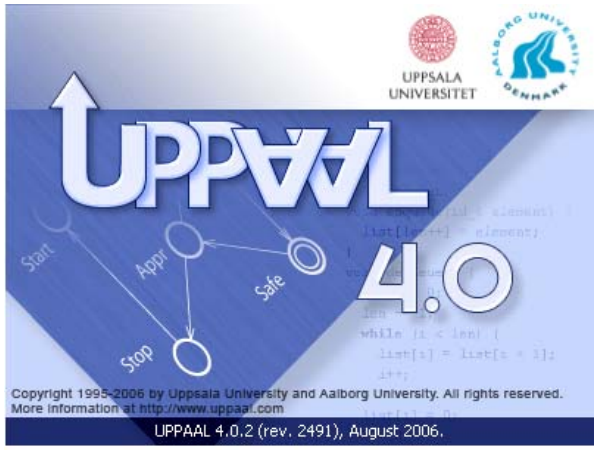


Validation, Synthesis and Performance Evaluation of Embedded Systems

using UPPAAL

Kim Guldstrand Larsen

Timed Automata

UPPAAL (1995-)

@AALborg

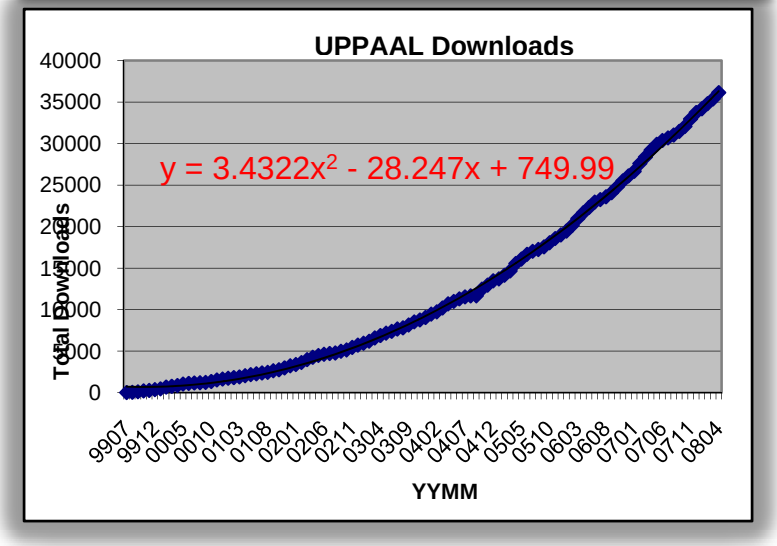
- Kim G Larsen
- Alexandre David
- Gerd Behrman
- Marius Mikucionis
- Jacob I. Rasmussen
- Arne Skou
- Brian Nielsen
- Shuhao Li

@UPPsala

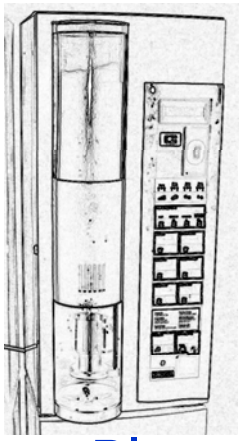
- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@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.....

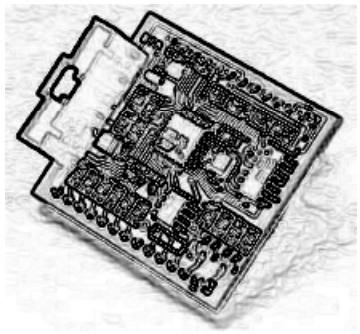


Real Time Systems



sensors

actuators



Plant

Continuous

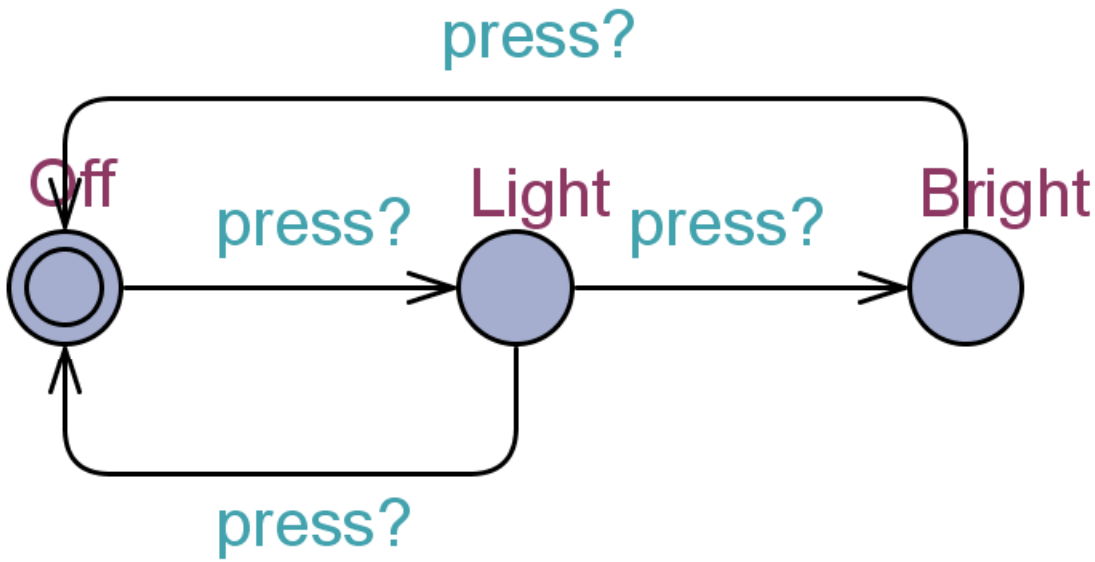
Controller Program

Discrete

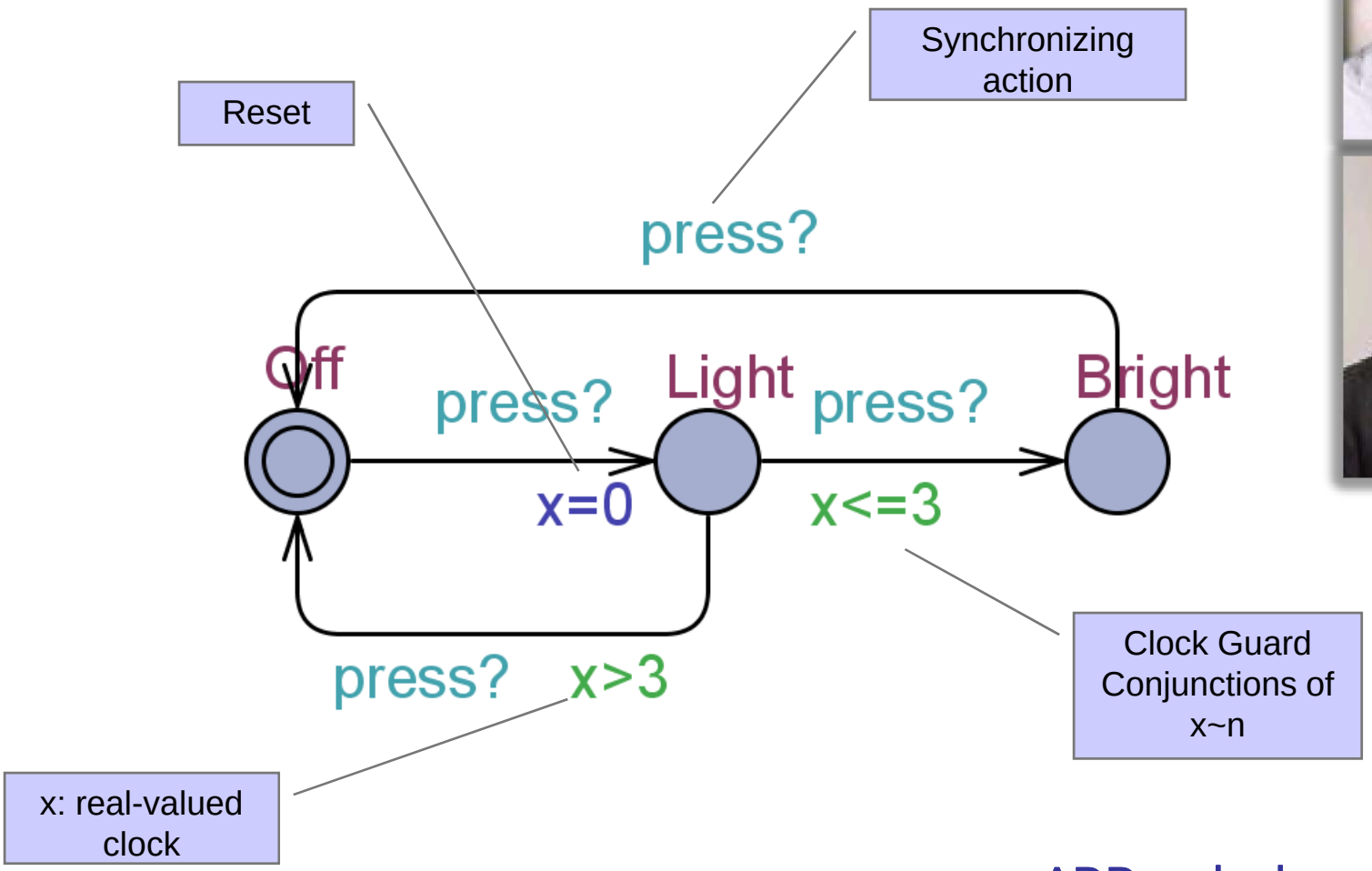
Eg.: Realtime Protocols
Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

