

Year 2 Review
Paris, November 8th and 9th, 2006

Real Time Validation Tools

Kim G. Larsen
CISS, Aalborg University

UPPAAL Branches

- Real Time Verification

CLASSIC

- Real Time Performance Evaluation & Optimal Scheduling and Planning

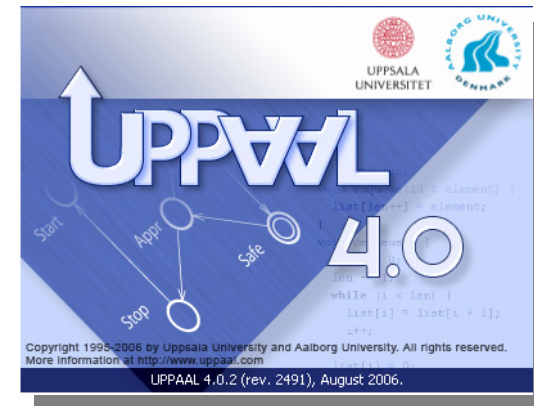
CORA

- Real Time (Optimal) Controller Synthesis

TIGA

- Real Time (On-Line) Testing

TRON

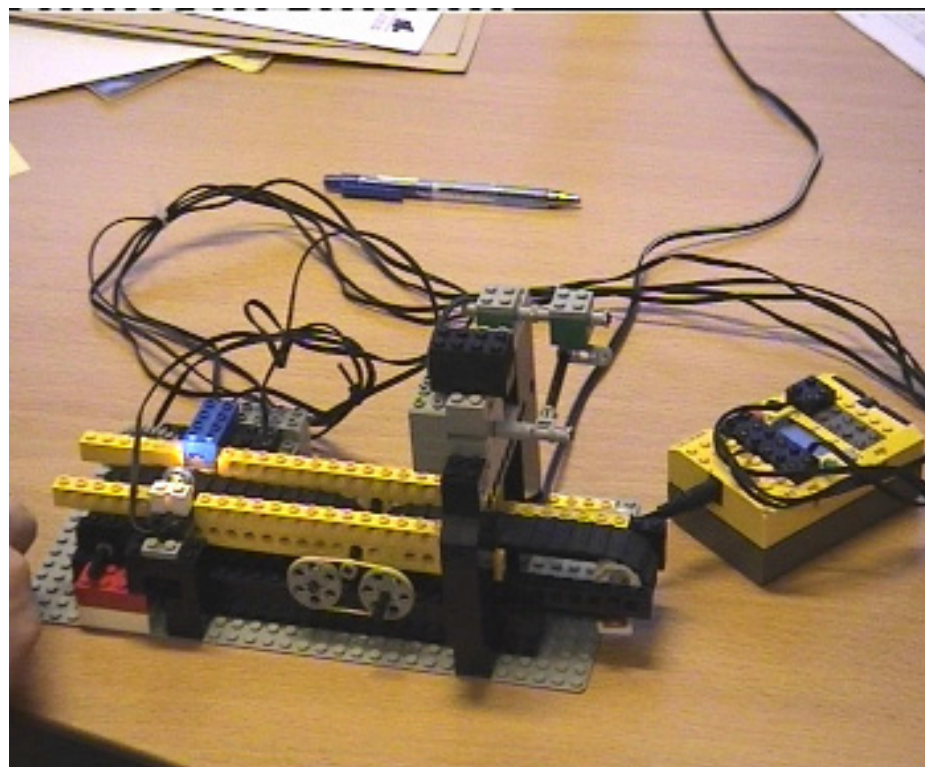


- Timed Automata
- Structured Datatypes
- C-code
- Synchronous/Broadcast Communication
- TCTL

