Université Mohamed Khider – Biskra.
Faculté des Sciences Exactes
Département d'Informatique

Année universitaire: **2024/2025**
Niveau: **3ème Année Licence**
Module : **Compilation**

# TD N°3

**Exercice 1 :** On considère la grammaire G définie par :

**S⟶ S ∨ S| S ∧ S |¬ S | (S) | b**

1. Montrer que cette grammaire est ambigüe
2. Construire une grammaire équivalente non ambigüe, avec les règles de priorité suivantes : l'opérateur de négation ¬ est le plus prioritaire. L'opérateur ∧ a la seconde plus haute priorité et est associatif à gauche L'opérateur ∨ est le moins prioritaire et est associatif à gauche.

**Exercice 2 :** Supprimer la récursivité gauche des grammaires suivantes:

1. **S⟶A a |b**
   **A⟶A c| S d| c**

2. **S⟶S a| T S c |d**
   **T⟶TbT|ε**

**Exercice 3 :**

On considère un langage de commandes, noté $L_1$, construit sur le vocabulaire : {c, ;, *, (, ), $ }.
Intuitivement, "c" représente une commande de base, ";" un opérateur binaire infixé de composition séquentielle, "*" un opérateur unaire post-fixé d'itération. Les parenthèses permettent de grouper plusieurs commandes et "$" est le caractère de fin de fichier.
La syntaxe complète du langage est donnée par la grammaire $G_1$ suivante :
Z → C $
C → c | C ; C | C∗ | (C)

1. Montrez que $G_1$ est une grammaire ambigüe.
2. On complète maintenant la définition du langage L2 en considérant :
   — que l'opérateur "*" est plus prioritaire que l'opérateur " ;".
   — que l'opérateur " ;" est associatif à gauche.

On note $L_2$ le langage obtenu en prenant en compte ces nouvelles contraintes.

1. Proposez une grammaire $G_2$, non ambigüe, qui décrive le langage $L_2$.
2. La grammaire $G_2$ que vous avez proposé à la question précédente est-elle LL(1) ? Si non transformez-là en une grammaire $G'_2$ qui soit LL(1) et qui décrive $L_1$. Justifiez vos réponses.

Illustrez le fonctionnement de l'analyseur LL(1) obtenu à la question précédente sur la séquence "c ; c ; c* $"

**Exercice 4 :** Soit G = ({X,A,B,C,D}, {a,b, c, d},X) avec :

**X→AB** **C→aCd |ε**
**A→CD** **D→bbD |ε**
**B→c |ε**

1. Calculer les ensembles **Premier** et **Suivant** de la grammaire.
2. Construisez la table d'analyse prédictive de G. Cette grammaire est-elle LL(1) ?
3. Analysez le mot adbb$.

**Exercice 5:** Soit la grammaire G :

**S → iBae**
**B → TB | ε**
**T → [eD] | di**
**D → ed | ε**

1. Calculer les ensembles Premier et Suivant de la grammaire.
2. Calculer la table d'analyse LL(1) pour G.

## Exercice 6 :

Soit G = ({S,T} {"a","b", ", "; "(", ")"}, S}) la grammaire suivante
**S → a | b|(T)**
**T→T,S | S**

1. G est-elle LL(1) ?
2. Eliminer la récursivité à gauche et factoriser si nécessaire.
3. La nouvelle grammaire est-elle LL(1) ?

**Exercice 7 :** Soit G la grammaire suivante :
**S → X | Yc**
**X →Xa | ε**
**Y →Yb | d**

1. Quel est le langage décrit par cette grammaire ?
2. Calculer les ensembles **Premier** et **Suivant** de la grammaire.
3. Cette grammaire n'est pas LL(1) : pourquoi ?
4. Donner une grammaire G1 qui décrit le même langage. Justifier en utilisant la table d'analyse que G1 est une grammaire LL(1).

## Exercice 8 :

On considère la grammaire G de terminaux {a; x; d; e}, de non-terminaux {S; A;B}, d'axiome S, et de productions :
S→eAd | eB
A→ aA | a | x
B → x

1. Il est évident que cette grammaire n'est pas LL(1) : pourquoi (en détail)? Transformez-la en Grammaire G1de type LL(1).
2. Construire la table d'analyse LL(1) pour G1.
3. Analyser le mot « *eaxd* » par un analyseur LL(1).;

## Exercice 9:

On considère la grammaire d'expression arithmétique suivante :

**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. Qu'est-ce qui empêche cette grammaire d'être LL(1) ?
2. Transformez-la en grammaire LL(1) ?