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

Real Time Validation Tools

Kim G. Larsen

CISS, Aalborg University

UPPAAL Branches

CLASSIC

- Real Time Verification

CORA

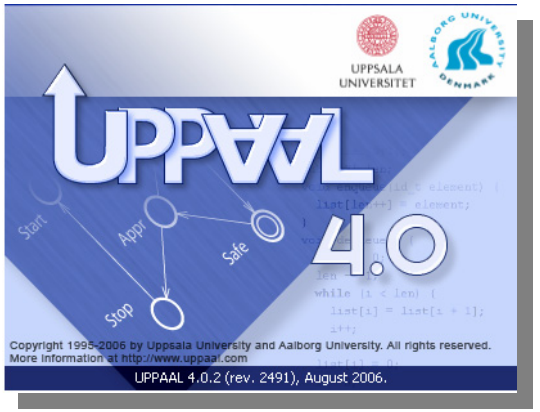
- Real Time Performance Evaluation & Optimal Scheduling and Planning

TIGA

- Real Time (Optimal) Controller Synthesis

TRON

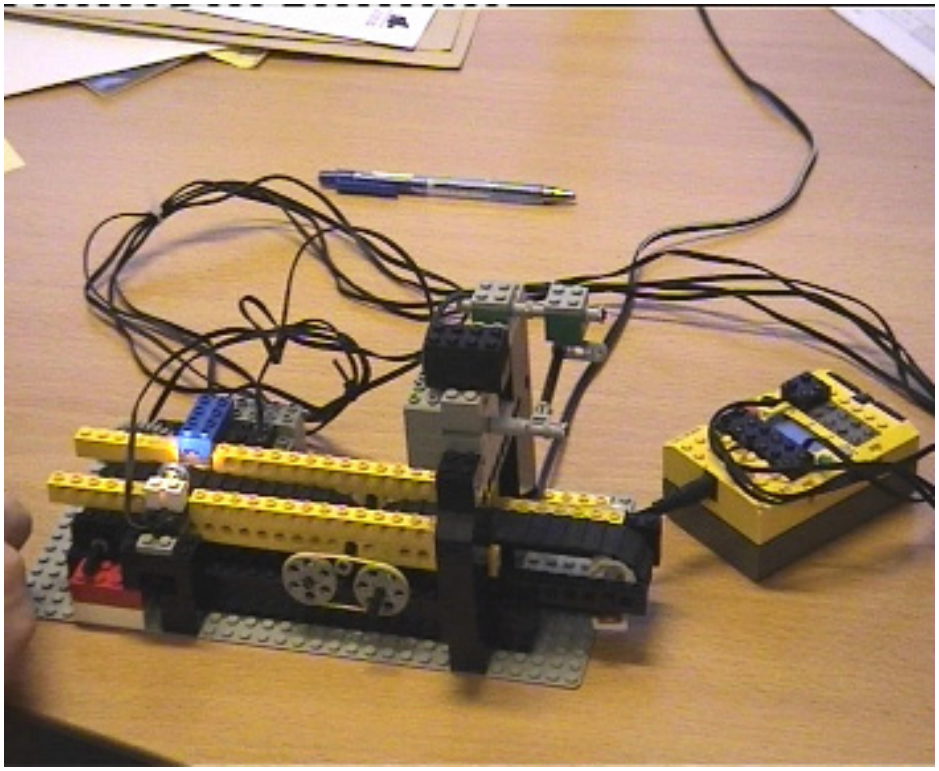
- Real Time (On-Line) Testing



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

A Real Real Timed System

The Plant
Conveyor Belt
&
Bricks

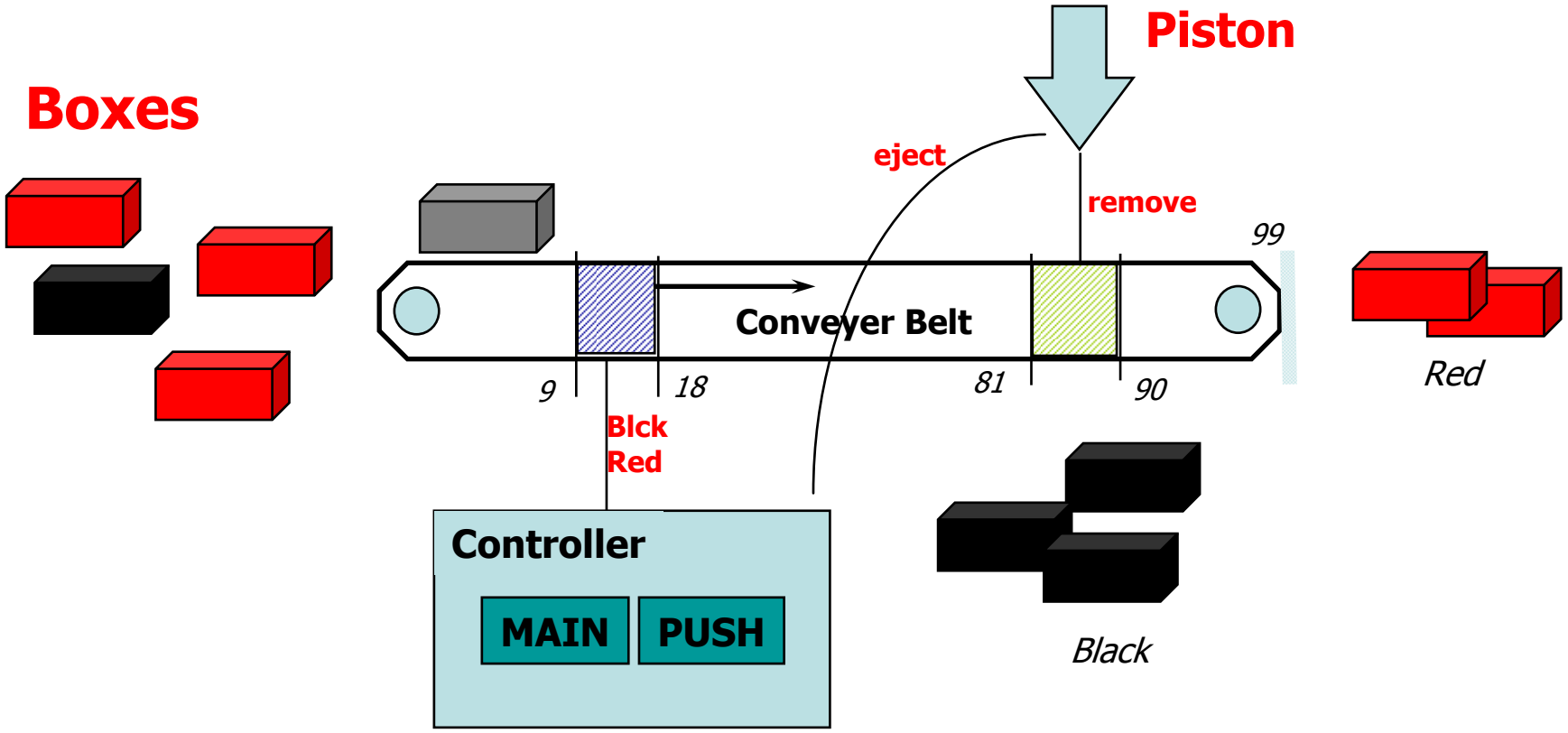


Controller Program
LEGO MINDSTORM

First UPPAAL model

Sorting of Lego Boxes

Ken Tindell

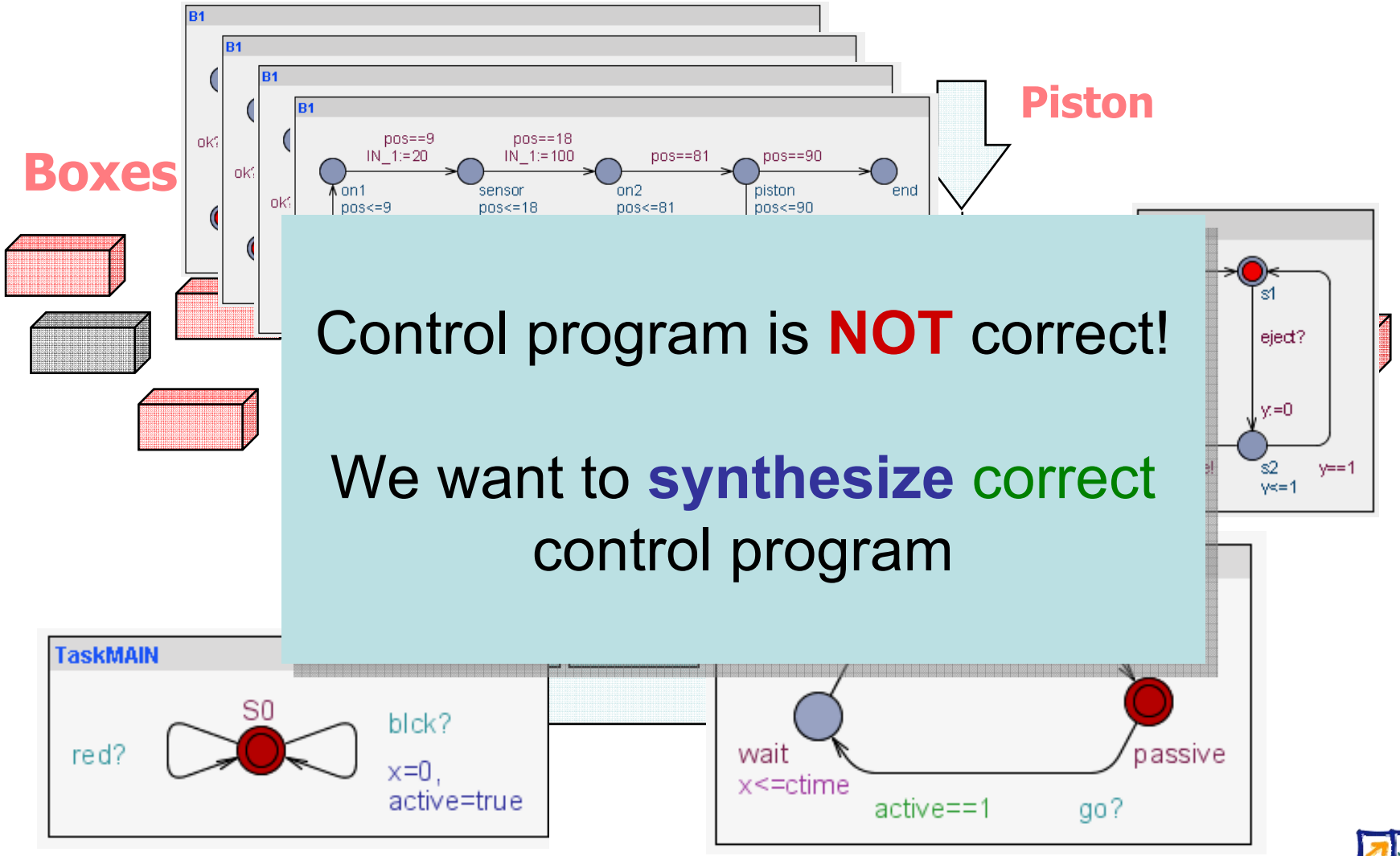


