Ryan Luo
CS81: Pomona College
Assignment 12
6 May 2014

All of the problems on this assignment are to be solved using explanations composed of clear English sentences. Appeals to the Church-Turing thesis are allowed. Writing such solutions is not easy; it takes some effort. Please take the time to make your explanations precise, elegant, and as brief as possible.

1 We can encode finite state machines using strings from the alphabet $\{\mathsf{a}, \mathsf{q}, \mathsf{h}, 0, 1\}$ in much the same way as we encoded Turing machines. Let $\text{FINITE}_{\text{DFSM}}$ be the language of descriptions $\langle M \rangle$ of DFSM's M such that $L(M)$ is finite. Prove that $\text{FINITE}_{\text{DFSM}}$ is decidable.

Hint: Compare Problem 7 on Assignment 5. The argument given there for PDA's can be adapted to finite state machines.

Let $< M >$ be a description of a finite state machine. M has a finite number of states k. The problem of whether the language decided by M is infinite is the same problem as whether the language accepts any strings w such that $|w| > k$, because by the pidgeon hole theorem, in computation at least one state must have been visited more than once, and so there exists a loop in the DFSM and so the language accepted by the machine is infinite. We can place an upper bound on the length of words that must be checked by noting that a word that loops twice through a machine can be at most $2 * |k| - 1$ characters long. So if M accepts a word such that $k \leq |w| \leq 2 * k - 1$, M is infinite.

This computation can be carried out by a turing machine by simulating M on all possible strings $s \in \Sigma^*$ such that $k \leq |s| \leq 2 * k - 1$. If a word is accepted by the DFSM, reject. If no word is accepted, accept. We have defined a decidable algorithm that halts on all inputs, and so the Church-Turing thesis states we can build a turing machine $\text{FINITE}_{\text{DFSM}}$ to compute this. This turing machine is decidable, and so $\text{FINITE}_{\text{DFSM}}$ is decidable. ∎

2 Is it decidable for a Turing machine M whether $L(M) = L(M)^R$? Or more precisely, is the language $\{\langle M\rangle \mid L(M) = L(M)^R\}$ decidable? Give a proof.

We can use Rice's theorem. Let P be defined on the set of languages that are equal to their reversal. P is true if $L(M) = L(M)^R$ and false otherwise.
The domain of P is the set of SD languages, since it is defined on the set of languages accepted by a Turing Machine.
P is nontrivial: P(a*) is true, P(a*b) is false.
Therefore, the language $\{\langle M\rangle \mid L(M) = L(M)^R\}$ is not decidable.

A more colloquial thought process (extra, not part of the solution):
The language $\{\langle M\rangle \mid L(M) = L(M)^R\}$ is not decidable. To check whether two languages accepted by turing machines are equal, a turing machine has to enumerate all strings that are accepted by both languages. This enumeration process is not decidable, as a language can include an infinite number of strings. Because this process of deciding language equality using turing machines is not decidable, the language $\{\langle M\rangle \mid L(M) = L(M)^R\}$ is not decidable. ∎

3 One of the following sets is semidecidable, and the other is not. Which is which? Give proof for both.

a. $\{\langle M \rangle \mid L(M) \text{ contains at least } 42^{47} \text{ elements}\}$

b. $\{\langle M \rangle \mid L(M) \text{ contains fewer than } 42^{47} \text{ elements}\}$

Hints: Recall that a set is decidable if and only if both it and its complement are semidecidable. Both languages qualify as "non-trivial" properties of Turing machines.

a. Semi-decidable. We can have an enumeration machine U that enumerates all strings that are accepted by the turing machine M. We use states to count how many we've seen, and once we've enumerated 42^{47} elements, we terminate. So we halt if we accept, or have seen 42^{47} elements, and if we haven't enumerated 42^{47} elements, we do not halt, but continue to search. This falls under the definition of semi-decidability.

We can continue by showing that a) not decidable by using Rice's Theorem:
Define P on the set of languages accepted by some Turing Machine. Let P be true if $L(M)$ has at least 42^{47} elements and false otherwise. The domain of P is the set of SD languages because P is defined on the set of languages accepted by Turing Machines.
P is nontrivial, as $P(a)$ is false and $P(a^*)$ is true.
So by Rice's Theorem, $\{\langle M \rangle \mid L(M) \text{ contains at least } 42^{47} \text{ elements}\}$ is not decidable. b. Not decidable because part (a) is the complement of part (b) and because part (a) is semi-decidable, part(b) cannot be semi-decidable (see previous homework solution).

Additional stuff: This language is essentially asking the following question: is the number of strings machine M accepts finite? The number 42^{47} is not important in this case; in order to definitively determine that there is fewer than a particular number of strings that are accepted by the language, the enumeration machine must enumerate all possible strings. Since the number of possible strings that can be created from an alphabet is infinite, there is no guarantee this machine will halt. The only time this machine will halt is if it has already enumerated 42^{47} elements, in which case we halt when we reject, not when we accept. This is also the complement of our previous language, and since we have proved that (a) is semi-decidable, this must be not decidable (if (b) were semi-decidable, then a and b would both be decidable). ∎

4 Let L be a non-empty semidecidable language. We know that there is a Turing machine M whose range is L. Show that there is a (possibly different) Turing machine which halts on all inputs and whose range is L.

