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

Danny Dubé
Automne 2020
Version: 12 octobre

Travail pratique #1
Analyses lexicale et syntaxique

Questions

1. (*5 points.*) Soit $w = \texttt{food}$ et soit $S = \{\texttt{d}, \texttt{f}, \texttt{fo}, \texttt{foo}, \texttt{o}, \texttt{od}, \texttt{oo}, \texttt{ood}\}$. Qu'est l'ensemble S par rapport à w?

   (a) L'ensemble des préfixes de w.

   (b) L'ensemble des préfixes propres de w.

   (c) L'ensemble des suffixes de w.

   (d) L'ensemble des suffixes propres de w.

   (e) L'ensemble des sous-chaînes de w.

   (f) L'ensemble des sous-chaînes propres de w.

   (g) L'ensemble des sous-séquences de w.

   (h) L'ensemble des sous-séquences propres de w.

2. (*5 points.*) Considérons le langage

$$\left\{ w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^* \;\middle|\; \begin{array}{l} w \text{ a } \texttt{cab} \text{ comme sous-chaîne mais} \\ w \text{ n'a pas } \texttt{bb} \text{ comme sous-chaîne} \end{array} \right\}.$$

   Parmi les choix suivants, un seul est capable de générer le langage. Lequel? Chacun des choix est soit une expression régulière, soit une définition régulière.

   (a) $[\texttt{abc}]^* \cdot \texttt{c} \cdot \texttt{a} \cdot \texttt{b} \cdot [\texttt{abc}]^* \;\middle|\; \texttt{b}^? \cdot \Big([\texttt{ac}] \cdot \texttt{b}^?\Big)^*$

   (b) $\texttt{b}^? \cdot \Big([\texttt{ac}] \cdot \texttt{b}^?\Big)^* \cdot \texttt{c} \cdot \texttt{a} \cdot \texttt{b} \cdot \Big([\texttt{ac}] \cdot \texttt{b}^?\Big)^*$

   (c) $f \;\rightarrow\; \texttt{aa} \mid \texttt{ab} \mid \texttt{ac} \mid \texttt{ba} \mid \texttt{bc} \mid \texttt{ca} \mid \texttt{cb} \mid \texttt{cc}$ // note: pas de $\texttt{bb}$
   $g \;\rightarrow\; f^* \cdot \texttt{c} \cdot \texttt{a} \cdot \texttt{b} \cdot f^*$

   (d) $f \;\rightarrow\; [\texttt{ac}]^+ \cdot \texttt{b} \cdot [\texttt{ac}]^+$
   $g \;\rightarrow\; f^* \cdot \texttt{c} \cdot \texttt{a} \cdot \texttt{b} \cdot f^*$

3. (*5 points.*) Considérons le langage des chaînes tirées de l'alphabet $\{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}\}$ dont les chiffres, additionnés ensemble, donnent une somme d'au moins 4. Parmi les choix suivants, un seul **n'**est **pas** capable de générer le langage. Lequel? Chacun des choix est soit une expression régulière, soit une définition régulière.

(a)
$$\begin{array}{lcl} d_0 & \to & \mathtt{0}^* \\ d_1 & \to & d_0 \cdot \mathtt{1} \cdot d_0 \\ d_2 & \to & d_0 \cdot \mathtt{2} \cdot d_0 \mid d_1 \cdot d_1 \\ d_3 & \to & d_0 \cdot \mathtt{3} \cdot d_0 \mid d_1 \cdot d_2 \mid d_2 \cdot d_1 \\ d_4 & \to & d_1 \cdot d_3 \mid d_2 \cdot d_2 \mid d_1 \cdot d_3 \\ d_5 & \to & d_1 \cdot d_4 \mid d_2 \cdot d_3 \mid d_3 \cdot d_2 \mid d_4 \cdot d_1 \\ d_6 & \to & d_1 \cdot d_5 \mid d_2 \cdot d_4 \mid d_3 \cdot d_3 \mid d_4 \cdot d_2 \mid d_5 \cdot d_1 \\ e & \to & [0\text{–}9]^* \\ f & \to & e \cdot \Big(d_4 \mid d_5 \mid d_6 \mid [4\text{–}9]\Big) \cdot e \end{array}$$

