USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

Exercice 3 :

Les grammaires qui génèrent les langages suivants :

1) L'ensemble des nombres binaires (un langage infini qui ne contient pas le mot vide)

Exemple : L = {0, 1, 01, 10, 00, 11, 001, 111, 00000, 10010, 00110, … }

G=<T, N, S, P> où T={0,1} N={S} et P=

S→ 0S / 1S / 0 / 1

Remarque : Les nombres commencent par un 0 ou 1. Apres la première lettre, il y a soit 0 soit 1 et ainsi de suite. ε n'appartient pas au langage (il n'est pas un nombre binaire)

2) L'ensemble des nombres binaires sans 0 inutiles en tête

Exemple : L={0, 1, 10, 101, 100110, 111, 100000, … }

G=<T, N, S, P> où T={0,1} N={S, A} et P=

S→ 1A / 0 /*La génération commence par 1 suivi par une suite aléatoire de 0 et 1*/

A→ 0A / 1A / ε /*La suite aléatoire qui se trouve après le premier 1*/

ou bien

G=<T, N, S, P> où T={0,1} N={S,A} et P=

S→ A / 0

A→ A0 / A1 / 1

Remarque : Dans la première grammaire la construction des mots se fait de la gauche vers la droite tandis que dans la deuxième elle se fait de la droite vers la gauche. 0 est un cas unique qu'on peut avoir directement à partir de l'axiome.

3) L'ensemble des nombres binaires de longueur paire.

Exemple : L = {00, 01, 10, 11, 1001, 0010, 1111, 0000, 100101, 111111, … }

G=<T, N, S, P> où T={0,1} N={S, A} et P=

S→ 1A / 0A /* on génère le premier 0 ou le premier 1*/

A→ 0S / 1S / 0 / 1 /* soit on génère 0 ou 1 et on s'arrête. Soit on génère 0 ou 1 et on reviens à S pour continuer la génération d'un autre doublet*/

Théorie des Langages
Solutions des Exercices de la Série 1

ou bien :

G=<T, N, S, P> où T={0,1} N={S} et P=

S→ 00S / 01S / 11S / 10S / 00 / 01 / 11 / 10

Remarque : Dans la première grammaire la génération se fait lettre par lettre. Dans la deuxième deux par deux, en considérant toutes les possibilités : 00, 01, 11, 10.

4) Les nombres décimaux éventuellement signés n'ayant pas de 0 inutiles. Rappelons que la partie (optionnelle) après la virgule ne se termine pas par un 0.

Exemple : L = {0, 2020, +123.45, -67.89, 0.98, +0.14, 987.65, +4321, … }

G=<T, N, S, P> où T= {0, …, 9, +, -, .} N={S, A, B, D} et P=

S→ A / +A / -A /* Les nombres peuvent être positif, négatifs ou sans signe*/

A→ 1B / 2B/ … / 9B / 0 / 0.D /* La partie entière qui ne commence pas par 0*/

B→ 0B / 1B / 2B/ … / 9B / .D / ε /* soit on s'arrête, soit on passe a la partie décimale */

D → 0D / 1D/ … / 9D / 1 /…/ 9 /* La partie décimale qui ne se termine pas par 0 */

5) L'ensemble des noms de variables (identificateurs) en Java. Un nom de variable en Java commence par une lettre alphabétique ou le caractère underscore (_) suivi par une suite quelconque de lettres alphabétiques, de chiffres et l'underscore.

Exemple : L={a, nom, nom_pere, p1, p2, _prix, _16_04_2020, thl_2020, …..}

Remarque : Chaque mot est composé de deux parties : début (premier symbole) qui peut être une lettre ou le underscore, et le reste du nom qui est une suite de lettres, de chiffres et de l'underscore.

G=<T, N, S, P> où T ={a, ….,z , A,….,Z, 0,….,9, _},

N={<NomJava>, <Suite>, <Lettre>, <Chiffre>} et P est :

<NomJava> → < Lettre > <Suite> / _<Suite>
/*commence par une lettre alphabétique ou underscore suivi d'une suite aléatoire formée de lettres, chiffres et/ou underscore*/

<Suite> → <Lettre> <Suite> / <Chiffre> <Suite> / _<Suite> / ε
/*une suite est une lettre alphabétique ou un chiffre ou un underscore suivi d'une suite. Minimum un mot vide*/

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

<Lettre> → a /…. / z / A /…. / Z /* les différentes possibilités d'une lettre */

<Chiffre> → 0 /…. / 9 /* les différentes possibilités d'un chiffre */

Il existe une deuxième grammaire dont voici l'idée :

Générer le nom en commençant de la droite vers la gauche. La sortie tout à fait à gauche se fait par une lettre ou underscrore.

<NomJava> → <NomJava><Lettre> /<NomJava><Chiffre> /<NomJava>_ / <Lettre> /_
/* un nom de variables Java est un nom de variables Java suivi d'une lettre ou d'un chiffre ou d'un underscore. La sortie par une lettre ou un underscore: le premier symbole du nom*/

<Lettre> → a /…. / z / A /…. / Z /* les différentes possibilités d'une lettre*/

<Chiffre> → 0 /…. / 9 /* les différentes possibilités d'un chiffre*/

6) L'ensemble des mots de passe de sécurité faible, qui sont formés que des lettres ou que des chiffres.

