

LIFPF – Programmation fonctionnelle

TD1 – introduction au λ -calcul

Licence informatique UCBL – Printemps 2025

Exercice 1 : Typage et évaluation en λ -calcul

1. Pour chacune des λ -expressions suivantes, donner son type en supposant que le seul type primitif est **number** puis l'évaluer *en explicitant chaque β -réduction*. Si l'expression n'est pas typable, expliquer pourquoi. On supposera que l'addition notée $+$ a pour type **number** \rightarrow **number** \rightarrow **number** et se réduit par calcul arithmétique usuel, de même que pour la multiplication notée $*$. Par exemple, $(\lambda x. x + 3) 2$ a pour type **number** et peut se réduire en $(\lambda x. x + 3) 2 \xrightarrow{\beta} 2 + 3 \rightsquigarrow 5$
 1. $((\lambda x. \lambda y. (x + y) \ 3) \ 4)$
 2. $((\lambda x. \lambda y. (x \ y) \ 3) \ 4)$
 3. $\lambda x. (x \ x)$
 4. $((\lambda x. \lambda y. (y \ (x + 2)) \ 5) \ \lambda z. (z * 3))$
 5. $(\lambda x. (\lambda y. (x + y) \ x) \ 5)$
 6. $(\lambda x. ((x \ \lambda y. (y + 2)) + (x \ \lambda z. (z * 3)))) \ (\lambda u. \lambda w. (w \ u) \ 5))$
 7. $(\lambda x. \lambda y. (x \ y) \ \lambda z. z)$
 8. $(\lambda x. (x + 3) \ \lambda y. y)$
 9. $\lambda x. \lambda y. x$
 10. $(\lambda x. \lambda y. x \ 3)$
 11. $((\lambda x. \lambda y. (y \ (3 + x)) \ 4) \ \lambda z. (z * 2))$
 12. $(\lambda x. (\lambda y. (y * x) \ x) \ 5)$
2. Donner deux séquences de réductions *différentes* de l'expression $(\lambda x. (x + 5) \ (\lambda z. z \ 2))$

Exercice 2 : Pour aller plus loin : codage des booléens dans le λ -calcul

Dans cet exercice on va s'intéresser au codage des booléens en λ -calcul pur appelé *booléens de Church*. Dans ce codage, les valeurs de vérités sont *des fonctions* et les fonctions booléennes *des fonctions d'ordre supérieur*. On définit les termes suivants qui représentent « vrai », « faux » et la fonction « et » :

- **T** = $\lambda x. \lambda y. x$
- **F** = $\lambda x. \lambda y. y$
- **AND** = $\lambda x. \lambda y. ((x \ y) \ \mathbf{F})$.

1. Vérifier les réductions suivantes

$$\mathbf{AND}(\mathbf{T})(\mathbf{T}) \rightsquigarrow \mathbf{T}$$

$$\mathbf{AND}(\mathbf{T})(\mathbf{F}) \rightsquigarrow \mathbf{F}$$

$$\mathbf{AND}(\mathbf{F})(\mathbf{T}) \rightsquigarrow \mathbf{F}$$

$$\mathbf{AND}(\mathbf{F})(\mathbf{F}) \rightsquigarrow \mathbf{F}$$

2. Quelle est la fonction booléenne associée à **XXX** $= \lambda x. \lambda y. ((x\ y)\ x)$?
3. Qu'est ce qui se passe quand on évalue $((\mathbf{AND}\ x)\ y)$ où x ou y n'est *pas* la représentation d'un booléen ?
4. Donner une représentation de la fonction booléenne « non ».
5. Donner une représentation de la fonction booléenne « ou ».
6. Montrer que le terme **IF** $= \lambda x. x$ permet de représenter l'instruction conditionnelle **IF** *cond* **THEN** *M* **ELSE** *N*.

Corrections

Solution de l'exercice 1

1. 1. Typage :

$$\underbrace{\underbrace{\underbrace{\underbrace{\lambda x. \lambda y. (x + y)}_{\text{number} \rightarrow \text{number}}}_{\text{number} \rightarrow \text{number} \rightarrow \text{number}}}_{\text{number} \rightarrow \text{number}}}_{\text{number}} \quad \underbrace{3}_{\text{number}} \quad \underbrace{4}_{\text{number}})$$

avec $x : \text{number}$ et $y : \text{number}$.

Évaluation :

$$((\lambda x. \lambda y. (x + y) \ 3) \ 4) \xrightarrow{x} (\lambda y. (3 + y) \ 4) \xrightarrow{y} (3 + 4) \rightsquigarrow 7$$

2. Typage : l'expression $((\lambda x. \lambda y. (x + y) \ 3) \ 4)$ n'est pas typable car la variable x doit à la fois être un entier (car on passe 3 en paramètre à $\lambda x. \lambda y. (x + y)$) et une fonction car x est appliquée à y . Par ailleurs, si on essaie d'effectuer les β -reductions, on obtient $(3 \ 4)$ qui n'est pas une valeur : seuls les entiers et les fonctions (*i.e.* les expressions commençant par λ) sont des valeurs.

3. Typage : l'expression $\lambda x. (x \ x)$ n'est pas typable. Si on dit que le type de x est un certain α , alors x doit prendre quelque chose de type α en argument (vu qu'on applique x à lui-même). Le type de x doit ainsi être solution de l'équation $\alpha \rightarrow \alpha' = \alpha$, *i.e.* en remplaçant α à gauche : $(\alpha \rightarrow \alpha') \rightarrow \alpha' = \alpha$, dans laquelle on peut encore remplacer α à gauche autant de fois qu'on veut.

