"glob" :

- le "*" pour dire toutes les suites de caractères
- le "?" pour dire un unique caractère
- le "" pour dire a ou b ou c

Expressions régulières POSIX : Il en existe deux types : basiques et étendues.

- le "." pour dire n'importe quel caractère sauf le saut de ligne \ n
- "car" pour le caractère car
- abc pour un a ou b ou c
- pour tout caractère sauf a, b, c
- "... $\|$ ..."
- "m*", "m+", "m{k,l}
- chapeau ou $
- ...
- Ce n'est pas du tout minimaliste

Expressions régulières à la Kleene : Les expressions régulières sont données par la grammaire suivante :

$ER ::= 0|1|a|ER{+}ER|ER.ER|ER^{*} (0 = \varnothing, 1 = \{""\}, a = \text{symbole}$

Les symboles appartiennent) un ensemble fini appelé alphabet généralement noté Σ.

Exemple : Avec $\Sigma = \{a, b\}$:

- 0
- 1
- a
- b
- $((a + 1)^{*}.b^{*})^{*} + a.b$

Priorités : * puis . puis $+$

L'addition est associative et commutative, la concaténation est associative. Le "." est souvent omis.

Rq : $r^{*} = 1 + rr^{*}$

Rq : Deux expressions régulières sont égales si elles correspondent aux mêmes chaines de caractères.

Langage associé : On associe à chaque regex r un ensemble de mots sur Σ appelé $L(r)$

Exemples

- $L(0) = \varnothing$
- $L(1) = \{\varepsilon\}$

- $L(a) = \{a\}$
- $L(r_1 + r_2) = L(r_1) \bigcup L(r_2)$
- $L(r_1 r_2) = \{mn, m \in L(r_1), n \in L(r_2)\} = L(r_1).L(r_2)$
- $L(r^*) = \{uu \dots u, u \in L(r)\}$
- $L(0^*) = \{\varepsilon\}$
- $L(1^*) = \{\varepsilon\}$

Définition : Un langage l est régulier si il existe une expression régulière r tel que $l = L(r)$ **Regex équivalentes** : Deux expressions sont équivalentes quand elles ont le même langage associé.
Question : Peut on vérifier si 2 expressions sont égales ?
Question : Existe-t-il des langages non réguliers ? OUI
Question : Le complémentaire d'un ensemble régulier est-il régulier ?
Question : Peut on tester rapidement si $u \in L(r)$?
Question : Quand a-t-on $\varepsilon \in L(E)$? réponse : on fait une fonction récursive à partir des cas de base :

- $\varepsilon \notin L(0)$
- $\varepsilon \in L(1)$
- $\varepsilon \notin L(s)$
- $\varepsilon \in L(E_1 + E_2) \Leftrightarrow \varepsilon \in L(E_1) \vee L(E_2)$

- $\varepsilon \in L(E_1E_2) \Leftrightarrow \varepsilon \in L(E_1) \wedge L(E_2)$
- $\varepsilon \in L(E^*)$

```
type regexp =
  | Zero
  | One
  | Symb of char
  | Sum of regexp * regexp
  | Product of regexp * regexp
  | Star of regexp

let rec contains_epsilon (r:regexp) :bool = match r with
  | One -> true
  | Zero -> false
  | Symb(_) -> false
  | Sum(r1,r2) -> contains_epsilon(r1) || contains_epsilon(r2)
  | Product(r1,r2) -> contains_epsilon(r1) && contains_epsilon(r2)
```

Test d'appartenance d'un mot quelconque à $L(R)$:

On définit R/a la dérivée de R le long de a (première lettre du mot) dont le langage vérifie :

$$w \in L(R/a) \equiv aw \in L(R)$$

.

