CS3133 - A Term 2009: Foundations of Computer Science Prof. Carolina Ruiz

Homework 1

WPI *By Li Feng, Shweta Srivastava, and Carolina Ruiz*

Chapter 2

Problem 1: (10 Points) Exercise 2.1.

Solution 1: Let ω be a string in Let Σ^*, the length of string ω, $Length(\omega)$, is defined as follows:

- **Basis**: $Length(\omega) = 0$, if $\omega = \lambda$.
- **Recursive Step**: $Length(\omega) = 1 + Length(y)$ where $\omega = ay$, $a \in \Sigma$, $y \in \Sigma^*$ and $\omega \in \Sigma^*$.

Problem 2: (10 Points) Exercise 2.4.

Solution 2:
a) Strings in set XY are $\{aa, aab, aaab, bb, bbb, bbab\}$

b) Strings of length 6 in X* are $\{aaaaaa, bbbbbb, aaaabb, aabbaa, bbaaaa, aabbbb, bbbbaa, bbaabb\}$

c) Strings of length 3 or less in Y* are $\{\lambda, b, ab, bb, bab, abb, bbb\}$

d) Strings of length 4 or less in X*Y* can be computed in the follwing way:
X* elements of legth 4 or less = $\{\lambda, aa, bb, aaaa, aabb, bbaa, bbbb\}$
Y* elements of legth 4 or less= $\{\lambda, b, ab, bb, bab, abb, bbb, abab, babb, bbab, abbb, bbbb\}$
Therefore elements of X*Y* of length four or less are:
$\{\lambda, b, ab, bb, bab, abb, bbb, abab, babb, bbab, abbb, bbbb, aa, aabaaab, aabb, aaaa, aabb, bbaa\}$

Problem 3: (10 Points) Exercise 2.5.

Solution 3:
a) Assume that L_0 denotes the set of all of the strings in the language L that are generated with zero applications of the recursive step (i.e., the basis), and L_i denotes the set of all of the strings in the language L that are generated with exactly i applications of the recursive step, for $i \geq 0$.

$L_0 = \{b\}$

$L_1 = \{bb, bab, bba\}$

$L_2 = \{bbb, babb, bbab, bbab, babab, bbaab, bbba, babba, bbaba, bbba, bbaba, bbbaa\}$

b) The string $bbaaba$ does not belong to L.

Explanation: If $bbaaba$ were in L, the only two possible ways to have generated it would be:

1. If the string $u_1 = bbaa$ were in L, because if it were, then applying the recursive step $u_1 ba$ will produce the string we want. But $u_1 = bbaa$ is not in L, because the only way to construct u_1 using the recursive step would be if $u_2 = ba$ were in L. But $u_2 = ba$ is not in L, because the only way to construct u_2 using the recursive step would be if $u_3 = \lambda$ were in L, but it is not.

2. If the string $u_1 = baab$ were in L, because if it were, then applying the recursive step $bu_1 a$ will produce the string we want. But $u_1 = baab$ is not in L, because if it were then EITHER baa would belong in L (but it can't because the recursive step would only construct it if ba were in L but it is not because λ in not in L); OR ba would belong in L, but we should showed it is not.

c) The string $bbaaaabb$ does not belong to L.

Explanation: Given the fact that $w = bbaaaabb$ ends with two bs, then the only way in which w could have been generated is using the recursive step $u_1 b$ where $u_1 = bbaaaab$ belongs to L. Now, let's see if u_1 belongs to L. If it did, it must have been generated by either using the recursive step ub or the recursive step uab.

- Hypothesis 1. $u_1 = bbaaaab$ was generated using the recursive step $u_2 b$ where $u_2 = bbaaaa$ belongs to L. Given the fact that u_2 ends on aa, then the only way it could have been generated is using the recursive step $bu_3 a$ where $u_3 = baaa$ belongs to L. Similarly, u_3 must have been generated from aa. But, aa does not belong to L, as each string in L contains at least one b (see basis). So hypothesis 1 fails.

