TD 7 : Grammaires LL(k)

Exercice 1 (Motifs Objective Caml)**.** Voici un extrait de la syntaxe des motifs d'Objective Caml :

$$\begin{aligned}
\langle pat \rangle &\to \texttt{VN} \\
&\mid \ \texttt{_} \\
&\mid \ \langle pat \rangle :: \langle pat \rangle \\
&\mid \ \langle pat \rangle \ \texttt{as VN} \\
&\mid \ [\ \langle patl \rangle\ ] \\
\langle patl \rangle &\to \langle pat \rangle \\
&\mid \ \langle patl \rangle\ ;\ \langle pat \rangle
\end{aligned}$$

Cette grammaire permet de générer des motifs tels que

```
[a; _; b] :: t as l
```

1. Donner une grammaire SLL(1) équivalente à ce fragment.
2. Préciser les ensembles First_1 et Follow_1 calculés.
3. Écrire un analyseur récursif descendant dans votre langage favori pour votre grammaire.

Exercice 2 (Calcul efficace de First_1 et Follow_1)**.** On appelle *expression relationnelle* un terme décrit par la syntaxe abstraite

$$e ::= R \mid e^* \mid e^{-1} \mid e \cdot e \mid e \cup e$$

où R est une relation "atomique". Une expression relationnelle définit une relation $R(e)$. La *taille* $|e|$ d'une expression relationnelle e est la somme des tailles des relations atomiques qui la composent. Pour une expression relationnelle e de taille finie, on peut calculer une image par sa relation $R(e)$ en temps $O(|e|)$. On souhaite calculer les ensembles First_1 et Follow_1 d'une grammaire G à l'aide d'expressions relationnelles de taille $O(|G|)$.

1. Proposer une expression relationnelle first sur $N \times \Sigma$ de taille $O(|G|)$ telle que
$$\text{First}_1(A) = R(\mathsf{first})(A) \cup \{\varepsilon \mid A \Rightarrow^* \varepsilon\} \ .$$
2. Proposer une expression relationnelle follow sur $N \times \Sigma$ telle que
$$\text{Follow}_1(A) = R(\mathsf{follow})(A) \cup \{\varepsilon \mid \exists \delta \in (N \uplus \Sigma)^*,\ S \Rightarrow^* \delta A\} \ .$$
Au moins dans un premier temps, on pourra se contenter d'une expression de taille $O(|G|^2)$.

Exercice 3 (Grammaires LL(0))**.**

1. Montrer que si l'automate expansion/vérification associé à une grammaire est déterministe, alors la grammaire est LL(0).
2. Montrer qu'une grammaire LL(0) engendre au plus un mot.

Exercice 4 (Langages LL(k))**.**

1. Montrer qu'une grammaire est LL(1) si et seulement si elle est SLL(1).
2. Montrer que si une grammaire est LL(k), alors il existe une grammaire SLL(k) équivalente.
3. Appliquer cette construction à la grammaire

$$S \to aAab \mid bAb$$
$$A \to cAB \mid \varepsilon \mid a$$
$$B \to \varepsilon$$

Il existe une hiérarchie stricte des langages LL(k) pour chaque valeur de k. Il existe aussi des langages déterministes qui ne sont pas LL(k).

Exercice 5 (Récursivité gauche, GNF, langages simples)**.** Un symbole non terminal $A \in N$ est *récursif gauche* s'il existe une dérivation $A \Rightarrow^+ A\alpha$ avec α dans $(N \uplus \Sigma)^*$. Une grammaire réduite est récursive gauche s'il existe au moins un non terminal A recursif gauche.

1. Montrer que si une grammaire réduite est récursive gauche, alors elle n'est pas LL.
2. Montrer que si une grammaire est simple, alors elle est SLL(1).
3. Montrer que si une grammaire sans ε-production est LL(k), alors on peut construire une grammaire LL(k) équivalente sous forme normale de Greibach.
4. Montrer que si une grammaire sans ε-production est LL(1), alors elle engendre un langage simple.
5. Donner une grammaire SLL(1) qui n'engendre pas un langage simple.