Behavior models and verification

Lecture 9

http://d3s.mff.cuni.cz

Jan Kofroň, František Plášil
CTL, LTL model checking is fine
→ sometimes however time is important
To model behavior of real-time systems over time, in 1994, Alur et al proposed

Timed Automata
Timed Automata

Markov chains
Timed automata
Labelled transition system
Kripke structure

Model

Property specification

PCTL
TCTL
LTL
CTL

AG(start → AF heat)

Model checker

Property satisfied

Property violated

Error report

Jan Kofroň, František Plášil, Lecture 9
Timed Automata

- Markov chains
- Timed automata
- Labelled transition system
- Kripke structure

**Model**

- open
- start empty
- close
- start close
- heat
- close empty

**Property specification**

\[ \text{AG} \text{(start } \rightarrow \text{ AF heat)} \]

**Model checker**

- Property satisfied
- Error report
- Property violated
Recall: Büchi automaton and \( \omega \)-regular languages

Finite automaton accepting infinite words

A word is accepted if

- An accepting state is visited infinitely many times (standard case)
- A state from each accepting set is visited infinitely many times (generalized case)

\[ (a+b)^*a^\omega \]

Büchi automaton accepting \((a+b)^*a^\omega\)
Timed languages

Timed sequence $t = t_1 t_2 t_3 \ldots$ is an infinite sequence of time values $t_i \in \mathbb{R}, t_i > 0$ satisfying:

1. Monotonicity, i.e., $\forall i \geq 1: t_i < t_{i+1}$
2. Progress, i.e., $\forall t \in \mathbb{R}, \exists i \geq 1: t_i > t$

Timed word is a tuple $(s, t)$, where

- $s$ is an infinite sequence of symbols
- $t$ is a timed sequence (above)
In addition to Büchi, finite set of real variables representing clocks (below: $x$)

- Initially set to 0, all incrementing at the same speed
- Can be reset to 0 at any transition
- Transition only allowed if the condition upon clocks holds
- Accepts timed words
Clock constraints

For a set $X$ of clocks, the set $\Phi(X)$ of clock constraints $\delta$ is defined:

$$\delta := x \leq c \mid c \leq x \mid \neg \delta \mid \delta_1 \land \delta_2$$

where $x$ is a clock in $X$ and $c$ is a constant in $\mathbb{Q}$.
A (nondeterministic) timed automaton $A$ is a tuple $(\Sigma, S, S_0, C, E, F)$, where

- $\Sigma$ is a finite alphabet,
- $S$ is a finite set of states,
- $S_0 \subseteq S$ is set of initial states,
- $C$ is a finite set of clocks,
- $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ is transition relation, where $2^C$ specifies the clocks to be reset, and $\Phi(C)$ is clock constraint over $C$,
- $F \subseteq S$ is the set of accepting states.
Timed automaton – another example

The automaton below accepts the language $L$:

$$\left\{ ((abcd)^\omega, t) \mid \forall j. \left( (t_{4j+3} < t_{4j+1} + 1) \land (t_{4j+4} > t_{4j+2} + 2) \right) \right\}$$
Properties of TA

Question: Is the class of timed regular languages closed under:
- Finite union?
Properties of TA

**Question:** Is the class of timed regular languages closed under:
- Finite union?

**Answer:** Yes
Question: Is the class of timed regular languages closed under:

- Finite union?

Answer: Yes

Proof: Since the TA are nondeterministic, union is represented by disjoint union of particular automata. (Similar to Büchi automata)
Properties of TA

Question: Is the class of timed regular languages closed under:
- Intersection?
Properties of TA

**Question:** Is the class of timed regular languages closed under:
- Intersection?

**Answer:** Yes
Question: Is the class of timed regular languages closed under:
- Intersection?

Answer: Yes

Proof: Simple modification of intersection of Büchi automata
Recall: definition for Büchi automata

Let $A_1 = (\Sigma, S_1, S_{01}, \Delta_1, F_1)$ and $A_2 = (\Sigma, S_2, S_{02}, \Delta_2, F_2)$ be Büchi automata.