Pour tester si $w \in L(R)$:

- Si $w = \varepsilon$ on utilise la méthode vue précédemment
- Si $w = aw'$ on vérifie récursivement si $w' \in L(R/a)$

Regles de calcul de R/a

- $L(0/a) = L(0)$
- $L(1/a) = L(0)$
- $L(b/a) = \begin{cases} 0 & \text{si } b \neq a \\ 1 & \text{si } b = a \end{cases}$

- $L((R_1 + R_2)/a) = L(R_1/a + R_2/a)$
- $L((R_1 R_2)/a) = \begin{cases} L((R_1/a)R_2 + R_2/a) & \text{si } \varepsilon \in L(R_1) \\ L((R_1/a)R_2) & \text{sinon} \end{cases}$
- $L(R^*/a) = L((R/a)R^*)$

A partir de ça on peut programmer l'appartenance d'un mot quelconque au langage d'une expression régulière quelconque. On dit qu'il y a une procédure effective pour le problème "appartenance d'un mot à un langage d'une regex".

Apparté : Montrons que le problème de l'arrêt d'un programme p avec un argument a n'a pas de solution effective. : On le démontre par l'absurde
Supposons qu'un programme résoud ce problème : termine(p,a) renvoie true si p(a) termine, false sinon.
On définit un programme oups(x) qui fait la chose suivante :
if termine(x,x) (boucle infinie)
else return true
Quelle est la valeur de oups(oups) ??? Il se termine si il ne se termine pas, sinon il se termine.

Conséquence : Il n'y a pas de regex dont le langage est exactement l'ensemble des programmes qui terminent.

Exercice 1

1.
   - $.*\backslash.jpg\$$
   - $.*\backslash.jpe?g\$$
   - $.*\backslash.(j|J)(p|P)(e|E)?(g|G)\$$
   - $.*\backslash.((j|J)(p|P)(e|E)?(g|G))|((g|G)(i|I)(f|F))|((p|P)(n|N)(g|G$

2.

$$projet - bak - [0-9] + \backslash.tgz$$

3.

$$((\backslash\backslash")|[\,\hat{}\,"])*$$

4.

$$(+|-)?(([0-9]+\backslash.[0-9]+)|([0-9]+\backslash.?)|(\backslash.[0-9]+))((e|E)(+|-)?[0-$$

Exercice 2

1. (a) $(a.a.a)^*$
   (b) $a.a.a.(a.a.a)^*$
   (c) $((a.a.a)^*) + ((a.a.a.a.a)^*)$
   (d) $(a.a.a.((a.a) + 1))^*$

2.

$$c^*((a.b)^*.(1 + a) + (b.a)^*(1 + b))$$

3.
$$((a+1).b^*)^*.(a+1)$$

4.
$$((a+1).(((b+1).c^*)^*.(b+1))^*.(a+1)$$

5. (a) Un mot de $L(R^*)$ est une concaténation de mots de $L(R)$, un mot de $L((R^*)^*)$ est une concaténation de concaténations de mots de $L(R)$, donc c'est une concaténation de mots de $L(R)$.

   (b) Le mot vide est clairement dans les deux langages. Un mot non vide de $L((R_1R_2)^*)$ est de la forme $r_{1,1}r_{2,1}\ldots r_{1,n}r_{2,n} = r_{1,1}(r_{2,1}\ldots r_{1,n})r_{2,n}$ donc c'est bien la concaténation d'un mot de $L(R_1)$, puis un mot de $L((R_1R_2)^*)$, puis un mot de $L(R_2)$, c'est à dire un mot de $L(R_1(R_1R_2)^*R_2)$.

   (c) Les mots de $L((R_1+R_2)^*)$ sont des concaténations de mots qui sont soit dans $L(R_1)$ soit dans $L(R_2)$. Considérons un tel mot u, on enlève les derniers mots qui sont dans $L(R_1)$ (éventuellement aucun), pour qu'il termine par un mot de $L(R_2)$. On a $u = v.w$ avec $w \in L(R_1^*)$. et v une concaténation de mots dans $L(R_1)$ et dans $L(R_2)$, le dernier de ces mots étant dans $L(R_2)$. On découpe cette concaténation dès qu'on rencontre un mot de $L(R_2)$, ce qui donne des concaténations dans $L(R_1^*R_2)$. La réciproque est triviale.

