An brief introduction to timed automata

part C: analysis

Mariëlle Stoelinga
Formal Methods & Tools
Timed Automata: small example

Controller of Railroad Crossing

- Gate is normally open
- If train approaches, gate closes in 10-15 seconds
- If train leaves, gate opens in 11-16 seconds
What can we do with TA?

Verify properties on TA

1. *Reachability*: can we reach certain locations and states?
2. *Invariance*: does a property hold for all states?
3. *Others*: next lecture (timed version of CTL)
What can we do with TA?

Verify properties on TA

- Reachability: $E<>P$
  - can we reach certain locations and states?

```
<table>
<thead>
<tr>
<th>Approach?</th>
<th>x:=0</th>
<th>x:=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>raising x ≤ 16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>closed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lowering x ≤ 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>open, x=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lowering, x=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>closed, x=10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

$E<> Gate.raising$

- witness trace
What can we do with TA?

Verify properties on TA

- **Reachability: \( E<>P \)**
can we reach certain locations and states?

\[
\begin{align*}
& \text{open} \\
\text{Approach?} & \quad x:=0 \\
\text{raising} & \quad x \leq 16 \\
& \text{closed} \\
& \quad x \geq 11 \\
& \quad x \geq 10 \\
& \text{Leave?} & \quad x:=0
\end{align*}
\]

- \( E<> \text{Gate.raising} \)

- \( E<> \text{Gate.closed} \& \ x > 100 \)

- \( E<> \text{Gate.raising} \& \ x > 100 \)

If a reachability property is satisfied, Uppaal gives you a witnessing trace
What can we do with TA?

Verify properties on TA

- **Invariance**: \( A[]P \)
  does a property hold for all states?

---

If a safety property is violated, Uppaal gives you a counterexample trace.

Next lectures: debugging & automatic synthesis.
What can we do with TA?

Verify properties on TA

- **Invariance**: $A[P]$
  does a property hold for all states?

![Diagram showing a train system with states and transitions]

- $A[]$ Train.at_crossing $\rightarrow$ Gate.closed
- $A[]$ Train.at_crossing $\rightarrow$ Gate.closed

- Controller:
  - Exit?
  - Detected?
  - Approach!
  - Leave!
In real life

- Multiple trains
  - Hence race conditions
- Detection takes time and may fail
  - Hence multiple detection points
- More complex but still feasible
Summary

- Timing behavior is crucial in systems correctness
- **Model checking**
  - does a system model meet a correctness property?
  - as good as models you provide
- **Timed automata**
  - model checking real-time behavior
  - finite state automata with real-valued clocks
Consider two of these traffic light models, $T$ and $U$

- $T$ is initially green, $U$ initially red

**Formulate correctness properties** formulating that

- $T$ and $U$ can be red at the same time
- $T$ and $U$ can never be green at the same time
- Think one (or more) other useful properties of the traffic lights
Thank you for your attention
& See you next time!