Real-Time Systems

Lecture 11: Timed Automata

2014-07-01

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:
- DC (un)decidability

This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - what’s notable about TA syntax? What's simple clock constraint?
  - what’s a configuration of a TA? When are two in transition relation?
  - what’s the difference between guard and invariant? Why have both?
  - what’s a computation path? A run? Zeno behaviour?

- **Content:**
  - Timed automata syntax
  - TA operational semantics
Content

Introduction

- **First-order Logic**
- **Duration Calculus** (DC)
- Semantical Correctness Proofs with DC
- DC Decidability
- DC Implementables
- **PLC-Automata**

- **Timed Automata** (TA), Uppaal
- Networks of Timed Automata
- Region/Zone-Abstraction
- Extended Timed Automata
- Undecidability Results

\[ \text{obs} : \text{Time} \rightarrow \mathcal{D}(\text{obs}) \]

\[ \langle \text{obs}_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle \text{obs}_1, \nu_1 \rangle, t_1 \ldots \]

- **Automatic Verification**...
- ...whether TA satisfies DC formula, observer-based
Example: Off/Light/Bright
Example
Example

- off
  - press?
  - $x := 0$

- light
  - press?
  - $x > 3$

- bright
  - press?
  - $x \leq 3$
Example

User:

- $\ell_0$: $y := 0$
- $\ell_1$: $y < 2$
- $\ell_2$: $y := 0$
- $\ell_3$: $y > 3$
- $\ell_4$: $y > 3$

- press!
- press!
- press!
- press!
- press!
Example Cont’d

Problems:
- Deadlock freedom
  [Behrmann et al., 2004]
- Location Reachability
  (“Is this user able to reach ‘bright’?”)
- Constraint Reachability
  (“Can the controller’s clock go past 5?”)
Plan

- **Pure TA** syntax
  - channels, actions
  - (simple) clock constraints
  - Def. TA

- **Pure TA** operational semantics
  - clock valuation, time shift, modification
  - operational semantics
  - discussion

- Transition sequence, computation path, run

- **Network of TA**
  - parallel composition (syntactical)
  - restriction
  - network of TA semantics

- **Uppaal Demo**
  - Region abstraction; zones

- **Extended TA**; Logic of Uppaal
Pure TA Syntax
Channel Names and Actions

To define timed automata formally, we need the following sets of symbols:

- A set \((a, b \in)\) Chan of channel names or channels.

- For each channel \(a \in\) Chan, two visible actions: \(a?\) and \(a!\) denote input and output on the channel \((a?, a! \notin\) Chan\).

- \(\tau \notin\) Chan represents an internal action, not visible from outside.

- \((\alpha, \beta \in)\) Act := \(\{a? \mid a \in\) Chan\} \(\cup\) \(\{a! \mid a \in\) Chan\} \(\cup\) \{\(\tau\)\} is the set of actions.

- An alphabet \(B\) is a set of channels, i.e. \(B \subseteq\) Chan.

- For each alphabet \(B\), we define the corresponding action set

  \[B?! := \{a? \mid a \in B\} \cup \{a! \mid a \in B\} \cup \{\tau\}.\]

- Note: Chan?! = Act.
Example

```
off
  press?
  x := 0
  press?
  x > 3

light
  press?
  x := 0
  x ≤ 3

bright
  press?
  x := 0

ℓ0
  press!
  y := 0
  y < 2

ℓ1
  press!
  y := 0
  y > 3

ℓ2
  press!
  y := 0

ℓ3

ℓ4
  press!
```

channel
input action
internal action
output action
Simple Clock Constraints

- Let \((x, y \in X)\) be a set of clock variables (or clocks).

- The set \((\varphi \in \Phi(X))\) of (simple) clock constraints (over \(X\)) is defined by the following grammar:

\[
\varphi ::= x \sim c \mid x - y \sim c \mid \varphi_1 \land \varphi_2
\]

where

- \(x, y \in X\),
- \(c \in \mathbb{Q}_0^+\), and
- \(\sim \in \{<, >, \leq, \geq\}\).

- Clock constraints of the form \(x - y \sim c\) are called difference constraints.
Example

```
press?

off

x := 0

press?

light

x \leq 3

press?

bright

x > 3

not a clock constraint

clock constraint

press!

y := 0

\ell_0

\ell_1

\ell_2

\ell_3

y < 2

y := 0

press!

\ell_4

y > 3

press!

press!

press!

press!

press!
```


Definition 4.3. [Timed automaton]
A (pure) **timed automaton** $A$ is a structure

$$A = (L, B, X, I, E, \ell_{\text{ini}})$$

where

- $(\ell \in L)$ is a finite set of **locations** (or **control states**),
- $B \subseteq \text{Chan}$,
- $X$ is a finite set of clocks,
- $I : L \rightarrow \Phi(X)$ assigns to each location a clock constraint, its **invariant**,
- $E \subseteq L \times B^?! \times \Phi(X) \times 2^X \times L$ a finite set of **directed edges**.

