Partiel de compilation

M1 Informatique Fondamentale, ENS Lyon
Mercredi 9 novembre 2011

Durée: 2 heures. Tous les documents sont autorisés.
Ce partiel est constitué de 2 exercices indépendants.
Chaque exercice sera traité sur une copie séparée.

Exercice 1. *Calculatrice en NPI*

Le but de l'exercice est de donner du pseudo code (ocaml|f|ϵ)lex. pour une calculatrice en notation polonaise inverse. La notation polonaise inverse (NPI) permet de noter les formules arithmétiques sans utiliser de parenthèses. Pour écrire une expression faisant appel à un opérateur $f : D^n \to D^m$, on donne d'abord n expressions qui deviendront les opérandes de l'opérateur f donné ensuite. Son évaluation produira m valeurs qui pourront servir comme arguments au prochain opérateur.

Analyse lexicale

Concentrons-nous d'abord sur l'analyse lexicale, en particulier sur l'analyse de nombres réels écrits en notation scientifique. Un nombre en notation scientifique est composé d'un signe s + ou − optionnel et d'une partie entière n, optionnellement suivi d'un point . et d'une partie fractionnelle f auxquels se rajoute optionnellement un facteur de mise à àl'échelle. Le facteur de mise à l'échelle se compose de la lettre e, d'un signe s + ou − optionnel et d'un entier E. La valeur associée à un tel nombre est $s(n + f)10^{sE}$. A titre d'illustration, voici quelques exemples :

4.3
-5.2e-5
+12.07
-0.0e-0
3e10

1. Montrez que le langage formé par les nombres écrit en notation scientifique est régulier en donnant une expression régulière qui l'engendre.

2. Donnez un automate fini qui reconnaît le même langage.

3. Minimisez votre automate.

Analyse syntaxique

Au premier abord, le langage des expressions en notation polonaise inverse ressemble à une suite finie de constantes et d'opérateurs.

1. Montrez que les suites finies de constantes et d'opérateurs forment un langage régulier.

2. Montrez que les expressions en notation polonaise inverse ne forment pas un langage régulier.

3. De quelle structure de donnée supplémentaire a-t-on besoin pour reconnaître uniquement les expressions bien formées.

4. A-t-on vraiment besoin des gros moyens comme (bison|((ocaml|ϵ)yacc)) pour réaliser notre calculatrice ?

Pseudo-code

1. Donnez du pseudo-code (ocaml|f|ϵ)lex pour une calculatrice en NPI minimale. i.e. gérant uniquement les entiers, les opérateurs binaires *plus*, *fois* et l'opérateur unaire *moins*.

Exercice 2. *Production de code*

On cherche à produire du code pour une machine avec un nombre illimité de registres $r_1, r_2, \ldots$, et les instructions suivantes:

registre = mémoire
mémoire = registre
registre3 = registre1 OP registre2

La première instruction transfère le contenu de l'emplacement mémoire dans le registre. La seconde instruction transfère le contenu du registre dans l'emplacement mémoire. La troisième instruction applique une opération arithmétique (+, -, *, /) entre registre1 et registre2, et stocke le résultat dans registre3.

On considère l'expression $E = (a + (b + c)) + ((e + f) + d)$, où a, b, c, d, e, et f sont des variables en mémoire.

Q1) Dessiner l'arbre de E en précisant le nombre de Strahler de chaque noeud (voir cours 4).
Quel est le nombre minimum de registres nécessaires à l'evaluation de cette expression?

Q2) En appliquant l'algorithme de Sethi-Ulmann, générer le code qui utilise un nombre minimum de registres.

Q3) Donnez une version transformée de E qui utilise moins de registres.

Q4) On dispose maintenant de l'instruction suivante: registre3 = registre1 OP mémoire.
Quel est maintenant le nombre minimum de registres nécessaires à l'évaluation de E?

Q5) Montrer que le nombre de Strahler d'une expression peut être arbitrairement grand. Conclusion?