   (d) Le mot vide est dans les deux langages. Les mots de la deuxième expression sont clairement dans $L(R^*)$. Vérifions que tout mot non vide de $L(R^*)$ est dans

$L((1+R)(RR)^*)$: S'il contient un mot de $L(R)$, c'est un mot de $L(1+R)$ suivi d'aucun mot de $L(RR)$. S'il contient deux mots c'est aucun mot de $L(1+R)$ suivi d'un mot de RR. Pour un nombre pair $n = 2k$ de mots, c'est aucun mot de $L(1+R)$ concaténé à k mots de $L(RR)$, si c'est un nombre impair $n = 2k+1$, c'est un mot de $L(1+R)$ concaténé à k mots de $L(RR)$.

6. Quand a-t-on $L(E) = \varnothing$? réponse : on fait une fonction récursive à partir des cas de base :

   - $L(0) = \varnothing$
   - $L(1) \neq \varnothing$
   - $L(s) \neq \varnothing$
   - $L(E_1 + E_2) = \varnothing \Leftrightarrow L(E_1) = \varnothing \wedge L(E_2) = \varnothing$
   - $L(E_1E_2) = \varnothing \Leftrightarrow L(E_1) = \varnothing \vee L(E_2) = \varnothing$
   - $L(E^*) \neq \varnothing$

Exo1, question1 :

1. ((a+b)(a+b))*
2. b*(ab*ab*)*
3. b*(ab*ab*)* + a*(ba*ba*)*
4. (impAimpBimpAimpB ou impApaireBimpA ou paireApaireB)
5. pairеA = (aa)*, paireB = (bb)*, impA = a

Exo2, question1 :

1. (ab)*
2. (a*b*)* = (a+b)*
3. (abc)* + (d(abc)*)* = (abc + d)*

Exo3, question1 :

1. (a*+ab)(aab+bb*a+bb)* OUI $a \in L(R)$
2. (b*a+b)(aab+bb*a+bb)* NON $b \notin L(R)$
3. (b*a+b)(aab+bb*a+bb)* NON $ab \notin L(R)$

Exo3, question2 :

1. R/a
   (a) aa(aaa)*

(b) aa(aaa)* + aaaa(aaaaa)*

(c) (aa + aaaa)(aaa + aaaaa)*

2. R/aa

   (a) a(aaa)*

   (b) a(aaa)* + aaa(aaaaa)*

   (c) (a + aaa)(aaa + aaaaa)*

3. R/aaa

   (a) (aaa)*

   (b) (aaa)* + aa(aaaaa)*

   (c) (1 + aa)(aaa + aaaaa)*

4. R/a^k

   (a) $\begin{cases} \text{(aaa)}* & \text{Si } k = 0[3] \quad \text{OUI} \\ \text{aa(aaa)}* & \text{Si } k = 1[3] \quad \text{NON} \\ \text{a(aaa)}* & \text{Si } k = 2[3] \quad \text{NON} \end{cases}$

