Software Verification
Exercise class 7: Real Time Systems

Carlo A. Furia
Recap of definitions and results
**Timed Automata: Syntax**

**Def. Nondeterministic Timed Automaton (TA):**

A tuple \([\Sigma, S, C, I, E, F]\):

- \(\Sigma\): finite nonempty (input) alphabet
- \(S\): finite nonempty set of locations (i.e., discrete states)
- \(C\): finite set of clocks
- \(I, F\): set of initial/final states
- \(E\): finite set of edges \([s, \sigma, c, \rho, s']\)
  
  - \(s \in S\): source location
  - \(s' \in S\): target location
  - \(\sigma \in \Sigma\): input character (also “label”)
  - \(c\): clock constraint in the form:
    \[c ::= x \approx k \mid x - y \approx k \mid - c \mid c_1 \land c_2\]
    
    - \(x, y \in C\) are clocks
    - \(k \in \mathbb{Z}\) is an integer constant
    - \(\approx\) is a comparison operator among \(<, \leq, >, \geq, =\)
  
  - \(\rho \subseteq C\): set of clock that are reset (to 0)
Timed Automata: Semantics

Def. A timed word \( w = w(1) w(2) \ldots w(n) \in (\Sigma \times \mathbb{R})^* \) is a sequence of pairs \([\sigma(i), t(i)]\) such that:

- the sequence of timestamps \( t(1), t(2), \ldots, t(n) \) is increasing
- \([\sigma(i), t(i)]\) represents the \( i \)-th character \( \sigma(i) \) read at time \( t(i) \)

Def. An accepting run of a TA \( A = [\Sigma, S, C, I, E, F] \) over input timed word \( w = [\sigma(1), t(1)] \ldots [\sigma(n), t(n)] \in (\Sigma \times \mathbb{R})^* \) is a sequence \( r = [s(0), v(0,1), \ldots, v(0,|C|)] \ldots [s(n), v(n,1), \ldots, v(n,|C|)] \in (S \times \mathbb{R}^{\mathbb{I}})^* \) of (extended) states such that:

- it starts from an initial state and ends in an accepting state: \( s(0) \in I \) and \( s(n) \in F \)
- initially all clocks are reset to 0: \( v(0,k) = 0 \) for all \( 1 \leq k \leq |C| \)
- for every transition \( 0 \leq i < n \):
  - \([ s(i), v(i,1) \ldots v(i,|C|) ] \rightarrow [ s(i+1), v(i+1,1) \ldots v(i+1,|C|) ] \)
  - some edge \([s(i), \sigma(i+1), c, \rho, s(i+1)]\) in \( E \) is followed:
    - the clock values \( v(i,1) + (t(i+1) - t(i)) \ldots v(i,|C|) + (t(i+1) - t(i)) \) satisfy the constraint \( c \)
    - \( v(i+1,k) = \text{if } k\text{-th clock is in } \rho \text{ then } 0 \text{ else } v(i,k) + t(i+1) - t(i) \)
Def. Any TA $A=([\Sigma, S, C, I, E, F]$ defines a set of input timed words $\langle A \rangle$:

$$\langle A \rangle \overset{def}{=} \{ w \in (\Sigma \times \mathbb{R})^* \mid \text{there is an accepting run of } A \text{ over } w \}$$

$\langle A \rangle$ is called the language of $A$
Metric (Linear) Temporal Logic: Syntax

Def. Propositional Metric Temporal Logic (MTL) formulae are defined by the grammar:

\[ F ::= p \mid \neg F \mid F \land G \mid F \mathbf{U}_{a,b} G \]

with \( p \in P \) any atomic proposition and \(<a,b>\) is an interval of the time domain (w.l.o.g. with integer endpoints).

