Real–Time Model Checking

Patricia Bouyer-Decitre
Kim G. Larsen
Nicolas Markey
Timed Automata
.. and Prices and Games

Patricia Bouyer–Decitre
Kim G. Larsen
Nicolas Markey
QUANTITATIVE Model Checking

System Description

Requirement

A\square ( \text{req} \Rightarrow A\lozenge )

A\square ( \text{req} \Rightarrow A\lozenge_{t<30s} \text{grant} )

A\square ( \text{req} \Rightarrow A\lozenge_{t<30s,c<5$} \text{grant} )

A\square ( \text{req} \Rightarrow A\lozenge_{t<30s,p>0.90} \text{grant} )

Time Cost Probability

Debugging Information

No!

Yes

Prototypes

Executable Code

Test sequences

Kim Larsen [3]
Synthesis

System Description

Requirement

\[ A \Box ( \text{req} \Rightarrow A \Diamond ) \]
\[ A \Box ( \text{req} \Rightarrow A \Diamond_{t<30s} \text{grant}) \]
\[ A \Box ( \text{req} \Rightarrow A \Diamond_{t<30s,c<5} \text{grant}) \]
\[ A \Box ( \text{req} \Rightarrow A \Diamond_{t<30s,p>0.90} \text{grant}) \]

Debugging Information

No!

Yes

Control Strategy

QMC, PhD School, March 3, 2010

Kim Larsen [4]
Overview

- Introduction to Timed Automata
- Decidability and undecidability results
- Timed Temporal Logics
- UPPAAL (hands-on)
- Timed Games
- Priced Timed Automata
- Open Problems
Timed Automata
UPPAAL (contributors)

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

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

@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
QMC, PhD School, March 3, 2010
Kim Larsen [8]
Timed Automata [Alur & Dill’89]

ADD a clock \( x \)

\( x: \) real-valued clock

Synchronizing action

Clock Guard Conjunctions of \( x \sim n \)

QMC, PhD School, March 3, 2010
Kim Larsen [10]
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)

QMC, PhD School, March 3, 2010
Kim Larsen [12]
Intelligent Light Controller

Transitions:

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

Note: \( (\text{Light}, x=0) \) delay 103 \( \rightarrow \)

Invariants ensures progress

QMC, PhD School, March 3, 2010

Kim Larsen [13]
Timed Automata (formally)

Constraints

Definition
Let $X$ be a set of clock variables. The set $\mathcal{B}(X)$ of clock constraints $\phi$ is given by the grammar:

$$\phi ::= x \leq c \mid c \leq x \mid x < c \mid c < x \mid \phi_1 \land \phi_2$$

where $c \in \mathbb{N}$ (or $\mathbb{Q}$).
Timed Automata (formally)

Clock Valuations and Notation

Definition
The set of clock valuations, $\mathbb{R}^C$ is the set of functions $C \rightarrow \mathbb{R}_{\geq 0}$ ranged over by $u, v, w, \ldots$.

Notation
Let $u \in \mathbb{R}^C$, $r \subseteq C$, $d \in \mathbb{R}_{\geq 0}$, and $g \in \mathcal{B}(X)$ then:

- $u + d \in \mathbb{R}^C$ is defined by $(u + d)(x) = u(x) + d$ for any clock $x$.

- $u[r] \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \not\in r$.

- $u \models g$ denotes that $g$ is satisfied by $u$.  

QMC, PhD School, March 3, 2010

Kim Larsen [15]
Timed Automata (formally)

Timed Automata

Definition
A timed automaton $A$ over clocks $C$ and actions $Act$ is a tuple $(L, l_0, E, I)$, where:

- $L$ is a finite set of locations
- $l_0 \in L$ is the initial location
- $E \subseteq L \times \mathcal{B}(X) \times Act \times \mathcal{P}(C) \times L$ is the set of edges
- $I : L \rightarrow \mathcal{B}(X)$ assigns to each location an invariant
Timed Automata (formally)

Semantics

Definition
The semantics of a timed automaton $A$ is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state $(l_0, u_0)$ and with the following transitions:

- $(l, u) \xrightarrow{\epsilon(d)} (l, u + d)$ iff $u \in I(l)$ and $u + d \in I(l)$,
- $(l, u) \xrightarrow{a} (l', u')$ iff there exists $(l, g, a, r, l') \in E$ such that
  - $u \models g$,
  - $u' = u[r]$, and
  - $u' \in I(l')$

$u_0(x) = 0$ for all $x \in C$
Example

![Example Diagram]

QMC, PhD School, March 3, 2010

Kim Larsen [18]
Example

\begin{align*}
    y &:= 0 \\
    y &\leq 2 \\
    y &\leq 2, x = 4 \\
    x &:= 0 \\
    x &\leq 2 \\
    (\ell_0, x = 0, y = 0)
\end{align*}
Example

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

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

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

\[ ^a \rightarrow (\ell_0, x = 1.4, y = 0) \]
Example

\[ (\ell_0, x = 0, y = 0) \]
\[ \xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4) \]
\[ \xrightarrow{a} (\ell_0, x = 1.4, y = 0) \]
\[ \xrightarrow{1.6} (\ell_0, x = 3.0, y = 1.6) \]
\[ \xrightarrow{a} (\ell_0, x = 3.0, y = 0) \]
Light Control Interface
Light Control Interface

- Press? \( d \) Release? \( \rightarrow \) Touch! \( \frac{1}{2} \leq d \leq 1 \)
- Press? \( 1 \) Release? \( \rightarrow \) Starthold!
- Press? \( d \) Release? \( \rightarrow \) Endhold! \( d > 1 \)

Press? 0.2 Release? \( \ldots \) Press? 0.7 Release? \( \ldots \) Press? 1.0 2.4 Release? \( \ldots \)

\( \emptyset \) Touch! Starthold! Endhold!

