Timed Automata and Logics for Real-time Systems

Luca Aceto
ICE-TCS, School of Computer Science
Reykjavik University, Iceland
luca@ru.is and luca.aceto@gmail.com
Goals of the course

Learning outcomes

At the end of the course, you will

- be familiar with the basic theory of timed automata,
- be able to model and verify real-time systems using UPPAAL,
- be familiar with
  - some of the behavioural equivalences over time automata and
  - some of the logics for real-time systems and their connections with behavioural equivalences.

The course will be evaluated via a small-group project.
Content of the first part of the course

- labelled transition systems with time
- timed automata
- timed and untimed bisimilarity
- timed and untimed language equivalence
- region graph and the reachability problem
- networks of timed automata
- model checking of basic properties of timed automata
Timeouts in protocols:
- In CCS, say, timeouts can be modelled using nondeterminism.
- Enough to prove that the protocol is safe.
- Maybe too abstract for certain questions. (What is the average time to deliver the message?)

Many real-life systems depend on timing:
- Real-time controllers (production lines, computers in cars, railway crossings).
- Embedded systems (mobile phones, remote controllers, digital watch).
- ...

Timeouts in protocols:
- In CCS, say, timeouts can be modelled using nondeterminism.
- Enough to prove that the protocol is safe.
- Maybe too abstract for certain questions. (What is the average time to deliver the message?)

Many real-life systems depend on timing:
- Real-time controllers (production lines, computers in cars, railway crossings).
- Embedded systems (mobile phones, remote controllers, digital watch).
- ...
Timed (labelled) transition system (TLTS)

TLTS is a triple \((\text{Proc}, \text{Act}, \{a \rightarrow \mid a \in \text{Act}\})\) where

- \(\text{Proc}\) is a set of states (or processes),
- \(\text{Act} = \mathbb{N} \cup \mathbb{R}^{\geq 0}\) is a set of actions (consisting of labels and time-elapsing steps), and
- for every \(a \in \text{Act}\), \(a \rightarrow \subseteq \text{Proc} \times \text{Proc}\) is a binary relation on states called the transition relation.

We write

- \(s \xrightarrow{a} s'\) if \(a \in \mathbb{N}\) and \((s, s') \in a \rightarrow\), and
- \(s \xrightarrow{d} s'\) if \(d \in \mathbb{R}^{\geq 0}\) and \((s, s') \in d \rightarrow\).
Typical requirements on the delay transitions

- **(Determinism)** If $s \xrightarrow{d} s'$ and $s \xrightarrow{d} s''$ ($d \in \mathbb{R}^{\geq 0}$) then $s' = s''$.
- **(Zero delay)** $s \xrightarrow{0} s'$ if, and only if, $s = s'$.
- **(Additivity 1)** If $s \xrightarrow{d} s' \xrightarrow{d'} s''$ then $s \xrightarrow{d + d'} s''$.
- **(Additivity 2)** If $s \xrightarrow{d} s''$ and $e \leq d$ then $s \xrightarrow{e} s' \xrightarrow{d-e} s''$ for some $s'$.

Why are these reasonable requirements?
How to Describe Timed Transition Systems?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>unknown entity</td>
<td>known entity</td>
</tr>
</tbody>
</table>
How to Describe Timed Transition Systems?

Syntax
unknown entity

Semantics
known entity

CCS

Labelled Transition Systems
How to Describe Timed Transition Systems?

Syntax
unknown entity

→

Semantics
known entity

CCS

Labelled Transition Systems

???

Timed Transition Systems
### Syntax

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>unknown entity</td>
<td>known entity</td>
</tr>
<tr>
<td>CCS</td>
<td>Labelled Transition Systems</td>
</tr>
<tr>
<td>???</td>
<td>Timed Transition Systems</td>
</tr>
</tbody>
</table>

**Timed Automata** [Alur, Dill’90]

Finite-state automata equipped with clocks.
Example: Light switch

- **Off**
  - Press $x := 0$

- **Light**
  - Press $x \leq 1.4$

- **Bright**
  - Press $x > 1.4$
Let $C = \{x, y, \ldots\}$ be a finite set of clocks.

