Real-Time Systems

Lecture 15: The Universality Problem for TBA

2013-07-02

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:
- Timed Words and Languages \textsf{[Alur and Dill, 1994]}

This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What’s a TBA and what’s the difference to (extended) TA?
  - What’s undecidable for timed (Büchi) automata?
  - What’s the idea of the proof?
- **Content:**
  - Timed Büchi Automata and timed regular languages \textsf{[Alur and Dill, 1994].}
  - The Universality Problem is undecidable for TBA \textsf{[Alur and Dill, 1994].}
  - Why this is unfortunate.
  - Timed regular languages are not everything.
Timed Büchi Automata

[Alur and Dill, 1994]
Recall: Timed Languages

Definition. A time sequence $\tau = \tau_1, \tau_2, \ldots$ is an infinite sequence of time values $\tau_i \in \mathbb{R}_0^+$, satisfying the following constraints:

(i) **Monotonicity:**
   $\tau$ increases strictly monotonically, i.e. $\tau_i < \tau_{i+1}$ for all $i \geq 1$.

(ii) **Progress:** For every $t \in \mathbb{R}_0^+$, there is some $i \geq 1$ such that $\tau_i > t$.

Definition. A timed word over an alphabet $\Sigma$ is a pair $(\sigma, \tau)$ where

- $\sigma = \sigma_1, \sigma_2, \cdots \in \Sigma^\omega$ is an infinite word over $\Sigma$, and
- $\tau$ is a time sequence.

Definition. A timed language over an alphabet $\Sigma$ is a set of timed words over $\Sigma$. 
Recall:

**Example: Timed Language**

**Timed word** over alphabet $\Sigma$: a pair $(\sigma, \tau)$ where
- $\sigma = \sigma_1, \sigma_2, \ldots$ is an infinite word over $\Sigma$, and
- $\tau$ is a time sequence (strictly (!) monotonic, non-Zeno).

$$L_{crt} = \{((ab)^\omega, \tau) \mid \exists i \forall j \geq i : (\tau_{2j} < \tau_{2j-1} + 2)\}$$
**Definition.** The set $\Phi(X)$ of **clock constraints** over $X$ is defined inductively by

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

where $x \in X$ and $c \in Q$ is a rational constant.

**Definition.** A **timed Büchi automaton** (TBA) $\mathcal{A}$ is a tuple $(\Sigma, S, S_0, X, E, F)$, where

- $\Sigma$ is an alphabet,
- $S$ is a finite set of states, $S_0 \subseteq S$ is a set of start states,
- $X$ is a finite set of clocks, and
- $E \subseteq S \times S \times \Sigma \times 2^X \times \Phi(X)$ gives the set of transitions.

An edge $(s, s', a, \lambda, \delta)$ represents a transition from state $s$ to state $s'$ on input symbol $a$. The set $\lambda \subseteq X$ gives the clocks to be reset with this transition, and $\delta$ is a clock constraint over $X$.

- $F \subseteq S$ is a set of **accepting states**.
Example: TBA

\[ \mathcal{A} = (\Sigma, S, S_0, X, E, F) \]
\[(s, s', a, \lambda, \delta) \in E\]

\[ \Sigma = \{a, b\} \]
\[ S = \{s_0, \ldots, s_3\} \]
\[ S_0 = \{s_0\} \]
\[ X = \{x\} \]
\[ E = \{(s_2, s_3, b, \emptyset, x < 2), \_\} \]
\[ F = \{s_2\} \]
(Accepting) TBA Runs

**Definition.** A run \( r \), denoted by \( (\bar{s}, \bar{\nu}) \), of a TBA \( (\Sigma, S, S_0, X, E, F) \) over a timed word \( (\sigma, \tau) \) is an **infinite** sequence of the form

\[
\begin{align*}
  r : & (s_0, \nu_0) \xrightarrow{\sigma_1, \tau_1} (s_1, \nu_1) \xrightarrow{\sigma_2, \tau_2} (s_2, \nu_2) \xrightarrow{\sigma_3, \tau_3} \ldots
\end{align*}
\]

with \( s_i \in S \) and \( \nu_i : X \rightarrow \mathbb{R}_0^+ \), satisfying the following requirements:

