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]

Timed Languages

Definition.
A time sequence τ = τ₁, τ₂, ... is an infinite sequence of time values τᵢ ∈ R⁺, satisfying the following constraints:

(i) Monotonicity:
τᵢ increases strictly monotonically, i.e., τᵢ < τᵢ₊₁ for all i ≥ 1.

(ii) Progress:
For every t ∈ R⁺, there is some i ≥ 1 such that τᵢ > t.

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

• σ = σ₁, σ₂, ... ∈ Σω is an infinite word over Σ,
• 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

• σ = σ₁, σ₂, ... is an infinite word over Σ,
• τ is a time sequence (strictly (!) monotonic, non-Zeno).

L_{cr} = \{((ab)ω, τ) | ∃i ∀j ≥ i : (τ₂j < τ₂j−1 + 2)\}
A timed language is the language of a TBA. For an FTA, the language is the set of all labeled sequences of states in which \(s\) is reached, where \(\delta(s, \tau, \nu) \neq \emptyset\) and \(\tau > 0\) for \(\nu\) and \(\sigma\) such that \(\sigma = \nu + \tau\). There is an edge in the graph from state \(s\) to \(s'\) for any \(\nu\) and \(\tau\) such that \(\delta(s, \tau, \nu) \neq \emptyset\) and \(\tau > 0\) for any \(\nu\) and \(\sigma\) such that \(\sigma = \nu + \tau\). For an FTA, the language is the set of all labeled sequences of states in which \(s\) is reached, where \(\delta(s, \tau, \nu) \neq \emptyset\) and \(\tau > 0\) for any \(\nu\) and \(\sigma\) such that \(\sigma = \nu + \tau\). There is an edge in the graph from state \(s\) to \(s'\) for any \(\nu\) and \(\tau\) such that \(\delta(s, \tau, \nu) \neq \emptyset\) and \(\tau > 0\) for any \(\nu\) and \(\sigma\) such that \(\sigma = \nu + \tau\).
The Universality Problem is Undecidable for TBA [Alur and Dill, 1994]

The Universality Problem

• Given: A TBA A over alphabet Σ.

• Question: Does A accept all timed words over Σ? In other words: Is \( L(A) = \{ (\sigma, \tau) | \sigma \in \Sigma^\omega, \tau \text{ timesequence} \} \).

Theorem 5.2. The problem of deciding whether a timed automaton over alphabet Σ accepts all timed words over Σ is \( \Pi_{11} \)-hard. ("The class \( \Pi_{11} \) consists of highly undecidable problems, including some non-arithmetical sets (for an exposition of the analytical hierarchy consult, for instance [Rogers, 1967].)"

Recall: With classical B"uchi Automata (untimed), this is different:

• Let B be a B"uchi Automaton over Σ.

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

• B' such that \( L(B') = L(B) \) is effectively computable.

• Language emptyness is decidable for B"uchi Automata.

Proof Idea: Theorem 5.2. The problem of deciding whether a timed automaton over alphabet Σ accepts all timed words over Σ is \( \Pi_{11} \)-hard.

Proof Idea:

• Consider a language \( L \text{undec} \) which consists of the recurring computations of a 2-counter machine \( M \).

• Construct a TBA \( A \) from \( M \) which accepts the complement of \( L \text{undec} \), i.e. with \( L(A) = L \text{undec} \).

• Then \( A \) is universal if and only if \( L \text{undec} \) is empty... which is the case if and only if \( M \) doesn't have a recurring computation.

Once Again: Two Counter Machines (Different Flavour)

A two-countermachine \( M \)

• 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 \( \langle i_j, c_j, d_j \rangle \).

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

The language of recurring computations is denoted by $\mathcal{L}$.

For all $\sigma, \tau \in \Sigma$, $\mathcal{L}(\sigma, \tau)$ is of the form $\langle i, c, d \rangle$.

$\mathcal{L}$ only if $\langle i, c, d \rangle \in M \equiv \{(i, c, d) \in \mathbb{N}^3 : \sigma, \tau \in \Sigma \}$.

$\mathcal{L}$ corresponds to the language of recurring computations.

Step 2: Construct "Observer" for $\mathcal{L}$

For all $\sigma, \tau \in \Sigma$, $\mathcal{L}(\sigma, \tau)$ is of the form $\langle i, c, d \rangle$.

$\mathcal{L}$ only if $\langle i, c, d \rangle \in M \equiv \{(i, c, d) \in \mathbb{N}^3 : \sigma, \tau \in \Sigma \}$.

$\mathcal{L}$ corresponds to the language of recurring computations.

Recurrence

Let $L$ be the language of recurring computations. (\sigma, \tau) \in \Sigma^2$

Then $\langle i, c, d \rangle \in M$.

