From Timed Automata to Stochastic Hybrid Games

Model Checking, Performance Analysis, Optimization, Synthesis, and Machine Learning

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
- ... ...
New Foundation

Discrete Models
(Boolean correctness)

Quantitative Models
(time, resources, probabilistic, stochastic, continuous,..)
(Quantitative correctness)
**QUANTITATIVE Model Checking**

- **Tool**
  - System Description
  - Requirement
  - Prototypes
  - Executable Code
  - Test sequences
  - Debugging Information

- **Open Time**

- **Prototypes**

- **Executable Code**

- **Test sequences**

- **Debugging Information**

**Formulas**

\[
A □ (\text{req} \implies A □ t<30s \text{ grant})
\]

\[
A □ (\text{req} \implies A □ t<30s, c<5 \text{ grant})
\]

\[
A □ (\text{req} \implies A □ t<30s, p>0.90 \text{ grant})
\]

*Kim Larsen [4]*
Synthesis

System Description

Requirement

- \( A \square ( req \Rightarrow A \Diamond \text{grant} ) \)
- \( A \square ( req \Rightarrow A \Diamond_{t<30s} \text{grant} ) \)
- \( A \square ( req \Rightarrow A \Diamond_{t<30s, c<5} \text{grant} ) \)
- \( A \square ( req \Rightarrow A \Diamond_{t<30s, p>0.90} \text{grant} ) \)
Origin of UPPAAL

TAU
CCS & Modal Transition Systems
Refinements
Modal Mu-Calculus
Explicit State Representation
Prolog

UPPAAL
Timed Automata
TCTL
Zones
C++ & Java

EPSILON
TCCS
Timed Refinements
Timed Mu-Calculus
Regions
Prolog

UP4ALL

Grundfos Prize
CAV Award

TU Graz, May 2017
UPPAAL Model Checker

Editor

Simulator

Discrete Control
Concurrency
Continuous Aspects
Stochasticity
Timing Constraints
Resources

Performance Analyses

Verifier

TU Graz, May 2017

Kim Larsen [7]
UPPAAL (1995– )

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

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

@Elsewhere
- Emmanuel Fleury, Didier Lime, Johan Bengtsson, Fredrik Larsson, Kåre J Kristoffersen, Tobias Amnell, Thomas Hune, Oliver Möller, Elena Fersman, Carsten Weise, David Griffeion, Ansgar Fehnker, Frits Vandraager, Theo Ruys, Pedro D'Argenio, J-P Katoen, Jan Tretmans, Judi Romijn, Ed Brinksma, Martijn Hendriks, Klaus Havelund, Franck Cassez, Magnus Lindahl, Francois Laroussinie, Patricia Bouyer, Augusto Burgueno, H. Bowmann, D. Latella, M. Massink, G. Faconti, Kristina Lundqvist, Lars Asplund, Justin Pearson...
UPPAAL Tool Suit

TU Graz, May 2017

Kim Larsen [9]
Topics

- **Timed Automata**
  - Decidability (regions)
  - Symbolic Verification (zones)

- **Priced Timed Automata**
  - Decidability (priced regions)
  - Symbolic Verification (priced zones)

- **Stochastic Timed Automata**
  - Stochastic Semantics
  - Statistical Model Checking
  - Stochastic Hybrid Automata

- **Timed Games & Interfaces**
  - Strategies, Symbolic Synthesis
  - Refinement

- **Stochastic Priced Timed Games**
  - Strategies
  - Symbolic Synthesis (zones)
  - Stochastic Strategies
  - Reinforcement Learning

---

TU Graz, May 2017

Kim Larsen [10]
From **Timed Automata** to **Stochastic Hybrid Games**
Model Checking, Optimization, Performance Evaluation, Synthesis and Machine Learning using UPPAAL

Technische Universität Graz, May 2017

**Kim Guldstrand Larsen**
CISS, Aalborg University, DENMARK

**Slides**
1. Timed Automata, UPPAAL
2. Priced Timed Automata, Energy Automata, UPPAAL CORA
3. Stochastic Timed Automata, UPPAAL SMC
4. Stochastic Timed Games, UPPAAL TIGA and UPPAAL STRATEGO

**Primary Reading**
Frits Vaandrager: [*A first introduction to UPPAAL*]
Alexandre David, Kim G Larsen: *More features in UPPAAL*
Timed Automata
Real Time Systems