- Hypothesis 2. $u_1 = bbaaaab$ was generated using the recursive step $u_4 ab$ where $u_4 = bbaaa$ belongs to L. Given the fact that u_4 ends on aa, then the only way it could have been generated is using the recursive step $bu_5 a$ where $u_5 = baa$ belongs to L. Similarly, u_5 must have been generated from a. But, a does not belong to L, as each string in L contains at least one b (see basis). So hypothesis 2 fails.

Hence, $w = bbaaaabb$ cannot belong to L since it cannot have been constructed from the basis using the recursive steps.

Problem 4: (10 Points) Exercise 2.8.

Solution 4:
Let L be the set of strings over $\Sigma = a, b$ which contain twice as many a's as b's, the language L can be defined recursively as follows:

- **Basis**: $\lambda \in L$.
- **Recursive Step**: if $u \in L$ and u can be written as $u = xyz\omega$, where $x, y, z, \omega \in \Sigma^*$, thus:
  1. $xayazb\omega \in L$,
  2. $xaybza\omega \in L$, and
  3. $xbyaza\omega \in L$
- **Closure**: A string u is L only if string u can be generated from λ using a finite number of recursive steps.

Problem 5: (10 Points) Exercise 2.14.

Solution 5: $a^* b^* c^*$

Problem 6: (10 Points) Exercise 2.16.

Solution 6: $(a \cup b \cup c)(a \cup b \cup c)(a \cup b \cup c)$
[You can abbreviate this regular expression as $(a \cup b \cup c)^3$]

Problem 7: (10 Points) Exercise 2.25.

Solution 7: $(a \cup bc \cup c)^*$

Problem 8: (10 Points) Exercise 2.26.

Solution 8: $(b^*ab^*ab^*ab*)^* \cup b^*$

Problem 9: (10 Points) Exercise 2.29.

Solution 9: $(b \cup c \cup ab \cup ac)^*a \cup (b \cup c \cup ab \cup ac)^* = (b \cup c \cup ab \cup ac)^*(a \cup \lambda)$

Problem 10: (10 Points) Exercise 2.34.

Solution 10:

Since the string should contain bb in any position and the length of the string must be odd, then the string should be of the shape: <u>odd-string</u> bb <u>even-string</u> $\bigcup$ <u>even-string</u> bb <u>odd-string</u>
where <u>even-string</u> and <u>odd-string</u> are just a shortcuts:
<u>even-string</u> = $((a \cup b)(a \cup b))^*$, and
<u>odd-string</u> = <u>even-string</u> $(a \cup b) = ((a \cup b)(a \cup b))^*(a \cup b)$

Note that the empty string λ belongs to <u>even-string</u>. Hence, the expression above allows bb to appear at the beginning, or at the end of the string, in addition to in the middle of the string. Now, writing this description as a regular expression, we have:

$$(((a \cup b)(a \cup b))^*(a \cup b)bb((a \cup b)(a \cup b))^*) \bigcup (((a \cup b)(a \cup b))^*bb((a \cup b)(a \cup b))^*(a \cup b)).$$

Problem 11: (10 Points) Exercise 2.40.

Solution 11: **a).** $[Cc]$
Output:

Cowards die many times before their deaths;
The valiant never taste of death but once.
Seeing that death, a necessary end,
Will come when it will come.

Note: grep will output all the lines containing letter c or C.

b). $[K - Z]$

Output:

The valiant never taste of death but once.
Of all the wonders that I yet have heard,
Seeing that death, a necessary end,

Will come when it will come.

Note: grep will output all the lines containing letter $K, L, \cdots, X, Y, Z$.

c). $\backslash < [a-z]\,\{6\} \backslash >$
Output:

Cowards die many times **before** their **deaths** ;
It seems to me most strange that men **should** fear;

Note: The lines contain words with exactly six lower case letters.

d). $\backslash < [a-z]\,\{6\} \backslash > | \backslash < [a-z]\,\{7\} \backslash >$
Output:

Cowards die many times **before** their **deaths**;
The **valiant** never taste of death but once.
Of all the **wonders** that I yet have heard,
It seems to me most **strange** that men **should** fear;

Note: The lines contain words with exactly six or seven lower case letters.

