Why continuous-time semantics?

Example [Alur91]

- under discrete-time, the output is always 0:
Why continuous-time semantics?

Example [Alur91]

- under discrete-time, the output is always 0:
Why continuous-time semantics?

Example [Alur91]

• under discrete-time, the output is always 0:
Why continuous-time semantics?

Example [Alur91]

- under discrete-time, the output is always 0:
Why continuous-time semantics?

Example [Alur91]

- under discrete-time, the output is always 0:
Why continuous-time semantics?

Example [Alur91]

- Under continuous-time, the output can be 1:
Why continuous-time semantics?

Example [Alur91]

Finding the correct granularity (if one exists) is hard!
The model of timed automata

This run reads the timed word

\[(\text{problem}, x:=0) \rightarrow \text{alarm} \rightarrow \text{repairing} \rightarrow \text{done, } 22 \leq y \leq 25\]

\[(\text{delayed}, y:=0) \rightarrow \text{failsafe} \rightarrow \text{repair} \rightarrow \text{done} \rightarrow \text{safe}\]
The model of timed automata

\[
x := 0, x \leq 15, y := 0, 15 \leq x \leq 16, y := 0, 2 \leq y \land x \leq 56, y := 0, 22 \leq y \leq 25.
\]

\[
\text{done, } 22 \leq y \leq 25.
\]

\[
\text{problem, } x := 0, \text{repair, } x \leq 15, y := 0, \text{delayed, } y := 0.
\]

\[
\text{repairing, } y := 0, \text{repair, } 2 \leq y \land x \leq 56, y := 0.
\]

\[
\text{failsafe, } y := 0.
\]
The model of timed automata

This run reads the timed word

(\text{problem}, 23)
(\text{delayed}, 38.6)
(\text{repair}, 40.9)
(\text{done}, 63)

\[
\begin{array}{c|c|c}
\text{safe} & 23 & \text{safe} \\
\text{x} & 0 & 23 \\
\text{y} & 0 & 23 \\
\end{array}
\]
The model of timed automata

This run reads the timed word

\[(\text{problem}, 23) (\text{delayed}, 38.6) (\text{repair}, 40.9) (\text{done}, 63)\]
The model of timed automata

This run reads the timed word

\[(\text{problem}, 23) (\text{delayed}, .6) (\text{repair}, 40) (\text{done}, 63)\]
The model of timed automata

This run reads the timed word

\( (\text{problem}, 23) (\text{delayed}, 38.6) (\text{repair}, 40.9) (\text{done}, 63) \)

<table>
<thead>
<tr>
<th>safe</th>
<th>23</th>
<th>safe</th>
<th>problem</th>
<th>alarm</th>
<th>15.6</th>
<th>alarm</th>
<th>delayed</th>
<th>failsafe</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>15.6</td>
<td>15.6</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
| y    | 0  | 23   | 23      | 38.6  | 0    | 0     | 0       | ...

failsafe

... 15.6

0
The model of timed automata

![Diagram of timed automata with states safe, alarm, and failsafe, transitions include repair, repair, and delayed, with conditions on x and y.]

This run reads the timed word

\[(\text{problem}, 23) \rightarrow (\text{delayed}, 0) \rightarrow (\text{repair}, 0) \rightarrow (\text{done}, 22 \leq y \leq 25)\]
The model of timed automata

This run reads the timed word 

\[
\begin{align*}
\text{problem, } x &= 0 \\
\text{alarm, } y &= 0 \\
\text{repair, } x &
\leq 15 \\
\text{delayed, } y &= 0 \\
\text{done, } 22 \leq y \leq 25 \\
\text{repair, } 2 \leq y \wedge x \leq 56 \\
\end{align*}
\]
The model of timed automata

This run reads the timed word

\((\text{problem}, 23)(\text{delayed}, 23.6)(\text{repair}, 40.9)(\text{done}, 63)\)

The model of timed automata

This run reads the timed word

\[
\begin{align*}
\text{safe} & \xrightarrow{23} \text{safe} & \text{problem} & \xrightarrow{15.6} \text{alarm} & \xrightarrow{delayed} \text{failsafe} \\
\text{safe} & \xrightarrow{2.3} \text{failsafe} & \text{repair} & \xrightarrow{22.1} \text{repairing} & \xrightarrow{done} \text{safe} \\
\end{align*}
\]

\[
\begin{align*}
\text{x} & \quad 0 & \quad 23 & \quad 0 & \quad 15.6 & \quad 15.6 & \quad \ldots \\
\text{y} & \quad 0 & \quad 23 & \quad 23 & \quad 38.6 & \quad 0 & \quad \ldots \\
\end{align*}
\]
The model of timed automata

This run reads the timed word:

\[(\text{problem}, 0) \rightarrow (\text{delayed}, 23.6) \rightarrow (\text{repair}, 0) \rightarrow (\text{done}, 63)\]

