Problem 1

Consider the problem

\[ A = \{ \langle M, w \rangle | \text{M moves left at some point during its computation on } w \} \]

Show that \( A \) is decidable.

Problem 1 Solution

We will show that \( A \) is decidable by building a Turing Machine \( M_A \) to decide it: Machine \( M_A \) on input: \( \langle M, w \rangle \)

i. Let \( n = |w|, k = |Q_{M_A}| \).

ii. Simulate \( M \) on \( w \) for \( n + k + 1 \) steps.

- If \( M \) moves left, ACCEPT.
- If \( M \) halts without moving left, REJECT.

iii. We have detected a loop and have not moved left, REJECT.

We first demonstrate that this machine halts on every input: Upon receiving input \( \langle M, w \rangle \), \( M_A \) will simulate \( M \) on \( w \). If \( M \) moves its head left, \( M_A \) will accept immediately. If \( M \) halts on \( w \) without having moved its head left, then \( M_A \) will reject. Since there are only a finite number of states and a finite number of symbols, we know that if \( M \) doesn’t halt on \( w \), it will eventually repeat a combination. If \( M \) runs for \( n + k + 1 \) steps without halting, we know that it must have reached some state/symbol configuration more than once, and so \( M_A \) will reject because we have detected a loop. Thus, \( M_A \) will halt on every input.

We must now show correctness. Suppose \( \langle M, w \rangle \in A \). Then \( M \) moves its head left when computing on \( w \). \( M_A \) will detect this and accept. Suppose \( \langle M, w \rangle \notin A \). Then \( M \) never moves its head left when computing on \( w \). This means that either \( M \) halts on \( w \) without moving its head left, in which case \( M_A \) rejects, or \( M \) neither halts on \( w \) nor moves its head left. \( M_A \) can detect if \( M \) is stuck on one of the tape squares in the input tape, and \( M_A \) can also detect if \( M \) is stuck in an infinite loop in the blank portion of the tape to the right of the input because of the finite number of state/symbol combinations, so in either case, \( M_A \) will detect this event and reject. Thus \( M_A \) decides \( A \).
Problem 2

Let $C$ be a language. Prove that $C$ is Turing-recognizable (i.e. computably enumerable) if and only if a decidable language $D$ exists such that $C = \{ x \mid \text{There exists a } y \text{ such that } \langle x, y \rangle \text{ is in } D \}$

**Problem 2 Solution**

⇒: Assume $C$ is Turing-recognizable. Thus, there exists a recognizer $M_C$ for $C$.

Define a language $D = \{ \langle x, y \rangle \mid M_C \text{ accepts } x \text{ within } y \text{ steps} \}$.

Define a TM $M_D$ the does the following on input $\langle x, y \rangle$:

i. Run $M_C$ on $x$ for $y$ steps.

ii. If it accepts, accept. Else, reject. Note that we can see if a machine did not accept within a fixed number of steps.

$M_D$ is a decider for $D$. It accepts only if $M_C$ accepted $x$ ($x$ is in $C$) and it did so within $y$ steps. It rejects when it did not ($x$ might be in $C$, but it was not accepted within $y$ steps.)

⇐: Assume there exists a decidable language $D$ exists such that $C = \{ x \mid \text{There exists a } y \text{ such that } \langle x, y \rangle \in D \}$

Define a TM $M_C$ the does the following on input $\langle x \rangle$:

i. Choose $y = \gamma$ to be the first possible value for $y$ in a lexicographic ordering.

ii. Run $M_D$ on $\langle x, y \rangle$.

iii. If it accepts, accept. Else, let $y$ be the next value in order and go to step ii.

If $\langle x \rangle$ is in $C$, eventually $M_C$ will find an appropriate value for $y$ and will accept. Otherwise, it will loop. Thus, $M_C$ is a recognizer for $C$. Note that there may be multiple $y$’s, but by definition $x \in C$ if there is some $y$ with $\langle x, y \rangle \in D$, we accept when we find the first one in lexicographic order.

Problem 3

Let $B = \{ \langle M_1 \rangle, \langle M_2 \rangle, \ldots \}$ be a Turing-recognizable language consisting of Turing machine descriptions. Show that there is a decidable language $C$ consisting of Turing machine descriptions such that every machine described in $B$ has an equivalent machine in $C$ and vice versa.
Problem 3 Solution  We prove the existence of a decidable language $C$ by construction.

The proof uses a clever device (aka trick) to tell a decider when to stop. The idea is to record, for each Turing machine description, the number of steps required to accept the description. By doing so, we tell the decider when it is safe to reject a description.

