# TME4 : Grammaires propres

## (définitions, exemples, programmation en Python)

**– version 2.0 (Janvier 2022) –**

Mathieu.Jaume@lip6.fr

## 1 Grammaires hors-contexte propres

Le code présenté dans cette section se trouve dans le fichier `proper_grammar.py`.

La grammaire hors-contexte $G = (V, \Sigma, R, S)$ est *propre* si $R$ ne contient ni $\varepsilon$-*production* (i.e. production de la forme $X \to \varepsilon$) ni *production unitaire* (i.e. production de la forme $X \to Y$ avec $X, Y \in V$).

### 1.1 Elimination des $\varepsilon$-productions

**Symboles annulables** Un symbole $X \in V$ est *annulable* s'il existe une dérivation $X \mapsto^* \varepsilon$. L'ensemble $\mathbb{C}(G) \subseteq V$ des symboles annulables de $G$ est obtenu en construisant la suite croissante et bornée (car $V$ est fini) :

$$\begin{array}{l} C_0(G) = \{X \mid X \to \varepsilon \in R\} \\ C_{n+1}(G) = C_n(G) \cup \{X \mid X \to u \in R \text{ et } u \in (C_n(G))^\star\} \end{array} \quad \mathbb{C}(G) = \bigcup_{n \geq 0} C_n(G)$$

Il s'agit donc de la construction d'un point fixe.

**Exemple** Considérons la grammaire hors-contexte $G_3$ définie dans le TME3. La suite $(C_n(G_3))_{n\in\mathbb{N}}$ est construite comme suit :

$$C_0(G_3) = \{A_6\} \quad C_1(G_3) = \{A_3, A_6\} \quad C_2(G_3) = \{S, A_3, A_6\} = C_3(G_3) = \mathbb{C}(G_3)$$

**Implantation** Le calcul de l'ensemble $\mathbb{C}(G)$ est effectué à partir de l'ensemble $C_0(G)$ en appliquant une fonction `next_canc` permettant de construire $C_{n+1}(G)$ à partir de $C_n(G)$ jusqu'à obtenir un point fixe, c-à-d un ensemble $C_{n+1}(G)$ égal à $C_n(G)$.

**Calcul de $C_0(G)$** Ce calcul est effectué à partir de la liste `r` des productions de $G$.

calcul de $C_0(G)$

```
def canc0(r):
    # r : liste de productions
    def is_empty_list(l):
        return len(l)==0
    return [s for s,ls in r if exists_such_that(ls,is_empty_list)]
```

exemple : $C_0(G_3)$

```
>>> canc0(g3_r)
['A6']
```

**Calcul de $C_{n+1}(G)$ à partir de $C_n(G)$** La fonction `next_canc` effectue ce calcul. Outre l'ensemble `prev` à partir duquel est effectué ce calcul, cette fonction utilise la liste `r` des productions de $G$, et la fonction d'égalité `eqnt` sur $V$.

**à compléter** : calcul de $C_{n+1}(G)$ à partir de $C_n(G)$

```
def next_canc(r,eqnt,prev):
    # r : liste de productions
    # eqnt : egalite sur les non terminaux
    # prev : liste de non terminaux de depart
```

exemple : $C_1(G_3)$

```
>>> next_canc(g3_r,g3_eqnt,canc0(g3_r))
['A3', 'A6']
```

**Calcul de $\mathbb{C}(G)$** Ce calcul correspond à la construction itérative d'un point fixe (et utilise donc la fonction `fixpoint_from` définie dans le fichier `ensembles.py`) à partir des fonctions `canc0` et `next_canc`. Ici encore, cette construction utilise la fonction `make_eq_set` (définie dans le fichier `ensembles.py`) qui permet de définir une fonction d'égalité sur des ensembles dépendant de l'égalité sur les éléments de ces ensembles.

calcul de $\mathbb{C}(G)$

```
def canc(r,eqnt):
    # r : liste de productions
    # eqnt : egalite sur les non terminaux
    def _next_canc(e):
        return next_canc(r,eqnt,e)
    return fixpoint_from(make_eq_set(eqnt),_next_canc,canc0(r))
```

exemple : $\mathbb{C}(G_3)$

```
>>> canc(g3_r,g3_eqnt)
['S', 'A3', 'A6']
```

**Elimination des $\varepsilon$-productions** Etant donnée une grammaire hors-contexte $G = (V, \Sigma, R, S)$, on peut construire une grammaire hors-contexte $\overline{\mathcal{E}}(G) = (V, \Sigma, R', S)$ sans $\varepsilon$-production telle que $\mathcal{L}(G) \setminus \{\varepsilon\} = \mathcal{L}(\overline{\mathcal{E}}(G))$ où :

$$R' = \left\{ \begin{array}{l} X \to \alpha_1 \cdots \alpha_{n+1} \mid n \geq 0 \\ \text{et } \exists X_1, \cdots, X_n \in \mathbb{C}(G)\ X \to \alpha_1 X_1 \alpha_2 \cdots \alpha_n X_n \alpha_{n+1} \in R \end{array} \right\} \setminus \{X \to \varepsilon \mid X \in V\}$$