Exercise: Design **Controller** so that **black** boxes are being pushed out

First UPPAAL model

Sorting of Lego Boxes

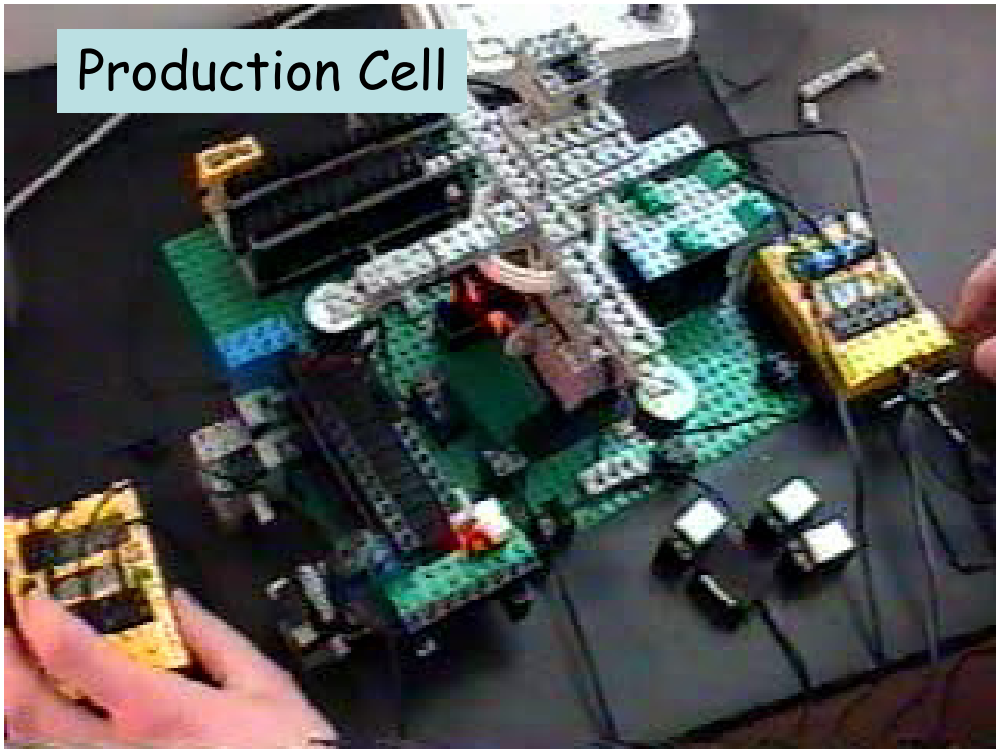
Ken Tindell

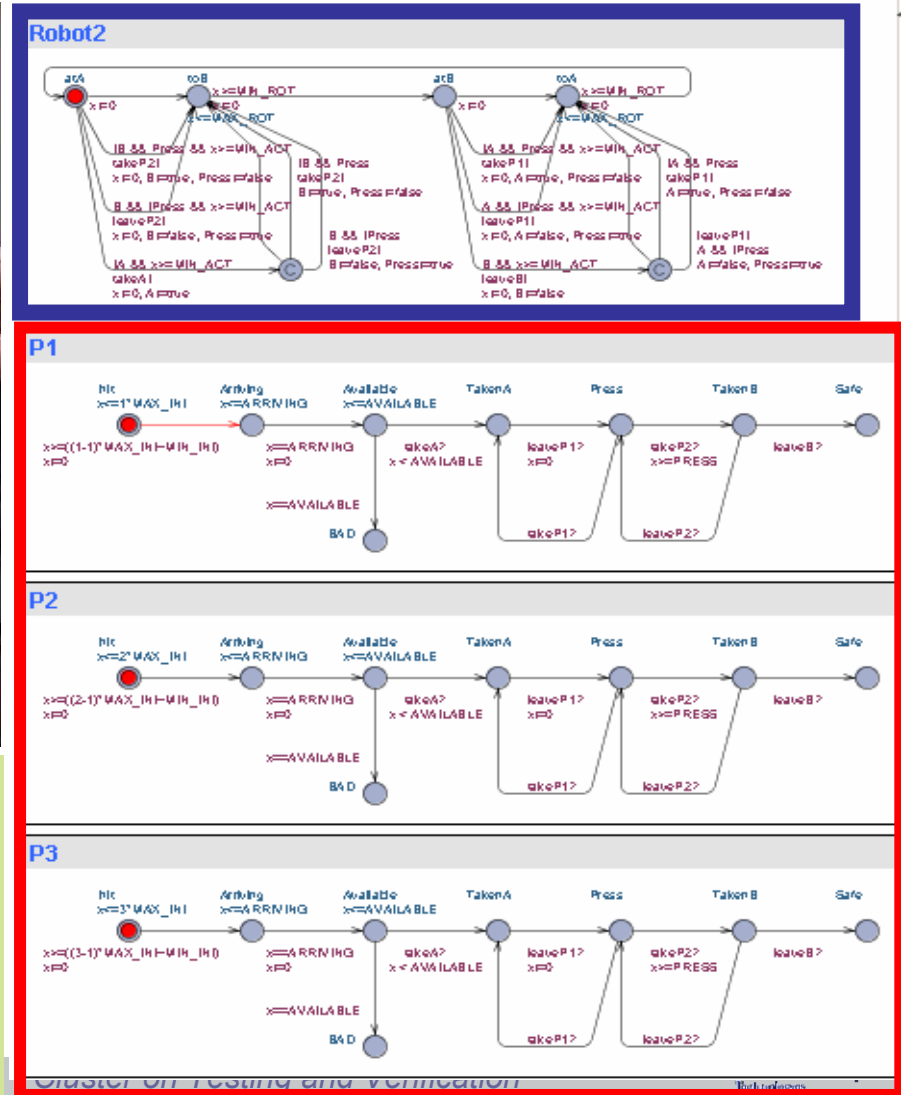


Control program is **NOT** correct!

We want to **synthesize** correct control program

Controller Synthesis and Timed Games





