Université Laval
Faculté des sciences et de génie
Département d'informatique et de génie logiciel
IFT-3101

Danny Dubé
Hiver 2019
~~Version : 26 février~~
Version : 27 février

Corrigé du travail pratique #1

Réponses

Note : les réponses aux différentes questions ne sont pas toujours uniques.

1. (a) ϵ,
   `c`,
   `ca`,
   `car`,
   `cart`,
   `carte`

   (b) Attention ! Il fallait éviter les préfixes dégénérés.
   `er`,
   `ker`,
   `oker`,
   `r`

   (c) Attention ! Il fallait éviter les sous-chaînes dégénérées.
   `a`,
   `ai`,
   `ain`,
   `i`,
   `in`,
   `m`,
   `ma`,
   `mai`,
   `n`

   (d) ϵ,
   `a`,
   `aa`,
   `aaa`,
   `aaaa`,
   `aaaab`,
   `aaab`,
   `aab`,
   `ab`,
   `b`

(e) abcabcab,
bcabcabc,
cabcabca

2. (a) Une apparition obligatoire de chacun des trois symboles b, c et c est placée suivant un certain ordre puis des symboles additionnels quelconques sont insérés avant, entre et après les apparitions obligatoires. Il y a 3 ordres pour les apparitions obligatoires.

$$\begin{array}{rcl} A & \rightarrow & (\texttt{a} \mid \texttt{b} \mid \texttt{c})^* \\ B & \rightarrow & A\,\texttt{b}\,A\,\texttt{c}\,A\,\texttt{c}\,A \\ & \mid & A\,\texttt{c}\,A\,\texttt{b}\,A\,\texttt{c}\,A \\ & \mid & A\,\texttt{c}\,A\,\texttt{c}\,A\,\texttt{b}\,A \end{array}$$

(b) On crée une définition régulière qui comporte d'abord ~~les alias d_0 à d_5~~ les alias d_0 à d_8. L'alias d_i génère ~~des chaînes~~ les chaînes faites de chiffres dans $\{0, \ldots, 4\}$ dont ~~dont~~ la somme des chiffres égale i. — L'alias f génère les chaînes qui se terminent par un chiffres élevé. — Enfin, on crée l'alias principal e qui génère le langage désiré.

$$\begin{array}{rcl} d_0 & \rightarrow & \texttt{0}^* \\ d_1 & \rightarrow & d_0\,\texttt{1}\,d_0 \\ d_2 & \rightarrow & d_1\,d_1 \mid d_0\,\texttt{2}\,d_0 \\ d_3 & \rightarrow & d_1\,d_2 \mid d_2\,d_1 \mid d_0\,\texttt{3}\,d_0 \\ d_4 & \rightarrow & d_1\,d_3 \mid d_2\,d_2 \mid d_3\,d_1 \mid d_0\,\texttt{4}\,d_0 \\ \cancel{d_5} & \cancel{\rightarrow} & \cancel{d_1\,d_4 \mid d_2\,d_3 \mid d_3\,d_2 \mid d_4\,d_1 \mid d_0\,\texttt{5}\,d_0} \\ \cancel{e} & \cancel{\rightarrow} & \cancel{d_5\,(\texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{6} \mid \texttt{7} \mid \texttt{8} \mid \texttt{9})^*} \\ d_5 & \rightarrow & d_1\,d_4 \mid d_2\,d_3 \mid d_3\,d_2 \mid d_4\,d_1 \\ d_6 & \rightarrow & d_1\,d_5 \mid d_2\,d_4 \mid d_3\,d_3 \mid d_4\,d_2 \\ d_7 & \rightarrow & d_1\,d_6 \mid d_2\,d_5 \mid d_3\,d_4 \mid d_4\,d_3 \\ d_8 & \rightarrow & d_1\,d_7 \mid d_2\,d_6 \mid d_3\,d_5 \mid d_4\,d_4 \\ f & \rightarrow & [0\text{–}4]^*\,[5\text{–}9] \\ e & \rightarrow & (d_5 \mid d_6 \mid d_7 \mid d_8 \mid f)\,[0\text{–}9]^* \end{array}$$

(c) Les deux sous-chaînes requises peuvent apparaître dans un ordre ou dans l'autre et elles peuvent même se chevaucher.

$$\begin{array}{rcl} A & \rightarrow & (\texttt{a} \mid \texttt{b} \mid \texttt{c} \mid \texttt{d})^* \\ B & \rightarrow & A\,\texttt{abc}\,A\,\texttt{bcd}\,A \mid A\,\texttt{bcd}\,A\,\texttt{abc}\,A \mid A\,\texttt{abcd}\,A \end{array}$$

