Travaux Dirigés Compilation: Feuille 3

Informatique 2ème année. ENSEIRB 2018/2019

—David Janin & Myriam Desainte-Catherine—

▸**Exercice 1.** *Vérification des types*

On considère la grammaire suivante :

$$
\begin{array}{lcl}
S & \rightarrow & DS|D \\
D & \rightarrow & BL\texttt{;} \\
B & \rightarrow & \texttt{int} \\
 & & |\texttt{float} \\
 & & |\texttt{int *} \\
 & & |\texttt{float *} \\
L & \rightarrow & \texttt{id}\ VL \\
 & & |\texttt{id} \\
V & \rightarrow & \texttt{,}
\end{array}
$$

avec en minuscule, les lexèmes. On souhaite définir pour chaque identificateur son type. On utilisera pour cela une table t stockant pour chaque identificateur son type. On utilisera les fonctions suivantes sur cette table :

— *$ajoute(t, id, type)$: insère dans la table t le nom de l'identificateur id, avec le type $type$.*

— *$type(t, id)$: retourne le type de l'identificateur id. S'il n'a pas été défini, retourne la valeur $undef$.*

Cette table s'appelle une table des symboles.

1. *Ecrire l'arbre syntaxique pour* `int a, b; int * c; float d;`
2. *Définir en langage algorithmique pour chaque règle de la grammaire les actions sémantiques permettant d'insérer dans la table des symboles t les identificateurs avec leur type. Pour ce faire, on utilisera la syntaxe YACC pour les attributs synthétisés, $$, $1, $2, etc.* (1) *Tester la correction de vos actions sémantiques sur les examples précédents.*
3. *On ajoute à la grammaire les règles suivantes :*

$$
\begin{array}{lcl}
S & \rightarrow & IS|I \\
I & \rightarrow & \texttt{id = } E\texttt{;} \\
E & \rightarrow & T + E|T - E|T \\
T & \rightarrow & F * T|F \\
F & \rightarrow & id|n|f|(E)
\end{array}
$$

   avec n un lexème représentant une valeur entière et f un lexème représentant un float. Les opérations autorisées en fonction des types sont les suivantes :

   — *entre deux ints : $+$, $-$, $*$ donnent un int*

   — *entre deux floats ou un int et un float : $+$, $-$ et $*$ donnent un float*

   — *entre deux pointeurs de même type : $-$ donne un int. Les autres opérations ne sont pas autorisées.*

   — *entre un pointeur et un int : $+$ et $-$ donnent un pointeur du même type. Les autres opérations avec pointeur ne sont pas autorisées.*

   Pour chaque opérateur, faire la table donnant pour chaque type d'opérande, le type du résultat ou $undef$.
4. *Ajouter en langage algorithmique des actions sémantiques calculant le type des expressions et vérifiant que les affectations sont toutes valides. On supposera qu'on ne peut donner à un identificateur qu'une valeur correspondant à son type. Les identificateurs doivent tous avoir été préalablement déclarés.*

(1). Pour la communication transversale à l'arbre de dérivation, on pourra utiliser l'attribut $0 qui désigne l'attribut du noeud précédent le non terminal gauche de la règle traitée dans l'arbre de dérivation. Par example, on pourra faire l'action

$$V \rightarrow \texttt{,}\ \{\texttt{\$\$ = \$-1;}\}$$

pour recopier, selon l'arbre de dérivation, l'attribut du premier T ou V à gauche du V courant.

▸**Exercice 2.** *Graphe de dépendances*

On considère la même grammaire que l'exercice précédent (avec types et affectations).

On souhaite construire le graphe de dépendance entre variables des programmes définis par cette grammaire. Il y a une dépendance d'un identificateur vers un autre si le premier est utilisé dans l'expression affectée au second.

L'intérêt d'un graphe de dépendance est important : il permet de changer l'ordre d'exécution des instructions, tout en respectant la sémantique (utile pour vectoriser, remplir les unités fonctionnelles, éviter les bulles dans le pipeline, mieux allouer les registres...). Les compilateurs ont tous une représentation d'un graphe de dépendance.

1. *Ajouter en langage algorithmique des actions sémantiques permettant de calculer un attribut $utilise$ donnant la liste des identificateurs utilisés dans une expression. On utilisera la fonction $creeliste(id)$ qui retourne une liste avec id comme unique élément et $concatene(liste2, liste3)$ qui retourne la liste qui est la concaténation des $liste2, liste3$.*
2. *Ajouter les actions sémantiques permettant de construire le graphe de dépendance. On utilisera la fonction* `ajoute-arc`$(g, id1, id2)$ *qui ajoute à un graphe g un arc $id1 \to id2$.*