Problem Set 4: Context-free grammars and PDAs
CS254, Fall 2019, Anna Rafferty

You are welcome to talk to others about the problems on this problem set, but you should **read and attempt each problem prior do discussing it with others.** The write up must be your own work. That means that **you should write your answers by yourself, without looking at notes from sessions with collaborators, and you must be able to explain any answer you write down.** You should list anyone you collaborated with on the Collaboration Form (linked off of moodle).

Due dates: All problems are due on Monday, October 7. As for all homeworks, if a problem is due on a specific day it must be turned in by 10PM on that day. See homework handout (a copy is posted on Moodle) for more specific homework policies. If you use a late day on this homework, you may not look at the solutions until you turn in the homework. Violating this policy is a violation of academic honesty policies.

1. Practice providing context free grammars. You should justify why your grammar generates exactly the language given (i.e., all and only strings in the language), but you do not need to use induction to prove this.[1]

   (a) Sipser 2.6b

   Solution: The complement of this language is $L = \{a^n b^m \mid n \neq m\} \cup \{w \mid w$ has at least one b before an $a\}$. We define a grammar $G = (V, \Sigma, R, S)$ such that $L(G) = L$ as follows:

   - $V = \{S, A, B\}$
   - $\Sigma = \{a, b\}$
   - The set of rules R has the following elements:
     - $S \to aSb \mid aA \mid bB \mid C$
     - $A \to aA \mid \epsilon$
     - $B \to bB \mid \epsilon$
     - $C \to DbaD$
     - $D \to DD \mid a \mid b \mid \epsilon$
   - The start variable is S

   This grammar works a lot like one we did in class: the rule $S \to aSb$ is used to add the same number of as and bs, and then to get rid of all variables, we at some point have to use the rule aA or bB. Since A and B each generate only as and only bs, respectively, we're guaranteed to get at least one more a than we have bs (or vice versa) from the aA, and we can get any number of extra as (or vice versa) using the $A \to aA$ transition. However, once we decide to add more as than bs, we never add more bs, so we know the number will never be equal.

   To generate the strings where we have at least one b before an a, we know that we'll have at least one b and one a (since the strings a and b are generated by the rules derived above). We generate this in $C \to DbaD$, and the D variable generates any string in $\{a, b\}^*$.

   (b) Sipser 2.6d

   Solution: We want a grammar for the language $L = \{x_1 \# x_2 \# \dots \# x_k \mid k \geq 1,$ each $x_i \in \{a, b\}^*$ and for some $i, j, x_i = x_j^R\}$. We define the grammar $G = (V, \Sigma, L, S)$ as follows:

   - $V = \{S, R, G\}$
   - $\Sigma = \{a, b, \#\}$
   - The set of rules L has the following elements:
     - $S \to G\#R \mid R\#G \mid G\#R\#G \mid R$

[1] For checking my CFGs and debugging them, I like the tool on this website: https://web.stanford.edu/class/archive/cs/cs103/cs103.1156/tools/cfg/

  - $R \rightarrow aRa \mid bRb \mid \epsilon \mid a \mid b \mid \# \mid \#G\#$
  - $G \rightarrow aG \mid bG \mid \#G \mid \epsilon$

- The start variable is S

The variable G generates any string in $\{a, b, \#\}^*$. (We want to allow consecutive # signs since each $x_k \in \{a, b\}^*$ and thus may be ϵ.)
The variable R is responsible for making the $x_i = x_j^R$. Characters for this section are generated with aRa and bRb. When we're done generating the reverse section, there are a few options:

- $i = j$ for the section that is the reverse of the other. Then, we replace the final R with an a, b, or ϵ (the first two substitutions would give an odd length palindromic section, the the last would give an even length palindromic section).
- $i + 1 = j$ - the two sections that are the reverse of one another are right next to each other. Then we replace the final R with a #.
- The two sections that are the reverse of one another are separated by at least one other section. Then we generate the intervening section(s) via G, and ensure that the sections that are the reverse of one another are not change by putting # before and after the G (substitution $R \rightarrow \#G\#$).

S is the start variable and encodes the following possibilities:

- The i, j such that $x_i = x_j^R$ are on the ends (or $i = j = 1$), through the rule $S \rightarrow R$
- One of i, j such that $x_i = x_j^R$ is on the far left ($S \rightarrow R\#G$) or far right ($S \rightarrow G\#R$).
- Neither i, j such that $x_i = x_j^R$ is on an end of the string ($S \rightarrow G\#GR\#G$).

