Computational Models — Lecture 10

Handout Mode

Iftach Haitner.

Tel Aviv University.

December 17, 2018

\[1\] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.
Talk Outline

- Mapping reduction, Rice’s theorem and friends, reminder
- Controlled executions
- Linear Bounded Automaton
- Configuration histories
- $\mathcal{RE}$-Completeness
- Sipser’s book, 5.1, 5.3

- Introduction to time complexity (moved to lecture 11)
- Sipser’s book, 7.1
Section 1

Controlled Executions
Bounded time and space

In the following a TM stands for a single-tape deterministic TM.

Definition 1
CET := \{ \langle M, w, k \rangle : M \text{ is a TM that accepts } w \text{ within } k \text{ steps} \}. Is CET ∈ \mathcal{R}?

Theorem 2
CET ∈ \mathcal{R}.

Proof?

Definition 3
CES := \{ \langle M, w, k \rangle : M \text{ is a TM that accepts } w \text{ using } k \text{ cells} \}. Is CES ∈ \mathcal{R}?

Theorem 4
CES is decidable.
Proving \(\text{CES} \in \mathcal{R}\)

How to check that the computation will not terminate?

Proof: \(m = |Q| \cdot |\Gamma|^k \cdot k\) is a bound on the number of \(k\)-cell configurations of \(M\).

Algorithm 5

On input \(\langle M, w, k \rangle\).

1. Emulate \(M(w)\) while maintaining a step counter.
   Counter is incremented by 1 per each simulated step (of \(M\)).

2. Reject if counter reaches \(m + 1\) or \(M\) uses more than \(k\) cells.

3. Accept if \(M\) does.

Correctness follows since if \(M\) does not accept \(w\) within \(m\) steps (using \(k\) cells), then it will never halt ♣
Proving $\text{All}_{\text{TM}} = \mathcal{L}_{\Sigma^*} = \{ \langle M \rangle : M \text{ is a TM and } L(M) = \Sigma^* \} \not\in \mathcal{RE}$

Proof: We show that $\overline{H_{\text{TM}}} \leq_m \text{All}_{\text{TM}}$. Namely, we define a computable $f$ with $f(\langle M, w \rangle) = \langle B_{M,w} \rangle$ such that

- $M(w)$ halts $\implies L(B_{M,w}) \neq \Sigma^*$.
- $M(w)$ does not halt $\implies L(B_{M,w}) = \Sigma^*$.

**Definition 6 ($B_{M,w}$)**

On input $y$:

1. Emulate $M(w)$ for $|y|$ steps.
2. Accept, if $M(w)$ did not halt (in that many steps); otherwise, reject.

- $M(w)$ halts after $k$ steps $\implies B_{M,w}$ accepts only $y$’s of length smaller than $k$ $\implies L(B_{M,w})$ is finite $\implies L(B_{M,w}) \neq \Sigma^*$.
- $M(w)$ does not halt $\implies B_{M,w}$ accepts all $y$’s $\implies L(B_{M,w}) = \Sigma^*$. 
Section 2

Computation Histories
Reduction via Computation Histories

Important technique for proving undecidability. Examples

- Basis for proof of undecidability in Hilbert’s tenth problem (given a polynomial with integer coefficients, does it have a solution over the integers?).

- Enables showing that
  \[ \text{All}_{\text{CFG}} := \{\langle S \rangle : S \text{ is a CFG and } L(S) = \Sigma^* \} \notin \mathcal{RE} \]

Recall that \[ \text{EMPTY}_{\text{CFG}} := \{\langle S \rangle : S \text{ is a CFG and } L(S) = \emptyset \} \in \mathcal{R} \].

- Proof: we use computation histories to show that
  \[ \overline{A_{TM}} \leq_m \text{All}_{\text{PDA}} := \{\langle P \rangle : P \text{ is a PDA and } L(P) = \Sigma^* \} \].

- Hence, \[ \text{All}_{\text{PDA}} \notin \mathcal{RE} \].
Reminder: Configurations

- Configuration: $1011q_70111$, means:
  - state is $q_7$
  - LHS of tape is 1011
  - RHS of tape is 0111
  - head is on RHS 0

- Yield relation
  
  - $uaq_i bv \implies uq_j acv$, if $\delta(q_i, b) = (q_j, c, L)$
  
  - $uaq_i bv \implies uacq_j v$, if $\delta(q_i, b) = (q_j, c, R)$

- Special cases: $q_i bv$ and $uaq_i$

- Special type of configurations: starting, accepting, rejecting, halting

- $h = C_1 \# C_2 \ldots \# C_\ell$ is accepting configuration history of $M$ on $w$,\(^2\) if
  1. $C_1$ is the starting configuration of $M$ on $w$
  2. $C_\ell$ is an accepting configuration of $M$
  3. $\forall i \in [\ell]$: $C_i \implies C_{i+1}$ according to $M$

