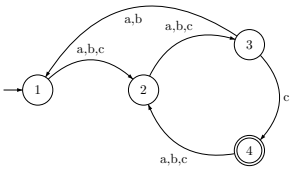
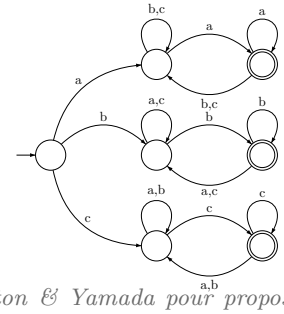
1. *Soit $\Sigma = \{a, b, c\}$.*
   (a) *Proposer un automate déterministe (pas nécessairement complet) qui reconnaît le langage sur Σ^* de tous les mots qui comprennent un nombre de lettres multiple de 3 et qui se terminent par c.*
   (b) *Proposer un automate déterministe (pas nécessairement complet) qui reconnaît le langage sur Σ^* de tous les mots de longueur supérieure ou égale à deux qui commencent et se terminent par la même lettre.*

   (a) La deuxième condition exclut le mot vide, le langage en question peut donc être défini : $\{u \in \Sigma^* \ / \ |u| = 3n, n > 0, \text{ et t.q. } u = vc, c \in \Sigma, v \in \Sigma^*\}$
   (b) $\{u \in \Sigma^* \ / \ |u| \geq 2 \text{ et } u = ava \text{ ou } u = bvb \text{ ou } u = cvc, v \in \Sigma^*\}$.





2. *Soit l'automate suivant. Appliquez l'algorithme de McNaughton & Yamada pour proposer une expression rationnelle caractérisant le langage reconnu.*

	a	b	c
$\rightarrow 1$	2		1
$\leftarrow 2$		2	1

La première étape donne l'automate généralisé (a), puis on supprime l'état 1 (au hasard), ce qui donne la version (b), et enfin on supprime l'état 2, ce qui donne l'expression (c)

(a)

$\nearrow$	1	2	F
I	ε	$\emptyset$	$\emptyset$
1	c	a	$\emptyset$
2	c	b	ε

(b)

$\nearrow$	2	F
I	$\emptyset \mid$	$\emptyset \mid$
	$\varepsilon.c^*.a$	$\varepsilon.c^*.\emptyset$
2	$b \mid$	ε
	$c.c^*.a$	$c.c^*.\emptyset$

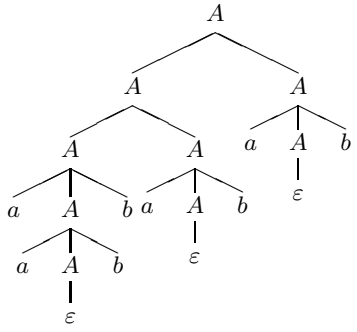
(c)

$\nearrow$	F
I	$\emptyset$
	$c^*a(b\|cc^*a)^*.\varepsilon$

3. *Soit la grammaire algébrique $\mathcal{G}$: $A \rightarrow aAb \mid AA \mid bAa \mid \varepsilon$.*
   (a) *Donner des mots de $L(\mathcal{G})$ de longueur 0, 2, 4.*
   longueur 0 : ε ; longueur 2 : ab, ba ; longueur 4 : $abba$, $aabb$, $baba$, ...
   (b) *Tout mot commençant par a se termine-t-il par b ?*
   Non, à cause de la règle $A \rightarrow AA$, on peut produire $a^n b^n b^k a^k$.
   (c) *Quel est le langage engendré par $\mathcal{G}$? On demande juste une caractérisation intuitive.*
   $L(\mathcal{G})$ est le langage de tous les mots contenant autant de a que de b. Cela peut se démontrer en faisant une récurrence sur le nombre de a et de b des proto-mots (récurrence sur les étapes de dérivation).
   (d) *Soit le mot $u = aabbbaab$. Donner pour u une dérivation droite, une dérivation gauche, et un arbre syntaxique.*

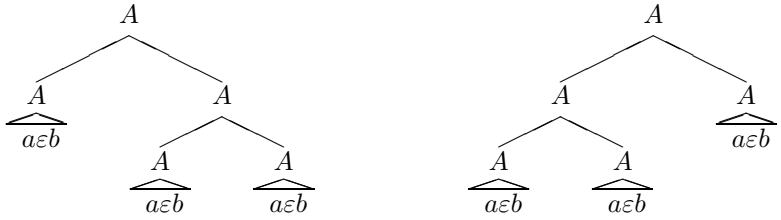
Dérivation gauche (parmi d'autres) :
$A \rightarrow AA \rightarrow AAA \rightarrow aAbAA \rightarrow aaAbbAA \rightarrow aabbAA \rightarrow aabbbAaA \rightarrow aabbbaA \rightarrow aabbbaaAb \rightarrow aabbbaab$