2. For each of the following languages, decide if it is a CFL. If it is, describe a PDA for the CFL; you do not need to prove your answer correct. I should be able to implement a PDA from the description you give, without having to make any guesses about how the machine should behave, but you can describe the behavior, not give me the exact tuple. If the language is not a CFL, prove it is not.

   (a) $\{0^n 1^m 2^p 3^q \mid n, m, p, q \geq 1 \text{ and } n = m \text{ and } p = q\}$

   Solution: This is a context free language. The PDA for it is very similar to the $0^n 1^n$ machine we showed in class, except we'll disallow ϵ and count twice. Start off with a transition on 0 that writes an empty stack symbol to the bottom of the stack, then push all the 0s on, then transition on 1 to another state that pops all the 0s off (1 for each 1), then transition on the empty stack symbol to another state. This state then transitions on a 2 to a new state and pushes all the 2s on. On a 3, it transitions to another state and pops all the 2s off, reading a 3 each time. Finally, it transitions to a final state on the empty stack symbol.

   (b) $\{0^n 1^m 2^p 3^q \mid n, m, p, q \geq 1 \text{ and } n = p \text{ and } m = q\}$

   Solution: This language is not context free. We prove this via the pumping lemma. Assume the language is context free. Then there must be some pumping length ℓ. Let $s = 0^\ell 1^\ell 2^\ell 3^\ell$. We must show there's no way to write this as $s = uvxyz$ such that the three conditions in the lemma hold. In order to make $|vxy| \leq p$, we know that v and y can contain at most two different consecutive characters, and they must contain at least one character (since $|vy| > 0$). Then for $i = 0$, uv^0xy^0z, there must be fewer 0s, 1s, 2s, or 3s, while keeping all remaining character counts the same, or there are fewer 0s and 1s (and the same number of 2s and 3s), fewer 1s and 2s (and the same number of 0s and 3s), or fewer 2s and 3s (and the same number of 0s and 1s). Thus, we can't have maintained both the equality of 0s and 2s as well as the equality of 1s and 3s. This is a contradiction, so our assumption that the language is context free must be false.

   (c) $\{0^n 1^m 2^p 3^q \mid n, m, p, q \geq 1 \text{ and } n = m \text{ and } m = p\}$

   Solution: This language is not context free, which we can show via the pumping lemma. Assume the language is context free. Then there must be some pumping length ℓ. Let $s = 0^\ell 1^\ell 2^\ell 3$. We must show there's no way to write this as $s = uvxyz$ such that the three conditions in the lemma hold. In order to make $|vxy| \leq p$, we know that v and y can contain at most two different consecutive characters, and they must contain at least one character (since $|vy| > 0$). Consider two cases: $|vy|$ contains the three, or it does not. If it contains the three, then uv^0xy^0z is not in the language because it contains no 3s (i.e., $q = 0$). Thus, it must not contain the three. Then for $i = 0$, uv^0xy^0z, there must be fewer of the 0s, 1s, or 2s while keeping all remaining character counts the same, or there are fewer 0s and 1s (and the same number of 2s) or fewer 1s and 2s (and the same number of 0s). Thus, we can't have maintained both the equality of 0s, 1s, and 2s. This is a contradiction, so our assumption that the language is context free must be false.

(d) $\{0^n1^m2^p3^q \mid n, m, p, q \geq 1 \text{ and } m = p \text{ or } n = q\}$

Solution: This language is context free. A PDA for it would have ϵ transitions for two options: (1) Read all the 0s, pushing nothing on the stack except the initial bottom of stack symbol, then match up the 1s and the 2s (push all the 1s on the stack, then pop one off each time a 2 is read), then read all the 3s, pushing nothing on the stack. (2) Push all the 0s on the stack, then read all the 1s without changing the stack, then read all the 2s without changing the stack, and finally read all the 3s, popping a 1 off the stack with each 3 read. Transitions to new states occur based on reading one of each symbol, so we can ensure that we have at least one of each symbol.