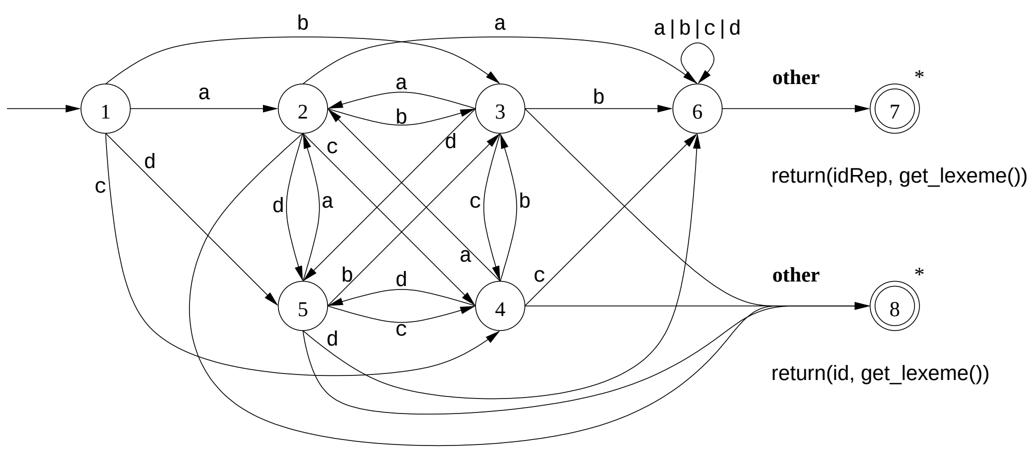
(d) A priori, on pourrait envisager de concevoir des mots du langage de la manière suivante. La forme du mot serait initialement '. . . c . . . a . . . b . . .', pour s'assurer d'avoir la sous-séquence '`cab`', puis on placerait la sous-chaîne '`abc`' en remplaçant l'un des '...' par '...`abc`...' et, enfin, on remplacerait les '...' par n'importe quelles suites de symboles. Cependant, ce serait sans tenir compte du fait que la sous-chaîne '`abc`', de par sa présence, fournit certains symboles nécessaires à la sous-séquence '`cab`'. En fait, on n'a que deux formes de chaînes à considérer : il suffit de se poser la question à savoir s'il apparaît un '`c`' indépendant avant la première apparition de '`abc`'.

$$\begin{array}{rcl} A & \rightarrow & (\mathtt{a} \mid \mathtt{b} \mid \mathtt{c})^* \\ B & \rightarrow & A\,\mathtt{abc}\,A\,\mathtt{a}\,A\,\mathtt{b}\,A \mid A\,\mathtt{c}\,A\,\mathtt{abc}\,A \end{array}$$

(e) On peut observer que toute chaîne du langage tombe dans l'un des cas suivants :
soit la chaîne ne contient pas de '`a`' ;
soit, après le premier '`a`', la chaîne ne contient pas de '`c`' ;
soit, après le premier '`c`' qui suit le premier '`a`', la chaîne ne contient pas de '`d`' ;
etc.

$$\begin{array}{rl} & (\mathtt{b} \mid \mathtt{c} \mid \mathtt{d})^* \\ \mid & (\mathtt{b} \mid \mathtt{c} \mid \mathtt{d})^*\,\mathtt{a}\,(\mathtt{a} \mid \mathtt{b} \mid \mathtt{d})^* \\ \mid & (\mathtt{b} \mid \mathtt{c} \mid \mathtt{d})^*\,\mathtt{a}\,(\mathtt{a} \mid \mathtt{b} \mid \mathtt{d})^*\,\mathtt{c}\,(\mathtt{a} \mid \mathtt{b} \mid \mathtt{c})^* \\ \mid & (\mathtt{b} \mid \mathtt{c} \mid \mathtt{d})^*\,\mathtt{a}\,(\mathtt{a} \mid \mathtt{b} \mid \mathtt{d})^*\,\mathtt{c}\,(\mathtt{a} \mid \mathtt{b} \mid \mathtt{c})^*\,\mathtt{d}\,(\mathtt{a} \mid \mathtt{b} \mid \mathtt{d})^* \end{array}$$

3. Tout d'abord, veuillez noter que la question n'était pas claire en ce qui concerne la question de la fin de l'identificateur. Il aurait fallu avoir un cinquième symbole, par exemple ␣, qui aurait permis de terminer les identificateurs sans avoir besoin de consommer tout le texte d'entrée. Peu importe, dans la correction, on va accepter que ayez supposé qu'il y avait un cinquième symbole ou pas. Nous ne présentons qu'un seul diagramme mais celui-ci a deux états acceptants, tout dépendant si une répétition a été aperçue ou pas.



