Feuille d'exercices n°12 - Grammaires non contextuelles

Notions abordées

- exemple de grammaires non contextuelles
- suite de dérivations, arbre de dérivation
- langage d'une grammaire non contextuelle, preuve par induction
- exemple de langages non contextuels : langage de Dyck, mots de Łukasiewicz
- grammaire décrivant des listes, décrivant des formules sous FNC
- régularité des langages non contextuels
- grammaire propre

Exercice 1 : Exemples de dérivations

Soit $\Sigma = \{a, b\}$, $\mathcal{V} = \{S, D, T, X\}$, et $\mathcal{G}$ la grammaire non contextuelle de symbole initial S, et contenant les règles de production ci-contre♣.

$$\begin{array}{l} S \to XSX \mid D \\ D \to aTb \mid bTa \\ T \to XTX \mid X \mid \varepsilon \\ X \to a \mid b \end{array}$$

Q. 1 Parmi les dérivations suivantes, lesquelles sont vraies ?

(a) $T \Rightarrow aba$ (b) $T \overset{\star}{\Rightarrow} aba$ (c) $T \Rightarrow T$ (d) $T \overset{\star}{\Rightarrow} T$

Q. 2 Parmi les dérivations suivantes, lesquelles sont vraies ?

(a) $XXX \overset{\star}{\Rightarrow} aba$ (b) $X \overset{\star}{\Rightarrow} aba$ (c) $T \overset{\star}{\Rightarrow} XX$ (d) $T \overset{\star}{\Rightarrow} XXX$ (e) $D \overset{\star}{\Rightarrow} \varepsilon$

Q. 3 Donner 3 mots de $\mathcal{L}(\mathcal{G})$. Justifier leur appartenance à $\mathcal{L}(\mathcal{G})$ par une dérivation.

Q. 4 Donner 3 mots de $\Sigma^\star$ qui ne sont pas dans $\mathcal{L}(\mathcal{G})$.

Q. 5 Donner une description en français de $\mathcal{L}(\mathcal{G})$. En première lecture de cet énoncé, on ne demande pas de preuve du résultat avancé.

Exercice 2 : Arbres de dérivations

Soit $\Sigma = \{a, b, c, \ldots, z, +, (,)\}$, $\mathcal{V} = \{S, P, F, T\}$, et la grammaire $\mathcal{G}$ de symbole initial S et de règles de production ci-contre.

$$\begin{array}{l} S \to S + P \mid P \\ P \to PF \mid F \\ F \to (S) \mid T \\ T \to a \mid b \mid c \mid \ldots \mid z \end{array}$$

Q. 1 Donner un arbre de dérivation dont la production est $ab + bc$.

Q. 2 Donner un arbre de dérivation de $b + (a + b)c + a(bc)$.

Q. 3 Que représente cette grammaire, *i.e.* que réprésentent les mots engendrés par $\mathcal{G}$?

♣. Lorsque plusieurs règles de production ont le même membre gauche, on peut les représenter factorisées en séparant les différents membres droits par $|$. Par exemple, on note $A \to w|w'$ pour les deux règles $A \to w$ et $A \to w'$.

Exercice 3 : Construction de grammaires

Q. 1 Soit $\Sigma = \{a, b\}$. Donner une grammaire non contextuelle $\mathcal{G}_i$ de langage L_i ci-dessous.
- $L_0 = \{w \in \Sigma^\star \mid |w| \geqslant 3\}$;
- $L_1 = \{w \in \Sigma^\star \mid |w|_a \geqslant 3\}$;
- $L_2 = \{w \in \Sigma^+ \mid w$ commence et finit par la même lettre$\}$;
- $L_3 = \emptyset$;

Q. 2 Soit $\Sigma = \{a, b\}$. Donner une grammaire non contextuelle $\mathcal{G}_i$ de langage L_i ci-dessous[♣].
- $L_4 = \{w \in \Sigma^\star \mid |w| \equiv 1[2]\}$;
- $L_5 = \{w \in \Sigma^\star \mid |w| \equiv 1[2]$ et sa lettre du milieu est $a\}$;
- $L_6 = \{w \in \Sigma^\star \mid w$ est un palindrome$\}$;
- $L_7 = L_6^C$;

Q. 3 Donner une grammaire non contextuelle $\mathcal{G}_i$ de langage L_i ci-dessous.
- $L_8 = \{a^n b^n \mid n \in \mathbb{N}\}$;
- $L_9 = \{a^n b^m \mid (n, m) \in \mathbb{N}^2,\ m \geqslant n\}$;
- $L_{10} = \{a^n b^m \mid (n, m) \in \mathbb{N}^2,\ m \geqslant 2n\}$;

Q. 4 Soit $\Sigma = \{a, b, \#\}$. Donner une grammaire non contextuelle $\mathcal{G}_{13}$ de langage L_{13} ci-dessous.
- $L_{10} = \{w\#x \mid (w, x) \in (\{a, b\}^\star)^2, \overline{w}^R$ est un sous-mot de x où $\overline{w}^R$ désigne le miroir de $w\}$.

