Real-Time Systems

Lecture 16: The Universality Problem for TBA

2011-07-19

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- Using Uppaal to check whether a TA satisfies a DC requirement:
  Testable DC properties

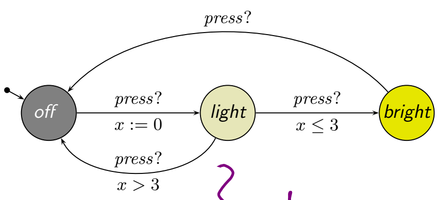
This Lecture:

- **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 [Alur and Dill, 1994].
  - The Universality Problem is undecidable for TBA [Alur and Dill, 1994]
  - Why this is unfortunate.
  - Timed regular languages are not everything.

Timed Büchi Automata

[Alur and Dill, 1994]

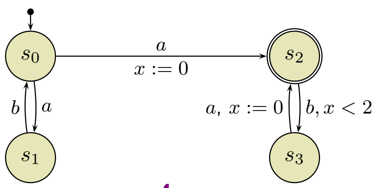
. . . vs. Timed Automata



produces

$$\xi = \langle \mathit{off}, 0\rangle, 0 \xrightarrow{1} \langle \mathit{off}, 1\rangle, 1 \xrightarrow{press?} \langle \mathit{light}, 0\rangle, 1 \xrightarrow{3} \langle \mathit{light}, 3\rangle, 4 \xrightarrow{press?} \langle \mathit{bright}, 3\rangle, 4 \xrightarrow{\because} \ldots$$

ξ is a computation path and run of $\mathcal{A}$.



accept

New: Given a **timed word**

$$(a, 1), (b, 2), (a, 3), (b, 4), (a, 5), (b, 6), \ldots,$$

does $\mathcal{A}$ **accept** it?

New: acceptance criterion is **visiting accepting state infinitely often**.

Timed Languages

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

(i) **Monotonicity**:
τ 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 $\tau_i > t$.

Infinite sequences of letters from Σ

Definition. A **timed word** over an alphabet Σ is a pair (σ, τ) where

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

Definition. A **timed language** over an alphabet Σ is a set of timed words over Σ.

Example: Timed Language

Timed word over alphabet Σ: a pair (σ, τ) where
- $\sigma = \sigma_1, \sigma_2, \dots$ is an infinite word over Σ, and
- τ is a time sequence (strictly (!) monotonic, non-Zeno).

a could be 'system beeps'
b could be 'system flashes light'

$$L_{crt} = \{((ab)^\omega, \tau) \mid \exists\, i\; \forall\, j \geq i : (\tau_{2j} < \tau_{2j-1} + 2)\}$$

timestamp of a 'b'

timestamp of the a before

a b a b a b ... a b a b a b ...

τ_1 τ_2 τ_3 τ_4 τ_5 τ_6 τ_{27} τ_{28} τ_{29} τ_{30} τ_{31} τ_{32}

finite prefix where timestamps don't matter

from here on ...

$\tau_{30} < \tau_{29} + 2$

(not including)

"b at most 2 time units after the a before"

Timed Büchi Automata

Definition. The set $\Phi(X)$ of **clock constraints** over X is defined inductively by

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

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

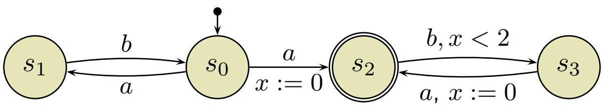
Definition. A **timed Büchi automaton** (TBA) $\mathcal{A}$ is a tuple $(\Sigma, S, S_0, X, E, F)$, where

- Σ 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 δ is a clock constraint over X.
- $F \subseteq S$ is a set of **accepting states**.

Example: TBA

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



- $\Sigma = \{a, b\}$
- $S = \{s_0, \dots, s_3\}$
- $S_0 = \{s_0\}$
- $X = \{x\}$
- $F = \{s_2\}$
- $E = \{(s_0, s_1, a, \emptyset, true),$
  $(s_1, s_0, b, \emptyset, true),$
  $(s_0, s_2, a, \{x\}, true),$
  $(s_2, s_3, b, \emptyset, x<2),$
  $(s_3, s_2, a, \{x\}, true)\}$

(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 (σ, τ) is an **infinite** sequence of the form

$$r : \underbrace{\langle s_0, \nu_0 \rangle \xrightarrow[\tau_1]{\sigma_1} \langle s_1, \nu_1 \rangle \xrightarrow[\tau_2]{\sigma_2} \langle s_2, \nu_2 \rangle}_{*} \xrightarrow[\tau_3]{\sigma_3} \dots$$

with $s_i \in S$ and $\nu_i : X \to \mathbb{R}_0^+$, 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 δ_i and
  - $\nu_i = (\nu_{i-1} + \underbrace{(\tau_i - \tau_{i-1})}))[\lambda_i := 0]$.

