USTHB Année 2019/2020
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Corrigé Devoir N°2

Exercice 1

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, …..}

Première Solution

Chaque nom de variable est composé de deux parties : -

- Le premier symbole qui peut être une lettre ou l'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*/*

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

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

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

Deuxième Solution

Un nom de variable Java est une suite de T* (T est l'alphabet du langage) qui commence par une lettre ou un underscore. A l'inverse de la solution précédente un nom de variable Java sera considéré comme un mot, qui est aussi un nom de variable Java, suivi d'une lettre ou d'un chiffre ou d'un underscore ; au minimum c'est une lettre ou un underscrore qui correspond au 1er symbole du nom.
Donc, L5={wx/ w∈L5 et x est une lettre ou undescore}∪{x/ x est une lettre}∪{underscore}}

G'5=(T, N, <NomJava>, P') où N={<NomJava>, <lettre>, <chiffre>} et P' est :

<NomJava> → <NomJava><chiffre>/<NomJava> <lettre> /<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 */*

USTHB Année 2019/2020
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Corrigé Devoir N°2

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

La question qu'on doit se poser est :
Peut-on trouver une grammaire de type 3 qui génère L5?

Dans ce cas, on peut trouver une grammaire de type 3 qui génère ce langage :
Il suffit de considérer la grammaire précédente qui génère le mot de la droite vers la gauche sans utiliser les non-terminaux <lettre> et <chiffre>.
Ainsi, la règle **<NomJava> → <NomJava><chiffre>** sera remplacée par un ensemble de dix règles où chacune correspond à un chiffre donné.

G''5=(T, {<NomJava>}, <NomJava>, P'') où P'' est :

<NomJava> → <NomJava> a /……/<NomJava>z/
<NomJava>A/……./<NomJava>Z/
<NomJava>0/……../<NomJava>9
/<NomJava>_ / _ /a./…../z/A/…../Z

Cette grammaire est régulière gauche.

8) 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, … }

Première solution
Un mot de passe comporte un chiffre puis une lettre ou inversement et une suite aléatoire avant, après et entre ces deux symboles.

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

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 */*

USTHB Année 2019/2020
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages

Corrigé Devoir N°2

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

Remarques :

1/ 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.

2/ Une solution similaire vient du constat que tout mot de passe contenant au moins un chiffre et au moins une lettre comporte forcément une lettre suivie directement d'un chiffre ou inversement. Ainsi, la première règle sera remplacée par :

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

Il existe une autre solution basée sur l'idée suivante :

Tout mot de passe commence par une lettre ou un chiffre. Dans le premier 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. Dans le second cas, le premier chiffre peut être suivi par une séquence de chiffres puis une lettre qui sera suivi par une séquence aléatoire de chiffres et de lettres.

G'8=(T, N, S, P') où N={<Password>, <SuiteChiffre>, <Suitelettre>, <lettre>, <chiffre>, <suite>}, S=<Password> et l'ensemble de productions P' :

<Password> →<chiffre> <SuiteChiffre> <lettre> <Suite> /

<lettre> <Suitelettre> <Chiffre> <Suite>

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

<SuiteChiffre>→ <Chiffre> <SuiteChiffre> / ε

*/*Une suite de chiffres est un chiffre suivi d'une suite de chiffres*/*

<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 */*

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

USTHB Année 2019/2020
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages
Corrigé Devoir N°2

Exercice 2 :

On donne les grammaires pour les langages suivants :

1) $L_3=\{a^i b^j / i \geq j+1, j\geq 0\}$
Notons que le nombre de a est strictement supérieur au nombre de b.
Exemple : 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.

On considère la condition $i \geq j+1$
$\Leftrightarrow$ **$i= j+k+1$ / $k\geq 0$**
Remplacer i dans les mots de L3 :
$a^i b^j = a^{j+k+1}b^j$

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

Première solution : $a^{j+1+K} b^j = a^{1+K} a^j b^j$ $j,k\geq 0$
On obtient deux parties indépendantes : les a^{1+K} et $a^j b^j$.
On peut utiliser deux non-terminaux A et B : A pour générer a^{1+K} et B pour générer $a^j b^j$.

Donc, une grammaire pour L_3 est la suivante :

G3=<T, N, S, P> où T={a,b} N={S, A, B} P :

S→ AB */*les deux parties du mot*/*

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

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

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

Remarques
1) La concaténation n'est pas commutative.
2) Ne pas prendre la relation j=i-k-1!!!!

Deuxième solution :
Puisque i= j+k+1, on peut écrire $a^i b^j = a^j a^{k+1} b^j$ avec $j,k\geq 0$ $^{1+K}$
Avec cette écriture, les a^{1+K} se trouvent entre a^j et b^j.
- Générer a^j et b^j à partir de S
- Utiliser un non-terminal A pour générer a^{1+K} au milieu à partir de S.