How do we attach a number to a Turing machine? Here’s the gimmick: store the number as the number of states in the machine. That is in the size of $Q$ for that machine. How can we increase the size of $Q$? Easy, just add extra states that the machine cannot get to and go nowhere. These dummy states do not affect the operation of the machine, but they allow the decider to know when it is time to quit and reject. Got it? Now read on...

$B$ is Turing-recognizable, so there is a recognizer, $R_B$ that will accept any $\langle M_i \rangle$ in $B$ in a finite number of steps. Use the notation $n_i$ to represent the number of steps $R_B$ requires to accept $\langle M_i \rangle$. Therefore, for each machine description $\langle M_i \rangle$ in $B$, we have a well-defined integer $n_i$. We can now construct $C$ using this information.

For each machine $M_i$ with description $\langle M_i \rangle$ in $B$, define machine $C_i$ to be the same machine as $M_i$ (symbols, states, transitions) except that it has at least $n_i$ states. If the number of states in $M_i$ is less than $n_i$, then add dummy states to $M_i$ to bring the number of states to $n_i$. Add that new machine, $C_i$ to the new set $C$. By this construction, there is a one-to-one correspondence between the descriptions in $B$ and $C$, and those machines are equivalent.

We now claim that $C$ is decidable. To do so, we create a decider, $D_C$ that takes $\langle M \rangle$ as input:

1. Examine $Q$ and count all dummy states, call this $n$.
2. Remove the dummy states (see below for details) to get $M$.
3. Run the recognizer $R_B$ on $M$ for $n$ steps.
4. If it accepts, accept. Else, reject.

This decider performs two finite operations: counting the number of dummy states and running $M_B$ for a finite number of steps. Therefore, $D_C$ always terminates. Furthermore, $D_C$ will accept strings in $C$ by construction of $C$. Similarly, strings that are not in $C$ will be rejected because $R_B$ will reject or will be stopped without accepting.

Storing Dummy States: How can the translation from $M_i$ to $M'_i$ identify the dummy states? Here is one technique. In the transition table for $M_i$, add a transition from each dummy state to itself. The motion direction and the symbol to write are irrelevant because no transitions will bring you to any of these states. Therefore, this machine will produce the same effect on any input as the un-extended version.
Problem 4

Prove that every infinite Turing-recognizable (i.e. computably enumerable) language C contains an infinite subset D that is decidable.

Problem 4 Solution

Let C be an infinite Turing-recognizable language. Because it is recognizable, it must have some enumerator E.

Idea: We know that if a enumerator enumerates in lexicographic order that the set it is enumerating is decidable. (We know when to accept, we accept when an element is printed, we also know when to reject, we reject when an element larger than the one we are looking for is printed.)

We can assume that C is infinite as we are looking for an infinite subset. (Every finite set is decidable.) Further the problem is only interesting if C is undecidable, if C is decidable and infinite, then C itself is an infinite decidable subset.

Define the following set D constructed in stages:

- Stage 0: Run E and add the first element printed, $x_0$, into D, set variable $last\_element = x_0$.
- Stage i: Continually run E, when an element, $x_i$ is printed, compare it to $last\_element$, if it is lexicographically larger, add it to D and set $last\_element = x_i$

You can also define D to be the set of elements printed by E that are printed in lexicographic order starting with the first element.

We need to show that D is a subset of C, that it is decidable and that it is infinite.

D is a subset of C. Since every element, $x$, in D was printed by the enumerator E for C, we know $x$ is in C.

D is decidable by the following turing machine, $M_D$ on input $x$

1. Run E, for every element printed, compare it to $x$ if the are equal, ACCEPT. If an element is printed that is larger than $x$, REJECT.

If $x$ is in D then it is in C and it will be printed by E. Further if $x$ is in D, then there was a stage when it was the largest printed element, and the above machine will accept that $x$. If $x$ is not in C and not in D eventually E will print a larger number and the machine will REJECT. (C is infinite, so there will eventually be a larger element than $x$ printed.) Similarly, if $x \in C$, but $x \notin D$ then this is because some larger element was printed before $x$ and the above machine will reject.

D is infinite. Assume not, then there is a maximum number in D. Let $max$ be that number. Then we know that enumerator E printed only smaller numbers than $max$ after printing $max$. This means there are only $max$ unique elements printed by E, which implies C is finite, which is a contradiction.

Problem 5

Consider the problem
\[ T = \{ \langle M \rangle | M \text{ is a Turing machine that does not halt on } w\,w \text{ if and only if it accepts } w \} \]

Here \( w\,w \) is \( w \) concatenated with itself. Show that \( T \) is undecidable.

