
# Controller Synthesis for Real Time systems

Kim G. Larsen

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

# UPPAAL Branches

- Real Time Verification


CLASSIC

- Real Time Controller Synthesis

TIGA

- Real Time Scheduling & Planning

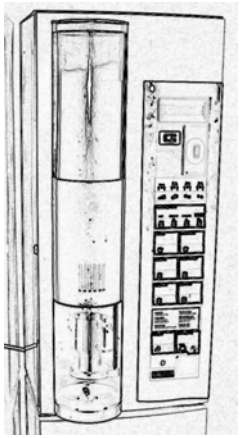
CORA

- Real Time Testing

TRON

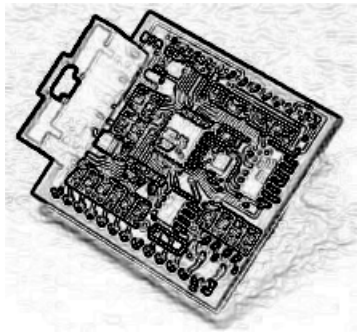
BRICS Basic Research in Computer Science

# Real Time Systems



sensors →

← actuators



**Plant**
*Continuous*

**Controller Program**
*Discrete*

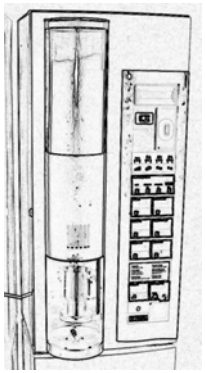
**Eg.:** Realtime Protocols
Pump Control
Air Bags
Robots
Cruise Control
ABS
CD Players
Production Lines

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

BRICS
Basic Research
in Computer Science

# Real Time Model Checking

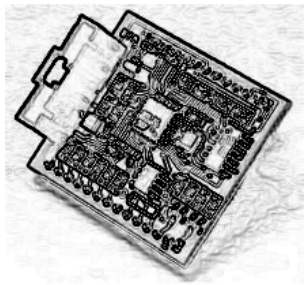
**Plant**
*Continuous*



**sensors**

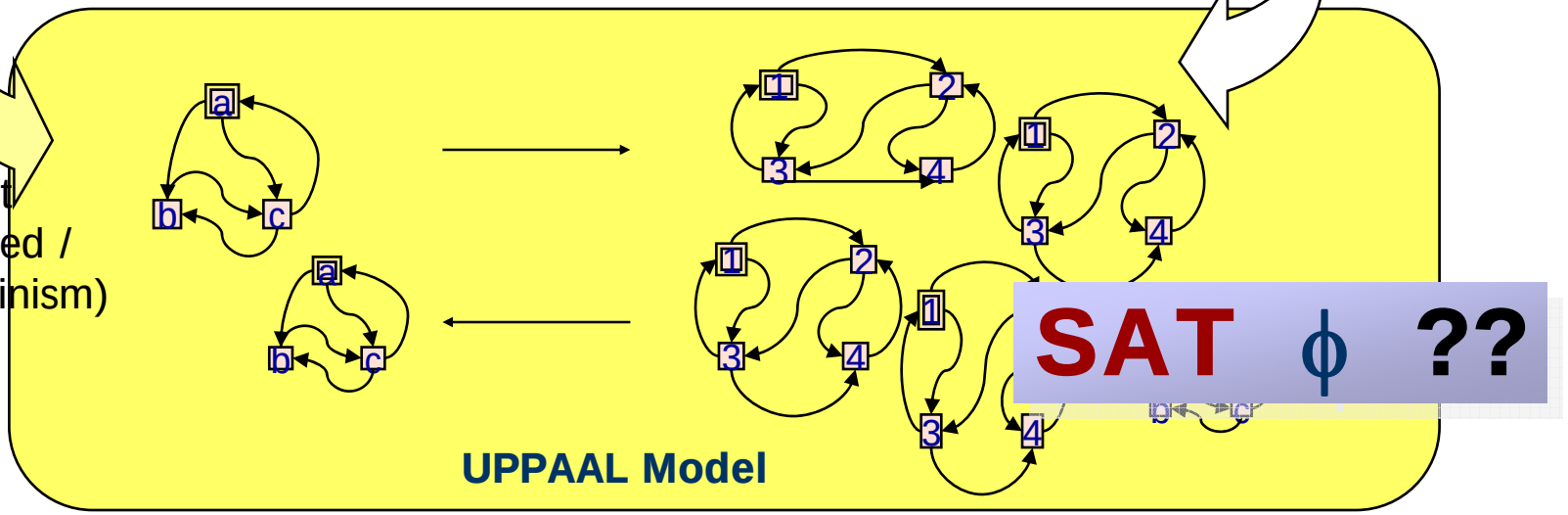
**actuators**

**Controller Program**
*Discrete*



Model of tasks (automatic?)

Model of environment (user-supplied / non-determinism)

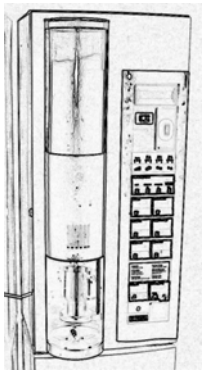



**SAT $\phi$ ??**

BRICS Basic Research in Computer Science

# Real Time Control Synthesis

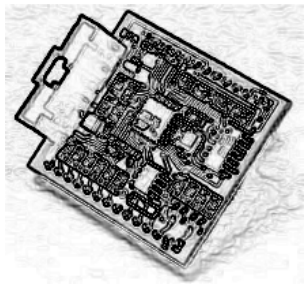
**Plant**
*Continuous*



**sensors**

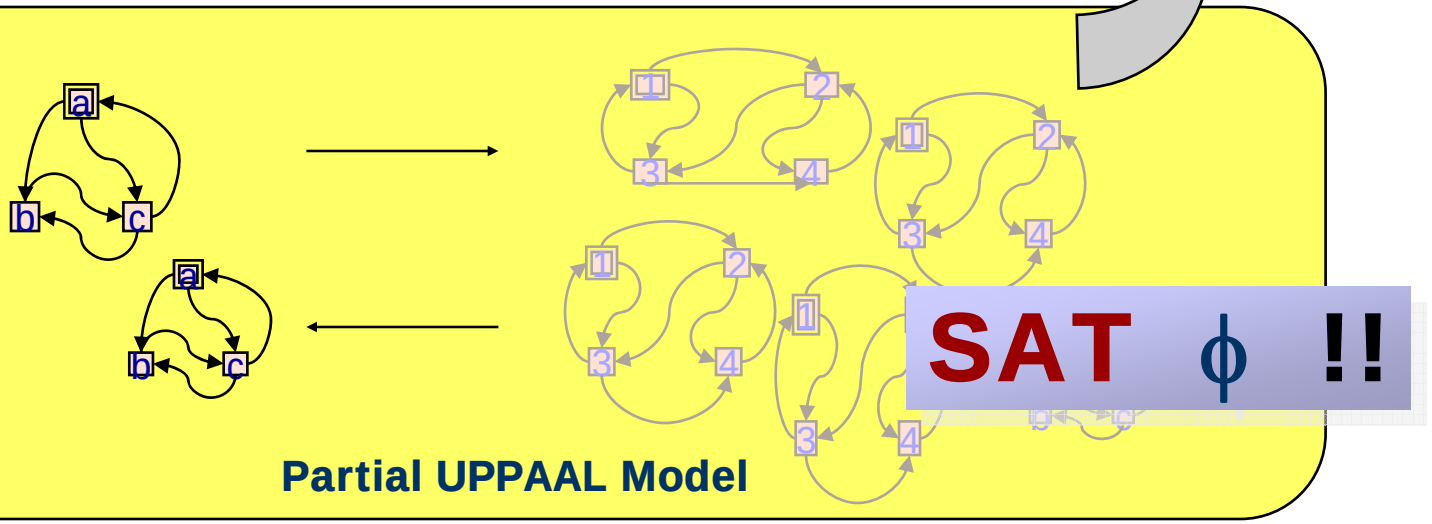
**actuators**

**Controller Program**
*Discrete*



Synthesis
of
tasks/scheduler
(automatic)

Model
of
environment
(user-supplied)




