Real-Time Systems

Lecture 15: Extended Timed Automata

2018-01-09

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

- Extended Timed Automata – Syntax
  - Data Variables
  - Urgent locations and channels
  - Committed locations

- Extended Timed Automata – Semantics
  - labelled transition system
  - extended valuations, timeshift, modification
  - examples for urgent / committed

- Extended vs. Pure Timed Automata

- The Reachability Problem
  of Extended Timed Automata

- Uppaal Query Language
  - Transition graph (!)
  - By-the-way: satisfaction relation decidable.
Example (Partly Already Seen in Uppaal Demo)

Extended Timed Automata

Templates:
- \( L: \)
  - press? \( \rightarrow \) light
    - \( z := 0 \)
    - \( x \leq 3 \)
    - \( x > 3 \)

- \( U: \)
  - \( v := 0 \)
  - \( y := 0 \)
  - \( y < 2 \)

System:
- \( x \) press? \( \rightarrow \) \( y \) press!
- \( L \) chan press \( \rightarrow \) \( U \) press!

Extensions:
- Data Variables (Expressions, Constraints, Updates)
- Structuring
- Urgent/Committed Locations, Urgent Channels
When modelling controllers as timed automata, it is sometimes desirable to have (local and shared) non-clock variables. E.g., count number of open doors, or intermediate positions of gas valve.

Adding variables with finite range (possibly grouped into finite arrays) to any finite-state automata concept is straightforward:

But: writing edges is tedious then.

So: have variables as “first class citizens” and let compilers do the work.

Interestingly, many case-studies in the literature live without non-clock variables. The more abstract the model is, i.e., the fewer information it keeps track of (e.g., in data variables), the easier the verification task.
Data Variables and Expressions

- Let \((v, w \in V)\) be a set of (integer) variables.
  \((\psi_{int} \in \Psi(V))\): integer expressions over \(V\) using function symbols \(+, -, \ldots\)
  \((\varphi_{int} \in \Phi(V))\): integer (or data) constraints over \(V\), using integer expressions, predicate symbols \(=, <, \leq, \ldots\), and logical connectives.

- Let \((x, y \in X)\) be a set of clocks.
  \((\varphi \in \Phi(X, V))\): The set of (extended) guards is defined by the following grammar:

\[
\varphi ::= \varphi_{clk} \mid \varphi_{int} \mid \varphi_1 \land \varphi_2
\]

where \(\varphi_{clk} \in \Phi(X)\) is a simple clock constraint (as defined before) and \(\varphi_{int} \in \Phi(V)\) an integer (or data) constraint.

Examples: Extended guard or not extended guard? Why?

(a) \(x < y \land v > 2\), (b) \(x < y \lor v > 2\), (c) \(v < 1 \lor v > 2\), (d) \(x < v\)

Modification or Reset Operation

- New: a modification or reset (operation) is
  \[x := 0, \quad x \in X,\]
  or
  \[v := \psi_{int}, \quad v \in V, \quad \psi_{int} \in \Psi(V).\]

- By \(R(X, V)\) we denote the set of all resets.
- By \(\vec{r}\) we denote a finite list \(\langle r_1, \ldots, r_n \rangle\), \(n \in \mathbb{N}_0\), of reset operations \(r_i \in R(X, V)\); \(\langle\rangle\) is the empty list.
- By \(R(X, V)^*\) we denote the set of all such lists of reset operations (also called reset vector).

Examples: Modification or not? Why? (\(x, y\) clocks; \(v, w\) variables)

(a) \(x := y\), (b) \(x := v\), (c) \(v := x\), (d) \(v := w\), (e) \(v := 0\)
Structuring Facilities

- Global declarations of clocks, data variables, channels, and constants.
- Binary and broadcast channels: chan $c$ and broadcast chan $b$.
- Templates of timed automata.
- Instantiation of templates (instances are called process).
- System definition: list of processes.

Restricting Non-determinism

- **Urgent locations** – enforce local immediate progress.

  ![Urgent Locations](image)

- **Committed locations** – enforce atomic immediate progress.

