Cocke-Younger-Kasami

Antoine Dequay

21 septembre 2022

Notes

- Prof : .
- Leçon : 907, 915, 923, 931.
- Références :
  - Le Barbenchon.

Définition 1 *Forme normale de* CHOMSKY. On dit qu'une grammaire algébrique $G = (\mathcal{N}, \mathcal{T}, S, \mathcal{R})$ est sous forme normale de CHOMSKY si elle ne contient que des règles de la forme $A \in \mathcal{N} \to a \in \mathcal{T}$ ou $A \in \mathcal{N} \to A_1A_2 \in$ "$\mathcal{N}\mathcal{N}$".

Définition 2 *Problème du mot.* On définit le problème du MOT :

- entrée : Un mot $w = w_1 \dots w_n$ une grammaire algébrique G sous forme normale de CHOMSKY,
- sortie : oui si $w \in L(G)$, non sinon.

On cherche un algorithme permettant de résoudre ce problème.

Pour cela, notons, pour $i, j \in [\![1, n]\!]$, $i \leq j$:

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

On a donc $w \in L(G) \iff S \in E_{1,n}$. On cherche donc à calculer $E_{1,n}$.

Proposition 3 Pour tout $i \in [\![1, n]\!]$,

$$E_{i,i} = \{A \in \mathcal{N},\ A \to w_i \in \mathcal{R}\}.$$

Preuve. Au vue de la forme d'une grammaire de CHOMSKY ($A \to \varepsilon$ n'existe pas), un mot construit en utilisant une règle de la forme $A \to BC$ contiendra au moins 2 lettres, d'où le résultat.

□

Proposition 4 Pour $1 \leq i < j \leq n$, on a :

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

Preuve.

- $\supseteq$: Immédiate ! Il suffit de l'écrire.
- $\subseteq$: Comme $i \neq j$, $w_i \dots w_j$ possède au moins 2 lettres, donc si $A \in E_{i,j}$, A s'écrit forcément sous la forme $A \to BC$, $B, C \in \mathcal{N}$, d'où $BC \to^* w_i \dots w_j$. Par lemme de factorisation, il existe $k \in \mathbb{N}$ et des dérivations $B \to^* w_i \dots w_k$ et $C \to^* w_{k+1} \dots w_j$. Par la forme d'une grammaire de CHOMSKY, il vient donc $i \leq k < j$ et $B \in E_{i,k}$, $C \in E_{k+1,j}$, d'où le résultat.

□

Pour calculer par programmation dynamique successivement les $E_{i,j}$ pour arriver à $E_{1,n}$, on utilise la représentation matricielle suivante :

$$\begin{pmatrix}
E_{1,1} & \dots & E_{1,i} & \dots & E_{1,j} & \dots & E_{1,n} \\
 & \ddots & \vdots & & \vdots & & \vdots \\
 & & E_{i,i} & \rightarrow & E_{i,j} & \dots & E_{i,n} \\
 & & & \ddots & \uparrow & & \vdots \\
 & & & & E_{j,j} & \dots & E_{j,n} \\
 & & & & & \ddots & \vdots \\
 & & & & & & E_{n,n}
\end{pmatrix}$$

et on parcours les diagonales dans le sens $\nearrow$.

Théorème 5 Ainsi, l'algorithme suivant permet bien de résoudre le problème du mot :

Algorithm 1: CYK(w, G)

Entrée: Un mot w et une grammaire G.

Sortie : Un booléen indiquant si $w \in L(G)$.

```
Initialiser E_{i,j} = ∅;
for i ← 1 to n do
    for A → a ∈ R do
        if a = w_i then
            E_{i,i} ← E_{i,i} ∪ {A}
for d ← 2 to n do
    for (i, j) sur la d-diagonale supérieure do
        for k ← i to j − 1 do
            for A → B_1 B_2 ∈ R do
                if B_1 ∈ E_{i,k} et B_2 ∈ E_{k+1,j} then
                    E_{i,j} ← E_{i,j} ∪ {A}
return w ∈ E_{1,n}.
```

Remarque On peut par exemple implémenter ces ensemble dans une matrice indicée par (i, j), ayant dans chaque case une liste de booléens indexée sur les non-terminaux. La complexité temporelle est alors en $O(n^3 \# \mathcal{R})$, et la complexité spatiale en $O(n^2 \# \mathcal{N})$.

Remarque — Quand se ramène à une forme de CHOMSKY, on perd ε s'il est dans le langage,

— Dans le cas des machines de TURING, le problème est indécidable,

— Il existe des problèmes indécidables dans le cas des grammaires algébriques.