Feuille d'exercices n°6 - Récursivité : introduction

Notions abordées

- définition récursive de fonctions déjà vues : puissance, factorielle, pgcd...
- méthode pour étudier la complexité d'une fonction récursive
- arbre de syntaxe (expressions arithmétiques, booléennes, appels de fonctions)
- notion d'appel récursif terminal

Définir de manière récursive

Toutes les fonctions de cette partie seront reprises dans l'exercice 8, si vous ne voulez pas avoir à recopier les définitions, laissez une demi-page après chacune d'elles.

Exercice 1 Factorielle

Question 1
Proposer une définition récursive de la factorielle d'un entier naturel.

Question 2
Établir une relation de récurrence vérifiée par la suite u_n qui à un entier n associe le nombre d'appels récursifs de la définition proposée appelée initialement sur l'entier n. Préciser aussi les termes initiaux de cette suite. En déduire le nombre d'appels récursifs nécessaires pour calculer $n!$ avec la définition proposée.

Exercice 2 Puissance

Question 1
Proposer une définition récursive de l'exponentiation d'un réel à une puissance entière et positive. *On attend ici une fonction naïve.*

Question 2
Établir une relation de récurrence vérifiée par la suite u_n qui à un entier n associe le nombre d'appels récursifs de la définition proposée appelée initialement sur un réel quelconque x et l'entier n. Préciser aussi les valeurs initiales de cette suite. En déduire le nombre d'appels récursifs nécessaires pour calculer x^n avec la définition proposée.

Question 3
Proposer une meilleure définition récursive de l'exponentiation d'un réel à une puissance entière et positive, c'est-à-dire une définition qui réaliserait moins d'appels récursifs que la précédente. Comme

précédemment, donner le nombre d'appels récursifs en fonction de l'exposant. *On attend ici une fonction équivalente à* `puissance_rapide`.

Exercice 3 Division euclidienne

Question 1
Proposer une définition récursive du quotient d'un entier naturel par un entier naturel non nul.

Question 2
Proposer une définition récursive du reste d'un entier naturel par un entier naturel non nul.

Question 3
Soit $(a, b) \in \mathbb{Z} \times \mathbb{Z}^*$.
On note q (resp. r) le quotient (resp. le reste) de a par b.
On note q' (resp. r') le quotient (resp. le reste) de $a' = |a|$ par $b' = |b|$.
En distinguant les trois cas suivants, exprimer q et r en fonction de a, b, q' et r'.

- $a \leqslant 0$ et $b > 0$
- $a \geqslant 0$ et $b < 0$
- $a \leqslant 0$ et $b < 0$

Question 4
En déduire une définition récursive de la division euclidienne (*i.e.* le couple quotient-reste) d'un entier quelconque par un entier non nul.

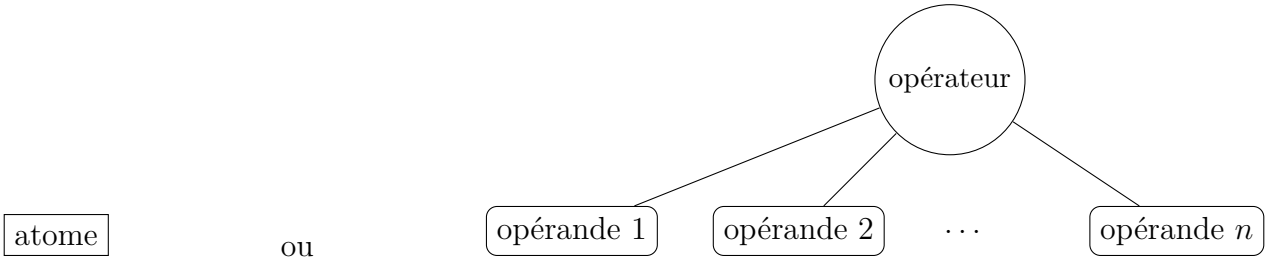
Exercice 4 PGCD

Question 1
Proposer une définition récursive du PGCD de deux entiers naturels non identiquement nuls. *On attend ici une définition analogue de l'algorithme d'Euclide.*

