From **Timed Automata** to **Stochastic Hybrid Games**
Model Checking, Performance Evaluation and Synthesis

Kim G. Larsen
Aalborg University, DENMARK
Aalborg University leads Danish ICT in terms of public investments (33%).
CISS – Center For Embedded Software Systems

Regional ICT Center (2003– )

- 3 research groups
  - Computer Science
  - Control Theory
  - HW/SW– codesign

- 20 Employed
- 25 Associated
- 20 PhD Students
- 50 Industrial projects
- 10 Elite–students
- 65 MDKK

- ARTIST Design
- ARTEMIS
ES are Pervasive

Characteristica:

- Dedicated function
- Complex environment
- SW/HW/Mechanics
- Autonomous
- Resource constrained
  - Energy
  - Bandwidth
  - Memory
  - ...
- Timing constraints
ES are often Safety Critical

How to achieve ES that are:
• correct
• predicable
• dependable
• fault tolerant
• ressource minimal
• cheap

Model-Based Development

300 horse power
100 processors
Model Checking

System Description

Requirement

\[ \forall (\text{req} \Rightarrow \exists ! \text{grant}) \]
\[ \forall (\text{req} \Rightarrow \exists ! \text{grant}, t<30s) \]
\[ \forall (\text{req} \Rightarrow \exists ! \text{grant}, t<30s, c<5$) \]
\[ \forall (\text{req} \Rightarrow \exists ! \text{grant}, t<30s, p>0.90) \]
Synthesis

System Description

Requirement

A □ (req ⇒ A □

A □ (req ⇒ A □ t<30s , p>0.90 grant)

SSFT2015 / Kim G Larsen
Origin of UPPAAL

EPSILON
TCCS
Timed Refinements
Timed Mu-Calculus
Regions
Prolog

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

UPPAAL
Timed Automata
TCTL
Zones
C++ & Java

UP4ALL

CAV Award

1995
2007
2013

SSFT2015 / Kim G Larsen
Contributors

@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 Griffioen, 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 Model Checker

Editor
- Discrete Control
- Concurrency
- Continuous Aspects
- Stochasticity
- Timing Constraints
- Resources

Simulator

Verifier
- Performance Analyses
Overview

- **Timed** Automata & UPPAAL
- **Symbolic** Verification & UPPAAL Engine, Options
- **Priced** Timed Automata and Timed Games
- **Stochastic** Timed Automata
  **Statistical** Model Checking
  **Optimal** Synthesis

(Lecture + Exercise)^4
From **Timed Automata** to **Stochastic Hybrid Games**

Model Checking, Performance Evaluation and Synthesis using UPPAAL

**Kim Guldstrand Larsen**

CISS, Aalborg University, DENMARK

Fifth Summer School on Formal Techniques
May 17 - May 22, 2015
Menlo College, Atherton, CA

Slides (preliminary – will be updated)

1. [Timed Automata and UPPAAL](#)
2. [Symbolic Verification and UPPAAL Engine](#)
3. [Priced Timed Automata and Timed Games](#)
4. [Stochastic Timed Automata and Statistical Model Checking](#)

Material available [here](#)

Exercises available [here](#)
Timed Automata
Real Time Systems

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

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

**Plant**

*Continuous*

**Controller Program**

*Discrete*
A Dumb Light Controller

Diagram:

- Off
- Light
- Bright

States:
- Off
- Light
- Bright

Transitions:
- Press?
- Press?
Timed Automata

ADD a clock

Synchronizing action

Clock Guard Conjunctions of $x \sim n$

$x$: real-valued clock

Off

Reset

press?

Light

$press? \quad x=0$

$press? \quad x \leq 3$

$press? \quad x > 3$

Bright

SSFT2015 / Kim G Larsen
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

Transitions:

- \( (\text{Off}, x=0) \)
- 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
Intelligent Light Controller

Transition diagram:

1. **Off**
   - *x = 0*
   - *press?*
   - *x <= 100*
   - *x = 0*
   - *x = 100*

2. **Light**
   - *x = 0*
   - *x <= 100*
   - *x <= 3 press?*
   - *x = 3*
   - *x > 3 press?*
   - *x = 0*

3. **Bright**
   - *x = 0*
   - *x <= 100*
   - *press?*
   - *x = 0*

Graphical simulation output:

Simulations (1)

Value

- 1.00
- 2.00
- 3.00

Time

- 0.000E+00
- 1.000E+00
- 2.000E+00
- 3.000E+00
- 4.000E+00
- 5.000E+00
- 6.000E+00
- 7.000E+00
- 8.000E+00
- 9.000E+00
- 1.000E+01
UPPAAL Demo

![UPPAAL Diagram](image-url)
Clock Valuations

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

**Set $B(C)$ of clock constraints over $C$**

$B(C)$ 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: $x \leq 3 \land y > 0 \land y - x = 2$
Clock Valuation – Operations

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 ($\nu \models g$)

- $\nu \models x < n \iff \nu(x) < n$
- $\nu \models x \leq n \iff \nu(x) \leq n$
- $\nu \models x = n \iff \nu(x) = n$
- $\vdots$
- $\nu \models x - y < n \iff \nu(x) - \nu(y) < n$
- $\nu \models x - y \leq n \iff \nu(x) - \nu(y) \leq n$
- $\vdots$
- $\nu \models g_1 \land g_2 \iff \nu \models g_1 \text{ and } \nu \models g_2$
Timed Automata – Syntax

Definition

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 \( \ell \xrightarrow{g,a,r} \ell' \) whenever \((\ell, g, a, r, \ell') \in E\).
Let \( A = (L, \ell_0, E, I) \) be a timed automaton.

