BU CS 332 – Theory of Computation

Lecture 16:

- Mapping Reducibility

Reading:

Sipser Ch 5.3

Mark Bun

March 25, 2020

Problems in language theory

A_{DFA} decidable	A_{CFG} decidable	A_{TM} undecidable
E_{DFA} decidable	E_{CFG} decidable	E_{TM} undecidable
EQ_{DFA} decidable	EQ_{CFG} ?	EQ_{TM} ?

Reductions

A reduction from problem A to problem B is an algorithm for problem A which uses an algorithm for problem B as a subroutine

If such a reduction exists, we say "A reduces to B"

Reminder: Using reductions to prove undecidability

If A reduces to B and A is undecidable, then B is also undecidable $\overline{E}_{TM}$ A_{TM}

Proof template:

1. Suppose to the contrary that B is decidable
2. Using B as a subroutine, construct an algorithm deciding A
3. But A is undecidable. Contradiction!

Equality Testing for TMs

$$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

Theorem: EQ_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for EQ_{TM}. We construct a decider for A_{TM} as follows:

On input $\langle M, w\rangle$: $A_{TM} = \{(M,w) \mid M \text{ is a TM which accepts on input } w\}$

1. Construct TMs M_1, M_2 as follows:

Idea: $L(M_1) = L(M_2) \iff M$ accepts input w

$L(M_1) = \begin{cases} \emptyset & \text{if } M \text{ does not accept } w \\ \Sigma^* & \text{if } M \text{ accepts } w \end{cases}$ $\quad L(M_2) = \Sigma^*$

2. Run R on input $\langle M_1, M_2\rangle$
3. If R accepts, accept. Otherwise, reject.

This is a reduction from A_{TM} to EQ_{TM}

Equality Testing for TMs

Claim: $L(M_1) = L(M_2) \iff M$ accepts w

$L(M) \neq \Sigma^* \iff M$ accepts w

$$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

Theorem: EQ_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for EQ_{TM}. We construct a decider for A_{TM} as follows:

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

$\Sigma = \{0, 1\}$ $\quad L(M_2) = \Sigma^*$

1. Construct TMs M_1, M_2 as follows:

M_1 = "On input x:
1) Ignore x
2) Run M on input w
3) If M accepts, accept
If M rejects, reject

M_2 = "On input x:
1) Ignore x, and accept"

$L(M_1) = \begin{cases} \emptyset & \text{if } M \text{ does not accept } w \\ \Sigma^* & \text{if } M \text{ accepts } w \end{cases}$

M_1 only recognizes one of these langs.

2. Run R on input $\langle M_1, M_2\rangle$
3. If R accepts, accept. Otherwise, reject.

This is a reduction from A_{TM} to EQ_{TM}

Problems in language theory

A_{DFA} decidable	A_{CFG} decidable	A_{TM} undecidable
E_{DFA} decidable	E_{CFG} decidable	E_{TM} undecidable
EQ_{DFA} decidable	EQ_{CFG} ?	EQ_{TM} undecidable

Context-free language testing for TMs

$$CFL_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is context free}\}$$

Theorem: CFL_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for CFL_{TM}. We construct a decider for A_{TM} as follows:

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

1. Construct a TM M' as follows:

Idea: $L(M')$ is context free $\Leftrightarrow$ M accepts w

$$L(M') = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ B & \text{if } M \text{ does not accept } w \end{cases}$$

B is your favorite non-context-free language

2. Run R on input $\langle M' \rangle$
3. If R accepts, accept. Otherwise, reject

This is a reduction from A_{TM} to CFL_{TM}

Context-free language testing for TMs

RE G PDA?

$$CFL_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is context free}\}$$

Theorem: CFL_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for CFL_{TM}. We construct a decider for A_{TM} as follows:

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

1. Construct a TM M' as follows:

   M' = "On input x,

   1. If $x \in \{0^n 1^n 2^n \mid n \geq 0\}$, accept
   2. Run TM M on input w
   3. If M accepts, accept."

2. Run R on input $\langle M' \rangle$
3. If R accepts, accept. Otherwise, reject

= B

M does not accept w:
M' Accept $x \iff x \in B$

M does accept w
M' Accept all $x \Rightarrow L(M') = \Sigma^*$

M accepts w $\iff$ L(M') is a CFL

Description of machine deciding A_{TM}

This is a reduction from A_{TM} to CFL_{TM}

Mapping Reductions

"Many-to-one reductions"

⚑⚑ Warning ⚑⚑

We know A_{TM} is recognizable

What's wrong with the following "proof"?

Bogus "Theorem": A_{TM} is not Turing-recognizable

Bogus "Proof": Suppose for contradiction that there exists a recognizer R for A_{TM}. We construct a recognizer for $\overline{A_{\mathrm{TM}}}$:

doesn't exist

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

1. Run R on input $\langle M, w \rangle$
2. If R accepts, reject. Otherwise, accept.

Problem: R loops forever on (M,w) ⇒ new machine also loops forever

This sure looks like a reduction from $\overline{A_{\mathrm{TM}}}$ to A_{TM}

Mapping Reductions: Motivation

1. How do we formalize the notion of a reduction?
2. How do we use reductions to show that languages are unrecognizable?
3. How do we protect ourselves from accidentally "proving" bogus theorems about recognizability?

Computable Functions

Definition:

A function $f: \Sigma^* \rightarrow \Sigma^*$ is computable if there is a TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape.

Example 1: $f(\langle x, y \rangle) = x + y$

$x, y \in \{0,1\}^*$

$\langle x, y \rangle = x \# y$

example encoding

Example 2: $f(\langle M, w \rangle) = \langle M' \rangle$ where M is a TM, w is a string, and M' is a TM that ignores its input and simulates running M on w

M' = "On input x
1) Ignore x
2) Run M on w
3) If M accepts, accept, else reject.."

Mapping Reductions

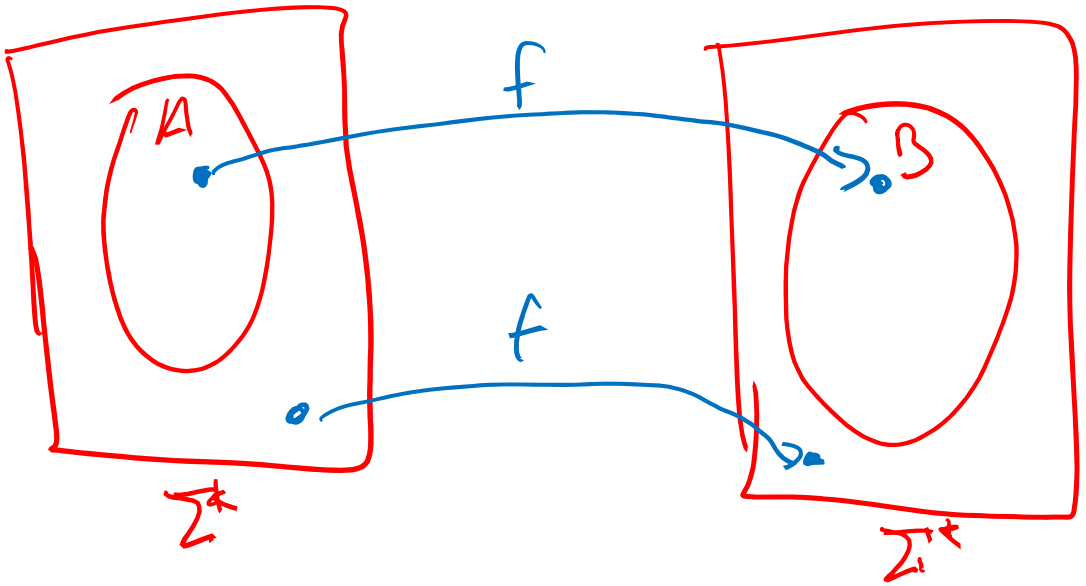
"A is no harder than B"

