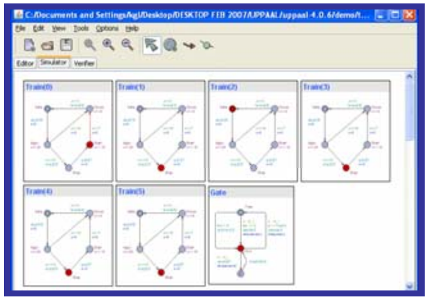
Real-Time Model Checking

Patricia Bouyer-Decitre
Kim G. Larsen
Nicolas Markey

Timed Automata
.. and Prices and Games

Patricia Bouyer-Decitre
Kim G. Larsen
Nicolas Markey

QUANTITATIVE Model Checking

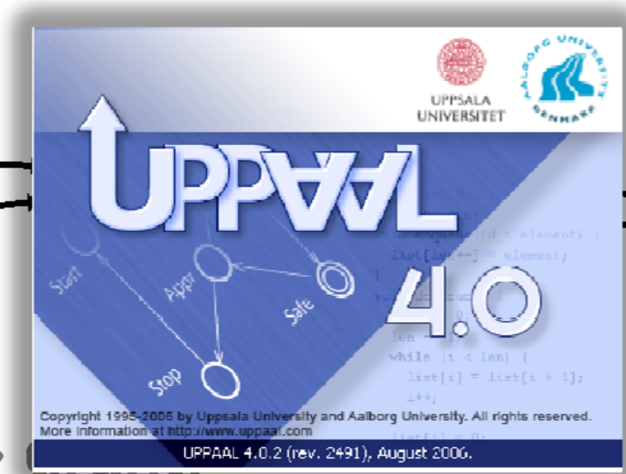


System Description

Time

Cost

Probability



No!
Debugging Information

Yes
Prototypes
Executable Code
Test sequences

Requirement

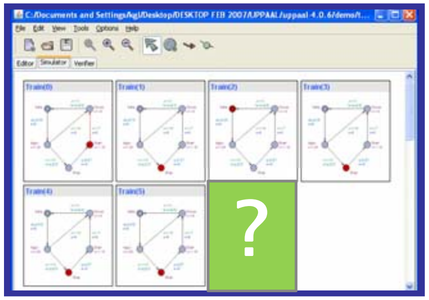
$A\Box(\, req \Rightarrow A\Diamond\, grant)$

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

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

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

Synthesis

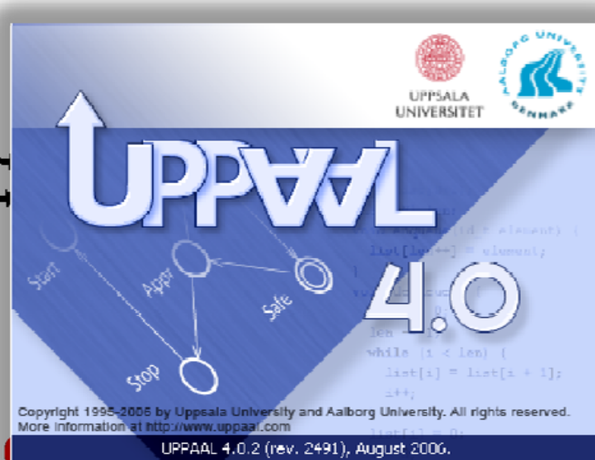


System Description

Time

Cost

Probability



No!
Debugging Information

Yes
Control Strategy

Requirement

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

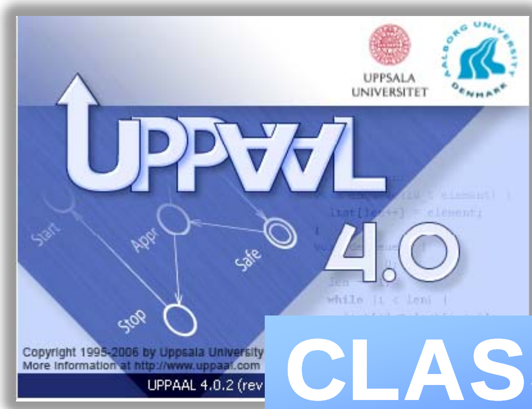
$$A\Box(\, req \Rightarrow A\Diamond_{t<30s}\, grant)$$

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

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

Overview

- Introduction to
  Timed Automata
- Decidability and
  undecidability results
- Timed Temporal Logics
- UPPAAL .. (hands-on)
- Timed Games
- Priced Timed Automata
- Open Problems



CLASSIC

CORA

TIGA

TRON

PRO

Timed Automata

UPPAAL (contributors)

@UPPsala

UPPSALA UNIVERSITET

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

@AALborg

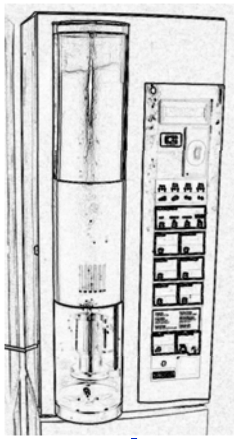
AALBORG UNIVERSITY DENMARK

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

@Elsewhere

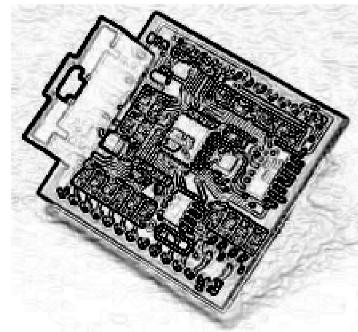
- Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffioen, Ansgar Fehnker, Jan Tretmans, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen,, Judi Romijn, Ed Brinksma, Martijn 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



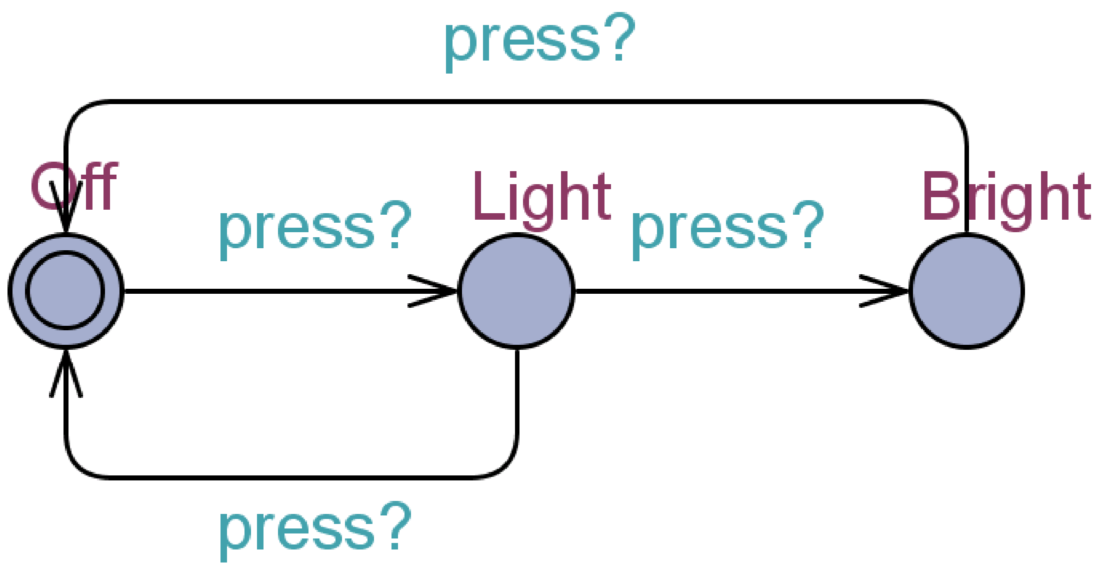
Plant
Continuous

Controller Program
Discrete

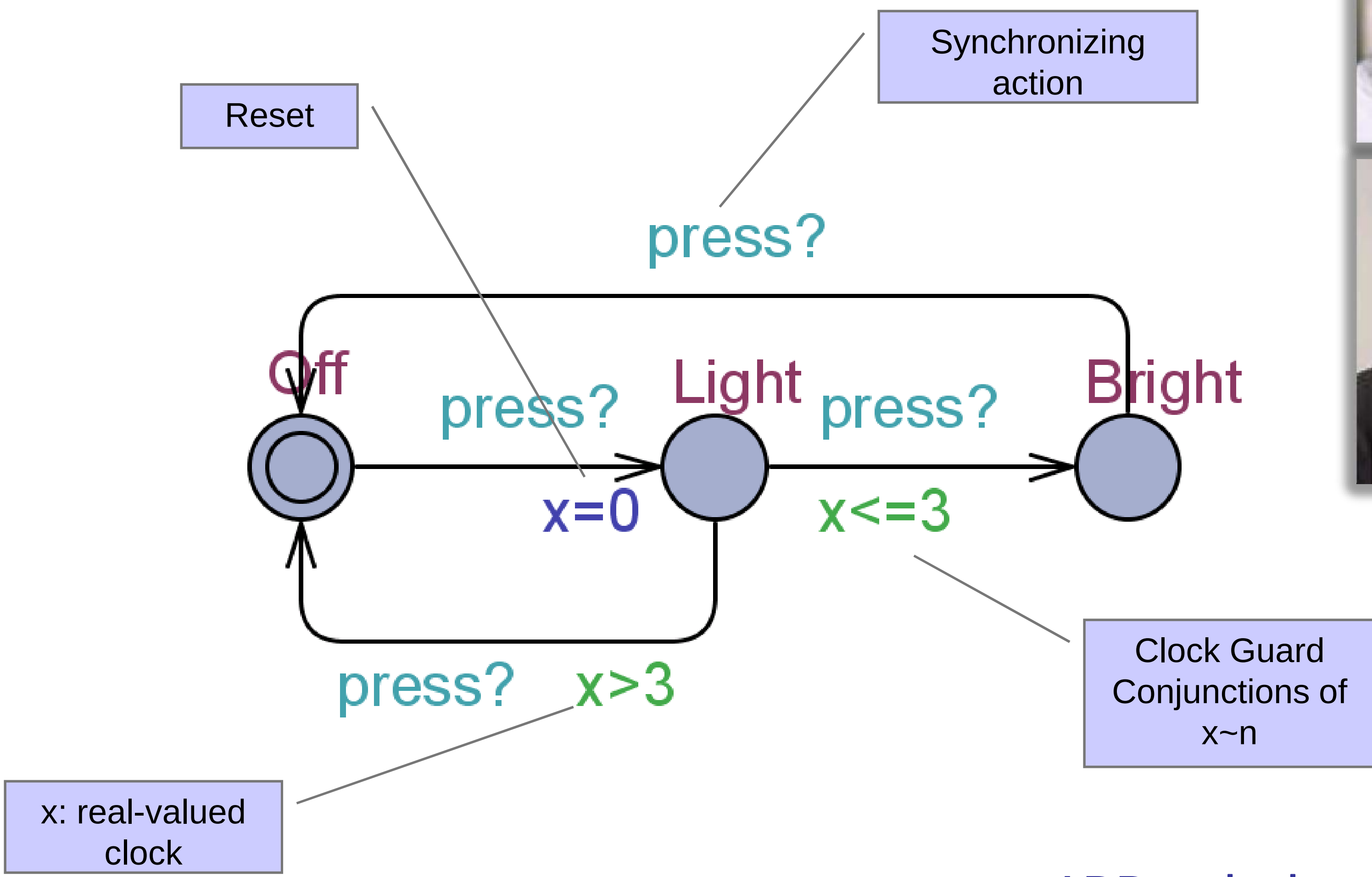
Eg.: Realtime Protocols
Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

