Gavin Grob
CS 510 Automata Theory
Final Exam

1. Give a regular expression for the following language where $\Sigma = \{0, 1\}$.
$L_1 = \{s | s$ starts with a 0 and has even length or stats with a 1 and has odd length$\}$
Justify that your regular expression is correct by explaining why all strings generated fit the description and all strings that fit the description are generated.

$(0(0 \cup 1) \cup 1)(00 \cup 01 \cup 10 \cup 11)^*$

I am assuming that this language doesn't accept the empty string, because the discription states that it either starts with a 0 or 1. So the regular expression doesn't accept the empty string. The expression has 2 steps, firsts generates a string starting with a 0 followed by 1 or 0 making it even, or it generates a 1 making it odd. Second it generates 2 more characters(all possible combinations of 0 and 1) keeping the generates string either even or odd accordingly.

2.Give a context free grammar generating the following language where s and t are strings made up of 0?s and 1?s.
$L_2\{s\#t | s^R$ is a substring of $t\}$ where s^R represents the reverse of s.
Justify your grammar is correct by explaining why all strings generated fit the description and all strings that fit the description are generated.

$S \to 1M1F|0M0F|\#F$
$M \to 1M1|0M0|\#F$
$F \to 1F|0F|\epsilon$

The grammar works by $1M1F \to 10M01F \to 10\#F01F$ then F can make any string with 0's and 1's

So It generates s and s^R at the same time, then leaves a F before and after s^R so t is able to be any string that has a substring of s^R
note: that s can also be the empty string, and the s^R of the empty string is the empty string. So it is valid to have the string #010 which it can also generate.

3. Classify the following languages and concisely justify your answers.

(a) $L_1 = \{w | w = 0^n 1^m 2^m 3^n$ for $n, m \geq 1\}$
i. the language is recursive but not context free

ii. the language is context free but not regular
iii. the language is regular

A context free for L_1
$S \to 0S3|M$
$M \to 1M2|\epsilon$

So I know it is at least CFG, we will use the pumping lemma to prove that it is not regular

we have a pumping length p,
$w \in L_1$ $w = 0^p 1^{2p} 2^{2p} 3^p$
This forces y to be 1^i where $1 \leq |xy| \leq p$ since $|xy| \leq n$
We can then unpump so we left with the string $0^{p-i} 1^{2p} 2^{2p} 3^p$ which is not in the language

(b) $L_2 = \{w|$ every prefix of w has more 1?s than 0?s but not more than three more 1?s than 0?s $\}$
i. the language is recursive but not context free
ii. the language is context free but not regular
iii. the language is regular

I will use the CFL pumping lemma to prove L_2 is not context, therefore not regular.

we have a pumping length p = 1
$w \in L_2$ $w = 11$

This then forces v or x to be 1, so then if we pump there will be more than three 1's than 0's.

(c.) $L_3 = \{< M, w > |M$ is a Turing Machine that does not read every character of its input $w\}$
i. recursive
ii. recursively enumerable but not recursive
iii. co-recursively enumerable but not recursive
iv. none of the above

I will prove this by first showing $\overline{L_3}$ is undecidable.

-TM_{accept} = on input (M, w)
- 1.Construct $TM_{transform}$ by modifying M on w

- a. copy w into the input tape (this is what is read when running)
- b. copy all the state of M into $TM_{transform}$
- c. add a case when M goes into the accepting state, $TM_{transform}$ reads all the input.
- 2. run $TM_{read-all}$ on $TM_{transform}$
- 3. if $TM_{read-all}$ rejects then accept, otherwise reject.

However we know that TM_{accept} is undecidable, making $TM_{read-all}$ is undecidable. But like L_u and $\overline{L_3}$ can list out all of the strings in this language making them RE. This then makes L_3 co-recursively enumerable, because L_3 and $\overline{L_3}$ are not RE(which would make them recursive).

4.(b) In the following one player board game you are given an n x n board. On each square of the board lies either a blue chip, a red chip or nothing. You win by removing chips from the board so that each column contains only chips of a single color and each row contains at least one chip. Winning may or may not be possible depending on the initial configuration of the board.
Let Puzzle = $\{< B > | B$ is a winnable starting game board $\}$
Prove that Puzzle is NP-complete by first showing it is in NP and then reducing an NP-complete problem we have studied in class to Puzzle. I suggest you use 3-SAT as your starting point.

Puzzle $\in NP$ because we are able to scan the the columns to remove any opposite color, in less than of equal n time, then scan to make sure there is still there is still a chip in the row int less than of equal n time, making it polynomial time of the input size.

3SAT $\leq_p$ Puzzle

We map our 3SAT input of n variables $x_1...x_n$ and m clauses $c_1...c_m$ we make a game board $B[m, n]$:
1. remove any clauses that contains x_i v $\overline{x_i}$.
2. Label each row with a clause number, 1 through m
3. Label each column with a variable number, 1 through n
4. Generating a board game $B[m, n]$

If x_i is in clause c_j then place a red chip in $B[j, i]$, on the contrary if $\overline{x_i}$ is in clause c_j then place a blue chip in $B[j, i]$.
If x_i is true, then remove the blue chips from the column i, on the contrary if x_i is false, then remove the red chips from the column i.
Each row with one color of chips represents a satisfied clause.
Any solution to the Game can be used to find if the Puzzle B is winnable, finding the color

in each column and setting the corresponding variable to true if the chip is blue and false the chip is red.
If the 3Sat instance is satisfiable then this game is winnable.