(b)
$$\begin{array}{lcl} d_0 & \to & \mathtt{0}^* \\ d_1 & \to & d_0 \cdot \mathtt{1} \cdot d_0 \\ d_2 & \to & d_1 \cdot \mathtt{1} \cdot d_0 \mid d_0 \cdot \mathtt{2} \cdot d_0 \\ d_3 & \to & d_2 \cdot \mathtt{1} \cdot d_0 \mid d_1 \cdot \mathtt{2} \cdot d_0 \mid d_0 \cdot \mathtt{3} \cdot d_0 \\ e & \to & \Big(d_3 \cdot [1\text{–}9] \mid d_2 \cdot [2\text{–}9] \mid d_1 \cdot [3\text{–}9] \mid d_0 \cdot [4\text{–}9]\Big) \cdot [0\text{–}9]^* \end{array}$$

(c)
$$\begin{array}{lcl} d_1 & \to & [1\text{–}9] \\ d_2 & \to & [2\text{–}9] \\ d_3 & \to & [3\text{–}9] \\ d_4 & \to & [4\text{–}9] \\ e & \to & [0\text{–}9]^* \\ f & \to & e \cdot d_4 \cdot e \\ & \mid & e \cdot d_1 \cdot e \cdot d_3 \cdot e \mid e \cdot d_2 \cdot e \cdot d_2 \cdot e \mid e \cdot d_3 \cdot e \cdot d_1 \cdot e \\ & \mid & e \cdot d_1 \cdot e \cdot d_1 \cdot e \cdot d_2 \cdot e \mid e \cdot d_1 \cdot e \cdot d_2 \cdot e \cdot d_1 \cdot e \mid e \cdot d_2 \cdot e \cdot d_1 \cdot e \cdot d_1 \cdot e \\ & \mid & e \cdot d_1 \cdot e \cdot d_1 \cdot e \cdot d_1 \cdot e \cdot d_1 \cdot e \end{array}$$

(d) $\mathtt{0}^* \cdot [\mathtt{1}\text{–}\mathtt{9}]^+ \cdot \mathtt{0}^* \cdot [\mathtt{1}\text{–}\mathtt{9}]^+ \cdot \mathtt{0}^* \cdot [\mathtt{1}\text{–}\mathtt{9}]^+ \cdot \mathtt{0}^* \cdot [\mathtt{1}\text{–}\mathtt{9}]^+ \cdot \mathtt{0}^*$

4. (*5 points.*) Considérons l'alphabet $\Sigma = \{\texttt{/}, \star, \texttt{␣}, \texttt{a}\}$ ainsi que l'expression régulière suivante.

$$\texttt{/} \cdot \star \cdot (\star^* \cdot [\texttt{␣a}])^* \cdot \star^+ \cdot \texttt{/}$$

Nous expliquons brièvement la notation adoptée ici. Dans Σ, nous retrouvons la barre oblique (i.e. le *slash*), l'astérisque, le caractère d'espacement ainsi que la lettre `a`. Notons que nous dénotons l'astérisque par $\star$ dans Σ afin de le distinguer de l'astérisque utilisé dans les expressions régulières pour indiquer la fermeture de Kleene. Aussi, les symboles `␣` et `a` servent à représenter symboliquement tous les autres caractères que l'on pourrait retrouver dans un jeu de caractères réaliste, comme ASCII ou ISO-Latin-1.

Quel langage est généré par l'expression régulière donnée?

