Modelling, Verification and Controller Synthesis for Real-Time Systems

Kim Guldstrand Larsen
Collaborators

@UPPsala
- Wang Yi
- Johan Bengtsson
- Paul Pettersson
- Fredrik Larsson
- Elena Fersman
- Tobias Amnell

@AALborg
- Kim G. Larsen
- Gerd Behrman
- Arne Skou
- Alexandre David
- Emmanuel Fleury
- Didier Lime
- Brian Nielsen
- Marius Mesciowski
- Thomas Hune

@Elsewhere
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

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

- **Real Time Verification**
  - Modelling, Timed Automata
  - Applications (demo)
  - Engine & Data Str.

- **Real Time Contr. Synth.**
  - Scheduling, Reachability
  - Optimality
  - Applications (demo)
  - Data Str.

- Concluding Remarks
Timed Automata

(Alur, Dill 1990)

A single TRAIN

= Finite State Control
+ Real Valued Clocks
+ Discrete Variables

Discrete Variables
(bounded integers, arrays, records)

Clock Updates

Invariants

Guards

Synchronizations

Finite State Control

Real Valued Clocks

Discrete Variables

Alur, Dill 1990
Timed Automata \textit{(semantics)}

Clocks: \( x, y \)

Guard
Boolean combination of integer bounds on clocks and clock-differences.

Reset
Action performed on clocks

State
\((\text{location}, x=v, y=u)\) where \(v,u\) are in \(\mathbb{R}\)

Transitions
\[
\begin{align*}
(n, x=2.4, y=3.1415) & \xrightarrow{a} (m, x=0, y=3.1415) \\
(n, x=2.4, y=3.1415) & \xrightarrow{e(1.1)} (n, x=3.5, y=4.2415)
\end{align*}
\]
Timed Automata (semantics)

Invariants

Clocks: \( x, y \)

Transitions

\[ (n, x=2.4, y=3.1415) \quad e(3.2) \]

\[ (n, x=3.5, y=4.2415) \]

Invariants ensure progress!!
The UPPAAL Model

= Networks of Timed Automata + Integer Variables + C-code + ..

Example transitions

\[(l1, m1, ..., x=2, y=3.5, i=3,....) \xrightarrow{\tau} (l2, m2, ..., x=0, y=3.5, i=7,....)\]

\[(l1, m1, ..., x=2.2, y=3.7, I=3,....)\]
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” Real Time System

