CMSC 303 Introduction to Theory of Computation, VCU
Assignment: 6
Name: Steven Hernandez

3.d uses http://cseweb.ucsd.edu/classes/sp06/cse105/homework8.pdf as reference
Total marks: 59 marks + 6 marks bonus for typing your solutions in LaTeX.

Unless otherwise noted, the alphabet for all questions below is assumed to be $\Sigma = \{0, 1\}$. This assignment will get you primarily to practice reductions in the context of decidability.

1. [10 marks] We begin with some mathematics regarding uncountability. Let $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ denote the set of natural numbers.

   (a) [5 marks] Prove that the set of integers $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3 \ldots\}$ has the same size as $\mathbb{N}$ by giving a bijection between $\mathbb{Z}$ and $\mathbb{N}$.

   Proof: By bijection, we create a function $f : N \mapsto Z$ such that all even numbers in N map into a some even number in Z and odd numbers in N map to negative numbers in Z.

$$f(x) = \begin{cases} -\dfrac{x+1}{2} & \text{if } x \text{ is odd} \\ \dfrac{x}{2} & \text{if } x \text{ is even} \end{cases}$$

   (b) [5 marks] Let B denote the set of all infinite sequences over $\{0, 1\}$. Show that B is uncountable using a proof by diagonalization.

   Proof: We can prove this by contradiction.

   Let's begin by only concern ourselves with very the large (infinite) strings. We assume $\exists$ list L of all these strings.

$$\begin{aligned} L = & 011001100\ldots \\ & 100110011\ldots \\ & 100100100\ldots \\ & 111111111\ldots \\ & \ldots\ldots\ldots \end{aligned} \tag{1}$$

   We can construct a string $x \in B$ which is not in L by taking the ith symbol of x to be the opposite symbol from the ith entry of L. Thus, contradiction.

2. [9 marks] We next move to a warmup question regarding reductions.

   (a) [2 marks] Intuitively, what does the notation $A \leq B$ mean for problems A and B?

   $A \leq B$ means that B is harder than A (or equally hard).

(b) [2 marks] What is a mapping reduction $A \leq_m B$ from language A to language B? Give both a formal definition, and a brief intuitive explanation in your own words.

A mapping reduction gives us a way to handle deciding whether a problem is decidable or not from knowing that some other problem is decidable or not.

So for example with an example of adding and multiplying. Multiplying $\leq$ adding because multiplying can be achieved by simply using adding. Thus, adding is a more powerful construct.

With this, we can say that if A is not decidable, B is not decidable either because B is harder than A. As well as, if B is decidable, A is also decidable, because A is not as hard as B.

(c) [2 marks] What is a computable function? Give both a formal definition, and a brief intuitive explanation in your own words.

A computable function is a function where $\exists$ a TM T where $\forall w \in \Sigma^*$, T halts with only $f(w)$ on its tape.

This means that there needs to exist a TM for a function to be a computable function. This somewhat implies potentially there are non-computable function meaning there may exist functions which we cannot compute with any TM.

(d) [3 marks] Suppose $A \leq_m B$ for languages A and B. Please answer each of the following with a brief explanation.

i. If B is decidable, is A decidable?
A is decidable because B is 'harder', yet is decidable.

ii. If A is undecidable, is B undecidable?
B is undecidable because B is 'harder' than A and A is not decidable.

iii. If B is undecidable, is A undecidable?
It is unknown whether A is decidable or not, because while B is not decidable, it is also said to be harder than A.

3. [40 marks] Prove using reductions that the following languages are undecidable.

(a) [8 marks] $L = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^*\}$.

Proof: By contradiction. Assume $\exists$ TM R deciding L.

If we can construct TM S to decide A_{TM} with R, this would be a contradiction.

First, given $\langle M, x \rangle$, we want to decide if $\langle M, x \rangle \in A_{TM}$. We will construct a new TM M_x such that

If $x \in L(M)$, then $L(M_x) = \Sigma^*$ and

if $x \notin L(M)$, then $L(M_x) \neq \Sigma^*$.

$$\begin{aligned} M_x = {}& \text{"On input } t \in L(M): \\ & \text{if } t = x, accept \\ & \text{if } t \neq x, \text{run } M \text{ on } x \text{ and accept if } M \text{ does."} \end{aligned} \tag{2}$$

This means if M accepts, $L(M_x) = \Sigma^*$, and if M does not accept, $L(M_x) = \Sigma^* - \{x\}$ thus $L(M_x) \neq \Sigma^*$.

Now we show how if we can create a TM S from R and M_x which decides A_{TM}, we will have a contradiction.

$$\begin{aligned} S = {}& \text{"On input } \langle M, x \rangle \text{ for} A_{TM}: \\ & \text{construct TM } M_x \\ & \text{Run } R \text{ on } \langle M_x \rangle \\ & \text{If R accepts, (this means } L(M_x) = \Sigma^*\text{), accept} \\ & \text{If R rejects, reject."} \end{aligned} \tag{3}$$

(b) [8 marks] $L = \{\langle M \rangle \mid M$ is a TM and $\{000, 111\} \subseteq L(M)\}$.

Proof: By contradiction. Assume $\exists$ TM R deciding L.

