Last time we saw how to prove decidability and undecidability of certain languages. So, how does one know whether a given language is decidable or not?

**Detecting undecidability. Rice’s Theorem.**

Sometimes it is very difficult to tell whether a given language is decidable, but there are some “rules of thumb” that may help:

- Questions about a particular behavior of Turing machines are *often, but not always*, undecidable. For example:
  - “Does a TM $M$ on input $w$ ever write a particular symbol on its tape?”
  - “Does a TM $M$ have a useless (never entered) state?”
  - “Does a TM $M$ on input $w$ ever tries to move its head left while on the left end of the tape?”

However, some questions like that are decidable, so be careful! For example:

- “Does a TM $M$ on input $w$ ever tries to move its head left?” is decidable. Here’s an idea for the decider:

Given $⟨M, w⟩$ simulate $M$ on $w$ for $|w|$ steps. If the machine hasn’t made any left moves, then the head must be pointing to the first $⊔$ after $w$. And, now do a search in the state diagram of $M$ by following all transitions that have $⊔$ as input. The search is similar to the one we did for $E_{DFA}$. We’re only interested in these transitions, because if the machine never moves left, the only available input symbol will be $⊔$. If none of the transitions move left, *accept*, otherwise *reject*.

- Questions about the language of Turing machines are almost always undecidable. This is known as *Rice’s Theorem*, and in essence, it says that every non-trivial property of the language of Turing machines is undecidable. Formally,

**Theorem (Rice’s Theorem):** Let $L$ be a language of the form

$$L = \{⟨M⟩ \mid L(M) \text{ has some property } P\}$$

where

1. $P$ is non-trivial, i.e. there exist at least one machine $M$ such that $⟨M⟩ \in L$, and at least one machine $M$ such that $⟨M⟩ \notin L$.

2. $P$ is indeed a property of the *language* of TMs, i.e. whenever $L(M_1) = L(M_2)$, we have $⟨M_1⟩ \in L$ if and only if $⟨M_2⟩ \in L$.

Then $L$ is undecidable.
Proof: The proof of this theorem is a generalization of all the undecidability proofs via reduction from $A_{TM}$ that we have done in the last two classes. Whenever we had to prove that some language $K$ is undecidable, we would reduce $A_{TM}$ to $K$ in the following way: on input $\langle M, w \rangle$ we would construct another Turing machine $\langle M_1 \rangle$, such that either $\langle M_1 \rangle \in K$ when $M$ accepts $w$, and $\langle M_1 \rangle \notin K$ when $M$ doesn’t accept $w$, or the other way around: $\langle M_1 \rangle \notin K$ when $M$ accepts $w$, and $\langle M_1 \rangle \in K$ when $M$ doesn’t accept $w$. Now, we generalize this construction.

Assume that $L$ is decidable, that is there exists a Turing machine $M_L$ that decides $L$. We’re going to construct the Turing machine $M_{ATM}$ that will decide $A_{TM}$. Since, $A_{TM}$ is undecidable, this will leads to contradiction.

Let $M_\emptyset$ be a Turing machine with an empty language, that is $L(M_\emptyset) = \emptyset$. Then, we have two choices: either $\langle M_\emptyset \rangle \in L$, or $\langle M_\emptyset \rangle \notin L$.

- If $\langle M_\emptyset \rangle \in L$, then let $M_x$ be another Turing machine, such that $\langle M_x \rangle \notin L$ (by condition 1, non-triviality, such $M_x$ exists). Then the machine $M_1$, whose code we will pass to the decider for $L$, will work as follows:

  $M_1 =$ “on input $x$

  1. Simulate $M$ on $w$.
  2. Reject if $M$ rejects $w$.
  3. If $M$ accepts $w$, simulate $M_x$ on $x$ and accept if $M_x$ accepts $x$, reject otherwise.”