Definition:

Language A is mapping reducible to language B, written

$$A \leq_{\mathrm{m}} B$$

if there is a computable function $f: \Sigma^* \to \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \iff f(w) \in B$



f is also a mapping reduction from A to B, i.e. $\overline{A} \leq_m \overline{B}$

Decidability

Theorem: If $A \leq_m B$ and B is decidable, then A is also decidable

Proof: Let M be a decider for B and let $f: \Sigma^* \to \Sigma^*$ be a mapping reduction from A to B. Construct a decider for A as follows:

$w \in A \Rightarrow f(w) \in B$
$\Rightarrow M$ accepts $f(w)$
$\Rightarrow$ Accept

$w \notin A \Rightarrow f(w) \notin B$
$\Rightarrow M$ rejects $f(w)$
$\Rightarrow$ Reject

On input w:

1. Compute $f(w)$ $\in B$
2. Run M on input $f(w)$
3. If M accepts, accept. Otherwise, reject.

Undecidability

Theorem: If $A \leq_m B$ and B is decidable, then A is also decidable

Corollary: If $A \leq_m B$ and A is undecidable, then B is also undecidable

Eg. $A_{TM} \leq_m B \Rightarrow B$ undecidable

Old Proof: Equality Testing for TMs

$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem: EQ_{TM} is undecidable

Proof: Suppose for contradiction that there exists a decider R for EQ_{TM}. We construct a decider for A_{TM} as follows:

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

mapping reduction f

1. Construct TMs M_1, M_2 as follows:

   M_1 = "On input x,
   1. Ignore x
   2. Run M on input w
   3. If M accepts, accept.
      Otherwise, reject."

   M_2 = "On input x,
   1. Ignore x and accept"

2. Run R on input $\langle M_1, M_2\rangle$
3. If R accepts, accept. Otherwise, reject.

This is a reduction from A_{TM} to EQ_{TM}

New Proof: Equality Testing for TMs

$$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

Theorem: $A_{\text{TM}} \leq_{\text{m}} EQ_{\text{TM}}$ hence EQ_{TM} is undecidable

Proof: The following TM computes the reduction:

$f: A_{TM} \to EQ_{TM}$, $f: \overline{A_{TM}} \to \overline{EQ_{TM}}$

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

1. Construct TMs M_1, M_2 as follows:

   M_1 = "On input x,
   1. Ignore x
   2. Run M on input w
   3. If M accepts, accept.
      Otherwise, reject."

   M_2 = "On input x,
   1. Ignore x and accept"

2. Output $\langle M_1, M_2\rangle$

TM computing reduction f

Mapping Reductions: Recognizability

Theorem: If $A \leq_{\mathrm{m}} B$ and B is recognizable, then A is also recognizable

Proof: Let M be a recognizer for B and let $f: \Sigma^* \to \Sigma^*$ be a mapping reduction from A to B. Construct a recognizer for A as follows:

On input w:

1. Compute $f(w)$
2. Run M on input $f(w)$
3. If M accepts, accept. Otherwise, reject.

Unrecognizability

Theorem: If $A \leq_{\mathrm{m}} B$ and B is recognizable, then A is also recognizable

Corollary: If $A \leq_{\mathrm{m}} B$ and A is unrecognizable, then B is also unrecognizable

Corollary: If $\overline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} B$, then B is unrecognizable

Recognizability and A_{TM}

Theorem: A language L is recognizable if and only if $L \leq_m A_{TM}$

Proof. ($\Leftarrow$) Follows from theorem on slide 20 and fact that A_{TM} is recognizable

($\Rightarrow$) Suppose L is recognizable and let M be a TM recognizing L. We want to show $L \leq_m A_{TM}$.

