Midterm

Mark all correct answers in each of the following questions.

4. A *logical expression* (for the purposes of this question) is an expression made of identifiers, and symbols for negation – !, disjunction – $\vee$, and conjunction – $\wedge$. (Here, assume that *identifier* has already been defined, and represents strings consisting of letters and digits, starting with a letter.) A *literal* is an identifier preceded by any number of negation signs. Any two identifiers in an expression are separated by either a disjunction or a conjunction symbol. For example, $x5$ and $!!!!x12$ are literals; $x5 \wedge !!x5 \vee !!!x12$ is a logical expression. On the other hand, $x!$ and $x!y$ are not literals; $\vee x$ and $x \wedge \wedge x$ are not logical expressions.

   (a) The following is a regular definition of logical expressions:

   $$\begin{array}{lcl} \textit{literal} & \rightarrow & !^* \textit{ identifier}, \\ \textit{logicalExpression} & \rightarrow & \textit{literal } ([\vee\wedge] \textit{ literal})^*. \end{array}$$

   (b) The following is a regular definition of logical expressions:

   $$\begin{array}{lcl} \textit{literal} & \rightarrow & !^* \textit{ identifier}, \\ \textit{logicalExpression} & \rightarrow & \textit{literal } ([\vee\wedge] \textit{ logicalExpression})?. \end{array}$$

   (c) Suppose the truth value of a logical expression, given truth values for the identifiers constituting it, is calculated from left to right, with negation having precedence over conjunction, which in turn has precedence over disjunction. A *true-true logical expression* is a logical expression having the property that, if all identifiers in it assume the value T, then the expression assumes the same value. For example, $x \wedge !!y \vee !x$ is a true-true expression, but $x \wedge !x \vee !y$ is

not. Then the following is a regular definition of true-true logical expressions:

$$\begin{array}{lcl} \textit{TTLiteral} & \rightarrow & (!!)^* \ \textit{identifier}, \\ \textit{TTConjunctiveExpression} & \rightarrow & \textit{TTLiteral} \ (\wedge \ \textit{TTLiteral})^*, \\ \textit{TTLogicalExpression} & \rightarrow & (\textit{logicalExpression} \ \vee)? \\ & & \quad \textit{TTConjunctiveExpression} \\ & & \quad (\vee \ \textit{logicalExpression})?. \end{array}$$

(Assume that *logicalExpression* is any correct regular definition of logical expressions, not necessarily one of the definitions in the previous parts.)

(d) A $false\text{-}false$ *logical expression* is a logical expression having the property that, if all identifiers in it assume the value F, then the expression assumes the same value. A *cute logical expression* is a logical expression which is both true-true and false-false. For example, $x \vee !!y$ is cute. Then there does not exist a regular definition of cute logical expressions.

In Questions 5 and 6, $G = (N, T, R, S)$ denotes an arbitrary context-free grammar, unless indicated otherwise.

5. (a) If $L(G)$ is closed under concatenation (namely, if $u, v \in L(G)$ then $uv \in L(G)$) and contains two non-commuting words (i.e., words u, v such that $uv \neq vu$), then G is ambiguous.

(b) If G is unambiguous and $L(G)$ has the property that $L(G)^m \cap L(G)^n = \emptyset$ for every $m > n \geq 0$, then the grammar

$$G' = (N \cup \{S'\}, T, R \cup \{S' \rightarrow SS', S' \rightarrow \varepsilon\}, S')$$

(where $S' \notin N$) is an unambiguous grammar such that $L(G') = L(G)^*$.

(c) The grammars G_1 and G_2, defined by the rules

$$\begin{array}{lcl} E & \rightarrow & E + A \mid E - A \mid A, \\ A & \rightarrow & a \mid b \mid c, \end{array}$$

and

$$\begin{aligned} E &\rightarrow A + E \mid A - E \mid A, \\ A &\rightarrow a \mid b \mid c, \end{aligned}$$

respectively, are designed to produce certain collections of arithmetic expressions. Now G_1 gives the right interpretation of all expressions and is unambiguous. However, G_2 does not give the right interpretation (of some expressions), and in particular is ambiguous.

(d) If $S \rightarrow SS \in R$ and $L(G)$ includes a word of length 10, then G is ambiguous.

6. (a) Suppose R includes (among others) the rules $S \rightarrow Abac$ and $S \rightarrow Abca$, and it is known that $A \stackrel{*}{\Longrightarrow} a^{100}b^{100}c^{100}$. Then G is not an $LL(101)$ grammar, but it may be an $LL(102)$ grammar.