$$
\text{(b)} \begin{cases}
(\mathrm{aaa}) * +(\mathrm{aaaaa})* & \text{Si } k = 0[15] \quad \text{OUI} \\
\mathrm{aa}(\mathrm{aaa}) * +\mathrm{aaaa}(\mathrm{aaaaa})* & \text{Si } k = 1[15] \quad \text{NON} \\
\mathrm{a}(\mathrm{aaa}) * +\mathrm{aaa}(\mathrm{aaaaa})* & \text{Si } k = 2[15] \quad \text{NON} \\
(\mathrm{aaa}) * +\mathrm{aa}(\mathrm{aaaaa})* & \text{Si } k = 3[15] \quad \text{OUI} \\
\mathrm{aa}(\mathrm{aaa}) * +\mathrm{a}(\mathrm{aaaaa})* & \text{Si } k = 4[15] \quad \text{NON} \\
\mathrm{a}(\mathrm{aaa}) * +(\mathrm{aaaaa})* & \text{Si } k = 5[15] \quad \text{OUI} \\
(\mathrm{aaa}) * +\mathrm{aaaa}(\mathrm{aaaaa})* & \text{Si } k = 6[15] \quad \text{OUI} \\
\mathrm{aa}(\mathrm{aaa}) * +\mathrm{aaa}(\mathrm{aaaaa})* & \text{Si } k = 7[15] \quad \text{NON} \\
\mathrm{a}(\mathrm{aaa}) * +\mathrm{aa}(\mathrm{aaaaa})* & \text{Si } k = 8[15] \quad \text{NON} \\
(\mathrm{aaa}) * +\mathrm{a}(\mathrm{aaaaa})* & \text{Si } k = 9[15] \quad \text{OUI} \\
\mathrm{aa}(\mathrm{aaa}) * +(\mathrm{aaaaa})* & \text{Si } k = 10[15] \quad \text{OUI} \\
\mathrm{a}(\mathrm{aaa}) * +\mathrm{aaaa}(\mathrm{aaaaa})* & \text{Si } k = 11[15] \quad \text{NON} \\
(\mathrm{aaa}) * +\mathrm{aaa}(\mathrm{aaaaa})* & \text{Si } k = 12[15] \quad \text{OUI} \\
\mathrm{aa}(\mathrm{aaa}) * +\mathrm{aa}(\mathrm{aaaaa})* & \text{Si } k = 13[15] \quad \text{NON} \\
\mathrm{a}(\mathrm{aaa}) * +\mathrm{a}(\mathrm{aaaaa})* & \text{Si } k = 14[15] \quad \text{NON}
\end{cases}
$$

$$
\text{(c)} \begin{cases}
\mathrm{R}* & \text{Si } k = 0 \quad \text{OUI} \\
\mathrm{R}_1 = (\mathrm{aa} + \mathrm{aaa})\mathrm{R}* & \text{Si } k = 1 \quad \text{NON} \\
\mathrm{R}_2 = (\mathrm{a} + \mathrm{aa})\mathrm{R}* & \text{Si } k = 2 \quad \text{NON} \\
\mathrm{R}_3 = (1 + \mathrm{a})\mathrm{R}* & \text{Si } k = 3 \quad \text{OUI} \\
\mathrm{R}_4 = \mathrm{R} * +\mathrm{R}_1 & \text{Si } k = 4 \quad \text{OUI} \\
\mathrm{R}_5 = \mathrm{R}_1 + \mathrm{R}_2 & \text{Si } k = 5 \quad \text{NON} \\
\mathrm{R}_6 = \mathrm{R}_2 + \mathrm{R}_3 & \text{Si } k = 6 \quad \text{OUI} \\
\mathrm{R}_7 = \mathrm{R}_3 + \mathrm{R} * +\mathrm{R}_1 & \text{Si } k = 7 \quad \text{OUI} \\
\mathrm{R}_8 = \mathrm{R} * +\mathrm{R}_1 + \mathrm{R}_2 & \text{Si } k = 8 \quad \text{OUI} \\
\mathrm{R}_9 = \mathrm{R}_1 + \mathrm{R}_2 + \mathrm{R}_3 & \text{Si } k = 9 \quad \text{OUI} \\
\mathrm{R}_{10} = \mathrm{R} * +\mathrm{R}_1 + \mathrm{R}_2 + \mathrm{R}_3 & \text{Si } k \geq 10 \quad \text{OUI}
\end{cases}
$$

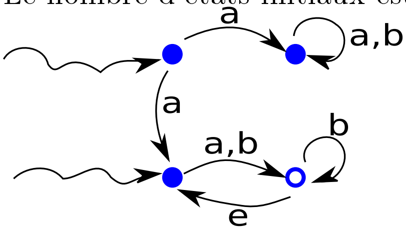
Exo3, Question 3

1. Les dérivées possibles sont (aaa)*+(bbb)*,aa(aaa)*,a(aaa)*,(aaa)*,0,

2. Les dérivées possibles sont R*,aaR*,aR*,0,bR*,bbR*,bbbR*,bbbbR*

Exo3, Question 5 On calcule le graphe et on le parcourt en suivant les lettres de l'expression demandée. On calcule la complexité sans tenir compte du temps de calcul du graphe : c'est le nombre de caractères du mot.

