Model checking
timed automata

part 1: semantics

Mariëlle Stoelinga
Formal Methods & Tools
Previously

Real-time is crucial

\[ E<> \text{Gate.closed} \land x > 100 \]

A[] Train.at_crossing implies Gate.closed

\[ y := 0 \quad \text{near} \]

\[ y > 10 \quad \text{at_crossing} \]

\[ y \geq 11 \]

\[ x \geq 10 \]

\[ x = 0 \quad \text{Leave?} \]

\[ z < 2 \]

\[ z := 0 \quad \text{Approach!} \]

\[ z \geq 10 \quad \text{Leave?} \]

\[ z \leq 16 \]

\[ x \leq 10 \]

\[ x \leq 16 \]

\[ x := 0 \quad \text{Approach!} \]

\[ E<> \text{Gate.closed} \land x > 100 \]

\[ \text{A[] Train.at_crossing implies Gate.closed} \]

\[ \text{model checker} \]

\[ \text{satisfied} \]

\[ \text{violated + counter example} \]

\[ \text{system} \]

\[ \text{meets??} \]

\[ \text{Safety Requirements} \]

\[ \text{model} \]

\[ \text{invariant: at_crossing } \rightarrow \text{gates closed} \]
Agenda

Safety Requirements

Invariant:
\[ \text{at\_crossing} \Rightarrow \text{gates closed} \]

Algorithms?

Data structures?

System model meets??

violated + counter example

satisfied
## New schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>at home (exercises / Video)</th>
<th>theory in class</th>
<th>exercises in class</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA #1</td>
<td>Mon 25 Nov</td>
<td>intro TA</td>
<td>intro TA</td>
</tr>
<tr>
<td>TA #2</td>
<td>Fri 29 Nov</td>
<td>Uppaal under the hood</td>
<td>WC: oefenen met RA</td>
</tr>
<tr>
<td>TA #3</td>
<td>Mon 2 Dec</td>
<td>Uppaal under the hood</td>
<td>Uppaal under the hood (algos)</td>
</tr>
<tr>
<td>TA #4</td>
<td>Fri 6 Dec</td>
<td>paper datapaths</td>
<td>- paper datapaths</td>
</tr>
<tr>
<td></td>
<td>Mon 9 Dec</td>
<td></td>
<td>- assignment</td>
</tr>
</tbody>
</table>

**In class exercises**
- Small: no discussion
- Big: discussion
- Questions always welcome!
Model checking TAs: how?

Is R reachable?

Yes:
- P, x\leftarrow0, y\leftarrow0
- P, x\leftarrow3, y\leftarrow3
- P, x\leftarrow0, y\leftarrow3
- P, x\leftarrow3, y\leftarrow3
- P, x\leftarrow4, y\leftarrow7
- Q, x\leftarrow0, y\leftarrow7
- R, x\leftarrow0, y\leftarrow7

1. What does reachability mean?
2. How to compute in TA if given location is reachable?

Idea?

- R reachable = some run (path) leads to R
- Runs work on states = locations + clock values
- Transitions can be delays or actions
- Transitions must respect invariants (delays) and guards (actions)
- Mathematically: TLTSs

Exercise (pairs)
- Write down some other runs of H
  - Some reach R
  - Some do not reach R
Model checking TAs: runs

Is R reachable?

Yes: \( P, x\leftarrow 0, y\leftarrow 0 \)

\( 3 \)

\( P, x\leftarrow 3, y\leftarrow 3 \)

\( b \)

\( P, x\leftarrow 0, y\leftarrow 3 \)

\( 4 \)

\( P, x\leftarrow 4, y\leftarrow 7 \)

\( c \)

\( Q, x\leftarrow 0, y\leftarrow 7 \)

\( d \)

\( R, x\leftarrow 0, y\leftarrow 7 \)

- R reachable = some trace leads to R
- Traces work on states = locations + clock values
- Transitions can be delays or actions
- Transitions must respect invariants (delays) and guards (actions)
- Mathematically: TLTSs

TLTSs
- Infinitely many states / transitions: not suitable for model checking
- Pin down meaning
  - No ambiguity
  - Equivalence of TAs
  - \( = \) same semantics
Timed Automata: equivalence

Equivalence
- Exactly same semantics
Model checking TAs: overview

1. What does reachability mean?
2. How to compute in TA if given location is reachable?

finite LTS
- timing info in states
- basic model checking
  - complexity
  - decidability

finite LTS
- timing info in states
- efficient model checking
- data structure: DBMs

infinite LTS
- states = lo
- wha
- reuse existing theory:
  - paths, traces, reachability
  - bisimulation, (timed) trace equivalence
  - ...