**Problem 5 Solution**

We show \( T \) undecidable by doing a reduction from \( A_{TM} \) to \( T \). Consider this function \( f \) that takes an input \( \langle M, w \rangle \) and outputs a machine \( M'(x) \) defined as follows:

1. If \( x \) is \( w \), Run \( M \) on \( w \). Do what it does. (If it accepts, accept, it it rejects, reject.)
2. If \( x \) is \( w\,w \) LOOP
3. In any other case, reject

Now to show that \( f \) is a many-one reduction from \( A_{TM} \) to \( T \).

**computable**: \( M' \) uses an existing machine \( M \) and does two comparisons.

\[ \langle M, w \rangle \in A_{TM} \Rightarrow M' = f(\langle M, w \rangle) \in T: \text{ The machine } M' \text{ accepts only one value, } w \text{ and loops on only one value } w\,w, \text{ therefore } f(\langle M, w \rangle) \text{ is in } T. \]

\[ \langle M, w \rangle \notin A_{TM} \Rightarrow M' = f(\langle M, w \rangle) \notin T: \text{ If } \langle M, w \rangle \notin A_{TM} \text{ then } M \text{ does not accept } w \text{ therefore } M' \text{ accepts nothing but still loops on } w\,w. \text{ Thus, } M' \text{ is a machine that loops on } w\,w \text{ but does not accept } w, \text{ so } M' \text{ is not in } T. \]

Since \( A_{TM} \) is undecidable, and \( A_{TM} \leq_{m} T \), \( T \) must be undecidable.

**Problem 6**

Show that a language, \( L \), is decidable if and only if \( L \leq_{m} \{1\}^* \).
Problem 6 Solution

⇒: Assume that $L$ is decidable then we need to show that $L \leq_m \{1\}^*$. Let $M_L$ be the decider for $L$. Consider the function $f$ which on input $x$ does the following:

i. Run $M_L$ on $x$
ii. If $M_L$ accepts $x$ output 1.
iii. If $M_L$ rejects $x$, output 0.

Since $M_L$ is a decider it will halt on every input and it will accept if $x \in L$ and it will reject if $x \notin L$. Then on input $x$ if $x$ is in $L$ then $f$ outputs 1 which is in $\{1\}^*$ If $x$ is not in $L$ it outputs 0, which is not in $\{1\}^*$.

⇐: Now assume that $L \leq_m \{1\}^*$. We will show that $L$ is decidable. First notice that $\{1\}^*$ is decidable using the following machine $D$. $D$ on input $x$

i. Read each character of the input if any non-blank character is not a 1, reject
ii. If we reach the end of the input (a blank) and we have only seen 1s then accept.

By assumption we have a reduction from $L$ to $\{1\}^*$, that is a function $f$ where if $x \in L$ then $f(x) \in \{1\}^*$ and if $x \notin L$ then $f(x) \notin \{1\}^*$. Consider machine $Q$ on input $x$.

i. Compute $f(x)$
ii. Run $D$ on input $f(x)$
iii. If $D$ accepts, accept. If $D$ rejects, reject.

Since $D$ is a decider and $f$ is computable the machine $Q$ halts on every input. Claim $Q$ decides $L$. We know that if $x \in L$ then $f(x) \in \{1\}^*$ and $D$ will accept, therefore $Q$ will accept. Also, if $x \notin L$ then $f(x) \notin \{1\}^*$ and $D$ will reject and $Q$ will reject. Therefore $Q$ decides $L$.

Problem 7

Consider the following language

$$L = \{w|w = 0x \text{ for some } x \in K \text{ or } w = 1x \text{ for some } x \in \overline{K}\}$$

Show that neither $L$ nor $\overline{L}$ are recognizable.
Problem 7 Solution

We will show both by doing reductions from $K$ to $L$ and from $\overline{K}$ to $\overline{L}$.

First we do a many one reduction from $K$ to $L$.

Consider the following function $f$: on input $x$, output $1x$.
Notice if $x \in K$ that $1x$ in $L$ by definition. Similarly, if $x \notin K$ then $1x$ is not in $L$. Thus, $f$ is a reduction from $K$ to $L$.
Since $K$ is unrecognizable we know that $L$ must be unrecognizable.

Next we do a many one reduction from $\overline{K}$ to $L$.

Consider the function $g$: on input $x$, output $0x$.
Notice if $x \in \overline{K}$ that $0x$ is not in $L$ and therefore in $L$ by definition. Similarly if $x \notin \overline{K}$, $x \in K$, then $0x$ is in $L$, $x \notin \overline{L}$. Thus $g$ is a reduction from $\overline{K}$ to $\overline{L}$.
Since $\overline{K}$ is unrecognizable we know that $\overline{L}$ must be unrecognizable.