CS 383
HW 8 Solutions

1. Alan Turing was interested In modeling computations rather than accepting/rejecting inputs. His TMs had not Accept state. Given an input they either halted (which is good) or ran forever. So let $\mathcal{L}_{halt}$ = {(M,w) | M is a TM that halts (whether or not in a final state) on input w} If you prefer you can write this as {m111w | m is the encoding of a TM that halts on input w}. Show that $\mathcal{L}_{halt}$ is recursively enumerable but not recursive.

   To see that $\mathcal{L}_{halt}$ is recursively enumerable, let M* be a TM that takes as input (M,w) and simulates the computation of M on w. If M ever enters a final state M* accepts (M,w). If M ever enters a situation where it is not in a final state and it has no transition fitting its current state and tape symbol (i.e. M halts not in a final state) the M* accepts (M,w). Otherwise M* keeps running. M* accepts the language $\mathcal{L}_{halt}$.

   To see that $\mathcal{L}_{halt}$ isn't recursive we reduce the universal language to it. Remember that the universal language is {(M,w) | M accepts w} Suppose we had a decider for $\mathcal{L}_{halt}$. Given an (M,w) pair ask if M halts on w; if not reject (M,w). If M halts on w run a simulator of M on w to see what state it halts in. If it halts in a final state accept (M,w) otherwise reject (M,w). This makes a decider for the universal language, which we know can't be.

2. We showed that if a language and its complement are both RE then both are recursive. Suppose we have 3 recursively enumerable languages that are disjoint (no string is in two of them) and whose union is the set of all strings. Show that all three must be recursive.

   One way to do this is to say that the union of two recursively enumerable languages is recursively enumerable, so the complement of each language, which is the union of the other two, must be RE. We know that if both a language and its complements are RE then the language must be recursive.

   Here's a more constructive solution. We start with languages L1, L2, and L3 and TMs M1, M2, and M3 that accept them. Build a 4-tape TM M* with input w on Tape 1, and tapes 2,3,and 4 simulating the tapes of M2, M3, and M4. The states of M* will be triples consisting of a state of M1, a state of M2 and a state of M3. Given an input M, M* runs simultaneous simulations of the computations of M1, M2, and M3 on w. If any of the TMs enter a final state M* halts. Every string is in one of the three languages so M* always halts. We could build 3 versions of M*. Version 1 accepts w if M* halts in and accept state for M1, and rejects w if M* halts in an accept state for M2 or M3. This

is a decider for L1. The other two versions give deciders for L2 and L3.

3. Suppose $\mathcal{L}_1$ and $\mathcal{L}_2$ are both recursively enumerable. Is the concatenation $\mathcal{L}_1\mathcal{L}_2$ RE? Why or why not?

   Yes. Suppose $\mathcal{L}_1$ and $\mathcal{L}_2$ are accepted by M1 and M2. Given a string w with |w|=n, we need to run n+1 simultaneous pairs of simulations: (ε on M1 and w on M2), (w[0] on M1 and w[1:] on M2), (w[0:2] on M1 and w[2:] on M2), etc. If any pair of simulations ever halts and both computations accept, our simulator halts and accepts w. Alternatively, use a nondetermi

4. We know $\mathcal{L}_{ne}$ is recursively enumerable but not recursive. Let $\mathcal{L}_{2ne}$ be {m | m encodes TM that accepts at least 2 strings} Rice's Theorem says $\mathcal{L}_{2ne}$ is not recursive. Is it recursively enumerable? Why or why not?

   Sure. Let N be a non-deterministic TM that, given an encoding m, writes two arbitrary strings after it, so we have (m,w1,w2). N then simulates the computation of m on w1 and the computation of m on w2. If both halt and accept N enters an accept state. This non-deterministic TM accepts $\mathcal{L}_{2ne}$ and we know that any language accepted by a non-deterministic TM is also accepted by a deterministic TM.

5. Let $\mathcal{L}_{inf}$ be {m | m encodes a TM that accepts infinitely many strings}. Is $\mathcal{L}_{inf}$ RE?

   I thought we should go out on a hard problem. Neither $\mathcal{L}_{inf}$ nor its complement are RE. To show $\mathcal{L}_{inf}$ is not RE we reduce the complement of $\mathcal{L}_{halt}$ (which we know from Q1 isn't RE) to it. Given an (M,w) pair build M': For input x, M' simulates the computation of M on w for |x| steps. If M is still running (i.e, after |x| steps M is at a state-tape combination for which there is a valid transition) then M' accepts x.

   If M halts on w after N steps, then M' does not accept any x with |x| > N. This means the language accepted by M' is finite.
   If M does not halt on w then M' accepts all strings, so the language accepted by M' is infinite.

   A recognizer for $\mathcal{L}_{inf}$ would allow us to recognize if M does not halt on w, so that would accept the complement of the halting language.

   You can construct almost the same argument around the complement of the "universal language" {(M,w) | M accepts w} We showed the universal language is RE but not recursive, so its complement isn't RE.

6. Let $\mathcal{L}_{\text{hippy-dippy}}$ be the set of encodings of Turing Machines that accept all strings. Our friend Happy (actually, his encoding) is a member of $\mathcal{L}_{\text{hippy-dippy}}$. The complement of $\mathcal{L}_{\text{hippy-dippy}}$ is $\mathcal{L}_{\text{skeptical}}$, the set of Turing Machines that fail to accept at least one string. Rice's Theorem tells us that neither of these sets is Recursive.
    a. Prove that $\mathcal{L}_{\text{hippy-dippy}}$ is not Recursively Enumerable. You might try reducing the complement of the halting langue to $\mathcal{L}_{\text{hippy-dippy}}$.
    b. Either prove that $\mathcal{L}_{\text{skeptical}}$ is Recursively Enumerable or prove it isn't.

Neither of these is recursively enumerable.

a) We can reduce the complement of the halting language to $\mathcal{L}_{\text{hippy-dippy}}$. to see this, start with an (M, w) pair. Create a new Turing Machine M' that runs on input x as follows: M' simulates M on w for |x| steps. If M halts on w within |x| steps, M' rejects x; otherwise M' accepts x. M does not halt on w exactly when M' is in $\mathcal{L}_{\text{hippy-dippy}}$. So a TM that recognizes the latter would recognize the complement of the halting language, and we know the latter is not recursively enumerable.

b) This one is easier. We reduce the complement of the universal language to this. Given an (M, w) pair create a new Turing Machine M'. For any input s, M' accepts s if s is not w, and M' simulates M on w if s is w. M' accepts all strings if an only if M accepts w. So if we could recognize if M' is in $\mathcal{L}_{\text{skeptical}}$ then we can recognize that (M, w) is in the complement of the universal language, which we know we can't do.