Partiel Automates 2020

Exercice 1

1) ε n'est pas déterministe car

* il y a des ε-transitions
* $|\delta_\varepsilon(A, a)| > 1$
* il y a deux états initiaux.

déterminisation : On pose $Q_0 = \{A, D\} \cup \varepsilon F(A) \cup \varepsilon F(D)$

$Q_0 = \{A, D, B\}$.

$\delta_\varepsilon(Q_0, a) = \{C, B\} \cup \varepsilon F(C) \cup \varepsilon F(B)$
$= \{C, B, D\} = Q_1$.

$\delta_\varepsilon(Q_0, b) = \{B\} \cup \varepsilon F(B) = \{B, D\} = Q_2$

$\delta_\varepsilon(Q_1, a) = \{D, C\} \cup \varepsilon F(D) \cup \varepsilon F(C)$
$= \{C, D, B\} = Q_1$.

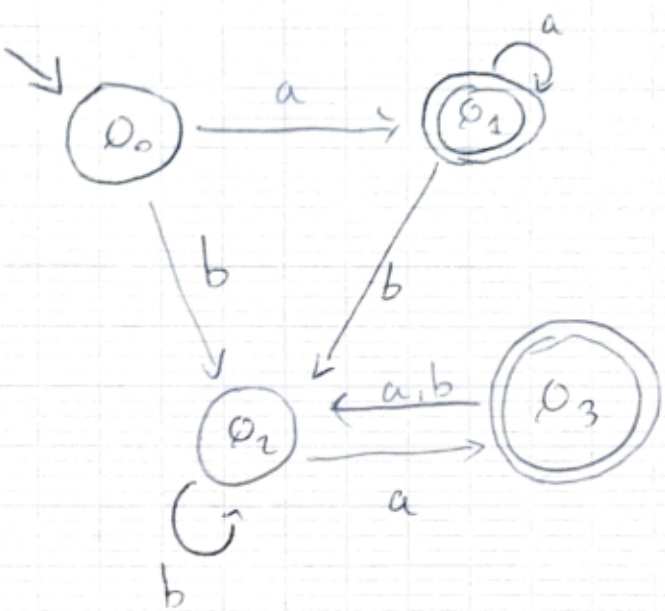
$\delta_\varepsilon(Q_1, b) = \{B\} \cup \varepsilon F(B) = \{B, D\} = Q_2$

$\delta_\varepsilon(Q_2, a) = \{C\} \cup \varepsilon F(C) = \{C\} = Q_3$

$\delta_\varepsilon(Q_1, b) = \{B\} \cup \varepsilon F(B) = \{B, D\} = Q_2$

$\delta_\varepsilon(Q_3, a) = \{D\} \cup \varepsilon F(D) = \{B, D\} = Q_2$

$\delta_\varepsilon(Q_3, b) = \{B\} \cup \varepsilon F(B) = \{B, D\} = Q_2$.



2) table de transition :

	État	a	b
→	Q_0	Q_1	Q_2
*	Q_1	Q_1	Q_2
	Q_2	Q_3	Q_2
*	Q_3	Q_2	Q_2

→ : initial

* : final

Exercice 2 :

1) $\begin{cases} L_E = b L_E + a L_G & (1) \\ L_G = b L_F + \Lambda & (2) \\ L_F = b L_F + a L_E & (3) \end{cases}$

2) (3) $\rightarrow$ $L_F = b L_F + a L_E = b^* a L_E$ (Lemme d'Arden)

(2) $\rightarrow$ $L_G = b L_F + \Lambda \underset{(3)}{=} b b^* a L_E + \Lambda$.

(1) $\rightarrow$ $L_E = b L_E + a L_G$ ~~$= b^* a L_G$ (lemme d'Arden)~~

$= b L_E + a b b^* a L_E + a$

$\begin{cases} L_E = (b + a b b^* a)^* a \\ L_F = b^* a (b + a b b^* a)^* a \\ L_G = b b^* a (b + a b b^* a)^* a + \Lambda \end{cases}$

