Veryfing Real–Time Systems
The UPPAAL Model Checker
Introduction to UPPAAL

UPPAAL is a toolbox for modeling, simulation and verification of real-time systems

- Uppsala University + Aalborg University = UPPAAL

Examples of real-time systems are

- real-time controllers
- communication protocols
- multimedia applications
Introduction to UPPAAL

Systems modeled as networks of Timed Automata enriched with

- integer variables
- structured data types
- channel synchronisations
- urgency

Properties to be verified can specified in a subset of CTL (computational tree logic)
About UPPAAL:

- first version released in 1995
- it consists of:
  - a graphical description tool
  - a simulator
  - a model-checker
- Java user interface and C++ verification engine
- freely available at http://www.uppaal.com/
A Timed Automaton is a finite–state machine extended with clock variables.

- A clock variable evaluates to a real number
- All the clocks progress synchronously

A system is modelled as a parallel composition of timed automata

An automaton may perform a transition separately or synchronise with another automaton (channel synchronisation)
Network of Timed Automata

The lamp example
Clock Valuations and Boolean Guards

Let $C$ be a set of clocks. A *clock valuation* is a function $u : C \to \mathbb{IR}_{\geq 0}$

Boolean guards are defined as follows:

$$B(C) = \text{true} \mid \text{false} \mid x \mathrel{\&} c \mid x - y \mathrel{\&} c \mid B(C) \land B(C) \mid B(C) \lor B(C) \mid \neg B(C)$$

where $x, y \in C$, $c \in \mathbb{IN}$, and $\mathrel{\&} \in \{<, \leq, =, \geq, >\}$.

Let $g \in B(C)$, we write $u \in g$ if $u$ is a clock valuation satisfying the boolean guard $g$. 
Location Invariants

Boolean guards are used as transition guards, but also as location invariants:

![Diagram](attachment:image.png)

The automaton can be in state **Start** only if the valuation of clock $x$ is smaller than or equal to 15.
Example: Transition Guards and Location Invariants

(a) Test.  (b) Observer.  (c) Behaviour: one possible run.
Example: Transition Guards and Location Invariants

The observer automaton is as before
Example: Transition Guards and Location Invariants

The observer automaton is as before.
Definition of Timed Automaton

**Definition (Timed Automaton)** A Timed Automaton is a tuple \((L, \ell_0, C, A, E, I)\), where:

- \(L\) is a set of locations
- \(\ell_0 \in L\) is the initial location
- \(C\) is the set of clocks,
- \(A\) is the set of actions (e.g. press!), co–actions (e.g. press?) and internal \(\tau\)--actions
- \(E \in L \times A \times B(C) \times 2^C \times L\) is a set of edges between locations with an action, a guard and a set of clocks to be reset
- \(I : L \rightarrow B(C)\) assigns invariants to locations
Semantics of a Timed Automaton

Definition (Semantics of a Timed Automaton) Let $(L, \ell_0, C, A, E, I)$ be a timed automaton. The semantics is defined as a labelled transition system $\langle S, s_0, \rightarrow \rangle$, where $S \subseteq L \times \mathbb{R}^C$ is the set of states, $s_0 = (\ell_0, u_0)$ is the initial state, and $\rightarrow \subseteq S \times \{\mathbb{R} \geq 0 \cup A\} \times S$ is the transition relation such that

$\blacktriangleright$ $(\ell, u) \xrightarrow{d} (\ell, u + d)$ if $\forall d'. 0 \leq d' \leq d \implies u + d' \in I(\ell)$

$\blacktriangleright$ $(\ell, u) \xrightarrow{a} (\ell', u')$ if there exists $e = (\ell, a, g, r, \ell') \in E$ such that $u \in g$, $u' = [r \mapsto 0]u$, and $u' \in I(\ell)$

where for $d \in \mathbb{R} \geq 0$, $u + d$ maps each clock $x$ in $C$ to the value $u(x) + d$, and $[r \mapsto 0]u$ denotes the clock valuation which maps each clock in $r$ to 0 and agrees with $u$ over $C \setminus r$. 
Definition of Network of Timed Automata

A *network of Timed Automata* over a common set of clocks and actions consists of $n$ Timed Automata $(L_i, \ell_i^0, C, A, E_i, I_i)$ with $1 \leq i \leq n$.

A *location vector* is a vector $\vec{\ell} = (\ell_1, \ldots, \ell_n)$.

Location invariant functions are composed into a common function over location vectors $I(\vec{\ell}) = I_1(\ell_1) \land \ldots \land I_n(\ell_n)$.

$\vec{\ell}[\ell_i/\ell_i']$ denotes the location vector where the $i$th element $\ell_i$ of $\vec{\ell}$ has been replaced by $\ell_i'$. 
Semantics of a Network of Timed Automata

Let $A_i = (L_i, \ell_i^0, C, A, E_i, l_i)$ be a network of Timed Automata. Let $\bar{\ell}_0 = (\ell_1^0, \ldots, \ell_n^0)$ be the initial location vector. The semantics is defined as a transition system $\langle S, s_0, \rightarrow \rangle$, where

$S = (L_1, \times \ldots \times L_n) \times \mathbb{IR}^C$ is the set of states, $s_0 = (\bar{\ell}_0, u_0)$ is the initial state, and $\rightarrow \subseteq S \times S$ is the transition relation defined by:

1. $(\bar{\ell}, u) \rightarrow (\bar{\ell}, u + d)$ if $\forall d'. 0 \leq d' \leq d \implies u + d' \in I(\bar{\ell})$
2. $(\bar{\ell}, u) \rightarrow ([\ell'_i/\ell_i], u')$ if there exists $(\ell_i, \tau, g, r, \ell'_i)$ such that $u \in g$, $u' = [r \mapsto 0]u$ and $u' \in I(\ell[\ell'_i/\ell_i])$
3. $(\ell, u) \rightarrow (\ell[\ell'_j/\ell_j, \ell'_i/\ell_i], u')$ if there exist $(\ell_i, c?, g_i, r_i, \ell'_i)$ and $(\ell_j, c!, g_j, r_j, \ell'_j)$ such that $u \in (g_i \land g_j)$, $u' = [r_i \cup r_j \mapsto 0]u$ and $u' \in I(\ell[\ell'_j/\ell_j, \ell'_i/\ell_i])$.
Extensions to Timed Automata

Some of the additional features of the UPPAAL modelling language are the following:

- **Bounded integer variables** are declared as `int[min,max] name`, where `min` and `max` are the lower and upper bound, respectively. Violating a bound leads to an invalid state that is discarded at run–time.

- **Arrays** are arrays.

- **Broadcast channels** One sender `c!` can synchronise with an arbitrary number of receivers `c?`. Any available receiver must synchronise. Broadcast sending is never blocking.
Example: Fisher’s Mutual Exclusion Protocol

With the following declarations (for 6 processes):

\[
\text{int}[0,6] \ id; \ \text{const k 2; clock x;}
\]

and the following parameter (for 6 processes): \[\text{int}[1,6] \ \text{pid;}\]
Extensions to Timed Automata

Some of the additional features of the UPPAAL modelling language are the following:

- **Urgent locations** Time is not allowed to pass when the system is an urgent location. They are semantically equivalent to adding an extra clock $x$ that is reset on all incoming edges, and having an invariant $x \leq 0$ on the location.

- **Committed locations** A committed location is the same as an urgent location but the next transition must involve an outgoing edge of at least one of the committed locations of the network.
Example: Urgent vs Commit

- When P0 is in S1, time can pass and any edge can be taken.
- When P1 is in S1, time cannot pass, but any edge can be taken.
- When P2 is in S1, time cannot pass and the only edge that can be taken is the one from S1 to S2 in P2.
Examples... 

Examples included in the **Uppaal** package

- The four vikings problem (*bridge.xml*)
- The train gate (*train-gate.xml*)
State Formulae

**Uppaal** uses a simplified version of CTL as its query language.

The query language consists of path formulae and state formulae.

- State formulae describe individual states
- Path formulae quantify over paths of the model

A state formula is an expression that can be evaluated for a state.

\[ x > 3 \quad i == 2 \quad x <= 3 \quad \text{and} \quad i == 5 \]

Moreover:

- the state formula \( P.\ell \) tests whether the Timed Automaton identified as process \( P \) is in a given location \( \ell \)
- the state formula deadlock is satisfied for all deadlock states of the network (there are no enabled transitions)
Path Formulae

Path formulae have the following syntax:

\[ PF ::= A \Box \phi \]
\[ \quad | A \Diamond \phi \]
\[ \quad | E \Box \phi \]
\[ \quad | E \Diamond \phi \]
\[ \quad | \phi \leadsto \psi \quad \text{that is } A \Box (\phi \rightarrow E \Diamond \psi) \]

Path formulae can be classified into

- reachability
- safety
- liveness
Path Formulae

A[] ϕ

E<> ϕ

ψ ⊢ ϕ

A<> ϕ

E[] ϕ

Verifying Real–Time Systems — The UPPAAL Model Checker
Path Formulae

Reachability Properties

They ask whether there exists a path starting at the initial state, such that a state formula $\phi$ is eventually satisfied.

Reachability properties are often used while designing a model to perform sanity checks (e.g. is it possible for a sender to send a message?).

These properties do not by themselves guarantee the correctness of the protocol (i.e. that any message is eventually delivered), but they validate the basic behavior of the model.

A reachability property is expressed by the path formula $E\Diamond \phi$. 

Path Formulae

Safety Properties
Something bad will never happen! (e.g. in a nuclear power plant the temperature of the core is always (invariantly) under a certain threshold)

A variation: something bad will possibly never happen! (e.g. in a game, a safe state in one in which the player can still win – will possibly not loose)

In UPFAAL these properties are formulated positively (something good is invariantly true) and they are expressed by the path formulae $A\Box \phi$ and $E\Diamond \phi$
Path Formulae

Liveness Properties

Something good will eventually happen! (e.g. when pressing the on button, then eventually the television should turn on)

A variation: if something good happen, then something else will eventually happen! (e.g. in a communication protocol, any message that has been sent should eventually be received)

These properties are formulated as $A\diamond \phi$, and $\phi \leadsto \psi$ (i.e. $\phi$ leads to $\psi$).
Model Checking Procedure

All path formulae can be expressed as reachability and invariance properties:

- $E\diamond \phi$ is reachability
- $E\Box \phi$ is invariance
- $A\diamond \phi = \neg E\Box \neg \phi$
- $A\Box \phi = \neg E\diamond \neg \phi$

The model–checking procedure implemented in Uppaal is based on a finite–state symbolic semantics of networks.

- The logic is interpreted with respect to symbolic states of the form $(\bar{l}, D)$, where $D$ is a constraint system.
- A symbolic state $(\bar{l}, D)$ represents all the states $(\bar{l}, u)$ where $u$ satisfies the constraint $D$
Model Checking Procedure

An algorithm for symbolic reachability analysis. Symbolic invariance should be similar.