NOLLE

Damien

L3 – Informatique

TL – Devoir 1 :

Note :	***Observation :***
/20	

Exercice 1)

Calcul de N_ε :

$N_\varepsilon^0 = \{A \mid A \in N \wedge \exists(A \rightarrow \varepsilon) \in R\}$

Donc :

$N_\varepsilon^0 = \{B\}$

Car $\exists(B \rightarrow \varepsilon) \in R$.

$N_\varepsilon^1 = N_\varepsilon^0 \cup \{A \mid A \in N \wedge \exists(A \rightarrow \alpha) \in R \wedge \alpha \in (N_\varepsilon^0)^*\}$

Donc :

$N_\varepsilon^1 = \{B, A\}$

Car $\exists(A \rightarrow BB) \in R$ et $BB \in (N_\varepsilon^0)^*$.

$N_\varepsilon^2 = N_\varepsilon^1 \cup \{A \mid A \in N \wedge \exists(A \rightarrow \alpha) \in R \wedge \alpha \in (N_\varepsilon^1)^*\}$

Sauf qu'il n'existe dans R, aucune règle de production de la forme $A \rightarrow \alpha$ et $\alpha \in (N_\varepsilon^1)^*$.

Donc : $\{A \mid A \in N \wedge \exists(A \rightarrow \alpha) \in R \wedge \alpha \in (N_\varepsilon^1)^*\} = \emptyset$ et $N_\varepsilon^1 \cup \emptyset = N_\varepsilon^1$.

On dit alors que l'ensemble N_ε se stabilise, et donc :

$N_\varepsilon = \{B, A\}$.

On définit G', la grammaire équivalent à G sans production vide. L'élimination des productions vides, entraine l'ajout d'un nouveau non-terminal et d'un nouvel axiome : S' et d'une nouvelle règle de production : S' → S.

Si S (l'axiome de G) ∈ N_ε, alors une autre règle de production est ajouté : S' → ε. Or, S ∉ N_ε, on n'ajoute pas de production vide pour S' et on se retrouve uniquement avec : S' → S.

A partir d'ici, il y a deux solutions : Soit on conserve S' ou non, car S' → S est une règle de production qui sert uniquement à passer d'un non-terminal à un autre sans faire évoluer le mot. Qu'on le conserve ou non, cela ne changera pas le langage engendré par la grammaire, ni l'avancement de l'élimination des productions vides.

Si nous avions eu : S' → S | ε, on aurait conservé obligatoirement S' et ses règles de productions afin de conservé le langage de la grammaire de départ.

On ne va donc pas conserver S' et on se retrouve donc avec : G' = ({S,A,B,C}, {x,y}, S, R')

et R' = { S → ABC | AC | BC | C,

A → BB | B | x,

B → CA | C,

C → AC | C | yB | y }

Exercice 2)

Q1 : L est un langage défini sur V = {a,b} contenant tous les mots α qui contiennent au moins une fois au moins deux lettres b consécutives.

Ce qui signifie que le mot le plus court, accepté dans L, est : bb.

Donc : RegEx = bb.

Les mots peuvent commencer par ε, a, b, aa, ab, ba, bb, etc…, donc :

RegEx = (a + b)*.RegEx = (a + b)*.bb.

Ils peuvent aussi se terminer par ε, a, b, aa, ab, ba, bb, etc…, donc :

Regex = RegEx.(a + b)* = (a + b)*.bb.(a + b)*.

L = L(RegEx) et RegEx ∈ ER.

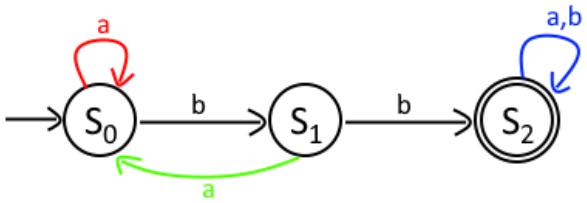
Q2 : D'après la description du langage L, on sait que le mot le plus cours est : bb, on réalise donc la première partie de l'automate correspondant à ce mot le plus court. On définit un état initial S_0, un état S_1 et un état final S_2 avec les transitions suivantes : $S_0.b = S_1$ et $S_1.b = S_2$.