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

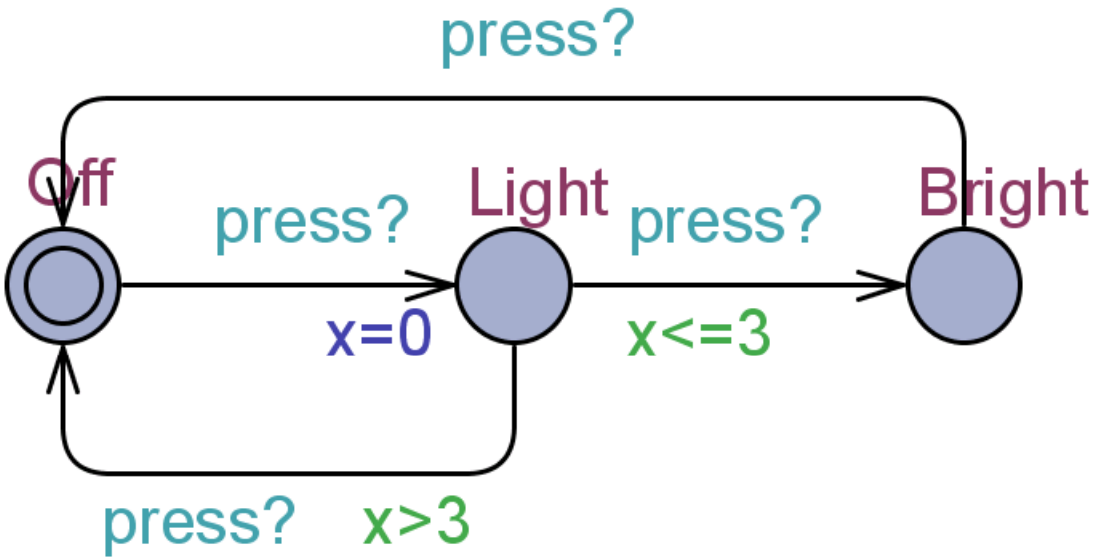


Timed Automata [Alur & Dill'89]



ADD a clock x

A Timed Automata (Semantics)



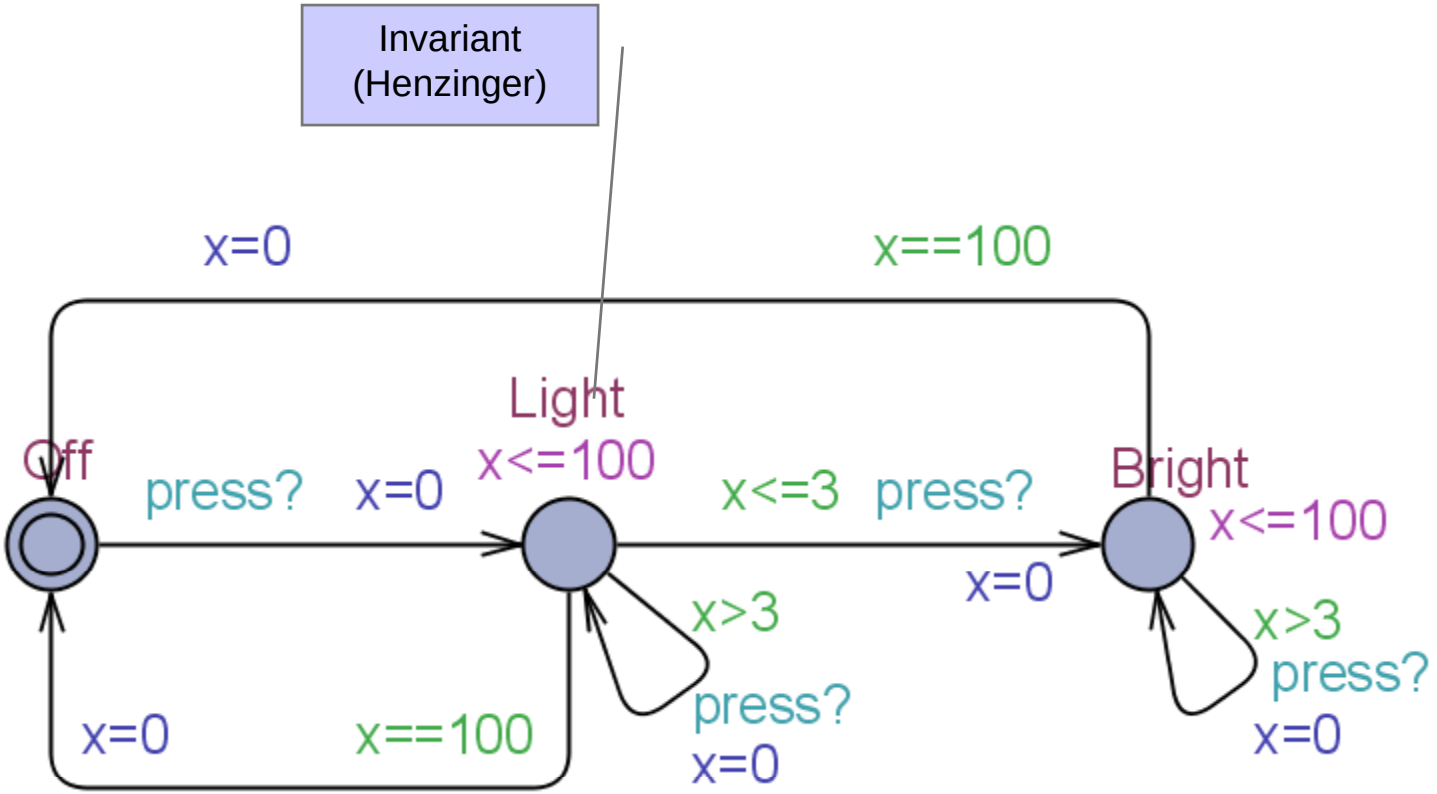
States:

(location , x=v) where v∈**R**

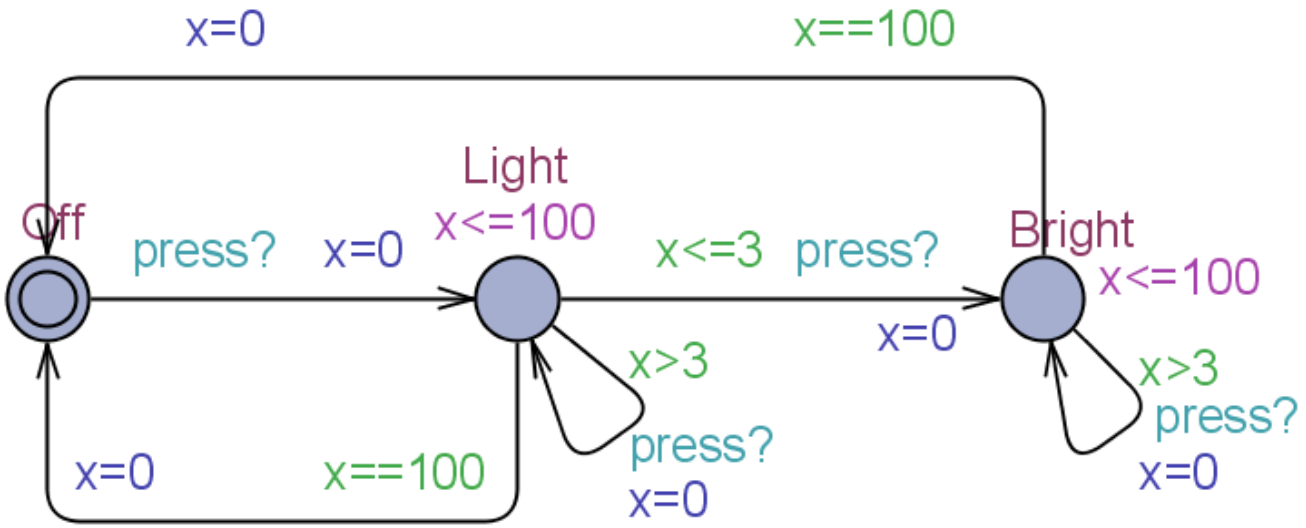
Transitions:

	(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

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 $C \longrightarrow \mathbb{R}_{\geq 0}$ ranged over by $u, v, w, \ldots$.