BRICS
Basic Research
in Computer Science


CLASSIC

# Real Time Verification

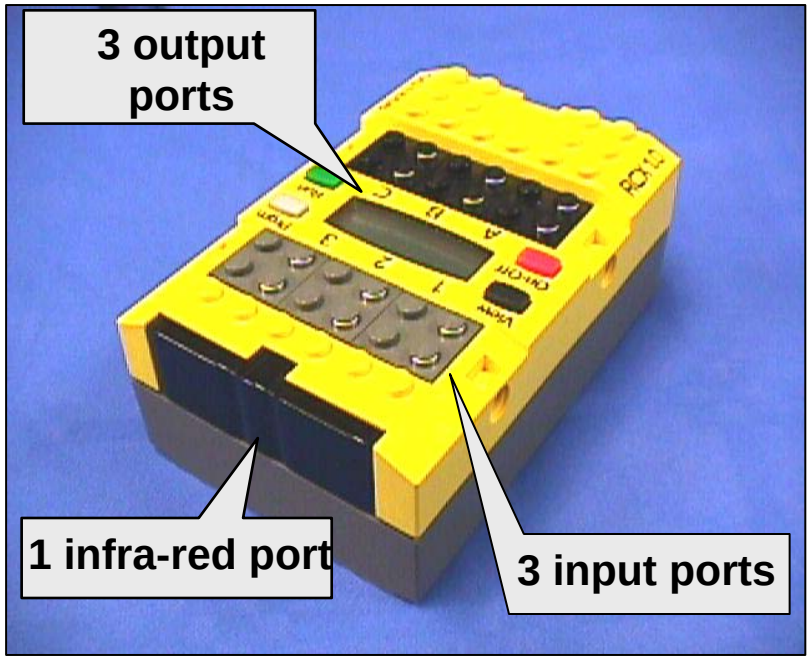
AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CSS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