- **Initiation:** \( s_0 \in S_0 \) and \( \nu(x) = 0 \) for all \( x \in X \).
- **Consecution:** for all \( i \geq 1 \), there is an edge in \( E \) of the form \((s_{i-1}, s_i, \sigma_i, \lambda_i, \delta_i)\) such that
  - \( (\nu_{i-1} + (\tau_i - \tau_{i-1})) \) satisfies \( \delta_i \) and
  - \( \nu_i = (\nu_{i-1} + (\tau_i - \tau_{i-1}))[\lambda_i := 0] \).
Example: TBA

\[ A = (\Sigma, S, S_0, X, E, F) \]

\[ (s, s', a, \lambda, \delta) \in E \]
(Accepting) TBA Runs

**Definition.** A run \( r \), denoted by \((\bar{s}, \bar{\nu})\), of a TBA \((\Sigma, S, S_0, X, E, F)\) over a timed word \((\sigma, \tau)\) is an **infinite** sequence of the form

\[
\begin{align*}
    r : (s_0, \nu_0) &\xrightarrow{\sigma_1} (s_1, \nu_1) &\xrightarrow{\sigma_2} (s_2, \nu_2) &\xrightarrow{\sigma_3} \ldots
\end{align*}
\]

with \( s_i \in S \) and \( \nu_i : X \to \mathbb{R}^+ \), satisfying the following requirements:

- **Initiation:** \( s_0 \in S_0 \) and \( \nu(x) = 0 \) for all \( x \in X \).
- **Consecution:** for all \( i \geq 1 \), there is an edge in \( E \) of the form \((s_{i-1}, s_i, \sigma_i, \lambda_i, \delta_i)\) such that
  - \( (\nu_{i-1} + (\tau_i - \tau_{i-1})) \) satisfies \( \delta_i \) and
  - \( \nu_i = (\nu_{i-1} + (\tau_i - \tau_{i-1})) \cdot [\lambda_i := 0] \).

The set \( \text{inf}(r) \subseteq S \) consists of those states \( s \in S \) such that \( s = s_i \) for infinitely many \( i \geq 0 \).

**Definition.** A run \( r = (\bar{s}, \bar{\nu}) \) of a TBA over timed word \((\sigma, \tau)\) is called (an) **accepting** (run) if and only if \( \text{inf}(r) \cap F \neq \emptyset \).
Example: (Accepting) Runs

\[ r : \langle s_0, \nu_0 \rangle \xrightarrow{\sigma_1^\tau_1} \langle s_1, \nu_1 \rangle \xrightarrow{\sigma_2^\tau_2} \langle s_2, \nu_2 \rangle \xrightarrow{\sigma_3^\tau_3} \ldots \text{ initial and } (s_{i-1}, s_i, \sigma_i, \lambda_i, \delta_i) \in E, \text{ s.t.} \]
\[ (\nu_{i-1} + (\tau_i - \tau_{i-1})) = \delta_i, \nu_i = (\nu_{i-1} + (\tau_i - \tau_{i-1}))[\lambda_i := 0]. \text{ Accepting iff } \inf(r) \cap \overline{F} \neq \emptyset. \]

Timed word: \((a, 1), (b, 2), (a, 3), (b, 4), (a, 5), (b, 6), \ldots\)

- Can we construct any run? Is it accepting?
  \[ \langle s_0, 0 \rangle \xrightarrow{\frac{1}{a}} \langle s_1, 1 \rangle \xrightarrow{\frac{1}{b}} \langle s_2, 0 \rangle \xrightarrow{\frac{1}{a}} \langle s_3, 1 \rangle \xrightarrow{\frac{1}{b}} \langle s_4, 0 \rangle \ldots \]
  \[ \inf(r) = \{s_2, s_3\} \]
  \[ \inf(r) \cap \overline{F} = \{s_2, s_3\} \cap \overline{F} = \{s_2\} \neq \emptyset \]

- Can we construct a non-run (get stuck)?
  NO

- Can we construct a (non-)accepting run?
  \[ \langle s_0, 0 \rangle \xrightarrow{\frac{1}{a}} \langle s_1, 1 \rangle \xrightarrow{\frac{1}{b}} \langle s_0, 2 \rangle \xrightarrow{\frac{1}{a}} \langle s_1, 3 \rangle \ldots \]
  \[ \inf(r') = \{s_0, s_1\} \]
The Language of a TBA

**Definition.** For a TBA \( A \), the **language** \( L(A) \) of timed words it accepts is defined to be the set

\[
\{(\sigma, \tau) \mid A \text{ has an accepting run over } (\sigma, \tau)\}.
\]

For short: \( L(A) \) is the **language of** \( A \).

