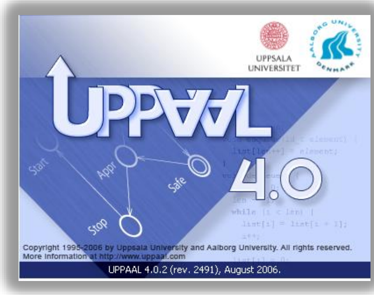
TAPAS

Tests and Proofs and Synthesis



Copyright 1995-2006 by Uppsala University and Aalborg University. All rights reserved.
More information at http://www.uppaal.com

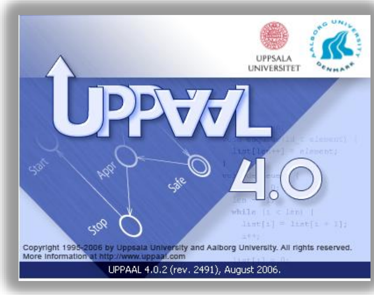
UPPAAL 4.0.2 (rev. 2491), August 2006.

Kim G Larsen

Aalborg University, DENMARK

TAPAS

From Testing and Verification to Performance Analysis and Synthesis of Cyber-Physical Systems



UPPSALA UNIVERSITET

UPPAAL 4.0

Copyright 1995-2006 by Uppsala University and Aalborg University. All rights reserved.
More information at http://www.uppaal.com

UPPAAL 4.0.2 (rev. 2491), August 2006.

Kim G Larsen
Aalborg University, DENMARK

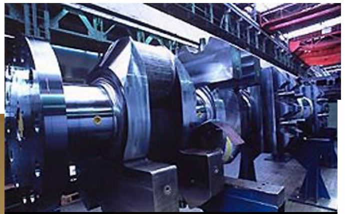
CISS –
Center For Embedded Software Systems

Regional ICT Center (2002-   )

- 3 research groups
  - Computer Science
  - Control Theory
  - Hardware
  - Wireless Communication

- 20 Employed
- 25 Associated
- 20 PhD Students
- 70 Industrial projects
- 10 Elite-students

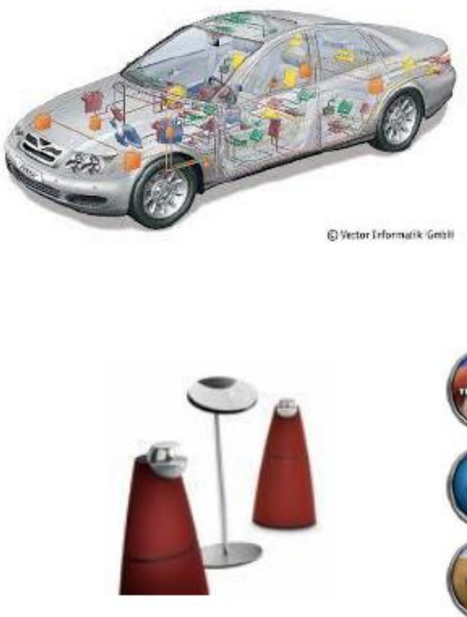
- ARTIST Design
- ARTEMIS / ECSEL
- … …

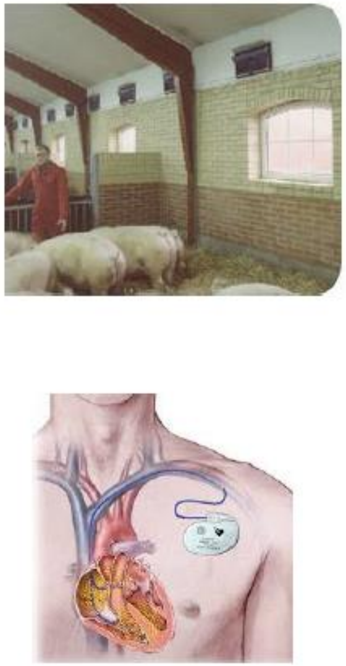


Information Society Technologies

ARTEMIS

From ES to CPS





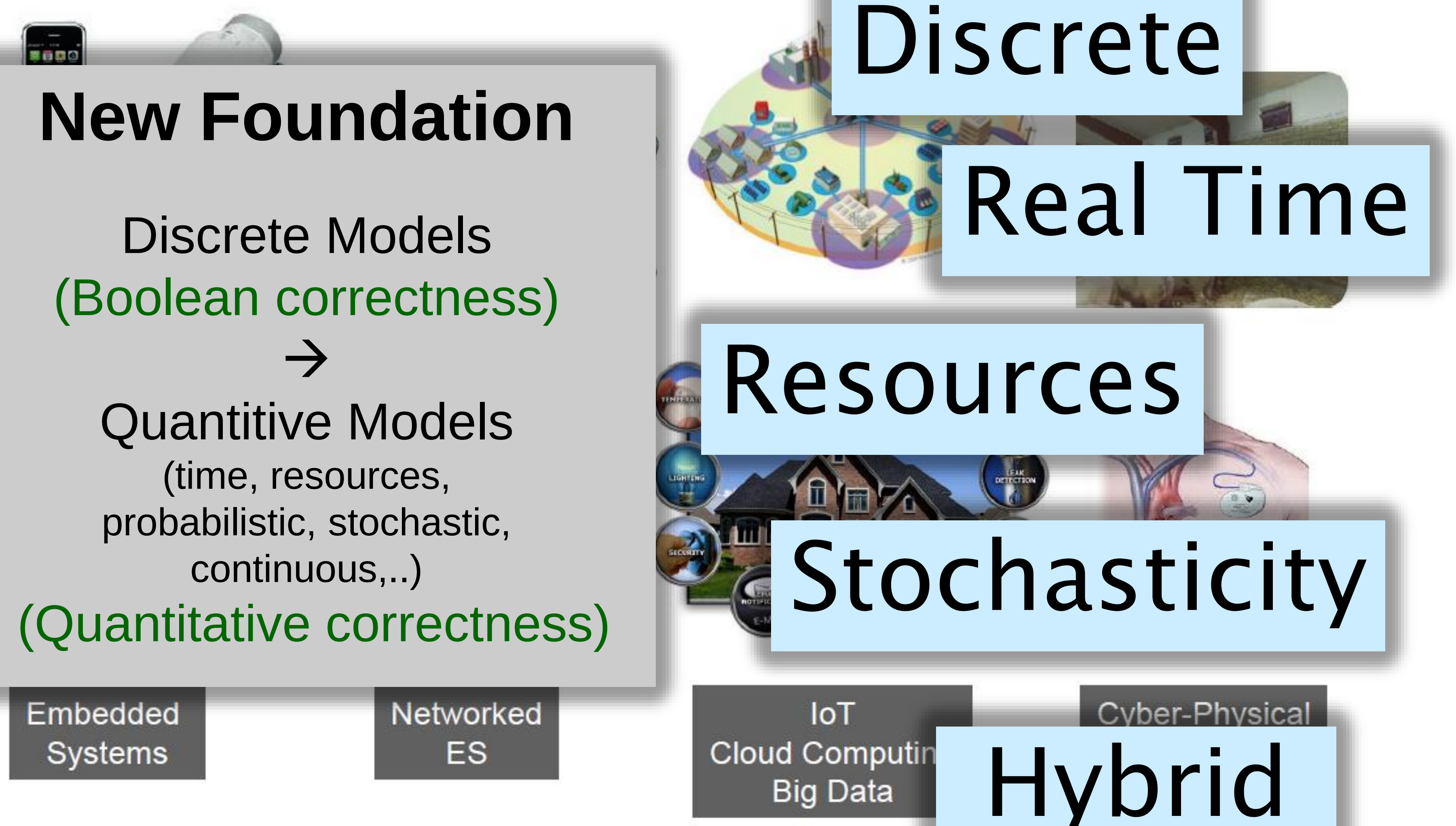
Embedded Systems

Networked ES

IoT
Cloud Computing
Big Data

Cyber-Physical Systems

From ES to CPS



New Foundation

Discrete Models
(Boolean correctness)
→
Quantitive Models
(time, resources, probabilistic, stochastic, continuous,..)
(Quantitative correctness)

Discrete

Real Time

Resources

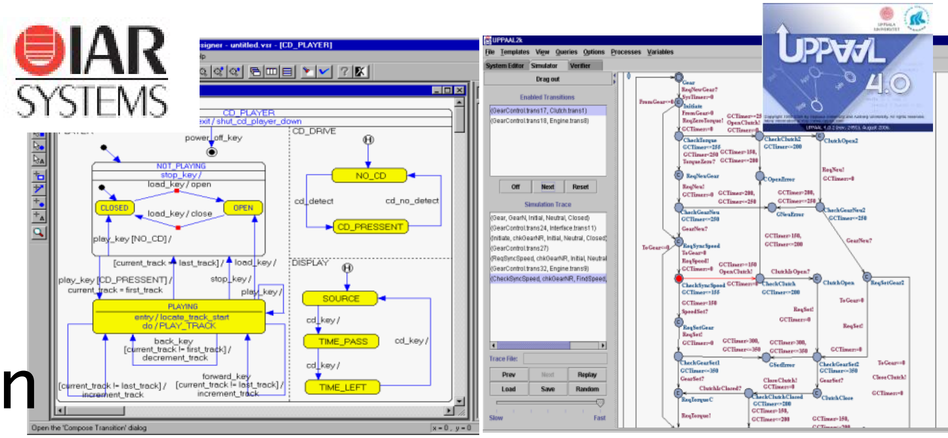
Stochasticity

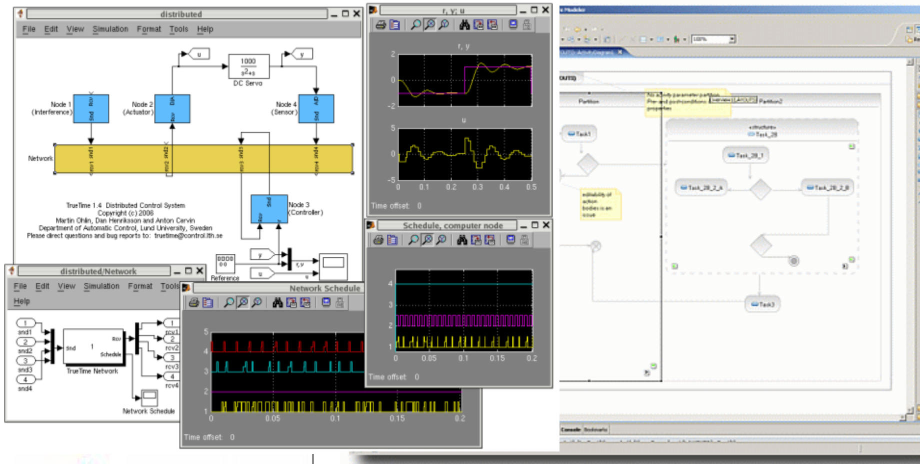
Hybrid

Model–Driven Development

- High–level designs
- Early design–space exploration
- Early error–detection
- Efficient code generation
- Automatization of testing.
- Verification & synthesis.
- Reduced time–to–market.
- Outsourcing
- Reuse and reconfiguration.

IAR SYSTEMS

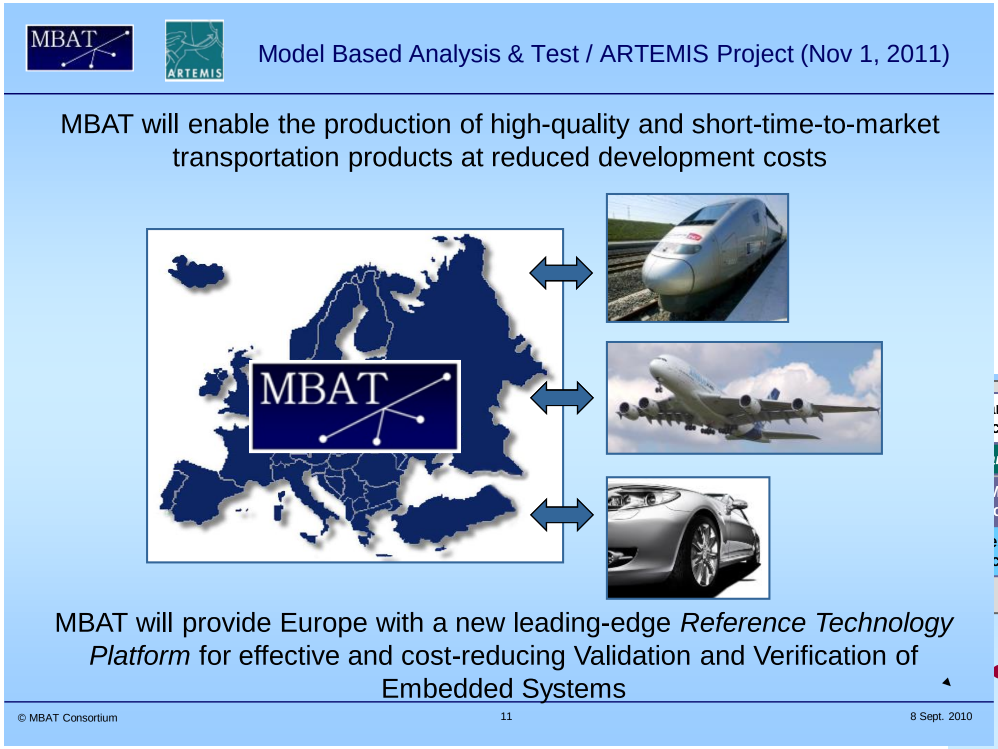




MATLAB SIMULINK

UNIFIED MODELING LANGUAGE

MBAT (2011-2014)
Model-Based Analysis & Test



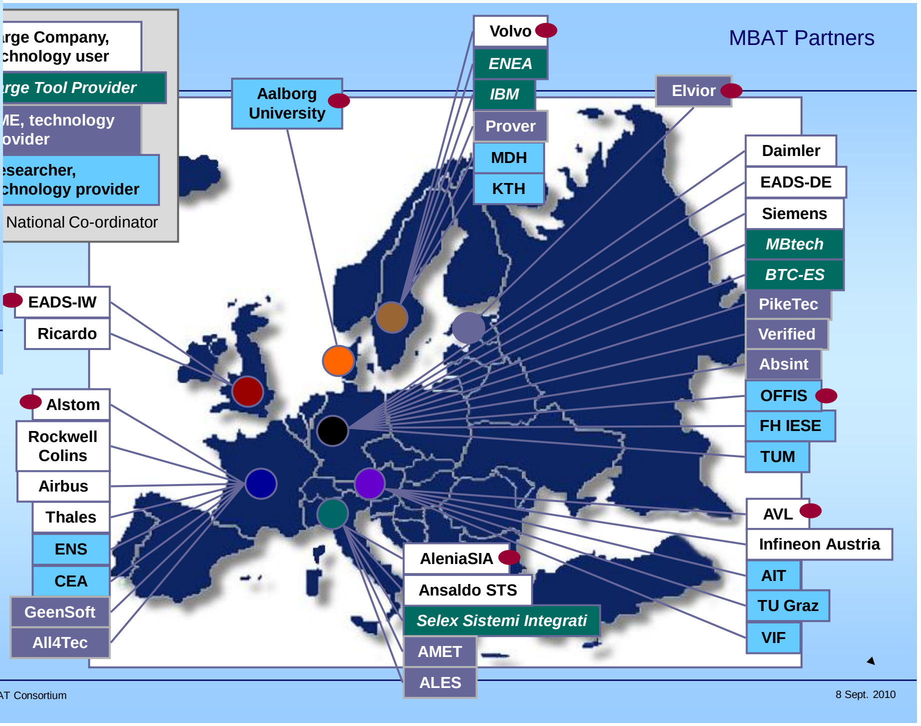
Model Based Analysis & Test / ARTEMIS Project (Nov 1, 2011)

MBAT will enable the production of high-quality and short-time-to-market transportation products at reduced development costs

MBAT
ARTEMIS
MBAT

MBAT will provide Europe with a new leading-edge *Reference Technology Platform* for effective and cost-reducing Validation and Verification of Embedded Systems

© MBAT Consortium 11 8 Sept. 2010



MBAT Partners

rge Company,
chnology user

rge Tool Provider

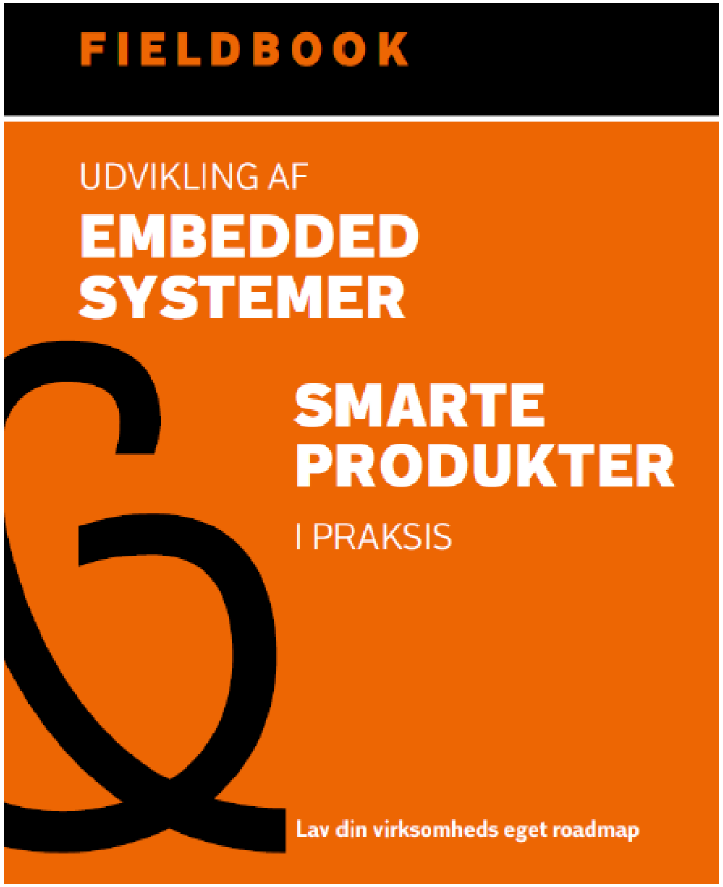
IE, technology
ovider

esearcher,
chnology provider

National Co-ordinator

© MBAT Consortium 8 Sept. 2010

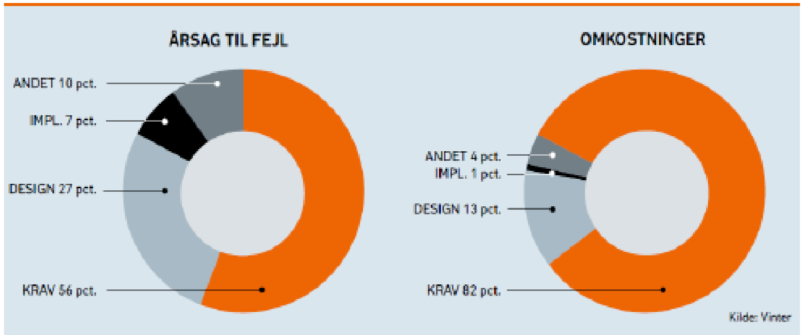
ITOS (2015)
Industrial Technology and Software



FIELDBOOK

UDVIKLING AF **EMBEDDED SYSTEMER** & **SMARTE PRODUKTER** I PRAKSIS

Lav din virksomheds eget roadmap

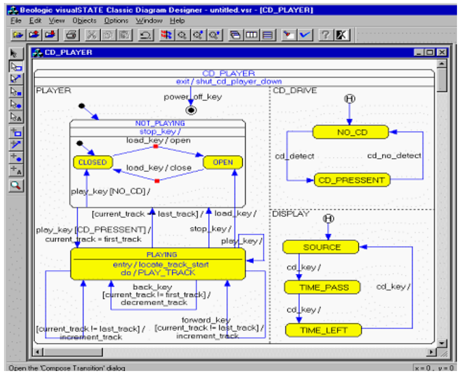


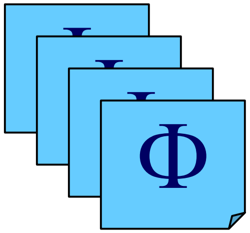
Kilde: Vinter

- 4 industry cases
  - GN Resound
  - MAN Diesel & Turbo
  - Seluxit
  - Terma
- 14 high tech companies
- 2 universities (AAU, DTU)
- 1 fieldbook

Model Driven Development

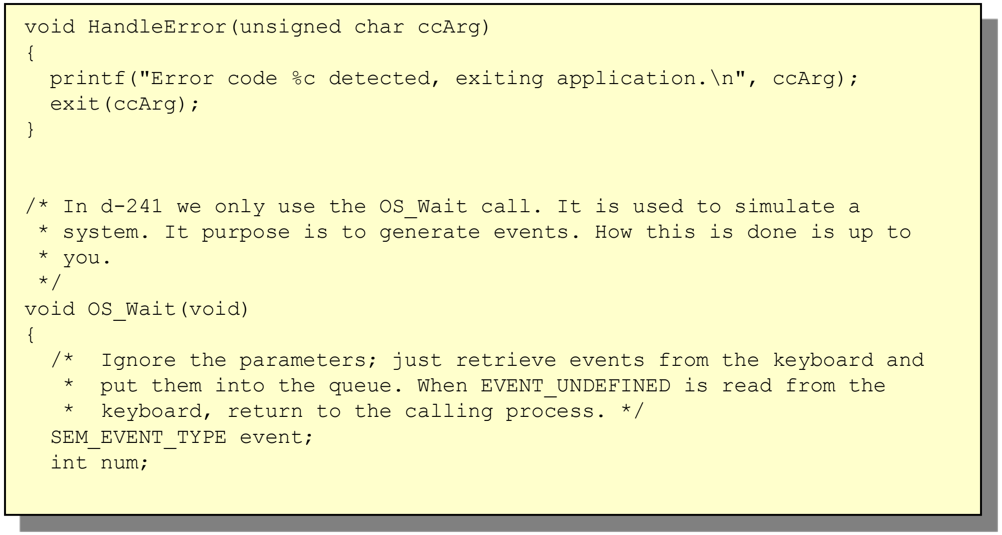
Model





Req

Code

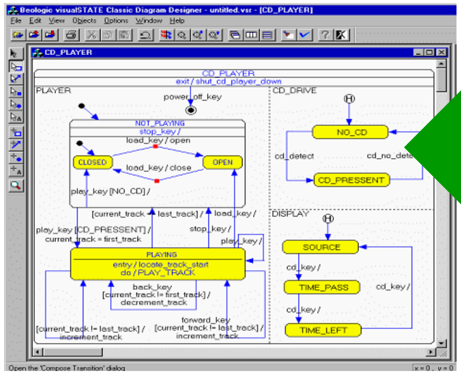


```
void HandleError(unsigned char ccArg)
{
  printf("Error code %c detected, exiting application.\n", ccArg);
  exit(ccArg);
}


/* In d-241 we only use the OS_Wait call. It is used to simulate a
 * system. It purpose is to generate events. How this is done is up to
 * you.
 */
void OS_Wait(void)
{
  /*  Ignore the parameters; just retrieve events from the keyboard and
   *  put them into the queue. When EVENT_UNDEFINED is read from the
   *  keyboard, return to the calling process. */
  SEM_EVENT_TYPE event;
  int num;
```

Running System

Model Checking

Model



Model Checking

Φ

Req

```
void HandleError(unsigned char ccArg)
{
  printf("Error code %c detected, exiting application.\n", ccArg);
  exit(ccArg);
}

                                                                   te a
                                                                   is up to




                                                                   board and
                                                                   om the
```

Coc

Characteristics:

Automata-based

Rich class of properties

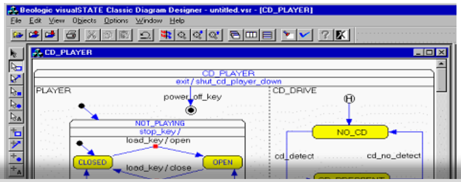
Exact Analysis

State-space Explosion

Running System

Testing & Statistical MC

Model


Beologic visualSTATE Classic Diagram Designer - untitled.vsr - [CD_PLAYER]
File Edit View Objects Options Window Help
CD_PLAYER
CD_PLAYER
exit / shut_cd_player_down
PLAYER
power_off_key
CD_DRIVE
NOT_PLAYING
stop_key /
load_key / open
CLOSED
OPEN
load_key / close
NO_CD
cd_detect
cd_no_detect

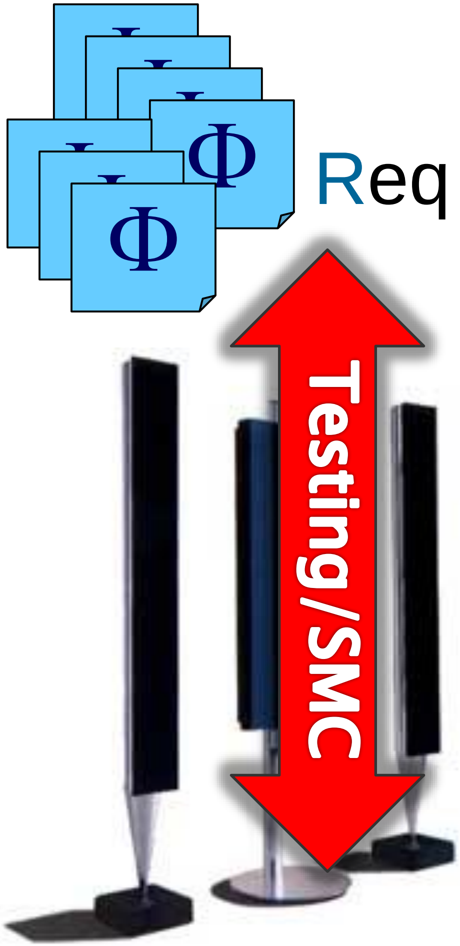
Characteristics:

- System-based
- Very rich properties
- (Under) approximate
- Scalable

Code

```
                                                 on.\n", ccArg);

/* In d-241 we only use the OS_Wait call. It is used to simulate a
 * system. It purpose is to generate events. How this is done is up to
 * you.
 */
void OS_Wait(void)
{
  /*  Ignore the parameters; just retrieve events from the keyboard and
   *  put them into the queue. When EVENT_UNDEFINED is read from the
   *  keyboard, return to the calling process. */
  SEM_EVENT_TYPE event;
  int num;
```

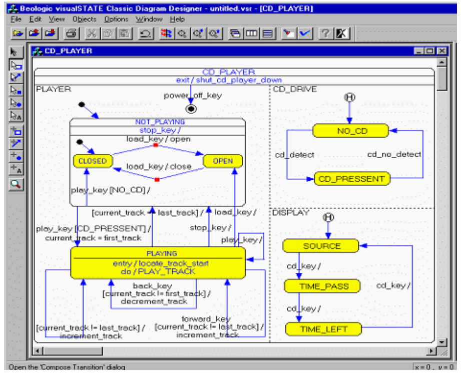
Req

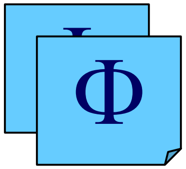


Running System

Static Analysis

Model

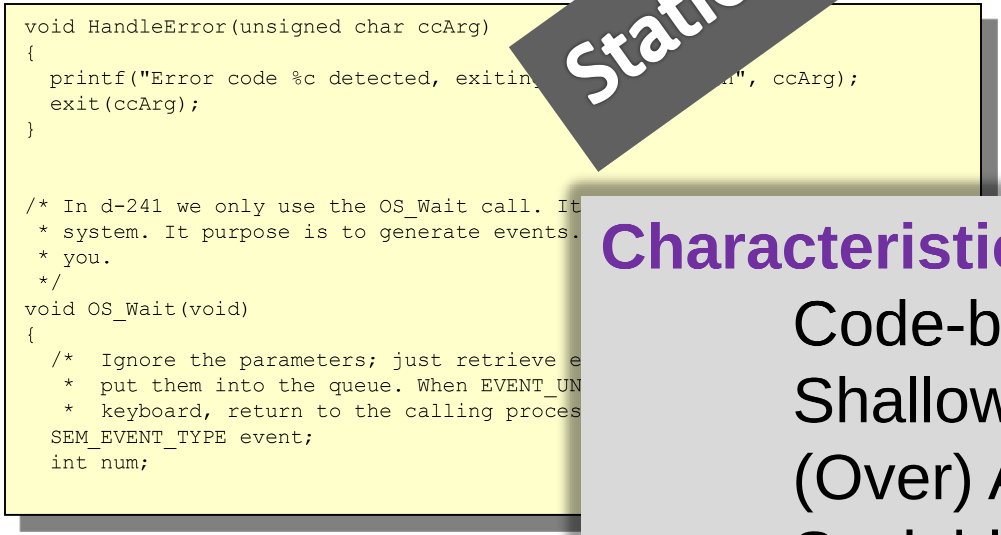




Req

Static Analysis

Code



```
void HandleError(unsigned char ccArg)
{
  printf("Error code %c detected, exitin          n", ccArg);
  exit(ccArg);
}


/* In d-241 we only use the OS_Wait call. It
 * system. It purpose is to generate events.
 * you.
 */
void OS_Wait(void)
{
  /*  Ignore the parameters; just retrieve e
   *  put them into the queue. When EVENT_UN
   *  keyboard, return to the calling proces
  SEM_EVENT_TYPE event;
  int num;
```

Characteristics:

Code-based

Shallow properties

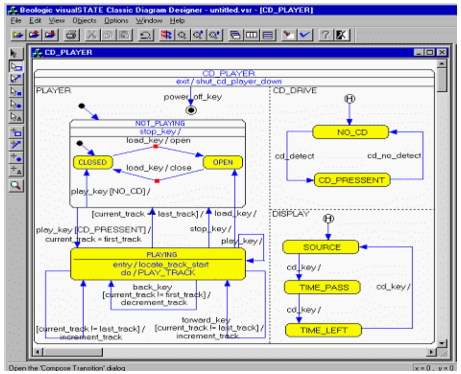
(Over) Approximate

Scalable

ystem

Synthesis

Model



Φ

Req

Synthesis

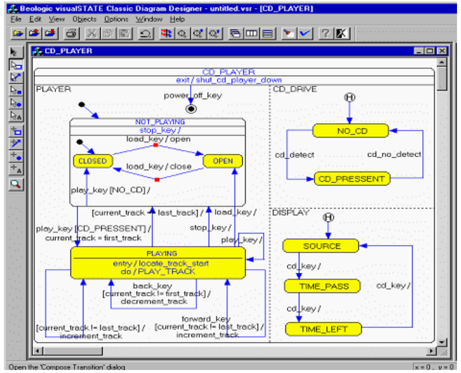
Code

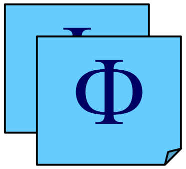
```
void HandleError(unsigned char ccArg)
{
  printf("Error code %c detected, exiting application [illegible]);
  exit(ccArg);
}


/* In d-241 we only use the OS_Wait call. It is used to simulate a
 * system. It purpose is to generate events. How this is done is up to
 * you.
 */
void OS_Wait(void)
{
  /*  Ignore the parameters; just retrieve events from the keyboard and
   *  put them into the queue. When EVENT_UNDEFINED is read from the
   *  keyboard, return to the calling process. */
  SEM_EVENT_TYPE event;
  int num;
```

Running System

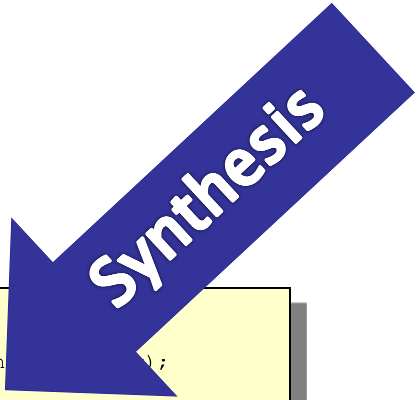
Synthesis

Model





Req



Code

```
void HandleError(unsigned char ccArg)
{
  printf("Error code %c detected, exiting application          );
  exit(ccArg);
}


/* In d-241 we only use the OS_Wait
 * system. It purpose is to generat
 * you.
 */
void OS_Wait(void)
{
  /*  Ignore the parameters; just r
   *  put them into the queue. When
   *  keyboard, return to the calli
  SEM_EVENT_TYPE event;
  int num;
```

Characteristics:

Rich Properties

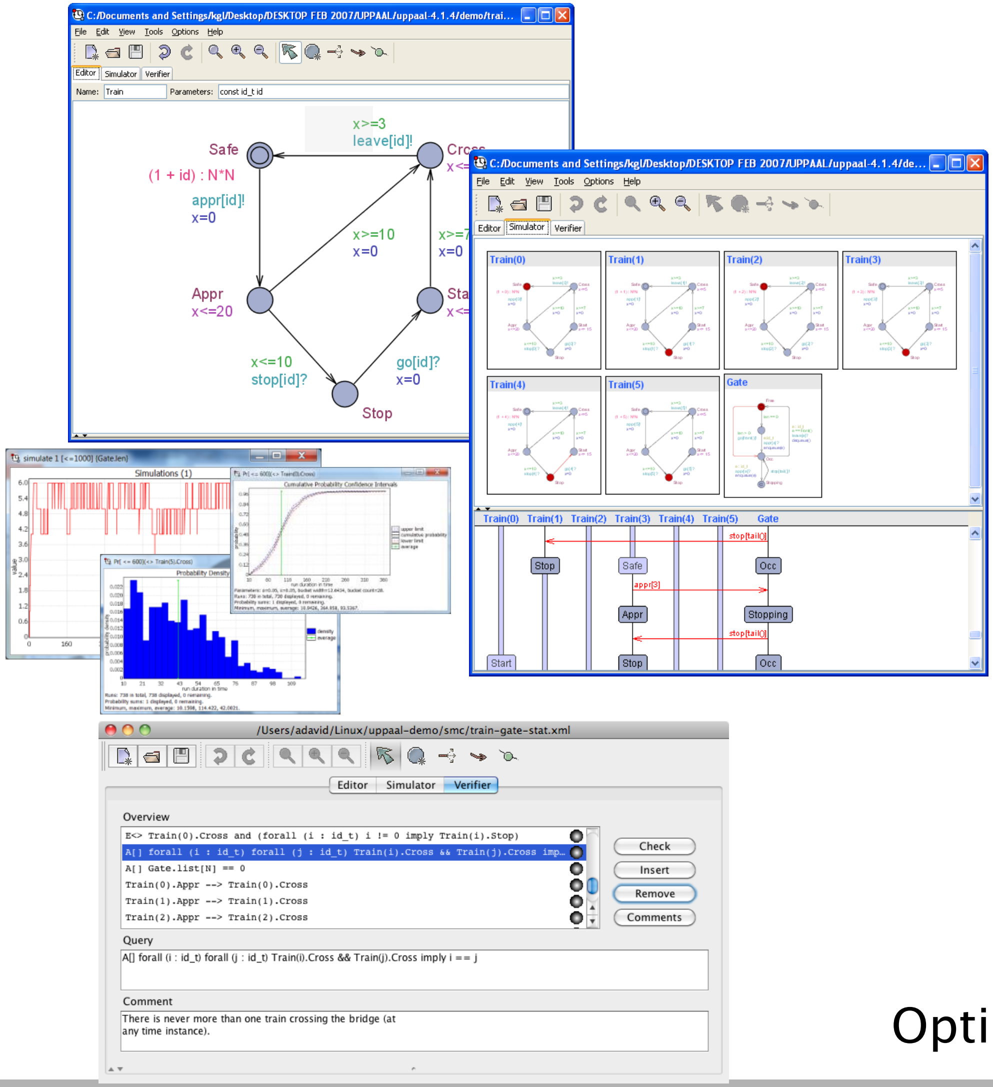
Automatic generation of code

Easy reprogrammable

Complexity

Running System

UPPAAL Tool Suit


C:/Documents and Settings/kgl/Desktop/DESKTOP FEB 2007/UPPAAL/uppaal-4.1.4/demo/trai...
File Edit View Tools Options Help
Editor Simulator Verifier
Name: Train Parameters: const id_t id
Safe
(1 + id) : N*N
appr[id]!
x=0
x>=3
leave[id]!
Cross
x>=10
x=0
Appr
x<=20
x<=10
stop[id]?
Stop
go[id]?
x=0
Train(0) Train(1) Train(2) Train(3) Train(4) Train(5) Gate
stop[tail()]
Stop
Safe
appr[3]
Occ
Appr
Stopping
Start
Simulations (1)
Cumulative Probability Confidence Intervals
Probability Density
/Users/adavid/Linux/uppaal-demo/smc/train-gate-stat.xml
Editor Simulator Verifier
Check
Insert
Remove
Comments

Overview

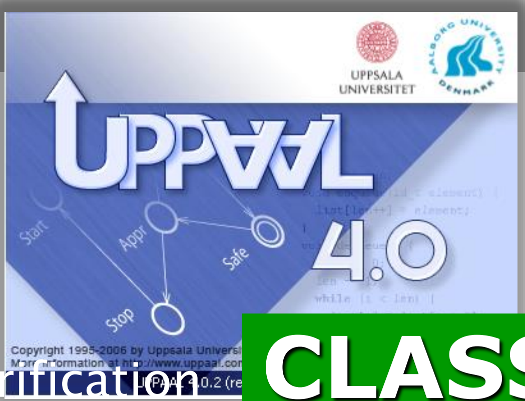
```
E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)
A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imp...
A[] Gate.list[N] == 0
Train(0).Appr --> Train(0).Cross
Train(1).Appr --> Train(1).Cross
Train(2).Appr --> Train(2).Cross
```

Query

```
A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j
```

Comment

There is never more than one train crossing the bridge (at any time instance).


UPPSALA UNIVERSITET
AALBORG UNIVERSITY DENMARK
UPPAAL
4.0
Start
Appr
Safe
Stop
Copyright 1995-2006 by Uppsala Universi
More information at http://www.uppaal.com

Verification	CLASSIC
Optimization	CORA
Synthesis	TIGA
Component	ECDAR
Testing	TRON
Performance Analysis	SMC
Optimal Synthesis	STRATEGO

Overview

- Timed Automata / UPPAAL
  - Verification
- Stochastic Priced Timed Automata / UPPAAL SMC
  - Performance Evaluation
  - SMC in a Nutshell
  - Stochastic Hybrid Automata
- Timed Games / UPPAAL TIGA
  - Controller Syntesis
- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synthesis
- Conclusion

Train Gate

Train Gate
Schedulability Analysis

Train Gate

Train Gate
Floor Heating
Adaptive Cruise Control

Overview

- Timed Automata / UPPAAL
  - Verification
- Stochastic Priced Timed Automata / UPPAAL SMC
  - Performance Evaluation
  - SMC in a Nutshell
  - Stochastic Hybrid Automata
- Timed Games / UPPAAL TIGA
  - Controller Syntesis
- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synthesis
- Conclusion

Train Gate

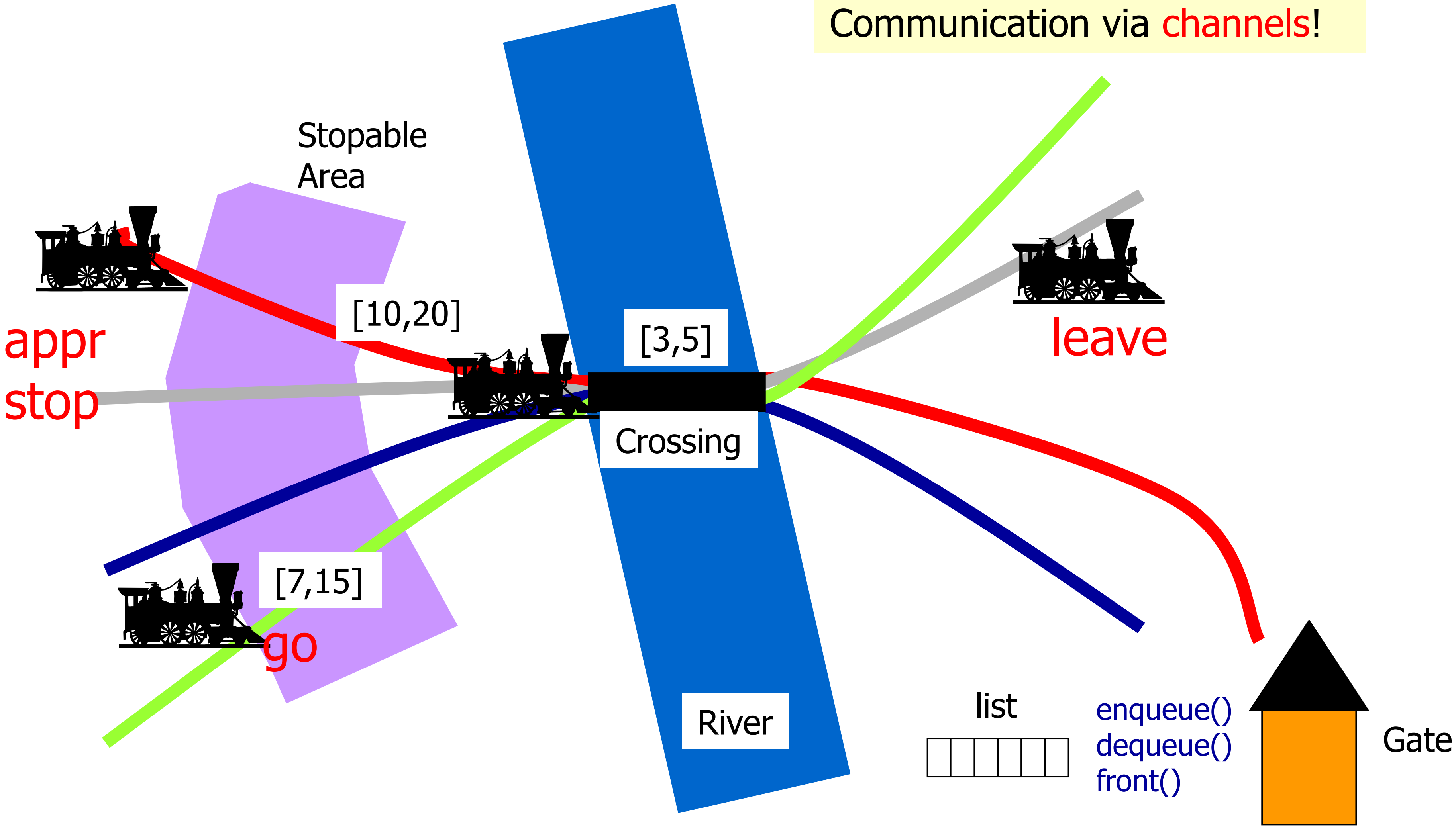
Train Gate
Schedulability Analysis

Train Gate

Train Gate
Floor Heating
Adaptive Cruise Control

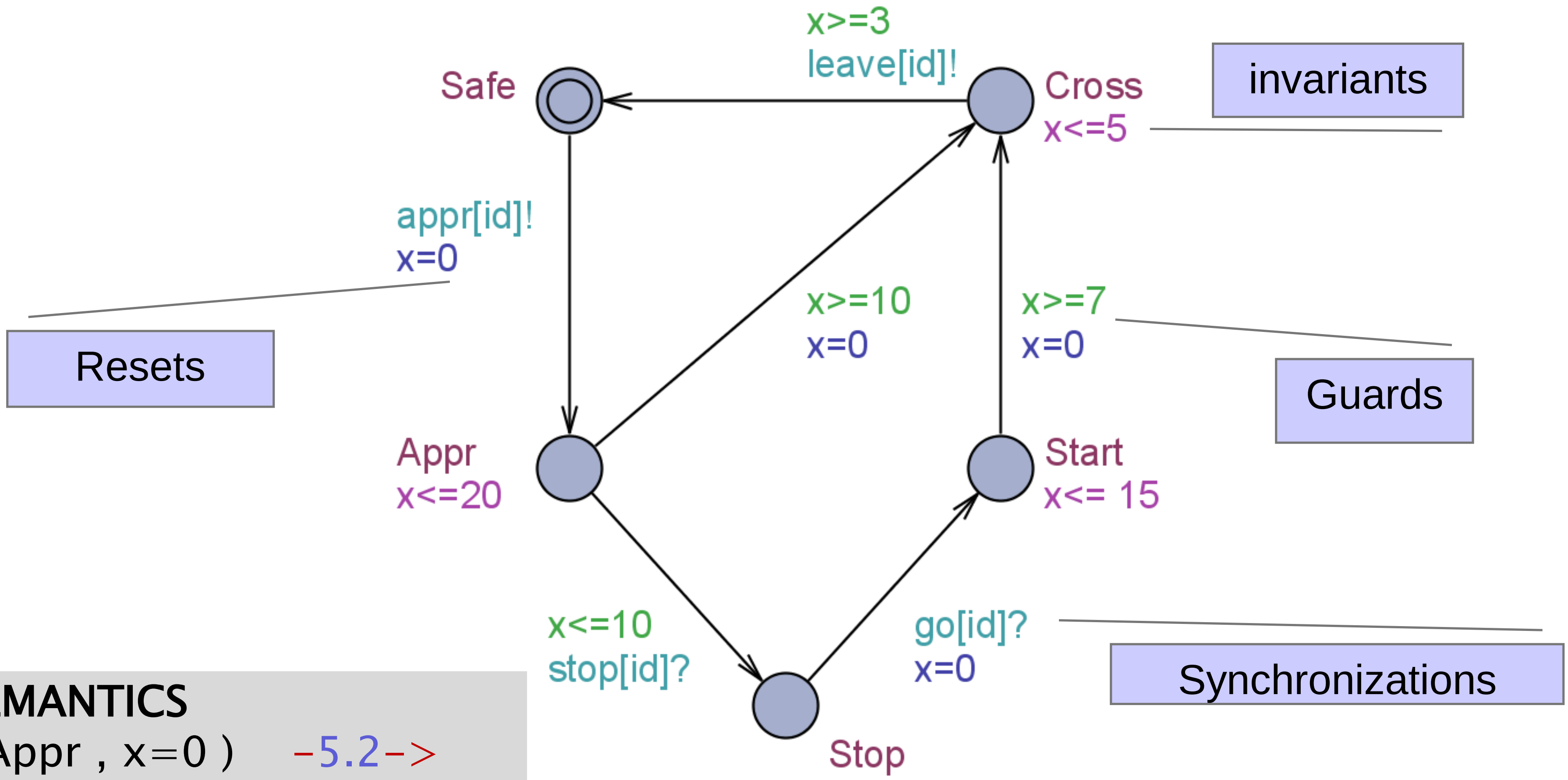
Train Scheduling

Communication via channels!



Timed Automata [Train]

= Finite State Control
+ Real Valued Clocks



SEMANTICS

(Appr , x=0) –5.2–>
(Appr , x=5.2) –stop? –>
(Stop , x=5.2)

Logical Specifications

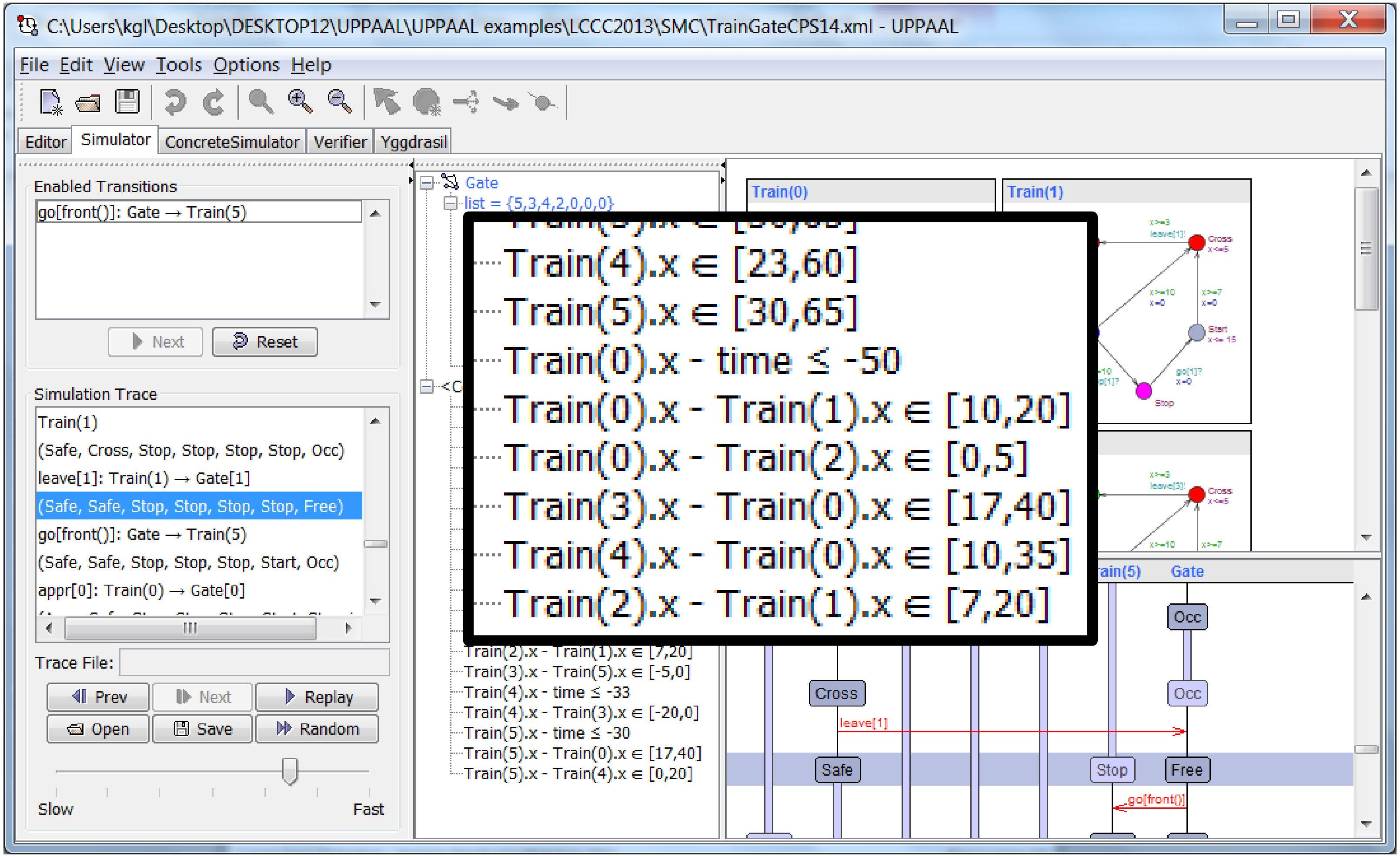
- Validation Properties
  - Possibly: E<> *P*
- Safety Properties
  - Invariant: A[] *P*
  - Pos. Inv.: E[] *P*
- Liveness Properties
  - Eventually: A<> *P*
  - Leadsto: $\mathrm{P} \rightarrow Q$
- Bounded Liveness
  - Leads to within: $P \rightarrow_{\leq t} Q$

The expressions *P* and *Q* must be type safe, side effect free, and evaluate to a boolean.

Only references to integer variables, constants, clocks, and locations are allowed (and arrays of these).

DEMO

THE ”secret” of UPPAAL



C:\Users\kgl\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS14.xml - UPPAAL

Enabled Transitions

go[front()]: Gate → Train(5)

Simulation Trace

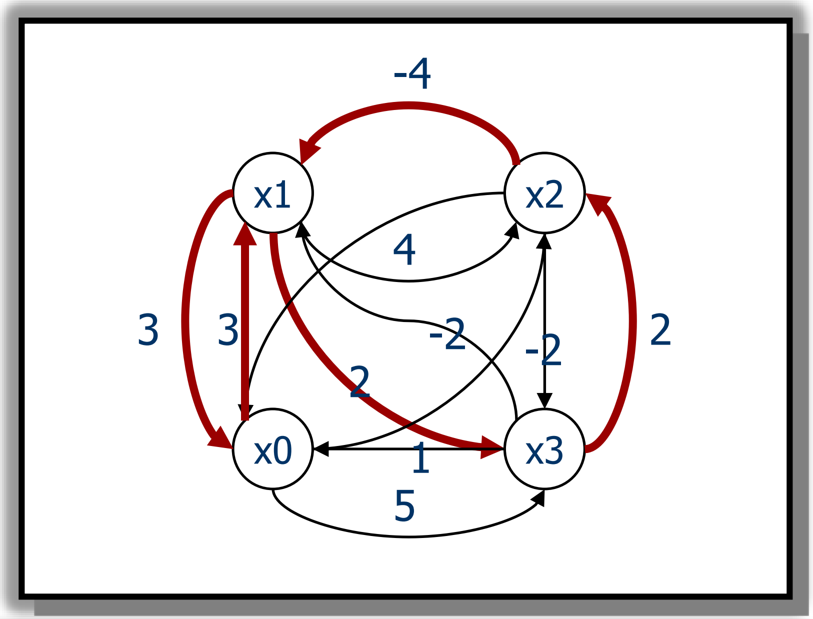
```
Train(1)
(Safe, Cross, Stop, Stop, Stop, Stop, Occ)
leave[1]: Train(1) → Gate[1]
(Safe, Safe, Stop, Stop, Stop, Stop, Free)
go[front()]: Gate → Train(5)
(Safe, Safe, Stop, Stop, Stop, Start, Occ)
appr[0]: Train(0) → Gate[0]
```

Trace File:

```
Gate
  list = {5,3,4,2,0,0,0}
Train(4).x ∈ [23,60]
Train(5).x ∈ [30,65]
Train(0).x - time ≤ -50
Train(0).x - Train(1).x ∈ [10,20]
Train(0).x - Train(2).x ∈ [0,5]
Train(3).x - Train(0).x ∈ [17,40]
Train(4).x - Train(0).x ∈ [10,35]
Train(2).x - Train(1).x ∈ [7,20]
Train(3).x - Train(5).x ∈ [-5,0]
Train(4).x - time ≤ -33
Train(4).x - Train(3).x ∈ [-20,0]
Train(5).x - time ≤ -30
Train(5).x - Train(0).x ∈ [17,40]
Train(5).x - Train(4).x ∈ [0,20]
```

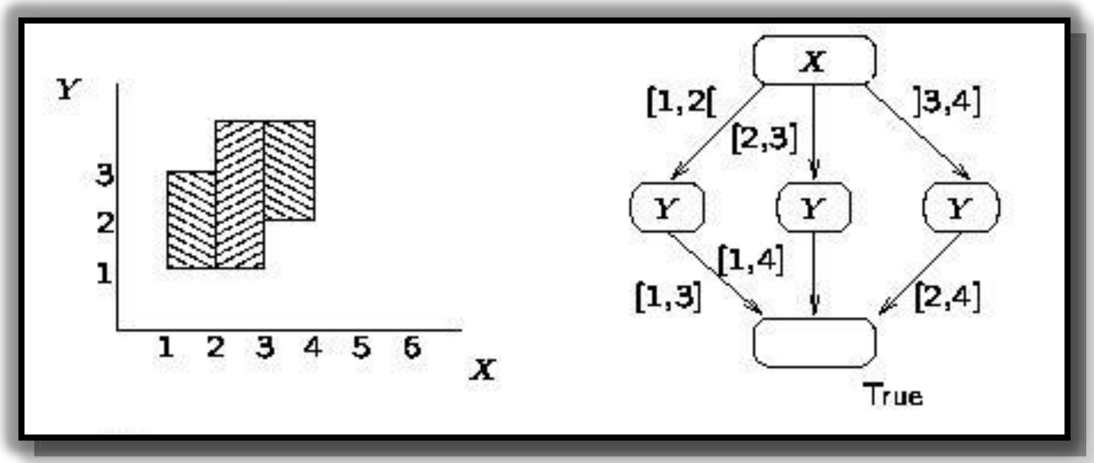
Datastructures for Zones

- Difference Bounded Matrices (DBMs)



- Minimal Constraint Form
  [RTSS97]

- Clock Difference Diagrams
  [CAV99]



Overview

- Timed Automata / UPPAAL
  - Verification

Train Gate

- Stochastic Priced Timed Automata / UPPAAL SMC
  - Performance Evaluation
  - SMC in a Nutshell
  - Stochastic Hybrid Automata