Exemple : L = {a, 7, aa, 2020, 04, thl, usthb, 16, … }

G=<T, N, S, P> où T={0,..9, a,…z, A, …, Z}

N={{<Password>, <SuiteLettres>, <SuiteChiffres>, <Lettre>, <Chiffre>} et

S= <Password> et P =

<Password>→ <SuiteLettres> / <SuiteChiffres> /*soit des lettres, soit des chiffres*/

<SuiteLettres>→ <Lettre> <SuiteLettres> / <Lettre> /*une ou plusieurs lettres*/

<SuiteChiffres>→ <Chiffre> <SuiteChiffres> / <Chiffre> /*un ou plusieurs chiffres*/

<Lettre>→a / b /…../ z / A / B /…../ Z /* les différentes possibilités de lettres*/

<Chiffre>→0 / 1 /…. / 9 /*les différentes possibilités de chiffres*/

USTHB
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Année 2022/2023

Théorie des Langages
Solutions des Exercices de la Série 1

7) L'ensemble des mots de passe de sécurité moyenne, qui comportent au moins une lettre **et** au moins un chiffre mais aucun caractère spécial.

Exemple : L = {a7, 9aa, 20z20, s400, thl2020, usthb16alger, … }

G=<T, N, S, P> où T={0,..9, a,…z, A, …, Z}

N={<Password>, <Suite>, <Chiffre>, <Lettre> } et S= <Password>

L'ensemble des productions =

<Password> →<Suite> <Chiffre> <Suite> <Lettre> <Suite> /

<Suite> <Lettre> <Suite> <Chiffre> <Suite

/* comporte un chiffre puis une lettre ou inversement et une suite aléatoire avant, après et entre ces deux symboles */

<Suite>→<Lettre> <Suite> / <Chiffre> <Suite> / ε

<Chiffre> → 0 / 1 / … / 9 /* les différentes possibilités du chiffre */

<Lettre>→ a / b /…../ z / A / B /…../ Z /* les différentes possibilités de la lettre */

Remarques :

Le non terminal <Suite> génère la partie aléatoire (combinaison entre lettres et/ou chiffres) qui se trouve avant, après ou entre la lettre et le chiffre.

Il existe une deuxième solution, dont voici l'idée :

Tout mot de passe commence par un chiffre ou une lettre. Dans le premier cas, le premier chiffre peut être suivi par une séquence de chiffres, puis une lettre qui sera suivie par une séquence aléatoire de chiffres et de lettres. Dans le second cas, la première lettre peut être suivie par une séquence de lettres puis un chiffre qui sera suivi par une séquence aléatoire de chiffres et de lettres. Ainsi, les productions de la grammaire sont :

<Password> →<Chiffre> <SuiteChiffre> <Lettre> <Suite> /

<Lettre> <Suitelettre> <Chiffre> <Suite>

/*commence par un chiffre (resp une lettre) suivi d'une suite de chiffres (resp de suite de lettres) puis une lettre (resp un chiffre) suivi d'une suite aléatoire de lettres et chiffres */

<SuiteLettre>→<Lettre> <SuiteLettre> /ε /* Une suite de lettres est une lettre suivie d'une suite de lettres (définition récursive)*/

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

<SuiteChiffre>→<Chiffre> <SuiteChiffre> / ε /* Une suite de chiffres est un chiffre suivi d'une suite de chiffres (définition récursive)*/

<Suite>→ <Lettre><Suite> / <Chiffre> <Suite> / ε /*une suite aléatoire de lettres et de chiffres*/

<Chiffre> → 0 / 1 / … / 9 /*les différentes possibilités d'un chiffre*/

<Lettre>→ a / b /…../ z / A / B /…../ Z /*les différentes possibilités d'une lettre*/

Exercice 4 :

On donne les grammaires pour les langages suivants :

1) $L_1 = \{(ab)^n a^{2p} (ba)^m \ / \ n, p \geq 0 \text{ et } m \geq 1 \}$

Exemples : L_1={ba, abba, aaba, ababaaaaaaba, …., **ab…ab aa…aa ba…ba**, … }

Il faut remarquer que les mots sont composés de trois parties **<u>indépendantes</u>** : les **ab**, les **aa**, les **ba**

Donc, première grammaire pour L_1 est la suivante :

G=<T, N, S, P> où T={a,b} N={S, A, B, C} P :

S→ ABC /* les trois parties du mot */

A→ abA / ε /* la première partie : une suite aléatoire de ab. La sortie par ε */

B→ aaB / ε /* la deuxième partie : une suite aléatoire de aa. La sortie par ε */

C→ baC / ba /*la troisième partie : une suite aléatoire de ba. Au moins un ba (m≥1)*/

Cette grammaire n'est pas de type 3 à cause de la première règle.
Elle est de type 2 car toutes les règles sont de la forme $A \rightarrow \alpha$, $A \in N$ et $\alpha \in (T \cup N)^*$

Une deuxième grammaire pour L_1 est la suivante :

G=<T, N, S, P> où T={a,b} N={S, A, B} P :

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

S→ abS / A /* on commence par générer la suite des **ab***/

A→ aaA / B /* dans une deuxième étape on génère la suite des **aa** */

B→ baB / ba /* dans la dernière, on génère la suite des **ba** avec minimum un ba*/

Cette grammaire est de type 3. Donc c'est elle qui va être retenue.