Edges $(\ell, \alpha, \varphi, Y, \ell')$ from location $\ell$ to $\ell'$ are labelled with an **action** $\alpha$, a **guard** $\varphi$, and a set $Y$ of clocks that will be **reset**.

- $\ell_{\text{ini}}$ is the **initial location**.
Graphical Representation of Timed Automata

\[ A = (L, B, X, I, E, \ell_{ini}) \]

- **Location (control states) and their invariants:**

\[ \ell \xrightarrow{I(\ell)} \]

- **Edges:**

\[ (\ell, \alpha, \varphi, Y, \ell') \in L \times B^?! \times \Phi(X) \times 2^X \times L \]
Pure TA Operational Semantics
Clock Valuations

- Let $X$ be a set of clocks. A **valuation $\nu$ of clocks** in $X$ is a mapping

\[ \nu : X \rightarrow \text{Time} \]

assigning each clock $x \in X$ the **current time** $\nu(x)$.

- Let $\varphi$ be a clock constraint.

The **satisfaction** relation between clock valuations $\nu$ and clock constraints $\varphi$, denoted by $\nu \models \varphi$, is defined inductively:

- $\nu \models x \approx c$ iff $\nu(x) \approx c$
- $\nu \models x \rightarrow y \approx c$ iff $\nu(x) \triangleleft \nu(y) \approx c$
- $\nu \models \varphi_1 \land \varphi_2$ iff $\nu \models \varphi_1$ and $\nu \models \varphi_2$
Clock Valuations

- Let $X$ be a set of clocks. A **valuation** $\nu$ of clocks in $X$ is a mapping

$$\nu : X \rightarrow \text{Time}$$

assigning each clock $x \in X$ the **current time** $\nu(x)$.

- Let $\varphi$ be a clock constraint. The satisfaction relation between clock valuations $\nu$ and clock constraints $\varphi$, denoted by $\nu \models \varphi$, is defined inductively:

  - $\nu \models x \sim c$ iff $\nu(x) \sim c$
  - $\nu \models x - y \sim c$ iff $\nu(x) - \nu(y) \sim c$
  - $\nu \models \varphi_1 \land \varphi_2$ iff $\nu \models \varphi_1$ and $\nu \models \varphi_2$

- Two clock constraints $\varphi_1$ and $\varphi_2$ are called (logically) equivalent if and only if for all clock valuations $\nu$, we have

$$\nu \models \varphi_1 \text{ if and only if } \nu \models \varphi_2.$$  

In that case we write $\models \varphi_1 \iff \varphi_2$. 
Operations on Clock Valuations

Let \( \nu \) be a valuation of clocks in \( X \) and \( t \in \text{Time} \).

- **Time Shift**
  We write \( \nu + t \) to denote the clock valuation (for \( X \)) with
  \[
  (\nu + t)(x) = \nu(x) + t.
  \]
  for all \( x \in X \),

- **Modification**
  Let \( Y \subseteq X \) be a set of clocks.
  We write \( \nu[Y := t] \) to denote the clock valuation with
  \[
  (\nu[Y := t])(x) = \begin{cases} 
  t, & \text{if } x \in Y \\
  \nu(x), & \text{otherwise}
  \end{cases}
  \]
  Special case **reset**: \( t = 0 \).
Definition 4.4. The operational semantics of a timed automaton

\[ A = (L, B, X, I, E, \ell_{ini}) \]

is defined by the (labelled) transition system

\[ T(A) = (Conf(A), \text{Time} \cup B?!, \{\overset{\lambda}{\rightarrow} | \lambda \in \text{Time} \cup B?!)\}, C_{ini}) \]

where

- \( Conf(A) = \{\langle \ell, \nu \rangle | \ell \in L, \nu : X \rightarrow \text{Time}, \nu \models I(\ell)\} \)
- \( \text{Time} \cup B?! \) are the transition labels,
- there are delay transition relations

\[ \langle \ell, \nu \rangle \overset{\lambda}{\rightarrow} \langle \ell', \nu' \rangle, \lambda \in \text{Time} \]

and action transition relations

\[ \langle \ell, \nu \rangle \overset{\lambda}{\rightarrow} \langle \ell', \nu' \rangle, \lambda \in B?! \]  \( \rightarrow \) later slides

- \( C_{ini} = \{\langle \ell_{ini}, \nu_0 \rangle\} \cap Conf(A) \) with \( \nu_0(x) = 0 \) for all \( x \in X \)

is the set of initial configurations.
\[ A = (L, B, X, I, E, \ell_{ini}) \]

\[ T(A) = (Conf(A), \text{Time} \cup B!!, \{ \lambda \rightarrow | \lambda \in \text{Time} \cup B!! \}, C_{ini}) \]

- **Time** or delay transition:

\[ \langle \ell, \nu \rangle \xrightarrow{t} \langle \ell, \nu + t \rangle \]

if and only if \( \forall t' \in [0, t] : \nu + t' \models I(\ell) \).

