

Examples of reductions

- Recall the language $\text{HALT} = \{\langle M, w \rangle \mid M \text{ halts on input } w\}$
- Last class we proved that HALT is undecidable. The idea was to use a proof by contradiction: assuming there is a TM M_{HALT} that decides HALT , we could use this TM to construct a Turing machine that decides A_{TM} . This contradicts the fact that A_{TM} is not decidable, so HALT must not be decidable.
- Now let's prove the same thing using reductions.
- **Theorem:** HALT is undecidable.
- Proof: Show $A_{\text{TM}} \leq_m \text{HALT}$. Define f as follows: $f(\langle M, w \rangle) = \langle M', w' \rangle$, where $w' = w$ and M' = "on input x ..."
 1. Run M on input x
 2. If M accepts, accept (i.e. halt)
 3. If M rejects, go into an infinite loop

Then,

$$\begin{aligned}
 \langle M, w \rangle \in A_{\text{TM}} &\implies M \text{ accepts } w \\
 &\implies M' \text{ accepts } w' = w \\
 &\implies M' \text{ halts on input } w' \\
 &\implies \langle M', w' \rangle \in \text{HALT} \\
 \langle M, w \rangle \notin A_{\text{TM}} &\implies M \text{ doesn't accept } w \\
 &\implies M' \text{ doesn't halt on input } w' = w \\
 &\implies \langle M', w' \rangle \notin \text{HALT}
 \end{aligned}$$

So $\langle M, w \rangle \in A_{\text{TM}} \iff f(\langle M, w \rangle) \in \text{HALT}$. Is f computable? We will invoke the Church-Turing thesis to say "yes"; but consider that to obtain M' from M we only need to modify M 's transition function so that instead of going to q_{reject} it goes to some state q_{loop} , and add a few new transitions so that once it is in q_{loop} it never halts.

- **Definition:** $\text{EMPTY} = \{\langle M \rangle \mid L(M) = \emptyset\}$.
- **Theorem:** $\overline{\text{EMPTY}}$ is undecidable.
- Proof: Show $A_{\text{TM}} \leq_m \overline{\text{EMPTY}}$. Define $f(\langle M, w \rangle) = \langle M' \rangle$, where M' = "on input x ..."
 1. Run M on input w
 2. If M accepts then accept
 3. If M rejects then reject

Then,

$$\begin{aligned}
 \langle M, w \rangle \in A_{\text{TM}} &\implies M \text{ accepts } w \\
 &\implies L(M') = \Sigma^* \\
 &\implies L(M') \neq \emptyset \\
 &\implies \langle M' \rangle \in \overline{\text{EMPTY}} \\
 \langle M, w \rangle \notin A_{\text{TM}} &\implies M \text{ doesn't accept } w \\
 &\implies L(M') = \emptyset \\
 &\implies \langle M' \rangle \notin \overline{\text{EMPTY}}
 \end{aligned}$$

- We proved several lectures ago that if L and \overline{L} are both recognizable, then they are both decidable.
- **Corollary:** If L is recognizable but not decidable, then \overline{L} is not recognizable.
- Proof: Suppose L is recognizable but not decidable, and for contradiction assume that \overline{L} is recognizable. Then L and \overline{L} are both recognizable, so L is decidable, contradicting the assumption that L is not decidable.
- **Theorem:** $\overline{\text{EMPTY}}$ is not recognizable.
- Proof: By the corollary above, we just need to prove that $\overline{\text{EMPTY}}$ is recognizable. Then since $\overline{\text{EMPTY}}$ is recognizable but not decidable, its complement EMPTY is not recognizable. Define $M_{\overline{\text{EMPTY}}}$ = “on input $\langle M \rangle$...
 1. for $t \leftarrow 1, 2, 3, \dots, \infty$ do
 2. for $j \leftarrow 1, 2, \dots, t$ do
 3. Run M on s_j for t steps
 4. If M accepts then accept

where s_1, s_2, s_3, \dots is an enumeration of Σ^* . If $L(M) \neq \emptyset$ then there is a string which it accepts, and $M_{\overline{\text{EMPTY}}}$ will eventually run M on one such string for sufficiently many steps that M accepts. So $L(M_{\overline{\text{EMPTY}}}) = \text{EMPTY}$.

- **Definition:** $\text{FINITE} = \{\langle M \rangle \mid L(M) \text{ is finite}\}$.
- **Theorem:** FINITE is unrecognizable.
- Proof: Show $\overline{A_{\text{TM}}} \leq_m \text{FINITE}$. Define $f(\langle M, w \rangle) = \langle M' \rangle$, where M' = “on input x ...
 1. Run M on input w
 2. If M accepts then accept

3. If M rejects then reject

Then,

$$\begin{aligned} \langle M, w \rangle \in \overline{A_{TM}} &\implies M \text{ doesn't accept } w \\ &\implies L(M') = \emptyset \\ &\implies L(M') \text{ is finite} \\ &\implies \langle M' \rangle \in \text{FINITE} \\ \langle M, w \rangle \notin \overline{A_{TM}} &\implies M \text{ accepts } w \\ &\implies L(M') = \Sigma^* \\ &\implies L(M') \text{ is not finite} \\ &\implies \langle M' \rangle \notin \text{FINITE} \end{aligned}$$

So $\langle M, w \rangle \in \overline{A_{TM}} \iff f(\langle M, w \rangle) \in \text{FINITE}$. By the Church-Turing thesis, f is computable.

- **Theorem:** Let A and B be languages. Then $A \leq_m B$ iff $\overline{A} \leq_m \overline{B}$.
- Proof: (\Rightarrow). Suppose $A \leq_m B$. Then there is a computable function f such that $\forall x \in \Sigma^*$:

$$\begin{aligned} x \in A &\iff f(x) \in B \\ x \notin A &\iff f(x) \notin B \\ x \in \overline{A} &\iff f(x) \in \overline{B} \end{aligned}$$

So the same function f shows $\overline{A} \leq_m \overline{B}$. The other direction is symmetric.

- **Theorem:** $\overline{\text{FINITE}}$ is unrecognizable.
- Proof: We need to show $\overline{A_{TM}} \leq_m \overline{\text{FINITE}}$. By the previous theorem this is equivalent to showing $A_{TM} \leq_m \text{FINITE}$. Define $f(\langle M, w \rangle) = \langle M' \rangle$, where M' = "on input x ...

1. Run M on input w for $|x|$ steps
2. If M accepts then reject
3. else accept

Then,

$$\begin{aligned} \langle M, w \rangle \in A_{TM} &\implies M \text{ accepts } w \text{ in } t \text{ steps} \\ &\implies L(M') = \{x \mid |x| < t\} \\ &\implies L(M') \text{ is finite} \\ &\implies \langle M' \rangle \in \text{FINITE} \\ \langle M, w \rangle \notin A_{TM} &\implies M \text{ doesn't accept } w \\ &\implies L(M') = \Sigma^* \\ &\implies L(M') \text{ is not finite} \\ &\implies \langle M' \rangle \notin \text{FINITE} \end{aligned}$$

So $\langle M, w \rangle \in A_{\text{TM}} \iff f(\langle M, w \rangle) \in \text{FINITE}$. By the Church-Turing thesis, f is computable.

Use the complement

- Remember: if you are trying to prove a language is undecidable or unrecognizable, it may be helpful to think about its complement. The following table shows the possible scenarios:

Scenario	L	complement of L
(1)	decidable	decidable
(2)	recognizable, but not decidable	unrecognizable
(3)	unrecognizable	recognizable, but not decidable
(4)	unrecognizable	unrecognizable