Validation, Synthesis and Performance Evaluation of Embedded Systems using UPPAAL

Kim Guldstrand Larsen
Timed Automata
UPPAAL (1995– )

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

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

@Elsewhere

yyymm

y = 3.4322x^2 - 28.247x + 749.99

UPPAAL Downloads

ARTIST PhD School, Beijing, 2011

Kim Larsen [3]
Real Time Systems

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

ARTIST PhD School, Beijing, 2011

Kim Larsen [5]
Timed Automata

[Alur & Dill’89]

- Synchronizing action
- Reset

-钟

- ADD a clock \( x \)

- x: real-valued clock

- Clock Guard
  Conjunctions of \( x = n \)

- x = 0
- x ≤ 3
- x > 3

- Off
- Press?

- Light
- Press?

- 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)

Off

press?  x=0

x=0

x==100

Light

x<=100

x<=3  press?

x=0

x>3

press?

x=0

x==100

Bright

x<=100

x>3  press?

x=0

x=0
Intelligent Light Controller

Transitions:

- (Off, x=0)
- delay 4.32 → (Off, x=4.32)
- press? → (Light, x=0)
- delay 4.51 → (Light, x=4.51)
- press? → (Light, x=0)
- delay 100 → (Light, x=100)
- τ → (Off, x=0)

Note:

(Off, x=0) delay 103 →

Invariants ensures progress

ARTIST PhD School, Beijing, 2011

Kim Larsen [9]
Constraints

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

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

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 $u$. 

ARTIST PhD School, Beijing, 2011

Kim Larsen [11]
Timed Automata (formally)

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
- $E \subseteq L \times B(X) \times Act \times \mathcal{P}(C) \times L$ is the set of edges
- $I : L \rightarrow B(X)$ 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
  - $u' \in I(l')$

\[ *u_0(x) = 0 \text{ for all } x \in C \]
Example

Is L1 reachable?
Example

\[ y = 0, x = 0 \]

\[ y < 2, x < 2 \]

\[ y < 2, x = 4 \]

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

\[ y := 0 \quad a \quad y \leq 2 \quad L_0 \quad x := 0 \quad b \quad x \leq 2 \quad c \quad y \leq 2, x = 4 \quad L_1 \]

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

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

\[\frac{1.4}{\text{a}} (\ell_0, x = 1.4, y = 1.4)\]

\[\text{a} \rightarrow (\ell_0, x = 1.4, y = 0)\]
(ℓ₀, x = 0, y = 0)

\[
\begin{align*}
\rightarrow^{1.4} & \quad (ℓ₀, x = 1.4, y = 1.4) \\
\rightarrow^{a} & \quad (ℓ₀, x = 1.4, y = 0) \\
\rightarrow^{1.6} & \quad (ℓ₀, x = 3.0, y = 1.6) \\
\rightarrow^{a} & \quad (ℓ₀, x = 3.0, y = 0)
\end{align*}
\]
Networks  Light Controller & User

Transition

- (Off, Rest, x=0, y=0) → (Off, Rest, x=20, y=20)
- (Light, Busy, x=0, y=0) → (Light, Busy, x=2, y=2)
- (Bright, Rest, x=0, y=0) → (Bright, Rest, x=0, y=0)
Network Semantics

\[ T_1 \parallel T_2 = (S_1 \times S_2, \rightarrow, S_0 \parallel_X S_0^2) \]

where

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

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

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

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

(URGENT synchronization)

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

\[ S_1 \xrightarrow{\mu^1} S_1' \quad S_2 \xrightarrow{\mu^2} S_2' \]

where

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

\[ S_1 \xrightarrow{a!} S_1' \quad S_2 \xrightarrow{a?} S_2' \]

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

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

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

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

Kim Larsen [21]

ARTIST PhD School, Beijing, 2011
Light Control Interface
Light Control Interface

- press? d release? → touch! 0.5 ≤ d ≤ 1
- press? 1 → starthold!
- press? d release? → endhold! d > 1

User

Press? 0.2 release? ... press? 0.7 release? ... press? 1.0 2.4 release? ...

Ø touch! starthold! endhold!

ARTIST PhD School, Beijing, 2011
Light Control Network

![Diagram of Light Control Network]
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
First UPPAAL model

Sorting of Lego Boxes

Exercise: Design **Controller** so that **black** boxes are being pushed out.
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);
  }
}

int active;
int DELAY;
int 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
Control Tasks & Piston

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 – 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

Production Cell

Rasmus Crüger Lund
Simon Tune Riemanni

ARTIST PhD School, Beijing, 2011
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
- 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