A partir de cet automate, on détermine les états et les transitions manquantes permettant de concevoir les mots les plus longs :

- A partir de l'état S_0, si on lit le caractère a au début du mot, on reste dans un état non final. On définit alors une nouvelle transition $S_0.a = S_0$ permettant ainsi de rester dans l'état initial, qui n'est pas final. Si on avait définit la transition $S_0.a = S_1$, S_1 étant aussi un état non final, que le caractère précèdent lu du mot est un a et qu'à partir de S_1 on lit le caractère b, on arriverait alors dans l'état S_3 qui est l'état final de l'automate. Le mot étant alors accepté dans le langage engendré par l'automate, il ne contiendrait pas au moins une fois au moins deux b consécutifs et donc ne correspondrait pas au langage décrits dans le sujet. Cette nouvelle transition, permet de prendre en compte tous les mots commençant par un ou plusieurs a.

- A partir de l'état S_0, si on lit un ou plusieurs caractères a au début du mot, qu'on lit ensuite un caractère b (on passe donc de l'état S_0 à S_1) et qu'on relit aussitôt un caractère a, on doit revenir sur l'état initial (S_0) car les caractères b ne seront pas consécutifs, brisant ainsi la définition du langage. On définit alors la transition $S_1.a = S_0$.

- Une fois arrivé à l'état final (S_3), peu importe ce qu'on lira comme caractère (que ça soit a ou b), le mot correspondra toujours à la définition du langage puisqu'il contiendra au moins une fois au moins deux b consécutifs. On restera alors toujours dans l'état final. On définit donc les transitions $S_3.a = S_3$ et $S_3.b = S_3$.



On a donc défini l'automate A = ({S_0,S_1,S_2},{a,b},S_0,.,{S_2}), avec les transitions : $S_0.b = S_1$, $S_1.b = S_2$, $S_0.a = S_0$, $S_1.a = S_0$, $S_3.a = S_3$ et $S_3.b = S_3$. Pour lequel, L(A) = L(RegEx) = L.

Q3 : Un automate d'états fini déterministe (AFD) est un automate ayant un seul état initial et qui, pour chaque état, il n'existe qu'une seule transition sortante par étiquette. On remarque que cette définition correspond déjà à l'automate A précédemment définit. Donc A est un AFD.

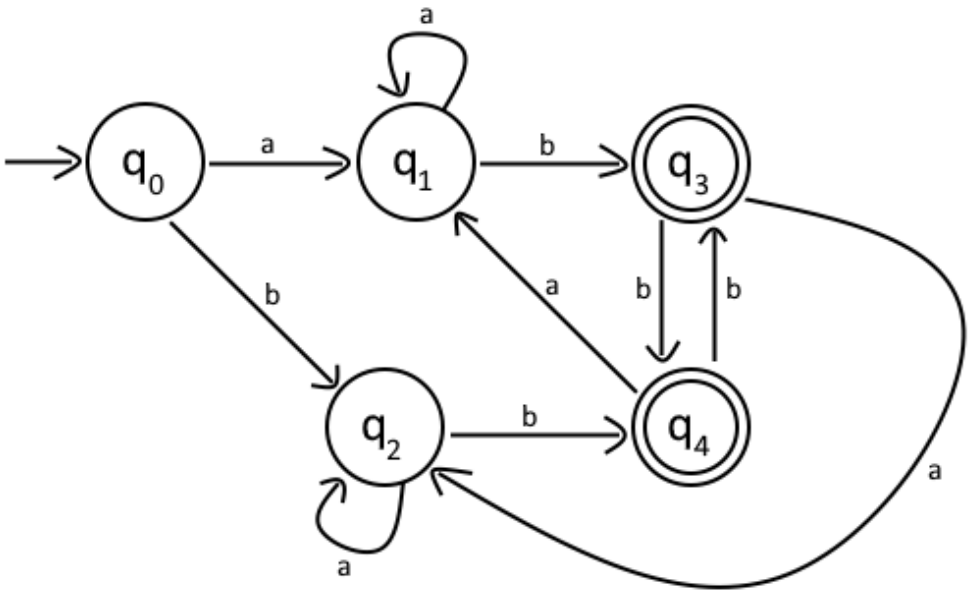
Si A était un automate d'états fini non déterministe (AFN), il aurait fallu utilisé la détermination pour définir un automate A' qui serait un AFD équivalent à l'automate A.