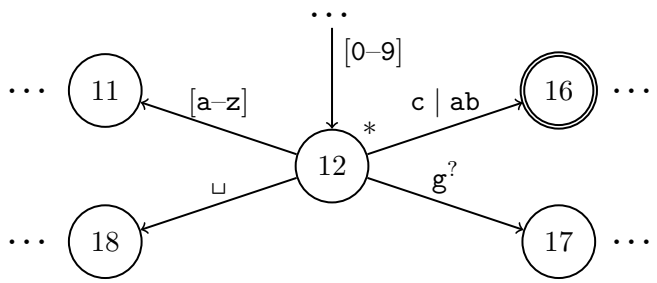
(a) Les commentaires classiques du langage C.

(b) Les constantes chaînes de caractères du langage C.

(c) Les commentaires du langage C où on impose la restriction supplémentaire visant à séparer les astérisques les uns des autres.

(d) Les expressions arithmétiques faites de multiplications, de divisions et de sous-expressions entre parenthèses.

5. (*5 points.*) Considérons le fragment de diagramme d'analyse lexicale ci-bas. Le fragment se concentre sur l'état 12 et son voisinage immédiat. Le fragment comporte plusieurs violations des règles de conception des diagrammes. Parmi les énoncés donnés plus bas, quel est le seul qui décrit un élément *correct* du diagramme?



(a) Le non-déterminisme causé par les choix de symboles offerts par les transitions $12 \to 11$, $12 \to 16$ et $12 \to 17$.

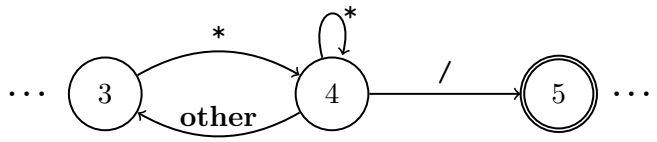
(b) La transition $12 \to 18$ sur un caractère d'espacement.

(c) Le nombre de symboles possiblement consommés sur la transition $12 \to 16$.

(d) Le nombre de symboles possiblement consommés sur la transition $12 \to 17$.

(e) La présence d'une étoile sur l'état 12.

6. (*5 points.*) Considérons le fragment de diagramme d'analyse lexicale suivant. Il s'agit d'un extrait d'un diagramme que l'on retrouve à la page 31 des notes de cours sur l'analyse lexicale.



La traduction de l'état 4 en pseudo-code donnerait la séquence suivante, en laissant quelques trous vides.

```
case 4:
   c := buffer[forward];
   forward ++;
   if (c == '*')
      [          ]
   else if [          ]
     state := 5;
   else
      [          ]
   break;
```

Quel pseudo-code est-ce qu'on devrait retrouver dans les trous laissés vides (en respectant l'ordre)?

(a) `state := 4;` `(c == '/')` `state := 3;`

(b) `state := 5;` `(c == OTHER)` `state := 3;`

(c) `  ` `state := 5;` `state := 3;`

(d) `state := 4;` `(c != '/')` `state := fail();`

7. (*5 points.*) Lequel de ces langages est hors-contexte? Rappelons que la notation w^R produit la chaîne inversée de la chaîne w (i.e. écrite de droite à gauche plutôt que de gauche à droite).

   (a) Le langage $\{\mathtt{a}^i\mathtt{b}^j \mid j = i^2\}$.

   (b) Le langage $\{u\mathtt{c}v\mathtt{c}w \mid u, v, w \in \{\mathtt{a}, \mathtt{b}\}^* \text{ et } |u| < |v| < |w|\}$.

   (c) Le langage

   $$\left\{u\mathtt{c}v^R \mid u, v \in \{\mathtt{a}, \mathtt{b}\}^* \text{ et } u \text{ est lexicographiquement plus petit que } v\right\} .$$

   (d) Le langage des programmes sources en C qui terminent leurs calculs de manière garantie.

   (e) Le langage des expressions arithmétiques faites de constantes entières, d'opérateurs $+$, $-$, $*$, / et de parenthèses qui valent 12.

