Embedded Systems Specification and Design
Model-based Design and Verification

David Kendall
Real-time Systems

Development of embedded systems is challenging due to:

- concurrency
- communication
- real-time
- resource constraints

So far, we have considered concurrency and communication. What about real-time?

How to include explicit reference to time in models and specifications?

If a message is sent, it is eventually delivered – OK
If a message is sent, it is delivered within 45ms – ????
The essence of a formal method

Informally...

The system meets its specification

Formally...

\[ \text{Sys} \models \text{Spec} \]

A formal language has well-defined

- syntax
- semantics
- rules for reasoning about the relationship between expressions
Model checking for embedded systems

Model-checking allows systems engineers to make practical use of formal methods

Model-checking provides the method for reasoning about the relationship between a model and a specification

For real-time, we need

- a language for models that includes explicit references to quantitative time - **timed automata**
- a language for specifications that includes explicit references to quantitative time - **timed temporal logic**
Timed Automaton

- Timed automaton – finite-state machine extended with clock variables.
- Each clock variable evaluates to a real number.
- All clocks progress synchronously (and keep perfect time!)
  - This is an abstraction
- Clocks can be compared against lower and upper bounds. Comparisons can be strict or non-strict
- An invariant condition is associated with each location.
- Each edge has a guard, an action label and a set of clocks to reset.
- A system is modelled as a network of timed automata in parallel.
The example shows

- Locations
- Edges
- Clock Guards and Invariants
- Synchronisation Labels
- Clock Resets
We need a User to interact with the Lamp
Exercise:

- Name the parts…
- Location, Edge, Label, Guard, Reset, Invariant
Some notation for clocks

Assume $C$ is a set of clock variables

Clock Guards and Invariants

- $B(C)$ is the set of conjunctions over simple conditions of the form $x \preceq k$ or $x - y \preceq k$, where $x, y \in C$, $k \in \text{Int}$ and $\preceq$ is one of $<, \leq, =, \geq, >$

Clock Valuations

- A clock valuation $\nu$ gives the current value of each clock in $C$, i.e. $\nu(x)$ gives the value of clock $x$
- $\nu \in [C \to \mathbb{R}_{\geq 0}]$ (can be written as $\nu \in \mathbb{R}^C_{\geq 0}$)

Clock Resets

- If $r \subseteq C$ then $\nu[r]$ is a clock valuation in which all the clocks in $r$ have the value 0, and all other clocks have the same value as given by $\nu$, i.e. $\forall x \in r : \nu[r](x) = 0$ and $\forall x \in C \setminus r : \nu[r](x) = \nu(x)$
Some notation for clocks (ctd)

Time passes

- $v + d$ is the clock valuation in which all clocks have their value in $v$ increased by $d$ time units
- formally $v + d$ maps each clock $x \in C$ to the value $v(x) + d$

Constraint satisfaction

- Assume $g \in B(C)$ is some constraint
- $v \models g$ is the notation that means $g$ is true for clock valuation $v$.

Zero valuation

- $0_C$ is a clock valuation in which all clocks in $C$ have the value 0
Exercise

Assume $C = \{x, y, z\}$, $v = [x \mapsto 1, y \mapsto 2, z \mapsto 3]$ and $r = \{x, z\}$.

- Write out the following valuations:
  1. $v + 3$
  2. $v[r]$
  3. $v[r] + 10$
  4. $(v + 10)[r]$

- Give clock valuations $v$ such that
  1. $v \models x \leq 5 \land z \geq 2$
  2. $v \models y \leq 5 \land z \geq 2$
A timed automaton is a tuple \((L, \ell^0, C, A, E, I)\), where

- \(L\) is a finite set of locations,
- \(\ell^0 \in L\) is the initial location,
- \(C\) is a finite set of clocks,
- \(A\) is a finite set of actions, co-actions and the internal \(\tau\) action,
- \(E \subseteq L \times B(C) \times A \times 2^C \times L\) is a set of edges between locations with a guard, an action and a set of clocks to be reset
  - we often write \(\ell \xrightarrow{g,a,r} \ell'\) for \((\ell, g, a, r, \ell') \in E\)
- \(I \in [L \rightarrow B(C)]\) assigns invariants to locations
A **timed transition system** is a tuple \((S, s^0, \mathcal{L}, \rightarrow)\) where

