Student ID: ______________________________

Collaborators: ______________________________

CS181 Winter 2018 – Problem Set 3

Due Monday, February 12, 11:59 pm

- Please write your student ID **and the names of anyone you collaborated with** in the spaces provided and attach this sheet to the front of your solutions. **Please do not include your name anywhere since the homework will be blind graded.**
- An extra credit of **5%** will be granted to solutions written using LaTeX. Here is one place where you can create LaTeXdocuments for free: `https://www.sharelatex.com/`. The link also has tutorials to get you started. There are several other editors you can use.
- If you are writing solutions by hand, please write your answers in a neat and readable hand-writing.
- Always explain your answers. When a proof is requested, you should provide a rigorous proof.
- If you don't know the answer, write "I don't know" along with a clear explanation of what you tried. For example: "I couldn't figure this out. I think the following is a start, that is correct, but I couldn't figure out what to do next. [[Write down a start to the answer that you are sure makes sense.]] Also, I had the following vague idea, but I couldn't figure out how to make it work. [[Write down vague ideas.]]" At least 20% will be given for such an answer.

  Note that if you write things that do not make any sense, no points will be given.
- The homework is expected to take anywhere between 8 to 14 hours. You are advised to start early.
- Submit your homework online on the course webpage on CCLE. You can also hand it in at the end of any class before the deadline.

Note: *Suggested practice problems from the book: 2.4 and 2.5. Please, do not turn in solutions to problems from the book.*

1.

Theorem 1. If A and B are regular languages, then A ∇ B = $\{xy \,|\, x \in A, y \in B \, and \, \|x\| = \|y\|\}$ is a context-free language

Proof. To start, notice that A ∇ B is just a concatenation of the two languages with an added condition that the length of x be equal to the length of y. Since we are given the fact that A and B are regular languages, we can use the closure properties of regular languages. Thus, A ∇ B is a regular language because of the closure property of concatenation.

Now that we have established A ∇ B is a regular language, we can say that there exists a DFA that accepts the language A ∇ B. Using this fact, we can also claim that there exists a PDA that accepts A ∇ B. This is because we can always convert a DFA to a PDA. To achieve such a conversion, merely keep the same transitions and push and pop nothing onto and off the stack.

Now that we have established that there exists a PDA that accepts A ∇ B, we can use the fact that every PDA can be converted into a Context-Free Grammar (CFG). We proved this in class on 2/5, so I will assume that this is a given. As a result, this means that there is a CFG that represents the language A ∇ B. Since this is true, A ∇ B is a context-free language. □

2a.

Theorem 2a. The language, L$_1$ = $\{x\$y |\, x, y \in \{0, 1\}^* \, and \, x \neq y\}$ over the alphabet Σ = {0,1,\$} is a context-free language.

Proof. We can prove the theorem by coming up with a PDA that accepts this language. If we can achieve this, then there is a CFG that represents the language and thus the language is a context-free language (CFL).

My plan is to build a PDA which consists of a helper PDA. In the first phase, the PDA will be pushing everything in the x portion of x\$y onto the stack. Then when x is exhausted, the helper PDA will step in and read x\$y from the back of the string. It will essentially compare the x and y portions of the input by comparing the current input to the top of the stack and popping off the stack. The PDA will accept if nothing is on the stack and y is not exhausted, if y is exhausted and something is on the stack or if an input y$_i$ $\neq$ x$_i$. Otherwise, the PDA will reject.

I will represent my PDA, M, and the helper PDA, N, with a 6-tuple. Q is the set of states in the PDA. Σ is the input alphabet. Γ is the stack alphabet. δ is the set of transitions in the PDA. q$_0$ is the start state. F is the set of accepting states.

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F) \tag{1}$$

$$N = (Q', \Sigma, \Gamma, \delta', {q_0}', F') \tag{2}$$

My helper PDA N will have these parameters:

$$Q' = \{q_3, q_4\} \tag{3}$$

$$q_0{}' = q_3 \tag{4}$$

$$F' = q_4 \tag{5}$$

The input alphabet and stack alphabet are both {0,1,\$}.

$$\Sigma = \{0, 1, \$\} \tag{6}$$

$$\Gamma = \{0, 1, \$\} \tag{7}$$

Onto the most complex part of our PDA, we have the δ transitions. To start off, the δ transtions will be written in the form of δ(q_i, x, y, z) = q_j where q_i is the current state, x is the current input, y is the top of the stack to be popped, z is the input being pushed onto the stack and q_j is the next state.

