Arbres

I Différents types d'arbres

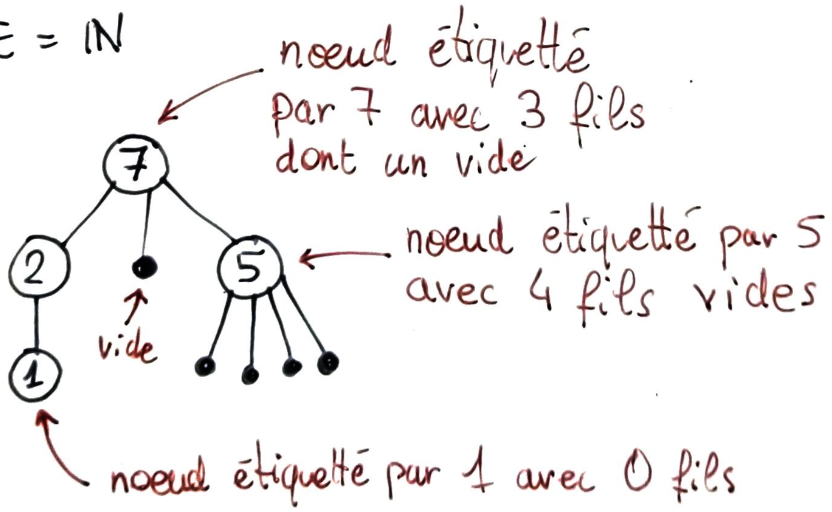
1 Arbres quelconques

def

Soit E un type
L'ensemble des arbres étiquettés par E est le plus petit ensemble tel que

1. L'abre vide est un arbre
2. Pour tout $n \in \mathbb{N}$, et $a_1, \dots, a_n$ des abres et $x \in E$, le noeud étiquettés et de fils $a_1, \dots, a_n$ est un arbre.

ex $E = \mathbb{N}$



def s

1. l'arité d'un noeud son nombre de fils

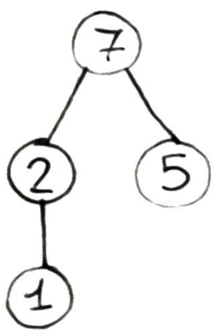
2. Une feuille est un noeud dont tous les fils sont vides

3. La racine d'un arbre non-vide est le seul noeud qui n'a pas de père

notn convention

On ne représente pas les abres vides

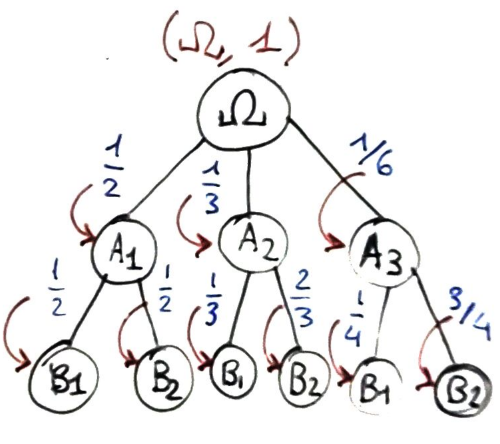
ex



ex d'utilisations

- Arbres de probabilités

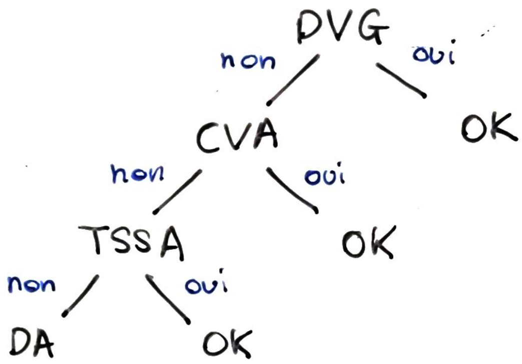
$E = \{\text{évts}\} \times [0, 1]$



pour "étiqueter" les arrêtes, on étiquette les fils avec des couples $(E = \{\text{évts}\} \times [0,1])$