Exercice 4 : Langage de Dyck

Le langage des mots de Dyck est le langage des mots sur $\Sigma = \{(,)\}$ bien parenthésés, c'est-à-dire le plus petit langage $L \subseteq \Sigma^\star$ tel que :
- $\varepsilon \in L$;
- $\forall u \in L$, $(u) \in L$;
- $\forall (u, v) \in L \times L$, $u \cdot v \in L$.

Q. 1 Donner une grammaire non contextuelle $\mathcal{G}$ reconnaissant le langage de Dyck.

Q. 2 Montrer que les mots u de $\mathcal{L}(\mathcal{G})$ sont tels que :
- $|u|_(= |u|_)$
- pour tout préfixe v de u, $|v|_(\geqslant |v|_)$

Q. 3 Montrer que les mots $u \in \Sigma^\star$ vérifiant ces deux conditions sont bien parenthésés.

Q. 4 En déduire une fonction OCaml testant si un mot est dans $\mathcal{L}(\mathcal{G})$.

Q. 5 Démontrer que ce langage n'est pas régulier.

Exercice 5 : Raisonner par induction sur une grammaire

Soit $\Sigma = \{a, b\}$, $\mathcal{V} = \{S\}$ et la grammaire $\mathcal{G}$ contenant les règles de production :

$$S \to aS \mid Sb \mid a \mid b \mid \varepsilon$$

Q. 1 Montrer que ba n'est sous-mot d'aucun mot de $\mathcal{L}(\mathcal{G})$.

Q. 2 Donner $\mathcal{L}(\mathcal{G})$.

Q. 3 Donner une grammaire reconnaissant $\mathcal{L}(\mathcal{G})$ et assurant que chaque mot est obtenu par une unique suite de dérivations immédiates depuis le symbole initial.

♣. Lorsque la définition d'un langage fait intervenir le complémentaire, celui-ci s'entend par rapport à $\Sigma^\star$.

Exercice 6 : Ambiguïté

Soit $\Sigma = \{a, b\}$, $\mathcal{V} = \{S\}$, et la grammaire $\mathcal{G}$ de symbole initial S et de règles de production :

$$S \to aS \mid aSbS \mid \varepsilon$$

Q. 1 Montrer que $\mathcal{G}$ est ambiguë.

Q. 2 Montrer que $\mathcal{L}(G) = \{v \in \Sigma^\star \mid \text{pour tout } u \text{ préfixe de } v, |u|_a \geqslant |u|_b\}$.

Q. 3 Donner une grammaire non ambiguë reconnaissant $\mathcal{L}(G)$.

Exercice 7 : 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 8 : 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 *)
```

♣. 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. 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 $p \vee q$
- $p \wedge q$
- $p \wedge (q \vee \neg p)$

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 9 : 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 10 : 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 11 : 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).

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

Q. 4 Conclure.

♣. 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

Exercice 12 : Mots de Łukasiewicz [Centrale 2008]

On considère l'ensemble de deux symboles $\Sigma = \{\square, \circ\}$, Σ contient donc uniquement le symbole "carré" : $\square$ et le symbole "rond" : $\circ$. On s'intéresse dans cet exercice à un sous-ensemble de $\Sigma^\star$ nommé **mots de Łukasiewicz**. On rappelle qu'étant donné un mot $w = w_1 w_2 \dots w_n \in \Sigma^\star$, la notation $|w|$ désigne la longueur de w, n ici.

On considère la fonction $\mathsf{va} : \Sigma \to \{-1, 1\}$ telle que $\mathsf{va}(\square) = -1$ et $\mathsf{va}(\circ) = 1$. À un mot $w = w_1 w_2 \dots w_n \in \Sigma^\star$ et à un entier $i \in [\![0, n]\!]$ on associe la valeur : $\overline{w}^i = \sum_{k=1}^{i} \mathsf{va}(w_i)$, en prenant comme convention que $\sum_{k=1}^{0} \mathsf{va}(w_i) = 0$.

Exemples : $\overline{\circ\circ\square\square\square}^4 = \mathsf{va}(\circ) + \mathsf{va}(\circ) + \mathsf{va}(\square) + \mathsf{va}(\square) = 1 + 1 - 1 - 1 = 0$
$\overline{\varepsilon}^0 = 0$

On définit alors l'ensemble des **mots de Łukasiewicz** comme étant l'ensemble des mots $w \in \Sigma^\star$ tels que $\forall i \in [\![0, |w| - 1]\!], \overline{w}^i \geqslant 0$ et $\overline{w}^{|w|} = -1$

Par exemple, $w = \circ\circ\square\square\square \in \mathcal{L}$. En effet les valeurs des $\overline{w}^i$ sont les suivantes.

i	0	1	2	3	4	5
$\overline{w}^i$	0	1	2	1	0	-1

Q. 1 Donner l'ensemble des éléments de $\mathcal{L}$ de longueur 1, 2, 3.

Q. 2 Donner (en justifiant le résultat) l'ensemble des éléments de $\mathcal{L}$ de longueur paire.

On suppose définis en OCaml les types et fonctions ci-contre. Le type `sym` est utilisé pour représenter l'ensemble Σ : le symbole $\circ$ est représenté par le constructeur `R` (pour "rond") et $\square$ est représenté par le constructeur `C` (pour "carré"). Le type `word` est utilisé pour représenter les mots de $\Sigma^\star$. La fonction `va` implémente la fonction va définie ci-dessus.

