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

Danny Dubé
Automne 2018
Version : 7 octobre

Travail pratique #1
Analyses lexicale et syntaxique

Questions

1. (*6 points.*) Le motif suivant décrit la structure lexicale des identificateurs du langage C. (Ignorez la question des mots-clés.)

   `[A-Za-z_][A-Za-z_0-9]*`

   (a) Est-ce que tout préfixe non-vide d'un identificateur est aussi un identificateur ? Justifiez.

   (b) Est-ce que tout suffixe non-vide d'un identificateur est aussi un identificateur ? Justifiez.

2. (*15 points.*) Exprimez chacun des langages suivants à l'aide du formalisme demandé. Donnez une explication dès que votre réponse est le moindrement complexe.

   (a) Donnez une expression régulière ou une définition régulière qui génère le langage défini sur $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ suivant.

   $$\{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \mathtt{aaaa} \text{ n'est pas une sous-chaîne de } w\}$$

   (b) Donnez une grammaire hors-contexte qui génère le langage défini sur $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ suivant. Considérez que les "ou" sont inclusifs.

   $$\{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i \leq j \text{ ou } i \leq k \text{ ou } j \leq k\}$$

   (c) Donnez une expression régulière qui génère le langage défini sur $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ suivant. Considérez que les "ou" sont inclusifs.

   $$\{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i \leq j \text{ ou } j \leq k \text{ ou } k \leq i\}$$

   (d) Donnez une expression régulière qui génère le langage défini sur $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ suivant.

   $$\{v \mid w \in \{\mathtt{abcd}\}^* \text{ et } v \text{ est une sous-séquence de } w\}$$

   (e) Donnez une grammaire hors-contexte qui génère le langage défini sur $\{\mathtt{a}, \mathtt{b}\}$ suivant. Inspirez-vous de l'exercice 4.2.3(c).

   $$\{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ comporte de 2 à 3 fois plus de } \mathtt{a} \text{ que de } \mathtt{b}, \text{ inclusivement}\}$$

3. (*8 points.*) Dessinez des diagrammes de transitions qui permettent de reconnaître les lexèmes qui correspondent au motif des constantes entières d'un langage hypothétique. En fait, votre réponse peut comporter un diagramme ou plus. Ces constantes peuvent être écrites dans l'une des bases 2, 8, 10 et 16. Une constante doit être écrite uniquement à l'aide des chiffres valides dans la base choisie, soit les chiffres 0 et 1 en binaire, les chiffres 0 à 7 en octal, les chiffres 0 à 9 en décimal et les chiffres 0 à F en hexadécimal, où A à F sont les chiffres au-delà de 9. Un suffixe `#`*base* suit les chiffres de la constante et il sert à spécifier, en décimal, la base utilisée. Dans le cas de la base 10, le suffixe est optionnel. L'analyse faite par vos diagrammes doit être vorace.

$$\texttt{[01]}^{+}\texttt{\#2} \quad | \quad \texttt{[0–7]}^{+}\texttt{\#8} \quad | \quad \texttt{[0–9]}^{+}\texttt{(\#10)}^{?} \quad | \quad \texttt{[0–9A–F]}^{+}\texttt{\#16}$$

4. (*10 points.*) Dans les notes de cours du chapitre 3, à la page 34, on retrouve deux diagrammes de transitions. En imitant le pseudo-code de la page 41, montrez comment transformer les diagrammes en fragments de code. Vous n'avez pas à vous soucier de la fonction `fail()`.

5. (*8 points.*) La grammaire hors-contexte suivante permet de générer les énoncés **if-then-else** sans ambiguïté :

$$\begin{array}{rcl}
\textit{stmt} & \rightarrow & \textit{matched_stmt} \\
 & | & \textit{open_stmt} \\
\textit{matched_stmt} & \rightarrow & \textbf{if } \textit{expr} \textbf{ then } \textit{matched_stmt} \textbf{ else } \textit{matched_stmt} \\
 & | & \textbf{other} \\
\textit{open_stmt} & \rightarrow & \textbf{if } \textit{expr} \textbf{ then } \textit{stmt} \\
 & | & \textbf{if } \textit{expr} \textbf{ then } \textit{matched_stmt} \textbf{ else } \textit{open_stmt}
\end{array}$$

(Pour compléter cette grammaire, supposons qu'on ajoute une production pour les expressions : *expr* $\rightarrow$ **e**.) Donnez la dérivation à gauche d'abord et l'arbre de dérivation pour la chaîne suivante.

if e then other else if e then if e then other else other

6. (*6 points.*) Revenons sur le langage de l'exercice 2e, plus haut. Démontrez que la grammaire suivante **ne** génère **pas** le langage.

