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

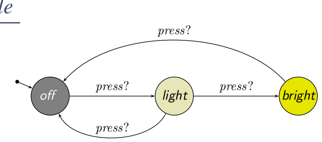
$obs : \mathsf{Time} \to \mathscr{D}(obs)$

$\langle obs_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle obs_1, \nu_1 \rangle, t_1 \ldots$

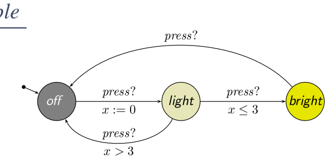
- **Automatic Verification**...
- ...whether TA satisfies DC formula, observer-based

Example: Off/Light/Bright

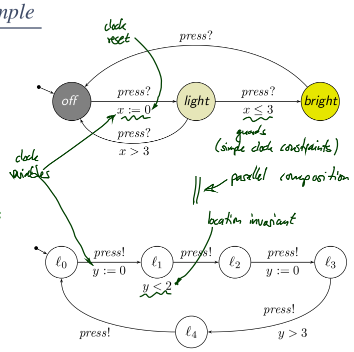
Example



Example

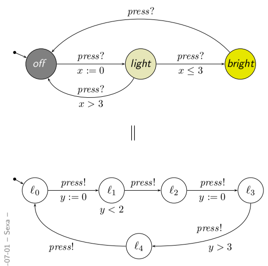


Example



User:

Example Cont'd

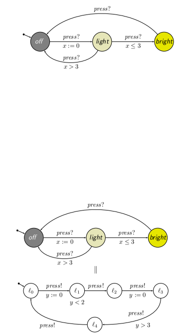


Problems:

- 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

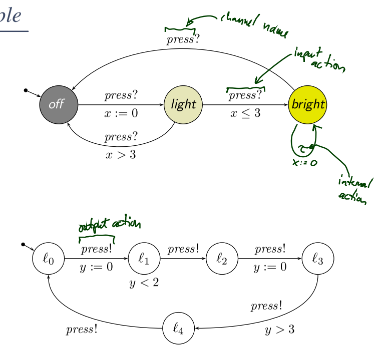
To define timed automata formally, we need the following sets of symbols:

- 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.
- $(\alpha, \beta \in)$ $Act := \{a? \mid a \in \text{Chan}\} \cup \{a! \mid a \in \text{Chan}\} \cup \{\tau\}$ 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: $\text{Chan}_{?!} = Act$.

Example



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:

$$\varphi ::= x \sim c \mid x - y \sim c \mid \varphi_1 \wedge \varphi_2$$

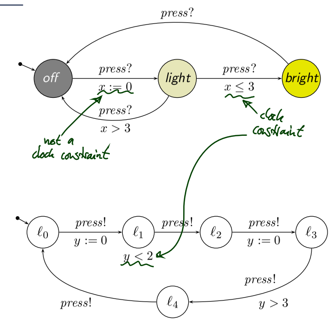
where

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

we may use $x = c$ (or $x = y$) as an abbreviation for $x \leq c \wedge x \geq c$ (or $x - y \leq 0 \wedge x - y \geq 0$)

- Clock constraints of the form $x - y \sim c$ are called **difference constraints**.

Example



Timed Automaton

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

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

where

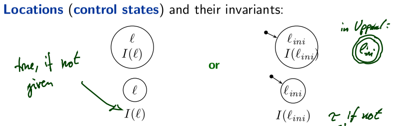
- $(\ell \in)\ L$ is a finite set of **locations** (or **control states**),
- $B \subseteq$ Chan,
- X is a finite set of clocks,
- $I : L \to \Phi(X)$ assigns to each location a clock constraint, its **invariant**,
- $E \subseteq L \times B_{?!} \times \Phi(X) \times 2^X \times L$ a finite set of **directed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

  Edges $(\ell, \alpha, \varphi, Y, \ell')$ from location ℓ to ℓ' are labelled with an **action** α, a **guard** φ, and a set Y of clocks that will be **reset**.
- ℓ_{ini} is the **initial location**.