Temporal (modal) operators:

- next: \( \mathbf{X} F \triangleq \text{True} \mathbf{U}_{[1,1]} F \)
- bounded until: \( F \mathbf{U}_{a,b} G \)
- bounded release: \( F \mathbf{R}_{a,b} G \triangleq \neg (\neg F \mathbf{U}_{a,b} \neg G) \)
- bounded eventually: \( \Diamond_{a,b} F \triangleq \text{True} \mathbf{U}_{a,b} F \)
- bounded always: \( \Box_{a,b} F \triangleq \neg \Diamond_{a,b} \neg F \)
- intervals can be unbounded; e.g., \([3, \infty)\)
- intervals with pseudo-arithmetic expressions, e.g.:
  - \( \geq 3 \) for \([3, \infty)\)
  - \( = 1 \) for \([1,1]\)
  - \([0, \infty)\) is simply omitted
Metric Temporal Logic: Semantics

Def. A timed word $w = [\sigma(1), t(1)] [\sigma(2), t(2)] ... [\sigma(n), t(n)] \in (2^p \times \mathbb{R})^*$ satisfies an LTL formula $F$ at position $1 \leq i \leq n$, denoted $w, i \models F$,
under the following conditions:

- $w, i \models p$ iff $p \in \sigma(i)$
- $w, i \models \neg F$ iff $w, i \not\models F$ does not hold
- $w, i \models F \land G$ iff both $w, i \models F$ and $w, i \models G$ hold
- $w, i \models F \mathcal{U} <a,b> G$ iff for some $i \leq j \leq n$ such that $t(j) - t(i) \in <a,b>$ it is:
  - $w, j \models G$ and for all $i \leq k < j$ it is $w, k \models F$
  - i.e., $F$ holds until $G$ will hold within $<a, b>$
Def. Satisfaction:

\[ w \vDash F \triangleq w, 1 \vDash F \]

i.e., timed word \( w \) satisfies formula \( F \) initially

Def. Any MTL formula \( F \) defines a set of timed words \( \langle F \rangle \):

\[ \langle F \rangle \triangleq \{ w \in (2^p \times \mathbb{R})^* \mid w \vDash F \} \]

\( \langle F \rangle \) is called the language of \( F \)
Clock Regions

Def. A clock region is an equivalence class of clock evaluations induced by the equivalence relation ~

• For a clock evaluation \( v = [v(1), \ldots, v(n)] \in \mathbb{R}_{\geq 0}^n \), \([[v]]\) denotes the clock region \( v \) belongs to.

• As a consequence of the definition of ~, any clock region can be uniquely characterized by a finite set of constraints on clocks:
  - \( v = [0.4; 0.9; 0.7; 0] \) for 4 clocks \( w, x, y, z \)
  - \([[v]]\) is \( z = 0 < w < y < x < 1 \)

• Fact: clock regions are always in finite number
Clock Regions (cont'd)

More systematically:

- given a set of clocks $C = [x(1), ..., x(n)]$
- with $M(i)$ the largest constant appearing in constraints on clock $x(i)$

a clock region is uniquely characterized by

- For each clock $x(i)$ a constraint in the form:
  - $x(i) = c$ with $c = 0, 1, ..., M(i)$; or
  - $c - 1 < x(i) < c$ with $c = 1, ..., M(i)$; or
  - $x(i) > M(i)$

- For each pair of clocks $x(i), x(j)$ a constraint in the form
  - $\text{frac}(x(i)) < \text{frac}(x(j))$
  - $\text{frac}(x(i)) = \text{frac}(x(j))$
  - $\text{frac}(x(i)) > \text{frac}(x(j))$

(These are unnecessary if $x(i) = c, x(j) = c, x(i) > M(i), \text{or } x(j) > M(j)$ )
Region Automaton Construction

For a timed automaton $A$ it is always possible to build an FSA $\text{reg}(A)$ (the “region automaton” of $A$) such that:

$$\langle A \rangle = \emptyset \quad \text{iff} \quad \langle \text{reg}(A) \rangle = \emptyset$$

**Def.** Given a TA $A = [\Sigma, S, C, I, E, F]$ its region automaton $\text{reg}(A) \triangleq [\Sigma, rS, rI, rE, rF]$ is defined as:

- $rS \triangleq \{ (s, r) \mid s \in S \text{ and } r \text{ is a clock region} \}$
- $rI \triangleq \{ (s, [0, 0, ..., 0]) \mid s \in I \}$
  - the clock region where all clocks are reset to 0
- $rE(\sigma, [s, r]) \triangleq \{ (s', r') \mid [s, \sigma, c, \rho, s'] \in E$
  and there exists a region $r'' \in \text{time-succ}(r)$
  such that $r''$ satisfies $c$, and $r'$ is obtained from $r''$ by resetting all clocks in $\rho$ to 0
- $rF \triangleq \{ (s, r) \mid s \in F \}$
Exercises:

Does the property hold?
Does the property hold?