  ![Committed Locations](image)

- **Urgent channels** – enforce cooperative immediate progress.

  urgent chan press;
Replace

\[ \ell \quad \text{urgent} \quad \varphi \quad \text{with} \quad \ell \quad \varphi \land z = 0 \]

where \( z \) is a fresh clock:
- reset \( z \) on all in-going edges,
- add \( z = 0 \) to invariant.

**Question:** How many fresh clocks do we need in the worst case for a network of \( N \) extended timed automata?

---

**Extended Timed Automata**

**Definition 4.39.** An extended timed automaton is a structure

\[ \mathcal{A}_e = (L, C, B, U, X, V, I, E, \ell_{\text{ini}}) \]

where \( L, B, X, I, \ell_{\text{ini}} \) are as in Definition 4.3, except that location invariants in \( I \) are downward closed, and where
- \( C \subseteq L \): committed locations,
- \( U \subseteq B \): urgent channels,
- \( V \): a set of data variables (with finite domain),
- \( E \subseteq L \times B \times \Phi(X, V) \times R(X, V)^* \times L \)

is a set of directed edges such that

\[ (\ell, \alpha, \varphi, \vec{r}, \ell') \in E \land \text{chan}(\alpha) \in U \implies \varphi = \text{true}. \]

Edges \((\ell, \alpha, \varphi, \vec{r}, \ell')\) from location \( \ell \) to \( \ell' \) are labelled with an action \( \alpha \), a guard \( \varphi \), and a list \( \vec{r} \) of reset operations.
Operational Semantics of Networks

Definition 4.40. Let

\[ A_{e,i} = (L_i, C_i, B_i, U_i, X_i, V_i, I_i, E_i, \ell_{ini,i}), \quad 1 \leq i \leq n, \]

be extended timed automata with pairwise disjoint sets of clocks \( X_i \). The operational semantics of \( N = C(A_{e,1}, \ldots, A_{e,n}) \) (closed!)

is the labelled transition system

\[ T_e(C(A_{e,1}, \ldots, A_{e,n})) = T(N) = \]

\[ (\text{Conf}, \text{Time} \cup \{\tau\}, \{\lambda \rightarrow \tau | \lambda \in \text{Time} \cup \{\tau\}\}, C_{ini}) \]

where

- \( X = \bigcup_{i=1}^n X_i \) and \( V = \bigcup_{i=1}^n V_i \),
- \( \text{Conf} = \{ (\ell, \nu) | \ell_i \in L_i, \nu : X \cup V \rightarrow \text{Time}, \nu \models \bigwedge_{k=1}^n I_k(\ell_k) \} \),
- \( C_{ini} = \{ (\ell_{ini}, \nu_{ini}) \} \cap \text{Conf} \),

The transition relations consists of transitions of the following three types.
Helpers: Extended Valuations and Timeshift

- **Now:** \( \nu : X \cup V \to \text{Time} \cup D(V) \)
- Canonically extends to \( \nu : \Psi(V) \to D(\nu) \) (valuation of expression).
- \( \models \) extends canonically to expressions from \( \Phi(X, V) \).
- Extended timeshift \( \nu + t, t \in \text{Time} \), applies to clocks only:
  - \((\nu + t)(x) := \nu(x) + t, x \in X, \)
  - \((\nu + t)(v) := \nu(v), v \in V. \)
- Effect of modification \( r \in R(X, V) \) on \( \nu \), denoted by \( \nu[r] \):
  - \( \nu[x := 0](a) := \begin{cases} 0, & \text{if } a = x, \\ \nu(a), & \text{otherwise} \end{cases} \)
  - \( \nu[v := \psi_{\text{int}}](a) := \begin{cases} \nu(\psi_{\text{int}}), & \text{if } a = v, \\ \nu(a), & \text{otherwise} \end{cases} \)
- We set \( \nu[\langle r_1, \ldots, r_n \rangle] := (\nu[r_1], \ldots, [r_n]) = ( \ldots (\nu[r_1])[r_2] \ldots, [r_n] \ldots ) \).