A Real **Real** Timed System

The Plant

Conveyor Belt
&
Bricks



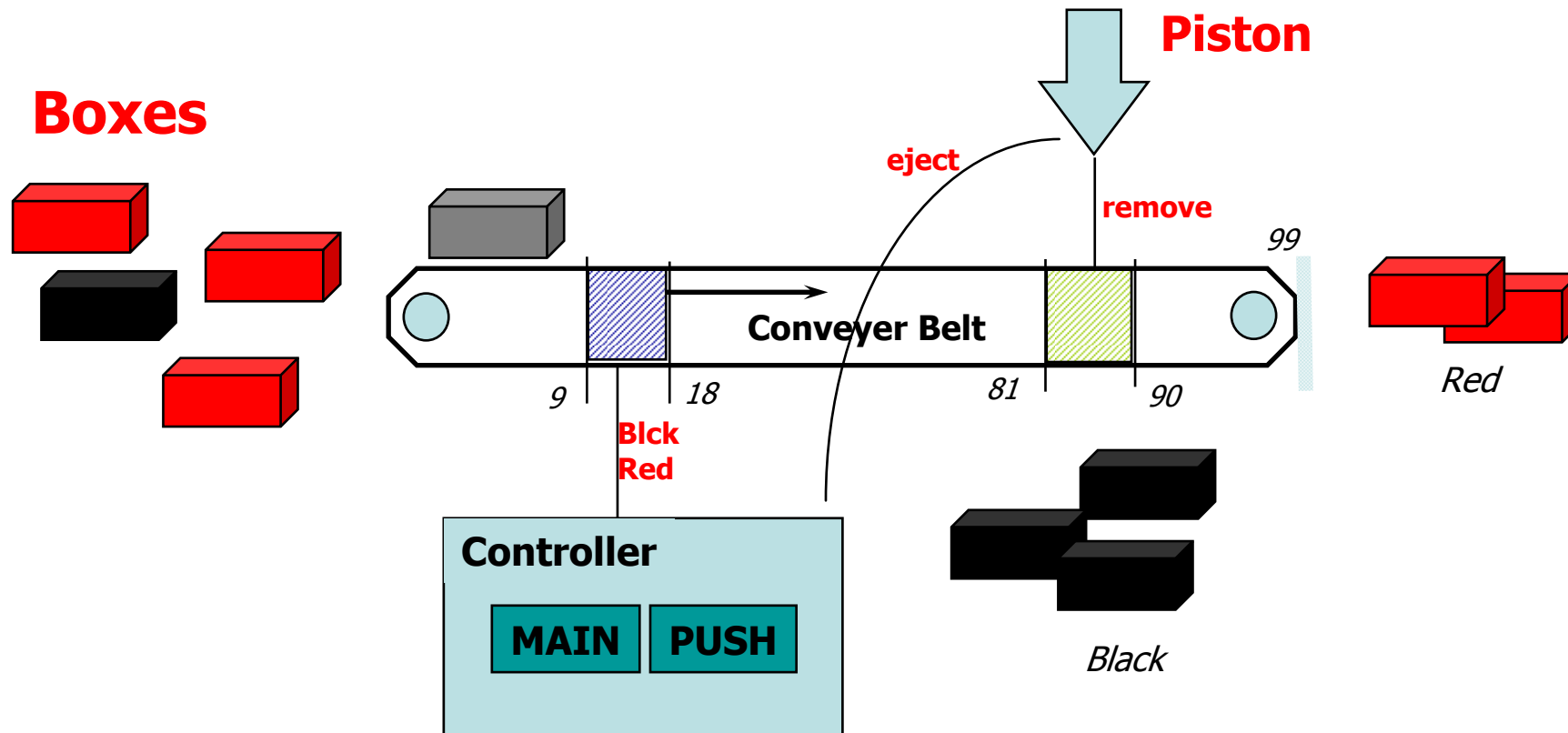
Controller Program

LEGO MINDSTORM

First UPPAAL model

Ken Tindell

Sorting of Lego Boxes

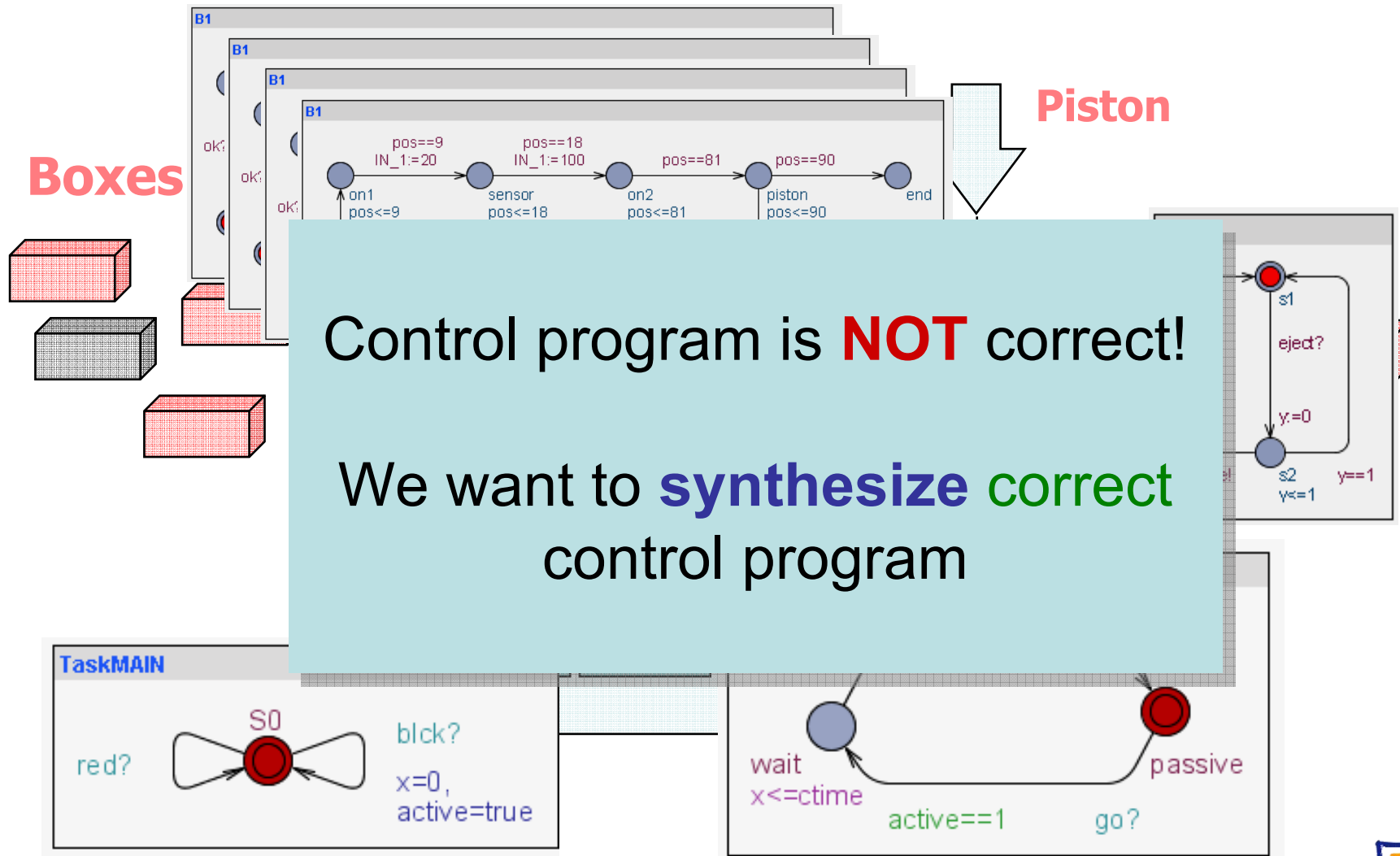