Main point: what is the language of $M_1$? If $M$ accepts $w$, then $M_1$ will act exactly like $M_x$, and so $L(M_1) = L(M_x)$. If $M$ doesn’t accept $w$, then $L(M_1) = \emptyset$. So, the decider for $A_{TM}$ can be constructed as follows:

$M_{ATM} =$ “on input $\langle M, w \rangle$

  1. Construct the code for the machine $M_1$, as described above.
  2. Simulate $M_L$ on input $\langle M_1 \rangle$.
  3. Accept if $M_L$ rejects, reject otherwise.“

$M_{ATM}$ decides $A_{TM}$ because it always halts, and on input $\langle M, w \rangle$, if $M$ accepts $w$, then $L(M_1) = L(M_x)$, and since $\langle M_x \rangle \notin L$, by condition 2, $\langle M_1 \rangle \notin L$, and so $M_L$ will reject $\langle M_1 \rangle$ and $M_{ATM}$ will accept. If, on the other hand, $M$ does not accept $w$, then $L(M_1) = L(M_\emptyset)$, and since $\langle M_\emptyset \rangle \in L$, by condition 2, $\langle M_1 \rangle \in L$, and so $M_L$ will accept $\langle M_1 \rangle$, and so $M_{ATM}$ will reject.

- In the second case, that is $\langle M_\emptyset \rangle \notin L$, then we take $M_x$ to be another Turing machine, such that $\langle M_x \rangle \in L$ (again, by condition 1, such $M_x$ exists). The machine $M_L$ will be exactly like in the previous case, however the machine $M_{ATM}$ for $A_{TM}$ will work in the opposite way from before:
\( M_{ATM} = \text{“on input } \langle M, w \rangle \text{”} \)

1. Construct the code for the machine \( M_1 \), as described above.
2. Simulate \( M_L \) on input \( \langle M_1 \rangle \).
3. Accept if \( M_L \) accepts, reject otherwise.”

Now, \( M_{ATM} \) decides \( ATM \) because it always halts, and on input \( \langle M, w \rangle \), if \( M \) accepts \( w \), then \( L(M_1) = L(M_x) \), and since \( \langle M_x \rangle \in L \), and so \( M_L \) will accept \( \langle M_1 \rangle \) and \( M_{ATM} \) will accept. If, on the other hand, \( M \) does not accept \( w \), then \( L(M_1) = L(M_\emptyset) \), and since \( \langle M_\emptyset \rangle \notin L \), by condition 2, \( \langle M_1 \rangle \notin L \), and so \( M_L \) will reject \( \langle M_1 \rangle \), and so \( M_{ATM} \) will reject.

\[ \square \]

Examples:

- \( ALL_{TM} = \{ \langle M \rangle \mid M \text{ is a TM }, \text{ and } L(M) = \Sigma^* \} \) is undecidable, by Rice’s theorem. The property \( P \) in this case is “\( L(M) = \Sigma^* \)”. \( P \) is non-trivial, because there is at least one TM that belongs to \( ALL_{TM} \) (e.g. the machine that accepts everything), and at least one machine that doesn’t belong to \( ALL_{TM} \) (e.g. the machine that rejects everything). Also, \( P \) is indeed a property of the language of TMs, because any for two machines \( M_1 \) and \( M_2 \), such that \( L(M_1) = L(M_2) \),

\[
\langle M_1 \rangle \in ALL_{TM} \iff L(M_1) = \Sigma^* \\
\iff L(M_2) = \Sigma^* \\
\iff \langle M_2 \rangle \in ALL_{TM}.
\]

- \( CFL_{TM} = \{ \langle M \rangle \mid M \text{ is a TM }, \text{ and } L(M) \text{ is context-free } \} \) is undecidable, by Rice’s theorem. The property \( P \) in this case is “\( L(M) \text{ is context-free} \)”. \( P \) is non-trivial, because there is at least one TM that belongs to \( CFL_{TM} \) (e.g. the machine that accepts everything), and at least one machine that doesn’t belong to \( CFL_{TM} \) (e.g. the machine that accepts \( 0^n1^n2^n \)). Also, \( P \) is indeed a property of the language of TMs, because any for two machines \( M_1 \) and \( M_2 \), such that \( L(M_1) = L(M_2) \),

\[
\langle M_1 \rangle \in CFL_{TM} \iff L(M_1) \text{ is context-free} \\
\iff L(M_2) \text{ is context-free} \\
\iff \langle M_2 \rangle \in CFL_{TM}.
\]

• Practice problems:

  - For decidability – all of the problems from Section 4.
  - For undecidability – 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.20, 5.28, 5.29, 5.30, 5.31, 5.35.