Université de Technologie de Compiègne

AI21 - Théorie des Langages de Programmation

TD4 : Corrigé

Exercice 1

1. On donne deux dérivations gauche du mot *abab* :

$$\begin{aligned} D1: \quad & S \underset{LM}{\Longrightarrow} aSbS \underset{LM}{\Longrightarrow} abSaSbS \underset{LM}{\Longrightarrow} abaSbS \underset{LM}{\Longrightarrow} ababS \underset{LM}{\Longrightarrow} abab \\ D2: \quad & S \underset{LM}{\Longrightarrow} aSbS \underset{LM}{\Longrightarrow} abS \underset{LM}{\Longrightarrow} abaSbS \underset{LM}{\Longrightarrow} ababS \underset{LM}{\Longrightarrow} abab \end{aligned}$$

   la grammaire est ambiguë, car Il existe deux dérivations gauches pour le mot *abab*.

2. On donne deux dérivations droite pour le mot *abab* :

$$\begin{aligned} D1: \quad & S \underset{RM}{\Longrightarrow} aSbS \underset{RM}{\Longrightarrow} aSb \underset{RM}{\Longrightarrow} abSaSb \underset{RM}{\Longrightarrow} abSab \underset{RM}{\Longrightarrow} abab \\ D2: \quad & S \underset{RM}{\Longrightarrow} aSbS \underset{RM}{\Longrightarrow} aSbaSbS \underset{RM}{\Longrightarrow} aSbaSb \underset{RM}{\Longrightarrow} aSbab \underset{RM}{\Longrightarrow} abab \end{aligned}$$

3. Les arbres de dérivations sont donnés par les figures (3) et (3).

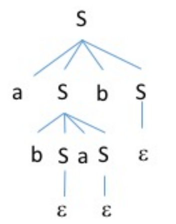

Figure 1 – D1 : Arbre de dérivation

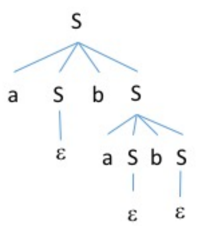

Figure 2 – D2 : Arbre de dérivation

4. Cette grammaire génère le langage L composé de toutes las chaines sur a et b tel que le nombre de a est égal au nombre de b.
   Démonstration :
   Soit l'hypothèse suivante, H : Le langage L est composé de n'importe quel mot sur a et b tel que le nombre de a est égale au nombre de b. On décompose cette hypothèse en deux parties :
   — Soit w_n un mot du langage L, composé de n caractères. On note par $|w_n|_a$ et $|w_n|_b$ le nombre de a et de b, respectivement, dans le mot w_n.
   H_1 : $|w_n|_a = |w_n|_b$, en conséquence n est pair.

— H_2 : La grammaire génère tous les mots possibles sur a et b tel que la dernière condition est vraie.

Vérification :

— pour $n = 0$ la grammaire peut générer ϵ .

— pour $n = 2$ la grammaire peut générer :

  — $S \underset{LM}{\Longrightarrow} aSbS \underset{LM}{\Longrightarrow} abS \underset{LM}{\Longrightarrow} ab$

  — $S \underset{LM}{\Longrightarrow} bSaS \underset{LM}{\Longrightarrow} baS \underset{LM}{\Longrightarrow} ba$

  Il est à noter qu'il y a 4 mots possibles sur a et b de longueur 2 : aa, ab, ba, bb. Seuls ab et ba vérifient la conditions H_1. En conséquence : H_1 et H_2 sont vraies pour $n = 0$ et $n = 2$.

On suppose que H est vraie (en particulier H_1 et H_2) jusqu'à l'ordre n.

Démonstration : Dans la suite on démontre H pour l'ordre $n + 1$.

Les mots w_{n+2} sont obtenus comme suit :

— $w_{n+2} = aw_lbw_m$ (en utilisant $S \longrightarrow aSbS$)

— $w_{n+2} = bw_law_m$ (en utilisant $S \longrightarrow bSaS$)

Il est clair que $l + m = n$. En utilisant l'hypothèse de récurrence sur l et m, on remarque que H_1 est vraie pour w_l et w_m. On a :

$$\begin{cases} |w_{n+2}|_a = 1 + |w_l|_a + |w_m|_a \\ |w_{n+2}|_b = 1 + |w_l|_b + |w_m|_b \end{cases}$$

On conclut que $|w_{n+2}|_a = |w_{n+2}|_b$ car $|w_l|_a = |w_l|_b$ et $|w_m|_a = |w_m|_b$ (Hypothèse de récurrence).