$$\begin{array}{rcl}
S & \rightarrow & \epsilon \\
 & | & \texttt{aab}S \quad | \quad \texttt{aa}S\texttt{b} \quad | \quad \texttt{a}S\texttt{ab} \quad | \quad S\texttt{aab} \\
 & | & \texttt{aba}S \quad | \quad \texttt{ab}S\texttt{a} \quad | \quad \texttt{a}S\texttt{ba} \quad | \quad S\texttt{aba} \\
 & | & \texttt{baa}S \quad | \quad \texttt{ba}S\texttt{a} \quad | \quad \texttt{b}S\texttt{aa} \quad | \quad S\texttt{baa} \\
 & | & \texttt{aaab}S \quad | \quad \texttt{aaa}S\texttt{b} \quad | \quad \texttt{aa}S\texttt{ab} \quad | \quad \texttt{a}S\texttt{aab} \quad | \quad S\texttt{aaab} \\
 & | & \texttt{aaba}S \quad | \quad \texttt{aab}S\texttt{a} \quad | \quad \texttt{aa}S\texttt{ba} \quad | \quad \texttt{a}S\texttt{aba} \quad | \quad S\texttt{aaba} \\
 & | & \texttt{abaa}S \quad | \quad \texttt{aba}S\texttt{a} \quad | \quad \texttt{ab}S\texttt{aa} \quad | \quad \texttt{a}S\texttt{baa} \quad | \quad S\texttt{abaa} \\
 & | & \texttt{baaa}S \quad | \quad \texttt{baa}S\texttt{a} \quad | \quad \texttt{ba}S\texttt{aa} \quad | \quad \texttt{b}S\texttt{aaa} \quad | \quad S\texttt{baaa}
\end{array}$$

7. (*5 points.*) La grammaire suivante génère les mots sur $\{\mathtt{a}\}$ dont les longueurs sont des multiples de 3 ou de 5. Cette grammaire est-elle ambiguë ? Justifiez.

$$\begin{aligned} A &\rightarrow \epsilon \mid \mathtt{aaa}B \mid \mathtt{aaaaa}C \\ B &\rightarrow \epsilon \mid \mathtt{aaa}B \\ C &\rightarrow \epsilon \mid \mathtt{aaaaa}C \end{aligned}$$

8. (*6 points.*) La grammaire hors-contexte ci-bas génère le langage suivant. Cette grammaire est ambiguë. Proposez une autre grammaire qui n'est pas ambiguë. Justifiez.

$$\{\mathtt{a}^i\mathtt{b}^j \mid i \leq 2j \text{ et } j \leq 2i\}$$

$$S \rightarrow \mathtt{aa}S\mathtt{b} \mid \mathtt{a}S\mathtt{b} \mid \mathtt{a}S\mathtt{bb} \mid \epsilon$$

9. (*10 points.*) Éliminez la récursion à gauche dans la grammaire suivante (pour des expressions fictives) en ordonnant les non-terminaux ainsi : E, T, F. Décrivez en détail les étapes effectuées par l'algorithme (boucles sur i et j).

$$\begin{aligned} E &\rightarrow E \oplus T \mid F \dagger E \mid T \\ T &\rightarrow T\surd \mid E \ominus F \mid F \\ F &\rightarrow \mathbf{cte} \mid \mathbf{var} \mid (\,E\,) \end{aligned}$$

10. (*6 points.*) Considérons une grammaire qui génère une variété d'énoncés. En imitant le pseudo-code présenté aux pages 39–42 des notes de cours du chapitre 4 (révision hiver 2014), donnez l'implantation des fonctions `stmt`, `tail` et `list` qui effectuent l'analyse syntaxique descendante prédictive des énoncés générés par *stmt*. Par souci de simplicité, nous prétendons que les expressions sont représentées par un simple jeton : **expr**. Supposez que les fonctions '*peekToken*' et '*readToken*' sont fournies. De plus, prenez pour acquis que les ensembles PREDICT des *list*-productions vous sont fournis :
PREDICT($list \rightarrow stmt\ list$) = $\{\mathbf{while}, \mathbf{if}, \mathbf{id}\}$ et
PREDICT($list \rightarrow \epsilon$) = $\{\mathbf{else}, \mathbf{end}\}$.

$$\begin{aligned} stmt &\rightarrow \mathbf{while}\ \mathbf{expr}\ \mathbf{do}\ list\ \mathbf{end} \\ &\mid \mathbf{if}\ \mathbf{expr}\ \mathbf{then}\ list\ tail \\ &\mid \mathbf{id}\ :=\ \mathbf{expr}\,; \\ tail &\rightarrow \mathbf{else}\ list\ \mathbf{end} \\ &\mid \mathbf{end} \\ list &\rightarrow stmt\ list \\ &\mid \epsilon \end{aligned}$$

11. (*5 points.*) À la façon de la page 48 des notes de cours du chapitre 4 (révision hiver 2014), donnez la trace des opérations effectuées par l'analyseur prédictif sans récursion sur la chaîne d'entrée `abbaaabcbaaabba`.

Non-terminal	Symbole d'entrée			
	`a`	`b`	`c`	$
S	$S \to \texttt{a}\,S\,\texttt{a}$	$S \to \texttt{b}\,S\,\texttt{b}$	$S \to \texttt{c}$	

12. (*10 points.*) Revenons à la grammaire de la question 10.

    (a) Calculez les ensembles FIRST des trois non-terminaux.

    (b) Calculez les ensembles FOLLOW des trois non-terminaux.

13. (*5 points.*) Revenons à la grammaire de la question 5. Construisez la table d'analyse pour cette grammaire. Les ensembles FIRST et FOLLOW des non-terminaux sont précalculés pour vous.

Non-term.	FIRST	FOLLOW
stmt	{**if**, **other**}	{$}
matched_stmt	{**if**, **other**}	{**else**, $}
open_stmt	{**if**}	{$}

Remise des travaux

Les modalités de la remise sont inscrites dans le plan de cours.