

Model-based

Verification, Optimization, Synthesis and Performance Evaluation of Real-Time Systems

Alexandre David



Kim G. Larsen



Aalborg University, DENMARK



Timed Automata

.. and Prices, Games, Probabilities

Alexandre David



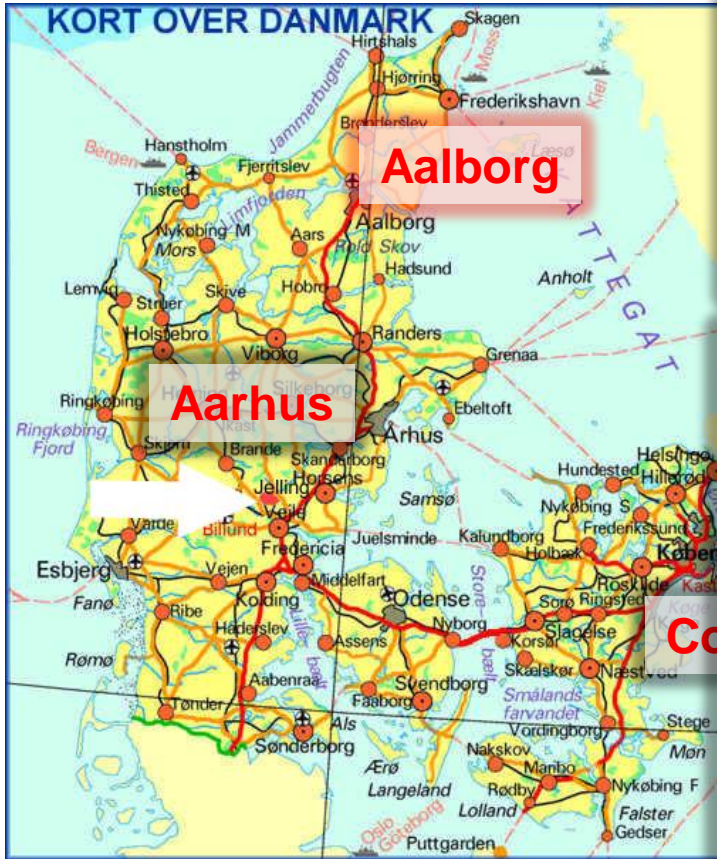
Kim G. Larsen



Aalborg University, DENMARK



Aalborg



Aalborg University lea
publ

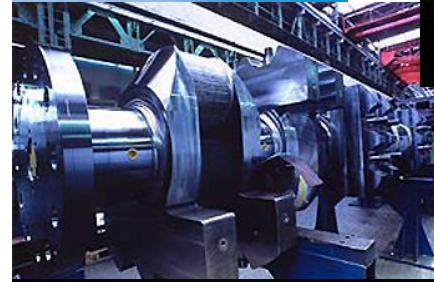
Jomfru Ane Gade




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




IDEA⁴CPS Foundations for CPS

I  Inst. of Software Chinese Academy of Sciences,
Beijing, China



D  Technical University of Denmark,
Lyngby, Denmark

E  East China Normal University,
Shanghai, China

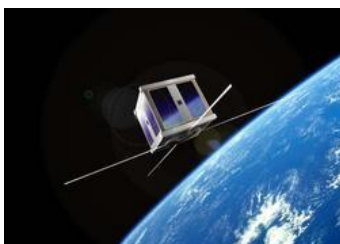
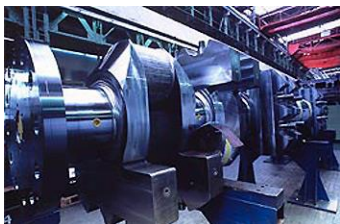
A  Aalborg University,
Denmark



(2011–2014) + (2014–2017)



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:

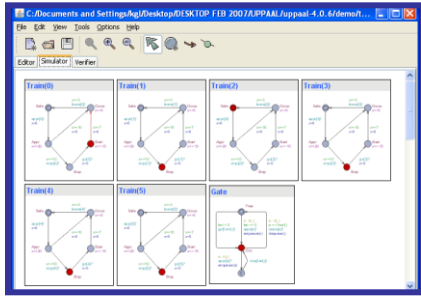
- correct
- predicable
- dependable
- fault tolerant
- resource minimal
- cheap



Model-Based Development



QUANTITATIVE Model Checking



System Description



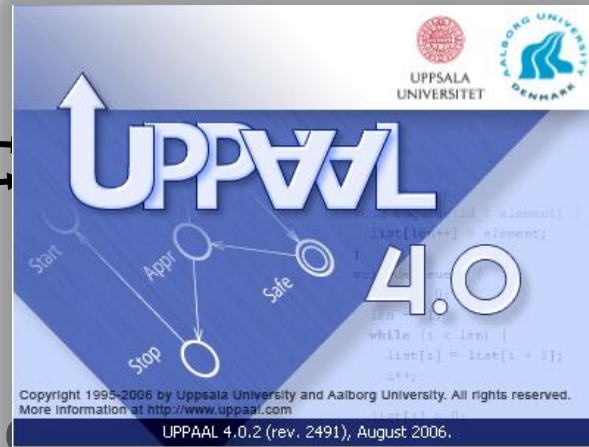
Time



Cost



Probability



Requirement

$$A \square (\text{req} \Rightarrow A \diamond \text{grant})$$

$$A \square (\text{req} \Rightarrow A \diamond_{t < 30s} \text{grant})$$

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

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

No!

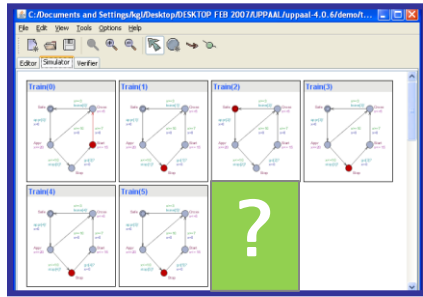
Debugging Information

Yes

Prototypes
Executable Code
Test sequences



Synthesis



System Description



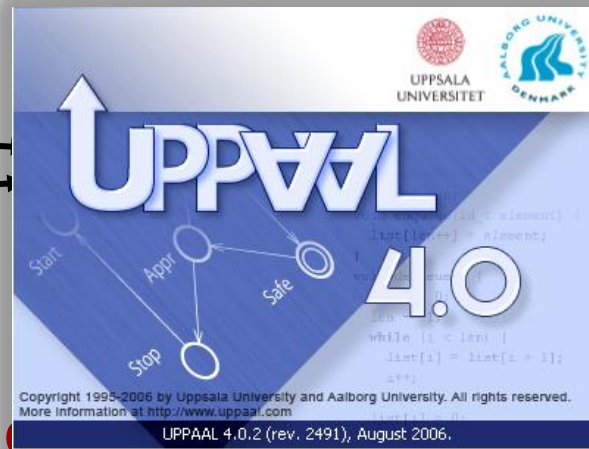
Time



Cost



Probability



No!

Debugging Information

Yes

Control Strategy

Requirement

$$A \square (\text{req} \Rightarrow A \diamond \text{grant})$$

$$A \square (\text{req} \Rightarrow A \diamond_{t < 30s} \text{grant})$$

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

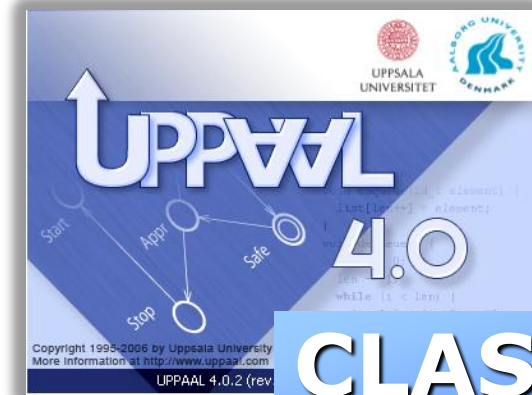
$$A \square (\text{req} \Rightarrow A \diamond_{t < 30s, p > 0.90} \text{grant})$$



Overview

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

(Lecture+Exercise)⁴



CLASSIC

CORA

TIGA

ECDAR

SMC

TRON

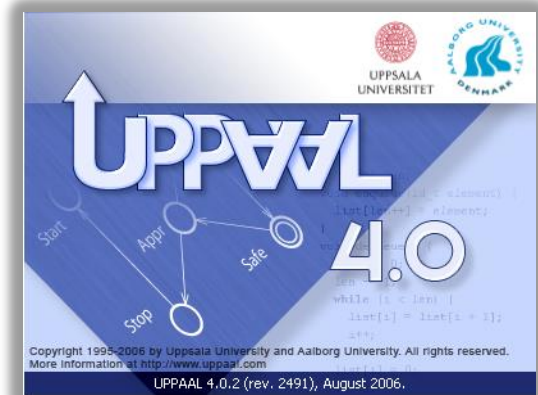




Model-Based **Verification**, **Optimization**, **Synthesis** and **Performance Evaluation** of **Real-Time Systems** using UPPAAL



Alexandre David & Kim Guldstrand Larsen
CISS, Aalborg University, DENMARK



www.uppaal.org



Timed Automata



UPPAAL (1995–)

@AALborg

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



@UPPSala

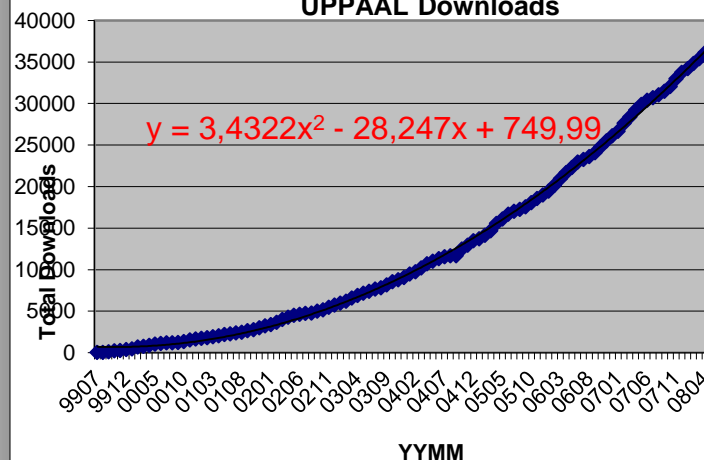
- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun



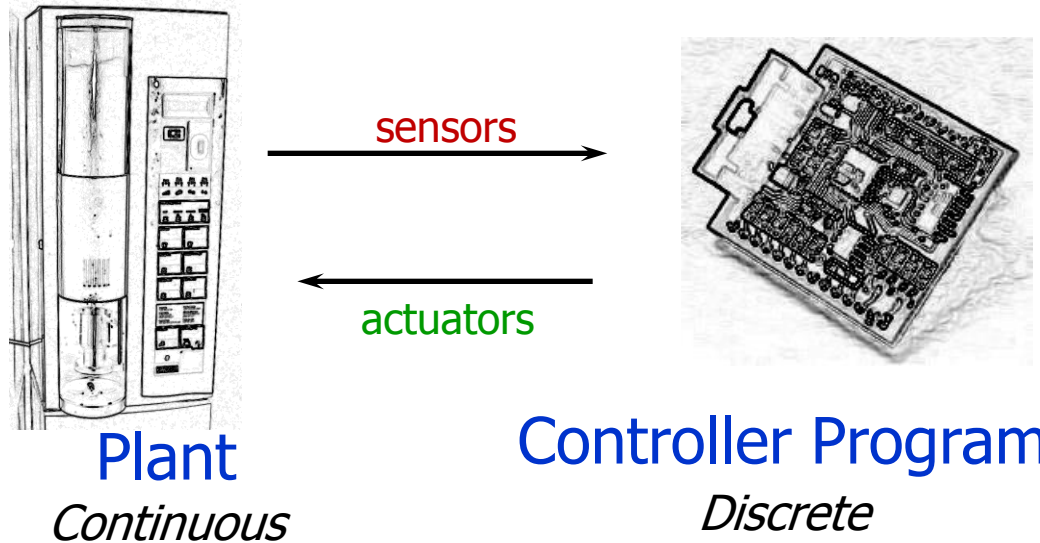
@Elsewhere

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, Jan Tretmans, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen,, 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 Downloads



Real Time Systems



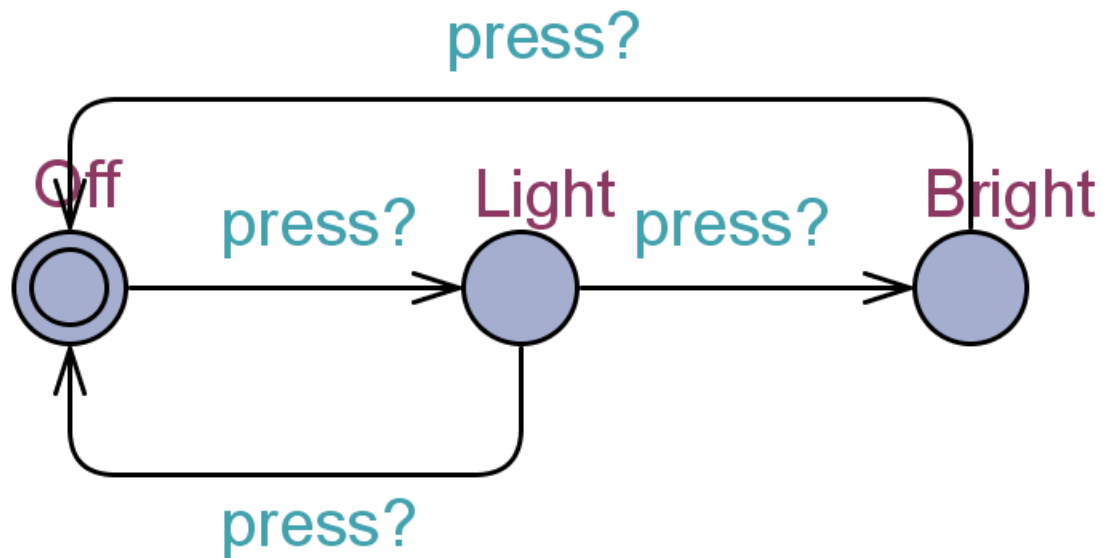
- Eg.:
- Realtime Protocols
 - Pump Control
 - Air Bags
 - Robots
 - Cruise Control
 - ABS
 - CD Players
 - Production Lines

Real Time System

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

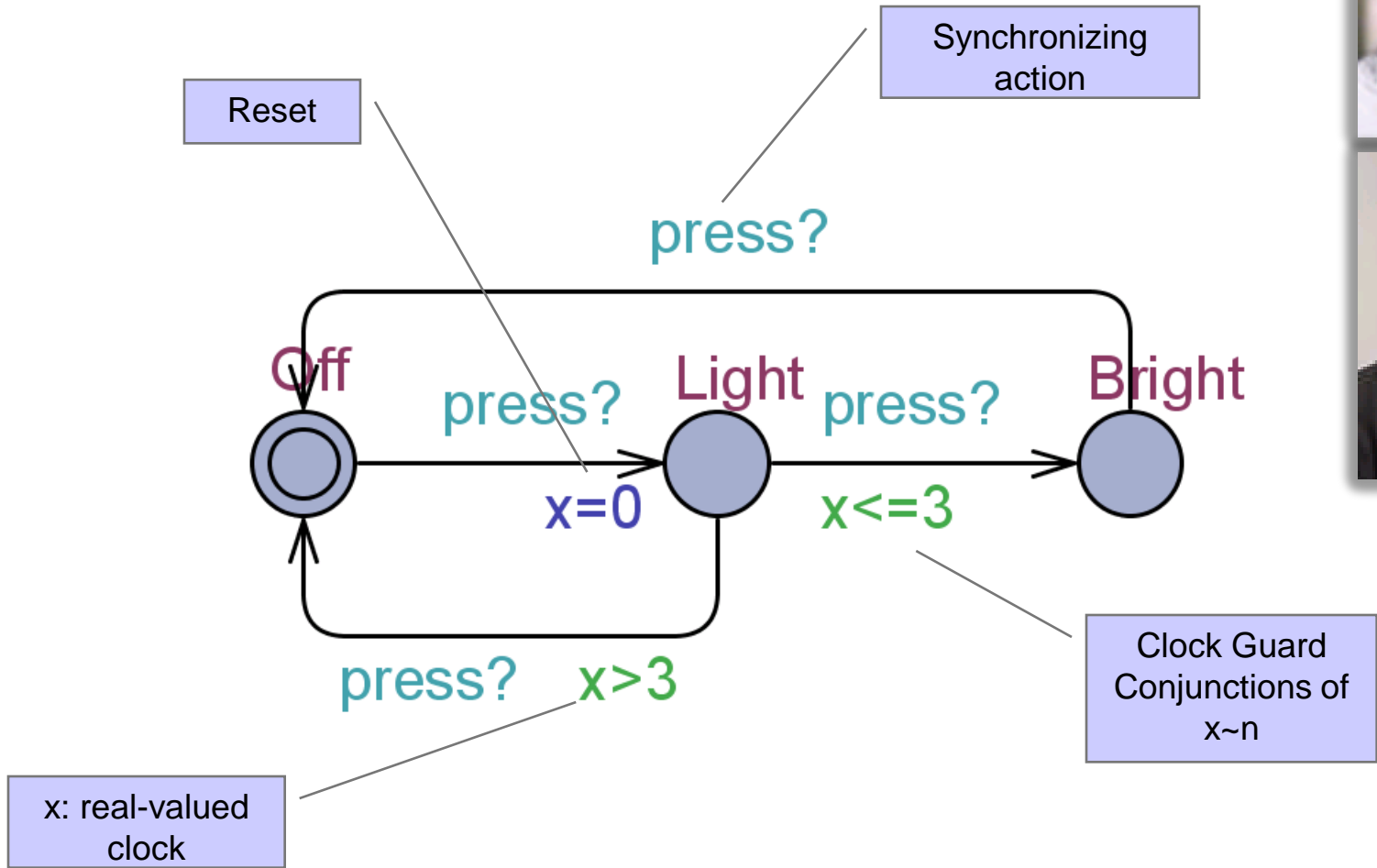
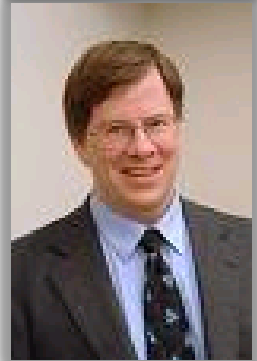


A Dumb Light Controller



Timed Automata

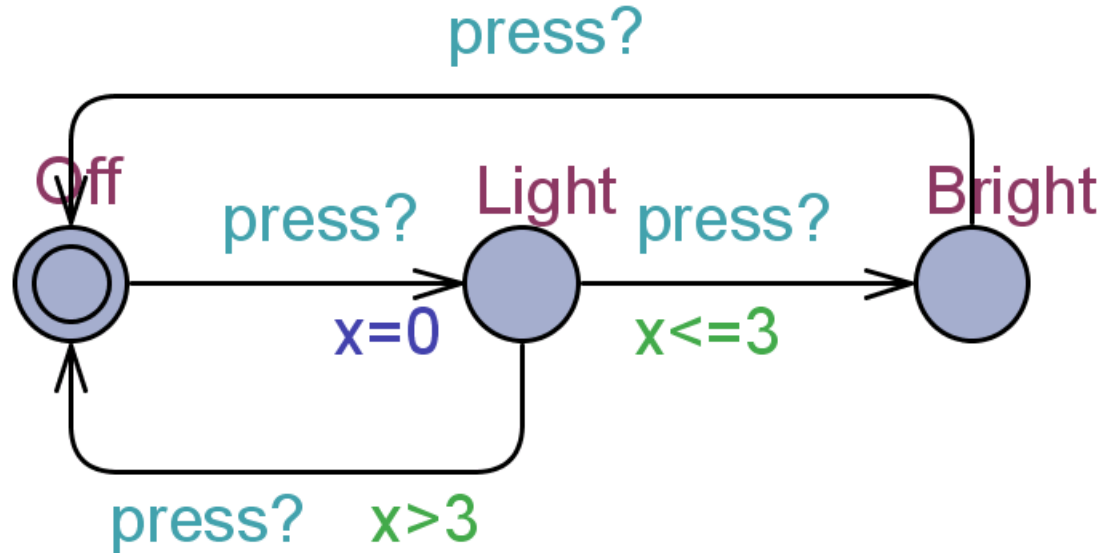
[Alur & Dill'89]



ADD a clock x



A Timed Automata (Semantics)



States:

