Construction d'un Automate Fini Déterministe à partir d'une Expression Régulière

Julien Devevey

2018-2019

Ref : Aho, Sethi, Ullman - Compilateurs : Principes et Outils p.159

Définition 1. (Arbre abstrait) Soit r une expression régulière. L'arbre abstrait de r est construit par induction sur la structure de r par l'induction suivante :
Si $r = a$: l'arbre est juste une feuille étiquetée par a.
Si $r = r_1 \diamond r_2$: $\diamond$ où T_i est l'arbre abstrait de $r_i, i \in \{1, 2\}$ et $\diamond \in \{|, \bullet\}$, $\bullet$ désignant l'opérateur de concaténation.

Si $r = r_0^*$: $*$ où T_0 est l'arbre abstrait de T_0.

Si $r = (r_0)$, l'arbre abstrait de r est le même que celui de r_0.

Définition 2. Quand on écrit un un mot via une expression régulière, on peut associer à chaque lettre un noeud de l'arbre, qu'on appelle position et qui correspond à la règle qui a permis d'écrire cette lettre. On définit alors 4 fonctions sur l'arbre : $Annulable$, $PosSuivante$, $PremierePos$ et $DernierePos$. $PosSuivante(i)$ correspond aux positions dans l'arbre qui peuvent suivre une position donnée i. $PremierePos(n)$ renvoie les positions qui peuvent correspondre à la première lettre d'une sous-chaîne engendrée par le sous-arbre enraciné en n, où n est un noeud de l'arbre. $DernierePos(n)$ renvoie aux contraires les positions correspondant à la dernière lettre d'un mot engendré par le sous-arbre enraciné en n. Enfin, $Annulable(n)$ vaut vrai si et seulement si le sous-arbre enraciné en n peut engendrer la chaîne vide.

Proposition 3. (Calcul des fonctions) Pour calculer $Annulable$ et $PremierePos$, on a les formules d'induction suivantes :

Noeud n	$Annulable(n)$	$PremierePos(n)$
n est une feuille étiquetée par ϵ	Vrai	$\emptyset$
n est une feuille étiquetée par i	Faux	$\{i\}$
	$Annulable(c_1) \vee Annulable(c_2)$	$PremierePos(c_1) \cup PremierePos(c_2)$
	$Annulable(c_1) \wedge Annulable(c_2)$	Si $Annulable(c_1)$ alors $PremierePos(c_1) \cup PremierePos(c_2)$ sinon $PremierePos(c_2)$
	Vrai	$PremierePos(c_1)$

Les règles pour $DernierePos$ sont les mêmes que pour $PremierePos$, mais en inversant les rôles de c_1 et c_2. A partir de ces trois fonctions, on peut calculer $PosSuivante$: Si n correspond à un noeud de concaténation entre c_1 et c_2, on a $\forall i \in DernierePos(c_1), PremierePos(c_2) \subset PosSuivante(i)$. Si n correspond à un noeud étoile, $\forall i \in DernierePos(n), PremierePos(n) \subset PosSuivante(i)$.

Remarque 4. Ces fonctions peuvent être construites par des parcours en profondeur successifs de l'arbre abstrait.

Algorithme 5. L'algorithme 1 construit un AFD à partir d'une expression régulière, les états de l'automate étant des ensembles de position.
Remarque 6. L'algorithme explore en fait tous les états pour lesquels on n'a pas encore construit les transitions sortantes, jusqu'à ce qu'il n'y en ait plus. Si on est dans un tel état, pour chaque lettre de l'alphabet, on cherche l'ensemble des positions dans lesquels on peut être à l'étape suivante en supposant qu'on vient d'écrire cette lettre, ce qui nous fait (potentiellement un nouvel état pas encore exploré et) une nouvelle transition. On construit ainsi l'automate.

Corollaire 7. L'ensemble des langage réguliers est inclus dans l'ensemble des langages reconnaissables par un automate.

Algorithm 1 Calcul des coefficients

Entrée: r une expression régulière
Sortie: Un Automate Fini Déterministe reconnaissant $L(r)$.

$A \leftarrow$ L'arbre abstrait de l'expression régulière étendue $(r)\#$
Construire $Annulable$, $PosSuivante$, $PremierePos$ et $DernierePos$
$Detats \leftarrow [PremierePos(racine(A))]$
$Dtran \leftarrow$ dict()
$marquage \leftarrow []$
tant que Il existe un élément T de $Detats$ qui n'est pas dans $marquage$ **faire**
    Ajouter T à $marquage$
    pour chaque symbole a de l'alphabet **faire**
        $U \leftarrow \bigcup_{p \text{ position de } T | a \text{ est l'étiquette de } p} PosSuivante(p)$
        si $U \neq \emptyset$ et $U \notin Detats$ **alors**
            Ajouter U à $Detats$
        fin si
        $Dtran[T, a] \leftarrow U$
    fin pour
fin tant que
$Dinit \leftarrow PremierePos(racine(A))$
$Dfinal \leftarrow \{U, U \in Detats$ et U contient la position étiquetée par $\#\}$
return $(Detats, Dinit, Dfinal, Dtran)$