time shift

"how much time has passed since reading $(\sigma_{i-1}, \tau_{i-1})$"

(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 (σ, τ) is an **infinite** sequence of the form

$$r : \langle s_0, \nu_0 \rangle \xrightarrow[\tau_1]{\sigma_1} \langle s_1, \nu_1 \rangle \xrightarrow[\tau_2]{\sigma_2} \langle s_2, \nu_2 \rangle \xrightarrow[\tau_3]{\sigma_3} \dots$$

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

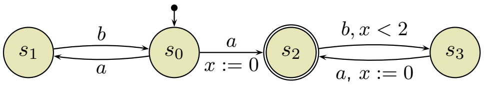
- **Initiation**: $s_0 \in S_0$ and $\nu(x) = 0$ for all $x \in X$.
- **Consecution**: for all $i \geq 1$, there is an edge in E of the form $(s_{i-1}, s_i, \sigma_i, \lambda_i, \delta_i)$ such that
  - $(\nu_{i-1} + (\tau_i - \tau_{i-1}))$ satisfies δ_i and
  - $\nu_i = (\nu_{i-1} + (\tau_i - \tau_{i-1}))[\lambda_i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

The set $inf(r) \subseteq S$ 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 (σ, τ) is called (an) **accepting** (run) if and only if $inf(r) \cap F \neq \emptyset$.

Example: (Accepting) Runs

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



Timed word: $(a, 1), (b, \underset{2.1}{\cancel{2}}), (a, 3), (b, 4), (a, 5), (b, 6), \ldots$

- Can we construct **any run**? Is it accepting?

$r : \langle s_0, x=0 \rangle \xrightarrow[1]{a} \langle s_2, x=0 \rangle \xrightarrow[2.1]{b} \langle s_3, x=1.1 \rangle \xrightarrow[3]{a} \langle s_2, x=0 \rangle \xrightarrow[4]{b} \langle s_3, x=1 \rangle \rightarrow \ldots$

$\mathit{inf}(r) = \{s_2, s_3\}$, r is accepting since $\mathit{inf}(r) \cap F = \{s_2\} \neq \emptyset$

- Can we construct a **non-run**?

$r' : \langle s_0, 0 \rangle \xrightarrow[1]{a} \langle s_2, 1 \rangle$ ← wrong valuation

- Can we construct a **(non-)accepting run**?

$r'' : \langle s_0, 0 \rangle \xrightarrow[1]{a} \langle s_1, 1 \rangle \xrightarrow[2.1]{b} \langle s_0, 2.1 \rangle \xrightarrow[3]{a} \langle s_1, 3 \rangle \rightarrow \ldots$

The Language of a TBA

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

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

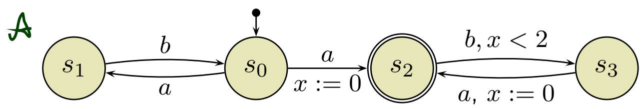
For short: $L(\mathcal{A})$ is the **language of** $\mathcal{A}$.

!

Definition. A timed language L is a **timed regular language** if and only if $L = L(\mathcal{A})$ for **some** TBA $\mathcal{A}$.

Example: Language of a TBA

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



Claim:

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

- $L_{crt} \subseteq L(\mathcal{A})$: "pick some" $(\sigma,\tau) \in L_{crt}$. Construct an accepting run in $\mathcal{A}$.
- $L(\mathcal{A}) \subseteq L_{crt}$: "pick some" $(\sigma,\tau) \in L(\mathcal{A})$, thus there is an accepting run $(\bar{s},\bar{\nu})$ over (σ,τ).

Question: Is L_{crt} timed regular or not?

The Universality Problem is Undecidable for TBA

[Alur and Dill, 1994]

The Universality Problem

- **Given:** A TBA $\mathcal{A}$ over alphabet Σ.
- **Question:** Does $\mathcal{A}$ accept all timed words over Σ?

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

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

... is universal

The Universality Problem

- **Given:** A TBA $\mathcal{A}$ over alphabet Σ.
- **Question:** Does $\mathcal{A}$ accept all timed words over Σ?

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

Theorem 5.2. The problem of deciding whether a timed automaton over alphabet Σ accepts all timed words over Σ is Π^1_1-hard.