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

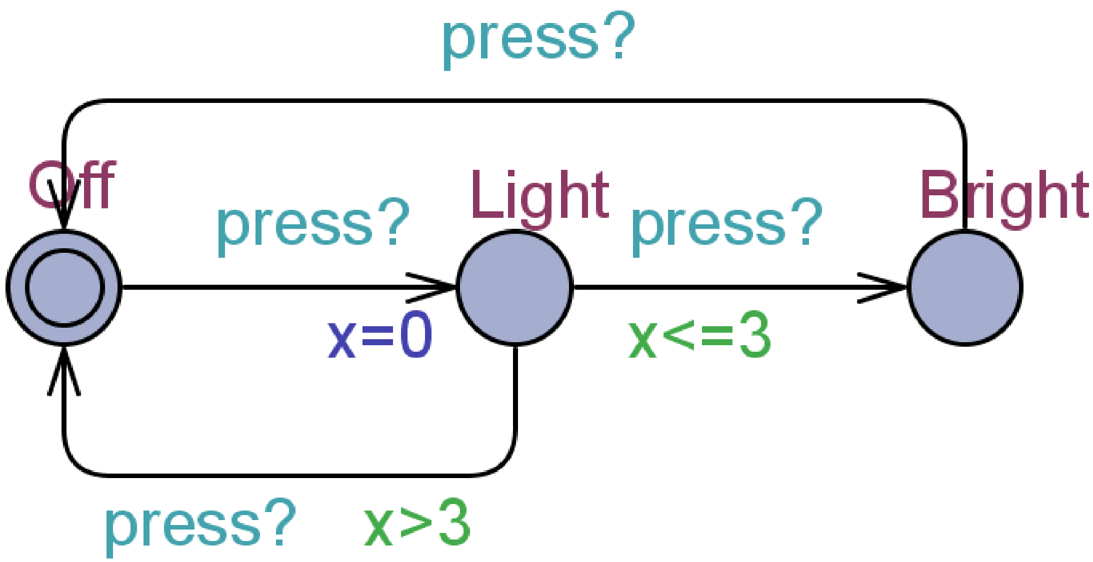


Timed Automata [Alur & Dill'89]



ADD a clock x

A Timed Automata (Semantics)

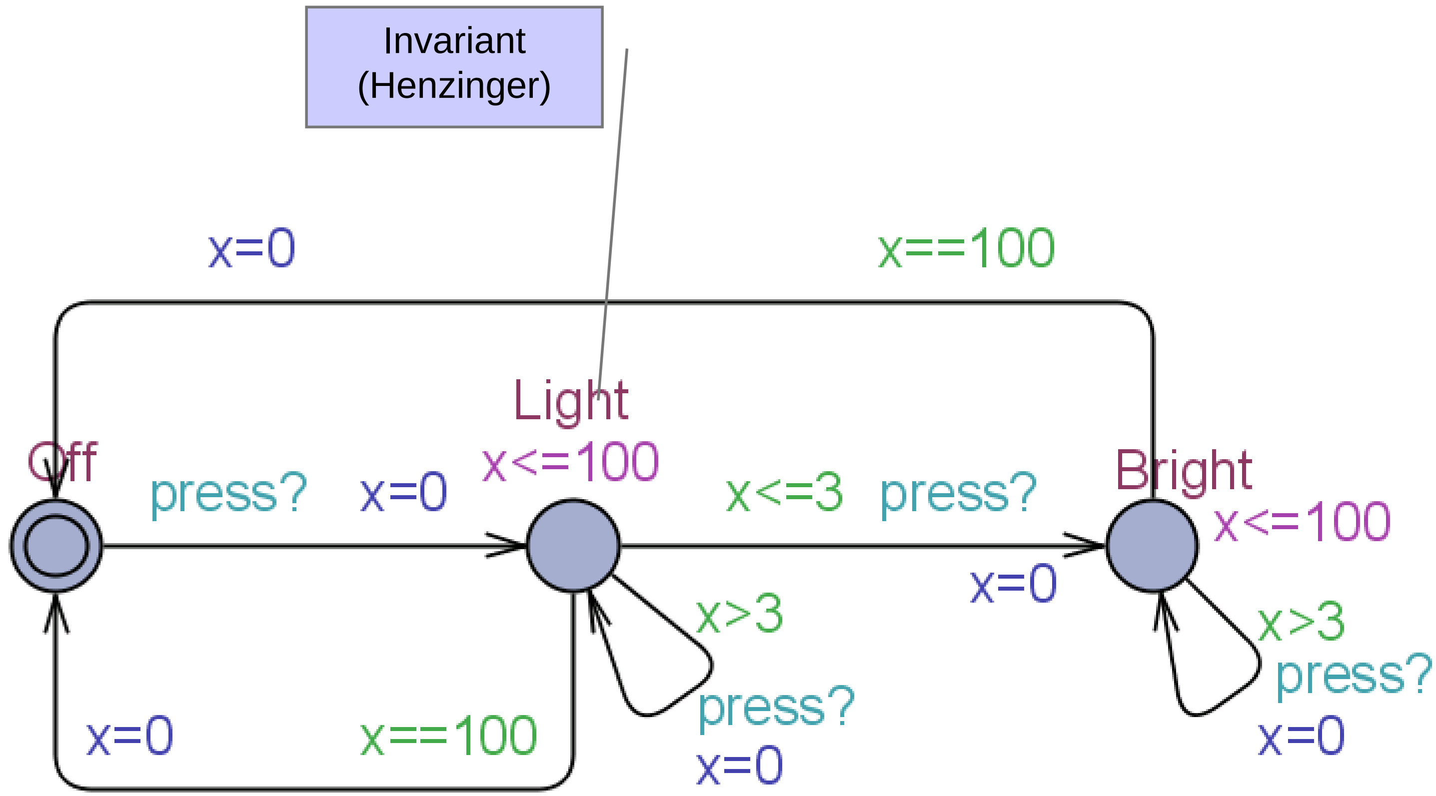


States:
(location , x=v) where v∈**R**

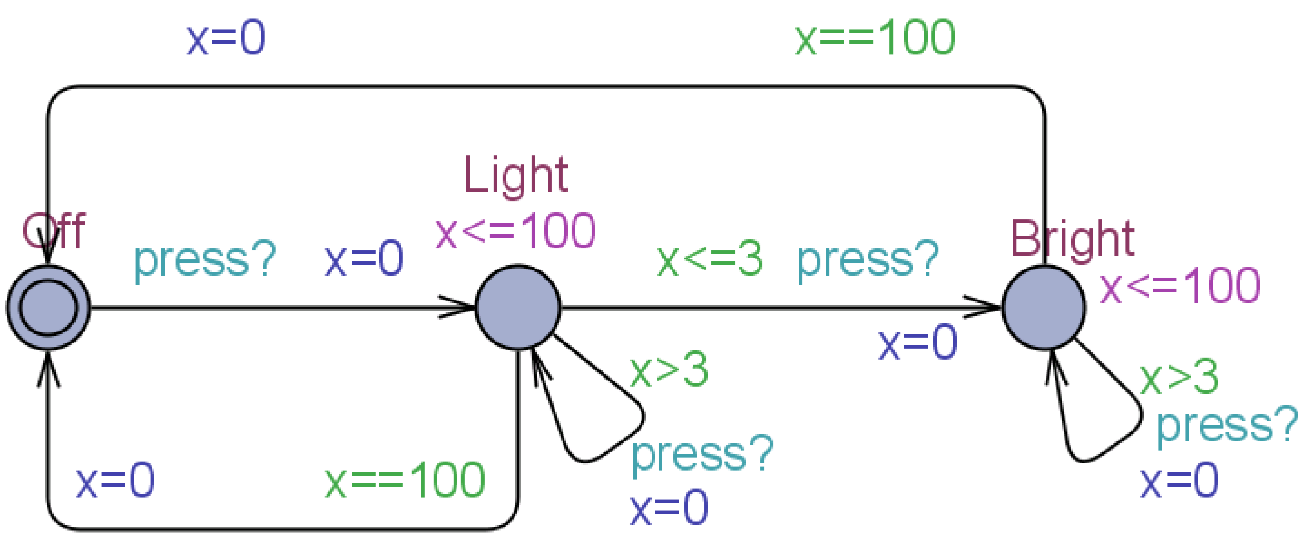
Transitions:

	(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: X

(Light , x=0) delay 103 →

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

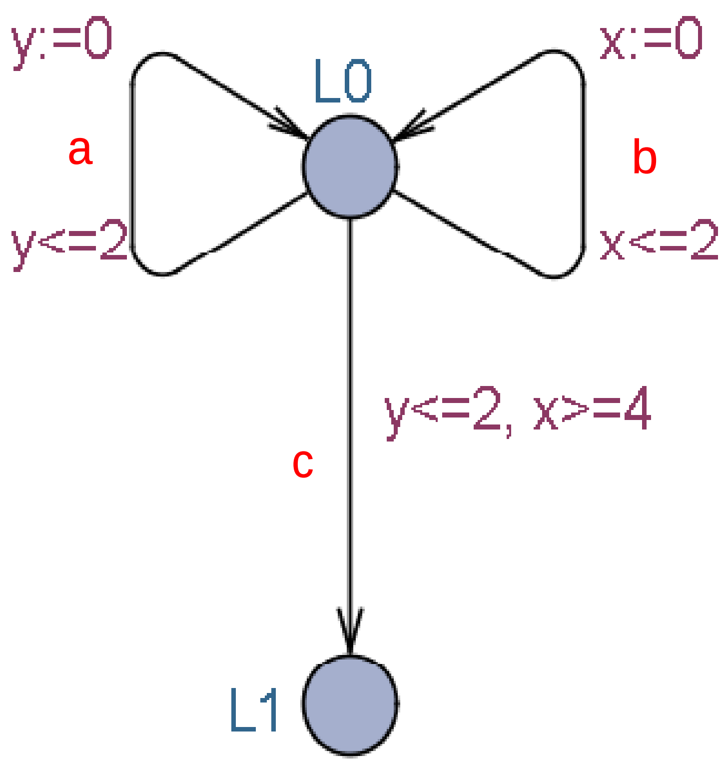
Definition

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:

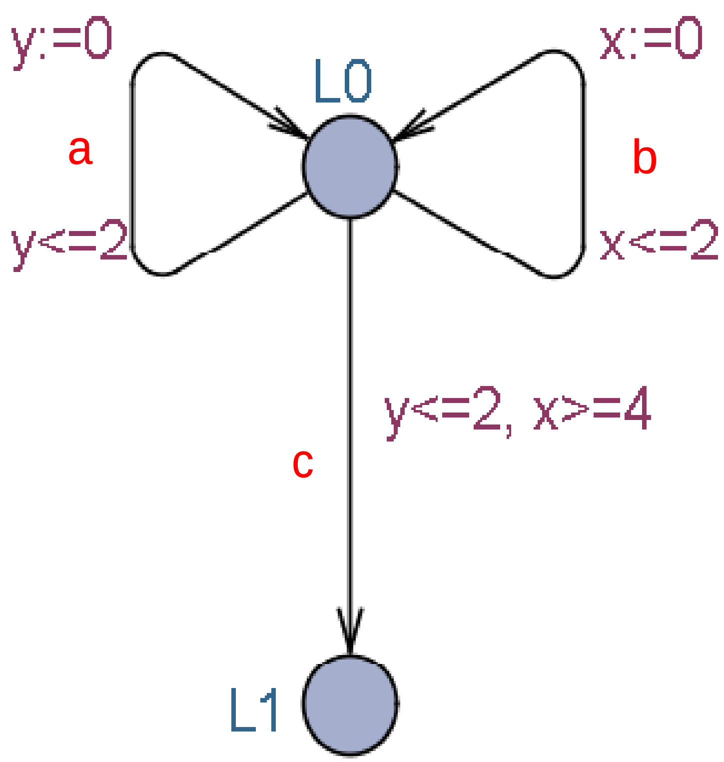
- $(l, u) \xrightarrow{\epsilon(d)} (l, u + d)$ iff $u \in I(l)$ and $u + d \in I(l)$,
- $(l, u) \xrightarrow{a} (l', u')$ iff there exists $(l, g, a, r, l') \in E$ such that
  - $u \models g$,
  - $u' = u[r]$, and
  - $u' \in I(l')$