Problem 12: (10 Points) Exercise 2.41.

Solution 12:

1. a number

   $[1-9][0-9]^*$

2. a street name

   $[A-Z][a-z]^+$

3. an identifier or abbreviation

   $Street|St|Road|Rd|Blvd|Ave|Avenue|Plaza|Pl$

   Note: it could be many, we don't expect you have a complete set here.

Summary: $[1-9][0-9]^*\square^+[A-Z][a-z]^+\square^+[Street|St|\cdots|Pl]$
Note: $\square^+$ denotes the space chars between numbers, street names and an identifier. $\square$ here means a *space char*.

Chapter 3

Problem 13: (10 Points) Exercise 3.2.

Solution 13:

a) Leftmost derivation for $aabbba$

$$\begin{array}{l} \text{DerivationSteps} \\ S \Rightarrow ASB \\ \ \Rightarrow aAbSB \\ \ \Rightarrow aaAbbSB \\ \ \Rightarrow aabbSB \\ \ \Rightarrow aabbB \\ \ \Rightarrow aabbba \end{array}$$

b) Rightmost derivation for $abaabbbabbaa$

$$\begin{array}{l} \text{DerivationSteps} \\ S \Rightarrow ASB \\ \ \Rightarrow ASbBa \\ \ \Rightarrow ASbbaa \\ \ \Rightarrow AASBbbaa \\ \ \Rightarrow AASbabbaa \\ \ \Rightarrow AAbabbaa \\ \ \Rightarrow AaAbbabbaa \\ \ \Rightarrow AaaAbbbabbaa \\ \ \Rightarrow Aaabbbabbaa \\ \ \Rightarrow aAbaabbbabbaa \\ \ \Rightarrow abaabbbabbaa \end{array}$$

c) See derivation trees for parts (a) and (b) included on the last pages of these solutions.

d) $L = \{(a^n b^n)^k (b^m a^m)^j | n \geq 0, k \geq 0, m > 0, j > 0\} \cup \{\lambda\}$

Problem 14: (10 Points) Exercise 3.6 Part (c).

Solution 14: $(ab)^n (cd)^m (ba)^m (dc)^n$, where $m, n \geq 0$.

Problem 15: (10 Points) Exercise 3.8

Solution 15: Grammar:
$S \rightarrow aScc | aAcc$
$A \rightarrow bAc | bc$
Note: λ is not in this language, since $m, n > 0$.

Problem 16: (10 Points) Exercise 3.25

Solution 16:
Grammar:

$$\begin{array}{l} S \to aA|bC|aB|bD|\lambda \\ C \to aA|bC|\lambda \\ A \to aC|bA \\ D \to aD|bB|\lambda \\ B \to aB|bD \end{array}$$

Note: A and C rules generate strings with even number of a's. B and D rules generate strings with odd number of b's. Note also, that the variable A "assumes" that an odd number of a's appear on the terminal prefix to the left of A, and the variable C "assumes" that an even number of a's appear on the terminal prefix to the left of C. Similarly, the variable B "assumes" that an even number of a's appear on the terminal prefix to the left of B, and the variable D "assumes" that an odd even number of a's appear on the terminal prefix to the left of D.