Set $\mathcal{B}(C)$ of clock constraints over $C$

$\mathcal{B}(C)$ is defined by the following abstract syntax

$$g, g_1, g_2 \ ::= x \sim n \mid x - y \sim n \mid g_1 \land g_2$$

where $x, y \in C$ are clocks, $n \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$.

Example: $x \leq 3 \land y > 0 \land y - x = 2$
Clock Valuation

Clock valuation $\nu$ is a function $\nu : C \rightarrow \mathbb{R}_{\geq 0}$. 

Let $\nu$ be a clock valuation. Then $\nu + d$ is a clock valuation for any $d \in \mathbb{R}_{\geq 0}$ and it is defined by $(\nu + d)(x) = \nu(x) + d$ for all $x \in C$.
Clock Valuation

Clock valuation $\nu$ is a function $\nu : C \rightarrow \mathbb{R}^{\geq 0}$.

Let $\nu$ be a clock valuation. Then

- $\nu + d$ is a clock valuation for any $d \in \mathbb{R}^{\geq 0}$ and it is defined by
  $$(\nu + d)(x) = \nu(x) + d$$ for all $x \in C$.

- $\nu[r]$ is a clock valuation for any $r \subseteq C$ and it is defined by
  $$\nu[r](x) \begin{cases} 
    0 & \text{if } x \in r \\
    \nu(x) & \text{otherwise}. 
  \end{cases}$$
Evaluation of clock constraints $(v \models g)$

<table>
<thead>
<tr>
<th>Expression</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v \models x &lt; n$</td>
<td>$v(x) &lt; n$</td>
</tr>
<tr>
<td>$v \models x \leq n$</td>
<td>$v(x) \leq n$</td>
</tr>
<tr>
<td>$v \models x = n$</td>
<td>$v(x) = n$</td>
</tr>
<tr>
<td>$v \models x - y &lt; n$</td>
<td>$v(x) - v(y) &lt; n$</td>
</tr>
<tr>
<td>$v \models x - y \leq n$</td>
<td>$v(x) - v(y) \leq n$</td>
</tr>
<tr>
<td>$v \models g_1 \land g_2$</td>
<td>$v \models g_1$ and $v \models g_2$</td>
</tr>
</tbody>
</table>
Syntax of Timed Automata

**Definition**

A **timed automaton** over a set of clocks $C$ and a set of labels $N$ is a tuple

$$(L, \ell_0, E, I)$$

where

- $L$ is a finite set of **locations**
- $\ell_0 \in L$ is the **initial location**
- $E \subseteq L \times \mathcal{B}(C) \times N \times 2^C \times L$ is the set of **edges**
- $I : L \rightarrow \mathcal{B}(C)$ assigns **invariants** to locations.

We usually write $\ell \overset{g, a, r}{\rightarrow} \ell'$ whenever $(\ell, g, a, r, \ell') \in E$. 

Example: Hammer

```
free
  ^
  |
  v

start

x:=0, y:=0

->

done

y>=5

busy
  ^
  |
  v

hit

x>=1

x:=0
```
Semantics of Timed Automata

Let $A = (L, \ell_0, E, I)$ be a timed automaton.

Timed transition system generated by $A$

$T(A) = (Proc, Act, \{a \rightarrow | a \in Act\})$ where

- $Proc = L \times (C \rightarrow \mathbb{R}^{\geq 0})$, i.e. states are of the form $(\ell, v)$ where $\ell$ is a location and $v$ a valuation
- $Act = N \cup \mathbb{R}^{\geq 0}$
- $\rightarrow$ is defined as follows:

$(\ell, v) \xrightarrow{a} (\ell', v')$ if there is $(\ell \xrightarrow{g,a,r} \ell') \in E$ s.t. $v \models g$ and $v' = v[r]$