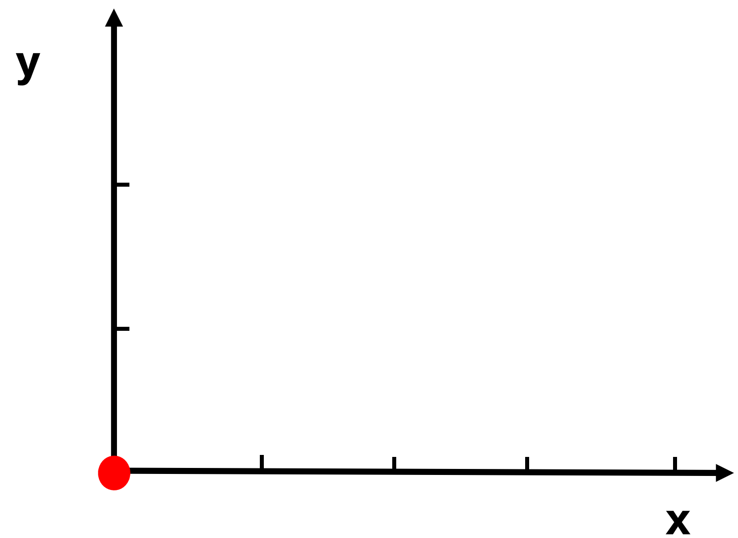
*$u_0(x) = 0$ for all $x \in C$

Example



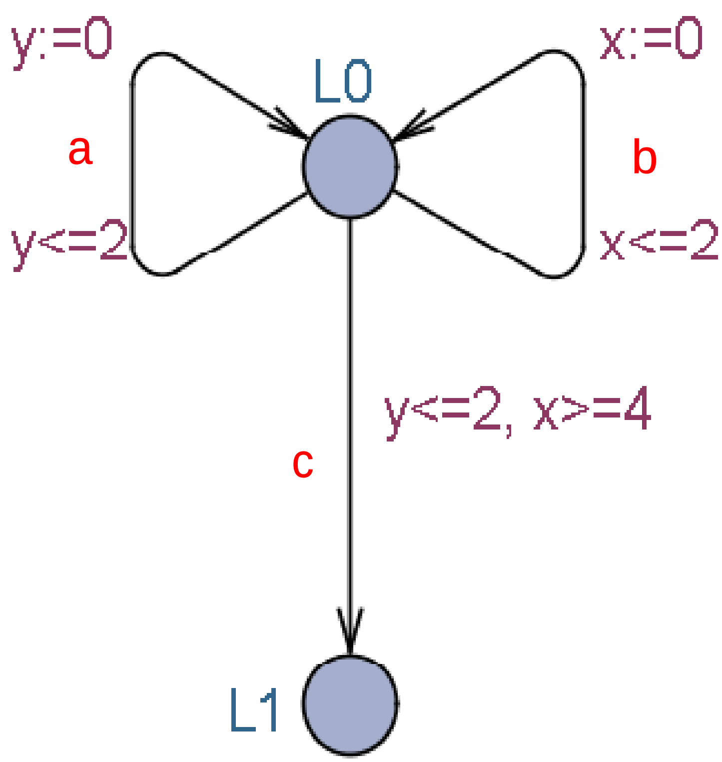
Example

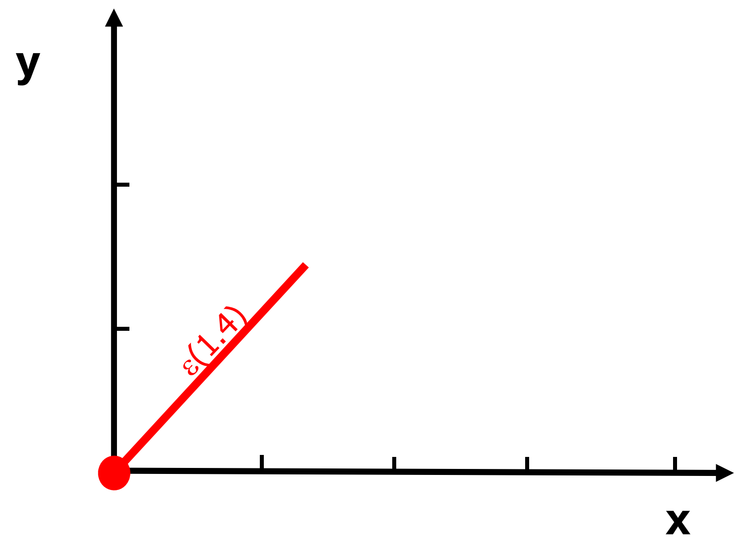




$(\ell_0, x = 0, y = 0)$

Example

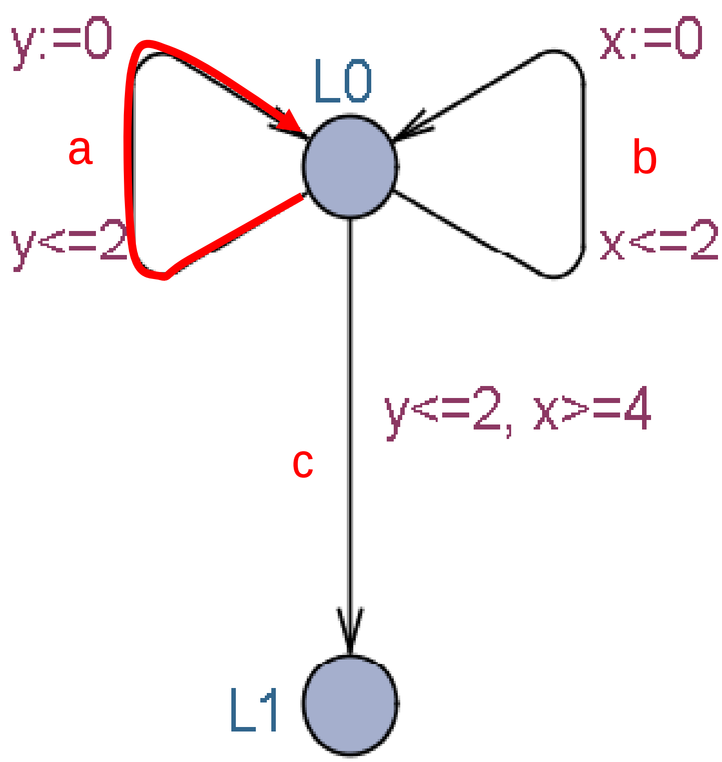


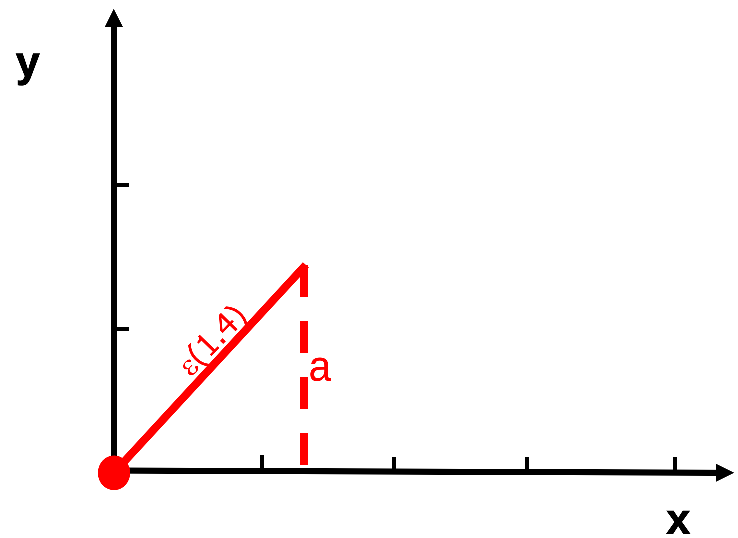


$$(\ell_0, x = 0, y = 0)$$

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

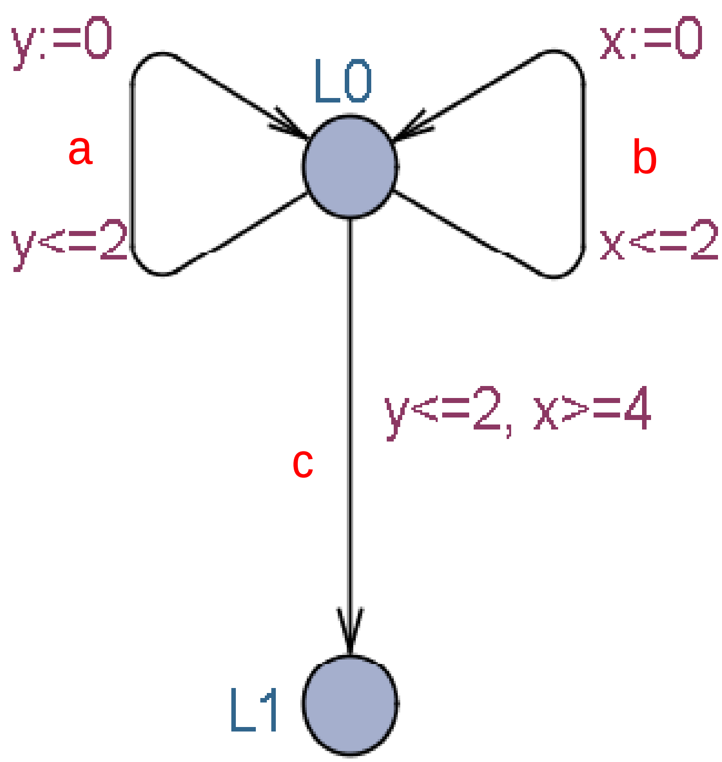
Example

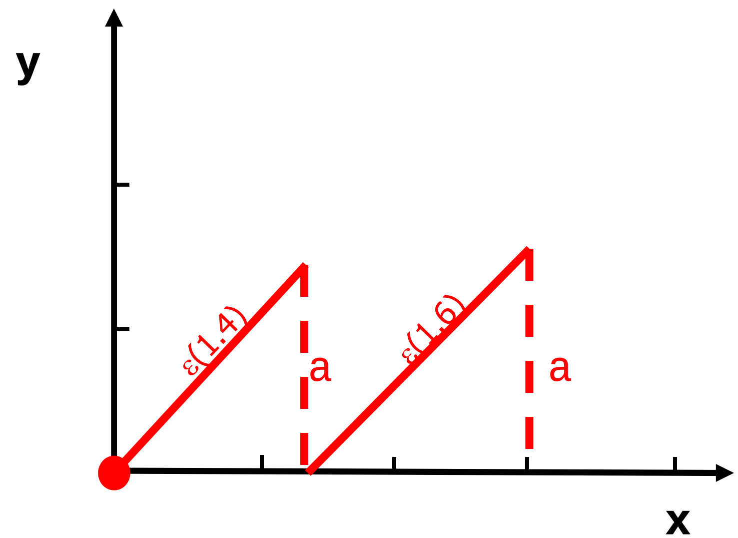




$$(\ell_0, x = 0, y = 0)$$
$$\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)$$
$$\xrightarrow{a} (\ell_0, x = 1.4, y = 0)$$

Example





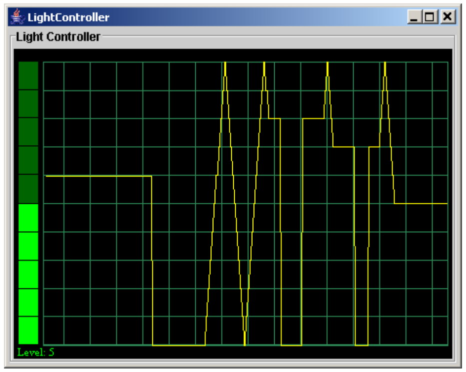
$$(\ell_0, x = 0, y = 0)$$

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

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

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

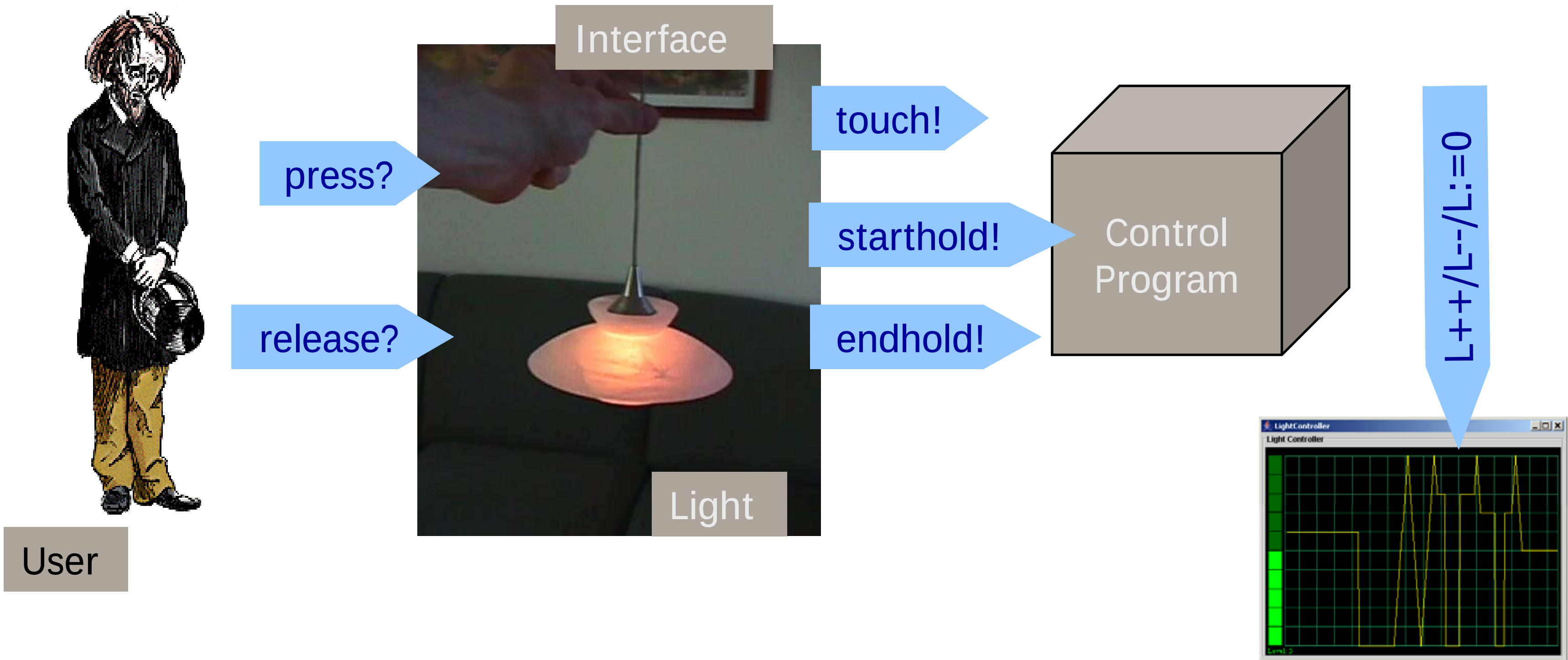
$$\xrightarrow{a} (\ell_0, x = 3.0, y = 0)$$



Light Control Interface

Light Control Interface

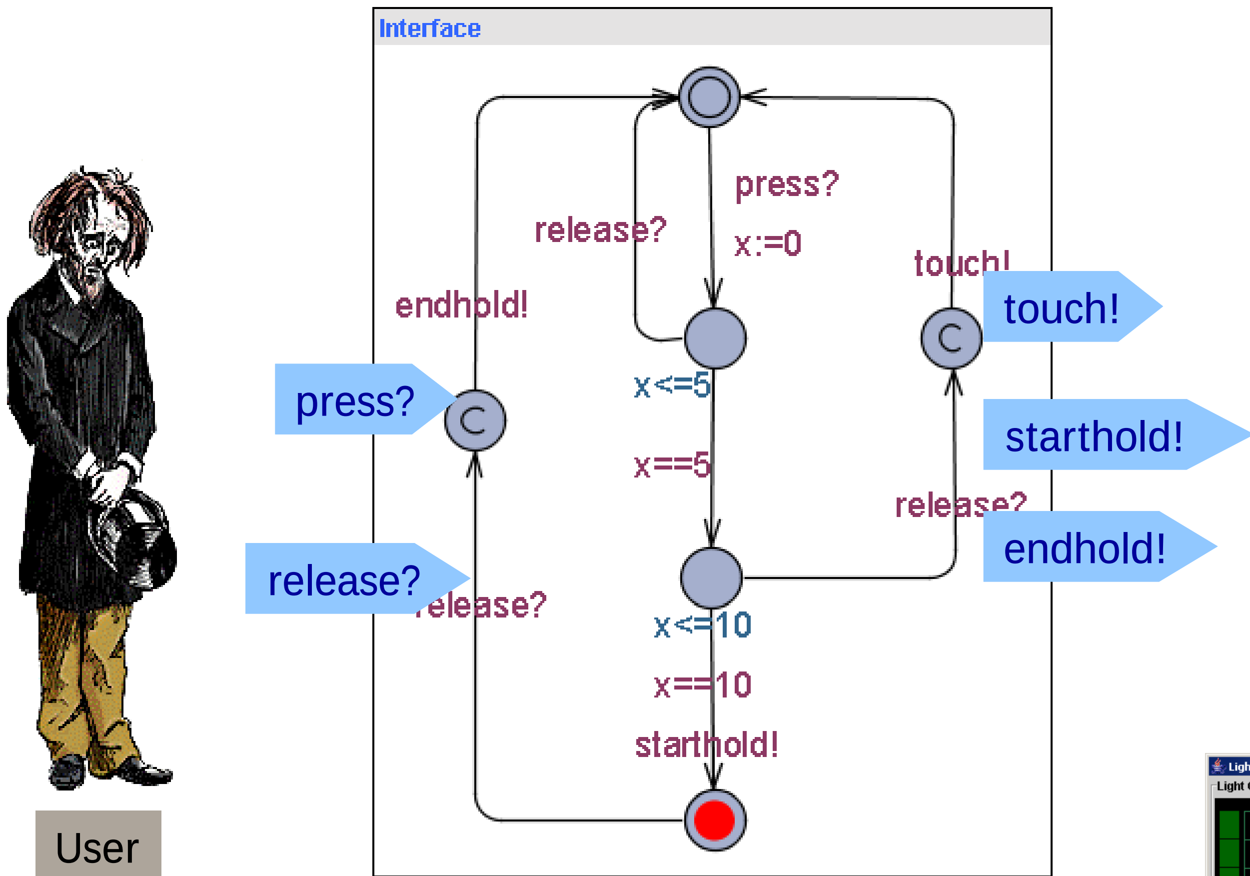
press? d release? → touch! $0.5 \leq d \leq 1$
press? 1 → starthold!
press? d release? → endhold! $d > 1$

