MA2C03: TUTORIAL 9 PROBLEMS
FORMAL LANGUAGES AND GRAMMARS

1) Consider the binary alphabet $\{0,1\}$, start symbol $\langle S\rangle$, set of non-terminals consisting of $\{\langle S\rangle, \langle A\rangle, \langle B\rangle, \langle C\rangle, \langle D\rangle, \langle E\rangle\}$, and production rules given by

(1) $\langle S\rangle \to \langle A\rangle\langle B\rangle\langle C\rangle$
(2) $\langle A\rangle\langle B\rangle \to 0\langle A\rangle\langle D\rangle$
(3) $\langle A\rangle\langle B\rangle \to 1\langle A\rangle\langle E\rangle$
(4) $\langle D\rangle\langle C\rangle \to \langle B\rangle 0\langle C\rangle$
(5) $\langle E\rangle\langle C\rangle \to \langle B\rangle 1\langle C\rangle$
(6) $\langle D\rangle 0 \to 0\langle D\rangle$
(7) $\langle D\rangle 1 \to 1\langle D\rangle$
(8) $\langle E\rangle 0 \to 0\langle E\rangle$
(9) $\langle E\rangle 1 \to 1\langle E\rangle$
(10) $0\langle B\rangle \to \langle B\rangle 0$
(11) $1\langle B\rangle \to \langle B\rangle 1$
(12) $\langle A\rangle\langle B\rangle \to \epsilon$
(13) $\langle C\rangle \to \epsilon$

(a) What type of grammar is this (context-free or phrase structure)? Justify your answer

(b) What language does this grammar generate? (Hint: Rules (12) and (13) show you that the word before the last non-terminals are swapped out can contain only non-terminals $\langle A\rangle$,$\langle B\rangle$,$\langle C\rangle$. Figure out how the other rules combine to give you words consisting of the terminals and $\langle A\rangle$,$\langle B\rangle$,$\langle C\rangle$.)

Solution: (a) Production rule (2) has two non-terminals on the left-hand side, so this grammar is clearly a phrase structure grammar as context-free grammars have production rules that allow only one non-terminal to be swapped for something else.

(b) Rules (1), (2), and (4) yield $0\langle A\rangle\langle B\rangle 0\langle C\rangle$, whereas rules (1), (3), and (5) yield $1\langle A\rangle\langle B\rangle 1\langle C\rangle$. We can get to $01\langle A\rangle\langle B\rangle 01\langle C\rangle$ from $0\langle A\rangle\langle B\rangle 0\langle C\rangle$ by applying rules (3), (8), (5), and (10). Similarly, we can get $w\langle A\rangle\langle B\rangle w\langle C\rangle$ for any word $w \in \{0,1\}^*$. Rules (6) to (11) are there to rearrange non-terminals so that rules (2) to (5) can be applied. Altogether, we generate the language $L = \{ww \mid w \in \{0,1\}^*\}$. Note that the language L CANNOT be generated by a context-free grammar as a context-free grammar could NOT keep track of the duplicate w by replacing only one non-terminal at a time. We have thus constructed an example of a language that is generated by a phrase structure grammar and cannot be generated by a context-free grammar thus showing that the former category is more general than the latter.

2) Let L be the language over the alphabet $\{0, 1\}$ consisting of all words where the string 00 occurs as a substring.

(a) Draw a finite state acceptor that accepts the language L. Carefully label all the states including the starting state and the finishing states as well as all the transitions. Make sure you justify it accepts all strings in the language L and no others.

(b) Write down the transition mapping of the finite state acceptor you drew in the previous part of the problem.

(c) Devise a regular grammar in normal form that generates the language L. Be sure to specify the start symbol, the non-terminals, and all the production rules.

Solution: (a) See diagram of finite state acceptor on the next page.

We can use three states $\{i, A, B\}$, where i is the initial state. Since we must ensure the word contains the string 00, when 1 is the input, we stay in the initial state i. For input 0, we move to a new state A. A is not an accepting state as we have so far only half of the string 00, the first zero. If we get input 1, we have to restart the process of capturing the string 00, so we get back to the initial state i. If we get input 0, then we will have received the second zero we want, so we'll move to a new state B, which is an accepting state. Once we have the substring 00, we don't care what follows, so the transitions for both 0 and 1 out of state B are back into B itself.

(b)

$$\begin{aligned} t(i,0) &= A \qquad & t(i,1) &= i \\ t(A,0) &= B \qquad & t(A,1) &= i \\ t(B,0) &= B \qquad & t(B,1) &= B \end{aligned}$$

(c) We shall use the algorithm discussed in lecture in order to generate the regular grammar in normal form corresponding to the finite state acceptor constructed above. The finite state acceptor had three states $\{i, A, B\}$, where i was the initial state. Correspondingly, we use three non-terminals in our regular grammar: the start symbol $\langle S\rangle$ corresponding to the initial state i, $\langle A\rangle$ corresponding to state A, and $\langle B\rangle$ corresponding to state B. We first write the production rules corresponding to the transitions out of the initial state i :

(1) $\langle S\rangle \to 1\langle S\rangle$.

(2) $\langle S\rangle \to 0\langle A\rangle$.

Next, we write the production rules corresponding to the transitions out of state A :

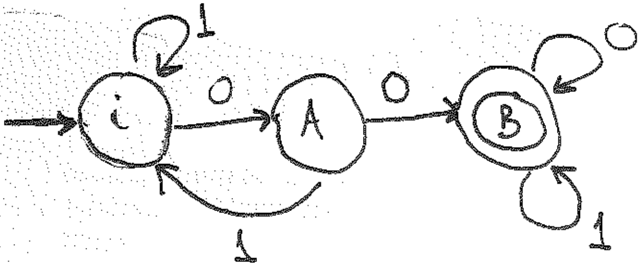
(3) $\langle A\rangle \to 1\langle S\rangle$.
(4) $\langle A\rangle \to 0\langle B\rangle$.

Finally, we write the production rules corresponding to the transitions out of state B :

(5) $\langle B\rangle \to 1\langle B\rangle$.
(6) $\langle B\rangle \to 0\langle B\rangle$.

Rules (1)-(6) are of type (i). For each accepting state, we will write down a rule of type (iii). Since there is only one accepting state, B, we have only one such rule:

(7) $\langle B\rangle \to \epsilon$.



1
0
0
0
i
A
B
1
1