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
- $I : L \longrightarrow \mathcal{B}(X)$ assigns to each location an invariant

Timed Automata (formally)

Semantics

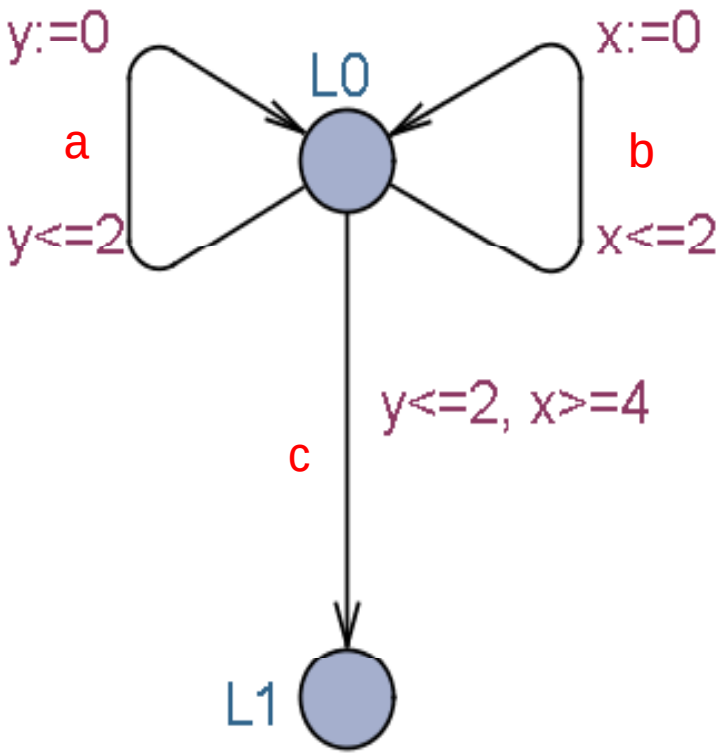
Definition

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
  - $u' \in I(l')$

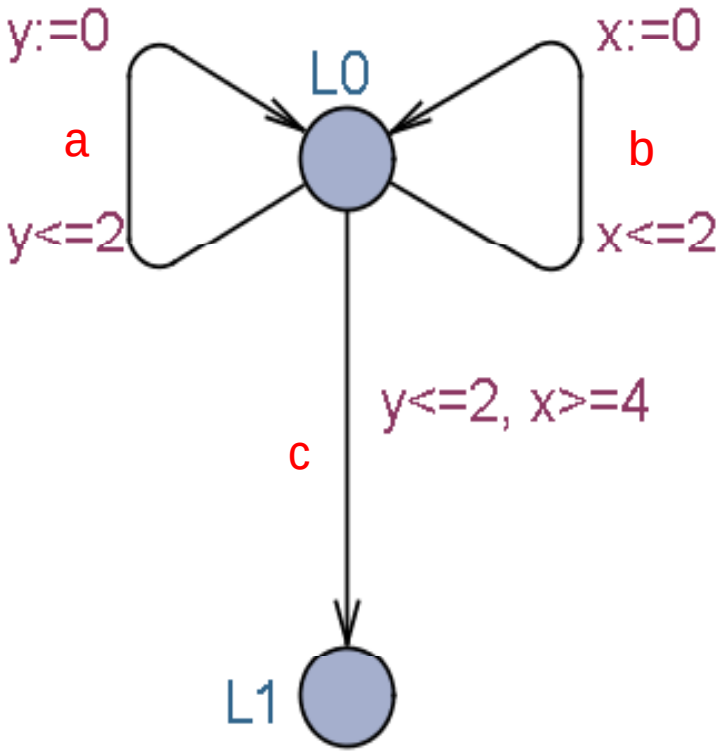
$^*u_0(x) = 0$ for all $x \in C$

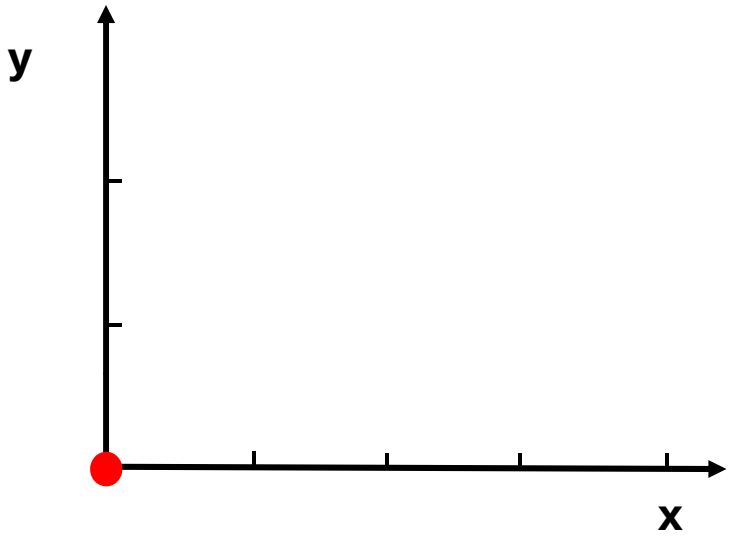
Example



Is L1 reachable ?

