CSE355 Homework Four Sample Solutions

The presidential election is underway![1] There are four candidates, Aaron, Beatrice, Charlie, and Deedee. Because we are friends, we call them `a`,`b`,`c`,`d`, respectively.

A candidate wins by *majority* if they receive more than half the votes cast. They win by *plurality* if they receive more votes than any other candidate.

At Elections Central on Election night, we will receive one string over $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^\star$ giving all of the votes.

Let $L_M = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^\star : \text{Charlie wins by majority}\}$.

Let $L_P = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^\star : \text{Charlie wins by plurality}\}$.

Let $L_W = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^\star : \text{Charlie wins by majority } \textbf{and} \text{ Beatrice beats Aaron}\}$.

Question 1

Is L_P regular? context-free but not regular? or not context-free? Justify your answer carefully. Suppose (to the contrary) L_P were context-free. Then, there is a pumping constant p for L_P. Choose the string $w = \mathtt{c}^{p+1}\mathtt{a}^p\mathtt{b}^p\mathtt{d}^p$. Note that $w \in L_P$ (because Charlie has strictly more votes than all other candidates) and $|w| \geq p$. We want to show that no matter how w is written as $uvxyz$, there is a value of i for which $uv^ixy^iz \notin L_P$. Note that since $|vxy| \leq p$, there are only at most 2 distinct characters in the string vxy. Therefore, we can split this into cases:

- Case 1: v and y both consist of the same symbol. Then, if the symbol is `a`, `b`, or `d`, we can pump up once to guarantee that symbol has at least the number of occurrences as `c`, and therefore Charlie does not win, showing that $uv^2xy^2z \notin L$. If the symbol is `c`, then we can pump down to get the string $\mathtt{c}^\alpha\mathtt{a}^p\mathtt{b}^p\mathtt{d}^p$ with $\alpha \leq p$. Therefore, Charlie does not win in this case either, showing that $uxz \notin L$.
- Case 2: v and y individually only consist of one single type of symbol, but the symbol each consists of is not the same. For example, v consists entirely of `c`'s but y consists entirely of `a`'s. Here, we only have 3 sub-cases: `c`'s vs. `a`'s, `a`'s vs. `b`'s, and `b`'s vs. `d`'s. For the second and third cases, even though v or y may be empty, we can always pump up to get some "section" (that is not `c`) to have length $\geq p + 1$, showing that Charlie is not the winner. However, for the first case, v or y being empty is important. Therefore, we split this into the following sub-cases:
  - Case 2a: `c`'s vs. `a`'s, $v = \epsilon$, and $y \neq \epsilon$. Then, we can just pump up once to get at least as many `a`'s as `c`'s.
  - Case 2b: `c`'s vs. `a`'s, $v \neq \epsilon$, and $y = \epsilon$. Then, we can just pump down once to get at most as many `c`'s as `a`'s.
  - Case 2c: `c`'s vs. `a`'s, $v \neq \epsilon$, and $y \neq \epsilon$. Then, if we pump down, then we will remove at least one character from the set of `c`'s, and therefore have at most p `c`'s, and since we did not modify the `b` or `d` sections, Charlie is not the winner.
- Case 3: v or y consist of two different symbols (Note: it cannot be the case that *both* v and y consist of two different symbols). For example, v can contain the substring `ca`. There are sub-cases to consider:
  - Case 3a: v or y contains the substring `ab` or `bd`. Then we can just pump up once to get at least one more vote for two non-Charlie candidates, making Charlie not the winner.

[1] Vote if you can and have not already done so.

– Case 3b: v or y contains the substring `ca`. Just like Case 2c, we can pump down and get at most the number of `b`'s and `d`'s, making Charlie not the winner.

These are all the cases, and a contradiction was reached in each case. Therefore, L_P is not context-free.

Question 2a

Show that L_M is not regular.

Suppose (to the contrary) that L_M is regular. Then there exists a pumping length p. We choose the string $w = \mathtt{c}^{p+1}\mathtt{d}^p$, for which the pumping lemma gives a way to decompose w into uvx. Since $|uv| \leq p$, we know that v will consist entirely of `c`'s, and because $|v| \geq 1$, it will contain at least one `c`. We pump down to obtain $uv^0x = ux$, which will contain at least one fewer `c` but the same number of `d`'s. Then `c` is no longer in the majority, so the string is not in L_M, a contradiction. Therefore, L_M is not regular.