S1

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

S2

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

□ a
Does the property hold?

Yes:
• it simply means that a holds at every position in the word (if any)
Does the property hold?

\( \square ( \Diamond = 1 \text{ a } ) \)
Does the property hold?

\[ \square ( \Diamond = 1 \ a ) \]

No:
- this requires that there is always a future position, 1 time unit in the future, where \( a \) holds
- but this is not the case in the last position of any (non-empty) timed word
Does the property hold?

\[ \square ( \square = 1 \ a ) \]
Does the property hold?

Yes:

- the formula just requires that there if there is a future position 1 time unit in the future, then a holds there
- the automaton accepts only a's every time unit, hence the property is satisfied by any word accepted by the automaton
Does the property hold?

□ ( a ⇒ ◊(0,1) c )
Does the property hold?

\[ \square (a \Rightarrow \Diamond (0,1) \ c) \]

Yes:
- clock \( x \) is reset upon reading \( a \)
- after that, it is checked upon reading \( c \)
- the constraint requires that \( x \) is in the range \((0,1)\)
Does the property hold?

\[ \square (a \Rightarrow \lozenge(0,1) \ b) \]
Does the property hold?

□ (a ⇒ ◊(0,1) b)

Yes:

• clock x is reset upon reading a; after that, it is checked upon reading c, which is always preceded by a reading of b
• if b occurs later than or exactly after 1 time unit since the reading of b, the same occurs for the reading of c
• in this case the constraint on x would be violated
Does the property hold?

\[ \square (a \Rightarrow (a \lor b) \cup (0,1) \ c) \]
Does the property hold?

\[ \square (a \Rightarrow (a \lor b) \cup (0,1) \ c) \]

Yes:
- clock \( x \) is reset upon reading \( a \)
- after that there is one reading of \( b \) followed by a reading of \( c \), which satisfies the sequence of events required by the until formula
- as far as timing is concerned, \( c \) must occur within interval of time \((0,1)\) since \( a \) occurred because of the clock constraint \( 0 < x, y < 1 \)
Does the property hold?

□ (a ⇒ (a ∨ b) U(1,2) c)
Does the property hold?

\[ \square ( a \Rightarrow (a \lor b) \cup (1,2) \ c) \]

No:
- if the “next” \( c \) is considered w.r.t when \( a \) occurs, it cannot happen in interval (1,2)
- if a successive occurrence of \( c \) is considered, it is preceded by at least another occurrence of \( c \), which is not admitted by \( a \lor b \)
Exercises:
Region automaton construction
Build the region automaton for:

\[ x := 0 \]
\[ x = 1 \]
\[ a \]

S1

\[ x := 0 \]
\[ x = 1 \]
\[ a \]

S2
Build the region automaton for:

\[ x := 0 \]
\[ x = 1 \]
\[ x := 0 \]
\[ x = 1 \]
Build the region automaton for:

\[ \text{S1} \]
\[ x := 0 \]
\[ 0 < x, y < 1 \]
\[ \text{S2} \]

\[ \text{S3} \]
\[ y := 0 \]
\[ x, y := 0 \]
Build the region automaton for:

- **S1**: \( x = y = 0 \)
- **S2**: 
  - \( x = 0 \)
  - \( 0 < y < 1 \)
  - \( y = 1 \)
  - \( y > 1 \)
- **S3**: 
  - \( y = 0 \)
  - \( 0 < x < 1 \)
  - \( x = 1 \)
  - \( x > 1 \)

Transitions:
- From **S1**: to **S2** (a), to **S3** (c)
- From **S2**: to **S1** (a), to **S2** (b)
- From **S3**: to **S2** (b), to **S3** (c)
Build the region automaton for:

Example from: Alur & Dill, 1994
Build the region automaton for:

Example from: Alur & Dill, 1994