Remarque : dans la première grammaire, les trois parties des mots ont été générées indépendamment l'une de l'autre (en parallèle). Dans la deuxième grammaire, les trois parties ont été générées dans l'ordre (en séquentiel).

2) $L_2 = \{a^{2i+3} b^{2j+2} \ / \ i, j \geq 0 \}$

Exemples : L_2 = {aaabb, aaaaabb, aaabbbb, aaaaabbbb, …., **aa…aa aaa bb…bb bb**, … }

Il faut remarquer que les mots sont composés de deux parties **<u>indépendantes</u>** : les **aa (suivis de aaa)**, les **bb (suivis de bb)**
Donc, première grammaire pour L_2 est la suivante :

G=<T, N, S, P> où T={a,b} N={S, A, B, C} P :

S→ AB /* les deux parties du mot */

A→ aaA / aaa /* la première partie : une suite aléatoire de aa qui se termine par aaa*/

B→ bbB / bb /* la deuxième partie : une suite aléatoire de bb qui se termine par bb*/

Cette grammaire n'est pas de type 3 à cause de la première règle.
Elle est de type 2 car toutes les règles sont de la forme $A\rightarrow\alpha$, $A\in N$ et $\alpha\in(T\cup N)^*$

Une deuxième grammaire pour L_2 est la suivante :

G=<T, N, S, P> où T={a,b} N={S, A, B} P :

S→ aaS / aaaB /* on commence par générer la suite des **aa** avec au moins **aaa***/

B→ bbB / bb /* on génère la suite de **bb** avec au moins **bb** */

Cette grammaire est de type 3. Donc c'est elle qui va être retenue.

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

3) $L_3=\{a^i\, b^j \;/\; i \geq j+1,\, j\geq 0\}$

Exemples : L_3 = {a, aab, aaab, aaaa, aaabb, aaaaab, …., }

Ici les deux parties du mot sont dépendantes : on ne peut pas générer les a et les b séparément.

$i \geq j+1 \;\Leftrightarrow\; i= j+k+1 \;/\; k\geq 0$

dans ce cas : $a^i\, b^j = a^{k+1}\, a^j\, b^j$ (par remplacement de i)

Donc, une grammaire pour L_3 est la suivante :

G=<T, N, S, P> où T={a,b} N={S, A, B} P :

S→ AB

A→ aA / a /* la partie des **a** seuls avec au moins un **a***/

B→ aBb / ε /* la partie $\mathbf{a^j b^j}$ qui est de la forme $\mathbf{a^n b^n}$*/

Cette grammaire n'est pas de type 3 à cause de la première et la quatrième règle.
Elle est de type 2 car toutes les règles sont de la forme $A\rightarrow\alpha$, $A\in N$ et $\alpha\in(T\cup N)^*$

Remarque : Puisque i= j+k+1, on peut écrire $a^i\, b^j = a^j\, a^{k+1}\, b^j$ avec $j,k\geq 0$ et dans ce cas on peut avoir une autre grammaire pour ce langage qui est la suivante :

G=<T, N, S, P> où T={a,b} N={S} P :

S→ aS / aSb / a /*avec chaque **b** à droite il y a un **a** à gauche mais on peut avoir des **a** supplémentaires à gauche. Minimum un a*/

4) $L_4=\{c^n\, w \;/\; w\in\{a,b\}^* \text{ et } |w| = n \text{ et } n\geq 0\}$

Exemples : L_4 = { ε, ca, cb, ccaa, ccba, cccaba, cccbba…., **ccc…(a∪b) (a∪b) (a∪b)**, … }
Les mots de L_4 sont composés de **n** occurrences de la lettre **c** suivis d'un mot de longueur **n** (composé de **n** lettres, chacune peut être **a** ou **b**).

Donc, une grammaire pour L_4 est la suivante :

G=<T, N, S, P> où T={a,b,c} N={S} P :

S→ cSa / cSb / ε

Pour chaque **c** à gauche, il y a un **a** ou un **b** à droite. La sortie se fait par **ε.**
Cette grammaire est de type 2.

