Model checking timed automata
Lecture 2: how

Mariëlle Stoelinga
Formal Methods & Tools
Agenda

Safety Requirements

System

Model

meets??

Invariant:

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

Algorithms? + Data structures?

violated + counter example

satisfied

UNIVERSITY OF TWENTE.
Agenda

1. Exercises
2. Parallel composition
3. Next exercises
4. Next screen cast
Agenda

1. Exercises
2. Parallel composition
3. Next exercises
4. Next screen cast
The underlying TLTS

\[
\begin{align*}
I_0 & : x > 2 \quad a \ y := 0 \\
I_1 & : y < 1 \quad 2 \leq x - y \leq 3 \\
I_2 & : y := 0
\end{align*}
\]

path

\[
\begin{align*}
l_0, x \leftarrow 0, y \leftarrow 0 \\
l_0, x \leftarrow 2.1, y \leftarrow 0 \quad 2.1 \\
l_1, x \leftarrow 2.1, y \leftarrow 0 \quad a \\
l_2, x \leftarrow 2.1, y \leftarrow 0
\end{align*}
\]

faster path

\[
\begin{align*}
l_0, x \leftarrow 0, y \leftarrow 0 \\
l_0, x \leftarrow 2.01, y \leftarrow 0 \quad 2.01 \\
l_1, x \leftarrow 2.01, y \leftarrow 0 \quad a \\
l_2, x \leftarrow 2.01, y \leftarrow 0
\end{align*}
\]

UNIVERSITY OF TWENTE.
The underlying TLTS

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
- $A = \Sigma$
- $\rightarrow =$

$\{ (<l,u>,d,<l,u+d>) \mid l \in L, u \in B(C), u+d \models \text{inv}(l) \}$

$\{ (<l,u>,a,<l',u[r:=0]> ) \mid (l,g,a,r,l') \in E, u \models g, u[r:=0] \models \text{inv}(l') \}$

- $S = \{l_0, l_1, l_2\} \times \{x,y \rightarrow R_{\geq 0}\}$
- $s_0 = <l_0,(x\leftarrow 0,y\leftarrow 0)>$
- $A = \{a\}$
  - Assumption
- $\rightarrow =$

Delay transitions

- In $l_0$ we can delay as long as we want
  - $\{(l_0, u>,d,<l_0, u>) \mid d \in R_{\geq 0}, u \in \{x,y \rightarrow R_{\geq 0}\} \}$
- In $l_1$ we can delay as long as $y$'s value $< 1$ [try yourself]
  - $\{(l_1, u>,d,<l_1, u>) \mid d \in R_{\geq 0}, u \in \{x,y \rightarrow R_{\geq 0}, u(y) < 1\} \}$
- In $l_2$ we can delay as long as we want
  - $\{(l_2, u>,d,<l_2, u>) \mid d \in R_{\geq 0}, u \in \{x,y \rightarrow R_{\geq 0}\} \}$
Let $H = (L, I_0, \Sigma, C, E, \text{inv})$ be a TA. The underlying TLTS of $H$ is given by $(S, s_0, A, \rightarrow)$ where

- $S = L \times (C \rightarrow R_{\geq 0})$
- $s_0 = (I_0, 0)$ is the initial state
- $A = \Sigma$
- $\rightarrow =$

- Action transitions
- In $I_0$ we can move to $I_1$
  $$\{(l_0, u, a, l_1, u[y:=0]) \mid u \in \{x, y\} \rightarrow R_{\geq 0}, u(x) > 2\} \cup$$
  $$\{(l_0, u, a, l_1, v) \mid u, v \in \{x, y\} \rightarrow R_{\geq 0}, u(x) > 2, u(x)=v(x), v(y)=0\} \cup$$
- In $I_1$ we can move to $I_1$
  $$\{(l_1, u), \tau, l_1, u[y:=0]\} \mid u \in \{x, y\} \rightarrow R_{\geq 0}\}$$
- In $I_2$ we can [try again]
  $$\{(l_1, u), \tau, l_2, u\} \mid u \in \{x, y\} \rightarrow R_{\geq 0}\} 2 \leq u(x) - u(y) \leq 3}$$