QMC, PhD School, March 3, 2010
Light Control Interface

Interface

User

Press?

Release?

Endhold!

Press?

Release?

Starthold!

Touch!

Touch!

Control Program

Dim

Switch

L<Max

x:=delay

on:=0

touch?

x:=0

L:=OL

L:=OL

L:=OL

L++/L-:=0

QMC, PhD School, March 3, 2010
Light Control Network
Task Graph Scheduling
Semantics:

(Idle, Init, B=0, x=0)

\[ \text{d(3.1415)} \rightarrow (\text{Idle}, \text{Init}, B=0, x=3.1415) \]

use \rightarrow (\text{InUse}, \text{Using}, B=6, x=0)

d(6) \rightarrow (\text{InUse}, \text{Using}, B=6, x=6)

done \rightarrow (\text{Idle}, \text{Done}, B=6, x=6)
Task Graph Scheduling – Example

Compute:
\[(D \ast (C \ast (A + B)) + ((A + B) + (C \ast D)))\]

using 2 processors

P1 (fast)  
P2 (slow)

time

13 pico-sec!!
Task Graph Scheduling – Example

Compute:
\[(D \times (C \times (A + B)) + ((A + B) + (C \times D))\]
using 2 processors

P1 (fast)
P2 (slow)

12 pico-sec
OPTIMAL!!
Task Graph Scheduling

\[ M = \{M_1, M_2\} \]
Task Graph Scheduling

\[ E<> \text{(Task1.End and ... and Task7.End)} \]
### Experimental Results

<table>
<thead>
<tr>
<th>name</th>
<th>#tasks</th>
<th>#chains</th>
<th># machines</th>
<th>optimal</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>437</td>
<td>125</td>
<td>4</td>
<td>1178</td>
<td>1182</td>
</tr>
<tr>
<td>000</td>
<td>452</td>
<td>43</td>
<td>20</td>
<td>537</td>
<td>537</td>
</tr>
<tr>
<td>018</td>
<td>730</td>
<td>175</td>
<td>10</td>
<td>700</td>
<td>704</td>
</tr>
<tr>
<td>074</td>
<td>1007</td>
<td>66</td>
<td>12</td>
<td>891</td>
<td>894</td>
</tr>
<tr>
<td>021</td>
<td>1145</td>
<td>88</td>
<td>20</td>
<td>605</td>
<td>612</td>
</tr>
<tr>
<td>228</td>
<td>1187</td>
<td>293</td>
<td>8</td>
<td>1570</td>
<td>1574</td>
</tr>
<tr>
<td>071</td>
<td>1193</td>
<td>124</td>
<td>20</td>
<td>629</td>
<td>634</td>
</tr>
<tr>
<td>271</td>
<td>1348</td>
<td>127</td>
<td>12</td>
<td>1163</td>
<td>1164</td>
</tr>
<tr>
<td>237</td>
<td>1566</td>
<td>152</td>
<td>12</td>
<td>1340</td>
<td>1342</td>
</tr>
<tr>
<td>231</td>
<td>1664</td>
<td>101</td>
<td>16</td>
<td>t.o.</td>
<td>1137</td>
</tr>
<tr>
<td>235</td>
<td>1782</td>
<td>218</td>
<td>16</td>
<td>t.o.</td>
<td>1150</td>
</tr>
<tr>
<td>233</td>
<td>1980</td>
<td>207</td>
<td>19</td>
<td>1118</td>
<td>1121</td>
</tr>
<tr>
<td>294</td>
<td>2014</td>
<td>141</td>
<td>17</td>
<td>1257</td>
<td>1261</td>
</tr>
<tr>
<td>295</td>
<td>2168</td>
<td>965</td>
<td>18</td>
<td>1318</td>
<td>1322</td>
</tr>
<tr>
<td>292</td>
<td>2333</td>
<td>318</td>
<td>3</td>
<td>8009</td>
<td>8009</td>
</tr>
<tr>
<td>298</td>
<td>2399</td>
<td>303</td>
<td>10</td>
<td>2471</td>
<td>2473</td>
</tr>
</tbody>
</table>

Symbolic A*  
Branch-&-Bound  
60 sec  

Abbeddaïm, Kerbaa, Maler

QP, PhD School, March 3, 2010

Kim Larsen [33]
Brick Sorting
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 Timed System

The Plant
Conveyor Belt & Bricks

Controller Program
LEGO MINDSTORM

QMC, PhD School, March 3, 2010
First UPPAAL model

Sorting of Lego Boxes

Exercise: Design **Controller** so that **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;
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 $\chi$ 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
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

QMC, PhD School, March 3, 2010
Kim Larsen [43]