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

→

A Two-Player Game

Timed Games

Reachability / Safety Games

Memoryless strategy:

$F : Q \rightarrow E_c [\lambda$

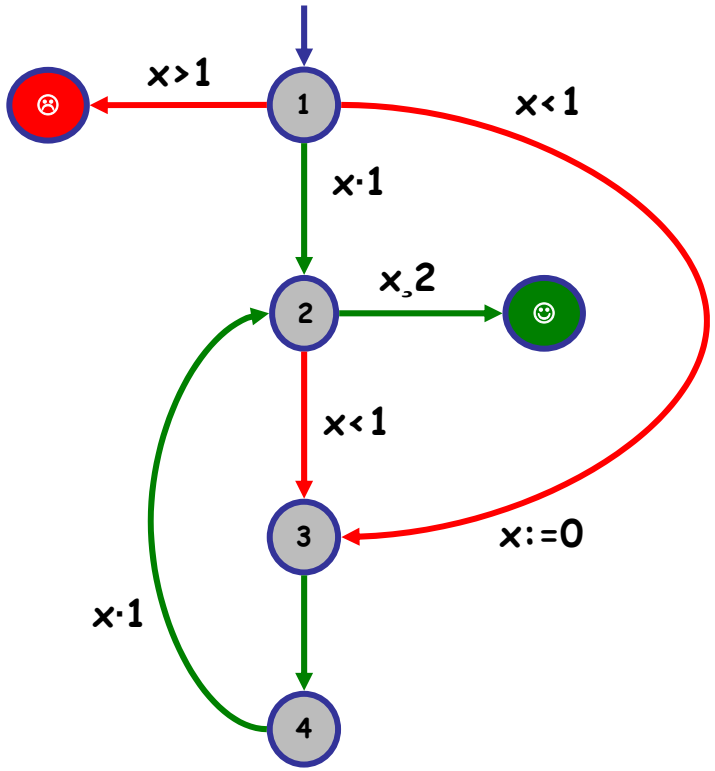
Winning Run:

States(ρ) Å G ≠ Ø

States(ρ) Å G = Ø

Winning Strategy:

Runs(F) µ WinRuns



Timed Games

Reachability / Safety Games

Memoryless strategy:

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

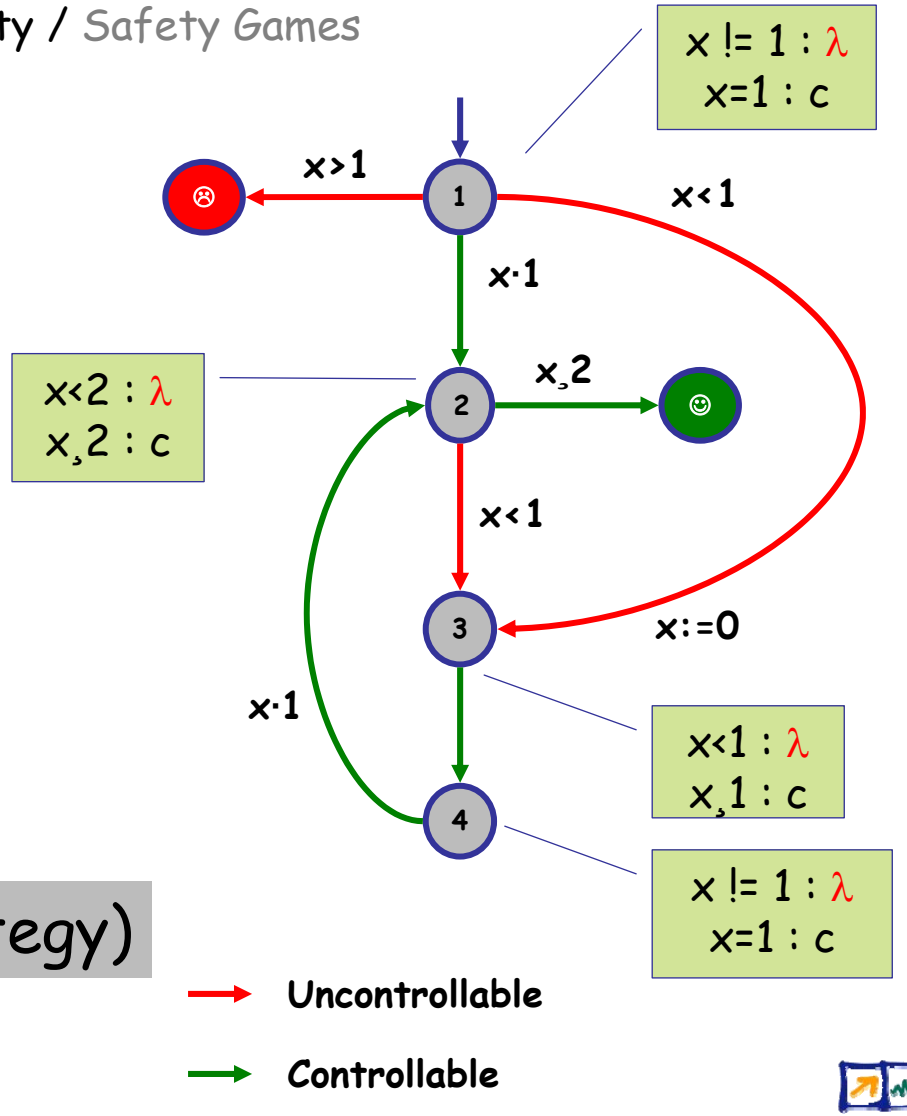
Winning Run:

States(ρ) Å G ≠ Ø

States(ρ) Å G = Ø

Winning Strategy:

Runs(F) µ WinRuns

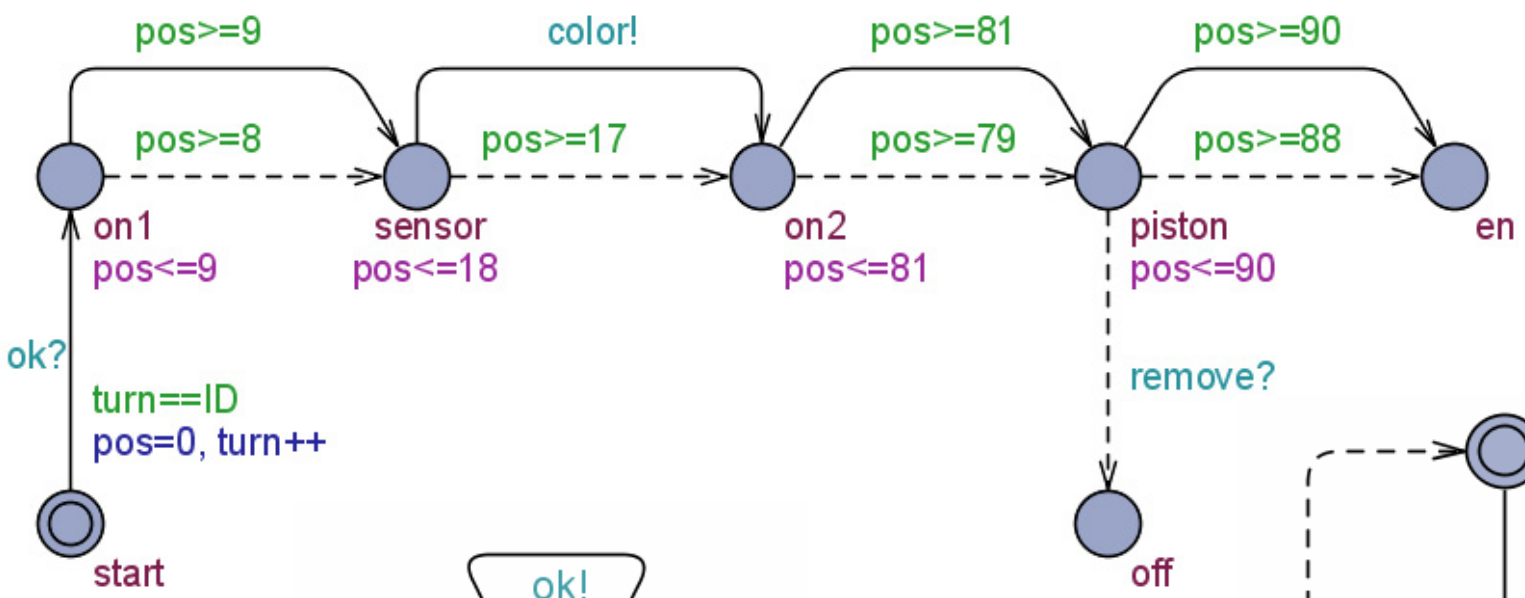


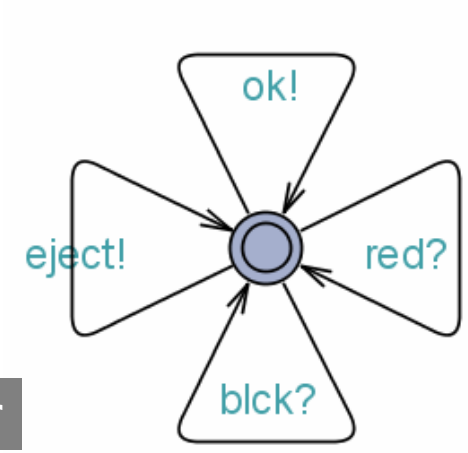
Winning (memoryless) strategy)

BRICK Sorting

Timing Uncertainty + Synthesis

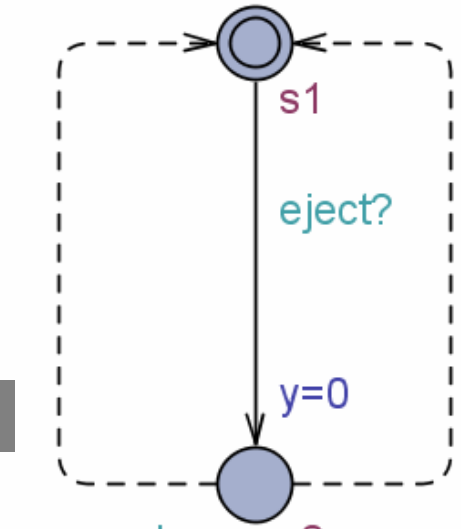
Generic Plate





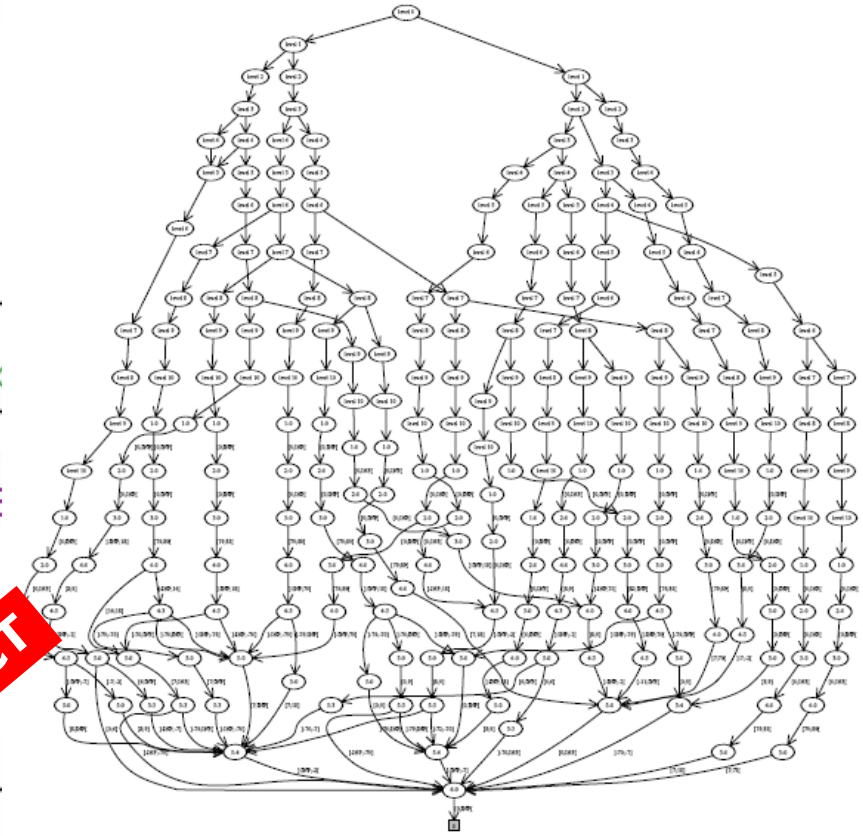
Controller

Piston



BRICK Sorting

Ti esis



Generic Plate

pos>=9

pos>=8

on1
pos<=9

sensor
pos<=18

en

ok?

turn==ID
pos=0, turn++

start

Strategy for EJECT

eject!

red?

blck?

Controller

Piston

s1

eject?

y=0

remove!

