1 Turing Machines

1. Are the following true or false?

   (a) Given TM M, there can be more than one distinct language L decided by M.

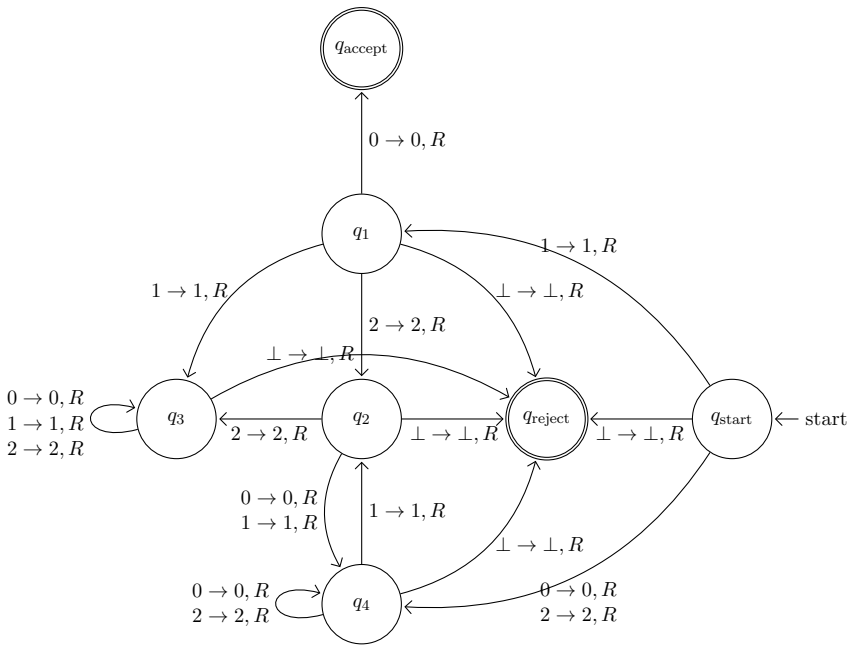
   Solution: False, a Turing machine can decide either zero or one languages.

   - Deciders are required to halt on all inputs, so any Turing machine that does not halt on some input is not a decider. These machines decide zero languages.
   - We now restrict ourselves to deciders. The language of a decider is the set of all (*finite*) strings that machine accepts. Let decider M decide languages L_1 and L_2 For sake of contradiction, suppose $L_1 \neq L_2$. That is, WLOG $\exists x \in L_1 \setminus L_2$. Turing machines are defined as deterministic, so M cannot both accept and reject x, so we've reached a contradiction. L_1 and L_2 must be equivalent, so any decider decides one language.

   (b) Given decidable language L, there can be more than one distinct TM M that decides L.

   Solution: True. Consider an arbitrary decidable language L and machine that decides it M. We can write a different machine M' that begins by transitioning one cell right, then one cell left, not writing either time, then has an identical transition function to M. Because M' is defined differently, it is a different machine. However, M' will trivially decide L, so M and M' decide the same language. In fact, there are infinite Turing machines for any decidable language.

2. Consider the Turing Machine whose state diagram is given below:



Which of the following statements is true about this Turing Machine?

(a) It accepts all strings that contain the substring "10."

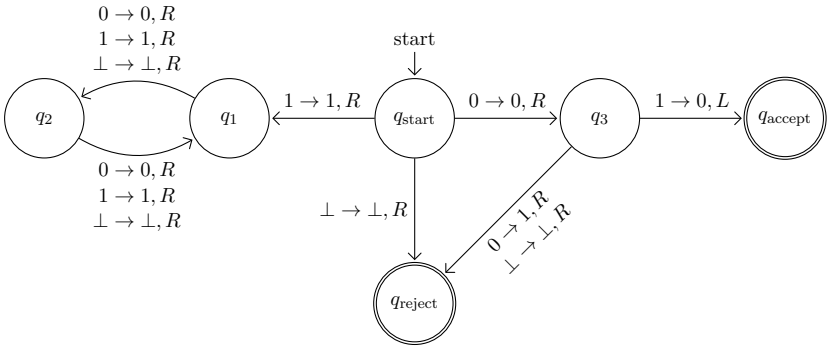
(b) It accepts all strings that start with the substring "10."

(c) It loops on any input string that contains only 2s.

(d) It will reject any string that contains $\perp$.