Plant
Continuous

Controller Program
Discrete

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

Real Time System
A system where correctness not only depends on the logical order of events but also on their timing!!
A Dumb Light Controller
Timed Automata

ADD a clock $x$

- **Reset**
- **Synchronizing action**
- **Clock Guard Conjunctions of $x \sim n$**

$x$: real-valued clock

**Off**
- press?
- $x=0$
- press?
- $x>3$

**Light**
- press?
- $x=0$
- press?
- $x<3$

**Bright**
- press?
A Timed Automata (Semantics)

States: 
( location , x=v) where v∈R

Transitions:
( Off , x=0 )
delay 4.32  →  ( Off , x=4.32 )
press?    →  ( Light , x=0 )
delay 2.51  →  ( Light , x=2.51 )
press?    →  ( Bright , x=2.51 )
Intelligent Light Controller

Invariant (Henzinger)
Intelligent Light Controller

Transitions:

- Delay 4.32 \( \rightarrow ( \text{Off}, x=4.32 ) \)
- Press? \( \rightarrow ( \text{Light}, x=0 ) \)
- Delay 4.51 \( \rightarrow ( \text{Light}, x=4.51 ) \)
- Press? \( \rightarrow ( \text{Light}, x=0 ) \)
- Delay 100 \( \rightarrow ( \text{Light}, x=100 ) \)
- \( \tau \) \( \rightarrow ( \text{Off}, x=0 ) \)

Note:

- \( ( \text{Light}, x=0 ) \) Delay 103 \( \rightarrow \)

Invariants ensures progress
Clock Valuations

Let $C = \{x, y, \ldots\}$ be a finite set of clocks.

Set $\mathcal{B}(C)$ of clock constraints over $C$

$\mathcal{B}(C)$ is defined by the following abstract syntax

$$g, g_1, g_2 ::= x \sim n \mid x - y \sim n \mid g_1 \land g_2$$

where $x, y \in C$ are clocks, $n \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$.

Example:
Clock valuations

Clock valuation $\nu$ is a function $\nu : C \rightarrow \mathbb{R}^{\geq 0}$.

Let $\nu$ be a clock valuation. Then

- $\nu + d$ is a clock valuation for any $d \in \mathbb{R}^{\geq 0}$ and it is defined by
  $$(\nu + d)(x) = \nu(x) + d \text{ for all } x \in C$$

- $\nu[r]$ is a clock valuation for any $r \subseteq C$ and it is defined by
  $$\nu[r](x) = \begin{cases} 0 & \text{if } x \in r \\ \nu(x) & \text{otherwise.} \end{cases}$$
Evaluation of clock constraints $(\nu \models g)$

\begin{align*}
\nu \models x < n & \quad \text{iff } \nu(x) < n \\
\nu \models x \leq n & \quad \text{iff } \nu(x) \leq n \\
\nu \models x = n & \quad \text{iff } \nu(x) = n \\
\vdots & \quad \vdots \\
\nu \models x - y < n & \quad \text{iff } \nu(x) - \nu(y) < n \\
\nu \models x - y \leq n & \quad \text{iff } \nu(x) - \nu(y) \leq n \\
\vdots & \quad \vdots \\
\nu \models g_1 \land g_2 & \quad \text{iff } \nu \models g_1 \text{ and } \nu \models g_2
\end{align*}
A **timed automaton** over a set of clocks $C$ and a set of labels $N$ is a tuple

$$(L, \ell_0, E, I)$$

where

- $L$ is a finite set of **locations**
- $\ell_0 \in L$ is the **initial location**
- $E \subseteq L \times \mathcal{B}(C) \times N \times 2^C \times L$ is the set of **edges**
- $I : L \rightarrow \mathcal{B}(C)$ assigns **invariants** to locations.

We usually write $\ell \overset{g,a,r}{\rightarrow} \ell'$ whenever $(\ell, g, a, r, \ell') \in E$. 
Let $A = (L, l_0, E, I)$ be a timed automaton.