**Timed transition system generated by \( A \)**

\[
T(A) = (Proc, Act, \{ \xrightarrow{a} \mid a \in Act \}) \text{ where}
\]

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

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

\[
(\ell, v) \xrightarrow{d} (\ell, v+d) \text{ for all } d \in \mathbb{R}_{\geq 0} \text{ s.t. } v \models l(\ell) \text{ and } v+d \models l(\ell)
\]
Timed Automata: Example

Synchronization

Guard

2 <= x &&

x <= 3

a!

Reset

x = 0
Timed Automata: Example

Simulations

2 <= x &&
 x <= 3

a!

x <= 3

x = 0

invariant

guard
Example

Is $L_1$ reachable?
Example

\[ y := 0 \quad a \quad y \leq 2 \]

\[ x := 0 \quad b \quad x \leq 2 \]

\[ y \leq 2, x \geq 4 \]

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

\begin{itemize}
\item \( y := 0 \) \quad a \quad y \leq 2 \quad L_0 \quad b \quad x \leq 2 \quad x := 0 \quad c \quad y \leq 2, x > 4 \quad L_1 \)
\end{itemize}

\begin{align*}
(\ell_0, x = 0, y = 0) & \quad \xrightarrow{1.4} \quad (\ell_0, x = 1.4, y = 1.4)
\end{align*}
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

\[(\ell_0, x = 0, y = 0)\]
\[
\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)
\]
\[
\xrightarrow{a} (\ell_0, x = 1.4, y = 0)
\]
\[
\xrightarrow{1.6} (\ell_0, x = 3.0, y = 1.6)
\]
\[
\xrightarrow{a} (\ell_0, x = 3.0, y = 0)
\]
Networks  Light Controller & User

Transition

( Off, Rest, x=0, y=0 )
delay 20  \rightarrow ( Off, Rest, x=20, y=20 )
press?!  \rightarrow ( Light, Busy, x=0, y=0 )
delay 2   \rightarrow ( Light, Busy, x=2, y=2 )
press?!  \rightarrow ( Bright, Rest, x=0, y=0 )
Network Semantics

\[ T_1 \parallel_X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \parallel_X s_0^2) \]

where

\[ S_1 \xrightarrow{\mu} S_1' \]
\[ S_1 \parallel_X S_2 \xrightarrow{\mu} S_1 \parallel_X S_2' \]

\[ S_2 \xrightarrow{\mu} S_2' \]
\[ S_1 \parallel_X S_2 \xrightarrow{\mu} S_1 \parallel_X S_2' \]

\[ S_1 \xrightarrow{a!} S_1' \]
\[ S_1 \parallel_X S_2 \xrightarrow{\tau} S_1 \parallel_X S_2' \]

\[ S_2 \xrightarrow{a?} S_2' \]
\[ S_1 \parallel_X S_2 \xrightarrow{\tau} S_1 \parallel_X S_2' \]

\[ S_1 \xrightarrow{e(d)} S_1' \]
\[ S_1 \parallel_X S_2 \xrightarrow{e(d)} S_1 \parallel_X S_2' \]

\[ S_2 \xrightarrow{e(d)} S_2' \]
\[ S_1 \parallel_X S_2 \xrightarrow{e(d)} S_1 \parallel_X S_2' \]
Network Semantics

(URGENT synchronization)

\[ T_1 \parallel X T_2 = (S_1 \times S_2, \rightarrow, s_0^1 \parallel X s_0^2) \]

where

\[ S_1 \xrightarrow{\mu} S_1' \]

\[ S_1 \parallel X S_2 \xrightarrow{\mu} S_1 \parallel X S_2' \]

\[ S_2 \xrightarrow{\mu} S_2' \]

\[ S_1 \parallel X S_2 \xrightarrow{\mu} S_1 \parallel X S_2' \]

\[ S_1 \xrightarrow{a!} S_1' \]

\[ S_2 \xrightarrow{a?} S_2' \]

\[ S_1 \parallel X S_2 \xrightarrow{\tau} S_1 \parallel X S_2' \]

\[ S_1 \xrightarrow{e(d)} S_1' \]

\[ S_2 \xrightarrow{e(d)} S_2' \]

\[ S_1 \parallel X S_2 \xrightarrow{e(d)} S_1 \parallel X S_2' \]

\[ \forall d' < d, \forall u \in UAct: \]

\[ \neg (s_1 \xrightarrow{e(d')} u? \land s_2 \xrightarrow{e(d')} u!) \]
UPPAAL

First Introduction
Light Control Interface

press? \(d\) release? \(\rightarrow\) touch! \(0.5 \leq d \leq 1\)
press? \(1\) \(\rightarrow\) startH!
press? \(d\) release? \(\rightarrow\) endH! \(d > 1\)

