CS 310, Assignment 4

Answers

1. Let $\Sigma = \{0, 1\}$. Give a context-free grammar that generate the following language:

$$\{0^{i+k}1^{2i} : k \geq 0, i \geq 2\} + \{0^{2i+k}1^{i+k} : i, k \geq 0\}$$

ANSWER: We have a union of two languages so we start with the following two rules:

$$S \to A \quad S \to B$$

A is responsible for the first language which I will rewrite as $\{0^k 0^i 1^{2i} : k \geq 0, i \geq 2\}$, a concatenation of two languages:

$$A \to CD$$

The rest is pretty standard:

$$C \to 0C \quad C \to \varepsilon \quad D \to 0D11 \quad D \to 001111$$

On now to the second language. In the same spirit I am going to rewrite it as

$$\{0^{2i}0^k 1^k 1^i : i, k \geq 0\}$$

and then the rules just write themselves:

$$B \to 00B1 \quad B \to E \quad E \to 0E1 \quad E \to \varepsilon$$

---

2. The following grammar with start symbol S generates the language $\{0^i 1^j 0^k : i = j \text{ or } j = k\}$:

$$\begin{array}{llll} S \to XO & S \to OY & O \to O0 & O \to \varepsilon \\ X \to 0X1 & X \to \varepsilon & Y \to 1Y0 & Y \to \varepsilon \end{array}$$

   (a) Show that this grammar is ambiguous.

ANSWER: We are looking for a string that satisfies both conditions from the disjunction defining the language. The simplest string that shows ambiguity is therefore 010. That string is generated by both XO and OY, as follows:

$$S \Rightarrow XO \Rightarrow 0X1O \Rightarrow 01O \Rightarrow 01O0 \Rightarrow 010$$
$$S \Rightarrow OY \Rightarrow O0Y \Rightarrow 0Y \Rightarrow 01Y0 \Rightarrow 010$$

Clearly these two derivations are not similar and so the grammar is ambiguous. The golden rule for showing that two derivations are or are not similar is to draw and then compare the respective parse trees. You are welcome (even encouraged!) to do so if you wish, but in this case the lack of similarity is obvious since with the exception of S the terminals that appear in the first derivation do not appear in the second and also the other way around.

(b) Is this language inherently ambiguous? State your thoughts one way or another (you only need to provide your thoughts, not a proof).

ANSWER: The language is inherently ambiguous. To see this note first that the language is a disjunction of two languages. If we generate each of these two "sub-languages" separately using two "sub-grammars" (like I did for this particular question) then we necessarily obtain an overall ambiguous grammar. Indeed, consider some string containing an equal number of 0, 1, and 0 symbols (like our friend form the previous question 010); this string belongs to the intersection of the two sub-languages and so can always be generated in two ways (one for each sub-grammar).

Now, is there a cleverer way or generating the two languages at once? That does not appear to be the case. We know how to generate an equal number of two symbols (we have seen this countless times in the course), but there is no way to implement the choice of *which* two symbols to match in an unambiguous manner.

Alternatively, we note that we can always construct a deterministic pushdown automaton for an unambiguous language (we have not explicitly talked about this in class, but even so this alternate answer could be instructive so I am including it anyway). Conversely, it is not possible to construct a deterministic pushdown automaton for an inherently ambiguous language. Consider now a pushdown automaton that accepts our language. This automaton will have to decide at the very beginning whether to compare (using the stack) the first bunch of 0's against the 1's, or ignore the prefix of 0's perform a comparison later (of 1's against the second bunch of 0's). Any successful such a comparison will do, but at the beginning of the input (when we only see 0's) there is no way to decide which one is the winner. Furthermore, if we decide to perform one comparison and our choice proves wrong, that happens much later in the input and we have no way to come back and perform the other comparison. For all these reasons a pushdown automaton accepting our language is necessarily nondeterministic and so the language is inherently ambiguous.

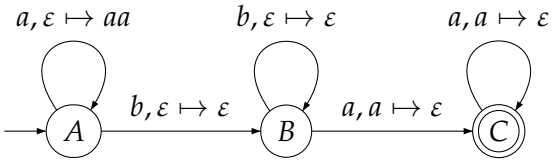
---

3. Design a *deterministic* pushdown automaton that recognizes the language

$$\{a^i b^k a^{2i} : i, k \geq 1\}$$

Draw a table that traces the behavior of your pushdown automaton on the input *aabbaaaa* and explain how this input is accepted or rejected (as the case might be).

ANSWER: I am going to push two symbols a for each a in the first part of the string. This way, when the first b appears I will have twice as many a's on the stack than I had in the input. This bodes well for the trailing a's: I need to have twice as many a's than I had at the beginning, and I already have this many symbols on the stack, so all I need is to match the symbols on the stack against the symbols in the input. In between the a's I have a number of b's that I do not need to match with anything else, so I will just eat them up without changing anything on the stack. I will sequence these three steps using three states. Overall we have:



Note in passing that it *is* a mistake to mark B as accepting, since is this is the case then the automaton will also accept strings for which $i = 0$ in contradiction to the requirements that $i \geq 1$.

The following is the trace of the automaton on the string *aabbaaaa*:

Input	State	Stack
aabbaaaa	*A*	ε
abbaaaa	*A*	*aa*
bbaaaa	*A*	*aaaa*
baaaa	*B*	*aaaa*
aaaa	*B*	*aaaa*
aaa	*C*	*aaa*
aa	*C*	*aa*
a	*C*	*a*
ε	*C*	ε

At the end of the day we end up at the end of the input, in a accepting state (C), and with an empty stack. Therefore the input is accepted.

---

4. Is the language $L = \{a^i b^i c^j d^i : i \geq 0, j \geq 0\}$ context-free? Prove one way or another.

ANSWER: Assume that L is context-free and let p be the constant threshold given by the pumping lemma. We consider the string $s = a^p b^p d^p \in L$. We note the following:

(a) We can choose a string that contains c, but we can also choose a string that does not contain this symbol. Indeed, the pumping lemma applies to *any* (long enough) string in the language. We chose to go the c-less route since we can pump the c's as many times as we want and so we cannot reach a contradiction when c's are present.

(b) $|s| \geq p$ and so the pumping lemma applies.

Therefore we can write $s = uvwxy$ with the substrings u, v, w, x, and y satisfy the conditions of the pumping lemma and so $uv^k wx^k y \in L$ for any $k \geq 0$.

(a) If either v or x contains occurrences of two or more distinct symbols, then uv^2wx^2y is not in $a^*b^*c^*d^*$ and so not in L, a contradiction.

(b) The only other possibility is that v and x each contains occurrences of at most one symbol. By the pumping lemma we know that v and x cannot both be ε. Thus the strings v and x contain occurrences of exactly one or of exactly two of the symbols a, b, d. This means that uv^2wx^2y cannot have equal numbers of a's, b's and d's, and so $uv^2wx^2y \notin L$, a contradiction.

In all it follows that L is not context free.