What was supposed to happen?
First UPPAAL model
Sorting of Lego Boxes

Boxes

Conveyor Belt

Controller

MAIN
PUSH

Exercises: Design Controller so that only red boxes are being pushed out

Ken Tindell
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);
    }
}
```
UPPAAL Demo
The Production Cell in LEGO

Course at DTU, Copenhagen

Production Cell

Rasmus Crüger Lund
Simon Tune Riemanni
Case Studies: Protocols

- Philips Audio Protocol [HS’95, CAV’95, RTSS’95, CAV’96]
- Collision-Avoidance Protocol [SPIN’95]
- Bounded Retransmission Protocol [TACAS’97]
- Bang & Olufsen Audio/Video Protocol [RTSS’97]
- TDMA Protocol [PRFTS’97]
- Lip-Synchronization Protocol [FMICS’97]
- Multimedia Streams [DSVIS’98]
- ATM ABR Protocol [CAV’99]
- ABB Fieldbus Protocol [ECRTS’2k]
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]
- Experimental Batch Plant (2000)
- RCX Production Cell (2000)
Application

Leader Election (2003-2005)

All,

Thanks for the spec. It seems to run fine. As expected, it's 2 or 3 orders of magnitude faster than TLC. I'm wondering if your algorithms could be used for checking specs written in a higher level language like TLA+.
THE UPPAAL ENGINE

Symbolic Reachability Checking
Zones
From infinite to finite

State
(n, x=3.2, y=2.5)

Symbolic state (set)
(n, 1≤x≤4,1≤y≤3)

Zone: conjunction of x-y≤n, x=>n
Symbolic Transitions

Thus \((n, 1 \leq x \leq 4, 1 \leq y \leq 3) = a \Rightarrow (m, 3 < x, y = 0)\)
Canonical Datastructure for Zones

*Difference Bounded Matrices*

Bellman’58, Dill’89

Shortest Path Closure $O(n^3)$

$x_1-x_2 \leq 4$
$x_2-x_1 \leq 10$
$x_3-x_1 \leq 2$
$x_2-x_3 \leq 2$
$x_0-x_1 \leq 3$
$x_3-x_0 \leq 5$
New Canonical Datastructure

Minimal collection of constraints

\[
\begin{align*}
-x_1 + x_2 & \leq 4 \\
x_2 - x_1 & \leq 10 \\
x_3 - x_1 & \leq 2 \\
x_2 - x_3 & \leq 2 \\
x_0 - x_1 & \leq 3 \\
x_3 - x_0 & \leq 5 \\
n & \geq 4
\end{align*}
\]

Shortest Path Closure \(O(n^3)\)

Shortest Path Reduction \(O(n^3)\)

Space worst \(O(n^2)\) practice \(O(n)\)

RTSS 1997
Clock Difference Diagrams

= Binary Decision Diagrams + Difference Bounded Matrices

- Nodes labeled with differences
- Maximal sharing of substructures (also across different CDDs)
- Maximal intervals
- Linear-time algorithms for set-theoretic operations.

CAV99
D-UPPAAL

File information:

Model: AN ASTRONOMICALLY BIG MODEL
Query: A VERY INTERESTING QUESTION

Model checking options

Search order: ○ breadth first ○ width first
State space reduction: ○ none ○ conservative ○ aggressive
State space representation: ○ DBM ○ compact data structure ○ under approximation ○ over approximation
New syntax: ○ no ○ yes

Distribution options

Number of CPUs: ○ 1 ○ 5 ○ 10 ○ 15 ○ 20 ○ 25 ○ 30 ○ 35 ○ 49

Run options

Max walltime (minutes): ○ 1 ○ 5 ○ 15 ○ 30 ○ 60 ○ 120 ○ 240

Contact information

Email: kg@cs.au.dk

Submit Query  Reset
Optimal Scheduling & Controller Synthesis

Jacob Illum, Gerd Behrmann, Kim Larsen
Thomas Hune, Ansgar Fehnker, Judi Romijn, Frits Vaandrager, Ed Brinksma, Paul Pettersson
Patricia Bouyer, Franck Cassez, Emmanuel Fleury

V H S
Verification of Hybrid Systems

AMETIST
advanced methods for timed systems
Rush Hour

OBJECTIVE: Get your CAR out
Rush Hour

\begin{align*}
\text{no} &= 3, \\
i &\geq 2 \\
i &= 0, \\
\text{no} &= \text{no} + 1, \\
\text{pos} &= 2
\end{align*}

\begin{align*}
\text{pos} + 2 &< \text{N} + 1, \\
\text{BOARD}[3][\text{pos} + 2] &= 0
\end{align*}

\begin{align*}
\text{BOARD}[3][\text{pos} - 1] &= 1, \\
\text{BOARD}[3][\text{pos} + 2 - 1] &= 0, \\
\text{pos} &= \text{pos} - 1
\end{align*}

\begin{align*}
\text{BOARD}[3][\text{pos} + 2] &= 1, \\
\text{BOARD}[3][\text{pos}] &= 0, \\
\text{pos} &= \text{pos} + 1
\end{align*}
Cost Optimality

leftHand

rightHand

Red

5
9
3
17

ICE-TCS, Reykjavik U, 2005
Kim G. Larsen
Steel Production Plant

- A. Fehnker
- Hune, Larsen, Pettersson
- Case study of Esprit-LTR project 26270 VHS
- Physical plant of SIDMAR located in Gent, Belgium.
- Part between blast furnace and hot rolling mill.

**Objective:** model the plant, obtain schedule and control program for plant.
Steel Production Plant

Input: sequence of steel loads ("pigs").

Load follows Recipe to become certain quality, e.g:
start; T1@10; T2@20; T3@10; T2@10;
end within 120.

Output: sequence of higher quality steel.
Steel Production Plant

**Input:** sequence of steel loads (“pigs”).

Load follows **Recipe** to become certain quality, e.g:

- start; **T1@10; T2@20; T3@10; T2@10;**
- end within 120.

**Output:** sequence of higher quality steel.

\[ \sum = 107 \]
Steel Production Plant

**Input:** sequence of steel loads ("pigs").

Load follows **Recipe** to obtain certain quality, e.g:
start; $T_1@10$; $T_2@20$; $T_3@10$; $T_2@10$; end within 120.

**Output:** sequence of higher quality steel.

$\sum = 127$
A single load (part of)

Crane B
### Experiment

<table>
<thead>
<tr>
<th>n</th>
<th>BFS</th>
<th>DFS</th>
<th>BSH</th>
<th>BFS</th>
<th>DFS</th>
<th>BSH</th>
<th>BFS</th>
<th>DFS</th>
<th>BSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.1</td>
<td>0.9</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>2</td>
<td>18.4</td>
<td>36.4</td>
<td>0.1</td>
<td>1</td>
<td>0.1</td>
<td>1.1</td>
<td>-</td>
<td>-</td>
<td>4.4</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>3.2</td>
<td>6.5</td>
<td>3.4</td>
<td>1.4</td>
<td>4.4</td>
<td>-</td>
<td>72.4</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>8.2</td>
<td>4.6</td>
<td>1.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>10.2</td>
<td>5.5</td>
<td>2.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>13.3</td>
