UMB CS 420
Mapping Reducibility
Thursday, November 17, 2022
Announcements

• Current hw: HW 9
  • Due Mon 11/21 11:59pm EST

• Next hw: HW 10
  • Out: Tue 11/22
  • Due: Mon 12/5
  • 2 weeks due to Thanksgiving break
Last time: Undecidable ...

- $\text{REGULAR}_{TM} = \{<M> \mid M\text{ is a TM and } L(M)\text{ is a regular language}\}$

- $\text{CONTEXTFREE}_{TM} = \{<M> \mid M\text{ is a TM and } L(M)\text{ is a CFL}\}$

- $\text{DECIDABLE}_{TM} = \{<M> \mid M\text{ is a TM and } L(M)\text{ is a decidable language}\}$

- $\text{FINITE}_{TM} = \{<M> \mid M\text{ is a TM and } L(M)\text{ is a finite language}\}$

- ...

- $\text{ANYTHING}_{TM} = \{<M> \mid M\text{ is a TM and “… anything …” about } L(M)\}$
Rice’s Theorem: $\text{ANYTHING}_{\text{TM}}$ is Undecidable

$\text{ANYTHING}_{\text{TM}} = \{<M> \mid M \text{ is a TM and } \text{... anything ... about } L(M)\}$

• “... Anything ...”, more precisely:
  • For any $M_1, M_2$, if $L(M_1) = L(M_2)$ ...
  • ... then $M_1 \in \text{ANYTHING}_{\text{TM}} \Leftrightarrow M_2 \in \text{ANYTHING}_{\text{TM}}$

• Also, “... Anything ...” must be “non-trivial”:
  • $\text{ANYTHING}_{\text{TM}} \neq \{\}$
  • $\text{ANYTHING}_{\text{TM}} \neq \text{set of all TMs}$
Rice’s Theorem: $\text{ANYTHING}_{\text{TM}}$ is Undecidable

$\text{ANYTHING}_{\text{TM}} = \{<M> | M \text{ is a TM and ... anything ... about } L(M)\}$

Proof by contradiction

• Assume some language satisfying $\text{ANYTHING}_{\text{TM}}$ has a decider $R$.
  • Since $\text{ANYTHING}_{\text{TM}}$ is non-trivial, then there exists $M_{\text{ANY}} \in \text{ANYTHING}_{\text{TM}}$
  • Where $R$ accepts $M_{\text{ANY}}$

• Use $R$ to create decider for $A_{\text{TM}}$:

  On input $<M, w>$:
  
  • Create $M_w$:
    - $M_w = \text{on input } x$:
      - Run $M$ on $w$
      - If $M$ rejects $w$: reject $x$
      - If $M$ accepts $w$: Run $M_{\text{ANY}}$ on $x$ and accept if it accepts, else reject
    
    If $M$ accepts $w$: $M_w = M_{\text{ANY}}$
    If $M$ doesn’t accept $w$: $M_w$ accepts nothing

  • Run $R$ on $M_w$
    • If it accepts, then $M_w = M_{\text{ANY}}$, so $M$ accepts $w$, so accept
    • Else reject

These two cases must be different, (so $R$ can distinguish when $M$ accepts $w$)

Wait! What if the TM that accepts nothing is in $\text{ANYTHING}_{\text{TM}}$!

Proof still works! Just use the complement of $\text{ANYTHING}_{\text{TM}}$ instead!
Prove that the following is undecidable:

\{<M> | M is a TM that installs malware\}
Rice’s Theorem Implication

\{<M> \mid M \text{ is a TM that installs malware}\}

Undecidable!
(by Rice’s Theorem)
Flashback: “Reduced”

Thm: $\text{HALT}_{\text{TM}}$ is undecidable

Proof, by contradiction:

• Assume: $\text{HALT}_{\text{TM}}$ has decider $R$; use it to create $A_{\text{TM}}$ decider:

  $S = \text{“On input } \langle M, w \rangle \text{, an encoding of a TM } M \text{ and a string } w:\n  1. \text{ Run TM } R \text{ on input } \langle M, w \rangle.\n  2. \text{ If } R \text{ rejects, reject.}\n  3. \text{ If } R \text{ accepts, simulate } M \text{ on } w \text{ until it halts.}\n  4. \text{ If } M \text{ has accepted, accept; if } M \text{ has rejected, reject.}$$

  Use $R$ to first check if $M$ will loop on $w$

  Then run $M$ on $w$ knowing it won’t loop

• Contradiction: $A_{\text{TM}}$ is undecidable and has no decider!