**Definition.** A timed language \( L \) is a **timed regular language** if and only if \( L = L(A) \) for some TBA \( A \).
Example: Language of a TBA

\[ L(A) = \{ (\sigma, \tau) \mid A \text{ has an accepting run over } (\sigma, \tau) \}. \]

Claim:

\[ L(A) = L_{\text{crt}} = \{ ((ab)^{\omega}, \tau) \mid \exists i \forall j \geq i : (\tau_{2j} < \tau_{2j-1} + 2) \} \]

- \( L_{\text{crt}} \subseteq L(A) \): pick some \((\sigma, \tau) \in L_{\text{crt}} \). Construct an accepting run of \( A \).
- \( L(A) \subseteq L_{\text{crt}} \): pick some \((\sigma, \tau) \in L(A) \). Then there is an accepting run \((s, \nu)\) over \((\sigma, \tau)\).

Question: Is \( L_{\text{crt}} \) timed regular or not?
The Universality Problem is Undecidable for TBA

[Alur and Dill, 1994]
The Universality Problem

- **Given:** A TBA $A$ over alphabet $\Sigma$.
- **Question:** Does $A$ accept all timed words over $\Sigma$?
  
  In other words: Is $L(A) = \{ (\sigma, \tau) | \sigma \in \Sigma^\omega, \tau \text{ time sequence} \}$.

$\Sigma = \{a, b, c\}$

$A$:  $\vdots$ $a$
    $b$
    $c$

... is universal
The Universality Problem

• **Given:** A TBA $A$ over alphabet $\Sigma$.

• **Question:** Does $A$ accept all timed words over $\Sigma$?

  In other words: Is $L(A) = \{(\sigma, \tau) \mid \sigma \in \Sigma^\omega, \tau$ time sequence$\}$.

---

**Theorem 5.2.** The problem of deciding whether a timed automaton over alphabet $\Sigma$ accepts all timed words over $\Sigma$ is $\Pi^1_1$-hard.

("The class $\Pi^1_1$ consists of highly undecidable problems, including some nonarithmetical sets (for an exposition of the analytical hierarchy consult, for instance [Rogers, 1967].)

**Recall:** With classical Büchi Automata (untimed), this is different:

• Let $B$ be a Büchi Automaton over $\Sigma$.

  $B$ is universal if and only if $L(B) = \emptyset$.

• $B'$ such that $L(B') = \overline{L(B)}$ is effectively computable.

• Language emptyness is decidable for Büchi Automata.
Theorem 5.2. The problem of deciding whether a timed automaton over alphabet $\Sigma$ accepts all timed words over $\Sigma$ is $\Pi^1_1$-hard.

Proof Idea:

- Consider a language $L_{undec}$ which consists of the recurring computations of a 2-counter machine $M$.

- Construct a TBA $A$ from $M$ which accepts the complement of $L_{undec}$, i.e. with

$$L(A) = \overline{L_{undec}}.$$ 

- Then $A$ is universal if and only if $L_{undec}$ is empty...

... which is the case if and only if $M$ doesn’t have a recurring computation.
A two-counter machine $M$

- has two counters $C$, $D$ and
- a finite program consisting of $n$ instructions.
- An instruction increments or decrements one of the counters, or jumps, here even non-deterministically.

A configuration of $M$ is a triple $\langle i, c, d \rangle$:

- program counter $i \in \{1, \ldots, n\}$, values $c, d \in \mathbb{N}_0$ of $C$ and $D$.

A computation of $M$ is an infinite consecutive sequence

\[ \langle 1, 0, 0 \rangle = \langle i_0, c_0, d_0 \rangle, \langle i_1, c_1, d_1 \rangle, \langle i_2, c_2, d_2 \rangle, \ldots \]

that is, $\langle i_{j+1}, c_{j+1}, d_{j+1} \rangle$ is a result executing instruction $i_j$ at $\langle i_j, c_j, d_j \rangle$.

\[ \langle 1, 0, 0 \rangle, \langle 2, 0, 1 \rangle, \langle 3, 1, 1 \rangle, \ldots \]

A computation of $M$ is called recurring iff $i_j = 1$ for infinitely many $j \in \mathbb{N}_0$. 

E.g.

1: inc $D_i$; goto 2
2: inc $C_i$; goto 1, 3
3: dec $D_i$; if ($D=0$) goto 1 else goto 4
4: inc $D_i$; goto 1, 2
Step 1: The Language of Recurring Computations

- Let $M$ be a 2CM with $n$ instructions.

**Wanted:** A timed language $L_{\text{undec}}$ (over some alphabet) representing exactly the recurring computations of $M$.
(In particular s.t. $L_{\text{undec}} = \emptyset$ if and only if $M$ has no recurring computation.)

- Choose $\Sigma = \{b_1, \ldots, b_n, a_1, a_2\}$ as alphabet.