$(\ell, v) \xrightarrow{d} (\ell, v + d)$ for all $d \in \mathbb{R}^{\geq 0}$ s.t. $v \models I(\ell)$ and $v + d \models I(\ell)$
A timed automaton and a fragment of its associated TLTS

\[
\begin{align*}
&x \leq 2 \quad \bigcirc \quad x \leq 1 \\
&(\ell_0, [x = 0]) \overset{0.6}{\rightarrow} (\ell_0, [x = 0.6]) \overset{0.4}{\rightarrow} (\ell_0, [x = 1]) \overset{0.3}{\rightarrow} (\ell_0, [x = 1.3]) \overset{0.7}{\rightarrow} (\ell_0, [x = 2])
\end{align*}
\]
Example: A small jobshop

Can you give a fragment of its associated TLTS?
Networks of Timed Automata

Timed Automata in Parallel

Intuition in CCS

Let \( C \) be a set of clocks and \( Chan \) a set of channels. We let \( Act = N \cup R_{\geq 0} \) where

\[
N = \{ c! \mid c \in Chan \} \cup \{ c? \mid c \in Chan \} \cup \{ \tau \}.
\]

Let \( A_i = (L_i, l_{i0}, E_i, I_i) \) be timed automata for \( 1 \leq i \leq n \).

Networks of Timed Automata

We call \( A = A_1 \parallel A_2 \parallel \cdots \parallel A_n \) a network of timed automata.

\[
(a.Nil \mid a.Nil) \setminus \{a\}
\]
Networks of Timed Automata

Timed Automata in Parallel

Intuition in CCS

Let $C$ be a set of clocks and $Chan$ a set of channels. We let $Act = N \cup R^{\geq 0}$ where
- $N = \{c! \mid c \in Chan\} \cup \{c? \mid c \in Chan\} \cup \{\tau\}$.

Let $A_i = (L_i, \ell_{i0}, E_i, I_i)$ be timed automata for $1 \leq i \leq n$.

Networks of Timed Automata
We call $A = A_1 | A_2 | \cdots | A_n$ a network of timed automata.
Example: Hammer, Worker, Nail

H: 

```
free

start? x:=0, y:=0

hit! x:=0

y>=5

done?
```

W: 

```
rest

start! z:=0

work

done! z:=0

z<=60
```

N: 

```
up

hit?

down
```

Luca Aceto
ICE-TCS, School of Computer Science Reykjavik
Example: Hammer, Worker, Nail

**H:**
- **Free:** x := 0, y := 0, start?
- **Busy:** x ≥ 1, hit!, y ≥ 5, done?

**W:**
- **Rest:** z := 0, start!
- **Work:** z ≤ 60, done!

**N:**
- **Hit:** half
- **Down:** τ

Luca Aceto
ICE-TCS, School of Computer Science Reykjavik
luca@ru.is and luca.aceto@gmail.com

Timed Automata and Logics for Real-time Systems 18 / 43
Example: Hammer, Worker, Nail

**H:** free
- **start?**
- **x:=0, y:=0**
- **y≥5**
- **done?**
- **hit!**

**W:** rest
- **start!**
- **z:=0**
- **z≥10**
- **done!**

**N:** up
- **half**
- **down**
- **τ**
- **hit?**
Timed Transition System Generated by $A = A_1 | \cdots | A_n$

\[
T(A) = (\text{Proc}, \text{Act}, \{ \xrightarrow{a} | a \in \text{Act} \}) \text{ where }
\]

- $\text{Proc} = (L_1 \times L_2 \times \cdots \times L_n) \times (C \rightarrow \mathbb{R}_{\geq 0})$, i.e. states are of the form $(((\ell_1, \ell_2, \ldots, \ell_n), v))$ where $\ell_i$ is a location in $A_i$