Solution: B
A is false. One counter-example is the string "010" which contains the substring "10" but would cause this machine to reject.
B is true. Any string that begins with "10" will cause this machine to reach the accept state, which means the machine will accept regardless of all the remaining characters of the string.
C is false because a string will eventually reach a blank symbol, then reject.
D is false (and nonsensical). The input string cannot contain the blank symbol.

3. Consider the Turing Machine whose state diagram is given below:



What language over $\Sigma = \{0, 1\}$ does this Turing machine decide, if any?

Note: We say that M decides language A if $A = \{w : M \text{ accepts } w\}$ and M halts on all inputs.

(a) $\{01\} \cup \{s \in \Sigma^* : s \text{ starts with } 1\}$

(b) $\{01\}$

(c) $\{s \in \Sigma^* : s \text{ starts with } 1\}$

(d) $\{\varepsilon, 0, 00\}$

(e) None of the above

Solution: This TM fails to halt on inputs of the form $1(0|1)^*$, so there is no language it decides.

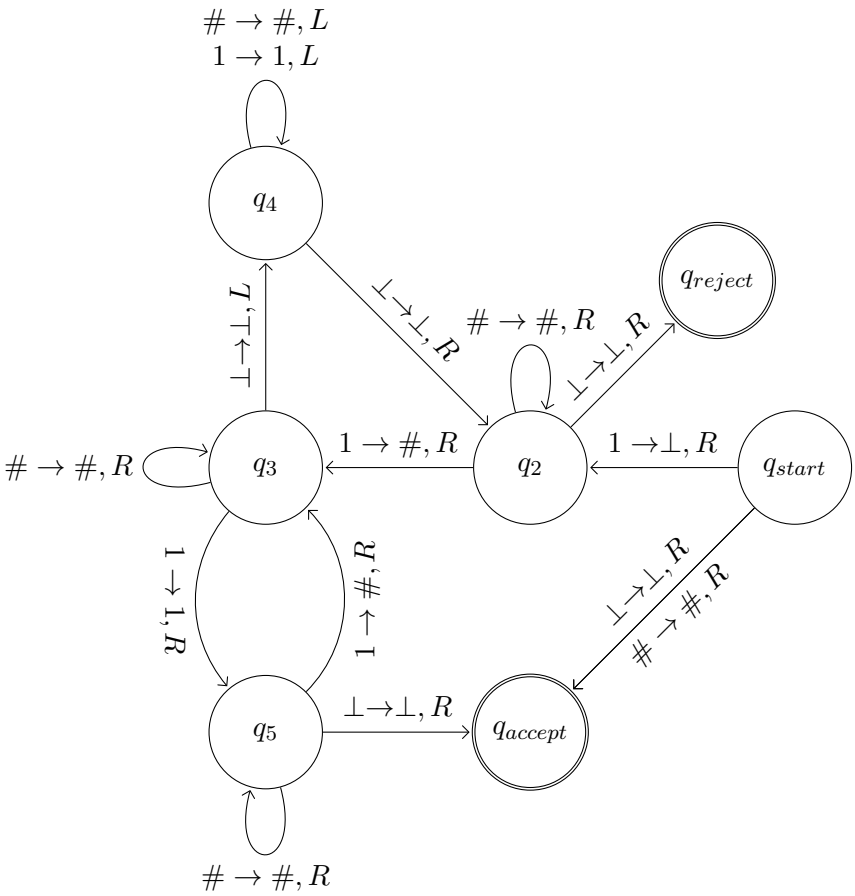
4. (A. Every / B. Some / C. No) **Turing machine** writes to only finitely many of its tape cells during its execution.

> **Solution: Some**
> Finitely Many: Occurs anytime a Turing Machine halted on an input. Halting implies that only finitely many transitions occurred and each transition causes exactly one cell on the tape to be written to (even if what is written back is the same thing as was already written).
> Infinitely Many: One example would be a Turing Machine that was passed an empty input and has a transition function including the transition $\delta(\perp, q_0) = (q_0, \perp, R)$. In general, a Turing Machine would write to infinitely many tape cells anytime the Turing Machine looped infinitely AND was not stuck in a cycle that caused the head to continue looping over a finite portion of the tape (e.g. move right, move left, move right, ...).

5. (A. Every / B. Some / C. No) **decider** writes to only finitely many of its tape cells during its execution.

