (1998)

Gear Controller

Lindahl, Pettersson, Yi 1998



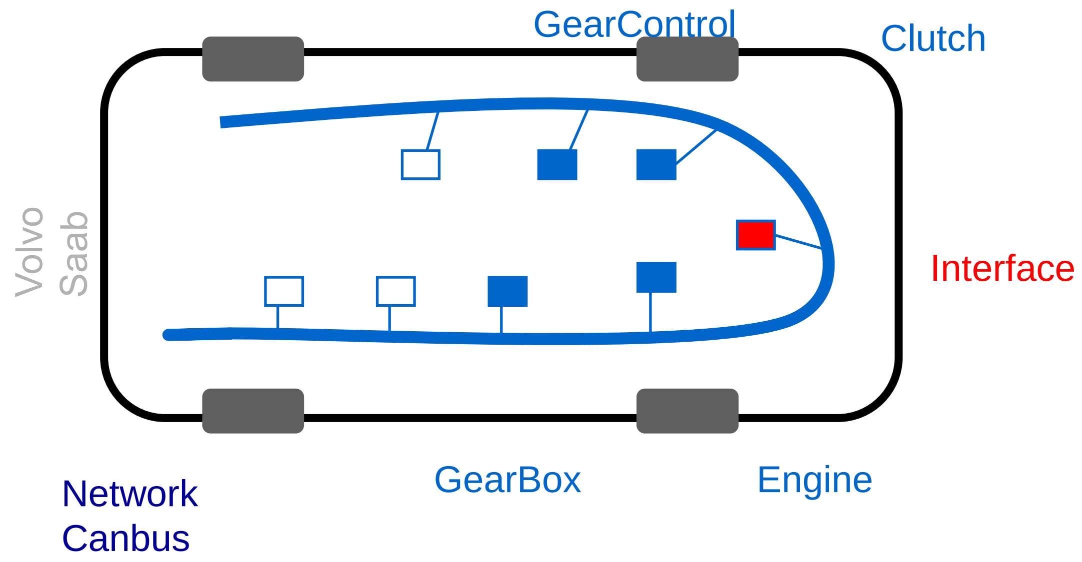




MECEL AB (1998)

Gear Controller

Lindahl, Pettersson, Yi 1998



$$\mathsf{GearControl@Initiate} \leadsto_{\leq 1500} ((\,\mathtt{ErrStat}=0\,) \Rightarrow \mathsf{GearControl@GearChanged}\,) \quad (1)$$

$$\mathsf{GearControl@Initiate} \leadsto_{\leq 1000} ((\,\mathtt{ErrStat}=0 \land \mathtt{UseCase}=0\,) \Rightarrow \mathsf{GearControl@GearChanged}\,) \quad (2)$$

$$\mathsf{Clutch@ErrorClose} \leadsto_{\leq 200} \mathsf{GearControl@CCloseError} \quad (3)$$

$$\mathsf{Clutch@ErrorOpen} \leadsto_{\leq 200} \mathsf{GearControl@COpenError} \quad (4)$$

$$\mathsf{GearBox@ErrorIdle} \leadsto_{\leq 350} \mathsf{GearControl@GSetError} \quad (5)$$

$$\mathsf{GearBox@ErrorNeu} \leadsto_{\leq 200} \mathsf{GearControl@GNeuError} \quad (6)$$

$$Inv\,(\,\mathsf{GearControl@CCloseError} \Rightarrow \mathsf{Clutch@ErrorClose}\,) \quad (7)$$

$$Inv\,(\,\mathsf{GearControl@COpenError} \Rightarrow \mathsf{Clutch@ErrorOpen}\,) \quad (8)$$

$$Inv\,(\,\mathsf{GearControl@GSetError} \Rightarrow \mathsf{GearBox@ErrorIdle}\,) \quad (9)$$

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]
- 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

Case Studies: Protocols

- Philips Audio Protocol [HS'95, CAV'95, RTSS'95, CAV'96]
- Bounded Retransmission Protocol [TACAS'97]
- Bang & Olufsen Audio/Video Protocol [RTSS'97]
- TDMA Protocol [PRFTS'97]
- Lip-Synchronization Protocol [FMICS'97]
- ATM ABR Protocol [CAV'99]
- ABB Fieldbus Protocol [ECRTS'2k]
- IEEE 1394 Firewire Root Contention (2000)
- Distributed Agreement Protocol [Formats05]
- Leader Election for Mobile Ad Hoc Networks [Charme05]

- Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal, 2006
- Formalizing SHIM6, a Proposed Internet Standard in UPPAAL, 2007
- Verifying the distributed real-time network protocol RTnet using Uppaal, 2007
- **Analysis of the Zeroconf protocol using UPPAAL, 2009**
- Analysis of a Clock Synchronization Protocol for Wireless Sensor Networks, 2009
- **Model Checking the FlexRay Physical Layer Protocol, 2010**

Using UPPAAL as 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

RELATED SITES: TIMES | UPPAAL CORA | UPPAAL TRON

UPPAAL

Home

Home | About | Documentation | Download | Examples | 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.

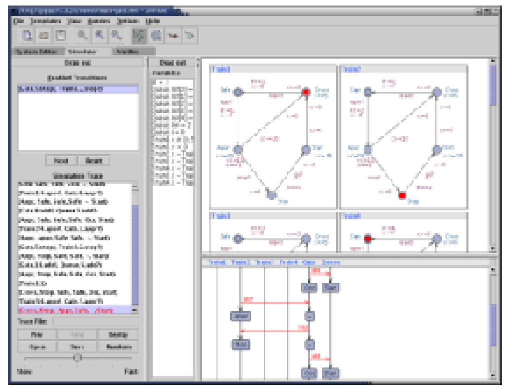


Figure 1: UPPAAL on screen.

Download

The current official release is UPPAAL 3.4.11 (Jun 23, 2005). A release of UPPAAL **3.6 alpha 3** (dec 20, 2005) is also available. For more information about UPPAAL version 3.4, we refer to this press release.

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

License

The UPPAAL tool is **free** for non-profit applications. For information about commercial licenses, please email sales(at)uppaal(dot)com.

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 forum page. Bugs should be reported using the bug tracking system. To email the development team directly, please use uppaal(at)list(dot)it(dot)uu(dot)se.

LAB-Exercises

http://people.cs.aau.dk/~kgl/SSFT2015/

Exercise 1 (Brick Sorter)

Excercise 19 (Train Crossing)

Exercise 2 (Coffee Machine)

Exercise 28 (Jobshop Scheduling)