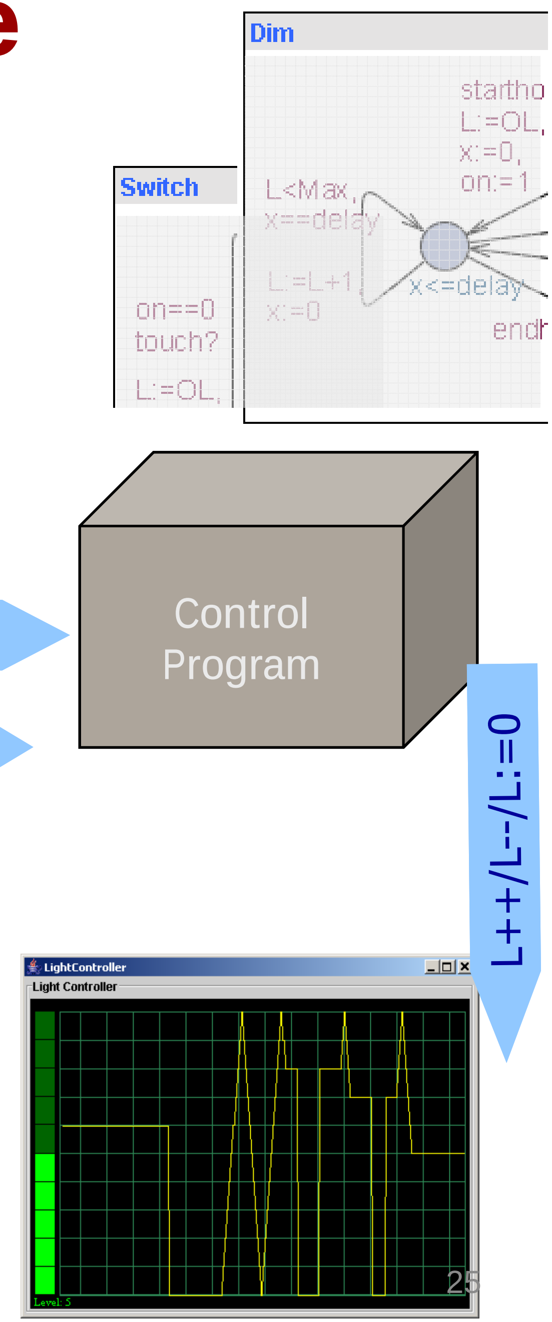


press? 0.2 release? … press? 0.7 release? … press? 1.0 2.4 release? …

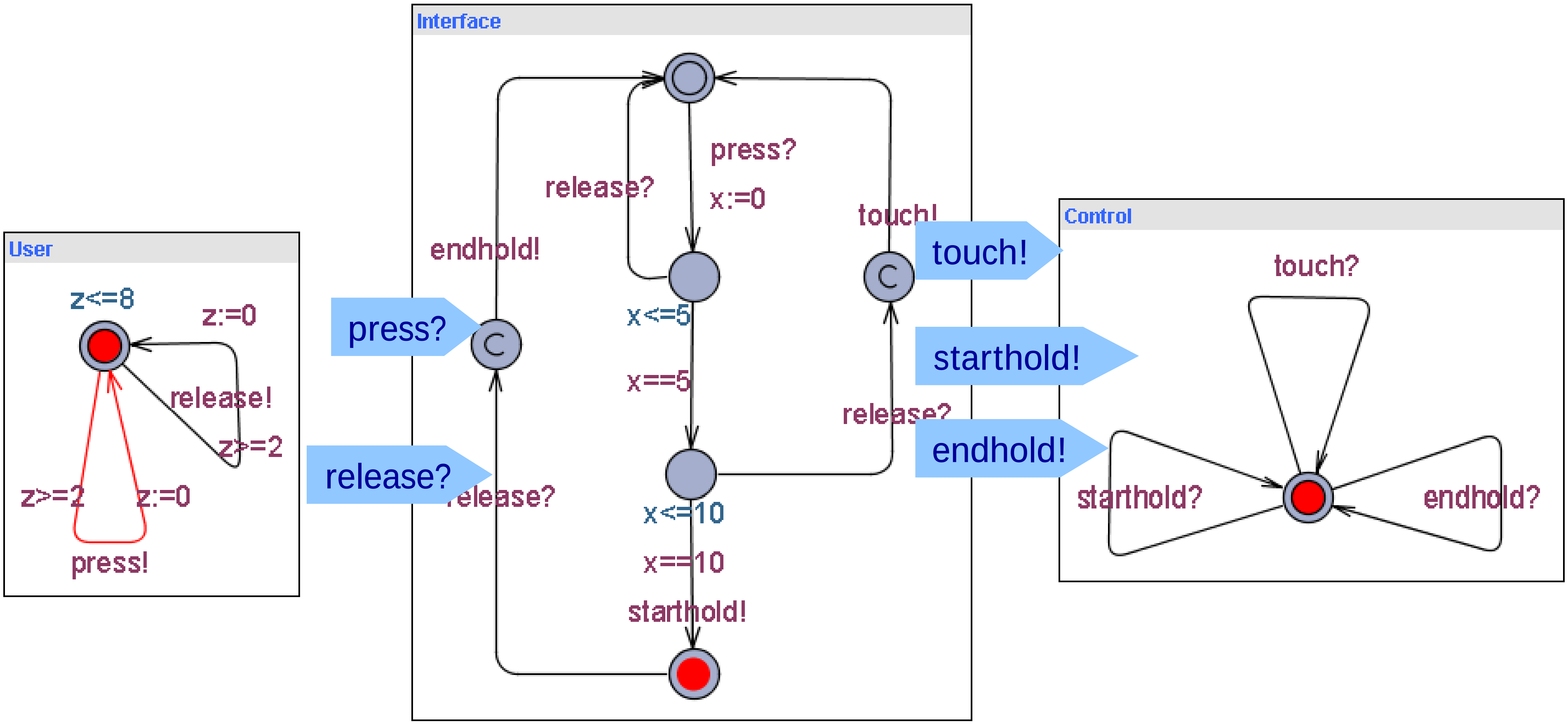
Ø touch! starthold! endhold!

Light Control Interface

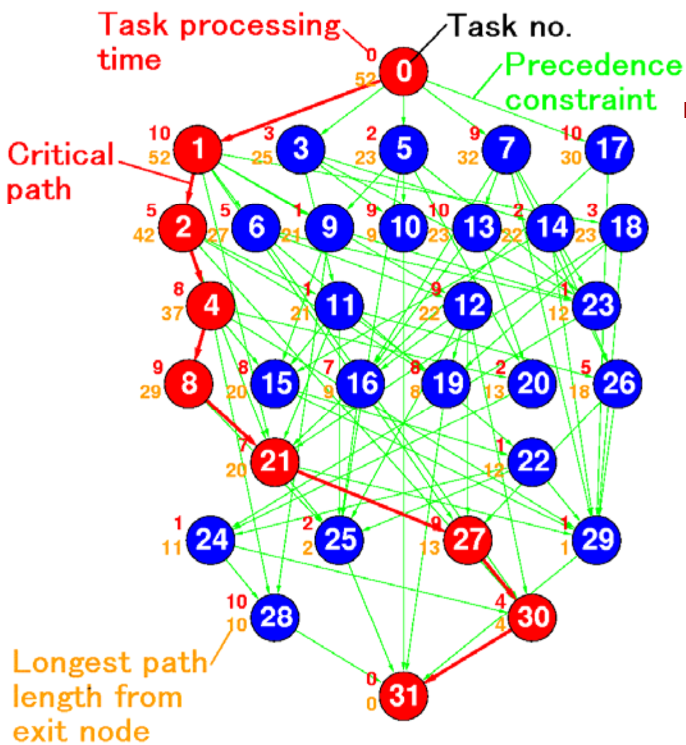




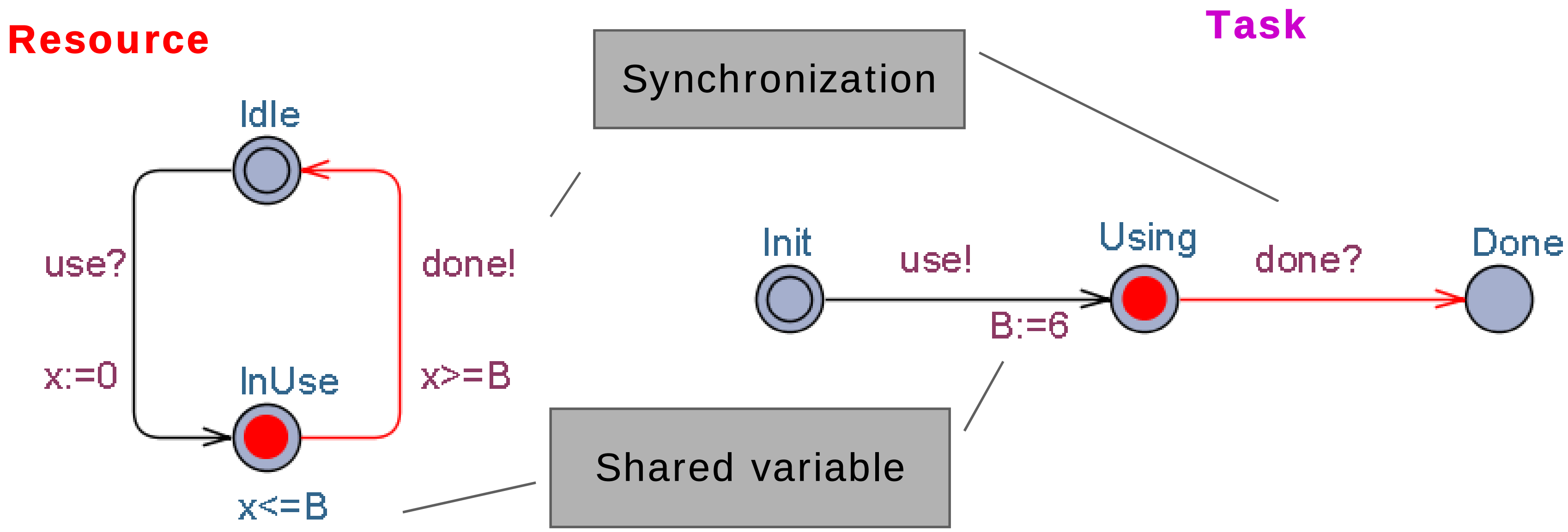
Light Control Network



Task Graph Scheduling



Resources & Tasks & Composition



Semantics:

