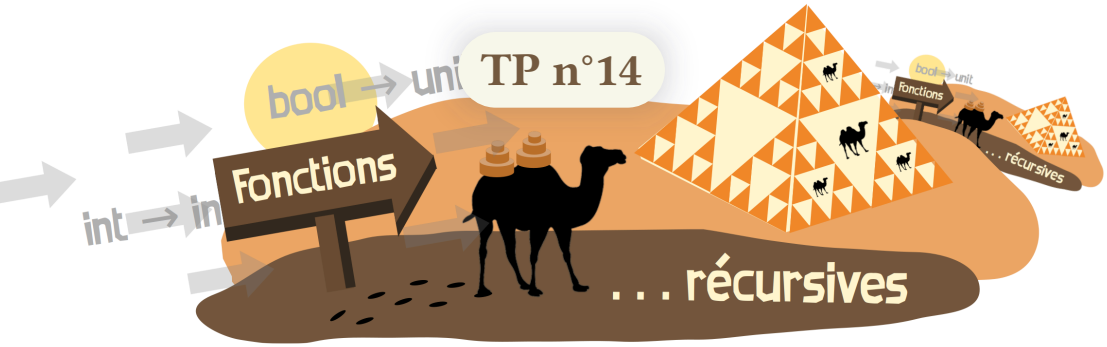


Définition de la récursivité

Si vous avez bien compris ce qu'est la récursivité, vous pouvez passer à la suite du TP. Sinon, commencez par lire la définition de la récursivité écrite au tout début du TP.

I. Types produit

Un premier type non simple : le n-uplet (ou tuple)

En OCaml, on peut regrouper des valeurs (qui peuvent être de types différents) dans des **n-uplets**. Pour cela, il suffit de séparer ces valeurs par des virgules et les entourer de parenthèses. Ex : `let t = (true, 108, 41.)`

Pour pouvoir récupérer les valeurs du n-uplet, on utilise un **let déstructurant**, qui permet de récupérer chacune des valeurs. Ex : `let a, b, c = t in ...` Ici, on a récupéré ces trois valeurs dans `a`, `b` et `c`, on peut les utiliser plus tard. Si certaines valeurs ne nous intéressent pas, on peut utiliser un underscore _ au lieu d'une variable.

1 Vérifier rapidement dans l'interpréteur comment se note le type d'un tel n-uplet. Par exemple un triplet de booléens, puis un couple formé de deux valeurs de types différents. Pourquoi est-ce appelé un **type produit** ?

2 Dans un fichier **tp14.ml**, implémenter deux fonctions nommées `fst` et `snd` qui renvoient respectivement le premier élément d'un couple et le second élément d'un couple. Quelle est la signature obtenue ? Ces fonctions existent déjà en OCaml : on pourra librement les utiliser par la suite.

3 Implémenter une fonction `swap` qui à un couple `(a, b)` quelconque associe le couple `(b, a)`. Quelle est sa signature ?

4 Implémenter une fonction qui, à un entier positif a et un entier strictement positif b, associe le couple formé du quotient et du reste de la division euclidienne de a par b.

5 Implémenter une fonction qui à un triplet d'entiers correspondant aux coefficients a, b et c d'un polynôme du second degré (avec a non nul) indique le nombre de racines de ce polynôme. Tester sur un exemple et l'afficher. Quelle est la signature de cette fonction ?

II. Application partielle et curryfication

6 Dans un interpréteur OCaml, implémenter une fonction de signature `mult : int -> int -> int` qui à deux entiers associe leur multiplication. Tester sur un exemple et l'afficher.

7 De même, implémenter une fonction de signature `maxi : int -> int -> int` qui à deux entiers associe le plus grand des deux. Tester sur un exemple et l'afficher.

8 Toujours dans l'interpréteur, écrire `let a = mult 3;;` puis appuyer sur **Entrée**. La fonction `mult` a besoin de 2 entiers pour fournir un résultat entier, mais que se passe-t-il ici ? Quel est le type obtenu ? En déduire à quoi celà correspond concrètement.

Application partielle

En OCaml, on peut appliquer une fonction à une partie de ses arguments seulement (ceux les plus à gauche). On obtient alors une nouvelle fonction, dans laquelle ces premiers arguments sont fixés, et qui attend les arguments restants : c'est l'**application partielle**.

9 Implémenter une fonction `fois_108` qui à un entier associe cet entier multiplié par 108. On utilisera une application partielle.