We define the product Büchi automaton to be $(\Sigma, S, S_0, \Delta, F)$, where:

- $S = S_1 \times S_2 \times \{1,2\}$
- $S_0 = S_{01} \times S_{02} \times \{1\}$
- $F = F_1 \times S_2 \times \{1\}$
- $\Delta$ as follows
Recall: \( \cap \) definition for Büchi automata

\[ \Delta : \]
- for all \( s, s' \in S_1, t, t' \in S_2, a \in \Sigma, \ i, j \in \{1,2\} : \]
\[
((s, t, i), a, (s', t', j)) \in \Delta \text{ iff } (s, a, s') \in \Delta_1, (t, a, t') \in \Delta_2, \]
and:
  - a) \( i = 1, s \in F_1, \) and \( j = 2, \) or
  - b) \( i = 2, t \in F_2, \) and \( j = 1, \) or
  - c) neither a) or b) above applies and \( j = i \)
$A_1, A_2$ are Büchi automata
Recall: Intersection for Büchi automata

\[ A = A_1 \cap A_2 \]
Intersection of Timed automata

Let $A_1, A_2$ are two timed automata with disjoint set of clocks

Denote $A = A_1 \cap A_2$

Denote $C_i$ the set of clocks

Transitions are $((s_1, s_2, i), (s'_1, s'_2, j), a, \lambda, \varphi)$

- $(s_1, s_2, i), (s'_1, s'_2, j), a$ as in the case of intersection of Büchi automata
- $\lambda = \lambda_1 \cup \lambda_2$ is the set of clock to be reset
- $\varphi = \varphi_1 \land \varphi_2$ is the transition constraint
Timed automata are NOT closed under complement

Even worse – inclusion of timed languages
$L(A) \subseteq L(B)$ is undecidable problem
Important property

- Recall LTL model checking algorithm

**Idea:** Construct Büchi $B$ automaton such that $B$ accepts the same language (up to timing) as the timed automaton under consideration
Clock regions I.

For a state $s$ of timed automaton, by $(s, n)$ denote extended state

- $s$ is a state
- $n$ is a clock interpretation (i.e., valuation of clock variables)

If $t \in \mathbb{R}$, $t = \lfloor t \rfloor + \text{frac}(t)$
Let $A = (\Sigma, S, S_0, C, E, F)$ be timed automaton

For $x \in C$, by $c_x$ denote largest $c$ such that $x \leq c$ or $c \leq x$ is a subformula of some clock constraints in $F$

The equivalence relation $\sim$ over clock interpretation – $n \sim n'$ iff all of the following holds:

1. For all $x \in C$, either $\lfloor n(x) \rfloor = \lfloor n'(x) \rfloor$ or $\lfloor n(x) \rfloor > c_x \land \lfloor n'(x) \rfloor > c_x$

2. For all $x, y \in C$ with $n(x) \leq c_x$ and $n(y) \leq c_y$: $\frac{n(x)}{n'(y)} \leq \frac{n'(x)}{n'(y)} \iff \frac{n'(x)}{n'(y)} \leq \frac{n'(y)}{n'(y)}$

3. For all $x \in C$ with $n(x) \leq c_x$: $\frac{n(x)}{n(x)} = 0 \iff \frac{n'(x)}{n'(x)} = 0$

Clock region for $A$ is equivalence class induced by $\sim$
Clock regions – example

![Graph showing clock regions](image-url)
Clock regions – example

6 corner regions: (0,0), (0,1), (1,0), …
Clock regions – example

6 corner regions: (0,0), (0,1), (1,0), …
14 open line segments: 0<x=y<1, 0<x<1 & y=0, 2<x & y=0,…
Clock regions – example

6 corner regions: (0,0), (0,1), (1,0), …
14 open line segments: 0<x=y<1, 0<x<1 & y=0, 2<x & y=0,…
8 open regions: 0<x<y<1, 2<x & 1<y, …
Clock regions III.

