Algorithme de Cocke-Younger-Kasami

Benjamin Hellouin

Hopcroft ?

Définition 1. *Soit Σ un alphabet fini. Une* grammaire hors-contexte *est la donnée d'un ensemble de symboles V, d'un symbole initial $S \in V$ et d'un ensemble de règles de la forme $A \to u$ où $A \in V$ et $u \in (V \cup \Sigma)^*$. Le langage reconnu par une grammaire hors contexte est l'ensemble des mots sur T constructibles depuis S. Elle est dite en forme normale de Chomsky si elle est constituée uniquement de régles du type :*

- $A \to BC$ *où* $A \in V, B, C \in V \backslash S$ *;*
- $A \to \alpha$ *où* $A \in \Sigma, \alpha \in \Sigma$ *;*
- $S \to \varepsilon$.

Définition 2 (Problème à résoudre)**.**
Entrée : une grammaire hors-contexte G et un mot $u = u_1 \dots u_n \in A^*$.
Sortie : OUI si $u \in L(G)$, NON sinon.

Théorème 1. *La reconnaissance de l'appartenance d'un mot $u \in \Sigma^*$ à une grammaire hors-contexte G en FNC est décidable.*

Preuve : On remarque que chaque dérivation de premier type augmente la longueur du mot. Voici une méthode de force brute :

- si $u = \varepsilon$, on regarde si $S \to \varepsilon \in G$
- sinon, on regarde toutes les mots obtenus avec n dérivations de premier type et n dérivations de second type.

Cet algorithme termine en temps $O(|G|^n)$.

On propose un algorithme basé sur la programmation dynamique pour résoudre ce problème : l'algorithme de Cocke-Younger-Kasami. On définit les variables $P[i, j, k]$ par VRAI si le mot $a_i \dots a_j$ peut être obtenu à partir du symbole R_k. Le problème consiste alors à calculer $P[1, n, S]$.

On définit les dépendances locales :

Calcul de $P[i, j, v]$**,** $j > i, v \in V$ **:**

Pour tout règle de production $v \to v' \cdot v'' \in G$ **faire**
  Pour $a \in [i, j-1]$ **faire**
    Si $P[i, a, v']$ et $P[a+1, j, v'']$ **alors**
      $P[i, j, v] \leftarrow VRAI$
Si $P[i, j, v] \neq VRAI$ **alors**
  $P[i, j, v] \leftarrow FAUX$

Cet algorithme, connaissant tous les $P[i', j', v']$ pour $i' + j' < i + j$ et $v' \in V$, calcule $P[i, j, v]$ en temps $O(n.|G|)$. De cette manière, l'ensemble de la table est calculée en temps $O(n^3.|G|^2)$, et l'algorithme se déroule en temps polynômial.

La souplesse de la programmation dynamique permet à cet algorithme de résoudre le même problème où les règles de production se voient assigner un poids et le but est de minimiser le poids total de la dérivation.