Assignment #7: Problem 1

□ Show that the set of recursively enumerable languages is closed under union.

Let $L_1$ and $L_2$ be two recursively enumerable languages, and $M_1$ and $M_2$ be their accepting Turning machines, respectively.

Let $M_{\text{union}}$ be a TM that comprises $M_1$ and $M_2$ running in parallel. Why do they have to run in parallel?

An input string $w$ is accepted by $M_{\text{union}}$ if it is accepted by either $M_1$ or $M_2$ or both.

$M_{\text{union}}$ is a TM that accepts $L_1 \cup L_2$ and therefore the set of recursively enumerable languages is closed under union.
Show that the set of **recursively enumerable** languages is closed under **intersection**.

- Similar to the proof for union.
- Let $M_{\text{intersect}}$ be a TM that comprises $M_1$ and $M_2$.

An input string $w$ is accepted by $M_{\text{intersect}}$ if it is accepted by both $M_1$ and $M_2$. Since both need to halt and accept, they can run serially.

$M_{\text{intersect}}$ is a TM that accepts $L_1 \cap L_2$ and therefore the set of recursively enumerable languages is closed under intersection.
Show that the set of **recursive** languages is closed under **union** and **intersection**.

- Similar to proofs for recursively enumerable languages, except that we don’t have to run $M_1$ and $M_2$ in parallel – one after the other will do.

- But because $L_1$ and $L_2$ are recursive, we know that their membership TMs $M_1$ and $M_2$ will always halt.

- Therefore, $M_{\text{union}}$ and $M_{\text{intersect}}$ will always halt, and so the set of recursive languages is closed under union and intersection.
Show that the set of recursive languages is closed under reversal.

Let $L$ be a recursive language and $M$ be its membership TM.

Then we can construct an membership TM for $L^R$ that reverses its input string and then calls TM $M$.

Therefore, the set of recursive languages is closed under reversal.
Assignment #7: Problem 4

- Show that language $L$ is recursive if it is accepted by a nondeterministic Turing machine that always halts on any input string.

- Theorem 10.2 of the textbook says that any nondeterministic TM can be simulated by (and is therefore equivalent to) a standard deterministic TM.

- Therefore, if the TM always halts, then $L$ must be recursive.
Assignment #7: Problem 5

Suppose a language $L$ has a function $f$ such that $f(w) = 1$ if $w \in L$ and $f(w) = 0$ otherwise. Show that function $f$ is Turing-computable if and only if the language $L$ is recursive.

- Let $L$ be recursive.
- Then $L$ must have a membership TM $M$ that always halts.
- Therefore, the TM for $f$ simply feeds its input string $w$ into $M$ and outputs $M$’s result as its own.
Suppose a language $L$ has a function $f$ such that $f(w) = 1$ if $w \in L$ and $f(w) = 0$ otherwise. Show that function $f$ is Turing-computable if and only if the language $L$ is recursive.

- Let $f$ be computable.

- Then $f$ has a TM $F$ that for input string $w$ outputs either 1 or 0 depending on whether or not it accepts $w$.

- Therefore, the TM for $L$ simply feeds its input into $F$ and outputs $F$’s result as its own.
Let $D$ be a recursive language of string pairs $<x, y>$. Let $C$ be the set of all strings $x$ for which there exists some $y$ such that $<x, y> \in D$. Show that $C$ is recursively enumerable.

Since $D$ is recursive, it has a membership TM $M_D$ that always halts.

Construct a TM $M_C$ that, for each input string $x$, it can generate all possible strings $y$ in proper order.

For each generated $y$, $M_C$ calls $M_D$ with the pair $<x, y>$.

$M_C$ accepts $x$ if $M_D$ halts and accepts some pair $<x, y>$.

Given $x$, $M_C$ might never find a $y$ such $M_D$ accepts $<x, y>$, and so $M_C$ might not halt.
Assignment #7: Problem 7

- Let $C$ be a recursively enumerable language. Show that there exists a recursive language $D$ of string pairs such that $C$ contains exactly the strings $x$ such that there exists some $y$ such that $<x, y> \in D$.

- Let TM $M_C$ accept $C$. Create a TM $M_D$.
- For each $x \in C$, choose a string $y$ that represents a positive integer.
- $M_D$ simulates $M_C$ on $x$ and lets $M_C$ run at most $y$ steps.
- If $M_C$ accepts $x$ within $y$ steps, then $M_D$ accepts $<x, y>$.
- Therefore, $M_D$ defines the recursive language $D$.

Why limit the number of steps?