Question 2b

Give a formal description of a PDA whose language is L_M.

The PDA is given by the 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$, $\Gamma = \{\$, +, -\}$, $F = \{q_3\}$, and δ is as follows:

$$
\begin{aligned}
\delta(q_0, \epsilon, \epsilon) &= \{(q_1, \$)\} \\
\delta(q_1, \epsilon, +) &= \{(q_2, \epsilon)\} \\
\delta(q_1, \sigma, \epsilon) &= \{(q_1, -)\} \quad (\sigma \in \{\mathtt{a}, \mathtt{b}, \mathtt{d}\}) \\
\delta(q_1, \sigma, +) &= \{(q_1, \epsilon)\} \quad (\sigma \in \{\mathtt{a}, \mathtt{b}, \mathtt{d}\}) \\
\delta(q_1, \mathtt{c}, \epsilon) &= \{(q_1, +)\} \\
\delta(q_1, \mathtt{c}, -) &= \{(q_1, \epsilon)\} \\
\delta(q_2, \epsilon, \$) &= \{(q_3, \epsilon)\} \\
\delta(q_2, \epsilon, +) &= \{(q_2, \epsilon)\} \\
\delta(q, \sigma, \gamma) &= \emptyset \quad \text{(all other cases)}
\end{aligned}
$$

In q_0 we push a \$ to mark the bottom of the stack, which we will need later. In q_1 we push a $+$ or pop a $-$ when seeing a `c`, and we push a $-$ or pop a $+$ when seeing an `a`, `b`, or `d`, so that the total number of $+$'s minus the total number of $-$'s present on the stack represents the number of votes by which `c` is in the majority. We can only transition to q_2 if we see a $+$, so there must be at least one extra vote for `c`. Then q_2 pops $+$'s (additional votes beyond the required majority for `c`) before finally transitioning to q_3 and accepting once the stack is again empty.

The only way to reach q_3 is for there to be just $+$'s on the stack when we reach q_2, which can in turn only happen if `c` received more votes than the rest of the candidates together, as required. Conversely, if we always choose pushing or popping so that the stack never contains both $+$ and $-$, then we will always accept when `c` has a majority, because in that case there will be at least one $+$ left on the stack at the end of the input to permit the transition to q_2 and then q_3.

Question 3a

Give a context-free grammar G whose language is $\overline{L_M}$. This is equivalent to saying that Charlie does not win by majority (i.e., gets either exactly half or fewer than half of the votes). So the total number of occurrences of `c` is at most $\lfloor \frac{|w|}{2} \rfloor$. We can instead think of the language $L = \{w \in \{0,1\}^\star : \text{the number of 0's in } w \text{ is at most the number of 1's}\}$, and have `a, b, d` all be encoded as a 1. Then for each occurrence of 1 on the RHS of any rule, we can add 3 rules, which correspond to each of `a, b, d`. Here is a CFG for L (S is the

start variable):

$$S \to AS \mid 1A \mid 1S$$
$$A \to AA \mid 0A1 \mid 1A0 \mid \epsilon$$

This grammar is correct because A produces strings that have the same number of 1's and 0's. If w contains more 1's than 0's, then it is either of the form (1) $1x$ with x having more 1's than 0's, (2) $1x$ where x has the same number of 1's and 0's, or (3) xy where x has an equal number of 1's and 0's, and y has more 1's than 0's. Therefore, the rules of S handle these three cases.

So we then build a grammar for $\overline{L_M}$. Each `c` is matched by a `a, b, d`, and there may be some `a, b, d` occurrences that are unmatched.

$$S \to AS \mid \mathtt{a}A \mid \mathtt{a}S \mid \mathtt{b}A \mid \mathtt{b}S \mid \mathtt{d}A \mid \mathtt{d}S$$
$$A \to AA \mid \mathtt{c}A\mathtt{a} \mid \mathtt{a}A\mathtt{c} \mid \mathtt{c}A\mathtt{b} \mid \mathtt{b}A\mathtt{c} \mid \mathtt{c}A\mathtt{d} \mid \mathtt{d}A\mathtt{c} \mid \epsilon$$

The variable A produces strings that have an equal number of `c`'s and the total number of `a, b, d` occurrences. Then S (an extension of the 3 cases above) produces unmatched `a, b, d`'s.