SSFT2015 / Kim G Larsen
Light Control Interface

User

SSFT2015 / Kim G Larsen
Light Control Network

SSFT2015 / Kim G Larsen
Full Light Controller
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.
A Real Timed System

The Plant
Conveyor Belt & Bricks

Controller Program
LEGO MINDSTORM

What is suppose to happen?
Exercise: Design Controller so that only black boxes are being pushed out
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
GLOBAL DECLARATIONS:
const int ctime = 75;

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

chan eject, ok;
urgent chan blck, red, remove, go;
From RCX to UPPAAL

- Model includes Round–Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000
The Production Cell

Course at DTU, Copenhagen

Production Cell
UPPAAL

Modeling & Specification
Train Crossing

Safe → Approaching → Crossing → Safe

River

Bridge

tracks

0 20 3 – 5

Time

SSFT2015 / Kim G Larsen
Train Crossing

Stop the train while it still stoppable!

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

SSFT2015 / Kim C Larsen
Train Crossing

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

Stopped → Restarted

River

Bridge

tracks

0 10 20

Time

3 – 5

7 – 15
Train Crossing

Safe → Approaching

Add **timing** + synchronization

Stopped

Crossing → Restarted

Safe

Add timing + synchronization

SSFT2015 / Kim G Larsen
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 are allowed (and arrays of these).
Editor

- Unlimited undo and redo
- Syntax and bracket highlighting
- Rectangular selection
- Customization of colors
- Tooltip
- Hiding of information
- Improved help menu with search component

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
- Gannt Charts

SSFT2015 / Kim G Larsen
Verifier

- Exhaustive & automatic checking of requirements
  - including validating, safety, liveness, bounded liveness and response properties
- generation of debugging information for visualisation in simulator.
- performance properties, e.g. probabilistic and expectation.
- plot composer
Bang & Olufsen (1997)

- Bug known to exist for 10 years
- Ill-described: 2,800 loc + 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.
- Follow-up project.

Arne Skou, Klaus Havelund

1st RTSS'97 talk, Klaus Havelund
GearControl@Initiate $\rightarrow_{\leq 1500} (\text{ErrStat} = 0) \Rightarrow \text{GearControl@GearChanged}$ \hspace{5cm} (1)

GearControl@Initiate $\rightarrow_{\leq 1000}$

$\quad (\text{ErrStat} = 0 \land \text{UseCase} = 0) \Rightarrow \text{GearControl@GearChanged}$ \hspace{5cm} (2)

Clutch@ErrorClose $\rightarrow_{\leq 200} \text{GearControl@CCloseError}$ \hspace{5cm} (3)

Clutch@ErrorOpen $\rightarrow_{\leq 200} \text{GearControl@COpenError}$ \hspace{5cm} (4)

GearBox@ErrorIdle $\rightarrow_{\leq 350} \text{GearControl@GSetError}$ \hspace{5cm} (5)

GearBox@ErrorNeu $\rightarrow_{\leq 200} \text{GearControl@GNeuError}$ \hspace{5cm} (6)

\[ \text{Inv} (\text{GearControl@CCloseError} \Rightarrow \text{Clutch@ErrorClose} ) \hspace{5cm} (7) \]

\[ \text{Inv} (\text{GearControl@COpenError} \Rightarrow \text{Clutch@ErrorOpen} ) \hspace{5cm} (8) \]

\[ \text{Inv} (\text{GearControl@GSetError} \Rightarrow \text{GearBox@ErrorIdle} ) \hspace{5cm} (9) \]
Case Studies: Controllers

- 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
Case Studies: Protocols

- Philips Audio Protocol [HS’95, CAV’95, RTSS’95, CAV’96]
- Bounded Retransmission Protocol [TACAS’97]
- Bang & Olufsen Audio/Video Protocol [RTSS’97]
- TDMA Protocol [PRFTS’97]
- Lip-Synchronization Protocol [FMICS’97]
- ATM ABR Protocol [CAV’99]
- ABB Fieldbus Protocol [ECRTS’2k]
- Distributed Agreement Protocol [Formats05]
- Leader Election for Mobile Ad Hoc Networks [Charme05]

- Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal, 2006
- Formalizing SHIM6, a Proposed Internet Standard in UPPAAL, 2007
- Verifying the distributed real-time network protocol RTnet using Uppaal, 2007
- Analysis of the Zeroconf protocol using UPPAAL, 2009
- Model Checking the FlexRay Physical Layer Protocol, 2010
Using UPPAAL as 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.

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.
LAB–Exercises

http://people.cs.aau.dk/~kgl/SSFT2015/

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