8. (*5 points.*) Quel est le langage généré par la grammaire suivante?

$$\begin{array}{lcl} S & \rightarrow & A\mathtt{a}A \mid B\mathtt{b}B \\ A & \rightarrow & A\mathtt{a}A \mid A\mathtt{a}A\mathtt{b}A \mid A\mathtt{b}A\mathtt{a}A \mid \epsilon \\ B & \rightarrow & B\mathtt{b}B \mid B\mathtt{a}B\mathtt{b}B \mid B\mathtt{b}B\mathtt{a}B \mid \epsilon \end{array}$$

   (a) Le langage des mots qui contiennent un nombre différent de `a` et de `b`.

   (b) Le langage des mots tirés de $\{\mathtt{a}, \mathtt{b}\}$ qui ont longueur qui est un nombre de Fibonacci.

   (c) Le langage des mots tirés de $\{\mathtt{a}, \mathtt{b}\}$ qui ont une longueur multiple de 3.

   (d) Le langage de tous les mots non-vides tirés de $\{\mathtt{a}, \mathtt{b}\}$.

   (e) Le langage des mots qui contiennent entre une et deux fois plus de `b` que de `a`.

9. (*5 points.*) Considérons la grammaire suivante. Pour une raison quelconque, nous souhaitons montrer que cette grammaire génère (entre autres) toutes les chaînes tirées de $\{\mathtt{a}\}$ dont la longueur est un nombre de Fibonacci. Inspectez la preuve plus bas et ensuite indiquez quelle affirmation est vraie parmi les choix de réponses.

$$\begin{array}{lcl} S & \to & A \mid B \mid C \\ A & \to & BC \mid \epsilon \\ B & \to & CA \mid \mathtt{a} \\ C & \to & AB \end{array}$$

Nous travaillons avec cette définition de la suite de Fibonacci.

$$f_0 = 0 \qquad f_1 = 1 \qquad f_{n+2} = f_n + f_{n+1}, \text{ pour } n \geq 0$$

La stratégie que nous adoptons pour démontrer la propriété voulue consiste à démontrer une propriété un peu plus spécifique.

Théorème. Pour tout nombre naturel j, on a $A \Rightarrow^* \mathtt{a}^{f_{3j}}$, $B \Rightarrow^* \mathtt{a}^{f_{3j+1}}$ et $C \Rightarrow^* \mathtt{a}^{f_{3j+2}}$.

Autrement dit, l'énoncé du théorème affirme que:
— A peut générer les chaînes de longueurs f_0, f_3, f_6, f_9, ...
— B peut générer les chaînes de longueurs f_1, f_4, f_7, f_{10}, ... et
— C peut générer les chaînes de longueurs f_2, f_5, f_8, f_{11}, ...
Il est clair que la propriété que l'on veut originalement démontrer découle de l'énoncé du théorème.

Preuve. Nous procédons par induction sur le rang des nombres de Fibonacci.

Base d'induction. Montrons que l'énoncé du théorème est valide pour les trois premiers nombres de Fibonacci.
— $A \Rightarrow \epsilon = \mathtt{a}^0 = \mathtt{a}^{f_0}$
— $B \Rightarrow \mathtt{a} = \mathtt{a}^1 = \mathtt{a}^{f_1}$
— $C \Rightarrow AB \Rightarrow B \Rightarrow \mathtt{a} = \mathtt{a}^1 = \mathtt{a}^{f_2}$

Hypothèse d'induction. Supposons que l'énoncé du théorème est valide pour les n premiers nombres de Fibonacci, i.e. pour f_0, f_1, ..., f_{n-1}.

Pas d'induction. Montrons que l'énoncé du théorème est valide pour le $(n+1)$ième nombre de Fibonacci, i.e. pour f_n, où $n \geq 3$. Pour ce faire, il faut distinguer les cas où n est un multiple de 3, un multiple de 3 plus 1 ou un multiple de 3 plus 2.