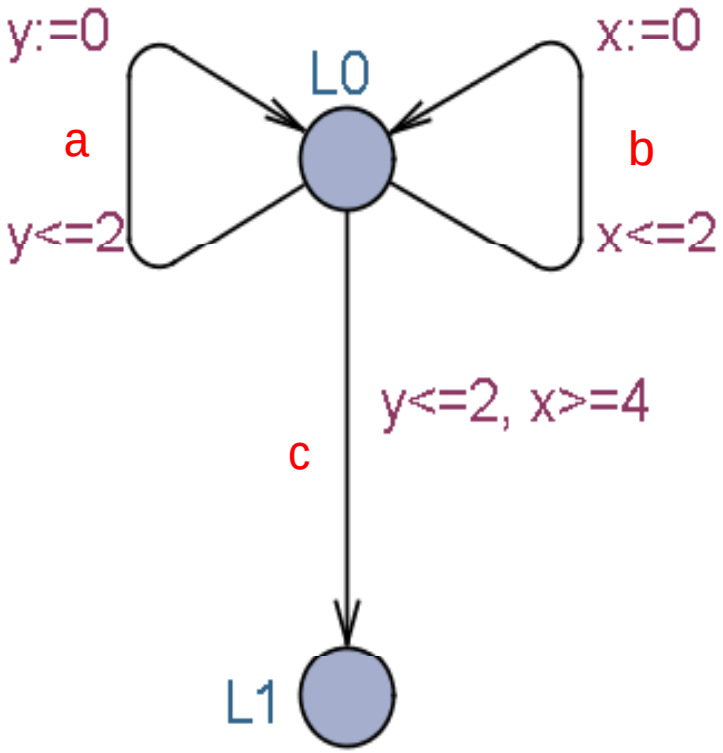
Example

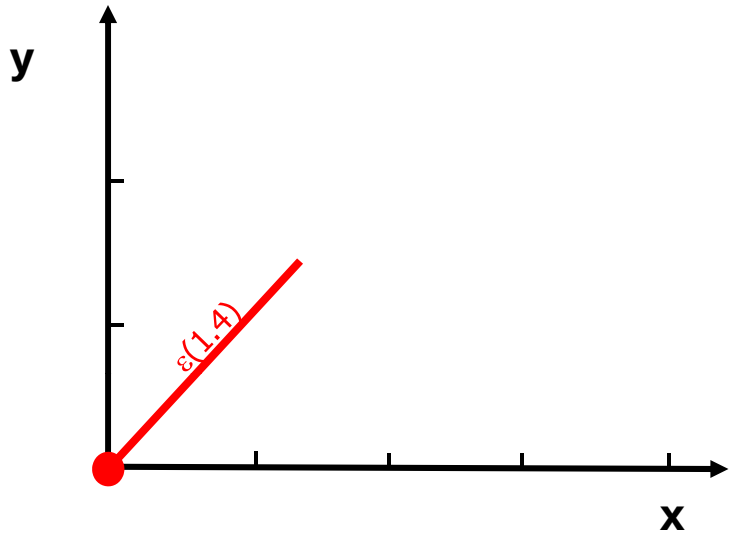




$(\ell_0, x = 0, y = 0)$

Example

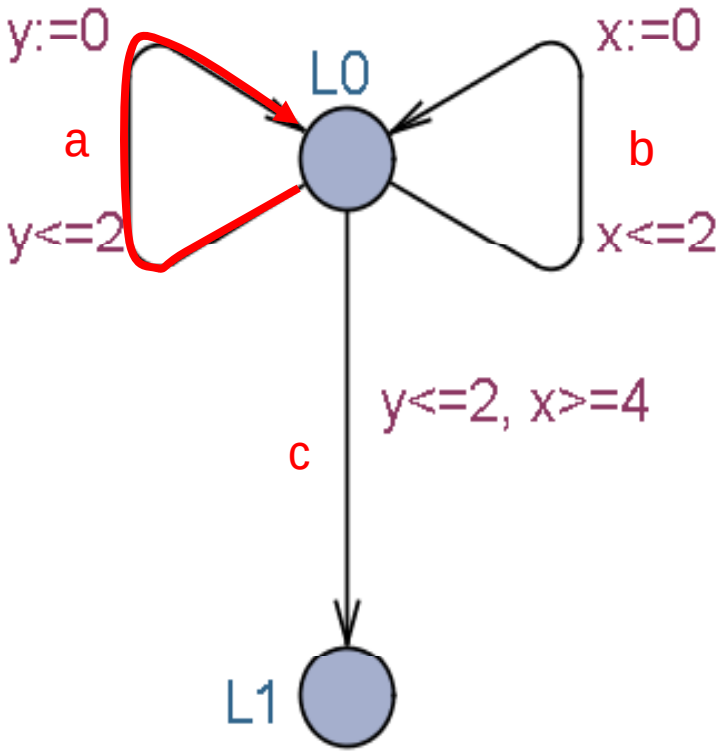


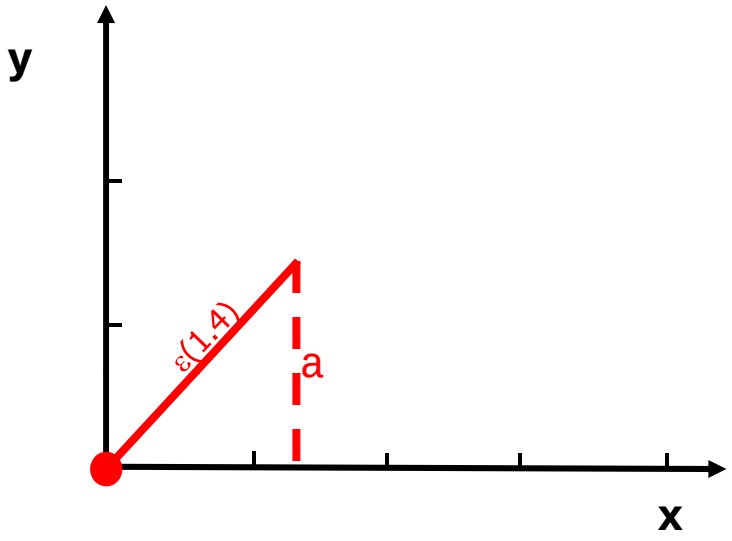


$$(\ell_0, x = 0, y = 0)$$

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

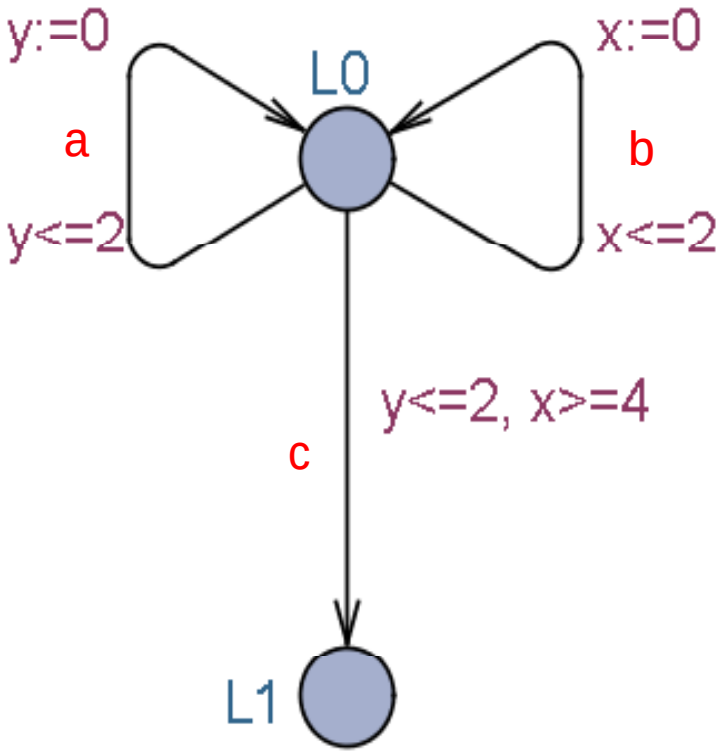
Example

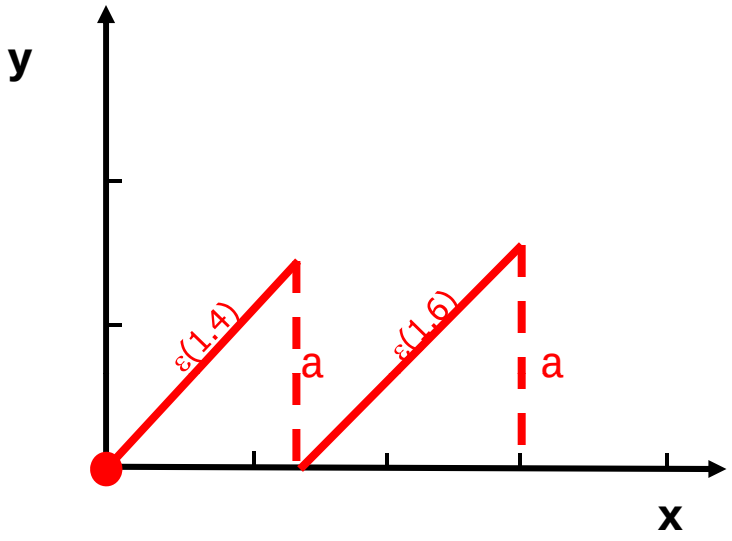




$$(\ell_0, x = 0, y = 0)$$
$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$
$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

Example





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

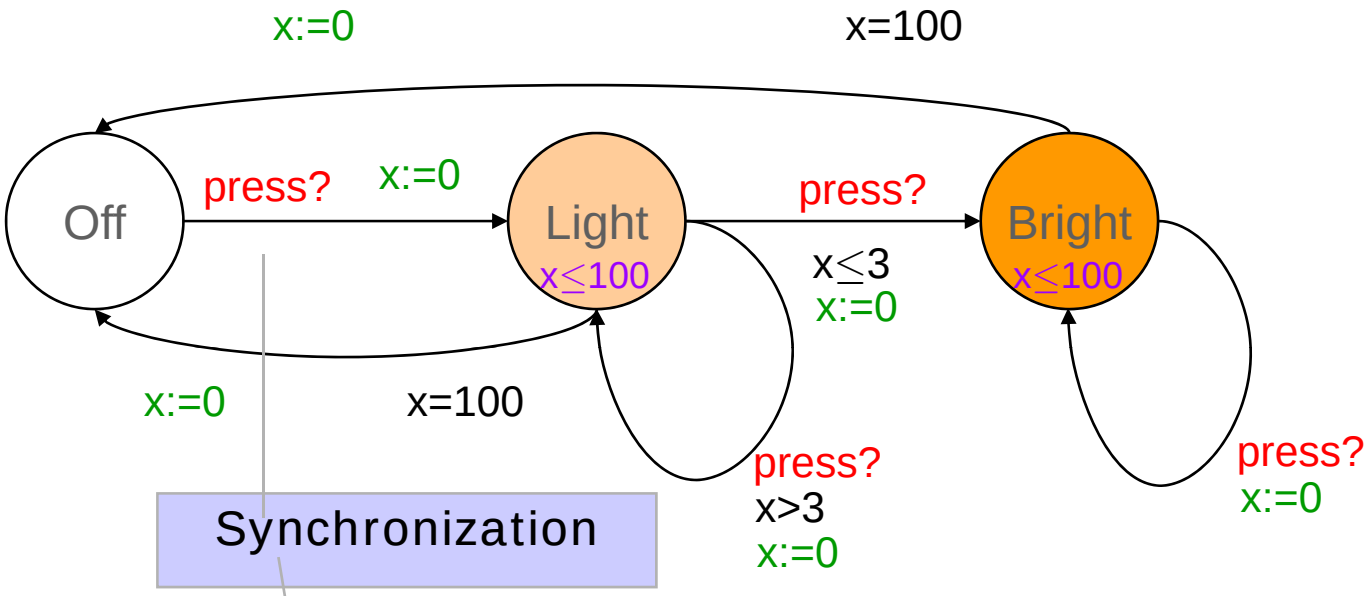
$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$

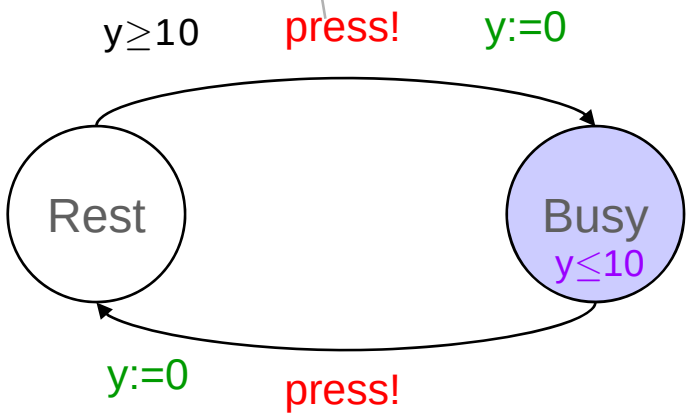
$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

