University of Khenchela
Faculty of Science and Technology

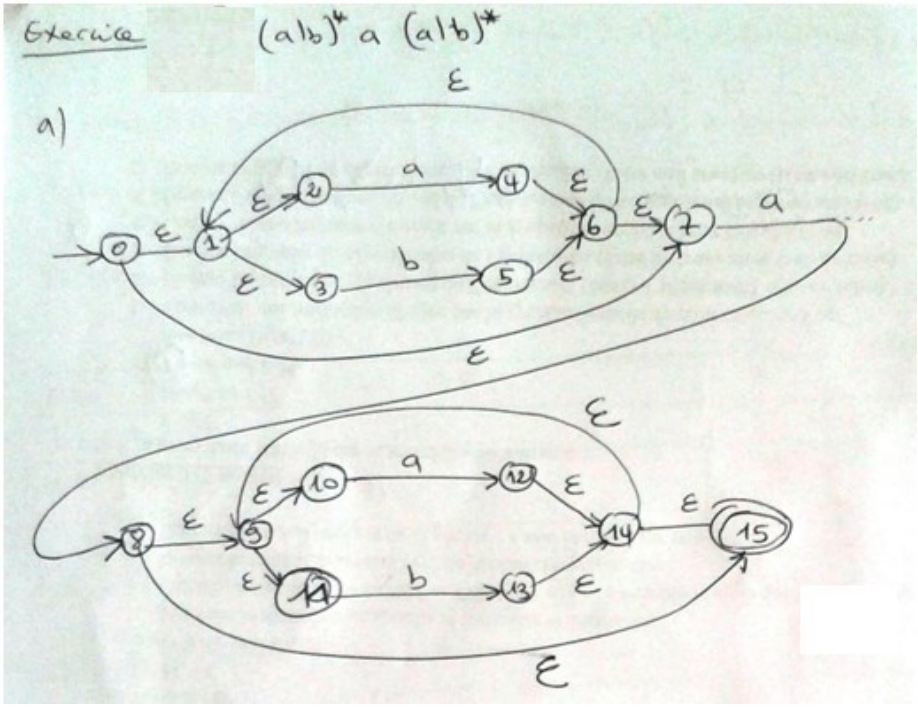
3 SI **Duration: 1h30, Date: 21/01/2025**

Final Exam in Compilation (solution)

Exercise 1 (3 points)

1. Uppercase [A-Z] (1.5 point)
   Lettre[a-zA-Z]
   Chiffre[0-9]
   { Uppercase |_}({Lettre}|{Chiffre})*

2. (1.5 point)



Exercise 2 (7 points)

1. FIRST and FOLLOW sets

FIRST sets

- FIRST(A) = { b, c } (from A → bc | cb)
- FIRST(B) = { a, c } (from B → ac | ca)
- FIRST(C) = { b, a } (from C → ba | ab)

For S:

S → aA | bB | cC

So:

- FIRST(S) = { a, b, c }

FOLLOW sets

S is the start symbol, so:

- FOLLOW(S) = { $ }

Now compute others:

From S → aA
A is at the end, so FOLLOW(A) ⊇ FOLLOW(S) = { $ }

From S → bB
B is at the end, so FOLLOW(B) ⊇ FOLLOW(S) = { $ }
From S → cC
C is at the end, so FOLLOW(C) ⊇ FOLLOW(S) = { $ }
Thus:

- FOLLOW(A) = { $ }
- FOLLOW(B) = { $ }
- FOLLOW(C) = { $ }

Non-terminal	**FIRST**	FOLLOW
S	{a, b ,c }	{ $ }
A	{ b, c}	{ $ }
B	{ a, c}	{ $ }
C	{ b, a}	{ $ }

2. LL(1) Parsing Table

Terminals: a, b, c, $
Non-terminals: S, A, B, C

	a	**b**	**c**	**$**
S	S → aA	S → bB	S → cC	
A		A → bc	A → cb	
B	B → ac		B → ca	
C	C → ab	C → ba		

No conflicts appear, so the grammar is **LL(1)**.

3. Parsing Trace for the Input `abc`

Stack	**Input**	**Action**
S$	abc$	S → aA
aA$	abc$	match a
A$	bc$	A → bc
bc$	bc$	match b
c$	c$	match c
$	$	accept

Exercise 3 (5 points)

1. Compute all LR(0) item sets

Augment the grammar:

0. S' → · E
1. E → not E

2. E → F
3. F → true

Compute all LR(0) item sets

- I0 = {S' → · E, E → · not E, E → · F, F → · true}
- I1 = Δ(I0, E)= {S' → E ·}
- I2 = Δ(I0, F){E → F ·}
- I3 = Δ(I0, not)={E → not · E, E → · not E, E → · F, F → · true}
- I4 = Δ(I0, true) = {F → true ·}
- I5 = Δ(I3, E) = {E → not E ·}
- I6 = Δ(I3, F) = {E → F ·}

2. **Compute FIRST and FOLLOW sets**

Step 2a: FIRST sets

- FIRST(F) = { true } (since F → true)
- FIRST(E) = { not, true } (since E → not E and E → F)

Step 2b: FOLLOW sets

- Start symbol E → FOLLOW(E) contains $
- Check productions:
  - E → not E → nothing after E → add FOLLOW(E) to E (already $)
  - E → F → nothing after F → add FOLLOW(E) to F → FOLLOW(F) = { $ }

Non-terminal	**FIRST**	FOLLOW
S’	{ not, true }	{ $ }
E	{ not, true }	{ $ }
F	{ true }	{ $ }

3. Fill the SLR table

Rules:

- If dot is before terminal → shift to the Δ state
- If dot is at the end → reduce by the corresponding rule
- If start symbol completed → accept

State table:

State	**not**	**true**	**$**	**E**	**F**
I0	s3	s4		1	2
I1			acc		
I2	r2	r2	r2		
I3	s3	s4		5	6
I4	r3	r3	r3		
I5	r1	r1	r1		
I6	r2	r2	r2		

Exercise 4 (5 points)

Solution

a) Left Recursion Elimination and L-Attributed SDD

a) Eliminate Left Recursion

Original:

$E \rightarrow E$ or $E \mid F$

Transformed grammar:

$E \rightarrow F\ E'$

$E' \rightarrow$ or $F\ E' \mid \varepsilon$

$F \rightarrow (E) \mid$ true $\mid$ false

---

b) L-Attributed Syntax-Directed Definition

Production	**Semantic Rules**
1. $E \rightarrow F\ E'$	E'.inh = F.val E.val = E'.syn
2. $E' \rightarrow$ or $F\ E'_1$	E'1.inh = E'.inh OR F.val E'.syn = E'_1.syn
3. $E' \rightarrow \varepsilon$	E'.syn = E'.inh
4. $F \rightarrow (E)$	F.val = E.val
5. $F \rightarrow$ true	F.val = true
6. $F \rightarrow$ false	F.val = false

c) Give the decorated (annotated) parse trees for the following expression: **(true or false) or (false or false)**