s2
y<=1

y>=1

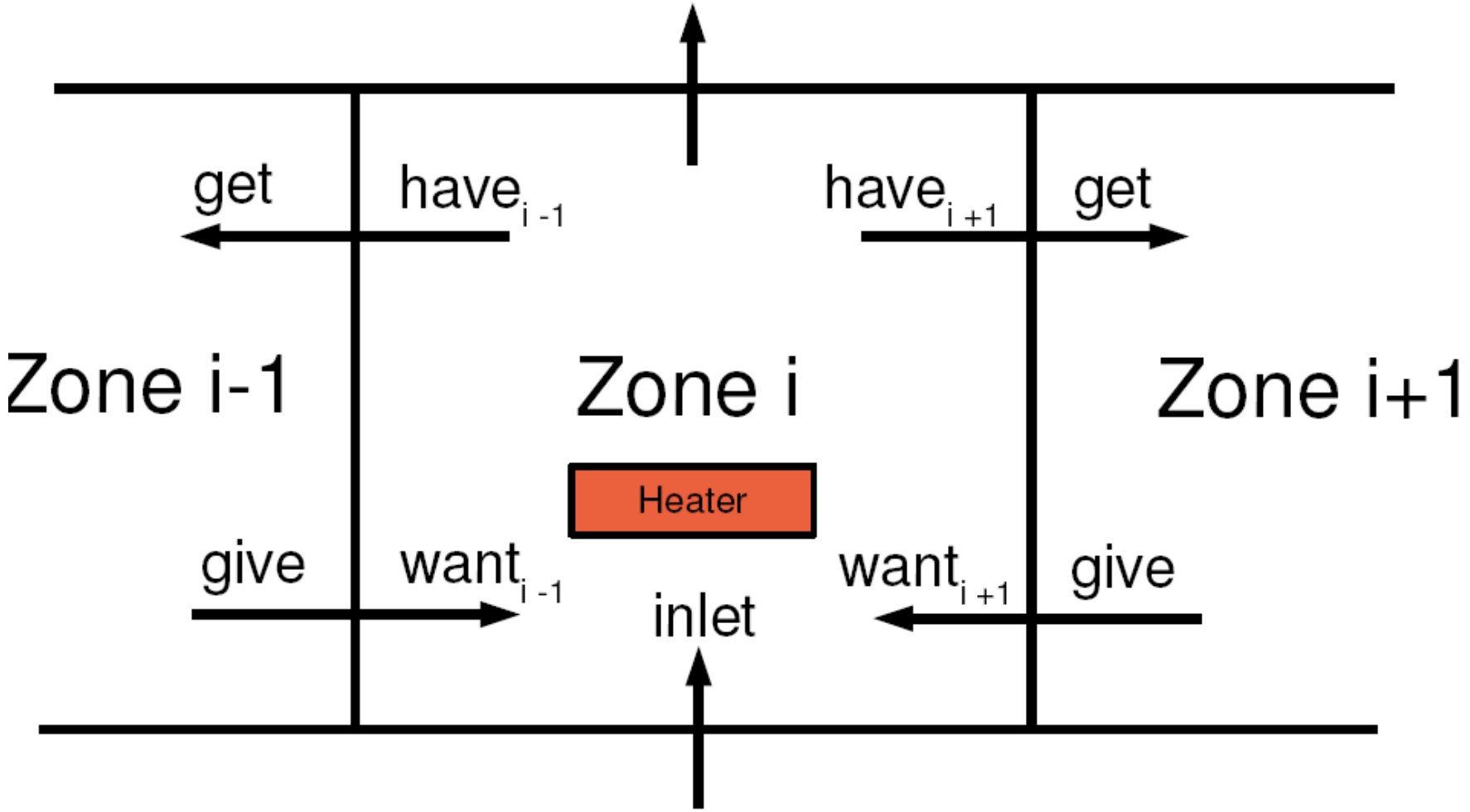
Society

Climate Control For Live Stock Production

Syvsten,
Northern Jutland,
DK

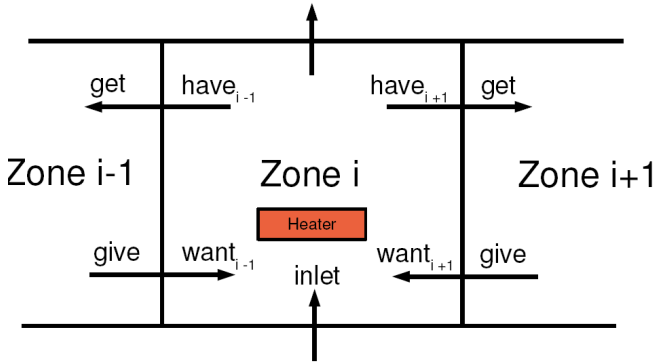
By Jan J. Jessen
Jacob I. Rasmussen

Climate Control For Live Stock Production



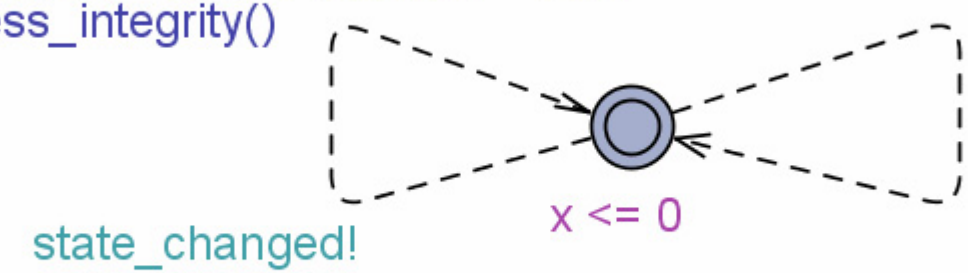
Climate Control For Live Stock Production

Neighboring zone



Neighbor wants to receive flow?