> **Solution: Every**
> A decider must accept or reject for every input in Σ^*, which means it halts after a finite number of steps on all inputs. If its execution is finite, the number of cells written to must also be finite.

6. Consider the Turing machine represented by the following figure, for input alphabet $\Sigma = \{1\}$ and tape alphabet $\Gamma = \{1, \#, \perp\}$. (Note that there are *two* extra characters in the latter.)



What language does this Turing machine decide (if any)? Justify your answer.
Hint: *What strings does it not accept?*

Solution: This Turing machine decides the language:

$$\{x \mid x \notin \{1\}^{2^n} \text{ for } n \in \mathbb{N}_0\}$$

We will consider some example runs of the machine to understand how this Turing machine works, and argue how this example generalizes to any string in $\{1\}^*$.

Consider the input string (which is in the above language) "11111111" and note that the rest of the tape is filled with $\bot$:

- First we will have q_{start} and transition to q_2 when we detect the first 1 giving us the string: "$\bot$ 1111111"
- We detect a 1 so we transition to state q_3 as follows: "$\bot$ #111111"
- We now do a loop making every other 1 a # until we hit the end of the string, the result is as follows: "$\bot$ #1#1#1#"

- We then move to start q_4 reversing directions now and moving left. The function of state q_4 is to take us back to the beginning of the string (the first $\perp$ symbol). After we hit the first $\perp$ symbol we are at q_2 with the following on our tape: "$\perp$ #1#1#1#"
- We repeat the process again replacing every other 1 with a # and when we reach the end of the string we have:

  "$\perp$ ###1###"
- We reset back to the beginning of the tape and are again at:

  "$\perp$ ###1###"
- After another iteration we are at the end of the tape with:

  "$\perp$ #######"
- We skip ever going to q_5 this time since we are already at the end of the tape after one replacement. We transition back to the start to state q_2:

  "$\perp$ #######"
- At q_2 we loop through the rest of the string until we hit a $\perp$ since it is composed entirely of # characters now. We notice that if we ever hit a $\perp$ symbol while at state q_2 we transition to state q_{reject}, and thus we have finished, rejecting the string $\{1\}^{2^3}$ as expected.

More generally, each pass of the machine over the input replaces every other 1 with #, and accepts if the number of 1s it encounters is an odd number greater than one. Therefore, the machine rejects if and only if every pass sees an even number of 1s, followed by a final pass with exactly one 1. The machine therefore rejects every string of the form $x \in \{1\}^{2^n}$, and accepts all others.

2 Decidability

1. Show the following statement is true: For any decidable language L, the language $L' = L \cup \{\varepsilon\}$ is decidable.

Solution: Let D_L decide L.

There are two cases.

If $\varepsilon \in L$, then $L = L'$ and the claim holds.

If $\varepsilon \notin L$, then we can use the following decider:

```
1: function D_{L'}(x)
2:     if x = ε then
3:         Accept
4:     else if D_L(x) accepts then
5:         Accept
6:     else
7:         Reject
```

$\mathrm{D}_{L'}$ halts on all inputs because D_L is a decider for L.

We have the following implications:

- $x \in L' \implies x \in L \cup \{\varepsilon\} \implies x = \varepsilon \vee x \in L$. If $x = \varepsilon$, then $\mathrm{D}_{L'}$ accepts on lines 2-3. If $x \in L$, then $D_L(x)$ accepts on lines 4-5.
- $x \notin L' \implies x \notin L \cup \{\varepsilon\} \implies x \neq \varepsilon \wedge x \notin L$. Because $x \neq \varepsilon$ and $x \notin L$, x satisfies neither of the conditions which would cause $\mathrm{D}_{L'}$ to accept, so $\mathrm{D}_{L'}$ rejects.

So $\mathrm{D}_{L'}$ decides L'.

2. Let A and B be languages. $A \triangle B$ denotes the *symmetric difference* of the sets A and B. That is, $A \triangle B = (A \cup B) \setminus (A \cap B)$.

   Suppose that A and B are decidable. Then $A \triangle B$ is decidable for (all/some/no) inputs. Provide justification to support your answer.

Solution: $A \triangle B$ is decidable for all inputs. A decider D for $A \triangle B$ on input x could run a decider for A on x and a decider for B on x and reject if either both accepted or both rejected. Otherwise D accepts.

