

# Série d'exercices #1

IFT-2035

4 mai 2021

## 1.1 Préfixe, postfixe et ASA

Pour chaque expression infixe ci-dessous, réécrire l'expression en notation préfixe et postfixe. Dessiner également l'arbre de syntaxe abstraite (ASA).

1.  $a + b + c$
2.  $a + (b + c)$
3.  $a \cdot b + c \cdot d$
4.  $a + b < a \cdot (c + d)$
5.  $(-b + \text{sqrt}(b \cdot b - 4 \cdot a \cdot c)) / (2 \cdot a)$

## 1.2 Postfixe et machine à pile

La notation postfixe s'évalue facilement à l'aide d'une pile. L'algorithme général est :

1. Lire l'expression de gauche à droite.
  - (a) S'il s'agit d'un nombre, l'empiler.
  - (b) S'il s'agit d'un opérateur :
    - i. dépiler le nombre correspondant de valeurs du sommet de la pile ;
    - ii. calculer le résultat ;
    - iii. et l'empiler.
2. Lorsque la lecture est terminée, le résultat est au sommet de la pile.

Illustrer cet algorithme avec les expressions de la section ??.

### 1.3 Si et seulement si

Voici une grammaire pour `if...then...else`. Les éléments  $E$  et  $X$  représentent des expressions et parties de la grammaire qu'il n'est pas important de spécifier ici.

$$\begin{array}{lcl} S & ::= & X \\ & | & \text{"if" } E \text{ "then" } S \\ & | & \text{"if" } E \text{ "then" } S \text{ "else" } S \end{array}$$

Cette grammaire est ambiguë.

1. Donner un exemple d'ambiguïté.
2. Donner une grammaire non ambiguë qui associe les `else` avec le `if` le plus proche, comme le font les langages de programmation habituels.

### 1.4 Ambiguïté et récursion

Soit la grammaire suivante pour des expressions arithmétiques

$$\begin{array}{lcl} \text{expr} & ::= & \text{expr} + \text{expr} \\ & | & \text{expr} * \text{expr} \\ & | & \text{number} \end{array}$$

1. Montrer que cette grammaire est *ambiguë*.
2. Réécrire cette grammaire de manière à éliminer les ambiguïtés.
3. Cette grammaire est *récursive à gauche*, ce qui pose problème pour certaines techniques d'analyse syntaxique : En effet, dans un parseur avec analyse descendante (top down), la portion du programme devant lire la catégorie *expr* va devoir d'abord faire appel à la portion du programme qui doit lire la catégorie *expr* ...  
Réécrire la grammaire de manière à éviter cette *récursion à gauche*.