- \(S\) is the set of states
- \(s^0\) is the initial state
- \(\mathcal{L} = \mathbb{R}_{\geq 0} \cup A\) is the set of labels (assuming \(A\) is some set of discrete actions disjoint from \(\mathbb{R}\))
- \(\rightarrow \subseteq S \times \mathcal{L} \times S\) is the transition relation, e.g. \(s \overset{3.5}{\rightarrow} s', s \overset{a}{\rightarrow} s'\).
The semantics of a timed automaton $A = (L, \ell^0, C, A, E, I)$ is given by a timed transition system, constructed as follows:

- The set $S$ of states is the set of all possible combinations of locations and clock valuations $(\ell, v)$ where $v \models I(\ell)$, i.e. the invariant is satisfied.
- The initial state $s^0$ is given by $(\ell^0, 0_C)$, i.e. the initial location with the zero clock valuation.
- The set $L$ of labels is $\mathbb{R}_{\geq 0} \cup A$.
- The transition relation $\rightarrow$ is given by the rules TA.1 and TA.2, following...
Timed Automaton Semantics (ctd)

**Definition (TA.1 - Action Transitions)**

\[(\ell, v) \xrightarrow{a} (\ell', v') \text{ iff } \exists \ell \xrightarrow{g,a,r} \ell' \in E : v \models g \land v' = v[r] \land v' \models l(\ell')\]

**Definition (TA.2 - Time Transitions)**

\[(\ell, v) \xrightarrow{d} (\ell, v + d) \text{ iff } d \in \mathbb{R}_{\geq 0} \text{ and } \forall d' : 0 \leq d' \leq d \Rightarrow v + d' \in l(\ell).\]
A behaviour (run, execution, trace) of a timed automaton is an alternating sequence of time transitions and action transitions of its timed transition system, starting in the initial state:

- Time passes, action occurs, time passes, action occurs, time passes, action occurs, ...
- e.g. \((idle, 0) \xrightarrow{150} (idle, 150) \xrightarrow{press?} (one, 0) \xrightarrow{250} (one, 250) \xrightarrow{press?} (two, 250) \ldots\)

Actions are instantaneous (take no time)

Time-consuming activities are modelled by distinct start and end actions, e.g. `start_transmission, end_transmission`
Consider just the Lamp automaton from an earlier slide

1. Write down the components of the tuple for the Lamp automaton
2. Use the rules TA.1 and TA.2 to derive a (finite prefix of a) possible behaviour of the Lamp automaton
Properties of Time

- **Deterministic** whenever \( s \xrightarrow{d} s' \) and \( s \xrightarrow{d} s'' \) then \( s' = s'' \), i.e. it’s not possible to reach different states simply by passage of time.

- **Dense** for any two time points \( d_1 < d_2 \), there exists a third point \( d \) such that \( d_1 < d < d_2 \).

- **Non-Zeno** there is no bound on the progress of time, i.e. for any real value \( c \), the time can progress beyond \( c \).
Deadlock and Time-deadlock

- A state $s$ in the TTS of a TA is a deadlocked state if there is no time delay $d$ and action $a$ such that $s \xrightarrow{d} s'' \xrightarrow{a} s'$

- A state $s$ in the TTS of a TA is a time-deadlocked state if every execution from $s$ is a Zeno execution. A Zeno execution is an infinite trace of a system in which the progress of time is bounded by some upper limit $c \in \mathbb{R}_{\geq 0}$

- Give examples of TA with deadlocked and time-deadlocked states.
www.uppaal.org
Integrated environment for modelling, simulation and verification of real-time systems
Developed by universities of Uppsala and Aalborg
Systems modelled as collection of extended TA
Applications: real-time controllers, communication protocols etc
Gearbox controller: Mecel AB and Uppsala