Final #2

Mark all correct answers in each of the following questions.
$G = (N, T, R, S)$ denotes a context-free grammar. All grammars are without useless letters.

4. (a) If G is unambiguous, then it is possible to add to R a rule (without changing T and N, but perhaps changing the language accepted by the grammar) in such a way that the grammar will still be unambiguous.

   (b) Suppose there exist at least two non-terminals A and B with ε-productions $A \to \varepsilon$ and $B \to \varepsilon$ and $L(G)$ is infinite. Then G is ambiguous.

   (c) If $L(G_1) = L(G_2)$, then G_2 is unambiguous if and only if G_1 is such.

   (d) Suppose $G_i = (N_i, T, R_i, S_i), i = 1, 2$, where $N_1 \cap N_2 = \emptyset$. Let $G = (N_1 \cup N_2 \cup \{S\}, T, R_1 \cup R_2 \cup \{S \to S_1 S_2\}, S)$ (where we assume that $S \notin N_1 \cup N_2$). (By the way, $L(G) = L(G_1)L(G_2)$.) If G is unambiguous, then so are G_1 and G_2.

5. (a) The following algorithm has been suggested for elimination of unit rules: First take all rules that are not unit rules. Next, for each pair of non-terminals A, B with $A \overset{+}{\Longrightarrow} B$, take all rules that are obtained from non-unit rules by replacing any occurrences of A on the right-hand side of a rule by B. (For example, the rule $C \to aAbBAba$ gives rise to three additional rules: $C \to aBbBAba$, $C \to aAbBBba$, $C \to aBbBBba$.) The algorithm yields within finitely many steps a grammar without unit rules that is equivalent to G.

(b) Suppose that $N = \{S, A\}$. The grammar rules are

$$S \to AAS \mid ASa \mid aba\,,$$
$$A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_{10}\,,$$

for some $\alpha_1, \alpha_2, \ldots, \alpha_{10} \in (N \cup T)^* - \{\varepsilon, S, A\}$. We employ the algorithm discussed in class for eliminating left-recursion (with $A_1 = S, A_2 = A$). Then the grammar we obtain includes at most 32 rules.

(c) Consider the algorithm for constructing a grammar in Chomsky Normal Form, that is equivalent to a given grammar without ε-rules and unit rules. There exists a function $f : \mathbf{N}^3 \to \mathbf{N}$ (where $\mathbf{N}$ denotes the set of positive integers) such that, denoting by R' the set of rules of the new grammar, we have $|R'| \leq f(|N|, |T|, |R|)$.

(d) Suppose all rules in R are of the form $A \to \alpha$, where $\alpha \in T^* \cup T^*NT^* \cup T^*NT^*NT^*$, and there are no ε-rules and unit rules. A slight modification of the CYK algorithm yields a parsing algorithm that still works in time $O(n^3)$ for words on length n.

6. (a) For any $\alpha \in (N \cup T)^* - \{\varepsilon\}$, denote by $\mathsf{DIRECT\_FIRST}(\alpha)$ the first letter of α. If R includes no ε-rules, and $\mathsf{DIRECT\_FIRST}(\alpha_1) \neq \mathsf{DIRECT\_FIRST}(\alpha_2)$ for every non-terminal A and rules $A \to \alpha_1$ and $A \to \alpha_2$, then G is $LL(1)$.

(b) The number of stages (not including stage 0) in the algorithm presented in class for computing the FOLLOW sets is at most $|N| \cdot |T|$. It is impossible to reduce this upper bound in general.

(c) If R includes the rules $A \to BAB$ and $A \to Bab$, then G is not $LL(1)$.

(d) If G is $LL(k)$ for some $k \geq 1$, then for every $w \in T^*$ there exists exactly one parse tree producing w.

Solutions

4. (a) The grammar given by the rules

$$S \to aS \mid bS \mid \varepsilon\,,$$

accepts the language $L((a \cup b)^*)$, and is easily seen to be unambiguous (and even $LL(1)$). By adding any rule, we make it ambiguous. Indeed, take any sequence of derivations which includes the new rule, that produces some word $w \in \{a, b\}^*$. Since w belongs to the language accepted by the original grammar, there is a parse tree producing w that does not use the new rule. Thus, w is produced by (at least) two parse trees, which means that the new grammar is ambiguous.

(b) The grammar given by the rules

$$S \to AB\,,$$
$$A \to aA \mid \varepsilon\,,$$
$$B \to bB \mid \varepsilon\,,$$

accepts the language $L(a^*b^*)$ and satisfies the required properties. It is easy to see that the grammar is unambiguous. (In fact, it is even $LL(1)$.)

(c) The grammar defined by the rules

$$S \to aS \mid \varepsilon\,,$$

accepts the language $L(a^*)$ and is clearly unambiguous, while the grammar defined by the rules

$$S \to aS \mid aaS \mid \varepsilon\,,$$

accepts the same language and is ambiguous (as the word aa, for example, may be produced by two distinct parse trees).

(d) Let us show that, say, G_1 is unambiguous. Let w be any word in $L(G_1)$. Take any word w' is $L(G_2)$. Then $ww' \in L(G)$, and there is a unique parse tree producing ww'. The root of this tree is labelled by S, and it has two children, labelled by S_1 and S_2. The uniqueness of the tree means in particular that the subtree rooted at S_1 is unique. Thus, there exists a unique parse tree for G_1 producing w. Hence G_1 is unambiguous. The proof for G_2 is analogous.

Thus, only (d) is true.

