Série d'exercices #1

IFT-2035

September 2, 2024

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. $(-b + \mathsf{sqrt}(b \cdot b - 4 \cdot a \cdot c))/(2 \cdot a)$

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 1.1.

1.3 Conversion de base

Écrire les fonctions suivantes en Haskell pour convertir des nombres en représentation binaire à décimal et vice versa (à remarquer que l'argument est un entier et que par exemple l'entier "cent-un" représente le nombre binaire "un-zéro-un" qui correspond à 5 en décimal). Vous aurez besoin des fonctions prédéfinies mod, et div.

$$\textit{bin2dec}\ 10001 \quad \rightsquigarrow^* \quad 17$$
$$\textit{dec2bin}\ 17 \quad \rightsquigarrow^* \quad 10001$$

Écrire la fonction *baseconv* en Haskell qui convertit d'une base à une autre (≤ 10). N'hésitez pas à définir des fonctions auxiliaires si nécessaire.

$$\textit{baseconv}\ 2\ 10\ 10001 \quad \rightsquigarrow^* \quad 17$$
$$\textit{baseconv}\ 10\ 2\ 17 \quad \rightsquigarrow^* \quad 10001$$

Donner aussi le type de chacune des fonctions que vous avez définies.

Note: La distinction entre un objet et sa représentation est un thème qui réapparait souvent dans ce cours, par exemple sous la forme de la différence entre la syntaxe et la sémantique des programmes.