Concernant la deuxième partie de la démonstration Supposons qu'il existe un mot W de $n+2$ caractères ayant le même nombre de a que de b, qui n'est pas généré par la grammaire.

Supposons que W commence par a (Le même raisonnement est valide pour un mot qui commence par b, ce cas ne sera pas détaillé dans la démonstration). Soit k le plus petit indice d'un caractère b tel que le nombre de a dans $W[1..k]$ est égal au nombre de b dans $W[1..k]$ (où $W[1..k]$ est la chaîne formée par les k premières caractères de la chaîne W).

Exemple : considérons le mot "aababbab", on aura $k = 6$, $W[1..6] = "aababb"$. Dans $W[1..6]$ le nombre de a est égal au nombre b est égal à 3.

En tenant compte que le premier caractère de W est égal à a et que le $k^{ème}$ caractère est égal à b on peut récrire W de la manière suivante :

$$W = aw_lbw_m, \text{ avec } l = k - 2, m = n - k$$

Si W existe alors w_l et / ou w_m ne sont pas générés par la grammaire. Absurde, on contredit l'hypothèse de récurrence pour w_l et / ou w_m où H est vrai car $l, m \leq n$.

Exercice 2

Précision : l'ensemble des terminaux de la grammaire est $T = \{(,), a, b, *, |\}$

1. la grammaire génère des expressions régulières exclusivement sur a et b (impossible d'avoir ϵ). L'expression peut contenir les méta symboles parenthèses et peut être composée des 3 opérateurs : concaténation, disjonction et fermeture.

Démonstration :

Soit w_k un mot du langage engendré par la grammaire obtenu après avoir appliqué k règles de production.

H : w_k est une expression régulière définie sur a et b. De plus, w_k peut contenir les méta symboles parenthèses et peut être composée des 3 opérateurs : concaténation, disjonction et fermeture.

Vérification : On remarque que la grammaire génère $w_1 = a$ et $w_1 = b$ deux expression régulières obtenues après avoir appliqué une règle de production (une des deux dernières).

On Suppose que H Vraie,$\forall k \leq n$.

On démontre que H est vraie pour $k = n + 1$. En utilisant les 4 premières règles de production de la grammaire, on remarque que :

$$w_{n+1} = \begin{cases} w_i w_j, & \forall i, j > 0; i + j = n \\ w_i | w_j, & \forall i, j > 0; i + j = n \\ w_i *, & i = n \\ (w_i) & i = n \end{cases} \tag{1}$$

On note que H est vraie pour i et j. A partir de (1), on conclut que que w_{n+1} est bien une expression régulière vérifiant les conditions listées ci-dessus.

2. On considère le mot aba, et on donne ces deux dérivations :

$$D1: \quad R \underset{LM}{\Longrightarrow} RR \underset{LM}{\Longrightarrow} aR \underset{LM}{\Longrightarrow} aRR \underset{LM}{\Longrightarrow} abR \underset{LM}{\Longrightarrow} aba \tag{2}$$

$$D2: \quad R \underset{LM}{\Longrightarrow} RR \underset{LM}{\Longrightarrow} RRR \underset{LM}{\Longrightarrow} aRR \underset{LM}{\Longrightarrow} abR \underset{LM}{\Longrightarrow} aba \tag{3}$$

Exercice 3

1. L'ensemble des terminaux de cette grammaire est $T = \{*, /, +, -, (,), id, nb\}$
2. Soit la grammaire $G = (T = \{*, /, +, -, (,), id, nb\}, V = \{E, T, F\}, E, P)$, avec P :

$$\begin{array}{lllll} E & \to E + T & | & E - T & | & T \\ T & \to T * F & | & T/F & | & F \\ F & \to (E) & | & id & | & nb \end{array}$$

Attention la grammaire suivante G_1, ou toute grammaire équivalente, ne gère pas la priorité de $*$ et $/$ par rapport à $+$ et $-$:

Soit la grammaire $G_1 = (T = \{*, /, +, -, (,), id, nb\}, V = \{E\}, E, P)$, avec P :

$$\begin{array}{lllllll} E & \to E + E & | & E - E & | & E * E & | & E/E \\ E & \to (E) & | & nb & | & id \end{array}$$

La grammaire G_1 est ambiguë. On donne deux dérivation gauche du mot "$id + id * id$" :

$$D1: \quad E \underset{LM}{\Longrightarrow} E + E \underset{LM}{\Longrightarrow} id + E \underset{LM}{\Longrightarrow} id + E * E \underset{LM}{\Longrightarrow} id + id * E \underset{LM}{\Longrightarrow} id + id * id$$

$$D2: \quad E \underset{LM}{\Longrightarrow} E * E \underset{LM}{\Longrightarrow} E + E * E \underset{LM}{\Longrightarrow} id + E * E \underset{LM}{\Longrightarrow} id + id * E \underset{LM}{\Longrightarrow} id + id * id$$

Si on remplace tous les id par la valeur 5, le résultat de $D1$ est 30 alors que le résultat de $D2$ est 50. Dans $D2$ la priorité de $*$ par rapport à $+$ n'est pas vérifiée.