**Exemple** Considérons à nouveau la grammaire $G_3$ définie dans le TME3. A partir de l'ensemble $\mathbb{C}(G_3) = \{S, A_3, A_6\}$, on définit la grammaire $G_4 = \overline{\mathcal{E}}(G_3) = (V_4, \Sigma, R_4, S)$ où :

$$R_4 = \{S \to A_3\ , \quad A_3 \to aA_3A_9 \mid aA_9 \mid A_6\ , \quad A_6 \to bA_6 \mid b\ , \quad A_9 \to c\} \qquad (1)$$

**Calcul de $\overline{\mathcal{E}}(G)$** Ce calcul s'effectue en engendrant récursivement toutes les productions possibles lorsque l'on supprime des symboles non-terminaux annulables des productions de $G$. Cette construction utilise la fonction `make_eq_prod` (définie dans le TME3) qui permet de construire une fonction d'égalité sur les parties droites des productions de $G$.

élimination des $\varepsilon$-productions

```
def remove_eps_prod(g):
    # g : ghc
    nt,t,r,si,eqnt = g
    canc_g = canc(r,eqnt)
    def make_new_prod(l):
        if len(l)==0:
            return [[]]
        else:
            res_rec = make_new_prod(l[1:])
            add_first = [[l[0]]+lrec for lrec in res_rec]
            if is_in(eqnt,l[0],canc_g):
                acc = add_first + res_rec
            else:
                acc = add_first
            return acc
    def make_new_prods(ls):
        res = []
        for l in ls:
            new_l = [x for x in make_new_prod(l) if len(x)>0]
            res = union(make_eq_prod(nt,eqnt),new_l,res)
        return res
    new_r = [(s,make_new_prods(ls)) for s,ls in r]
    return (nt,t,new_r,si,eqnt)
```

exemple : $G_4 = \overline{\mathcal{E}}(G_3)$

```
>>> g4_g = remove_eps_prod(g3_g)
>>> g4_nt,g4_t,g4_r,g4_si,g4_eqnt = g4_g
>>> g4_r
[('S', [['A3']]), ('A3', [['A6'], ['a', 'A9'], ['a', 'A3', 'A9']]),
 ('A6', [['b'], ['b', 'A6']]), ('A9', [['c']])]
```

## 1.2 Elimination des productions unitaires

**Paires unitaires** L'ensemble $\mathbb{U}(G) \subseteq V \times V$ des *paires unitaires* de $G$ est obtenu en contruisant la suite croissante et bornée (car $R$ est fini) :

$$\begin{array}{l} U_0(G) = \{(X,Y) \mid X \to Y \in R\} \\ U_{n+1}(G) = U_n \cup \{(X,Z) \mid (X,Y) \in U_n(G) \text{ et } Y \to Z \in R\} \end{array} \qquad \mathbb{U}(G) = \bigcup_{n \geq 0} U_n(G)$$

Il s'agit donc de la construction d'un point fixe.

**Exemple** Considérons la grammaire hors-contexte $G_4$ définie en (**??**). La suite $(U_n(G_4))_{n\in\mathbb{N}}$ est construite comme suit :

$$U_0(G_4) = \{(S,A_3),(A_3,A_6)\} \quad U_1(G_4) = \{(S,A_3),(A_3,A_6),(S,A_6)\} = U_2(G_4) = \mathbb{U}(G_4)$$

**Implantation** Le calcul de l'ensemble $\mathbb{U}(G)$ est effectué à partir de l'ensemble $U_0(G)$ en appliquant une fonction `next_unit_pair` permettant de construire $U_{n+1}(G)$ à partir de $U_n(G)$ jusqu'à obtenir un point fixe, c-à-d un ensemble $U_{n+1}(G)$ égal à $U_n(G)$. On utilise ici une fonction `make_eq_pair_nt` qui permet de construire une fonction d'égalité sur les paires de symboles de $V$.

égalité sur les paires de symboles non terminaux

```
def make_eq_pair_nt(eqnt):
    # eqnt :  egalite sur les non terminaux
```

```
    def _eq_pair_nt(p1,p2):
        x1,y1 = p1
        x2,y2 = p2
        return eqnt(x1,x2) and eqnt(y1,y2)
    return _eq_pair_nt
```

**Calcul de $U_0(G)$** Ce calcul est effectué à partir de la liste `nt` représentant $V$, de la liste `r` des productions de $G$ et de la fonction d'égalité `eqnt` sur $V$.

**à compléter** : calcul de $U_0(G)$

```
def unit_pair0(nt,r,eqnt):
    # nt : symboles non terminaux
    # r : liste de productions
    # eqnt : egalite sur les non terminaux
```

exemple : $U_0(G_4)$

```
>>> unit_pair0(g4_nt,g4_r,g4_eqnt)
[('A3', 'A6'), ('S', 'A3')]
```

**Calcul de $U_{n+1}(G)$ à partir de $U_n(G)$** La fonction `next_unit_pair` effectue ce calcul. Outre l'ensemble `prev` à partir duquel est effectué ce calcul, cette fonction utilise la liste `nt` représentant $V$, la liste `r` des productions de $G$ et la fonction d'égalité `eqnt` sur $V$.