\(^2\)In Lecture 7, we called such $C_1, \ldots, C_\ell$, an accepting valid sequence of configurations with respect to $M$ and $w$. 
Warmup: $\overline{A_{TM}} \leq_m A_{All_{TM}}$ (reproving that $A_{All_{TM}} \not\in \mathcal{RE}$)

Proof: We define computable $f$ such that the TM $B_{M,w} = f(\langle M, w \rangle)$ has the following properties:

1. if $M$ does not accept $w$, then $L(B_{M,w}) = \Sigma^*$
2. if $M$ does accept $w$, then $L(B_{M,w}) \neq \Sigma^*$

$B_{M,w}$ accepts all strings but the accepting configuration history of $M$ on $w$. (accepts all strings if $w \notin L(M)$)

**Algorithm 7 ($B_{M,w}$)**

Accepts input $h = C_1 \# C_2 \ldots \# C_\ell$, if one of the followings holds:

1. $C_1$ not the starting configuration of $M$ on $w$
2. $C_\ell$ is not an accepting configuration of $M$
3. $\exists i \in [\ell - 1]$ s.t. $C_i \Leftrightarrow C_{i+1}$ according to $M$

It is easy to see that $f$ is computable
Proof: We define computable $f$ such that PDA the $P_{M,w} = f(\langle M, w \rangle)$ has the following properties:

1. if $M$ does not accept $w$, then $L(P_{M,w}) = \Sigma^*$
2. if $M$ does accept $w$, then $L(P_{M,w}) \neq \Sigma^*$

$P_{M,w}$ accepts all strings but the accepting configuration history of $M$ on $w$.

Algorithm 8 ($P_{M,w}$)

Accepts input $h = C_1 \# C_2 \ldots \# C_\ell$, if one of the followings holds

1. $C_1$ not the starting configuration of $M$ on $w$
2. $C_\ell$ is not an accepting configuration of $M$
3. $\exists i \in [\ell - 1]$ s.t. $C_i \not\Rightarrow C_{i+1}$ according to $M$

The (only) hard part is checking $C_i \not\Rightarrow C_{i+1}$ (the PDA will “guess” the right $i$ if such exists)
Checking $C_i \leftrightarrow C_{i+1}$

Algorithm 9 (Checking $C_i \leftrightarrow C_{i+1}$)

1. Push $C_i$ onto the stack till #.
2. Scan $C_{i+1}$ and pop matching symbols of $C_i$.
   Check if $C_i$ and $C_{i+1}$ match everywhere, except around the head position, where difference dictated by transition function for $M$.

Problem

When $C_i$ is popped from stack, it is in reverse order.

But we only trying to identify (ignoring the local changes around head position) the language $x \# y$, with $x \neq y$.
This can be done a PDA (see Lecture 5), but next slide we give a simpler solution.
Checking $C_i \iff C_{i+1}$, take 2

- So far, we used a “straight” notion of accepting computation histories

  $\# \rightarrow \# \rightarrow \# \rightarrow \# \rightarrow \# \rightarrow \# \cdot \cdot \cdot \# \rightarrow \#$

- But why not employ an alternative notion of accepting computation history, one that will make the life of our PDA much easier?

  **A solution:** write the accepting computation history so that every other configuration is in reverse order.

        $\# \rightarrow \# \leftarrow \# \rightarrow \# \leftarrow \# \cdot \cdot \cdot \# \leftarrow \#$

- This resolves the difficulty in the proof.
Section 3

Linear Bounded Automaton
Linear Bounded Automaton – LBA

- A restricted form of TM, that on input $w$ uses space (at most) $|w|$.
- We want a commutable membership check:
  $$\delta(\omega, q) = (\cdot, \omega, L)$$ for any $q \in Q$.
- Size of input determines size of memory.
Why “linear”? 

Question 10

Why such machines called “linear”?

Answer: Using a tape alphabet larger than the input alphabet increases memory by a constant factor.
LBAs are powerful!

- The **deciders** we seen for the following languages are all LBAs.
  - $A_{\text{DFA}}$ (does a DFA accept a string?)
  - $A_{\text{CFG}}$ (is string in a CFG?)
  - $\text{EMPTY}_{\text{DFA}}$ (is a DFA trivial?)
  - $\text{EMPTY}_{\text{CFG}}$ (is a CFL empty?)

- Every **CFL** can be decided by an LBA.

- Not too easy to find a **natural, decidable language** that cannot be decided by an LBA.

- Almost all the algorithms in the data-structure and algorithm courses are decided by LBAs!
Acceptance for LBAs – $A_{\text{LBA}}$

$$A_{\text{LBA}} = \{ \langle M, w \rangle : M \text{ is an LBA } \land w \in \mathcal{L}(M) \}$$

**Question 11**
Is $A_{\text{LBA}}$ decidable?

**Theorem 12**
$A_{\text{LBA}}$ is decidable.

Proof’s idea: Similar to controlled executions.