(b) Let $G_i = (N_i, T, R_i, S_i)$ for $i = 1, 2$, where $N_1 \cap N_2 = \emptyset$. Put

$$G = (N_1 \cup N_2 \cup \{S\}, T, R_1 \cup R_2 \cup \{S \rightarrow S_1,\ S \rightarrow S_2\}, S),$$

where $S \notin N_1 \cup N_2$. (Recall that $L(G) = L(G_1) \cup L(G_2)$.) If G_1, G_2 are $LL(10)$ grammars, then G is an $LL(11)$ grammar.

(c) Suppose the rules in R for replacing the non-terminals $A, B \in N$ are:

$A \rightarrow bSDabS \mid aacD,$
$B \rightarrow bSDabS \mid aacD.$

Suppose also that there exist words $\alpha \in (N \cup T)^*$ and $w \in L(G)$ such that all three non-terminals A, B, D occur in α at least once and $S \stackrel{*}{\Longrightarrow} \alpha \stackrel{*}{\Longrightarrow} w$. Then G is not an $LL(1)$ grammar.

(d) The grammar defined by the rules

$S \rightarrow Ab \mid cB,$
$A \rightarrow Ac \mid dB \mid aaba,$
$B \rightarrow dBBcS \mid bSA,$

then it is not an $LL(1)$ grammar, but it is an $LL(k)$ grammar for every sufficiently large k.

Solutions

4. (a) The negation symbols preceding the identifier are represented by $!^*$, and hence the first line of the suggested definition represents literals. A logical expression consists of a literal, after which we may have an arbitrary number of occurrences of a disjunction/conjunction symbol together with a literal. Hence the suggested definition indeed represents logical expressions.

   (b) expression that is of the same specifications as those of a logical expression. However, in a regular definition one cannot use in any line terms not previously defined. Hence the suggested definition is not a regular definition.

   (c) A logical expression consisting of a single literal is clearly true-true if and only if the identifier is preceded by an even number of occurrences of the negation symbol. Thus, true-true literals are represented by the right-hand side of the first line of the definition. Due to the precedence rules, after the values of all literals have been computed, we compute the truth values of all (maximal) blocks of literals separated by conjunction symbols only, and then calculate the truth value of the whole expression. A block of literals separated by conjunction symbols assumes the value T if and only if all literals within it are true-true. Hence the whole expression is true-true if and only if at least one of the blocks within it consists of true-true literals. It follows that the suggested definition is correct.

   (d) A logical expression consisting of a single literal is false-false if and only if the identifier is preceded by an even number of occurrences of the negation symbol (same as the condition for being true-true). Similarly to the considerations in the preceding part, we conclude that a logical expression is false-false if and only if all (maximal) blocks of literals separated by conjunction symbols include at least one literal preceded by an even number of occurrences of the negation symbol. One concludes easily that the logical expressions that

are both true-true and false-false are represented by the definition:

$$\begin{array}{lcl} \mathit{FFLiteral} & \to & \mathit{TTLiteral}, \\ \mathit{FFConjunctiveExpression} & \to & (\mathit{literal} \wedge)^* \ \mathit{FFLiteral} \ (\wedge \ \mathit{literal})^*, \\ \mathit{TTFFLogicalExpression} & \to & (\mathit{TTLogicalExpression} \vee)? \\ & & \quad \mathit{FFConjunctiveExpression} \\ & & \quad (\vee \ \mathit{TTLogicalExpression})?. \end{array}$$

Note that it would suffice to find a regular definition for false-false logical expressions. In fact, the collection of cute expressions is the intersection of those of true-true expressions and of false-false expressions. Once both are known to have regular definitions, so does the intersection (even if finding an explicit definition may be tiresome).

Thus, (a) and (c) are true.

5. (a) The grammar defined by the rules

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

is unambiguous (and even $LL(1)$), yet $L(G) = \{a, b\}^*$, which language includes pairs of non-commuting words.