Problem 17: (10 Points) Exercise 3.32

Solution 17:
a) a^+b^+

b) Derivation 1:

$$\begin{array}{l} \mathrm{DerivationSteps} \\ S \Rightarrow aS \\ \Rightarrow aSb \\ \Rightarrow aabb \end{array}$$

Derivation 2:

$$\begin{array}{l} \mathrm{DerivationSteps} \\ S \Rightarrow Sb \\ \Rightarrow aSb \\ \Rightarrow aabb \end{array}$$

c) See the derivation trees for part (b) included on the last pages of these solutions.

d) An unambiguous grammar G' that is equivalent to G is:

$$\begin{array}{l} S \to AB \\ A \to aA\ |a \\ B \to bB\ |b \end{array}$$

Problem 18: (10 Points) Exercise 3.34

Solution 18: **a).**

$a^+b^+b \bigcup \lambda$

b). The key idea is to show that there is a unique leftmost derivation of every string in $L(G)$. In this language, λ can be generate with the rule $S \to \lambda$ only. Other string are of the form $a^i b^j$, where $i \geq 1$ and $j \geq 2$. To generate the given string $a^i b^j$, the only leftmost derivation should be in the following form:

$$\begin{array}{ll}
\text{DerivationSteps} & \text{Rule} \\
S \Rightarrow aA & S \to aA \\
\overset{i-1}{\Rightarrow} a^i A & A \to aA \\
\Rightarrow a^i bB & A \to bB \\
\overset{j-2}{\Rightarrow} a^i b b^{j-2} B & B \to bB \\
\Rightarrow a^i b b^{j-2} b & B \to b \\
\Rightarrow a^i b^j &
\end{array}$$

Clearly, each step you only can use one rule to generate string $a^i b^j$. Starting with $S \to aA$, you need exactly $i-1$ step with the rule $A \to aA$ to generate correct number of a's. Similarly, to j number of b's, you need apply $j-2$ times of the rule $B \to bB$. Otherwise, you could not get the correct string. Thus, we can say G is unambiguous.

Problem 19: (10 Points) Exercise 3.37

Solution 19:

$$\begin{array}{ll}
L_1: & S_1 \to aAbC|abc \\
& A \to aAb|ab \\
& C \to cC|c
\end{array}$$

$$\begin{array}{ll}
L_2: & S_2 \to DbBc|abc \\
& D \to aD|a \\
& B \to bBc|bc
\end{array}$$

$$\begin{array}{ll}
L(G) = L_1 \cup L_2: & S \to S_1|S_2 \\
& S_1 \to aAbC|abc \\
& S_2 \to DbBc|abc \\
& A \to aAb|ab \\
& C \to cC|c \\
& D \to aD|a \\
& B \to bBc|bc
\end{array}$$

To prove that G is ambiguous, we only need to find a string $w \in L(G)$, and there exists two leftmost derivation to generate w.

Let $w = aabbcc$, clearly, $w \in L_1$, and $w \in L_2$.

Leftmost derivation 1,

$$\begin{array}{l}
S \Rightarrow S_1 \\
\Rightarrow aAbC \\
\Rightarrow aabbC \\
\Rightarrow aabbcC \\
\Rightarrow aabbcc
\end{array}$$

Leftmost derivation 2,

$$\begin{array}{l}
S \Rightarrow S_2 \\
\Rightarrow DbBc \\
\Rightarrow aDbBc \\
\Rightarrow aabBc \\
\Rightarrow aabbcc
\end{array}$$

The string $aabbcc$ can be generated by two different leftmost derivations in G, and so G is an ambiguous grammar.

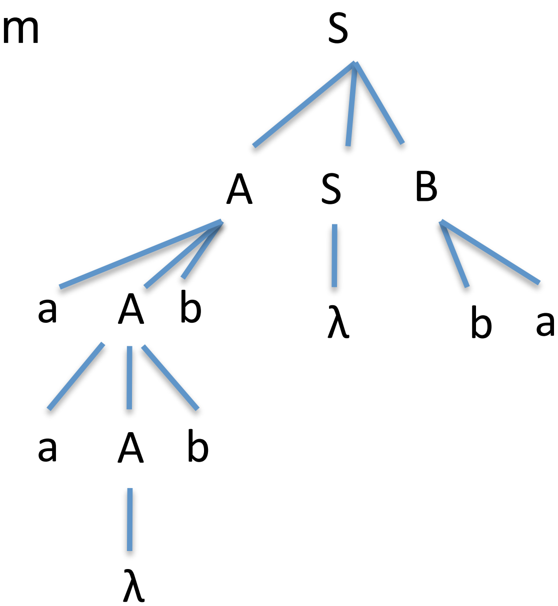
Intuitively, any grammar that generates the language $L_1 \bigcup L_2$ will have two different ways to derive strings of the form: $a^i b^i c^i$ for $i > 0$, as those string belong to both L_1 and to L_2.