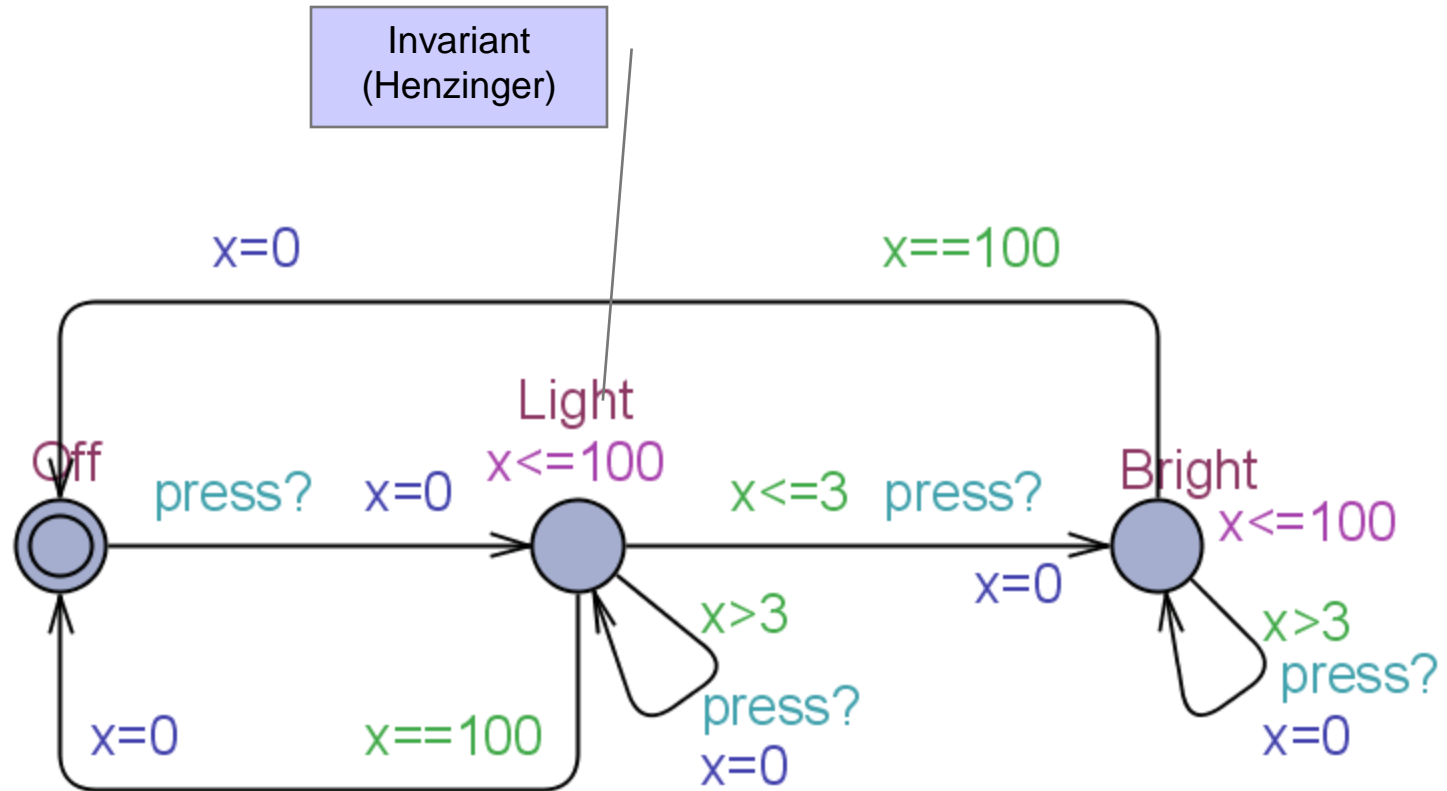
(location , $x=v$) where $v \in \mathbf{R}$

Transitions:

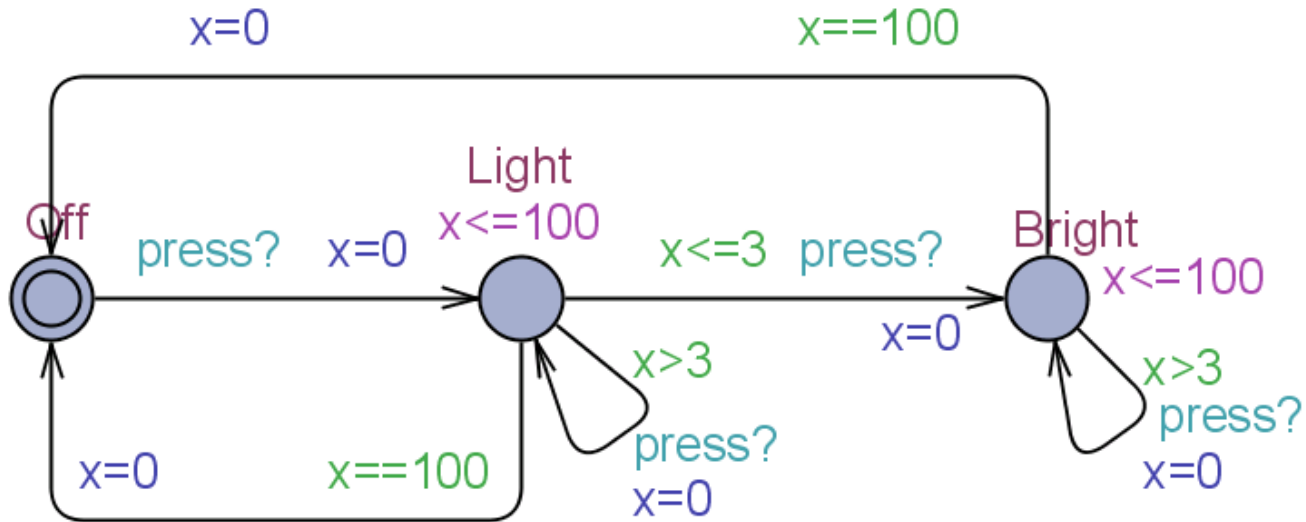
	(Off , $x=0$)
delay 4.32	\rightarrow (Off , $x=4.32$)
press?	\rightarrow (Light , $x=0$)
delay 2.51	\rightarrow (Light , $x=2.51$)
press?	\rightarrow (Bright , $x=2.51$)



Intelligent Light Controller



Intelligent Light Controller



Transitions:

	$(\text{Off} , x=0)$
delay 4.32	$\rightarrow (\text{Off} , x=4.32)$
<code>press?</code>	$\rightarrow (\text{Light} , x=0)$
delay 4.51	$\rightarrow (\text{Light} , x=4.51)$
<code>press?</code>	$\rightarrow (\text{Light} , x=0)$
delay 100	$\rightarrow (\text{Light} , x=100)$
τ	$\rightarrow (\text{Off} , x=0)$

Note:

X
 $(\text{Light} , x=0)$ delay 103 \rightarrow

Invariants ensures progress



Timed Automata (formally)

Constraints

Definition

Let X be a set of clock variables. The set $\mathcal{B}(X)$ of *clock constraints* ϕ is given by the grammar:

$$\phi ::= x \leq c \mid c \leq x \mid x < c \mid c < x \mid \phi_1 \wedge \phi_2$$

where $c \in \mathbb{N}$ (or \mathbb{Q}).



Timed Automata (formally)

Clock Valuations and Notation

Definition

The set of *clock valuations*, \mathbb{R}^C is the set of functions $C \rightarrow \mathbb{R}_{\geq 0}$ ranged over by u, v, w, \dots

Notation

Let $u \in \mathbb{R}^C$, $r \subseteq C$, $d \in \mathbb{R}_{\geq 0}$, and $g \in \mathcal{B}(X)$ then:

- $u + d \in \mathbb{R}^C$ is defined by $(u + d)(x) = u(x) + d$ for any clock x
- $u[r] \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \notin r$.
- $u \models g$ denotes that g is satisfied by u .



Timed Automata (formally)

Timed Automata

Definition

A timed automaton A over clocks C and actions Act is a tuple (L, l_0, E, I) , where:

- L is a finite set of locations
- $l_0 \in L$ is the initial location
- $E \subseteq L \times \mathcal{B}(X) \times Act \times \mathcal{P}(C) \times L$ is the set of edges
- $I : L \rightarrow \mathcal{B}(X)$ assigns to each location an invariant



Timed Automata (formally)

Semantics

Definition

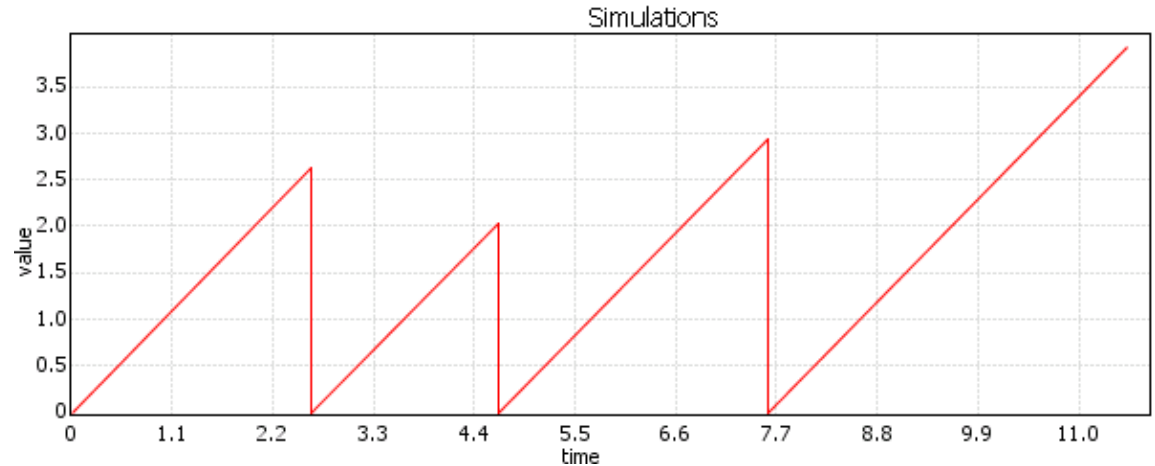
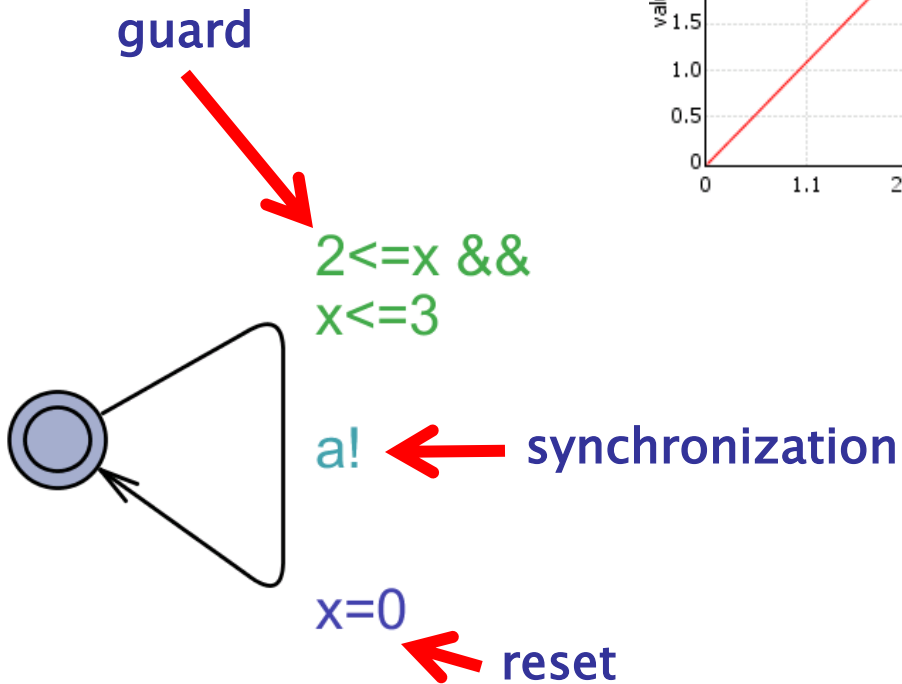
The semantics of a timed automaton A is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state $(l_0, u_0)^*$ and with the following transitions:

- $(l, u) \xrightarrow{\epsilon(d)} (l, u + d)$ iff $u \in I(l)$ and $u + d \in I(l)$,
- $(l, u) \xrightarrow{a} (l', u')$ iff there exists $(l, g, a, r, l') \in E$ such that
 - $u \models g$,
 - $u' = u[r]$, and
 - $u' \in I(l')$

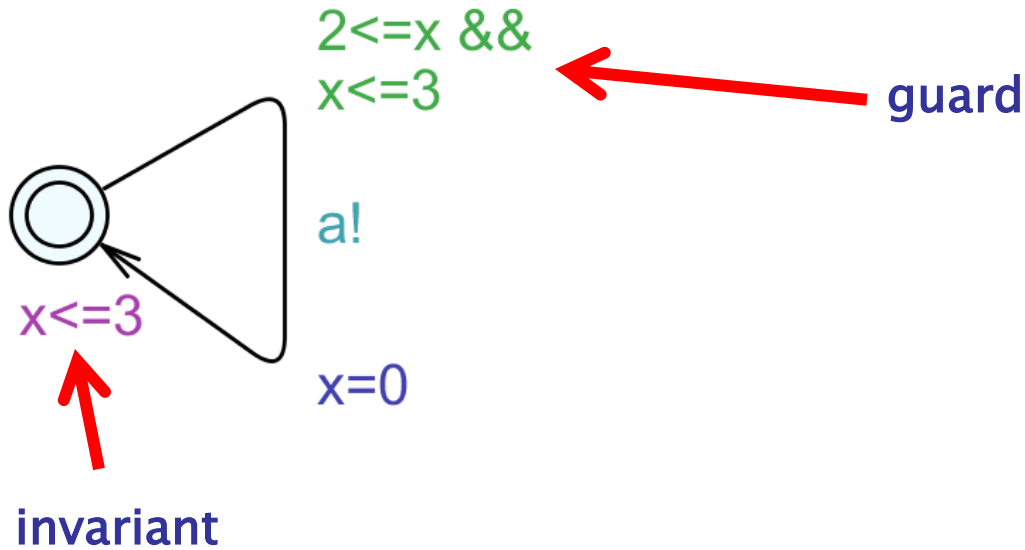
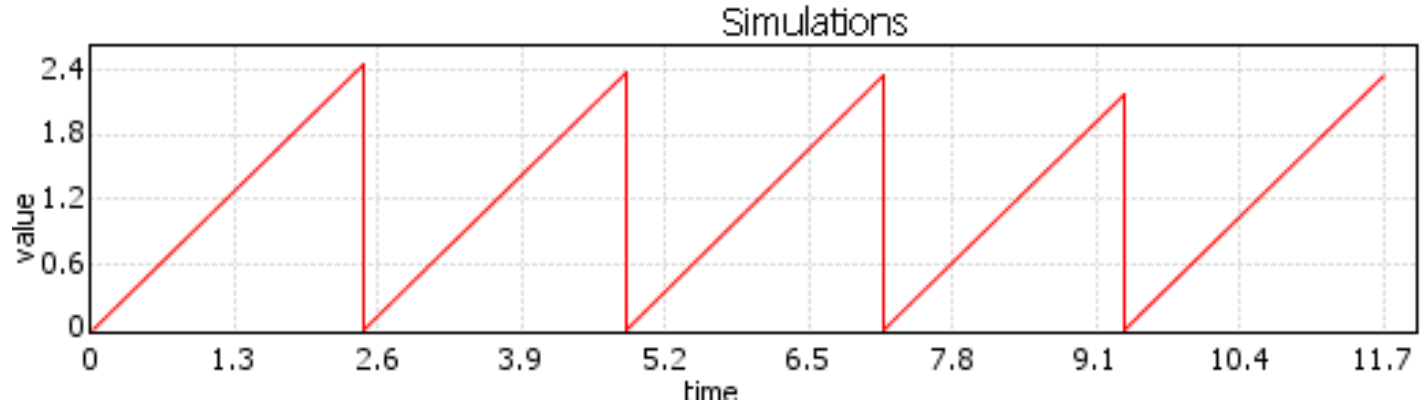
* $u_0(x) = 0$ for all $x \in C$



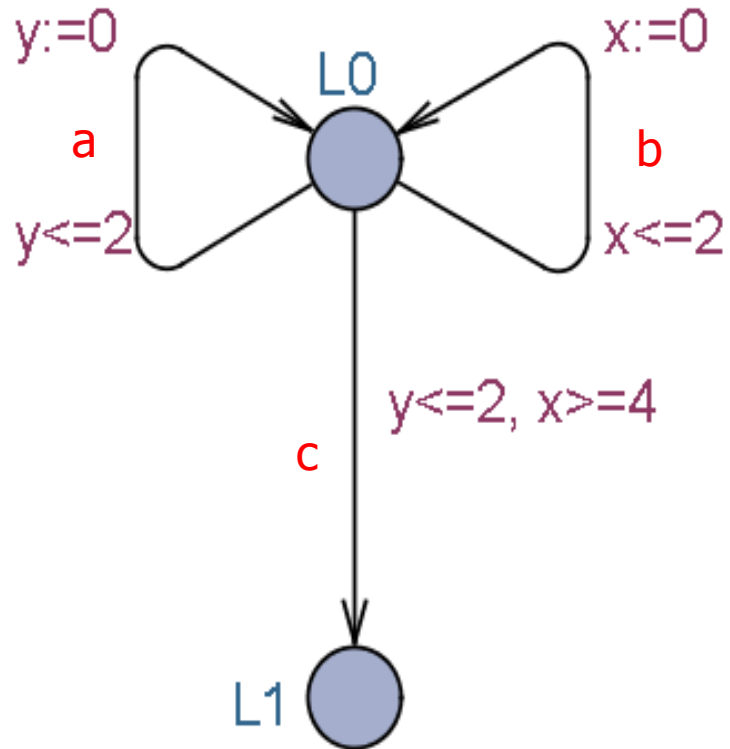
Timed Automata: Example



Timed Automata: Example



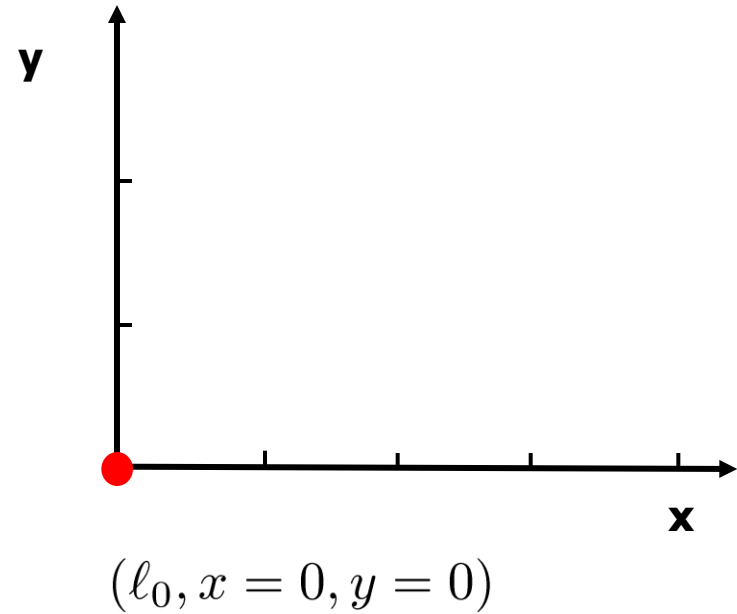
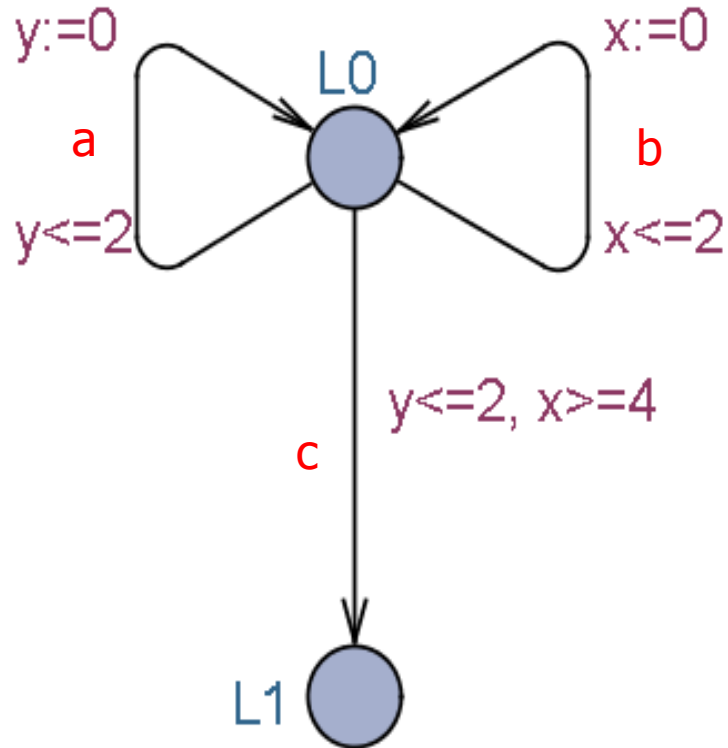
Example