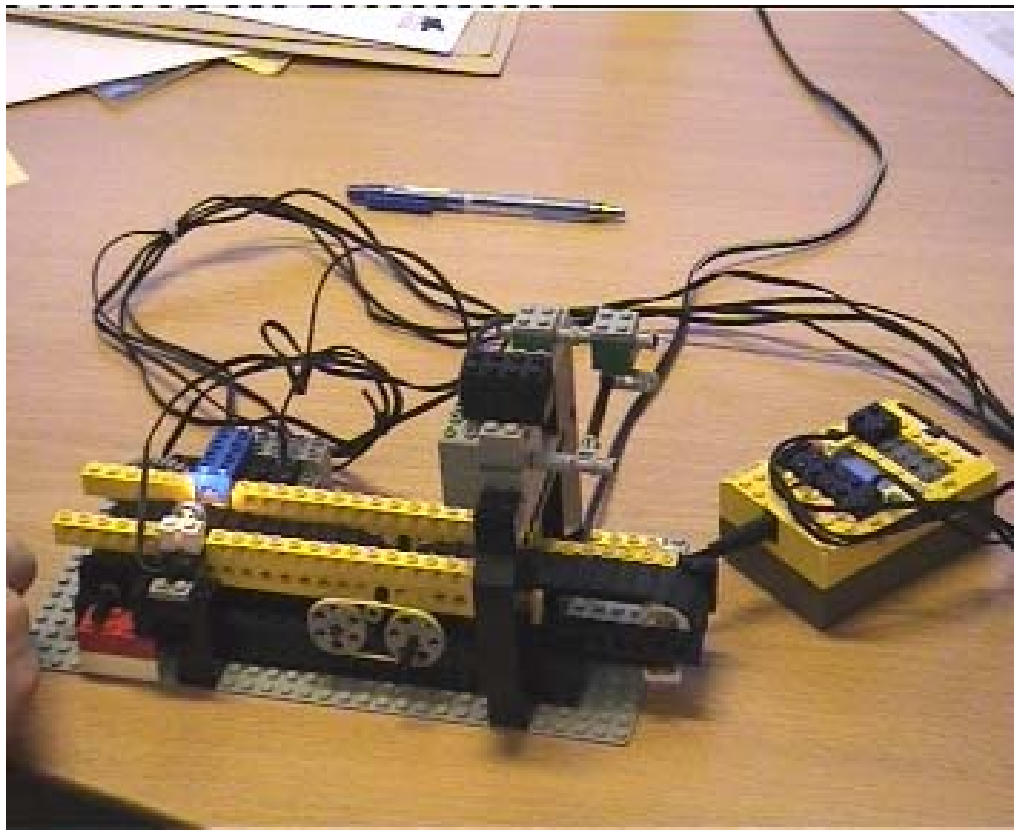
# LEGO Mindstorms/RCX

- 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



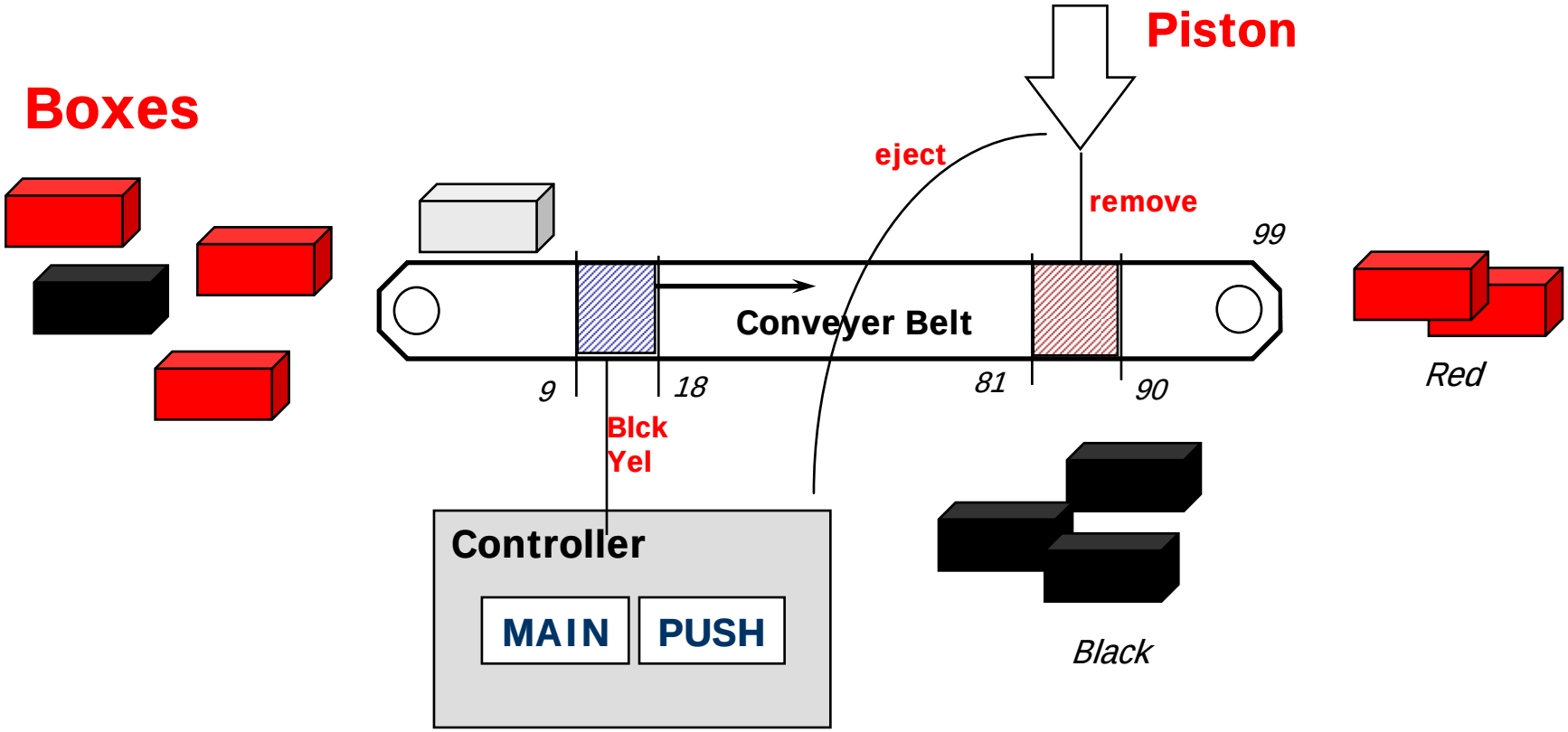
**Controller Program**
LEGO MINDSTORM

BRICS
Basic Research
in Computer Science

# 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;
```

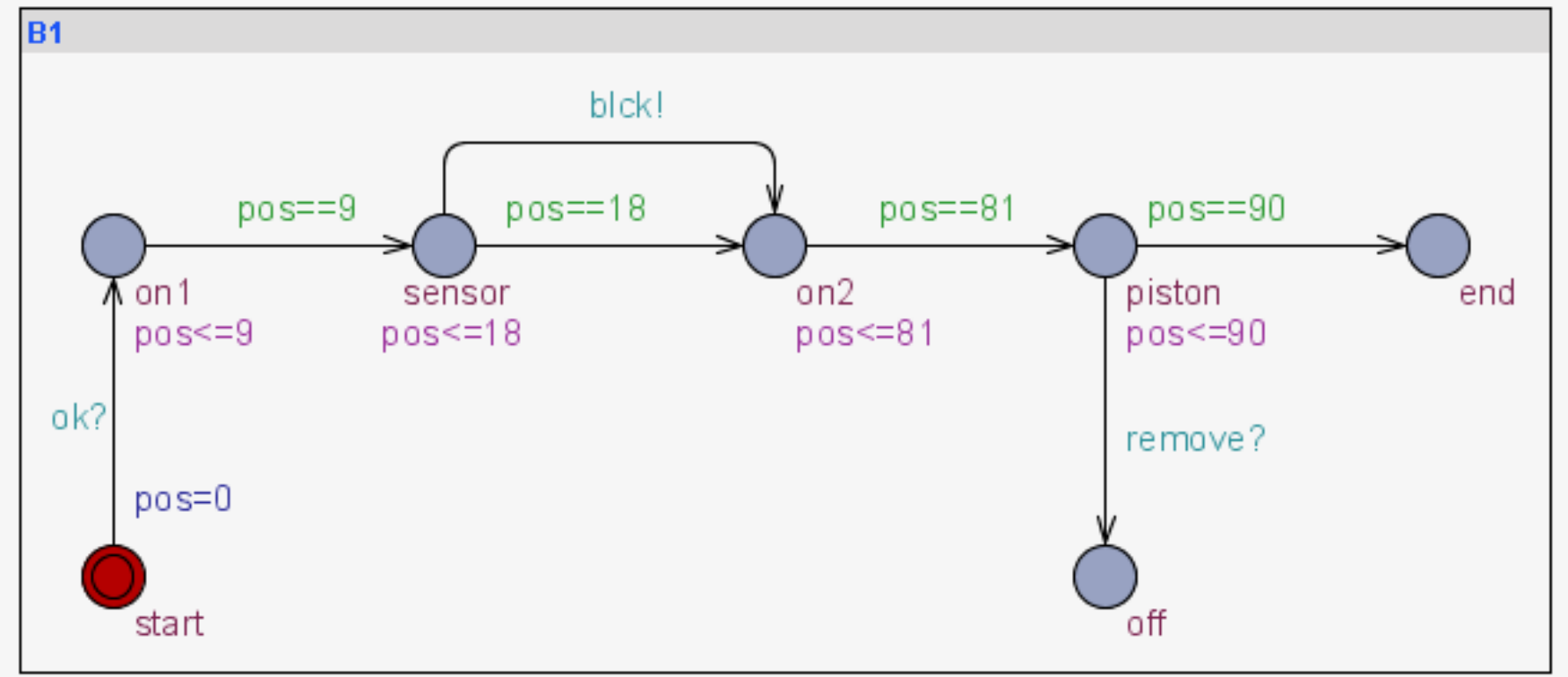
```
task MAIN{
 DELAY=75;
 LIGHT_LEVEL=35;
 active=0;
 Sensor(IN_1, IN_LIGHT);
 Fwd(OUT_A,1);
 Display(1);

 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);
  }
}
```

# UPPAAL Demo

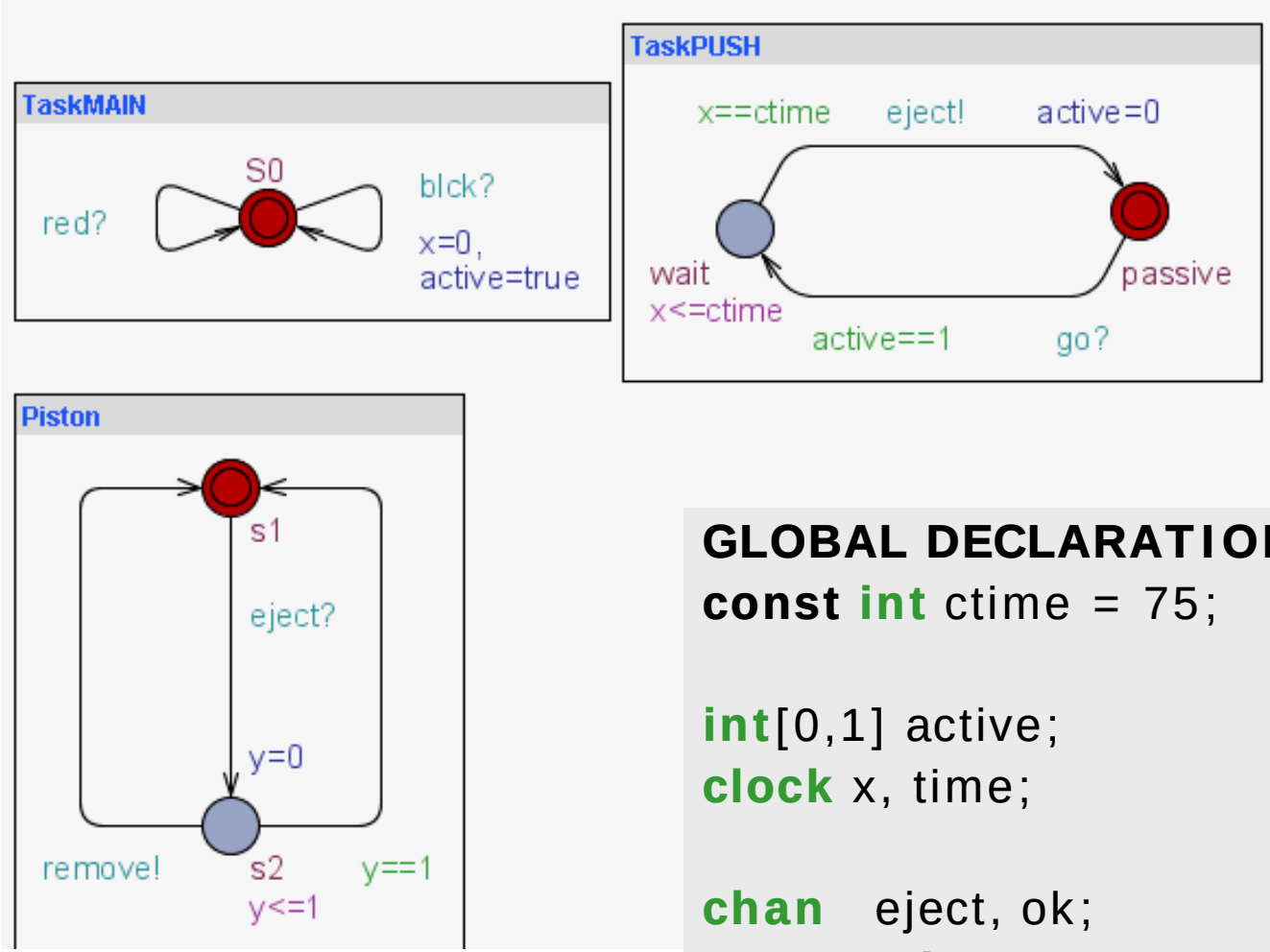

BRICS
Basic Research
in Computer Science

# A Black Brick




BRICS
Basic Research
in Computer Science

# 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;
```

