Real-Time Systems

Lecture 14: Extended Timed Automata

2013-06-25

Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:
- Decidability of the location reachability problem:
  - region automaton
  - zones

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - By what are TA extended? Why is that useful?
  - What’s an urgent/committed location? What’s the difference?
  - What’s an urgent channel?
  - Where has the notion of “input action” and “output action” correspondences in the formal semantics?

Content:
- Extended TA:
  - Data-Variables
  - Structuring Facilities
  - Restriction of Non-Determinism
  - The Logic of Uppaal
Extended Timed Automata

Example (Partly Already Seen in Uppaal Demo)

Templates:
- \( L \):
  - \( \text{off} \)  
  - \( \text{press?} \):
    - \( x := 0 \)  
    - \( x > 3 \)  
  - \( \text{light} \):
    - \( y := 0 \)  
    - \( y < 2 \)  
  - \( \text{bright} \):
    - \( v := 1 \)  

- \( U \):
  - \( \text{press!} \):
    - \( y := 0 \)  
    - \( y > 3 \)  

System:
- \( x \) press?
- \( \text{chan press} \)
- \( v \) press!

Extensions:
- Data Variables (Expressions, Constraints, Updates)
- Structuring
- Urgent/Committed Location, Urgent Channel
Data-Variables

• When modelling controllers as timed automata, it is sometimes desirable to have (local and shared) 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 range as a variable \( v \) with \( D(v) = \{0, 1, 2\} \),
  • then just use \( L = L_0 \times D(v) \) as control locations, i.e. encode the current value of \( v \) in the control location, and consider updates of \( v \) in the \( \lambda \rightarrow \).

\( L \) is still finite, so we still have a proper TA.

• But: writing \( \lambda \rightarrow \) is tedious.

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

• Interestingly, many examples in the literature live without 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.
  
  \((\psi_{\text{int}}) \in \Psi(V)\): integer expressions over \(V\) using func. symb. \(+, -, \ldots\)
  
  \((\varphi_{\text{int}}) \in \Phi(V)\): integer (or data) constraints over \(V\)
  using integer expressions, predicate symbols \(=, <, \leq, \ldots\), and
  boolean logical connectives. (incl. \(v, \neg, \land, \lor, \implies, \equiv\))

• Let \((x, y) \in X\) be a set of clocks.
  
  \((\varphi) \in \Phi(X, V)\): (extended) guards, defined by
  
  \[ \varphi ::= \varphi_{\text{clk}} \mid \varphi_{\text{int}} \mid \varphi_1 \land \varphi_2 \]
  
  where \(\varphi_{\text{clk}} \in \Phi(X)\) is a simple clock constraint (as defined before)
  and \(\varphi_{\text{int}} \in \Phi(V)\) 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) \(x < v \leq\)

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 \(\bar{r}\) we denote a finite list \((r_1, \ldots, r_n), n \in \mathbb{N}_0\),
  of reset operations \(r_i \in R(X, V)\);
  \((\emptyset)\) is the empty list.

• By \(R(X, V)^*\) we denote the set of all such lists of reset operations.

Examples: Modification or not? Why?

(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.

![Diagram of Structuring Facilities]

### Restricting Non-determinism

- **Urgent locations** — enforce local immediate progress.
  
  ![U]

- **Committed locations** — enforce atomic immediate progress.
  
  ![C]

- **Urgent channels** — enforce cooperative immediate progress.
  
  ```
  urgent chan press;
  ```
Urgent Locations: Only an Abbreviation...

Replace

\[
\begin{align*}
\ell & \quad \text{urgent} \\
\varphi \\
\end{align*}
\]

with

\[
\begin{align*}
\ell & \\
z := 0 & \quad \varphi \land z = 0 \\
\end{align*}
\]

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

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

where \( L, B, X, I, \ell_{ini} \) are as in Def. 4.3, except 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,
- \( E \subseteq L \times B \times \Phi(X, V) \times R(X, V)^* \times L \): 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.
Definition 4.40. Let $A_{e,i} = (L_i, C_i, B_i, U_i, X_i, V_i, I_i, E_i, \ell_{ini,i})$, $1 \leq i \leq n$, be extended timed automata with pairwise disjoint sets of clocks $X_i$.

The operational semantics of $C(A_{e,1}, \ldots, A_{e,n})$ (closed!) is the labelled transition system

$$T_e(C(A_{e,1}, \ldots, A_{e,n})) = (\text{Conf}, \text{Time} \cup \{\tau\}, \{\lambda \mapsto | \lambda \in \text{Time} \cup \{\tau\}\}, C_{ini})$$

where

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

and the transition relation consists of transitions of the following three types.