Dérivation droite (même analyse) :
$A \rightarrow AA \rightarrow AaAb \rightarrow Aab \rightarrow AAab \rightarrow AbAaab \rightarrow Abaab \rightarrow aAbbaab \rightarrow aaAbbbaab \rightarrow aabbbaab$



(e) *La grammaire $\mathcal{G}$ est-elle ambigüe ? Justifier.*

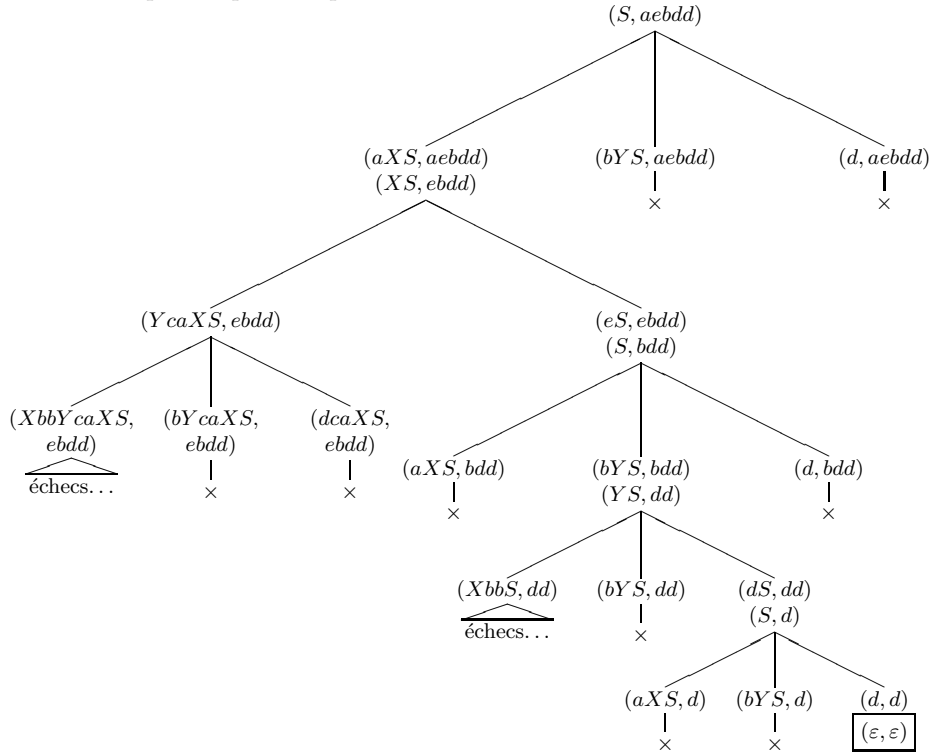
Oui, la grammaire est ambigüe, ce qui est facile à démontrer : il suffit de trouver un mot qui est produit par deux dérivations gauches différentes (ou qui a deux arbres syntaxiques différents). En voici un : *ababab* (l'arbre syntaxique est légèrement simplifié).



4. *Soit la grammaire* $S \to aXS \mid bYS \mid d$
$X \to YcaX \mid e$
$Y \to Xbb \mid bY \mid d$

(a) *Représenter l'« arbre d'exploration » d'une analyse descendante du mot $u = aebdd$. On ne demande pas de détailler toutes les branches.*

L'arbre contient plusieurs branches infinies, car la grammaire est récursive gauche. Les deux branches notées « échecs » comprennent des parties infinies si on applique l'algorithme standard qui compare les préfixes terminaux.



Si on ne fait pas figurer les transitions qui aboutissent directement à une paire $(x\alpha, y\beta)$, avec $x \neq y$ (x, y terminaux, α et β quelconques), cela simplifie l'arbre de façon importante : cf. figure 1, gauche.

(b) *Proposer une grammaire équivalente sous forme de Greibach qui reconnaisse le même langage (il peut être nécessaire de passer par plusieurs étapes).*

La grammaire étant récursive gauche (indirecte), il faut d'abord la dérécursiver avant de la traduire en forme de Greibach.

Le problème de récursivité vient des règles de la forme $X \to Y\alpha$ et $Y \to X\beta$. On remplace les règles $X \to Y\alpha$ par des règles où Y est « développé » (un autre ordre aurait été possible).

Au lieu de $X \to YcaX \mid e$, on aura donc : $X \to XbbcaX \mid bYcaX \mid dcaX \mid e$.