Train Gate
Schedulability Analysis

- Timed Games / UPPAAL TIGA
  - Controller Syntesis

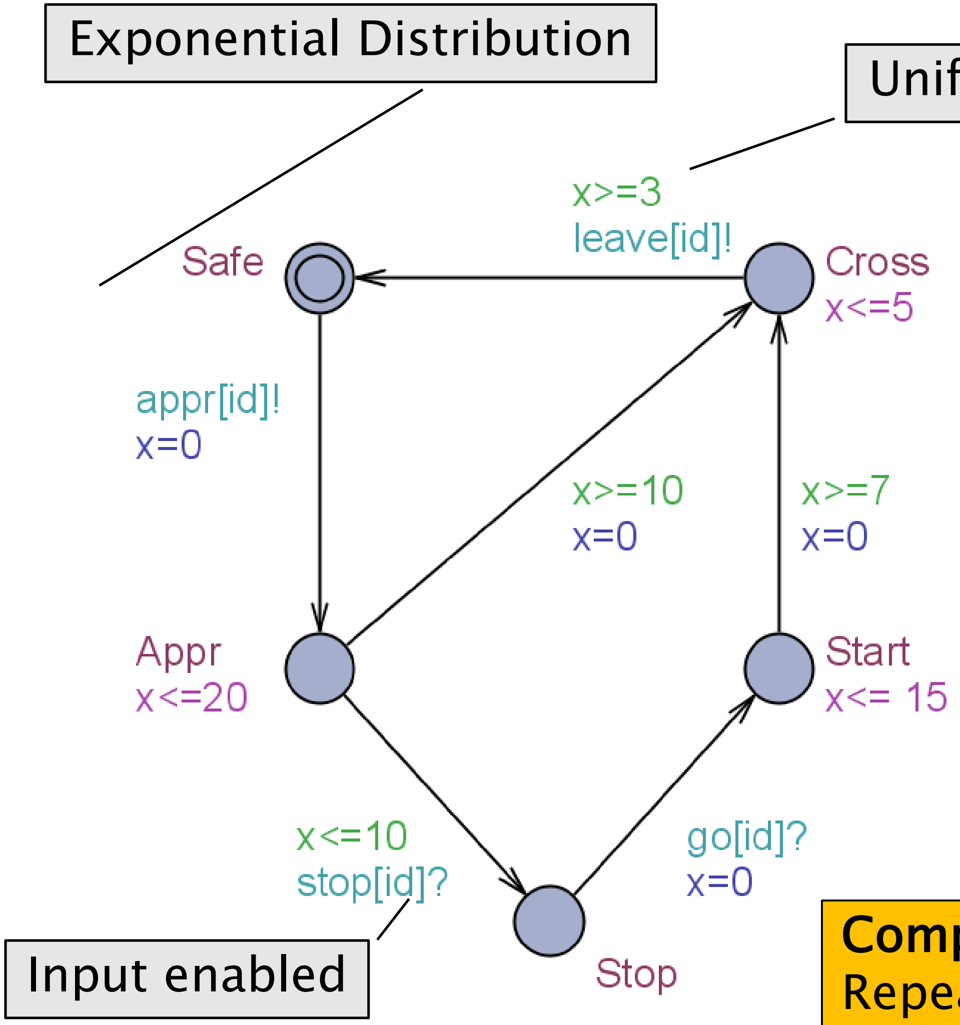
Train Gate

- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synthesis

Train Gate
Floor Heating
Adaptive Cruise Control

- Conclusion

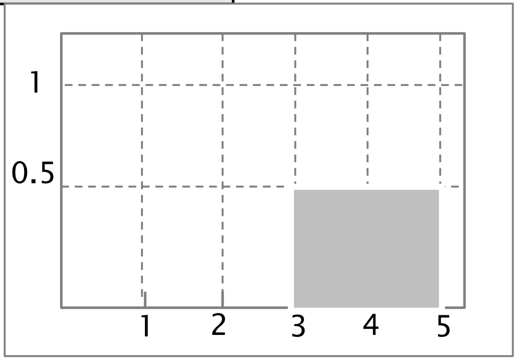
Stochastic Semantics of TA

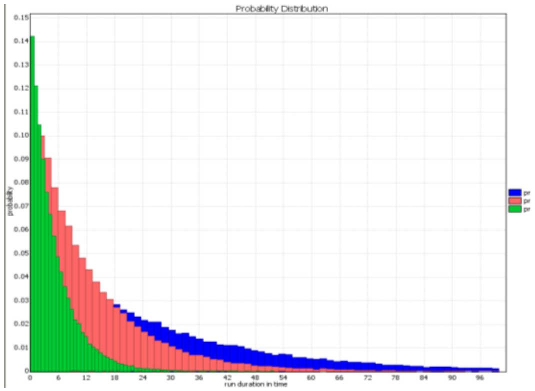


Exponential Distribution

Uniform Distribution

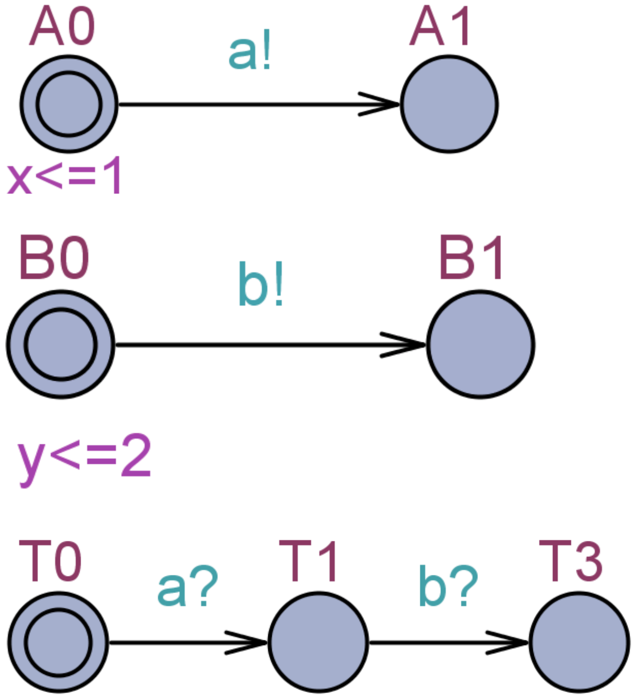
Input enabled





Composition =
Repeated races between components
for **outputting**

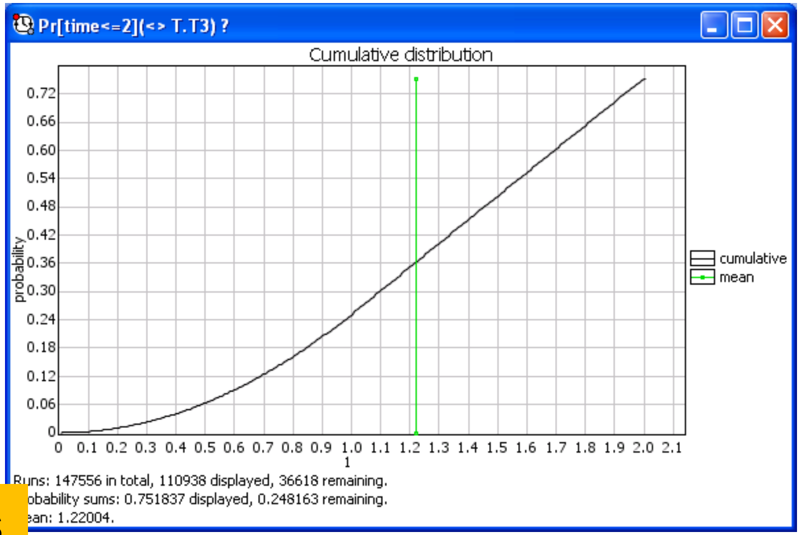
Composition of STA



Pr[time<=2](<> T.T3) ?

$$= \int_{t_a=0}^{1} 1 \cdot \int_{t_b=t_a}^{2} ½ \; dt_b \quad dt_a = 3/4$$

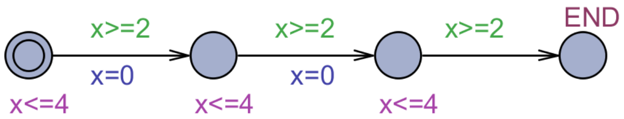
Pr[time<=T](<> T.T3) ?

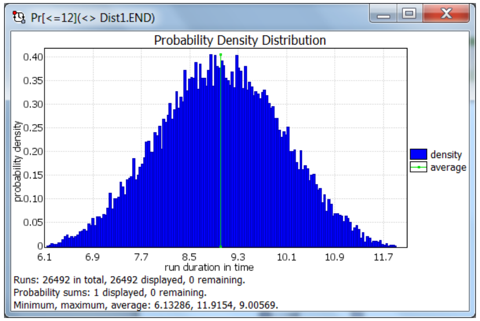


Runs: 147556 in total, 110938 displayed, 36618 remaining.
obability sums: 0.751837 displayed, 0.248163 remaining.
ean: 1.22004.

Composition = Race between components for outputting

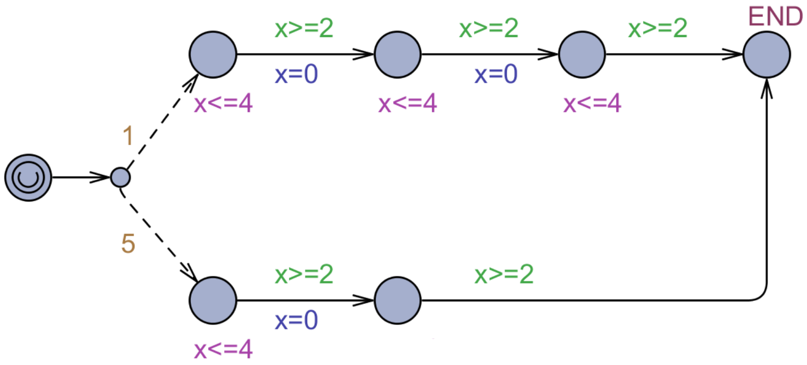
Beyond Uniform / Exponential Dist.

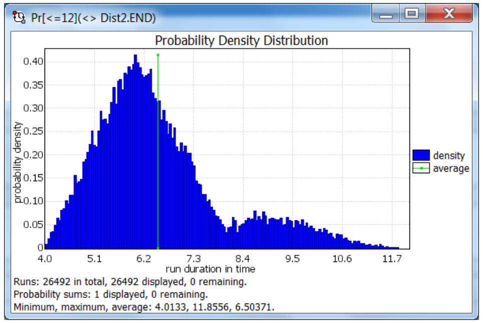




Runs: 26492 in total, 26492 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 6.13286, 11.9154, 9.00569.

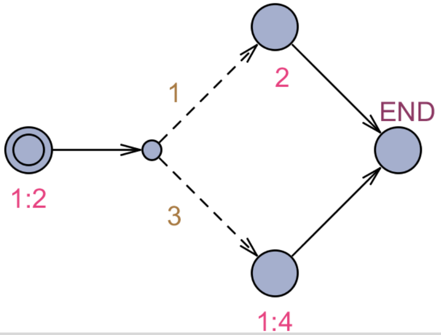
$\Pr(\langle\rangle_{\leq 9}\ \mathrm{END}) = ½$

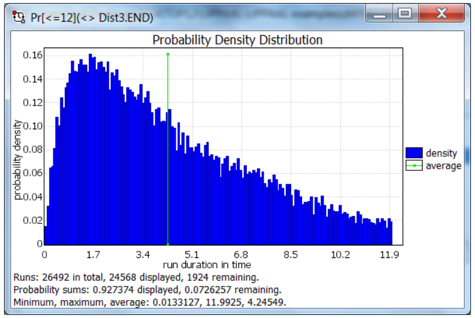




Runs: 26492 in total, 26492 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 4.0133, 11.8556, 6.50371.

$\Pr(\langle\rangle_{\leq 7}\ \mathrm{END}) \geq ½$





Runs: 26492 in total, 24568 displayed, 1924 remaining.
Probability sums: 0.927374 displayed, 0.0726257 remaining.
Minimum, maximum, average: 0.0133127, 11.9925, 4.24549.

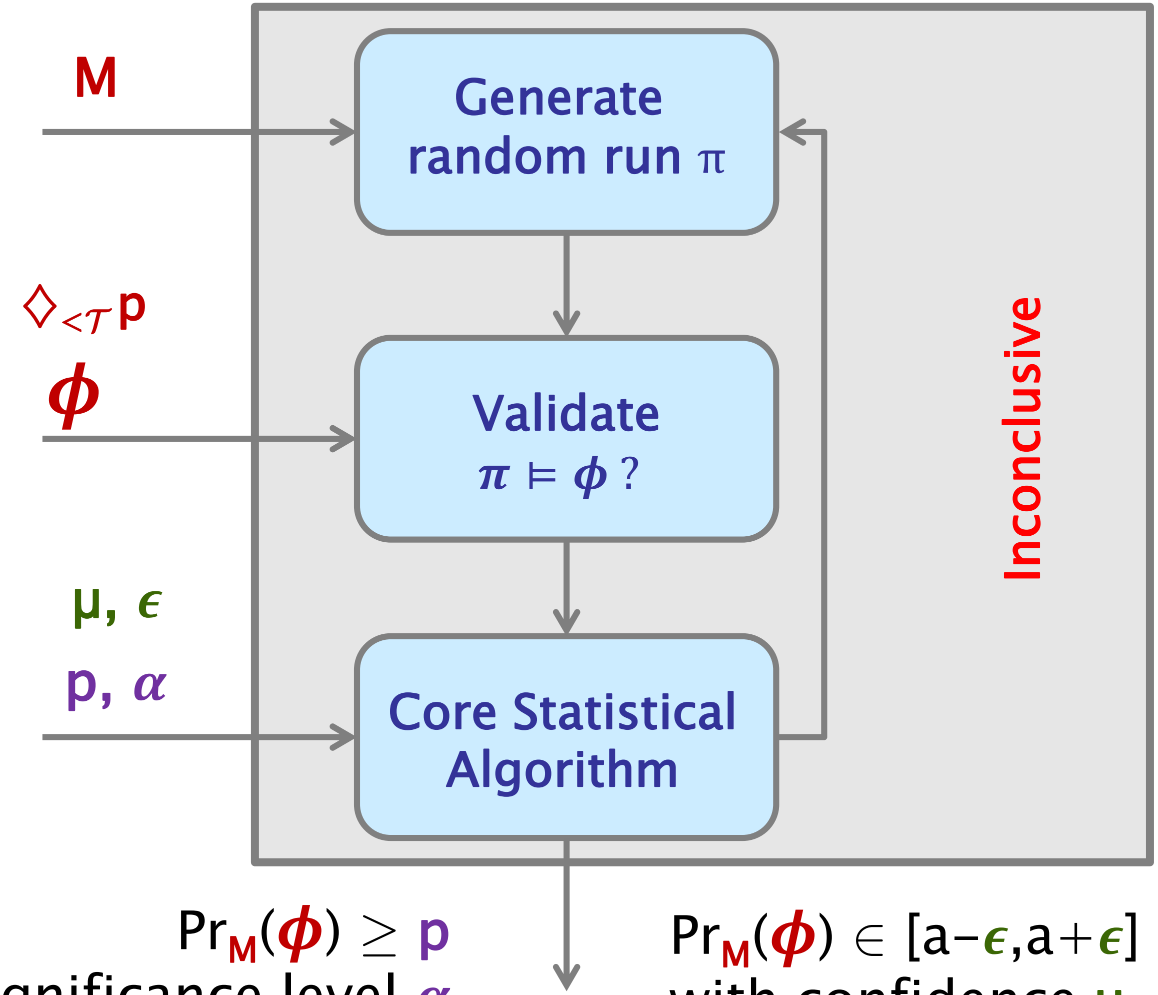
Includes all Phase-Type Distributions.

Can encode any distribution with arbitrary precision.

σ-algebra with prob. measure from cylinders $\boldsymbol{C(I_0\ \ell_0\ I_1\ \ell_1\ I_2\ \ldots\ \ I_n\ \ell_{n+1})}$

Statistical Model Checking

[FORMATS11, LPAR12, RV12]



Reachability
MITL

Hypothesis testing

$\Pr_M(\phi) \geq p$
at significance level α

Confidence Interval

$\Pr_M(\phi) \in [a-\epsilon, a+\epsilon]$
with confidence μ

Queries in UPPAAL Syntax

- Evaluation

  `Pr[<=100](<> expr)` $\mathbf{Pr}(\Phi)\!: \Phi \in \mathit{MITL}$

  Hypothesis testing

  ```
  Pr[<=100](<> expr) >= 0.1
  c<=100 #<=50 [] expr <=0.5
  ```

- Comparison

  ```
  Pr[<=20](<> e1) >= Pr[<=10](<> e2)
  ```

- Expected value

  ```
  E[<=10;1000](min: expr)
  ```

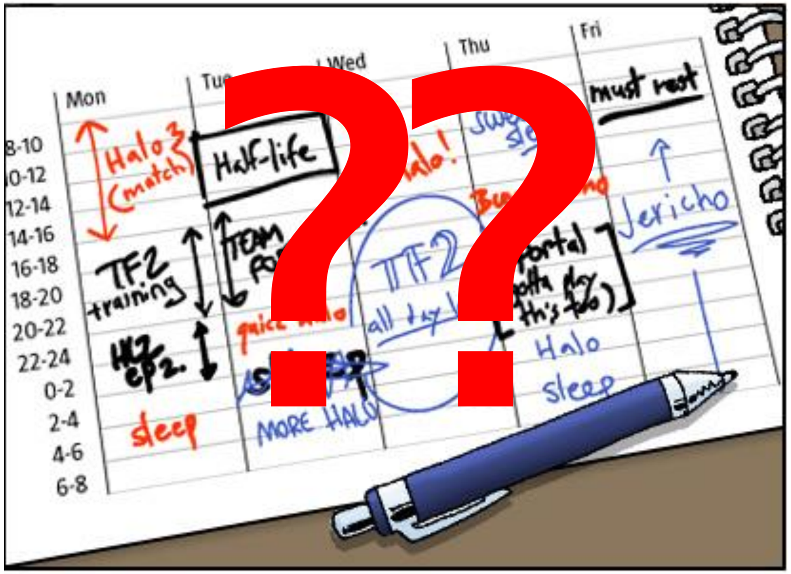
  Explicit number of runs. Min or max.

- Simulations

  ```
  simulate 10 [<=100]{expr1,expr2}
  ```

DEMO

Schedulability
& Performance Analysis



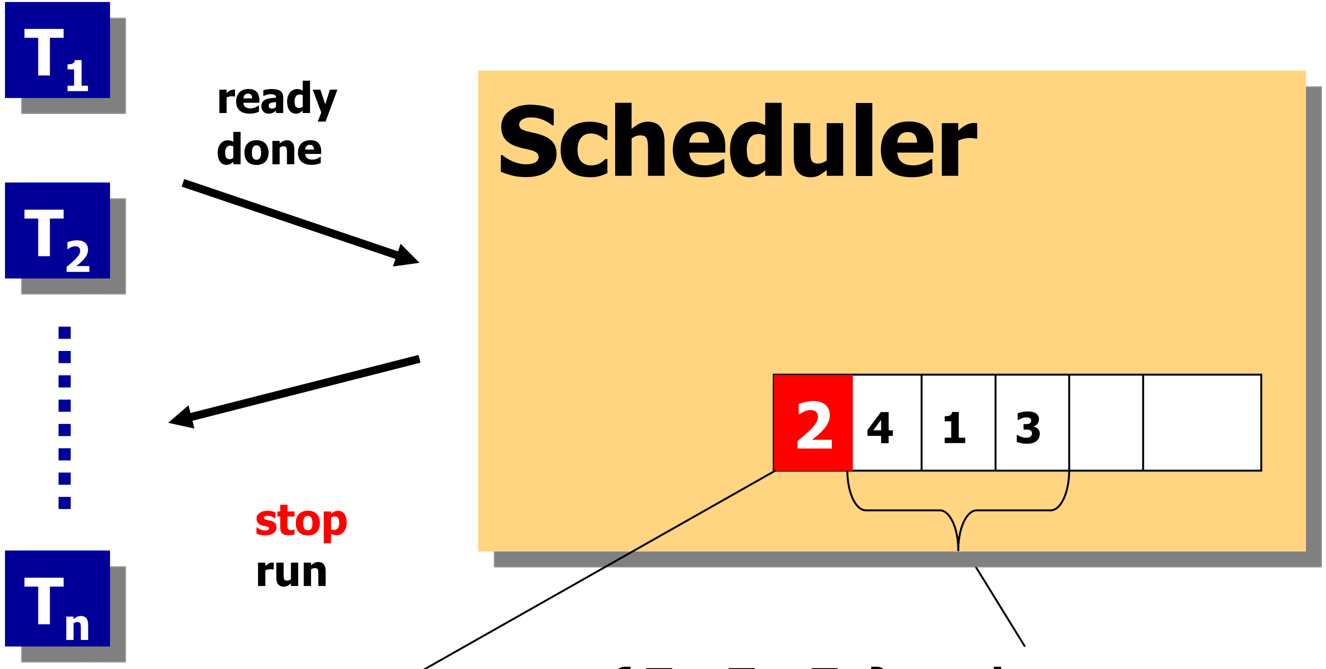
AALBORG UNIVERSITY DENMARK

Task Scheduling *utilization of CPU*

P(i), UNI[E(i), L(i)], .. : period or earliest/latest arrival or .. for T_i

C(i), UNI[BC(i),WC(i)] : execution time for T_i

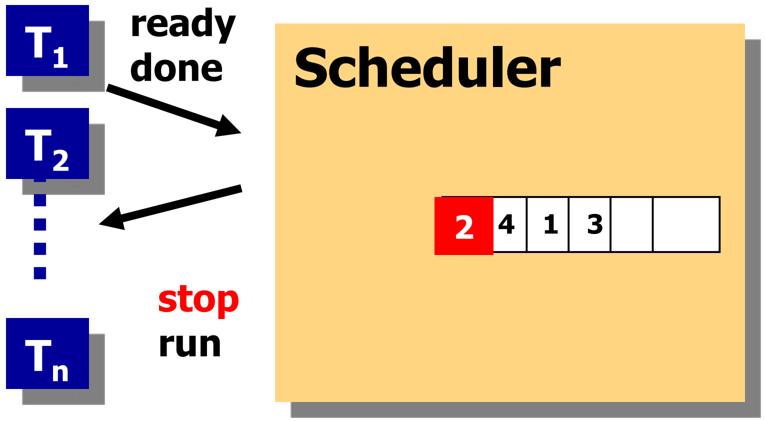
D(i): deadline for T_i

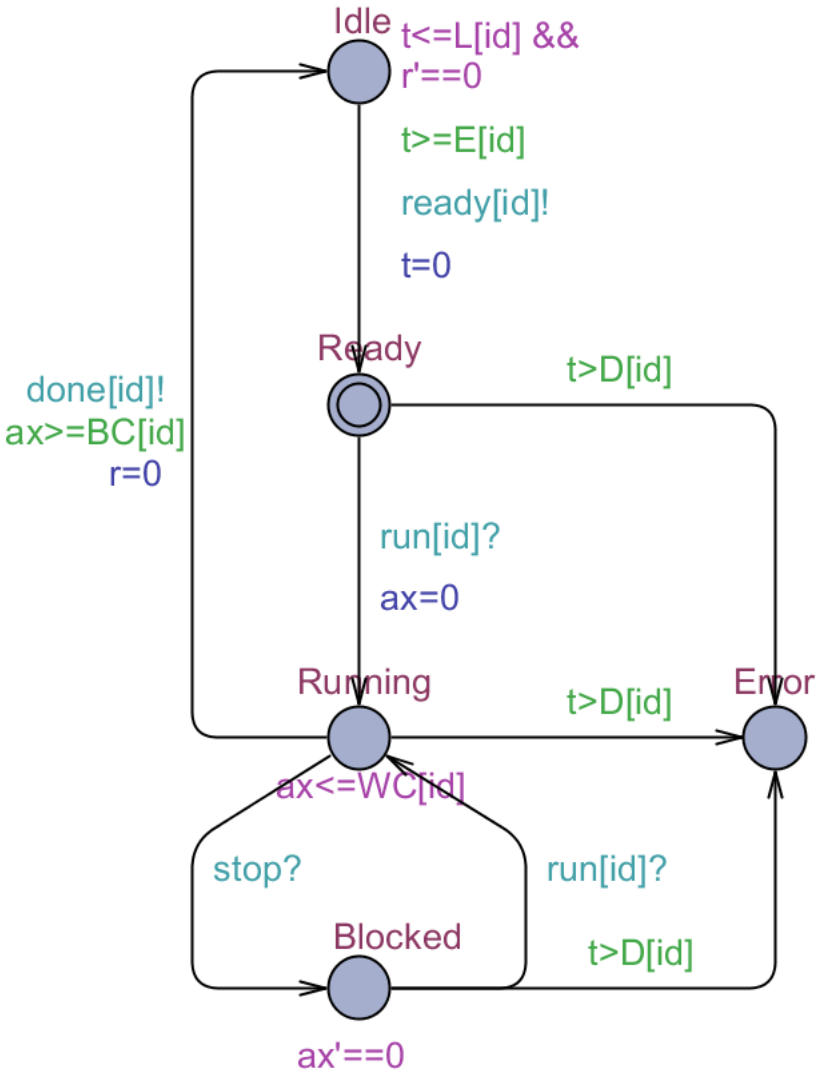


T_2 is running

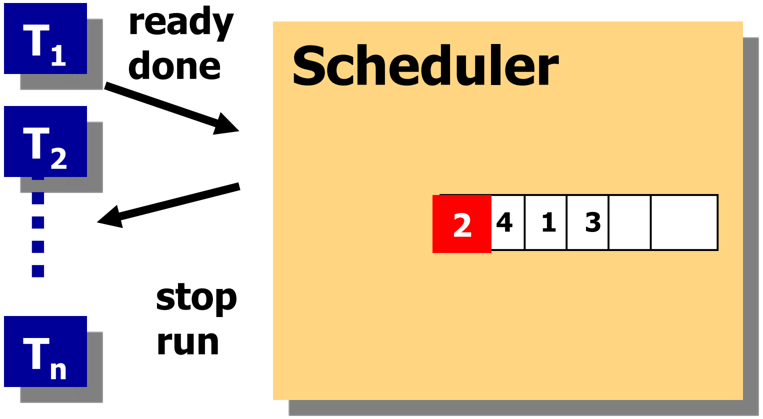
{ T_4 , T_1 , T_3 } ready ordered according to some given priority: (e.g. Fixed Priority, Earliest Deadline,..)