Définition : Il peut y avoir zéro ou plusieurs transitions différentes avec la même lettre à partir d'un seul état. On autorise aussi des flèches ε pour changer d'état sans consommer de lettre. Le nombre d'états initiaux est illimité.



Définition officielle : Un automate non déterministe sur l'alphabet Σ est donné par :

- Un ensemble fini d'états : S
- Une fonction de transition $f : S \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(S)$

Exemple :

δ	ε	a	b
$\to 1$	$\varnothing$	$\{2,3\}$	$\varnothing$
2	$\{3,4\}$	$\{2\}$	$\{2\}$
$\to (3)$	$\varnothing$	$\{4\}$	$\{4\}$
(4)	$\{3\}$	$\{1,2,3\}$	$\{4\}$

Définition : Un mot w est accepté par un automate non déterministe s'il existe au moins un chemin qui part d'un état initial et qui arrive à un état final en suivant des transitions correspondant aux lettres de w, ou bien des transitions ε

Union : Pour faire l'union de deux automates non déterministes il suffit de les mettre côte à côte.

Produit : Si A_1 et A_2 sont deux automates, on cherche un automate pour les mots qui commencent par un mot de A_1 et continuent par un mot de A_2. On conserve les états initiaux de A_1 et les états finaux de A_2. On ajoute des transitions ε entre les anciens états finaux de A_1 aux anciens états initiaux de A_2.

Etoile : On ajoute un état qui sera le seul état initial et final, on le relie par une transition ε vers tous les états initiaux et depuis tous les états finaux.

Langage régulier : Reconnu par un automate (déterministe ou non) ou donné par une expression régulière. Tous les langages ne sont pas réguliers!!!

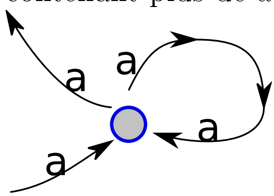
Exemples :

1. Sur l'alphabet $\Sigma = \{a\}$, $L = \{a^{n^2}, n \in \mathbb{N}\}$

2. Sur l'alphabet $\Sigma = \{a, b\}$, $L =$ ensemble des mots ayant autant de a que de b

   Preuve :

   Supposons qu'il existe un automate A déterministe qui reconnait L. Soit n le nombre d'états de A. Cet automate accepte $a^{n+1}b^{n+1}$. En lisant les $n+1a$, il y a forcément un moment où on repasse par un même état. Alors en cet état il y a deux sorties a, ce qui est impossible dans un automate déterministe. De plus, comme on ne sait pas le nombre de passage dans la boucle, il reconnaitrait forcément des mots contenant plus de a que de b.

3. Sur l'alphabet $\Sigma = \{a, b\}$, $L = \{a^nb^n, n \in \mathbb{N}\}$

Lemme de pompage : Si un langage est régulier alors il existe un nombre n avec la propriété suivante : Si w est un mot du langaga de taille supérieure à n, alors $w = u_1u_2u_3$ avec $u_1u_2^*u_3$ est dans L et la taille de u_1u_3 est inférieure ou égale à n.

Rq : On l'utilise en général pour montrer qu'un langage n'est pas régulier.

Théorème de Myhill-Nerode : Un langage est régulier ssi il a un nombre fini de dérivées.

Idée de preuve :

- nombre fini de dérivée $\Rightarrow$ automate des dérivées
- d'un automate déterministe donné il suffit de changer l'état initial pour avoir celui de la dérivée.