The δ' transitions basically represent the process of reading y backwards and matching it with the contents of the stack. If there is any mismatch, then the PDA accepts.

$$\delta'(q_3, 1, 1, \epsilon) = q_3 \tag{8}$$

$$\delta'(q_3, 0, 0, \epsilon) = q_3 \tag{9}$$

$$\delta'(q_3, \$, \$, \epsilon) = q_3 \tag{10}$$

$$\delta'(q_3, 1, 0, \epsilon) = q_4 \tag{11}$$

$$\delta'(q_3, 0, 1, \epsilon) = q_4 \tag{12}$$

$$\delta'(q_3, \$, \{0, 1\}, \epsilon) = q_4 \tag{13}$$

$$\delta'(q_3, \{0, 1\}, \$, \epsilon) = q_4 \tag{14}$$

$$\delta'(q_4, \Sigma, \Sigma, \epsilon) = q_4 \tag{15}$$

Now onto the main PDA M. There are only five states in the PDA, one of which, q_0, is the start state. It also includes two states of the helper PDA N.

$$Q = \{q_0, q_1, q_2\} \cup Q' \tag{16}$$

The PDA starts at q_0, the start state. It has an ϵ transition to q_1. Once at q_1, the PDA will keep looping to the current state and pushing x onto the stack until the input x is exhausted. Once this happens and the next input is \$, the PDA transitions to q_2 and then epsilon transitions to the helper PDA N. The δ transitions are like so:

$$\delta(q_0, \epsilon, \epsilon, \$) = q_1 \tag{17}$$

$$\delta(q_1, 1, \epsilon, 1) = q_1 \tag{18}$$

$$\delta(q_1, 0, \epsilon, 0) = q_1 \tag{19}$$

$$\delta(q_1, \$, \epsilon, \epsilon) = q_2 \tag{20}$$

$$\delta(q_2, \epsilon, \epsilon, \epsilon) = q_3 \tag{21}$$

The start state is trivially q_0 while the accepting state is the same as our helper PDA's accepting state.

$$q_0 = q_0 \tag{22}$$

$$F = F' \tag{23}$$

Our PDA works because it will check the reverse of the string y to the contents of the stack, which stores x in reverse. Now that there is a PDA constructed for L_1, we can say that there is a CFG that represents L_1 since all PDAs can be converted to a CFG. If there is a CFG for L_1, then it must be context-free (CFL). □

2b.

Theorem 2b. The language, $L_2 = \{xy|\, x, y \in \{0,1\}^*, |x| = |y|,\, and\, x \neq y\}$ is a context-free language.

Proof. My plan for the proof is to make a grammar that generates the language, L_2. The grammar will have at least one difference between the first half of xy and the second half of xy, so that it is guaranteed that x ≠ y. The rest of the grammar can be arbitrary. Once we achieve a grammar for L_2, then we will have shown that L_2 is a context-free language (CFL).

Let us call our grammar G. G features an alphabet Σ = {0,1}. Our grammar G has the following rules:

$$S \rightarrow XY|YX \tag{24}$$

$$X \rightarrow 0|0X0|0X1|1X0|1X1 \tag{25}$$

$$Y \rightarrow 1|0Y0|0Y1|1Y0|1Y1 \tag{26}$$

Using our grammar G, we can see that it generates the string $a_1xa_2b_1yb_2$ where $|a_1| = |a_2|$, $|b_1| = |b_2|$ and x ≠ y. To complete the proof, we can use the fact that a_2 and b_1 are interchangeable. This is due to the fact that both strings are generated independently of the rest of the string. Therefore, we have the following equivalence:

$$a_1xa_2b_1yb_2 = a_1xb_1a_2yb_2 \tag{27}$$

Now, notice that x and y both maintain their original positions within the string. This is due to the fact that we simply flipped the positions of a_2 and b_1. This proves that our grammar is context-free. Our grammar also generates strings that satisfy the conditions that |x| = |y| and x ≠ y. Therefore, our grammar G is a CFG that generates L_2. This proves that L_2 is a CFL. □