Exercise: Design **Controller** so that **black** boxes are being pushed out

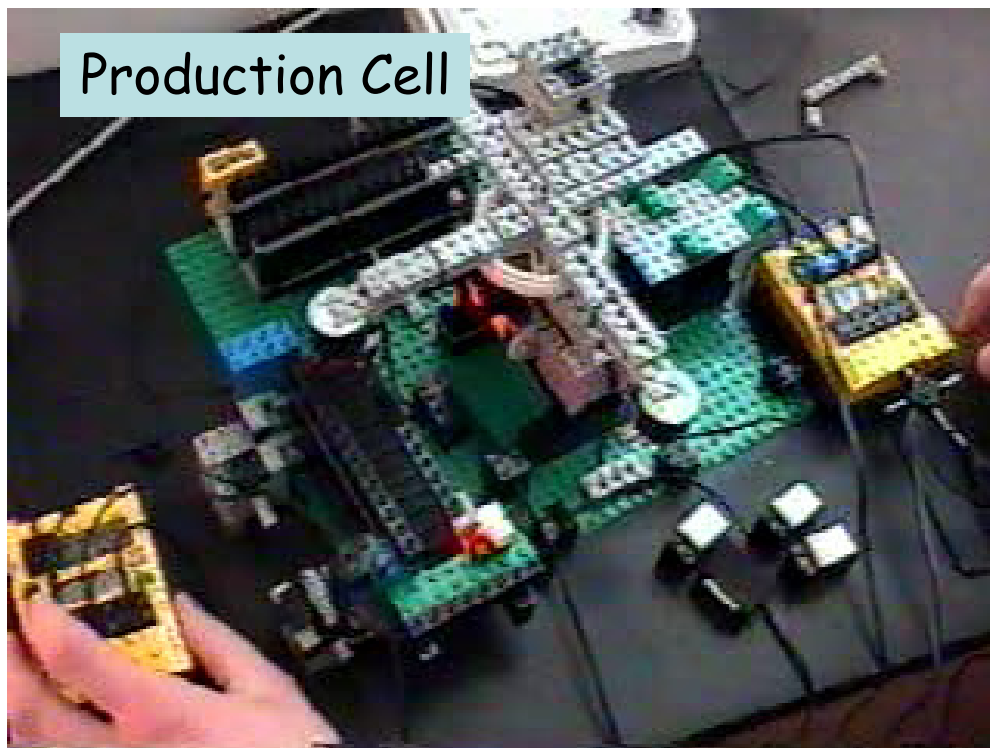
First UPPAAL model

Ken Tindell

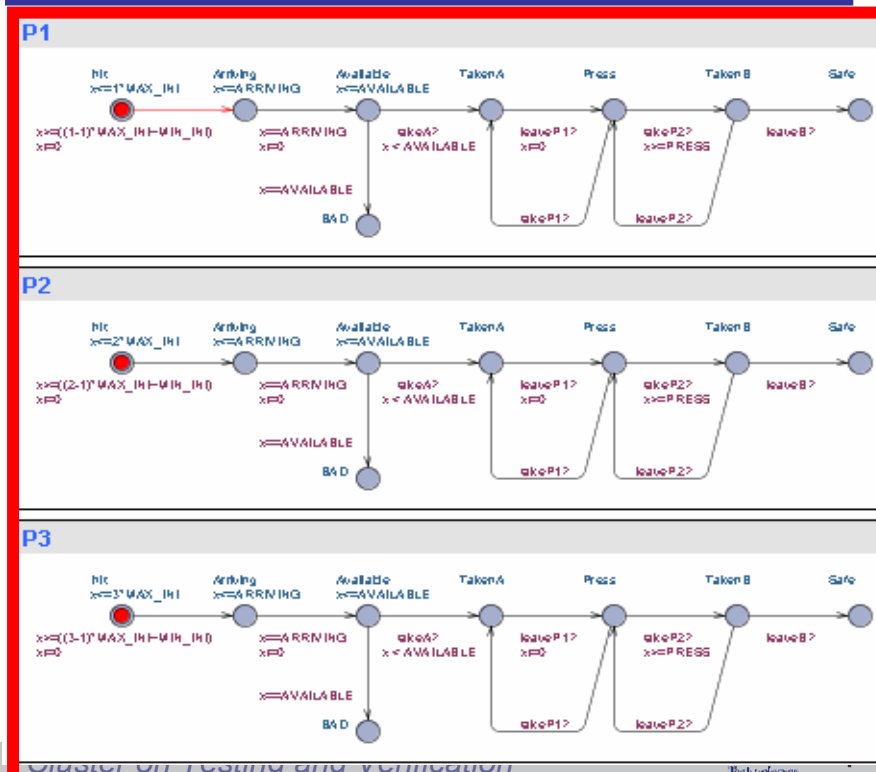
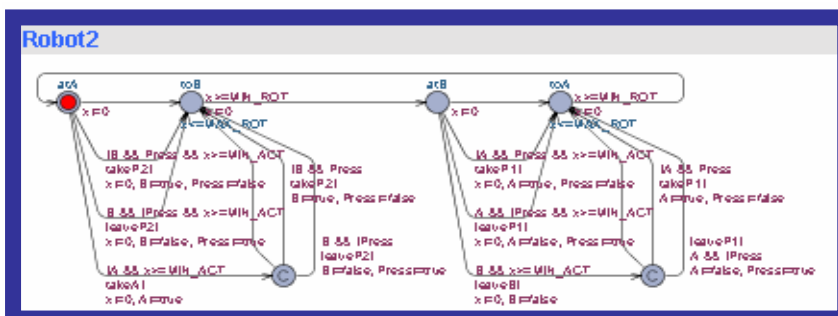
Sorting of Lego Boxes