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

6) $L_6=\{a^m\ b^n\ c^p \ /\ m > n$ ou $2n \leq p$ et $m,n,p\geq 0\ \}$

Exemples : $L_6 = \{a, acc, aabcc, bcc, aabbcccc, \ldots\}$
Ici, il y a deux conditions combinées par un **ou**. Donc, on peut voir L_6 comme étant l'union de deux langages : le premier respectant la première condition et le deuxième respectant la deuxième condition.
On va étudier les deux conditions séparément :

Condition 1) $m > n \Leftrightarrow m= n+k$ avec $k >0$.
Dans ce cas : $a^m\ b^n\ c^p = a^k\ a^n\ b^n\ c^p$ avec $n,p\geq 0$ et $k >0$ (k, n, p sont indépendants)
Donc la grammaire est :
$G_1=<T_1, N_1, S_1, P_1>$ où $T_1=\{a,b,c\}$ $N_1=\{S_1, A_1, B_1, C_1\}$ P_1 :

$\mathbf{S_1 \rightarrow A_1B_1C_1}$ /* les trois parties du mot */

$\mathbf{A_1 \rightarrow aA_1\ /\ a}$ /* la première partie : une suite aléatoire de a. La sortie par a (k>0)*/

$\mathbf{B_1 \rightarrow aB_1b\ /\ \varepsilon}$ /* la deuxième partie : la séquence a^nb^n. La sortie par ε */

$\mathbf{C_1 \rightarrow cC_1\ /\ \varepsilon}$ /* la troisième partie : une suite aléatoire de c. La sortie par ε */

Condition 2) $p \geq 2n \Leftrightarrow p= 2n+j$ avec $j \geq 0$.
Dans ce cas : $a^m\ b^n\ c^p \Leftrightarrow a^m\ b^n\ c^{2n}\ c^j$ avec $m, n, j \geq 0$ (m, n, j sont indépendants)
Donc la grammaire est :
$G_2=<T_2, N_2, S_2, P_2>$ où $T_2=\{a,b,c\}$ $N_2=\{S_2, A_2, B_2, C_2\}$ P_2 :

$\mathbf{S_2 \rightarrow A_2B_2C_2}$ /* les trois parties du mot */

$\mathbf{A_2 \rightarrow aA_2\ /\ \varepsilon}$ /* la première partie : une suite aléatoire de a. La sortie par ε */

$\mathbf{B_2 \rightarrow bB_2cc\ /\ \varepsilon}$ /* la deuxième partie : la séquence a^nb^n. La sortie par ε */

$\mathbf{C_2 \rightarrow cC_2\ /\ \varepsilon}$ /* la troisième partie : une suite aléatoire de c. La sortie par ε */

Ainsi, on obtient la grammaire globale, en ajoutant juste un nouvel axiome S et une nouvelle règle $S \rightarrow S_1\ /\ S_2$ et en maintenant toutes les productions de G_1 et de G_2.

Ainsi, la grammaire qui génère L_6 est la suivante :

$G=<T, N, S, P>$ où $T=\{a,b,c\}$ $N=N_1\cup N_2\cup\{S\}$, $P=P_1\cup P_2\cup \{S\rightarrow S_1\ /\ S_2\}$

Remarques : les non terminaux de la grammaire globale, sont ceux de la première, plus ceux de la deuxième, plus S (nouvel axiome). S_1 et S_2 ne sont plus axiomes. L'ensemble des productions de la grammaire globale est composé des productions de la première grammaire, plus celles de la deuxième en leur ajoutant la règle $S\rightarrow S_1\ /\ S_2$.
Cette grammaire est de type 2.

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

7) $L_7=\{a^m b^n \ / \ m \neq n$ et $m, n \geq 0 \}$

Exemples : $L_7 = \{$a, b, aaaab, abbb, aaa, bbbb, aaabbbbbb, …$\}$

On remarque que : **$m \neq n \Leftrightarrow m>n$ ou $m<n$** donc on se retrouve dans le cas de deux conditions combinées par un **ou**.

Condition 1) $m > n \Leftrightarrow m= n+k$ avec $k>0$.
Dans ce cas : $a^m b^n = a^k a^n b^n$ avec $n\geq 0$ et $k>0$ (k, n sont indépendants)

Donc la grammaire est :

$G_1=<T_1, N_1, S_1, P_1>$ où $T_1=\{a,b\}$ $N_1=\{S_1, A_1, B_1\}$ P_1 :

$S_1 \rightarrow A_1B_1$ /* les deux parties du mot */

$A_1 \rightarrow aA_1$ / a /* la première partie : une suite aléatoire de a. La sortie par a (k>0)*/

$B_1 \rightarrow aB_1b$ / ε /* la deuxième partie : la séquence a^nb^n. La sortie par ε */

Condition 2) $m < n \Leftrightarrow n= m+j$ avec $j>0$.

Dans ce cas : $a^m b^n = a^m b^m b^j$ avec $m\geq 0$ et $j>0$ (m, j sont indépendants)

Donc la grammaire est :
$G_2=<T_2, N_2, S_2, P_2>$ où $T_2=\{a,b\}$ $N_2=\{S_2, A_2, B_2\}$ P_2 :

$S_2 \rightarrow A_2B_2$ /*les deux parties du mot*/

$A_2 \rightarrow aA_2b$/ ε /*la première partie : la séquence a^mb^m. La sortie par ε */

$B_2 \rightarrow bB_2$ / b /*la deuxième partie : une suite aléatoire de b. La sortie par b (j>0)*/

Ainsi, on obtient la grammaire globale, en ajoutant juste un nouvel axiome S et une nouvelle règle $S \rightarrow S_1 / S_2$ et en maintenant toutes les productions de G_1 et de G_2.

