Dev 1 : ALGORITHME DE COCKE-YOUNGER-KASAMI

Notations :
$G = (\Sigma, N, R, S)$ une grammaire algébrique, $R \subset N \times (\Sigma \cup N)^*$, $S \in N$
$w \in \Sigma^*$, $\hat{\Sigma} := \Sigma \cup N$, $n = |w|$
La taille d'une règle est la somme des longueurs des côtés droits et gauches
La taille d'une grammaire est la somme des tailles de ses règles.

L'algorithme CYK décide en temps $O(|w|^3 |G|)$ si le mot w appartient au langage engendré par la grammaire G.

Mise sous forme 2NF :
G est en 2NF si pour toute règle $X \to \alpha$, on a $|\alpha| \leq 2$.
Pour chaque règle $X \to x_1 \dots x_p$, avec $\forall i \in [\![1, \dots, p]\!]$, $x_i \in \hat{\Sigma}$:
Si $p > 2$, on remplace la règle par l'ensemble de règles :
$\{X \to x_1 X_1, X_{p-2} \to x_{p-1} x_p\} \cup \{X_i \to x_{i+1} X_{i+1}, i \in [\![1, \dots, p-3]\!]\}$
Taille de l'ancienne règle : $p+1$
Taille de la nouvelle règle : $3(p-1)$
$\Rightarrow$ On obtient une nouvelle grammaire G' en 2NF, avec $|G'| = \Theta(|G|)$
On suppose désormais que G est en 2NF

Symboles annulables :

```
ANNULABLES(G):
Todo = []
Pour tout A ∈ N
    Lien(A) = []
    Nul(A) = Faux
Pour tout A -> B ∈ R
    Lien(B) = Lien(B) ∪ {A}
Pour tout A -> BC ∈ R
    Lien(B) = Lien(B) ∪ {<A,C>}
    Lien(C) = Lien(C) ∪ {<A,B>}
Pour tout A -> ε ∈ R
    Nul(A) = Vrai
    Todo = Todo ∪ {A}
Tant que Todo ≠ ∅
    Retirer un B de Todo
    Pour tout A ou <A,C> dans Lien(B) avec Nul(A) = Faux
        Si Nul(A) = Faux
            Nul(A) = Vrai
            Todo = Todo ∪ {A}
Renvoyer Nul
```

On note $\mathcal{E} := \{X \in N, X \to^* \varepsilon\}$.
Grâce à l'algorithme, $\mathcal{E}$ est calculable en temps $\Theta(|G|)$
Si $w = \varepsilon$, il suffit alors de tester si $S \in \mathcal{E}$
On suppose désormais que $w \neq \varepsilon$

Prédécesseurs :
Pour $y \in \hat{\Sigma}$, $P(y) := \{x \in \hat{\Sigma}, x \to^* y\}$
Pour $M \subset \hat{\Sigma}$, $P(M) := \bigcup_{y \in M} P(y)$

Graphe des prédécesseurs :
$\mathcal{G}_f := (\hat{\Sigma}, \mathcal{A})$ avec $\mathcal{A} := \{(x, X), X \to \alpha x \beta$ où $\alpha, \beta \in \mathcal{E} \cup \{\varepsilon\}\}$
On le construit en mettant tous les états, puis pour chaque règle $X \to x$ on ajoute l'arête, enfin pour les règles $X \to xy$, on teste si x ou $y \in \mathcal{E}$, et le cas échéant, on ajoute l'arête ou les deux arêtes.
On peut donc le construire en temps $\Theta(|G|)$.

Lemme 1 : $x \to^* y$ dans $G \iff y \rightsquigarrow^* x$ dans $\mathcal{G}_f$

$\Rightarrow$: récurrence sur la longueur de $x \to^* y$, notée p :
$p = 0$: on a $x = y$ donc $y \rightsquigarrow^* x$ dans $\mathcal{G}_f$
Soit $p > 0$, on suppose que $\forall x, y$, si $x \to^p y$ alors $y \rightsquigarrow^* x$
Soit $x, y \in \hat{\Sigma}$ avec $x \to^p y$.
On note α tq $x \to \alpha \to^{p-1} y$: α a donc 1 ou 2 symboles.
Si α n'a qu'un seul symbole, par HR : $y \rightsquigarrow^* \alpha$ et $\alpha \rightsquigarrow x \Rightarrow y \rightsquigarrow^* x$.
Sinon, $\alpha = \alpha_1 \alpha_2$ et $x \to \alpha_1 \alpha_2$, $\alpha_1 \to^{p_1} y$ et $\alpha_2 \to^{p_2} \varepsilon$ (ou l'inverse), avec
$p_1 + p_2 = p - 1 \Rightarrow p_1 < p$ et $p_2 < p \Rightarrow \alpha_1 \rightsquigarrow x$ et par HR, $y \rightsquigarrow^* \alpha_1$, d'où $y \rightsquigarrow^* x$