expression régulière : $L_E = \boxed{(b + a b b^* a)^* a}$

Exercice 3 : On pose $q_0 = (a|\Lambda)((ba)^*|ba)$

$D_a(q_0) = (D_a(a|\Lambda)((ba)^*|ba)) | D_a((ba)^*|ba)$

$= ((ba)^*|ba) | \underbrace{D_a((ba)^*)}_{\phi} | \underbrace{D_a(ba)}_{=q}$

$= (ba)^*|ba = q_1.$

$D_b(q_0) = (D_b(a|\Lambda)((ba)^*|ba)) | D_b((ba)^*)$

$|D_b(ba)$

$= \phi | D_b(ba)(ba)^*|a$

$= a(ba)^*|a = q_2$

$D_a(q_1) = \phi$

$D_b(q_1) = q_2$

$D_a(q_2) = (ba)^*|\Lambda = q_3.$

$D_b(q_2) = \phi$

$D_a(q_3) = \phi$

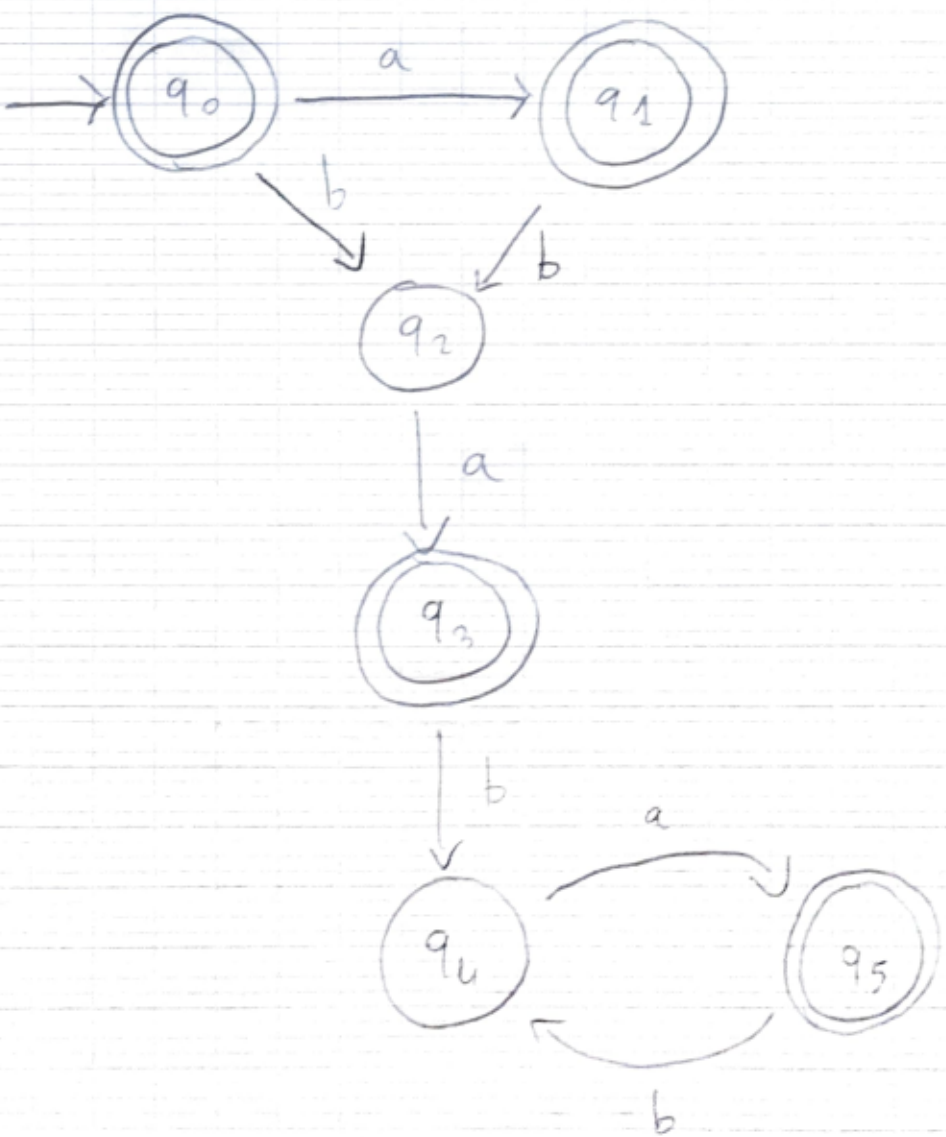
$D_b(q_3) = a(ba)^* = q_4$

$D_a(q_4) = (ba)^* = q_5$.

$D_b(q_4) = \phi$

$D_a(q_5) = \phi$

$D_b(q_5) = a(ba)^* = q_4$



Exercice 4:

1) Nouvelle grammaire sans récursivité à gauche :

$$\begin{cases} N \to R\,p\,I \mid R \mid I \\ I \to R\,i \\ R \to c\,R' \\ R' \to c\,R' \mid \Lambda \end{cases}$$

transformation en grammaire régulière :

