Feuille d'exercices n°14 - Grammaires non contextuelles - partie 2

Notions abordées

- grammaire décrivant des listes
- grammaire décrivant des formules sous FNC
- régularité des langages non contextuels
- grammaire propre

Exercice 1 : Listes OCaml

On considère l'alphabet $\Sigma = \{\texttt{[}, \texttt{]}, \texttt{;}, \texttt{true}, \texttt{false}\}$. Donner une grammaire non contextuelle dont le langage est l'ensemble des listes OCaml de booléens♣.

Exercice 2 : Forme normale conjonctive

Soit un ensemble fini de variables propositionnelles $\mathcal{Q} = \{\texttt{p}, \texttt{q}, \ldots\}$. Soit $\mathcal{O}$ l'ensemble des symboles $\{\texttt{\&}, \texttt{|}, \texttt{-}\}$♡. Soit l'alphabet $\Sigma = \mathcal{Q} \cup \mathcal{O}$. Soit finalement le langage FNC sur l'alphabet Σ des mots qui décrivent une formule de la logique propositionnelle sous forme normale conjonctive.
Par exemple FNC contient :

- ε qui représente la conjonction vide ;
- `p` qui représente la conjonction réduite à une disjonction, elle-même réduite au seul littéral `p` ;
- `p|-q` qui représente la conjonction réduite à une disjonction : $(p \vee \neg q)$;
- `p|q&r|-s&-p` qui représente la forme normale conjonctive : $(p \vee q) \wedge (r \vee \neg s) \wedge \neg p$

Q. 1 Donner une grammaire non contextuelle non ambiguë $\mathcal{G}$ de langage FNC.

On se munit des types OCaml suivants, permettant la représentation des formes normales conjonctives en OCaml.

```
type var = char                 (* une variable *)
type lit = var * bool           (* un littéral *)
type cls = lit list             (* une clause *)
type fnc = cls list             (* une FNC *)
```

Q. 2 Grâce aux types précédemment définis, donner les expressions OCaml représentant les éléments suivants.

- la variable p
- le littéral p
- le littéral $\neg q$
- la clause $p \vee q$
- la formule $p \vee q$
- $p \wedge q$
- $p \wedge (q \vee \neg p)$

♣. On ne décrit pas l'ensemble des expressions OCaml de type `bool list` car on limite les expressions booléennes autorisées à `true` et `false` seulement

♡. Jouant respectivement les rôles de $\{\wedge, \vee, \neg\}$

Q. 3 Donner 5 fonctions OCaml mutuellement récursives :

```
- parse_f (s: string) (start: int) (len: int) : fnc
- parse_c (s: string) (start: int) (len: int) : fnc
- parse_d (s: string) (start: int) (len: int) : cls
- parse_l (s: string) (start: int) (len: int) : lit
- parse_v (s: string) (start: int) (len: int) : char
```

prenant en arguments une chaîne de caractères `s`, un indice de début `start` et une longueur `len` et retournant la forme normale conjonctive (resp. la FNC non vide, la clause disjonctive, le littéral, la variable) représentée par la chaîne de caractères `String.sub s i len`. On déclenchera une erreur dans l'éventualité où la chaîne de caractères n'est pas de la forme attendue.

Exemples :

> `(parse_v "p|q&r|-s&-p" 2 1)` s'évalue en `'q'`
> `(parse_l "p|q&r|-s&-p" 9 2)` s'évalue en `('p', false)`
> `(parse_l "p|q&r|-s&-p" 4 4)` s'évalue en `[('r', true); ('s', false)]`

Exercice 3 : Un lemme d'itération

Dans cet exercice on s'intéresse à un équivalent du lemme de l'étoile pour les langages non contextuels. Ce résultat donne des contraintes que doivent vérifier les langages engendrés par des grammaires non contextuelles, ainsi on pourra établir que certains langages ne peuvent être engendrés par des grammaires non contextuelles en montrant qu'ils ne vérifient pas lesdites propriétés.
Soit une grammaire $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$. On suppose dans la suite que $P \neq \emptyset$. Soit donc $p = \max(\{1\} \cup \{|w| \mid V \to w \in P\})$ la longueur du plus long mot apparaissant à droite d'une règle, ou 1 si tous les mots à droite des règles sont le mot vide.

Q. 1 Donner et prouver une majoration sur la longueur des mots de $(\Sigma \cup \mathcal{V})^*$ admettant un arbre de dérivation de hauteur h.

Q. 2 En déduire qu'il existe un entier N tel que si w est un mot de $\mathcal{L}(\mathcal{G})$ de longueur $\geqslant N$, alors tout arbre de dérivation de w admet une branche contenant au moins deux fois le même symbole non terminal.