Operational Semantics of Networks: Internal Transitions

- An **internal transition** \( \langle \vec{\ell}, \nu \rangle \xrightarrow{\tau} \langle \vec{\ell}', \nu' \rangle \) occurs if there is \( i \in \{1, \ldots, n\} \) such that
  - there is a \( \tau \)-edge \( (\ell_i, \tau, \varphi, \vec{r}, \ell'_i) \in E_i, \)
  - \( \nu \models \varphi, \)
  - \( \vec{\ell}' = \vec{\ell}[\ell_i := \ell'_i], \)
  - \( \nu' = \nu[\vec{r}], \)
  - \( \nu' \models I_i(\ell'_i), \)
  - (\( \clubsuit \)) if \( \ell_k \in C_k \) for some \( k \in \{1, \ldots, n\} \) then \( \ell_i \in C_i, \)
A synchronisation transition \( \langle \vec{\ell}, \nu \rangle \xrightarrow{\tau} \langle \vec{\ell}', \nu' \rangle \) occurs if there are \( i, j \in \{1, \ldots, n\} \) with \( i \neq j \) such that

- there are edges \((\ell_i, b!, \varphi_i, \vec{r}_i, \ell_i') \in E_i \) and \((\ell_j, b?, \varphi_j, \vec{r}_j, \ell_j') \in E_j \),
- \( \nu \models \varphi_i \land \varphi_j \),
- \( \vec{\ell}' = \vec{\ell}[\ell_i := \ell_i'][\ell_j := \ell_j'] \),
- \( \nu' = (\nu[\vec{r}_i] \cap \nu[\vec{r}_j]) \),
- \( \nu' \models I_i(\ell_i') \land I_j(\ell_j') \),
- (\( \spadesuit \)) if \( \ell_k \in C_k \) for some \( k \in \{1, \ldots, n\} \) then \( \ell_i \in C_i \) or \( \ell_j \in C_j \).

A delay transition \( \langle \vec{\ell}, \nu \rangle \xrightarrow{\mu} \langle \vec{\ell}, \nu + \tau \rangle \) occurs if

- \( \nu + \tau \models \wedge_{k=1}^n I_k(\ell_k) \),
- (\( \spadesuit \)) there are no \( i, j \in \{1, \ldots, n\} \) and \( b \in U \) with \((\ell_i, b!, \varphi_i, \vec{r}_i, \ell_i') \in E_i \) and \((\ell_j, b?, \varphi_j, \vec{r}_j, \ell_j') \in E_j \),
- (\( \spadesuit \)) there is no \( i \in \{1, \ldots, n\} \) such that \( \ell_i \in C_i \).
### Restricting Non-determinism: Example

#### Property 1
- $w$ can become 1

#### Property 2
- $y \leq 0$ holds when $Q$ is in $q_1$

#### Property 3
- $(x \geq y \Rightarrow y \leq 0)$ holds when in $p_1$ and $q_1$

| $\mathcal{N} := P \parallel Q \parallel R$ | ✔ | ✗ | ✗ |
| $\mathcal{N}, q_1$ urgent | ✓ | ✓ | ✓ |
| $\mathcal{N}, q_1$ committed | ✗ | ✓ | ✓ |
| $\mathcal{N}, b$ urgent | ✓ | ✗ | ✓ |

#### Content

- **Extended Timed Automata – Syntax**
  - Data Variables
  - Urgent locations and channels
  - Committed locations

- **Extended Timed Automata – Semantics**
  - labelled transition system
  - extended valuations, timeshift, modification
  - examples for urgent / committed

- **Extended vs. Pure Timed Automata**

- The Reachability Problem of Extended Timed Automata

- **Uppaal Query Language**
  - Transition graph (!)
  - By-the-way: satisfaction relation decidable.
Extended vs. Pure Timed Automata

\[ A_e = (L, C, U, X, V, I, E, \ell_{ini}) \]
\[ (\ell, \alpha, \varphi, \vec{r}, \ell') \in L \times B_{\mathbb{N}} \times \Phi(X, V) \times R(X, V)^* \times L \]

vs.

\[ A = (L, B, X, I, E, \ell_{ini}) \]
\[ (\ell, \alpha, \varphi, Y, \ell') \in E \subseteq L \times B_{\mathbb{N}} \times \Phi(X) \times 2^X \times L \]

- \( A_e \) is in fact (or specialises to) a **pure** timed automaton if
  - \( C = \emptyset \),
  - \( U = \emptyset \),
  - \( V = \emptyset \),
  - for each \( \vec{r} = (r_1, \ldots, r_n) \), every \( r_i \) is of the form \( x := 0 \) with \( x \in X \).