4. Voici le pseudo-code.

```
while (true)
{
  switch(state)
  {
    ...
    case 1:
      c := buffer[forward];
      forward ++;
      if (c == '-')
        state := 3;
      else if (is_digit(c))
        state := 2;
      else
        state := fail();
      break;
    case 2:
      c := buffer[forward];
      forward ++;
      if (c == '.')
        state := 5;
      else if (is_digit(c))
        state := 2;
      else
        state := 4;
      break;
    case 3:
      c := buffer[forward];
      forward ++;
      if (is_digit(c))
        state := 2;
      else
        state := fail();
      break;
    case 4:
      forward --;
      jeton := num;
      attribut := gettoken();
      token_beginning := forward;
      return jeton;
    case 5:
      c := buffer[forward];
      forward ++;
      if (is_digit(c))
        state := 6;
      else
        state := fail();
      break;
    case 6:
      c := buffer[forward];
      forward ++;
      if (is_digit(c))
        state := 6;
      else
        state := 7;
      break;
    case 7:
      forward --;
      jeton := num;
      attribut := gettoken();
      token_beginning := forward;
      return jeton;
    ...
  }
}
```

5. Notons tout d'abord que la fonction de bilan possède une propriété fort utile :

$$f(uv) \quad = \quad f(u) + f(v) \ .$$

Je fais une preuve par induction sur la longueur des chaînes dans L_5. Notons que, pour respecter la contrainte sur le nombre de '$\mathtt{a}$' et de '$\mathtt{b}$', toute chaîne dans L_5 a une longueur qui est un multiple de 3.

Base de l'induction. Considérons la chaîne w de longueur 0. Cette chaîne est nécessairement $w = \epsilon$. À cause de la première production, G_5 est capable de générer w.

Hypothèse d'induction (HI)**.** Supposons que, pour toute chaîne w dans L_5 de longueur de moins de n symboles, G_5 est capable de générer w.

Pas d'induction. Soit $w \in L_5$ de longueur $n > 0$. Considérons la suite des bilans calculés sur tous les préfixes de $w = c_1 \dots c_n$:

$$0 = f(\epsilon),\ f(c_1),\ \dots,\ f(c_1 \dots c_{n-1}),\ f(c_1 \dots c_n) = 0 \ .$$

Premier cas : la suite de bilans passe par zéro en un autre endroit qu'aux extrémités. En d'autre mots, la suite de bilans a la forme :

$$0, \dots, 0, \dots, 0 \ .$$

Dans ce cas, on peut découper w en deux sous-chaînes non-vides qui ont chacune un bilan nul ; i.e. $w = uv$ et $f(u) = 0 = f(v)$. On note que chacune de u et v est de longueur plus petite que n. On peut donc utiliser l'HI et savoir que $S \Rightarrow^* u$ et $S \Rightarrow^* v$. On peut donc donner une dérivation pour w :

$$S \Rightarrow S\,S \Rightarrow^* u\,S \Rightarrow^* u\,v = w \ .$$

Deuxième cas : w commence par '$\mathtt{a}$' et se termine par '$\mathtt{a}$'. Dans ce cas, la suite de bilan doit avoir la forme suivante :

$$0, +1, \dots, -1, 0 \ .$$

Or, pour que le bilan soit passé du positif au négatif sans passer par zéro, il faut nécessairement qu'un '$\mathtt{b}$' ait fait passer le bilan de $+1$ à -1. Donc, la suite de bilan doit avoir la forme suivante :

$$0, +1, \dots, +1, -1, \dots, -1, 0 \ .$$

Ceci implique qu'on peut décomposer w en la concaténation $\mathtt{a}u\mathtt{b}v\mathtt{a}$ où $f(u) = f(v) = 0$. Bien entendu, u et v sont de longueur plus petite que n, donc l'HI s'applique et on peut donc donner une dérivation de w :

$$S \Rightarrow \mathtt{a}S\mathtt{b}S\mathtt{a} \Rightarrow^* \mathtt{a}u\mathtt{b}S\mathtt{a} \Rightarrow^* \mathtt{a}u\mathtt{b}v\mathtt{a} = w \ .$$