Q. 3 En déduire qu'il existe un entier N ♣ tel que pour tout mot $w \in \mathcal{G}$ tel que $|w| \geqslant N$, il existe des mots u, x, c, y et v de $\Sigma^\star$ tels que :

$i)$ $w = uxcyv$
$ii)$ $|xy| \geqslant 1$
$iii)$ $|xcy| \leqslant N$
$iv)$ $\forall n \in \mathbb{N}, ux^n cy^n v \in \mathcal{L}(\mathcal{G})$

Q. 4 En déduire que le langage $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ n'est pas engendré par une grammaire non contextuelle.

Q. 5 En déduire que l'ensemble des langages non contextuels n'est stable ni par intersection, ni par complémentaire.

♣. évidemment cet entier dépend de la grammaire fixée en début d'exercice

Exercice 4 : Grammaires propres

Grammaires propres. $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$ est dite **propre** dès lors que P ne contient aucune règle de la forme :

1. $V \to \varepsilon$ avec $V \in \mathcal{V}$
2. $V \to V'$ avec $(V, V') \in \mathcal{V}^2$

Q. 1 Proposer un algorithme prenant en argument une grammaire $\mathcal{G}$ et calculant une grammaire reconnaissant le langage $\mathcal{L}(\mathcal{G}) \setminus \{\varepsilon\}$ et ne contenant aucune règle de la forme 1. ci-dessus.

Q. 2 Proposer un algorithme prenant en argument une grammaire $\mathcal{G}$ ne contenant pas de règles de production de la forme 1. ci-dessus et la transformant en une grammaire ne contenant aucune règle de la forme 2. ci-dessus (on s'efforcera de ne pas produire une grammaire contenant des règles de la forme 1.). Conclure.

Q. 3 Démontrer que dans une grammaire contextuelle propre $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$, si $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow w_3 \ldots \Rightarrow w_n$ alors $|w_n|_{\mathcal{V}} + 2|w_n|_{\Sigma} \geqslant 1 + n$ [♣].

Q. 4 En déduire que l'ensemble des langages des grammaires non contextuelles est décidable.

Exercice 5 : Les langages réguliers sont non contextuels

Dans cet exercice, on se propose de donner deux preuves alternatives du résultat de cours : tout langage régulier est le langage d'une grammaire non contextuelle. Ces preuves reposent sur deux définitions qu'on peut donner d'un langage régulier, à savoir que c'est le langage d'un automate, ou que c'est le langage d'une expression régulière.

1. Avec des automates

Soit L un langage régulier et $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ un automate [♡] le reconnaissant. On considère alors la famille d'automates $(\mathcal{A}_q)_{q \in Q} = (\Sigma, Q, \{q\}, F, \delta)$. On se munit par ailleurs de la famille de symboles non terminaux $(X_q)_{q \in Q}$, qu'on note être finie.

Q. 1 Proposer un ensemble de règles de production P tel qu'en notant $\overset{\star}{\Rightarrow}$ la relation de dérivation induite par P, on a $\forall q \in Q, \mathcal{L}(\mathcal{A}_q) = \{u \in \Sigma^\star, X_q \overset{\star}{\Rightarrow} u\}$. Démontrer cette propriété.

Q. 2 Conclure.

2. Avec des expressions régulières

Soit L un langage régulier sur un alphabet fini Σ et e une expression régulière de langage L. On définit inductivement les **sous-expressions régulières** de e comme suit.

$$\mathsf{ssexp}(e) = \{e\} \cup \begin{cases} \mathsf{ssexp}(e_1) \cup \mathsf{ssexp}(e_2) & \text{si } e = e_1|e_2 \text{ ou } e = e_1 \cdot e_2 \\ \mathsf{ssexp}(e_1) & \text{si } e = (e_1)^\star \\ \emptyset & \text{sinon, } \textit{i.e.} \text{ si } e = \varepsilon \text{ ou } e \in \Sigma \text{ ou } e = \emptyset \end{cases}$$

On se munit alors de la famille de symboles non terminaux $(X_r)_{r \in \mathsf{ssexp}(e)}$ (qu'on remarque être finie).

♣. Lorsque w est un mot sur un alphabet et A un sous-ensemble de cet alphabet, $|w|_A$ désigne le nombre de lettres de w appartenant à A.

♡. sans ε-transitions

Q. 3 Proposer un ensemble de règles de production P tel qu'en notant $\stackrel{\star}{\Rightarrow}$ la relation de dérivation induite par P, on a $\forall r \in \mathsf{ssexp}(e), \mathscr{L}(r) = \{u \in \Sigma^\star, X_r \stackrel{\star}{\Rightarrow} u\}$. Démontrer cette propriété.

Q. 4 Conclure.