- $\text{Act} = \{\tau\} \cup \mathbb{R}_{\geq 0}$

- $\xrightarrow{}$ is defined as follows:

\[
((\ell_1, \ldots, \ell_i, \ldots, \ell_n), v) \xrightarrow{\tau} ((\ell_1, \ldots, \ell'_i, \ldots, \ell_n), v') \text{ if there is } (\ell_i \xrightarrow{g, \tau, r} \ell'_i) \in E_i \text{ s.t. } v = g \text{ and } v' = v[r] \text{ and } v' \models l_i(\ell'_i) \land \bigwedge_{k \neq i} l_k(\ell_k)
\]

\[
((\ell_1, \ldots, \ell_n), v) \xrightarrow{d} ((\ell_1, \ldots, \ell_n), v + d) \text{ for all } d \in \mathbb{R}_{\geq 0} \text{ s.t. } v \models \bigwedge_k l_k(\ell_k) \text{ and } v + d \models \bigwedge_k l_k(\ell_k)
\]
Continuation

$$(((\ell_1, \ldots, \ell_i, \ldots, \ell_j, \ldots, \ell_n), v) \xrightarrow{\tau} (((\ell_1, \ldots, \ell'_i, \ldots, \ell'_j, \ldots, \ell_n), v'))$$

if $i \neq j$ and there are $(\ell_i \xrightarrow{g_i, a!} \ell'_i) \in E_i$ and $(\ell_j \xrightarrow{g_j, a?} \ell'_j) \in E_j$ s.t.

$v \models g_i \land g_j$ and $v' = v[r_i \cup r_j]$ and $v' \models l_i(\ell'_i) \land l_j(\ell'_j) \land \bigwedge_{k \neq i, j} l_k(\ell_k)$
The light switch and a fast user

Off

press? \( x > 14 \)

press? \( x := 0 \)

Light

press? \( x \leq 14 \)

press?

Bright

y \leq 0

U

press!

y := 0

U'

y \leq 3

press!

\( y = 3 \)

\( y := 0 \)
A fragment of the TLTS for the previous network

\[(\text{Off, } U) \xrightarrow{\tau} (\text{Light, } U') \xrightarrow{3} (\text{Light, } U') \xrightarrow{\tau} (\text{Bright, } U')\]

\[(\text{Off, } U') \xleftarrow{3} (\text{Off, } U') \xleftarrow{\tau} (\text{Bright, } U')\]
The lazy Worker and his demanding Employer

---

The Lazy Worker and his Demanding Employer

- **Definition**
- **Example**
- **Logical Properties in UPPAAL**

---

The diagram illustrates a Timed Automaton with two states: Rw (Start) and Ww (Hit). The transition conditions are:

- **Rw to Ww**: 
  - Start: $x := 0$
  - $y := 0$
  - $y \geq 1$

- **Ww to Rw**: 
  - $x \leq 60$
  - $y := 0$

The states are:

- **Rw**: $x \geq 5$
- **Ww**: $x = 0$

---

The diagram also shows another Timed Automaton with two states: Re (Start) and We (End). The transition conditions are:

- **Re to We**: 
  - Start: $z := 0$
  - $y \leq 4$

- **We to Re**: 
  - $z \leq 10$
  - $y \geq 40$

The states are:

- **Re**: $z \leq 10$
- **We**: $z = 0$

---

Luca Aceto
ICE-TCS, School of Computer Science
Reykjavik
luca@ru.is
luca.aceto@gmail.com

Timed Automata and Logics for Real-time Systems 23 / 43
Logic for Timed Automata in UPPAAL

Let $\phi$ and $\psi$ be local properties (checkable locally in a given state). Example: $(\text{H.busy} \land W.\text{rest} \land 20 \leq z \leq 30)$

UPPAAL can check the following formulae (subset of TCTL)

- $A[]\phi$ — invariantly $\phi$
- $E\langle\rangle\phi$ — possibly $\phi$
- $A\langle\rangle\phi$ — always eventually $\phi$
- $E[]\phi$ — potentially always $\phi$
- $\phi \rightarrow \psi$ — $\phi$ always leads to $\psi$ (same as $A[](\phi \implies A\langle\rangle\psi)$)

Legend:
- $A$ and $E$ are so called path quantifiers, and
- $[]$ and $\langle\rangle$ quantify over states of a selected path.
Fact

Even very simple timed automata generate timed transition systems with infinitely (even uncountably) many reachable states.
Automatic Verification of Timed Automata

Fact

Even very simple timed automata generate timed transition systems with infinitely (even uncountably) many reachable states.