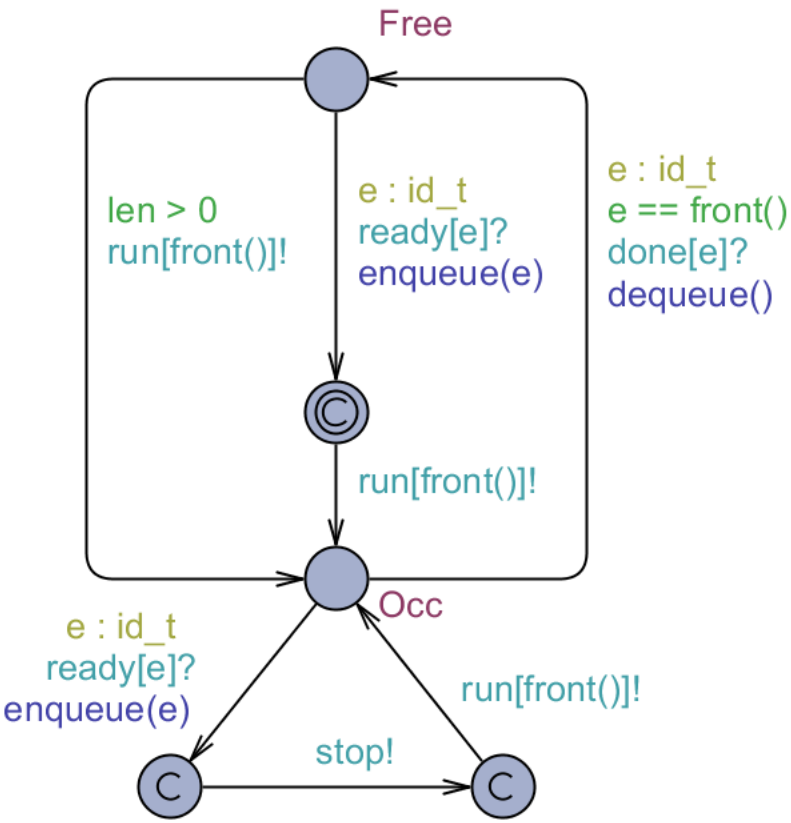
Modeling Task



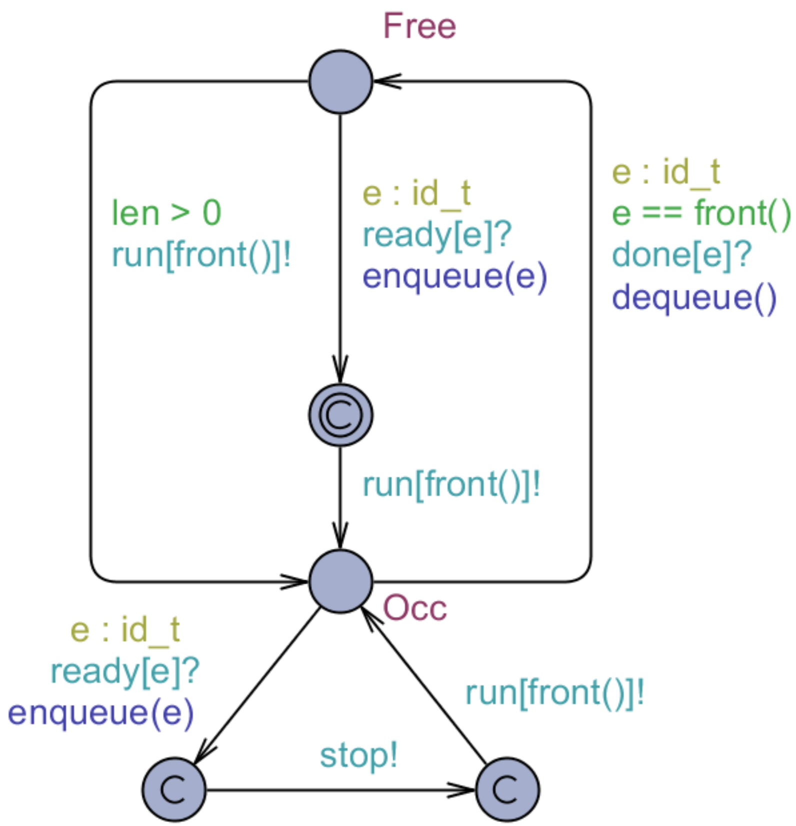


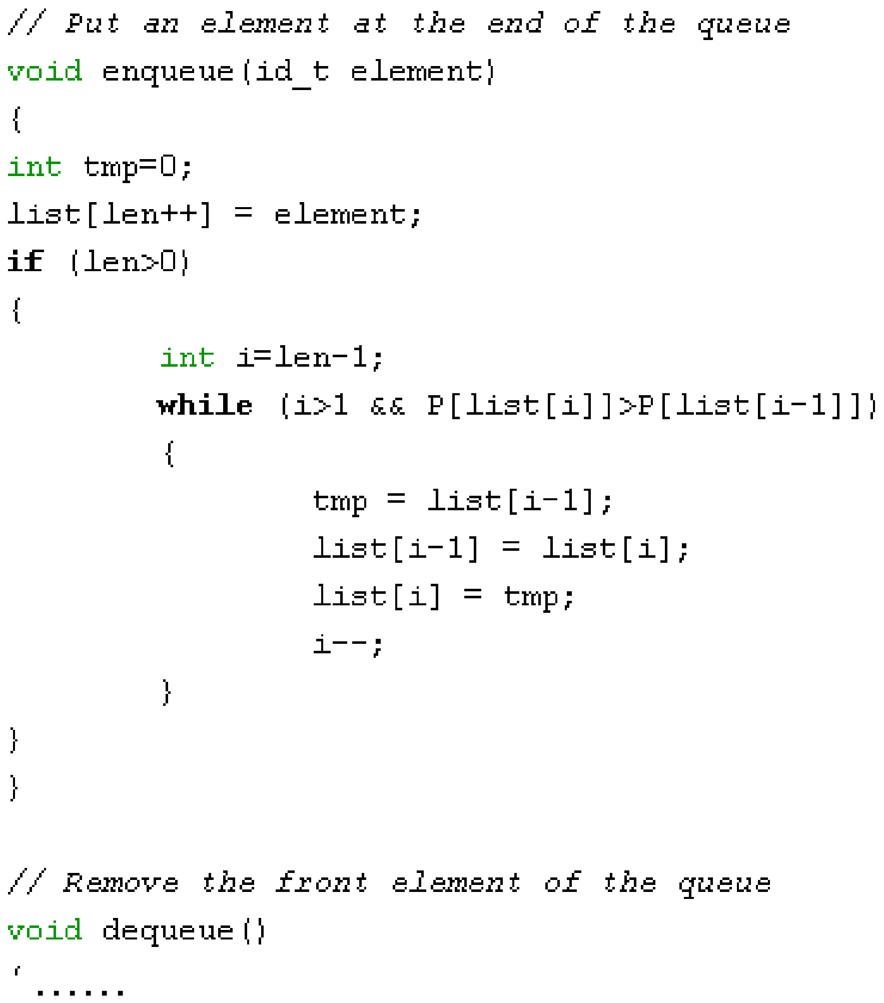
Modeling Scheduler





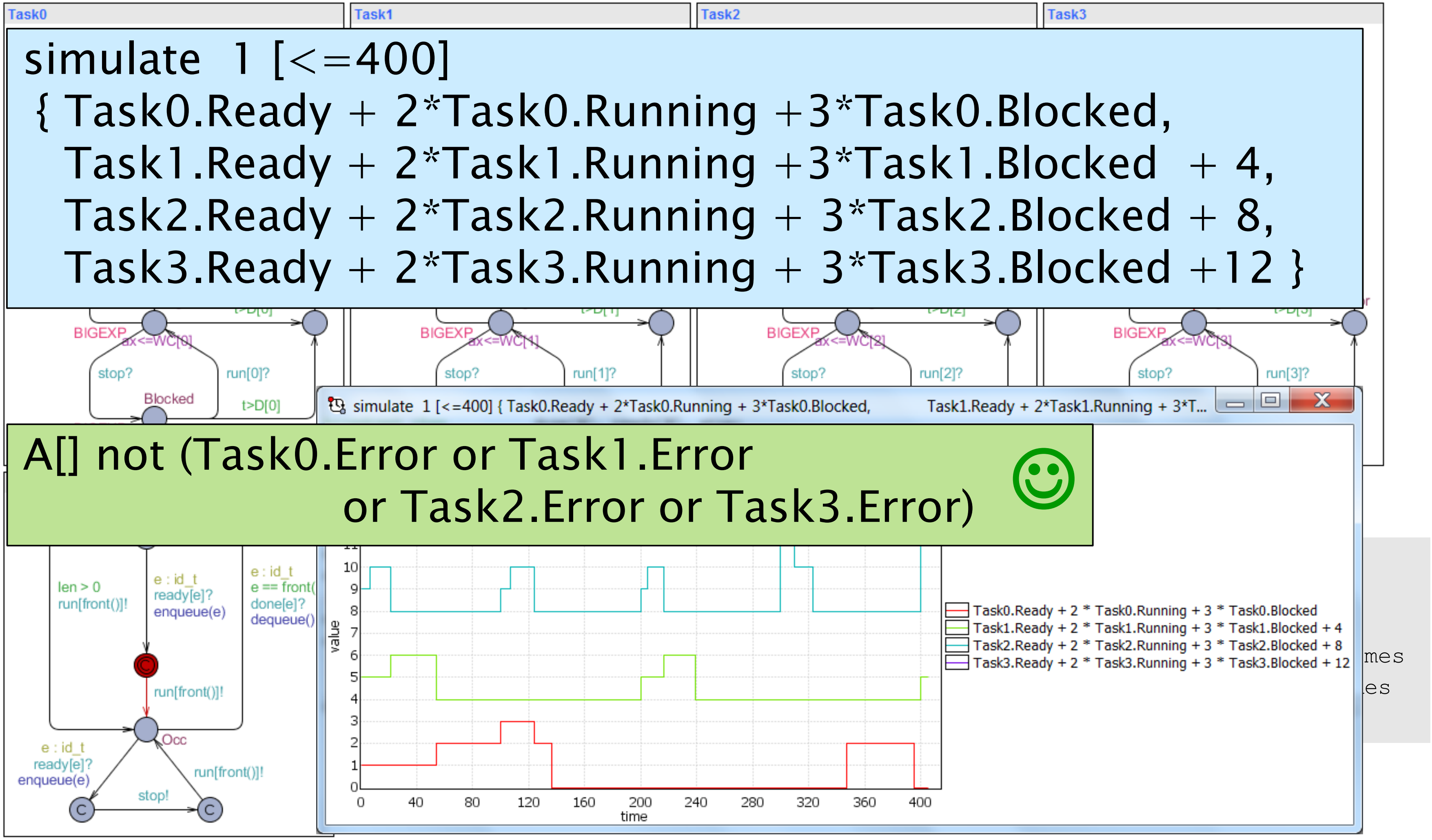
Modeling Queue





```
// Put an element at the end of the queue
void enqueue(id_t element)
{
int tmp=0;
list[len++] = element;
if (len>0)
{
        int i=len-1;
        while (i>1 && P[list[i]]>P[list[i-1]])
        {
                tmp = list[i-1];
                list[i-1] = list[i];
                list[i] = tmp;
                i--;
        }
}
}

// Remove the front element of the queue
void dequeue()
{ ......
```

Schedulability Analysis



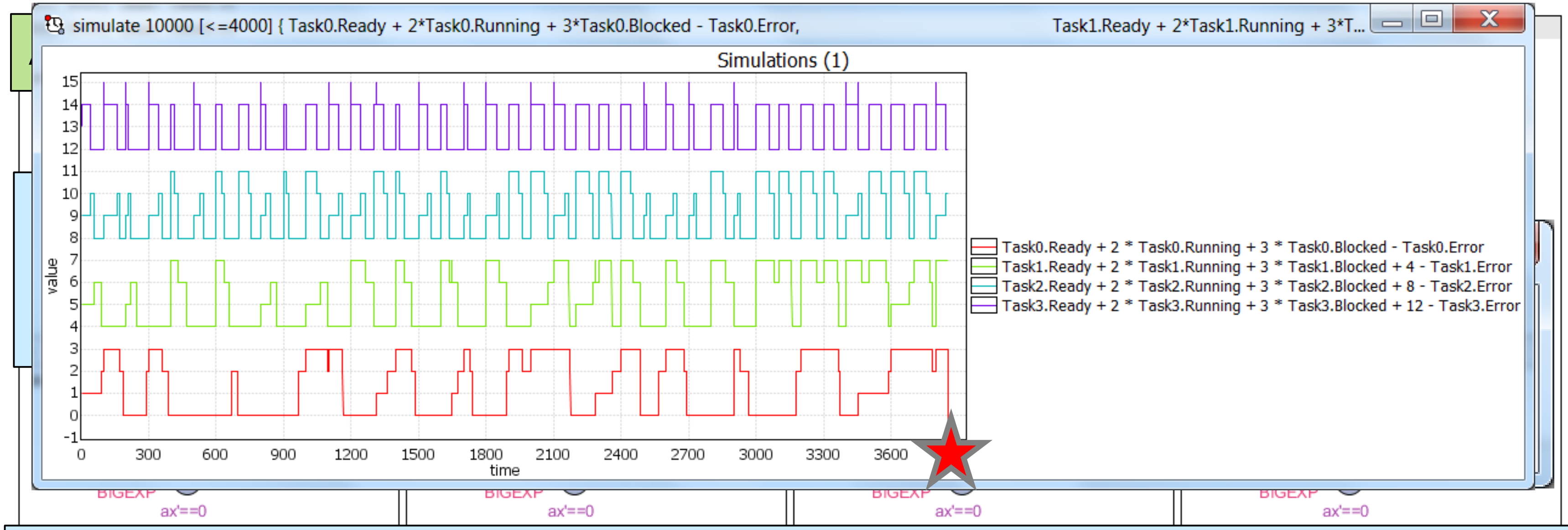
```
simulate  1 [<=400]
 { Task0.Ready + 2*Task0.Running +3*Task0.Blocked,
   Task1.Ready + 2*Task1.Running +3*Task1.Blocked  + 4,
   Task2.Ready + 2*Task2.Running + 3*Task2.Blocked + 8,
   Task3.Ready + 2*Task3.Running + 3*Task3.Blocked +12 }
```

```
A[] not (Task0.Error or Task1.Error
                   or Task2.Error or Task3.Error)
```

☺

Task0
Task1
Task2
Task3
BIGEXP
x<=WC[0]
x<=WC[1]
x<=WC[2]
x<=WC[3]
t>D[0]
t>D[1]
t>D[2]
t>D[3]
stop?
run[0]?
run[1]?
run[2]?
run[3]?
Blocked
simulate 1 [<=400] { Task0.Ready + 2*Task0.Running + 3*Task0.Blocked, Task1.Ready + 2*Task1.Running + 3*T...
len > 0
run[front()]!
e : id_t
ready[e]?
enqueue(e)
e : id_t
e == front()
done[e]?
dequeue()
run[front()]!
Occ
e : id_t
ready[e]?
enqueue(e)
stop!
run[front()]!
C
C
value
time
0
1
2
3
4
5
6
7
8
9
10
0
40
80
120
160
200
240
280
320
360
400
Task0.Ready + 2 * Task0.Running + 3 * Task0.Blocked
Task1.Ready + 2 * Task1.Running + 3 * Task1.Blocked + 4
Task2.Ready + 2 * Task2.Running + 3 * Task2.Blocked + 8
Task3.Ready + 2 * Task3.Running + 3 * Task3.Blocked + 12
mes
es

Schedulability Analysis



simulate 10000 [<=4000] { Task0.Ready + 2*Task0.Running + 3*Task0.Blocked - Task0.Error, Task1.Ready + 2*Task1.Running + 3*T...

Simulations (1)
value
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
0
300
600
900
1200
1500
1800
2100
2400
2700
3000
3300
3600
time
🟥 Task0.Ready + 2 * Task0.Running + 3 * Task0.Blocked - Task0.Error
🟩 Task1.Ready + 2 * Task1.Running + 3 * Task1.Blocked + 4 - Task1.Error
🟦 Task2.Ready + 2 * Task2.Running + 3 * Task2.Blocked + 8 - Task2.Error
🟪 Task3.Ready + 2 * Task3.Running + 3 * Task3.Blocked + 12 - Task3.Error
BIGEXP
ax'==0
BIGEXP
ax'==0
BIGEXP
ax'==0
BIGEXP
ax'==0

simulate 10000 [<=400]

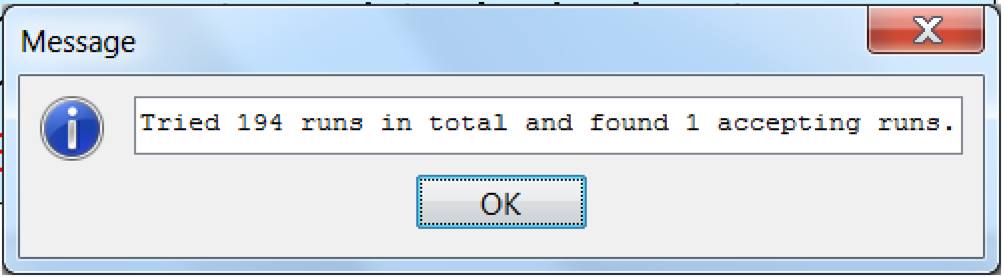
{ Task0.Ready + 2*Task0.Running +3*Task0.Blocked,

Task1.Ready + 2*Task1.Running +3*Task1.Blocked + 4,

Task2.Ready + 2*Task2.Rur

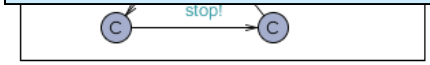
Task3.Ready + 2*Task3.Rur

: 1 : (Task0.Error or Task1.E

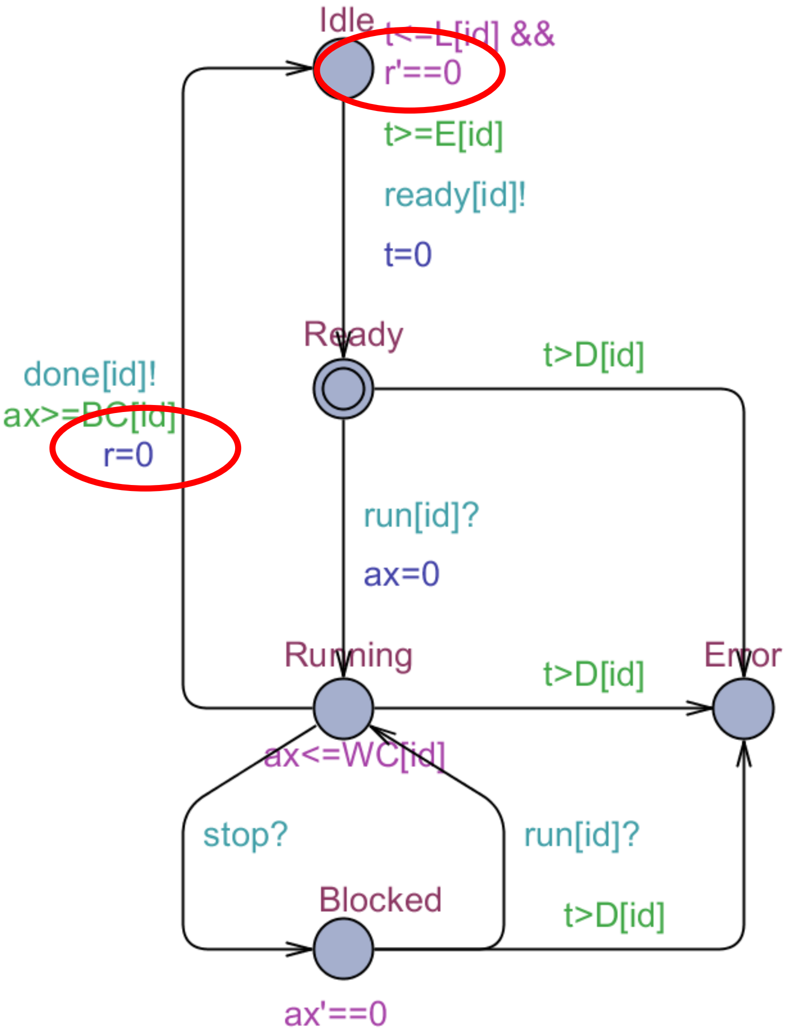


Message

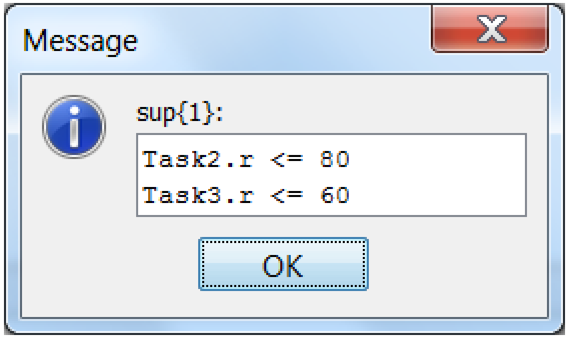
```
Tried 194 runs in total and found 1 accepting runs.
```



Performance Analysis



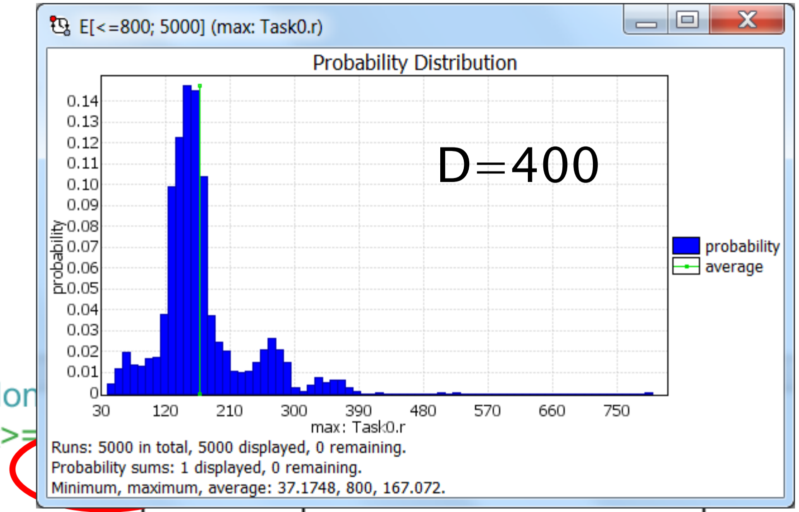
sup : Task2.r, Task3.r



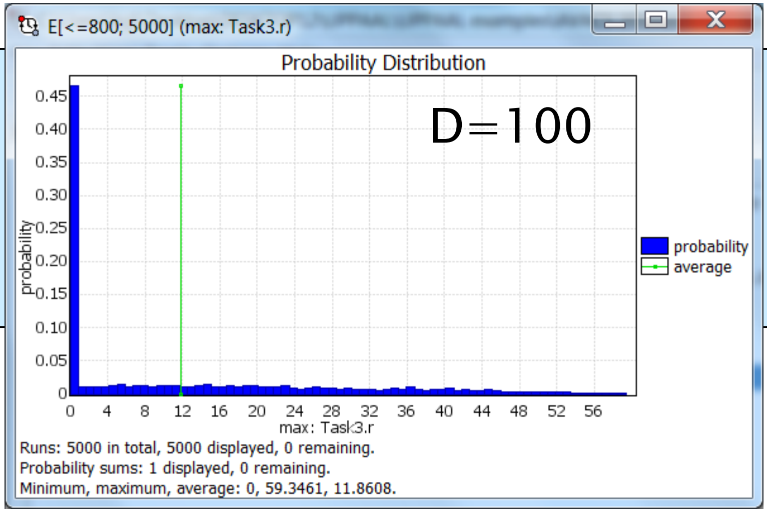
sup{1}:

```
Task2.r <= 80
Task3.r <= 60
```

Performance Analysis



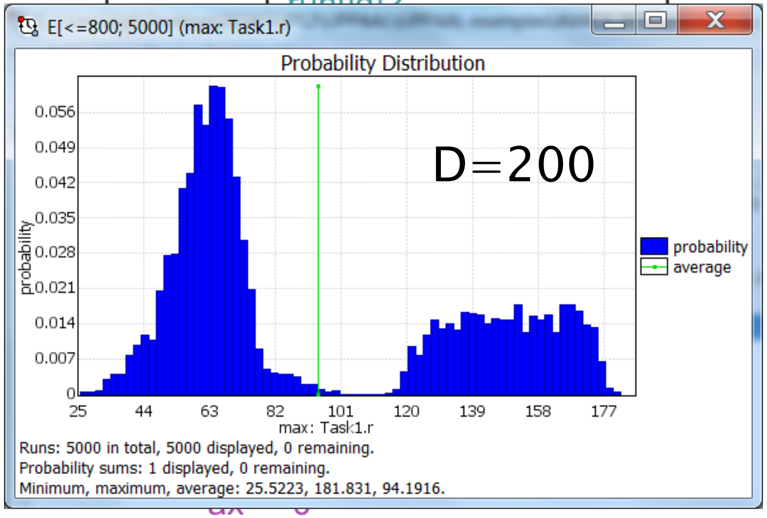
Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 37.1748, 800, 167.072.



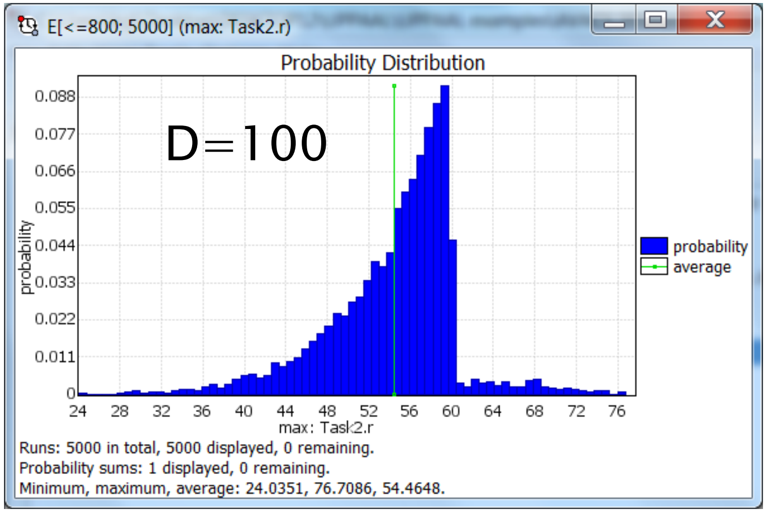
Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 0, 59.3461, 11.8608.

E[
E[
E[
E[

don
ax>=

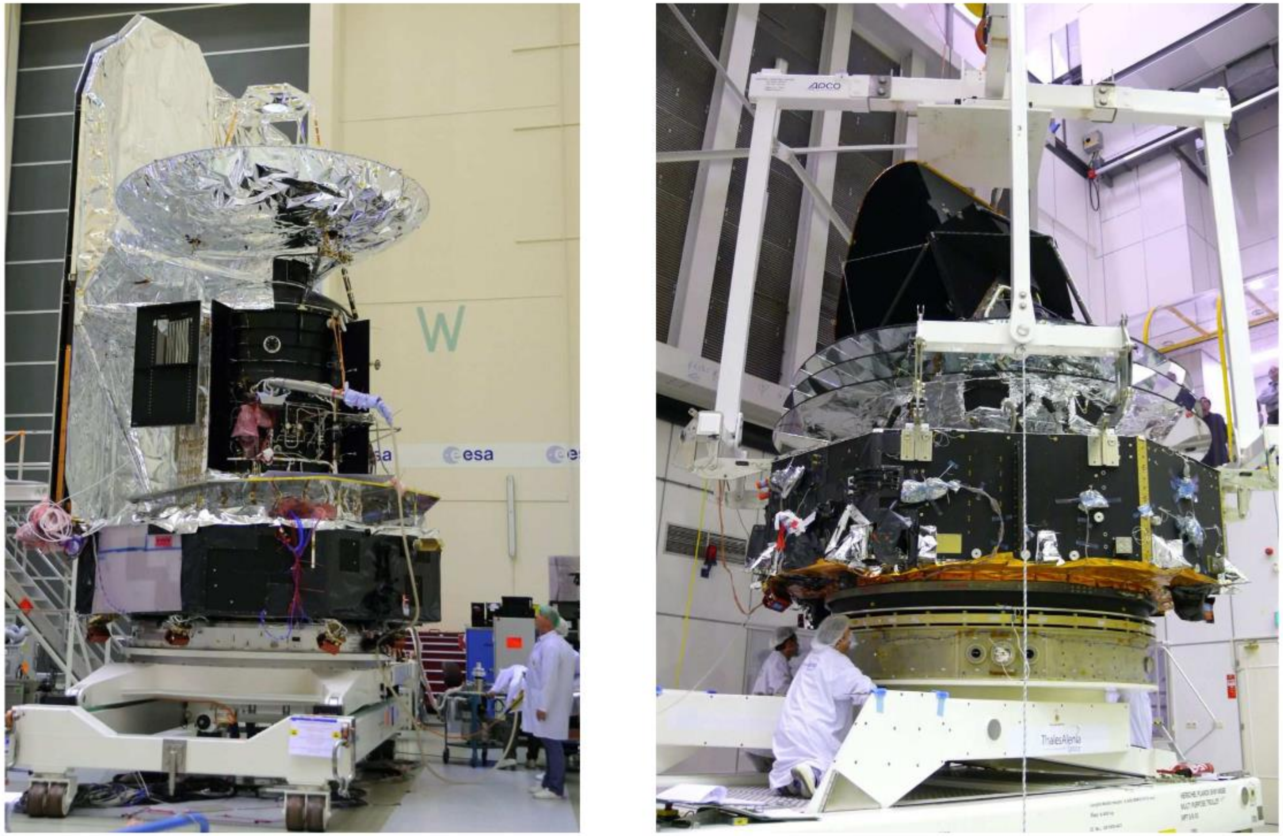


Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 25.5223, 181.831, 94.1916.



Runs: 5000 in total, 5000 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 24.0351, 76.7086, 54.4648.

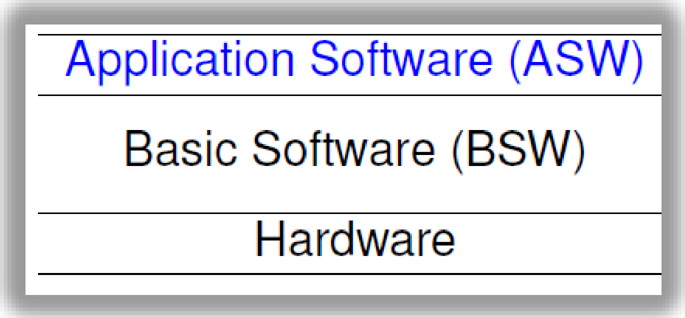
Herschel–Planck Scientific Mission at ESA



Attitude and Orbit Control Software
TERMA A/S Steen Ulrik Palm, Jan Storbank Pedersen, Poul Hougaard

Herschel & Planck Satelites

- Application software (ASW)
  - built and tested by Terma:
  - does attitude and orbit control, tele-commanding, fault detection isolation and recovery.
- Basic software (BSW)
  - low level communication and scheduling periodic events.
- Real-time operating system (RTEMS)
  - Priority Ceiling for ASW,
  - Priority Inheritance for BSW
- Hardware
  - single processor, a few buses, sensors and act

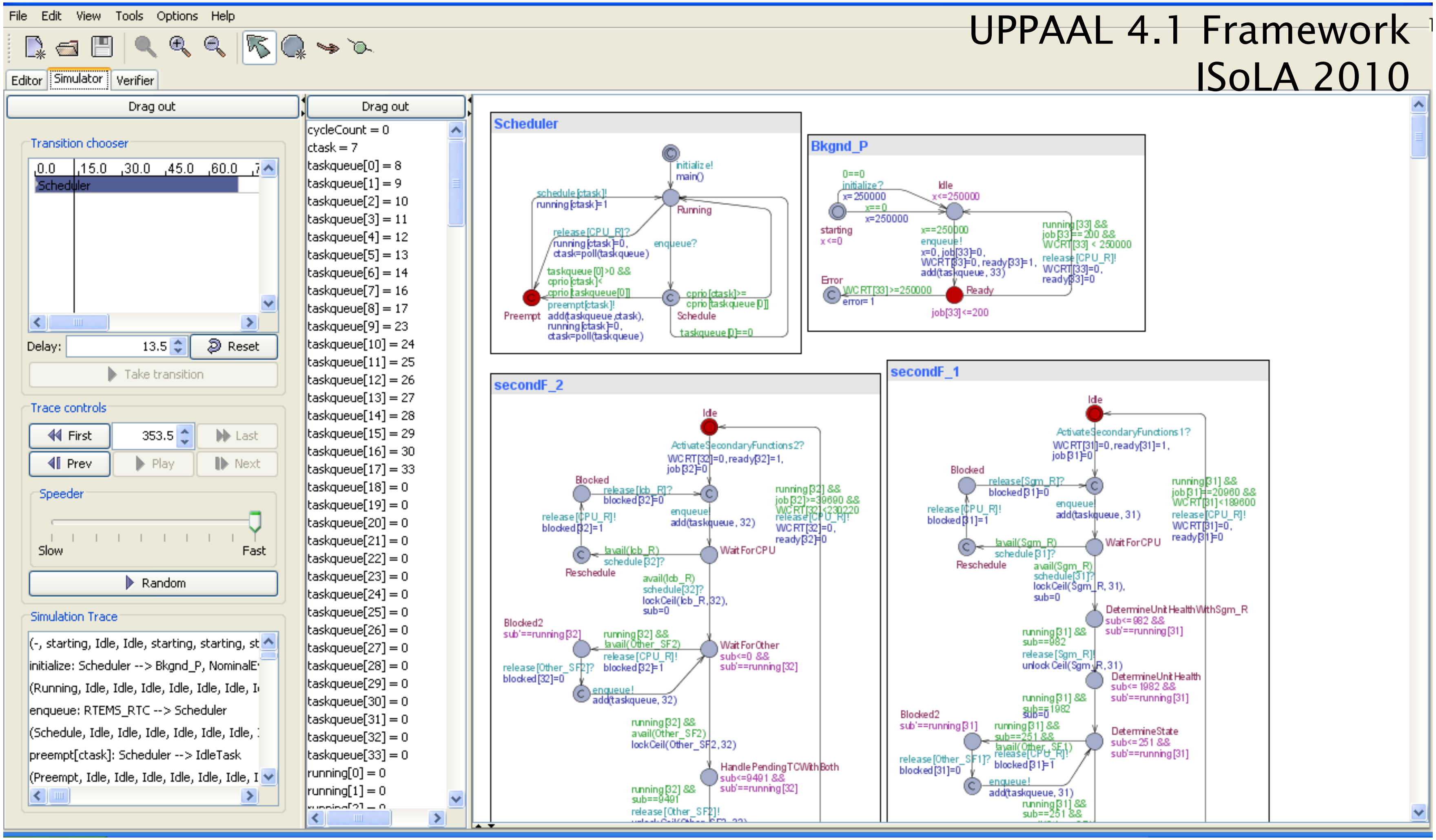


Requirements:

Software tasks should be schedulable.
CPU utilization should not exceed 50% load

Modeling in UPPAAL

TERMA®



UPPAAL 4.1 Framework
ISoLA 2010

(-, starting, Idle, Idle, starting, starting, st
initialize: Scheduler --> Bkgnd_P, NominalE
(Running, Idle, Idle, Idle, Idle, Idle, Idle, I
enqueue: RTEMS_RTC --> Scheduler
(Schedule, Idle, Idle, Idle, Idle, Idle, Idle, I
preempt[ctask]: Scheduler --> IdleTask
(Preempt, Idle, Idle, Idle, Idle, Idle, Idle, I

```
cycleCount = 0
ctask = 7
taskqueue[0] = 8
taskqueue[1] = 9
taskqueue[2] = 10
taskqueue[3] = 11
taskqueue[4] = 12
taskqueue[5] = 13
taskqueue[6] = 14
taskqueue[7] = 16
taskqueue[8] = 17
taskqueue[9] = 23
taskqueue[10] = 24
taskqueue[11] = 25
taskqueue[12] = 26
taskqueue[13] = 27
taskqueue[14] = 28
taskqueue[15] = 29
taskqueue[16] = 30
taskqueue[17] = 33
taskqueue[18] = 0
taskqueue[19] = 0
taskqueue[20] = 0
taskqueue[21] = 0
taskqueue[22] = 0
taskqueue[23] = 0
taskqueue[24] = 0
taskqueue[25] = 0
taskqueue[26] = 0
taskqueue[27] = 0
taskqueue[28] = 0
taskqueue[29] = 0
taskqueue[30] = 0
taskqueue[31] = 0
taskqueue[32] = 0
taskqueue[33] = 0
running[0] = 0
running[1] = 0
```

Gantt Chart 1. cycle

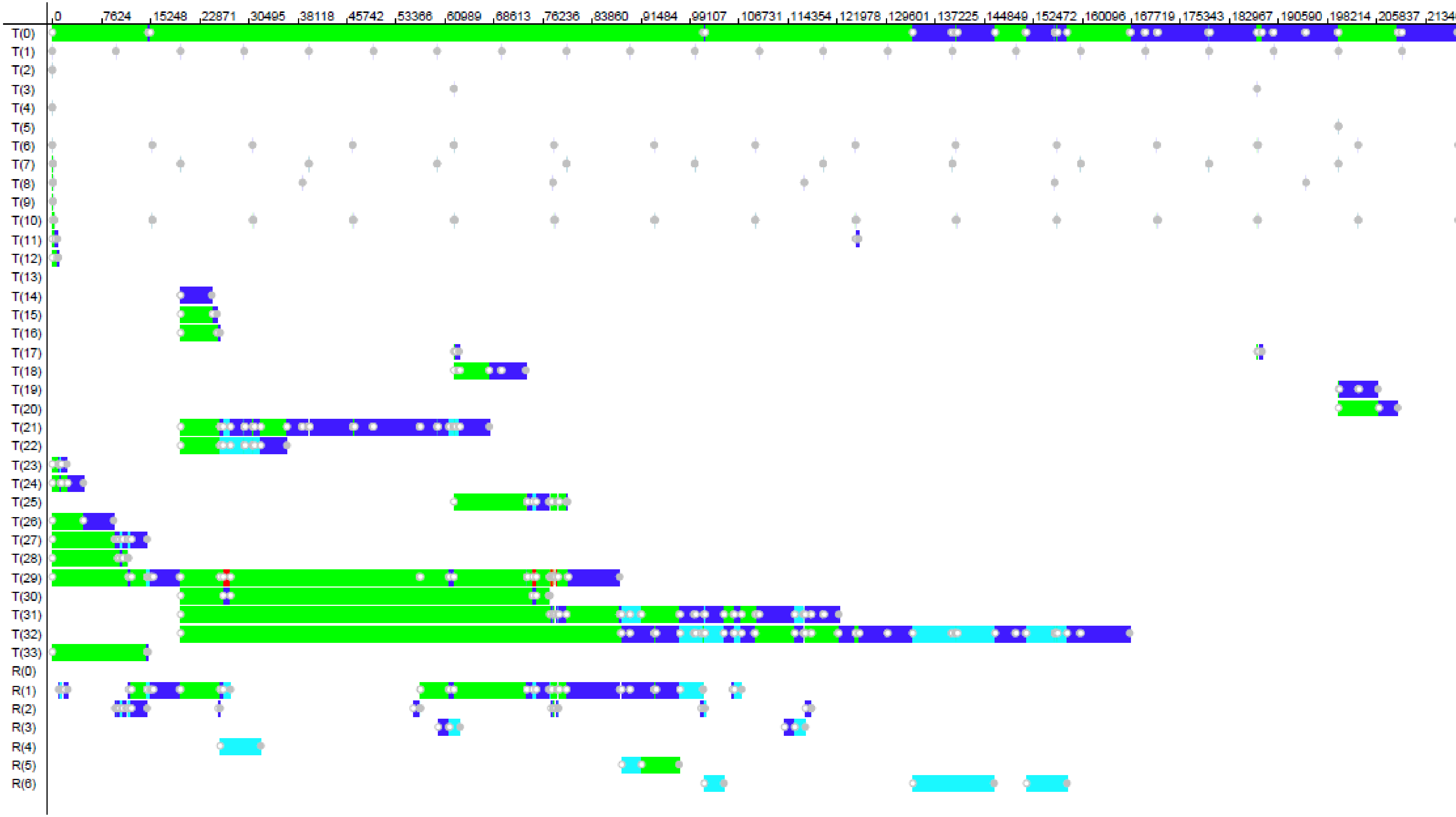
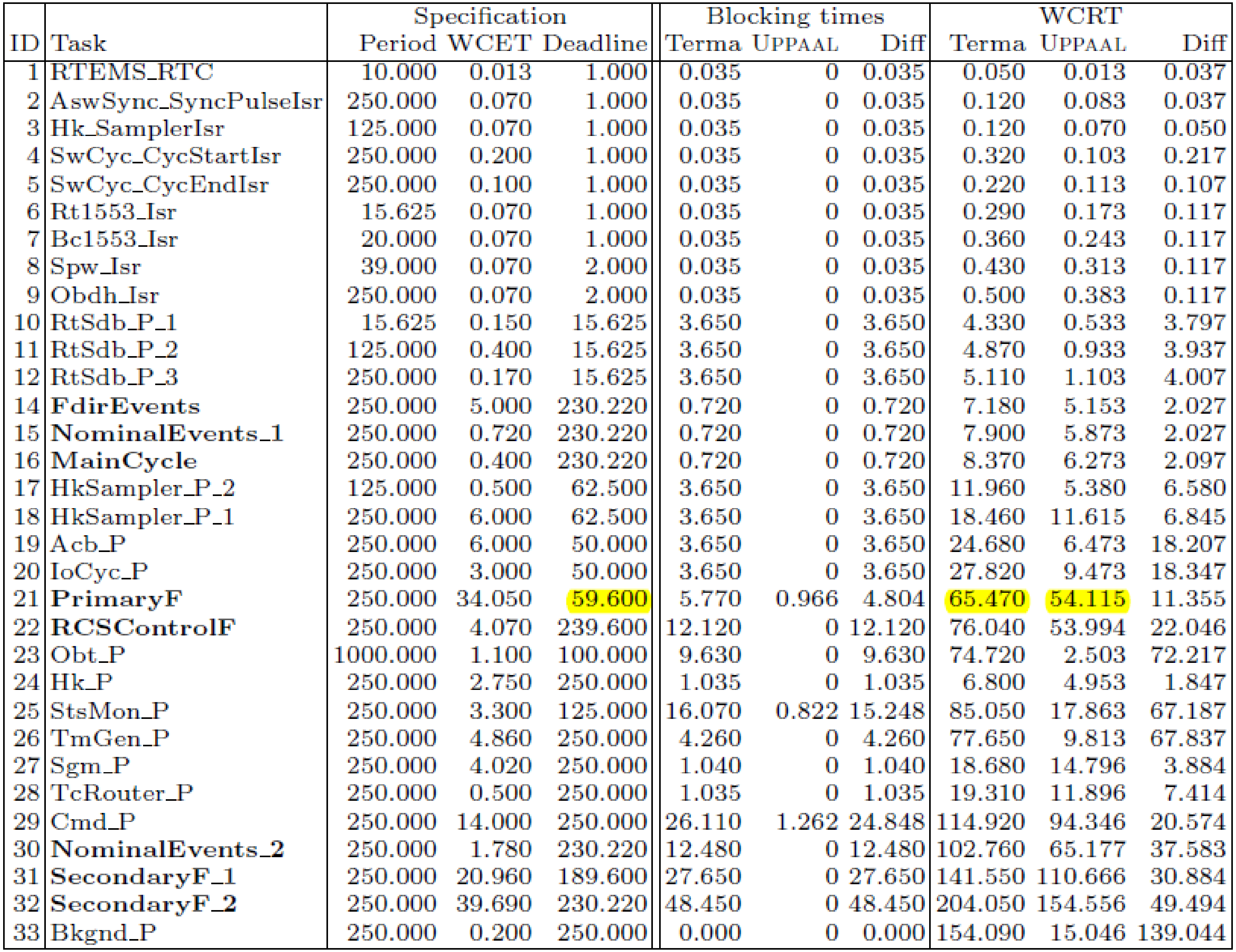


Fig. 11. Gantt chart of a schedule from the first cycle: green means ready, blue means running, cyan means suspended, red means blocked. R stand for resources: CPU_R=0, Icb_R=1, Sgm_R=2, PmReq_R=3, Other_RCS=4, Other_SF1=5, Other_SF2=6.

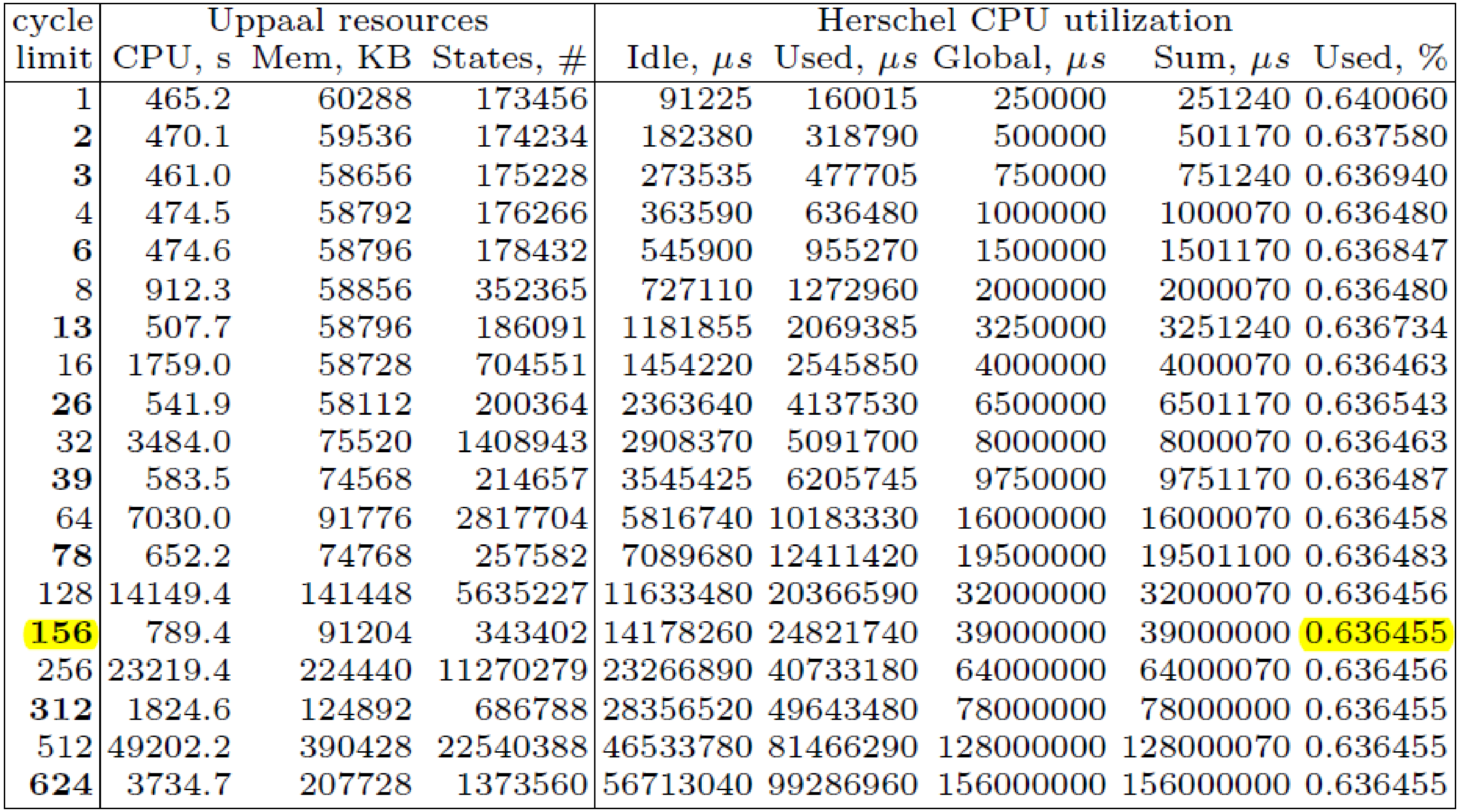
Blocking & WCRT



		Specification			Blocking times			WCRT		
ID	Task	Period	WCET	Deadline	Terma	Uppaal	Diff	Terma	Uppaal	Diff
1	RTEMS_RTC	10.000	0.013	1.000	0.035	0	0.035	0.050	0.013	0.037
2	AswSync_SyncPulseIsr	250.000	0.070	1.000	0.035	0	0.035	0.120	0.083	0.037
3	Hk_SamplerIsr	125.000	0.070	1.000	0.035	0	0.035	0.120	0.070	0.050
4	SwCyc_CycStartIsr	250.000	0.200	1.000	0.035	0	0.035	0.320	0.103	0.217
5	SwCyc_CycEndIsr	250.000	0.100	1.000	0.035	0	0.035	0.220	0.113	0.107
6	Rt1553_Isr	15.625	0.070	1.000	0.035	0	0.035	0.290	0.173	0.117
7	Bc1553_Isr	20.000	0.070	1.000	0.035	0	0.035	0.360	0.243	0.117
8	Spw_Isr	39.000	0.070	2.000	0.035	0	0.035	0.430	0.313	0.117
9	Obdh_Isr	250.000	0.070	2.000	0.035	0	0.035	0.500	0.383	0.117
10	RtSdb_P_1	15.625	0.150	15.625	3.650	0	3.650	4.330	0.533	3.797
11	RtSdb_P_2	125.000	0.400	15.625	3.650	0	3.650	4.870	0.933	3.937
12	RtSdb_P_3	250.000	0.170	15.625	3.650	0	3.650	5.110	1.103	4.007
14	**FdirEvents**	250.000	5.000	230.220	0.720	0	0.720	7.180	5.153	2.027
15	**NominalEvents_1**	250.000	0.720	230.220	0.720	0	0.720	7.900	5.873	2.027
16	**MainCycle**	250.000	0.400	230.220	0.720	0	0.720	8.370	6.273	2.097
17	HkSampler_P_2	125.000	0.500	62.500	3.650	0	3.650	11.960	5.380	6.580
18	HkSampler_P_1	250.000	6.000	62.500	3.650	0	3.650	18.460	11.615	6.845
19	Acb_P	250.000	6.000	50.000	3.650	0	3.650	24.680	6.473	18.207
20	IoCyc_P	250.000	3.000	50.000	3.650	0	3.650	27.820	9.473	18.347
21	**PrimaryF**	250.000	34.050	59.600	5.770	0.966	4.804	65.470	54.115	11.355
22	**RCSControlF**	250.000	4.070	239.600	12.120	0	12.120	76.040	53.994	22.046
23	Obt_P	1000.000	1.100	100.000	9.630	0	9.630	74.720	2.503	72.217
24	Hk_P	250.000	2.750	250.000	1.035	0	1.035	6.800	4.953	1.847
25	StsMon_P	250.000	3.300	125.000	16.070	0.822	15.248	85.050	17.863	67.187
26	TmGen_P	250.000	4.860	250.000	4.260	0	4.260	77.650	9.813	67.837
27	Sgm_P	250.000	4.020	250.000	1.040	0	1.040	18.680	14.796	3.884
28	TcRouter_P	250.000	0.500	250.000	1.035	0	1.035	19.310	11.896	7.414
29	Cmd_P	250.000	14.000	250.000	26.110	1.262	24.848	114.920	94.346	20.574
30	**NominalEvents_2**	250.000	1.780	230.220	12.480	0	12.480	102.760	65.177	37.583
31	**SecondaryF_1**	250.000	20.960	189.600	27.650	0	27.650	141.550	110.666	30.884
32	**SecondaryF_2**	250.000	39.690	230.220	48.450	0	48.450	204.050	154.556	49.494
33	Bkgnd_P	250.000	0.200	250.000	0.000	0	0.000	154.090	15.046	139.044

Marius Micusionis

Effort and Utilization

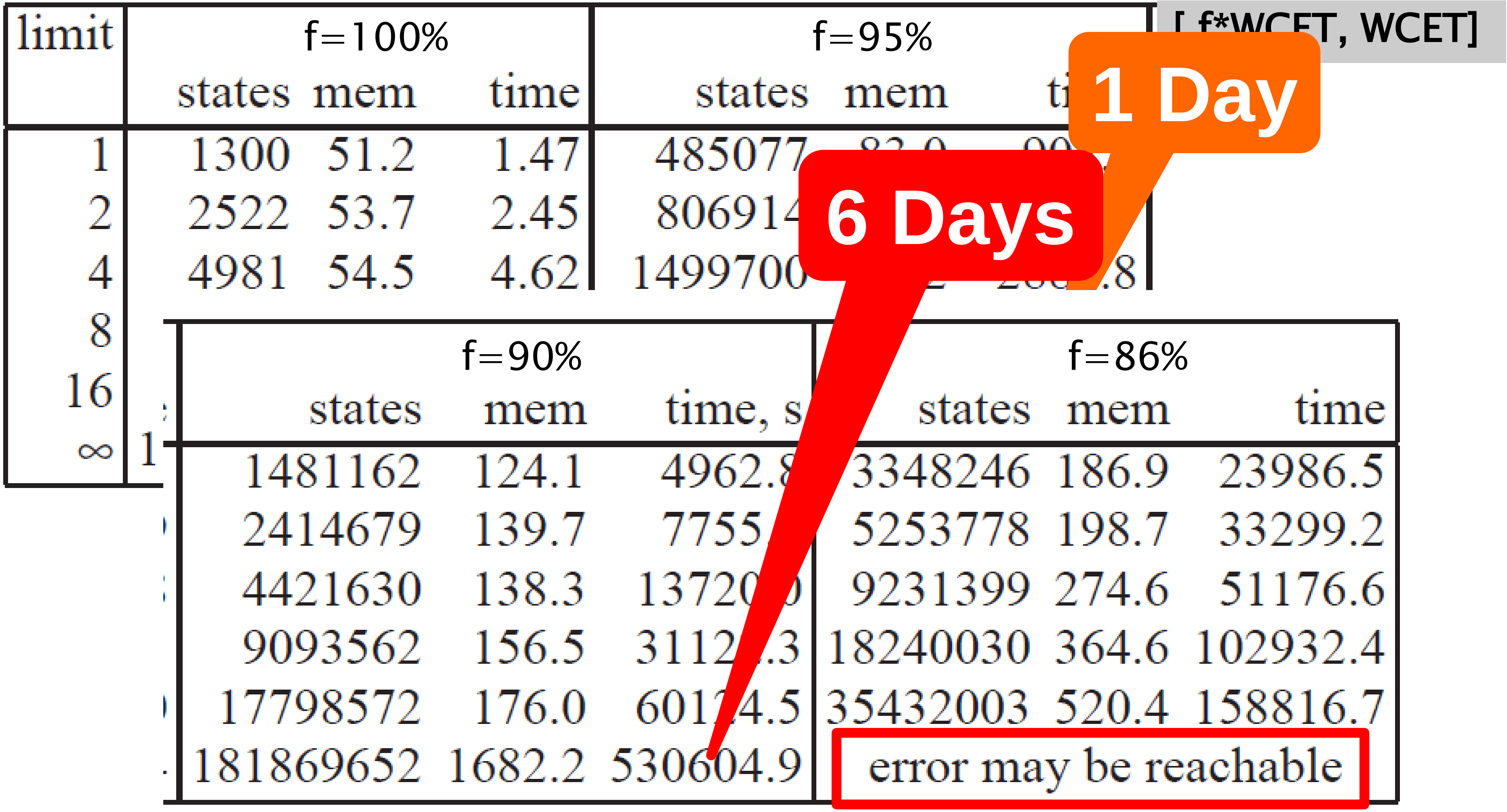


cycle	Uppaal resources			Herschel CPU utilization				
limit	CPU, s	Mem, KB	States, #	Idle, μs	Used, μs	Global, μs	Sum, μs	Used, %
1	465.2	60288	173456	91225	160015	250000	251240	0.640060
2	470.1	59536	174234	182380	318790	500000	501170	0.637580
3	461.0	58656	175228	273535	477705	750000	751240	0.636940
4	474.5	58792	176266	363590	636480	1000000	1000070	0.636480
6	474.6	58796	178432	545900	955270	1500000	1501170	0.636847
8	912.3	58856	352365	727110	1272960	2000000	2000070	0.636480
13	507.7	58796	186091	1181855	2069385	3250000	3251240	0.636734
16	1759.0	58728	704551	1454220	2545850	4000000	4000070	0.636463
26	541.9	58112	200364	2363640	4137530	6500000	6501170	0.636543
32	3484.0	75520	1408943	2908370	5091700	8000000	8000070	0.636463
39	583.5	74568	214657	3545425	6205745	9750000	9751170	0.636487
64	7030.0	91776	2817704	5816740	10183330	16000000	16000070	0.636458
78	652.2	74768	257582	7089680	12411420	19500000	19501100	0.636483
128	14149.4	141448	5635227	11633480	20366590	32000000	32000070	0.636456
156	789.4	91204	343402	14178260	24821740	39000000	39000000	0.636455
256	23219.4	224440	11270279	23266890	40733180	64000000	64000070	0.636456
312	1824.6	124892	686788	28356520	49643480	78000000	78000000	0.636455
512	49202.2	390428	22540388	46533780	81466290	128000000	128000070	0.636455
624	3734.7	207728	1373560	56713040	99286960	156000000	156000000	0.636455

Marius Micusionis

TERMA Case Follow-Up

ISOLA 2012



[f*WCET, WCET]

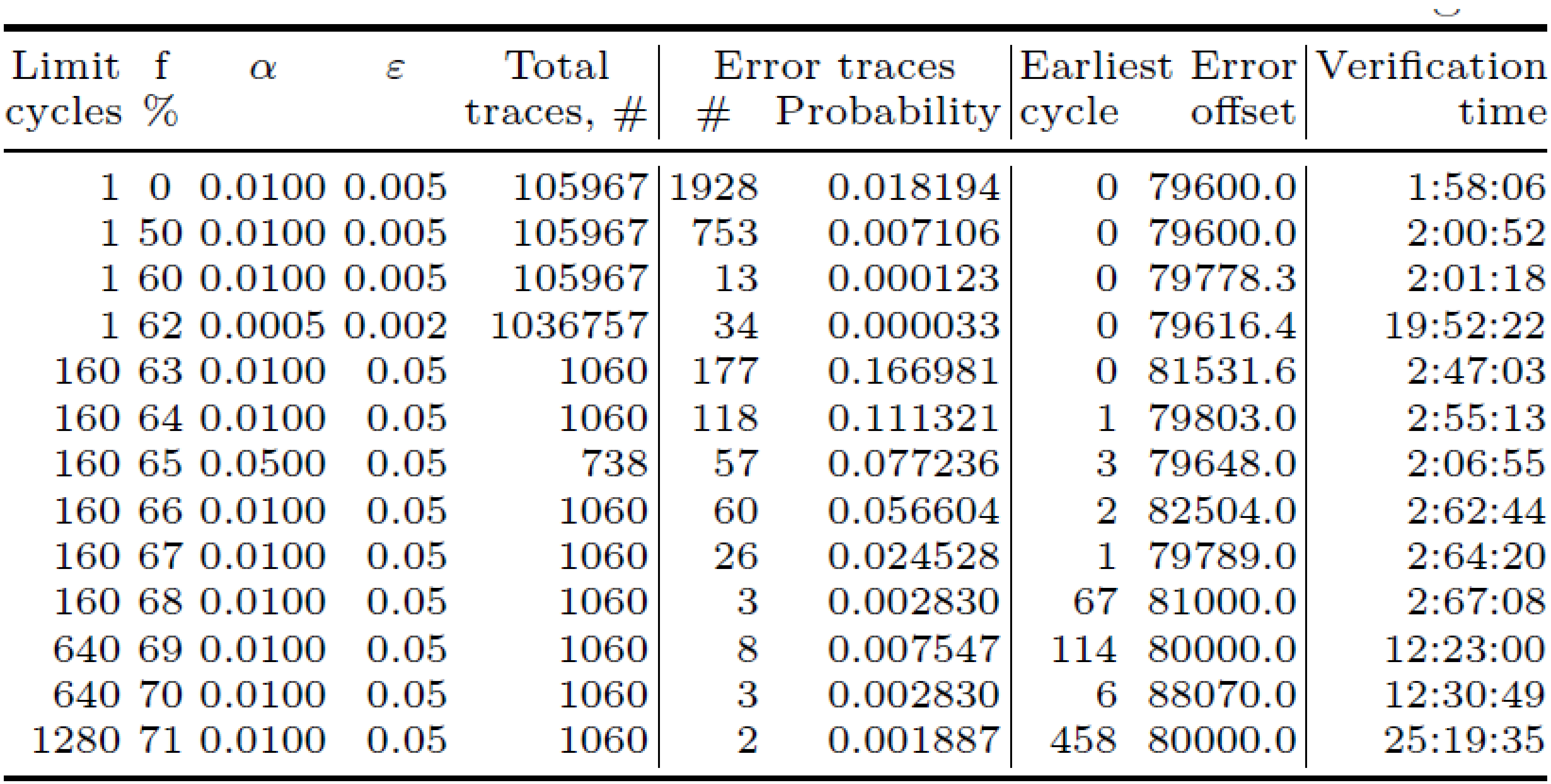
limit	f=100%			f=95%		
	states	mem	time	states	mem	ti[illegible]
1	1300	51.2	1.47	485077	[illegible]	[illegible]
2	2522	53.7	2.45	806914	[illegible]	[illegible]
4	4981	54.5	4.62	1499700	[illegible]	[illegible].8
8						
16						
∞						

f=90%			f=86%		
states	mem	time, s	states	mem	time
1481162	124.1	4962.8	3348246	186.9	23986.5
2414679	139.7	7755.[illegible]	5253778	198.7	33299.2
4421630	138.3	13720.0	9231399	274.6	51176.6
9093562	156.5	3112[illegible].3	18240030	364.6	102932.4
17798572	176.0	601[illegible]4.5	35432003	520.4	158816.7
181869652	1682.2	530604.9	error may be reachable		

