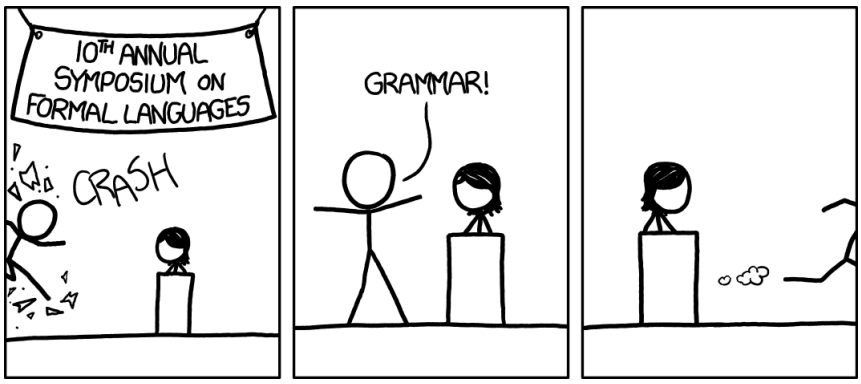
MPI* Info

# TD Grammaires




Olivier Caffier

1. Expliquer comment on peut se ramener à cela sans perte de généralité.
**Corrigé :**

Il suffit juste de créer une nouvelle grammaire avec les mêmes règles et un nouveau symbole initial qui ne peut que se dériver en l'ancien symbole initial.

## 1 Grammaire $\mathcal{G}_2$

On définit $\mathcal{G}_2 = (\mathcal{V}_2, \Sigma_2, S_2, R_2)$ avec $\mathcal{V}_2 = \{S_2, E, T\}, \Sigma_2 = \{a, +, *\}$ et $R_2$ :

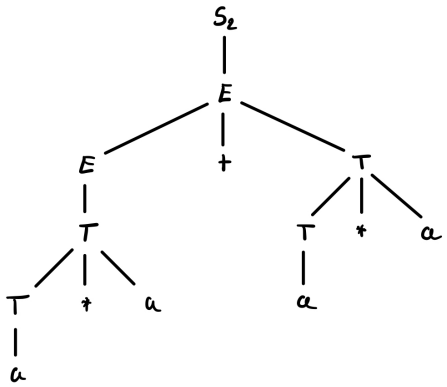
$$\begin{aligned} S_2 &\to E \\ E &\to E + T \mid T \\ T &\to T * a \mid a \end{aligned}$$

2. Est-ce que le mot $a * a + a * a$ est généré par $\mathcal{G}_2$ ? Si oui, donner un arbre de dérivation pour ce mot et une dérivation gauche pour ce mot.
**Corrigé :**

On a

$$\begin{aligned} S_2 &\to E \\ &\to E + T \\ &\to T + T \\ &\to T * a + T \\ &\to a * a + T \\ &\to a * a + T * a \\ &\to a * a + a * a \end{aligned}$$

D'où l'arbre de dérivation suivant :

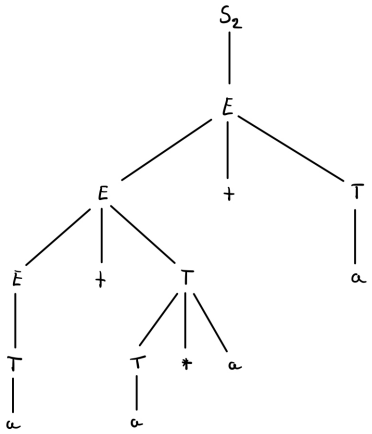



3. Est-ce que le mot $a + a * a + a$ est généré par $\mathcal{G}_2$ ? Si oui, donner un arbre de dérivation pour ce mot et une dérivation gauche pour ce mot.
**Corrigé :**

On a

$$\begin{aligned} S_2 &\to E \\ &\to E + T \\ &\to E + a \\ &\to E + T + a \\ &\to E + T * a + a \\ &\to E + a * a + a \\ &\to T + a * a + a \\ &\to a + a * a + a \end{aligned}$$

On peut obtenir l'arbre suivant :




4. On considère la grammaire $\mathcal{G}_2'$ ayant les mêmes caractéristiques que $\mathcal{G}_2$ mais de symbole initial $T$. Donner (sans justification) une expression régulière décrivant le langage engendré par $\mathcal{G}_2'$.
**Corrigé :**

On a

$$\boxed{L(\mathcal{G}_2') = a(* \ a)^*}$$

5. On considère la grammaire $\mathcal{G}_2''$ ayant les mêmes caractéristiques que $\mathcal{G}_2$ mais de symbole initial $E$. Même question avec $\mathcal{G}_2''$.
**Corrigé :**