Timed transition system generated by $A$

$T(A) = (Proc, Act, \{ \frac{a}{\rightarrow} \mid a \in Act \})$ where

- $Proc = L \times (C \rightarrow \mathbb{R}^{\geq 0})$, i.e. states are of the form $(\ell, v)$ where $\ell$ is a location and $v$ a valuation
- $Act = N \cup \mathbb{R}^{\geq 0}$
- $\rightarrow$ is defined as follows:

$(\ell, v) \xrightarrow{a} (\ell', v')$ if there is $(\ell \xrightarrow{g,a,r} \ell') \in E$ s.t. $v \models g$ and $v' = v[r]

(\ell, v) \xrightarrow{d} (\ell, v + d)$ for all $d \in \mathbb{R}^{\geq 0}$ s.t. $v \models I(\ell)$ and $v + d \models l(\ell)$
Example

\[
\begin{align*}
2 \leq x \land \land \\
x \leq 3
\end{align*}
\]

\[
\begin{align*}
a!
\end{align*}
\]

\[
\begin{align*}
x = 0
\end{align*}
\]
Example

\[
\text{2} \leq x \land \land \\
\text{x} \leq 3
\]

2 \leq x \land x \leq 3

x = 0

Simulations

\begin{align*}
\text{value} & \quad 0.6 & 1.2 & 1.8 & 2.4 \\
\text{time} & \quad 0 & 1.3 & 2.6 & 3.9 & 5.2 & 6.5 & 7.8 & 9.1 & 10.4 & 11.7
\end{align*}
Example

Is \( L_1 \) reachable?
Example

TU Graz, May 2017
Kim Larsen [29]
Example

\[ (l_0, x = 0, y = 0) \]
\[ \xrightarrow{1.4} (l_0, x = 1.4, y = 1.4) \]
Example

\[(\ell_0, x = 0, y = 0)\]

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

\[\xrightarrow{a} (\ell_0, x = 1.4, y = 0)\]
Example

\[
\begin{align*}
(\ell_0, x = 0, y = 0) \\
\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4) \\
\xrightarrow{a} (\ell_0, x = 1.4, y = 0) \\
\xrightarrow{1.6} (\ell_0, x = 3.0, y = 1.6) \\
\xrightarrow{a} (\ell_0, x = 3.0, y = 0)
\end{align*}
\]
UPPAAL
First Introduction
Light Control Interface

User

Interface

Light

Control Program

- press? d release? → touch! \(0.5 \leq d \leq 1\)
- press? 1 → starthold!
- press? d release? → endhold! \(d > 1\)

- press? 0.2 release? \(\emptyset\) → touch!
- press? 0.7 release? \(\emptyset\) → starthold!
- press? 1.0 release? \(\emptyset\) → endhold!

- press? 2.4 release? \(\emptyset\) → endhold!

TU Graz, May 2017
Light Control Interface

User

Press?

Release?

Press?

Release?

Touch!

StartHold!

EndHold!

Control Program

Dim

Switch

on = 0

touch?

x = 0

L = OL,

L = L + 1

x <= delay

L <= Max,

x = delay

x = 0

on = 0

Touch?

L = OL,

L = L + 1

L <= delay

End!
Light Control Network

![Diagram of Light Control Network]

TU Graz, May 2017
Full Light Controller
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.

![LEGO Mindstorms/RCX](image)
A Real Real Timed System

The Plant
Conveyor Belt & Bricks

Controller Program
LEGO MINDSTORM
First UPPAAL model

Sorting of Lego Boxes

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

Ken Tindell

Tu Graz, May 2017

Kim Larsen [40]
NQC programs

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

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

    start PUSH;

    while(true){
        wait(IN_1<=LIGHT_LEVEL);
        ClearTimer(1);
        active=1;
        PlaySound(1);

        wait(IN_1>LIGHT_LEVEL);
    }
}

task PUSH{
    while(true){
        wait(Timer(1)>DELAY && active==1);
        active=0;
        Rev(OUT_C,1);
        Sleep(8);
        Fwd(OUT_C,1);
        Sleep(12);
        Off(OUT_C);
    }
}```
A Black Brick

![Diagram](image-url)
GLOBAL DECLARATIONS:

const int ctime = 75;

int[0,1] active;
clock x, time;

chan eject, ok;
urgent chan blk, red, remove, go;
From RCX to UPPAAL – and back

- Model includes Round–Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000 in Stockholm.

- From UPPAAL to RCX: Martijn Hendriks.
The Production Cell in LEGO

Course at DTU, Copenhagen
UPPAAL

Modeling & Specification
Train Crossing

Safe → Approaching → Crossing → Safe

Tracks

River

Bridge

Time

0 → 20 → 3 – 5

TU Graz, May 2017
Train Crossing

Safe → Approaching → Crossing → Safe
Safe → Approaching → Crossing → Safe

Stop the train while it still stoppable!