Troisième cas : w commence par '$\mathtt{a}$' et se termine par '$\mathtt{b}$'. Dans ce cas, la suite de bilan doit avoir la forme suivante :

$$0, +1, \dots, +2, 0 \ .$$

Or, il faut nécessairement qu'un '$\texttt{a}$' ait fait passer le bilan de $+1$ à $+2$. Donc, la suite de bilan doit avoir la forme suivante :

$$0, +1, \dots, +1, +2, \dots, +2, 0 \ .$$

Ceci implique qu'on peut décomposer w en la concaténation $\texttt{a}u\texttt{a}v\texttt{b}$ où $f(u) = f(v) = 0$. Encore une fois, u et v sont de longueur plus petite que n, donc l'HI s'applique et on peut donc donner une dérivation de w :

$$S \Rightarrow \texttt{a}S\texttt{a}S\texttt{b} \Rightarrow^* \texttt{a}u\texttt{a}S\texttt{b} \Rightarrow^* \texttt{a}u\texttt{a}v\texttt{b} = w \ .$$

Quatrième cas : w commence par '$\texttt{b}$' et se termine par '$\texttt{a}$'. Ce cas se traite de manière similaire au cas précédent.
Cinquième cas : w commence par '$\texttt{b}$' et se termine par '$\texttt{b}$'. Dans ce cas, la suite de bilan doit avoir la forme suivante :

$$0, -2, \dots, +2, 0 \ .$$

Or, il est inutile de considérer ce cas car la suite de bilans doit nécessairement passer par zéro entre le -2 et le $+2$ et cette situation a déjà été captée par le premier cas. Ceci complète le pas d'induction et, par la même occasion, la preuve que $L_5 \subseteq L(G_5)$.

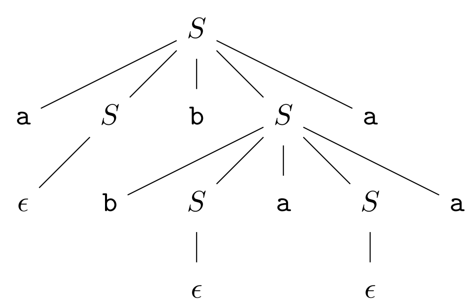
6. Dérivation à gauche d'abord :

$$
\begin{aligned}
& S \\
\Rightarrow\ & \mathtt{a}\,S\,\mathtt{b}\,S\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,\mathtt{b}\,S\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,\mathtt{b}\,\mathtt{b}\,S\,\mathtt{a}\,S\,\mathtt{a}\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,\mathtt{b}\,\mathtt{b}\,\mathtt{a}\,S\,\mathtt{a}\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,\mathtt{b}\,\mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}
\end{aligned}
$$

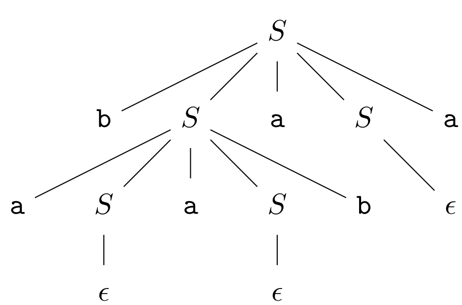
Dérivation à droite d'abord :

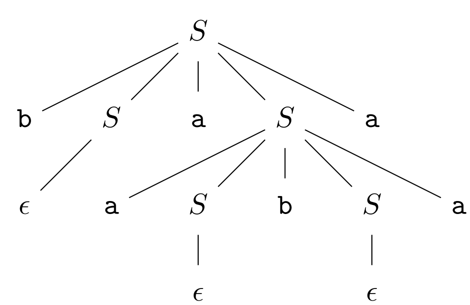
$$
\begin{aligned}
& S \\
\Rightarrow\ & \mathtt{a}\,S\,\mathtt{b}\,S\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,S\,\mathtt{b}\,\mathtt{b}\,S\,\mathtt{a}\,S\,\mathtt{a}\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,S\,\mathtt{b}\,\mathtt{b}\,S\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,S\,\mathtt{b}\,\mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\
\Rightarrow\ & \mathtt{a}\,\mathtt{b}\,\mathtt{b}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}
\end{aligned}
$$

Arbre de dérivation :



7. On peut donner deux arbres de dérivation différents pour la chaîne `baabaa`.





8. Nous commençons avec la grammaire originale.

$$\begin{array}{rcl} E & \rightarrow & E\,\textbf{ou}\,T \mid T \\ T & \rightarrow & T\,\textbf{et}\,F \mid F \\ F & \rightarrow & \textbf{non}\,F \mid A \\ A & \rightarrow & (\,E\,) \mid \textbf{faux} \mid \textbf{vrai} \mid \textbf{id} \end{array}$$

Pour $i = 1$ **:** On nettoie les E-productions.