Q4 : Un automate est complétement spécifié si, pour chaque état de l'automate, il existe au moins une transition sortante pour toutes les étiquettes. En clair, quel que soit l'état de départ (s), il existe au moins un état cible (s') pour lequel : $\forall a \in V$ (ensemble du vocabulaire compris par l'automate, soit les étiquettes), $\exists(s.a \rightarrow s')$.

On remarque que l'automate A correspond à cette définition. En effet, il existe pour chaque état une transition sortante pour toutes les étiquettes.

Exercice 3)

Q1 :



Q2 : Pour donner un automate minimal A_{min} équivalent à l'automate A, il faut déjà que l'automate A soit déterministe avance d'utiliser la minimisation.

Pour rappel : Un automate d'états fini déterministe (AFD) est un automate ayant un seul état initial et qui, pour chaque état, il n'existe qu'une seule transition sortante par étiquette.

L'automate A dispose d'un seul état initial (q_0) et pour chaque état, on a qu'une seule transition sortante par étiquettes. De plus, chaque état dispose d'une transition sortante pour toutes les étiquettes. A est donc un automate complétement spécifié.

Minimisation de l'automate A :

- Réalisation du tableau des transitions :

	q_0	q_1	q_2	q_3	q_4
a	$q_0.a = \mathbf{q_1}$	$q_1.a = \mathbf{q_1}$	$q_2.a = \mathbf{q_2}$	$q_3.a = \mathbf{q2}$	$q_4.a = \mathbf{q_1}$
B	$q_0.b = \mathbf{q_2}$	$q_1.b = \mathbf{q_3}$	$q_2.b = \mathbf{q_4}$	$q_3.b = \mathbf{q4}$	$q_4.b = \mathbf{q_3}$

- Bilan epsilon :

Pour rappel :

Supposons s, s'et s'', trois états de l'automate A.

- s.ε = s, s'il n'existe aucune transition sortante d'un état étiqueté ε dans A.
- s.ε = s', s'il existe une transition (s,ε,s') dans A.
- s.ε = s', s'il existe une transition (s,ε,s'') et une autre (s'',ε,s') dans A.

Pour le bilan epsilon de la minimisation :

En supposant s et s', deux état de A quelconques :

- Si s.ε = s' et s' ∈ S' (l'ensemble des états finaux de A), alors s' appartient à une classe attribué arbitrairement.
- Si s.ε = s' et s' ∉ S', alors s' appartient à une autre classe attribué arbitrairement distinctes de la classe précédente.

En clair, il s'agit de déterminer les états finaux et les séparer des autres états.

Dans l'automate A, il n'y a aucune transition étiqueté ε, donc s.ε = s. En supposant que la classe I correspond aux autres états (non finaux) et la classe II, les états finaux. On a donc :

	q_0	q_1	q_2	q_3	q_4
Bilan ε	I	I	I	II	II

- Premier bilan :

On reprend le tableau des transitions de l'automate A et on remplace les états résultant des transitions par sa classe équivalent au bilan epsilon. Après cela, on vient redéfinir des classes arbitrairement et réunir dans ces classes toutes les colonnes identiques.

	q_0	q_1	q_2	q_3	q_4
Bilan ε	I	I	I	II	II
a	I	I	I	I	I
b	I	II	II	II	II
1er bilan	I	II	II	III	III

- Deuxième bilan :

