Analyse lexicale

Licence info S5
TD COMPIL – 2011 - 2012

Exercice 1 : Un automate qui fonctionne comme un an.lex.

Tiré de [Appel], exercice 2.8.

On s'intéresse à un analyseur lexical qui produirait des classes de symboles associés aux 5 langages suivants :

- L_+ : le langage contenant l'opérateur `+` ;
- L_- : le langage contenant l'opérateur `-` ;
- L_i : le langage des identificateurs à la Java composés uniquement de chiffres et de lettres (exemple : `ps2pdf`) ;
- L_e : le langage des entiers non signés (exemple : `209`) ;
- L_r : le langage composé d'un sous-ensemble des réels comportant syntaxiquement une suite de chiffres suivis de la lettre `e` suivis d'une suite de chiffres éventuellement signés (exemple : `12e-3`, `4e26`).

Noter l'absence de tout séparateur.

Q 1.1 : Donner pour chacun des 5 langages un AFD (automate à nombre fini d'états déterministe) qui le reconnaît. □

Q 1.2 : Donner un AFD fonctionnant comme un analyseur lexical qui reconnaît dans une chaîne de caractères les classes de symboles. □

Q 1.3 : Donner pour chaque symbole une description régulière qui le définit. □

Q 1.4 : Comment l'analyseur lexical décompose-t-il les chaînes suivantes ? Dérouler l'analyse sur l'automate.

- $+-$;
- $aa + 9$;
- $a9e + 9$;
- $13e12e4$;
- $9e + a$.

□

Q 1.5 : Reprendre la question précédente en cherchant combien de caractères l'analyseur lexical doit lire après la fin d'un symbole avant de déclarer qu'il a reconnu ce symbole. Peut-on borner ce nombre ? □

Exercice 2 : Vive l'automatisation

Le but de cet exercice est de vous sensibiliser aux avantages de la génération de code comme solution aux tâches répétitives et fastidieuses. On considère un langage de programmation contenant les mot-clés `do`, `od` et les identificateurs à la Java.

Q 2.1 : Donner un AFD qui effectue l'analyse lexicale de ces trois classes de symboles. □

Q 2.2 : Vous avez implanté à la main l'analyseur lexical à partir de l'AFD obtenu. Votre chef est content, vous aussi. Mais votre chef décide que tout compte fait, `done` serait un mot-clé plus agréable que `od` ! Recommencez... □

Exercice 3 : Analyse lexicale de Init

Q 3. 1 : Donner les classes de symboles nécessaires à l'analyse lexicale de INIT, et pour chaque classe une expression ou description régulière qui la reconnaît. Préciser éventuellement quels symboles sont prioritaires sur quels symboles. □

Exercice 4 : Un langage de commande

On souhaite écrire un analyseur lexical pour le langage de commandes suivant :

- une *commande* est composée d'un nom de commande, suivi d'une liste optionnelle d'arguments, suivie d'une liste facultative d'options ;
- une *liste d'arguments* est une suite d'arguments ;
- une *liste d'options* est une suite non vide d'options encadrée par `[` et `]`, à l'intérieur de laquelle les options sont séparées par `,` ;
- une *option* est un caractère précédé d'un tiret ;
- un *argument* est un identificateur, de même qu'un *nom de commande*.

Par exemple, `macom arg1 arg2 [-a,-b]` est une commande, ainsi que `macom [-f]` et `macom`.

Q 4. 1 : Quelles sont les unités lexicales nécessaires à la description d'une commande ? Donner pour chacune d'entre elles une description régulière qui la définit. □