COMP30026 Models of Computation

Lecture 21: Reducibility

Mak Nazecic-Andrlon and William Umboh

Semester 2, 2024

Some material from Michael Sipser's slides

Where are we?

Last time:
- A_{TM} is undecidable
- The diagonalization method
- $\overline{A_{\text{TM}}}$ is T-unrecognizable

Today: (Sipser §5.1, §5.3)
- The Reducibility Method for proving undecidability and T-unrecognizability.
- General reducibility
- Mapping reducibility

Why study reducibility?

Hard problems are everywhere:

- Checking ambiguity of CFGs
- Checking if a program will ever run into an infinite loop (Halting problem)
- Testing equivalence of programs
  - Example: A new Google intern refactored the entire codebase. Does it retain the same behavior/functionality?

Knowing when a problem you want to solve is undecidable can save you time!

The Reducibility Method

If we know that some problem (say A_{TM}) is undecidable,
we can use that to show other problems are undecidable.

Defn: $HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ halts on input } w\}$

Recall Theorem: $HALT_{\mathrm{TM}}$ is undecidable

Proof by contradiction, showing that A_{TM} is reducible to $HALT_{\mathrm{TM}}$:

Assume that $HALT_{\mathrm{TM}}$ is decidable and show that A_{TM} is decidable (false!).
Let TM R decide $HALT_{\mathrm{TM}}$.
Construct TM S deciding A_{TM}.

$S =$ "On input $\langle M, w\rangle$

1. Use R to test if M on w halts. If not, *reject*.
2. Simulate M on w until it halts (as guaranteed by R).
3. If M has accepted then *accept*.
   If M has rejected then *reject*.

TM S decides A_{TM}, a contradiction. Therefore $HALT_{\mathrm{TM}}$ is undecidable.

```
def S(⟨M,w⟩):
  if not R(⟨M,w⟩):
    reject
  else:
    return M(w)
```

Reducibility – Concept

If we have two languages (or problems) A and B, then
A is reducible to B means that we can use B to solve A.

Example 1: Measuring the area of a rectangle
is reducible to measuring the lengths of its sides.

Example 2: We showed that A_{NFA} is reducible to A_{DFA} .

If A is reducible to B then solving B gives a solution to A.
- then B is easy $\rightarrow$ A is easy.
- then A is hard $\rightarrow$ B is hard.

this is the form we will focus on today

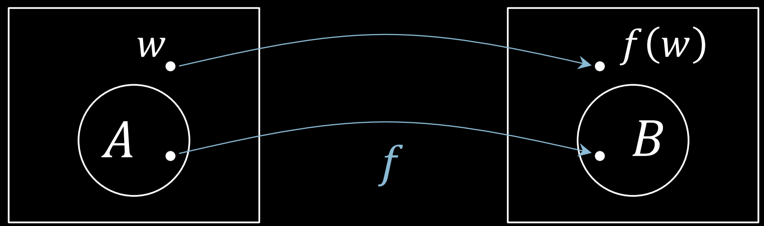
Mapping Reducibility

Defn: Function $f : \Sigma^* \rightarrow \Sigma^*$ is computable if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Examples:

- String concatenation $f(\langle x, y \rangle) = xy$.
- If L is decidable, then the following function is computable: $f(x) = 1$ if x in L and 0 otherwise
- DFA/NFA manipulation procedures we've seen so far

Defn: A is mapping-reducible to B ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.

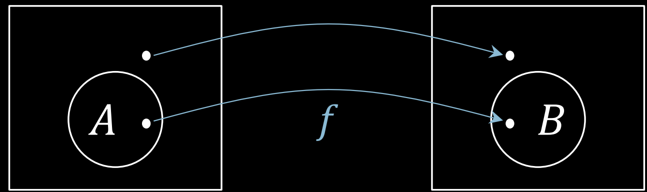


Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:



$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

```
def S(w):
    return R(f(w))
```

Examples for decidability:

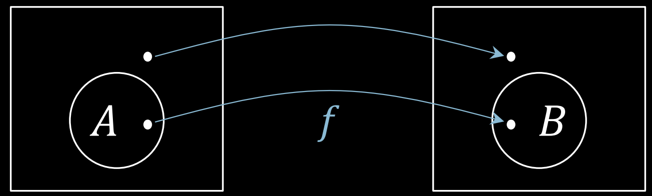
- $A_{\mathrm{NFA}} \leq_{\mathrm{m}} A_{\mathrm{DFA}}$
- $A_{\mathrm{REX}} \leq_{\mathrm{m}} A_{\mathrm{DFA}}$
- $A_{\mathrm{PDA}} \leq_{\mathrm{m}} A_{\mathrm{CFG}}$
- $EQ_{\mathrm{DFA}} \leq_{\mathrm{m}} E_{\mathrm{DFA}}$

