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.
Extended Timed Automata
Example (Partly Already Seen in Uppaal Demo)

Templates:

- $\mathcal{L}$:
  - Off
  - Light
  - Bright

- $\mathcal{U}$:
  - C
  - U

System:

- $x$
- press?
- chan press
- $y$
- $v$
- 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.
Data-Variables

- 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:
  - If we have control locations $L_0 = \{\ell_1, \ldots, \ell_n\}$,
  - and want to model, e.g., the valve position as a variable $v$ with domain $D(v) = \{0, 1, 2\}$,
  - then just use $L = L_0 \times D(v)$ as control locations, i.e. encode the current value of $v$ in locations, and consider updates of $v$ in the edges.

  $L$ is still finite, so we still have a proper timed automaton.

- 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.
Let \((v, w \in V)\) be a set of (integer) variables.

- \((\psi_{\text{int}} \in \Psi(V))\): integer expressions over \(V\) using function symbols \(+, -, \ldots\)
- \((\phi_{\text{int}} \in \Phi(V))\): 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.

- \((\phi \in \Phi(X,V))\): The set of (extended) guards is defined by the following grammar:

\[
\phi ::= \phi_{\text{clk}} \mid \phi_{\text{int}} \mid \phi_1 \land \phi_2
\]

where \(\phi_{\text{clk}} \in \Phi(X)\) is a simple clock constraint (as defined before) and \(\phi_{\text{int}} \in \Phi(V)\) 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

\[
    v := \psi_{\text{int}}, \quad v \in V, \quad \psi_{\text{int}} \in \Psi(V).
\]

- 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? \((x, y \text{ clocks}; v, w \text{ variables})\)

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

- ✗  
- ✗  
- ✗  
- ✅  
- ✅
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.

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

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

urgent chan press;
Replace

\[
\begin{array}{c}
\text{urgent} \\
\varphi
\end{array}
\]

with

\[
\begin{array}{c}
\text{urgent} \\
\varphi \\
z := 0
\end{array}
\]

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?
Definition 4.39. An extended timed automaton is a structure

\[ A_e = (L, C, B, U, X, V, I, E, \ell_{ini}) \]

where \( L, B, X, I, \ell_{ini} \) 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),
- \( E \subseteq L \times B!\varphi \times \Phi(X, V) \times R(X, V)^* \times L \) is a set of directed edges such that

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

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.
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**.
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

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

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

where

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

The transition relations consists of transitions of the following three types.
• **Now:** $\nu : X \cup V \rightarrow \text{Time} \cup \mathcal{D}(V)$

• Canonically extends to $\nu : \Psi(V) \rightarrow \mathcal{D}$ (valuation of expression).

• “$|=|$” extends canonically to expressions from $\Phi(X, V)$.

• Extended **timeshift** $\langle \nu + t \rangle$, $t \in \text{Time}$, 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 $\nu[\langle r_1, \ldots, r_n \rangle] := (\nu[r_1] \ldots [r_n]) = ( ( ( \nu[r_1] )[r_2] )[r_3] \ldots )[r_n]$. 
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 \((\ell_j, b?, \varphi_j, \vec{r}_j, \ell_j') \in E_j\),
- \( \nu \models \varphi_i \land \varphi_j \),
- \( \vec{\ell}' = \vec{\ell}[\ell_i := \ell_i'][\ell_j := \ell_j'] \),
- \( \nu' = (\nu[\vec{r}_i])[\vec{r}_j] \),
- \( \nu' \models I_i(\ell_i') \land I_j(\ell_j') \),
- (♣) 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 \( \langle \vec{l}, \nu \rangle \xrightarrow{t} \langle \vec{l}, \nu + t \rangle \) occurs if

- \( \nu + t \models \bigwedge_{k=1}^{n} I_k(\ell_k) \),

- \((\clubsuit)\) there are no \( i \neq j \in \{1, \ldots, n\}\) 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

\[ P \]
\[ x := 0 \]
\[ p_0 \rightarrow p_1 \rightarrow p_2 \]
\[ p_1 \]
\[ \text{b?} \]

\[ Q \]
\[ y := 0, v := 1 \]
\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]
\[ q_1 \]
\[ b!, v := 2 \]
\[ v := 3 \]

\[ R \]
\[ w := v \]
\[ r_0 \rightarrow r_1 \]

Property 1: \( w \) can become 1
Property 2: \( y \leq 0 \) holds when \( Q \) is in \( q_1 \)
Property 3: \( (x \geq y \implies y \leq 0) \) holds when in \( p_1 \) and \( q_1 \)

\[ \mathcal{N} := P \parallel Q \parallel R \]
- \( \checkmark \)
- \( \times \)
- \( \times \)

\[ \mathcal{N}, q_1 \text{ urgent} \]
- \( \checkmark \)
- \( \checkmark \)
- \( \checkmark \)

\[ \mathcal{N}, q_1 \text{ committed} \]
- \( \times \)
- \( \checkmark \)
- \( \checkmark \)

\[ \mathcal{N}, b \text{ urgent} \]
- \( \checkmark \)
- \( \times \)
- \( \checkmark \)
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
\[ A_e = (L, C, B, U, X, V, I, E, \ell_{\text{ini}}) \]
\[(\ell, \alpha, \varphi, \vec{r}, \ell') \in L \times B_1 \times \Phi(X, V) \times R(X, V)^* \times L \]

vs.
\[ A = (L, B, X, I, E, \ell_{\text{ini}}) \]
\[(\ell, \alpha, \varphi, Y, \ell') \in E \subseteq L \times B_2 \times \Phi(X) \times 2^X \times L \]

- \( A_e \) 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.

$$\mathcal{T}_e(C(A_1, \ldots, A_n)) = \mathcal{T}(\text{chan } b_1, \ldots, b_m \bullet (A_1 \parallel \ldots \parallel A_n)).$$
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.
Reachability Problems for Extended Timed Automata
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 \( \tilde{w} \) extended timed automata?
What About 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).
The Logic of Uppaal

\[ \sqrt{\text{EF } \varphi} \equiv \text{E} \bigcirc (\omega = 1) \]
Consider $\mathcal{N} = C(A_1, \ldots, A_n)$ 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:**
  
  $$\text{term ::= atom } \mid \neg \text{term } \mid \text{term}_1 \land \text{term}_2$$

- **existential path formulae:**
  
  ($\text{"exists finally"}, \text{"exists globally"}$)

  $$\text{e-formula ::= } \exists \diamond \text{term } \mid \exists \square \text{term}$$

- **universal path formulae:**
  
  ($\text{"always finally"}, \text{"always globally"}, \text{"leads to"}$)

  $$\text{a-formula ::= } \forall \diamond \text{term } \mid \forall \square \text{term } \mid \text{term}_1 \rightarrow \text{term}_2$$

- **formulae:**
  
  $$F ::= \text{e-formula } \mid \text{a-formula}$$
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
References