- We represent a configuration $\langle i, c, d \rangle$ of $M$ by the sequence

  $$b_i \underbrace{a_1 \ldots a_1}_{c \text{ times}} \underbrace{a_2 \ldots a_2}_{d \text{ times}} = b_1 a_1^c a_2^d$$
Let $L_{undec}$ be the set of the timed words $(\sigma, \tau)$ with

- $\sigma$ is of the form $b_1 a_1 c_1 d_1 b_2 a_2 c_2 d_2 \ldots$
- $\langle i_1, c_1, d_1 \rangle, \langle i_2, c_2, d_2 \rangle, \ldots$ is a recurring computation of $M$.
- For all $j \in \mathbb{N}_0$,
  - the time of $b_{ij}$ is $j$.
  - if $c_{j+1} = c_j$: for every $a_1$ at time $t$ in the interval $[j, j + 1]$ there is an $a_1$ at time $t + 1$,
  - if $c_{j+1} = c_j + 1$: for every $a_1$ at time $t$ in the interval $[j + 1, j + 2]$, except for the last one, there is an $a_1$ at time $t - 1$,
  - if $c_{j+1} = c_j - 1$: for every $a_1$ at time $t$ in the interval $[j, j + 1]$, except for the last one, there is an $a_1$ at time $t + 1$,

And analogously for the $a_2$'s.
**Step 2: Construct “Observer” for \( L_{\text{undec}} \)**

**Wanted:** A TBA \( \mathcal{A} \) such that

\[
L(\mathcal{A}) = \overline{L_{\text{undec}}},
\]

i.e., \( \mathcal{A} \) accepts a timed word \((\sigma, \tau)\) if and only if \((\sigma, \tau) \notin L_{\text{undec}}\).

**Approach:** What are the reasons for a timed word not to be in \( L_{\text{undec}} \)?

**Recall:** \((\sigma, \tau)\) is in \( L_{\text{undec}} \) if and only if:

- \( \sigma = b_{i_1} a_{i_1}^{c_1} a_{i_2}^{d_1} b_{i_2} a_{i_1}^{c_2} a_{i_2}^{d_2} \)
- \( \langle i_1, c_1, d_1 \rangle, \langle i_2, c_2, d_2 \rangle, \ldots \) is a recurring computation of \( M \).
- the time of \( b_{i_j} \) is \( j \),
- if \( c_{j+1} = c_j \) (\( = c_j + 1 \), \( = c_j - 1 \)): \( \ldots \)
Step 2: Construct “Observer” for $L_{\text{undec}}$

**Wanted:** A TBA $A$ such that

\[ L(A) = \overline{L_{\text{undec}}}, \]

i.e., $A$ accepts a timed word $(\sigma, \tau)$ if and only if $(\sigma, \tau) \notin L_{\text{undec}}$.

**Approach:** What are the reasons for a timed word not to be in $L_{\text{undec}}$?

(i) The $b_i$ at time $j \in \mathbb{N}$ is missing, or there is a spurious $b_i$ at time $t \in ]j, j + 1[$.

(ii) The prefix of the timed word with times $0 \leq t < 1$ doesn't encode $\langle 1, 0, 0 \rangle$.

(iii) The timed word is not recurring, i.e. it has only finitely many $b_i$.

(iv) The configuration encoded in $[j + 1, j + 2[$ doesn’t faithfully represent the effect of instruction $b_j$ on the configuration encoded in $[j, j + 1[$.

**Plan:** Construct a TBA $A_0$ for case (i), a TBA $A_{\text{init}}$ for case (ii), a TBA $A_{\text{recur}}$ for case (iii), and one TBA $A_i$ for each instruction for case (iv).

Then set

\[ A = A_0 \cup A_{\text{init}} \cup A_{\text{recur}} \cup \bigcup_{1 \leq i \leq n} A_i \]
Step 2.(i): Construct $A_0$

(i) The $b_i$ at time $j \in \mathbb{N}$ is missing, or there is a spurious $b_i$ at time $t \in ]j, j + 1[$.

[Alur and Dill, 1994]: “It is easy to construct such a timed automaton.”
Step 2.(ii): Construct $A_{init}$

(ii) The prefix of the timed word with times $0 \leq t < 1$ doesn’t encode $\langle 1, 0, 0 \rangle$.

- It accepts

$$\{(\sigma_j, \tau_j)_{j \in \mathbb{N}_0} \mid (\sigma_0 \neq b_1) \lor (\tau_0 \neq 0) \lor (\tau_1 \neq 1)\}.$$
Step 2.(iii): Construct $A_{\text{recur}}$

(iii) The timed word is not recurring, i.e. it has only finitely many $b_i$.