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:



$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Corollary: If $A \leq_{\mathrm{m}} B$ and A is undecidable then so is B

Theorem: If $A \leq_{\mathrm{m}} B$ and B is T-recognizable then so is A

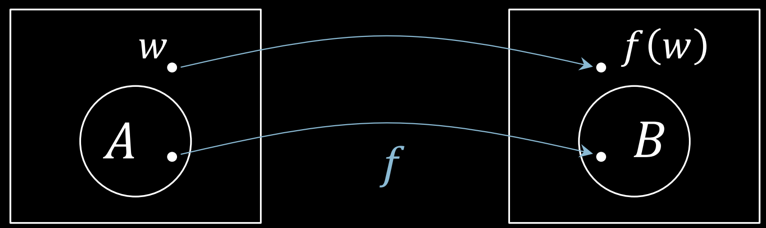
Proof: Same as above.

Corollary: If $A \leq_{\mathrm{m}} B$ and A is T-unrecognizable then so is B

Mapping Reducibility

Defn: Function $f: \Sigma^* \to \Sigma^*$ is <u>computable</u> if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Defn: <u>A is mapping-reducible to B</u> ($A \le_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



Check-in 21.1

Suppose $A \le_{\mathrm{m}} B$.

What can we conclude?

Check all that apply.

(a) $B \le_{\mathrm{m}} A$

(b) $\overline{A} \le_{\mathrm{m}} \overline{B}$

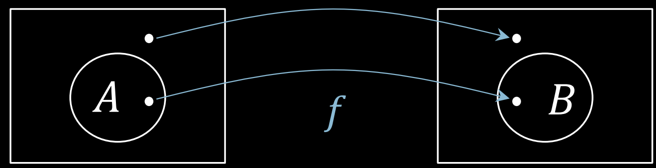
(c) $\overline{B} \le_{m} \overline{A}$

(d) None of the above

Mapping vs General Reducibility

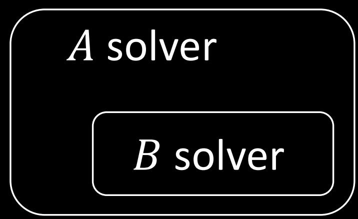
Mapping Reducibility of A to B: Translate A-questions to B-questions.

- A special type of reducibility
- Useful to prove T-unrecognizability



(General) Reducibility of A to B: Use B solver to solve A.

- May be conceptually simpler
- Useful to prove undecidability



Noteworthy difference:

- A is reducible to $\overline{A}$
- BUT A may not be mapping reducible to $\overline{A}$.

  For example $\overline{A_{\mathrm{TM}}} \not\leq_{\mathrm{m}} A_{\mathrm{TM}}$

Check-in 21.2

We showed that if $A \leq_{\mathrm{m}} B$ and B is T-recognizable then so is A.

Is the same true if we use general reducibility instead of mapping reducibility?

(a) Yes

(b) No

Example of general reduction that is not mapping reduction: Halting problem

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is A_{TM})
- Template: Assume TM R decides B.
  Construct TM S deciding A. Contradiction.
- This is called a general reduction.

To prove B is T-unrecognizable:

- Show T-unrecognizable A is mapping reducible to B. (often A is $\overline{A_{\mathrm{TM}}}$)
- Template: give <u>computable</u> reduction function f.
- This is called a mapping reduction
- This also works to prove A is undecidable and is useful for undecidability proofs

Note: mapping reduction is special case of general reduction

E_{TM} is undecidable

Let $E_{\mathrm{TM}} = \{\langle M\rangle |\ M$ is a TM and $L(M) = \emptyset\ \}$

Theorem: E_{TM} is undecidable

Proof by contradiction. Show that A_{TM} is reducible to E_{TM}.

Assume that E_{TM} is decidable and show that A_{TM} is decidable (false!).

Let TM R decide E_{TM}.

Construct TM S deciding A_{TM}.