Let’s formalize this conversion, i.e., mapping reducibility
Flashback: $A_{\text{NFA}}$ is a decidable language

$A_{\text{NFA}} = \{ \langle B, w \rangle \mid B \text{ is an NFA that accepts input string } w \}$

Decider for $A_{\text{NFA}}$:

$N = \text{“On input } \langle B, w \rangle, \text{ where } B \text{ is an NFA and } w \text{ is a string:} \quad$

1. Convert NFA $B$ to an equivalent DFA $C$, using the procedure $\text{NFA} \rightarrow \text{DFA}$
2. Run TM $M$ on input $\langle C, w \rangle$.
3. If $M$ accepts, accept; otherwise, reject.”

We said this $\text{NFA} \rightarrow \text{DFA}$ algorithm is a decider TM, but it doesn’t accept/reject?

More generally, our analogy has been: “programs = TMs”, but programs do more than accept/reject?
Definition: Computable Functions

A function \( f : \Sigma^* \rightarrow \Sigma^* \) is a **computable function** if some Turing machine \( M \), on every input \( w \), halts with just \( f(w) \) on its tape.

- **A computable function** is represented with a TM that, instead of accept/reject, “outputs” its final tape contents.

- **Example 1**: All arithmetic operations

- **Example 2**: Converting between machines, like DFA→NFA
  - E.g., adding states, changing transitions, wrapping TM in TM, etc.
**Definition:** Mapping Reducibility

Language $A$ is **mapping reducible** to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every $w$,

$$w \in A \iff f(w) \in B.$$  

“The function $f$ is called the reduction from $A$ to $B$.”

A function $f : \Sigma^* \to \Sigma^*$ is a **computable function** if some Turing machine $M$, on every input $w$, halts with just $f(w)$ on its tape.
Flashback: Equivalence of Contrapositive

“If $X$ then $Y$” is equivalent to ... ?

- “If $Y$ then $X$” (converse)
  - No!

- “If not $X$ then not $Y$” (inverse)
  - No!

✓ “If not $Y$ then not $X$” (contrapositive)
  - Yes!
Definition: Mapping Reducibility

Language $A$ is **mapping reducible** to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

$$w \in A \iff f(w) \in B.$$  

"if and only if"

The function $f$ is called the **reduction** from $A$ to $B$.

"forward" direction ($\Rightarrow$): if $w \in A$ then $f(w) \in B$

"reverse" direction ($\Leftarrow$): if $f(w) \in B$ then $w \in A$

Equivalent (contrapositive): if $w \notin A$ then $f(w) \notin B"
Proving Mapping Reducibility: 2 Steps

Language $A$ is **mapping reducible** to language $B$, written $A \leq_{m} B$, if there is a *computable function* $f: \Sigma^{*} \rightarrow \Sigma^{*}$, where for every $w$,

$$w \in A \iff f(w) \in B.$$ 

The function $f$ is called the **reduction** from $A$ to $B$.

**Step 1:**
Show there is computable function $f$... by creating a TM

**Step 2:**
Prove the iff is true

---

**Step 2a:** “forward” direction ($\Rightarrow$): if $w \in A$ then $f(w) \in B$

**Step 2b:** “reverse” direction ($\Leftarrow$): if $f(w) \in B$ then $w \in A$

**Step 2b, alternate (contrapositive):** if $w \notin A$ then $f(w) \notin B$

---

A function $f: \Sigma^{*} \rightarrow \Sigma^{*}$ is a **computable function** if some Turing machine $M$, on every input $w$, halts with just $f(w)$ on its tape.
Thm: $A_{TM}$ is mapping reducible to $HALT_{TM}$

To show: $A_{TM} \leq_m HALT_{TM}$

Step 1: create computable fn $f$: $<M, w> \rightarrow <M', w>$ where:

Step 2: show $<M, w> \in A_{TM}$ if and only if $<M', w'> \in HALT_{TM}$

The following machine $F$ computes a reduction $f$.