```
type sym = | R | C
type word = sym list

let va (s: sym) : int =
  match s with
  | R ->  1
  | C -> -1
```

Q. 3 L'ensemble des mots de Łukasiewicz est-il régulier ?

Q. 4 Définir une fonction `est_luka : word -> bool` permettant de tester si un mot est de Łukasiewicz, *i.e.* permettant de tester, pour $w \in \Sigma^\star$, si $w \in \mathcal{L}$. On proposera une implémentation ayant une complexité pire cas en $\mathcal{O}(n)$.

Q. 5 On appelle préfixe strict d'un mot $w \in \Sigma^\star$, un mot $u \in \Sigma^\star$ tel qu'il existe un mot $v \in \Sigma^\star \setminus \{\varepsilon\}$ tel que $w = u \cdot v$. Montrer qu'un préfixe strict d'un mot de $\mathcal{L}$ n'est pas un mot de $\mathcal{L}$.

Q. 6 Montrer que si $u \in \mathcal{L}$ et $v \in \mathcal{L}$ alors $\circ \cdot u \cdot v \in \mathcal{L}$.

Q. 7 Montrer que si $w \in \mathcal{L}$ et $|w| \neq 1$ alors il existe un unique couple $(u, v) \in \mathcal{L} \times \mathcal{L}$ tel que $w = \circ \cdot u \cdot v$.

Q. 8 Montrer qu'il existe une grammaire contextuelle non ambiguë reconnaissant l'ensemble des mots de Łukasiewicz.

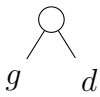
Q. 9 Donner une fonction `decompose: word -> word * word` prenant en argument un mot de $\mathcal{L}$ de taille différente de 1 (et de 0 car $\varepsilon \notin \mathcal{L}$) et calculant la décomposition de la **Q. 7**.

Exemples : `decompose [R; R; C; C; C] = ([R; C; C], [C])`
`decompose [R; R; C; C; R; C; C] = ([R; C; C], [R; C; C])`

On définit maintenant l'ensemble des arbres binaires $\mathbb{B}$ **sans étiquette** par les deux règles de construction :

```
type btree =
  | N of btree * btree          (* Nœud interne *)
  | E                           (* Arbre vide *)
```

Autrement énoncé :

- $\square \in \mathbb{B}$
- Si $g \in \mathbb{B}$ et $d \in \mathbb{B}$ alors  $\in \mathbb{B}$

Q. 10 Donner une bijection naturelle de $\mathbb{B}$ dans $\mathcal{L}$, induite par la **Q. 8**

Q. 11 Définir en OCaml deux fonctions `to_word : btree -> word` et `from_word: word -> btree` telles que $\forall w \in \mathcal{L}$, `(to_word (from_word w))` $= w$ et $\forall b \in \mathbb{B}$, `(from_word (to_word b))` $= b$.

Exemples :
```
from_word [R; R; C; C; R; C; C] = N (N (E, E), N (E, E))
to_word (N (N (E, E), N (E, E))) = [R; R; C; C; R; C; C]
```

Q. 12 Démontrer que $\forall w \in \mathcal{L}, |w| \geqslant 3 \Rightarrow \exists (u, v) \in \Sigma^\star, w = u \cdot \circ\square\square \cdot v$.