3. Prove that the class of context-free languages is closed under union, concatenation, and star.

Solution: First, we show that if we have context-free languages A, B, then $A \cup B$ is also context free. Let $G_A = (V_A, \Sigma_A, R_A, S_A)$ be the grammar that generates A and $G_B = (V_B, \Sigma_B, R_B, S_B)$ be the grammar that generates B. We construct the grammar that recognizes $A \cup B$ as $G = (V_A \cup V_B \cup S, \Sigma_A \cup \Sigma_B, R_A \cup R_B \cup R, S)$. S is the new start symbol, and R contains exactly one additional rule: $S \to S_A \mid S_B$. We claim this generates the language $A \cup B$:

$$\begin{aligned} w \in A \cup B &\iff w \in A \text{ or } w \in B \\ &\iff S_A \overset{*}{\Rightarrow} w \text{ or } S_B \overset{*}{\Rightarrow} w \\ &\iff S \Rightarrow S_A \overset{*}{\Rightarrow} w \text{ or } S \Rightarrow S_B \overset{*}{\Rightarrow} w \\ &\iff w \in L(G) \end{aligned}$$

Next, we show that if we have context-free languages A, B, then $A \circ B$ is also context free. Let $G_A = (V_A, \Sigma_A, R_A, S_A)$ be the grammar that generates A and $G_B = (V_B, \Sigma_B, R_B, S_B)$ be the grammar that generates B. We construct the grammar that recognizes $A \circ B$ as $G = (V_A \cup V_B \cup S, \Sigma_A \cup \Sigma_B, R_A \cup R_B \cup R, S)$. S is the new start symbol, and R contains exactly one additional rule: $S \to S_A S_B$. We claim this generates the language $A \cup B$:

$$\begin{aligned} w \in A \circ B &\iff w = xy, x \in A, y \in B \\ &\iff w = xy, S_A \overset{*}{\Rightarrow} x, S_B \overset{*}{\Rightarrow} y \\ &\iff S \Rightarrow S_A S_B \overset{*}{\Rightarrow} xy \\ &\iff w \in L(G) \end{aligned}$$

Next, we show that if we have a context-free language A, then A^* is also context free. Let $G_A = (V_A, \Sigma_A, R_A, S_A)$ be the grammar that generates A. We construct the grammar that recognizes A^* as $G = (V_A \cup S, \Sigma_A, R_A \cup R, S)$. S is the new start symbol, and R contains exactly one additional rule: $S \to \epsilon \mid SS_A$. As above, we can show this generates A^*.

4. Sipser 2.36

Solution: Let the language be $F = \{a^ib^jc^kd^\ell \mid i, j, k, \ell \geq 0 \text{ and if } i = 1 \text{ then } j = k = \ell\}$. This language is a variation of example 2.36. Let $p = 2$ and consider two cases. We have three cases $i = 0$, $i = 1$, or $i > 1$. If $i = 0$, when let $u = \epsilon$, $v = b$, $x = \epsilon$, and $y = \epsilon$. Then $uv^mxy^mz = b^mz$, where z is the remainder of the string s and thus has some combination of b, c, d in the right order. Adding more bs keeps us in the same language ($i = 0$), so condition 1 is met. The length of $vy = 1$, so condition 2 is met, and $|vxy| = 1 \leq p$ so condition 3 is met.

If $i = 1$, then we let $u = \epsilon$, $v = a$, $x = \epsilon$, and $y = \epsilon$. Then $uv^mxy^mz = a^{m+1}z$, where z is the remainder of the string s and thus has some combination of b, c, d in the right order and has all the same number of each character (since $s \in F$ and $i = 1$). Adding more as keeps us in the same language, so condition 1 is met. The length of $vy = 1$, so condition 2 is met, and $|vxy| = 1 \leq p$ so condition 3 is met.

If $i = 2$, we let $u = \epsilon$, $v = aa$, $x = \epsilon$, and $y = \epsilon$. Then $uv^m xy^m z = a^{2m}z$, where z is the remainder of the string s and thus has some combination of b, c, d in the right order. If $m = 0$, we have no as, so we stay in the same language, and if $m > 0$, we have more than one a, so we stay in F. The length of $vy = 2$, so condition 2 is met, and $|vxy| = 2 \leq p$ so condition 3 is met.