BRICS
Basic Research
in Computer Science


TIGA

Efficient

# Real Time Controller Synthesis

with

Franck Cassez, Agnes Counard, Alexandre David
Emmanuel Fleury, Didier Lime

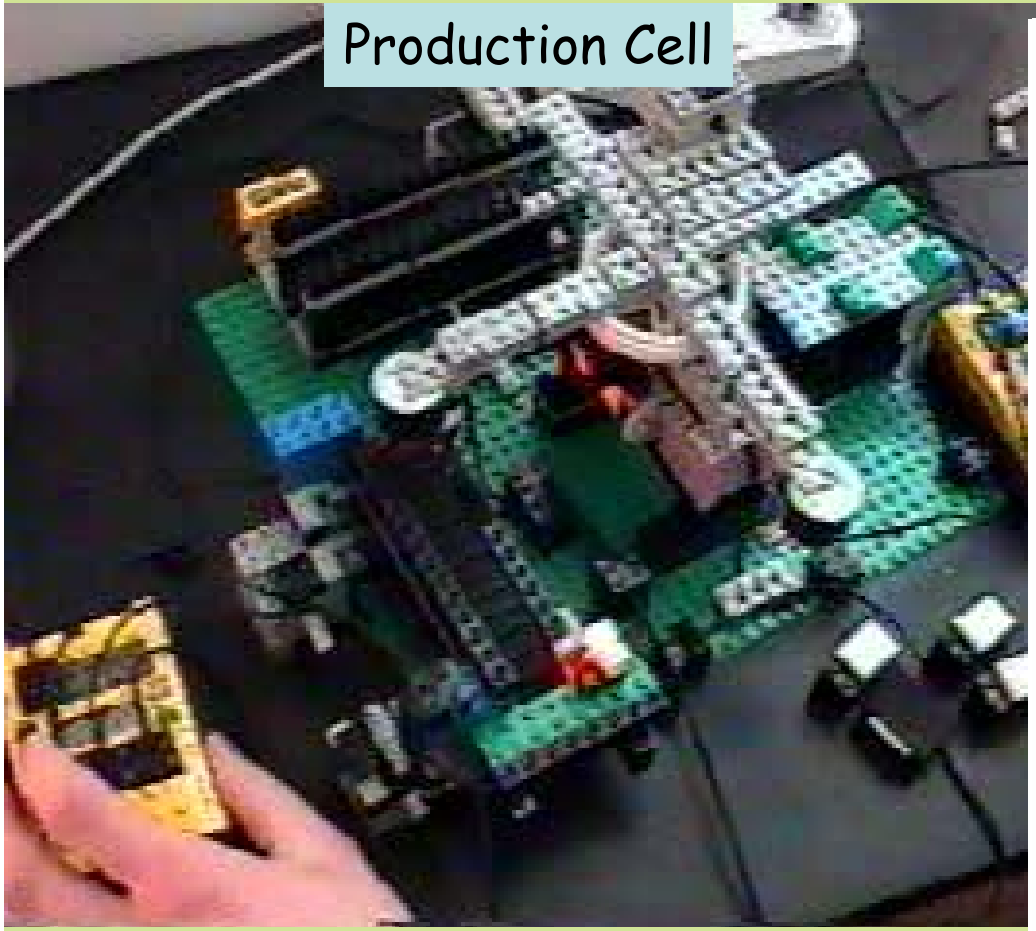
Presented at CONCUR'05

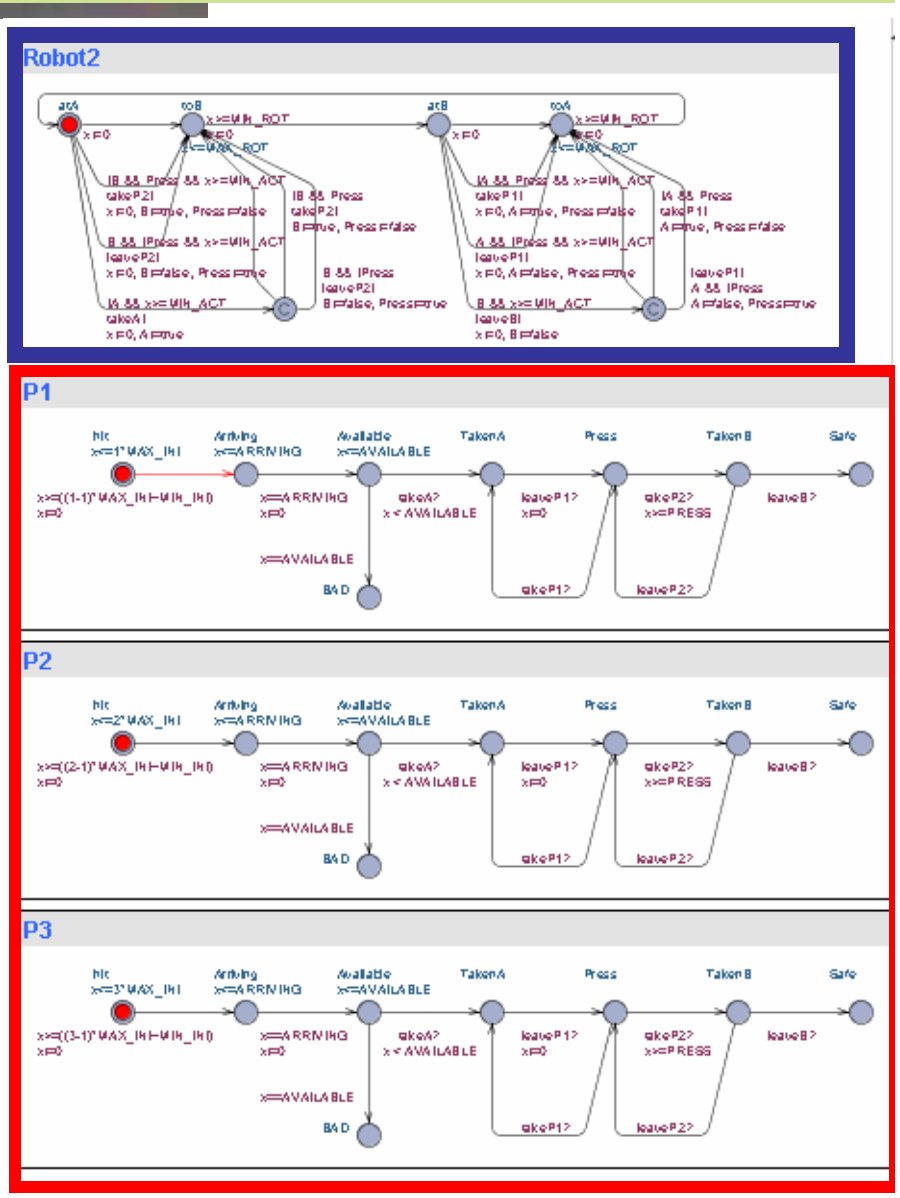
AALBORG UNIVERSITY DENMARK

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

# Controller Synthesis and Timed Games



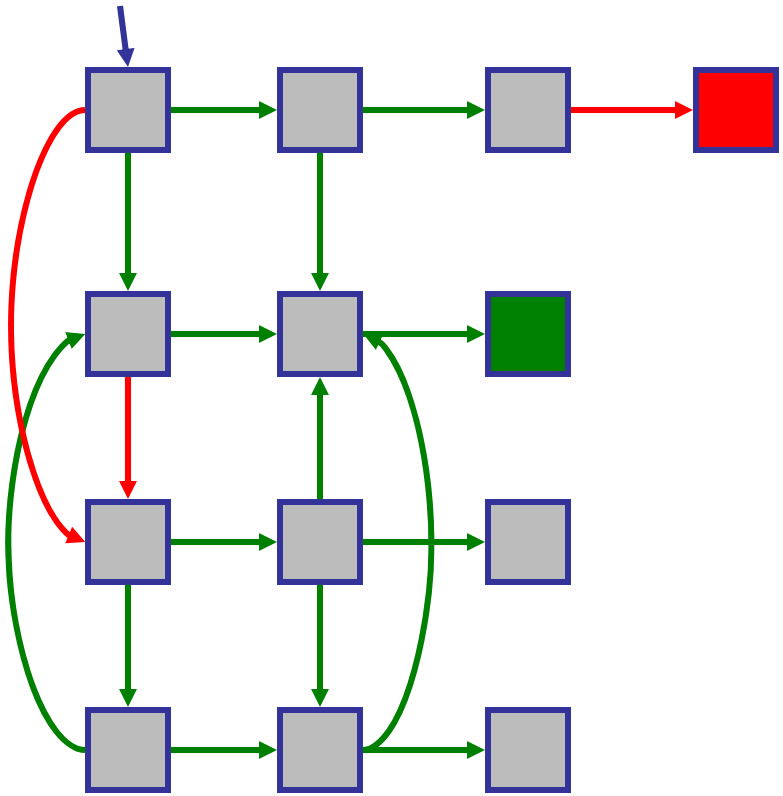

**GIVEN** System moves $S$,
Controller moves $C$, and property $\phi$
**FIND** strategy $s_C$ such that $s_C || S \models \phi$