“Some **time** \( t \in \text{Time} \) **elapses** respecting invariants, location unchanged.”

- **Action** or discrete transition:

\[ \langle \ell, \nu \rangle \xrightarrow{\alpha} \langle \ell', \nu' \rangle \]

if and only if there is \( (\ell, \alpha, \varphi, Y, \ell') \in E \) such that

\[ \nu \models \varphi, \quad (\nu' = \nu[Y := 0]), \quad \text{and} \quad \nu' \models I(\ell'). \]

“An action occurs, location may change, some clocks may be reset, **time does not advance**.”
Transition Sequences, Reachability

- A **transition sequence** of \( A \) is any finite or infinite sequence of the form

\[
\langle \ell_0, \nu_0 \rangle \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle \xrightarrow{\lambda_3} \ldots
\]

with

- \( \langle \ell_0, \nu_0 \rangle \in C_{ini} \),
- for all \( i \in \mathbb{N} \), there is \( \xrightarrow{\lambda_{i+1}} \) in \( T(A) \) with \( \langle \ell_i, \nu_i \rangle \xrightarrow{\lambda_{i+1}} \langle \ell_{i+1}, \nu_{i+1} \rangle \)

- A **configuration** \( \langle \ell, \nu \rangle \) is called **reachable** (in \( A \)) if and only if there is a transition sequence of the form

\[
\langle \ell_0, \nu_0 \rangle \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle \xrightarrow{\lambda_3} \ldots \xrightarrow{\lambda_n} \langle \ell_n, \nu_n \rangle = \langle \ell, \nu \rangle
\]

- A **location** \( \ell \) is called **reachable** if and only if any configuration \( \langle \ell, \nu \rangle \) is reachable, i.e. there exists a valuation \( \nu \) such that \( \langle \ell, \nu \rangle \) is reachable.
Example

\[ \langle \text{off}, x = 0 \rangle \xrightarrow{2.5} \langle \text{off}, x = 2.5 \rangle \xrightarrow{1.7} \langle \text{off}, x = 4.2 \rangle \]
\[ \xrightarrow{\text{press?}} \langle \text{light}, x = 0 \rangle \xrightarrow{2.1} \langle \text{light}, x = 2.1 \rangle \]
\[ \xrightarrow{\text{press?}} \langle \text{bright}, x = 2.1 \rangle \xrightarrow{10} \langle \text{bright}, x = 12.1 \rangle \]
\[ \xrightarrow{\text{press?}} \langle \text{off}, x = 12.1 \rangle \]
\[ \xrightarrow{\text{press?}} \langle \text{light}, x = 0 \rangle \xrightarrow{0} \langle \text{light}, x = 0 \rangle \]
\( x = \{ x \mid y \} \)

\[ e_0 \xrightarrow{\tau} e_1 \]

\([e_0, x=0, y=0] \xrightarrow{\tau} [e_1, x=0, y=0] \]

\([e_0, x=2, y=1] \xrightarrow{2.1} [e_0, x=2, y=1] \]

\([e_0, x=2, y=1] \xrightarrow{\tau} [e_1, x=0, y=2, y=1] \]

\[ e_0 \xrightarrow{x \leq 12:00} e_1 \]

\[ x \geq 11:45 \]
Recall the user model for our light controller:

\[ \ell_0 \xrightarrow{\text{press!}} y := 0 \xrightarrow{y < 2} \ell_1 \xrightarrow{\text{press!}} \ell_2 \xrightarrow{y := 0} \ell_3 \xrightarrow{\text{press!}} \ell_4 \xrightarrow{y > 3} \ell_3 \]

- **“Good” configurations:**
  \[ \langle \ell_1, y = 0 \rangle, \langle \ell_1, y = 1.9 \rangle, \langle \ell_2, y = 1000 \rangle, \langle \ell_2, y = 0.5 \rangle, \langle \ell_3, y = 27 \rangle \]

- **“Bad” configurations:**
  \[ \langle \ell_1, y = 2.0 \rangle, \langle \ell_1, y = 2.5 \rangle \]
Two Approaches to Exclude “Bad” Configurations

- **The approach taken for TA:**
  - Rule out **bad** configurations in the step from $A$ to $T(A)$.
    “Bad” configurations are not even configurations!
  