- \( I(\ell), \varphi \in \Phi(X) \) is then a consequence of \( V = \emptyset \).
Theorem 4.41. If $A_1, \ldots, A_n$ specialise to pure timed automata, then the operational semantics of

\[ C(A_1, \ldots, A_n) \]

and

\[ \text{chan } b_1, \ldots, b_m \bullet (A_1 \parallel \ldots \parallel A_n) \]

where \( \{b_1, \ldots, b_m\} = \bigcup_{i=1}^{n} B_i \), coincide, i.e.

\[ T_e(C(A_1, \ldots, A_n)) = T(\text{chan } b_1, \ldots, b_m \bullet (A_1 \parallel \ldots \parallel A_n)) \]

Content

- Extended Timed Automata – Syntax
  - Data Variables
  - Urgent locations and channels
  - Committed locations

- Extended Timed Automata – Semantics
  - labelled transition system
  - extended valuations, timeshift, modification
  - examples for urgent / committed

- Extended vs. Pure Timed Automata

- The Reachability Problem
  of Extended Timed Automata

- Uppaal Query Language
  - Transition graph (!)
  - By-the-way: satisfaction relation decidable.
Recall

**Theorem 4.33.** [Location Reachability]
The location reachability problem for pure timed automata is **decidable**.

**Theorem 4.34.** [Constraint Reachability]
Constraint reachability is **decidable** for pure timed automata.

- And what about tea `with extended timed automata?`
Extended Timed Automata add the following features:

- **Data-Variables**
  - As long as the domains of all variables in $V$ are finite, adding data variables doesn’t harm decidability.
  - If they’re infinite, we’ve got a problem (encode two-counter machine!).

- **Structuring Facilities**
  - Don’t hurt – they’re merely abbreviations.

- **Restricting Non-determinism**
  - Restricting non-determinism doesn’t affect the configuration space $Conf$.
  - Restricting non-determinism only removes certain transitions, so it makes the reachable part of the region automaton even smaller (not necessarily strictly smaller).
Consider $\mathcal{N} = \mathcal{C}(A_1, \ldots, A_n)$ over data variables $V$.

- **basic formula:**
  \[
  \text{atom} ::= A_i.\ell \mid \varphi
  \]
  where $\ell \in L_i$ is a location and $\varphi$ a constraint over $X_i$ and $V$.

- **configuration formulae:**
  \[
  \text{term} ::= \text{atom} \mid \neg \text{term} \mid \text{term}_1 \land \text{term}_2
  \]

- **existential path formulae:**
  \[\exists F \mid \exists G\] ("exists finally", "exists globally")
  \[
  e\text{-formula} ::= \exists \Diamond \text{term} \mid \exists \Box \text{term}
  \]

- **universal path formulae:**
  ("always finally", "always globally", "leads to")
  \[
  a\text{-formula} ::= \forall \Diamond \text{term} \mid \forall \Box \text{term} \mid \text{term}_1 \rightarrow \text{term}_2
  \]

- **formulae:**
  \[
  F ::= e\text{-formula} \mid a\text{-formula}
  \]

---

Tell Them What You’ve Told Them...

- **For convenience**, time automata can be **extended** by
  - data variables, and
  - urgent / committed locations.

- **None** of these extensions harm decidability, as long as the data variables have a **finite** domain.

- Properties to be checked for a timed automata model can be specified using the **Uppaal Query Language**,
  - which is a **tiny little fragment** of Timed CTL (TCTL),
  - and as such **by far** not as expressive as Duration Calculus.

- **TCTL** is another **means** to **formalise** requirements.
References