Each region can be characterized by specifying:

1. for each clock $x$ one clock constraint from set:
   \[
   \{x = c | c = 0, 1, \ldots, c_x\} \cup \{c - 1 < x < c | c = 1, 2, \ldots, c_x\} \cup \{x > c_x\}
   \]

2. for each pair of clock $x$ and $y$ such that $c - 1 < x < c$ and $d - 1 < y < d$ appear in 1. for some $c, d$ whether $\text{frac}(x)$ is less than, greater than, or equal to $\text{frac}(y)$

Note that number of regions is finite
A clock region $b$ is a successor of a clock region $a$ iff for each $n \in a$ there exists a positive $t \in \mathbb{R}$ such that $n + t \in b$. 

Clock regions IV.
How to construct the successors of region \( a \)?

- If for each clock \( x \) satisfies \( x > x_c \), then the only successor of \( a \) is this region itself.

- Denote \( C_0 \) set of clocks such that \( x = c \), for a clock \( x \in C_0 \) in the clock set, successors of \( a \) are defined as set \( b \) as follows:
  - If \( x = c_x \), then \( b \) satisfies \( x > c_x \), otherwise \( b \) satisfies \( c < x < c + 1 \).
  - For \( x \notin C_0 \) the constraint in \( b \) is the same as in \( a \).

- If neither of the above applies, then...
Let $C_0$ be a set of clocks $x$ such that region $a$ does not satisfy $x > c_x$ and for all $y \in C_0$:

\[ \text{frac}(y) \leq \text{frac}(x) \]

Let $b$ be the clock region:

- For $x \in C_0$ if $a$ satisfies $c - 1 < x < c$ then $b$ satisfies $x = c$, for $x \notin C_0$ the constraint in $b$ is the same as in $a$
- For clocks $x, y$ such that $c - 1 < x < c$ and $d - 1 < y < d$ appearing above, the ordering in $b$ between fractional parts is the same as in $a$

Successors of $a$ include $a, b$ and all successors of $b$
Informally:

- Successors of a region are all regions that can be directly reached by moving diagonally up, i.e., increasing the time of all clocks
- The successor relation is transitive
Region successors – example
Region automaton

For a timed automaton $A = (\Sigma, S, S_0, C, E, F)$, corresponding region automaton $R(A)$ is defined:

- States of $R(A)$ are of the form $(s, a)$ where $s \in S$ and $a$ is a clock region.
- Initial states are of the form $(s_0, [n_0])$ where $s_0 \in S_0$ and $n_0(x) = 0$ for all $x \in C$.
- $R(A)$ has edge $((s, a), (s', a'), m)$ iff there is edge $(s, s', m, \lambda, \varphi) \in E$ and region $a''$ such that:
  - $a''$ is successor of $a$.
  - $a''$ satisfies $\varphi$.
  - $a' = [\lambda \rightarrow 0]a''$.

Jan Kofroň, František Plášil, Lecture 9
Region automaton – example
Lemma: If \( r \) is a progressive run of \( R(A) \) over \( s \), then there exists a time sequence \( t \) and a run \( r' \) of \( A \) over \((s, t)\) such that \( r \) equals \([r']\).

- Progressive means that for all clocks there is no bound
- We can consider just progressive runs
  - Proof skipped 😊
Theorem: Given Timed automaton $A = (\Sigma, S, S_0, \Delta, F)$, there exists Büchi automaton which accepts $Untime(L(A))$.

Idea:

1. Construct region automaton $R(A)$
2. Set of accepting states $F' = \{(s, a) | s \in F\}$
3. Omit time
Network of TA

For modeling communicating parts of system in independent way

Each part represented by a single TA
- Communicates with other parts through input/output actions

Composition resulting in parallel synchronous product
Network of TA
A tool for verification of TA models

Academic, but quite well established and used in industry nowadays

Allows modeling, verification, simulation

Successfully applied on communication protocols, multimedia applications, ...

Available at http://www.uppaal.org/ and http://www.uppaal.com