Algorithme de Cocke - Younger - Kasami

Référence : Partie 4.2.3 du Carton

Idée : Décider si un mot est engendré par une grammaire. Utile pour transformer une suite de lexèmes en un arbre de syntaxe en analyse syntaxique lors de la compilation.

Définition 1 : On appelle **grammaire hors-contexte** un tuple (Σ, N, S, R) avec :

- Σ un alphabet de symboles terminaux
- N un alphabet de symboles non-terminaux
- S un symbole de départ appelé **axiome**
- R un ensemble de règles de $N \times (\Sigma \cup N)^*$ noté $X \to u_1 \dots u_n$.

Définition 2 : On dit que u **se dérive en** v si $u = \alpha X \beta$, $v = \alpha u_1 \dots u_n \beta$ et $X \to u_1 \dots u_n \in R$.
On appelle ce processus la **dérivation**.
Un mot appartient au langage d'une grammaire s'il existe une suite de dérivations de l'axiome au mot.

Propriété 3 : Toute grammaire hors contexte est équivalente à une grammaire dont les règles sont de la forme $X \to a$ ou $X \to YZ$, $a \in \Sigma$ et $Y, Z \in N$. (grammaire quadratique).

Problème 4 : Entrée : grammaire G hors contexte sous forme quadratique, mot w

Sortie : Décide si $w \in L(G)$.

Idées : On utilise la programmation dynamique pour calculer les $E_{i,j}$ pour $i,j \in [\![1,n]\!]$ avec $n = |w|$, où $E_{i,j}$ est l'ensemble des variables X telles que $w[i,j] \in L_G(X)$.

Propriété 6 : On a les relations suivantes :
- $\forall i \in [\![1;n]\!]$, $E_{i,i} = \{X \in N \mid X \to w[i] \in R\}$
- $\forall i \in [\![1;n]\!]$, $\forall j \in [\![i+1,n]\!]$, $E_{i,j} = \{X \in N \mid \exists k \in [\![i,j-1]\!] \; \forall Y,Z \in N$ tels que $X \to YZ \in R$, $Y \in E_{i,k}$ et $Z \in E_{k+1,j}\}$

Fonction CYK(W : mot) :

```
n := longueur(w)
E_{i,j} := [] pour i,j ∈ [1;n]
Pour i de 1 à n faire
    Pour S ∈ N faire
        Si S → w[i] ∈ R faire
            E_{i,i} := S :: E_{i,i}
        Fin si
Pour i de n à 1 faire
    Pour j de (i+1) à n faire
        Pour k de i à j-1 faire
            Pour Z → XY ∈ R faire
                Si X ∈ E_{i,k} et Y ∈ E_{k+1,j} faire
                    E_{i,j} := Z :: E_{i,j}
Retourner S ∈ E_{1,n}
```

Théorème 6 : Cet algorithme est cubique en la taille du mot.

Preuve : On itère i, j et k qui peuvent prendre un nombre de valeurs majoré chacun par n.
On a alors une complexité temporelle de $O(n^3 \times |R| \times |N|)$ pour des structures de données naïves, ce qui est cubique en n.

Remarque 7 : On ne prend pas en compte la taille de la grammaire car elle est supposée fixe. On l'intègre donc à la constante.

Exemple 8 :

$S \to 0$
$S \to S S_+$
$S_+ \to V_+ S$
$V_+ \to +$
$S' \to S S_+$

	0	+	0	+	0
0	S	[]	S,S'	[]	S,S'
+	/	V_+	S_+	[]	S_+
0	/	/	S	[]	S,S'
+	/	/	/	V_+	S_+
0	/	/	/	/	S