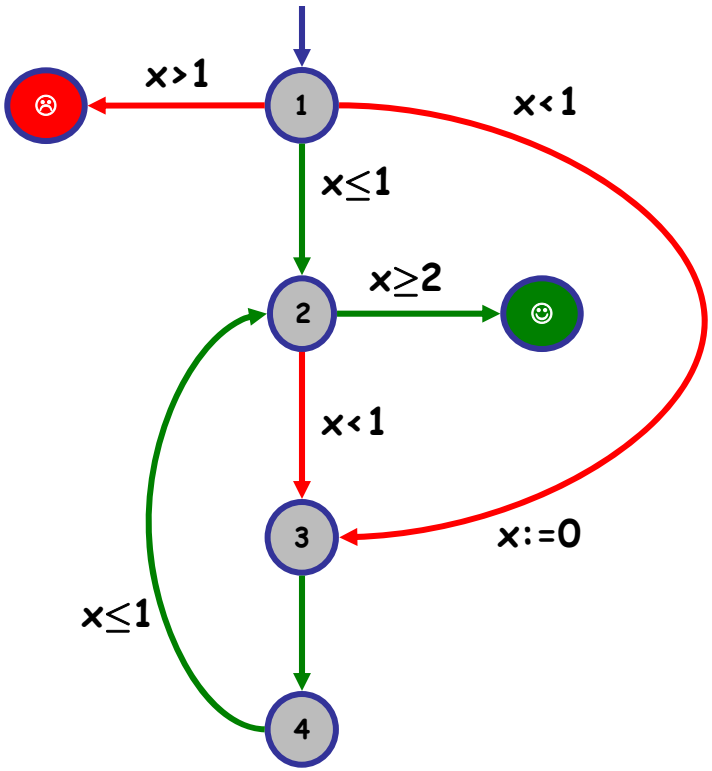
→

**A Two-Player Game**

# Untimed and Timed Games

Reachability / Safety Games



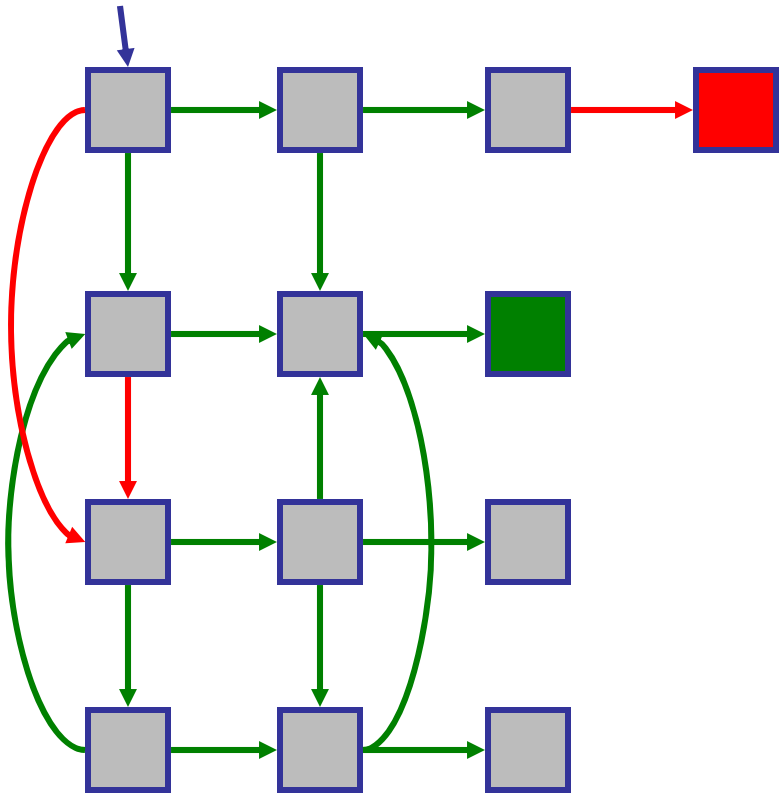



Uncontrollable

Controllable

# Untimed Games

Reachability / Safety Games



Strategy:

$F : Run(A) \rightarrow E_c$

Memoryless strategy:

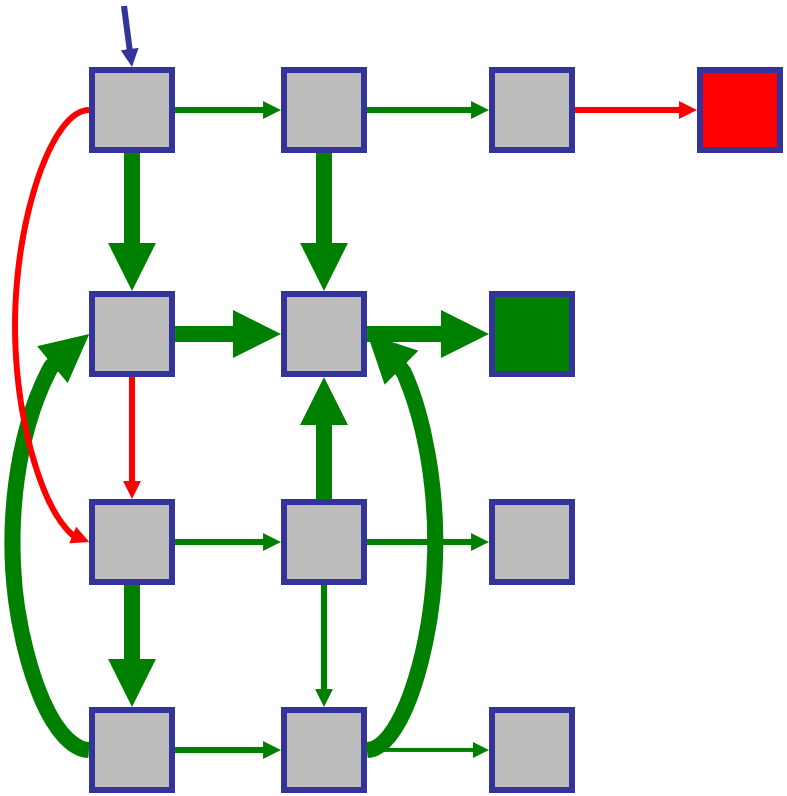
$F : Q \rightarrow E_c$

Uncontrollable

Controllable

# Untimed Games

Reachability / Safety Games



Strategy:

$F : Run(A) \rightarrow E_c$

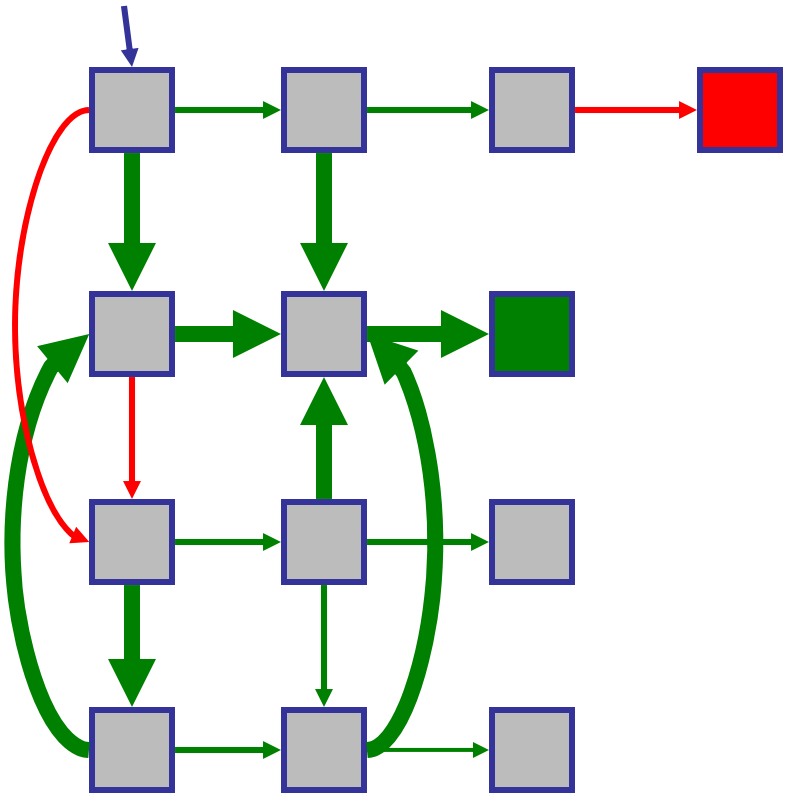
Memoryless strategy:

$F : Q \rightarrow E_c$

Uncontrollable

Controllable

# Untimed Games



Reachability / Safety Games

Strategy:

$F : Run(A) \rightarrow E_c$

Memoryless strategy:

$F : Q \rightarrow E_c$

Winning Run:

$States(\rho) \cap G \neq \emptyset$

$States(\rho) \cap B = \emptyset$

Winning Strategy:

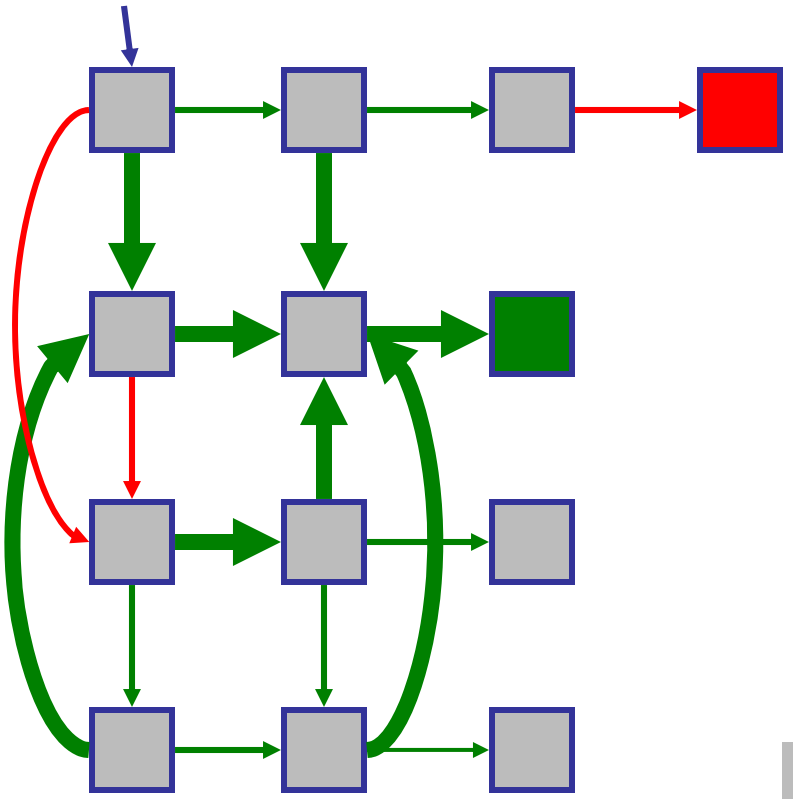
$Runs(F) \subseteq WinRuns$

Loosing (memoryless) strategy

→ Uncontrollable

→ Controllable

# Untimed Games



Reachability / Safety Games

Strategy:

$F : Run(A) \rightarrow E_c$

Memoryless strategy:

$F : Q \rightarrow E_c$

Winning Run:

$States(\rho) \cap G \neq \emptyset$

$States(\rho) \cap B = \emptyset$

Winning Strategy:

$Runs(F) \subseteq WinRuns$

**Winning (memoryless) strategy)**

→ Uncontrollable

→ Controllable

# Timed Games

Reachability / Safety Games

Strategy:

$F : Run(A) \rightarrow E_c \cup \lambda$

Memoryless strategy:

$F : Q \rightarrow E_c \cup \lambda$

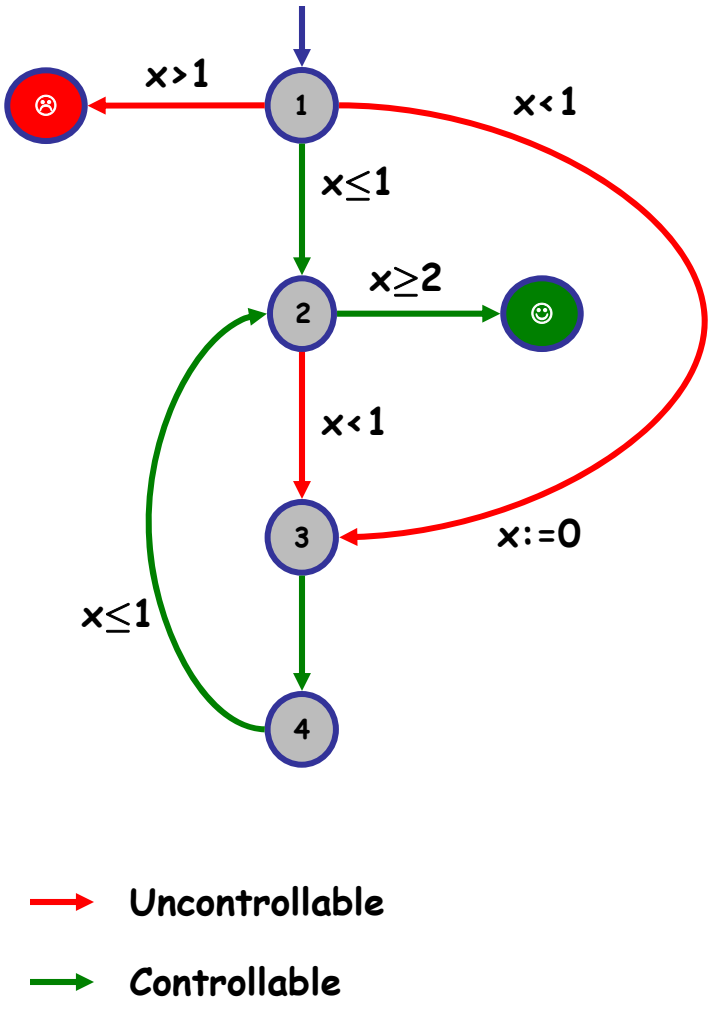
Winning Run:

$States(\rho) \cap G \neq \emptyset$

$States(\rho) \cap G = \emptyset$

Winning Strategy:

$Runs(F) \subseteq WinRuns$



# Timed Games

Reachability / Safety Games

Strategy:

$F : Run(A) \rightarrow E_c \cup \lambda$

Memoryless strategy:

$F : Q \rightarrow E_c \cup \lambda$

Winning Run:

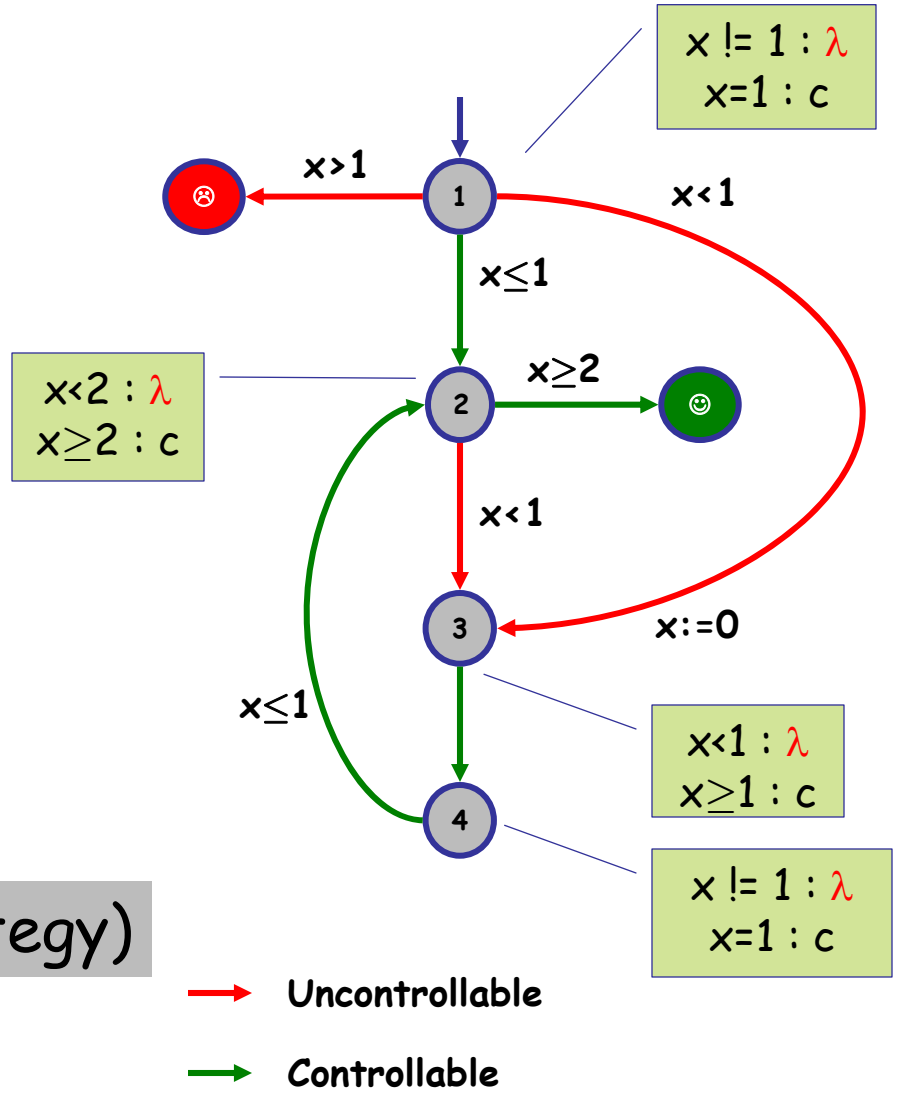
$States(\rho) \cap G \neq \emptyset$

$States(\rho) \cap G = \emptyset$

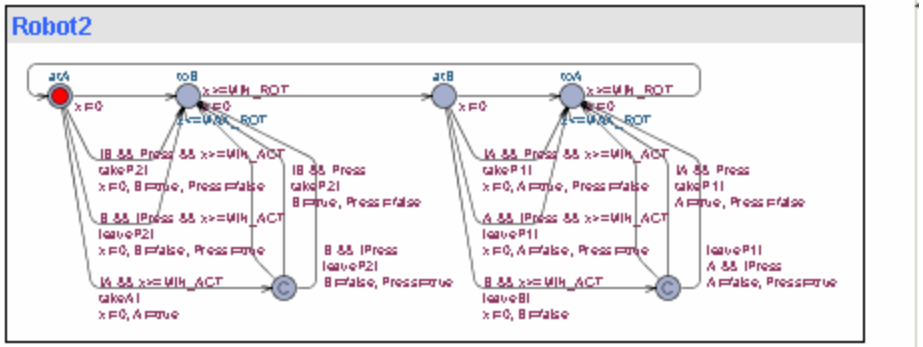
Winning Strategy:

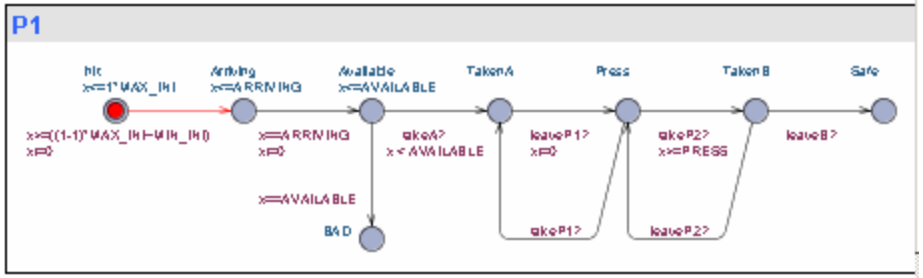
$Runs(F) \subseteq WinRuns$

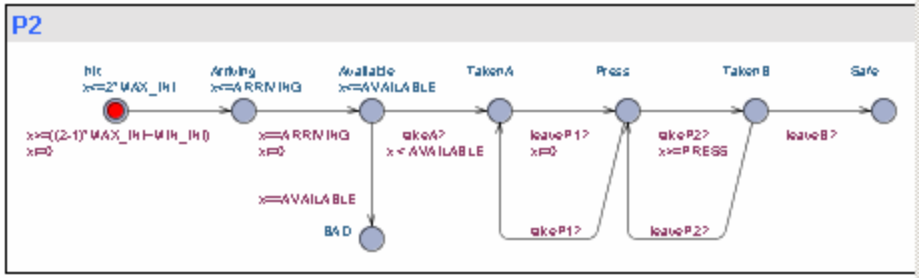
Winning (memoryless) strategy)

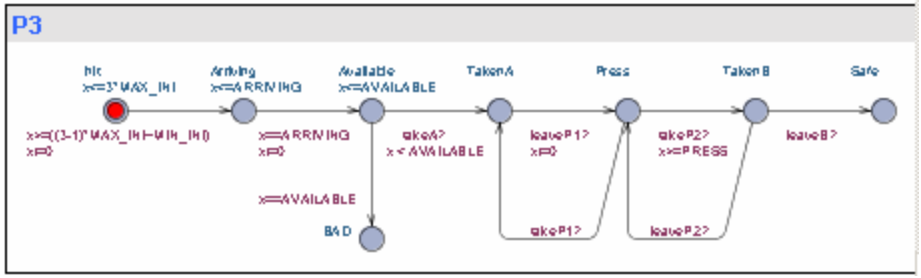


# Implementation & Demo

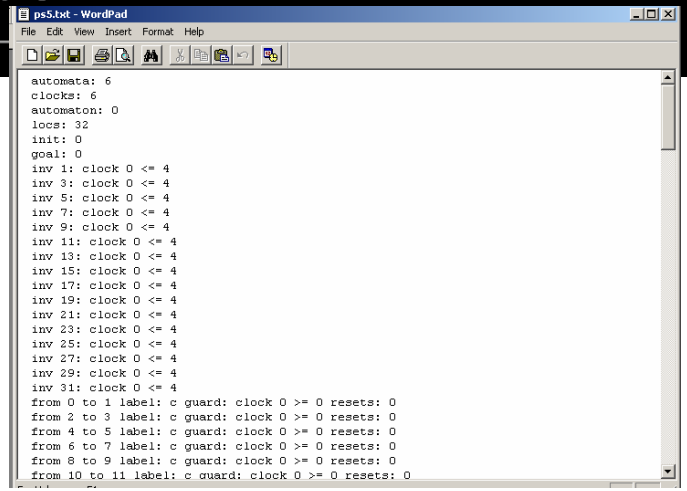



```
C:\UPPAAL Tiga>timed_games.exe ps5.txt
Parsing done !
--------------------------------
        Initial state
Locations: 0 0 0 0 0 0
<=0     <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
--------------------------------
posts: 76249
pres: 88678

--------------------------------
        Winning zones
1 DBM 7x7 {
<=0     <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
<=12    <=0     <=0     <=0     <=0     <=0     <=0
}
--------------------------------
        Losing zones
0 DBM 7x7 {
}
--------------------------------
```