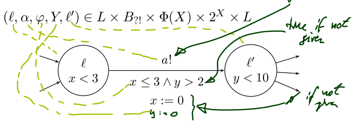
Graphical Representation of Timed Automata

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

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



- **Edges**: $(\ell, \alpha, \varphi, Y, \ell') \in L \times B_{?!} \times \Phi(X) \times 2^X \times L$



Pure TA Operational Semantics

Clock Valuations

- Let X be a set of clocks. A **valuation** ν **of clocks** in X is a mapping

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

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

- Let φ be a clock constraint.
  The **satisfaction** relation between clock valuations ν and clock constraints φ, 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 $\nu(x) - \nu(y) \sim c$
  - $\nu \models \varphi_1 \wedge \varphi_2$ iff $\nu \models \varphi_1$ and $\nu \models \varphi_2$

Clock Valuations

- Let X be a set of clocks. A **valuation ν of clocks** in X is a mapping

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

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

- Let φ be a clock constraint.
The **satisfaction** relation between clock valuations ν and clock constraints φ, 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 $\nu(x) - \nu(y) \sim c$
  - $\nu \models \varphi_1 \wedge \varphi_2$ iff $\nu \models \varphi_1$ and $\nu \models \varphi_2$
- Two clock constraints φ_1 and φ_2 are called (**logically**) **equivalent** if and only if for all clock valuations ν, 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 ν be a valuation of clocks in X and $t \in$ 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**: $t = 0$.

Operational Semantics of TA

Definition 4.4. The **operational semantics** of a timed automaton

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

is defined by the (labelled) transition system

$$\mathcal{T}(\mathcal{A}) = (Conf(\mathcal{A}), \text{Time} \cup B_{?!}, \{\xrightarrow{\lambda} \mid \lambda \in \text{Time} \cup B_{?!}\}, C_{ini})$$

where

- $Conf(\mathcal{A}) = \{\langle \ell, \nu \rangle \mid \ell \in L, \nu : X \to \text{Time}, \ \nu \models I(\ell)\}$
- Time $\cup B_{?!}$ are the transition labels,
- there are **delay transition relations**
$\langle \ell, \nu \rangle \xrightarrow{\lambda} \langle \ell', \nu' \rangle, \lambda \in$ Time
and **action transition relations**
$\langle \ell, \nu \rangle \xrightarrow{\lambda} \langle \ell', \nu' \rangle, \lambda \in B_{?!}$. (→ later slides)
- $C_{ini} = \{\langle \ell_{ini}, \nu_0 \rangle\} \cap Conf(\mathcal{A})$ with $\nu_0(x) = 0$ for all $x \in X$ is the set of **initial configurations**.

Operational Semantics of TA Cont'd

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

$$\mathcal{T}(\mathcal{A}) = (Conf(\mathcal{A}), \text{Time} \cup B_{?!}, \{\xrightarrow{\lambda} \mid \lambda \in \text{Time} \cup B_{?!}\}, C_{ini})$$

- **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** $t \in$ 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

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

"An action occurs, location may change, some clocks may be reset, **time does not advance**."

Transition Sequences, Reachability

- A **transition sequence** of $\mathcal{A}$ is any finite or infinite sequence of the form

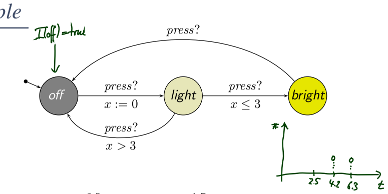
$$\langle \ell_0, \nu_0 \rangle \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle \xrightarrow{\lambda_3} \ldots$$

with
  - $\langle \ell_0, \nu_0 \rangle \in C_{ini}$,
  - for all $i \in \mathbb{N}$, there is $\xrightarrow{\lambda_{i+1}}$ in $\mathcal{T}(\mathcal{A})$ 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** ℓ is called **reachable** if and only if **any** configuration $\langle \ell, \nu \rangle$ is reachable, i.e. there exists a valuation ν such that $\langle \ell, \nu \rangle$ is reachable.

