1 Make your own TMs!

An essential skill in Turing reductions involves constructing a Turing machine for a given language or behaviour. Let's practice this concept. Note that most one them have multiple answers, so it's time for you to be creative!

1. Construct a Turing machine M such that

   (a) $L(M) = \Sigma^*$

   Solution:

   M = on input w:
   1: Ignore w
   2: **accept**"

   (b) $L(M) = \emptyset$

   Solution:

   M = on input w:
   1: Ignore w
   2: **reject**"

   (c) $L(M) = \{3, 7, 6\}$

   Solution:

   M = on input w:
   1: **if** $w \in \{3, 7, 6\}$ **then accept** ▷ Iterate through the set and check if match
   2: **else reject**"

   (d) $L(M)$ is finite

   Solution: There are a lot of ways to construct a TM with finite language. We could reuse our TM from (c) because the set $\{3, 7, 6\}$ is finite.

   (e) $|L(M)|$ is even

   Solution: Notice that 0 is even, so we can reuse our TM from (b).

   (f) M loops on x if $x \notin \{3, 7, 6\}$

Solution: Again, there are multiple TMs for this language. One example is as follows:

M = on input w:
1: **if** $w \in \{3, 7, 6\}$ **then accept**
2: **else loop**"

Note that we could reject on line 1 instead, as long as M halts if $w \in \{3, 7, 6\}$ (but we need to specify clearly). On line 2, we don't necessarily have to explicitly describe how we want the machine to loop, it could be something simple like

M = on input w:
1: **for** $x = 1, 2, \ldots$ **do**
2: PRINT(x)

2. Construct two Turing machines M_1 and M_2 such that

(a) $L(M_1) \cup L(M_2) = \Sigma^*$

Solution: There are a lot of ways to make the union of two languages Σ^*. For instance,

- $L \cup \bar{L}$
- $\Sigma^* \cup \{\varepsilon\}$
- $\Sigma^* \cup \Sigma^*$

For $\Sigma^* \cup \Sigma^*$, we can just reuse our TM from 1(a) for both M_1 and M_2.

(b) $L(M_1) \cup L(M_2) = \emptyset$

Solution: Example: $\emptyset \cup \emptyset = \emptyset$. We can reuse our TM from 1(b) for both M_1 and M_2.

(c) $L(M_1) \cap L(M_2) = \emptyset$

Solution: Some useful facts: For any language L, $L \cap \bar{L} = \emptyset$ and $\emptyset \cap L = \emptyset$. So we can do the same thing as in 2(b) by having $L(M_1) = L(M_2) = \emptyset$. Of course, you could also pick any two languages that does not intersect ad $L(M_1)$ and $L(M_2)$.

(d) $L|(M_1) \cap L(M_2)| = 1$

Solution: Example: $\{a, b, c\} \cap \{a, d, e\} = \{a\}$, which has a size of 1.

M_1 = on input w:
1: **if** $w \in \{a, b, c\}$ **then accept**
2: **else reject**"

M_2 = on input w:
1: **if** $w \in \{a, d, e\}$ **then accept**
2: **else reject**"

2 Decidability and Turing Reductions

1. Which of the following languages is decidable?

◯ $L_{\text{HALT}} = \{(\langle M \rangle, x) : M$ is a Turing machine and M loops on $x\}$

◯ $L_{203} = \{\langle M \rangle : M$ is a Turing machine that halts on input '203'$\}$

√ $L_{376C} = \{\langle M \rangle : M$ **is a Turing machine that halts on input '376' in exactly 376 steps**$\}$

◯ $L_{376D} = \{\langle M \rangle : M$ is a Turing machine that halts on input '376' after a number of steps which is a multiple of 376 $\}$

◯ $L_{376E} = \{\langle M \rangle : M$ is Turing machine that halts (on empty input) on the 376th tape cell$\}$

Solution:

- Option A is undecidable. We have proven in lecture that L_{HALT} is undecidable.
- Option B is undecidable. Similarly to how we had shown that $L_{\varepsilon-\text{HALT}}$ is undecidable, we could replace the ε in that proof with 203 and everything else would be the same.
- Option C is decidable. We can make a decider for L_{376C} by on input $\langle M \rangle$, simulate M on '$376'$ for 376 steps and see what happens. Regardless of what happens, we would know what to return from the function in 376, or more importantly, a finite number of steps.
- Option D is undecidable. Intuitively, this would be undecidable since we M might keep running on an input and we would never be able to say with certainty that M would either loop or halt. A reduction to prove that this language is undecidable would likely use L_{HALT}.
- Option E is undecidable. There would be no way to determine in general what cell a Turing Machine would halt on since there is no way to tell if a Turing Machine will halt in the first place. A reduction to prove that this language is undecidable would likely also use L_{HALT}