Considérons tout d'abord le cas où n est un multiple de 3, i.e. il existe j tel que $n = 3j$. Dans ce cas-ci, on doit montrer que c'est le non-terminal A qui peut générer $\mathtt{a}^{f_n}$.

$$\begin{array}{lcll} A & \Rightarrow & BC & \\ & \Rightarrow^* & \mathtt{a}^{f_{n-2}}C & // \text{ par H.I. sachant que } n-2 = 3j-2 = 3(j-1)+1 \\ & \Rightarrow^* & \mathtt{a}^{f_{n-2}}\mathtt{a}^{f_{n-1}} & // \text{ par H.I. sachant que } n-1 = 3j-1 = 3(j-1)+2 \\ & = & \mathtt{a}^{f_{n-2}+f_{n-1}} & \\ & = & \mathtt{a}^{f_n} & \end{array}$$

Dans les cas où n est un multiple de 3 plus 1 ou un multiple de 3 plus 2, les raisonnements sont similaires. Il suffit de commencer ceux-ci avec les productions $B \to CA$ et $C \to AB$, respectivement.

(a) On ne peut pas simplement prétendre que les cas “multiple de 3 plus 1” et “multiple de 3 plus 2” sont similaires; on n'indique même pas les bonnes productions à utiliser pour débuter les raisonnements.

(b) Le nombre de cas traités dans la base d'induction aurait pu être réduit à 2 en adaptant le reste de la preuve, sans compromettre la validité du raisonnement.

(c) La preuve doit comporter une erreur quelque part puisque le non-terminal C est capable de générer une chaîne de longueur f_{3j+1}, pour un certain nombre naturel j.
$C \Rightarrow AB \Rightarrow B \Rightarrow \mathtt{a} = \mathtt{a}^1$; or on a $1 = f_1 = f_{3\cdot 0+1}$.

(d) La preuve présentée est invalide; la grammaire devrait comporter une production additionnelle: $C \to \mathtt{a}$.

(e) Le théorème n'est pas valide parce qu'il y a d'autres chaînes qui peuvent être générées par la grammaire; par exemple, $\mathtt{a}^4$ peut être générée et 4 n'est pas un nombre de Fibonacci.
$S \Rightarrow A \Rightarrow BC \Rightarrow \mathtt{a}C \Rightarrow \mathtt{a}AB \Rightarrow \mathtt{a}BCB \Rightarrow \mathtt{aa}CB \Rightarrow^* \mathtt{aaa}B \Rightarrow \mathtt{aaaa}$

10. (*5 points.*) À la page 21 des notes de cours sur l'analyse syntaxique, on présente une façon non-ambiguë de générer les énoncés conditionnels, i.e. des **if-then-else**. Il peut sembler paradoxal de voir une utilisation de *matched_stmt* dans la dernière production puisque celle-ci vise à produire des énoncés conditionnels incomplets. Supposons qu'on propose plutôt la variante suivante de la grammaire, où ce sont des énoncés quelconques qui peuvent être placés dans la branche **then**. Désignons par G la grammaire ambiguë de la page 21, par G' la grammaire non-ambiguë de la page 21 et par G'' la variante présentée ici. G'' n'arrive malheureusement pas à atteindre l'objectif double de générer $L(G)$ tout en étant dénuée d'ambiguïté. Quel est le défaut de G''?

G'':

stmt	→	*matched_stmt*
	|	*open_stmt*
matched_stmt	→	**if** *expr* **then** *matched_stmt* **else** *matched_stmt*
	|	**other**
open_stmt	→	**if** *expr* **then** *stmt*
	|	**if** *expr* **then** *stmt* **else** *open_stmt*

(a) G'' n'est pas capable de générer toutes les chaînes de $L(G)$.

(b) G'' génère certaines chaînes qui ne font pas partie de $L(G)$.

(c) G'' est ambiguë.

(d) G'' n'est en fait plus une grammaire hors-contexte.