Question

Is any automatic verification approach (like bisimilarity checking, model checking or reachability analysis) possible at all?
**Fact**

Even very simple timed automata generate timed transition systems with infinitely (even uncountably) many reachable states.

**Question**

Is any automatic verification approach (like bisimilarity checking, model checking or reachability analysis) possible at all?

**Answer**

Yes, using region graph techniques.

Key idea: infinitely many clock valuations can be categorized into finitely many equivalence classes.
Preliminaries

Let $d \in \mathbb{R}_{\geq 0}$. Then

- let $\lfloor d \rfloor$ be the integer part of $d$, and
- let $\text{frac}(d)$ be the fractional part of $d$.

Any $d \in \mathbb{R}_{\geq 0}$ can be now written as $d = \lfloor d \rfloor + \text{frac}(d)$.

Example: $\lfloor 2.345 \rfloor = 2$ and $\text{frac}(2.345) = 0.345$. 
Let $d \in \mathbb{R}_{\geq 0}$. Then

- let $\lfloor d \rfloor$ be the integer part of $d$, and
- let $\text{frac}(d)$ be the fractional part of $d$.

Any $d \in \mathbb{R}_{\geq 0}$ can be now written as $d = \lfloor d \rfloor + \text{frac}(d)$.

Example: $\lfloor 2.345 \rfloor = 2$ and $\text{frac}(2.345) = 0.345$.

Let $A$ be a timed automaton and $x \in C$ be a clock. We define $c_x \in \mathbb{N}$ as the largest constant with which the clock $x$ is ever compared either in the guards or in the invariants present in $A$. 
Let $v, v' : C \rightarrow \mathbb{R}^\geq 0$ be clock valuations. Let $\sim$ denote untimed bisimilarity of timed transition systems.

**Our Aim**

Define an equivalence relation $\equiv$ over clock valuations such that:

1. $v \equiv v'$ implies $(\ell, v) \sim (\ell, v')$ for any location $\ell$
2. $\equiv$ has only finitely many equivalence classes.
Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations $v$ and $v'$ are equivalent ($v \equiv v'$) iff

1. for all $x \in C$ such that $v(x) \leq c_x$ or $v'(x) \leq c_x$ we have $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$

2. for all $x \in C$ such that $v(x) \leq c_x$ we have $\frac{v(x)}{c_x} = 0$ iff $\frac{v'(x)}{c_x} = 0$

3. for all $x, y \in C$ such that $v(x) \leq c_x$ and $v(y) \leq c_y$ we have $\frac{v(x)}{c_x} \leq \frac{v(y)}{c_y}$ iff $\frac{v'(x)}{c_x} \leq \frac{v'(y)}{c_y}$
Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations \( v \) and \( v' \) are equivalent (\( v \equiv v' \)) iff

1. for all \( x \in C \) such that \( v(x) \leq c_x \) or \( v'(x) \leq c_x \) we have

\[
\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor
\]
Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations $v$ and $v'$ are equivalent ($v \equiv v'$) iff

1. for all $x \in C$ such that $v(x) \leq c_x$ or $v'(x) \leq c_x$ we have
   \[\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor\]

2. for all $x \in C$ such that $v(x) \leq c_x$ we have
   \[\frac{v(x)}{1} = 0 \iff \frac{v'(x)}{1} = 0\]
Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations $v$ and $v'$ are equivalent ($v \equiv v'$) iff

1. for all $x \in C$ such that $v(x) \leq c_x$ or $v'(x) \leq c_x$ we have
   \[ \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \]

2. for all $x \in C$ such that $v(x) \leq c_x$ we have
   \[ \frac{v(x)}{1} = 0 \iff \frac{v'(x)}{1} = 0 \]

3. for all $x, y \in C$ such that $v(x) \leq c_x$ and $v(y) \leq c_y$ we have
   \[ \frac{v(x)}{1} \leq \frac{v(y)}{1} \iff \frac{v'(x)}{1} \leq \frac{v'(y)}{1} \]
Let $v$ be a clock valuation. The $\equiv$-equivalence class represented by $v$ is denoted by $[v]$ and defined by $[v] = \{ v' \mid v' \equiv v \}$.