$\Leftarrow$: $y \rightsquigarrow^* x \Rightarrow \exists p \in \mathbb{N}, y \rightsquigarrow \beta_1 \rightsquigarrow \dots \rightsquigarrow \beta_p \rightsquigarrow x$.
Alors $\beta_1 \to^* y$, $\forall i \in \{1, \dots, p-1\}$, $\beta_{i+1} \to^* \beta_i$ et $x \to^* \beta_p$ car par définition de $\mathcal{G}$, chaque arête $x \rightsquigarrow X$ correspond à une règle $X \to \alpha x \beta$ avec $\alpha, \beta \in \mathcal{E} \cup \{\varepsilon\} \Rightarrow X \to^* x$.
Donc $x \to^* y$.

Finalement, $P(M) = \{$sommets accessibles depuis M dans $\mathcal{G}\}$, calculable grâce à un parcours quelconque en $\Theta(|G|)$.

Notations:

Pour $1 \leq i \leq j \leq m$, $T_{i,j} := P(T'_{i,j})$
Pour $1 \leq i \leq m$, $T'_{i,i} = \{w_i\}$
Pour $1 \leq i < j \leq m$, $T'_{i,j} = \{X \in V, \exists k \in \{i, \dots, j-1\}, \exists \beta_1 \in T_{i,k}, \exists \beta_2 \in T_{k+1,j}, X \to \beta_1 \beta_2\}$.

Lemme 2 : $T_{i,j} = \{x \in \hat{\Sigma}, x \to^* w_i \dots w_j\}$
On a déjà $T_{i,j} = \{x \in \hat{\Sigma}, \exists y \in T'_{i,j}, x \to^* y\}$ et $T'_{i,j} \subset T_{i,j}$
$\subset$: Par récurrence sur $j-i$:
$i=j$: $T'_{i,i} = \{w_i\}$ et soit $x \in T_{i,i} \Rightarrow \exists y \in T'_{i,i}, x \to^* y$
$\Rightarrow y = w_i \Rightarrow x \to^* w_i \Rightarrow x \in \{x \in \hat{\Sigma}, x \to^* w_i\}$.
$j > i$: soit $x \in T_{i,j} \Rightarrow \exists y \in T'_{i,j}, x \to^* y$.
Vu que $y \in T'_{i,j}$, $\exists k \in \{i, \dots, j-1\}, \exists \beta_1 \in T_{i,k}, \exists \beta_2 \in T_{k+1,j}, y \to \beta_1 \beta_2\}$
Par HR, $\beta_1 \to^* w_i \dots w_k$ et $\beta_2 \to^* w_{k+1} \dots w_j$
$\Rightarrow y \to^* w_i \dots w_j \Rightarrow x \to^* w_i \dots w_j \Rightarrow x \in \{x \in \hat{\Sigma}, x \to^* w_i\}$.
$\supset$: Par récurrence sur la longueur de $x \to^* w_i \dots w_j$ avec $1 \leq i \leq j \leq m$.
$p=0$: $x = w_i \dots w_j \Rightarrow i=j \Rightarrow x = w_i \Rightarrow x \in T'_{i,i} \subset T_{i,i}$
$p>0$: Soit $x \in \hat{\Sigma}$ et tel que $x \to^p w_i \dots w_j \Rightarrow x \in V$
(i) : $x \to y \to^{p-1} w_i \dots w_j$. Par HR, $y \in T_{i,j}$. Vu que $x \to y \Rightarrow x \in T_{i,j}$.
(ii) : $x \to y_1 y_2$ et $y_1 \to^{p_1} w_i \dots w_k$ et $y_2 \to^{p_2} w_{k+1} \dots w_j$ avec $k \in \{i-1, \dots, j\}$.
Si $k = i-1$, $y_2 \to^{p_2} w_i \dots w_j$ donc par HR, $y_2 \in T_{i,j}$.
Vu que $y_1 \in \mathcal{E}$, $x \to^* y_2$ donc $x \in T_{i,j}$.
Si $k=j$: idem en sens inverse que $k = i-1$
Sinon, par HR, $y_1 \in T_{i,k}$, $y_2 \in T_{k+1,j}$ et $k \in \{i, \dots, j-1\}$
Donc $x \in T'_{i,j} \subset T_{i,j}$

