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
Introduction

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

- **PLC-Automata**

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

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

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

- **Automatic Verification**...
- ...whether TA satisfies DC formula, observer-based

Example: Off/Light/Bright
Example

Example
Example

User:

\[ \ell_0 \quad \text{press!} \quad y := 0 \quad y < 2 \quad \text{press!} \quad y > 3 \]

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 \((a, b \in \text{Chan})\) of **channel names** or **channels**.

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

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

- \((\alpha, \beta \in \text{Act})\) is the set of **actions**.

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

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

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

- Note: \(\text{Chan?!} = \text{Act}\).

**Example**

```
Example

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

- light
  - press?
    - x ≤ 3

- bright
  - press?
    - y := 0
    - y < 2
  - press!
    - y := 0
    - y > 3
```
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

\[\text{press?} \quad \text{press?} \quad \text{press?} \quad \text{press?}\]

\[\text{off} \quad \text{press?} \quad \text{light} \quad \text{press?} \quad \text{bright}\]

\[x = 0 \quad x > 3 \quad x \leq 3 \quad x > c\]

We may use \(x = c\) (or \(x = y\)) as an abbreviation for \(x \leq c \land x \geq c\) (or \(x - y \leq 0 \lor x - y \geq 0\)).
**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**

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

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

- **Edges**: $(\ell, \alpha, \varphi, Y, \ell') \in L \times B \times \Phi(X) \times 2^X \times L$
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 $\nu \not\models \varphi_1$ and $\nu \not\models \varphi_2$
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
\[ T(A) = (Conf(A), Time \cup B??, \{ \lambda \rightarrow \lambda \mid \lambda \in Time \cup B?? \}, C_{ini}) \]
where
- \( Conf(A) = \{ (\ell, \nu) \mid \ell \in L, \nu : X \to Time, \nu \models I(\ell) \} \)
- Time \( \cup \) B?? are the transition labels,
- there are delay transition relations
  \[ (\ell, \nu) \xrightarrow{\lambda} (\ell', \nu'), \lambda \in Time \]
  and action transition relations
  \[ (\ell, \nu) \xrightarrow{\alpha, \varphi, Y, \ell'} (\ell', \nu'), \lambda \in B??. \]
- \( C_{ini} = \{ (\ell_{ini}, \nu_0) \} \cap Conf(A) \) with \( \nu_0(x) = 0 \) for all \( x \in X \)
is the set of initial configurations.

Operational Semantics of TA Cont’d

\[ A = (L, B, X, I, E, \ell_{ini}) \]
\[ T(A) = (Conf(A), Time \cup B??, \{ \lambda \rightarrow \lambda \mid \lambda \in Time \cup B?? \}, C_{ini}) \]

- Time or delay transition:
  \[ (\ell, \nu) \xrightarrow{t} (\ell, \nu + t) \]
  if and only if \( \forall t' \in [0, t] : \nu + t' \models I(\ell) \).
  “Some time \( t \in Time \) elapses respecting invariants, location unchanged.”

- Action or discrete transition:
  \[ (\ell, \nu) \xrightarrow{\alpha, \varphi, Y, \ell'} (\ell', \nu') \]
  if and only if there is \( (\ell, \alpha, \varphi, Y, \ell') \in E \) such that
  \[ \nu \models \varphi, \quad \nu' = \nu[Y := 0], \quad \text{and} \quad I(\ell'). \]
  “An action occurs, location may change, some clocks may be reset, time does not advance.”
A transition sequence of \( \mathcal{A} \) 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
- \( \langle \ell_0, \nu_0 \rangle \in C_{\text{ini}} \),
- for all \( i \in \mathbb{N} \), there is \( \xrightarrow{\lambda_{i+1}} \) in \( T(\mathcal{A}) \) 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 \( \mathcal{A} \)) 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

\[
\begin{align*}
\langle \text{off}, x = 0 \rangle & \xrightarrow{2.5} \langle \text{off}, x = 2.5 \rangle \xrightarrow{1.7} \langle \text{off}, x = 4.2 \rangle \\
& \xrightarrow{\text{press?}} \langle \text{light}, x = 0 \rangle \xrightarrow{2.1} \langle \text{light}, x = 2.1 \rangle \\
& \xrightarrow{\text{press?}} \langle \text{bright}, x = 2.1 \rangle \xrightarrow{10} \langle \text{bright}, x = 12.1 \rangle \\
& \xrightarrow{\text{press?}} \langle \text{off}, x = 12.1 \rangle \\
& \xrightarrow{\text{press?}} \langle \text{light}, x = 0 \rangle \xrightarrow{0} \langle \text{light}, x = 0 \rangle
\end{align*}
\]
Discussion: Set of Configurations

Recall the user model for our light controller:

- **“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:
  - \( \text{Conf}(A) = \{ \langle \ell, \nu \rangle \mid \ell \in L, \nu : X \rightarrow \text{Time, } \nu \models I(\ell) \} \)
  - \( C_{\text{ini}} = \{ \langle \ell_{\text{ini}}, \nu_0 \rangle \} \cap \text{Conf}(A) \)
  - Note: Being in \( \text{Conf}(A) \) doesn’t mean to be reachable.

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

Computation Path, Run
Computation Paths

- \( (\ell, \nu), t \) is called time-stamped configuration

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

- Time-stamped action transition: \( (\ell, \nu), t \xrightarrow{\alpha} (\ell', \nu'), t \)
  
  iff \( \alpha \in B \) and \( (\ell, \nu) \xrightarrow{\alpha} (\ell', \nu') \).

- A sequence of time-stamped configurations

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

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

- A computation path (or path) is a computation path starting at \( (\ell_0, \nu_0), 0 \)
  
  where \( (\ell_0, \nu_0) \in C_{\text{ini}} \).

Timelocks and Zeno Behaviour

\[ \ell \quad \begin{array}{c} \xrightarrow{x \leq 2} \end{array} \quad \ell' \quad \begin{array}{c} \xrightarrow{x \leq 3} \end{array} \]

- Timelock: \( (\ell, x = 0), 0 \xrightarrow{2} (\ell, x = 2), 2 \)
  
  \( (\ell', x = 0), 0 \xrightarrow{3} (\ell', x = 3), 3 \xrightarrow{\alpha^2} (\ell', x = 3), 3 \xrightarrow{\alpha^2} \ldots \)

- Zeno behaviour:
  
  \( (\ell, x = 0), 0 \xrightarrow{1/2} (\ell, x = 1/2), \frac{1}{2} \xrightarrow{1/4} (\ell, x = 3/4), \frac{3}{4} \ldots \)
  
  \( \frac{1/2^n}{\rightarrow} (\ell, x = (2^n - 1)/2^n), \frac{2^n - 1}{2^n} \ldots \)
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):
  \[ \forall t \in \text{Time} \exists i \in \mathbb{N}_0 : t < t_i \]

**Run**

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

**Example:**

\[ x \leq 2 \]
Example

\begin{center}
\begin{tikzpicture}
\node (l0) at (0,0) {\ell_0};
\node (l1) at (2,0) {\ell_1};
\draw[->] (l0) -- node[above] {$a!$} (l1);
\draw[->] (l0) -- node[below] {$x \geq 10$} (l1);
\end{tikzpicture}
\end{center}

$s?$, $x < 10, x := 0$

References