TU Graz, May 2017
Train Crossing

Safe → Approaching → Crossing → Safe
Safe → Approaching → Crossing → Safe

Stopped → Restarted

Bridge

tracks

0 10 20

3 – 5 7 – 15

River

TU Graz, May 2017
Timed Automata [Train] = Finite State Control + Real Valued Clocks

- Invariants
- Guards
- Synchronizations
- Resets

Diagram:

- Safe
- Cross
- Start
- Stop

Transitions:
- appr[id]!
- x=0
- x<=20
- x=0
- x<=10
- stop[id]?
- go[id]?
- x=0
- x<=15
- x<=5
- x>=3
- leave[id]!
- x=0
- x>=10
- x=0
- x>=7
- x=0

TU Graz, May 2017
Kim Larsen [51]
Timed Automata [Gate] = Finite State Control + Real Valued Clocks + Discrete Variables

```c
id_t list[N+1];
int[0,N] len;

// Put an element at the end of the queue
void enqueue(id_t element)
{
    list[len++] = element;
}

// Remove the front element of the queue
void dequeue()
{
    int i = 0;
    len -= 1;
    while (i < len)
    {
        list[i] = list[i + 1];
        i++;
    }
    list[i] = 0;
}
```
UPPAAL Help

UPPAAL is a tool for modeling, validation and verification of real-time systems. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks (i.e. timed automata), communicating through channels and (or) shared data structures. Typical application areas include real-time controllers, communication protocols, and other systems in which timing aspects are critical.

The UPPAAL tool consists of three main parts:

- a graphical user interface (GUI),
- a verification server, and
- a command line tool.

The GUI is used for modelling, simulation, and verification. For both simulation and verification, the GUI uses the verification server. In simulation, the server is used to compute successor states. The command line tool is a stand-alone verifier, appropriate for e.g. batch verifications.

More information can be found at the UPPAAL web site: http://www.uppaal.com.
Logical Specifications

- **Validation Properties**
  - Possibly: $E<> P$

- **Safety Properties**
  - Invariant: $A[] P$
  - Pos. Inv.: $E[] P$

- **Liveness Properties**
  - Eventually: $A<> P$
  - Leadsto: $P \rightarrow Q$

- **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 (and arrays of these) are allowed.
Logical Specifications

- Validation Properties
  - Possibly: \( E<> P \)

- Safety Properties
  - Invariant: \( A[] P \)
  - Pos. Inv.: \( E[] P \)

- Liveness Properties
  - Eventually: \( A<> P \)
  - Leadsto: \( P \rightarrow Q \)

- Bounded Liveness
  - Leads to within: \( P \rightarrow_{\leq t} Q \)
Logical Specifications

- **Validation Properties**
  - Possibly: \( E <> P \)

- **Safety Properties**
  - Invariant: \( A[P] \)
  - Pos. Inv.: \( E[P] \)

- **Liveness Properties**
  - Eventually: \( A<> P \)
  - Leadsto: \( P \rightarrow Q \)

- **Bounded Liveness**
  - Leads to within: \( P \rightarrow_{\leq t} Q \)
Logical Specifications

- Validation Properties
  - Possibly: $E<> P$

- Safety Properties
  - Invariant: $A[] P$
  - Pos. Inv.: $E[] P$

- Liveness Properties
  - Eventually: $A<> P$
  - Leadsto: $P \rightarrow Q$

- Bounded Liveness
  - Leads to within: $P \rightarrow^* Q$

TU Graz, May 2017
Logical Specifications

- **Validation Properties**
  - Possibly: \( E<> P \)

- **Safety Properties**
  - Invariant: \( A[] P \)
  - Pos. Inv.: \( E[] P \)

- **Liveness Properties**
  - Eventually: \( A<> P \)
  - Leadsto: \( P \rightarrow Q \)

- **Bounded Liveness**
  - Leads to within: \( P \rightarrow_{\leq t} Q \)
Demo 2
Editor

**Language**
- User defined functions (C-like)
- New types (records, type declarations, meta variables, scalars)
- Partial instantiation of templates
- Select clauses on edges
- Forall and exist quantifiers
Concrete Simulator

Graphical Simulator
- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gantt Charts
Symbolic Simulator

Graphical Simulator

- visualization and recording
- inexpensive fault detection
- inspection of error traces
- Message Sequence Charts
- Gantt Charts

TU Graz, May 2017

Kim Larsen [63]
Verifier

- Exhaustive & automatic checking of requirements
- .. including validating, safety, liveness, bounded liveness and response properties
- .. performance properties, e.g. probabilistic and expectation.
- .. generation of debugging information for visualisation in simulator.
- .. plot composer
Applications
(some)
Bang & Olufsen IR–Link

- Bug known to exist for 10 years
- Ill-described:
  - 2,800 lines of assembler code + 3 flowchart + 1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.

