CS375 HW 7 Solution Set (20 points)

Due date: April 3, 2024

1. (6 points)
   The language generated by the following grammar is (2 points) $\boxed{\{ab^m, acb^n \mid m, n \in N\}}$

   S → acB | aB   B → bB | Λ

   This grammar is LL($\boxed{2}$). (1 point)
   Use left-factoring we can find an equivalent LL(k) grammar for this grammar where k is as small as possible. In the following, fill out the blank in the middle portion to make the resulting grammar such an LL(k) grammar.

   S → aT

   $T \to cB \mid B$
   or
   $T \to cB \mid B \mid \Lambda$
   or
   $T \to bB \mid cbB \mid c \mid \Lambda$

   B → bB | Λ   (2 points)

   What is the value of k? k = $\boxed{1}$ (1 point)

2. (6 points)
   The language generated by the following grammar is (2 points) $\boxed{\{\, c(aac)^n \mid n \in N\}}$

   S → SaaS | c

   This grammar is left recursive, hence, it is not LL(k) for any k.
   Fill out the blank below to make the resulting grammar an equivalent grammar of the above grammar but with no left-recursion.

   S → cT

   $T \to aaST \mid \Lambda$
   or
   $T \to aacTT \mid \Lambda$

   (2 points)

Is the resulting grammar LL(k)? x Yes ☐ No (1 point)

If your answer is YES, then what is the value of k? k = 1 (1 point)

For any given string of this language, say 'caacaac', the left most letter is always 'c'. Since the first production of the new grammar '$S \to cT$' is the unique first step, we don't need to scan any thing to use this production. Once this production is applied, we would automatically get a match for the first letter of the input screen. So our first scan starts with the second letter of the input string.
The 2nd letter would be an 'a', so the production to use would be '$T \to aacTT$'. This production gives a match of 'aac', the 2nd through the 4th letters of the input string. So our next scan would be the 5th letter which would again be an 'a' and we would still have to use the production '$T \to aacTT$' for the next derivation step and which again gives a match of 'aac', the 5th through the 7th letters. So our next scan would be the 8th letter of the input string. In general, our n-th scan would be the (2+3(n-1))-th letter of the input string which would always be an 'a' unless it is 'Λ'. In such a case we apply the production '$T \to \Lambda$'. Otherwise, we apply the production '$T \to aacTT$' and get a match of 'aac', the (2+3(n-1))-th through the (2+3n-1)-th letters of the input string. Each time, we only need to scan one letter to determine the production needed for the next derivation step, so that is why it is LL(1).

3. (5 points; 1 point each blank)
The following given grammar is a left recursive grammar

$$S \to Sbab \mid ba \mid b$$

The language generated by this grammar is of the following form:

$$L = \{\, ba(bab)^m,\ b(bab)^n \mid m\ \ n \in N \,\}$$

This left recursive grammar can be transformed to a right recursive grammar as follows:

$S \to$ baT | bT

$T \to$ babT $\quad \Lambda$

This right recursive grammar is an LL(2) grammar.

4. (6 points; 1 point each blank)

The following grammar is an indirect left recursive grammar

S → Bb | a   B → Sa | b

Strings of the language generated by this grammar are of the following form:

$$L = \{\ a(w)^m,\ bb(w)^n \mid w = \boxed{ab}\ m, n \in N\ \}$$

This indirect left recursive grammar can be transformed to a right recursive grammar as follows:

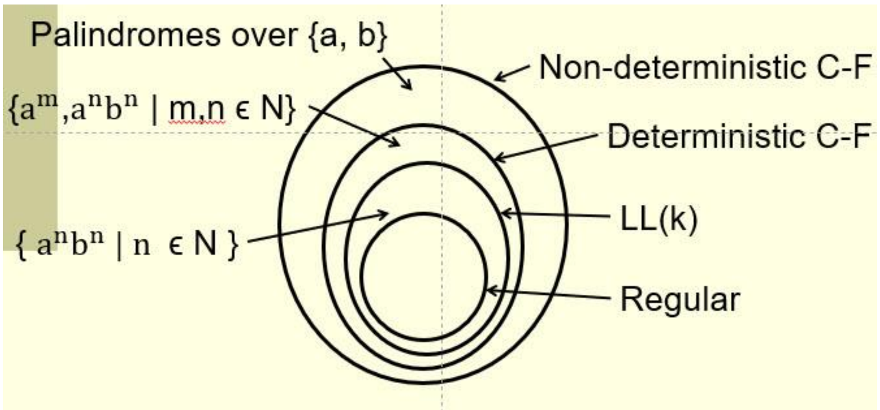
S → $\boxed{bbT}$ | $\boxed{aT}$

T → $\boxed{abT}$ $\boxed{\Lambda}$

This right recursive grammar is an LL($\boxed{1}$) grammar.