1 Day

6 Days

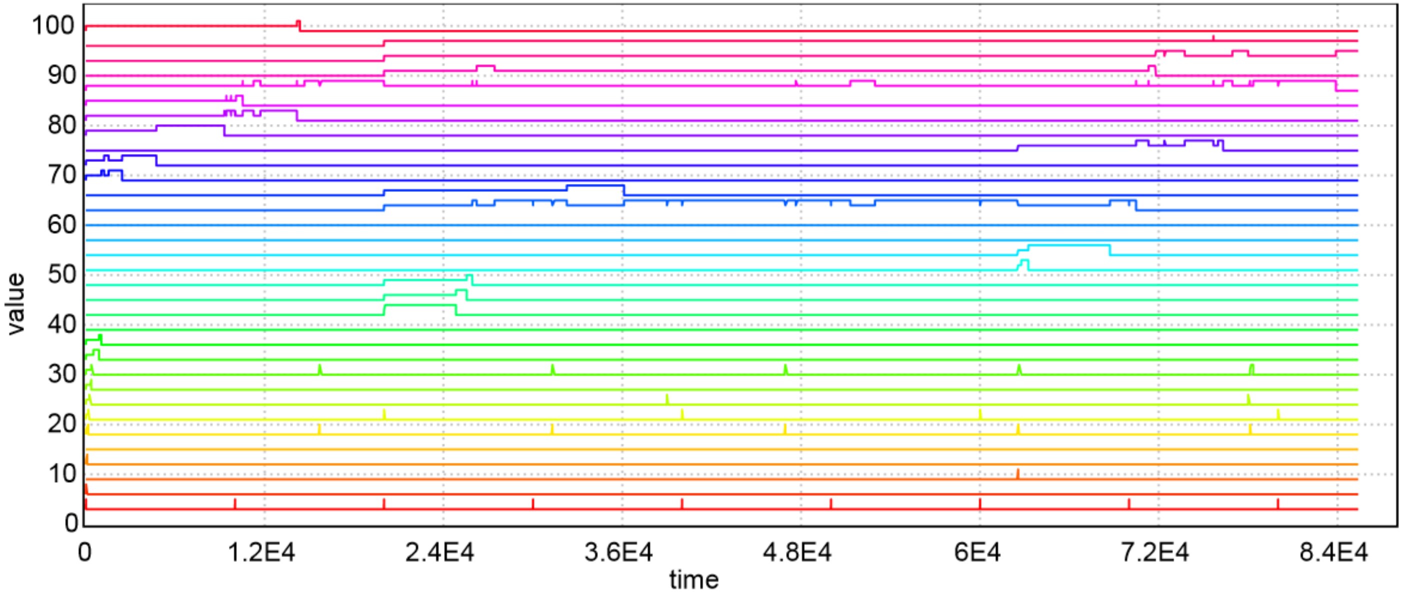
TERMA Case – Statistical MC



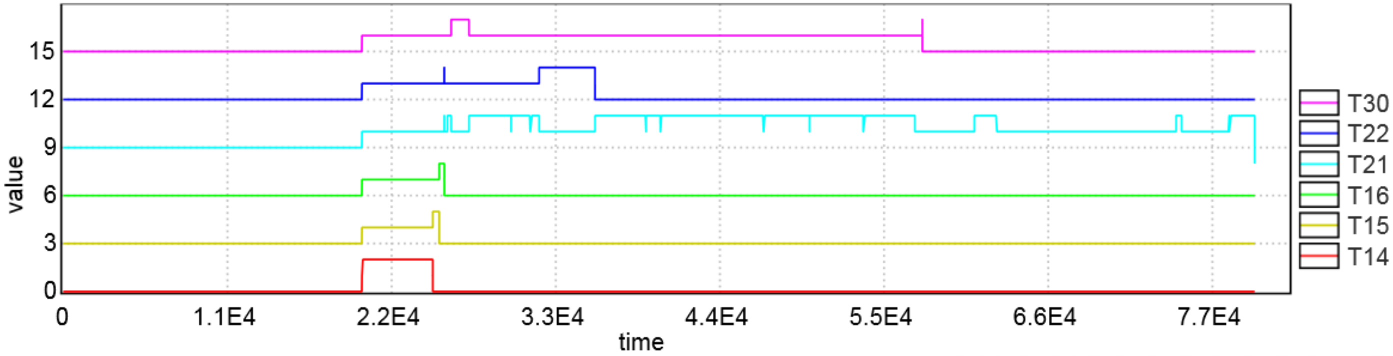
Limit	f	α	ε	Total	Error traces		Earliest Error		Verification
cycles	%			traces, #	#	Probability	cycle	offset	time
1	0	0.0100	0.005	105967	1928	0.018194	0	79600.0	1:58:06
1	50	0.0100	0.005	105967	753	0.007106	0	79600.0	2:00:52
1	60	0.0100	0.005	105967	13	0.000123	0	79778.3	2:01:18
1	62	0.0005	0.002	1036757	34	0.000033	0	79616.4	19:52:22
160	63	0.0100	0.05	1060	177	0.166981	0	81531.6	2:47:03
160	64	0.0100	0.05	1060	118	0.111321	1	79803.0	2:55:13
160	65	0.0500	0.05	738	57	0.077236	3	79648.0	2:06:55
160	66	0.0100	0.05	1060	60	0.056604	2	82504.0	2:62:44
160	67	0.0100	0.05	1060	26	0.024528	1	79789.0	2:64:20
160	68	0.0100	0.05	1060	3	0.002830	67	81000.0	2:67:08
640	69	0.0100	0.05	1060	8	0.007547	114	80000.0	12:23:00
640	70	0.0100	0.05	1060	3	0.002830	6	88070.0	12:30:49
1280	71	0.0100	0.05	1060	2	0.001887	458	80000.0	25:19:35

TERMA Case – Conclusion

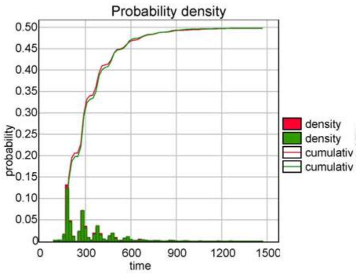
Herschel simulation run with $f = 90\%$:

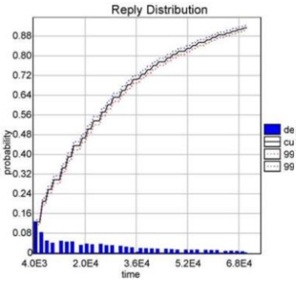


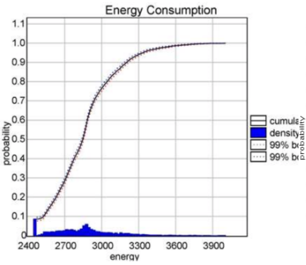
Herschel deadline violation with $f = 50\%$:

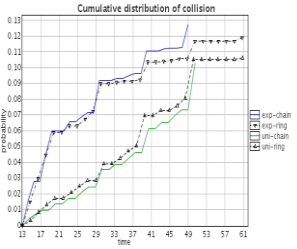


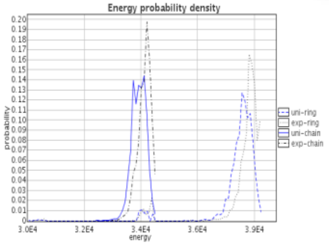
Statistical Model Checking
of Stochastic Hybrid Systems

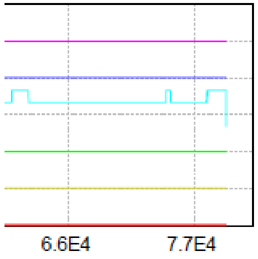










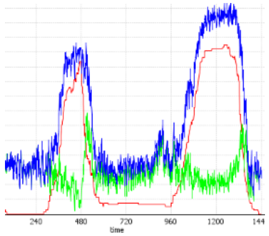


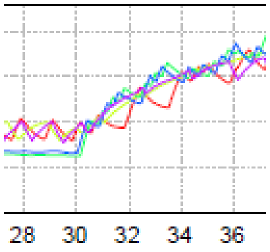
FIREWIRE

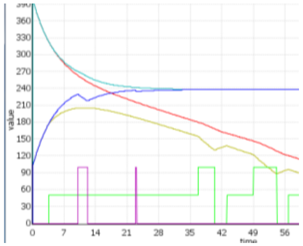
BLUETOOTH

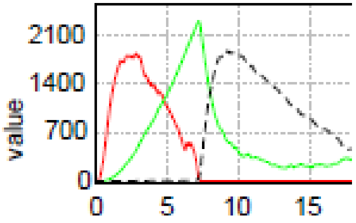
10 node LMAC

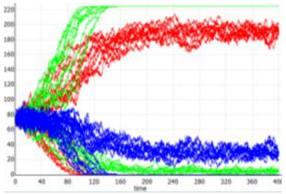
Schedulability Analysis for Mix Cr Sys











Smart Grid Demand / Response

Energy Aware Buildings

Battery Scheduling

Genetic Oscilator (HBS)

Cell Cycle Swithch

Overview

- Timed Automata / UPPAAL
  - Verification

Train Gate

- Stochastic Priced Timed Automata / UPPAAL SMC
  - Performance Evaluation
  - SMC in a Nutshell
  - Stochastic Hybrid Automata

Train Gate
Schedulability Analysis

- Timed Games / UPPAAL TIGA
  - Controller Syntesis

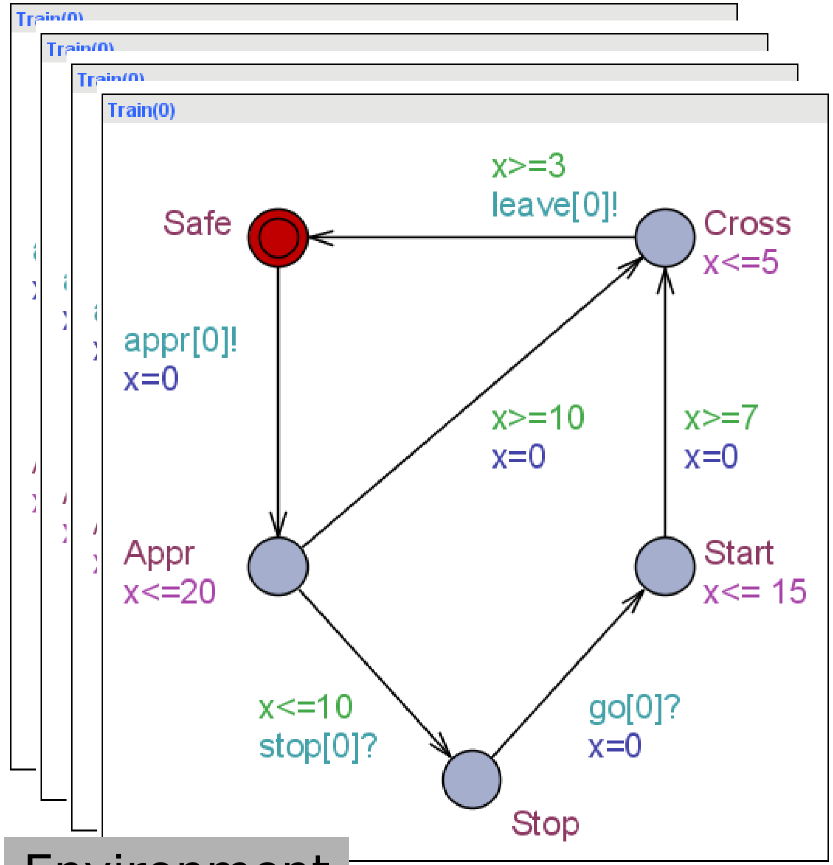
Train Gate

- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synthesis

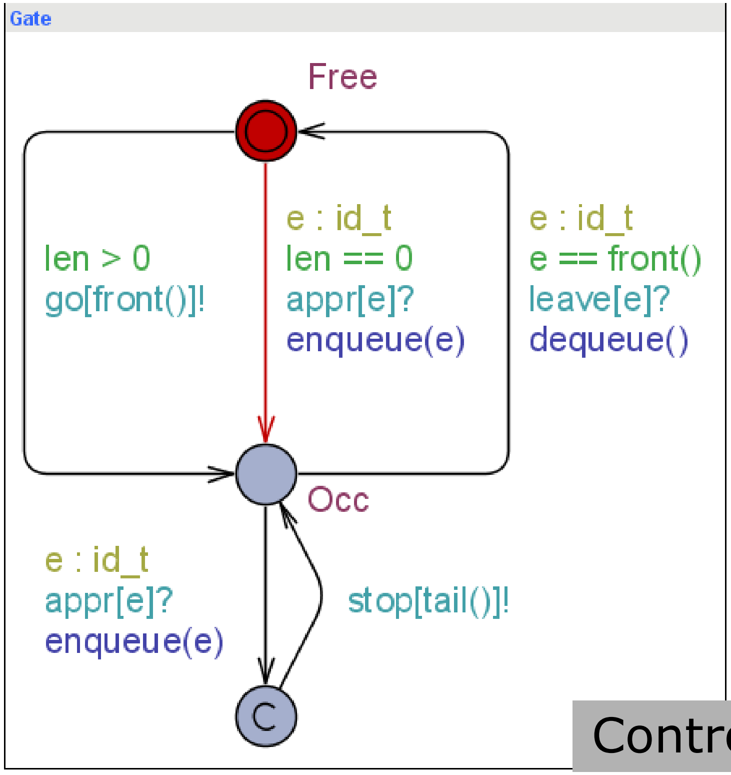
Train Gate
Floor Heating
Adaptive Cruise Control

- Conclusion

Model Checking (ex Train Gate)



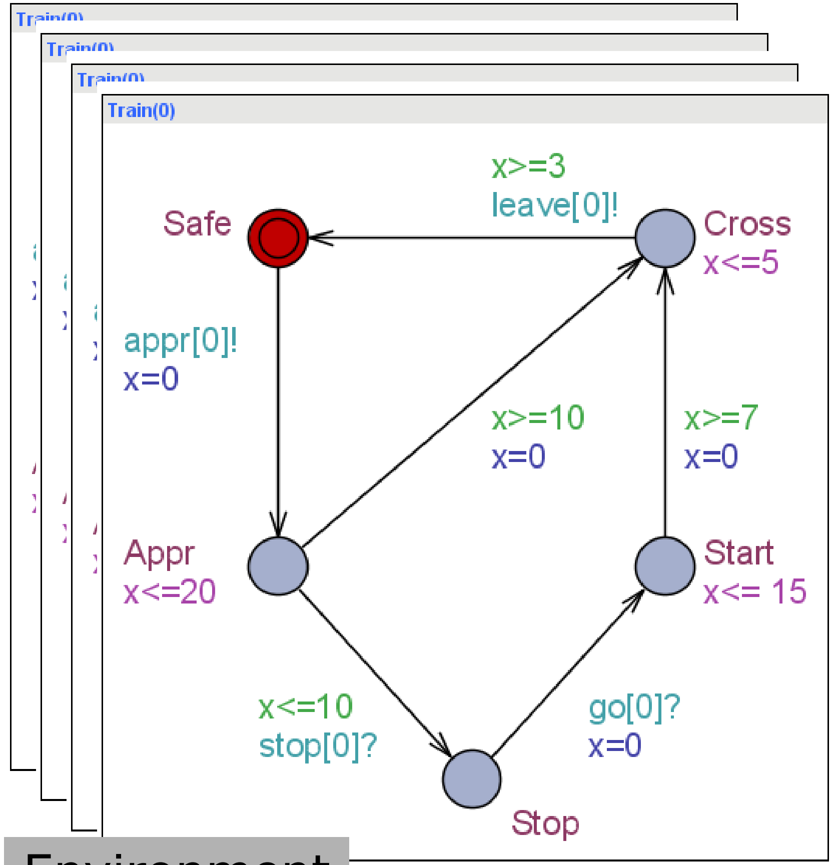
Environment



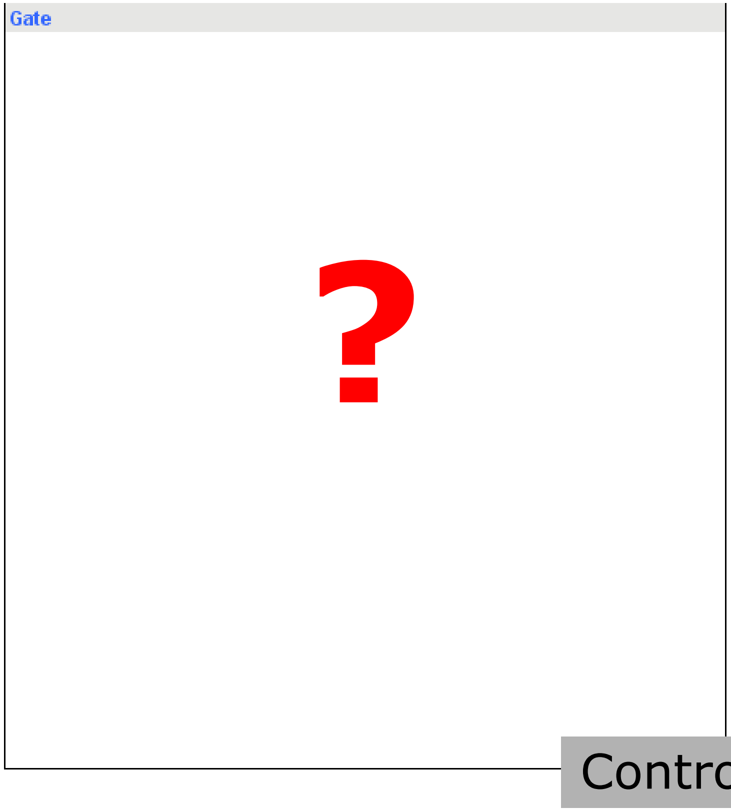
Controller

ϕ: Never two trains at the crossing at the same time

Synthesis (ex Train Gate)



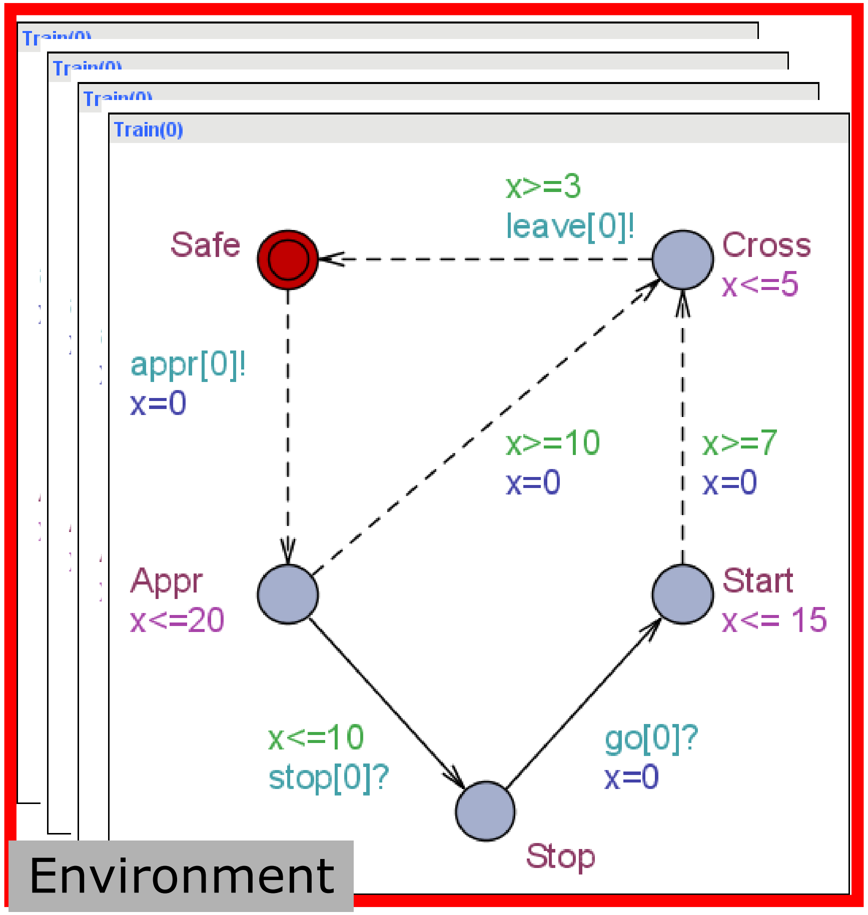
Environment

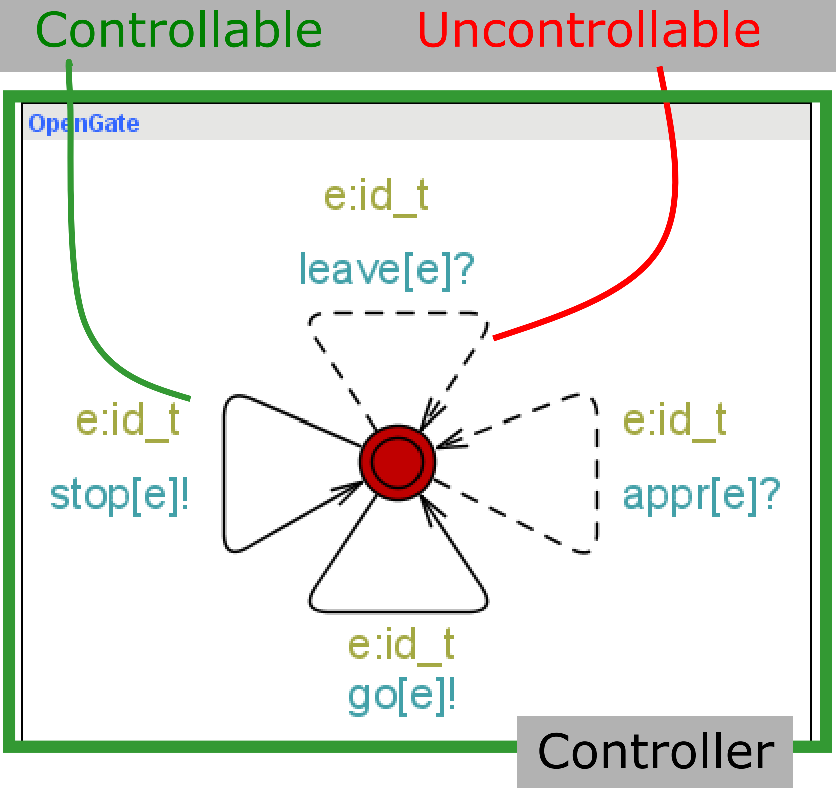


Controller

ϕ: Never two trains at the crossing at the same time

Timed Games



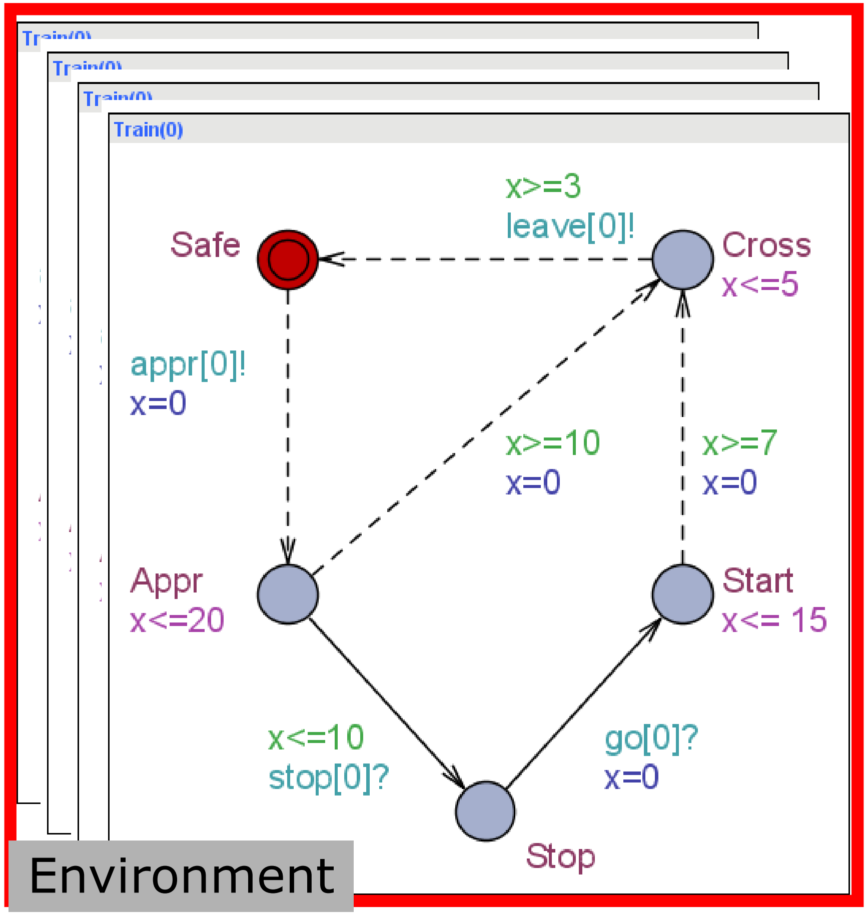


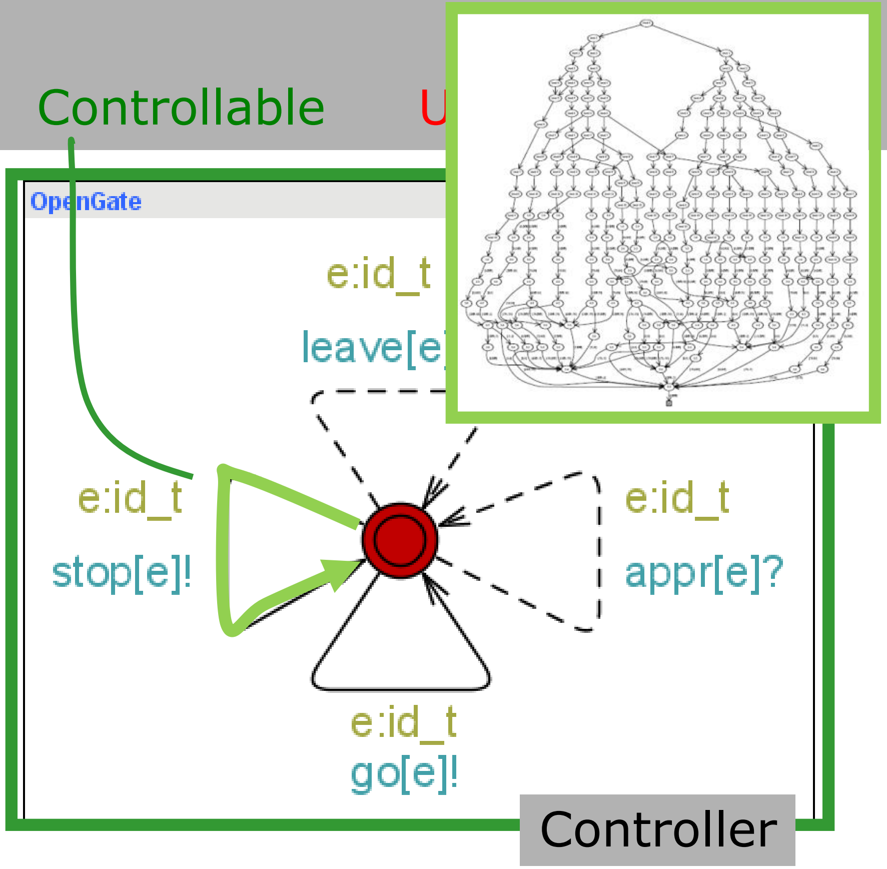
Synthesize strategy for controllable actions st behaviour satisfies ϕ

ϕ: Never two trains at the crossing at the same time

DEMO

Timed Games





Find strategy for controllable actions st behaviour satisfies ϕ

ϕ: Never two trains at the crossing at the same time

Overview

- Timed Automata / UPPAAL
  - Verification

Train Gate

- Stochastic Priced Timed Automata / UPPAAL SMC
  - Performance Evaluation
  - SMC in a Nutshell
  - Stochastic Hybrid Automata