Controller Synthesis and Timed Games



Production Cell



GIVEN System moves S ,
 Controller moves C , and property ϕ
 FIND strategy s_c such that $s_c \parallel S^2 \phi$



A Two-Player Game

Timed Games

Reachability / Safety Games

Memoryless strategy:

$$F : Q \rightarrow E_c[\lambda]$$

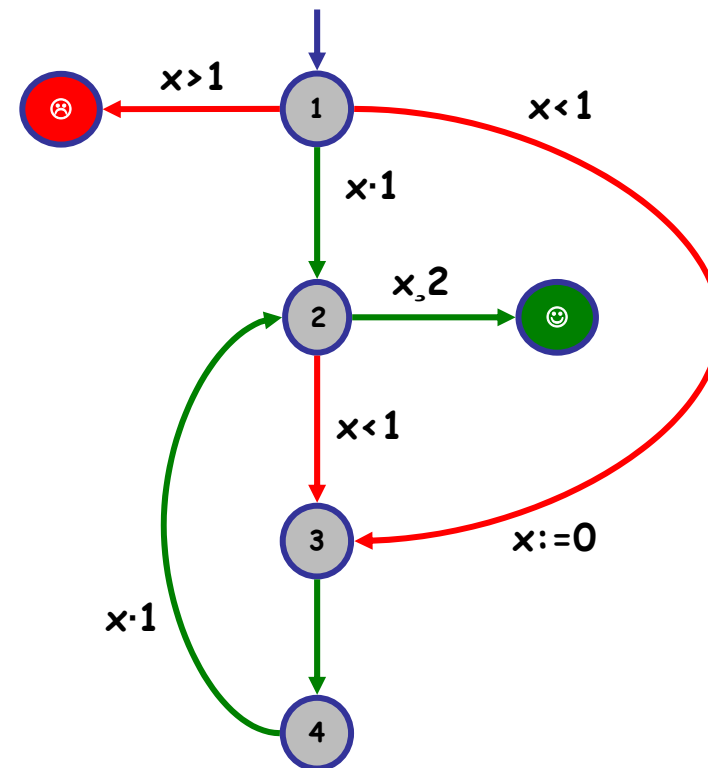
Winning Run:

$$\text{States}(\rho) \mathbf{A} G \neq \emptyset$$

$$\text{States}(\rho) \mathbf{A} G = \emptyset$$

Winning Strategy:

$$\text{Runs}(F) \mu \text{ WinRuns}$$



→ Uncontrollable

→ Controllable

Timed Games

Reachability / Safety Games

Memoryless strategy:

$$F : Q \rightarrow E_c[\lambda]$$

Winning Run:

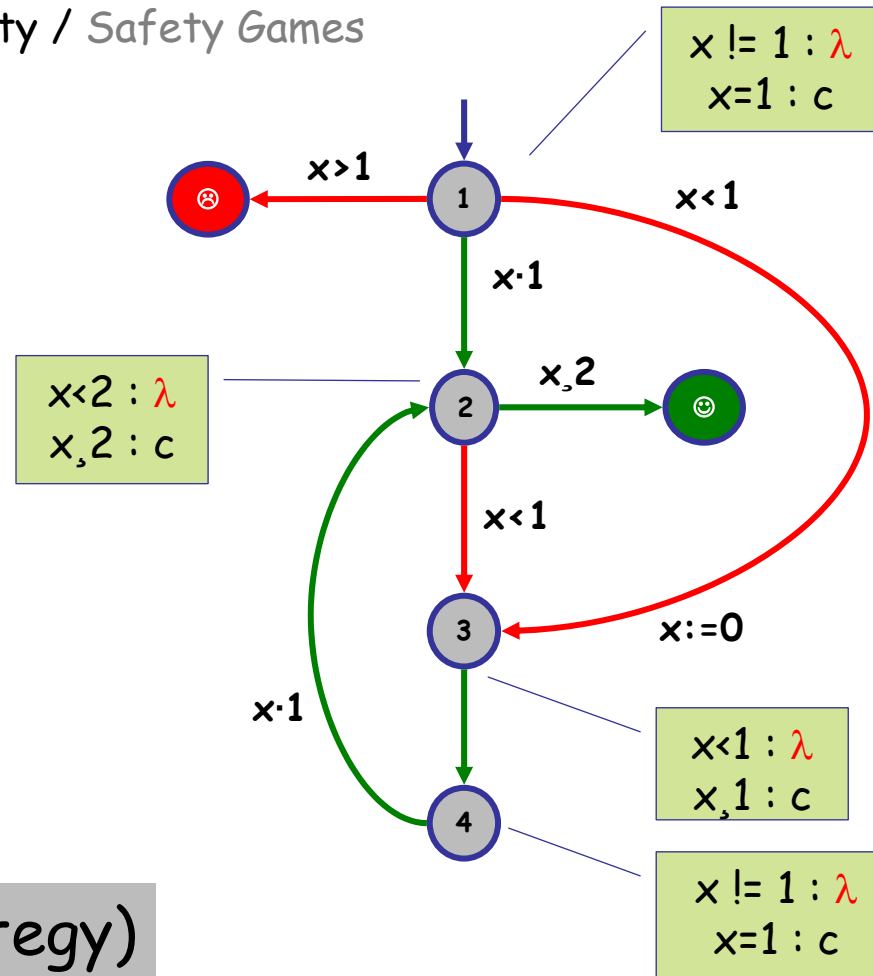
$$\text{States}(\rho) \mathbf{A} G \neq \emptyset$$