3. Il existe des techniques pour obtenir une grammaire non récursive à gauche. On les abordera dans les prochains cours.
4. On donne l'arbre représenté dans la figure 4

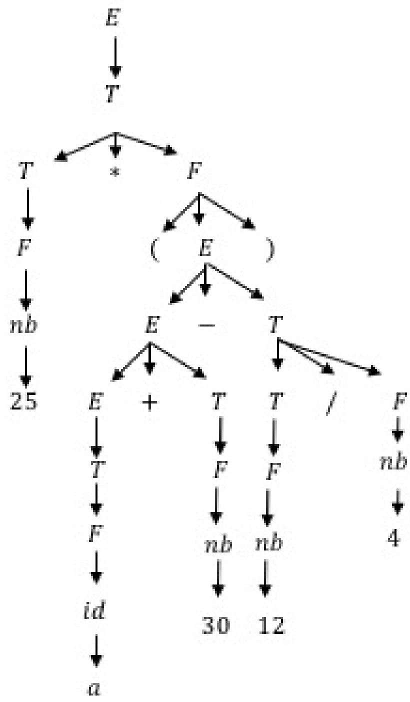

FIGURE 3 – Arbre de dérivation de la question 4

Exercice 4

Dans une forme FNC, les règles de productions sont sous la forme suivante :

$$A \ \rightarrow BC \quad | \quad a$$

Avec a est un terminal et A, B et C des variables. De plus, seul l'axiome peut générer le mot vide.

1. On considère la grammaire $G = (T\{a, b\}, V = \{S, A, B\}, S, P)$, avec P :

$$\begin{aligned} S \ &\rightarrow A \quad | \quad Bbb \\ A \ &\rightarrow aB \quad | \quad bS \quad | \quad \epsilon \\ B \ &\rightarrow ABb \quad | \quad Bb \quad | \quad \epsilon \end{aligned}$$

— **Étape 1 :** On Remarque que A et B peuvent générer le mot vide. On supprime $A \rightarrow \epsilon$ et $B \rightarrow \epsilon$. Il faudra modifier la grammaire de manière à prendre en compte cette modification. Par exemple, si on considère $S \rightarrow Bbb$, en tenant en compte que B peut générer le mot vide et qu'on supprime la règles $B \rightarrow \epsilon$, $S \rightarrow Bbb$ se transforme en $S \rightarrow Bbb \quad | \quad bb$

On obtient la grammaire $G = (T\{a, b\}, V = \{S, A, B\}, S, P)$, avec P :

$$\begin{aligned}
S &\to A \quad | \quad Bbb \quad | \quad \mathbf{bb} \quad | \quad \epsilon \\
A &\to aB \quad | \quad bS \quad | \quad \mathbf{a} \\
B &\to ABb \quad | \quad Bb \quad | \quad \mathbf{b} \quad | \quad \mathbf{Ab}
\end{aligned}$$

— **Étape 2 :** On Supprime la règle de production $S \to A$. On obtient la grammaire $G = (T\{a, b\}, V = \{S, A, B\}, S, P)$, avec P :

$$\begin{aligned}
S &\to \mathbf{aB} \quad | \quad \mathbf{bS} \quad | \quad \mathbf{a} \quad | \quad Bbb \quad | \quad bb \quad | \quad \epsilon \\
A &\to aB \quad | \quad bS \quad | \quad a \\
B &\to ABb \quad | \quad Bb \quad | \quad b \quad | \quad Ab
\end{aligned}$$

— **Étape 3 :** On modifie les règles de productions où à droite il y a une combinaison de terminaux et de non terminaux. On obtient la grammaire $G = (T\{a, b\}, V = \{S, A, B, \mathbf{C}, \mathbf{D}\}, S, P)$, avec P :

$$\begin{aligned}
\mathbf{C} &\to \mathbf{a} \\
\mathbf{D} &\to \mathbf{b} \\
S &\to \mathbf{C}B \quad | \quad \mathbf{DS} \quad | \quad a \quad | \quad B\mathbf{DD} \quad | \quad \mathbf{DD} \quad | \quad \epsilon \\
A &\to \mathbf{C}B \quad | \quad \mathbf{D}S \quad | \quad a \\
B &\to AB\mathbf{D} \quad | \quad B\mathbf{D} \quad | \quad b \quad | \quad A\mathbf{D}
\end{aligned}$$