Definition of a Region

An $\equiv$-equivalence class $[v]$ represented by some clock valuation $v$ is called a region.
Regions

Let \( v \) be a clock valuation. The \( \equiv \)-equivalence class represented by \( v \) is denoted by \([v]\) and defined by \([v] = \{ v' \mid v' \equiv v \}\).

Definition of a Region

An \( \equiv \)-equivalence class \([v]\) represented by some clock valuation \( v \) is called a region.

Theorem

For every location \( \ell \) and any two valuations \( v \) and \( v' \) from the same region (\( v \equiv v' \)) it holds that

\[
(\ell, v) \sim (\ell, v')
\]

where \( \sim \) stands for untimed bisimilarity.
Symbolic States and Region Graph

\[ \text{state } (\ell, v) \rightsquigarrow \text{symbolic state } (\ell, [v]) \]

Note: \( v \equiv v' \) implies that \((\ell, [v]) = (\ell, [v'])\).

Region Graph

Region graph of a timed automaton \( A \) is an unlabelled (and untimed) transition system where

- states are symbolic states
- \( \rightsquigarrow \) on symbolic states is defined as follows:
  
  \( (\ell, [v]) \rightsquigarrow (\ell', [v']) \) iff \((\ell, v) \xrightarrow{a} (\ell', v')\) for some label \( a \)

  \( (\ell, [v]) \rightsquigarrow (\ell, [v']) \) iff \((\ell, v) \xrightarrow{d} (\ell, v')\) for some \( d \in \mathbb{R}_{\geq 0} \)
Symbolic States and Region Graph

\[
\text{state } (\ell, v) \sim\rightarrow \text{ symbolic state } (\ell, [v])
\]

Note: \( v \equiv v' \) implies that \((\ell, [v]) = (\ell, [v'])\).

**Region Graph**

Region graph of a timed automaton \( A \) is an unlabelled (and untimed) transition system where

- states are symbolic states
- \( \sim\rightarrow \) on symbolic states is defined as follows:
  
  \[
  (\ell, [v]) \sim\rightarrow (\ell', [v']) \iff \ell, v \xrightarrow{a} (\ell', v') \text{ for some label } a
  \]
  
  \[
  (\ell, [v]) \sim\rightarrow (\ell, [v']) \iff \ell, v \xrightarrow{d} (\ell, v') \text{ for some } d \in \mathbb{R}_{\geq 0}
  \]

**Fact**

A region graph of any timed automaton is finite.
Partitioning of the valuations for a simple timed automaton
Symbolic exploration of the simple timed automaton
Application of Region Graphs to Reachability

We write \((\ell, v) \rightarrow (\ell', v')\) whenever
- \((\ell, v) \xrightarrow{a} (\ell', v')\) for some label \(a\), or
- \((\ell, v) \xrightarrow{d} (\ell', v')\) for some \(d \in \mathbb{R}^{\geq 0}\).

Reachability Problem for Timed Automata

**Instance (input):** Automaton \(A = (L, \ell_0, E, I)\) and a state \((\ell, v)\).

**Question:** Is it true that \((\ell_0, v_0) \xrightarrow{*} (\ell, v)\) (where \(v_0(x) = 0\) for all \(x \in C\))?
Application of Region Graphs to Reachability

We write \((\ell, v) \rightarrow (\ell', v')\) whenever
- \((\ell, v) \xrightarrow{a} (\ell', v')\) for some label \(a\), or
- \((\ell, v) \xrightarrow{d} (\ell', v')\) for some \(d \in \mathbb{R}^{\geq 0}\).

Reachability Problem for Timed Automata

**Instance (input):** Automaton \(A = (L, \ell_0, E, I)\) and a state \((\ell, v)\).

**Question:** Is it true that \((\ell_0, v_0) \rightarrow^* (\ell, v)\) (where \(v_0(x) = 0\) for all \(x \in C\))?}

Reduction of Timed Automata Reachability to Region Graphs

Reachability for timed automata is decidable because

\((\ell_0, v_0) \rightarrow^* (\ell, v)\) in a timed automaton if and only if
\((\ell_0, [v_0]) \xrightarrow{}^* (\ell, [v])\) in its (finite) region graph.
Applicability of Region Graphs

Pros

Region graphs provide a natural abstraction which enables to prove decidability of e.g.