$$\text{States}(\rho) \mathbf{A} G = \emptyset$$

Winning Strategy:

$$\text{Runs}(F) \mu \text{ WinRuns}$$

Winning (memoryless) strategy



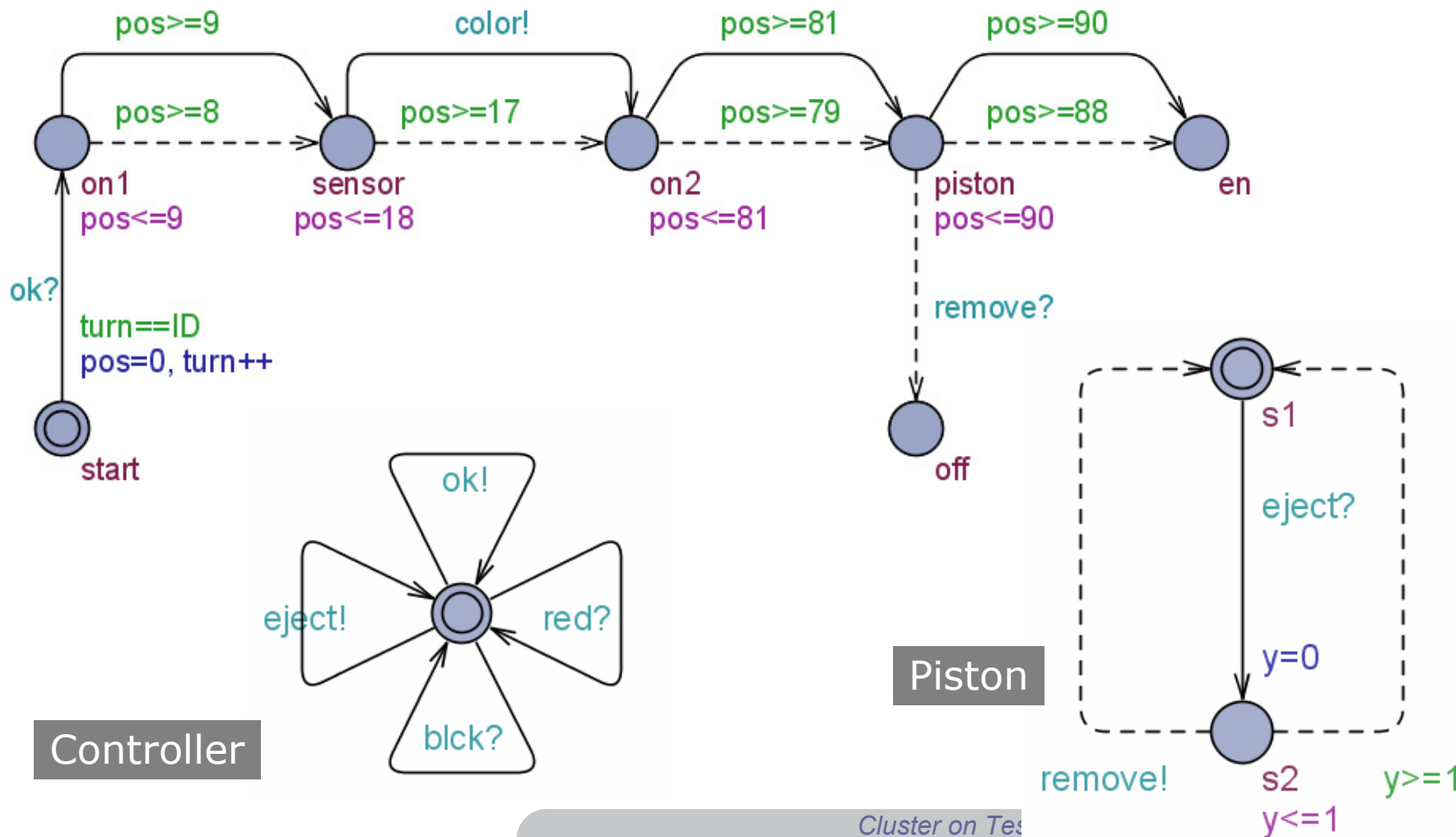
→ Uncontrollable

→ Controllable

BRICK Sorting

Timing Uncertainty + Synthesis

