In our notation of Turing machines, we write “x = ⟨M⟩” to indicate that we’re interpreting x as the description of a TM M. If x is not a valid encoding of a TM, we by default interpret it as the TM M₀ that always rejects.

Last time, we saw that \( A_{DFA} = \{⟨M, w⟩ | M \text{ is a DFA and } M \text{ accepts } w\} \) is decidable. On the other hand, \( A_{TM} = \{⟨M, w⟩ | M \text{ is a TM and } M \text{ accepts } w\} \) is Turing-recognizable but undecidable. What about \( A_{TM}^C \)?

**Topics Covered**

1. Decidability Under Complements
2. TM Reductions
3. Rice’s Theorem

### 1 Decidability Under Complements

**Theorem** If a language \( L \) is decidable, then so is \( L^C \).

**Proof** Suppose \( L \) is decidable; let \( D \) be its decider. Consider the following TM \( D' \):

\[
D', \text{ on input } x:
\]

1. Run \( D \) on \( x \).

2. Accept if \( D \) rejects. Otherwise, reject.

We claim that \( D' \) is a decider for \( L^C \). If \( x \in L^C \), then \( x \notin L \), so \( D \) will reject \( x \) and \( D' \) will accept. If \( x \notin L^C \), then \( x \in L \), so \( D \) will accept \( x \) and \( D' \) will reject. Because \( D \) halts, \( D' \) also halts. Thus, \( D \) is a decider for \( L^C \). ■
Corollary  If \( L \) is undecidable, so is \( L^C \).

In particular, \( A_{TM}^C \) is undecidable because \( A_{TM} \) is undecidable. Is \( A_{TM}^C \) Turing-recognizable? Our intuition (thanks to Tracy) might say that given a TM recognizing \( L \), unless it’s a decider, it isn’t clear how we can construct a machine to recognize \( L^C \). This brings us to another theorem.

**Theorem**  If \( L \) and \( L^C \) are Turing-recognizable, then \( L \) is decidable.

**Proof**  Suppose \( M \) recognizes \( L \) and \( M' \) recognizes \( L^C \). Consider the TM \( D \) that operates as follows:

\[
D, \text{ on input } w:
\]

1. Run the following in parallel:
   
   (a) Run \( M \) on \( w \).
   
   (b) Run \( M' \) on \( w \).
   
   (c) If \( M \) accepts, accept. If \( M' \) accepts, reject.

The TM \( D \) is a decider for \( L \). If \( w \in L \), then \( M \) will accept and \( D \) will accept. If \( w \in L^C \), then \( M' \) will accept and \( D \) will reject. Any \( w \) is either in \( L \) or \( L^C \), and will thus be accepted by either \( M \) or \( M' \). Therefore \( D \) halts on any \( w \), and decides \( L \). \( \blacksquare \)

**Corollary**  If \( L \) is recognizable and undecidable, then \( L^C \) is not recognizable. This corollary implies that \( A_{TM}^C \) is not Turing-recognizable.

### 2 TM Reductions

Consider the language \( HALT_{TM} = \{(M, w) \mid M \text{ is a TM that halts on input } w\} \). To prove that \( HALT_{TM} \) is undecidable, we can reduce \( A_{TM} \) to \( HALT_{TM} \). This means that if we could decide \( HALT_{TM} \), then we could decide \( A_{TM} \). In particular, we reduce something we know is undecidable to something we wish to show is undecidable. The outline for our reduction is to first suppose that \( HALT_{TM} \) were decidable. Then, we use the decider \( D \) for \( HALT_{TM} \) to construct a decider \( D' \) for \( A_{TM} \). Since \( A_{TM} \) is not decidable, this yields a contradiction, showing that \( HALT_{TM} \) is undecidable.
Proof that $HALT_{TM}$ is Undecidable  Suppose $HALT_{TM}$ were decidable and let $D$ be its decider. Now construct a decider $D'$ for $A_{TM}$:

$D'$, on input $\langle M, w \rangle$:

1. Run $D$ on input $\langle M, w \rangle$.
2. If $D$ accepts, then run $M$ on $w$. Accept if $M$ accepts and reject if $M$ rejects.
3. Otherwise, if $D$ does not accept, then reject.

In analyzing this construction, we need to show that if $D$ decides $HALT_{TM}$, then $D'$ decides $A_{TM}$. Suppose $\langle M, w \rangle \in A_{TM}$. Then $M$ halts and accepts $w$. $D'$ will send its input to $D$, which will say that it halts. Then, running $M$ on $w$ will accept, and $D'$ will accept. Alternatively, suppose $\langle M, w \rangle \notin A_{TM}$. Either $M$ rejects or loops forever on input $w$. First, consider the case when $M$ rejects on input $w$. Then $D$ will say that $M$ halts on $w$, so $D'$ will run $M$ on $w$. This rejects, so $D'$ rejects. In the second case, $M$ loops on $w$. Then $D$ rejects because $M$ does not halt on $w$, and $D'$ will reject.

Thus, $D'$ accepts $\langle M, w \rangle \in A_{TM}$ and rejects $\langle M, w \rangle \notin A_{TM}$, so $D'$ decides $A_{TM}$. However, since we know that $A_{TM}$ is undecidable, $D'$ cannot exist. Hence, our assumption was incorrect and $HALT_{TM}$ is undecidable. ■

The language $E_{TM} = \{ \langle M \rangle | M \text{ does not accept anything; i.e. } L(M) = \emptyset \}$ is also undecidable.