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

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

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

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

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

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

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

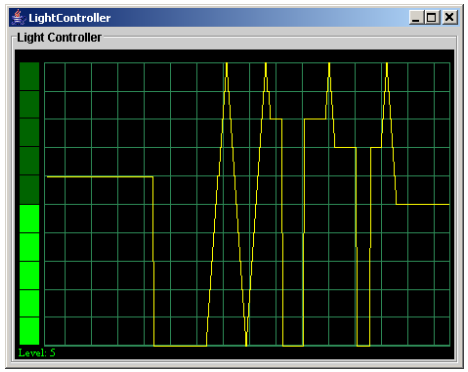
$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1' \|_X s_2} \qquad \frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{\mu} s_1 \|_X s_2'}$$

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

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

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

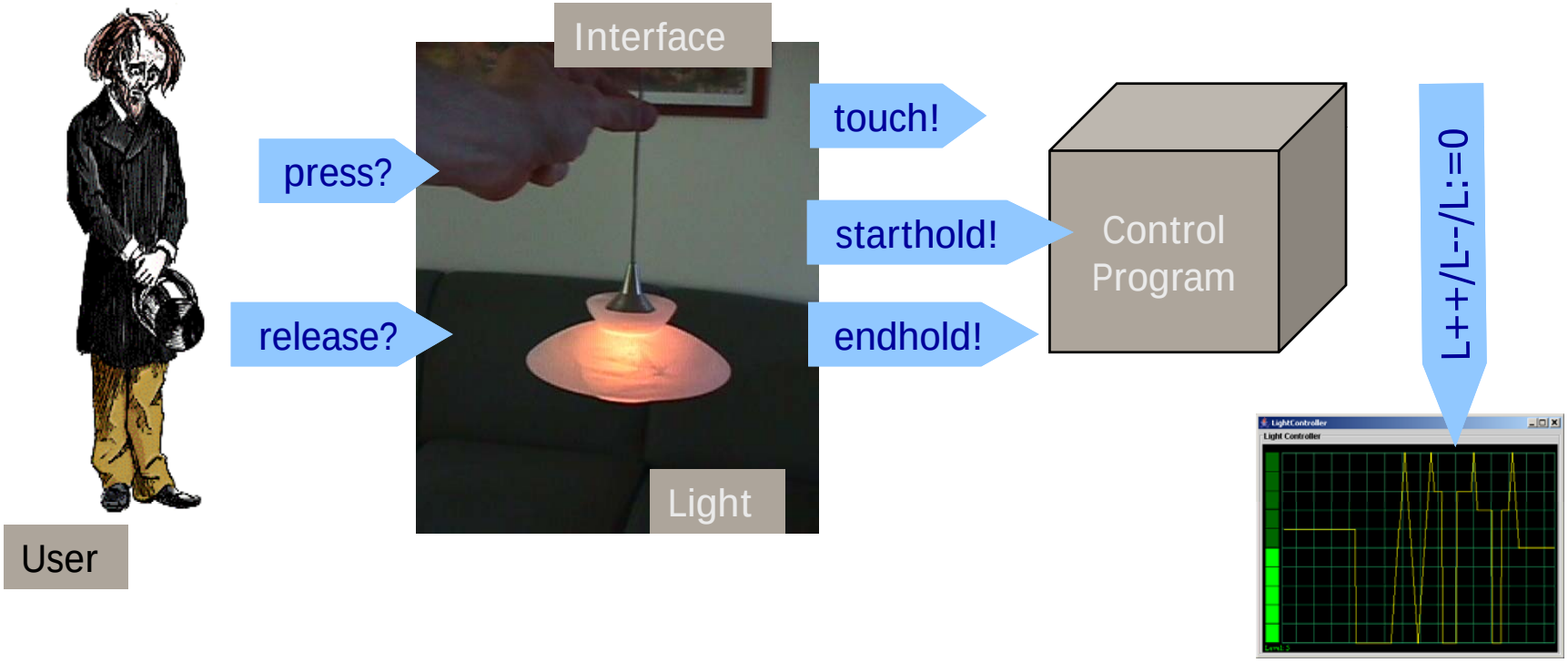
$$\frac{s_1 \xrightarrow{e(d)}_1 s_1' \qquad s_2 \xrightarrow{e(d)}_2 s_2'}{s_1 \|_X s_2 \xrightarrow{e(d)} s_1' \|_X s_2'}$$



Light Control Interface

Light Control Interface

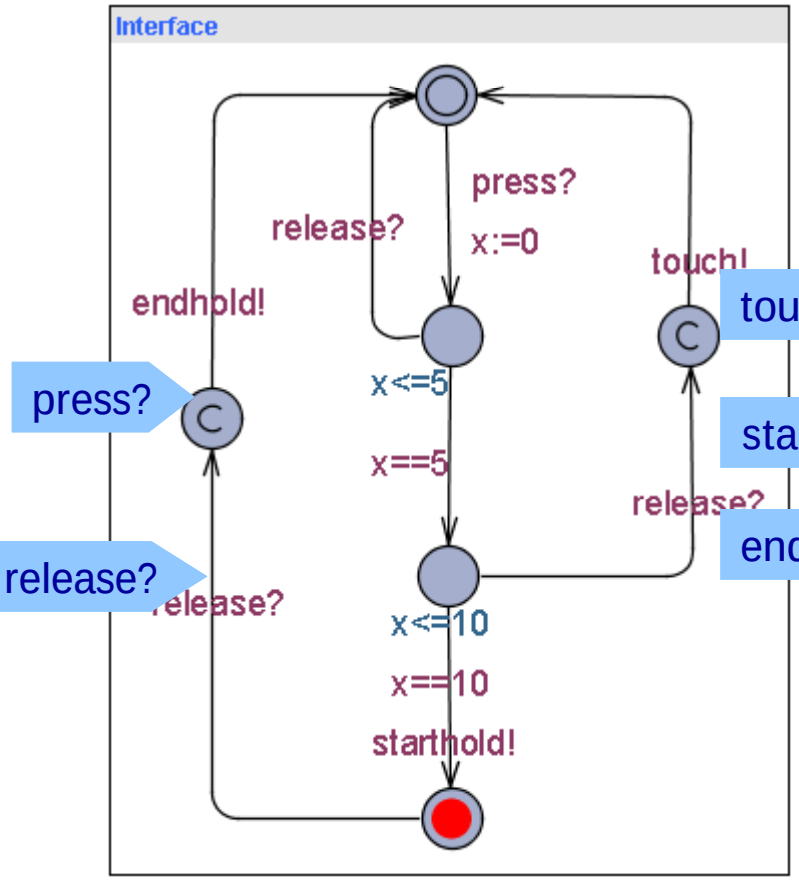
press? d release? → touch! $0.5 \leq d \leq 1$
press? 1 → starthold!
press? d release? → endhold! d >1



press? 0.2 release? … press? 0.7 release? … press? 1.0 2.4 release? …
↓ ↓ ↓ ↓
Ø touch! starthold! endhold!

Light Control Interface

User



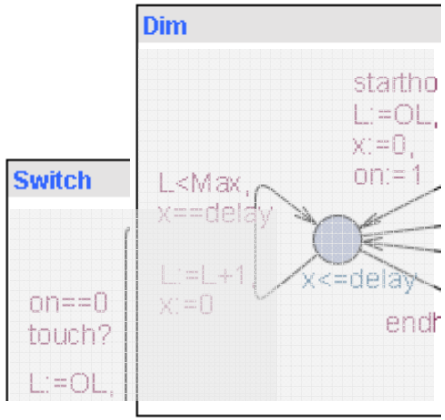
touch!

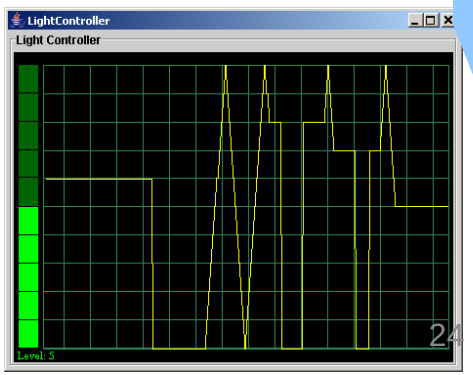
starthold!

endhold!