("The class Π^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 $\mathcal{B}$ be a Büchi Automaton over Σ. complement in Σ^ω
- $\mathcal{B}$ is universal if and only if $\overline{L(\mathcal{B})} = \emptyset$.
- $\mathcal{B}'$ such that $L(\mathcal{B}') = \overline{L(\mathcal{B})}$ is effectively computable.
- Language emptyness is decidable for Büchi Automata.

Proof Idea

Theorem 5.2. The problem of deciding whether a timed automaton over alphabet Σ accepts all timed words over Σ is Π_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 $\mathcal{A}$ from M which accepts the complement of L_{undec}, i.e. with
  $$L(\mathcal{A}) = \overline{L_{undec}}.$$
- Then $\mathcal{A}$ 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.

Once Again: Two Counter Machines (Different Flavour)

something like:
1: inc C; goto 2,3
2: dec C; goto 1; goto 3
3: inc D; goto 3

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.

  will care for $c, d \geq 0$

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

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

- A **computation** of M is an infinite consecutive sequence
  $$\langle 1, 0, 0 \rangle = \langle i_0, c_0, d_0 \rangle, \langle i_1, c_1, d_1 \rangle, \langle i_2, c_2, d_2 \rangle, \dots$$
  that is, $\langle i_{j+1}, c_{j+1}, d_{j+1} \rangle$ is a result executing instruction i_j at $\langle i_j, c_j, d_j \rangle$.

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

Step 1: The Language of Recurring Computations

- Let M be a 2CM with n instructions.

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

- Choose $\Sigma = \{b_1, \dots, b_n, a_1, a_2\}$ as alphabet.
- We represent a configuration $\langle i, c, d \rangle$ of M by the sequence

$$b_i \underbrace{a_1 \dots a_1}_{c \text{ times}} \underbrace{a_2 \dots a_2}_{d \text{ times}} = b_1 a_1^c a_2^d$$

Step 1: The Language of Recurring Computations

$\langle i, c, d \rangle$ represented by $b_1 a_1^c a_2^d$

Let L_{undec} be the set of the timed words (σ, τ) with

Step 1: The Language of Recurring Computations

$\langle i, c, d \rangle$ represented by $b_1 a_1^c a_2^d$

Let L_{undec} be the set of the timed words (σ, τ) with

- σ is of the form $b_{i_1} a_1^{c_1} a_2^{d_1} b_{i_2} a_1^{c_2} a_2^{d_2} \ldots$
- $\langle i_1, c_1, d_1 \rangle, \langle i_2, c_2, d_2 \rangle, \ldots$ is a recurring computation of M.
- For all $j \in \mathbb{N}_0$,
  - the time of b_{i_j} is j.
  - if $c_{j+1} = c_j$ then for every a_1 at time t in the interval $[j, j+1]$ there is an a_1 at time $t+1$,
  - if $c_{j+1} = c_j + 1$ then 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 $c_{j+1} = c_j - 1$ then 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 $\overline{L_{undec}}$

Wanted: A TBA $\mathcal{A}$ such that

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

Step 2: Construct "Observer" for $\overline{L_{undec}}$

Wanted: A TBA $\mathcal{A}$ such that

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

What are the reasons for a timed word **not to be** in L_{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_i on the configuration encoded in $[j, j+1[$.

– 16 – 2011-07-19 – Suniv –

Step 2: Construct "Observer" for $\overline{L_{undec}}$

Wanted: A TBA $\mathcal{A}$ such that

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

What are the reasons for a timed word **not to be** in L_{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_i on the configuration encoded in $[j, j+1[$.

Plan: Construct a TBA $\mathcal{A}_0$ for case (i), a TBA $\mathcal{A}_{init}$ for case (ii), a TBA $\mathcal{A}_{recur}$ for case (iii), and one TBA $\mathcal{A}_i$ for each instruction for case (iv).

Then set

$$\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_{init} \cup \mathcal{A}_{recur} \cup \bigcup_{1 \leq i \leq n} \mathcal{A}_i$$

– 16 – 2011-07-19 – Suniv –

Step 2.(i): Construct $\mathcal{A}_0$

(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 $\mathcal{A}_{init}$

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

- It accepts

$$\{(\sigma_j, \tau_j)_{j \in \mathbb{N}_0} \mid (\sigma_0 \neq b_1) \vee (\tau_0 \neq 0) \vee (\tau_1 \neq 1)\}.$$

Step 2.(iii): Construct $\mathcal{A}_{recur}$

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

- $\mathcal{A}_{recur}$ accepts words with only finitely many b_i.