$F =$ “On input $<M, w>$:
1. Construct the following machine $M'$.

$M'$ = “On input $x$:
1. Run $M$ on $x$.
2. If $M$ accepts, accept.
3. If $M$ rejects, enter a loop.”

2. Output $<M', w>$.”

$M'$ is like $M$, except it always loops when it doesn’t accept.

Output new $M'$

Converting $M$ to $M'$

Language $A$ is mapping reducible to language $B$, written $A \leq_m B$, if there is a computable function $f$: $\Sigma^* \rightarrow \Sigma^*$, where for every $w$:

$w \in A \iff f(w) \in B$.

The function $f$ is called the reduction from $A$ to $B$.

A function $f$: $\Sigma^* \rightarrow \Sigma^*$ is a computable function if some Turing machine $M$, on every input $w$, halts with just $f(w)$ on its tape.
⇒ If $M$ accepts $w$, then $M'$ halts on $w$
  - $M'$ accepts (and thus halts) if $M$ accepts

⇐ If $M'$ halts on $w$, then $M$ accepts $w$

⇐ (Alternatively) If $M$ doesn’t accept $w$, then $M'$ doesn’t halt on $w$ (contrapositive)
  - Two possibilities for non-acceptance:
    1. $M$ loops: $M'$ loops and doesn’t halt
    2. $M$ rejects: $M'$ loops and doesn’t halt

The following machine $F$ computes a reduction $f$.

$F = “On$ input $\langle M, w \rangle$:
  1. Construct the following machine $M'$.
    $M' = “On$ input $x$:
      1. Run $M$ on $x$.
      2. If $M$ accepts, accept.
      3. If $M$ rejects, enter a loop.”
  2. Output $\langle M', w \rangle$.”

Step 2: $M$ accepts $w$ if and only if $M'$ halts on $w$
Uses of Mapping Reducibility

• To prove Decidability

• To prove Undecidability
Thm: If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

**PROOF** We let $M$ be the decider for $B$ and $f$ be the reduction from $A$ to $B$. We describe a decider $N$ for $A$ as follows.

$N = \text{“On input } w:\n1. \text{ Compute } f(w).\n2. \text{ Run } M \text{ on input } f(w) \text{ and output whatever } M \text{ outputs.”} \]

Language $A$ is *mapping reducible* to language $B$, written $A \leq_m B$, if there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

\[ w \in A \iff f(w) \in B. \]

The function $f$ is called the *reduction* from $A$ to $B$. 
Coro: If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.

- **Proof** by contradiction.

- **Assume** $B$ is decidable.

- **Then $A$ is decidable** (by the previous thm).

- **Contradiction:** we already said $A$ is undecidable
Summary: Showing Mapping Reducibility

Language $A$ is **mapping reducible** to language $B$, written $A \leq_m B$, if there is a **computable function** $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

$$w \in A \iff f(w) \in B.$$  

The function $f$ is called the **reduction** from $A$ to $B$.

**Step 1:** Show there is computable fn $f \ldots$ by creating a TM

**Step 2:** Prove the iff is true

**Step 2a:** “forward” direction ($\Rightarrow$): if $w \in A$ then $f(w) \in B$

**Step 2b:** “reverse” direction ($\Leftarrow$): if $f(w) \in B$ then $w \in A$

**Step 2b, alternate (contrapositive):** if $w \notin A$ then $f(w) \notin B$

A function $f : \Sigma^* \rightarrow \Sigma^*$ is a **computable function** if some Turing machine $M$, on every input $w$, halts with just $f(w)$ on its tape.
Summary: Using Mapping Reducibility

To prove decidability...

- If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

To prove undecidability...

- If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.