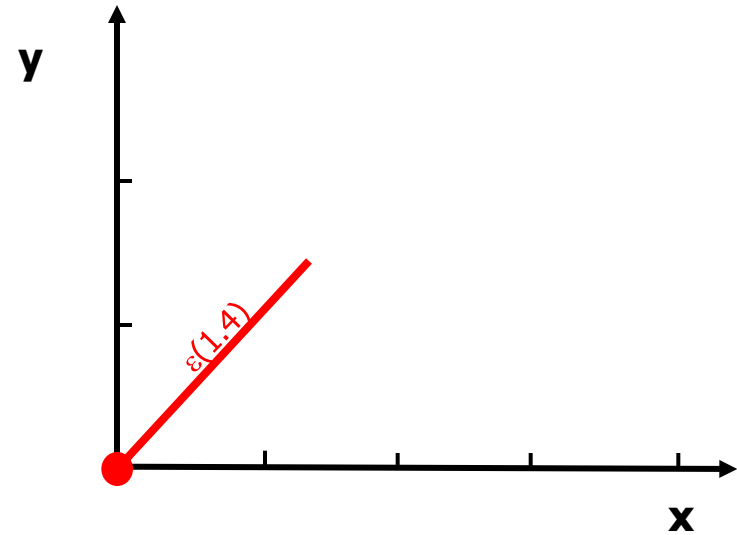
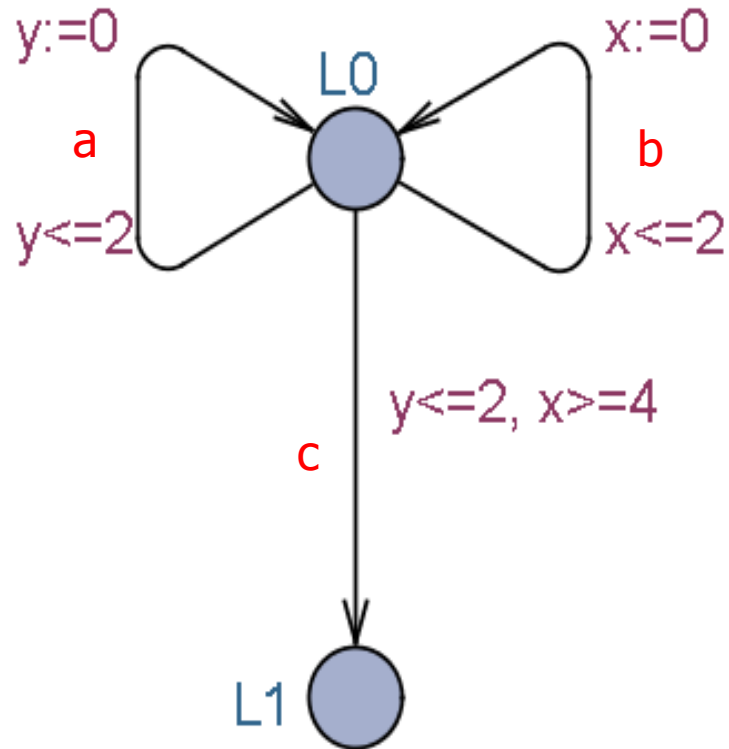
Is L1 reachable ?



Example



Example

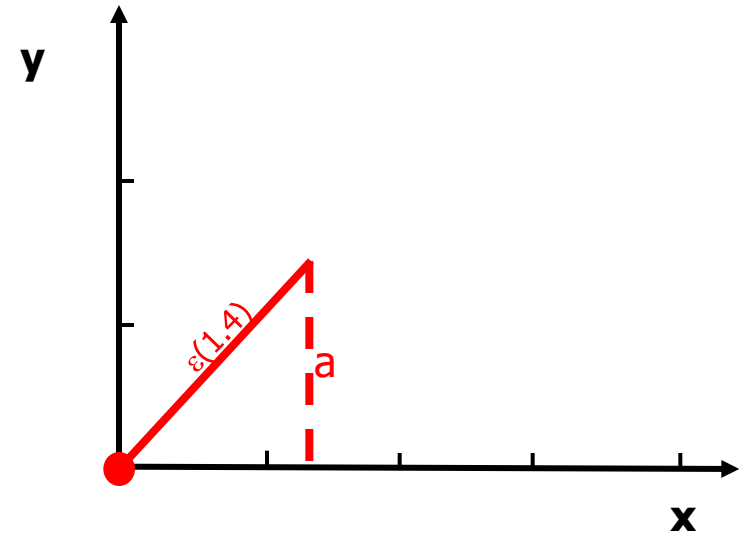
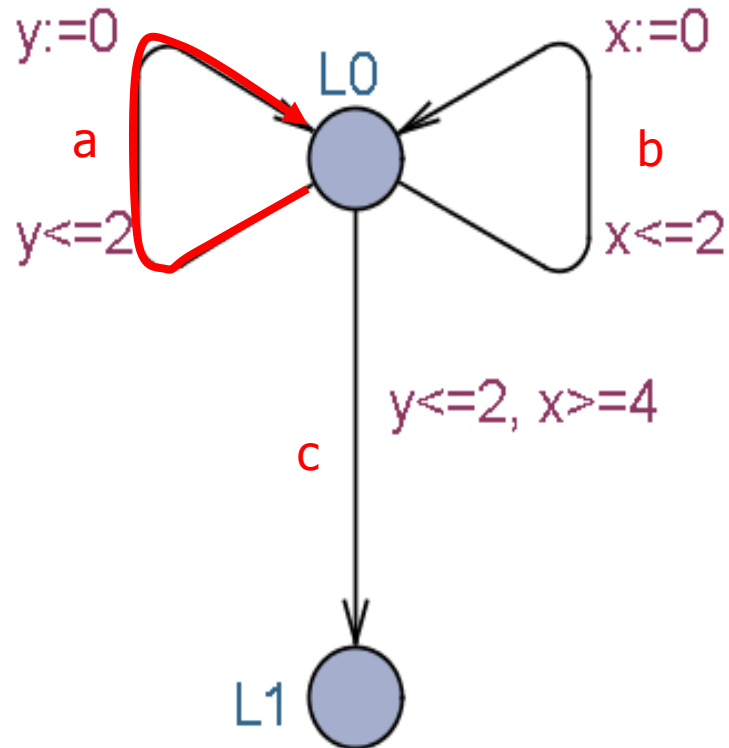


$$(\ell_0, x = 0, y = 0)$$

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



Example



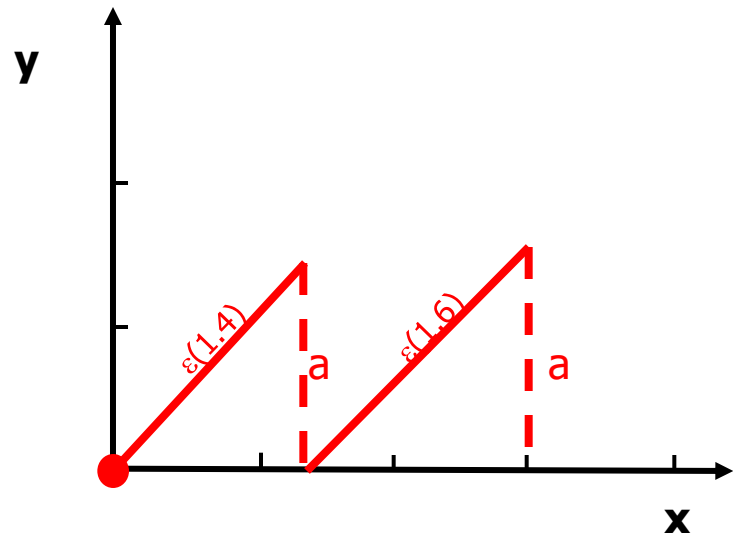
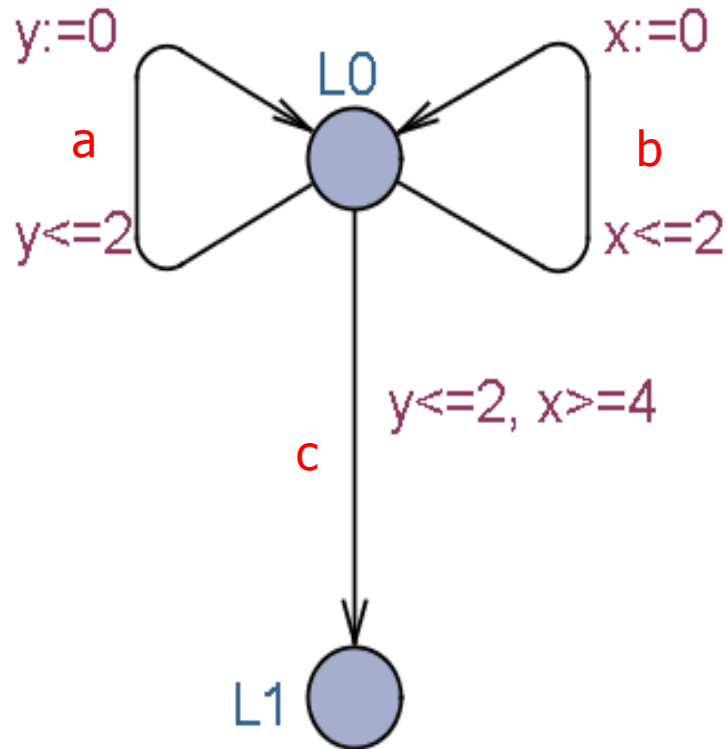
$$(\ell_0, x = 0, y = 0)$$

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

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



Example



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

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

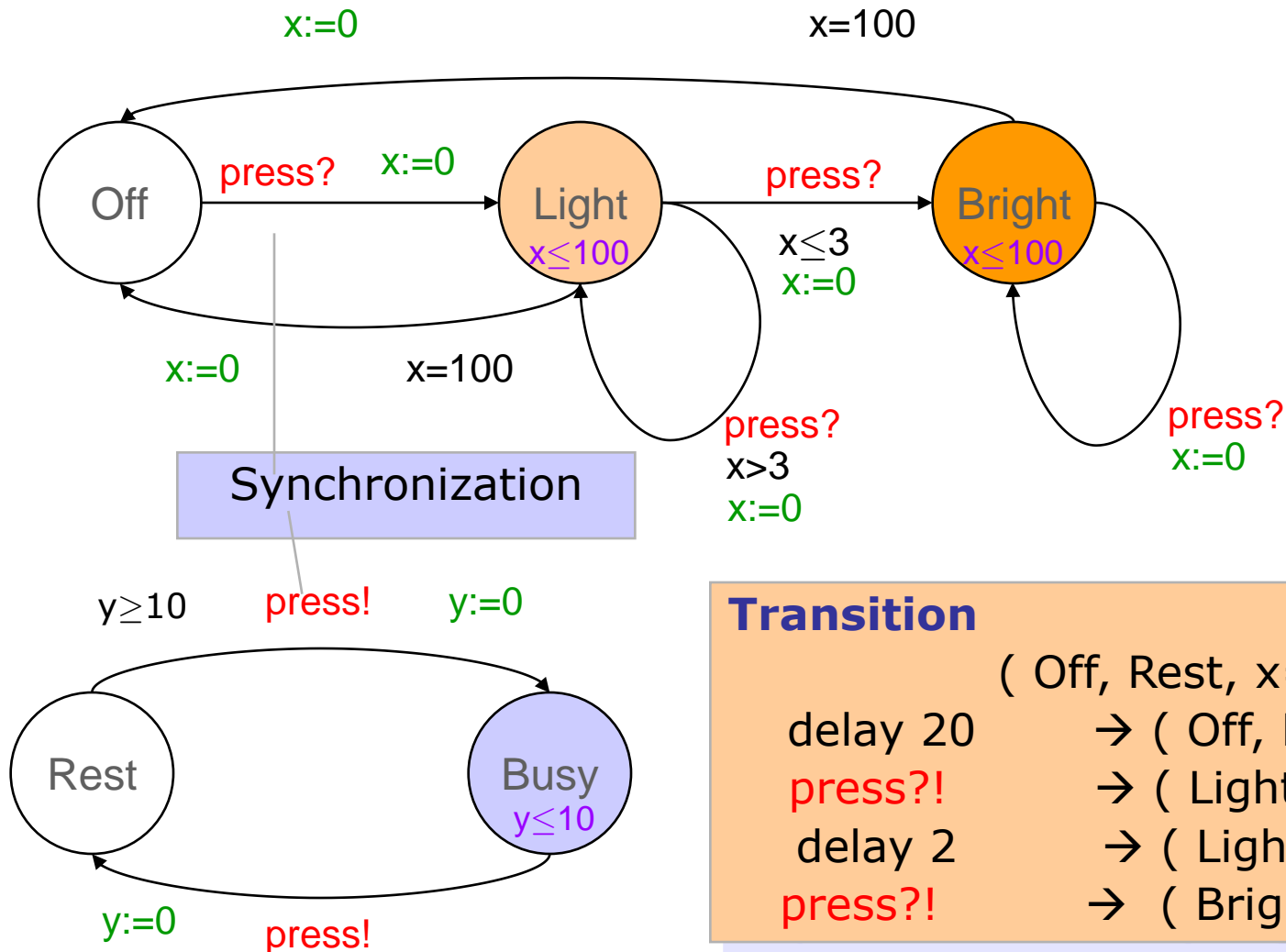
$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