For all $\sigma, \tau \in \Sigma$, $\mathcal{L}(\sigma, \tau)$ is of the form $\langle i, c, d \rangle$.

$\mathcal{L}$ only if $\langle i, c, d \rangle \in M \equiv \{(i, c, d) \in \mathbb{N}^3 : \sigma, \tau \in \Sigma \}$.

$\mathcal{L}$ corresponds to the language of recurring computations.

Step 1: The Language of Recurring Computations

The language of recurring computations is denoted by $\mathcal{L}$.

For all $\sigma, \tau \in \Sigma$, $\mathcal{L}(\sigma, \tau)$ is of the form $\langle i, c, d \rangle$.

$\mathcal{L}$ only if $\langle i, c, d \rangle \in M \equiv \{(i, c, d) \in \mathbb{N}^3 : \sigma, \tau \in \Sigma \}$.

$\mathcal{L}$ corresponds to the language of recurring computations.

Step 2: Construct "Observer" for $\mathcal{L}$

For all $\sigma, \tau \in \Sigma$, $\mathcal{L}(\sigma, \tau)$ is of the form $\langle i, c, d \rangle$.

$\mathcal{L}$ only if $\langle i, c, d \rangle \in M \equiv \{(i, c, d) \in \mathbb{N}^3 : \sigma, \tau \in \Sigma \}$.

$\mathcal{L}$ corresponds to the language of recurring computations.

Recurrence

Let $L$ be the language of recurring computations. (\sigma, \tau) \in \Sigma^2$

Then $\langle i, c, d \rangle \in M$.

For all $\sigma, \tau \in \Sigma$, $\mathcal{L}(\sigma, \tau)$ is of the form $\langle i, c, d \rangle$.

$\mathcal{L}$ only if $\langle i, c, d \rangle \in M \equiv \{(i, c, d) \in \mathbb{N}^3 : \sigma, \tau \in \Sigma \}$.

$\mathcal{L}$ corresponds to the language of recurring computations.
Step 2.(i): Construct $A_0$

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

[Alur and Dill, 1994]: "It is easy to construct such a timed automaton."

Step 2.(ii): Construct $A_{\text{init}}$

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

• It accepts $\{ (\sigma_j, \tau_j) : j \in \mathbb{N}_0 | (\sigma_0 \neq b_1) \lor (\tau_0 \neq 0) \lor (\tau_1 \neq 1) \}$. 

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 $b_i$. 

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 and jump non-deterministically to instruction 3 or 5.

Once again: stepwise. $A_7$ is $A_17 \cup \cdots \cup A_{67}$.

• $A_{17}$ 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 a 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

\[A_7 = A_1 \cup \cdots \cup A_{67}.\]

- \(A_1\) accepts words with \(b_7\) at time \(j\) but neither \(b_3\) nor \(b_5\) at time \(j + 1\).

"Easy to construct."

- \(A_{27}\) is \(\ell_0 \ell_1 \ell_2^* b_7 x := 0^* a_1 x < 1 \neg a_1\), \(x = 1\) if \(x \neq 1\).

• \(A_{37}\) accepts words which encode unexpected increment of counter C.

• \(A_{47}, \ldots, A_{67}\) accept words with missing decrement of D.

Consequences: Language Inclusion

- Given: Two TBAs \(A_1\) and \(A_2\) over alphabet \(B\).

- Question: Is \(L(A_1) \subseteq L(A_2)\)?

Possible applications of a decision procedure:

- Characterize the allowed behavior as \(A_2\) and model the design as \(A_1\).

- Automatically check whether the behavior of the design is a subset of the allowed behavior.

If language inclusion was decidable, then we could use it to decide universality of \(A\) by checking \(L(A_{univ}) \subseteq L(A)\) where \(A_{univ}\) 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 \(A\) such that \(L(A) = W\)).

- Question: Is \(W\) timed regular?

Possible applications of a decision procedure:

- Characterize the allowed behavior as \(A_2\) and model the design as \(A_1\).

- Automatically construct \(A_3\) with \(L(A_3) = L(A_2)\) and check \(L(A_1) \cap L(A_3) = \emptyset\), that is, whether the design has any non-allowed behavior.

- Taking for granted that:
  - The intersection automaton is effectively computable.
  - The emptiness 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 $L(A) = W$).

Question:
Is $W$ 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 $\Pi_{11}$-hardness of the inclusion problem." [Alur and Dill, 1994]

An non-complementable TBA $A$:

$$L(A) = \{(a_\omega, (t_i)) | \exists i \in \mathbb{N}_0 \exists j > i : (t_j = t_i + 1)\}$$

Complement language:

$$L(A) = \{(a_\omega, (t_i)) | \not \exists k \in \mathbb{N}_0 : (t_{i+k}) - (t_i) = 1\}.$$