Langages et compilation

Langages réguliers - Application à l'analyse lexicale

1 Définition : langages réguliers

Soit V un **alphabet**, c'est-à-dire un ensemble fini non vide de symboles. On appelle **langage** sur V tout sous-ensemble de V^*.

L'ensemble des **langages réguliers** sur V peut-être défini récursivement par les règles suivantes :
- le langage vide $\emptyset$ est un langage régulier ;
- le langage $\{\varepsilon\}$ est un langage régulier ;
- pour tout x de V, $\{x\}$ est un langage régulier ;
- si $R1$ et $R2$ sont des langages réguliers alors : $R1 \cup R2$, $R1.R2$ et $R1^*$ sont des langages réguliers.

Notons que certains langages ne sont pas réguliers, comme par exemple le langage $\{a^n.b^n \mid n > 0\}$ défini sur le vocabulaire $\{a, b\}$. En particulier la plupart des langages de programmation ne sont pas des langages réguliers.

2 Expressions régulières

Tout langage régulier sur V peut être décrit par une **expression régulière**, c'est-à-dire un terme construit sur l'alphabet $V \cup \{+, ., *, \underline{\varepsilon}, \underline{\emptyset}\}$ de la manière suivante :
- $\underline{\emptyset}$ décrit le langage vide $\emptyset$
- $\underline{\varepsilon}$ décrit le langage $\{\varepsilon\}$
- pour tout x appartenant à V, x décrit le langage $\{x\}$
- si $r1$ et $r2$ sont des expressions régulières sur V décrivant respectivement les langages $R1$ et $R2$ alors :
  - $r1 + r2$ décrit le langage $R1 \cup R2$
  - $r1.r2$ décrit le langage $R1.R2$
  - $r1^*$ décrit le langage $R1^*$

On considérera dans la suite que l'opérateur unaire $*$ est plus prioritaire que l'opérateur ., lui-même plus prioritaire que l'opérateur $+$.

Exemples :

On donne ci-dessous quelques exemples d'expressions régulières définies sur le vocabulaire $\{a, b, c\}$, ainsi que les langages (réguliers) correspondants :

Expression régulière	Eléments du langage
$a + b.c$	$\{a, bc\}$
$(ab)^*$	$\{\varepsilon, ab, abab, ababab, \ldots\}$
$ab + c*$	$\{\varepsilon, ab, c, cc, ccc, \ldots\}$

3 Automate d'état fini

Un intérêt des langages réguliers est que, pour tout langage régulier R défini sur V, il existe un algorithme efficace pour décider si un mot quelconque de V^* appartient ou non à R. Cet algorithme repose sur une caractérisation de R à l'aide d'un automate d'état fini déterministe.

On appelle automate d'état fini un quintuplet $A = (Q, V, \delta, q_0, F)$ dans lequel :
- Q est un ensemble fini d'états ;
- V est un ensemble fini de symboles (le vocabulaire d'entrée) ;
- $\delta \subseteq Q \times V \cup \{\varepsilon\} \times Q$ est la relation de transition ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble des états terminaux.

Un automate d'état fini A est dit **déterministe** si et seulement si la relation de transition est une fonction (partielle), c'est-à-dire si elle vérifie la propriété suivante :

$$(\forall (p, q, q') \in Q^3).\ (\forall x \in V).\ \delta(p, x, q) \wedge \delta(p, x, q') \Longrightarrow q = q'$$

Tout mot fini $w = x_0.x_1.x_2.\ldots.x_n$ de V^* est **accepté** par un automate A si et seulement si il existe une séquence $q_0.q_1.q_2.\ldots.q_n.q_{n+1}$ de Q^* telle que :
- pour tout $0 \leq i \leq n$, $(q_i, x_i, q_{i+1}) \in \delta$
- q_{n+1} appartient à l'ensemble F

Le **langage accepté** (ou reconnu) par un automate A, noté $L(A)$ est constitué de l'ensemble des mots acceptés par A.

On a les propriétés suivantes :
- le langage accepté par un automate d'état fini est un langage régulier ;
- à partir de tout automate d'état fini acceptant un langage (régulier) R il est possible de construire un automate d'état fini **déterministe** acceptant le même langage ;
- à partir de toute expression régulière r décrivant un langage (régulier) R il est possible de construire un automate d'état fini déterministe acceptant ce même langage.
- le complément d'un langage régulier est un langage régulier, l'intersection de deux langages réguliers est un langage régulier.

4 Application à l'analyse lexicale

L'intérêt des langages réguliers est qu'il existe un algorithme efficace permettant de décider si un mot fini w appartient ou non à un langage régulier L : il suffit en effet de construire un automate A **déterministe** acceptant L et de vérifier que cet automate accepte bien le mot w. L'automate A étant déterministe, cette vérification peut se faire au fur et à mesure d'un parcours unique de w (elle est donc linéaire en fonction de la taille de ce mot).

Par conséquent il est très important pour l'efficacité de l'analyse lexicale que l'**ensemble des lexèmes** d'un langage de programmation soit un **langage régulier** (même si ce langage lui-même n'est pas régulier).

Pour définir cet ensemble de lexèmes, plutôt que d'utiliser un automate d'état fini ou une expression régulière (dont les tailles pourraient être rédhibitoires), il existe des formalismes plus compact en pratique comme par exemple la notation BNF (*Backus-Naur Form*). Cette notation correspond en fait à une séquence d'expressions régulières de la forme : $a_i ::= r_i$ (pour $i \in 1 \ldots n$), dans laquelle les a_i sont des identificateurs et les r_i sont des expressions régulières définies sur $V \cup \{a_1, a_2, \ldots a_{i-1}\}$. On note r'_i l'expression régulière définie sur V et associée à chaque identificateur a_i. r'_i peut être obtenue en substituant dans r_i chaque occurrence de $a_j \in \{a_1, a_2, \ldots a_{i-1}\}$ par l'expression régulière r'_j correspondante.