Ainsi, la grammaire qui génère L_6 est la suivante :

$G=<T, N, S, P>$ où $T=\{a,b,c\}$ $N=N_1 \cup N_2 \cup \{S\}$, $P=P_1 \cup P_2 \cup \{S \rightarrow S_1 / S_2\}$

Cette grammaire est de type 2.

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

8) $L_8=\{wuw^R \ / \ w\in\{a, b\}^*$ et $u\in\{c\}^* \}$

Exemples : $L_8 = \{\varepsilon$, c, ccc, aca, abbcbba, aacaa, bbbcbbb, baabcbaab, … }
Les mots de L_8 sont composés comme suit : à gauche on trouve un mot quelconque, à droite son miroir et au milieu une suite aléatoire de c.

Donc, une grammaire pour L_8 est la suivante :

G=<T, N, S, P> où T={a,b,c} N={S, C} P :

S→ aSa / bSb/ C /* chaque **a** à gauche (ou **b**), lui correspond un **a** à droite (ou **b**)*/

C→ cC / ε /* pour générer la suite aléatoire de **c** qui joue le rôle de séparateur*/

Cette grammaire est de type 2.

9) $L_9=\{w\in\{a,b,c\}^* \ / \ |w|_c=3p+1, \quad p\geq 0\}$

Exemples : L_9 = { c, cb, acab, cccc, bacaacabbcbcaa, cccbcbaaaa, … }
Les mots de L_9 sont composés des mots où le nombre de **c** est un multiple de 3 plus 1 (3p+1). Aucune condition sur le nombre de **a** ou le nombre de **b**.

Remarque : d'une manière générale, le nombre de **c** dans un mot quelconque peut être : soit un multiple de 3, soit un multiple de 3 plus 1, soit un multiple de 3 plus 2. Notons qu'un nombre multiple de 3 plus 3 est multiple de 3.

On peut écrire $|w|_c \equiv r$ [3] qui se lit le nombre de **c** est congrue à **r** modulo 3 avec $r \in\{0, 1, 2\}$.
Donc, on peut répartir les mots de X* en 3 classes d'équivalences (3p, 3p+1 et 3p+2).

Si on ajoute une lettre **a** ou **b** à un mot, le nombre de **c** reste inchangé mais si on ajoute un **c**, il augmente de 1 (modulo 3), comme illustré dans le tableau suivant :

$\|w\|_c$\lettre	a	b	c
$\|w\|_c = 3p$	3p	3p	3p+1
$\|w\|_c = 3p+1$	3p+1	3p+1	3p+2
$\|w\|_c = 3p+2$	3p+2	3p+2	3p

Donc, une grammaire pour L_8 est la suivante :

G=<T, N, S, P> où T={a,b,c} N={S, A, B}

S : représente les mots qui ont le nombre de **c** multiple de 3 (3p)

USTHB
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Année 2022/2023

Théorie des Langages

Solutions des Exercices de la Série 1

A : représente les mots qui ont le nombre de **c** multiple de 3 plus 1 (3p+1).

B : représente les mots qui ont le nombre de **c** multiple de 3 plus 2 (3p+2).

L'ensemble des productions P est :

S→ aS / bS / cA /* Si on génère **a** ou **b**, le nombre de **c** ne change pas. Si on génère **c**, il devient 3p+1*/

A→ aA / bA / cB / ε /* Si on génère **a** ou **b**, le nombre de **c** ne change pas. Si on génère **c**, il devient 3p+2. Ici, on peut s'arrêter à n'importe quel moment*/

B→ aB / bB / cS /* Si on génère **a** ou **b**, le nombre de **c** ne change pas. Si on génère **c**, il redevient 3p*/

Cette grammaire est de type 3.

10) $L_{10}=\{w\in\{0, 1\}^* / w \text{ est divisible par } 3\}$

Exemples : L_{10} = { 0, 11, 011, 110, 000, 0011, 1001, 001001, 1100, … }

D'une manière générale, si $w\in\{0, 1\}^*$ alors :

w0=2*w (ajouter 0 à droite revient à multiplier w par 2. Exp : si w=1 alors w0=10 qui est 2)

w1=2*w+1 (ajouter 1 à droite revient à multiplier w par 2 et d'ajouter 1. Exp : si w=1 alors w1=11 qui est 3)

Pour notre exercice, un mot quelconque en binaire est soit un multiple de 3 (reste de sa division sur 3 est 0), soit un multiple de 3 plus 1 (reste de sa division sur 3 est 1), soit un multiple de 3 plus 2 (reste de sa division sur 3 est 2).

w\chiffre	**0**	**1**
w = 3p	3p	3p+1
w = 3p+1	3p+2	3p
w = 3p+2	3p+1	3p+2

Donc, une grammaire pour L_{10} est la suivante :

G=<T, N, S, P> où T={0,1} N={S, Z, U, D}

Où **S :** c'est l'axiome.

Z : représente les nombres binaires multiple de 3 (reste = **Z**éro).

U : représente les nombres binaires multiple de 3 plus 1 (reste = **U**n).