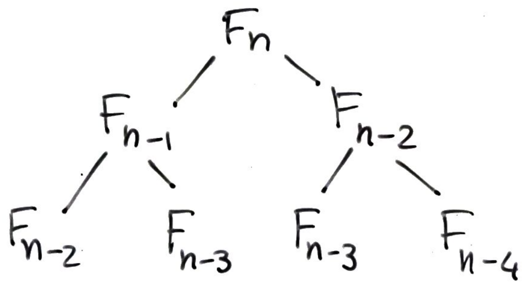
- Arbres de décision

ex

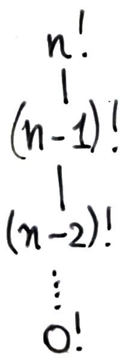


- Arbre des appels récursifs de la pile d'éxécution d'une fonction récursive

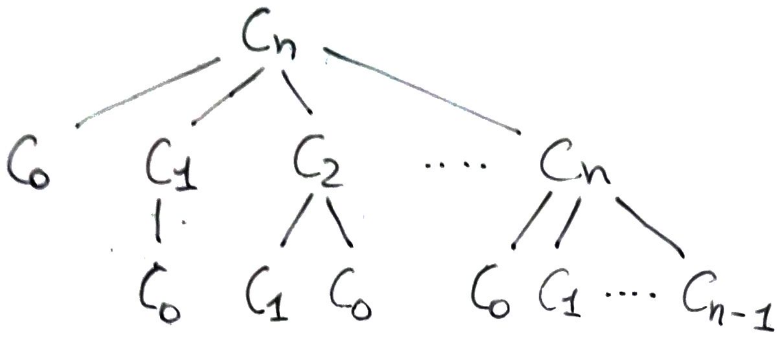
ex Fibonacci naïf



ex $n!$ naïf



ex nombres de Catalan naïf



$$C_{n+1} = \sum_{k=0}^{n} C_k \, C_{n-1-k}$$

implém

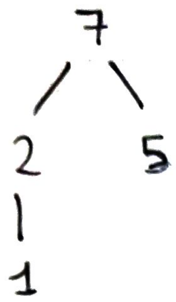
1. (naturelle) écrire un type inductif.

$$\texttt{type 'a tree = Empty | Node of (}\underbrace{\texttt{'a}}_{\text{étiquette}} * \underbrace{\texttt{'a tree list}}_{\text{nombre arbitraire d'abres enfants}}\texttt{)}$$

remq Cela nous donne une structure immuable.
Pour modifier un tree, il faut en créer un nouveau.

2. (muable) avec des tableaux

```
type 'a cell = Empty_cell | Node_cell of 'a * int self
type 'a tree_muable = 'a cell array
```



(7, [1; 2; 3])	(2, [4])	∅	(5, [5; 6; 7; 8])
0	1	2	3

(1, ∅)	∅	∅	∅	∅
4	5	6	7	8

2 Arbres binaires

On se limite aux arbres où chaque noeud est d'arité au plus 2 (en utilisant la convention vide $\Leftrightarrow$ absent)

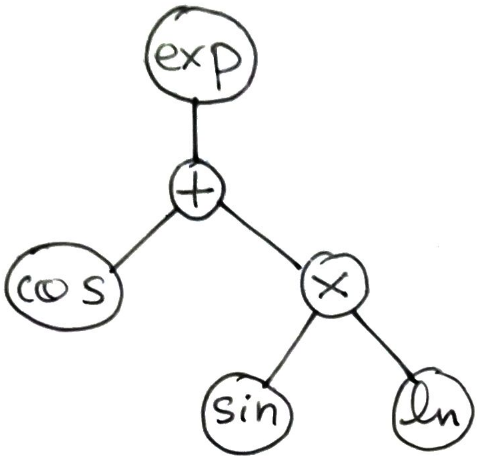
<u>def</u>

Soit E un type.
L'ensemble des arbres binaires est le plus petit ensemble tel que

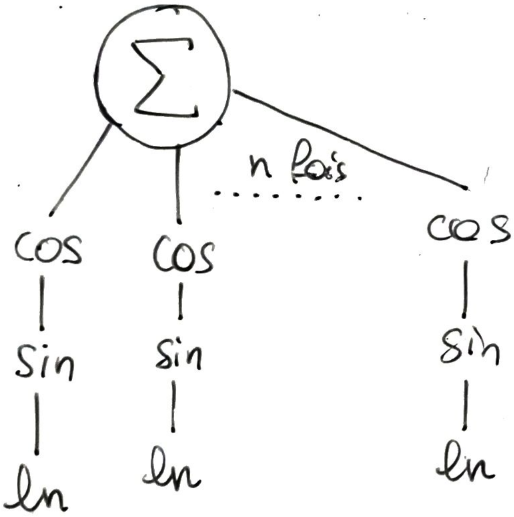
- L'arbre vide est un arbre binaire
- Soient $x \in E$, f_g, f_d deux arbres binaires, le noeud étiqueté par x et de fils gauche f_d et de fils droite f_d