On a

$$\boxed{L(\mathcal{G}_2'') = a(* \ a)^* \left(+ \ a(* \ a)^*\right)^*}$$

6. En déduire une description du langage engendré par $\mathcal{G}_2$ et justifier que cette grammaire n'est pas ambiguë. Montrer, à l'aide d'un schéma, la forme d'un arbre de dérivation de cette grammaire.
**Corrigé :**

D'une part,

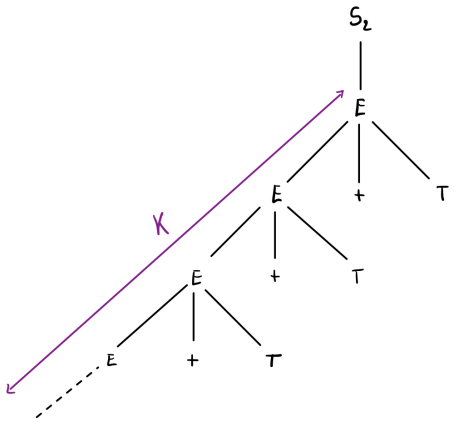
$$\boxed{L(\mathcal{G}_2) = L(\mathcal{G}_2'')}$$

Il s'agit donc du langage composé de $a$ avec un $+$ ou un $*$ entre eux.

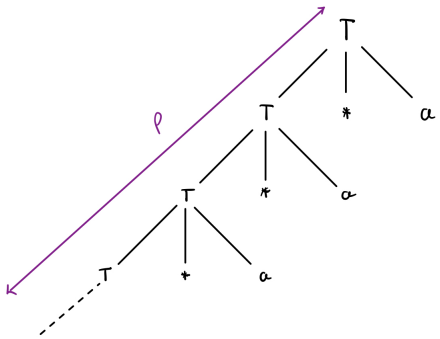
D'autre part, soit $m \in L(\mathcal{G}_2)$. Alors $m$ est de la forme :

$$a * a * \cdots * a \underbrace{+}_{\text{1ère occurence de "+" dans } m} a * \cdots * a \underbrace{+}_{\text{2ème}} \cdots$$

Notons $k \in \mathbb{N}$ le nombre d'occurence de "+" dans $m$. Alors l'arbre de dérivation gauche est de la forme suivante :



Entre deux signes "+", on a $\underbrace{a * a * \cdots * a}_{\text{longueur } l}$. On a alors un sous-arbre de la forme :




Les valeurs $k$ et $l$ étant entièrement déterminées par $m$ et la forme d'arbre ci-dessus étant la seule représentation possible au vu de $\mathcal{G}_2$, on en déduit le caractère non-ambiguë.

7. Écrire une fonction `bool genere_2(char* m)` qui prend en entrée une chaîne de caractère quelconque et renvoie `true` si et seulement si celle-ci correspond à un mot généré par $\mathcal{G}_2$. On attend une complexité linéaire en la longueur de $m$.
**Corrigé :**

```
bool genere_2(char* m){
  int ind = 0;
  while (m[ind + 1] != '\0'){
    if (m[ind] == 'a'){
      if (m[ind+1] == 'a'){
        // caractere illegal
        return false;
      }
      ind +=1;
    }
    else{
      if (m[ind+1] != 'a'){
        // caractere illegal
        return false;
      }
      ind +=1;
    }
  }
  return m[ind] == 'a';
}
```

## 2 Grammaire $\mathcal{G}_1$ et réductions

Considérons la grammaire $\mathcal{G}_1 = (\mathcal{V}_1, \Sigma_1, S, R_1)$ avec $\mathcal{V}_1 = \{S, R, T\}, \Sigma_1 = \{a, b\}$ et $R_1$ :

$$\begin{aligned} S &\to R \mid T \\ R &\to aRb \mid ab \\ T &\to aTbb \mid abb \end{aligned}$$

8. Montrer proprement que le langage engendré par $\mathcal{G}_1$ est inclus dans $\{a^n b^n \mid n \in \mathbb{N}^*\} \cup \{a^n b^{2n} \mid n \in \mathbb{N}^*\}$
**Corrigé :**

— Si $n \in \mathbb{N}^*$, notons $H_n$ : "après $n$ dérivations de $R$ on obtient le mot $a^n b^n$, après $n$ dérivations de $T$ on obtient $a^n b^{2n}$".
— $n = 1$ : immédiat

$H_1$ vraie

— $H_n \Rightarrow H_{n+1}$ :
On a $R \Rightarrow^{n+1} aub$ avec $R \Rightarrow^n u$. Par H.R, $u = a^n b^n$. D'où le résultat voulu.
De même pour $T$.