On reprend les étapes précédentes mais en prenant comme référent, le bilan précédent avec les nouvelles classes définies.

	q_0	q_1	q_2	q_3	q_4
Bilan ε	I	I	I	II	II
a	I	I	I	I	I
b	I	II	II	II	II
1er bilan	I	II	II	III	III
a	II	II	II	II	II
B	II	III	III	III	III
2ème bilan	I	II	II	III	III

On constate que le 1er bilan est identique au deuxième bilan. On arrête donc la minimisation ici. Il nous manque plus qu'à construire l'automate A_{min} :

Classe I = $\{q_0\}$

Classe II = $\{q_1,q_2\}$

Classe III = $\{q_3,q_4\}$

L'état initial de ce nouvel automate correspond à la classe qui contient l'état initial de l'automate de départ. q_0 étant l'état initial de A et $q_0 \in$ Classe I, alors la classe I sera l'état initial de l'automate A_{min}.

Le ou les états finaux de ce nouvel automate correspond à la ou les classes qui contiennent au moins un état final de l'automate de départ. Les états finaux de A sont q_3 et q_4 et q_3, $q_4 \in$ Classe III, alors la classe III sera l'état final de l'automate A_{min}.

Il nous reste plus qu'à déterminer les transitions :

- Classe I = $\{q_0\}$, on sait que :
  $q_0.a = q_1$ et $q_0.b = q_2$, on peut donc affirmer que I.a = I.b = $\{q_1,q_2\}$ = Classe II.

- Classe II = $\{q_1,q_2\}$, on sait que :
  $q_1.a = q_1$
  $q_1.b = q_3$
  $q_2.a = q_2$
  $q_2.b = q_4$

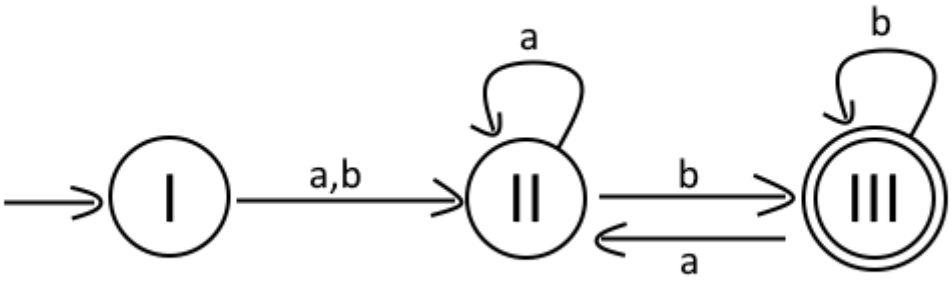
  On peut donc affirmer que II.a = $\{q_1, q_2\}$ = Classe II et II.b = $\{q_3,q_4\}$ = Classe III.

- Classe III = $\{q_3,q_4\}$, on sait que :
  $q_3.a = q_2$
  $q_3.b = q_4$
  $q_4.a = q_1$

$q_4.b = q_3$

On peut donc affirmer que III.a = $\{q_2,q_1\}$ = $\{q_1,q_2\}$ (Puisque dans un ensemble au sens mathématique, il n'y a pas d'ordre ni de position) = Classe II et III.b = $\{q_4,q_3\}$ = $\{q_3,q_4\}$ = Classe III.

On a donc : A_{min} = ({I,II,III},{a,b},I,.,{III}) avec les transitions : I.a = II, I.b = II, II.a = II, II.b = III, III.a = II et III.b = III.



En sachant que L(A) = L(A_{min}) donc A_{min} est un automate équivalent à A.