(b) The grammar defined by the rules

$$S \to a \mid ab \mid ba,$$

is clearly unambiguous. Since each word in $L(G)^m$ contains exactly m occurrences of the letter a, we have $L(G)^m \cap L(G)^n = \emptyset$ for $m > n$. However, G' is ambiguous. For example, the word aba can be produced in two different ways, namely

$$S' \Longrightarrow SS' \Longrightarrow aS' \Longrightarrow aSS' \Longrightarrow abaS' \Longrightarrow aba,$$

and

$$S' \Longrightarrow SS' \Longrightarrow abS' \Longrightarrow abSS' \Longrightarrow abaS' \Longrightarrow aba.$$

(c) It is true that G_2 constructs for some expressions a parse tree that would hint at calculating the expressions not in accordance with

the usual precedence rules. For example, the expression $a-b-c$ is derived as follows:

$$E \Longrightarrow A-E \Longrightarrow a-E \Longrightarrow a-A-E$$
$$\Longrightarrow a-b-E \Longrightarrow a-b-A \Longrightarrow a-b-c.$$

Thus, it would correspond to the computation of $a-(b-c)$. However, this is irrelevant to the properties of the grammar. In fact, both grammars are unambiguous. Let us show it for G_2 first. Take a word w. We need to show it has a unique parse tree (if it belongs to $L(G_2)$ at all). If the word is of length 1 we must use the rule $E \to A$ first, and then use for A the rule taking it to the only letter of w. Now let $|w| \geq 2$. We have to start with the rule $A \to A+E$ if the second letter of w is $'+'$ and the rule $A \to A-E$ if that letter is $'-'$. Next for A we must use the rule taking it to the first letter of w. Now it remains to derive from E the suffix of w consisting of all the word but the first two letters. Thus, an inductive argument shows that the parse tree is unique. In fact, our argument shows that G_2 is $LL(2)$.

The argument for G_1 is analogous, but here we need to start looking at the word from its end backwards. Note that G_1 is not $LL(k)$ for any k.

(d) Let w be any word in $L(G)$. We claim that the word w^3 has at least two parse trees. In fact, one may obtain this word by applying twice the rule $S \to SS$ to get SSS, and then obtain a w from each of the three occurrences of S. Since the second derivation above might have been either that of the first S in the word or that of the second, we have two distinct derivations of w^3.

Thus, (d) is true.

6. (a) G is not even $LL(301)$ in this case. In fact, both words $a^{100}b^{100}c^{100}bac$ and $a^{100}b^{100}c^{100}bca$ belong to $L(G)$, since

$$S \Longrightarrow Abac \stackrel{*}{\Longrightarrow} a^{100}b^{100}c^{100}bac$$

and

$$S \Longrightarrow Abca \stackrel{*}{\Longrightarrow} a^{100}b^{100}c^{100}bca.$$

The two words have the same first 301 letters, yet already the derivation of S applied at the first stage is distinct for the two words. We mention that G may be $LL(302)$.

(b) The conditions clearly do not guarantee that the languages $L(G_1)$ and $L(G_2)$ are disjoint. If they are not disjoint, then every word in $w \in L(G_1) \cap L(G_2)$ can be produced in G in two ways: Either replace S by S_1 and produce w according to the rules of G_1, or analogously via S_2.

(c) G may well be an $LL(1)$ grammar. For example, suppose it is given by the rules:

$S \to aABD \mid b,$
$A \to bSDabS \mid aacD,$
$B \to bSDabS \mid aacD,$
$D \to d.$

The condition in the question is satisfied since

$$S \Longrightarrow aABD \stackrel{*}{\Longrightarrow} aaacDaacDd \stackrel{*}{\Longrightarrow} aaacdaacdd.$$

The grammar is obviously $LL(1)$ since the right-hand sides of the rules for replacing each of the terminals start with distinct terminals. The fact that A and B may be replaced by exactly the same strings means that one can simplify the grammar (by omitting B from N, deleting the rules for replacing B, and then replacing B by A wherever it appears on the right-hand side of a rule), but is has nothing to do with the grammar being $LL(1)$.

(d) We have

$$A \Longrightarrow dB \Longrightarrow dbSA \Longrightarrow dbAbA \Longrightarrow dbaababA,$$

and therefore

$$A \stackrel{*}{\Longrightarrow} (dbaabab)^n A \Longrightarrow (dbaabab)^n aaba, \qquad n = 0, 1, 2 \ldots.$$

It follows that

$$S \Longrightarrow Ab \stackrel{*}{\Longrightarrow} (dbaabab)^n aabab, \qquad n = 0, 1, 2 \ldots,$$

and

$$S \Longrightarrow Ab \Longrightarrow Acb \stackrel{*}{\Longrightarrow} (dbaabab)^n aabacb, \qquad n = 0, 1, 2 \ldots .$$

Thus, when parsing $(dbaabab)^n aabab$ and $(dbaabab)^n aabacb$, after replacing S by Ab, we do not know which production to use for A by knowing the first $7n + 4$ letters of the word. Hence G is not $LL(k)$ for any k.

Thus, none of the claims is true.