Contents & Goals

Last Lecture:

• Timed Words and Languages
  [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
    [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.

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).

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

Recall:

Timed Büchi Automaton

Definition.

The set \( \Phi(X) \) of clock constraints over \( X \) is defined inductively by

\[ \delta :: = x \leq c | c \leq x | \neg \delta | \delta_1 \land \delta_2 \]

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

Definition.

A timed Büchi automaton (TBA) \( A \) 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 clock to be reset with this transition, and \( \delta \) is a clock constraint over \( X \).

• \(F \subseteq S\) is a set of accepting states.

Recall:

Real-Time Systems

Lecture 15: The Universality Problem for TBA

2013-07-02

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
The Language of a TBA

Example: (Accepting) Runs

The Language of a TBA

Example: (Accepting) Runs

The Language of a TBA

Example: (Accepting) Runs
Claim:
$L(A) = L_{crt}(=\{(σ,τ) | \exists i \forall j > i: τ_{2j} < τ_{2j−1} + 2\})$.

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 $A$ over alphabet $Σ$.
• Question: Does $A$ accept all timed words over $Σ$?

In other words: Is $L(A) = \{(σ,τ) | σ ∈ Σ ω, τ timesequence\}$.

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 non-arithmetic 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 $Σ$.
  • $B$ is universal if and only if $L(B) = ∅$.
  • $B'$ such that $L(B') = L(B)$ is effectively computable.
  • Language emptiness is decidable for Büchi Automata.

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 $L(A) = L_{undec}$.
• 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.

Once Again: Two Counter Machines (Different Flavour)

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 $⟨i, c, d⟩$: program counter $i ∈ \{1, . . . , n\}$, values $c, d ∈ N_0$ of $C$ and $D$.
• A computation of $M$ is an infinite consecutive sequence $⟨1, 0, 0⟩ = ⟨i_0, c_0, d_0⟩, ⟨i_1, c_1, d_1⟩, ⟨i_2, c_2, d_2⟩, . . .$ that is, $⟨i_j+1, c_{j+1}, d_{j+1}⟩$ is a result executing instruction $i_j$ at $⟨i_j, c_j, d_j⟩$.
A computation of $M$ is called recurring iff $i_j = 1$ for infinitely many $j ∈ N_0$. 
For case (ii), a TBA doesn't encode $b, j \in \mathbb{N}$, doesn't encode $1$, or there is a spurious $N \in j$.

Let $\tau(i) = \sigma(j)$ if and only if:

$\exists \text{ recurring computation of } i$.

Choose $0$.

It is easy to construct such a timed automaton.

In particular, $t$. Wanted.

Step 1: The Language of Recurring Computations

Step 2. (ii): Construct

We accept timed words representing exactly $W$.

Step 2. (i): Construct

If $\langle a, c, d, b \rangle$:

$\langle t \rangle$.

Then set $\tau(i)$ for case (iv).

For case (iii), and one TBA $A$

for case (i), a TBA $A$.
Complement language: The emptyness problem for Büchi automata is recursively timelimited (i.e., there is a Büchi automaton which is easy to construct). Any language in inclusion $\omega$-language is decidable.

Question: Character is the allowed behavior as...
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 that are not timed regular.
- There exist timed languages \( L \) such that there exists no \( A \) with \( \text{L}(A) = L \).

Example:
\[
\ell_{1}^{0} a, x := 0 \quad\ell_{2}^{2} x = 3 \quad y = 0
\]
\[
\{ \left( (abc)^{\omega}, \tau \right) \mid \forall j. (\tau_{3}j - \tau_{3}j - 1) = 2(\tau_{3}j - 1 - \tau_{3}j - 2) \}\}
\]