2. Show that $L_{\text{ACC}} \leq_T \overline{L_{\text{ACC}}}$, where $\overline{L_{\text{ACC}}}$ is the complement of

$$L_{\text{ACC}} = \{(\langle M \rangle, x) : M \text{ is a Turing machine and } M \text{ accepts } x\}.$$

Then, conclude that $\overline{L_{\text{ACC}}}$ is undecidable.

Solution: Let E be a black box decider that decides $\overline{L_{\text{ACC}}}$. By definition, $E(\langle M \rangle, x)$ accepts if M does not accept x, and $E(\langle M \rangle, x)$ rejects if M accepts x. We can construct a decider D for L_{ACC} as follows:

D = on input $\langle M \rangle, x$:
1: Run E on $(\langle M \rangle, x)$
2: **if** $E(\langle M \rangle$ accepts **then reject**
3: **else accept**"

Analysis:

- $(\langle M \rangle, x) \in L_{\text{ACC}} \implies M$ accepts $x \implies (\langle M \rangle, x) \notin \overline{L_{\text{ACC}}} \implies E$ rejects $x \implies D$ accepts x
- $(\langle M \rangle, x) \notin L_{\text{ACC}} \implies M$ does not accept $x \implies (\langle M \rangle, x) \in \overline{L_{\text{ACC}}} \implies E$ accepts $x \implies D$ rejects x

Since E is a decider, it necessarily halts on all inputs. Thus, D must halt on all inputs. Since D accepts all $(\langle M \rangle, x) \in L_{\text{ACC}}$ and rejects all $(\langle M \rangle, \notin L_{\text{ACC}})$, D is a decider for L_{ACC}.

We have shown that $L_{\text{ACC}} \leq_T \overline{L_{\text{ACC}}}$, since L_{ACC} is undecidable, $\overline{L_{\text{ACC}}}$ is also undecidable.

3. Let L_{LOOPS} be defined as follows:

$$L_{\text{LOOPS}} = \{(\langle M \rangle, x) : M \text{ is a Turing machine and } M \text{ loops on } x\}.$$

Prove via the reduction $L_{\text{HALT}} \leq_T L_{\text{LOOPS}}$ that L_{LOOPS} is undecidable.

Solution: Let E be a black box decider that decides L_{LOOPS}. By definition, $E(\langle M \rangle, x)$ accepts if M loops on x, and $E(\langle M \rangle, x)$ rejects if M halts on x. We can construct a decider D for L_{ACC} as follows:

D = on input $\langle M \rangle, x$:
1: Run E on $(\langle M \rangle, x)$
2: **if** $E(\langle M \rangle$ accepts **then reject**
3: **else accept**"

Analysis:

- $(\langle M\rangle, x) \in L_{\text{HALT}} \implies$ M halts on $x \implies (\langle M\rangle, x) \notin L_{\text{LOOPS}} \implies E$ rejects $x \implies D$ accepts x
- $(\langle M\rangle, x) \notin L_{\text{HALT}} \implies$ M loops on $x \implies (\langle M\rangle, x) \in L_{\text{LOOPS}} \implies E$ accepts $x \implies D$ rejects x

Since E is a decider, it necessarily halts on all inputs. Thus, D must halt on all inputs. Since D accepts all $(\langle M\rangle, x) \in L_{\text{HALT}}$ and rejects all $(\langle M\rangle, \notin L_{\text{HALT}})$, D is a decider for L_{HALT}.

We have shown that $L_{\text{HALT}} \leq_T L_{\text{LOOPS}}$, since L_{HALT} is undecidable, L_{LOOPS} is also undecidable.

4. Suppose A and B are languages over $\{a, b\}$ defined as follows:

$$A = \{a^n : n \geq 0\}, B = \{b^n : n \geq 0\}.$$

Show that $A \leq_T B$ (which implies that if B is decidable, then A is decidable). Although not necessary for a correct reduction, we require that you use the blackbox decider for B in your solution.

