Algorithme de Cocke-Younger-Kasami

Référence : 131 Développements pour l'oral, D. Lesesvre

Introduction. L'algorithme CYK permet de résoudre le problème du mot pour les grammaires algébriques. On dit qu'une grammaire $G = (V, \Sigma, R, S)$ est sous forme normale de Chomsky si elle ne contient que des règle de la forme $A \to a$ (avec $a \in Sigma$ et $A \in V$), ou $A \to A_1A_2$ $(A_1, A_2 \in V)$.

Définition 1 (Problème de mot)**.** *Étant donné un mot $w = w_1...w_n$ et une grammaire G sous forme normale de Chomsky, a-t-on $w \in L(G)$?*

Algorithme CYK. On utilisera de la programmation dynamique. Pour tous les indices $1 \leqslant i \leqslant j \leqslant n$, on note

$$E_{i,j} = \{A \in V, A \to^* w_i...w_j\}$$

Ainsi, $w \in L(G)$ ssi $S \in E_{1,n}$. On va donc calculer $E_{1,n}$ par programmation dynamique.

Initialisation. Montrons que pour tout $1 \leqslant i \leqslant n$, on a

$$E_{i,i} = \{A \in V : A \to w_i \in R\}$$

En effet, si $A \in E_{i,i}$, alors $A \to^* w_i$. Or,puisque G est sous forme normale de Chomsky, on a jamais $X \to^* \epsilon$ pour $X \in V$. Ainsi, la première règle de $A \to^* w_i$ n'est pas de la forme $A \to A_1A_2$ qui donnerait un mot d'au moins 2 lettres. On en déduit qu'on applique une règle de la forme $A \to a$, et donc $a = w_i$. Ainsi, $A \to w_i \in R$.

L'autre inclusion est triviale.

Récurrence. Soit $1 \leqslant i < j \leqslant n$, montrons

$$E_{i,j} = \bigcup_{k=i}^{j-1} \bigcup_{\substack{B \in E_{i,k} \\ C \in E_{k+1,j}}} \{A \in V : A \to BC \in R\}$$

Montrons l'inclusion indirecte. Soit $A \in V$ tel qu'il existe k tel que $i \leqslant k \leqslant j-1$, $B \in E_{i,k}$ et $C \in E_{k+1,j}$ avec $A \to BC \in R$.

On a alors $B \to^* w_i....w_k$, et $C \to^* w_{k+1}...w_j$. Ainsi, on a :

$$A \to BC \to^* w_i....w_kC \to^* w_i....w_j$$

Ainsi, $A \in E_{i,j}$.

Réciproquement, soit $A \in E_{i,j}$, c'est-à-dire $A \to^* w_i...w_j$. On a au moins deux lettres dans le mot $w_i...w_j$, la première règle appliquée est donc de la forme $A \to BC$. On utilise alors l'arbre de dérivation de $A \to^* w_i...w_j$, A est la racine et a deux enfants, B et C. Pour exhiber k, il suffit de prendre le nombre l de feuilles dans l'arbre enraciner en B, et on pose $k = i + l - 1$. Ainsi, on a $B \to^* w_i...w_k$ et $C \to^* w_{k+1}...w_j$. Cela conclut la preuve.

Algorithme. L'algorithme va donc d'abord calculer les $A_{i,i}$ et remonter jusqu'à l'ensemble $E_{1,n}$ en calculant diagonale par diagonale.

Algorithme 1 : CYK(w,G)

```
pour 1 ⩽ i ⩽ j ⩽ n faire
    E_{i,j} ← ∅ ;
pour i = 1...n faire
    pour A → a ∈ R faire
        si a = w_i alors
            A_{i,i} ← E_{i,i} ∪ {A} ;
pour d = 2...n faire
    pour (i, j) sur la d-diagonale supérieure faire
        pour k = i...j − 1 faire
            pour A → B_1 B_2 faire
                si B_1 ∈ E_{i,k} et B_2 ∈ E_{k+1,j} alors
                    E_{i,j} → E_{i,j} ∪ {A} ;
retourner S ∈ E_{1,n} ;
```

Implémentation et complexité. On veut pouvoir ajouter et vérifier rapidement si des non-terminaux appartiennent à un ensemble $E_{i,j}$. On peut alors, pour tout $1 \leqslant i \leqslant j \leqslant n$, utiliser un tableau booléen de taille $|V|$ pour avoir les deux opérations en temps constant.

On a alors une complexité temporelle en $\mathcal{O}(|R| \times n^3)$. pour la complexité spatiale, on a besoin de $\frac{n(n+1)}{2}$ tableaux de taille $|V|$, donc un $\mathcal{O}(n^2|V|)$.

Commentaires. Il fait d'abord transformer une grammaire pour la mettre sous forme normale de Chomsky. Cette transformation peut faire exploser la taille de la grammaire. Il faut aussi vérifier qu'on a bien $\epsilon \notin L(G)$, ce qui peut se faire par saturation.