D : représente les nombres binaires multiple de 3 plus 2 (reste = **D**eux).

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

L'ensemble des productions P est :

S→ 0Z / 1U	**/*** Si on génère **0**, on obtient le nombre 0 qui est multiple de 3. Si on génère **1**, on obtient 1 qui est multiple de 3 plus 1*/
Z→ 0Z / 1U / ε	**/*** Si on génère **0**, le nombre reste toujours multiple de 3. Si on génère **1** il devient multiple de 3 plus 1. On ne peut s'arrêter qu'ici */
U→ 0D / 1Z	**/*** Si on génère **0**, le nombre devient multiple de 3 plus 2. Si on génère **1** il devient multiple de 3 */
D→ 0U / 1D	**/*** Si on génère **0**, le nombre devient multiple de 3 plus 1. Si on génère **1** il devient multiple de 3 plus 2 */

Cette grammaire est de type 3.

Remarque : une autre solution serait de considérer **Z** comme étant l'axiome. Dans ce cas, il faut prévoir deux sorties : une à partir de Z par 0 et l'autre à partir de U par 1 (il faut ajouter les règles Z→0 et U→1)

11) $L_{11}=\{w\in\{a,b\}^* / |w|_a = |w|_b\}$

Exemples : L_{12} = {ε, ab, ba, aabb, aabbba, ababab, baab, baabba, … }

Le L_{11} est composé des mots qui contiennent autant de **a** que de **b**. Pour les avoir, il faut générer les **a** et les **b** en même temps en considérant toutes les combinaisons possibles.

Donc, une grammaire pour L_{12} est la suivante :

G=<T, N, S, P> où T={a,b} N={S} P :

S→ aSb / bSa / SS / ε

Remarque :
Dans cette grammaire, les deux premières règles expriment le fait que chaque **a** lui correspond un **b** et inversement. La troisième pour exprimer toutes les compositions possibles d'un mot du langage.

Cette grammaire est de type 2.

Il existe une deuxième grammaire pour ce langage. Ses règles de productions sont les suivantes :

S→ SaSbS / SbSaS / ε

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

L'idée de cette grammaire est la suivante : pour chaque **a** il y a un **b** (et inversement). Avant le **a**, après le **b** ou entre le **a** et le **b** il y a des sous mots qui ont autant de **a** que de **b**.

Il existe une troisième grammaire pour ce langage qui est la suivante :

G=<T, N, S, P> où T={a,b} N={S, A, B} P :

S → aB / bA / ε /* si on génère un **a**, il doit être suivi par un mot contenant un **b** et si on génère un **b**, il doit être suivi par un mot contenant un **a***/

A→ aS / bAA /* soit on génère le a manquant, soit encore un autre **b** et il faut donc deux A pour générer les deux **a** manquants*/.

B → bS / aBB /* soit on génère le b manquant, soit encore un autre **a** et il faut donc deux B pour générer les deux **b** manquants*/.

Remarque :
Le nombre de A dans le mot indique le nombre de **a** non encore générés pour avoir le même nombre que **b**.
Le nombre de B dans le mot indique le nombre de **b** non encore générés pour avoir le même nombre que **a**.

Exemple de dérivation : On numérote les règles de la grammaire précédente dans l'ordre et on indique la règle utilisée à chaque pas de dérivation :

$S \Rightarrow_{(1)} aB \Rightarrow_{(7)} aaBB \Rightarrow_{(6)} aabSB \Rightarrow_{(2)} aabbAB \Rightarrow_{(5)} aabbbAAB \Rightarrow_{(4)} aabbbaSAB$
$\Rightarrow_{(3)} aabbbaAB \Rightarrow_{(4)} aabbbaaSB \Rightarrow_{(3)} aabbbaaB \Rightarrow_{(6)} aabbbaabS \Rightarrow_{(3)} aabbbaab$

A chaque étape de la dérivation, on a l'invariant :

nombre de <u>a</u> + nombre de <u>A</u> = nombre de <u>b</u> + nombre de <u>B</u>

USTHB
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)
Année 2022/2023

Théorie des Langages
Solutions des Exercices de la Série 1

Exercice 5 :

Pour chacune des grammaires suivantes, donner son type et le langage qu'elle génère.

Pour trouver le langage généré à partir d'une grammaire, il faut explorer toutes les possibilités des dérivations à partir de l'axiome jusqu'à l'obtention des mots composés uniquement de terminaux.

1) Soit G_1=<{a,b}, {S,A}, S, P_1> où

S→aS / aaA / bb **A→bbA / ε**

$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaS \Rightarrow \ldots\Rightarrow a^nS \Rightarrow a^n(aaA\cup bb)$ avec $n \geq 0$.
On commence par appliquer la règle récursive (règle 1) **n** fois ce qui donne **a^nS ($n\geq 0$)**. Le dernier **S** donne **aaA** ou **bb,** d'après les deux règles d'arrêt (règle 2 et 3). On remarque que les mots dérivés à partir de S sont en fonction du non terminal **A**. Donc, on va déterminer les mots qui sont dérivés à partir de A.

