Contrôle continu - Compilation (L3 Info)
Durée : 2h - documents interdits

Conventions

— Axiome : symbole non terminal à gauche de la première production de la grammaire
— Symboles non terminaux : lettres *MAJUSCULES ITALIQUES*
— Symboles terminaux : lettres `minuscules true-type` ou caractères spéciaux simples

Exercice 1 - Vrai ou faux *(4 pts)*

Pour chacune des affirmations suivantes dites si elle est vraie ou fausse en justifiant brièvement et de façon convaincante votre réponse (seules les réponses accompagnées d'une justification seront prises en compte).

1. Les grammaires LL ne sont pas ambiguës
2. Pour tout langage hors contexte, il existe un automate à pile qui permet de le reconnaître
3. Les grammaires hors-contexte récursives à gauche sont ambiguës
4. Tout langage hors-contexte peut être reconnu en temps linéaire

Exercice 2 - Écriture de grammaire *(6 pts)*

Les arbres de dérivation produits par l'analyseur syntaxique développé en TP sont représentés à l'aide d'un balisage XML. L'arbre XML correspondant à la dérivation suivante : $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$ par exemple, sera représenté par le mot suivant : `<S> a <S> a <S> </S> b </S> b </S>`

1. Ecrire une grammaire G qui permet de générer les arbres XML. Le vocabulaire terminal de G est : `CHAINE, DEBUT_BO, DEBUT_BF, CHEVRON_FERMANT`.
   `CHAINE` correspond à une séquence de lettres majuscules ou minuscules, `DEBUT_BO` correspond à `<`, `DEBUT_BF` correpond à la séquence `</` et `CHEVRON_FERMANT` correspond à `>`.
   Votre grammaire doit être la plus simple possible, indiquez à quoi correspond chaque symbole non terminal de cette dernière.
2. Ecrire l'arbre XML de la dérivation suivante : $E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + b * E \Rightarrow a + b * -E \Rightarrow a + b * -c$
3. Ecrire l'arbre de dérivation que G associe à l'arbre XML de la question précédente.
4. Est ce que l'arbre XML suivant `<A> <B> </A> </B>` est un arbre bien formé ? Est ce que la grammaire G peut le générer ? si c'est le cas, est il possible de modifier G de manière à empêcher de générer de tels mots ?

Exercice 3 - Analyse $LL(1)$ *(6 pts)*
Soit la grammaire G_1 :

1	A	$\rightarrow$	$B\ A$
2	A	$\rightarrow$	ε
3	B	$\rightarrow$	`a` C
4	B	$\rightarrow$	`d`
5	C	$\rightarrow$	$D\ C$
6	C	$\rightarrow$	ε
7	D	$\rightarrow$	`b`
8	D	$\rightarrow$	`c`

1. Décrivez en une phrase ou deux ou en notation mathématique le langage engendré par G_1
2. Calculez les ensembles PREMIER et SUIVANT de G_1
3. Construisez la table LL(1) de G_1
4. Simulez l'analyse $LL(1)$ du mot `abcdac`. Les configurations sont représentées par le triplet (w, α, y), où w est la partie de la bande de lecture qui commence au caractère sous la tête de lecture (ce qui reste à analyser), α est la pile et y est la séquence de symboles de sortie (numéros des productions appliquées).

Exercice 4 - Récursivité à gauche *(4 pts)*
Soit la grammaire G_2 :

$$\begin{array}{lcl} A & \rightarrow & A\ \texttt{a} \mid BC \\ B & \rightarrow & AC \mid CC \\ C & \rightarrow & BA \mid AC \mid \texttt{b} \end{array}$$

1. Ecrire une grammaire G_2' non récursive à gauche telle que $L(G_2) = L(G_2')$, en respectant l'ordre suivant sur les symboles non terminaux : $A < B < C$.