Algorithme:

```
CYK(G, w)
Construire E et G                                  ← Θ(|G|)
Pour i = 1 à m                                     } Θ(m|G|)
    T_{i,i} = P(w_i)                               ← Θ(|G|)
Pour j = 2 à m                                     } O(m²)
    Pour i = j-1 à 1
        T_{ij} = ∅                                 ← Θ(1)
        Pour k = i à j-1
            Pour tout X -> xy                      } Θ(|G|)
                Si x ∈ T_{i,k} et y ∈ T_{k,j}
                    T_{ij} = T_{ij} ∪ {X}
        T_{ij} = P(T_{i,j})                        ← Θ(|G|)
Renvoyer (S ∈ T_{1,m})
```

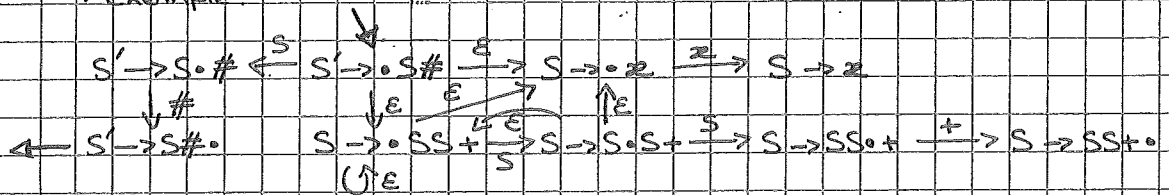
$\Theta(m|G|)$ $\Theta(m|G|)$ $\Theta(m|G|)$ $\Theta(m|G|)$ $\Theta(m^3|G|)$ $\Theta(m^3|G|)$

GRAMMAIRE LR(0)

Notations :
$G=(\Sigma, N, R, S)$ une grammaire, $R \subset N \times (\Sigma \cup N)^*$, $S \in N$
→ Exemple : $G_0 = (\{S\}, \{x, +\}, \{S \to SS+, S \to x\}, S)$.

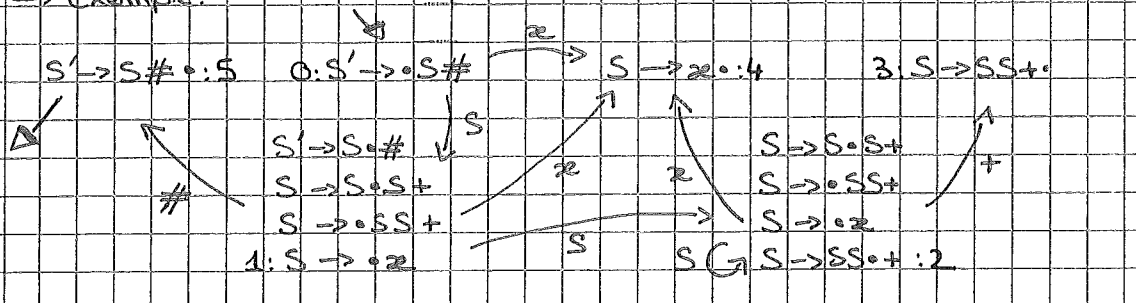
Item de la grammaire :
Pour $(A, \alpha, \beta) \in N \times (\Sigma \cup N)^*$, tq $A \to \alpha\beta \in R$, on note $[A \to \alpha \bullet \beta]$ l'item associé.
Un item $[A \to \bullet \beta]$ est initial ($\alpha = \varepsilon$)
Un item $[A \to \alpha \bullet]$ est final ($\beta = \varepsilon$).
On rajoute en plus des items $[S' \to \bullet S\#]$, $[S' \to S \bullet \#]$ et $[S' \to S\# \bullet]$ correspondant à la règle $S' \to S\#$ avec $S' \notin \Sigma \cup N$ et $\# \notin \Sigma \cup N$, avec $\#$ qui est un symbole qu'on ajoute à la fin pour signifier la fin de l'entrée.

Automate des items :
$\mathcal{A} = (Q, \Sigma \cup N, \Delta, \{[S' \to \bullet S\#]\}, \{[S' \to S\# \bullet]\})$, avec Q l'ensemble des items et $\Delta([A \to \alpha \bullet B\beta], \varepsilon) = \{[B \to \bullet\gamma] \mid$ et $\Delta([A \to \alpha \bullet X\beta], X) = \{[A \to \alpha X \bullet \beta]\}$.
→ Exemple :