Q3 :

Pour déterminer une grammaire G, tel que L(G) = L(A_{min}), il suffit de lire l'automate et :

En supposant s, s' et s'', trois états quelconques de l'automates A ainsi que a et b, deux étiquettes quelconques du vocabulaires compris par l'automate :

- Si s.a = s' et s' ∉ S', alors la règles de production sera de la forme : s → as' où s et s' deviennent des non terminaux et a un terminal.
- Si s.a = s' et s' ∈ S' alors s → a.
- Si s.a = s', s' ∈ S' et s'.b = s'', alors s → as' | a

Donc :

I.a = II et II n'étant pas un état final, alors : I → aII

I.b = II, alors : I → bII

II.a = II, alors : II → aII

II.b = III et III étant un état final (en sachant que III.a = II et III.b = III) donc : II → bIII | b

III.a = II, alors : III → aII

III.b = III, alors : III → bIII | b

I étant l'état initial, il sera alors l'axiome dans la grammaire G, donc :

G = ({I,II,III},{a,b},I,R) avec :

R = { I → aII | bII,

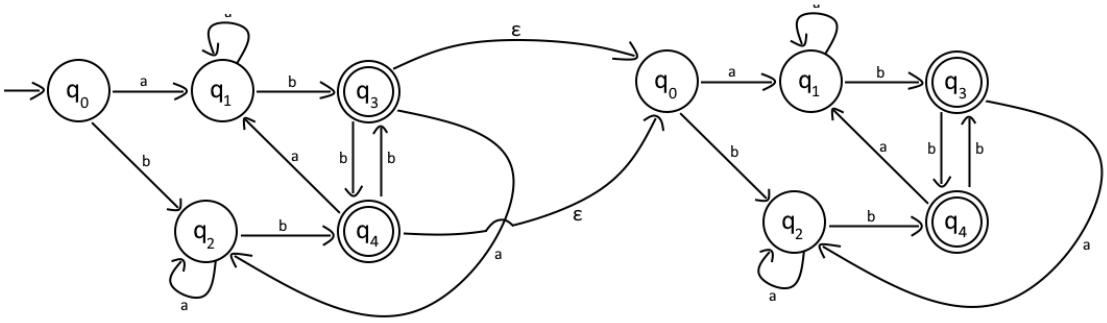
II → aII | bIII | b,

III → aII | bIII | b }

Avec L(G) = L(A_{min}).

Q4 :

Pour représenter le langage L(A).L(A), il suffit de recopier deux fois l'automate A et de joindre les états finaux du premier automate à l'état initial du deuxième automate avec des transition epsilon. L'état initial du second automate disparait.



Exercice 4)

Q1 :

Une grammaire sous forme normal de Greibach (FNG) est une grammaire ayant des règles de production de la forme : A → a.α avec a ∈ T et α ∈ N*.

Pour transformer une grammaire en sa forme normal de Greibach, il faut d'abord s'assurer que la grammaire de départ soit une grammaire algébrique propre. C'est-à-dire, une grammaire ayant des règles de production de la forme A → α avec α ∈ (N U T)* et n'ayant aucune production vide ou règle de la forme A → B où A, B ∈ N, ne faisant pas évoluer le mot dans sa dérivation puisque l'on passe uniquement d'un non terminal à un autre.

La transformation en FNG se déroule de la manière suivante :

- Elimination des récursivités gauches (immédiates et non immédiates).
- Les règles de la forme S_i → S_j.α avec α ∈ (N U T)* et j ≠ i, on vient remplacer S_j par les déterminants des règles de productions ayant S_j en membre gauche.
- On remplace les terminaux présents dans α par des nouveaux non terminaux qui dériveront en ses terminaux.

On constate que la grammaire G est algébrique, car ses règles de production sont de la forme : $A \rightarrow \alpha$ avec $\alpha \in (N \cup T)^*$. En revanche elle n'est pas propre, car il y a la production vide $B \rightarrow \varepsilon$, mais il n'y a aucune règle de la forme $A \rightarrow B$. Pour rendre G propre, il suffit d'éliminer les productions vides.

- Elimination des productions vides de G :

  Calcul de N_ε :

  $N_\varepsilon^0 = \{A \mid A \in N \wedge \exists(A \rightarrow \varepsilon) \in R\}$

  Donc :
  $N_\varepsilon^0 = \{B\}$
  Car $\exists(B \rightarrow \varepsilon) \in R$.