TLTS

today

Zone-construction
Upbaal

region construction

group states

Region Automaton

group more states

Zone Automaton

TA

Semantics

Zone Automaton

Region Automaton

TLTS

Region-construction
1a: What is a Timed Labeled Transition System? definition

A **TLTS** is a 4-tuple \((S, s_0, A, \rightarrow)\) given by

- \(S\) is the set of **states**
- \(s_0 \in S\) is the **initial state**
- **A** is the **action alphabet**
- \(\rightarrow \subseteq S \times (A \cup R_{\geq 0}) \times S\) is the **transition relation** s.t.

  for all \(s, s_1, s_3 \in S\) and \(d_1, d_2 \in R_{\geq 0}\)
  - \(s^0 \rightarrow s\)
  - \(\exists s_2 \in S. s_1 \rightarrow s_2\) and \(s_2 \rightarrow s_3\) if and only if \(s_1 \rightarrow s_2\)

\[\begin{align*}
\text{P, } x&<0, y<0 \\
\text{R, } x&<0, y<7 \\
\text{Q, } x&<0, y<7 \\
\text{P, } x&<3, y<3 \\
\text{P, } x&<0, y<3 \\
\text{P, } x&<4, y<7 \\
\end{align*}\]
A **TLTS** is a 4-tuple \((S, s_0, A, \rightarrow)\) given by

- \(S\) is the set of *states*
- \(s_0 \in S\) is the *initial state*
- \(A\) is the *action alphabet*
- \(\rightarrow \subseteq S \times (A \cup R_{\geq 0}) \times S\) *transition relation*
  
  for all \(s, s_1, s_3 \in S\) and \(d_1, d_2 \in R_{\geq 0}\)

- \(s \rightarrow s\)
- \(\exists s_2 \in S. s_1 \rightarrow s_2\) and \(s_2 \rightarrow s_3 \iff s_1 \rightarrow s_2\)

**Go through the formal definition**

- All 4 ingredients
- Some notation needed

**Model checking TAs: runs**

**Immediate transition in TLTS** *(must respect guard of transition taken in TA)*

**Delay transition in TLTS** *(must respect invariant in P)*

**State in TLTS**

- \(P, x \leftarrow 0, y \leftarrow 0\)
- \(P, x \leftarrow 3, y \leftarrow 3\)
- \(P, x \leftarrow 4, y \leftarrow 7\)
- \(Q, x \leftarrow 0, y \leftarrow 7\)
- \(R, x \leftarrow 0, y \leftarrow 7\)
1b: How to obtain underlying TLTS from TA?

**Examples**
- (P, <x⇐0, y⇐7>)
- (P, <x⇐1.2, y⇐3.4>)
- (P, <x⇐√2, y⇐log 37>)
- (Q, <x⇐3, y⇐3>)
- Invariant violated
- Still a state
- Not all are reachable
- E.g. if invariant violated
- Infinitely many (reachable) states

**Initial state in TLTS**
- Initial location
- All clocks equal 0

**Immediate actions in TLTS**
- All actions in TA

**States in TLTS**
- Tuple with
  - Location
  - Clock valuation in v: C → R≥0

**Initial state in H**
- (P, <x⇐0, y⇐0>) i.e. (P,0)

**Immediate actions in H?** {a,b,c,d}
A **TLTS** is a 4-tuple \((S, s_0, A, \rightarrow)\) given by
- \(S\) is the set of **states**
- \(s_0 \in S\) is the **initial state**
- \(A\) is the **action alphabet**
- \(\subseteq S \times (A \cup R_{\geq 0}) \times S\) transition relation
  for all \(s, s_1, s_3 \in S\) and \(d_1, d_2 \in R_{\geq 0}\)
  - \(s \rightarrow s\)
  - \(\exists s_2 \in S. s_1 \rightarrow s_2\) and \(s_2 \rightarrow s_3 \iff s_1 \rightarrow s_2\)

Go through the formal definition
- All 4 ingredients
- Some notation needed

**Immediate transition** in TLTS (must respect guard of transition taken in TA)

**Delay transition** in TLTS (must respect invariant in P)
1b: How to obtain underlying TLTS from TA?

**Delay transitions**
- Time can progress as long as invariant holds
- In location $l$, you can delay for $d$ iff increasing all clock by $d$ still satisfies the invariant of $l$.
- All delay transitions of TA given by
  - $\{ (<l,u>, d, <l,u+d>) \mid l \in L, u: C \rightarrow R_{\geq 0}, u+d \models \text{inv}(l) \}$

**Notation**
- $u: C \rightarrow R_{\geq 0}$ a clock valuation

**Example**
- $<x \leftarrow 4, y \leftarrow 7>$ means $u: \{x,y\} \rightarrow R_{\geq 0}$ with $u(x) = 4, u(y) = 7$

**Exercise (pairs)**
- Think of some examples with / without $u \models x - z < 6$ & $z > 0$

**Satisfaction:** Given
- $u: C \rightarrow R_{\geq 0}$ a clock valuation,
- $g \in B(C)$ a clock guard
- $u$ satisfies $g$
- $u \models g$ iff $g$ holds for values from $u$
1b: How to obtain underlying TLTS from TA?