Here is the mapping reduction: $f(w) = \langle M, w \rangle$

Why does this work?

1) If $w \in L$, then M accepts on input w, so $\langle M, w \rangle \in A_{TM}$

2) If $w \notin L$, then M does not accept on input w,

so $\langle M, w \rangle \notin A_{TM}$

Hence f is indeed a mapping reduction.

Takeaway: A_{TM} is the "hardest" recognizable language

Consequences of $A_{\mathrm{TM}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$

1. Since A_{TM} is undecidable, EQ_{TM} is also undecidable

2. $A_{\mathrm{TM}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$ implies $\overline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} \overline{EQ_{\mathrm{TM}}}$
   Since $\overline{A_{\mathrm{TM}}}$ is unrecognizable, $\overline{EQ_{\mathrm{TM}}}$ is unrecognizable

EQ_{TM} itself is also unrecognizable

$$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

Theorem: $\overline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$ hence EQ_{TM} is unrecognizable

Proof: The following TM computes the reduction:

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

1. Construct TMs M_1, M_2 as follows:

   M_1 = "On input x,
   1. Ignore x
   2. Run M on input w
   3. If M accepts, accept.
      Otherwise, reject."

   M_2 = "On input x, reject
   1. Ignore x and ~~accept~~"

2. Output $\langle M_1, M_2\rangle$

More on Reductions and Undecidability

Problems in language theory

A_{DFA} **decidable**	A_{CFG} **decidable**	A_{TM} **undecidable**
E_{DFA} **decidable**	E_{CFG} **decidable**	E_{TM} **undecidable**
EQ_{DFA} **decidable**	EQ_{CFG} **?**	EQ_{TM} **undecidable**

ALL_{CFG} is undecidable

$$ALL_{\text{CFG}} = \{\langle G \rangle \mid G \text{ is a CFG with terminal set } \Sigma \text{ and } L(G) = \Sigma^*\}$$

Theorem: $\overline{A_{\text{TM}}} \leq_{\text{m}} EQ_{\text{CFG}}$ hence EQ_{CFG} is undecidable

ALL_CFG ALL_CFG

Proof idea: "Computation history method"

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

1. Construct a CFG G such that:

   $L(G) = \Sigma^* \iff M$ does not accept w

   $L(G) = \{ w \mid w$ is not an "computation history" showing that M accepts $w \}$

   Accepting computation history: $C_1 \# C_2^R \# C_3 \# C_4^R \# \ldots \# C_{\ell-1}^R \# C_\ell$

   - Each C_i is a configuration of M
   - C_i yields C_{i+1} in M
   - C_1 = start configuration and C_ℓ is an accepting configuration

   Reverses are needed to ensure this can be generated by a PDA/CFG

2. Output $\langle G \rangle$

Problems in language theory

A_{DFA} decidable	A_{CFG} decidable	A_{TM} undecidable
E_{DFA} decidable	E_{CFG} decidable	E_{TM} undecidable
EQ_{DFA} decidable	EQ_{CFG} undecidable	EQ_{TM} undecidable

Proved by reduction from ALL_{CFG}
Exercise 5.1 in Sipser

Undecidable problems outside language theory

Post Correspondence Problem (PCP):

Domino: $\left[\frac{a}{ab}\right]$. Top and bottom are strings.

Input: Collection of dominos.

$$\left[\frac{aa}{aba}\right], \left[\frac{ab}{aba}\right], \left[\frac{ba}{aa}\right], \left[\frac{abab}{b}\right]$$

Match: List of some of the input dominos (repetitions allowed) where top = bottom

$$\left[\frac{ab}{aba}\right], \left[\frac{aa}{aba}\right], \left[\frac{ba}{aa}\right], \left[\frac{aa}{aba}\right], \left[\frac{abab}{b}\right]$$

Problem: Does a match exist? This is undecidable