5. (7 points)

In slide 41 of the notes "Context-free Languages and Pushdown Automata IV", it is shown that the set of LL(k) languages is a proper subset of the set of deterministic C-F languages (or see the following figure). In particular, it points out that the language $\{a^m, a^n b^n \mid m, n \in N\}$ is a deterministic C-F, but not LL(k) for any k.



To show the language is not LL(k) for any k, note that a grammar for this language is

$S \to A \mid B \quad A \to$ $\boxed{aA}$ | $\boxed{\Lambda}$ $\quad B \to$ $\boxed{aBb}$ | $\boxed{\Lambda}$

or

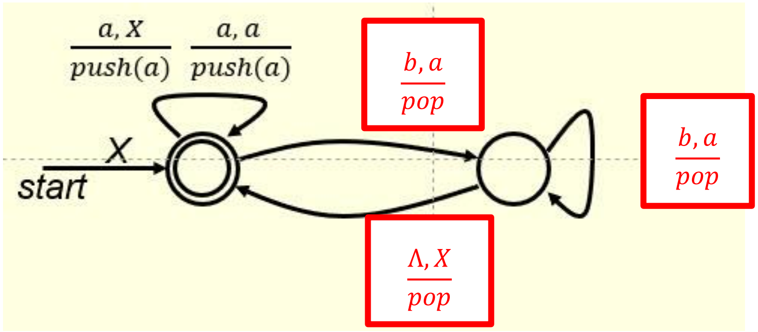
$S \to A \mid B \quad A \to$ $\boxed{aAb}$ | $\boxed{\Lambda}$ $\quad B \to$ $\boxed{aB}$ | $\boxed{\Lambda}$

(4 points)

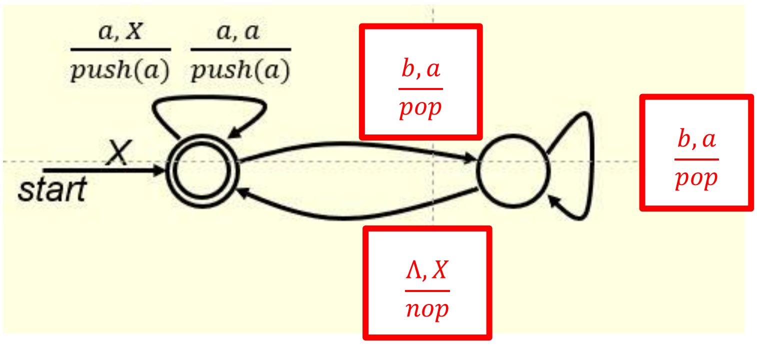
(you only need to answer one case here, either one). The language contains Λ as an element. Now consider the case k = 1 and consider the input string ab. When the first symbol is scanned, we get an 'a'. This information alone is not enough for us to make a proper choice. So we don't even know what to do with the first step in the parsing process.

For k = 2, if we consider the input string aabb, we face the same problem. For any k > 2, the input string $a^k b^k$ would cause exactly the same problem. So this grammar is not LL(k) for any k.

On the other hand, by putting proper instructions into the blanks of the following figure, we get a deterministic final-state PDA that accepts the language $\{a^m, a^n b^n \mid m, n \in N\}$.



or



(3 points)

(again, you only need to answer one case here, either one). Hence, this language is indeed deterministic C-F, but not LL(k) for any k.

6. (4 points)

Fill out the following blanks for the instructions of a Turing machine that accepts the language $\{a^n b \mid n \in N\}$. Use smallest possible non-negative integers to represent the states of the TM.

(0, a, a, R, $\boxed{0}$)
($\boxed{0}$, b, b, R, $\boxed{1}$)
($\boxed{1}$, Λ, Λ, S, halt)

7. (6 points)

Fill out the following blanks for the instructions of a Turing machine that accepts the language $\{a^n bb \mid n \in N\}$. Use smallest possible non-negative integers to represent the states of the TM.

(0, a, a, R, $\boxed{0}$)
($\boxed{0}$, b, b, R, $\boxed{1}$)
($\boxed{1}$, b, b, R, $\boxed{2}$)
($\boxed{2}$, Λ, Λ, S, halt)