- reachability
- timed and untimed bisimilarity
- untimed language equivalence and language emptiness.
Applicability of Region Graphs

Pros

Region graphs provide a natural abstraction which enables to prove decidability of e.g.

- reachability
- timed and untimed bisimilarity
- untimed language equivalence and language emptiness.

Cons

Region graphs have too large state spaces. State explosion is exponential in

- the number of clocks
- the maximal constants appearing in the guards.
Zones and Zone Graphs

Zones provide a more efficient representation of symbolic state spaces. A number of regions can be described by one zone.

Zone

A zone is described by a clock constraint $g \in \mathcal{B}(C)$.

$$[g] = \{ \nu \mid \nu \models g \}$$
Zones and Zone Graphs

Zones provide a more efficient representation of symbolic state spaces. A number of regions can be described by one zone.

**Zone**

A zone is described by a clock constraint \( g \in B(C) \).

\[
[g] = \{ v \mid v \models g \}
\]

**Region Graphs**

symbolic state: \((\ell, [v])\)

where \( v \) is a clock valuation

**Zone Graphs**

symbolic state: \((\ell, [g])\)

where \( g \) is a clock constraint
Zones and Zone Graphs

Zones provide a more efficient representation of symbolic state spaces. A number of regions can be described by one zone.

**Zone**

A zone is described by a clock constraint $g \in \mathcal{B}(C)$.

\[
[g] = \{ \nu \mid \nu \models g \}
\]

**Region Graphs**

symbolic state: $(\ell, [\nu])$
where $\nu$ is a clock valuation

**Zone Graphs**

symbolic state: $(\ell, [g])$
where $g$ is a clock constraint

A zone is usually represented (and stored in the memory) as DBM (Difference Bound Matrix).
Timed Bisimilarity

Let $A_1$ and $A_2$ be timed automata.

**Timed Bisimilarity**

We say that $A_1$ and $A_2$ are **timed bisimilar** iff the transition systems $T(A_1)$ and $T(A_2)$ generated by $A_1$ and $A_2$ are strongly bisimilar.

Remark: both

- $\xrightarrow{a}$ for $a \in N$ and
- $\xrightarrow{d}$ for $d \in \mathbb{R}_{\geq 0}$

are considered as normal (visible) transitions.
Example of Timed Bisimilar Automata

\[
\begin{align*}
A & \xrightarrow{a} B & x=1 \\
B & \xrightarrow{a} C & x \leq 2 \\
C & \xrightarrow{a} C' & x := 0
\end{align*}
\]

\[
\begin{align*}
A' & \xrightarrow{a} B' & x=1 \\
B' & \xrightarrow{a} C' & x \leq 1 \\
C' & \xrightarrow{a} C' & x := 0
\end{align*}
\]
Example of Timed Non-Bisimilar Automata

\[ A \xrightarrow{a} B \xrightarrow{a} C \]

\[ x \leq 1 \]
\[ x := 0 \]
\[ x \leq 3 \]

\[ A' \xrightarrow{a} B' \xrightarrow{a} C' \]

\[ x \leq 2 \]
\[ x := 0 \]
\[ x \leq 3 \]
Untimed Bisimilarity

Let $A_1$ and $A_2$ be timed automata. Let $\epsilon$ be a new (fresh) action.

**Untimed Bisimilarity**

We say that $A_1$ and $A_2$ are untimed bisimilar iff the transition systems $T(A_1)$ and $T(A_2)$ generated by $A_1$ and $A_2$ where every transition of the form $d \rightarrow$ for $d \in \mathbb{R}_{\geq 0}$ is replaced with $\epsilon \rightarrow$ are strongly bisimilar.