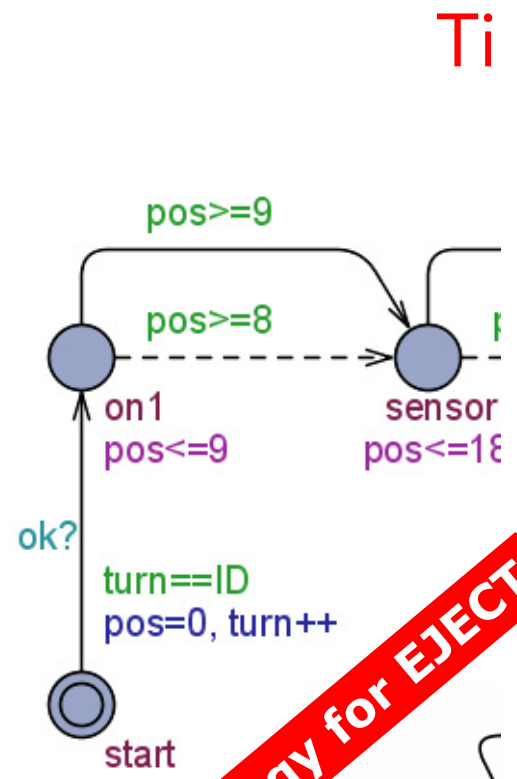
Generic Plate



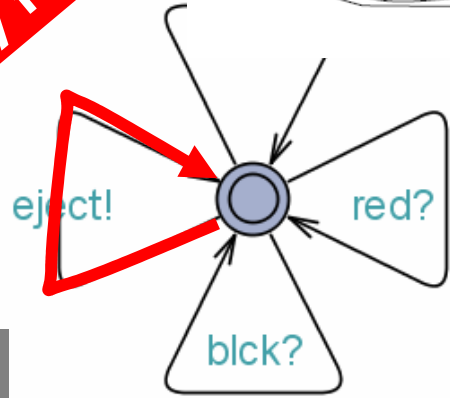
Controller

Piston

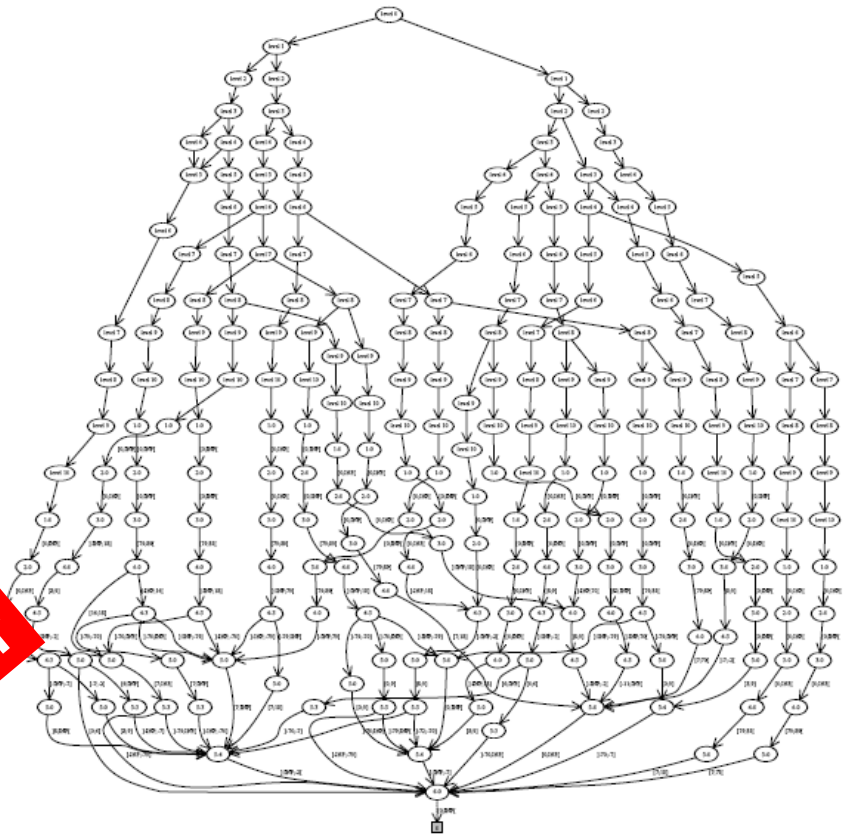
BRICK Controller



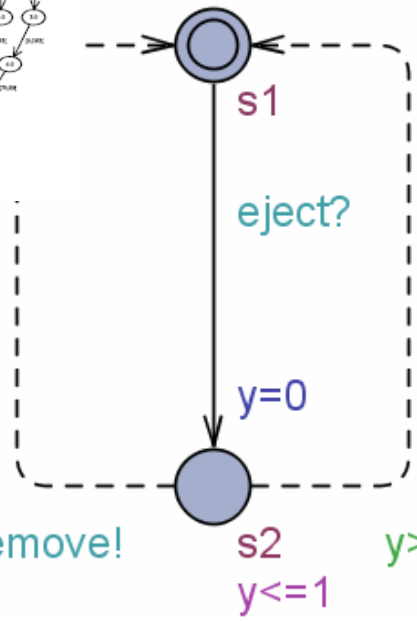
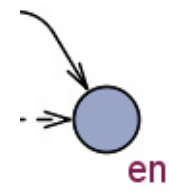
Strategy for EJECT



