Adam Gincel

*I pledge my honor that I have abided by the Stevens Honor System*

4.1

a) 0100 is accepted

b) 011 is not accepted

c) No. The input only has one component; it’s not in a correct form.

d) No. The first component is not a regular expression, so the input is not in the correct form.

e) M’s language is not empty, so no.

f) M accepts the same language as itself, so yes.

4.2

We define a language L = {<M, R> | M is a DFA and R is a Regular Expression with L(M) = L(R)}

As seen previously, we can make a Turing Machine F that decides the language EQdfa = {<A, B> | A and B are DFAs and L(A) = L(B)}. We can make a TM T that decides L

T = “On input {M, R> where M is a DFA and R is a regular expression

Convert R into a DFA using the algorithm used in the proof of Kleene’s Theorem

Run TM F on <M, R>

If F accepts, accept. Otherwise, reject.

4.3

We know Edfa is decidable, and we can build a Turing Machine E that decides it. We can also construct a Turing Machine R that decides ALLdfa

R = On input <A> where A is a DFA

Construct a DFA B that recognizes the complement of L(A), by simply swapping accept/reject states

Run Turing Machine E on input <B>

If E accepts, accept. Otherwise, reject.

4.7

Say we try to list out every infinitely long binary string in a list. Once we think we have them all, we diagonally digits from each string, taking the first digit from the first string, the second digit from the second string, etc. At the end we would have an infinitely long binary string, and then we take its inverse, toggling each bit. We have now guaranteed we have a string that was not in our set already, even though we thought we had listed them all. This is possible forever, so it is impossible to begin to list every infinitely long binary string. B is uncountably infinite.