Question 3b

Convert G into Chomsky normal form using the methods from class.

Step 1

Here we have to ensure that S is not recursive. Since it is, we have to add a new start variable, S_0:

$$S_0 \to S$$
$$S \to AS \mid \mathtt{a}A \mid \mathtt{a}S \mid \mathtt{b}A \mid \mathtt{b}S \mid \mathtt{d}A \mid \mathtt{d}S$$
$$A \to AA \mid \mathtt{c}A\mathtt{a} \mid \mathtt{a}A\mathtt{c} \mid \mathtt{c}A\mathtt{b} \mid \mathtt{b}A\mathtt{c} \mid \mathtt{c}A\mathtt{d} \mid \mathtt{d}A\mathtt{c} \mid \epsilon$$

Step 2

Here we have to remove ϵ-rules (except possibly for the start variable). Since $A \to \epsilon$ is a rule, we can see that S does not produce ϵ (and so therefore S_0 does not either). So we have to add rules for all possibilities where A can produce ϵ (and remove that rule):

$$S_0 \to S$$
$$S \to AS \mid \mathtt{a}A \mid \mathtt{a}S \mid \mathtt{b}A \mid \mathtt{b}S \mid \mathtt{d}A \mid \mathtt{d}S \mid \mathtt{a} \mid \mathtt{b} \mid \mathtt{d}$$
$$A \to AA \mid \mathtt{c}A\mathtt{a} \mid \mathtt{a}A\mathtt{c} \mid \mathtt{c}A\mathtt{b} \mid \mathtt{b}A\mathtt{c} \mid \mathtt{c}A\mathtt{d} \mid \mathtt{d}A\mathtt{c} \mid \mathtt{ca} \mid \mathtt{ac} \mid \mathtt{cb} \mid \mathtt{bc} \mid \mathtt{cd} \mid \mathtt{dc}.$$

We would add the rules $A \to A$ and $S \to S$: these are useless rules, so we can safely remove them without changing the language.

Step 3

Here we have to remove unit rules. The only unit rule is $S_0 \to S$. Therefore, we can copy the rules corresponding to S to S_0:

$$S_0 \to AS \mid \mathtt{a}A \mid \mathtt{a}S \mid \mathtt{b}A \mid \mathtt{b}S \mid \mathtt{d}A \mid \mathtt{d}S \mid \mathtt{a} \mid \mathtt{b} \mid \mathtt{d}$$
$$S \to AS \mid \mathtt{a}A \mid \mathtt{a}S \mid \mathtt{b}A \mid \mathtt{b}S \mid \mathtt{d}A \mid \mathtt{d}S \mid \mathtt{a} \mid \mathtt{b} \mid \mathtt{d}$$
$$A \to AA \mid \mathtt{c}A\mathtt{a} \mid \mathtt{a}A\mathtt{c} \mid \mathtt{c}A\mathtt{b} \mid \mathtt{b}A\mathtt{c} \mid \mathtt{c}A\mathtt{d} \mid \mathtt{d}A\mathtt{c} \mid \mathtt{ca} \mid \mathtt{ac} \mid \mathtt{cb} \mid \mathtt{bc} \mid \mathtt{cd} \mid \mathtt{dc}.$$

Step 4

Here we have to ensure that the RHS of any rule is either a single terminal, or consists entirely of (at least 2) variables. We can add new variables $U_{\mathtt{a}}, U_{\mathtt{b}}, U_{\mathtt{c}}, U_{\mathtt{d}}$ to make each of the characters in the alphabet, and replace them accordingly:

$$
\begin{aligned}
S_0 &\to AS \mid U_{\mathtt{a}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{b}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{d}}A \mid U_{\mathtt{d}}S \mid \mathtt{a} \mid \mathtt{b} \mid \mathtt{d} \\
S &\to AS \mid U_{\mathtt{a}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{b}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{d}}A \mid U_{\mathtt{d}}S \mid \mathtt{a} \mid \mathtt{b} \mid \mathtt{d} \\
A &\to AA \mid U_{\mathtt{c}}AU_{\mathtt{a}} \mid U_{\mathtt{a}}AU_{\mathtt{c}} \mid U_{\mathtt{c}}AU_{\mathtt{b}} \mid U_{\mathtt{b}}AU_{\mathtt{c}} \mid U_{\mathtt{c}}AU_{\mathtt{d}} \mid U_{\mathtt{d}}AU_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{a}} \mid U_{\mathtt{a}}U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{b}} \mid U_{\mathtt{b}}U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{d}} \mid U_{\mathtt{d}}U_{\mathtt{c}} \\
U_{\mathtt{a}} &\to \mathtt{a} \\
U_{\mathtt{b}} &\to \mathtt{b} \\
U_{\mathtt{c}} &\to \mathtt{c} \\
U_{\mathtt{d}} &\to \mathtt{d}
\end{aligned}
$$

Step 5

Here we have to break up the long RHSes of rules. The only rules we need to fix are the ones with 3 variables on the RHS of rules corresponding to A, which we can do by introducing new variables:

$$
\begin{aligned}
S_0 &\to AS \mid U_{\mathtt{a}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{b}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{d}}A \mid U_{\mathtt{d}}S \mid \mathtt{a} \mid \mathtt{b} \mid \mathtt{d} \\
S &\to AS \mid U_{\mathtt{a}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{b}}A \mid U_{\mathtt{a}}S \mid U_{\mathtt{d}}A \mid U_{\mathtt{d}}S \mid \mathtt{a} \mid \mathtt{b} \mid \mathtt{d} \\
A &\to AA \mid Y_1U_{\mathtt{a}} \mid Y_2U_{\mathtt{c}} \mid Y_1U_{\mathtt{b}} \mid Y_3U_{\mathtt{c}} \mid Y_1U_{\mathtt{d}} \mid Y_4U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{a}} \mid U_{\mathtt{a}}U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{b}} \mid U_{\mathtt{b}}U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{d}} \mid U_{\mathtt{d}}U_{\mathtt{c}} \\
U_{\mathtt{a}} &\to \mathtt{a} \\
U_{\mathtt{b}} &\to \mathtt{b} \\
U_{\mathtt{c}} &\to \mathtt{c} \\
U_{\mathtt{d}} &\to \mathtt{d} \\
Y_1 &\to U_{\mathtt{c}}A \\
Y_2 &\to U_{\mathtt{a}}A \\
Y_3 &\to U_{\mathtt{b}}A \\
Y_4 &\to U_{\mathtt{d}}A
\end{aligned}
$$

Question 4

Is L_W regular? context-free but not regular? or not context-free? Justify your answer carefully.

We argue that L_W is not context-free and thus also not regular.

Suppose (to the contrary) that L_W is context-free. Then there exists a pumping length p for L_W. We choose the string $w = \mathtt{a}^p\mathtt{b}^{p+1}\mathtt{c}^{2p+2}$, for which the pumping lemma gives a decomposition of w into $uvxyz$. There are $2p+1$ symbols other than $\mathtt{c}$ and $2p+2$ $\mathtt{c}$'s, so Charlie is in the majority, and there is one more $\mathtt{b}$ than $\mathtt{a}$, so Beatrice beats Aaron, as required for w to be in L_W. Since $|vxy| \leq p$, it is impossible for v and y to contain both an $\mathtt{a}$ and a $\mathtt{c}$. We obtain the following cases:

- Case 1: vy does not contain any $\mathtt{c}$'s. Then it must contain $\mathtt{a}$'s or $\mathtt{b}$'s (or both), since $|vy| \geq 1$. We pump up to uv^2xy^2z. This adds at least one $\mathtt{a}$ or $\mathtt{b}$, so Charlie is no longer in the majority, making $uv^2xy^2z \notin L_W$, a contradiction.
- Case 2: vy does contain $\mathtt{c}$'s. We pump down to $uv^0xy^0z = uxz$. If there was a $\mathtt{b}$, we now have at most p $\mathtt{b}$'s remaining, so Beatrice does not beat Alice. On the other hand, if there were no $\mathtt{b}$'s, then we still have $2p+1$ non-$\mathtt{c}$'s but at most $2p+1$ $\mathtt{c}$'s, so Charlie does not have a majority. Either way, we see that $uxz \notin L_W$, a contradiction.

Because we obtain a contradiction in all cases, we conclude that L_W is not context-free.