Fiche de TD no. 9

Le langage EQN

EQN est un langage de composition de texte permettant d'écrire des formules mathématiques. Chaque formule est contenue dans une boîte virtuelle. Une boîte peut être en indice d'une autre. Dans ce cas le corps de caractères (taille) de la boîte indice est plus petit. De même pour une boîte en exposant. La hauteur d'une boîte (**B.ht**) dépend
– de la hauteur normale des caractères : **texte.hn**,
– du corps de caractères : **B.cc**,
– des indices ou exposants à l'intérieur.
Voici un grammaire attribuée pour le langage EQN :

Production	Règle sémantique
$S \rightarrow B$	$B.cc := 10$ $S.ht := B.ht$
$B \rightarrow B\ B$	$B_1.cc := B.cc$ $B_2.cc := B.cc$ $B.ht := max(B_1.ht, B_2.ht)$
$B \rightarrow B$ `sub` B	$B_1.cc := B.cc$ $B_2.cc := diminue(B.cc)$ $B.ht := position(B_1.ht, B_2.ht)$
$B \rightarrow$ `texte`	$B.ht :=$ `texte`$.hn * B.cc$

Exercice 1. En utilisant des marqueurs, transformez cette grammaire de façon a pouvoir repérer les attributs hérités dans la pile dans une traduction ascendante.
Procédez comme suit :

1. Ajoutez un marqueur distingué avant chaque occurrence de B à la droite d'une production.
2. Engendrez des règles de copie de l'attribut hérité *cc* pour les marqueurs.
3. Modifiez les règles de calcul des attributs hérités pour B, comme hérités à partir de son frère aîné, le marqueur.
4. Effacez les marqueurs et les règles que vous pensez être inutiles.
5. Transformez les règles sémantiques ainsi calculée en des règles qui calculent les attributs par rapport au somment de la pile (et au nouveau sommet de la pile).

Types et contrôle de type

Exercice 2. Écrivez des expressions de type pour les types suivants :
– un tableau de pointeurs vers des réels, de taille 100,
– un tableau à deux dimensions, de taille 9×10, dont les éléments appartiennent au type précédent.

Exercice 3. Écrivez un programme `.tic` pour définir la syntaxe abstraite de la grammaire de type présentée en cours :

$$T \rightarrow \texttt{bool} \mid \texttt{char} \mid \texttt{int} \mid \texttt{tableau}\,[\,\texttt{nb}\,]\,\texttt{de}\,T \mid \uparrow T$$

Exercice 4. Considérez les déclarations C suivantes :

```
typedef struct {
  int a,b;
} CELLULE, *PCELLULE;

CELLULE toto [100];
PCELLULE truc (int x, CELLULE y);
```

Écrivez des expressions de type pour le type de `toto` et `truc`.

Exercice 5. Pour chacune des programmes suivants, construisez son arbre de dérivation et étiquetez chaque noeud de l'arbre par son expression de type :

1. ```
   c : caractere ; i : entier ;
   c mod i mod 3
   ```

2. ```
   p : ^entier; a : tableau [10] de entier ;
   a [ p^ ]
   ```

3. ```
   f : entier -> bool;
   i : entier; j : entier; k : entier;
   tant que f ( i) faire
   k:= i ;
   i:= j mod i;
   j:= k;
   ```

Exercice 6. Trouver un unificateur des deux expressions de type

$$pointeur(\alpha) \times (\beta \to \gamma) \qquad\qquad \beta \times (\gamma \to \delta)\,.$$
   ```