Analysis:

Observe that

$$\begin{aligned} x \in A \triangle B &\implies x \in (A \cup B) \setminus (A \cap B) \\ &\implies (x \in A \setminus B) \vee (x \in B \setminus A) \\ &\implies \text{exactly one of the deciders for } A \text{ and } B \text{ accepts} \\ &\implies D(x) \text{ accepts.} \end{aligned}$$

On the other hand,

$$\begin{aligned} x \notin A \triangle B &\implies x \in (A \cap B) \cup \overline{(A \cup B)} \\ &\implies \text{both of the deciders for } A \text{ and } B \text{ accept or reject} \\ &\implies D(x) \text{ rejects.} \end{aligned}$$

3. The set difference of S and T is written $S \setminus T$, and defined as $S \setminus T = \{s \in S : s \notin T\}$.

   If S and T are both decidable, $S \setminus T$ is decidable for (all/some/no) inputs. Provide justification to support your answer.

Solution: $S \setminus T$ is decidable for all inputs.

A decider for $S \setminus T$ can be constructed by running a decider for S and a decider for T and performing the appropriate checks.

function $D_{S \setminus T}(x)$
  if $D_S(x)$ accepts and $D_T(x)$ rejects **then**
    accept
  reject

Analysis: Observe that

$$\begin{aligned} x \in A \setminus B &\implies x \in (A \cap \bar{B}) \\ &\implies (x \in A) \wedge (x \in \bar{B}) \\ &\implies D_S(x) \text{ accepts and } D_T(x) \text{ rejects} \\ &\implies D(x) \text{ accepts.} \end{aligned}$$

On the other hand

$$\begin{aligned} x \notin A \setminus B &\implies x \notin A \vee x \notin \bar{B} \\ &\implies x \notin A \vee x \in B \\ &\implies \text{If } x \notin A \text{ then } D_S \text{ rejects; if } x \in B \text{ then } D_T \text{ accepts.} \\ &\implies \text{In both cases } D(x) \text{ rejects.} \end{aligned}$$

3 Countability and Diagonalization

1. (a) Show that the set consisting of all the (finite-length) ASCII strings is countable.

Solution: We know that this set represents all the strings using ASCII characters of finite length. Let Σ represents all the 128 ASCII characters. We can simply describe each string as an element Σ^*. We can enumerate all the strings by first listing all the strings of length 0, then listing all the strings of length 1, then all the strings of length 2, and so on. There are 128^k strings of length exactly k, so for any string of length k', there are finitely many strings that have the same or smaller length. Thus, this list will eventually get to any string of length k'. Because we will eventually enumerate all ASCII strings of finite length we know that this set must be countable. This is the same as creating a bijection between the natural numbers and the set of all finite length ASCII strings.

   (b) Show that the set of all programs in C++ is countable.
   Hint: Every C++ program can be mapped to a string in some subset of ASCII strings

(in particular, those corresponding to well-formatted C++ programs).

> **Solution:** Building off of part (a), we confirmed that the set of all ASCII strings of finite length is countable. We realize that all programs in C++ can be actually mapped to a ASCII string and thus by the same reasoning we can simply list all programs in ASCII string format in order of their length, as in part (a). Since we can list all C++ programs, the set of all programs in C++ is countable.

(c) Prove that the set of all decidable languages over a given alphabet Σ is countable.

> **Solution:** Since every decidable language requires a Turing Machine that decides it, decidable languages over a given alphabet Σ are in bijection with some subset of all Turing Machines over Σ. We showed that there are only countably many such Turing Machines, so we may conclude that there are only countably many decidable languages.

2. Let x, y be binary strings of the same length n over $\Sigma = \{0, 1\}$. The *Hamming distance* between x and y, written $d_H(x, y)$, is the number of positions $i \in \{1, 2, \ldots, n\}$ for which $x_i \neq y_i$. For example, $d_H(11100, 10101) = 2$ because the two strings only differ in the second and fifth characters.

   Consider an infinite list of infinite binary sequences:

$$\begin{aligned} s_1 &= b_{11}b_{12}b_{13}\cdots \\ s_2 &= b_{21}b_{22}b_{23}\cdots \\ s_3 &= b_{31}b_{32}b_{33}\cdots \\ &\vdots \end{aligned}$$

   where each $b_{ij} \in \{0, 1\}$. Cantor's diagonalization argument shows that the sequence $\overline{b}_{11}\overline{b}_{22}\overline{b}_{33}\cdots$ has Hamming distance at least 1 from every sequence in the list, where $\overline{b}$ denotes the complement of the bit b.