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



Temperature in neighbor zone (lower/higher)

Clin

Zone C

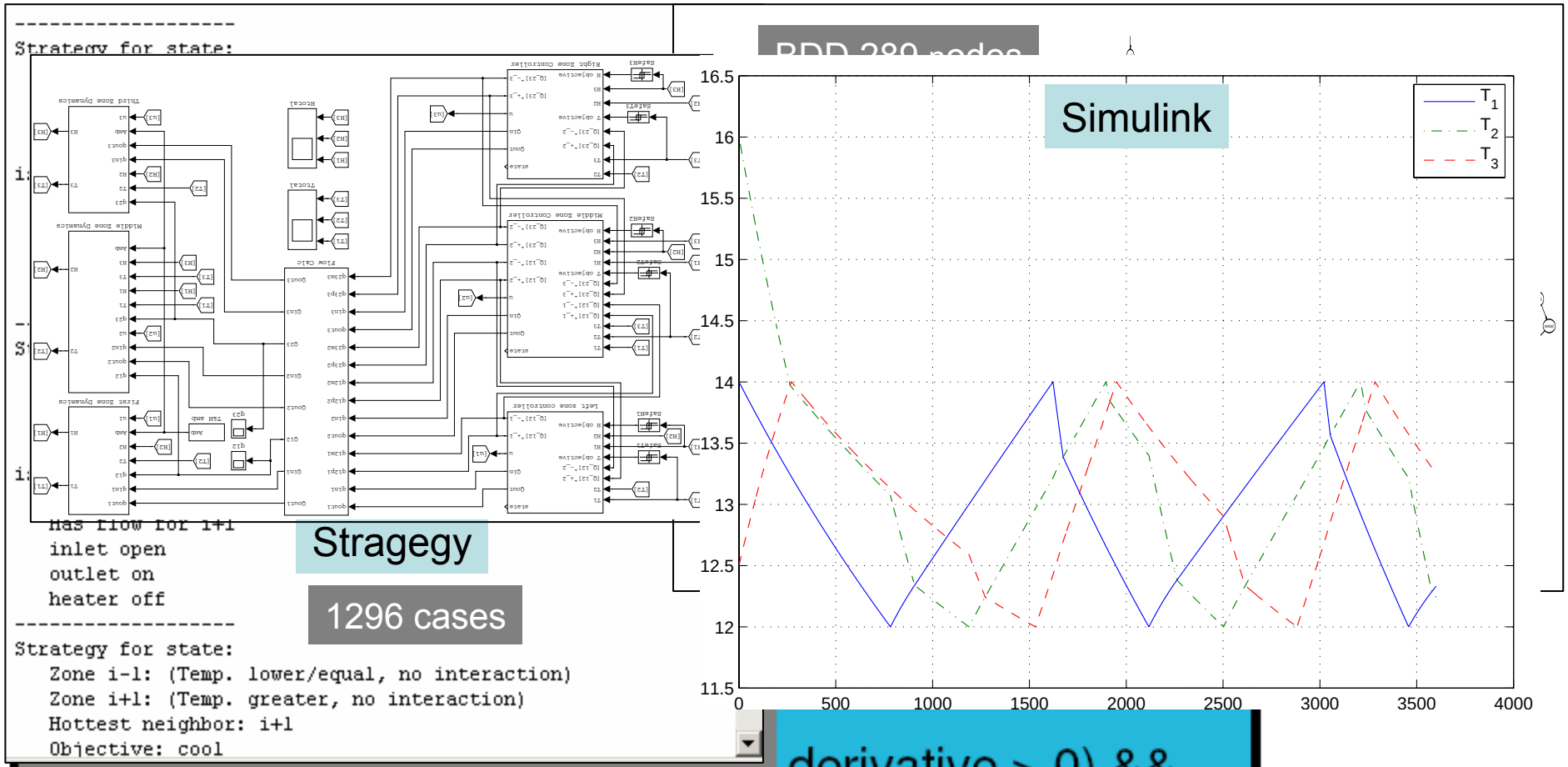
```
int compute_temperature(const choice_t c0, const choice_t c1, const intbool_t in, const intbool_t out,
{
int o,i,amp;
//active out-flow
o = out + (c0 == HAVE && n[0] == WANT) + (c1 == HAVE && n[1] == WANT);
//active in-flow
i = in  + (c0 == WANT && n[0] == HAVE) + (c1 == WANT && n[1] == HAVE);
i = 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
}
}
```

state_ch

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

Init

Obtaining executable code



RELATED SITES: TIMES | UPPAAL CORA | UPPAAL TRON | UPPAAL TIGA

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.

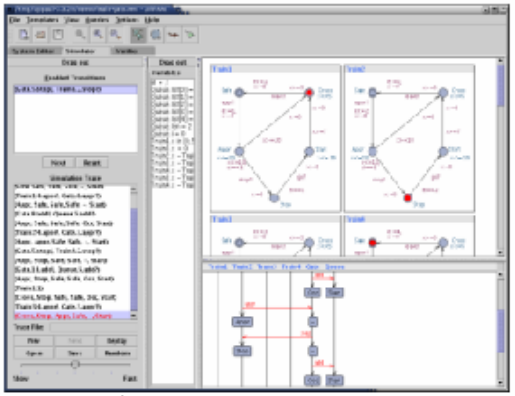


Figure 1: UPPAAL on screen.

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