— **Étape 4 :** On modifie les règles de productions où à droite il y a plus de 2 variables. On obtient : la grammaire $G = (T\{a, b\}, V = \{S, A, B, C, D, \mathbf{X}\}, S, P)$, avec P :

$$\begin{aligned}
\mathbf{X} &\to \mathbf{BD} \\
C &\to a \\
D &\to b \\
S &\to CB \quad | \quad DS \quad | \quad a \quad | \quad \mathbf{X}D \quad | \quad DD \quad | \quad \epsilon \\
A &\to CB \quad | \quad DS \quad | \quad a \\
B &\to A\mathbf{X} \quad | \quad BD \quad | \quad b \quad | \quad AD
\end{aligned}$$

La grammaire obtenue est bien sous forme FNC.

2. Les règles de production d'une grammaire de type 2 s'écrivent sous le format suivant :

$$A \quad \to \alpha, \quad \alpha \in (V \cup T)^*$$

On propose un algorithme à 4 étapes qui permet d'obtenir une grammaire équivalente sous forme FNC.

— **Étape 1 :** Pour chaque variable $A \in V$, tel que $A \neq S$ et que A peut générer le mot vide :
  — supprimer $A \to \epsilon$
  — pour chaque règle de production sous la forme :

$$B \to \alpha A \beta, \text{ avec } \alpha, \beta \in (V \cup T)^*$$

  ajouter l'alternative :

$$B \to \alpha\beta$$

Répéter cette étape tant qu'il existe au moins une variable différente de l'axiome qui peut générer le mot vide.
À la fin de cette étape on obtient une grammaire tel que les règles de productions s'écrivent sous le format suivant :

$$\begin{aligned} S &\to \alpha, \quad \alpha \in (V \cup T)^* \\ A &\to \beta, \quad A \neq S, \beta \in (V \cup T)^+ \end{aligned}$$

- **Étape 2 :** Pour chaque règle de production sous la forme $A \to B$:
  - Supprimer $A \to B$.
  - Pour chaque règle de production $B \to \alpha$, on ajoute la règle de production $A \to \alpha$.

  Répéter cette étape tant qu'il existe une règle de production sous la forme $A \to B$.
  À la fin de cette étape, on obtient une grammaire tel que les règles de productions s'écrivent sous le format suivant. Soit $\alpha = T \quad | \quad (V \cup T)(V \cup T)^+$.

$$\begin{aligned} S &\to \epsilon \quad | \quad \alpha \\ A &\to \alpha, \quad A \neq S \end{aligned}$$

- **Étape 3 :** Pour chaque règle de production sous la forme $A \to \alpha, \quad \alpha = (V \cup T)^* t (V \cup T)^+ | (V \cup T)^+ t (V \cup T)^*$ avec $t \in T$:
  - ajouter la règle de production $X \to t$
  - remplacer dans α, t par X

  Répéter cette étape tant qu'il existe une règle de production sous la forme $A \to \alpha, \quad \alpha = (V \cup T)^* T (V \cup T)^+ | (V \cup T)^+ T (V \cup T)^*$.
  À la fin de cette étape, on obtient une grammaire tel que les règles de productions s'écrivent sous le format suivant. Soit $\alpha = T \quad | \quad VV^+$.

$$\begin{aligned} S &\to \epsilon \quad | \quad \alpha \\ A &\to \alpha, \quad A \neq S \end{aligned}$$

- **Étape 4 :** Pour chaque règle de production sous la forme $A \to BC\alpha$, avec $\alpha = V^+$:
  - ajouter la règle de production $Y \to BC$
  - remplacer dans $A \to BC\alpha$ par $A \to Y\alpha$

  Répéter cette étape tant qu'il existe une règle de production sous la forme $A \to BC\alpha$, avec $\alpha = V^+$.
  À la fin de cette étape, on obtient une grammaire tel que les règles de productions s'écrivent sous le format suivant. Soit $\alpha = T \quad | \quad VV$.

$$\begin{aligned} S &\to \epsilon \quad | \quad \alpha \\ A &\to \alpha, \quad A \neq S \end{aligned}$$

La grammaire obtenue est sous forme FNC.