Train Gate
Schedulability Analysis

- Timed Games / UPPAAL TIGA
  - Controller Syntesis

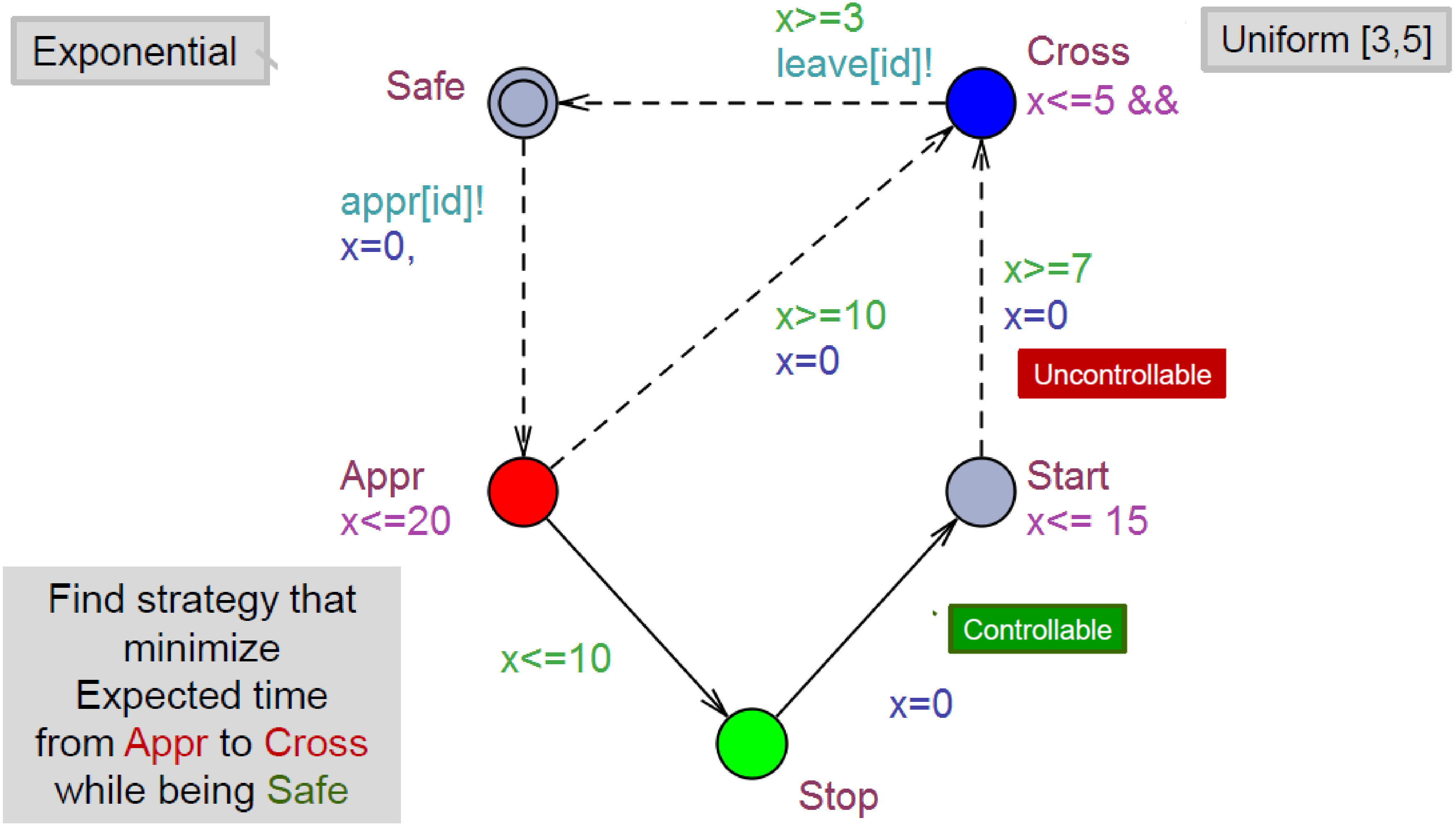
Train Gate

- Stochastic Priced Timed Games / UPPAAL STRATEGO
  - Optimal & Safe Synthesis

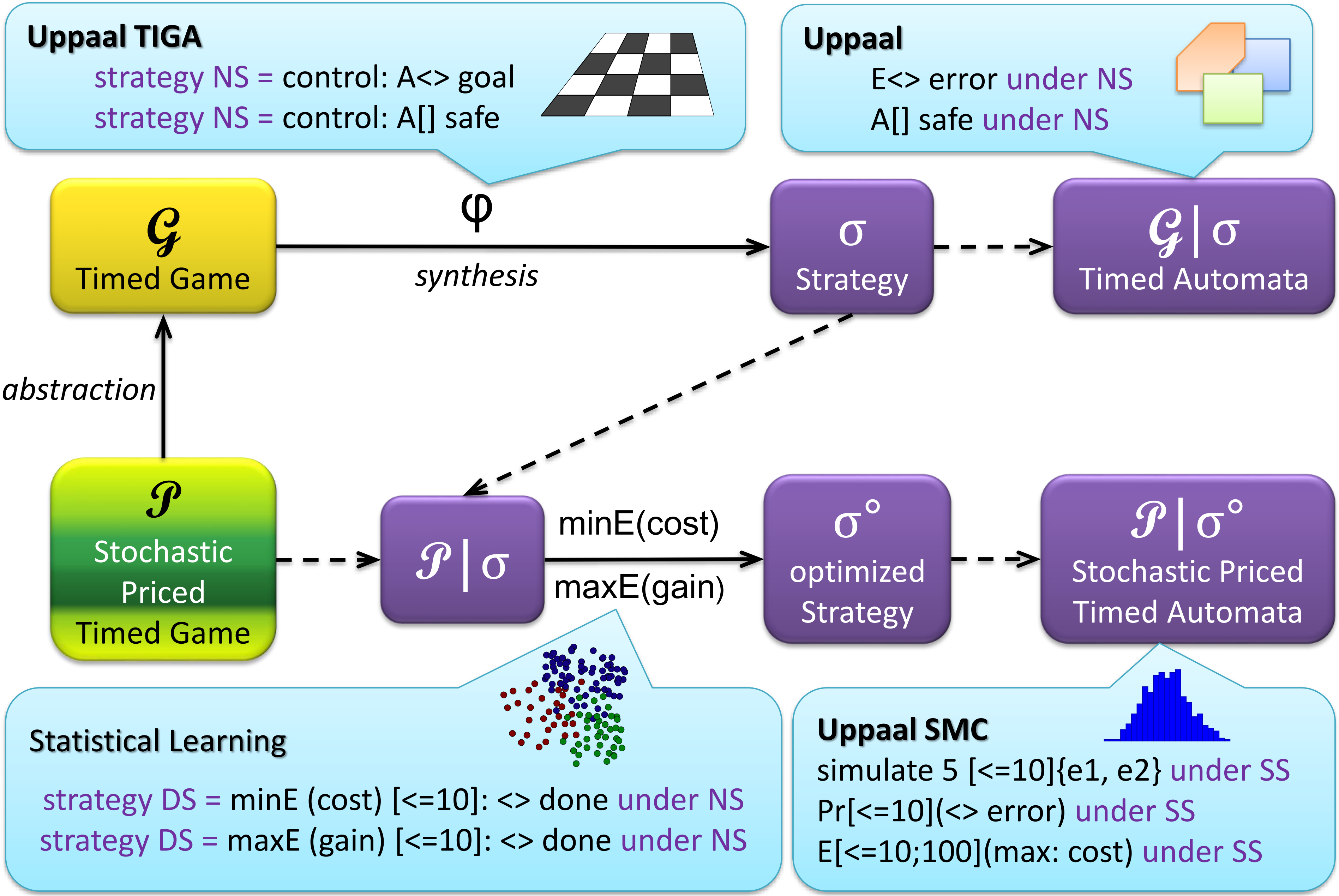
Train Gate
Floor Heating
Adaptive Cruise Control

- Conclusion

Stochastic Timed Game



Find strategy that minimize Expected time from Appr to Cross while being Safe



Uppaal TIGA

strategy NS = control: A<> goal

strategy NS = control: A[] safe

Uppaal

E<> error under NS

A[] safe under NS

$\mathcal{G}$ Timed Game

φ

synthesis

σ Strategy

$\mathcal{G} | \sigma$ Timed Automata

abstraction

$\mathcal{P}$ Stochastic Priced Timed Game

$\mathcal{P} | \sigma$

minE(cost)

maxE(gain)

σ° optimized Strategy

$\mathcal{P} | \sigma^\circ$ Stochastic Priced Timed Automata

Statistical Learning

strategy DS = minE (cost) [<=10]: <> done under NS

strategy DS = maxE (gain) [<=10]: <> done under NS

Uppaal SMC

simulate 5 [<=10]{e1, e2} under SS

Pr[<=10](<> error) under SS

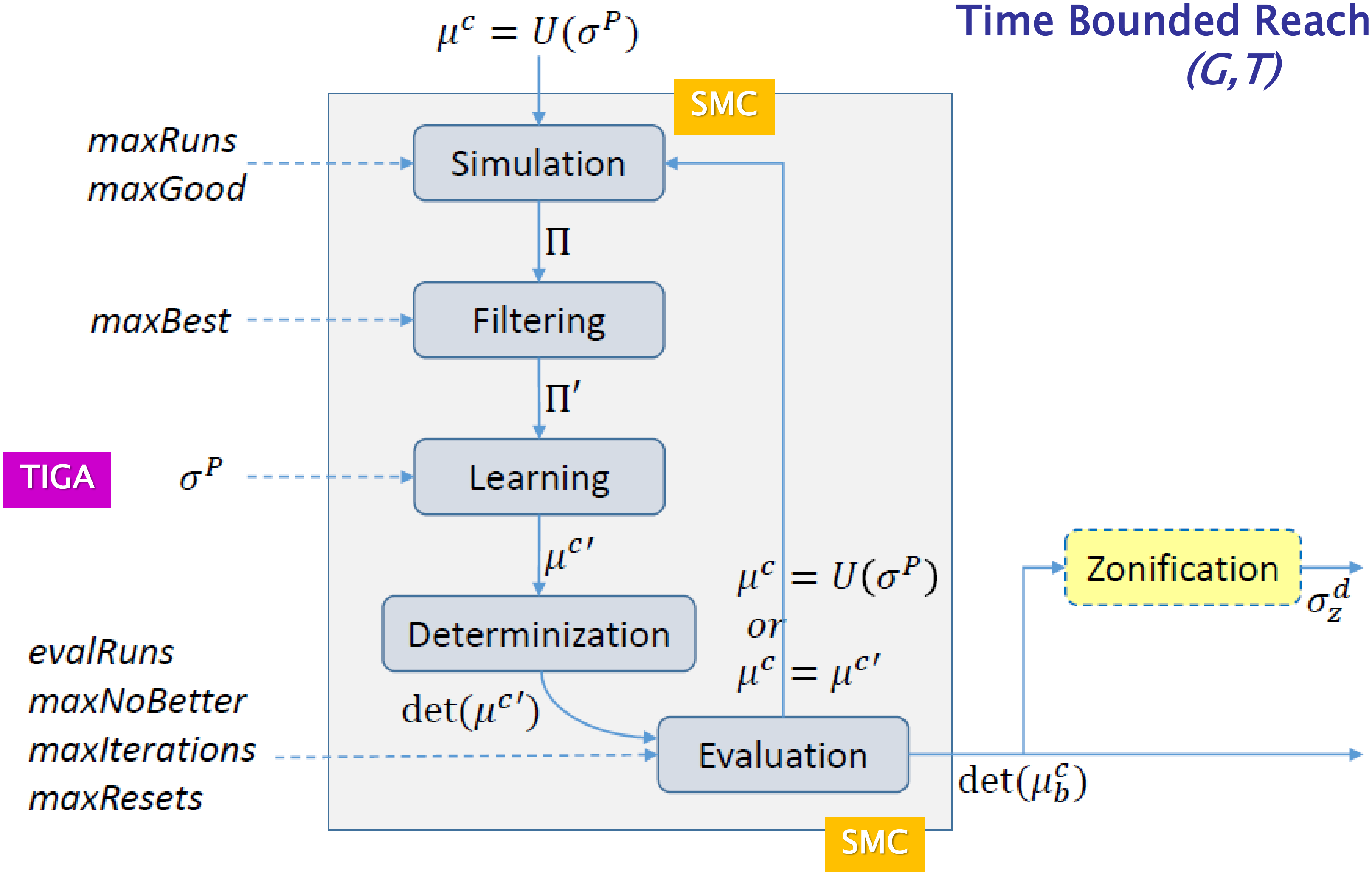
E[<=10;100](max: cost) under SS

DEMO

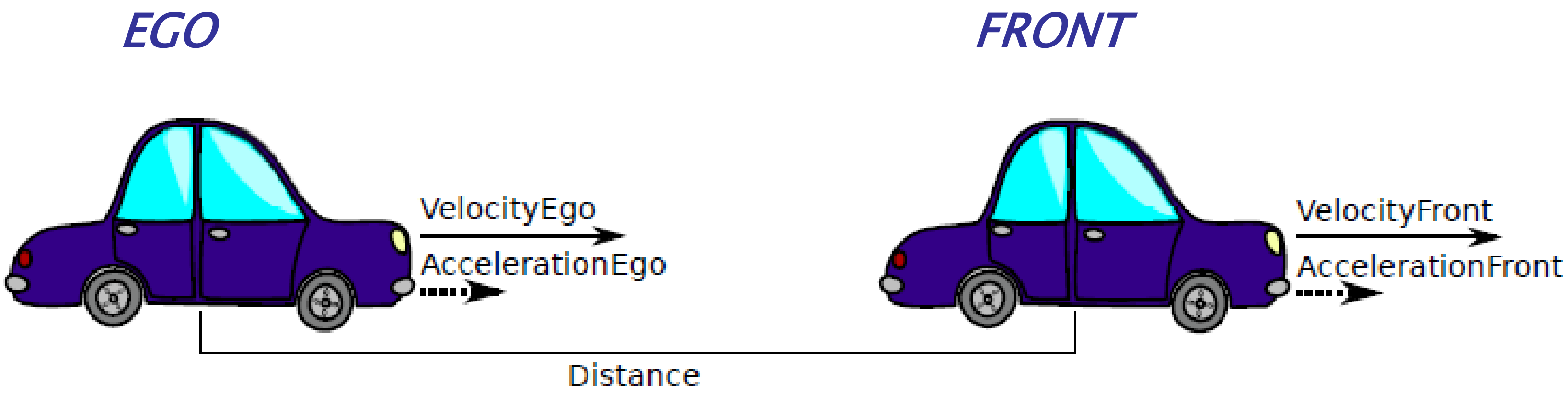
Reinforcement Learning

Time Bounded Reachability

(G,T)

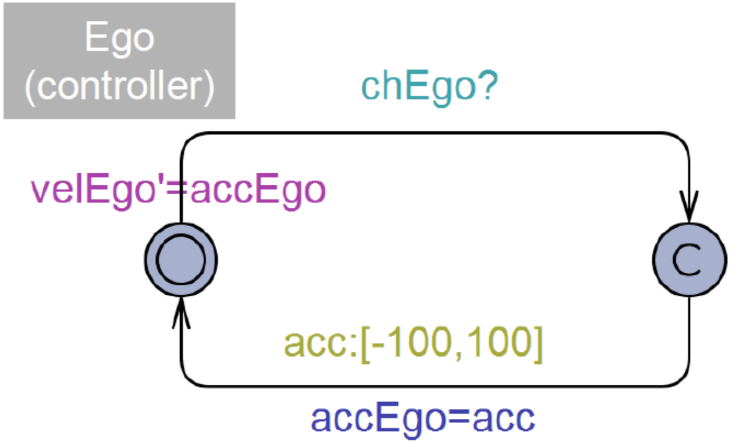


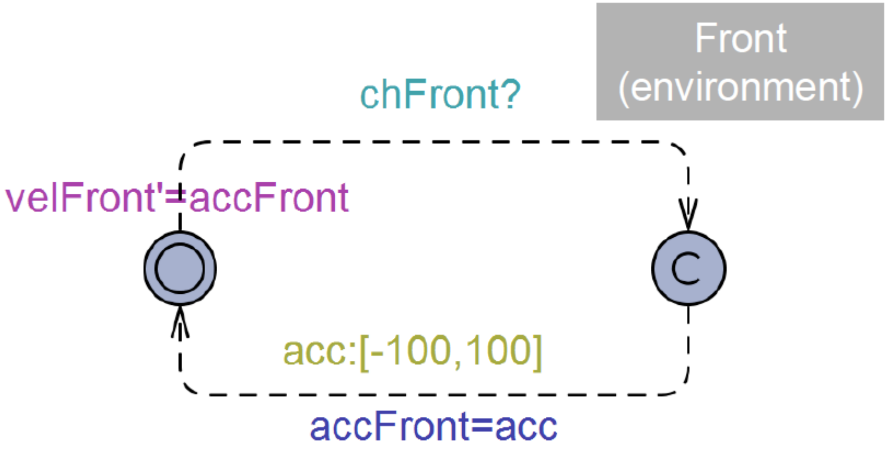
Synthesis of
Safe & Adaptive Cruice Control

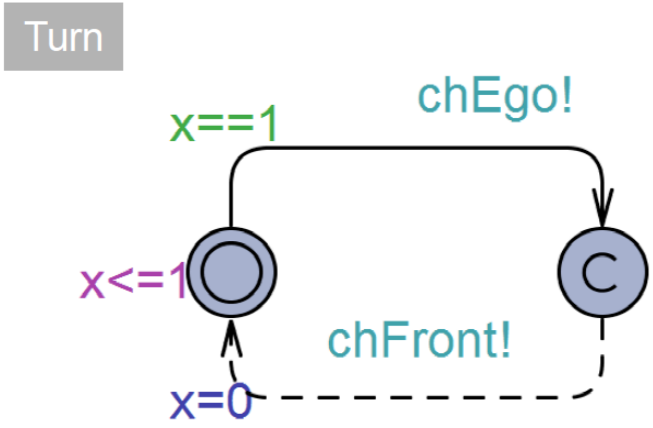


Q1: Find a safety **strategy** for *Ego* such no crash will ever occur no matter what *Front* is doing.
Q2: Find the **most permissive strategy** ensuring safety
Q3: Find the **optimal sub-strategy** that will allow *Ego* to go as far as possible (without overtaking).

Two Player Game (simplified)



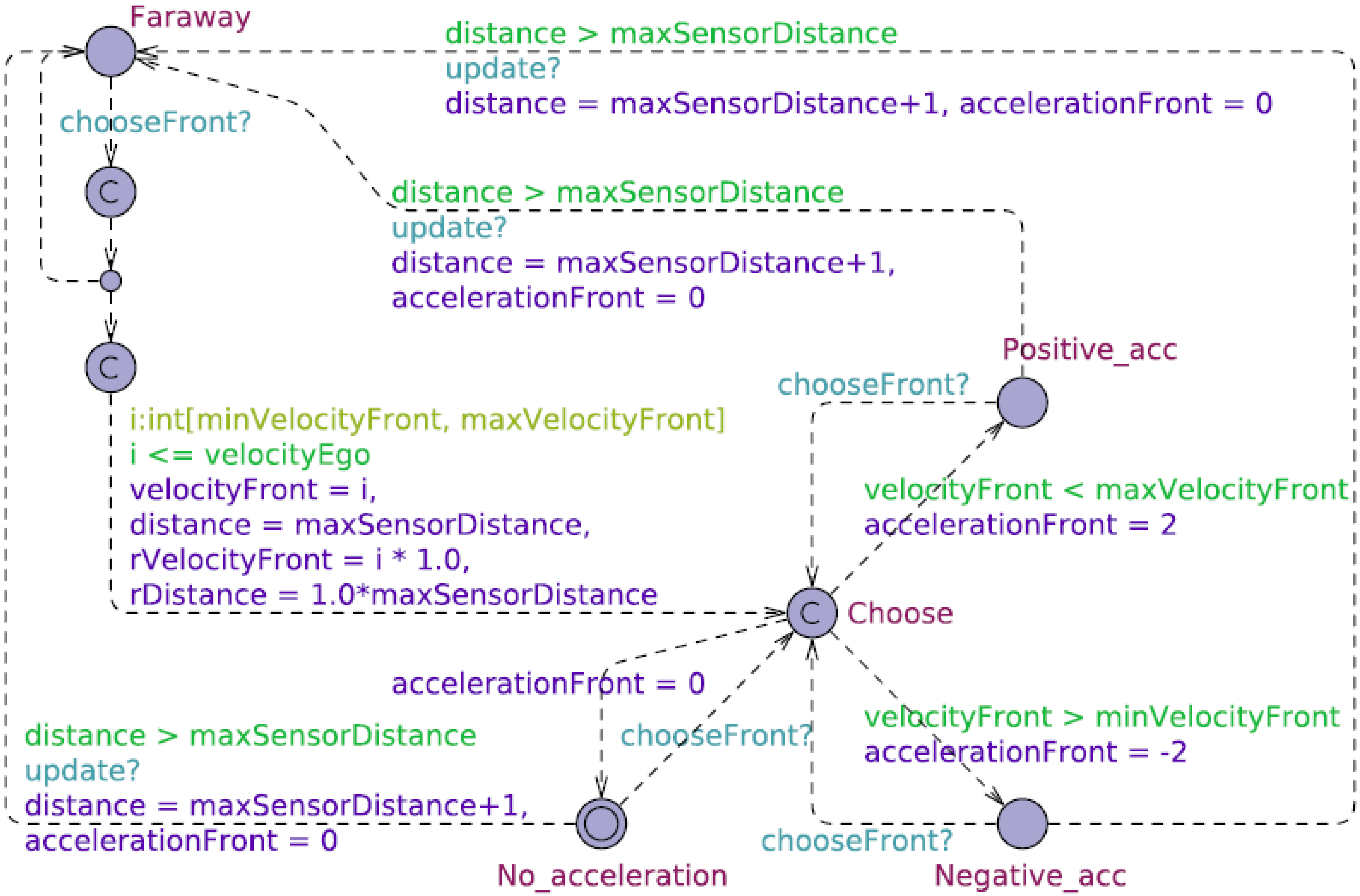




distance' == (velEgo-velFront) &&
D' == distance

Q: find strategy for Ego

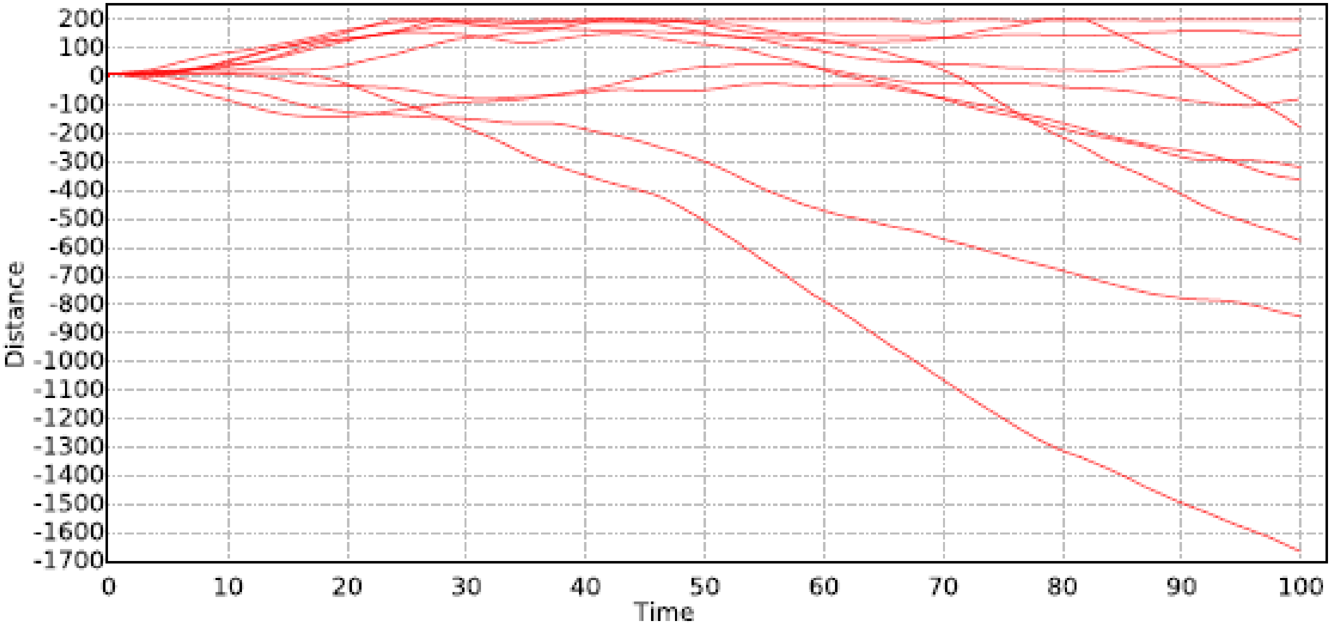
Front (complete)


Faraway
distance > maxSensorDistance
update?
distance = maxSensorDistance+1, accelerationFront = 0
chooseFront?
C
distance > maxSensorDistance
update?
distance = maxSensorDistance+1,
accelerationFront = 0
C
Positive_acc
chooseFront?
i:int[minVelocityFront, maxVelocityFront]
i <= velocityEgo
velocityFront = i,
distance = maxSensorDistance,
rVelocityFront = i * 1.0,
rDistance = 1.0*maxSensorDistance
velocityFront < maxVelocityFront
accelerationFront = 2
C
Choose
accelerationFront = 0
distance > maxSensorDistance
update?
distance = maxSensorDistance+1,
accelerationFront = 0
chooseFront?
velocityFront > minVelocityFront
accelerationFront = -2
chooseFront?
No_acceleration
Negative_acc

No Strategy

`Pr[<=100] (<> distance <= 5)`

`A[] distance > 5`

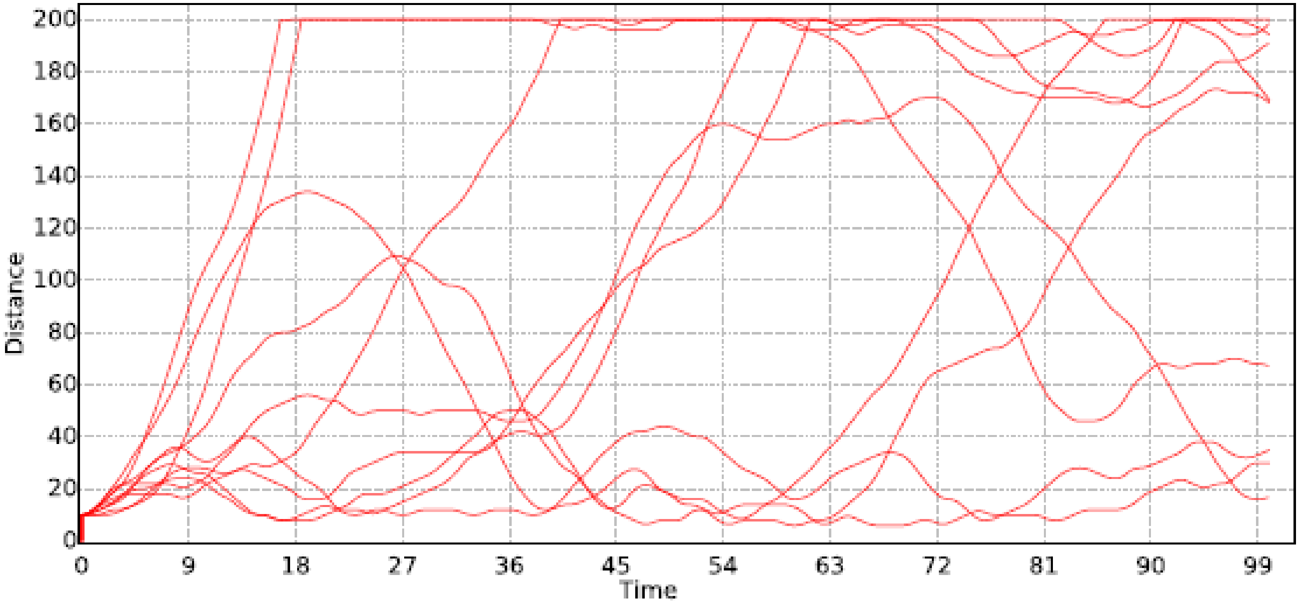


y
ative
je

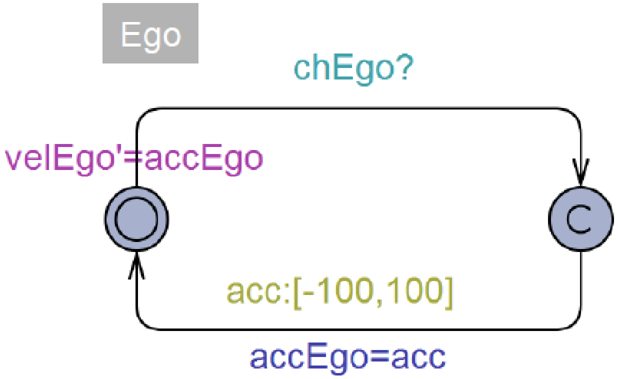
Safety Strategy

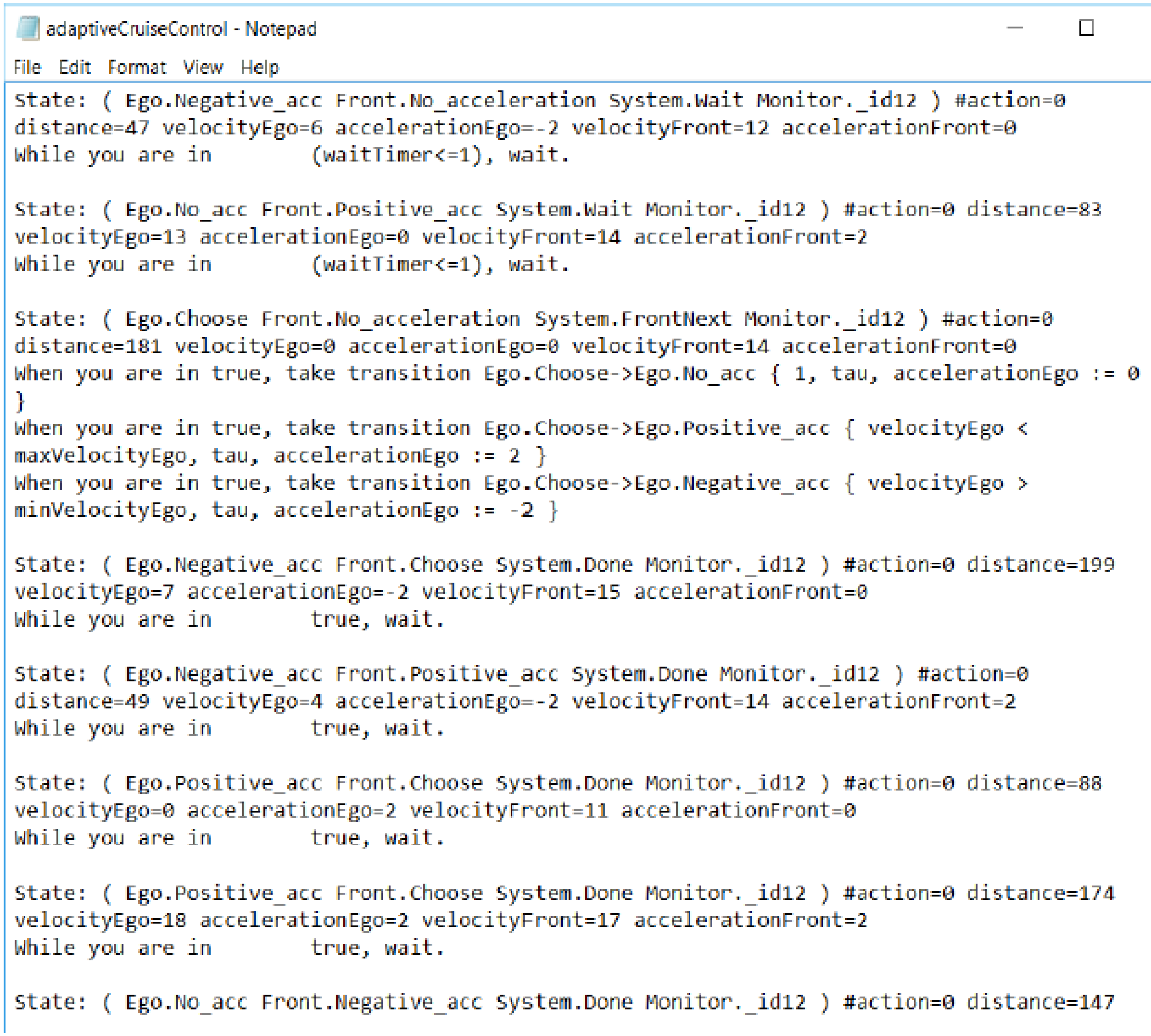
```
strategy safe = control: A[] distance > 5
```

```
A[] distance > 5 under safe
```



Safety Strategy (Code)