   (a) Extend the argument by constructing, with justification, a sequence that has Hamming distance at least d from each sequence in the list, where d is some arbitrary given positive integer.

> **Solution:** Using the same principle that we used to construct a binary sequences with Hamming distance 1 from every other sequence in the list, we can construct a sequence s that has Hamming distance d from every other sequence in the list in the following way:
>
> $$s = \overline{b}_{1,1}, \overline{b}_{1,2}, ..., \overline{b}_{1,d}, \overline{b}_{2,d+1}, ..., \overline{b}_{2,d+d}, \overline{b}_{3,2d+1}, ...$$
>
> This solution produces a Hamming distance of d as given any sequence s_k from the list, s differs from s_k in at least d places. Namely, $s_{d(k-1)}$ to $s_{d(k-1)+d}$.

(b) Construct, with justification, a binary sequences that has *infinite* Hamming distance from each sequence in the list, i.e., it differs from each string in an infinite number of positions.

Solution: Let p_k denote the k^{th} prime number. So $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, \ldots$. If we produce a sequence s by flipping every $(p_k)^n$-th bit of every string s_k (i.e., flip bits b_{k,p_k^n}) for $n \geq 1$, then s will differ from every sequence in the list in an infinite number of places.

This works as a consequence of the Fundamental Theorem of Arithmetic which states that the prime factorization of any positive integer is unique. There will never be any collisions in this construction since if $p^n = q^m$ for some primes p, q with $n, m > 0$, it is necessarily the case that $p = q$ and $n = m$.

3. In lecture, we proved the existence of undecidable languages using the diagonalization method. The idea of diagonalization is not limited to reason about the countability of an infinite sets, we could also use it to prove undecidability a language. In this problem, we outline the proof that the language

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

is undecidable using the diagonalization method.

Suppose for the sake of contradiction that L_{ACC} is decidable with a decider H. In the following table, we list all Turing machines down the rows, $M_1, M_2, \ldots$, and all their descriptions across the columns, $\langle M_1 \rangle, \langle M_2 \rangle, \ldots$. The entries tell whether the machine in a given row accepts the input in a given column. The entry is *accept* if the machine accepts the input but is blank if it rejects or loops on that input. Suppose we have the following behaviour table:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\ldots$
M_1	*accept*		*accept*		$\ldots$
M_2	*accept*	*accept*	*accept*	*accept*	$\ldots$
M_3					$\ldots$
M_4	*accept*	*accept*			$\ldots$
$\vdots$					

(a) Based on the example behavior table above, fill in the following table where the i, j entry is the behavior of H when taking in $\langle M_i, \langle M_j \rangle \rangle$ as input.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\ldots$
M_1					$\ldots$
M_2					$\ldots$
M_3					$\ldots$
M_4					$\ldots$
$\vdots$					

Solution:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
M_1	*accept*	*reject*	*accept*	*reject*	...
M_2	*accept*	*accept*	*accept*	*accept*	...
M_3	*reject*	*reject*	*reject*	*reject*	...
M_4	*accept*	*accept*	*reject*	*reject*	...
$\vdots$					

(b) Construct a new Turing machine D that uses H as a subroutine to determine the behavior of a Turing machine M when M is given its own description as input. Design D in such a way that it leads to a contradiction under the assumption that H can decide L_{ACC}. Explain where the contradiction occur and conclude that L_{ACC} is undecidable.

Solution:

Let D be a Turing machine that calls H to determine what M does when the input to M is its own description $\langle M \rangle$. Once D has determined this information, it does the **opposite**.

D = "on input ($\langle M \rangle$):

1: Run H on input ($\langle M, \langle M \rangle \rangle$)
2: **if** H accepts **then**
3: *reject*
4: **else**
5: *accept*"

By assumption, H is a decider for L_{ACC}, which means it should correctly predicts D's behavior. If $H(\langle D, \langle D \rangle \rangle)$ output *accept*, indicating that D accepts its input, then by the design of D, it should reject, which contradicts $H's$ prediction. Similarly, if H outputs *reject*, D should accept, again contradicting $H's$ prediction.

Since $D's$ design ensures that H's prediction is always wrong, H cannot exists as a decider for L_{ACC}. Therefore, there do not exists a decider for L_{ACC} and hence it's undecidable.