### (Clock) Valuation

<table>
<thead>
<tr>
<th>State</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>23</td>
<td>0</td>
<td>15.6</td>
</tr>
<tr>
<td>alarm</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>38.6</td>
<td>15.6</td>
<td>y</td>
</tr>
<tr>
<td>delayed</td>
<td>0</td>
<td>0</td>
<td>15.6</td>
<td>38.6</td>
<td>15.6</td>
<td>y</td>
</tr>
</tbody>
</table>

...
### The model of timed automata

This run reads the timed word:

\[(\text{problem}, 23)(\text{delayed}, 38.6)(\text{repair}, 40.9)(\text{done}, 63)\]
The train crossing example
Modelling the train crossing example

Train\(_i\) with \(i = 1, 2, \ldots\)

Far

Before, \(x_i < 30\)

20 < \(x_i < 30\), \(a, x_i := 0\)

App!, \(x_i := 0\)

On, \(x_i < 20\)

10 < \(x_i < 20\), Exit!
The train crossing example – cont’d

The gate:

- Open
- Lowering, $H_g < 10$
- Raising, $H_g < 10$
- Close

Transitions:
- $H_g < 10$, a
- GoDown?, $H_g := 0$
- GoUp?, $H_g := 0$
- $H_g < 10$, a
The train crossing example – cont’d

The controller:

\[ c_0, H_c \leq 20 \quad \text{Exit?} \]

\[ H_c = 20, \text{GoUp!} \quad \text{Exit?} \]

\[ c_1, H_c \leq 20 \]

\[ \text{App?}, H_c := 0 \]

\[ H_c \leq 10, \text{GoDown!} \]

\[ c_2, H_c \leq 10 \quad \text{Exit?} \]

\[ \text{App?} \quad \text{Exit?} \]
The train crossing example – cont’d

We use the synchronization function $f$:

<table>
<thead>
<tr>
<th>Train₁</th>
<th>Train₂</th>
<th>Gate</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>App!</td>
<td>.</td>
<td>.</td>
<td>App?</td>
</tr>
<tr>
<td>.</td>
<td>App!</td>
<td>.</td>
<td>App?</td>
</tr>
<tr>
<td>Exit!</td>
<td>.</td>
<td>.</td>
<td>Exit?</td>
</tr>
<tr>
<td>.</td>
<td>Exit!</td>
<td>.</td>
<td>Exit?</td>
</tr>
<tr>
<td>a</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>a</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>a</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>GoUp?</td>
<td>GoUp!</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>GoDown?</td>
<td>GoDown!</td>
</tr>
</tbody>
</table>

NB: the parallel composition does not add expressive power!
The train crossing example – cont’d

Some properties one could check:

- Is the gate closed when a train crosses the road?
Some properties one could check:

- Is the gate closed when a train crosses the road?
- Is the gate always closed for less than 5 minutes?
Examples of regions

Let us consider the partition of $\mathbb{R}^{\{x, y\}}$ made of five pieces

\[ \mathcal{R} = \{ R_0, R_1, R_2, R_3, R_4 \} \]

- $R_0$: $(x \geq 0)$ and $(y = 0)$
- $R_1$: $(0 \leq x < 1)$ and $(0 \leq y \leq 1)$ and $(x < y)$
- $R_2$: $(x \geq 0)$ and $(0 < y \leq 1)$ and $(x \geq y)$
- $R_3$: $(x > 1)$ and $(y > 1)$ and $(x \geq y)$
- $R_4$: $(x \geq 0)$ and $(y > 1)$ and $(x < y)$

It is easy to verify that $\mathcal{R}$ is a set of regions for the set of clocks $\{x, y\}$ and for the set of constraints $\{y \leq 1, y > 1, x \geq y, x < y\}$. 
Example of region graph

\[ R_0 \]

- \[ x \geq 0 \]
- \[ y = 0 \]

\[ R_1 \]

- \[ 0 \leq x < 1 \]
- \[ 0 \leq y \leq 1 \]
- \[ x < y \]

\[ R_2 \]

- \[ x \geq 0 \]
- \[ 0 < y \leq 1 \]
- \[ x \geq y \]

\[ R_3 \]

- \[ x > 1 \]
- \[ y > 1 \]
- \[ x \geq y \]

\[ R_4 \]

- \[ x \geq 0 \]
- \[ y > 1 \]
- \[ x < y \]

→ time elapsing
→ reset of clock \( x \)
→ reset of clock \( y \)
Example of region automaton

\[ y > 1, a, x := 0 \]

\[ x < y \land y \leq 1, e \]

\[ c, y := 0 \]
\[
\begin{align*}
\text{State } s_0: & \quad x > 0, a \quad y := 0 \\
\text{State } s_1: & \quad y = 1, b \quad x < 1, c \quad x < 1, c \\
\text{State } s_2: & \quad y = 1, b \quad x < 1, c \quad x < 1, c \\
\text{State } s_3: & \quad x > 1, d \quad y < 1, a \quad y := 0
\end{align*}
\]