Problem 20: (10 Points) Exercise 3.38

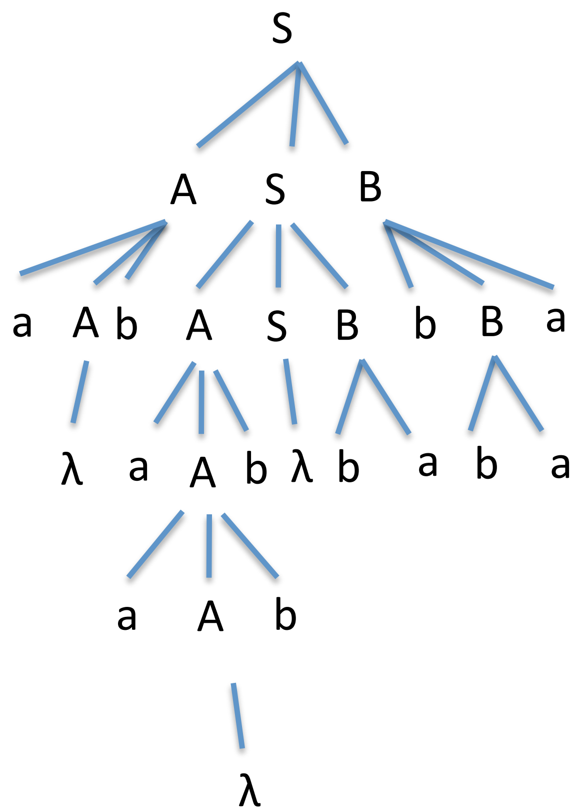
Solution 20:

$$
\begin{array}{l}
\text{DerivationSteps} \\
< Literal > \Rightarrow < FloatingPointLiteral > \\
\Rightarrow < Digits > . < Digits > < ExponentPart > \\
\Rightarrow < Digit > . < Digits > < ExponentPart > \\
\Rightarrow < NonZeroDigit > . < Digits > < ExponentPart > \\
\Rightarrow 1. < Digits > < ExponentPart > \\
\Rightarrow 1. < Digit > < ExponentPart > \\
\Rightarrow 1. < NonZeroDigit > < ExponentPart > \\
\Rightarrow 1.3 < ExponentPart > \\
\Rightarrow 1.3 < ExponentIndicator > < SignedInteger > \\
\Rightarrow 1.3e < SignedInteger > \\
\Rightarrow 1.3e < Digits > \\
\Rightarrow 1.3e < Digit > \\
\Rightarrow 1.3e < NonZeroDigit > \\
\Rightarrow 1.3e2
\end{array}
$$

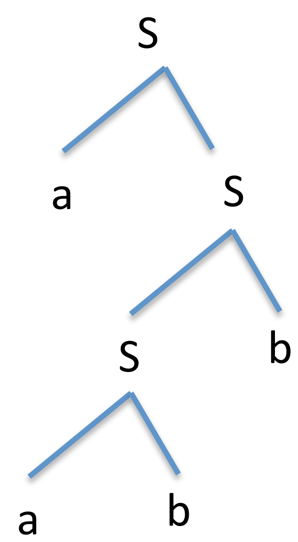
Chapter 3, problem 2.c for string aabbba



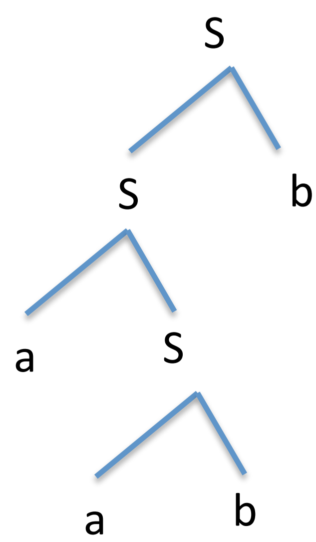
Chapter 3, problem 2.c for string abaabbbabbaa