**à compléter** : calcul de $U_{n+1}(G)$ à partir de $U_n(G)$

```
def next_unit_pair(nt,r,eqnt,prev):
    # nt : symboles non terminaux
    # r : liste de productions
    # eqnt : egalite sur les non terminaux
    # prev : liste de non terminaux de depart
```

exemple : $U_1(G_4)$

```
>>> next_unit_pair(g4_nt,g4_r,g4_eqnt,unit_pair0(g4_nt,g4_r,g4_eqnt))
[('S', 'A6'), ('A3', 'A6'), ('S', 'A3')]
```

**Calcul de $\mathbb{U}(G)$** Ce calcul correspond à la construction itérative d'un point fixe (et utilise donc la fonction `fixpoint_from` définie dans le fichier `ensembles.py`) à partir des fonctions `unit_pair0` et `next_unit_pair`.

calcul de $\mathbb{U}(G)$

```
def unit_pair(nt,r,eqnt):
    # nt : symboles non terminaux
    # r : liste de productions
    # eqnt : egalite sur les non terminaux
    def _next_unit_pair(e):
        return next_unit_pair(nt,r,eqnt,e)
    return fixpoint_from(make_eq_set(make_eq_pair_nt(eqnt)),
                         _next_unit_pair,unit_pair0(nt,r,eqnt))
```

exemple : $\mathbb{U}(G_4)$

```
>>> unit_pair(g4_nt,g4_r,g4_eqnt)
[('S', 'A6'), ('A3', 'A6'), ('S', 'A3')]
```

**Elimination des productions unitaires** Etant donnée une grammaire $G = (V, \Sigma, R, S)$, on peut construire la grammaire sans production unitaire $\overline{\mathcal{U}}(G) = (V, \Sigma, R', S)$ telle que $\mathcal{L}(G) = \mathcal{L}(\overline{\mathcal{U}}(G))$ où :

$$\begin{array}{rl} R' = & (R \setminus \{X \to Y \mid X \to Y \in R\}) \\ \cup & \{X \to u \mid u \notin V \text{ et } \exists Y \in V\ (X, Y) \in \mathbb{U}(G) \text{ et } Y \to u \in R\} \end{array}$$

**Exemple** Considérons à nouveau la grammaire $G_4$ définie en (**??**). A partir de l'ensemble $\mathbb{U}(G_4) = \{(S, A_3), (A_3, A_6), (S, A_6)\}$, on définit $G_5 = \overline{\mathcal{U}}(G_4) = (V_4, \Sigma, R_5, S)$ où :

$$R_5 = \left\{ \begin{array}{lcl} S \to aA_3A_9 \mid aA_9 \mid bA_6 \mid b & , & A_3 \to aA_3A_9 \mid aA_9 \mid bA_6 \mid b \\ A_6 \to bA_6 \mid b & , & A_9 \to c \end{array} \right\} \tag{2}$$

**Calcul de $\overline{\mathcal{U}}(G)$** Ce calcul s'effectue en utilisant la fonction `prods_s` (définie page **??**) permettant d'obtenir la liste des productions d'un symbole non-terminal donné et la fonction `add_prod` (définie dans le TME3) permettant d'ajouter une production pour un symbole non-terminal donné.

**à compléter** : élimination des productions unitaires

```
def remove_unit_pairs(g):
    # g : ghc
```

exemple : $G_5 = \overline{\mathcal{U}}(G_4)$

```
>>> g5_g = remove_unit_pairs(g4_g)
>>> g5_nt,g5_t,g5_r,g5_si,g5_eqnt = g5_g
>>> g5_r
[('A3', [['b', 'A6'], ['b'], ['a', 'A9'], ['a', 'A3', 'A9']]),
 ('A6', [['b'], ['b', 'A6']]),
 ('A9', [['c']]),
 ('S', [['a', 'A3', 'A9'], ['a', 'A9'], ['b', 'A6'], ['b']])]
```

## 1.3 Construction d'une grammaire propre

La construction d'une grammaire propre $G'$ à partir d'une grammaire (réduite) $G$ telle que $\mathcal{L}(G') = \mathcal{L}(G) \setminus \{\varepsilon\}$ s'obtient simplement en éliminant les $\varepsilon$-productions et les productions unitaires :

$$G' = \overline{\mathcal{U}}(\overline{\mathcal{E}}(G))$$

Par exemple, la grammaire $G_5$ définie en (**??**) correspond à la grammaire propre construite à partir de la grammaire $G_1$ définie dans le TME3.

construction d'une grammaire propre

```
def make_gp(g):
    # g : ghc
    return remove_unit_pairs(remove_eps_prod((g))
```