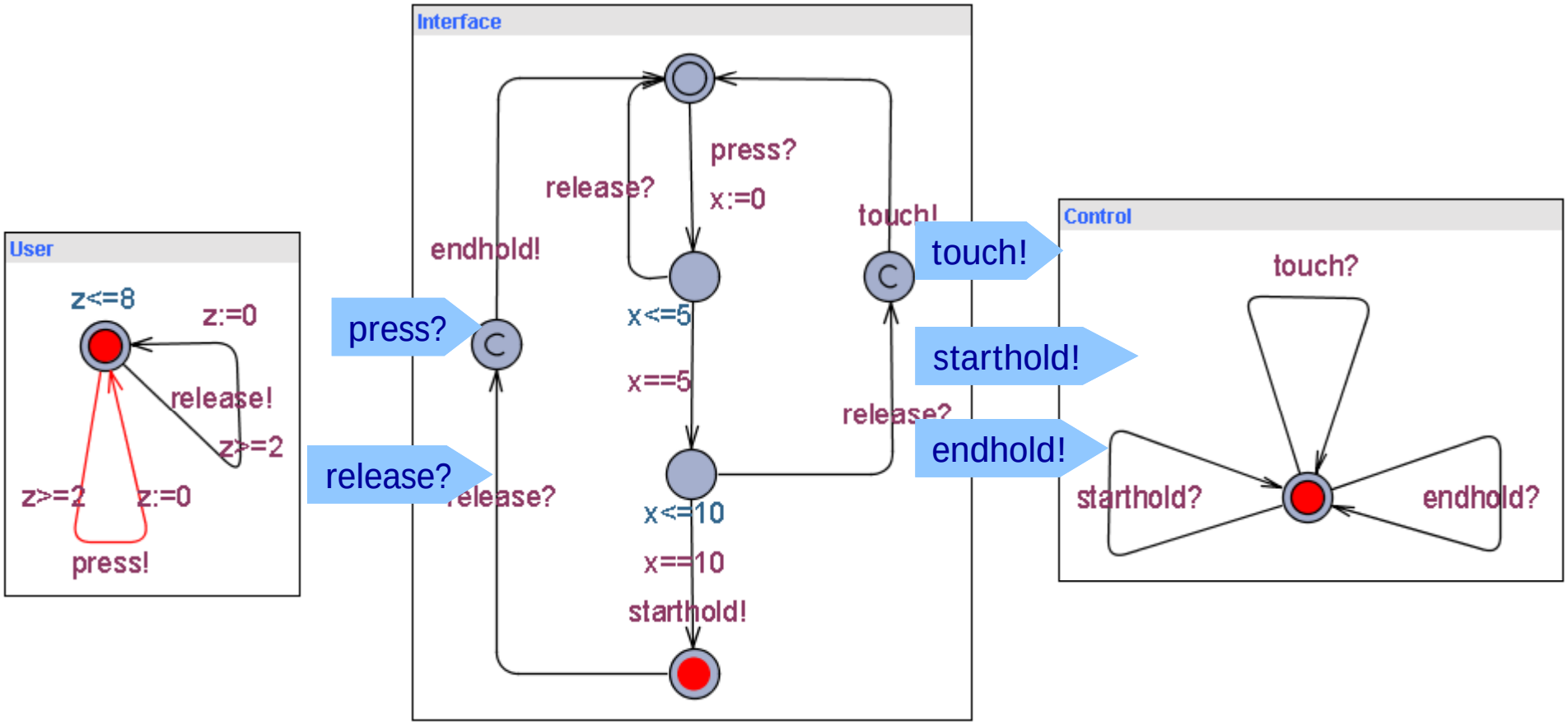
Control Program

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





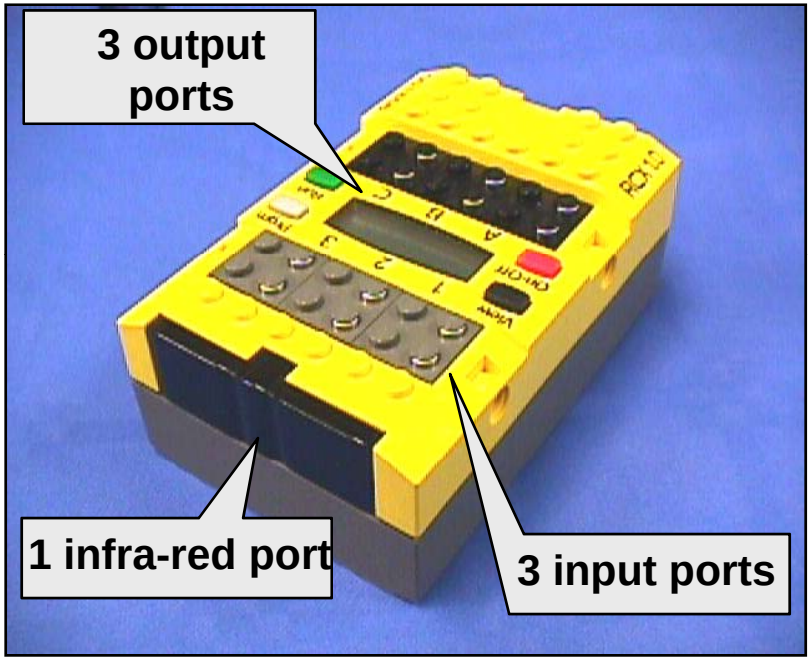
Light Control Network



Brick Sorting

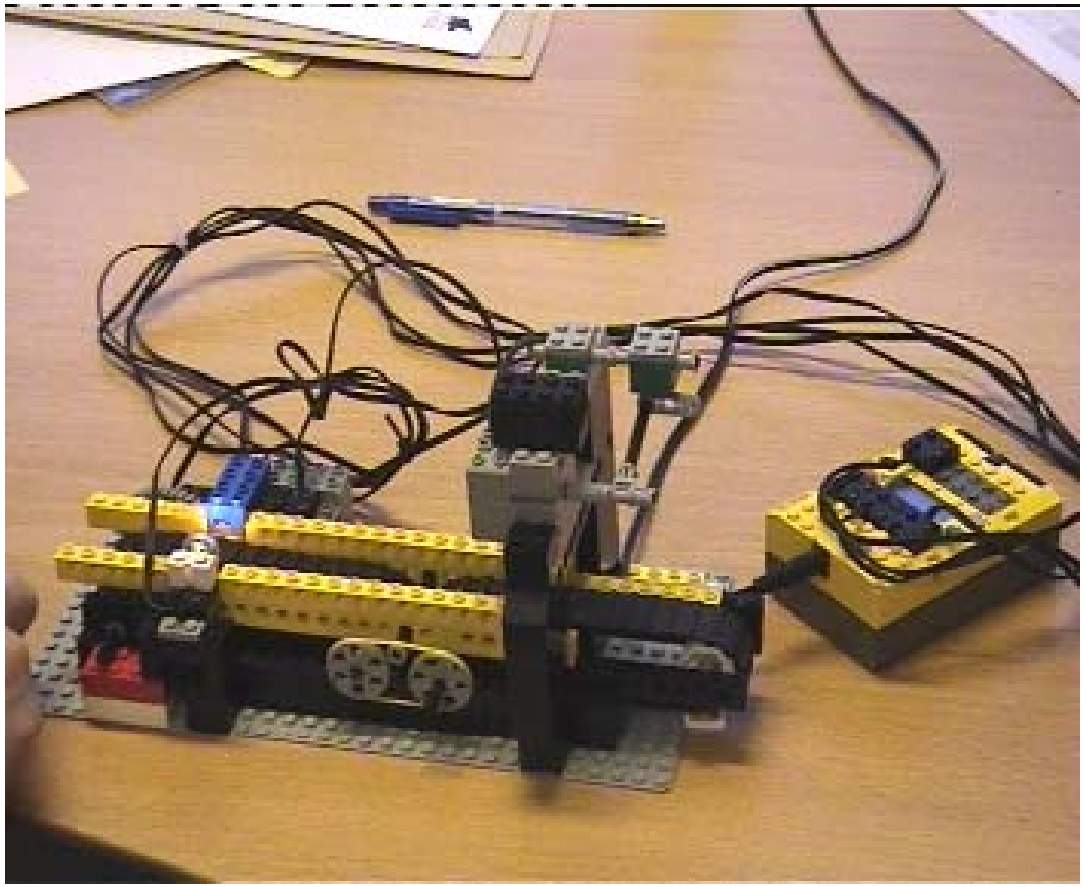
LEGO Mindstorms/RCX

- 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

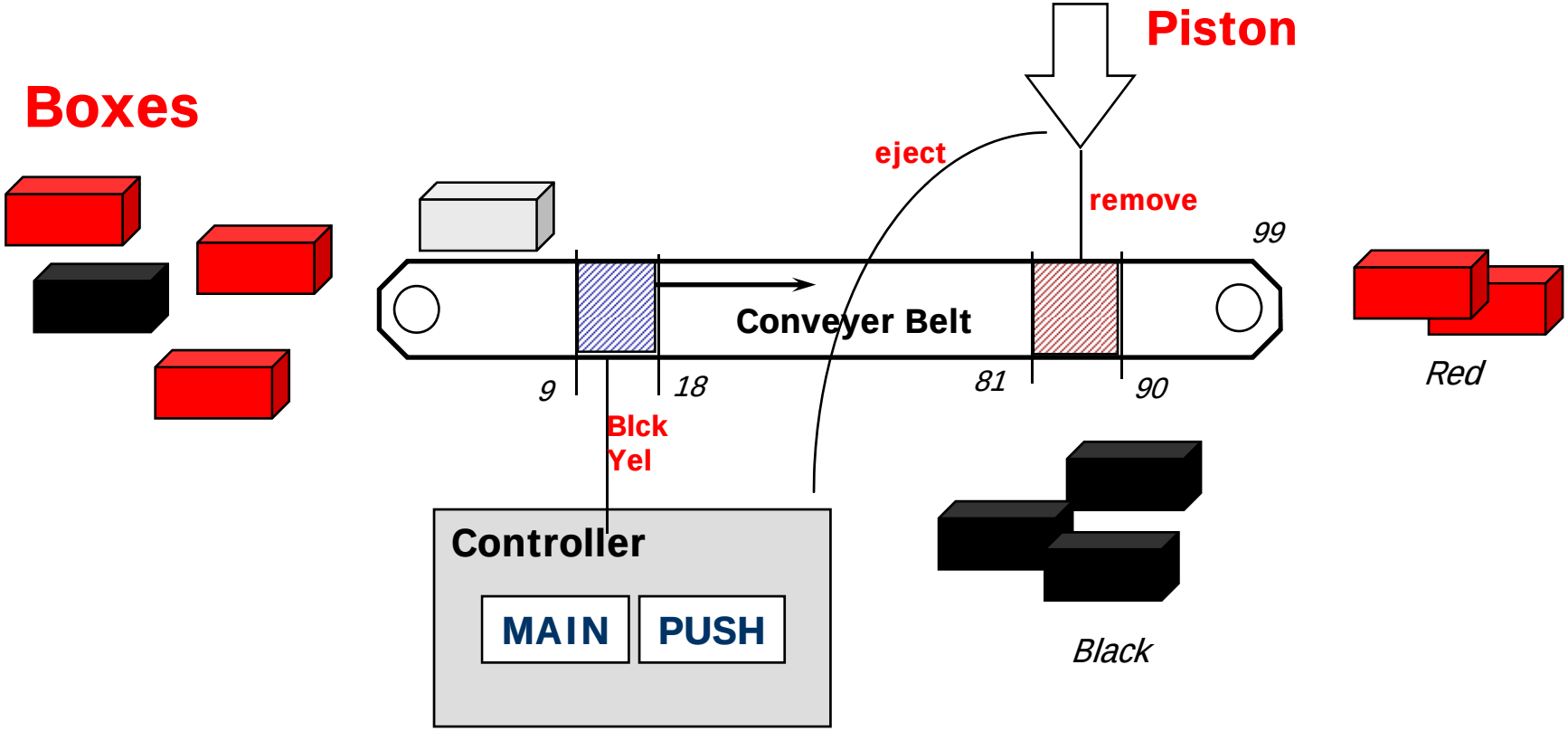


Controller Program
