Logic and Computability
Homework 10: Reductions and Decidability

Please read Handout 5 on reductions before getting started. It demonstrates what's required in these proofs.

Challenge 1: Proving that Languages are Undecidable [40 Points]

Prove that each of the following languages is not decidable (aka not "recursive") by describing reductions from the Halting Problem.

1. $L_{42} = \{\langle M \rangle \mid |\mathrm{L}(M)| = 42\}$. That is, M accepts exactly 42 different inputs.
2. $L_{union} = \{\langle M_1, M_2, M_3 \rangle \mid \mathrm{L}(M_1) \cup \mathrm{L}(M_2) = \mathrm{L}(M_3)\}$.

Answer 1:

1. We show that L is undecideable by a reduction from L_{halt}. By wasy of contradiction, assume that L is decideable. Then, we construct a decider for L_{halt}, M_h as follows: Let M_{42} be our built decider for L. Our input will be $< M >, w$, and when M_h is given the input, we will first feed that input to a machine called M'-builder that produces the encoding of a new TM, M'. Let x denote the input to M'. The TM M' has states that move the tape head to a blank region of the tape (either to the left or right of the string), and then writes $< M >, w$ on the tape. Next M' simulates the execution of M on w. If the simulation of M halts on w, then M' moves its tape head back to the front of the string x and then accepts x if x is the string in $L(M)$ such that $|L(M)| = 42$. $< M' >$ is then given to our decider M_{42}. We then wire the acceptance for M_{42} to our acceptance for acceptance from M_h and our rejection to the rejectin for M_h. This concludes the construction of M'. We do not run M', we only construct its TM code. Now, we take that code for $< M' >$ and feed it to the decider for L. If the decider for L_{42} accepts $< M' >$, then M_h accepts its own input $< M >, w$ and otherwise rejects it.

   Now, we prove that M_h is indeed a halt checker and thus a decider for L_{halt}. Notice that $L(M')$ is either the empty set or L_{42}. Therefore, if the decider for L accepts M' then it must mean that $L(M')$ is L_{42} and thus M halts on w or if the decider for L rejects M', then $L(M') \neq L_{42}$ and $L(M') = \emptyset$. Therefore, we have shown that a decider for L allows us to construct a decider for L_{halt} which is a contradiction.

2. We show that L is undecideable by a reduction from L_{halt}. By way of contradiction, assume that L is decideable. Then, we construct a decider for L_{halt}, M_h as follows: Let M_{union} be our built decider for L. Our input will be $< M >, w$ and when M_h is given the input, we will first feed that input to a machine called M'-builder that produces the encoding of a new TM, M'. The TM, M' has states that move the tape head to a blank region of the tape, and then writes $< M >, w$ on the tape. Next, M' simulates the execution of M on w being careful to stay away from x by moving x whenever the simulation of M on w would touch it. If the simulation of M halts on w then M' moves its tape head back to the front of the string and then M' accepts x. Therefore, it follows that $L(M') = \Sigma^*$ if M halts on w or $\emptyset$ if M does not halt on w.

   Now, let $M' = M_1$, and let M_2 be a TM that rejects all string that are inputed into it and let M_3 be a TM that accepts all inputed strings. Then, we can construct a tape of TM codes for $< M_1 >, < M_2 >, < M_3 >$ to send to our decider M_{union} to decide whether or not to accept, and let if M_{union} accepts the input, then M_h accepts our input as a whole, rejecting if not.

   Now, consider that this decies the halting problem as it follows that $L_(M_1)$ must be Σ^* since $L(M_1) \cup L(M_2) = L(M_3)$ since $L(M_3) = \Sigma^*$ and $L(M_2) = \emptyset$, which further implies that M_h halts on our input $< M >, w$. However, we can see that at the same time, if $L(M_1) \cup L(M_2) \neq L(M_3)$, then it follows that we subsequently know that therefore, $L(M_1) \neq \Sigma^*$ which subsequently means that M_1 does not halt on w. Therefore, we have shown that a decider for L allows us to construct a decider for L_{halt} which is a contradiction.

Challenge 2: Virus Detection is Undecidable! [60 Points]

You've been hired as a summer intern at Macrosoft, a leading software company. You and your team have been tasked to write a program that detects the notorious "42 virus." This virus attacks its computer by placing the binary representation of the number 42 somewhere in the computer's memory. It's known that if this binary string ever appears in memory, the processor will superheat and melt to a smelly liquid mush. The task is to write a virus detector program V that takes any program M as input and determines whether or not M contains the virus (that is, writes the number 42 in memory).

After months of effort and many failed attempts to build the virus detector, your team asks you for help. You look at the problem and suspect that it's impossible to solve. Your task is now to prove it. To do so, let's abstract this to a Turing machine problem so that we can be more precise and rigorous. That is, show that the following language is undecidable:

$$L_{virus} = \{\langle M\rangle \mid \exists x \text{ s.t. on input } x, M \text{ writes 42 in binary on its tape during its computation.}\}$$

Notice that we don't care if M runs forever or not. We just want to know if there is some input which causes it to write the virus on the tape at any time during its computation. Prove that L_{virus} is undecidable by describing a reduction from the Halting Problem.

Note: This result implies that a perfect virus detector cannot exist! That doesn't mean that we can't write a virus detector that detects *some* occurrences of a virus in *some* programs, but it does imply that detecting *every* possible occurrence of a given virus in *any* program is impossible.

Answer 2:

We show that L is undecidable by a reduction from L_{halt}. By way of contradiction, assume that L is decideable. Then, we construct a decider for L_{halt} as follows: Let M_{virus} be our built decider for L. Our input will be $< M >, w$ and when M_h is given the input, we will first feed the input to a machine called M'-builder that produces the encoding of a new TM, M'. The TM, M' has states that move the tape head to a blank region of the tape and then writes $< M >, w$ on the tape, except for instead of writing down each 0 or 1, which might give us the binary encoding of 42, we instead have the bijection, $f(0) = a$ and $f(1) = b$ so as to encode the meaning of the TM instead with different symbols instead of 42 (since the binary 42 is what gives us the issue and we can simply construct a TM, M' that executes the same simulation, just using a instead of 0 and b instead of 1 for each of our transition functions [changing the alphabet]). Next M' simulates the execution of M on w. If the simulation of M halts on w, then it follows that we have found the virus and thus we accept the input, rejecting otherwise. We can then wire our acceptance for M_{virus} to our acceptance to acceptance for M_h and our rejection for M_{virus} to rejection for M_h.

Now, consider that this decides the halting problem, as it follows that the language of M' is L if M_{virus} accepts our input, and $\emptyset$ otherwise. Therefore, we have shown that a decider for L allows us to construct a decider for L_{halt} which is a contradiction.