Arne Skou, Klaus Havelund

TU Graz, May 2017
Bug known to exist for 10 years
Ill-described:
   2,800 lines of assembler code + 3 flowchart + 1 B&O eng.
3 months for modeling.
UPPAAL detects error with 1,998 transition steps (shortest)
Error trace was confirmed in B&O laboratory.
Error corrected and verified in UPPAAL.
Philips Bounded Retransmission Protocol

Pedro D’Argenio
Joost-Pieter Katoen
Theo Ruys
Jan Tretmans

I should tell you that I am quite disappointed with this new release of Uppaal ;). You take all the fun out of it!! With this new release I could verify everything in a couple of minutes, including a couple of properties that were impossible before!! Moreover, I was playing with the simulator and I found a silly deadlock in the specification.

I found this new Uppaal a quite huge leap from the previous version. As a user, I have had a really good first impression. I will compile a list of comments for tomorrow afternoon.

Regards,
FlexRay

BMW, Bosch, Daimler, Freescale, General Motors, NXP Semiconductors, and Volkswagen

Fault-tolerance
Timed hardware model
Parameterized error models (glitches, jitter)
Voting & bit-clock alignment

[Gerke, Ehlers, Finkbeiner, Peters, 2010]
Gear Controller

with MECEL AB

Flowgraph

TU Graz, Mai 2017
Gear Controller
with MECEL AB

Network Canbus

Interface

Volvo Saab

GearBox Engine

Floorplan

Timed Automata Models

<Global variables>
UseCase = 0
FromGear = 0
ToGear = 0
ErrStat = 0

<Constraints>
CTimer ≥ 0
ETimer ≥ 0
GBTimer ≥ 0
GCTimer ≥ 0
SysTimer ≥ 0
GearControl.GCTimer ≥ 0

CTimer = ETimer
ETimer = GBTimer
GBTimer = GCTimer
GCTimer = SysTimer
SysTimer = GearControl.GCTimer
GearControl.GCTimer = CTimer

ClutchClosed?
GCTimer = 0

CheckClutchClosed
GCTimer <= 200
GCTimer > 150 and
GCTimer <= 200

Changed
CCloseError

Engine

GearBox

Interface

ErrorClose

CTimer = 150

TU Graz, May 2017

Magnus Lindahl
Paul Pettersson
Wang Yi
2001
Gear Controller

with MECEL AB

Magnus Lindahl
Paul Pettersson
Wang Yi
2001

GearControl\@Initiate \(\Rightarrow_{\leq 1500} ( ( \text{ErrStat} = 0 ) \Rightarrow \text{GearControl}\@\text{GearChanged} ) \)

GearControl\@Initiate \(\Rightarrow_{\leq 1000} ( ( \text{ErrStat} = 0 \land \text{UseCase} = 0 ) \Rightarrow \text{GearControl}\@\text{GearChanged} ) \)

Clutch\@ErrorClose \(\Rightarrow_{\leq 200} \text{GearControl}\@\text{CCloseError} \)

Clutch\@ErrorOpen \(\Rightarrow_{\leq 200} \text{GearControl}\@\text{COpenError} \)

GearBox\@ErrorIdle \(\Rightarrow_{\leq 350} \text{GearControl}\@\text{GSetError} \)

GearBox\@ErrorNeu \(\Rightarrow_{\leq 200} \text{GearControl}\@\text{GNeuError} \)

Inv ( GearControl\@CCloseError \(\Rightarrow\) Clutch\@ErrorClose )

Inv ( GearControl\@COpenError \(\Rightarrow\) Clutch\@ErrorOpen )

Inv ( GearControl\@GSetError \(\Rightarrow\) GearBox\@ErrorIdle )

Inv ( GearControl\@GNeuError \(\Rightarrow\) GearBox\@ErrorNeu )

Inv ( Engine\@ErrorSpeed \(\Rightarrow\) ErrStat \(\neq\) 0 )

Inv ( Engine\@Torque \(\Rightarrow\) Clutch\@Closed )
UPPAAL Model Checking – Demo