11. (*5 points.*) On s'intéresse à générer le langage suivant. À cette fin, on propose diverses grammaires qui génèrent le langage. Une seule n'est pas ambiguë. Laquelle?

$$\left\{ \; \mathtt{a}^i\mathtt{b}^j \;\middle|\; {}^i\!/_2 \leq j \leq 2i \; \right\}$$

(a)
$$\begin{array}{rcl} S & \to & A \mid B \\ A & \to & \mathtt{aa}A\mathtt{b} \mid \mathtt{a}A\mathtt{b} \mid \epsilon \\ B & \to & \mathtt{a}B\mathtt{b} \mid \mathtt{a}B\mathtt{bb} \mid \epsilon \end{array}$$

(b)
$$\begin{array}{rcl} S & \to & \mathtt{aa}A\mathtt{b} \mid M \mid \mathtt{a}B\mathtt{bb} \\ A & \to & \mathtt{aa}A\mathtt{b} \mid \mathtt{a}A\mathtt{b} \mid \epsilon \\ M & \to & \mathtt{a}M\mathtt{b} \mid \epsilon \\ B & \to & \mathtt{a}B\mathtt{b} \mid \mathtt{a}B\mathtt{bb} \mid \epsilon \end{array}$$

(c)
$$S \to \mathtt{aa}S\mathtt{b} \mid \mathtt{a}S\mathtt{b} \mid \mathtt{a}S\mathtt{bb} \mid \epsilon$$

(d)
$$\begin{array}{rcl} S & \to & \mathtt{aa}A\mathtt{b} \mid \mathtt{a}S\mathtt{b} \mid \mathtt{a}B\mathtt{bb} \mid \epsilon \\ A & \to & \mathtt{aa}A\mathtt{b} \mid \epsilon \\ B & \to & \mathtt{a}B\mathtt{bb} \mid \epsilon \end{array}$$

12. (*5 points.*) La grammaire suivante comporte des récursions à gauche.

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

Supposons qu'on obtient la grammaire modifiée suivante après avoir éliminé la récursion à gauche à l'aide de l'algorithme 4.19. Quel est l'ordre sur les non-terminaux qui aurait pu être adopté pour mener à une telle transformation?

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

(a) F, A, E, T
(b) T, F, A, E
(c) T, E, A, F
(d) A, E, T, F
(e) F, T, A, E
(f) F, E, T, A

13. (*5 points.*) Supposons que l'une des quatre grammaires de la question 11 subit la factorisation des préfixes communs à la façon de l'algorithme 4.21. Le résultat est présenté ici. Laquelle des grammaires a subi la transformation?

$$
\begin{array}{rcl}
S & \rightarrow & A \mid B \\
A & \rightarrow & \mathtt{a}A' \mid \epsilon \\
A' & \rightarrow & \mathtt{a}A\mathtt{b} \mid A\mathtt{b} \\
B & \rightarrow & \mathtt{a}B\mathtt{b}B' \mid \epsilon \\
B' & \rightarrow & \epsilon \mid \mathtt{b}
\end{array}
$$

(a) (a)

(b) (b)

(c) (c)

(d) (d)

14. (*5 points.*) Considérons les T-productions de la grammaire **modifiée** de la question 12. On traduit ces productions en une fonction d'analyse syntaxique `T()`. Parmi les choix ci-bas, indiquez lequel fournit correctement les parties manquantes au pseudo-code.

```
void T()
{
  if      (peekToken().type == 'non')
    { readToken('non'); F(); Tprime(); }
  else if (peekToken().type ==  [          ] )
    { readToken('('); E();  [          ] }
  else if (peekToken().type == 'faux')
    { readToken('faux'); Tprime(); }
  else if (peekToken().type == 'vrai')
    { readToken('vrai'); Tprime(); }
  else
    {  [          ] Tprime(); }
}
```

(a) `OTHER` `readToken(')');` `readToken('id');`

(b) `'('` `readToken(')'); Tprime();` `[ ]`

(c) `'('` `readToken(')'); Tprime();` `readToken('id');`

(d) `')'` `readToken(')');` `readToken('id');`

15. (*5 points.*) Considérons la table d'analyse suivante. Quelques cases ont un contenu mystérieux (indiqué par **?**). En regardant la trace d'une analyse faite à l'aide de cette table, identifiez le contenu de chacune des cases mystérieuses.

