Université de Paris XIII (USPN) — L2 N3 info-DL
Institut Galilée — 2023-24

Structures de données et algorithmes

Feuille de TD-TP n°4

Expressions arithmétiques et arbres syntaxiques

Exercice 1. Expressions arithmétiques préfixes

Soit l'alphabet $A = \{+, *, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Un *mot* sur A est une suite finie $a_0 a_1 \cdots a_{n-1}$, $n \geqslant 0$, telle que $a_i \in A$ pour tout $i \in \{0, 1, \dots, n-1\}$; l'entier n est appelé la *longueur* du mot. L'ensemble des mots sur A est noté A^* et le mot vide, c'est-à-dire l'unique mot de longueur 0, est noté ϵ.

On définit l'ensemble $\mathcal{E}_A$ par induction : c'est le plus petit sous-ensemble $\mathcal{F}$ de A^* qui satisfasse les propriétés suivantes :

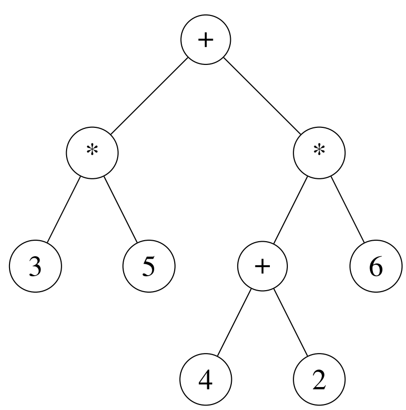
1. pour tout $x \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $x \in \mathcal{F}$;
2. pour tous $e, f \in A^*$, si $e \in \mathcal{F}$ et $f \in \mathcal{F}$, alors $+ef \in \mathcal{F}$ et $*ef \in \mathcal{F}$.

Par exemple, les mots 5, $+73$, $**123$ et $+*35*+426$ sont des éléments de $\mathcal{E}_A$, alors que ϵ, $+$, $3*4$ et $+*217+57$ n'en sont pas.

Un mot de $\mathcal{E}_A$ peut être « lu » comme une expression arithmétique en écriture préfixe [1], construite à partir de constantes entières comprises entre 0 et 9 [2]. On peut facilement reconstruire l'écriture infixe, plus traditionnelle : par exemple, $((3*5)+((4+2)*6))$ est l'écriture infixe correspondant à l'expression préfixe $+*35*+426$ [3].

$a)$ La valeur d'un mot de $\mathcal{E}_A$ est la valeur numérique de l'expression arithmétique qu'elle représente. Définissez par induction la valeur d'un élément de $\mathcal{E}_A$.

À chaque mot de $\mathcal{E}_A$ on associe *l'arbre syntaxique* de l'expression arithmétique qu'il représente. C'est l'arbre binaire dont les feuilles sont étiquetées par des chiffres décimaux et les autres sommets par des symboles d'opération, **construit de telle sorte que le parcours préfixe de l'arbre redonne l'écriture préfixe de l'expression**. Par exemple, l'arbre syntaxique de l'expression arithmétique préfixe $+*35*+426$ est



1. Dans l'écriture préfixe, le symbole d'opération (addition ou multiplication) précède les deux opérandes.
2. Cette restriction permet d'éviter toute ambiguïté dans la lecture des éléments de $\mathcal{E}_A$. Pour autoriser des constantes entières naturelles arbitraires sans créer d'ambiguïté, plusieurs solutions sont envisageables : ajouter à A un symbole pour chaque valeur constante ou ajouter à A un symbole permettant de « séparer » deux constantes qui se suivent dans une expression arithmétique préfixe, sans quoi on ne saurait pas si $+217$ doit être lu $(2+17)$ ou $(21+7)$.
3. Les parenthèses sont indispensables pour l'écriture infixe, alors qu'on peut s'en passer pour l'écriture préfixe.

Dans la suite de l'exercice, on suppose que `item` redéfinit le type `char` et qu'on peut appeler toute fonction de la bibliothèque `binary_tree.h` (cf. les feuilles de TD et de TP n°3 et la feuille ci-jointe).

b) Écrivez la définition d'une fonction `parse_expr` qui reçoit en entrée une expression arithmétique préfixe, *i.e.* une chaîne de caractère implémentant un mot de $\mathcal{E}_A$; elle construit l'arbre syntaxique de cette expression, en allouant la mémoire nécessaire sur le tas, et renvoie son adresse. Dans l'horrible franglais des informaticiens, on dit parfois que l'expression est « parsée » par (ou dans) son arbre syntaxique. [Indication. La fonction `parse_expr` pourra en appeler une autre, qui construira récursivement l'arbre syntaxique à partir de ceux associés à des sous-expressions.]

c) Écrivez la définition d'une fonction `eval_tree` qui reçoit en entrée l'adresse d'un arbre syntaxique, *i.e.* l'adresse d'un arbre binaire obtenu en « parsant » une expression arithmétique ; elle renvoie la valeur de l'expression arithmétique.

d) Écrivez la définition d'une fonction `tree_to_expr` qui reçoit en entrée l'adresse d'un arbre syntaxique, *i.e.* l'adresse d'un arbre binaire obtenu en « parsant » une expression arithmétique ; elle construit et renvoie la chaîne de caractères implémentant l'expression arithmétique « parsée » par l'arbre, après avoir alloué la mémoire nécessaire sur le tas.

Exercice 2. Bonus

a) Écrivez la définition d'une fonction `eval_expr` qui reçoit en entrée une expression arithmétique préfixe, *i.e.* une chaine de caractère implémentant un mot de $\mathcal{E}_A$; elle renvoie la valeur de l'expression arithmétique **sans construire son arbre syntaxique**.

b) On souhaite généraliser la définition d'une expression arithmétique préfixe en autorisant des constantes entières prenant leur valeur dans $\mathbb{N}$. Pour ce faire, on ajoute à l'alphabet A un symbole permettant de « séparer » les constantes, par exemple un symbole « espace ». On peut alors avoir des expressions comme $+ * 5\ 10 * 12\ 3$, dont l'écriture infixe correspondante est sans ambiguïté $((5*10)+(12*3))$. Modifiez la définition de la fonction `eval_expr` de la question précédente de façon à calculer et renvoyer la valeur d'une expression arithmétique généralisée reçue en entrée.