est un arbre binaire

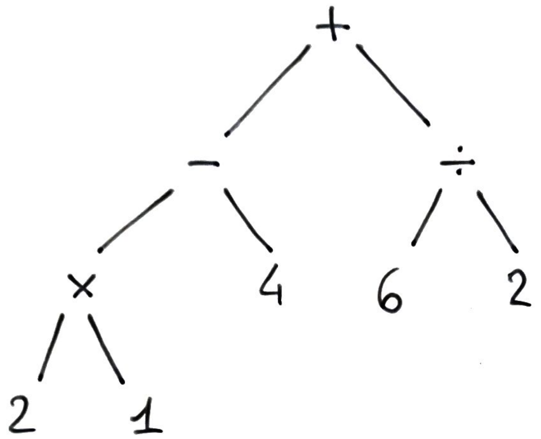
ex dévissage de $x \mapsto e^{\cos x + \sin x \cdot \ln x}$



c-ex dévissage de $x \mapsto \sum_{k=0}^{n} \cos(\sin(\ln x))$



ex expression syntaxique $((2 \cdot 1) - 4) + (6 \div 2))$



$+ - \times 2\ 1\ 4 \div 6\ 2$

notation polonaise directe

implém

type 'a binary_tree = Empty | Node of 'a * 'a binary_tree * 'a binary_tree

pour les expr. syntaxiques

type étiquette = $\underbrace{\text{int}}_{2,1,4,6} \mid \underbrace{\text{int} \to \text{int} \to \text{int}}_{\times,\ +,\ -,\ \div}$

Implém

+	−	×	2	1	4	÷	6	2

L'étiquette de la racine figure à l'indice 0 du tableau

Pour chaque noeud figurant à l'indice k du tableau, on fait figurer son fils $\begin{cases} \text{gauche à l'indice } 2k+1 \\ \text{droite à l'indice } 2k+2 \end{cases}$

3 Arbres binaires entiers

def arbre binaire entier

Un arbre binaire non-vide dans lequels

tout les noeuds ont

- 0 (feuille), ou
- 2

fils.

Ceci permet de typer différemment les feuilles et les autres noeuds

```
type ('a, 'b) abe =
  | Feuille of 'a
  | Noeud of ('b * ('a, 'b) abe * ('a, 'b) abe)
```

ex fonction d'évaluation

II Taille hauteur

1 Définitions

def taille d'un arbre T

$$\#\text{noeuds}(T)$$

def profondeur d'un noeud n

Le nombre de fois qu'il faut descendre en partant de la racine pour arriver à n.

def hauteur d'un arbre T

$$\max \overrightarrow{\text{profondeur}}(\text{noeuds}(T))$$

ex

profondeur

$+ \quad \longleftarrow \quad 0$

$- \quad \div \quad \longleftarrow \quad 1$

$\times \quad 4 \quad 6 \quad 2 \quad \longleftarrow \quad 2$

$\underbrace{2 \quad 1}_{\text{taille } 9} \quad \longleftarrow \quad 3 \quad = \text{hauteur}$

remq la hauteur d'un arbre à une feuille est 0

convention

$$\text{hauteur}(\emptyset) = -1$$

2 Implémentation

Implémenter des fonctions taille et hauteur sur les arbres,
sur les AB et
sur les AB E

implém taille sur AB :

```
let rec taille tree = match tree with
  | Empty              -> 0
  | Node (x, fd, fg)   -> 1 + taille fd + taille fg
```

implém hauteur sur AB

```
let rec hauteur tree = match tree with
  | Empty              -> -1
  | Node (x, fd, fg)   -> 1 + max (hauteur fg) (hauteur fd)
```

implém taille sur ABE

```
let rec taille tree = match tree with
  | Leaf _             -> 1
  | Node (x, fd, fg)   -> 1 + taille fd + taille fg
```

implém hauteur sur ABE

Le cas de base devient Leaf _

implém * sur les arbres quelconques

Exercice

3 Propriétés

Dans le cas des arbres qcq, taille et hauteur n'entretiennent pas de relations spectaculaires.

On s'intéresse au cas des arbres binaires {entiers,}

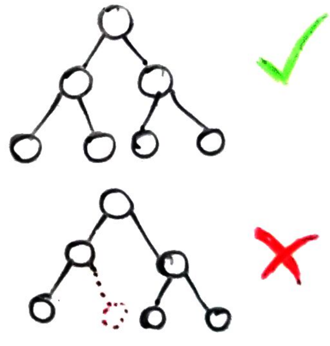
prop

Soit A un arbre de hauteur h et de taille n.