- $A_{\text{recur}}$ accepts words with only finitely many $b_i$. 

\begin{align*}
&b_1 b_2 b_3 \cdots b_n \cdots b_1 b_2 b_3 b_2 b_3 \cdots \\
&\text{(no run)} \\
&\text{(run, not accepting)} \quad \text{if word is accepted}
\end{align*}
Step 2.(iv): Construct $A_i$

(iv) The configuration encoded in $[j + 1, j + 2]$ doesn’t faithfully represent the effect of instruction $b_i$ on the configuration encoded in $[j, j + 1]$.

Example: assume instruction $7$ is:

Increment counter $D$ and jump non-deterministically to instruction 3 or 5.

Once again: stepwise. $A_7$ is $A_{71} \cup \cdots \cup A_{77}$.

- $A_{71}$ accepts words with $b_7$ at time $j$ but neither $b_3$ nor $b_5$ at time $j + 1$. “Easy to construct.”
- $A_{72}$ is

\[
\begin{align*}
&\bullet \quad \text{\(\ell_0\)} \quad x := 0 \\
&\circ \quad \text{\(\ell_1\)} \quad x < 1 \quad a_1 \quad x := 0 \\
&\circ \quad \text{\(\ell_2\)} \quad x \neq 1
\end{align*}
\]

- $A_{73}$ accepts words which encode unexpected increment of counter $C$.
- $A_{74}, \ldots, A_{77}$ accept words with missing increment of $D$. 
Aha, And...?
Consequences: Language Inclusion

- **Given:** Two TBAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over alphabet $B$.
- **Question:** Is $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$?

Possible applications of a decision procedure:

- Characterise the allowed behaviour as $\mathcal{A}_2$ and model the design as $\mathcal{A}_1$.
- Automatically check whether the behaviour of the design is a subset of the allowed behaviour.

- If **language inclusion** was decidable, then we could use it to decide universality of $\mathcal{A}$ by checking

  $$\mathcal{L}(\mathcal{A}_{univ}) \subseteq \mathcal{L}(\mathcal{A})$$

  where $\mathcal{A}_{univ}$ is any universal TBA (which is easy to construct).
Consequences: Complementation

- **Given**: A timed regular language $W$ over $B$ (that is, there is a TBA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = W$).

- **Question**: Is $\overline{W}$ timed regular?

Possible applications of a decision procedure:

- Characterise the allowed behaviour as $\mathcal{A}_2$ and model the design as $\mathcal{A}_1$.

- Automatically construct $\mathcal{A}_3$ with $L(\mathcal{A}_3) = \overline{L(\mathcal{A}_2)}$ and check

\[
\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_3) = \emptyset,
\]

that is, whether the design has any non-allowed behaviour.

- Taking for granted that:
  - The intersection automaton is effectively computable.
  - The emptyness problem for Büchi automata is decidable.

  (Proof by construction of region automaton [Alur and Dill, 1994].)
Consequences: Complementation

- **Given:** A timed regular language $W$ over $B$ (that is, there is a TBA $A$ such that $\mathcal{L}(A) = W$).
- **Question:** Is $\overline{W}$ timed regular?
- If the class of timed regular languages were closed under complementation, “the complement of the inclusion problem is recursively enumerable. This contradicts the $\Pi^1_1$-hardness of the inclusion problem.” [Alur and Dill, 1994]

A non-complementable TBA $A$:

\[
\begin{align*}
  \mathcal{L}(A) &= \{ (a^\omega, (t_i)_{i \in \mathbb{N}_0}) \mid \exists i \in \mathbb{N}_0 \exists j > i : (t_j = t_i + 1) \} \\
  \overline{\mathcal{L}(A)} &= \{ (a^\omega, (t_i)_{i \in \mathbb{N}_0}) \mid \text{no two } a \text{ are separated by distance 1} \}.
\end{align*}
\]
Beyond Timed Regular
Beyond Timed Regular

With clock constraints of the form

\[ x + y \leq x' + y' \]

we can describe timed languages which are not timed regular.

In other words:

- There are strictly timed languages than timed regular languages.
- There exists timed languages \( \mathcal{B} \) such that there exists no \( A \) with \( L(A) = \mathcal{B} \).

Example:

\[ \{(abc)^\omega, \tau) \mid \forall j. (\tau_{3j} - \tau_{3j-1}) = 2(\tau_{3j-1} - \tau_{3j-2}) \} \]
References
References