Finally, if $i > 2$, we let $u = \epsilon$, $v = a$, $x = \epsilon$, and $y = \epsilon$. Then $uv^m xy^m z = a^{m+i-1}z$, where z is the remainder of the string s and thus has some combination of b, c, d in the right order. Since $i > 2$, the exponent on a is larger than 1 regardless of the value of m, so we meet condition 1. The length of $vy = 1$, so condition 2 is met, and $|vxy| = 11 \leq p$ so condition 3 is met.

However, this language is not context free: there's no way to count and make sure $j = k = \ell$. We can show this by closure properties. The context free languages are closed under intersection with regular languages: we can prove this using the same sort of construction as in our proof about regular languages being closed under intersection (take the Cartesian product of the two sets of states). We have that $\{ab^j c^k d^\ell \mid j, k, \ell \geq 0\} \cap F = \{ab^j c^k d^\ell \mid j, k, \ell \geq 0, j = k = \ell\}$. We can prove this language is non-context free using the pumping lemma. Let $s = ab^p c^p d^p$. Then v or y either contains an a (can't be pumped), or contains no more than two of b, c, d. Then there's no way to maintain equality of the counts of these letters and allow pumping.

5. As we discussed in class, there has been considerable debate about the class of natural languages, with many linguists having decided that they are context sensitive. Context sensitive languages are the next step up from context free grammars in the Chomsky hierarchy. There's a good discussion of the complexity of natural languages by Geoffroy Pullum in "Syntactic and Semantic Parsability, " *Proceedings of the 10th International Conference on Computational Linguistics (ACL 84)* (1984), available at http://www.aclweb.org/anthology/P84-1026.[2] Pullum talks both about whether languages are regular and whether they are context free; we'll focus on whether they are context free, but if you're interested, try to think of how you would show English is non-regular and compare your ideas to Pullum's examples.

   Pullum's most compelling example for natural language not being a CFL (reporting on the work of Christopher Culy) is for the African language Bambara. In Bambara, for a noun x, the construction "x-o-x" means "whichever x." For example, cat-o-cat means "whichever cat" as in "Eris can stare down whichever cat you try to bring into my house" (Eris is my cat). In Bambara, compound nouns can be formed in the same was as in English. For example, cat-enabler (one who enables cats) or cat-enabler-fancier (one who fancies enablers of cats). Thus, one can grammatically say cat-enabler-fancier-o-cat-enabler-fancier ("whichever person fancies the enablers' of cats") but not cat-enabler-fancier-o-cat-fancier-enabler. Define a language $L = \{cat\ w\ o\ cat\ w \mid w \in \{fancier, enabler\}^*\}$. Prove that L is not context free. You are welcome to think of L as a language over $\Sigma = \{cat, o, fancier, enabler\}$ and to abbreviate any of these symbols to save writing (simply write your abbreviation key once so the grader understands).

   Solution: This is the language $\{cw0cw \mid w \in \{f, e\}^*\}$. We can prove this is non context free using the pumping lemma, following a pattern similar to the ww example in the book. Assume the language is context free and has a pumping length p. Let $s = cf^p e^p ocf^p e^p$. This string is longer than p. We have several options for breaking s into parts $uvxyz$. First, note that if vxy does not contain o, then this string cannot be pumped: the string before the o wouldn't end up being the same length as the string before the o, and thus we would not longer be part of the same language. Thus, vxy must contain o, and in particular, x must contain the o, since we must have exactly one o. Then we must have $v \neq \epsilon$ and $y \neq \epsilon$ since both cannot be ϵ and if only one is ϵ, we have the same problem as before where the string on one side of the o is longer than the string on the other side. x must also contain the c, since we cannot have multiple c's. We know that y then contains 0 or more es, and to meet our constraint that $|vxy| \leq p$, v contains 0 or more fs. Then letting $i = 2$, uv^2xy^2z is not in the language, as the string right before the o ends in more e's then the string after the o. Thus, our assumption is incorrect and this language is not context free.

6. **Optional** suggested problems (nothing to turn in here!): If the examples from class and the worksheets have left you feeling like you need additional practice on CFGs, I encourage you to try a few of the Sipser exercises such has 2.2, 2.4a and d, 2.6 a and c, and 2.7. Most of these have answers in Sipser, but if they don't and you'd like to check your answer, please feel free to come talk to me in office hours or to post on Slack.

[2] Hat tip to David Liben-Nowell for the reference and this question.