10 Implémenter une fonction `valeur_positive` qui renvoie la valeur positive d'un entier (ex: 6 pour 6 et 0 pour −7). Le faire en une seule ligne avec une application partielle d'une fonction déjà existante.

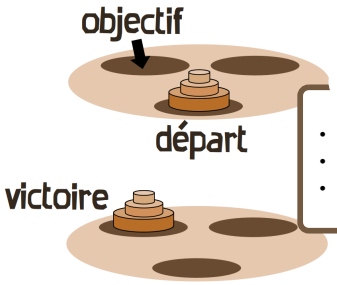
Fonction curryfiée

Une fonction est dite **curryfiée** si elle est définie comme une succession de fonctions à un seul argument. Les fonctions curryfiées peuvent profiter de l'application partielle pour obtenir rapidement de nouvelles fonctions : on cherchera donc le plus possible à implémenter des fonctions curryfiées.

11 Les trois arguments de `nb_racines` sont enfermés dans un n-uplet, ce qui empêche de réaliser une application partielle. Proposer une nouvelle fonction curryfiée de `nb_racines`. Quelle est sa nouvelle signature ?

12 Implémenter une fonction `curryfie` qui prend en argument une fonction non curryfiée dont l'unique argument est un triplet, et renvoie une fonction curryfiée équivalente. Appliquer cette fonction à votre ancienne version de `nb_racines`. Vérifier que le résultat est identique.

III. Tours de Hanoï



Règles du jeu

- (1) On ne peut déplacer qu'un jeton en haut d'une pile, un seul à la fois;
- (2) Il est interdit de placer un jeton sur un autre jeton plus petit que lui;
- (3) Le but du jeu est de déplacer tous les jetons sur la pile **Objectif**.

13 Résoudre à la main le jeu des tours de Hanoï fourni. Combien de coups sont requis au minimum ?

14 Expliquer comment résoudre ce jeu récursivement à l'aide de schémas (au crayon de papier).

15 Suivre la démonstration au tableau pour implémenter deux fonctions OCaml mutuellement récursives `bouge_a_gauche` et `bouge_a_droite` qui prennent un entier n associé au nombre n de disques à déplacer (dans la pile à gauche et dans la pile à droite respectivement) et affichent une séquence de coups permettant de les déplacer.

16 Peut-on résoudre le jeu à l'aide d'une seule fonction ?

IV. Fonctions récursives en OCaml

Fonction récursive en OCaml

On retiendra qu'en OCaml il est possible de définir une fonction qui s'appelle elle-même : on nomme cela une **fonction récursive**. Il faut alors ajouter le mot-clef `rec` lors de la liaison globale pour indiquer que la définition qui suit est récursive et fait référence à l'identifiant qui est en train d'être lié.

Pour qu'une fonction récursive termine, on fera attention à la présence d'un cas de base qui devra être atteint.

17 Dans votre fichier **tp14.ml**, implémenter une fonction récursive sans cas de base, qui affiche un message puis se rappelle elle-même. Vérifier son comportement en l'appelant.

18 Implémenter une fonction récursive permettant de calculer deux à la puissance n avec $n \geq 0$. Quelle est sa signature ?

19 Implémenter une fonction récursive permettant de calculer a à la puissance b, pour a et b deux entiers positifs. Quelle est sa signature ?

20 Implémenter une fonction récursive permettant de calculer la factorielle d'un entier positif. Quelle est sa signature ?

21 Implémenter une fonction récursive permettant de calculer la somme des nombres de 1 à n (sans utiliser la formule) pour $n \geq 0$. Quelle est sa signature ?

22 Implémenter une fonction récursive permettant de calculer le n-ième nombre de Fibonacci, pour $n \geq 0$. Cette suite est définie par $F_0 = 0$, $F_1 = 1$ et $\forall n \geq 2 : F_n = F_{n-1} + F_{n-2}$. Quelle est sa signature ?

23 Implémenter une fonction récursive permettant de calculer le nombre de diviseurs positifs d'un entier positif donné. Quelle est sa signature ?

24 Implémenter une fonction récursive permettant de tester si un entier positif est premier ou non. Quelle est sa signature ?

25 Implémenter une fonction récursive qui prend une fonction `f : int -> int` et un entier n, et renvoie la fonction qui applique n fois successivement f à son argument. On note $f^{(n)}$ cette fonction iterée. Tester