$H_{n+1}$ vraie

Ainsi, en utilisant les règles de dérivations de $\mathcal{G}_1$, on est bien dans $\{a^n b^n \mid n \in \mathbb{N}^*\} \cup \{a^n b^{2n} \mid n \in \mathbb{N}^*\}$.

D'où l'inclusion recherchée.

9. Démontrer l'inclusion réciproque afin de caractériser le langage engendré par $\mathcal{G}_1$.
**Corrigé :**

Soit $m \in \{a^n b^n \mid n \in \mathbb{N}^*\} \cup \{a^n b^{2n} \mid n \in \mathbb{N}^*\}$.

➢ Si $m \in \{a^n b^n \mid n \in \mathbb{N}^*\}$ : alors $\exists N \in \mathbb{N}^*$ tq. $m = a^N b^N$. Or, d'après la question précédente, $N$ dérivations du symbole $R$ permet d'obtenir ce mot.

D'où $m \in L(\mathcal{G}_1)$.

➢ Si $m \in \{a^n b^{2n} \mid n \in \mathbb{N}^*\}$ : de même en dérivant $N$ fois $T$.

D'où $m \in L(\mathcal{G}_1)$.

D'où l'inclusion réciproque.

D'où

$$\boxed{L(\mathcal{G}_1) = \{a^n b^n \mid n \in \mathbb{N}^*\} \cup \{a^n b^{2n} \mid n \in \mathbb{N}^*\}}$$

10. Donner une dérivation gauche depuis le symbole initial pour le mot $aaabbb$.
**Corrigé :**

On a

$$\begin{aligned} S &\to R \\ &\to aRb \\ &\to aaRbb \\ &\to aaabbb \end{aligned}$$

11. Donner une dérivation droite depuis le symbole initial pour le mot $aabbbb$.
**Corrigé :**

On a

$$\begin{aligned} S &\to T \\ &\to aTbb \\ &\to aabbbb \end{aligned}$$

12. Écrire une fonction `bool genere_1(char* m)` qui prend en entrée une chaîne de caractères quelconque et renvoie `true` si et seulement si celle-ci correspond à un mot généré par $\mathcal{G}_1$. On attend une complexité linéaire en la longueur de $m$.
**Corrigé :**

```
bool genere_1(char* m){
  int count_a = 0;
  int count_b = 0;
  int ind = 0;
  while (m[ind + 1] != '\0'){
    if (m[ind] == 'b'){
      count_b +=1;
      if (m[ind+1] == 'a'){
        return false;
      }
      ind +=1;
    }
    else{
      count_a +=1;
      ind+=1;
    }
  }
  if (m[ind]=='b'){
    count_b +=1;
    return (count_a == count_b) || (2 * count_a == count_b);
  }
  else{
    return false;
  }
}
```

On définit maintenant la notion de réduction comme l'opération contraire à la dérivation. On dit que pour $\mathcal{G} = (\mathcal{V}, \Sigma, S, R)$, $u \in (\mathcal{V} \cup \Sigma)^*$ se réduit en $v \in (\mathcal{V} \cup \Sigma)^*$ ssi $v$ se dérive en $u$. On notera cette réduction $u \hookrightarrow^* v$. Une réduction directe se notera $u \hookrightarrow v$ et correspondra à $v \Rightarrow u$.
Une réduction de $u$ est une réduction de $u$ jusqu'au symbole initial.
On dit qu'une réduction de $u$ est gauche ssi, à chaque étape, on réduit le facteur le plus à gauche possible.

13. Donner une réduction gauche pour $aabbbb$ dans $\mathcal{G}_1$.
**Corrigé :**

On a

$$aabbbb \hookrightarrow^* aTbb$$

14. Montrer qu'une réduction gauche correspond à une dérivation droite pour toute grammaire.
**Corrigé :**

Soit $m \hookrightarrow u_1 \hookrightarrow \cdots \hookrightarrow u_k = S$ une réduction gauche.
On considère la dérivation : $s \Rightarrow u_{k-1} \Rightarrow \cdots \Rightarrow u_2 \to m$
Supposons que cette dérivation ne soit pas droite, alors $\exists i \in [\![1, k]\!]$ tel que $\begin{cases} u_i &= v_1 V_1 v_2 V_2 v_3 \\ u_{i-1} &= v_1 h v_2 V_2 v_3 \end{cases}$.

avec $v_1, v_2, v_3, h \in (\mathcal{V} \cup \Sigma)^*$ et $V_1, V_2 \in \mathcal{V}$.
Autrement dit, on dérive le non terminal $V_1$ avant $V_2$.
On a alors

$$u_{i-1} = v_1 h v_2 V_2 v_3 \hookrightarrow v_1 V_1 v_2 V_2 v_3 = u_i$$

Mais cela signifie que $V_2$ est obtenu précédemment lors d'une réduction $u_j \hookrightarrow u_{j+1}$ (avec $j < i-1$) alors que la réduction $h$ était possible.

