Mohamed Khider University – Biskra.
Faculty of Exact Sciences
Computer Science department

3rd year License
Compilers
2025/2026

Work No. 3

Exercise 1: Consider the grammar G defined by:

S⟶ S ∨ S| S ∧ S |¬ S | (S) | b

1. Show that this grammar is ambiguous.
2. Construct an equivalent unambiguous grammar with the following priority rules: the negation operator ¬ has the highest priority. The operator ∧ has the second highest priority and is associative on the left. The operator ∨ has the lowest priority and is associative on the left.

Exercise 2:

Consider a command language, denoted L1, built on the vocabulary:{c, ;, *, (,), $}.

Intuitively, "c" represents a basic command, ";" a binary infix operator for sequential composition, and "*" a postfix unary operator for iteration. Parentheses are used to group multiple commands, and "$" is the end-of-file character.
The complete syntax of the language is given by the following grammar G1:
Z → C $
C → c | C ; C | C∗ | (C)

1. Show that G1 is an ambiguous grammar.
2. We now complete the definition of language L2 by considering:

— that the operator "*" has higher priority than the operator ";".
— that the operator ";" is left-associative.
We denote the language obtained by taking these new constraints into account as L2.

1. Propose a non-ambiguous grammar G2 that describes language L2.
2. Is the grammar G2 that you proposed in the previous question LL(1)? If not, transform it into a grammar G'2 that is LL(1) and describes L1. Justify your answers.
3. Illustrate how the LL(1) parser obtained in the previous question works on the sequence "c ; c ; c* $".

Exercise 3: Remove left recursion from the following grammars:

1) **S⟶A a |b A⟶A c| S d| c**

2) **S⟶S a| T S c |d T⟶TbT|ε**

Exercise 4: Consider the grammar of postfix expressions:

E → EE+| EE*| nb

1. Eliminate left recursion.
2. Factorise on the left.
3. Calculate the First and Next sets of the new grammar.
4. Is the new grammar LL(1)? If not, make it LL(1).

Exercise 4: Let G = ({X,A,B,C,D}, {a,b, c, d},X) with:

X→AB
A→CD
B→c |ε
C→aCd |ε
D→bbD |ε

1.. Calculate the **First** and **Next** sets of the grammar.
2. Construct G's predictive analysis table. Is this grammar LL(1)?
3. Analyse the word adbb$.

Exercise 6: Let G = ({S,T} {"a","b", ", "; "(", ")"}, S}) be the following grammar

S → a | b|(T)
T→T,S | S

1. Is G LL(1)?
2. Eliminate left recursion and factorise if necessary.
3. Is the new grammar LL(1)?

Exercise 7: Let G be the following grammar:

S → X | Yc
X →Xa | ε
Y →Yb | d

1. What is the language described by this grammar?
2. Calculate the **First** and **Next** sets of the grammar.
3. This grammar is not LL(1): why?
4. Give a grammar G1 that describes the same language. Justify using the analysis table that G1 is an LL(1) grammar.

Exercise 8: Consider the grammar G of terminals {a; x; d; e}, non-terminals {S; A; B}, axiom S, and productions:

S→ eAd | eB
A→ aA | a | x
B→ x

1. It is clear that this grammar is not LL(1): why (in detail)? Transform it into an LL(1) grammar G1.
2. Construct the LL(1) analysis table for G1.
3. Parse the word "*eaxd*" using an LL(1) parser.

Exercise 9: Consider the following arithmetic expression grammar:

E →T + E | T- E | T
T→F *T | F/T | F
F→P |- P
P→ Q ↑ P| Q
Q → a |b|c |(E)| *f*(E;E)

1. What prevents this grammar from being LL(1)?
2. Transform it into an LL(1) grammar?