Overview of Timed Automata and UPPAAL
Table of Contents

- Timed Automata
  - Introduction
  - Example
  - The Query Language

- UPPAAL
  - Introduction
  - Example
  - Editor
  - Simulator
  - Verifier

- Conclusions
Introduction to Timed Automata
It’s time for timed automata: why? what? how?

- Timing behaviour is critical in many applications:
- Examples:
  - Radiation period in X-ray machine
  - Clock for a chess game
  - Car Engine Controller
  - Heart pacemaker
  - Industrial Process Controller
Model checking approach

- system
- requirements

- system model
  - formalize
  - DTMCs
  - CTMCs
  - SDF
  - TAs

- property
  - formalize
  - PCTL
  - CSL
  - SDF logic
  - TCTL

- model checker
  - UPPAAL

- debugging
  - Violated + Counterexample

- Tool usage
  - Uppaal modeling + analysis

- systems with timing behavior:
  - deadlines
  - processing-speed
  - communication delays, ...

- against timing requirements:
  - can state be reached within 5 sec?
  - is task always completed in 1ms?
  - min/max time to finish all tasks?
  - ...

UNIVERSITY OF TWENTE.
Timed Automata: Introduction

- Timed automata are finite state machines with *clocks*.
  - Clocks are variables which can evaluate to a *real number*.
  - Clocks can be defined in each automaton to measure the time progress.
  - All clocks evolve at the *same pace* to represent the global progress of time.
  - The actual value of a clock can be either tested or reset (*not assigned*).
Timed Automata: small example

Light bulb
- single click: normal light
- double click: bright light

Clock constraints
- guards: transition may be taken (enabling conditions)
- invariants: transition must be taken (deadlines); ensure progress
- syntax:
  - $x - y \leq c$
  - $x - y < c$
  - $c$ integer

Terminology + uses
- click $x := 0$
- click $x \geq 5$
- click $x < 5$
- click $x \leq 300$
- click $x = 300$
- (clock) guard
- (clock) reset
- (location) invariant
- action / label
- location / node

Edge

(0,0)
Timed Automata: small example

Light bulb
- single click: normal light
- double click: bright light

Terminology + uses
- click \( x \geq 5 \)
- click \( x := 0 \)
- click \( x < 5 \)
- click \( x = 300 \)
- click \( x \leq 300 \)

Clocks
- All clocks progress with same speed
- Syntax invariants \( x - y \leq c \)
The Query Language

- A model checker verifies the model w.r.t requirement specification.

- UPPAAL uses a simplified version of Timed Computation Tree Logic (TCTL).

- It comprises of path formulae and state formulae.

- State formulae describes individual states.

- Path formulae quantifies over path or traces.
  - Can be classified into reachability, safety and liveness.

- Example
  - Along all the paths, is X-ray machine eventually stopped within 5 minutes starting at the initial state?
State Formulae

- State formula evaluates for a state.

- Example,
  - \( i == 7 \) is true in a state whenever \( i \) equals 7

  - To test whether a particular automaton is in a given location using expression “P.1”.

- Deadlock is checked using keyword “deadlock” in UPPAAL.
Path Formulae

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E \Diamond p$</td>
<td>there exists a path where $p$ eventually holds (reachability)</td>
</tr>
<tr>
<td>$A \Box p$</td>
<td>for all paths, $p$ always holds (invariance, safety)</td>
</tr>
<tr>
<td>$E \Box p$</td>
<td>there exists a path where $p$ always holds</td>
</tr>
<tr>
<td>$A \Diamond p$</td>
<td>for all paths, $p$ will eventually hold</td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>whenever $p$ holds, $q$ will eventually hold (leads to/response)</td>
</tr>
</tbody>
</table>

Uppaal can provide shortest / fastest path to $p$
Introduction to UPPAAL
UPPAAL: Introduction

- UPPAAL is a tool box for
  - validation (via graphical simulation) and
  - verification (via automatic model checking) of real-time systems.

- It consists of two main parts:
  - a Graphical User Interface (GUI) and
  - a model-checker engine

- It has been jointly developed by Uppsala University in Sweden and Aalborg University in Denmark
UPPAAL: Synchronisations

- A system is defined as a network of timed automata, called processes put in parallel.

- The mechanism called *synchronisation* is used to coordinate the action of two or more processes.

- It causes two (or more) processes to take a transition at the same time.

  - A **channel** \((c)\) is declared,
  - one process will have an edge annotated with \(c!\) (*sender*) and
  - the other(s) process(es) another edge annotated with \(c?\) (*receiver(s)*).
Another Example

- Three Locations i.e. off, low, bright
- Synchronisation between User and Lamp by done by *press*. 
UPPAAL: Overview

Menu
Icons
Tabs

Project
- Declarations
  - Template
- System declarations
Following TA has following properties

1. If a pedestrian arrives,
   1. the light switches to yellow (5 sec),
   2. then to red (45 sec), and
   3. then back to green.

2. The green period must remain on at least 20 seconds before switching to yellow.
UPPAAL: Simulator

Simulation Trace:

(Green, Loop)
x: Process2 \rightarrow Process1
(Yellow, Loop)
Process1
(Red, Loop)
Process1
(Green, Loop)
x: Process2 \rightarrow Process1
(Yellow, Loop)
Process1
(Red, Loop)
Process1
(Green, Loop)
x: Process2 \rightarrow Process1
(Yellow, Loop)
Process1
(Red, Loop)
Conclusions

- It is applied successfully in applications ranging from communication protocols to multimedia.

- State space grows quickly with model complexity.
  - Suitable level of abstraction of model.
  - Features related to the properties to be verified should be modelled only.