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

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



Networks Light Controller & User



Network Semantics

$$T_1 \parallel_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \parallel_X s_0^2) \quad \text{where}$$

$$\frac{S_1 \xrightarrow{\mu} S_1'}{S_1 \parallel_X S_2 \xrightarrow{\mu} S_1' \parallel_X S_2}$$

$$\frac{S_2 \xrightarrow{\mu} S_2'}{S_1 \parallel_X S_2 \xrightarrow{\mu} S_1 \parallel_X S_2'}$$

$$\frac{S_1 \xrightarrow{a!} S_1' \quad S_2 \xrightarrow{a?} S_2'}{S_1 \parallel_X S_2 \xrightarrow{\tau} S_1' \parallel_X S_2'}$$

$$\frac{S_1 \xrightarrow{e(d)} S_1' \quad S_2 \xrightarrow{e(d)} S_2'}{S_1 \parallel_X S_2 \xrightarrow{e(d)} S_1' \parallel_X S_2'}$$



Network Semantics

(URGENT synchronization)

+ Urgent synchronization

$$T_1 \parallel_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \parallel_X s_0^2) \quad \text{where}$$

$$\frac{s_1 \xrightarrow{\mu} s_1'}{s_1 \parallel_X s_2 \xrightarrow{\mu} s_1' \parallel_X s_2}$$

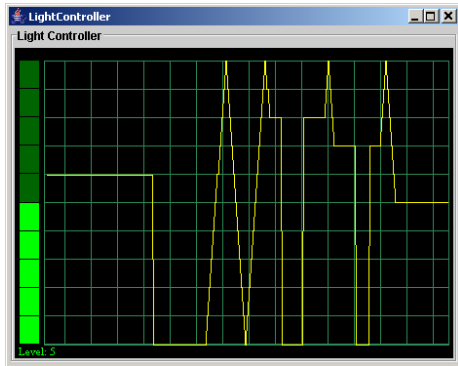
$$\frac{s_2 \xrightarrow{\mu} s_2'}{s_1 \parallel_X s_2 \xrightarrow{\mu} s_1 \parallel_X s_2'}$$

$$\frac{s_1 \xrightarrow{a!} s_1' \quad s_2 \xrightarrow{a?} s_2'}{s_1 \parallel_X s_2 \xrightarrow{\tau} s_1' \parallel_X s_2'}$$

$\forall d' < d, \forall u \in \text{UAct}: \\ \neg (s_1 \xrightarrow{e(d') u?} \rightarrow \wedge s_2 \xrightarrow{e(d') u!} \rightarrow)$

$$\frac{s_1 \xrightarrow{e(d)} s_1' \quad s_2 \xrightarrow{e(d)} s_2'}{s_1 \parallel_X s_2 \xrightarrow{e(d)} s_1' \parallel_X s_2'}$$





Light Control Interface



Light Control Interface

press? 0.2 release? ... press? 0.7 release? ... press? 1.0 2.4 release? ...
 ↓ ↓ ↓ ↓ ↓
 ∅ touch! starthold! endhold!



User

Interface



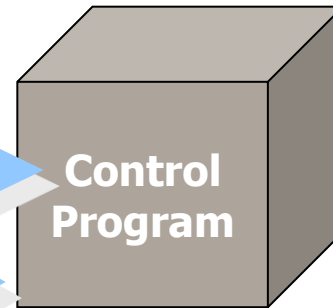
press?

release?

touch!

starthold!

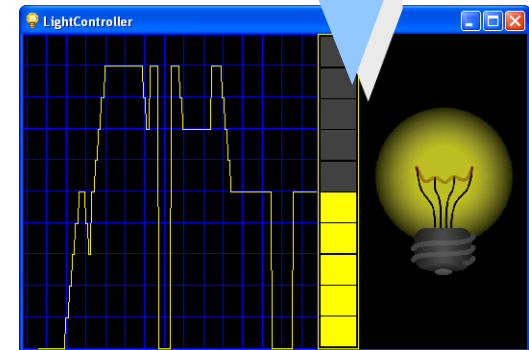
endhold!



Control Program

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

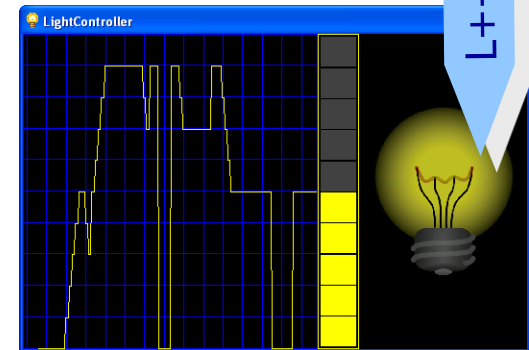
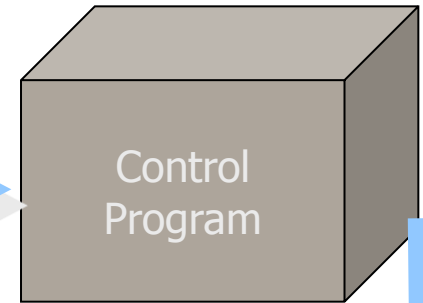
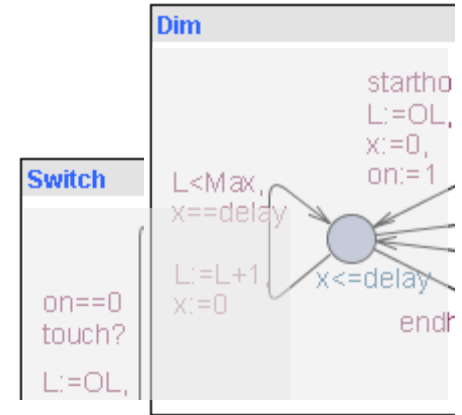
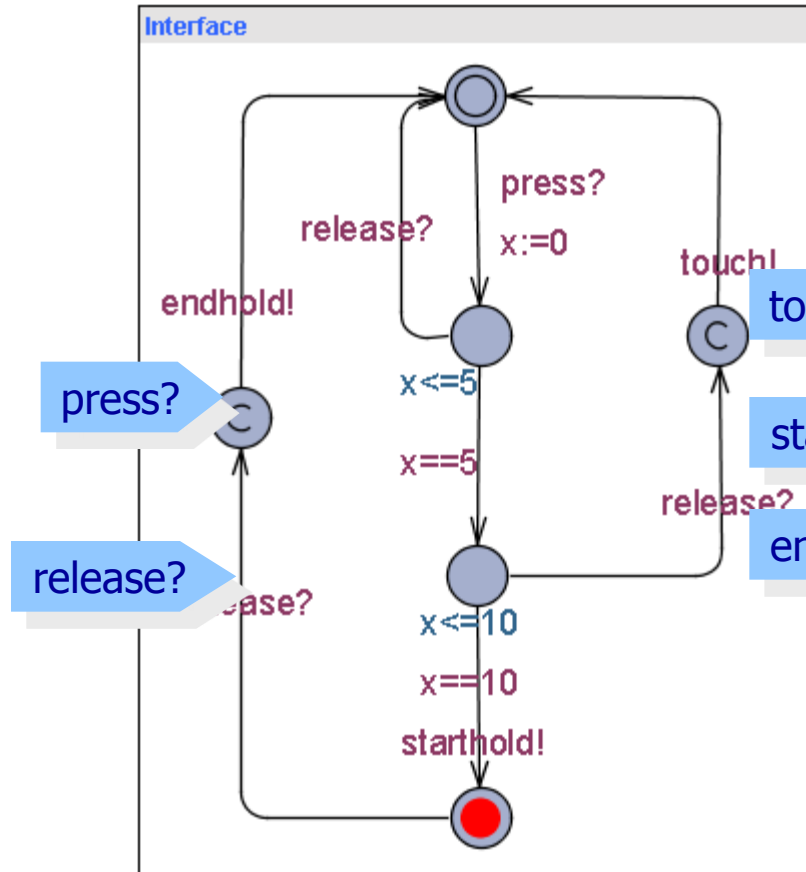
press? d release? → touch! $0.5 \leq d \leq 1$
 press? 1 → starthold!
 press? d release? → endhold! $d > 1$



