Lab 7 - Solution

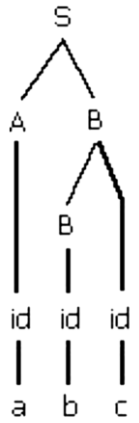
LL(1) Parsing

Exercise 1

Prove that the following grammar is ambiguous:

S → A B
A → A id | id
B → B id | id
id → a | b | c

Solution



One tree representing the substring "abc"

Another tree representing the substring "abc"

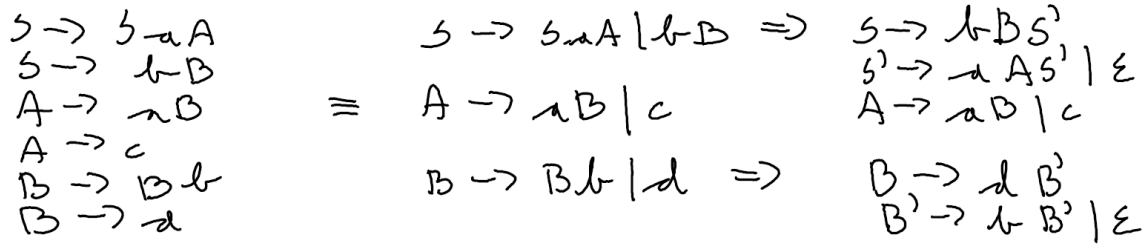
As it can be seen in the figures above, the grammar provided for this exercise is ambiguous. More than one tree can be constructed to represent the same substring.

Exercise 2

The following grammar has left recursion. Find an equivalent grammar without left recursion.

S →SaA
S → bB
A → aB
A → c
B → Bb
B →d

Solution



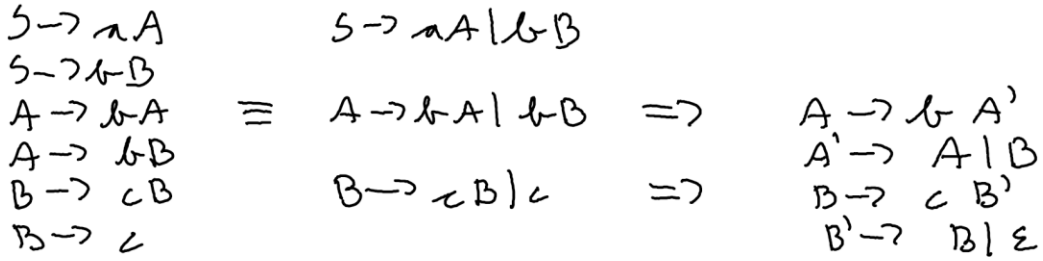
Note: S, A, et B are non-terminals. S is the starting symbol, the symbols a, b, c, and d are terminals.

Exercise 3

We consider the following grammar (a, b, and c are terminals). Use left factoring to find an equivalent grammar.

S → aA
S → bB
A → bA
A → bB
B → cB
B → c

Solution



Exercise 4

Determine the sets FIRST and FOLLOW for all non-terminals in the following grammar:

bexpr → bterm A'
A' → **or** bterm A' | **ε**
bterm → bfactor B'
B' → **and** bfactor B' | **ε**
bfactor → **not** bfactor | **(**bexpr**)** | **true** | **false**

Solution

(a) FIRST(bexpr) = First(bterm) = First(bfactor) = {not, (, true, false}

FIRST(A') = {or, ε}

FIRST(B') = {and, ε}

(b) FOLLOW(bexpr) = Follow(A') = {), $}

FOLLOW(bterm) = Follow(B') = {First(A') + Follow(A')

= {or,), $}

FOLLOW(bfactor) = {First(B') + Follow(B')} = {and, or,), $}

Exercise 5

Determine the sets FIRST and FOLLOW for all non-terminals in the following grammar:

A→BC | bA
B→a | **ε**
C→ cBc

Solution

FIRST(B) = {a, ε }
FIRST(C) = {c}
FIRST(A)=[FIRST(B) - ε]∪ FIRST(C) ∪ FIRST(b) = {a,b, c }
FOLLOW(A)={$}
FOLLOW(B)=FIRST(C) ∪ FIRST(c) = {c}
FOLLOW(C)={$}