Cela donne une règle avec récursivité directe, qu'on va transformer en :
$X \to bYcaXX' \mid dcaXX' \mid eX'$
$X' \to bbcaXX' \mid \varepsilon$
La grammaire non récursive gauche est donc la suivante :

S	$\to$	$aXS \mid bYS \mid d$
X	$\to$	$bYcaXX' \mid dcaXX' \mid eX'$
X'	$\to$	$bbcaXX' \mid \varepsilon$
Y	$\to$	$Xbb \mid bY \mid d$

Il faut maintenant passer en forme de Greibach, ce qui n'est pas très compliqué, car une seule règle est concernée, c'est la règle $Y \to Xbb$, qu'on va « développer » :

S	$\to$	$aXS \mid bYS \mid d$
X	$\to$	$bYcaXX' \mid dcaXX' \mid eX'$
X'	$\to$	$bbcaXX' \mid \varepsilon$
Y	$\to$	$bYcaXX'bb \mid dcaXX'bb \mid eX'bb \mid bY \mid d$

En toute rigueur, il faut aussi supprimer les productions ε. Ici, seul X' s'efface, il faut donc ajouter à toutes les règles où X' apparaît en partie droite une règle où X' n'apparaît pas[1]. Cela donne :

S	$\to$	$aXS \mid bYS \mid d$
X	$\to$	$bYcaXX' \mid dcaXX' \mid eX' \mid bYcaX \mid dcaX \mid e$
X'	$\to$	$bbcaXX' \mid bbcaX$
Y	$\to$	$bYcaXX'bb \mid dcaXX'bb \mid eX'bb \mid bYcaXbb \mid dcaXbb \mid ebb \mid bY \mid d$

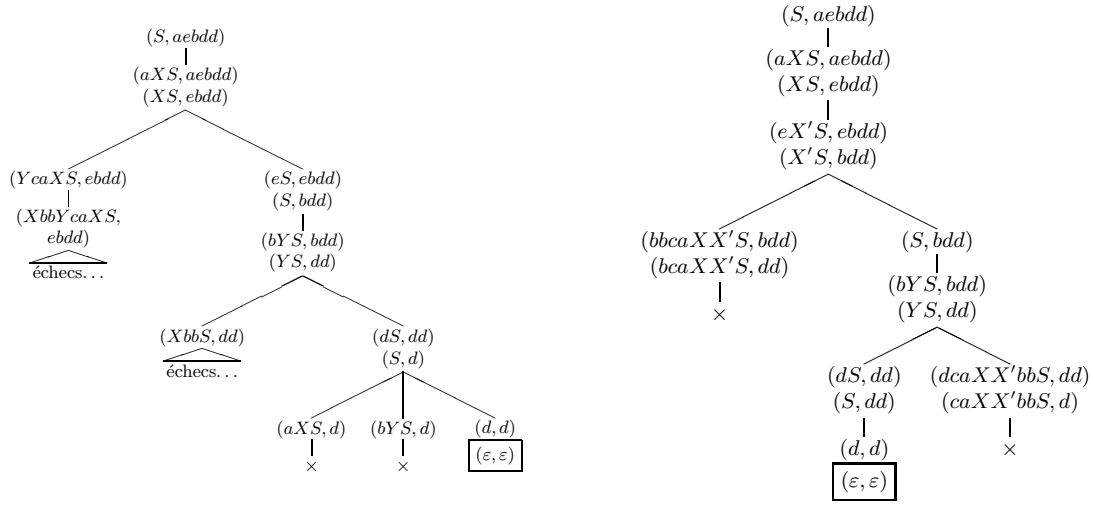
(c) *Représenter l'« arbre d'exploration » d'une analyse descendante du mot u avec cette nouvelle grammaire. On ne demande pas de détailler toutes les branches.*

On donne simplement l'arbre d'exploration simplifié : cf. figure 1, droite.

(d) *Peut-on (intuitivement) estimer le coût et le bénéfice de l'opération ?*

Le plus grand bénéfice est la finitude de l'arbre d'exploration : on garantit une analyse descendant en un temps fini. Les coûts induits sont d'une part la plus grande complexité de la grammaire (en terme de nombre de règles), d'autre part le coût de l'application des algorithmes de transformation. Enfin, l'arbre syntaxique, puisque la grammaire est altérée, n'est plus le même (pas dans tous les cas : pour notre mot *aebdd* l'analyse est la même).

FIG. 1 – Arbres simplifiés, 1re & 2e grammaire



[1] Cette opération peut donner lieu, en général, à une combinatoire coûteuse, mais avec cette grammaire-ci ce n'est pas le cas.