Light Control Interface

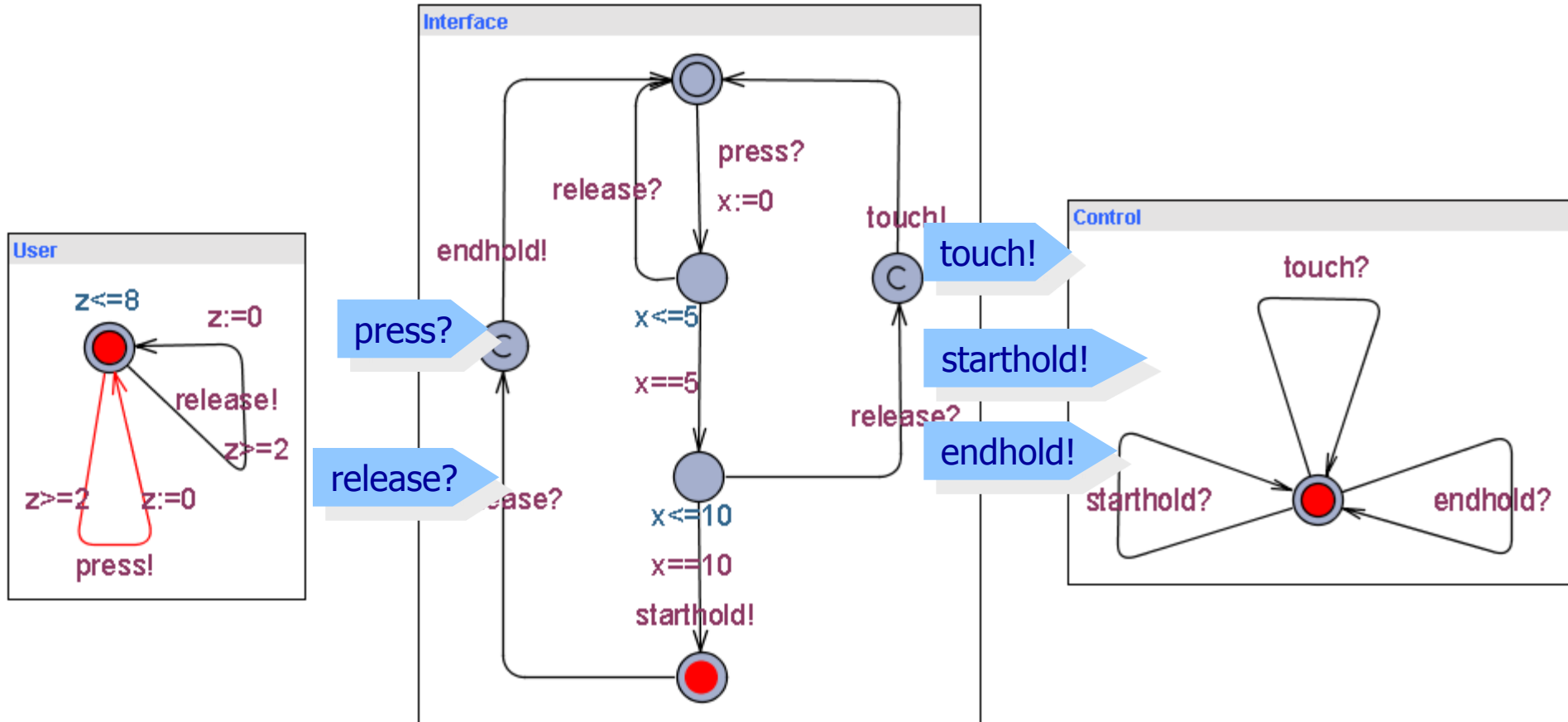


User

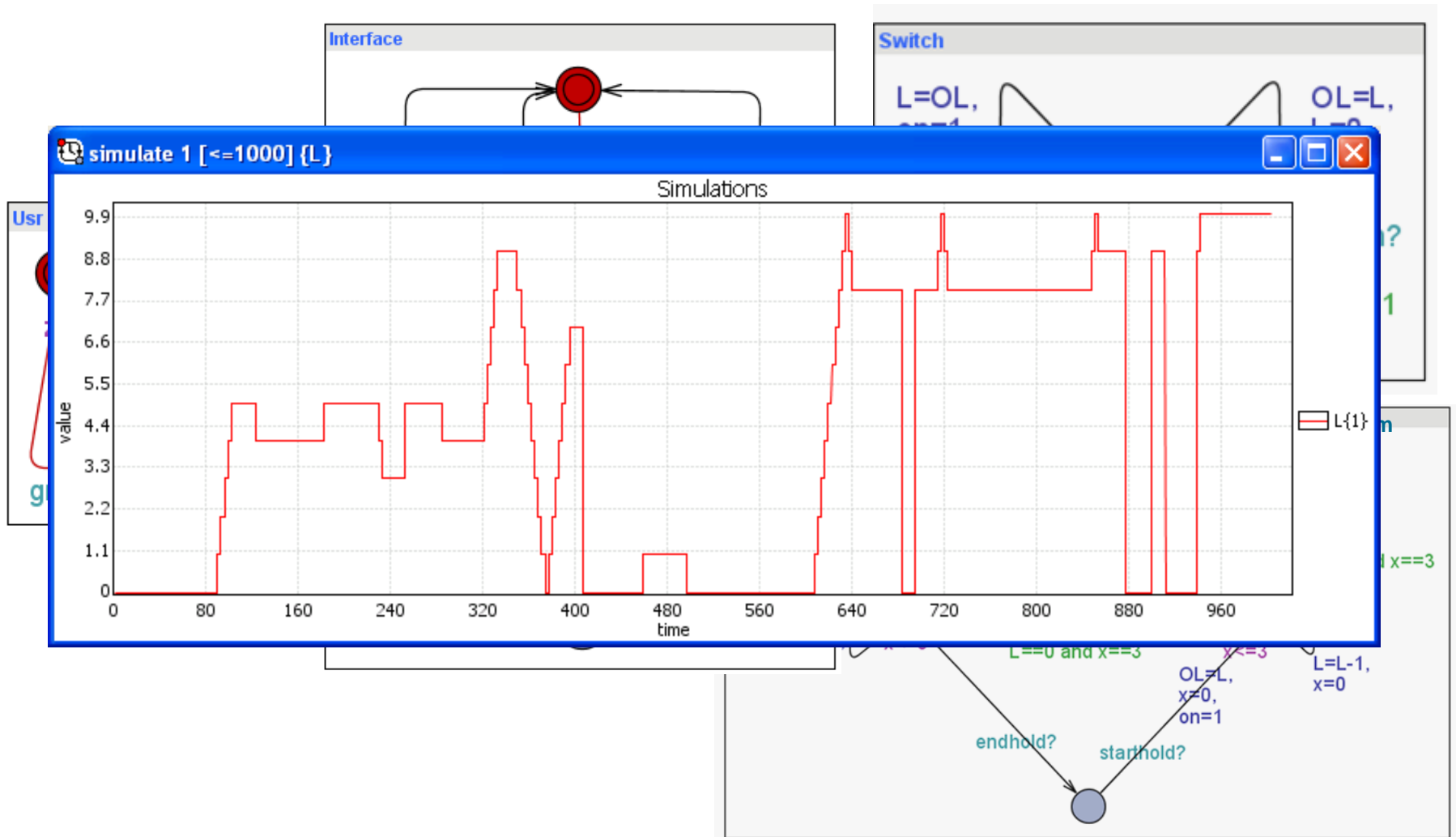


L++/L-/L:=0

Light Control Network



Full Light Controller

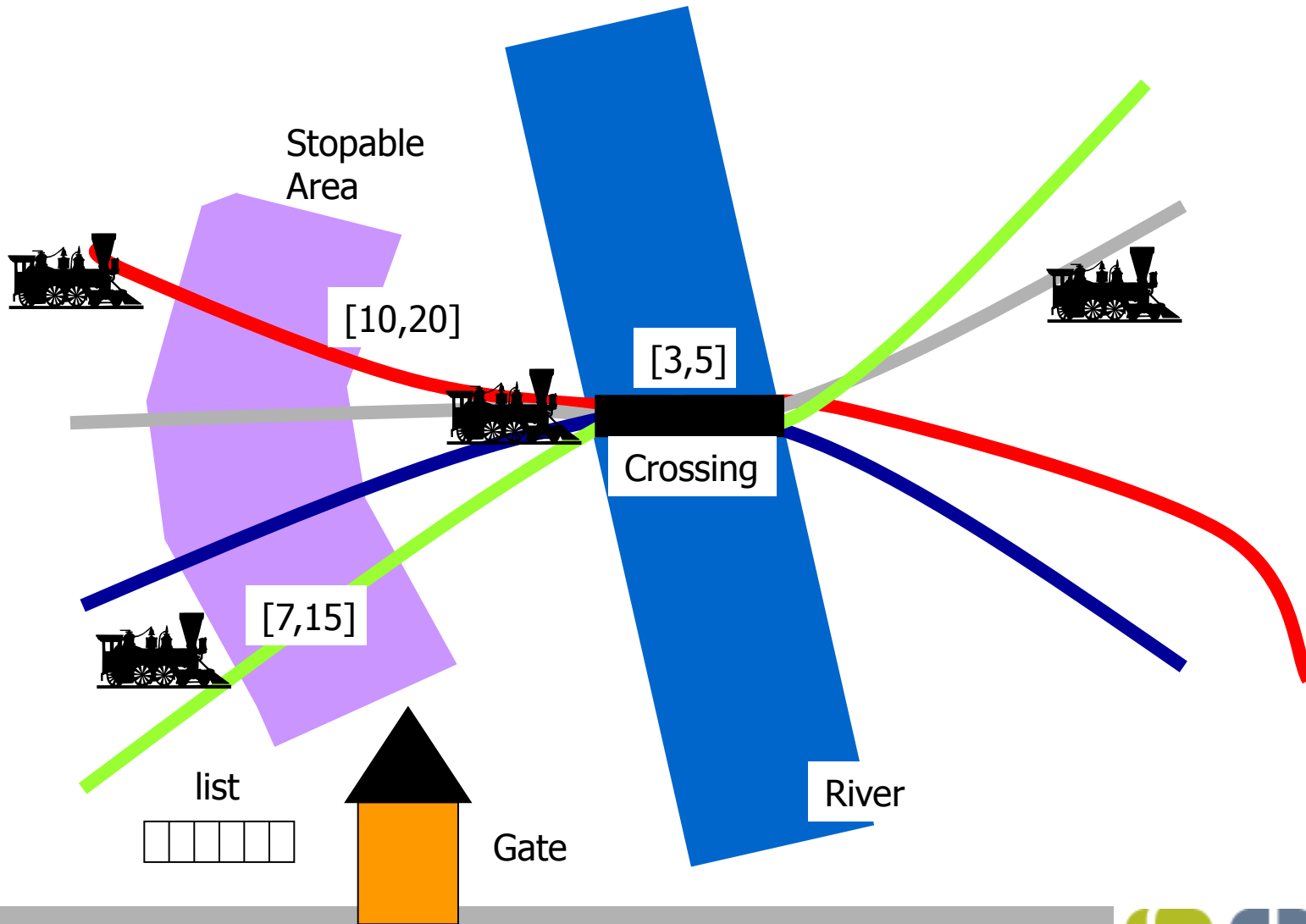


UPPAAL

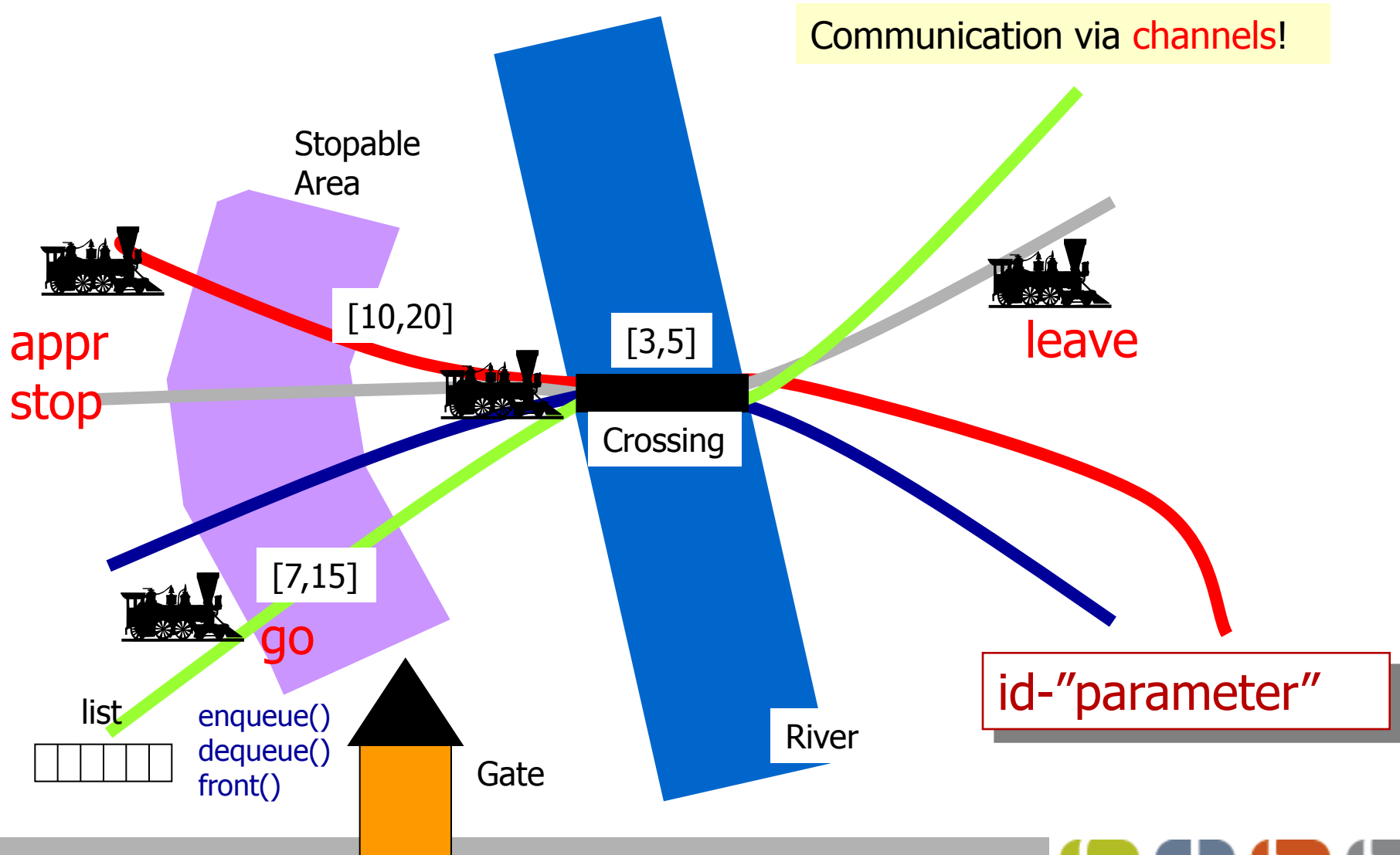
Modeling & Specification



Train Crossing



Train Crossing



Declarations

The screenshot shows the UPPAAL System Editor interface. The left pane displays a project tree for 'train-gate' with the following structure:

- train-gate
 - Global declarations
 - Train
 - Gate
 - IntQueue
 - Process assignments
 - System definition

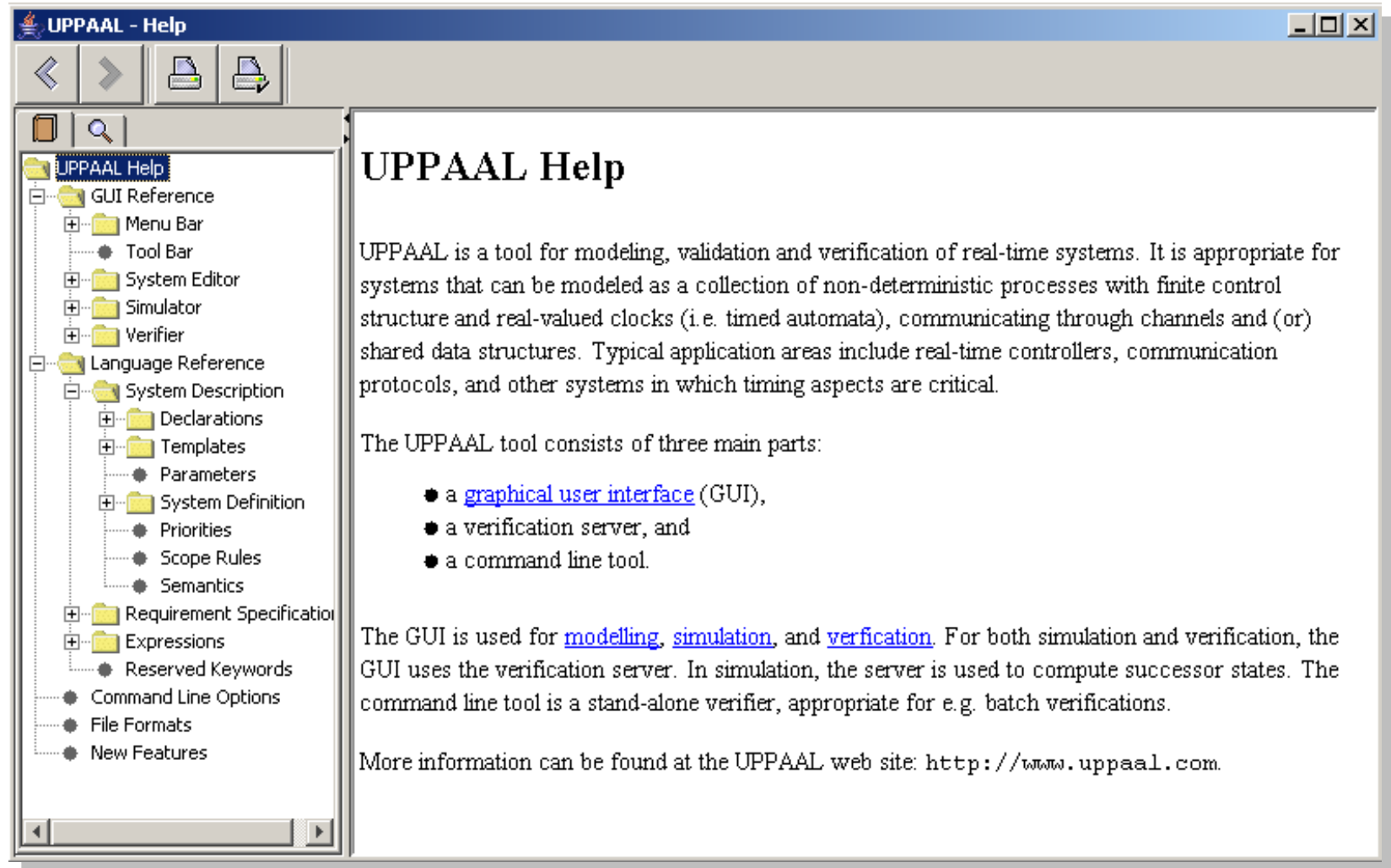
The main editor displays the following code:

```
/*  
 * For more details about this example, see  
 * "Automatic Verification of Real-Time Communicating Systems by Constraint Solving",  
 * by Wang Yi, Paul Pettersson and Mats Daniels. In Proceedings of the 7th International  
 * Conference on Formal Description Techniques, pages 223-238, North-Holland. 1994.  
 */  
  
const N      5;           // # trains + 1  
int[0,N]    el;  
chan        appr, stop, go, leave;  
chan        empty, notempty, hd, add, rem;  
  
clock x;  
  
int[0,N] list[N], len, i;  
  
Train1:=Train(el, 1);  
Train2:=Train(el, 2);  
Train3:=Train(el, 3);  
Train4:=Train(el, 4);  
  
system  
    Train1, Train2, Train3, Train4,  
    Gate, Queue;
```

On the right side of the image, a blue-bordered box lists the following concepts:

- Constants
- Bounded integers
- Channels
- Clocks
- Arrays
- Types
- Functions
- Templates
- Processes
- Systems

UPPAAL Help