La réduction ne serait alors pas gauche. D'où l'absurdité.

15. Montrer que si une grammaire est non-ambiguë alors tout mot engendré par celle-ci admet une unique réduction gauche.
**Corrigé :**

Si la grammaire est non-ambiguë, alors elle admet une unique dérivation droite. Ainsi, d'après la question précédente, elle admet une unique réduction gauche.

16. Montrer que si on a $u \hookrightarrow v$ alors il existe $x, h, y \in (\mathcal{V} \cup \Sigma)^*$ et une règle $T \to h \in R$ tels que $u = xhy$ et $v = uTy$. On dit que $h$ est un facteur réductible de $u$.
**Corrigé :**

Si $u \hookrightarrow v$ alors $v \Rightarrow u$ et donc par définition de la dérivation, il existe $x, h, y \in (\mathcal{V} \cup \Sigma)^*$ tq $v = xTy, u = xhy$ et il existe une telle règle $R : T \to h$. D'où le résultat voulu.

17. Soit $\omega = u_1 \hookrightarrow u_2 \hookrightarrow \cdots \hookrightarrow u_k$ une réduction gauche avec $\omega \in \Sigma^*$. On a alors pour chaque $u_i$ l'existence d'une factorisation sous la forme $u_i = xhy$ et d'une règle sous la forme $T \to h$ tq $u_{i+1} = xTy$. Justifier que $y$ est commposé uniquement de symboles terminaux.
**Corrigé :**

Si $y$ contient un symbole non-terminal, alors celui-ci est apparu lors d'une étape précédente comme résultat d'une réduction, mais alors la réduction ne peut pas être gauche car elle a réduit un facteur à droite d'un facteur irréductible :

— $h$ si déjà présent lors de la réduction
— l'un de ses facteurs non-terminaux sinon

Absurde

On appelle mot valide un mot $m \in (\mathcal{V} \cup \Sigma)^*$ pour lequel il existe $u \in L(\mathcal{G})$ tel que $m$ apparait dans une réduction gauche de $u$.

18. Dire (et justifier) si les mots suivants sont valides pour la grammaire $\mathcal{G}_1$ :
(a) $aaRbb$
(b) $aTbb$
(c) $aaRbbbbb$
(d) $aaabbbbbb$

**Corrigé :**

(a) $aaRbb$ est valide car appartient à la réduction gauche suivante :

$$\begin{aligned} aaabbb &\hookrightarrow aaRbb \\ &\hookrightarrow aRb \\ &\hookrightarrow R \\ &\hookrightarrow S \end{aligned}$$

(b) De même :

$$\begin{aligned} aabbbb &\hookrightarrow aTbb \\ &\hookrightarrow T \\ &\hookrightarrow S \end{aligned}$$

(c) $aaRbbbbb$ n'est pas un mot valide (aucune réduction n'existe pour faire apparaître ce mot).
(d) $aaabbbbbb \in L(\mathcal{G}_1)$ donc il s'agit bien d'un mot valide.

19. Pour tout mot valide $m$, on appelle facteur réductible gauche, un facteur réductible de $m$ situé le plus à gauche possible tel que si on lui applique la réduction on obtient un mot valide. Pour chacun des mots valides identifiés dans la question précédente, donner son facteur réductible gauche.
**Corrigé :**

— Pour $aaRbb$ : $ab$
— Pour $aTbb$ : $aTbb$
— Pour $aaabbbbbb$ : $abb$

20. Une grammaire est dite déterministe si pour tout mot $m$ valide de $(\mathcal{V} \cup \Sigma)^*$, il existe une unique réduction directe gauche vers un mot valide de règle $T \to h$ avec $m = xhy$ et que pour tout $y' \in \Sigma^*$, cette même réduction est l'unique réduction gauche de $m' = xhy'$ vers un mot valide quand $m'$ est valide.
Justifier à l'aide des mots $aaabbb$ et $aaabbbbbb$ que $\mathcal{G}_1$ n'est pas déterministe.
**Corrigé :**

On a

$$m = \underbrace{aa}_{x}\underbrace{ab}_{h}\underbrace{bb}_{y}$$

est un mot valide donc $h = ab$ et $h$ unique pour $m$.
Or

$$\begin{aligned} m' &= aaabbbbbb \\ &= xabbbbbb \end{aligned}$$

ainsi, $m'$ admet $h' = abb$ comme unique réduction gauche.
Finalement,

$$h \neq h'$$

D'où le caractère non-déterministe de $\mathcal{G}_1$.