Exercice 1 :

1. $\exists n$ tq $\forall$ $w \in L$ tq taille$(w) \geq n$ tq $\exists$ x, y, z tq $w = xyz$ et taille$(xy) \leq n$ et $\forall$ k $xy^kz \in L$

2. (a)
   - Soit $n = 2k \in \mathbb{N}$
   - Je choisis $w = (ba^n)^2$
   - Pour un découpage xyz avec taille$(xy) \leq n$ on aura $y = (b+1)a^p$ avec $p < n$
   - En choisissant $k = 0$ on a $xy^kz = (b+1)a^{n-p}ba^n$. Si il n'y a pas le b il est clair que le mot ne peut pas contenir deux fois le même mot car il n'y a qu'un seul b dans le mot. Si le b y est, le premier mot commence par b donc le deuxième mot aussi, et du coup ils n'ont pas le même nombre de lettres ($n + 1 - p$ pour le premier, $n + 1$ pour le second)

   (b) C'est $L(aa(aa)^*)$

3. (a)
   - Soit $n \in \mathbb{N}$
   - Je choisis $w = a^nb^{n+1}$
   - Pour un découpage xyz avec taille$(xy) \leq n$ on aura $y = a^p$ avec $1 \leq p < n$
   - En choisissant $k = 3$ on a $xy^kz = a^{n-p}(a^p)^3b^{n+1}$.Il est alors clair qu'il y a plus de a que de b ($n - p + 3p = n + 2p \geq n + 2$. Il y a au moins $n + 2$ a et exactement $n + 1$ b)

(b)
- Soit $n \in \mathbb{N}$
- Je choisis $w = a^n b^{n-1}$
- Pour un découpage xyz avec taille$(xy) \leq n$ on aura $y = a^p$ avec $1 \leq p < n$
- En choisissant $k = 0$ on a $xy^k z = a^{n-p} b^{n-1}$.Il est alors clair qu'il y a moins de a que de b ($p \geq 1$ donc $n - p \leq n - 1$ donc $n - p \not> n - 1$)

4. (a)
- Soit $n \in \mathbb{N}$
- Je choisis $w = a^{n \times n + 2n + 1}$
- Pour un découpage xyz avec taille$(xy) \leq n$ on aura $y = a^p$ avec $1 \leq p < n$
- En choisissant $k = 0$ on a $xy^k z = a^{n \times n + 2n + 1 - p}$. Comme $1 \leq p < n$, $n^2 < n \times n + 2n + 1 - p < (n+1)^2$

(b)
- Soit $n \in \mathbb{N}$
- Je choisis $w = a^{2^{n+1}}$
- Pour un découpage xyz avec taille$(xy) \leq n$ on aura $y = a^p$ avec $1 \leq p < n < 2^n$
- En choisissant $k = 0$ on a $xy^k z = a^{2^{n+1} - p}$. Comme $1 \leq p < n$, $2^n < 2^{n+1} - p < 2^{n+1}$

(c)
- Soit $n \in \mathbb{N}$
- Je choisis $w = a^m$ avec m premier supérieur strict à $2n$
- Pour un découpage xyz avec taille$(xy) \leq n$ on aura $y = a^p$ avec $1 \leq p < m$
- En choisissant $k = m + 1$ on a $xy^k z = a^{m(p+1)}$, et $m(p + 1)$ n'est pas premier.

5.

Exercice 2

1.

2. Son complémentaire n'est pas régulier

3. Si le langage L des mots bien parenthésés était régulier, alors $L \bigcup L((^*)^*)$ serait régulier donc le langage des $(^n)^n$ serait régulier.

4. Il suffit de remplacer tous les s de l'expression régulière par w

5. Si L était régulier alors en remplaçant les symboles sauf les parenthèses par ε on aurait que le langage des mots bien parenthésés serait régulier.

Exercice 3

1.