The screenshot shows a window titled "UPPAAL - Help". On the left is a tree view of the help content:

- 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 Specification
 - Expressions
 - Reserved Keywords
 - Command Line Options
 - File Formats
 - New Features

The main content area on the right is titled "UPPAAL Help" and contains the following text:

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 [verification](#). 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: $E \langle \rangle P$
- **Safety Properties**
 - Invariant: $A[] P$
 - Pos. Inv.: $E[] P$
- **Liveness Properties**
 - Eventually: $A \langle \rangle P$
 - Leadsto: $P \rightarrow Q$
- **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).



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 Hydraulic 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

- Voodoo: 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.**



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.

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](#).

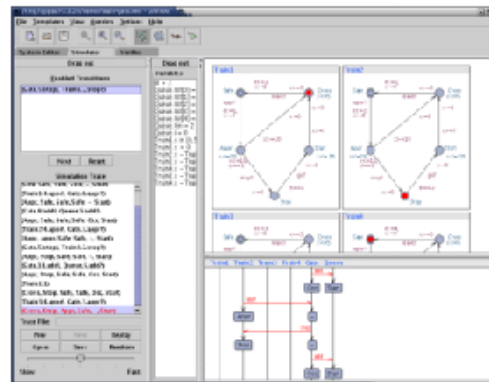


Figure 1: UPPAAL on screen.

License

The UPPAAL tool is **free** for non-profit applications. For information about commercial licenses, please email [sales\(at\)uppaal\(dot\)com](mailto:sales@uppaal.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](mailto:uppaal(at)list(dot)it(dot)uu(dot)se).



UPPSALA
UNIVERSITET



AALBORG UNIVERSITY



LAB-Exercises

www.cs.aau.dk/~kgl/Shanghai2013/exercises

Exercise 19 (Train Crossing)

Exercise 2 (Coffee Machine)

Exercise 28 (Jobshop Scheduling)

Exercise 14 (Gossiping Girls)