- La boucle sur j est vide. (C'est-à-dire qu'il n'y a aucun non-terminal placé *avant* E dans l'ordre des non-terminaux.)
- Il faut éliminer la récursion à gauche immédiate (c'est-à-dire, toute production de la forme $E \rightarrow E\alpha$). On obtient la grammaire :

$$\begin{array}{rcl} E & \rightarrow & T\,E' \\ E' & \rightarrow & \textbf{ou}\,T\,E' \mid \epsilon \\ T & \rightarrow & T\,\textbf{et}\,F \mid F \\ F & \rightarrow & \textbf{non}\,F \mid A \\ A & \rightarrow & (\,E\,) \mid \textbf{faux} \mid \textbf{vrai} \mid \textbf{id} \end{array}$$

Pour $i = 2$ **:** On nettoie les T-productions.

- La boucle sur j se résume à $j = 1$. Il faut éliminer toute production de la forme $T \rightarrow E\alpha$. Rien à faire.
- Il faut éliminer la récursion à gauche immédiate. On obtient la grammaire :

$$\begin{array}{rcl} E & \rightarrow & T\,E' \\ E' & \rightarrow & \textbf{ou}\,T\,E' \mid \epsilon \\ T & \rightarrow & F\,T' \\ T' & \rightarrow & \textbf{et}\,F\,T' \mid \epsilon \\ F & \rightarrow & \textbf{non}\,F \mid A \\ A & \rightarrow & (\,E\,) \mid \textbf{faux} \mid \textbf{vrai} \mid \textbf{id} \end{array}$$

Pour $i = 3$ **:** On nettoie les F-productions.

- Pour $j = 1$, il faut éliminer toute production de la forme $F \rightarrow E\alpha$. Rien à faire.
- Pour $j = 2$, il faut éliminer toute production de la forme $F \rightarrow T\alpha$. Rien à faire.
- Il n'y a pas de récursion à gauche immédiate chez F. Rien à faire.

Pour $i = 4$ **:** On nettoie les A-productions.

- Pour $j = 1$, il faut éliminer toute production de la forme $A \to E\alpha$. Rien à faire.
- Pour $j = 2$, il faut éliminer toute production de la forme $A \to T\alpha$. Rien à faire.
- Pour $j = 3$, il faut éliminer toute production de la forme $A \to F\alpha$. Rien à faire.
- Il n'y a pas de récursion à gauche immédiate chez A. Rien à faire.

La dernière grammaire donnée est la grammaire souhaitée.

9. Voici le pseudo-code.

```
void S()
{
  if (peekToken().type == 'a')
    {
      readToken('a');
      S();
      readToken('a');
    }
  else if (peekToken().type == 'b')
    {
      readToken('b');
      S();
      readToken('b');
    }
  else
    {
      readToken('c');
    }
}
```

10.

Pile	Entrée	Sortie
S \$	bbbaabdccddd\$	$S \to \mathtt{b}\,S\,\mathtt{d}$
b S d \$	bbbaabdccddd\$	(Consomme b)
S d \$	bbaabdccddd\$	$S \to \mathtt{b}\,S\,\mathtt{d}$
b S d d \$	bbaabdccddd\$	(Consomme b)
S d d \$	baabdccddd\$	$S \to \mathtt{b}\,S\,\mathtt{d}$
b S d d d \$	baabdccddd\$	(Consomme b)
S d d d \$	aabdccddd\$	$S \to \mathtt{a}\,S\,\mathtt{c}$
a S c d d d \$	aabdccddd\$	(Consomme a)
S c d d d \$	abdccddd\$	$S \to \mathtt{a}\,S\,\mathtt{c}$
a S c c d d d \$	abdccddd\$	(Consomme a)
S c c d d d \$	bdccddd\$	$S \to \mathtt{b}\,S\,\mathtt{d}$
b S d c c d d d \$	bdccddd\$	(Consomme b)
S d c c d d d \$	dccddd\$	$S \to \epsilon$
d c c d d d \$	dccddd\$	(Consomme d)
c c d d d \$	ccddd\$	(Consomme c)
c d d d \$	cddd\$	(Consomme c)
d d d \$	ddd\$	(Consomme d)
d d \$	dd\$	(Consomme d)
d \$	d\$	(Consomme d)
\$	\$	(Succès)