(Idle , Init , B=0, x=0)

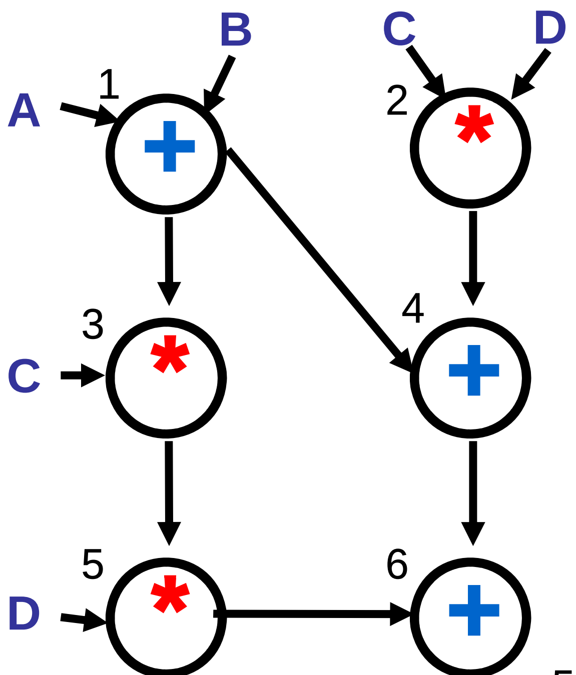
d(3.1415) → (Idle , Init , B=0 , x=3.1415)

use → (InUse , Using , B=6, x=0)

d(6) → (InUse , Using , B=6, x=6)

done → (Idle , Done , B=6 , x=6)

Task Graph Scheduling – Example



Compute :

(D * (C * (A + B)) + ((A + B) + (C * D))

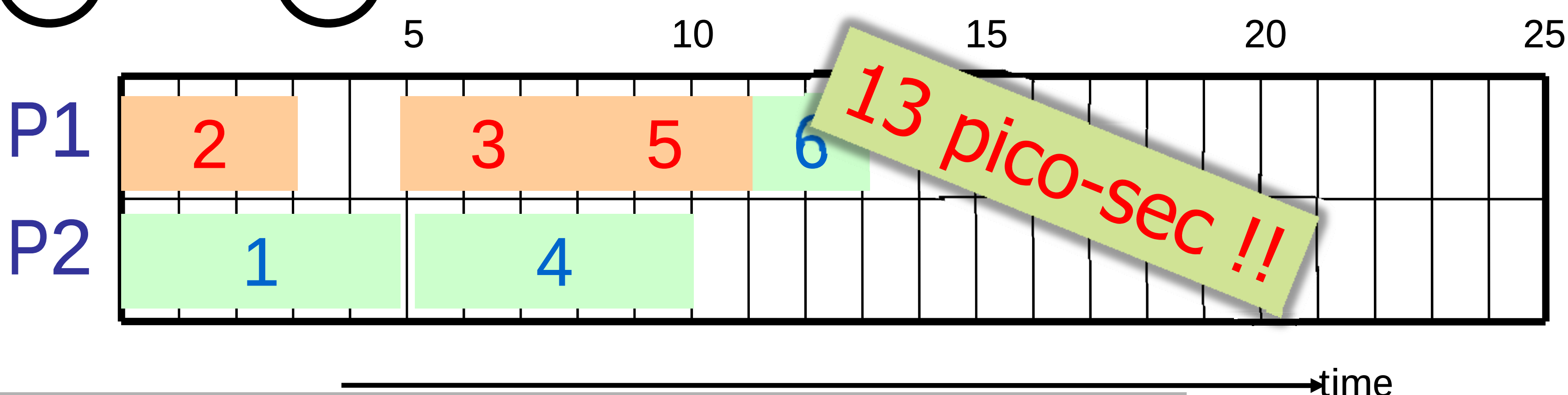
using 2 processors

P1 (fast)

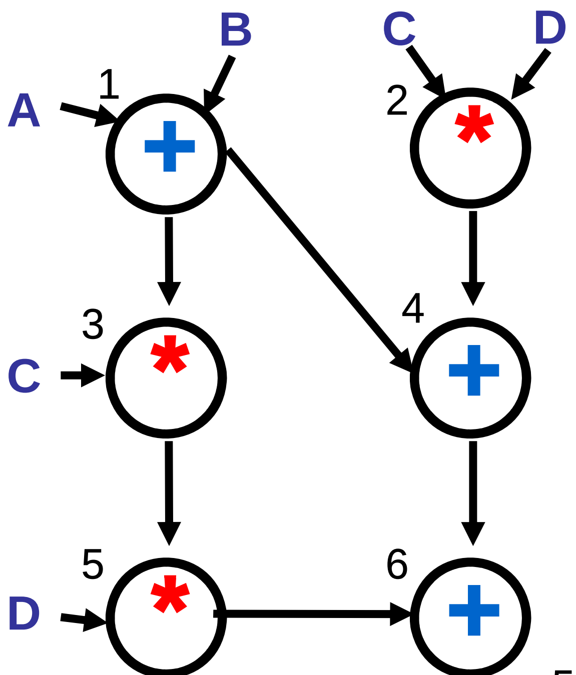
+	2ps
*	3ps

P2 (slow)

+	5ps
*	7ps



Task Graph Scheduling – Example



Compute :

(D * (C * (A + B)) + ((A + B) + (C * D))

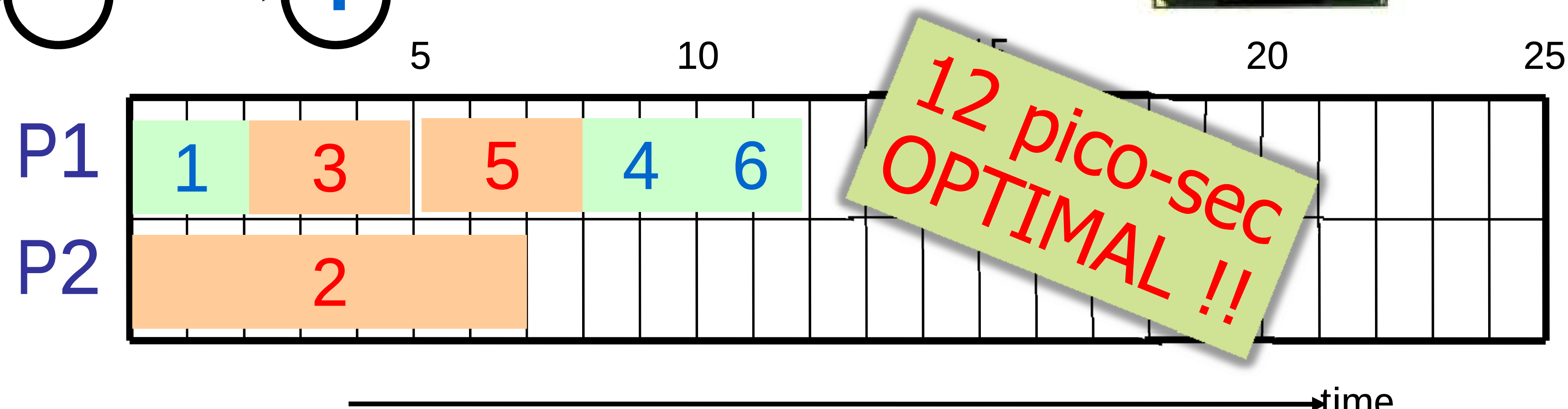
using 2 processors

P1 (fast)

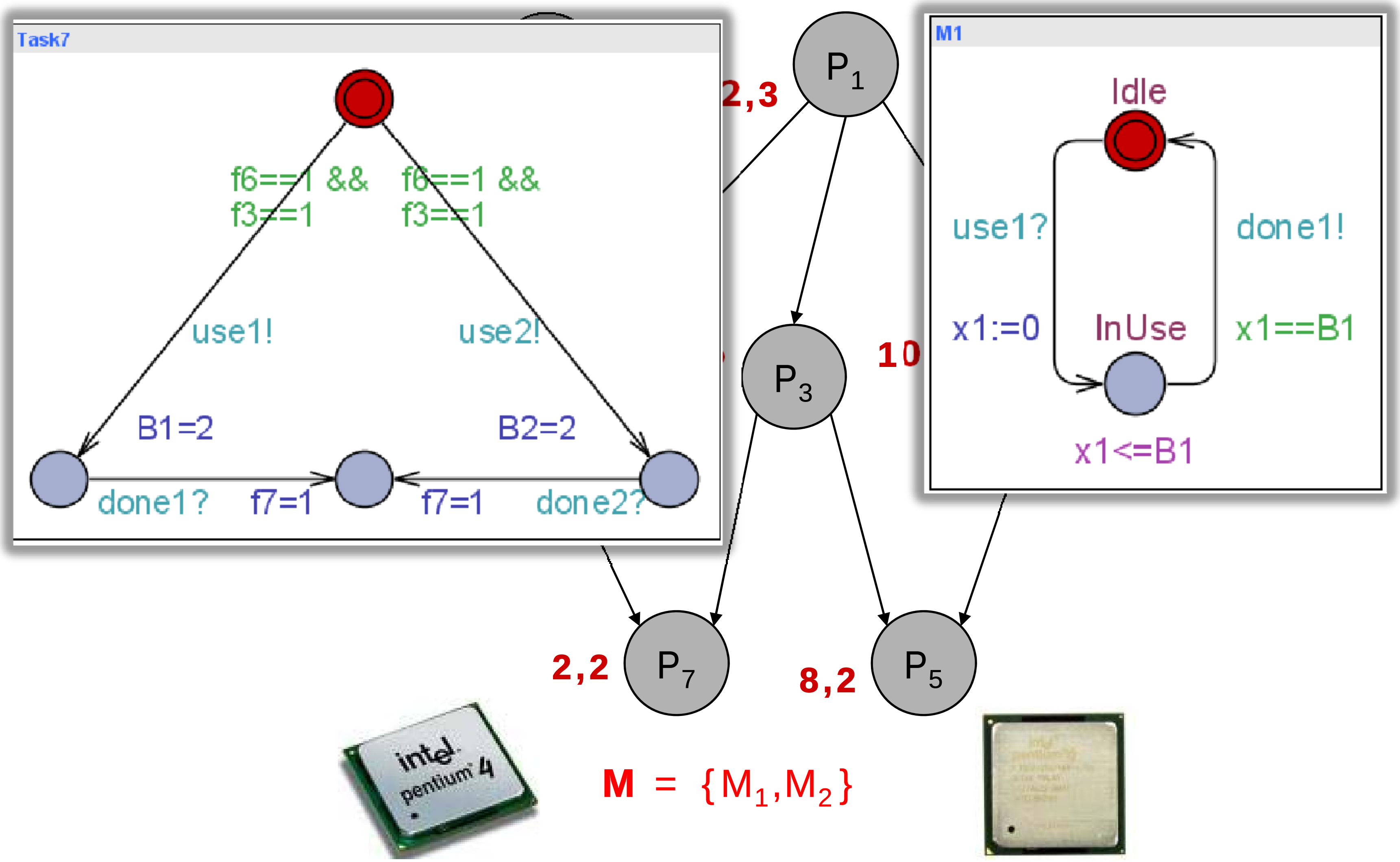
+	2ps
*	3ps

P2 (slow)