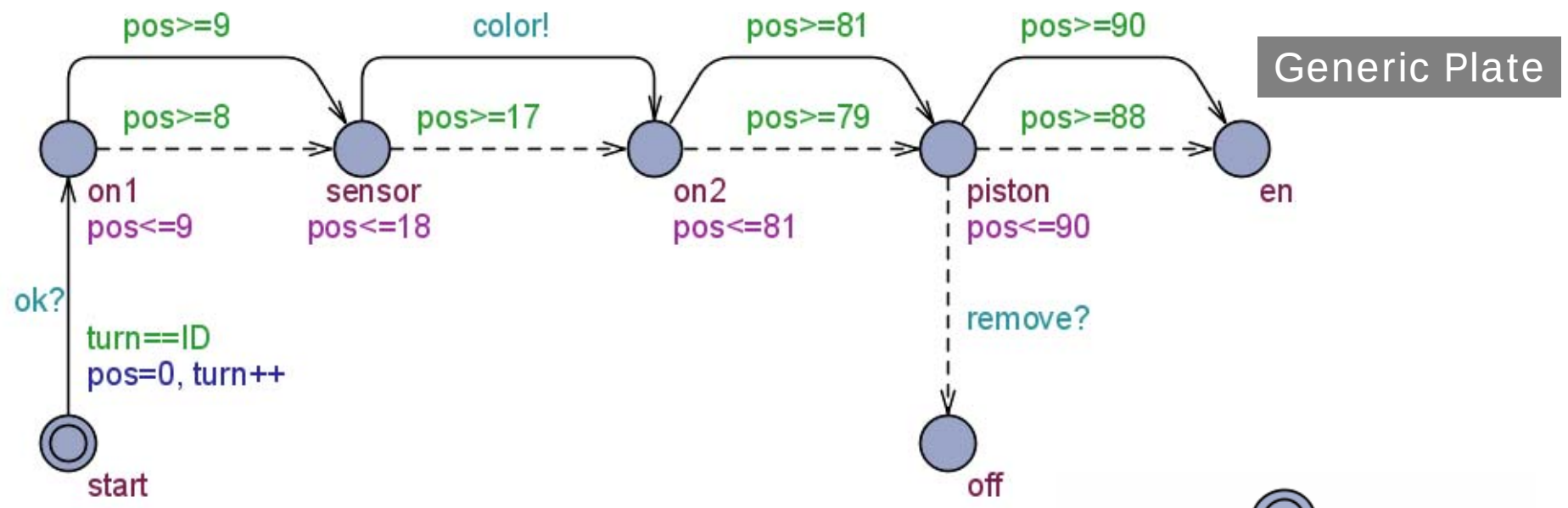


# Experimental Results

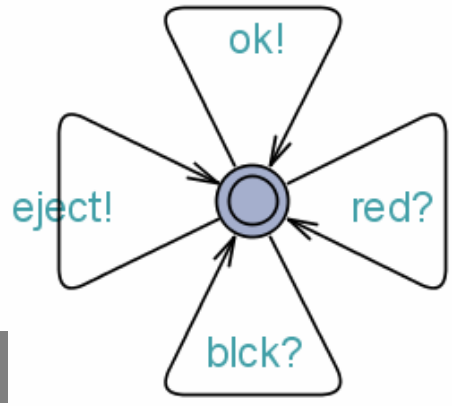
| Plates | | Basic | | Basic +inc | | Basic +inc +pruning | | Basic+lose +inc +pruning | | Basic+lose +inc +topt | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | time | mem | time | mem | time | mem | time | mem | time | mem |
| 2 | win | 0.0s | 1M | 0.0s | 1M | 0.0s | 1M | 0.0s | 1M | 0.04s | 1M |
| | lose | 0.0s | 1M | 0.0s | 1M | 0.0s | 1M | 0.0s | 1M | n/a | n/a |
| 3 | win | 0.5s | 19M | 0.0s | 1M | 0.0s | 1M | 0.1s | 1M | 0.27s | 4M |
| | lose | 1.1s | 45M | 0.1s | 1M | 0.0s | 1M | 0.2s | 3M | n/a | n/a |
| 4 | win | 33.9s | 1395M | 0.2s | 8M | 0.1s | 6M | 0.4s | 5M | 1.88s | 13M |
| | lose | - | - | 0.5s | 11M | 0.4s | 10M | 0.9s | 9M | n/a | n/a |
| 5 | win | - | - | 3.0s | 31M | 1.5s | 22M | 2.0s | 16M | 13.35s | 59M |
| | lose | - | - | 11.1s | 61M | 5.9s | 46M | 7.0s | 41M | n/a | n/a |
| 6 | win | - | - | 89.1s | 179M | 38.9s | 121M | 12.0s | 63M | 220.3s | 369M |
| | lose | - | - | 699s | 480M | 317s | 346M | 135.1s | 273M | n/a | n/a |
| 7 | win | - | - | 3256s | 1183M | 1181s | 786M | 124s | 319M | 6188s | 2457M |
| | lose | - | - | - | - | 16791s | 2981M | 4075s | 2090M | n/a | n/a |

# BRICK Sorting

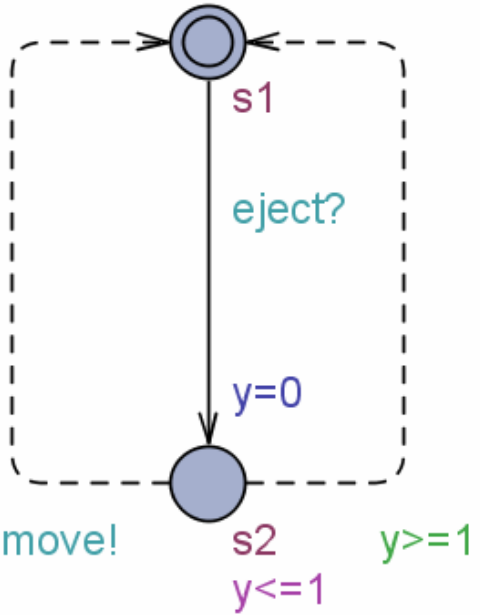
## Timing Uncertainty + Synthesis




Generic Plate



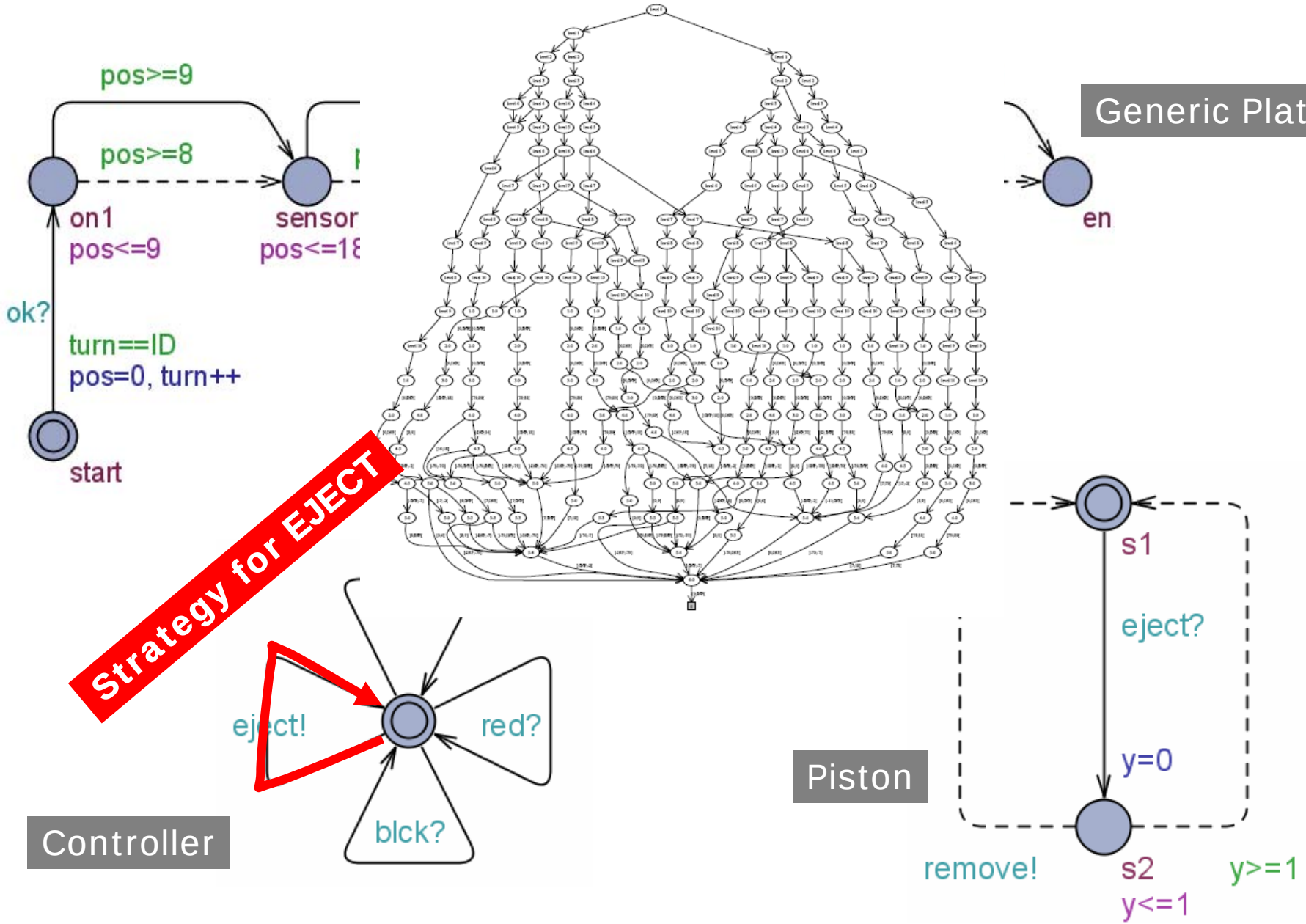

Controller




Piston

# BRICK Sorting

## Timing Uncertainty + Synthesis



# UPPAAL

*Home*



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.

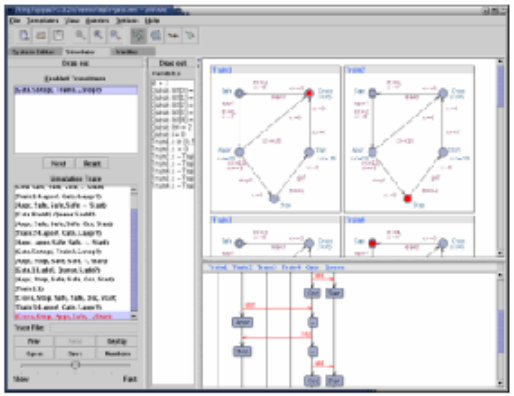


Figure 1: UPPAAL on screen.

## Download

**News:** We are proud to officially release UPPAAL 4.0.2 (Aug 7, 2006), available from the Download page. The 4.0 release is the result of over 2.5 years of 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).

## 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.

UPPSALA UNIVERSITET

AALBORG UNIVERSITY