Overview

E<> GearControl.GearChanged
E<> ( Interface.Gear5 )
E<> ( Interface.GearR )
E<> ( GearControl.GearChanged and ( SysTimer<=1000 ) )
A[] not ( GearBox.Neutral and ( Interface.Gear1 or Interface.Gear2 or Interface.Gear3 or Interface.Gear4 or Interface.Gear5 ) )
A[] not ( GearBox.Idle and Interface.GearN )
A[] ( Interface.GearN imply GearBox.Neutral )
A[] ( ( ErrStat==0 and UseCase==0 and SysTimer>=900 ) imply ( GearControl.GearChanged... ) )
E<> ( ErrStat==0 and UseCase==0 and SysTimer>899 and SysTimer<900 and not ( Gear... ) )
A[] ( ( ErrStat==0 and UseCase==0 and ( SysTimer<150 ) ) imply not ( GearControl.Gear... ) )

Query

E<> GearControl.GearChanged

Comment

P1. It is possible to change gear.

a) If the clutch is not closed properly (i.e. a timeout occurs) the gearbox controller will enter the location CCloseError within 200 ms.
Advanced Noise Reduction Techniques

Airport Surveillance

Costal Surveillance

Sweep Integration

Frequency Diversity

9.170 GHz
9.438 GHz

echo

9.438 GHz

Combiner (VP3)

AMETIST
advanced methods for timed systems

Kim Larsen [75]

TU Graz, May 2017
Attitude and Orbit Control Software
TERMA A/S Steen Ulrik Palm, Jan Storbak Pedersen, Poul Hougaard
**METAMOC**

Modular Execution Time Analysis using MOdel Checking

with

Andreas Dalsgaard
Mads Christian Olesen
Martin Toft
René Rydhof Hansen

Abstract process model and value analysis

Abstract hardware model with caching and pipelining

Timed automata models for hardware components and process functions:

WCET

42 cycles
Controllers in UPPAAL

- Gearbox Controller [TACAS’98]
- Bang & Olufsen Power Controller [RTPS’99,FTRTFT’2k]
- SIDMAR Steel Production Plant [RTCSA’99, DSVV’2k]
- Real–Time RCX Control–Programs [ECRTS’2k]
- Terma, Verification of Memory Management for Radar (2001)
- Scheduling Lacquer Production (2005)
- Memory Arbiter Synthesis and Verification for a Radar Memory Interface Card [NJC’05]
- Adapting the UPPAAL Model of a Distributed Lift System, 2007
- Analyzing a χ model of a turntable system using Spin, CADP and Uppaal, 2006
- Designing, Modelling and Verifying a Container Terminal System Using UPPAAL, 2008
- Model–based system analysis using Chi and Uppaal: An industrial case study, 2008
- Climate Controller for Pig Stables, 2008
- Optimal and Robust Controller for Hydraulic Pump, 2009
(Wireless) Protocols in UPPAAL

- Bang & Olufsen IR Link
- Philips Audio Protocol
- Collision–Avoidance Protocol
- Bounded Retransmission Protocol
- TDMA Protocol
- Multimedia Streams
- ATM ABR Protocol
- Lamport’s Leader Election Protocol
- ABB Fieldbus Protocol
- IEEE 1394 Firewire Root Contention
- Bluetooth Protocol
- Distributed Agreement Protocol
- FlexRay
- CHESS MAC Protocol
- Proprietary WSN, Other Big Danish Company
- MESH Protocol (MAC & Routing), NEOCORTEC

TU Graz, May 2017

Kim Larsen [79]
UPPAAL as a Back–End

- Vooduu: verification of object-oriented designs using Uppaal, 2004
- Formalising the ARTS MPSOC Model in UPPAAL, 2007
- Timed automata translator for Uppaal to PVS
- Component-Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008
- Verification of COMDES–II Systems Using UPPAAL with Model Transformation, 2008
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: The current official release is UPPAAL 4.0.13 (Sep 27, 2010). Compared to version 3, the 4.0 release is the result of over 2.5 years of additional 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).

Since Feb 26 2008, we also distribute development snapshots of the tool. The current version is

License

The UPPAAL tool is free for non-commercial applications in academia only. For commercial applications a commercial license is required. Please see the Download section or www.uppaal.com for more information.

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
Excercises

http://people.cs.aau.dk/~kgl/GRAZ17/

Exercise 1 (Brick Sorter)
Exercise 19 (Train Crossing)
Exercise 2 (Coffee Machine)
Exercise 28 (Jobshop Scheduling)