$N \to cR'pI \mid cR' \mid cR'i$

$\to c(R'pI \mid R' \mid R'i)$

$\to cR'(\underbrace{pI \mid i \mid \Lambda}_{X})$

$\to cY$

$Y \to R'X \to cR'X \mid \quad X \to cY \quad \mid pI \mid i \mid \Lambda$

On pose $z \to \Lambda$

d'où $Y \to cY \mid pI \mid iz \mid \Lambda$

Pour l'instant, nous avons

$$N \to c\,Y$$
$$Y \to c\,Y \mid p\,I \mid ;\,z \mid \Lambda$$
$$z \to \Lambda$$

$$I \to R\,; \to c\underbrace{R\,';z}_{A} \to c\,A$$

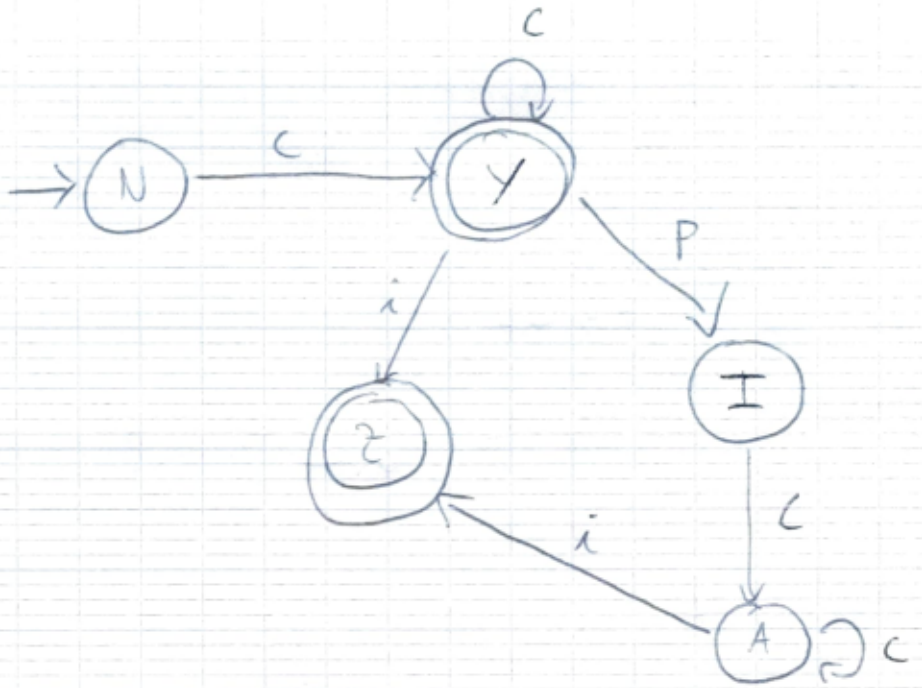
$$A \to R\,';z \to c\underbrace{R\,';z}_{A} \mid ;z$$

$$A \to c\,A \mid ;\,z$$

finalement :

$$\begin{cases} N \to c\,Y \\ Y \to c\,Y \\ Y \to p\,I \\ Y \to ;\,z \\ Y \to \Lambda \\ I \to c\,A \\ A \to c\,A \mid ;\,z \\ z \to \Lambda \end{cases}$$

2)



Exercice 5

1)

(a) G_0 est récursive à gauche à cause de la règle (3)

G_1 :

$$\begin{cases} O \to \{\} \\ O \to \{L\} \\ L \to F\ L' \\ L' \to ,\ F\ L' \\ F \to id : num \\ L' \to \Lambda \end{cases}$$

par la suite nous noterons '{' et '}' pour les symboles terminaux { et } pour éviter toute confusion dans les ensembles...

(b)

$SD(G \to \{\}) = \{\{\}\}$

$SD(G \to \{L\}) = \{\{\}\}$.

$SD(L' \to FL') = First(F) = id$ car $\Lambda \notin First(F)$

$SD(L' \to ,FL') = \{,\}$.

$SD(L' \to \Lambda) = Follow(L') = Follow(L) = \{\}\}$.

$SD(F \to id : num) = \{id\}$

(c) G_1 n'est pas LL(1) car $SD(G \to \{1) \cap SD(G \to \{2) = \{\{\}\} \neq \emptyset$

G_2

$G \to \{X$

$X \to \}$

$X \to L\}$.

$L \to FL'$

$L' \to ,FL'$

$L' \to \Lambda$

$F \to id : num$

Nous ne recalculerons que les symboles directeurs de G et de X (les autres étant inchangés) :

$SD(G \to \{X) = \{\{\}\}$

$SD(X \to \}) = \{\}\}$

$SD(X \to L\}) = First(L) = First(F) = \{id\}$

Cette fois G_2 est bien LL(1) puisque l'intersection des symboles directeurs des symboles non terminaux de G_2 est vide.

2)

(a) La construction monadique dans la programmation des analyseurs descendants récursifs permet d'éviter les lourdeurs que l'on aurait sans (pattern matching imbriqués, gestion du cas d'erreur, etc.)

(b) * L'analyseur lexical permet de découper une chaîne de caractères (stream), en petits bouts appelés tokens. Cela va nous permettre de reconnaître des patterns dans l'analyseur syntaxique. Une erreur est levée si un "mot" n'est pas conforme à au moins l'une des règles du lexique.

* L'analyseur syntaxique nous permets de vérifier que les tokens reconnus par l'analyseur lexical respectent bien certaines règles de production. C'est à dire que les tokens se trouvent dans le bon ordre vis-à-vis des règles imposées. Si le stream ne suit pas les règles de production, une erreur est levée.

(c) Dans les analyseurs syntaxique, les actions consistent à : vérifier qu'un token a reconnu pour la règle en question (sinon on retourne une erreur), le passer à une autre règle si le premier symbole est non-terminal ou consommer le symbole s'il match avec le premier symbole terminal.