<td>25.3</td>
<td>16.1</td>
<td>9.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>-</td>
<td>31.6</td>
<td>51.2</td>
<td>48.1</td>
<td>22.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>-</td>
<td>61.8</td>
<td>89.6</td>
<td>332</td>
<td>46.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>-</td>
<td>-</td>
<td>104</td>
<td>144</td>
<td>87.2</td>
<td>83.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
<td>-</td>
<td>166</td>
<td>216</td>
<td>124.2</td>
<td>136</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>209</td>
<td>250</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **BFS** = breadth-first search, **DFS** = depth-first search, **BSH** = bit-state hashing.
- “-“ = requires >2h (on 450MHz Pentium III), >256 MB, or suitable hash-table size was not found.
- **System size**: 2\(n+5\) automata and 3\(n+3\) clocks, if \(n=35\): 75 automata and 108 clocks.
- **Schedule generated for \(n=60\) on Sun Ultra with 2x300MHz with 1024MB in 2257s**.
LEGO Plant Model

- LEGO RCX Mindstorms.
- Local controllers with control programs.
- IR protocol for remote invocation of programs.
- Central controller.

Synthesis
LEGO Plant Model

Belt/Machine Unit.
Example: Aircraft Landing

- **E** earliest landing time
- **T** target time
- **L** latest time
- **e** cost rate for being early
- **l** cost rate for being late
- **d** fixed cost for being late

Planes have to keep separation distance to avoid turbulences caused by preceding planes.
Example: Aircraft Landing

Planes have to keep separation distance to avoid turbulences caused by preceding planes.

Modelled as a **PRICED** Timed Automata

[HSIC01, TACAS01, CAV01, TACAS04]

- 4 earliest landing time
- 5 target time
- 9 latest time
- 3 cost rate for being early
- 1 cost rate for being late
- 2 fixed cost for being late
# Aircraft Landing

## Source of examples:

Baesley et al’2000

<table>
<thead>
<tr>
<th>problem instance</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of planes</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>44</td>
</tr>
<tr>
<td>number of types</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>optimal value</td>
<td>700</td>
<td>1480</td>
<td>820</td>
<td>2520</td>
<td>3100</td>
<td>24442</td>
<td>1550</td>
</tr>
<tr>
<td>explored states</td>
<td>481</td>
<td>2149</td>
<td>920</td>
<td>5693</td>
<td>15069</td>
<td>122</td>
<td>662</td>
</tr>
<tr>
<td>cpu time (secs)</td>
<td>4.19</td>
<td>25.30</td>
<td>11.05</td>
<td>87.67</td>
<td>220.22</td>
<td>0.60</td>
<td>4.27</td>
</tr>
</tbody>
</table>

| 2               | 90 | 210 | 60 | 640 | 650 | 554 | 0  |
| explored states | 1218| 1797| 669| 28821| 47993| 9035| 92 |
| cpu time (secs) | 17.87| 39.92| 11.02| 755.84| 1085.08| 123.72| 1.06|

| 3               | 0  | 0   | 0  | 130 | 170 | 0   | N/A |
| explored states | 24 | 46  | 84 | 207715| 189602| 62  | N/A |
| cpu time (secs) | 0.36| 0.70| 1.71| 14786.19| 12461.47| 0.68| N/A |

| 4               | N/A| N/A| N/A| 0  | 0  | N/A| N/A|
| explored states | N/A| N/A| N/A| 65 | 64 | N/A| N/A|
| cpu time (secs) | 1.97| 1.53| N/A| N/A| N/A| N/A| N/A|

ICE-TCS, Reykjavik U, 2005

Kim G. Larsen
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 but we have a license agreement that all users must fill in before downloading and using the tool. 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 group at Yahoo! Groups intended for users of the tool. To join the group, email uppaal-subscribe@yahoogroups.com, to post use uppaal@yahoogroups.com. To email the development team directly, please use buc-uppaal@docs.uu.dk for bug reports and uppaal@docs.uu.dk otherwise.

Figure 1: UPPAAL on screen.

The current official release is UPPAAL 3.4.8 (Feb 17, 2005). A development snapshot of UPPAAL 3.5.4 (Feb 4, 2005) is also available. For more information about UPPAAL version 3.4, we refer to this press release.
Welcome!
UPPAAL CORA
UPPAAL for Cost Optimal Reachability Analysis

Welcome!
UPPAAL TRON
UPPAAL for Testing Realtime Systems Online

Latest News
UPPAAL Got New Home Page
25 Jan 2005

Latest News
New name and home
1 Mar 2004
Version 1.3.1 released under new name UPPAAL TRON and with a new home page similar to other UPPAAL pages.

UPPAAL CORA 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.).

UPPAAL CORA is a branch of UPPAAL for Cost Optimal Reachability Analysis. The CORA team has been a part of the VTH and AMETIST projects. The CORA methodology provides an easy-to-use framework for checking of timed automata, UPPAAL CORA uses an extensively annotated model for powerful verification. LPTA allows you to annotate the model with the notions of time, certain situations or the cost of particular actions, thereby matching goal conditions.

UPPAAL CORA has been used in a number of case studies and demonstration available at the case study page of this site. If you come up with interesting applications, we are interested in hearing what you did.

Due to different internal data structures, UPPAAL CORA provides two versions:

- A version for the simplified case of time optimal reachability analysis (TODA).
- A version for the full language of LPTA.

UPPAAL TRON is a testing tool, based on UPPAAL engine, suited for black-box conformance testing of timed systems, mainly targeted for embedded software commonly found in various controllers. By online we mean that tests are defined, executed and checked simultaneously while maintaining the connection to the system in real-time.

Here is a screen-shot of demo example where TRON is attached to a smart lamp light controller simulator in Java via TCP/IP socket connection.