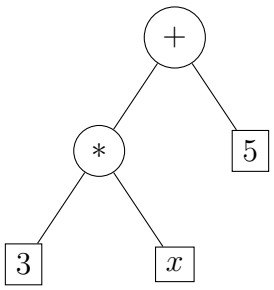
Question 2
Proposer une autre définition récursive du PGCD de deux entiers naturels non identiquement nuls. *On attend ici une définition analogue à* `pgcd_bis`.

Détecter si un appel récursif est terminal

En général, une expression peut être l'écriture d'une simple valeur, on parle alors d'**expression atomique**, ou composée à partir d'autres expressions (alors appelées opérandes (n.m)) grâce à des opérateurs, on parle alors d'**expression composée**.
Cette construction des expressions permet de les représenter sous forme d'arbre (appelé **arbre de syntaxe**) dont les nœuds internes sont les opérateurs (représentés dans des cercles) et les feuilles sont les expressions atomiques (représentés dans des rectangles). Ainsi l'arbre de syntaxe d'une expression simple est réduit à une feuille, tandis que l'arbre d'une expression composée est obtenu en inscrivant l'opérateur "principal" dans un cercle, puis en dessinant en dessous les arbres de chaque opérande, et en reliant l'opérateur à chacun de ses opérandes par un trait appelé arête.



Par exemple, l'expression $3x + 5$ est l'addition des termes $3x$ et 5, qui sont respectivement la multiplication du nombre 3 et de la variable x, et le nombre 5. L'arbre de syntaxe associé est donc le suivant.



Exercice 5 Arbre de syntaxe niveau 1

Une **expression arithmétique** est un nombre (par ex. 3, π, 55.2...), ou une variable représentant un nombre (par ex. x, y, i, j, α...), ou une expression obtenue en combinant d'autres expressions arithmétiques grâce aux opérateurs binaires d'addition, de soustraction, de multiplication, de division réelle, de quotient ou de reste (qu'on notera dans cet exercice comme en C), et à l'opérateur unaire de passage à l'opposé.

Question 1
Donner l'arbre de syntaxe des expressions arithmétiques suivantes :

1. $3 + 4 \times 8$
2. $(3 + 4) \times 8$
3. $3 + (4 \times 8)$
4. $(-1) \times 2$
5. $-(1 \times 2)$
6. 58
7. $(2 * 3) \bmod 10$

Une **expression booléenne**[1] est quant à elle une valeur explicite (vrai ou faux), ou une variable booléenne, ou une expression obtenue en combinant d'autres expressions booléennes grâce aux opérateurs binaires de conjonction, disjonction et l'opérateur unaire de négation (qu'on notera Et, Ou, Non dans cet exercice).

Question 2
Donner l'arbre de syntaxe des expressions booléennes suivantes :

1. vrai
2. A Et B
3. Non (A ou B)
4. (A ou C) Et (A ou B) Et (A ou D)[2]

Exercice 6 Arbre de syntaxe niveau 2

On s'intéresse dans cet exercice à des expressions de type `bool` écrites en C. Celles-ci peuvent faire intervenir des expressions arithmétiques combinées par des opérateurs de comparaisons (`==`, `<=`, ...).