Piston



Generic Plate



Controller

Climate Control For Live Stock Production

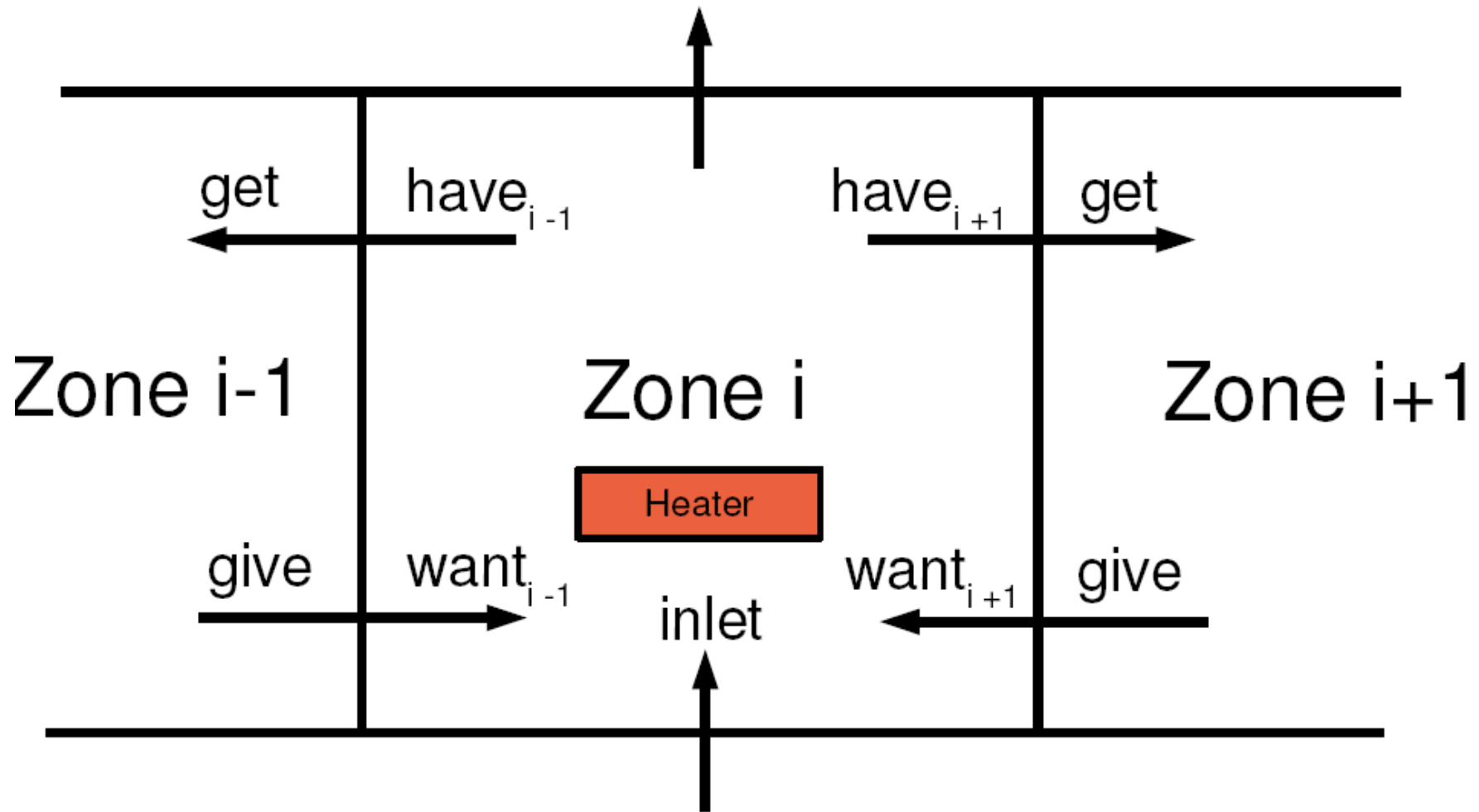


Syvsten,
Northern Jutland,
DK

By Jan J. Jessen
Jacob I. Rasmussen

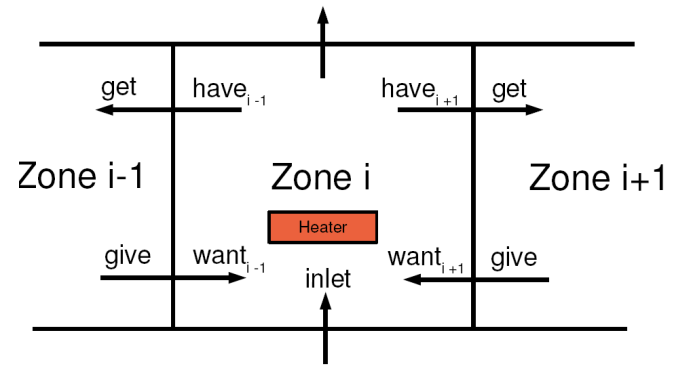


Climate Control For Live Stock Production



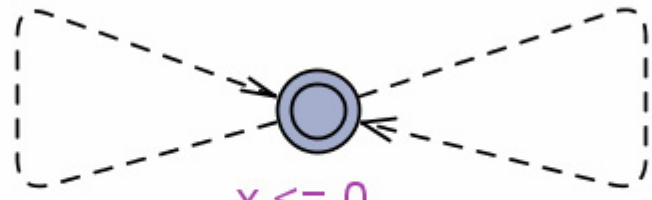
Climate Control For Live Stock Production

Neighboring zone



Neighbor wants to receive flow?

temp[id]? temp[id] = false : temp[id] = true,
check_hotness_integrity()



state_changed!

$x \leq 0$

c : choice_t
state_changed!
n[id] = c

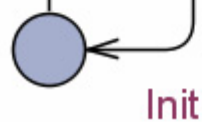
Temperature in neighbor zone (lower/higher)

