

# Compilateurs & Interprètes

## *Arbres d'expressions arithmétiques (suite)*

Octobre 2019

### But

Construction d'un arbre de dérivation pour évaluer une formule d'expression arithmétique contenant des variables.

### Enoncé

On rajoute à la grammaire de la série précédente quelques règles pour définir une formule contenant des variables :

$PROG \rightarrow LISTVAR FORM$

$LISTVAR \rightarrow DECLVAR LISTVAR$

$DECLVAR \rightarrow \# id = nb$

$LISTVAR \rightarrow \varepsilon$

$FORM \rightarrow E$

$E \rightarrow T D$

$D \rightarrow + E$

$D \rightarrow \varepsilon$

$T \rightarrow F G$

$G \rightarrow * T$

$G \rightarrow \varepsilon$

$F \rightarrow ( E )$

$F \rightarrow nb$

$F \rightarrow id$  (règle ayant été rajoutée)

L'axiome est la première règle. Elle nous dit qu'un programme est donné par une liste de variables et une formule. Les symboles non-terminaux sont en majuscules et les symboles terminaux sont en minuscules. Le symbole terminal *id* désigne un identifiant de variable qui devra commencer obligatoirement par une lettre de l'alphabet. Le symbole terminal *nb* indique un nombre réel et enfin le symbole  $\varepsilon$  désigne le mot vide (pour forcer l'arrêt de la récursivité).

Ecrire un programme qui interprète un *PROG* défini par cette grammaire. Cette fois il faudra construire une table des symboles par rapport aux variables qui auront été spécifiées. Tester l'interpréteur en écrivant plusieurs formules.

**Le listing de ce travail pratique est à rendre au plus tard le 11 novembre 2018. Il pourra être réalisé par groupe de 2 personnes ; une démonstration sera effectuée au laboratoire à l'enseignant.**