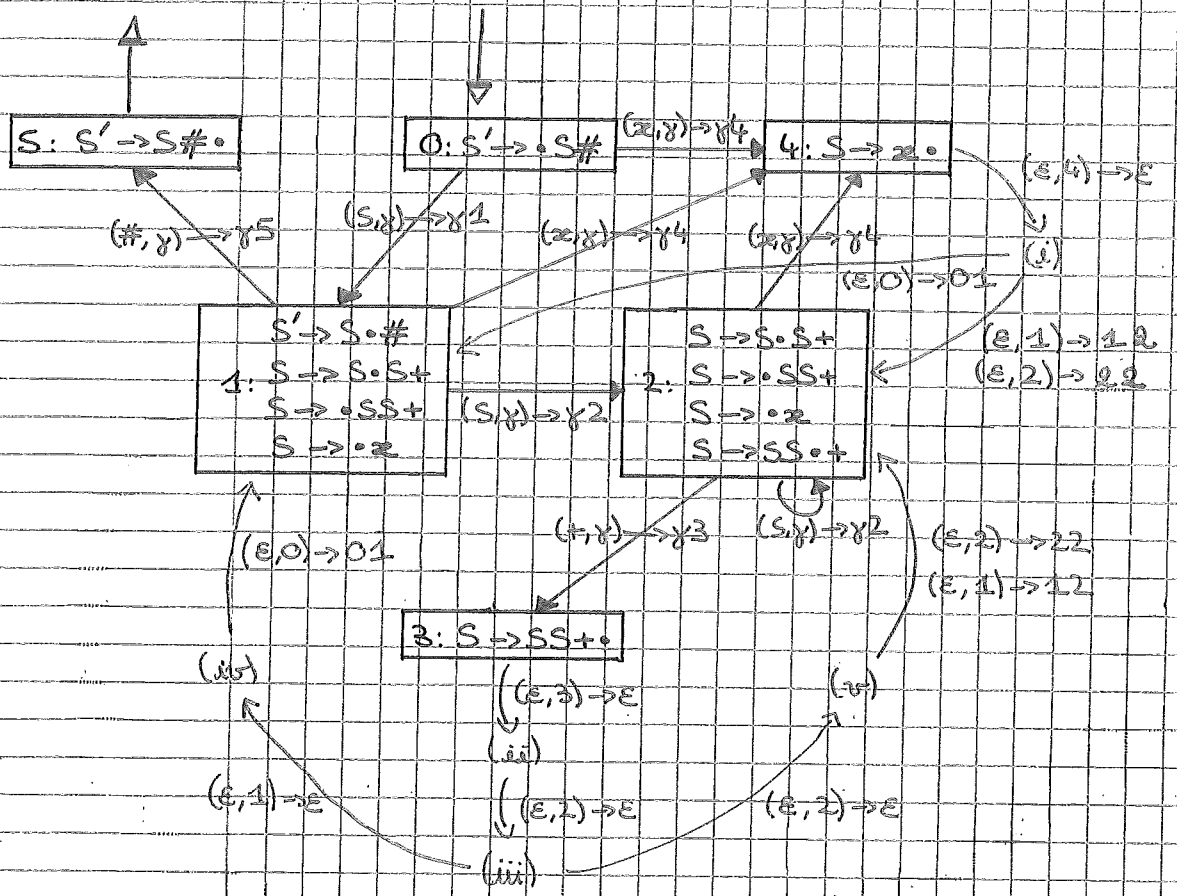
Propriété :
Pour tout $\gamma \in (\Sigma \cup N)^*$, pour tout item $[A \to \alpha \bullet \beta]$ tel que $\exists \delta, \gamma = \delta\alpha$:
$[A \to \alpha \bullet \beta] \in \Delta(S', \delta\alpha) \iff \exists w$ tq $S \to^* \delta A w \to \delta\alpha\beta w$
Preuve :
$\Rightarrow$: par induction sur la longueur de $S \to^* \delta A w$
$\Leftarrow$: par induction

Suite de la construction :
On déterminise l'automate des items :
→ Exemple :



Automate à pile
Pour chaque transition $q_i \xrightarrow{\alpha} q_j$, on modifie la transition (de lecture) :
$(q_i, \beta) \xrightarrow{\alpha} (q_j, q_i\beta)$.
On numérote les états : 0 est l'état initial.
Le symbole de pile initial est 0.
Le seul état final est l'état initial
On rajoute des transitions correspondants à des réductions : pour chaque état avec un item complet, on veut rebrousser le chemin correspondant à la partie droite de la règle et refaire le chemin de la partie gauche.

→ Exemple :


S: S' -> S#•
0: S' -> •S#
(x,y) -> y4
4: S -> x•
(ε,4) -> ε
(#,y) -> y5
(S,y) -> y1
(x,y) -> y4
(x,y) -> y4
(i)
(ε,0) -> 01
S' -> S•#
1: S -> S•S+
S -> •SS+
S -> •x
(S,y) -> y2
2: S -> S•S+
S -> •SS+
S -> •x
SS -> SS•+
(ε,1) -> 12
(ε,2) -> 22
(+,y) -> y3
(S,y) -> y2
(ε,2) -> 22
(ε,1) -> 12
(ε,0) -> 01
3: S -> SS+•
(ε,3) -> ε
(iii)
(ε,1) -> ε
(ε,2) -> ε
(ε,2) -> ε
(iv)
(v)
(vi)