  $N_\varepsilon^1 = N_\varepsilon^0 \cup \{A \mid A \in N \wedge \exists(A \rightarrow \alpha) \in R \wedge \alpha \in (N_\varepsilon^0)^*\}$

  Sauf qu'il n'existe dans R, aucune règle de production de la forme $A \rightarrow \alpha$ et $\alpha \in (N_\varepsilon^0)^*$.

  Donc : $\{A \mid A \in N \wedge \exists(A \rightarrow \alpha) \in R \wedge \alpha \in (N_\varepsilon^0)^*\} = \emptyset$ et $N_\varepsilon^0 \cup \emptyset = N_\varepsilon^0$.

  On dit alors que l'ensemble N_ε se stabilise, et donc :

  $N_\varepsilon = \{B\}$.

  On a donc :

  G' = ({S,A,B,C},{x,y},S,R') avec :

  R' = {S → ABC | AC | BA | A,

  A → AyC | xBC | xC,

  B → CC,

  C → Cx | yB | y }

  Mais G' n'est toujours pas propre car il existe : S → A.

En réalité, il n'est pas obligé de rendre la grammaire propre avant la transformation en FNG. Car l'élimination des productions vides est obligatoire pour l'élimination des récursivités gauches et sera donc faite quoi qu'il arrive (ici, l'élimination des productions vides a été faite en amont pour gagner du temps).

Les règles de la forme : $A \rightarrow B$ peuvent disparaitre au moment de la transformation.

Néanmoins, ne pas rendre la grammaire propre avant peut poser un risque de se retrouver avec une grammaire qui ne sera pas à l'arrivée en FNG, voir qui ne sera pas du tout équivalente à la grammaire de départ. Je décide ici de prendre ce risque.

- Elimination des récursivités gauches immédiates :

R' = {S → ABC | AC | BA | A,
A → AyC | xBC | xC,
B → CC,
C → Cx | yB | y }

Il y a deux récursivité gauches immédiates : A → AyC et C → Cx.

On créer donc deux nouveau terminaux A' et C' avec les règles de productions suivantes :
C → Cx | yB | y
C' → Cx
A → AyC | xBC | xC
A' → AyC

On supprime les règles de productions à l'origine de la récursivité dans C et dans A :
C → yB | y
C' → Cx
A → xBC | xC
A' → AyC

On supprime le non terminal à l'origine de la récursivité dans C' et A' :
C → yB | y
C' → x
A → xBC | xC
A' → yC

On met dans tous les déterminants, le non terminal C' dans C et A' dans A ainsi que C' dans C' et A' dans A' le plus à droite possible :
C → yBC' | yC'
C' → xC'
A → xBCA' | xCA'
A' → yCA'

Et enfin, on rajoute une production vide dans C' et A' :
C → yBC' | yC'
C' → xC' | ε
A → xBCA' | xCA'
A' → yCA' | ε

On se retrouve donc avec :
R'' = {S → ABC | AC | BA | A,
A → xBCA' | xCA',
A' → yCA' | ε,
B → CC,
C → yBC' | yC',
C' → xC' | ε }

- Elimination des récursivités gauches non-immédiates :
  Il n'y a aucune récursivités gauches non-immédiates dans R'', ce dernier reste donc incahangé.

- Elimination des règles de production de la forme : $S_i \rightarrow S_j.\alpha$ avec $\alpha \in (N \cup T)^*$ et $j \neq i$:
  R''' = {S → xBCA'BC | xCA'BC | xBCA'C | xCA'C | yBC'CA | yC'CA | xBCA' | xCA',
  A → xBCA' | xCA',
  A' → yCA' | ε,
  B → yBC'C | yC'C,
  C → yBC' | yC',
  C' → xC' | ε }

- On remplace les terminaux présents dans α par des nouveaux non terminaux qui dériveront en ses terminaux :
  Il n'y a pas besoin de le réaliser car il n'y a plus aucun terminal dans α pour chaque de productions dans R'''. Toutes les règles sont de la forme : A → a.α avec a ∈ T et α ∈ N*.