Step 2.(iv): Construct $\mathcal{A}_i$

(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. $\mathcal{A}_7$ is $\mathcal{A}_7^1 \cup \cdots \cup \mathcal{A}_7^6$.

Step 2.(iv): Construct $\mathcal{A}_i$

(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. $\mathcal{A}_7$ is $\mathcal{A}_7^1 \cup \cdots \cup \mathcal{A}_7^6$.

- $\mathcal{A}_7^1$ accepts words with b_7 at time j but neither b_3 nor b_5 at time $j+1$. "Easy to construct."

Step 2.(iv): Construct $\mathcal{A}_i$

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

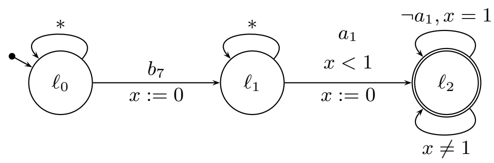
Example: assume instruction 7 is:

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

Once again: stepwise. $\mathcal{A}_7$ is $\mathcal{A}_7^1 \cup \cdots \cup \mathcal{A}_7^6$.

- $\mathcal{A}_7^1$ accepts words with b_7 at time j but neither b_3 nor b_5 at time $j+1$. "Easy to construct."

- $\mathcal{A}_7^2$ is



Step 2.(iv): Construct $\mathcal{A}_i$

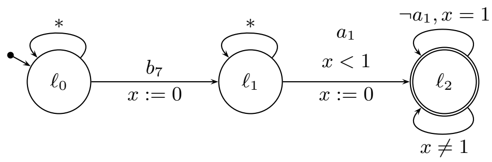
(iv) The configuration encoded in $[j+1, j+2[$ doesn't faithfully represent the effect of instruction b_i on the configuration encoded in $[j, j+1[$.

Example: assume instruction 7 is:

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

Once again: stepwise. $\mathcal{A}_7$ is $\mathcal{A}_7^1 \cup \cdots \cup \mathcal{A}_7^6$.

- $\mathcal{A}_7^1$ accepts words with b_7 at time j but neither b_3 nor b_5 at time $j+1$.
  "Easy to construct."
- $\mathcal{A}_7^2$ is



- $\mathcal{A}_7^3$ accepts words which encode unexpected increment of counter C.
- $\mathcal{A}_7^4, \ldots, \mathcal{A}_7^6$ accept words with missing decrement 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.

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

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

  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 $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = W$).
- **Question:** Is $\overline{W}$ timed regular?

Consequences: Complementation

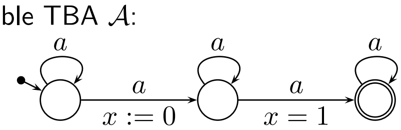
- **Given:** A timed regular language W over B
(that is, there is a TBA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = W$).
- **Question:** Is $\overline{W}$ timed regular?
- If the class of timed regular languages were closed under **complementation**, "the complement of the inclusion problem is recursively enumerable. This contradicts the Π_1^1-hardness of the inclusion problem." [Alur and Dill, 1994]

– 16 – 2011-07-19 – Sjaund –

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?
- If the class of timed regular languages were closed under **complementation**, "the complement of the inclusion problem is recursively enumerable. This contradicts the Π_1^1-hardness of the inclusion problem." [Alur and Dill, 1994]

A non-complementable TBA $\mathcal{A}$:



$$\mathcal{L}(\mathcal{A}) = \{(a^\omega, (t_i)_{i \in \mathbb{N}_0}) \mid \exists\, i \in \mathbb{N}_0 \; \exists\, j > i : (t_j = t_i + 1)\}$$

Complement language:

$$\overline{\mathcal{L}(\mathcal{A})} = \{(a^\omega, (t_i)_{i \in \mathbb{N}_0}) \mid \text{no two } a \text{ are separated by distance } 1\}.$$

Beyond Timed Regular

Beyond Timed Regular

With clock constraints of the form

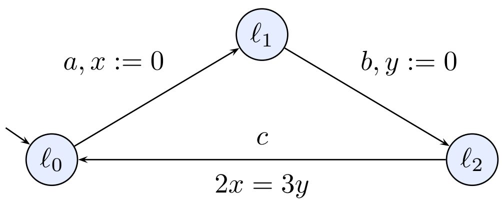
$$x + y \leq x' + y'$$

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 L such that there exists no $\mathcal{A}$ with $L(\mathcal{A}) = L$.

Example:



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

References

References

[Alur and Dill, 1994] Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235.

[Olderog and Dierks, 2008] Olderog, E.-R. and Dierks, H. (2008). *Real-Time Systems - Formal Specification and Automatic Verification*. Cambridge University Press.