Analyse Syntaxique - Annee 2010-2011
Feuille 5

Construction d'analyseurs syntaxiques avec `yacc`

Exercice 5.1
Reprendre la grammaire des expressions arithmétiques du TD 3 (avec soustractions et divisions). Utiliser les attributs pour que l'analyseur affiche la valeur numérique de l'expression analysée. On affichera un message d'erreur en cas de division par zéro.

Exercice 5.2
Soit la grammaire suivante qui permet de calculer des sommes d'entiers, y compris des sommes de fonctions entières sur un intervalle. Par exemple :
`x = somme (i = 1 .. 3, f(i))`.
On considère seulement deux fonctions : $carre(i)$ et $cube(i)$. La grammaire est donnée dans le format reconnu par `yacc` :

```
%token ID  NUM INT SOMME CARRE CUBE
%left PLUS

%%
axiom : E ';'
;

E :  E  PLUS E
  |  S
  |  INT
;

S : SOMME '(' ID '=' INT '.''.' INT ',' F '(' ID ')' ')'
;

F : CARRE
  | CUBE
;

%%
```

Ajouter pour chaque règle de grammaire, un calcul d'attributs qui permette d'évaluer les expressions. Le résultat sera afficher lors de l'action associée à la règle de l'axiome.

Exercice 5.3
Soit G la grammaire suivante écrite dans le format reconnu par `yacc` :

```
%token ID CHAR INT DOUBLE STRUCT NUM
%%
declar : type ID tab ';'
       ;

type : CHAR
     | INT
     | DOUBLE
     | STRUCT '{' list '}'
     ;

list : list declar
     |
     ;

tab : '[' NUM ']'
    |
    ;
%%
```

Afin de réserver l'espace mémoire correct dans la pile lorsqu'un identificateur est déclaré, on veut connaître la taille en octets de chaque variable.
Ajoutez dans le fichier `yacc` des règles de calcul d'attributs qui permettent de connaître la taille en octets d'une variable déclarée selon les règles de cette grammaire. Lorsque l'analyseur a reconnu une déclaration de variable, il devra afficher le nombre d'octets nécessaires à l'initialisation de cette variable (on ne demande pas de faire l'allocation mémoire).
On considèrera qu'un entier ou un caractère occupe 1 octet, un réel double occupe 4 octets. Une variable de type structure occupe la place nécessaire pour tous les champs de la structure. Une variable de type tableau de taille n occupe la place nécessaire pour les n cases du tableau.
On supposera que l'attribut du token `NUM` a été initialisé avec sa valeur dans le fichier `lex`.

Exercice 5.4

Une grammaire simplifiée des déclarations en langage C stipule qu'une déclaration est constituée d'un *type*, suivi d'un *déclarateur* ; un *déclarateur* est un *déclarateur-direct* précédé d'un nombre quelconque (y compris nul) d'étoiles $*$, et un *déclarateur-direct* a l'une des formes suivantes :

nom
(*déclarateur*)
déclarateur-direct ()
déclarateur-direct [taille facultative]

Les deux dernières formes de déclarations concernent respectivement les fonctions et les tableaux.

1. Ecrire la grammaire correspondante dans un fichier `yacc` et tester sur les déclarations suivantes :

```
char **argv
int (*x)[13]
int *y[13]
void *f()
void (*g)()
void (*h)()[5]
```

2. Modifier l'analyseur pour qu'il traduise en français la nature de chacune des variables ainsi déclarées, en employant les mots "fonction", "pointeur" et "tableau". Par exemple, char $**$argv devra être traduit par "un pointeur sur un pointeur sur un caractère".
   La traduction en français sera sauvegardée dans un attribut de type chaîne de caractères.