Climate

Zone C

state_ch

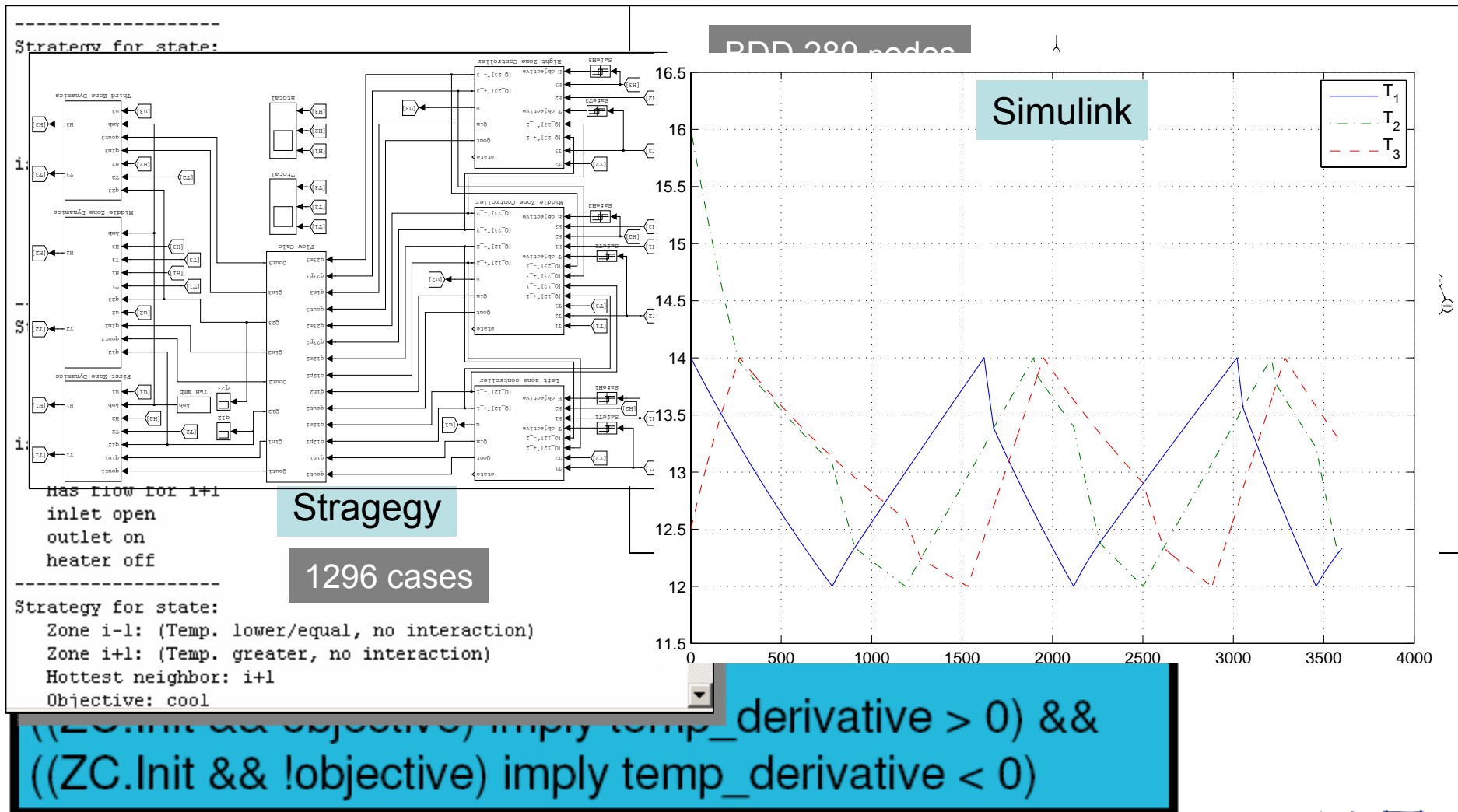
```
int compute_temperature(const choice_t c0, const choice_t c1, const intbool_t in, const intbool_t out,
{
int o,i,amp;
//active out-flow
o = out + (c0 == HAVE && n[0] == WANT) + (c1 == HAVE && n[1] == WANT);
//active in-flow
i = in + (c0 == WANT && n[0] == HAVE) + (c1 == WANT && n[1] == HAVE);
i = i >? 1;
//Multiplier per incoming flow
amp = (o * PER_OUT_CONTRIBUTION) / i;
if (objective) //heating
{
return heat
+ amp*(c0 == WANT && n[0] == HAVE ? (temp[0] ? (!hottest ? 3 : 1) : -1) : 0)
+ amp*(c1 == WANT && n[1] == HAVE ? (temp[1] ? ( hottest ? 3 : 1) : -1) : 0)
+ amp*(in ? -3 : 0)
+ -(c0 == HAVE) //Motivation for participation, even when not neighbor doesn't want, og has air
+ -(c1 == HAVE); //Motivation for participation, even when not neighbor doesn't want, og has ai
}
else //cooling
{
return (heat ? -3 : 0)
+ amp*(in ? 5 : 0)
+ amp*(c0 == WANT && n[0] == HAVE ? (!temp[0] ? ( hottest ? 3 : 1) : -1) : 0)
+ amp*(c1 == WANT && n[1] == HAVE ? (!temp[1] ? (!hottest ? 3 : 1) : -1) : 0)
+ (c0 == HAVE) //Motivation for participation, even when not neighbor doesn't want, og has air
+ (c1 == HAVE); //Motivation for participation, even when not neighbor doesn't want, og has air
}
}
```



inlet = in,
outlet = out,
temp_derivative = compute_temperature(c0,c1,in,out,heat)

Init

Obtaining executable code



UPPAAL

Home

www.uppaal.com

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.

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](#)).

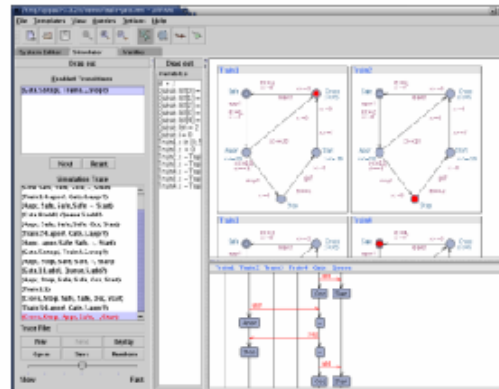


Figure 1: UPPAAL on screen.

License

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



UPPSALA
UNIVERSITET



AALBORG UNIVERSITY



Information Society
Rekordregion