Algorithme de Cocke-Younger-Kasami
Mise sous forme normale de Chomsky

Léo Gayral

2017-2018

ref : Floyd – Le langage des machines – p.270, p.318
An Efficient Yet Presentable Version of the CYK Algorithm

Définition 1. Soit G une grammaire. On note Σ ses terminaux, V ses non-terminaux et $S \in V$ son axiome. On dit que G est sous forme normale de Chomsky (FNC) lorsque toute règle de G est sous la forme :
- $X \to YZ$ avec $X \in V$ et $Y, Z \in V \backslash \{S\}$,
- $X \to a$ avec $X \in V$ et $a \in \Sigma$.

Théorème 1. Soit G une grammaire sous FNC. On considère l'algorithme CYK :

```
1  def CYK(V, G, w) :
2      n = |w|
3      B = [0]_{(i,j,X)∈⟦1,n⟧²×V}
4      pour i de 1 a n :
5          pour (X → a) ∈ G :
6              si w[i] = a :
7                  B[i, i, X] = 1
8      pour k de 1 a n − 1 :
9          pour i de 1 a n − k :
10             j = i + k
11             pour (X → YZ) ∈ G :
12                 pour r de i a j − 1 :
13                     si B[i, r, Y] = B[r + 1, j, Z] = 1 :
14                         B[i, j, X] = 1
```

Cet algorithme calcule $B[i,j,X] = \mathbb{1}_{w_i \dots w_j \in L(X)}$ par programmation dynamique. Cet algorithme a une complexité en $O\left(|G| \times n^3\right)$.

Démonstration.
Par induction structurelle, on voit que tout mot de $\bigcup_{X \in V} L(X)$ est de longueur strictement positive. On en déduit qu'en procédant par récurrence sur la

longueur des facteurs de w – ce qui correspond à la grosse boucle *for* selon k – que $w_i \dots w_j \in L(X)$ ssi on a une certaine règle $X \to YZ$ et une factorisation $i \leq r < j$ telle que $w_i \dots w_r \in L(Y)$ et $w_{r+1} \dots w_j \in L(Z)$. C'est exactement ce que vérifient les deux boucles *for* internes. □

Théorème 2. Soit G une grammaire quelconque. On peut ramener cette grammaire à une grammaire G' sous FNC de sorte que $L(S)\ \{\epsilon\} = L(S')$. On a de plus $|G'| = O\left(|G|^2\right)$.

Démonstration.
On va procéder en plusieurs étapes.

Premièrement, on commence par ajouter une règle $S_0 \to S$ à G, de sorte que le nouvel axiome S_0 n'apparaisse à droite d'aucune règle. Ensuite, on remplace chaque occurrence du terminal $a \in \Sigma$ par un nouveau non-terminal N_a, et on ajoute une règle $N_a \to a$. Ceci étant fait, on *éclate* les membres droits qui contiennent au moins trois symboles ; on remplace une règle $X \to Y_1 \dots Y_n$ par $X \to Y_1 Z_1$, $Z_1 \to Y_2 Z_2$ et ainsi de suite jusqu'à $Z_{n-1} \to Y_{n-1} Y_n$. A ce stade, on a parcouru une fois chaque règle de G, et triplé sa taille dans le pire des cas. On nomme G_q cette grammaire.

On cherche alors à éliminer les ϵ de G_q. Pour ce faire, on doit en premier lieu déterminer quels non-terminaux X sont *annulables*, vérifient $\epsilon \in L(X)$. On peut déterminer inductivement cet ensemble de non-terminaux W : on initialise $W_0 = \{X, (X \to \epsilon) \in G_q\}$, et on détermine $W_{n+1} = W_n \cup \{X, (X \to YZ) \in G_q, Y, Z \in W_n\}$. Dès que $W_{n+1} = W_n$, on peut s'arrêter. Pour passer de W_n à W_{n+1}, on doit dans le pire des cas lire une à une toutes les règles de G_q. En outre, $|W_n|$ est majoré par le nombre de non-terminaux de G_q, donc la complexité temporelle de cette étape est en $O\left(|G_q|^2\right) = O\left(|G|^2\right)$.

Une fois W déterminé, on commence par effacer les règles $(X \to \epsilon)$. Ensuite, on « remplace » les occurrences des $X \in W$ à droite des règles par des $X|\epsilon$. Ainsi, lorsque $Y \in W$ mais $Z \notin W$, $X \to YZ$ devient $X \to Z|YZ$. Lorsque $Y, Z \in W$, on remplace $X \to YZ$ par $X \to Y|Z|YZ$. On obtient enfin une grammaire G_e, dont la taille est le triple de celle de G_q dans le pire des cas.

Il reste enfin à éliminer les règles du type $X \to Y$, avec un seul non-terminal à droite. On considère le graphe orienté formé par les non-terminaux de G_e. Si on a un cycle $X_1 \to \dots \to X_n \to X_1$, alors $L(X_1) = \dots = L(X_n)$, et on peut identifier tous ces symboles par la suite. On se ramène ainsi au cas d'un graphe acyclique. Dans ce cas, il suffit alors de considérer une arrête $X \to Y$ sans successeur dans le graphe, puis de supprimer la

règle correspondante en ajoutant une règle $X \to \alpha$ pour chaque $Y \to \alpha$. Ce faisant, on ajoute de façon cumulative les règles de tous ses successeurs à chaque sommet du graphe. En fin de compte, on a donc une grammaire G' sous forme normale de Chomsky, telle que $|G'| = O\left(|G|^2\right)$.

□

Remarque 1. Pour G quelconque, une fois qu'on a effectué un pré-traitement pour la mettre sous FNC, on peut tester si $w \in L(S)$ en temps $O\left(|G|^2 \times n^3\right)$.

Notons cependant que, si on interrompt la mise sous FNC après l'élimination des cycles mais avant celle des autres règles $X \to Y$, on a une grammaire sous forme quadratique, de taille comparable à G.

Quitte à utiliser une version généralisée de CYK – où on ordonne les règles selon leur variable à gauche, en *remontant* au sein du graphe orienté acyclique avec un tri topologique – on peut ultimement tester si $w \in L(G)$ en temps $O\left(|G| \times n^3\right)$.