$A \Rightarrow bbA \Rightarrow (bb)^2A \Rightarrow (bb)^3A \Rightarrow \ldots\Rightarrow (bb)^mA \Rightarrow (bb)^m = b^{2m}$ avec $m \geq 0$.

Les mots de la forme b^{2m} sont obtenus à partir de **A** en appliquant **m** fois la règle 4 et la sortie avec **ε** se fait en appliquant la règle 5.

On remplace A dans la forme des mots obtenus à partir de S :

$S \Rightarrow \ldots \Rightarrow a^nS \Rightarrow a^n(aaA\cup bb) = a^n(aa\, b^{2m} \cup bb)$ avec $n, m \geq 0$.

Ainsi, les mot composés de terminaux obtenus par dérivation (directe ou indirecte) à partir de l'axiome ont la forme générale suivante : $\mathbf{a^n(a^2\, b^{2m} \cup b^2)}$

Donc $\mathbf{L(G_1) = \{ a^n(a^2\, b^{2m} \cup b^2) / n, m \geq 0\}}$ ou encore $\mathbf{L(G_1) = \{ a^{n+2}\, b^{2m} \cup a^nb^2) / n, m \geq 0\}}$

2) Soit G_2 =({a, b}, {S, A, B}, S, P_2) où P_2 est

S → aSab / abABab **A → Ab / ε** **B → aaB / a**

On procède à l'exploration de toutes les dérivations possibles à partir de l'axiome.

$S \Rightarrow aSab \Rightarrow a^2S(ab)^2\Rightarrow a^3S(ab)^3 \Rightarrow \ldots\Rightarrow a^nS(ab)^n \Rightarrow a^n\, abABab\, (ab)^n$ $n\geq 0$.

On commence par appliquer la règle récursive (règle 1) **n** fois ce qui donne $\mathbf{a^nS(ab)^n}$ ($n\geq 0$). Le dernier **S** donne un **abABab** par la règle d'arrêt (règle 2). On remarque que les mots dérivés à partir de S sont en fonction des non terminaux **A et B**. Donc, on va déterminer les mots qui sont dérivés à partir de **A** et à partir de **B**.

$A \Rightarrow Ab \Rightarrow Ab^2 \Rightarrow Ab^3 \Rightarrow \ldots\Rightarrow Ab^m \Rightarrow b^m$ avec $m \geq 0$.

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Solutions des Exercices de la Série 1

Les mots de la forme b^m sont obtenus à partir de **A** en appliquant **m** fois la règle 3 et la sortie avec **ε** se fait en appliquant la règle 4.

Pour le non terminal B :
$B \Rightarrow aaB \Rightarrow (aa)^2B \Rightarrow (aa)^3B \Rightarrow \ldots\ldots \Rightarrow (aa)^jB \Rightarrow (aa)^ja = a^{2j+1}$ avec $j \geq 0$.
Les mots de la forme a^{2j+1} sont obtenus à partir de **B** en appliquant **j** fois la règle 5 et la sortie avec **a** se fait en appliquant la règle 6.

On remplace A et B dans la forme des mots obtenus à partir de S :

$S \Rightarrow \ldots \Rightarrow a^nS(ab)^n \Rightarrow a^n\,abABab\,(ab)^n = a^n\,ab\,b^m\,a^{2j+1}\,ab\,(ab)^n$ avec $n, m, j \geq 0$.

Ainsi, les mot composés de terminaux obtenus par dérivation (directe ou indirecte) à partir de l'axiome ont la forme générale suivante : $\mathbf{a^{n+1}\,b^{m+1}\,a^{2j+2}\,b\,(ab)^n}$ avec $n, m, j \geq 0$. Remarquez que le plus petit mot généré par la grammaire est : $abaab = aba^2b$

Donc $\mathbf{L(G_2) = \{ a^{n+1}\,b^{m+1}\,a^{2j+2}\,b\,(ab)^n}$ avec $\mathbf{n, m, j \geq 0\}}$

3) Soit $G_3=<\{a,b,c\}, \{S\}, S, P_3>$ où P_3 :
S→aSa / aSb / c

On procède à l'exploration de toutes les dérivations possibles à partir de l'axiome.

$S \Rightarrow aS(a\cup b) \Rightarrow a^2S(a\cup b)^2 \Rightarrow a^3S(a\cup b)^3 \Rightarrow \ldots\ldots \Rightarrow a^nS(a\cup b)^n \Rightarrow a^n\,c\,(a\cup b)^n$ $n \geq 0$.

On commence par appliquer les deux règles récursives (règle 1 et 2, dans n'importe quel ordre) **n** fois ce qui donne $\mathbf{a^nS(a\cup b)^n}$. Le dernier **S** donne un **c** par la règle d'arrêt (règle 3), et on aura des mots de la forme $\mathbf{a^nc(a\cup b)^n}$ qui sont composés uniquement de terminaux. Remarquez que **c** appartient au langage, car on pouvait dériver **S** directement en **c** (règle 3).