Delay transitions
- Time can progress as long as invariant holds
- In location $l$, you can delay for $d$ iff increasing all clock by $d$ still satisfies the invariant of $l$.
- All delay transitions of TA given by
  - $\{ (\langle l,u \rangle, d, \langle l, u+d \rangle) \mid l \in L, \ u: C \rightarrow R_{\geq 0}, u+d \models \text{inv}(l) \}$

More notation
- $R \subseteq C$ subset of clocks, $d \ R_{\geq 0}$
- $u[R:=0]$ valuation with all clocks in $R$ equal to 0 and other clocks as in $u$.
- $u + d$ valuation with all clocks increased by $d$.

Exercise (pairs)
- Think of some examples with / without $u \models x - z < 6 \ & \ z > 0$

Satisfaction: Given
- $u: C \rightarrow R_{\geq 0}$ a clock valuation,
- $g \in B(C)$ a clock guard
- $u$ satisfies $g$
- $u \models g$ iff $g$ holds for values from $u$
Timed Automata: executing transitions

1. Start in location $L_1$
   - Invariant $\text{Inv}_1$ holds
2. Check if guard $G$ holds
   - otherwise transition cannot be taken
3. Take the action $a$
   - multiple outgoing $a$ actions: nondeterminism
4. Reset all clocks in reset set
   - other clock retain their values
5. Check if invariant in target state holds
   - otherwise transition cannot be taken
6. Move to the target location $x:=0, y:=0$
1b: How to obtain underlying TLTS from TA?

What are the transitions in $\mathcal{H}$?

- $(P, \langle x=4, y=7 \rangle)$ \( b \rightarrow (P, \langle x=6.2, y=9.2 \rangle) \)
- $(Q, \langle x=0, y=\log 7 \rangle)$ \( d \rightarrow (R, \langle x=3, y=10 \rangle) \)
- $(P, \langle x=4, y=7 \rangle)$ \( c \rightarrow (Q, \langle x=0, y=9.2 \rangle) \)

Action transitions in $(L, l_0, \Sigma, E, \text{Inv})$

\[
\{ (\langle l,u \rangle, a, \langle l',u[r:=0]\rangle) \mid (l,g,a,r,l') \in E, \ u \models g, \ u[r:=0] \models \text{inv}(l') \} 
\]
tuple \((L, l_0, \Sigma, E, \text{Inv})\) ➔ tuple \((S, s_0, A, \rightarrow)\)
2b: How to obtain underlying TLTS from TA?

Let \( H = (L, l_0, \Sigma, C, E, \text{inv}) \) be a TA. The underlying TLTs of \( H \) is given by \((S, s_0, A, \rightarrow)\) where

- \( S = L \times (C \rightarrow R_{\geq 0}) \)
- \( s_0 = (l_0, 0) \) is the initial state
- \( A = \Sigma \)
- \( \rightarrow = \)

\[
\begin{align*}
\{ (\langle l, u \rangle, d, \langle l, u+d \rangle) & \mid l \in L, u \in B(C), u+d \models \text{inv}(l) \} \\
U \\
\{ (\langle l, u \rangle, a, \langle l', u[r:=0] \rangle) & \mid (l, g, a, r, l') \in E, u \models 9, u[r:=0] \models \text{inv}(l') \}
\end{align*}
\]

### TLTSs

- Infinitely many states / transitions: not suitable for model checking
- Pin down meaning
  - No ambiguity
  - Equivalence of TAs
  - = same semantics
1. What does reachability mean?
2. How to compute in TA if given location is reachable?

Model checking TAs: overview

finite LTS
- timing info in states
- basic model checking
  - complexity
  - decidability

infinite LTS
- states = lo
- wha
- reuse existing theory:
  - paths, traces, reachability
  - bisimulation, (timed) trace equivalence
- ...

infinite LTS
- states = lo
- wha
- reuse existing theory:
  - paths, traces, reachability
  - bisimulation, (timed) trace equivalence
- ...

finite LTS
- timing info in states
- efficient model checking
- data structure: DBMs

TLTS
- Region-construction
- Uppaal

Today
- TA
- Semantics

region construction
- group states

finite LTS
- timing info in states
- basic model checking
- complexity
- decidability

Region Automaton

finite LTS
- timing info in states
- efficient model checking
- data structure: DBMs

Zone Automaton

Zone-construction

Zone Automaton

Zone Automaton

19
Thank you for your attention & See you next time!