

Please think carefully about how you are going to organise your answers *before* you begin writing. Make sure your answers are complete, clean, concise and rigorous.

1. Design a context-free grammar that generates the language

$$\{x \in \{0, 1\}^* : x \text{ is not of the form } ww \text{ for any } w \in \{0, 1\}^*\}.$$

Argue that your CFG is correct.

[10 points]

2. Consider the following CFG:

$$S \longrightarrow 1S00 \mid 00S1 \mid SS \mid 0S1S0 \mid \varepsilon$$

- 2.1. Give a simple description of the language it generates, in the form $\{x \in \{0, 1\}^* : \langle \text{some simple property of } x \rangle\}$. The simple property might be expressed as an equation or a sentence or some combination of the two.

[5 points]

- 2.2. Prove the correctness of your answer. You may want to mimic the style of the proof we did in class for a closely related language.

[10 points]

3. For each of the following languages, say whether or not it is a CFL and prove your answer, either by designing an appropriate CFG or PDA or by using closure properties and/or the pumping lemma. If designing a CFG/PDA, please explain your construction in brief so the grader can understand your design.

3.1. $\{a^n b^n c^m : n \leq m \leq 2n\}$.

[8 points]

3.2. $\{a^n b^{n^2} : n \geq 0\}$.

[8 points]

3.3. $\{x_1 \# x_2 \# \dots \# x_k : k \geq 1, \text{ each } x_i \in \{a, b\}^*, \text{ and for some } i \text{ and } j \text{ (possibly equal), } x_i = x_j^R\}$.

[9 points]

3.4. $\{b_i \# b_{i+1} \in \{0, 1, \#\}^* : i \geq 1\}$, where b_i is the binary representation of the integer i with no leading 0's (e.g. $b_5 = 101, b_{18} = 10010$).

[10 points]

3.5. $(a \cup b)^* - \{(a^n b^n)^n : n \geq 1\}$.

[10 points]

4. Define the *length* of a rule in a CFG to be the number of characters required to write down the rule. Thus, if $G = (V, \Sigma, R, S)$ is a CFG and " $A \rightarrow w$ " is a rule in R with $w \in (V \cup \Sigma)^*$, then the length of this rule is $2 + |w|$ if $w \neq \varepsilon$ and 3 if $w = \varepsilon$ (because we do have to write one character to represent ε , even though $|\varepsilon| = 0$). Define the *complexity* of a CFG to be the sum of the lengths of all the rules in the CFG.

Note that things like " $A \rightarrow w \mid x \mid y$ " are not rules; they are convenient notation for, in this case, three separate rules " $A \rightarrow w$ ", " $A \rightarrow x$ ", and " $A \rightarrow y$ ", so the lengths of these three rules must be figured separately and added up.

4.1. Suppose the PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is in normal form as discussed in class (i.e., each move is either a one-character push or a pop, but not both; there is only one accept state; and the stack is empty upon acceptance). Suppose we use the procedure of Sipser's Lemma 2.27 to convert it into a CFG G . Give the best possible asymptotic upper bound on the complexity of G , in terms of $|Q|$. Assume $|\Sigma|$ and $|\Gamma|$ are constants. By "asymptotic" I mean that you don't need to give an exact bound, and should use big- O notation to simplify, where possible. You should prove why your bound always holds. Make sure you assume nothing about M : your upper bound must hold even for the most outlandish of PDAs. [5 points]

4.2. We proved that every regular language is context-free as follows: a regular language is recognized by a DFA, which is automatically a PDA (that ignores its stack), which is equivalent to some CFG. However, having solved the previous problem, you know that if we start with a DFA with n states, the complexity of the CFG that results by following this proof may be **huge** (in terms of n , assuming a constant-sized alphabet). Come to the rescue by proving that any n -state DFA (over a constant-sized alphabet) can be converted into an equivalent CFG whose complexity is only $O(n)$. Hint: Prove that any DFA can be converted into a CFG where every rule is either of the form " $A \rightarrow aB$ ", or of the form " $A \rightarrow \varepsilon$ ", where A, B are variables and a is a terminal. What would be a natural choice for the set of variables? [15 points]

5. Do problem 2.27 from your textbook (Sipser). [10 points]

Challenge Problems

CP5: A deterministic pushdown automaton (DPDA) is just like a PDA, except that its transition function must be deterministic. Formally, it is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ that satisfies all of the conditions for a PDA and the following three additional conditions:

$$\begin{aligned} |\delta(q, a, s)| &\leq 1, & \forall q \in Q, a \in \Sigma_\varepsilon, s \in \Gamma_\varepsilon. \\ |\delta(q, \varepsilon, s)| = 1 &\Rightarrow |\delta(q, a, s)| = 0, & \forall q \in Q, a \in \Sigma, s \in \Gamma_\varepsilon. \\ |\delta(q, a, \varepsilon)| = 1 &\Rightarrow |\delta(q, a, s)| = 0, & \forall q \in Q, a \in \Sigma_\varepsilon, s \in \Gamma. \end{aligned}$$

Unlike in the finite automaton case, where nondeterminism did not add extra power, nondeterminism provably adds power to pushdown automata. Your challenge is to prove this.

It may help to consider the language $\{0^n 1^n : n \geq 0\} \cup \{0^n 1^{2n} : n \geq 0\}$.