Donc $\mathbf{L(G_3) = \{ a^nc(a\cup b)^n / n \geq 0\}}$ ou encore $\mathbf{L(G_3) = \{ a^ncw / w\in\{a,b\}}$ **et** $\mathbf{|w|=n}$ **avec** $\mathbf{n \geq 0\}}$

4) Soit $G_4=(\{a, b\}, \{S, A, B\}, S, P_4)$ où P_4 est
S → abS / abA **A→ aAB / ε** **B → bB / b**

On commence toujours par explorer les différentes dérivations à partir de l'axiome.
$S \Rightarrow abS \Rightarrow ababS \Rightarrow abababS \Rightarrow \ldots\ldots \Rightarrow (ab)^nS \Rightarrow (ab)^nabA$ avec $n \geq 0$.
On commence par appliquer la règle récursive (règle 1) **n** fois ce qui donne $\mathbf{(ab)^nS\ (n \geq 0)}$. Le dernier **S** donne **abA,** d'après la règle d'arrêt (règle 2). On remarque que les mots dérivés à partir de S sont en fonction du non terminal **A.** Donc, on va déterminer les mots qui sont dérivés à partir de A.
$A \Rightarrow aAB \Rightarrow aaABB \Rightarrow aaaABBB \Rightarrow \ldots\ldots \Rightarrow a^mAB^m \Rightarrow a^m\,B^m$ avec $m \geq 0$.

USTHB Année 2022/2023
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages

Solutions des Exercices de la Série 1

Les mots de la forme $a^m B^m$ sont obtenus à partir de **A** en appliquant **m** fois la règle 3 et la sortie avec **ε** se fait en appliquant la règle 4.

On remarque que les mots dérivés à partir de **A** sont en fonction du non terminal **B**. Donc, on va déterminer les mots qui sont dérivés à partir de **B**.

$B \Rightarrow bB \Rightarrow bbB \Rightarrow bbbB \Rightarrow \ldots\ldots \Rightarrow b^jB \Rightarrow b^jb = b^{j+1}$ avec $j \geq 0$.
Les mots de la forme b^{j+1} sont obtenus à partir de **B** en appliquant **j** fois la règle 5 et la sortie avec **b** se fait en appliquant la règle 6.

A présent, on remplace B dans A et on obtient :

$A \Rightarrow aAB \Rightarrow \ldots \Rightarrow a^m B^m \Rightarrow a^m (b^{j+1}, j \geq 0)^m$ avec avec $m \geq 0$.

Remarque :

$(b^{j+1}, j \geq 0)^m = \{b, b^2, \ldots, b^{j+1}, \ldots\} \ldots\ldots\ldots \{b, b^2, \ldots, b^{j+1}, \ldots\}$ (m fois) $m \geq 1$

$= \{b^{m+k} / k \geq 0\}$ $m \geq 1$

B génère une séquence de b avec au minimum un b et donc B^m génère une séquence de b avec au minimum m occurrences de b. Si m=0, B^m génère ε.

Donc, les mots obtenus à partir de A ont la forme :
$\{a^m (b^{j+1}, j \geq 0)^m / m \geq 0\} = \{ a^m b^{m+k} / k \geq 0 \text{ et } m \geq 1\} \cup \{\varepsilon\}$

On remarque que la nouvelle forme des mots obtenus à partir de A est formée uniquement de terminaux. Donc, on peut remplacer A dans la forme des mots obtenus à partir de S :

$S \Rightarrow abS \Rightarrow \ldots \Rightarrow (ab)^n abA \Rightarrow (ab)^n ab (\varepsilon \cup (a^m b^{m+k} / k \geq 0 \text{ et } m \geq 1))$ avec $n \geq 0$.

Ainsi, les mot composés de terminaux obtenus par dérivation (directe ou indirecte) à partir de l'axiome ont la forme générale suivante : $(ab)^n ab(\varepsilon \cup (a^m b^{k+m,} k \geq 0 \text{ et } m \geq 1))$ avec $n \geq 0$.

Donc $\mathbf{L(G_4) = \{ (ab)^{n+1} (\varepsilon \cup (a^m b^{m+k}} / k \geq 0 \text{ et } m \geq 1\mathbf{)) / n \geq 0\}}$
$\mathbf{= \{ (ab)^{n+1} a^m b^{m+k}} / k, n \geq 0 \text{ et } m \geq 1\mathbf{\} \cup \{(ab)^{n+1} / n \geq 0\}}$

Une autre forme pour LG4) :

$S \Rightarrow abS \Rightarrow \ldots \Rightarrow (ab)^n abA \Rightarrow (ab)^n ab\ a^m (b^{j+1} / j \geq 0)^m$ avec $m, n \geq 0$.
Ainsi, les mot composés de terminaux obtenus par dérivation (directe ou indirecte) à partir de l'axiome ont la forme générale suivante : $\mathbf{(ab)^{n+1} a^m (b^{j+1} / j \geq 0)^m}$ avec $n, m \geq 0$.

Donc $\mathbf{L(G_4) = \{ (ab)^{n+1} a^m (b^{j+1} / j \geq 0)^m / n, m \geq 0 \}}$

Remarque : Les deux formulations de $L(G_4)$ sont équivalentes.