Be careful with the direction of the reduction!
Alternate Proof: The Halting Problem

\( \text{HALT}_{TM} \) is undecidable

- If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable.

- Must be known

- \( A_{TM} \leq_m \text{HALT}_{TM} \)

- Since \( A_{TM} \) is undecidable,
- \( \ldots \) and we showed mapping reducibility from \( A_{TM} \) to \( \text{HALT}_{TM} \),
- then \( \text{HALT}_{TM} \) is undecidable.
Flashback: \( EQ_{TM} \) is undecidable

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

Proof by contradiction:

- **Assume** \( EQ_{TM} \) has decider \( R \); use it to create \( E_{TM} \) decider:
  \[
  E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}
  \]

\[
S = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:}
\]

1. Run \( R \) on input \( \langle M, M_1 \rangle \), where \( M_1 \) is a TM that rejects all inputs.
2. If \( R \) accepts, *accept*; if \( R \) rejects, *reject.*
Alternate Proof: \( EQ_{TM} \) is undecidable

\[ EQ_{TM} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]

Show mapping reducibility: \( E_{TM} \leq_m EQ_{TM} \)

**Step 1:** create computable fn \( f: \langle M \rangle \rightarrow \langle M_1, M_2 \rangle \), computed by \( S \)

\[ S = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:} \]

1. **Construct:** \( \langle M, M_1 \rangle \), where \( M_1 \) is a TM that rejects all inputs.
2. **Output:** \( \langle M, M_1 \rangle \)

**Step 2:** show iff requirements of mapping reducibility (exercise)

And use theorem ...

If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable.
Flashback: \(E_{TM}\) is undecidable

\[E_{TM} = \{\langle M \rangle | M \text{ is a TM and } L(M) = \emptyset\}\]

Proof, by contradiction:

- Assume \(E_{TM}\) has *decider* \(R\); use it to create \(A_{TM}\) *decider*:

  \[S = \text{"On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\]

  1. Use the description of \(M\) and \(w\) to construct the TM \(M_1\):
      \[M_1 = \text{"On input } x:\]
      1. If \(x \neq w\), reject.
      2. If \(x = w\), run \(M\) on input \(w\) and *accept* if \(M\) does."

  2. Run \(R\) on input \(\langle M_1 \rangle\).

  3. If \(R\) accepts, *reject*; if \(R\) rejects, *accept*.

- So this only reduces \(A_{TM}\) to \(E_{TM}\)

  If \(M\) accepts \(w\), \(M_1\) not in \(E_{TM}\)!
Alternate Proof: $E_{TM}$ is undecidable

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

Show mapping reducibility??: $A_{TM} \leq_m E_{TM}$

Step 1: create computable fn $f$: $\langle M, w \rangle \rightarrow \langle M' \rangle$, computed by $S$

$S =$ “On input $\langle M, w \rangle$, an encoding of a TM $M$ and a string $w$:
1. Use the description of $M$ and $w$ to construct the TM $M_1$

    $M_1 =$ “On input $x$:
    1. If $x \neq w$, reject.
    2. If $x = w$, run $M$ on input $w$ and accept if $M$ does.”

2. Output: $\langle M_1 \rangle$.
3. If $R$ accepts, reject; if $R$ rejects, accept.”

- So this only reduces $A_{TM}$ to $\overline{E_{TM}}$
- It’s good enough! Still proves $E_{TM}$ is undecidable
  - If ... undecidable langs are closed under complement

Step 2: show iff requirements of mapping reducibility (exercise)
Undecidable Langs Closed under Complement

Proof by contradiction

• **Assume** some lang $L$ is undecidable and $\overline{L}$ is decidable ...
  • Then $\overline{L}$ has a decider

• **... then** we can create decider for $L$ from decider for $\overline{L}$ ...
  • Because decidable languages are closed under complement (hw10?)!

Contradiction!
Next Time: Turing Unrecognizable?

Is there anything out here?

Where do these undecidable languages go?

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \} \]

\[ EQ_{CFG} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \]

\[ EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]
Check-in Quiz 11/17

On gradescope