CS308 Homework 2
Exercises for Algorithm Design and Analysis by Li Jiang, 2016 Autumn Semester

Name:Gao Chao   Student ID:5142029014   Email: gaoc96@163.com

Coverage : Syntax Analysis.

1. (Section 4.2, Exercises 4.2.1) Consider the context-free grammar:

$$S -> S S + \mid S S * \mid a$$

and the string aa + a*.

   (a) Give a leftmost derivation for the string.
   (b) Give a rightmost derivation for the string.
   (c) Give a parse tree for the string.

Solution.

   (a) S => S S * => S S + S * => a S + S * => a a + S * => a a + a *
   (b) S => S S * => S a * => S S + a* => S a + a *=> a a + a *
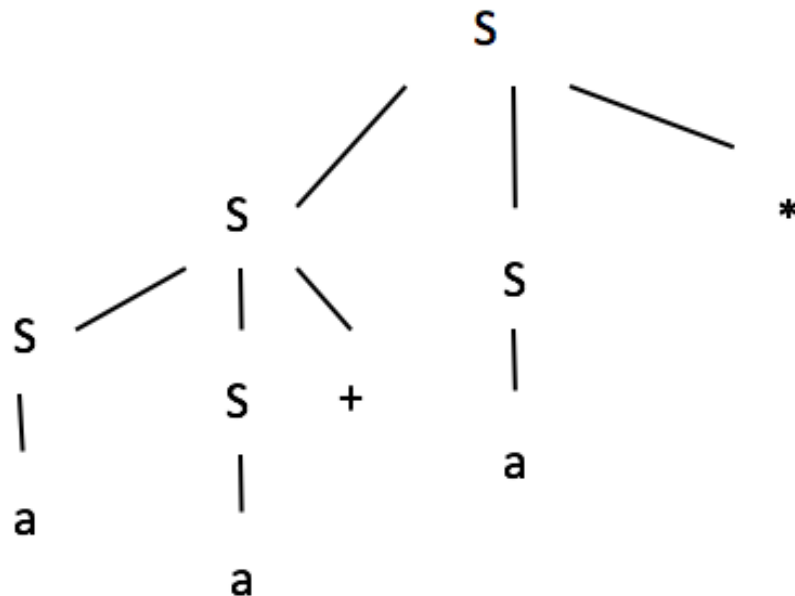   (c) below is the parse tree

图 1: parse tree

2. (Section 4.3, Exercises 4.3.1) The following is a grammar for regular expressions over symbols a and b only, using + in place of | for union, to avoid conflict with the use of vertical bar as a metasymbol in grammars:

$$rexpr -> rexpr + rterm \mid rterm$$
$$rterm -> rterm \; rfactor \mid rfactor$$
$$rfactor -> rfactor \; ^* \mid rprimary$$
$$rprimary -> a \mid b$$

(a) Left factor this grammar.

(b) Does left factoring make the grammar suitable for top-down parsing?

(c) In addition to left factoring, eliminate left recursion from the original grammar.

(d) Is the resulting grammar suitable for top-down parsing?

Solution.

(a) There is no common non-empty prefix, so we can't do left factoring, this grammar stays the same.

(b) No. Because this is the left-recursive grammar. If we use top-down parsing, it will get into an infinite loop.

(c) Below is the grammar after left recursion elimination.

$$rexpr -> rterm \; A$$
$$A -> + \; rterm \; A \mid \epsilon$$
$$rterm -> rfactor \; B$$
$$B -> rfactor \; B \mid \epsilon$$
$$rfactor -> rprimary \; C$$
$$C \; \text{->} \; ^* \; C \mid \epsilon$$
$$rprimary -> a \mid b$$

(d) Yes. Because there is no left recursion.

□

3. (Section 4.4, Exercises 4.4.3) Compute FIRST and FOLLOW for the grammar:

   (a) S − > 0 S 1 | 0 1 (with string 000111).
   (b) S − > + S S | * S S | a (with string + * aaa).

   Solution.

   (a) FIRST(S) = [0], FOLLOW(S) = [1, $].
   (b) FIRST(S) = [+, *, a], FOLLOW(S) = [+, *, a, $].

   □