Computational Models — Lecture 10[1]

Handout Mode

Ronitt Rubinfeld and Iftach Haitner.

Tel Aviv University.

May 23/25, 2016

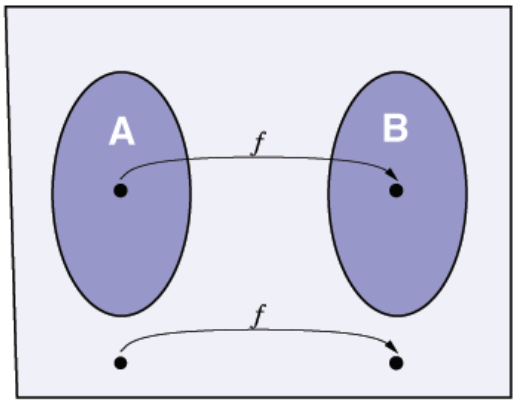
[1] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University. Also with modifications of Yishay Mansour.

Talk Outline

- Rice's theorem and freinds
- Controlled executions
- Configuartion histories
- $\mathcal{RE}$-Completeness

- Sipser's book, Chapter 5, Sections 5.1, 5.3

Mapping reductions (review)



$$x \in \mathcal{A} \iff f(x) \in \mathcal{B}$$

Converts questions about membership in $\mathcal{A}$ to membership in $\mathcal{B}$.

Theorem 1

If $\mathcal{A} \leq_m \mathcal{B}$ and $\mathcal{B} \in \mathcal{R}$, then $\mathcal{A} \in \mathcal{R}$.

Theorem 2

If $\mathcal{A} \leq_m \mathcal{B}$ and $\mathcal{B} \in \mathcal{RE}$, then $\mathcal{A} \in \mathcal{RE}$.

Corollary 3

If $\mathcal{A} \leq_m \mathcal{B}$ and $\mathcal{A} \notin \mathcal{R}$ [resp., $\mathcal{A} \notin \mathcal{RE}$], then $\mathcal{B} \notin \mathcal{R}$ [resp., $\mathcal{B} \notin \mathcal{RE}$] .

Section 1

Rice's Theorem

Non-trivial properties of $\mathcal{RE}$ languages

A few examples

- L is finite.
- L is infinite.
- L contains the empty string.
- L contains no prime number.
- L is co-finite.
- ...

All these are non-trivial properties of $\mathcal{RE}$ – for each of them there is $L_1, L_2 \in \mathcal{RE}$ such that L_1 satisfies the property but L_2 does not.

Question 4

Are there any trivial properties of $\mathcal{RE}$ languages?

Rice's Theorem

Theorem 5

For non-empty $\mathcal{C} \subsetneq \mathcal{RE}$, it holds that
$\mathsf{L}_{\mathcal{C}} = \{\langle M \rangle : \ M \text{ is a TM and } \mathsf{L}(M) \in \mathcal{C}\} \notin \mathcal{R}$.

Proof's idea: Reduction from H_{TM}.

Given M and w, we construct $M_w^{\mathcal{C}}$ such that:

- If M halts on w, then $\langle M_w^{\mathcal{C}} \rangle \in \mathsf{L}_{\mathcal{C}}$.
- If M does not halt on w, then $\langle M_w^{\mathcal{C}} \rangle \notin \mathsf{L}_{\mathcal{C}}$.

Proving Rice's Theorem

Assume $\emptyset \notin \mathcal{C}$ (Will take care of the other case later).

Fix $\mathcal{A} \in \mathcal{C}$ and let $M_{\mathcal{A}}$ be a TM accepting it (recall $\mathcal{C} \subseteq \mathcal{RE}$).

Algorithm 6 (M_w^C)

On input y:

1. Emulate $M(w)$.
2. Emulate $M_{\mathcal{A}}(y)$:

   Accept, if $M_{\mathcal{A}}$ accepts; reject, if $M_{\mathcal{A}}$ rejects.

Let $f(\langle M, w \rangle) := \langle M_w^C \rangle$

Claim 7

f is a mapping reduction from H_{TM} to $\mathrm{L}_{\mathcal{C}}$

Since f is clearly computable (?), it is left to show that

$$\langle M, w \rangle \in \mathrm{H}_{\mathrm{TM}} \iff f(\langle M, w \rangle) \in \mathrm{L}_{\mathcal{C}}$$

$\langle M, w \rangle \in \mathrm{H_{TM}} \iff f(\langle M, w \rangle) \in \mathrm{L}_{\mathcal{C}}$

Proof:

- If $\langle M, w \rangle \in \mathrm{H_{TM}}$, then $M_w^{\mathcal{C}}$ gets to Step 2, and emulates $M_{\mathrm{L}}(y)$.
  Hence $\mathrm{L}(M_w^{\mathcal{C}}) = \mathrm{L} \in \mathcal{C} \Longrightarrow M_w^{\mathcal{C}} \in \mathrm{L}_{\mathcal{C}}$.
- Otherwise, $M_w^{\mathcal{C}}$ never gets to Step 2.
  Hence $\mathrm{L}(M_w^{\mathcal{C}}) = \emptyset \notin \mathcal{C} \Longrightarrow M_w^{\mathcal{C}} \notin \mathrm{L}_{\mathcal{C}}$.
- Thus, $\langle M, w \rangle \in \mathrm{H_{TM}}$ iff $\langle M_w^{\mathcal{C}} \rangle \in \mathrm{L}_{\mathcal{C}}$.