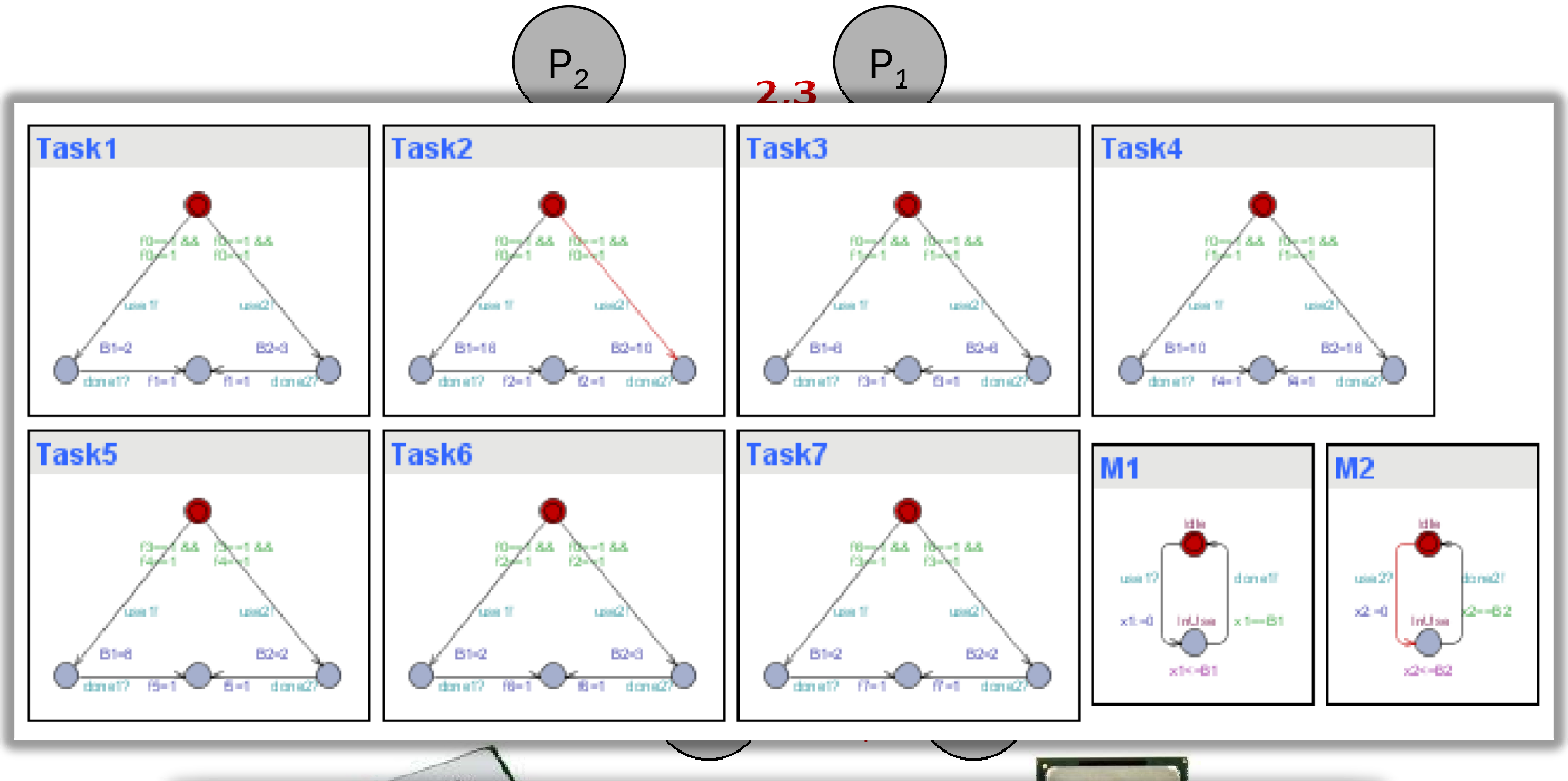
+	5ps
*	7ps



Task Graph Scheduling



Task Graph Scheduling



E<> (Task1.End and … and Task7.End)

Experimental Results

name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.	1150
233	1980	207	19	1118	1121
294	2014	141	17	1257	1261
295	2168	965	18	1318	1322
292	2333	318	3	8009	8009
298	2399	303	10	2471	2473