Non-terminal	Symbole d'entrée					
	int	**float**	[	]	**num**	$
T	$T \to B\,C$	**?**				
B	$B \to$ **int**	**?**				
C			**?**			**?**

Pile	**Entrée**	**Sortie**
T $	**float** [**num**] $	**?**
$B\ C$ $	**float** [**num**] $	**?**
float C $	**float** [**num**] $	(consomme **float**)
C $	[**num**] $	**?**
[**num**] C $	[**num**] $	(consomme [)
num] C $	**num**] $	(consomme **num**)
] C $	] $	(consomme])
C $	$	**?**
$	$	(Succès!)

(a) $T \to B\,C$ $B \to$ **float** $C \to$ [**num**] C $C \to \epsilon$

(b) $T \to B\,C$ $B \to$ **num** $C \to$ [**float**] C $C \to \$$

(c) $B \to T\,C$ $T \to$ **float** $C \to$ [**num**] C $C \to \epsilon$

(d) $T \to C\,B$ $B \to$ **float** $C \to C$] **num** [$C \to \epsilon$

16. (*5 points.*) Quel est l'ensemble FIRST(A)?

$$
\begin{aligned}
E &\to \texttt{b}\,E \mid A\,\texttt{c} \mid B\,B\,\texttt{a} \\
A &\to \texttt{c}\,E \mid B\,\texttt{e}\,A \\
B &\to A\,\texttt{e}\,B \mid \texttt{d}\,E \mid \epsilon
\end{aligned}
$$

(a) $\{\texttt{c}, \texttt{d}, \texttt{e}, \epsilon\}$.

(b) $\{\texttt{c}, \texttt{d}, \texttt{e}\}$.

(c) $\{\texttt{a}, \texttt{c}, \texttt{d}, \texttt{e}\}$.

(d) $\{\texttt{c}, \$\}$.

17. (*5 points.*) Dans la grammaire de la question 16, quel est l'ensemble FOLLOW(B)?

(a) $\{\texttt{a}, \texttt{c}, \texttt{d}, \texttt{e}\}$.

(b) $\{\texttt{a}, \texttt{e}, \$\}$.

(c) $\{\texttt{a}, \texttt{e}\}$.

(d) $\{\texttt{c}, \texttt{d}, \texttt{e}, \epsilon\}$.

18. (*5 points.*) Quel devrait être le contenu des cases mystérieuses dans la table d'analyse ci-bas? La grammaire ainsi que les ensembles FIRST et FOLLOW correspondants sont les suivants.

$$
\begin{array}{lcl}
T & \to & F\,T' \\
T' & \to & *\,F\,T' \mid /\,F\,T' \mid \epsilon \\
F & \to & \textbf{sqrt}\,F \mid A \\
A & \to & \textbf{num} \mid \textbf{id}
\end{array}
$$

Non-term.	FIRST	FOLLOW
T	$\{\textbf{num}, \textbf{id}, \textbf{sqrt}\}$	$\{\$\}$
T'	$\{*, /, \epsilon\}$	$\{\$\}$
F	$\{\textbf{num}, \textbf{id}, \textbf{sqrt}\}$	$\{*, /, \$\}$
A	$\{\textbf{num}, \textbf{id}\}$	$\{*, /, \$\}$

Non-terminal	Symbole d'entrée					
	$*$	$/$	**sqrt**	**num**	**id**	\$
T			$T \to F\,T'$	$T \to F\,T'$	$T \to F\,T'$	**?**
T'	$T' \to *\,F\,T'$	$T' \to /\,F\,T'$				**?**
F			**?**	$F \to A$	$F \to A$	
A			**?**	$A \to \textbf{num}$	**?**	

(a) $T \to \epsilon$ | | $F \to A$ | | $A \to \textbf{id}$

(b) $T' \to F\,T'$ | $T' \to \epsilon$ | | $F \to \textbf{sqrt}\,F$ | $A \to \textbf{id}$