♣

Completing the proof: The case $\emptyset \in \mathcal{C}$

Since $\mathcal{C} \subsetneq \mathcal{R}$ and $\mathcal{C}$ is non empty, it follows that $\emptyset \notin \overline{\mathcal{C}} \subsetneq \mathcal{RE}$, and $\overline{\mathcal{C}}$ is non-empty.

$\Longrightarrow$ (by the first part of the proof) $L_{\overline{\mathcal{C}}} \notin \mathcal{R}$

$\Longrightarrow$ $\overline{L_{\mathcal{C}}} \notin \mathcal{R}$ (?)

$\Longrightarrow$ $L_{\mathcal{C}} \notin \mathcal{R}$

Example: $\text{Primes}_{\text{TM}} \notin \mathcal{R}$

- $\text{Primes} = \{p \in \mathbb{N} : p \text{ is a prime}\}$
- $\text{Primes}_{\text{TM}} = \{\langle M \rangle : M \text{ is a TM and } L(M) = \text{Primes}\}$

Theorem 8

$\text{Primes} \in \mathcal{R}$

Proof: (?)

Theorem 9

$\text{Primes}_{\text{TM}} \notin \mathcal{R}$.

Proof: L_{Primes} is a non triviall subset of $\mathcal{RE}$.

Reflections on Rice theorem

- Rice's theorem can be used to show undecidability of properties like
  - *Does* $\mathsf{L}(M)$ *contain infinitely many primes*
  - *Does* $\mathsf{L}(M)$ *contain an arithmetic progression of length* 15
  - *Is* $\mathsf{L}(M)$ *empty*
- Decidability of properties related to the encoding itself cannot be inferred from Rice.
  - The question *does* $\langle M \rangle$ *has an even number of states* is decidable.
  - The question *does* M *reaches state* q_6 *on the empty input string* is undecidable, but this does not follow from Rice's theorem.
- Rice says nothing on membership in $\mathcal{RE}$
- **Rice's Theorem is a powerful tool, but use it with care!**

Section 2

Proving Non-Enumarability

Proving non-enumarability

Theorem 10

For non-empty $\mathcal{C} \subseteq (\mathcal{R} \setminus \{\Sigma^*\})$*, it holds tnat* $\mathsf{L}_{\mathcal{C}} = \{\langle M \rangle : M \text{ is a TM and } \mathsf{L}(M) \in \mathcal{C}\} \notin \mathcal{RE}$.

Proof: We will show that $\overline{\mathsf{A}_{\mathsf{TM}}} \leq_m \mathsf{L}_{\mathcal{C}}$.

Let D be a decider for some $\mathcal{A} \in \mathcal{C}$.
Define $f(x = \langle M, w \rangle)$ to return $B_{M,w}$ if x is well formatted, and D otherwise.

Definition 11 ($B_{M,w}$)

On input x

- Emulate $\mathsf{D}(x)$ and accept if D does.
- Emulate $M(w)$ and accept if M does.

- $\langle M, w \rangle \in \overline{\mathsf{A}_{\mathsf{TM}}} \implies \mathsf{L}(B_{M,w}) = \mathcal{A} \in \mathcal{C}$
- $\langle M, w \rangle \notin \overline{\mathsf{A}_{\mathsf{TM}}} \implies \mathsf{L}(B_{M,w}) = \Sigma^* \notin \mathcal{C}$. □

Corollary: $\mathsf{Primes}_{\mathsf{TM}} \notin \mathcal{RE}$.

Is $\mathsf{L}_\emptyset \in \mathcal{RE}$? Is $\mathsf{All}_{\mathsf{TM}} := \mathsf{L}_{\{\Sigma^*\}} \in \mathcal{RE}$?

Section 3

Controlled Executions

Bounded time and space

(in the following a TM stands for a determinstic one)

Definition 12

CET $:= \{\langle M, w, k\rangle : \ M$ accepts w within k steps$\}$.

Is CET $\in \mathcal{R}$?

Theorem 13

CET $\in \mathcal{R}$.

Proof?

Definition 14

CES $:= \{\langle M, w, k\rangle : \ M$ accepts w using k cells$\}$.