Proof that $E_{TM}$ is Undecidable  Suppose we had a decider $D$ for $E_{TM}$. Then consider $D'$, a decider for $A_{TM}$:

$D'$, on input $\langle M, w \rangle$:

1. Transform the input into another TM, $\langle X \rangle$. $X$ has instructions, “On input $x$, run $M$ on $w$, and accept if $M$ accepts.”
2. Run $D$ on $\langle X \rangle$. Reject if $D$ accepts. Accept if $D$ rejects.

If $\langle M, w \rangle \in A_{TM}$, $X$ will accept every input so $\langle X \rangle \notin E_{TM}$. $D$ will reject and $D'$ will accept. If $\langle M, w \rangle \notin A_{TM}$, $X$ will reject on every input, so $\langle X \rangle \in E_{TM}$. In this case, $D$ will accept and $D'$ will reject. Thus, $D'$ decides $A_{TM}$, which we know is undecidable. This implies
that $E_{TM}$ must be undecidable.

Another undecidable language is $\text{CONTAINSEMPTY}_{TM} = \{ \langle M \rangle \mid \varepsilon \in L(M) \}$. We will shorten this to $CE_{TM}$.

**Proof that $CE_{TM}$ is Undecidable**  Suppose $CE_{TM}$ were decidable, and let $D$ be its decider. Then construct a decider $D'$ for $A_{TM}$ as follows:

\[
D', \text{ on input } \langle M, w \rangle: \\
1. \text{ Transform the input into another TM } \langle X \rangle. X \text{ has instructions, “On input } w', \text{ if } w' = \varepsilon, \text{ run } M \text{ on } w \text{ and accept if } M \text{ accepts. Else, reject.”} \\
2. \text{ Run } D \text{ on input } \langle X \rangle. \text{ Accept if } D \text{ accepts. Reject if } D \text{ rejects.}
\]

If $\langle M, w \rangle \in A_{TM}$, then $\varepsilon \in L(X)$ and $\langle X \rangle \in CE_{TM}$. Then $D$ accepts and $D'$ accepts. If $\langle M, w \rangle \notin A_{TM}$, then $\varepsilon \notin L(X)$ and $\langle X \rangle \notin CE_{TM}$. $D$ rejects, so $D'$ rejects. Thus $D'$ decides $A_{TM}$, which is a contradiction, so $CE_{TM}$ must be undecidable.

3 Rice’s Theorem

Rice’s Theorem tells us that any language of the form $P = \{ \langle M \rangle \mid L(M) \text{ satisfies some nontrivial property} \}$ is undecidable. A nontrivial property is a property such that there exist machines $M_{yes}$ and $M_{no}$ such that $L(M_{yes})$ satisfies the property and $L(M_{no})$ does not. More formally, consider the following statement of the theorem:

**Rice’s Theorem**  Let $P$ be a language of TM descriptions such that $P$ satisfies the following two conditions:

1. $P$ is nontrivial; that is, there exists a TM whose description is in $P$, and there exists a TM whose description is not in $P$.

2. $P$ is a property of the TM’s language. Whenever $L(M_1) = L(M_2)$, we have $\langle M_1 \rangle \in P$ if and only if $\langle M_2 \rangle \in P$.

Then $P$ is an undecidable language.

**Proof**  We break the proof down into two cases, based on whether or not the empty language $\emptyset$ satisfies the property.
1. First, suppose the empty language does not satisfy the property. For example, a machine whose language is empty is not included in the languages \( \text{NonReg}_{TM} = \{M \mid L(M) \text{ is not regular}\} \) and \( \text{Infinite}_{TM} = \{M \mid L(M) \text{ contains an infinite number of strings}\} \). Note that these languages both contain the description of some TM \( M_{yes} \). For the proof, we suppose that \( P \) were decidable, and let \( D \) be its decider. Then we construct the following decider \( D' \) for \( A_{TM} \):

\[
D', \text{ on input } (M, w):
\]

1. Construct \( X \), where \( X \)'s instructions are, “On input \( x \), run \( M \) on \( w \). If \( M \) accepts, run \( M_{yes} \) on \( x \).”

2. Run \( D \) on input \( (X) \). Accept if \( D \) accepts. Otherwise, reject.

If \( (M, w) \in A_{TM} \), then \( L(X) = L(M_{yes}) \). This means that \( (X) \in P \), so \( D \) accepts and \( D' \) accepts. If \( (M, w) \notin A_{TM} \), \( L(X) = \emptyset \). Then \( (X) \notin P \), so \( D \) rejects and \( D' \) rejects. Thus, \( D \) is a decider for \( A_{TM} \), which implies by contradiction that \( P \) is undecidable.

2. In the second case, suppose the empty language does satisfy the property. Then consider \( P^C = \{M \mid L(M) \text{ does not satisfy the property}\} \). Note that \( P^C \) falls under the first case, as \( \emptyset \) is in either \( P \) or \( P^C \). If \( P^C \) is undecidable, then \( P \) is also undecidable. \( \blacksquare \)

There are some languages which are undecidable, but cannot be proven so by Rice’s Theorem. One example is \( \text{EQ}_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\} \). Another is the Post correspondence problem (PCP), which has to do with the existence of matches in a layout of dominoes over an alphabet \( \Sigma \).