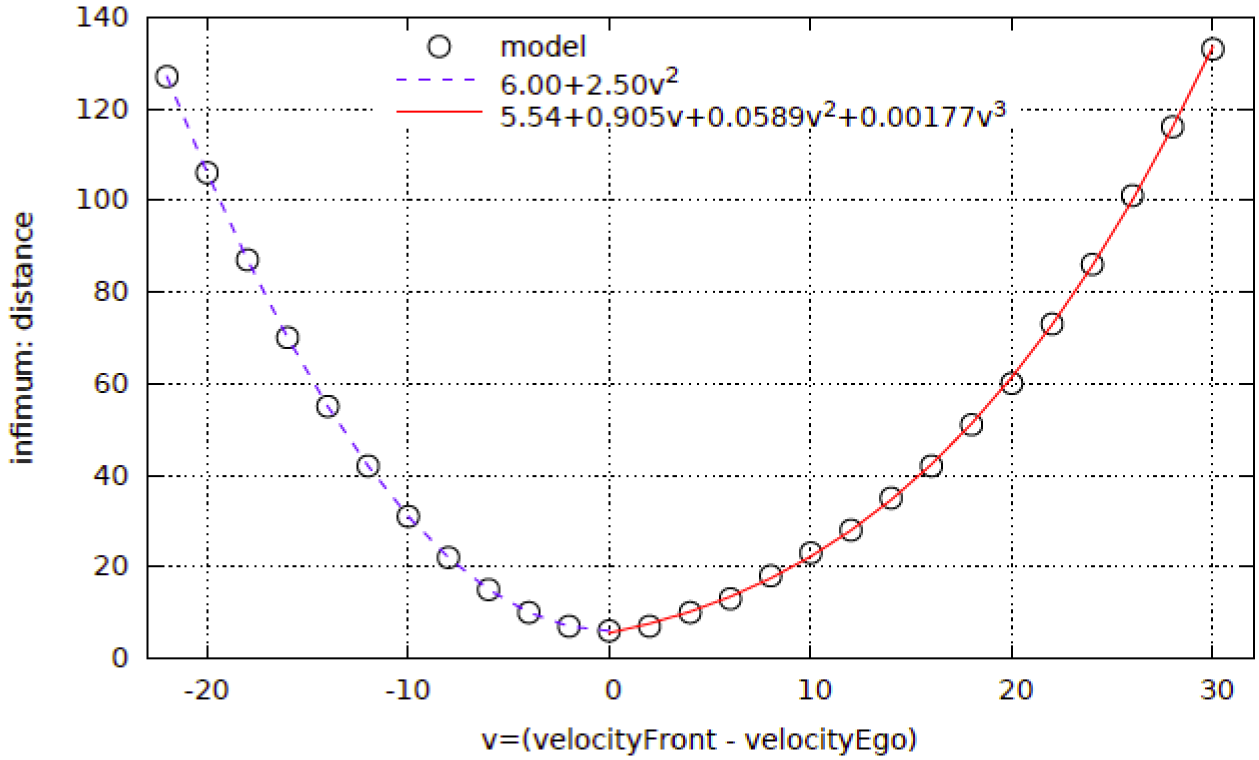




```
State: ( Ego.Negative_acc Front.No_acceleration System.Wait Monitor._id12 ) #action=0
distance=47 velocityEgo=6 accelerationEgo=-2 velocityFront=12 accelerationFront=0
While you are in        (waitTimer<=1), wait.

State: ( Ego.No_acc Front.Positive_acc System.Wait Monitor._id12 ) #action=0 distance=83
velocityEgo=13 accelerationEgo=0 velocityFront=14 accelerationFront=2
While you are in        (waitTimer<=1), wait.

State: ( Ego.Choose Front.No_acceleration System.FrontNext Monitor._id12 ) #action=0
distance=181 velocityEgo=0 accelerationEgo=0 velocityFront=14 accelerationFront=0
When you are in true, take transition Ego.Choose->Ego.No_acc { 1, tau, accelerationEgo := 0
}
When you are in true, take transition Ego.Choose->Ego.Positive_acc { velocityEgo <
maxVelocityEgo, tau, accelerationEgo := 2 }
When you are in true, take transition Ego.Choose->Ego.Negative_acc { velocityEgo >
minVelocityEgo, tau, accelerationEgo := -2 }

State: ( Ego.Negative_acc Front.Choose System.Done Monitor._id12 ) #action=0 distance=199
velocityEgo=7 accelerationEgo=-2 velocityFront=15 accelerationFront=0
While you are in        true, wait.

State: ( Ego.Negative_acc Front.Positive_acc System.Done Monitor._id12 ) #action=0
distance=49 velocityEgo=4 accelerationEgo=-2 velocityFront=14 accelerationFront=2
While you are in        true, wait.

State: ( Ego.Positive_acc Front.Choose System.Done Monitor._id12 ) #action=0 distance=88
velocityEgo=0 accelerationEgo=2 velocityFront=11 accelerationFront=0
While you are in        true, wait.

State: ( Ego.Positive_acc Front.Choose System.Done Monitor._id12 ) #action=0 distance=174
velocityEgo=18 accelerationEgo=2 velocityFront=17 accelerationFront=2
While you are in        true, wait.

State: ( Ego.No_acc Front.Negative_acc System.Done Monitor._id12 ) #action=0 distance=147
```

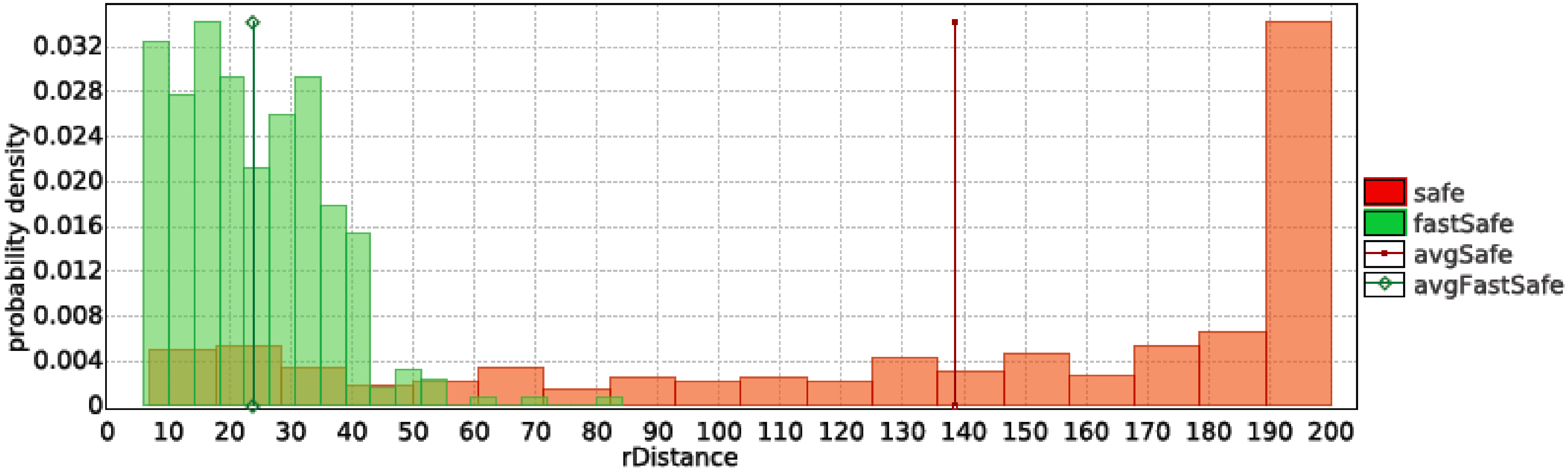
Safety Strategy

```
inf{velosityFront-velosityEgo==v}: distance under safe
```



Optimal and Safe Strategy

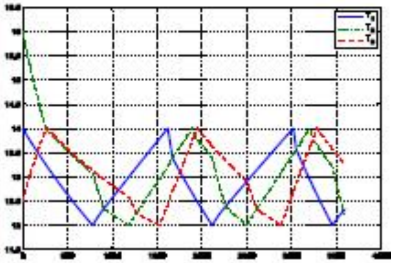
```
strategy safeFast = minE (D) [<=100]: <> time >= 100 under safe
```



Synthesis of Climate Controllers

- Zone-based climate control for pig-stables.

SKOV

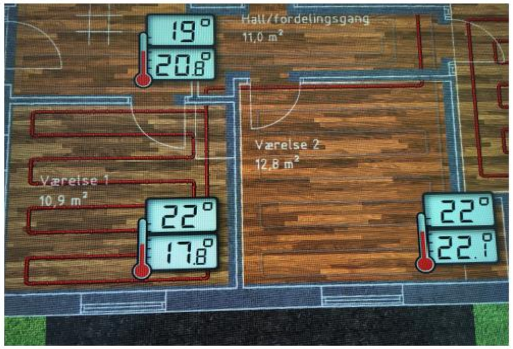


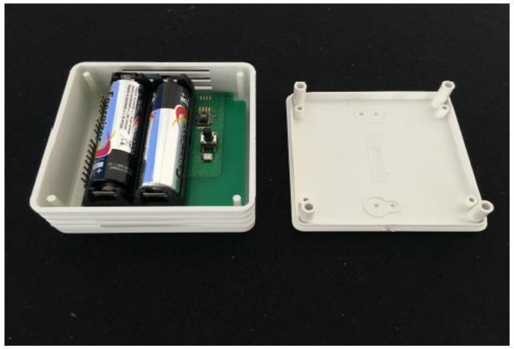
- Floor-Heating

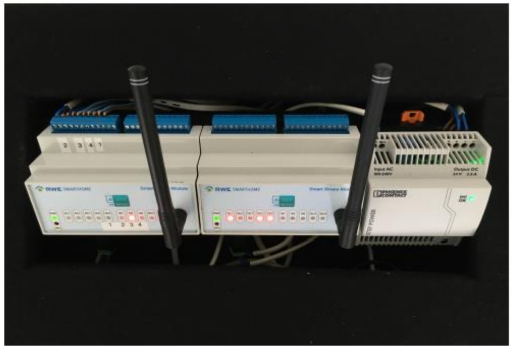
seluxit

Cassting

TACAS16







Synthesis of Climate Controllers

- Zone-based climate control for pig-stables.

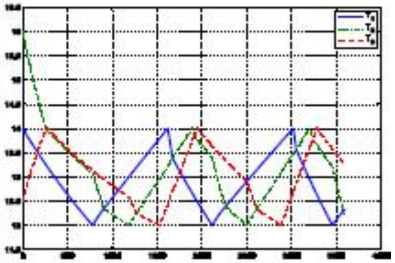
SKOV

- Floor-Heating

seluxit

Cassting

TACAS16



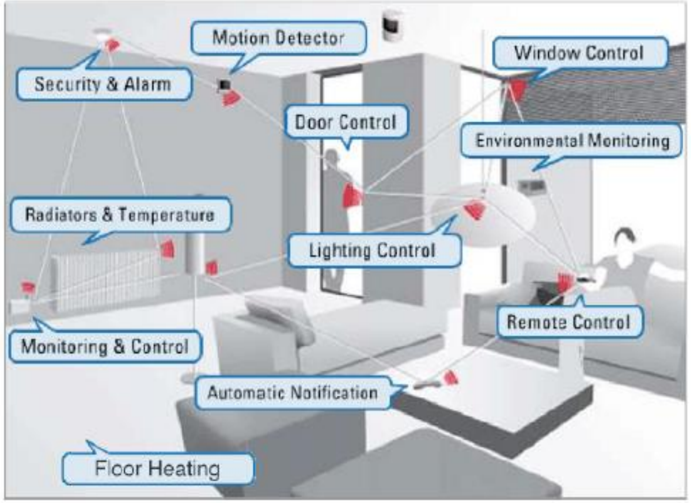
3 day scenario

Weather	Distance			Energy		
	Bang-Bang	Stratego	imp.	Bang-Bang	Stratego	imp.
Aalborg	14583	8342	**43**%	14180	12626	**10**%
Anadyr	2385515	1483272	**37**%	23040	22475	**2**%
Ankara	17985	10464	**41**%	17468	15684	**10**%
Minneapolis	22052	12175	**44**%	18165	15882	**12**%
Murmansk	399421	187941	**52**%	22355	21011	**6**%

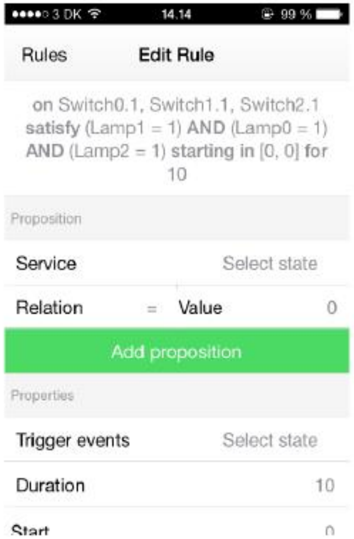
Weather	Distance			Energy		
	Bang-Bang	Stratego	imp.	Bang-Bang	Stratego	imp.
Aalborg	14583	8552	**41**%	14180	12590	**11**%
Anadyr	2385515	1503448	**36**%	23040	22371	**2**%
Ankara	17985	10511	**41**%	17468	15697	**10**%
Minneapolis	22052	12725	**42**%	18165	15837	**12**%
Murmansk	399421	191441	**52**%	22355	20923	**6**%

Modified parameters (0-20%)

Synthesis of Home Automation



Mathias G Sørensen
Best Embedded Systems Thesis, 2013 offered by Federation of Danish Industries


●●●●○ 3 DK 14.14 99 %

Rules **Edit Rule**

on Switch0.1, Switch1.1, Switch2.1 **satisfy** (Lamp1 = 1) **AND** (Lamp0 = 1) **AND** (Lamp2 = 1) **starting in** [0, 0] **for** 10

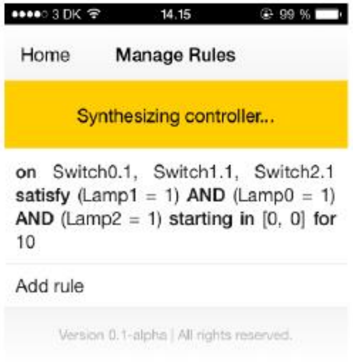
Proposition

Service	Select state
Relation = Value	0

Add proposition

Properties

Trigger events	Select state
Duration	10
Start	0


●●●●○ 3 DK 14.15 99 %

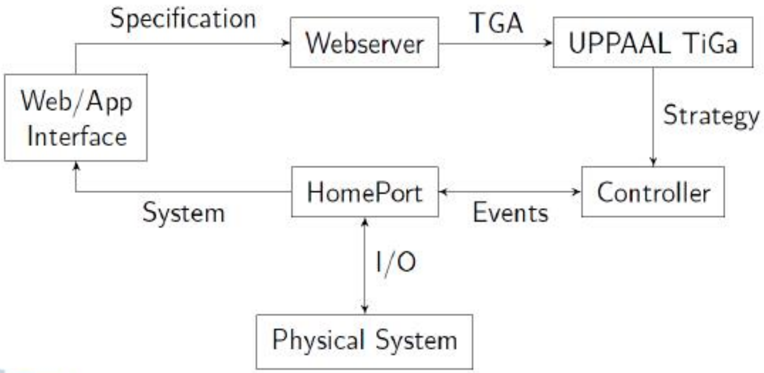
Home **Manage Rules**

Synthesizing controller...

on Switch0.1, Switch1.1, Switch2.1 **satisfy** (Lamp1 = 1) **AND** (Lamp0 = 1) **AND** (Lamp2 = 1) **starting in** [0, 0] **for** 10

Add rule

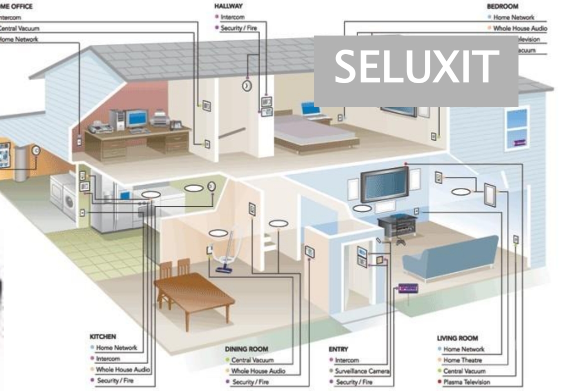
Version 0.1-alpha | All rights reserved.

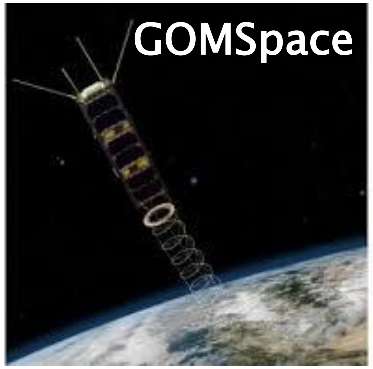


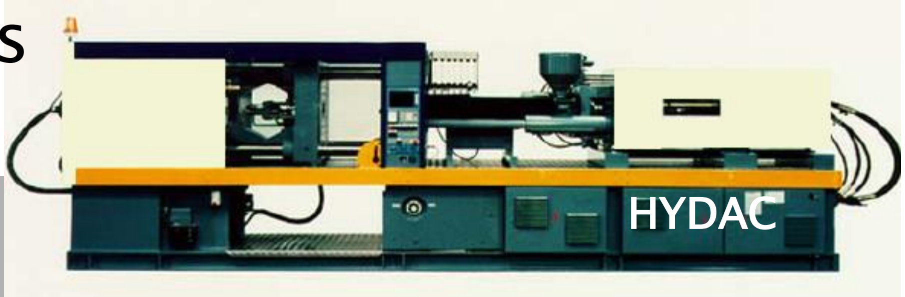
Raspberry Pi Model B
(entire tool chain)

Industrial Applications

- Safe and optimal adaptive cruise control
- Zone-based climate control pig-stables
- Profit-optimal, energy-aware schedules for satelittes
- Personalized light control in home automation
- Energy- and comfort-optimal floor heating
- Safe and energy optimal control of hydralic pumps







Conclusion & Future Work

- Strategies – Representation
  - Non-determinstic strategies $\sigma^{n}_{(\ell,v)} \subseteq (\Sigma_c \cup \{\lambda\})$
  - Stochastic strategies $\mu^{s}_{(\ell,v)} : (\Sigma_c \cup \{\lambda\}) \to [0,1]$
  - Verification of learned strategy
- Better learning methods (Q-learning)
- Beyond safety objectives (MITL)
  - Most (or maximal) permissive strategies
- Verification of discrete strategy for hybrid games
- Partial observability

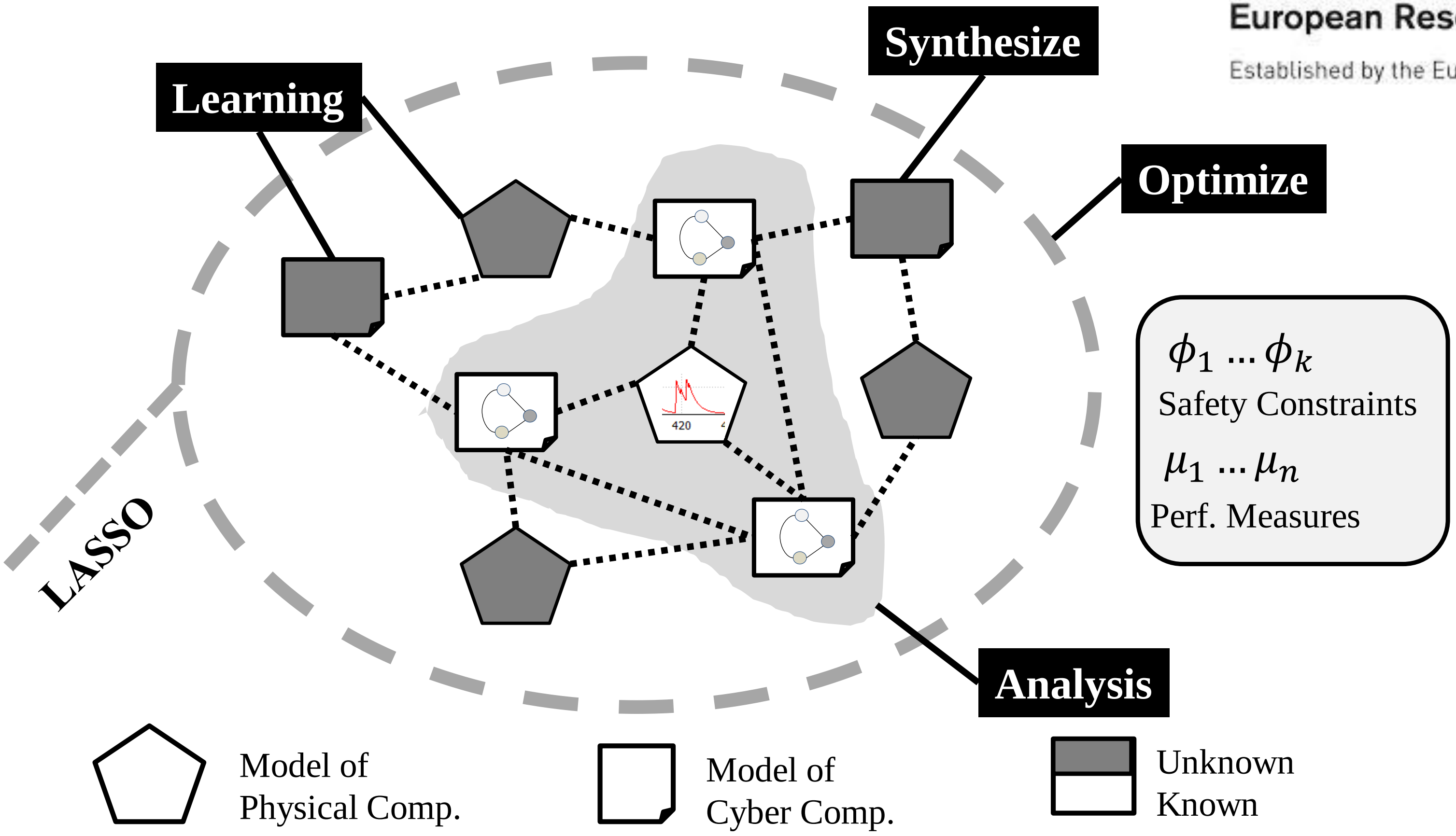
LASSO

Future Work

Learning, Analysis, SynthesiS and Optimization of Cyber-Physical Systems

European Research Council

Established by the European Commission



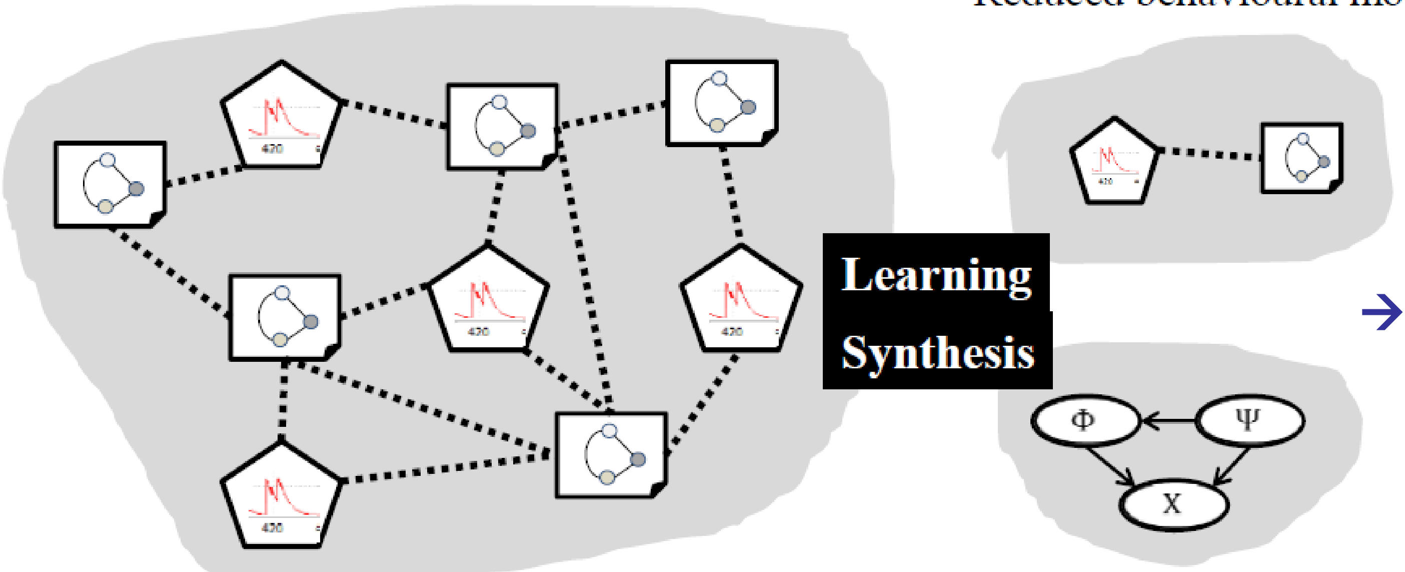
LASSO

Future Work

Learning, Analysis, SynthesiS and Optimization of Cyber-Physical Systems

European Research Council

Established by the European Commission



Reduced behavioural model

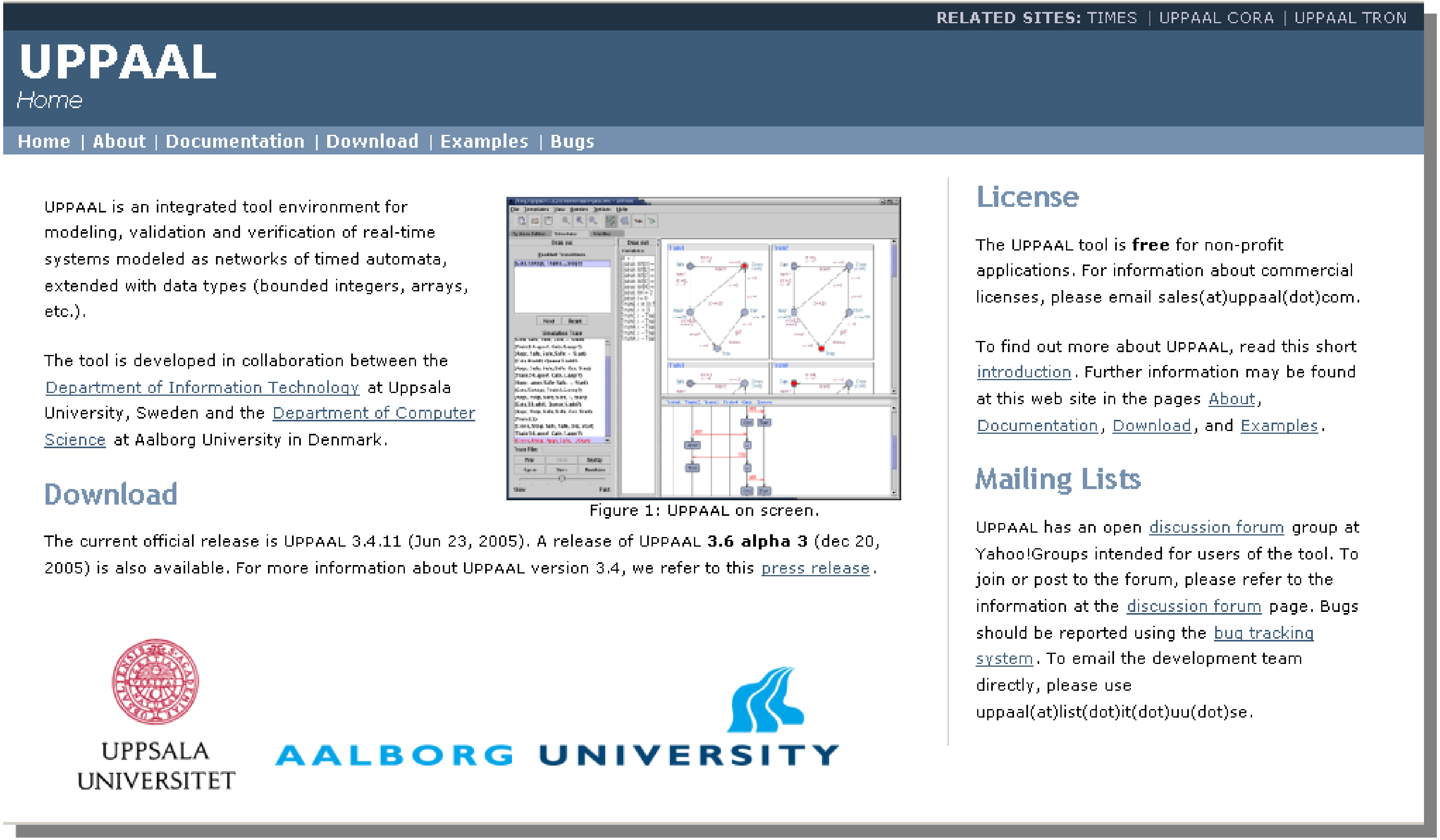
Learning Synthesis

Reduced graphical model

→ **METRICS**

Contact: kgl@cs.aau.dk

www.uppaal.org



RELATED SITES: TIMES | UPPAAL CORA | UPPAAL TRON

UPPAAL

Home

Home | About | Documentation | Download | Examples | 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

The current official release is UPPAAL 3.4.11 (Jun 23, 2005). A release of UPPAAL **3.6 alpha 3** (dec 20, 2005) is also available. For more information about UPPAAL version 3.4, we refer to this press release.

UPPSALA UNIVERSITET

AALBORG UNIVERSITY

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.