- **Reject** if $M$ is not an LBA. (?)
- Emulate $M(w)$ for limited number of steps (number depends on $M$ and $|w|$), accepts if $M$ does.
Lemma 13

Let $M$ be a LBA with $q$ states, $g$ symbols in tape alphabet. Then on input of size $n$, $M$ has at most $qng^n$ distinct configurations.

Proof: A configuration involves:

- control state ($q$ possibilities)
- head position ($n$ possibilities)
- tape contents ($g^n$ possibilities)
Decider for $A_{\text{LBA}}$

**Algorithm 14**

On input $\langle M, w \rangle$.

1. **Reject** if $M$ is not an LBA.
2. Emulate $M(w)$ while maintaining a step counter.
3. Counter incremented by 1 per each simulated step (of $M$).
4. Keep emulating $M$ for $qng^n + 1$ steps, or until it halts (whichever comes first)
5. **Accept** if $M$ has halted and accepted; otherwise, **Reject**
Emptiness for LBAs – \( \text{EMPTY}_{\text{LBA}} \)

\[
\text{EMPTY}_{\text{LBA}} = \{ \langle M \rangle : M \text{ is an LBA} \land L(M) = \emptyset \}
\]

**Question 15**

Is \( \text{EMPTY}_{\text{LBA}} \) decidable?

**Theorem 16**

\( \text{EMPTY}_{\text{LBA}} \) is undecidable.

Proof’s idea: Show that \( A_{\text{TM}} \leq_m \text{EMPTY}_{\text{LBA}} \implies \text{EMPTY}_{\text{LBA}} \not\in \mathcal{R} \implies \text{EMPTY}_{\text{LBA}} \not\in \mathcal{R} \).

Proof uses computation histories, similar to proving that \( \overline{A_{\text{TM}}} \leq_m \text{All}_{\text{PDA}} \).

- Given a TM \( M \) and input \( w \), construct LBA \( B_{M,w} \) such that \( \langle M, w \rangle \in A_{\text{TM}} \) iff \( L(B_{M,w}) \) contains the accepting computation history for \( M \) on \( w \). (?)
- Hence, \( M \) accepts \( w \) iff \( L(B_{M,w}) \neq \emptyset \).
Section 4

RE-Completeness
**RE-Completeness**

**Question 17**
Is there a language \( L \) that is **hardest** in the class \( \mathcal{RE} \)?

**Answer:** Well, you have to **define** what you mean by “hardest language”...

**Definition 18 (\( \mathcal{RE} \)-complete)**

A language \( L_0 \subseteq \Sigma^* \) is called \( \mathcal{RE} \)-complete, if the following holds

- \( L_0 \in \mathcal{RE} \) (membership).
- for every \( L \in \mathcal{RE} \) we have \( L \leq_m L_0 \) (hardness).

- The second item means that \( \forall L \in \mathcal{RE} \), there is a mapping reduction \( f_L \) from \( L \) to \( L_0 \).
- The reduction \( f_L \) depends on \( L \) and will typically differ from one language to another.
An \( \mathcal{RE} \)-complete language

**Question 19**

Are there \( \mathcal{RE} \)-complete languages?

**Theorem 20**

\( A_{TM} \) is \( \mathcal{RE} \)-complete.

**Proof:**

- Clearly \( A_{TM} \in \mathcal{RE} \).
- Let \( L \in \mathcal{RE} \), and let \( M_L \) be a TM accepting it. Then \( f_L(w) = \langle M_L, w \rangle \) is a mapping reduction from \( L \) to \( A_{TM} \).
Other $\mathcal{RE}$-Complete problems

Question 21
Are there other $\mathcal{RE}$-complete languages?

Observations 22
Reductions are transitive:

$$A \leq_m B, \quad B \leq_m C \quad \Rightarrow \quad A \leq_m C$$

Theorem 23
Let $L$ be a language such

1. $L \in \mathcal{RE}$
2. $A_{TM} \leq_m L$

then $L$ is $\mathcal{RE}$-complete.

Hence, $H_{TM}$ and $H_{TM,\varepsilon}$ and ..., are all $\mathcal{RE}$-complete.
Between $\mathcal{R}$ and $\mathcal{RE}$-complete

- Are there languages in $\mathcal{RE} \setminus \mathcal{R}$ that are not $\mathcal{RE}$-complete?
- Yes, but not natural ones. See work of Emil Post.
Section 5

Computability, Summary
Summary

- Turing Machine - a universal computational model
- Language classes we considered: \( \mathcal{R} \), \( \mathcal{RE} \) and \( co-\mathcal{RE} \).
- Undecidable languages (not in \( \mathcal{R} \))
  - Acceptance/Halting problem
  - Any non-trivial property of a program (Rice Theorem)
  - Questions with respect to Grammars.
  - Many more exists ...
- Non-enumerable languages (not in \( \mathcal{RE} \))
  - Some non-trivial properties of a program
  - Much more exists ...

- What does this imply to verification of software and hardware? Well...