LEGO MINDSTORM

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



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

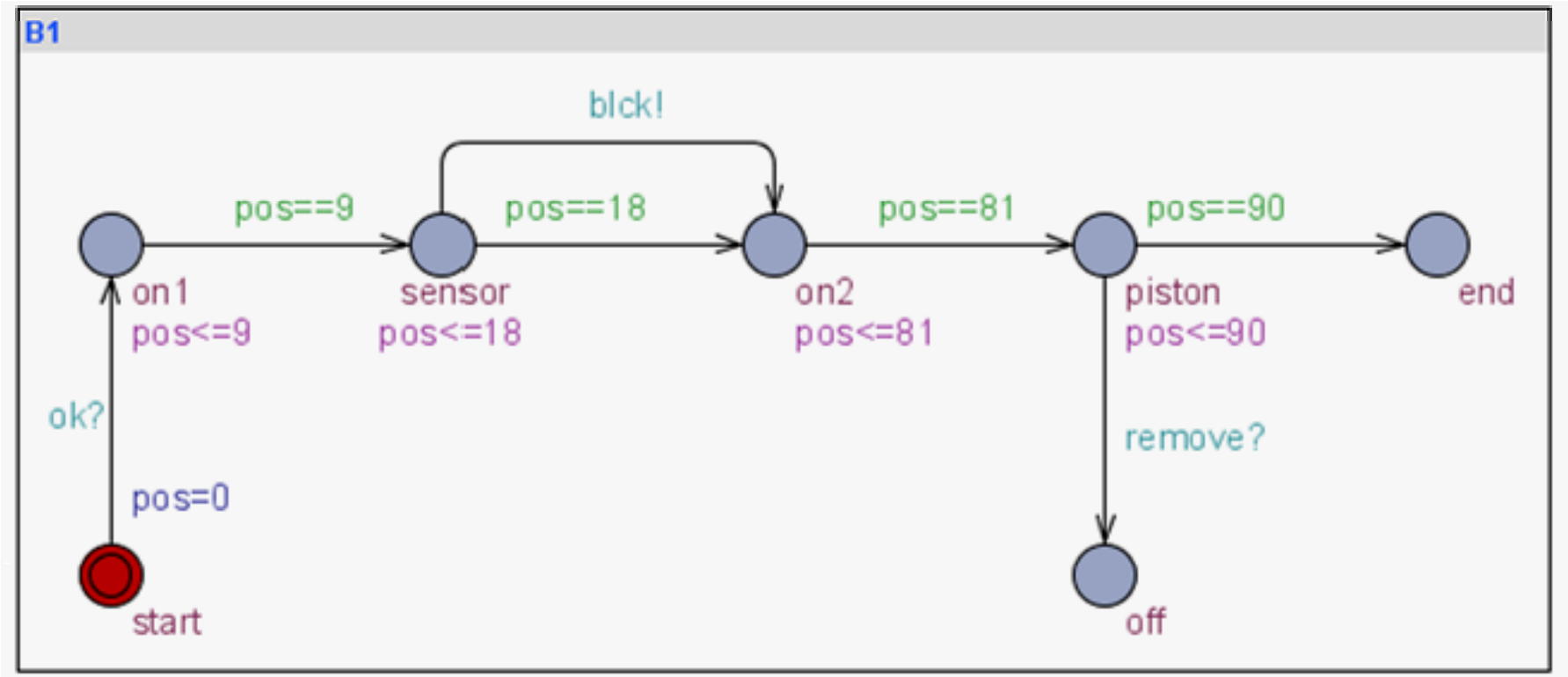
NQC programs

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

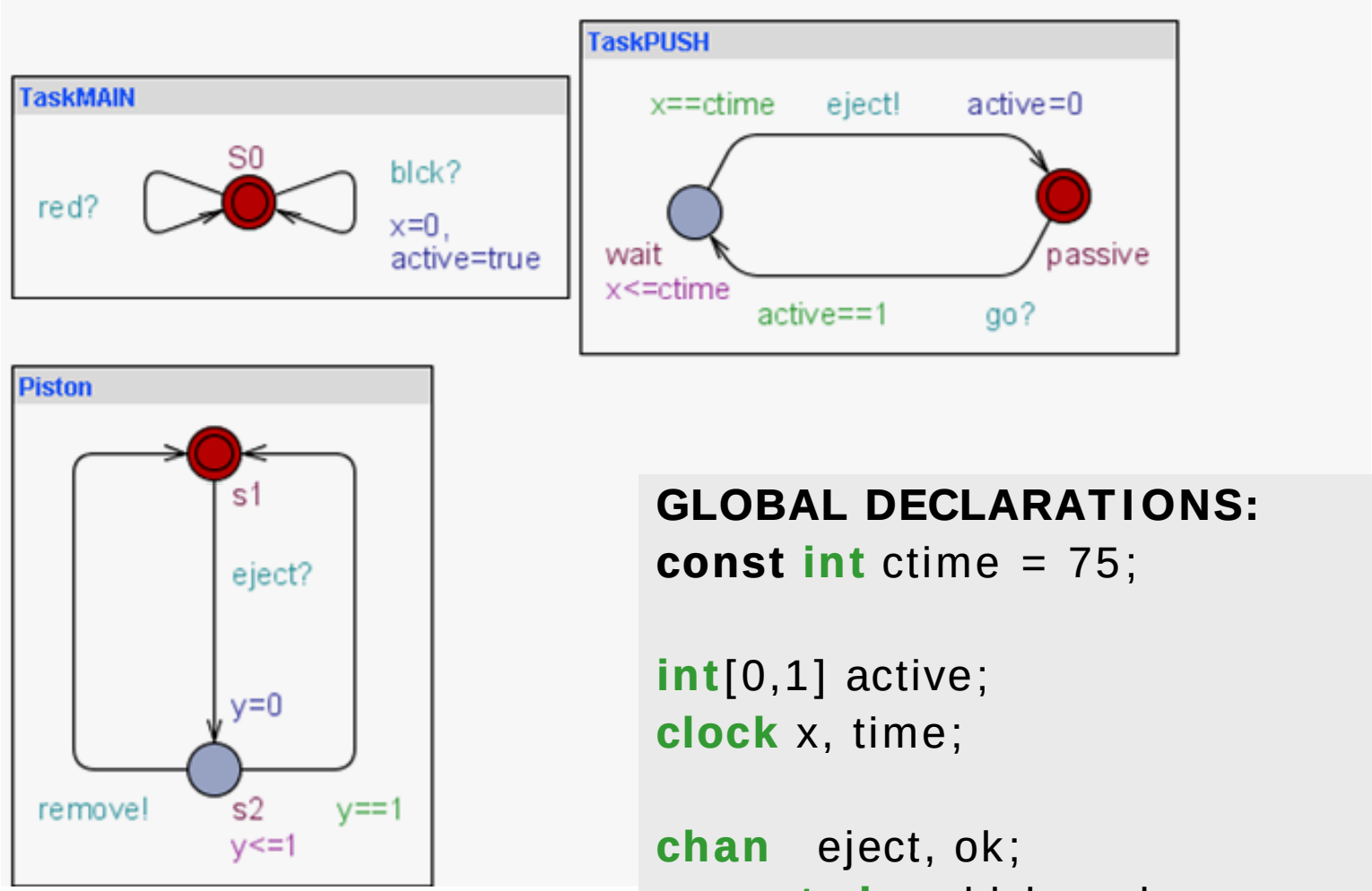
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 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