$S =$ "On input $\langle M, w\rangle$

1. Transform M to new TM $M_w =$ "On input x
    1. If $x \neq w$, *reject*.
    2. else run M on w
    3. *Accept* if M accepts."
2. Use R to test whether $L(M_w) = \emptyset$
3. If YES [so M rejects w] then *reject*.
   If NO [so M accepts w] then *accept*.

```
def S(M,w):
    def M_w(x):
        if x ≠ w:
            Reject
        Else:
            return M(w)
    return not(R(M_w))
```

M_w works like M except that it always rejects strings x where $x \neq w$.

So $L(M_w) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ rejects } w \end{cases}$

E_{TM} is T-unrecognizable

Recall $E_{\mathrm{TM}} = \{\langle M\rangle \mid M$ is a TM and $L(M) = \emptyset\}$

Theorem: E_{TM} is T-unrecognizable

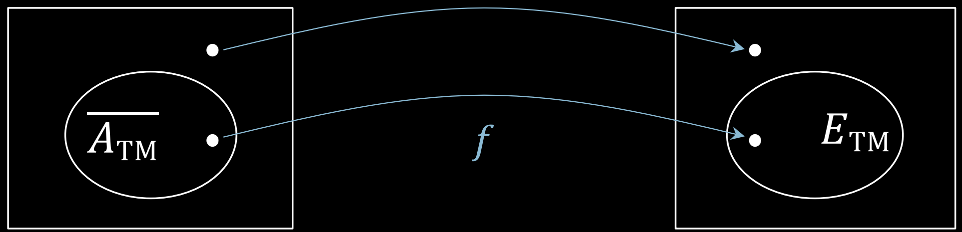
Proof: Show $\overline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} E_{\mathrm{TM}}$

Reduction function: $f(\langle M, w\rangle) = \langle M_w\rangle$

Recall TM M_w = "On input x
1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."

Explanation: $\langle M, w\rangle \in \overline{A_{\mathrm{TM}}}$ iff $\langle M_w\rangle \in E_{\mathrm{TM}}$

M rejects w iff $L(\langle M_w\rangle) = \emptyset$



EQ_{TM} and $\overline{EQ_{\mathrm{TM}}}$ are T-unrecognizable

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

Theorem: Both EQ_{TM} and $\overline{EQ_{\mathrm{TM}}}$ are T-unrecognizable

Proof: (1) $\overline{A_{\mathrm{TM}}} \le_{\mathrm{m}} EQ_{\mathrm{TM}}$

(2) $\overline{A_{\mathrm{TM}}} \le_{\mathrm{m}} \overline{EQ_{\mathrm{TM}}}$

For any w let $T_w =$ "On input x

1. Ignore x.
2. Simulate M on w."

T_w acts on all inputs the way M acts on w.

```
def T_w(x):
   return M(w)
```

(1) Here we give f which maps $\overline{A_{\mathrm{TM}}}$ problems (of the form $\langle M, w\rangle$) to EQ_{TM} problems (of the form $\langle T_1, T_2\rangle$).

$f(\langle M, w\rangle) = \langle T_w, T_{\text{reject}}\rangle$ $\qquad T_{\text{reject}}$ is a TM that always rejects.

(2) Similarly $f(\langle M, w\rangle) = \langle T_w, T_{\text{accept}}\rangle$ $\qquad T_{\text{accept}}$ always accepts.

Reducibility terminology

Why do we use the term "reduce"?

When we reduce A to B, we show how to solve A by using B
and conclude that A is no harder than B. (suggests the $\leq_{\mathrm{m}}$ notation)

Possibility 1: We bring A's difficulty down to B's difficulty. (A no harder than B)

Possibility 2: We bring B's difficulty up to A's difficulty. (B no easier than A)

Defn. When $A \leq_{\mathrm{m}} B$ and $B \leq_{\mathrm{m}} A$, then $A =_{\mathrm{m}} B$. The two problems are equivalent, i.e. A is recognizable/decidable iff B is recognizable/decidable.

Defn. When $A \leq_{\mathrm{m}} B$ and $B \leq_{\mathrm{m}} C$, then $A \leq_{\mathrm{m}} C$.

For example, if we showed $A_{TM} \leq_{\mathrm{m}} B$, then to show C is undecidable, we can also show $B \leq_{\mathrm{m}} C$. Moral: We don't always have to reduce from A_{TM}!

Quick review of today

- The Reducibility Method for proving undecidability and T-unrecognizability.
- General reducibility
- Mapping reducibility