1. $n \in [h+1, 2^{h+1}-1]$
2. A binaire entier $\Rightarrow n \in [2h+1, 2^{h+1}-1]$

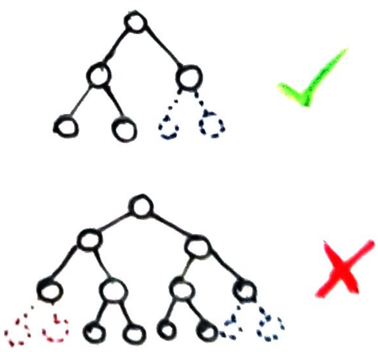
def caractère complet de A

$$\text{taille } A = 2^{\text{hauteur } A + 1} - 1$$



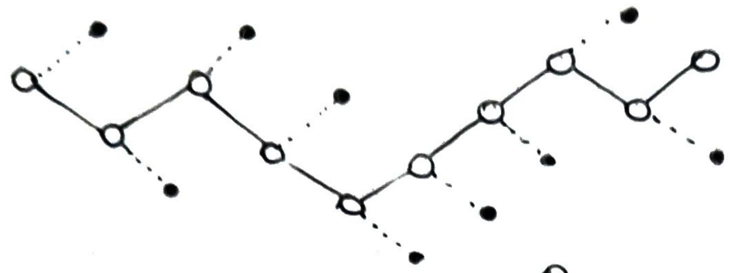
def caractère quasi-complet de A

A complet sauf au dernier niveau où les noeuds sont le plus à gauche possible

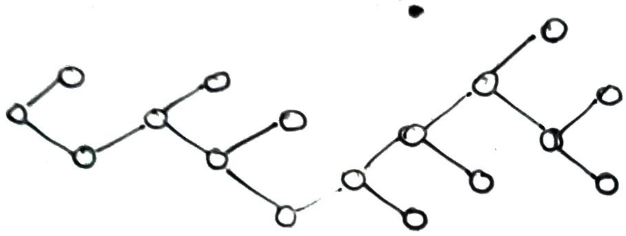


def caractère filiforme de A

$$\begin{cases} \text{taille } A = \text{hauteur } A + 1 \Leftarrow A \text{ pas entier} \\ \text{taille } A = 2\,\text{hauteur } A + 1 \Leftarrow A \text{ entier} \end{cases}$$



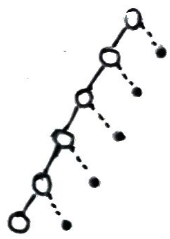
Binaire pas entier



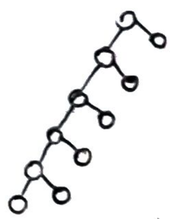
Binaire entier

def peigne

- pour un non-entier
  - à gauche: arbre binaire avec tout ses fils gauches vides
  - à droite: analogue.



- pour un entier
  À gauche/droite: que des feuilles



coroll

Soit A un arbre binaire. Soient $h, n = (\text{hauteur}, \text{taille}) \to A$

1. $\lceil \log_2(n+1) \rceil - 1 \le h \le n - 1$

2. A entier $\Rightarrow h \le \frac{n-1}{2}$

En particulier: $\lfloor \log_2 n \rfloor \le h$

III Parcours d'arbres

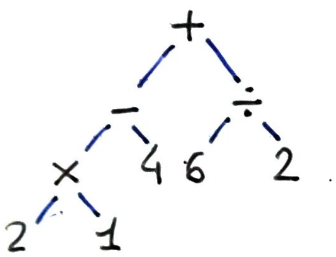
On se limite aux ABE. Parcourir un arbre, c'est énumérer les étiquettes de ses nœuds / feuilles.

1 En profondeur préfixe

def On visite

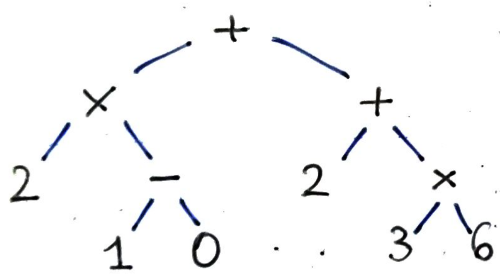
1. La racine
2. Le fils gauche (réc.)
3. le fils droit (réc.)

ex



donne $+ - \times\ 2\ 1\ 4 \div 6\ 2$

ex



donne $+ \times 2 - 1\ 0 + 2 \times 3\ 6$

remq On pourrait aussi définir les parcours en profondeur

- suffixe. fg, fd, racine
- infixe: fg, racine, fd