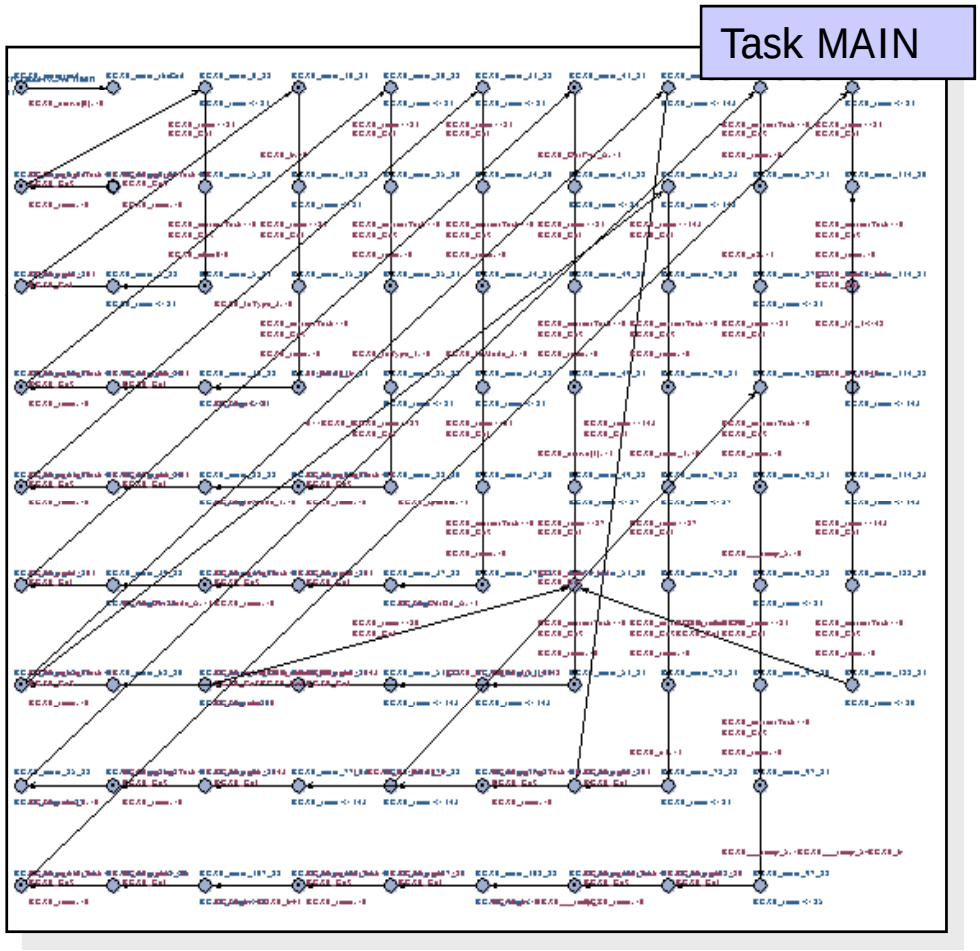



```
GLOBAL DECLARATIONS:
const int ctime = 75;

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

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

From RCX to UPPAAL – and back

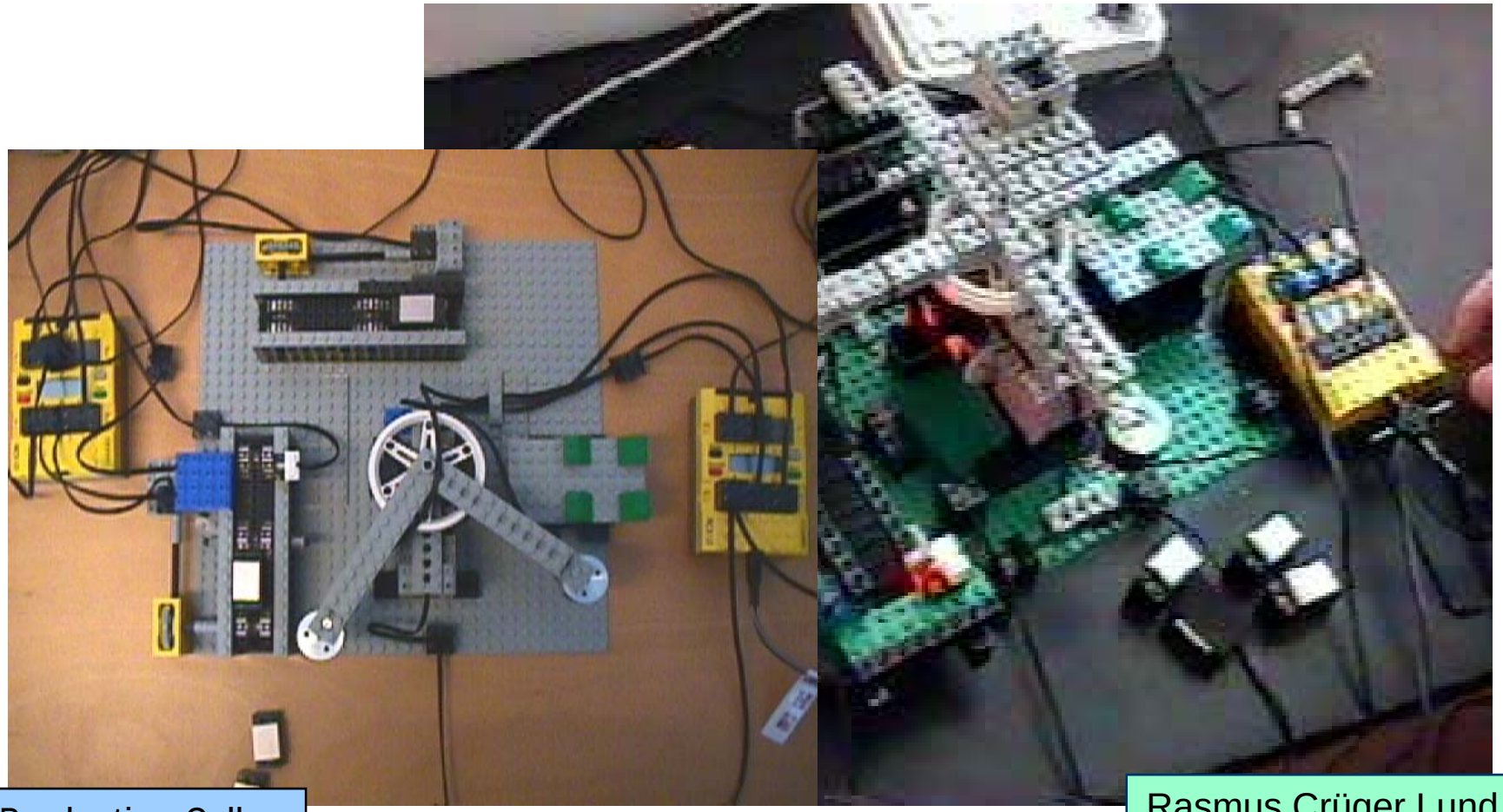
- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000 in Stockholm.

- From UPPAAL to RCX: Martijn Hendriks.



The Production Cell in LEGO

Course at DTU, Copenhagen



Production Cell

Rasmus Crüger Lund
Simon Tune Riemanni

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.{com,org}

UP4ALL

DESIGN VERIFICATION FOR EMBEDDED SYSTEMS

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

HOME PRODUCT SOLUTIONS PARTNERS SUPPORT WEB HELP CONTACT US

DOWNLOAD & BUY UPPAAL SOFTWARE

NON PROFIT USER?

NEWS & EVENTS

Aug 20 2010: release of commercial version 4.0.12.

P.O. Box 337, SE-75105 Uppsala, Sweden +46 21 151741 sales@uppaa

RELATED TOOLS: TIMES | 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.

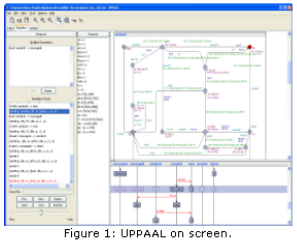


Figure 1: UPPAAL on screen.

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 a development snapshot of the forthcoming UPPAAL 4.2. The current development snapshot version is 4.1.4 released Jul 11, 2011.

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

License

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

Localization

In our ongoing work to localize the UPPAAL GUI we would like to acknowledge contributions by the following external people:

- Hiroshi Fujimoto (JA)