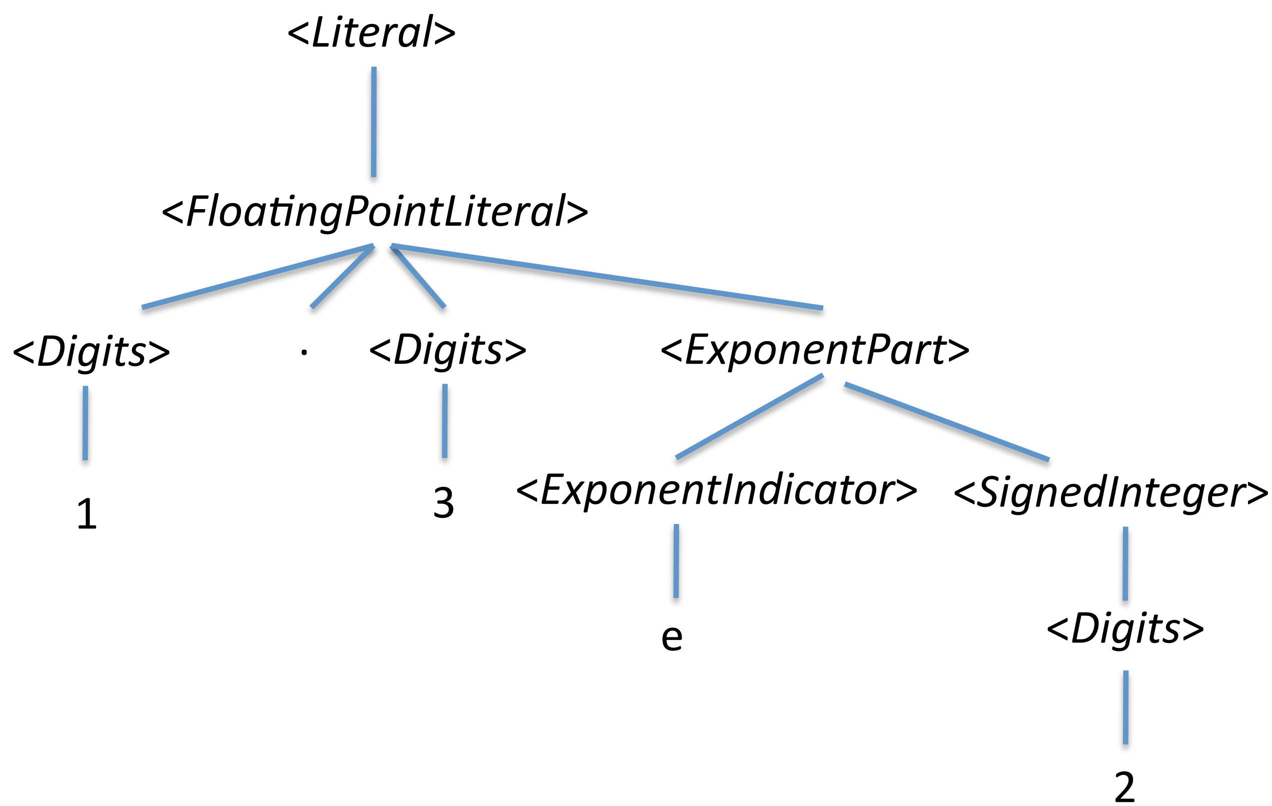


Chapter 3, problem
32.c for string aabb



Chapter 3, problem 32.c for string aabb




<Literal>
<FloatingPointLiteral>
<Digits>
.
<Digits>
<ExponentPart>
1
3
<ExponentIndicator>
<SignedInteger>
e
<Digits>
2