- $S = \{I_0, I_1, I_2\} \times \{\{x, y\} \rightarrow R_{\geq 0}\}$
- $s_0 = (I_0, 0)$
- $A = \{a\}$
  - Assumption
- $\rightarrow =$

The underlying TLTS
Example 2: Fisher’s Mutual Exclusion Protocol

Mutex algorithms are essential
- webshops: do not sell same ticket twice
- operating systems: access to bus, disk, …

Mutex algos are tricky:
- errors are easily made

One solution:
- Fisher’s algorithm:

Fisher’s Protocol
- N different processes
  - with pids 1,2,… N
- ≤ 1 in critical section (CS)
- ensures mutex via timing and shared variable id
  - id = pid: process pid is at CS or tries to enter
  - id = 0: no process (trying to get) in CS
Agenda

1. Exercises
2. Parallel composition
3. Next exercises
4. Next screen cast
Example 2: Fisher’s Mutual Exclusion Protocol

Mutex algorithms are essential
- webshops: do not sell same ticket twice
- operating systems: access to bus, disk, …

Mutex algos are tricky:
- errors are easily made

One solution:
- Fisher’s algorithm:

Fisher’s Protocol
- N different processes
  - with pids 1, 2, .. N
- ≤ 1 in critical section (CS)
- ensures mutex via timing and shared variable id
  - id = pid: process pid is at CS or tries to enter
  - id = 0: no process (trying to get) in CS

Behavior
- If id=0, request to enter CS
- Stay between [0, 2] seconds
- In wait: no changes to id
- for 2 seconds,
  - enter CS
  - otherwise try again
- Reset id:=0 upon leaving CS
Network of timed automata: Product Construction

Network of TAs
- several TAs run in parallel
- synchronization:
  - on shared actions: b
    - compulsory: if no other b-actions
    - optional: if other b-actions
  - independent on other actions: a,c
- powerful modeling construct
  - model each component individually

But: state space grows exponentially with nr of components
Plan for today

network of TA

Product construction

1. Semantics

TA

Product construction

2b. Region-construction

TLTS

finite state LTS

4. Zone-construction

Uppaal

2a. group states

RA

finite state LTS

3. model checking

yes/no

4. group more states

ZA

finite state LTS

5. model checking

yes/no

UNIVERSITY OF TWENTE.
Plan for today

network of TA

Product construction

network of TA

1. Semantics

2a. group states

finite state LTS

2b. Region-construction

TLTS

3. model checking

finite state LTS

4. Zone-construction

Uppaal

4. group more states

RA

yes/no

5. model checking

ZA

yes/no

UNIVERSITY OF TWENTE.
Agenda

1. Exercises
2. Parallel composition
3. Next exercises
4. Next screen cast
Standard BFS algorithm

- **Input**: Automaton $A = (S, s_0, A, \rightarrow)$, target $T$
- **Output**: is $T$ reachable from $s_0$

- **Method**:
  - `passed := {}` % seen all successors
  - `wait := \{s_0\}` % to be processed
  - while `wait ≠ {}` do
    - move s from `wait` to `passed`
    - if `s in T` then return YES
    - else
      - (compute successors)
      - If `s'` is a successor of `s` and `s` notin `passed` then add
  - endwhile
Regions

Idea: group together equivalent states
- equivalent states satisfy same clock constraints
- KEY INSIGHT:
  if one state enables transition,
  then all equivalent enable the transition and move to equivalent states
→ Group equivalent states into one
→ There are finitely many groups / classes
= Finite Region Automaton

When are states equivalent?
1. single clock case
2. multiple clock case

Infinite TLTS

Finite RA

1. Exercises
2. Parallel composition
3. Next exercises
4. Next screen cast
Model checking timed automata?
Lecture 2: how

Mariëlle Stoelinga
Formal Methods & Tools