Theorem 15

CES *is decidable.*

Proving 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 16

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 $\text{All}_{\text{TM}} = \{\langle M \rangle : M \text{ is a TM and } \text{L}(M) = \Sigma^*\} \notin \mathcal{RE}$

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

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

Definition 17 ($B_{M,w}$)

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 $\implies$ $\text{L}(B_{M,w})$ is finite $\implies$ $\text{L}(B_{M,w}) \neq \Sigma^*$.
- $M(w)$ does not halt $\implies$ $B_{M,w}$ accepts all y's $\implies$ $\text{L}(B_{M,w}) = \Sigma^*$.

♣

Section 4

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

  $\overline{A_{\text{TM}}} \leq_m \text{All}_{\text{PDA}} = \{\langle P \rangle : P \text{ is a PDA and } L(P) = \Sigma^*\} \notin \mathcal{RE}$

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
  - $uaq_ibv \implies uq_jacv$, if $\delta(q_i, b) = (q_j, c, \mathrm{L})$
  - $uaq_ibv \implies uacq_jv$, if $\delta(q_i, b) = (q_j, c, \mathrm{R})$
  - Special cases: q_ibv and uaq_i
- Special type of configurations: starting, accepting, rejecting, halting
- $h = C_1 \# C_2 \ldots \# C_\ell$ is an accpeting configuration history of M on w, if
  1. $\forall i \in [\ell]$: C_i is a valid configuration of M
  2. C_1 is the starting configuration of M on w
  3. C_ℓ is an accepting configuration of M
  4. $\forall i \in [\ell]$: $C_i \implies C_{i+1}$ according to M

Warmup: $\overline{A_{TM}} \leq_m All_{TM}$ (reproving that $All_{TM} \notin \mathcal{RE}$)

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 configuation history* of M on w. (accepts all strings if $w \notin L(M)$)

Algorithm 18 ($B_{M,w}$)

Accepts input $h = C_1 \# C_2 \ldots \# C_\ell$, (only) if one of the conditions below holds

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

It is easy to see that f is computable

$\overline{A_{TM}} \leq_m All_{PDA}$

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 configuation history of M on w.

Algorithm 19 ($P_{M,w}$)

Accepts input $h = C_1 \# C_2 \ldots \# C_\ell$, (only) if one of the conditions below holds

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

Algorithm 20 (Checking $C_i \not\Rightarrow C_{i+1}$)

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

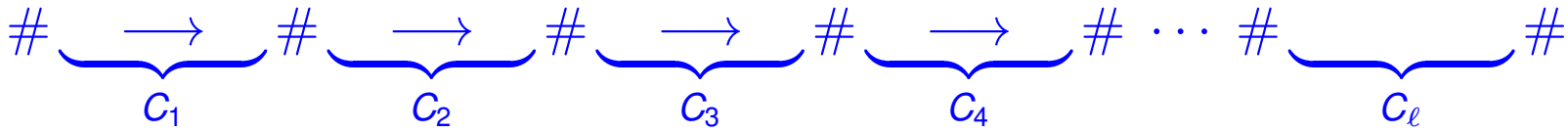
When C_i is popped from stack, it is in reverse order.

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.

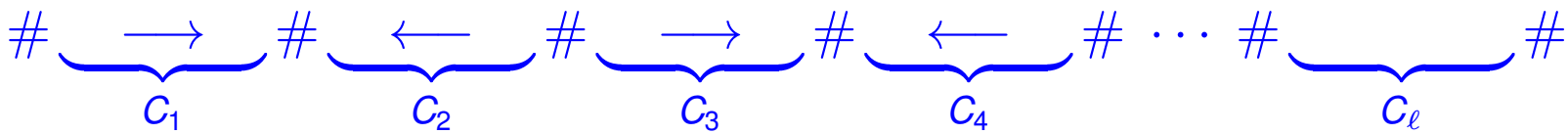
Checking $C_i \Rightarrow C_{i+1}$, take 2

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



- 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.



- This resolves the difficulty in the proof.

Section 5

RE-Completeness

$\mathcal{RE}$-Completeness

Question 21

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 22 ($\mathcal{RE}$-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.

A_{TM} is $\mathcal{RE}$-complete.

Question 23

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

Theorem 24

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} (why?).

♣

Other $\mathcal{RE}$-Ccomplete problems

Question 25

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

Observations 26

Reductions are transitive:

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

Theorem 27

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 6

Decidability, Summary

Summary

- Turing Machine - a universal computational model
- Language classes RE, co-RE, and R.
- Not Decidable
  - Acceptance/Halting problem
  - Any non-trivial property of a program (Rice Theorem)
  - Questions with respect to Grammars.
  - much more exists ...
- Not in RE (what does it mean?)

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