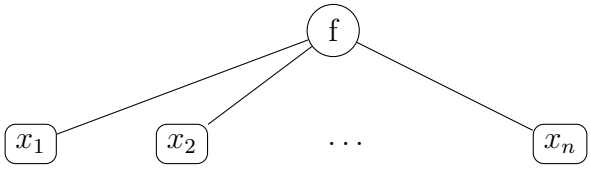
Question 1
Donner l'arbre de syntaxe des expressions suivantes :

1. `(x%2) == 0`
2. `(n < 0) || ((n == 0) && (m == 2))`
3. `(5*x + 3) != (8*y + 12)`

Exercice 7 Arbre de syntaxe niveau 3

On se propose maintenant de donner un arbre de syntaxe pour des expressions faisant intervenir des appels de fonctions. Il s'agit en fait d'une généralisation de ce qu'on a fait jusqu'ici puisqu'un opérateur binaire (resp. unaire) n'est rien d'autre qu'une fonction à deux arguments (resp. un argument). Ainsi l'arbre de syntaxe d'un appel de fonction est obtenu en inscrivant le nom de la fonction dans un cercle, relié à l'arbre de syntaxe de chacun de ses arguments placés sous ce cercle.

Si f est une fonction d'arité n (*i.e.* à n arguments), l'arbre de syntaxe associé à l'appel $f(x_1, x_2, ..., x_n)$ est donc le suivant.



Question 1
En supposant que f (resp. g) est une fonction d'arité 1 (resp. 2), donner l'arbre de syntaxe des expressions suivantes : 1. $f(3x+4)$ 2. $g(1,2)+g(3,4)$ 3. $g(f(5), f(6))$ 4. $g(g(1,2), g(3,4))$

[1] au sens restreint des formules du calcul propositionnel.

[2] Ici on a omis les parenthèses car le Et est associatif. On peut représenter l'arbre en supposant qu'il y a associativité à gauche, ou bien supposer que Et est un opérateur d'arité quelconque.

Un appel de fonction est dit **terminal** dans une expression s'il est évalué en dernier lors de l'évaluation de cette expression. Pour les opérateurs unaires et binaires considérés jusqu'ici (*i.e.* les opérateurs arithmétiques, booléens et de comparaison), l'ordre d'évaluation est le même : d'abord on évalue chaque opérande, puis on évalue le résultat de l'opération, Si on considère l'évaluation paresseuse du Et (resp. du Ou), un opérande est évalué seulement si l'opérande précédent a été évalué à vrai (resp. faux). Pour un appel de fonction, le principe est le même : on commence par évaluer les opérandes.

Question 2 ✍
Dans les arbres de syntaxes vus jusqu'ici, comment se traduit le fait d'être terminal pour un appel de fonction ?

Question 3 ✍
Dans les arbres de syntaxe de la question précédente, distinguer les appels de fonctions terminaux et non terminaux.

Exercice 8 Arbre de syntaxe niveau 4

Afin de pouvoir représenter toutes les expressions écrites jusqu'ici, on considère un nouvel opérateur que l'on notera `si/alors/sinon`. Le premier opérande de cet opérateur ternaire est une expression de type booléen et les deux autres opérandes sont des expressions d'un même type quelconque.

Question 1
Donner l'arbre de syntaxe des expressions suivantes :

1. $\begin{cases} f(5) \text{ si } x = 3 \\ 0 \quad \text{sinon} \end{cases}$
2. $\begin{cases} 3x + 2 \text{ si } x \mod 2 = 0 \\ 5x + 3 \text{ sinon} \end{cases}$
3. $\begin{cases} \text{vrai} \quad \text{si } a \neq 0 \\ y \text{ Et } z \text{ sinon} \end{cases}$

Question 2 ✍
Selon vous, quel est l'ordre d'évaluation de l'opérateur `si/alors/sinon`.

Question 3 ✍
Pour chaque définition récursive proposée dans la première partie (*i.e.* exercices 1 à 4), donner l'arbre de syntaxe des expressions utilisées, puis identifier les appels récursifs terminaux et non terminaux.

Question 4 ✍
Pour les définitions récursives dans lesquelles apparaît un appel récursif non terminal, proposer une définition alternative ne comportant que des appels récursifs terminaux. Cela peut nécessiter de faire appel à une fonction récursive auxiliaire.