Solution: Given a black box M_B that decides B, we can construct a machine M_A that decides A as follows:

$M_A =$ "on input w"

$M_A =$ on input w:
1: Create a new string w': Iterate through input w. If the cell reads a, append a b to w'; if the cell reads b, append an a to w'.
2: Run M_B on w'
3: **if** $M_B(w')$ accepts **then accept**
4: **else reject**"

Analysis:

- $w \in A \implies w = a^n \implies w' = b^n \implies M_B(w')$ accepts $\implies M_A(w)$ accepts
- $w \notin A \implies w \in \{a, b\}^*$: some of the characters are b or the length of w is not $n \implies w' \in \{a, b\}^*$: some of the characters are a of the length of w' is not $n \implies w' \notin B \implies M_B(w')$ rejects $\implies M_B(w')$ rejects

Both machines will always halt since inputs are finite length. Now we can say M_A decides A. We are able to construct a decider for A using a black box decider for B, so we conclude that $A \leq_T B$.

3 Turing Reductions Involving Creating TMs

1. Let $L_{\text{singleton}}$ be defined as follows:

$$L_{\text{singleton}} = \{\langle M\rangle : L(M) = \{\text{“eecs376”}\}\}.$$

Prove via the reduction $L_{\text{ACC}} \leq_T L_{\text{singleton}}$ that $L_{\text{singleton}}$ is undecidable.

Solution: Let $D_{L_{\text{singleton}}}$ be a black-box decider for $L_{\text{singleton}}$.

$D_{L_{\text{ACC}}}$ = on input $(\langle M\rangle, x)$):
Construct a machine M' as follows:

> M' = on input $\langle w\rangle$:
> **if** $w \neq$ “eecs376” **then**, reject
> Run $M(x)$ and output the same

Run $D_{L_{\text{singleton}}}$ on input $\langle M'\rangle$
if $D_{L_{\text{singleton}}}(\langle M'\rangle)$ accepts **then** accept
else reject”

Analysis: We claim that $D_{L_{\text{ACC}}}$ is a decider for L_{ACC}.

Suppose $\langle M, x\rangle \in L_{\text{ACC}}$. Then M' on input w accepts if and only if $M(x)$ accepts (otherwise, it rejects or loops). This implies that $L(M') = \{\text{“eecs376”}\}$. This implies that $D_{L_{\text{singleton}}}(M')$ accepts, so $D_{L_{\text{ACC}}}$ accepts $\langle M, x\rangle$.

Suppose $\langle M, x\rangle \notin L_{\text{ACC}}$. Then M' on input w does not accept if $w \neq$ “eecs376”, or $M(x)$ does not accept. This implies that $L(M') = \emptyset$, so $D_{L_{\text{singleton}}}(M')$ rejects, and $D_{L_{\text{ACC}}}$ rejects.

2. For each of the following languages, state whether it is decidable or undecidable. If decidable, describe and analyze a program that decides it. If undecidable, show that it is Turing reducible from an undecidable language L of your own choice.

(a) $L_{\text{OnlyOnes}} = \{x \in \{0, 1\}^* : x \text{ consists only of 1's}\}$.

Solution: Yes, the language is decidable. We will show this by constructing a decider that checks if x contains 0-s:

```
function D(x)
    Let x_i be ith character in x
    for x_i ∈ x do
        if x_i = 0 then reject
    accept
```

D always halts since it makes one pass through a finite string. Now we must prove that D is a decider $L_{OnlyOnes}$:

- $x \in L_{OnlyOnes} \implies x$ consists of only 1's $\implies D$ accepts
- $x \notin L_{OnlyOnes} \implies x$ contains a 0 $\implies D$ rejects

(b) $L_{\text{TuringOnlyOnes}} = \{\langle M \rangle : L(M) = L_{\text{OnlyOnes}}\}$.

Solution: We will show that $L_{\text{TuringOnlyOnes}}$ is undecidable via Turing Reduction from L_{ACC}. Let T be a black box for $L_{\text{TuringOnlyOnes}}$ and let D be a decider for $L_{OnlyOnes}$. Define a decider A for L_{ACC} as follows:

$A =$ on input $(\langle M \rangle, x)$):
1: Construct a machine M' as follows:

> $M' =$ on input $\langle w \rangle$:
> 1: **if** $D(w)$ accepts **then**, run $M(x)$ and output same
> 2: **else** reject"

2: Run T on input $(\langle M' \rangle)$
3: **if** T accepted **then** accept
4: **else** reject"