  Il ne manque plus qu'à éliminer les productions vides pour que la nouvelle grammaire soit propre :

  Calcul de N_ε :

  $N_\varepsilon^0 = \{A \mid A \in N \wedge \exists(A \rightarrow \varepsilon) \in R'''\}$

  Donc :
  $N_\varepsilon^0 = \{A', C'\}$
  Car $\exists(C' \rightarrow \varepsilon) \in R'''$ et $\exists(A' \rightarrow \varepsilon) \in R'''$.

  $N_\varepsilon^1 = N_\varepsilon^0 \cup \{A \mid A \in N \wedge \exists(A \rightarrow \alpha) \in R''' \wedge \alpha \in (N_\varepsilon^0)^*\}$

  Sauf qu'il n'existe dans R''', aucune règle de production de la forme $A \rightarrow \alpha$ et $\alpha \in (N_\varepsilon^0)^*$.

  Donc : $\{A \mid A \in N \wedge \exists(A \rightarrow \alpha) \in R''' \wedge \alpha \in (N_\varepsilon^0)^*\} = \emptyset$ et $N_\varepsilon^0 \cup \emptyset = N_\varepsilon^0$.

  On dit alors que l'ensemble N_ε se stabilise, et donc :

  $N_\varepsilon = \{A', C'\}$

  Donc :

  R'''' = {S → xBCA'BC | xBCBC | xCA'BC | xCBC | xBCA'C | xBCC | xCA'C | xCC | yBC'CA | yBCA | yC'CA | yCA | xBCA' | xBC | xCA' | xC,
  A → xBCA' | xBC | xCA' | xC,
  A' → yCA' | yC,
  B → yBC'C | yC'C | yBC | yC,
  C → yBC' | yC' | yB | y,

C’ → xC’ | x }

On se retrouve donc avec G’ = ({S,A,A’,B,C,C’},{x,y},S,R’’’’) qui est en FNG et qui est l’équivalente de G, donc L(G) = L(G’).

Q2 : On sait qu’à partir d’une grammaire en FNG, on peut construire un automate à pile simple.

Soit A = (V,Γ,Z_0,P) un automate à pile simple.

V = T (l’ensemble des terminaux).

Γ = N (l’ensemble des non-terminaux).

Z_0 = X (l’axiome).

P (les règles de transitions).

A chaque règle de production Z → aβ ∈ R, on associe une transition (a, Z, t(β)) où t(β) est le mot miroir de β.

Donc :

A = ({x,y},{S,A,A’,B,C,C’},S,P) avec :

P = {

(x,S,CBA’CB),

(x,S,CBCB),

(x,S,CBA’C),

(x,S,CBC),

(x,S,CA’CB),

(x,S,CCB),

(x,S,CA’C),

(x,S,CC),

(y,S,ACC’B),

(y,S,ACB),

(y,S,ACC’),

(y,S,AC),

(x,S,A’CB),

(x,S,CB),

(x,S,A'C),

(x,S,C),

(x,A,A'CB),

(x,A,CB),

(x,A,A'C),

(x,A,C),

(y,A',A'C),

(y,A',C),

(y,B,CC'B),

(y,B,CC'),

(y,B,CB),

(y,B,C),

(y,C,C'B),

(y,C,C'),

(y,C,B),

(y,C,ε),

(y,C',C'),

(x,C',ε),

}

Exercice 5)

Q1 :

- Calculons premier(A) où A correspond à tous les non-terminaux de G :

premier(X) = premier(aYbX) U premier(bZaX) U premier(ε) = premier(a) U premier(b) U premier(ε) = {a,b,ε}

premier(Y) = premier(aYbY) U premier(ε) = premier(a) U premier(ε) = {a, ε}

premier(Z) = premier(bZaZ) U premier(ε) = premier(b) U premier(ε) = {b, ε}

- Calculons suivant(A) où A correspond à tous les non-terminaux de G :

Prenons X :

suivant(X) = {#} car X est l'axiome.