**Remark:**
- $a \rightarrow$ for $a \in N$ is treated as a visible transition, while
- $d \rightarrow$ for $d \in \mathbb{R}_{\geq 0}$ are all labelled by a single visible action $\epsilon \rightarrow$. 
Untimed Bisimilarity

Let $A_1$ and $A_2$ be timed automata. Let $\epsilon$ be a new (fresh) action.

Untimed Bisimilarity

We say that $A_1$ and $A_2$ are untimed bisimilar iff the transition systems $T(A_1)$ and $T(A_2)$ generated by $A_1$ and $A_2$ where every transition of the form $d \rightarrow$ for $d \in \mathbb{R}^{\geq 0}$ is replaced with $\epsilon \rightarrow$ are strongly bisimilar.

Remark:

- $a \rightarrow$ for $a \in N$ is treated as a visible transition, while
- $d \rightarrow$ for $d \in \mathbb{R}^{\geq 0}$ are all labelled by a single visible action $\epsilon \rightarrow$.

Corollary

Any two timed bisimilar automata are also untimed bisimilar.
Timed Non-Bisimilar but Untimed Bisimilar Automata

\[
\begin{align*}
A & \xrightarrow{x \leq 1} B \\
& \xrightarrow{a} \downarrow \quad \downarrow \\
& \quad \downarrow \quad \downarrow \\
B & \xrightarrow{x \leq 3} C \\
& \xrightarrow{a} \downarrow \\
C & \\
\end{align*}
\]

\[
\begin{align*}
A' & \xrightarrow{x \leq 2} B' \\
& \xrightarrow{a} \downarrow \quad \downarrow \\
& \quad \downarrow \quad \downarrow \\
B' & \xrightarrow{x \leq 3} C' \\
& \xrightarrow{a} \downarrow \\
C' & \\
\end{align*}
\]
Decidability of Timed and Untimed Bisimilarity

Theorem [Cerans'92]
Timed bisimilarity for timed automata is decidable in EXPTIME (deterministic exponential time).
Decidability of Timed and Untimed Bisimilarity

**Theorem [Cerans’92]**
Timed bisimilarity for timed automata is decidable in EXPTIME (deterministic exponential time).

**Theorem [Larsen, Wang’93]**
Untimed bisimilarity for timed automata is decidable in EXPTIME (deterministic exponential time).
Timed Traces

Let $A = (L, \ell_0, E, I)$ be a timed automaton over a set of clocks $C$ and a set of labels $N$.

Timed Traces

A sequence $(t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$ where $t_i \in \mathbb{R}^{\geq 0}$ and $a_i \in N$ is called a timed trace of $A$ iff there is a transition sequence

$$(\ell_0, v_0) \xrightarrow{d_1} . \xrightarrow{a_1} . \xrightarrow{d_2} . \xrightarrow{a_2} . \xrightarrow{d_3} . \xrightarrow{a_3} \ldots$$

in $A$ such that $v_0(x) = 0$ for all $x \in C$ and

$$t_i = t_{i-1} + d_i \quad \text{where} \quad t_0 = 0.$$

Intuition: $t_i$ is the absolute time (time-stamp) when $a_i$ happened since the start of the automaton $A$. 
Timed and Untimed Language Equivalence

The set of all timed traces of an automaton $A$ is denoted by $L(A)$ and called the timed language of $A$.

Theorem [Alur, Courcoubetis, Dill, Henzinger’94]
Timed language equivalence (the problem whether $L(A_1) = L(A_2)$ for given timed automata $A_1$ and $A_2$) is undecidable.
Timed and Untimed Language Equivalence

The set of all timed traces of an automaton $A$ is denoted by $L(A)$ and called the timed language of $A$.

**Theorem [Alur, Courcoubetis, Dill, Henzinger’94]**

Timed language equivalence (the problem whether $L(A_1) = L(A_2)$ for given timed automata $A_1$ and $A_2$) is undecidable.

We say that $a_1a_2a_3\ldots$ is an untimed trace of $A$ iff there exist $t_1, t_2, t_3,\ldots \in \mathbb{R}_{\geq 0}$ such that $(t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$ is a timed trace of $A$.

**Theorem [Alur, Dill’94]**

Untimed language equivalence for timed automata is decidable.