(c) | $T' \to \epsilon$ | $F \to \textbf{sqrt}\,F$ | | $A \to \textbf{id}$

(d) | $T' \to *\,F\,T'$
$T' \to /\,F\,T'$ | $F \to \textbf{sqrt}\,F$ | | |

(e) $T' \to \epsilon$ | $T' \to \epsilon$ | $F \to \textbf{sqrt}\,F$ | $A \to \textbf{sqrt}\,A$ | $A \to \textbf{id}$

19. (*5 points.*) Que peut-on dire à propos de la validité des deux affirmations suivantes?

Affirmation 1. Il existe des grammaires qui sont ambiguës et LL(1).

Affirmation 2. La plus petite solution à la contrainte

$$S \;\supseteq\; \Big(S \cdot \{\texttt{a}\} - \{\texttt{aa}\}\Big) \cup \{\epsilon\}$$

rend l'ensemble S égal à $\{\epsilon, \texttt{a}\}$.

(a) L'affirmation 1 est vraie et l'affirmation 2 est vraie.

(b) L'affirmation 1 est vraie et l'affirmation 2 est fausse.

(c) L'affirmation 1 est fausse et l'affirmation 2 est vraie.

(d) L'affirmation 1 est fausse et l'affirmation 2 est fausse.

20. (*5 points.*) Que fait le programme C suivant quand on l'exécute? Je vous suggère de copier-coller ce programme dans un fichier avant de le compiler et de l'exécuter.

```
#include <stdio.h>

char *b = "\\";
char *g = "\"";
char *n = "\n";
char *p = "%";
char *ff = "    %s%s%s,%s";
char *f[] =
  {
    "#include <stdio.h>%s%schar *b = %s%s%s%s;%schar *g = %s%s%s%s;",
    "%schar *n = %s%sn%s;%schar *p = %s%s%s;%schar *ff = %s%s%s;%sch",
    "ar *f[] =%s  {%s",
    "    %s%s%s  };%s%svoid main(void)%s{%s  int i;%s%s  printf(f[0]",
    ", n, n, g, b, b, g, n, g, b, g, g);%s  printf(f[1], n, g, b, g,",
    " n, g, p, g, n, g, ff, g, n);%s  printf(f[2], n, n);%s  for (i ",
    "= 0 ; i < 11 ; i++)%s    printf(ff, g, f[i], g, n);%s  printf(f",
    "[3], g, g, n, n, n, n, n, n, n);%s  printf(f[4], n);%s  printf(",
    "f[5], n, n);%s  printf(f[6], n, n);%s  printf(f[7], n, n);%s  p",
    "rintf(f[8], n, n, n);%s  printf(f[9], n, n);%s  printf(f[10], n",
    ", n);%s}%s",
    ""
  };

void main(void)
{
  int i;

  printf(f[0], n, n, g, b, b, g, n, g, b, g, g);
  printf(f[1], n, g, b, g, n, g, p, g, n, g, ff, g, n);
  printf(f[2], n, n);
  for (i = 0 ; i < 11 ; i++)
    printf(ff, g, f[i], g, n);
  printf(f[3], g, g, n, n, n, n, n, n, n);
  printf(f[4], n);
  printf(f[5], n, n);
  printf(f[6], n, n);
  printf(f[7], n, n);
  printf(f[8], n, n, n);
  printf(f[9], n, n);
  printf(f[10], n, n);
}
```

(a) Il affiche le tradionnel texte "`Hello world!`".

(b) Il affiche seulement le contenu du tableau `f`.

(c) Il affiche le source d'un virus informatique.

(d) Il affiche son propre source; i.e. c'est un programme auto-répliquant ou *quine*.

Remise des travaux

Vous devez remettre votre travail en complétant le questionnaire du TP#1 sur **monPortail**. Notez que le questionnaire ne répète pas les questions; il ne présente que les divers choix.