Travaux Dirigés de Compilation n°2

Ingénieurs 2000 - IR2

—2005-2006—

Introduction à l'analyse lexicale

Le but de ce TD est d'utiliser le générateur de compilateurs TATOO (`http://igm.univ-mlv.fr/~roussel/Tatoo/`) pour récupérer des lexèmes reçus sur un flot, puis d'implanter, à la main, un analyseur lexical pour des règles simplistes.

▸ Exercice 1. *Simulation d'un analyseur lexical*

Récupérez sous les classes `DFAabstar` *et* `DFAa` *ainsi que l'interface* `DFA`*.*

Les classes `DFAabstar` *et* `DFAa` *implantent des automates à états finis qui reconnaissent, respectivement,* `(ab)*` *et* `a`*. Elles implantent l'interface* `DFA` *qui contient une méthode* `step()` *qui active la transition de l'automate qui correspond au caractère passé en argument. Si l'automate n'a pas une telle transition la méthode retourne la constante* `DFA.REJECT`*. Si un état final est atteint, elle retourne* `DFA.FINAL_ACCEPT` *s'il n'a pas de transition sortante et* `DFA.ACCEPT` *sinon. Enfin, si elle atteint un état quelconque, elle retourne* `DFA.CONTINUE`*. Elle contient également une méthode* `reset()` *qui permet de remettre l'automate dans l'état initial.*

En utilisant ces classes, écrire un analyseur lexical qui reconnaît dans une chaîne de caractères passée en argument, les token correspondants à ces classes. Pour cela, récupérer son squelette sous et écrire la méthode `next()` *qui retourne, au fur et à mesure, les sous-chaînes reconnues. Elle retourne* `null` *lorsque la fin de la chaîne à traiter est atteinte et une exception* `LexerException` *en cas de caractère non reconnu.*

La méthode `next()` *se décompose dans les phases suivantes :*

- *vérifier que l'on est pas en fin de chaîne ;*
- *tant que l'on est pas en fin de chaîne et qu'il y a un automate qui accepte des caractères (actif), faire avancer tous les automates actifs en parallèle en sauvant éventuellement leur dernière position d'acceptation intermédiaire (*`DFA.ACCEPT` *ou* `DFA.FINAL_ACCEPT`*).*
- *repérer l'automate qui a reconnu la plus longue sous-chaîne ;*
- *récupérer la plus longue sous-chaîne reconnue et revenir à la position suivant le dernier caractère reconnu ;*
- *réinitialiser les automates dans leur état initial.*

▸ **Exercice 2.** ***Analyse Lexicale avec*** TATOO

Écrire, en utilisant le générateur de compilateurs TATOO*, un programme qui extrait les commentaires Java d'un fichier dont le nom est reçu en argument sur la ligne de commande.*

Vous utiliserez pour cela les expressions rationnelles définies lors du TD précédents.

Écrire un programme équivalent qui utilise un état qui indique si l'analyseur est en dehors d'un commentaire, dans un commentaire sur une ligne ou dans un commentaire sur plusieurs lignes et qui utilise cet état dans un `Conditioner` *pour déterminer l'ensemble des règles à utiliser.*

Pour cela, Commencer par traiter le cas des commentaires sur une ligne, puis ajouter les commentaires sur plusieurs lignes. Il nécessaire de décomposer le contenu du commentaire en plusieurs tokens.