AMETIST
advanced methods for timed systems

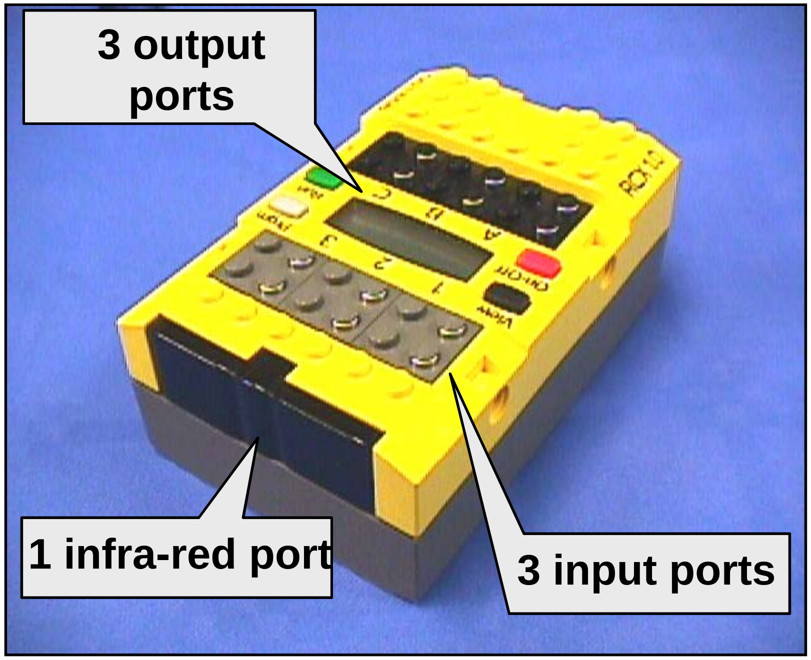
Symbolic A*
Branch-&-Bound
60 sec

Abdeddaïm, Kerbaa, Maler

Brick Sorting

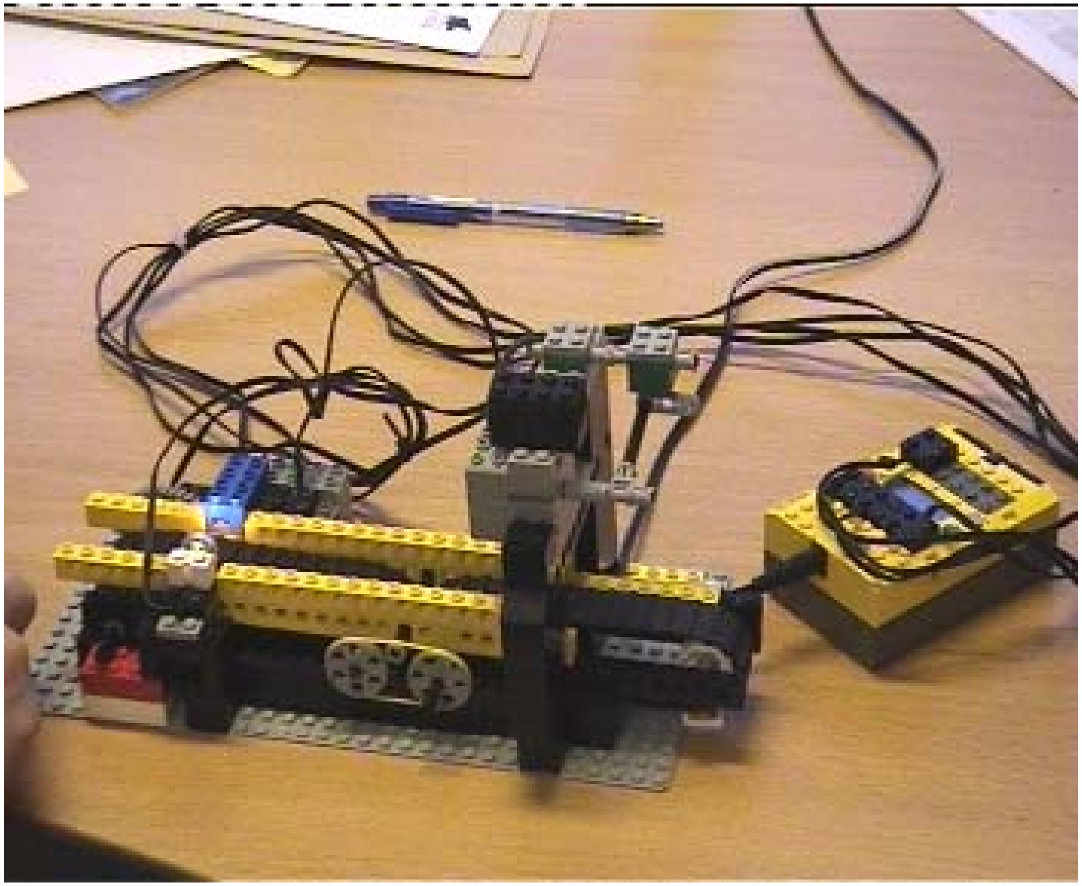
LEGO Mindstorms/RCX

- 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

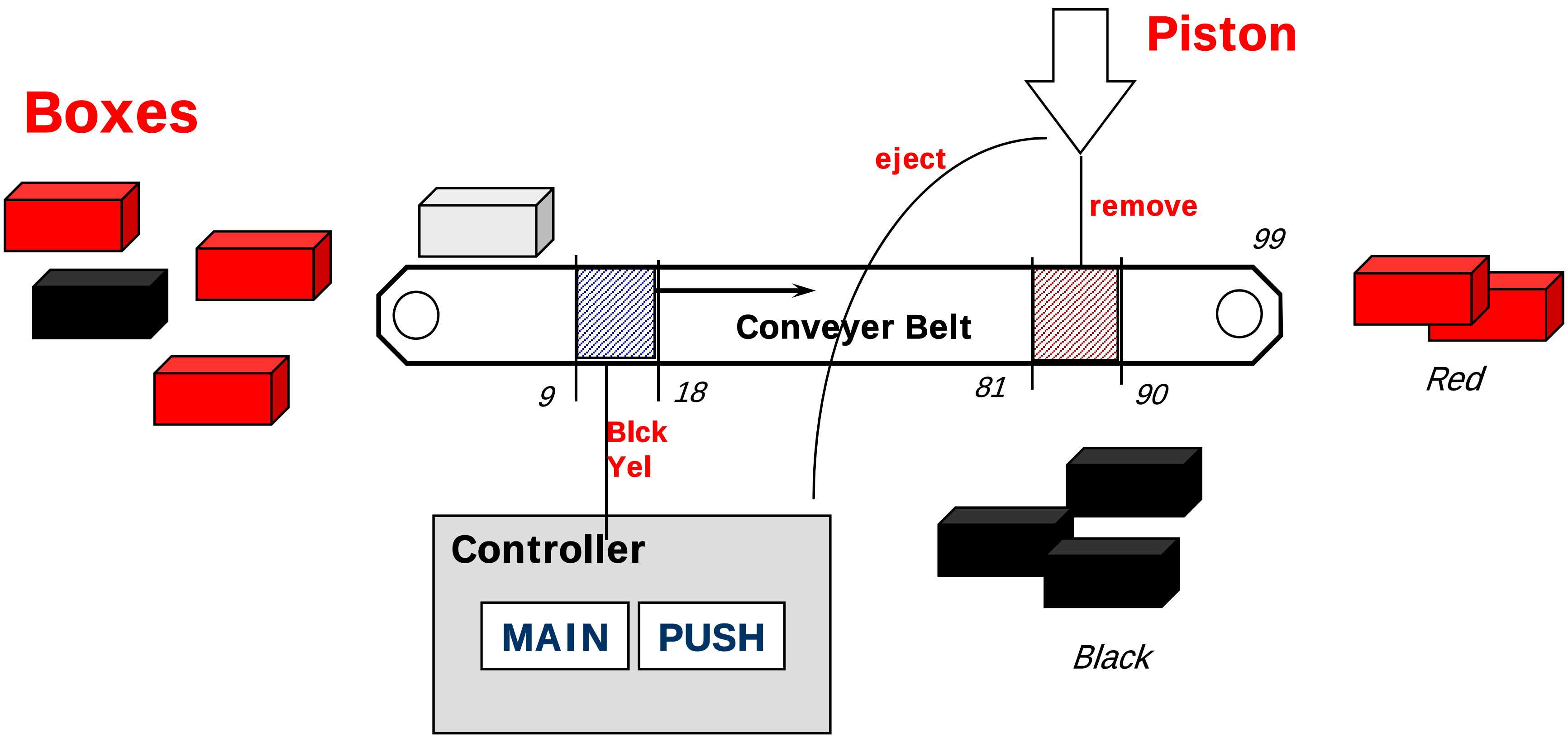


Controller Program
LEGO MINDSTORM

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



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

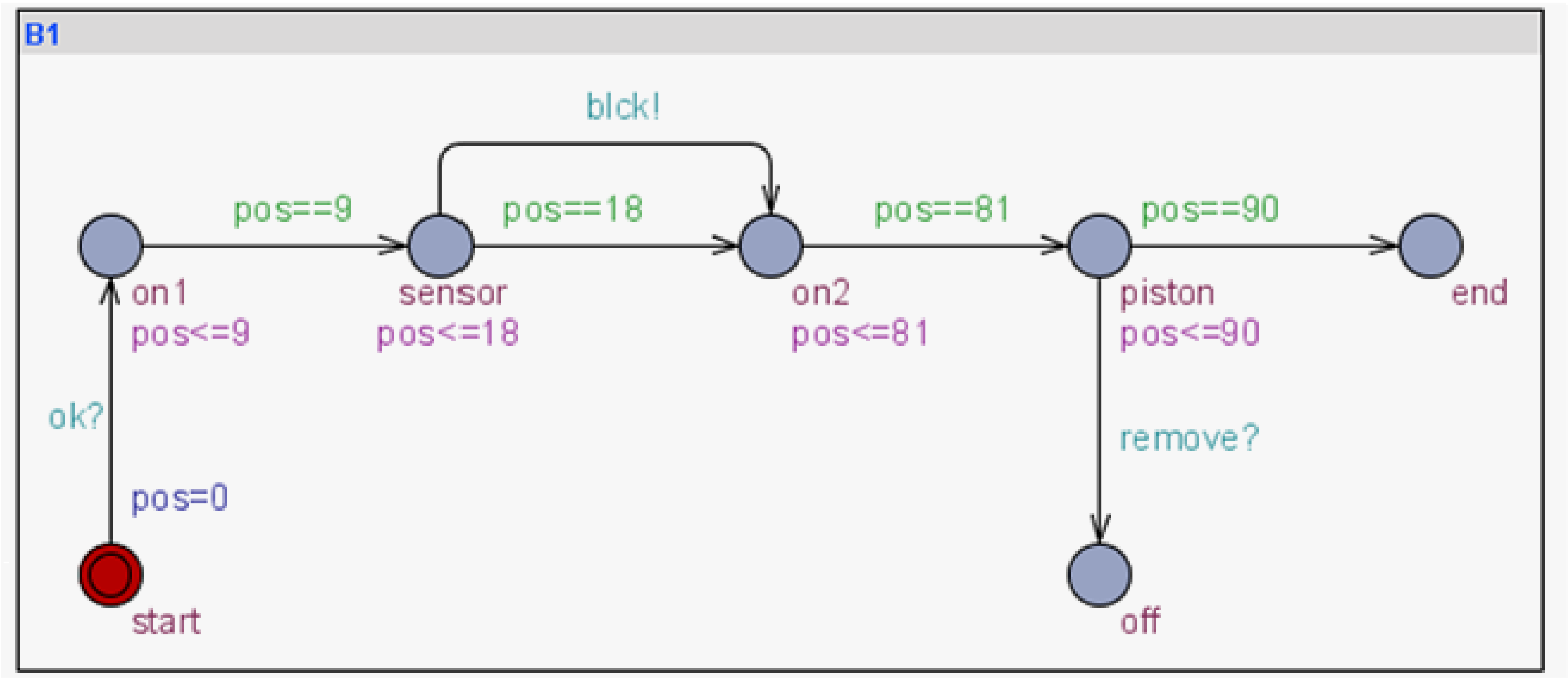
NQC programs

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

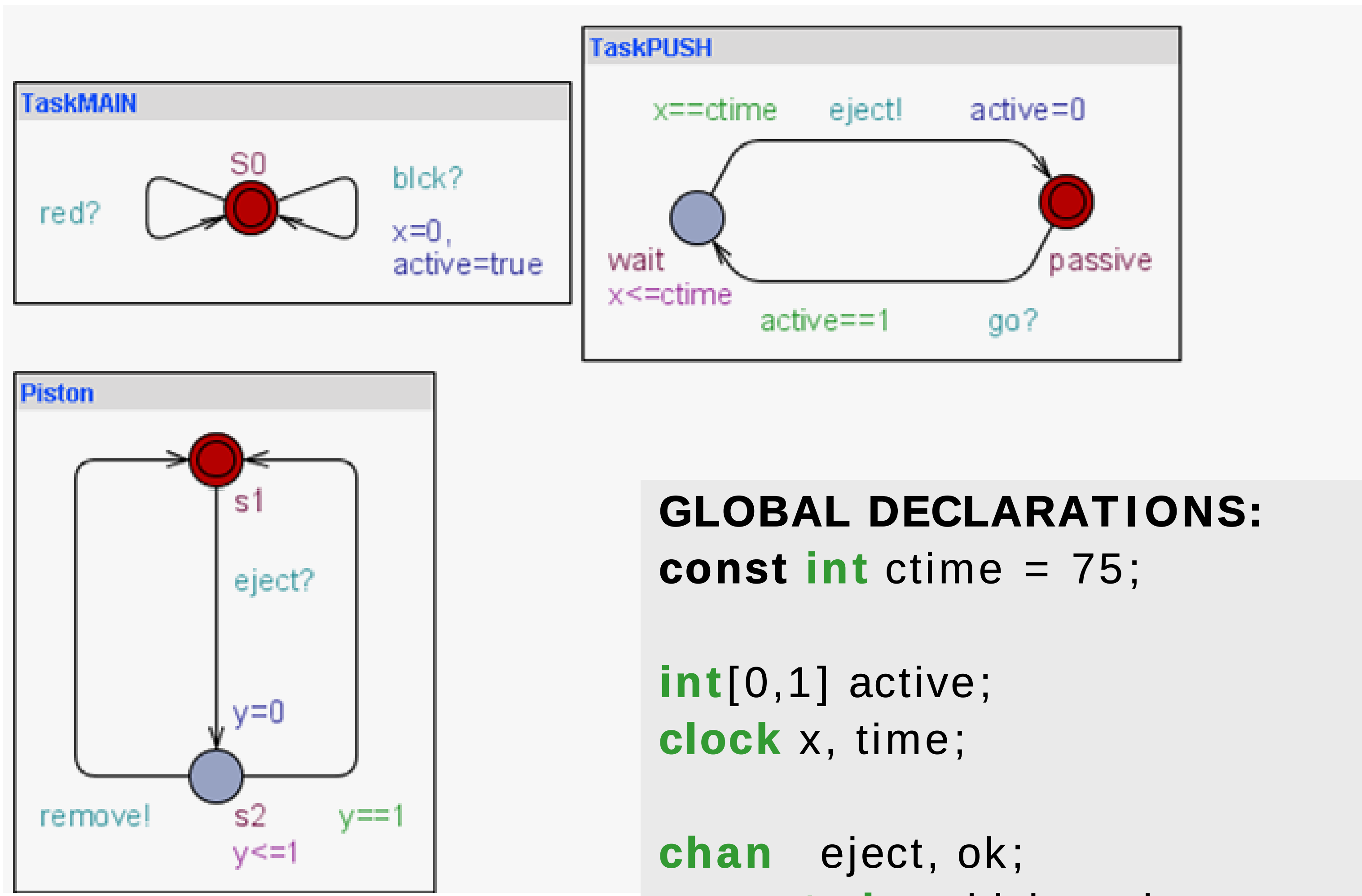
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 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;

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

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