5. (a) All parsing options in the suggested grammar are possible in the original grammar as well using suitable sequences of derivations, and therefore the language accepted by the suggested grammar is contained in that accepted by the original grammar. However, the algorithm fails to capture some of the possibilities of the given grammar. For example, applying the suggested algorithm to the grammar defined by the rules

   $S \to A \mid b\,,$

   $A \to a\,,$

   we obtain the grammar defined by

   $S \to b\,,$

   (with the letter A, which became useless, omitted). The first grammar accepts the language $\{a, b\}$ while the second accepts $\{b\}$.

   (b) Since the right-hand side of no rule for S starts with S, we start dealing with the rules for A. Suppose k out of the words $\alpha_1, \alpha_2, \ldots, \alpha_{10}$ start with S. Each of the k rules $A \to \alpha_i$ with these α_i's is replaced by three rules, in which the leading S is replaced by the right-hand sides of the rules for S, namely AAS, ASa and aba. At this stage we have $3k + (10 - k)$ rules for A. Out of the $3k$ new rules, the right-hand side of $2k$ rules starts with A. Let l be the number of such rules among the $10 - k$ original rules for A. Thus, the right-hand side of $2k + l$ of the rules for A start with A, while that of the other $10 - l$ does not. Now we add a new non-terminal A' with $2k + l + 1$ rules (including $A' \to \varepsilon$) for it, whereas for A we have only $10 - l$ rules left.

   Altogether, in the new grammar there are $3+(2k+l+1)+(10-l) = 2k + 14$ rules. In principle, this may be as large as 34. However, this bound is obtained for $k = 10$ only. If $k = 10$, then the non-terminal A is useless. Therefore $k \leq 9$, so that the new grammar has at most 32 rules.

   (c) The numbers $|N|$, $|T|$ and $|R|$ are unrelated to the lengths of the right-hand sides of the rules in the grammar. Since rules of the

form $A \to B_1 B_2 \dots B_k$ are replaced by $k-2$ rules, no function as required may possibly exist. For example, suppose the grammar has a single rule, say $S \to w$, where w is a word of length k in T^*. We first add $|T|$ non-terminals to N, each corresponding to one of the terminals, with a single corresponding rule for each, and then replace the resulting rule $S \to B_1 B_2 \dots B_k$ by $k-2$ rules. Thus, in the given grammar we have $|N| = 1$, arbitrary fixed $|T|$ and $|R| = 1$, yet $|R'| = |N| + k - 2$ may be arbitrarily large.

(d) We proceed as in the CYK algorithm. Consider a typical stage, when we have just finished finding out for which non-terminals A and subwords u of length up to some k of the input word w it is the case that $A \stackrel{*}{\Longrightarrow} u$. We need now to find out the same for subwords u of length $k+1$. To find out whether $A \stackrel{*}{\Longrightarrow} u$, we go over all rules for A. Consider a typical rule, say $A \to v_1 B_1 v_2 B_2 v_3$, where $v_1, v_2, v_3 \in T^*$ and $B_1, B_2 \in N$. (If the right-hand side includes less than two non-terminals, the problem is even simpler.) If v_1 does not coincide with the prefix of corresponding length of u, or v_3 does not coincide with the suffix of corresponding length of u, there is nothing to check. If both coincide, we need to check whether $B_1 v_2 B_2 \stackrel{*}{\Longrightarrow} u'$, where u' is the same as u, but with the prefix v_1 and the suffix v_3 removed. For each j between 1 and $|u'|-1-|v_2|$ we need to check whether it is the case that $B_1 \stackrel{*}{\Longrightarrow} u'_1$, $B_2 \stackrel{*}{\Longrightarrow} u'_2$, where u'_1 and u'_2 are the prefix of length j and suffix of length $|u'| - j - |v_2|$ of u', respectively, and the subword of u' in between them coincides with v_2. If this is the case for some j, then the answer is positive. Obviously, the runtime is the same as that of the CYK algorithm.

Thus, (b) and (d) are true.

6. (a) Consider the grammar defined by the rules:

$S \to A \mid a\,,$
$A \to a\,,$

We have $\mathsf{DIRECT\_FIRST}(A) = A \neq a = \mathsf{DIRECT\_FIRST}(a)$, so

that the grammar satisfies the property in question. However, the grammar is obviously not $LL(1)$. (In fact, it is even ambiguous.)

(b) The asserted upper bound is correct, but a much better bound exists. In fact, in the beginning we place in $\mathsf{FOLLOW}(A)$ those terminals a for which there exists a rule of the form $B \to \alpha A a \beta$. In the following stages, the initial sets grow due to the observation that, if $B \to \alpha A \beta$, where $\mathsf{Nullable}(\beta)$, then $\mathsf{FOLLOW}(A) \supseteq \mathsf{FOLLOW}(B)$. Thus, if a terminal a belongs to the FOLLOW set of some nonterminal A, then it will be known within at most $|N| - 1$ stages.

(c) If $\mathsf{FIRST}(B) \neq \emptyset$ then $\mathsf{FIRST}(BAB) \cap \mathsf{FIRST}(Bab) \neq \emptyset$, so the grammar is not $LL(1)$. Thus, suppose $\mathsf{FIRST}(B) = \emptyset$. The rule $A \to Bab$ shows then that $a \in \mathsf{FIRST}(A)$. It follows that a belongs both to $\mathsf{FIRST}(BAB)$ and to $\mathsf{FIRST}(Bab)$, and again the grammar is not $LL(1)$.

(d) Only words in $L(G)$ have parse trees producing them. Hence, unless $L(G) = T^*$, there are words in T^* with no parse trees producing them.

Thus, only (c) is true.