Example



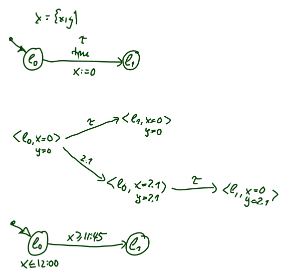
$\langle \underbrace{off}_{\ell_0}, \underbrace{x = 0}_{\nu_0} \rangle \xrightarrow{2.5} \langle off, x = 2.5 \rangle \xrightarrow{1.7} \langle off, x = 4.2 \rangle$

$\xrightarrow{press?} \langle light, x = 0 \rangle \xrightarrow{2.1} \langle light, x = 2.1 \rangle$

$\xrightarrow{press?} \langle bright, x = 2.1 \rangle \xrightarrow{10} \langle bright, x = 12.1 \rangle$

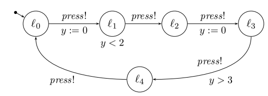
$\xrightarrow{press?} \langle off, x = 12.1 \rangle$

$\xrightarrow{press?} \langle light, x = 0 \rangle \xrightarrow{0} \langle light, x = 0 \rangle$



Discussion: Set of Configurations

Recall the user model for our light controller:



- **"Good" configurations**:

$$\langle \ell_1, y = 0 \rangle, \langle \ell_1, y = 1.9 \rangle, \quad \langle \ell_2, y = 1000 \rangle,$$

$$\langle \ell_2, y = 0.5 \rangle, \quad \langle \ell_3, y = 27 \rangle$$

- **"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 $\mathcal{A}$ to $\mathcal{T}(\mathcal{A})$.
    "Bad" configurations are not even configurations!
- **Recall Definition 4.4**:
  - $Conf(\mathcal{A}) = \{\langle \ell, \nu \rangle \mid \ell \in L, \nu : X \to \text{Time}, \nu \models I(\ell)\}$
  - $C_{ini} = \{\langle \ell_{ini}, \nu_0 \rangle\} \cap Conf(\mathcal{A})$
- **Note**: Being in $Conf(\mathcal{A})$ 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

$$Conf(\mathcal{A}) = \{\langle \ell, \nu \rangle \mid \ell \in L, \nu : X \to \text{Time} \cancel{, \nu \models I(\ell)}\}$$

  - "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

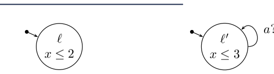
Computation Paths

- $\langle \ell, \nu \rangle, t$ is called **time-stamped configuration**
- **time-stamped delay transition**: $\langle \ell, \nu \rangle, t \xrightarrow{t'} \langle \ell, \nu + t' \rangle, t + t'$
  iff $t' \in$ 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

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

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**:

$$\langle \ell, x = 0 \rangle, 0 \xrightarrow{2} \langle \ell, x = 2 \rangle, 2$$

$$\langle \ell', x = 0 \rangle, 0 \xrightarrow{3} \langle \ell', x = 3 \rangle, 3 \xrightarrow{a?} \langle \ell', x = 3 \rangle, 3 \xrightarrow{a?} \dots$$

- **Zeno** behaviour:

$$\langle \ell, x = 0 \rangle, 0 \xrightarrow{1/2} \langle \ell, x = 1/2 \rangle, \frac{1}{2} \xrightarrow{1/4} \langle \ell, x = 3/4 \rangle, \frac{3}{4} \dots$$

$$\xrightarrow{1/2^n} \langle \ell, x = (2^n - 1)/2^n \rangle, \frac{2^n - 1}{2^n} \dots$$

Real-Time Sequence

Definition 4.9. An infinite sequence

$$t_0, t_1, t_2, \ldots$$

of values $t_i \in \mathsf{Time}$ 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**):
$$\forall\, t \in \mathsf{Time}\ \exists\, i \in \mathbb{N}_0 : t < t_i$$

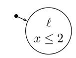
Run

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 $\mathcal{A}$

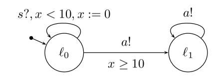
$$\xi = \langle \ell_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle, t_1 \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle, t_2 \xrightarrow{\lambda_3} \ldots$$

where $(t_i)_{i \in \mathbb{N}_0}$ is a real-time sequence.
If $\langle \ell_0, \nu_0 \rangle \in C_{ini}$ and $t_0 = 0$, then we call ξ a **run** of $\mathcal{A}$.

Example:



Example



References

[Behrmann et al., 2004] Behrmann, G., David, A., and Larsen, K. G. (2004). A tutorial on uppaal 2004-11-17. Technical report, Aalborg University, Denmark.

[Olderog and Dierks, 2008] Olderog, E.-R. and Dierks, H. (2008). Real-Time Systems - Formal Specification and Automatic Verification. Cambridge University Press.