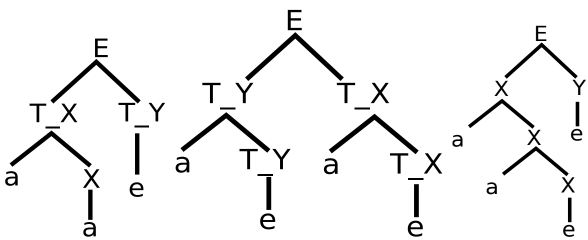
Exercice 1 :

1. (a)
   - $E ::= T_X T_Y$
   - $X ::= a$
   - $T_X ::= X|XT_X$
   - $Y ::= a|b|c$
   - $T_Y ::= \varepsilon|YT_Y$

   (b)
   - $E ::= T_Y T_x$
   - $T_Y ::= \varepsilon|aT_Y|bT_Y|cT_Y$
   - $T_X ::= a|aT_X$

   (c)
   - $E ::= XY$
   - $X ::= \varepsilon|aX|bX$
   - $Y ::= \varepsilon|bX|cX$

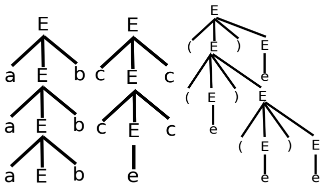
   (d)
   - $E ::= YaaZ$
   - $Z ::= \varepsilon|abZ|bcZ|caZ$
   - $Y ::= \varepsilon|abY|bCY$
   - $C ::= \varepsilon|cC$

   (e) 

2. (a) $E ::= \varepsilon|aEb$

(b) $E ::= \varepsilon | c | d | e | cEc | dEd | eEe$

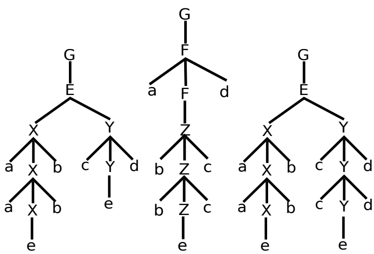
(c) $E ::= \varepsilon | (E)E$

(d) 

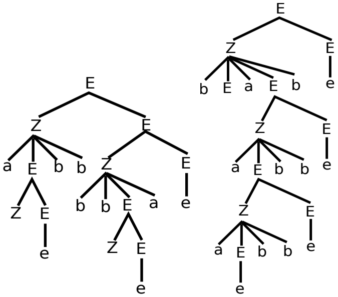
3. (a)
   - $E ::= XY$
   - $X ::= \varepsilon | aXb$
   - $Y ::= \varepsilon | cYd$

   (b)
   - $E ::= X | aEd$
   - $X ::= \varepsilon | bXc$

   (c)
   - $G ::= E | F$
   - $E ::= XY$
   - $X ::= \varepsilon | aXb$
   - $Y ::= \varepsilon | cYd$
   - $F ::= Z | aFd$
   - $Z ::= \varepsilon | bZc$

   (d) 

4. (a)
   - $E ::= \varepsilon | ZE$
   - $Z ::= aEbb | bEaEb | bbEa$

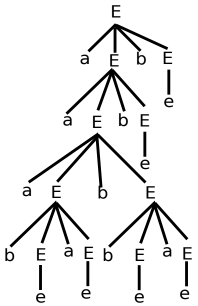


(b)

(c) C'est l'ensemble des mots qui ont deux fois plus de b que de a

5. (a) $E ::= \varepsilon | aEbE | bEaE$

   (b) $E = (1 + aEb + bEa)*$



(c)

Exercice 2

1. - $E ::= [L]$: un crochet, puis une liste, puis un crochet
   - $L ::= \varepsilon | PU$: rien ou (un nombre du langage, puis une liste éventuellement vide commençant par une virgule)
   - $U ::= \varepsilon | ,PU$: une partie de la liste ',' puis un nombre du langage puis une liste $\rightarrow$ liste commençant par une virgule
   - $P ::= 0Z | SN$: un nombre du langage (que des zéros ou un signe et un nombre ne commençant pas par 0)

- $Z ::= \varepsilon | 0Z$: des 0 les uns à côté des autres
- $N ::= 1M|2M|3M|4M|5M|6M|7M|8M|9M$: commence par autre chose que 0 puis des chiffres entre O et 9
- $M ::= \varepsilon | 0M | N$: des chiffres entre 0 et 9 qui se suivent
- $S ::= \varepsilon | + | -$: le signe