Since T is a decider, A necessarily halts. Therefore it remains to show that A is a decider for L_{ACC} :

- $(\langle M \rangle, x) \in L_{\text{ACC}} \implies M$ accepts $x \implies M'$ accepts all strings in $L_{OnlyOnes} \implies T(\langle M' \rangle)$ accepts $\implies A$ accepts
- $(\langle M \rangle, x) \notin L_{\text{ACC}} \implies M$ rejects or loops on $x \implies M'$ rejects or loops on all strings in $L_{OnlyOnes} \implies T(\langle M' \rangle)$ rejects $\implies A$ rejects

3. Let L_{EVEN} be defined as follows:

$$L_{\text{EVEN}} = \{\langle M \rangle : |L(M)| \text{ is even}\}.$$

Prove via the reduction $L_{\text{ACC}} \leq_T L_{\text{EVEN}}$ that L_{EVEN} is undecidable.

Solution: L_{EVEN} is undecidable, thus we are going to do a proof via Turing reduction from L_{ACC} to L_{EVEN} ($L_{\text{ACC}} \leq_T L_{\text{EVEN}}$). Assume that there is some decider D of L_{EVEN}.

We are going to use the decider D to construct a decider T for L_{ACC}. Given a Turing Machine M and an input x as input, T works as follows:

$T =$ on input $\langle M\rangle, x$:
1: Construct Turing machine M_x as follows:

$M_x =$ on input w:
1: **if** $w = x$ and $M(w)$ accepts **then accept**
2: **else reject"**

2: **if** $D(M_x)$ accepts **then reject**
3: **else accept** "

Note that

$$L(M_x) = \begin{cases} \{x\}, & \text{if } x \in L(M) \\ \emptyset, & \text{if } x \notin L(M), \end{cases}$$

and in particular $|L(M_x)|$ is even if and only if $x \notin L(M_x)$. That is, $(\langle M\rangle, x) \in L_{\text{ACC}}$ if and only if $|L(M_x)|$ is not even. From this idea, if we have a $\langle M\rangle, x \in L_{\text{ACC}}$, then $|L(M_x)|$ will not be even, and D will reject, so T will accept. If we have a $\langle M\rangle, x \notin L_{\text{ACC}}$, then $|L(M_x)|$ will be even, and D will accept, so T will reject. Ultimately, we can see that T decides L_{ACC}, so $L_{\text{ACC}} \leq_T L_{\text{EVEN}}$. Since L_{ACC} is undecidable, we may conclude that L_{EVEN} is undecidable. $\square$

4. Let L_u be defined as follows:

$$L_u = \{(\langle M_1\rangle, \langle M_2\rangle) : L(M_1) \cup L(M_2) = \emptyset\}$$

Prove via the reduction $L_{ACC} \leq_T L_u$ that L_u is undecidable.

Solution: We show the Turing reduction $L_{ACC} \leq_T L_u$, meaning that given a black-box U that decides L_u, we can construct a machine A that decides L_{ACC}.

$A =$ on input $(\langle M\rangle, x))$:
1: Construct a machine M' as follows:

$M' =$ on input $\langle w\rangle$:
1: Run $M(x)$ and output same

2: Query $U(\langle M'\rangle, \langle M'\rangle)$
3: **if** U accepted **then** reject
4: **else** accept"

Analysis:

- $(\langle M\rangle, x) \in \text{Ł}_{ACC \Rightarrow} M$ accepts $x \Rightarrow M'$ accepts all inputs $\Rightarrow L(M') = \Sigma^* \Rightarrow L(M') \cup L(M') = L(M') = \Sigma^* \Rightarrow (\langle M'\rangle, \langle M'\rangle) \notin L_u \Rightarrow U$ rejects input $(\langle M'\rangle, \langle M'\rangle) \Rightarrow A$ accepts

- $(\langle M \rangle, x) \notin$ Ł$ACC \Rightarrow M$ does not accept $x \Rightarrow M'$ does not accept any input $\Rightarrow L(M') = \emptyset \Rightarrow L(M') \cup L(M') = L(M') = \emptyset \Rightarrow (\langle M' \rangle, \langle M' \rangle) \in L_u \Rightarrow U$ accepts input $(\langle M' \rangle, \langle M' \rangle) \Rightarrow A$ rejects

Therefore, $L_{ACC} \leq_T L_u$, but since we know that L_{ACC} is undecidable, then L_u is undecidable as well.