G3'=<T, N, S, P'> où T={a,b} N={S, A} et l'ensemble des productions P' :

USTHB Année 2019/2020
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages

Corrigé Devoir N°2

S→aSb/A */* A chaque b correspond un a pour avoir a^j et b^j de part et d'autre de S et la sortie par A */*

A → aA/a */*la séquence des a^{k+1} au milieu avec au minimum un a*/*

Mais, on peut utiliser un seul non-terminal comme suit :

G''5=<T, N, S, P''> où T={a,b}, N={S} et l'ensemble de productions 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*/*

Les deux grammaires précédentes ne sont pas de type 3 à cause de la règle S→aSb.
Elles sont de type 2 car toutes les règles sont de la forme A→α, A∈N et α∈(T∪N)*

Toutes les grammaires proposées sont de type 2. Peut-on trouver une grammaire de type 3 qui génère L5. Intuitivement, on peut affirmer qu'il n'est pas possible de trouver une grammaire régulière pour L3.
Si on essaye d'écrire une grammaire régulière droite, on commence par générer la séquence des a et lorsqu'on veut générer la séquence des b, on n'aura aucune information sur le nombre de a préalablement généré.

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

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.

Exercice 3

1) $L_5 = \{a^{2m} b^{2n} c^{2p} / 2m+n+1 = p, n\geq 1 \text{ et } m,p\geq 0\}$
Exemples : L_5={bbcccc, aabbccccccc, … }

D'après la condition, on remplace p par 2m+n+1 dans la forme des mots et on obtient :

$a^{2m} b^{2n} c^{2p} = a^{2m} b^{2n} c^{2(2m+n+1)}$
$= a^{2m} b^{2n} c^{4m+2n+2}$
$= a^{2m} b^{2n} c^2 c^{2n} c^{4m}$, m≥0 et n ≥1

USTHB Année 2019/2020
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages

Corrigé Devoir N°2

Dans la nouvelle forme des mots, les correspondances sont les suivantes :
Pour 2 **a** à gauche (extérieur), ça leur correspond 4 **c** à droite : $\underline{\mathbf{a^{2m}}}\ \mathbf{b^{2n}}\ \mathbf{c^2}\ \mathbf{c^{2n}}\ \underline{\mathbf{c^{4m}}}$
Pour 2 **b** à gauche (intérieur), ça leur correspond 2 **c** à droite : $\mathbf{a^{2m}}\ \underline{\mathbf{b^{2n}}}\ \mathbf{c^2}\ \underline{\mathbf{c^{2n}}}\ \mathbf{c^{4m}}$
Il y a 2 **c** au milieu.

Donc, une grammaire pour L_4 est la suivante :

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

S→ aaScccc / A */* pour 2 **a** à gauche, ça correspond 4 **c** à droite*/*
A→ bbAcc / bbcccc */* pour 2 **b** à gauche, ça correspond 2 **c** à droite. La sortie par **bbcccc** car au moins n=1 et donc p=2*/*

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 $\alpha \in (T \cup N)^*$

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

Exemples : L_5 = {a, acc, aabcc, bcc, aabbcccc, …}
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
  $L_{61}=\{a^m\ b^n\ c^p / m > n\}$
- et le deuxième respectant la deuxième condition.
  $L_{62}=\{a^m\ b^n\ c^p / 2n \leq p\}$

On va étudier les deux langages séparément :

1/ Langage L_{61} : $m > n \Leftrightarrow m= n+k$ avec $k >0$**.**
Dans ce cas : $a^m\ b^n\ c^p = a^{n+k}\ b^n\ c^p$
$= a^k\ a^n\ b^n\ c^p$ avec $n,p \geq 0$ et $k >0$ (k, n, p sont indépendants)
Donc, on peut utiliser trois non-terminaux :
- A1 pour générer les a^k
- B1 pour générer les a^nb^n
- C1 pour générer les c^p

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 :

S_1→ $A_1B_1C_1$ */* les trois parties du mot */*

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

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

C_1→ cC_1 / ε */* la troisième partie : une suite aléatoire de c. La sortie par ε */*

Théorie des Langages
Corrigé Devoir N°2

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 $\alpha \in (T \cup N)^*$

2) Langage L_{62} : $p \geq 2n \Leftrightarrow p = 2n+j$ avec $j \geq 0$.

Dans ce cas : $a^m b^n c^p = a^m b^n c^{2n+j}$

$= a^m b^n c^{2n} c^j$ avec m, n, j ≥ 0 (m, n, j sont indépendants).

Donc, on peut utiliser trois non-terminaux A2, B2, C2 pour générer respectivement les a^m, les $b^n c^{2n}$ et les c^j.

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 :

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

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

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

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

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 $\alpha \in (T \cup N)^*$

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 $S \rightarrow S_1 / S_2$.

Exercice 4

$L_8 = \{w \in \{a,b,c\}^* / |w|_c = 3p+1, \quad p \geq 0\}$

Exemples : L_8 = { c, cb, acab, ccc, bacaacabbcbcaa, cccbcbaaaa, … }
Les mots de L_8 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**.

USTHB Année 2019/2020
Faculté d'Electronique & Informatique
Département d'Informatique
L2-ACAD (Sections A & C)

Théorie des Langages

Corrigé Devoir N°2

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)

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.