  - **Recall Definition 4.4:**
    - $\text{Conf}(A) = \{\langle l, \nu \rangle \mid l \in L, \nu : X \rightarrow \text{Time}, \nu \models I(l)\}$
    - $C_{\text{ini}} = \{\langle l_{\text{ini}}, \nu_0 \rangle\} \cap \text{Conf}(A)$
  
  - **Note:** Being in $\text{Conf}(A)$ doesn’t mean to be **reachable**.

- **The approach not taken for TA:**
  - consider every $\langle l, \nu \rangle$ to be a configuration, i.e. have
    
    \[ \text{Conf}(A) = \{\langle l, \nu \rangle \mid l \in L, \nu : X \rightarrow \text{Time} \} \]
  
  - “bad” configurations not in transition relation with others, i.e. have, e.g.,
    
    \[ \langle l, \nu \rangle \xrightarrow{t} \langle l, \nu + t \rangle \]
    
    if and only if $\forall t' \in [0, t] : \nu + t' \models I(l)$ **and** $\nu + t' \models I(l')$. 

Computation Path, Run
Computation Paths

- \langle \ell, \nu \rangle, t \text{ is called time-stamped configuration}

- **time-stamped delay transition**: \langle \ell, \nu \rangle, t \xrightarrow{t'} \langle \ell, \nu + t' \rangle, t + t'
  
  iff \( t' \in \text{Time} \) and \langle \ell, \nu \rangle \xrightarrow{t'} \langle \ell, \nu + t' \rangle.

- **time-stamped action transition**: \langle \ell, \nu \rangle, t \xrightarrow{\alpha} \langle \ell', \nu' \rangle, t
  
  iff \( \alpha \in B_? \) and \langle \ell, \nu \rangle \xrightarrow{\alpha} \langle \ell', \nu' \rangle.

- A sequence of time-stamped configurations

  \[ \xi = \langle \ell_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle, t_1 \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle, t_2 \xrightarrow{\lambda_3} \ldots \]

  is called computation path (or path) of \( \mathcal{A} \) starting in \( \langle \ell_0, \nu_0 \rangle, t_0 \)
  
  if and only if it is either infinite or maximally finite.

- A computation path (or path) is a computation path starting at \( \langle \ell_0, \nu_0 \rangle, 0 \)
  
  where \( \langle \ell_0, \nu_0 \rangle \in C_{ini} \).
Timelocks and Zeno Behaviour

- **Timelock:**
  \[
  \langle \ell, x = 0 \rangle, 0 \xrightarrow{2} \langle \ell, x = 2 \rangle, 2 \\
  \langle \ell', x = 0 \rangle, 0 \xrightarrow{3} \langle \ell', x = 3 \rangle, 3 \xrightarrow{a?} \langle \ell', x = 3 \rangle, 3 \xrightarrow{a?} \ldots
  \]

- **Zeno behaviour:**
  \[
  \langle \ell, x = 0 \rangle, 0 \xrightarrow{1/2} \langle \ell, x = 1/2 \rangle, \frac{1}{2} \xrightarrow{1/4} \langle \ell, x = 3/4 \rangle, \frac{3}{4} \ldots \\
  \xrightarrow{1/2^n} \langle \ell, x = (2^n - 1)/2^n \rangle, \frac{2^n - 1}{2^n} \ldots
  \]
Definition 4.9. An infinite sequence

\[ t_0, t_1, t_2, \ldots \]

of values \( t_i \in \text{Time} \) for \( i \in \mathbb{N}_0 \) is called real-time sequence if and only if it has the following properties:

- **Monotonicity:**
  \[
  \forall i \in \mathbb{N}_0 : t_i \leq t_{i+1}
  \]

- **Non-Zeno behaviour** (or unboundedness or progress):
  \[
  \forall t \in \text{Time} \exists i \in \mathbb{N}_0 : t < t_i
  \]
Definition 4.10. A run of $\mathcal{A}$ starting in the time-stamped configuration $\langle \ell_0, \nu_0 \rangle$, $t_0$ is an infinite computation path of $\mathcal{A}$

$$\xi = \langle \ell_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle, t_1 \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle, t_2 \xrightarrow{\lambda_3} \ldots$$

where $(t_i)_{i \in \mathbb{N}_0}$ is a real-time sequence.

If $\langle \ell_0, \nu_0 \rangle \in C_{ini}$ and $t_0 = 0$, then we call $\xi$ a run of $\mathcal{A}$.

Example:
Example

\[ s?, x < 10, x := 0 \]

\[ a! \]

\[ \ell_0 \rightarrow \ell_1 \]

\[ x \geq 10 \]
References