Hint: Consider a machine that takes as input a pair (w, x). It simulates M on w for at most $|x|$ steps. If the simulation of M halts, its output is the output of the new machine. Otherwise the output after $|x|$ steps is some fixed element of L.

Let S be a machine that takes as input a pair (w, x) and simulates M on w for $|x|$ steps. Let R be an indexed infinite set, or list, of L. S simulates M on w for $|x|$ steps, and if the simulation halts, it outputs what is left on the tape, which is what M would output. If the simulation does not halt after $|x|$ steps, then output $R[|x|]$, or the $|x|$th element of R. S is a different machine from M that shares a range with M.
In this way, S is a machine that will have the same range as M, as if M does not halt after $|x|$ steps, S will output an existing element of L. ∎

5 Let $\text{TOTAL}_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a Turing machine that halts on } \textit{all} \text{ inputs}\}$. In class we showed that TOTAL_{TM} is not decidable. Prove that TOTAL_{TM} is not semi-decidable.

Hint: Use the result of the previous problem to obtain a *total* Turing machine whose range is that language. Then diagonalize to obtain a contradiction.

Assume TOTAL_{TM} is semi-decidable. We use the following theorem: The language $\neg H$ is undecidable.

We define a mapping reduction R from $\neg H$ to TOTAL_{TM}:
R takes a machine description and input $< M, w >$ and constructs and returns $< M\# >$ which, upon input x, puts x on a second tape, erases the tape, writes w on the tape, runs M on w for $|x|$ steps or until it halts. If M halts, then $M\#$ loops infinitely, if M does not halt, $M\#$ halts.
TOTAL_{TM} takes the machine description $< M\# >$ outputted by R, which can be created with a turing machine. In doing so, if the machine that decides TOTAL_{TM} halts, then $M\#$ halts on all inputs. If $M\#$ halts on all inputs, then M does not halt on w.
If the machine that decides TOTAL_{TM} does not halt, then $M\#$ does not halt on all inputs. This implies that there was an input x into $M\#$ such that M halted on w after $|x|$ steps, and so M does halt on w.
If TOTAL_{TM} is semi-decidable, then it can semi-decide the language $\neg H$ through the examples above. We reach a contradiction: $\neg H$ is undecidable. Therefore, our hypothesis was incorrect, and the language TOTAL_{TM} is undecidable.

■

6 Recall that a language L_0 is *Turing reducible* to a language L_1 if there is a computable function f with the property that $w \in L_0$ if and only if $f(w) \in L_1$. (There are many other kinds of reducibility. One is *polynomial time* reducibility, in which the function is not just computable but computable in polynomial time.)

a. Show that H_{TM} is reducible to TOTAL_{TM}.

b. Show that any semidecidable language is reducible to H_{TM}.

a. Define a Reduction R that takes $< M, w >$ and creates a description of a machine $< M\# >$ that does the following:
1) Erase the tape
2) Write w on the tape
3) Run M on w

Note that $M\#$ will ignore all inputs and simply simulate M on input w.
Create a turing machine that will output $< M\# >$ upon input $< M, w >$ (represented by R).
The machine that semi-decides TOTAL_{TM} can take $< M\# >$ as input. Since $M\#$ ignores its own input, it will be simulated on multiple inputs by the machine that semi-decides TOTAL_{TM} and halt if M halts on w and not halt if M does not halt on w. We have reduced H_{TM} to TOTAL_{TM}.

b. Let L be a semi-decidable language.
Define a reduction R that takes w and creates an output $< M, w >$. R is a turing machine that concatenates w to a description of a turing machine M that semi-decides L.
A machine that semi-decides H_{TM} can take the input $< M, w >$ and simulate M on input w, effectively simulating a machine that semi-decides languages L.

We have chosen L arbitrarily, and therefore, every semi-decidable language can be reduced to H_{TM}.

■

7

a. Suppose that L_0 is reducible to L_1. Prove that if L_1 is semidecidable, then L_0 is also semidecidable.

b. Use the result of part a and Problem to show that the complement of TOTAL_{TM} is not semidecidable.

Comment: We now see that the language TOTAL_{TM} is rather complex set. Neither it nor its complement is computable. Can you think of other examples of similarly rich and complex sets.

a. L_0 is reducible to L_1. Then, there is a computable function f with the property that $w \in L_0$ if and only if $f(w) \in L_1$. We know that L_1 is semi-decidable, so a turing machine that decides L_1 halts if $f(w) \in L_1$ and does not halt otherwise. Since each w is mapped to a specific $f(w) \in L_1$, the machine that decides L_0 will halt on w if the machine that decides L_1 halts on $f(w)$.
Therefore, if L_1 is semidecidable and L_1 is reduced from L_0,L_0 is semidecidable.
b. From 6a, we have that H_{TM} can be reduced to TOTAL_{TM}. Then, by the definition of reducability, $\neg H_{\text{TM}}$ can be reduced to $\neg\text{TOTAL}_{\text{TM}}$. From 7a, we can derive if L_0 is not semi-decidable, then L_1 is not semi-decidable using the contrapositive. Because $\neg H_{\text{TM}}$ is not semi-decidable, $\neg\text{TOTAL}_{\text{TM}}$ is not semi-decidable. ∎