5.3.4 Exemples

Exemple 2

Expressions arithmétiques postfixées, un attribut synthétisé. Il s'agit du calcul du nombre de registres nécessaires à une évaluation.

(1) $S^0 \to S^1 S^2 b$
$S^0.\alpha := \Phi$[4]
(2) $S \to a$
$S.\alpha := 0$

Exemple 2′

On repart de l'exemple précédent, on ajoute une règle (3), et un attribut hérité (β).

(1) $S^0 \to S^1 S^2 b$
$S^0 := \Phi$
$S^1.\beta := S^0.\beta + 1$
$S^2.\beta := S^0.\beta \times 2$
(2) $S \to a$
$S.\alpha := S.\beta$
(3) $S' \to S$
$S.\beta := 1$

Exemple 2″

Variante de l'exemple précédent (seul changement : 3e règle associée à (1)), où le calcul est plus compliqué : on ne peut pas d'abord calculer tous les β, puis tous les α.

(1) $S^0 \to S^1 S^2 b$
$S^0 := \Phi$
$S^1.\beta := S^0.\beta + 1$
$S^2.\beta := S^1.\alpha \times 2$
(2) $S \to a$
$S.\alpha := S.\beta$
(3) $S' \to S$
$S.\beta := 1$

On voit bien sur ce exemple la difficulté de prédire qu'un système d'attributs donné sera calculable.

Exemple 3

Deux attributs synthétisés (α et γ), et un attribut hérité (β).

(1) $S^0 \to S^1 T b$
$S^0.\alpha := max(S^1.\alpha, T.\alpha)$
$S^0.\gamma := S^1.\gamma + T.\gamma$
$S^1.\beta := S^0.\alpha$
$T.\beta := S^0.\alpha + S^0.\beta$
(2) $T^0 \to T^1 c$
$T^0.\alpha := T^1.\alpha + 1$
$T^0.\gamma := T^1.\gamma + 1$
$T^1.\beta := T^0.\beta$
(3) $T \to c$
$T.\alpha := 1$
$T.\gamma := T.\beta$
(4) $S \to a$
$S.\alpha := 2$
$S.\gamma := S.\beta + 1$

Exemple 4

Exemple de système impossible à calculer.

(1) $S^0 \to S^1 S^2 b$
$S^0.\alpha := S^1.\alpha + S^2.\alpha$
$S^1.\beta := S^0.\beta$
$S^2.\beta := S^0.\beta$
(2) $S \to a$
$S.\alpha := S.\beta$
(3) $S' \to S$
$S.\beta := 2 \times S.\alpha$

[4] On note Φ l'expression :
$\left[(S^1.\alpha == S^2.\alpha) \ ? \ S^1.\alpha + 1 \ : \ max(S^1.\alpha, S^2.\alpha)\right]$

5.4 Test de circularité, algorithme d'évaluation d'attributs

Remarque On peut avoir des systèmes d'attributs calculables pour certains arbres de dérivation et pas pour d'autres ; dans l'exemple précédent le calcul est impossible sur tout arbre de dérivation.

Comment établir formellement l'impossibilité du calcul ?

5.4.1 Test de circularité

On associe à chaque arbre syntaxique dans une grammaire attribuée donnée, un graphe : chaque nœud du graphe est une valeur d'attribut (autrement dit, chaque nœud (branchant) de l'arbre correspond à k sommets du graphe, s'il y a k attributs). Le graphe est orienté, on trace un arc du sommet s vers le sommet t ssi le calcul de t a besoin de la valeur de s (les flèches vont dans le sens de la propagation des valeurs[5]). On remarque que pour les attributs synthétisés, les arcs vont des fils vers le père (ou dans le sommet lui-même), alors que pour les attributs hérités, les arcs vont du père vers les descendants (ou dans le sommet lui-même).

Théorème Le calcul d'attributs est possible pour le mot $f \in \mathcal{L}(\mathcal{G})$ ssi le graphe associé à son arbre de dérivation est sans circuit.