Helpers: Extended Valuations and Timeshift

- **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 $\Phi(X, V)$.

$$\Psi: \nu \mid f(\varphi_1, \ldots, \varphi_n)$$

Assume $I(\ell): \mathbb{Z}^n \rightarrow \mathbb{Z}$

$$I(\ell) = \nu(\vec{\mathcal{V}}) \in \mathcal{D}(V)$$

$I(f(\varphi_1, \ldots, \varphi_n), \nu) = I_f(I(\varphi_1), \ldots, I(\varphi_n))$

$I(\nu+\omega, [\nu_{ini}\nu_{ini}])$

$I(\nu+\omega, [\nu_{ini}\nu_{ini}]) = I\Phi(\varphi_{ini}, \nu_{ini}) \in \mathcal{D}(\nu_{ini} \nu_{ini})$
** Helpers: Extended Valuations and Timeshift **

- **Now:** \( \nu : X \cup V \to \text{Time} \cup \mathcal{D}(V) \)
- Canonically extends to \( \nu : \Psi(V) \to \mathcal{D} \) (valuation of expression).
- “\( = \)” extends canonically to expressions from \( \Psi(X, V) \).
- **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 \( \nu(r_1, \ldots, r_n) := \nu[r_1] \cdot \ldots \cdot [r_n] = (((\nu[r_1])[r_2]) \ldots)[r_n] \).

---

**Operational Semantics of Networks: Internal Transitions**

- An **internal transition** \( \langle \ell, \nu \rangle \xrightarrow{\tau} \langle \ell', \nu' \rangle \) occurs if there is \( i \in \{1, \ldots, n\} \) such that:
  - there is a \( \tau \)-edge \( (\ell_i, \tau, \varphi, r_i', \ell_i') \in E_i \),
  - \( \nu \models \varphi \),
  - \( \ell = \ell[i := \ell_i'] \),
  - \( \nu' = \nu[r] \),
  - \( \nu' \models I_i(\ell_i') \),
  - if \( \ell_k \in C_k \) for some \( k \in \{1, \ldots, n\} \) then \( \ell_i \in C_i \).
Operational Semantics of Networks: Synchronisation Transitions

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

Operational Semantics of Networks: Delay Transitions

- A delay transition \( \langle \vec{\ell}, \nu \rangle \xrightarrow{t} \langle \vec{\ell}, \nu + t \rangle \) occurs if
  - \( \nu + t \models \bigwedge_{k=1}^{n} I_k (\ell_k) \),
  - \( (\clubsuit) \) 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 \((\ell_j, b?, \varphi_j, \vec{r}_j, \ell'_j) \in E_j\),
  - \( (\clubsuit) \) there is no \( i \in \{1, \ldots, n\} \) such that \( \ell_i \in C_i \).
### Restricting Non-determinism: Example

<table>
<thead>
<tr>
<th>Property 1</th>
<th>Property 2</th>
<th>Property 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>∃◊w = 1</td>
<td>∀□Q.q₁ ⇒ y ≤ 0</td>
<td>∀□(P.p₁ ∧ Q.q₁ ⇒ (x ≥ y ⇒ y ≤ 0))</td>
</tr>
</tbody>
</table>

| N := P∥Q∥R | ✓ | × | × |
| N, q₁ urgent | ✓ | ✓ | ✓ |
| N, q₁ comm. | ✓ | ✓ | ✓ |
| N, b urgent | ✓ | ✓ | ✓ |

### Restricting Non-determinism: Urgent Location

<table>
<thead>
<tr>
<th>Property 1</th>
<th>Property 2</th>
<th>Property 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>∃◊w = 1</td>
<td>∀□Q.q₁ ⇒ y ≤ 0</td>
<td>∀□(P.p₁ ∧ Q.q₁ ⇒ (x ≥ y ⇒ y ≤ 0))</td>
</tr>
</tbody>
</table>

| N := P∥Q∥R | ✓ | × | × |
| N, q₁ urgent | ✓ | ✓ | ✓ |
| N, q₁ comm. | ✓ | ✓ | ✓ |
| N, b urgent | ✓ | ✓ | ✓ |
### Restricting Non-determinism: Committed Location

<table>
<thead>
<tr>
<th>Property 1</th>
<th>Property 2</th>
<th>Property 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists w = 1$</td>
<td>$\forall Q.q_1 \implies y \leq 0$</td>
<td>$\forall(P.p_1 \land Q.q_1 \implies (x \geq y \implies y \leq 0))$</td>
</tr>
<tr>
<td>$\mathcal{N}$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\mathcal{N}, q_1$ urgent</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$\mathcal{N}, q_1$ comm.</td>
<td>$\times$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$\mathcal{N}, b$ urgent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Restricting Non-determinism: Urgent Channel

<table>
<thead>
<tr>
<th>Property 1</th>
<th>Property 2</th>
<th>Property 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists w = 1$</td>
<td>$\forall Q.q_1 \implies y \leq 0$</td>
<td>$\forall(P.p_1 \land Q.q_1 \implies (x \geq y \implies y \leq 0))$</td>
</tr>
<tr>
<td>$\mathcal{N}$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$\mathcal{N}, q_1$ urgent</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$\mathcal{N}, q_1$ comm.</td>
<td>$\times$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$\mathcal{N}, b$ urgent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extended vs. Pure Timed Automata

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

vs.

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

- \( A_e \) 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 \( V = \emptyset \).
Operational Semantics of Extended TA

**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

$$\text{chan } b_1, \ldots, b_m \cdot (A_1 \parallel \ldots \parallel A_n),$$

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

$$\mathcal{T}_c(C(A_1, \ldots, A_n)) = \mathcal{T}(\text{chan } b_1, \ldots, b_m \cdot (A_1 \parallel \ldots \parallel A_n)).$$

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] The constraint reachability problem for pure timed automata is **decidable**.

- And what about tea ~W **extended** timed automata?

References
References