C’est le seul calcul que l’on peut faire car :

X → aYbX

Donc : suivant(X) = suivant(X) U suivant(X) = {#} U {#}.

Sauf que dans un ensemble au sens mathématique, il n’est pas possible d’avoir plusieurs fois un même élément, donc : {#} U {#} = {#} = suivant(X).

X → bZaX

Donc : suivant(X) = suivant(X) U suivant(X) = {#} U {#}.

Sauf que dans un ensemble au sens mathématique, il n’est pas possible d’avoir plusieurs fois un même élément, donc : {#} U {#} = {#} = suivant(X).

Prenons Y :

suivant(Y) = {}

X → aYbX

Donc : suivant(Y) = suivant(Y) U (Premier(bX) / {ε}) = Ø U (Premier(b) / {ε}) = Ø U ({b} / {ε}) = Ø U {b} = {b}.

C’est le seul calcul que l’on peut faire car :

Y → aYbY

Ici on pourrait avoir deux calculs :

suivant(Y) = suivant(Y) U (Premier(bX) / {ε}) = {b} U (Premier(b) / {ε}) = {b} U ({b} / {ε}) = {b} U {b}.

Sauf que dans un ensemble au sens mathématique, il n’est pas possible d’avoir plusieurs fois un même élément, donc : {b} U {b} = {b} = suivant(Y).

Et

Donc : suivant(Y) = suivant(Y) U suivant(Y) = {Y} U {Y} = {b} U {b}.

Sauf que dans un ensemble au sens mathématique, il n’est pas possible d’avoir plusieurs fois un même élément, donc : {b} U {b} = {b} = suivant(Y).

prenons Z :

suivant(Z) = {}

X → bZaX

Donc : suivant(Z) = suivant(Z) U (Premier(aX) / {ε}) = Ø U (Premier(a) / {ε}) = Ø U ({a} / {ε}) = Ø U {a} = {a}.

C’est le seul calcul que l’on peut faire car :

Z → bZaZ

Ici on pourrait avoir deux calculs :

Donc : suivant(Z) = suivant(Z) U (Premier(aZ) / {ε}) = {a} U (Premier(a) / {ε}) = {a} U ({a} / {ε}) = {a} U {a}.

Sauf que dans un ensemble au sens mathématique, il n’est pas possible d’avoir plusieurs fois un même élément, donc : {a} U {a} = {a} = suivant(Z).

Et

Donc : suivant(Z) = suivant(Z) U suivant(Z) = {Z} U {Z} = {a} U {a}.

Sauf que dans un ensemble au sens mathématique, il n’est pas possible d’avoir plusieurs fois un même élément, donc : {a} U {a} = {a} = suivant(a).

- Calcul de la table d’analyse M :

	a	**b**	**#**
X	X → aYbX	X → bZaX	X → ε
Y	Y → aYbY	Y → ε	
Z	Z → ε	Z → bZaZ	

Q2 :

Une grammaire est LL(1) si la table d’analyse est déterministe (c’est-à-dire qu’elle contient au plus une règles de production par case). Si la grammaire est LL(1) alors elle est non ambiguë et analysable par la procédure de l’analyse descendante LL(1) en lisant un caractère à l’avance.

On remarque que dans la table M calculé à la question 1, chaque case ne contient qu’une seule règle de production lorsqu’il y en a une.

La table d’analyse M est donc bien déterministe et la grammaire G est donc en LL(1). Etant LL(1), G est donc analysable par la procédure de l’analyse descendante LL(1) en lisant un caractère à l’avance.

Q3 :

Pile	**Mot**	**Règle | Action**
#X	aabb#	X → aYbX
#XbYa	aabb#	lecture
#XbY	abb#	Y → aYbY
#XbYbYa	abb#	lecture
#XbYbY	bb#	Y → ε
#XbYb	bb#	lecture
#XbY	b#	Y → ε
#Xb	b#	lecture
#X	#	X → ε
#	#	Succès

Q4 :