If we can construct TM S to decide A_{TM} with R, this would be a contradiction.

First we create a subprocessed TM M_x.

$$\begin{aligned} M_x = \text{"On input} w : \\ &\text{If } w = 000\text{, accept} \\ &\text{Else, run } M \text{ on } w. \\ &\text{If } M \text{ accepts, accept, if } M \text{ rejects, reject.} \end{aligned} \tag{4}$$

$$\begin{aligned} S = \text{"On input } \langle M, x \rangle : \\ &\text{Construct } M_x \\ &\text{Run } R \text{ on } \langle M_x \rangle \\ &\text{If } R \text{ accepts, accept, if } R \text{ rejects, reject.} \end{aligned} \tag{5}$$

(c) [8 marks] $L = \{\langle M \rangle \mid M$ is a TM which accepts all strings of even parity$\}$. (Recall the *parity* of a string $x \in \{0, 1\}$ is the number of 1's in x.)

Proof: By contradiction. Assume $\exists$ TM R deciding L.

If we can construct TM S to decide A_{TM} with R, this would be a contradiction.

First we create a subprocessed TM M_x.

$$\begin{aligned} M_x = \text{"On input} w : \\ &\text{If } w \text{ is of the form } ((0^*)1(0^*)1(0^*))^*\text{, accept} \\ &\text{Else, run } M \text{ on } w. \\ &\text{If } M \text{ accepts, accept, if } M \text{ rejects, reject.} \end{aligned} \tag{6}$$

By step one, this will accept all inputs with the form $((0^*)1(0^*)1(0^*))^*$ which are all of even parity. In fact, this is ALL string of even parity.

$$\begin{aligned} S = \text{"On input } \langle M, x \rangle : \\ &\text{Construct } M_x \\ &\text{Run } R \text{ on } \langle M_x \rangle \\ &\text{If } R \text{ accepts, reject, if } R \text{ rejects, accept.} \end{aligned} \tag{7}$$

If S rejects, this would mean that the $L(M_x) = ((0^*)1(0^*)1(0^*))^*$ (which R accepts), which would mean M_x rejected the input w. On the other hand, S would accept if $L(M_x) = ((0^*)1(0^*)1(0^*))^* + \{w\}$, which R rejects, because it was determined that w is not a string of even parity.

(d) [8 marks] $L = \{\langle M \rangle \mid M$ is a TM that accepts w^R whenever it accepts $w\}$. Recall here that w^R is the string w written in reverse, i.e. $011^R = 110$.

Using http://cseweb.ucsd.edu/classes/sp06/cse105/homework8.pdf as reference:

Proof: By contradiction. Assume $\exists$ TM R deciding L.

If we can construct TM S to decide A_{TM} with R, this would be a contradiction.

We begin by creating a TM M_x which defined as such:

$$M_x = \text{"On input } \langle x \rangle : \\ \quad \text{If } x \text{ is not 01 or 10, reject} \\ \quad \text{If } x \text{ is 01, accept} \\ \quad \text{Otherwise } x = 01\text{, so we run } M \text{ on } w \text{ and accept if } M \text{ accepts.} \tag{8}$$

Note for this above machine, we could use any pairs x and y such that $x = y^R$. $L(M_x) = \{01\}$ if M does not accept input 10 or $L(M_x) = \{01, 10\}$ if M does accept input 10. We can use this to show that building a TM S which uses both R and M_x, we would be able to decide A_{TM}, a contradiction.

$$S = \text{"On input } \langle M, w \rangle : \\ \quad \text{construct TM } M_x \\ \quad \text{Run } R \text{ on } \langle M_x \rangle \\ \quad \text{If R accepts, accept} \\ \quad \text{If R rejects, reject."} \tag{9}$$

(e) [8 marks] Consider the problem of determining whether a TM M on an input w ever attempts to move its head left when its head is on the left-most tape cell. Formulate this problem as a language and show that it is undecidable.

Using http://cseweb.ucsd.edu/classes/sp06/cse105/homework8.pdf (5.12) as reference:

$L = \{\langle M \rangle \mid M$ is a TM that moves its head left when its head is on the left-most tape cell. $\}$

Suppose $\exists$ TM R which decides L. If we can construct a TM S which uses R to decide A_{TM}, contradication. We can create a new TM; Q, with the same attributes as M, with the a few attributes changed.

$S =$ "On input $\langle M \rangle$:

1. Construct TM Q which matches M, with the following changes:

   $Q =$ "On input w:

   Add steps to the beginning which adds a new tape symbol, we can label $\#$ to the first slot

   Then the TM moves the contents of the tape over one slot to the right.

   (It is important to understand, our first step did not overwrite the content of the first slot.

   It can be assumed that both steps happen at the same time without losing any tape contents.)

   Additionally, before the TM normally transitions into an accept state, we have the TM

   move to the left until it reaches $\#$.

   Then, move the head left once.

   Also, if the head ever hits $\#$ during normal operation, the TM should perform as if it was the left most

2. We now run R on Q and return the result.

(10)

Because we removed the possiblity for M to move left when it is on the left most cell in its normal operation by transforming it into Q, we know the only reason This would happen is if M accepts, which would be a contradiction for A_{TM}.