Remarque Dans la pratique, ce théorème n'est pas intéressant : on a besoin de savoir si un système d'attribut est calculable pour **tous** les arbres de dérivation.
Il existe des versions plus « fortes » de ce théorème, qui permettent de vérifier qu'un système est calculable en étudiant un nombre fini d'arbres.

5.4.2 Algorithme

Rappel Propriété des graphes sans circuit : il existe au moins un sommet sans prédécesseur. Par ailleurs, on sait aussi que si on ôte un sommet à un graphe sans circuit, il reste sans circuit. Il est donc possible de munir l'ensemble des sommets d'un ordre partiel (« tri topologique »).

C'est de l'algorithme classique de tri topologique que l'on s'inspire pour proposer un algorithme de parcours du graphe associé à un arbre de dérivation, qui permet de calculer les attributs :

$\mathcal{G}$ = graphe associé initial ;
Tant que $\mathcal{G}$ non vide {
    Calculer la valeur des attributs sur le(s) sommet(s) sans prédécesseur ;
    $\mathcal{G} := \mathcal{G} \setminus \{$sommet(s) sans prédécesseur$\}$
}

Exercice Faire le graphe associé à l'exemple 2″.

[5] On peut aussi concevoir le graphe « inverse », dans lequel les flèches représentent la « dépendance » : arc de a vers b si a a besoin de b.

5.5 Génération de code avec des grammaires attribuées

L'idée est simple : l'un des attributs va contenir le code généré petit-à-petit (synthétisé), et on va ajouter d'autres attributs pour les informations nécessaires *at compile time*.

5.5.1 Exemple 1 : ETF

On introduit deux attributs : `nb` pour le nombre de registres nécessaires, et `code` pour le code lui-même. Ils sont tous deux synthétisés.[6]

$$
\begin{array}{lcll}
F \to a_i & : & F.\texttt{code} & := \texttt{LOAD } i \texttt{ R1} \\
 & & F.\texttt{nb} & := 1 \\
F \to (\,E\,) & : & F.\texttt{code} & := E.\texttt{code} \\
 & & F.\texttt{nb} & := E.\texttt{nb} \\
T \to F & : & T.\texttt{code} & := F.\texttt{code} \\
 & & T.\texttt{nb} & := F.\texttt{nb} \\
E \to T & : & E.\texttt{code} & := T.\texttt{code} \\
 & & E.\texttt{nb} & := T.\texttt{nb} \\
E \to E + T & : & E^0.\texttt{code} & := \left[\begin{array}{l} (E^1.\text{nb} \geq T.\texttt{nb}) \;\; ? \\ E^1.\texttt{code}\,;\,\mathcal{T}(T.\texttt{code})\,;\,\texttt{ADD R1 R2} \;\; : \\ \text{T.code}\,;\,\mathcal{T}(E^1.\texttt{code})\,;\,\texttt{ADD R1 R2} \end{array}\right] \\
 & & E^0.\texttt{nb} & := \Big[(E^1.\text{nb} == T.\texttt{nb})\;?\;E^1.\texttt{nb}+1 : \; max(E^1.\text{nb},T.\texttt{nb})\Big] \\
T \to T \times F & : & T^0.\texttt{code} & := \left[\begin{array}{l} (T^1.\text{nb} \geq F.\texttt{nb}) \;\; ? \\ T^1.\texttt{code}\,;\,\mathcal{T}(F.\texttt{code})\,;\,\texttt{MUL R1 R2} \;\; : \\ \text{F.code}\,;\,\mathcal{T}(T^1.\texttt{code})\,;\,\texttt{MUL R1 R2} \end{array}\right] \\
 & & T^0.\texttt{nb} & := \Big[(T^1.\text{nb} == F.\texttt{nb})\;?\;T^1.\texttt{nb}+1 : \; max(T^1.\text{nb},F.\texttt{nb})\Big]
\end{array}
$$

Code « translaté » : Décalage des numéros de registre.
Par exemple, $\mathcal{T}(\texttt{LDA A R1}) = \texttt{LDA A R2}$.

Exercice Reprendre l'exemple donné plus haut : $2 \times 3 + 4$.

5.5.2 Exemple 2 : `if`, `while` et les booléens

à voir en cours.

[6] `LOAD` = `LDA` = `STO`.