11. Posons les contraintes sur les ensembles FIRST qui sont nécessaires au calcul de FIRST(S).

$$\begin{aligned}
\mathrm{FIRST}(S) &= \mathrm{FIRST}(\epsilon) \cup \mathrm{FIRST}(\mathtt{a}S\mathtt{a}S\mathtt{b}) \cup \mathrm{FIRST}(\mathtt{a}S\mathtt{b}S\mathtt{a}) \\
&\quad \cup \mathrm{FIRST}(\mathtt{b}S\mathtt{a}S\mathtt{a}) \cup \mathrm{FIRST}(SS) \\
&= \{\epsilon\} \cup \{\mathtt{a}\} \cup \{\mathtt{a}\} \cup \{\mathtt{b}\} \cup \mathrm{FIRST}(SS) \\
&= \{\epsilon, \mathtt{a}, \mathtt{b}\} \cup (\mathrm{FIRST}(S) - \{\epsilon\}) \cup \mathrm{FIRST}(S) \\
&= \{\epsilon, \mathtt{a}, \mathtt{b}\} \cup \mathrm{FIRST}(S)
\end{aligned}$$

La plus petite solution à cette contrainte nous donne l'ensemble suivant.

	FIRST
S	$\{\mathtt{a}, \mathtt{b}, \epsilon\}$

Maintenant, posons les contraintes sur l'ensemble FOLLOW qui sont nécessaires au calcul de FOLLOW(S).

$$\begin{array}{lllll}
\mathrm{FOLLOW}(S) & \supseteq & \{\$\} & & \text{symb. de départ} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(\mathtt{a}\,S\,\mathtt{b}) - \{\epsilon\} \;=\; \{\mathtt{a}\} & & \text{appar. dans 2e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(\mathtt{b}) - \{\epsilon\} \;=\; \{\mathtt{b}\} & & \text{appar. dans 2e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(\mathtt{b}\,S\,\mathtt{a}) - \{\epsilon\} \;=\; \{\mathtt{b}\} & & \text{appar. dans 3e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(\mathtt{a}) - \{\epsilon\} \;=\; \{\mathtt{a}\} & & \text{appar. dans 3e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(\mathtt{a}\,S\,\mathtt{a}) - \{\epsilon\} \;=\; \{\mathtt{a}\} & & \text{appar. dans 4e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(\mathtt{a}) - \{\epsilon\} \;=\; \{\mathtt{a}\} & & \text{appar. dans 4e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(S) - \{\epsilon\} \;=\; \{\mathtt{a}, \mathtt{b}\} & & \text{appar. dans 5e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FIRST}(\epsilon) - \{\epsilon\} \;=\; \{\} & & \text{appar. dans 5e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FOLLOW}(S) & & \text{appar. fin 5e } S\text{-prod.} \\
\mathrm{FOLLOW}(S) & \supseteq & \mathrm{FOLLOW}(S) & & \text{appar. fin 5e } S\text{-prod.}
\end{array}$$

La plus petite solution à ces contraintes nous donne l'ensemble suivant.

	FOLLOW
S	$\{\mathtt{a}, \mathtt{b}, \$\}$

12. Voici les ensembles PREDICT qui justifient les insertions de productions dans la table d'analyse.

Production	PREDICT
$A \rightarrow B$	$\{$a, b, c$\}$
$A \rightarrow \texttt{d}\,B$	$\{$d$\}$
$B \rightarrow \texttt{c}\,A$	$\{$c$\}$
$B \rightarrow C\,\texttt{b}$	$\{$a, b$\}$
$B \rightarrow \texttt{a}$	$\{$a$\}$
$C \rightarrow \texttt{a}\,B$	$\{$a$\}$
$C \rightarrow \epsilon$	$\{$b$\}$

Non-terminal	Symbole d'entrée				
	a	b	c	d	$
A	$A \rightarrow B$	$A \rightarrow B$	$A \rightarrow B$	$A \rightarrow \texttt{d}\,B$	
B	$B \rightarrow C\,\texttt{b}$ $B \rightarrow \texttt{a}$	$B \rightarrow C\,\texttt{b}$	$B \rightarrow \texttt{c}\,A$		
C	$C \rightarrow \texttt{a}\,B$	$C \rightarrow \epsilon$			