4. Typage :

$$\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{\lambda x. \lambda y. (\underbrace{y}_{\text{number} \rightarrow \text{number}} \quad \underbrace{(x + 2)}_{\text{number}})}_{\text{number} \rightarrow \text{number} \rightarrow \text{number}}}_{\text{number} \rightarrow \text{number}}}_{\text{number} \rightarrow (\text{number} \rightarrow \text{number}) \rightarrow \text{number}}}_{(\text{number} \rightarrow \text{number}) \rightarrow \text{number}}}_{\text{number}} \quad \underbrace{5}_{\text{number}} \quad \underbrace{\lambda z. (z * 3)}_{\text{number} \rightarrow \text{number}})$$

avec $x : \text{number}$, $y : \text{number} \rightarrow \text{number}$ et $z : \text{number}$.

Évaluation :

$$((\lambda x. \lambda y. (y \ (x + 2)) \ 5) \ \lambda z. (z * 3)) \xrightarrow{x} (\lambda y. (y \ (5 + 2)) \ \lambda z. (z * 3)) \xrightarrow{y} (\lambda z. (z * 3) \ (5 + 2)) \xrightarrow{z} ((5 + 2) * 3) \rightsquigarrow 21$$

5. Typage :

$$\underbrace{\underbrace{\underbrace{\underbrace{\lambda x. (\underbrace{\lambda y. (x + y)}_{\text{number} \rightarrow \text{number}} \ x)}_{\text{number} \rightarrow \text{number}}}_{\text{number}}}_{\text{number} \rightarrow \text{number}}}_{\text{number}} \quad \underbrace{5}_{\text{number}})$$

Évaluation, 2 stratégies possibles :

$$(\lambda x.(\lambda y.(x + y) \ x) \ 5) \xrightarrow{x} (\lambda y.(5 + y) \ 5) \xrightarrow{y} (5 + 5) \rightsquigarrow 10$$

et

$$(\lambda x.(\lambda y.(x + y) \ x) \ 5) \xrightarrow{y} (\lambda x.(x + x) \ 5) \xrightarrow{x} (5 + 5) \rightsquigarrow 10$$

C'est la première qui correspond le plus à OCaml où on évalue d'abord les arguments.

6. Typage :

$$\begin{array}{c} (\lambda x.((x \ \underbrace{\lambda y.(y+2)}_{\text{number}}) + (x \ \underbrace{\lambda z.(z*3)}_{\text{number}})) \ (\lambda u. \ \underbrace{\lambda w.(\underbrace{w \ u}_{\text{number}})}_{\text{number}} \ \underbrace{5}_{\text{number}})) \\ \underbrace{\underbrace{\underbrace{\text{number} \rightarrow \text{number}}_{\text{number}} \rightarrow \text{number}}_{\text{number}} \rightarrow \text{number}}_{\text{number}} \rightarrow \underbrace{\underbrace{\underbrace{\text{number} \rightarrow (\text{number} \rightarrow \text{number}) \rightarrow \text{number}}_{\text{number}}}_{\text{number}} \rightarrow \text{number}}_{\text{number}} \\ \underbrace{\underbrace{\underbrace{\text{number} \rightarrow (\text{number} \rightarrow \text{number}) \rightarrow \text{number}}_{\text{number}}}_{\text{number}} \rightarrow \text{number}}_{\text{number}} \rightarrow \text{number} \end{array}$$

avec $x : (\text{number} \rightarrow \text{number}) \rightarrow \text{number}$, $y : \text{number}$, $z : \text{number}$, $u : \text{number}$, $w : \text{number} \rightarrow \text{number}$

Évaluation :

$$\begin{array}{l} (\lambda x.((x \ \lambda y.(y+2)) + (x \ \lambda z.(z*3))) \ (\lambda u.\lambda w.(w \ u) \ 5)) \\ \xrightarrow{u} (\lambda x.((x \ \lambda y.(y+2)) + (x \ \lambda z.(z*3))) \ \lambda w.(w \ 5)) \\ \xrightarrow{x} ((\lambda w_1.(w_1 \ 5) \ \lambda y.(y+2)) + (\lambda w_2.(w_2 \ 5) \ \lambda z.(z*3))) \\ \xrightarrow{w_1} ((\lambda y.(y+2) \ 5) + ((\lambda w_2.(w_2 \ 5) \ \lambda z.(z*3)))) \\ \xrightarrow{y} ((5+2) + (\lambda w_2.(w_2 \ 5) \ \lambda z.(z*3))) \\ \xrightarrow{w_2} ((5+2) + (\lambda z.(z*3) \ 5)) \\ \xrightarrow{z} ((5+2) + (5*3)) \rightsquigarrow 22 \end{array}$$

Remarque : ici, on a distingué les 2 occurrences de w en renommant w en w_1 et w_2 pour montrer leur indépendance. Si on ne fait que des β -reduction, il ne faudrait pas mettre d'indice, mais on perd en clarté.

7. $(\lambda x.\lambda y.(x \ y) \ \lambda z.z) :: \text{number} \rightarrow \text{number}$
 $(\lambda x.\lambda y.(x \ y) \ \lambda z.z) \xrightarrow{x} \lambda y.(\lambda z.z \ y) \xrightarrow{y} \lambda y.y$
8. $(\lambda x.(x + 3) \ \lambda y.y)$ n'est pas typable
 $(\lambda x.(x + 3) \ \lambda y.y) \xrightarrow{x} (\lambda y.y + 3)$
9. $\lambda x.\lambda y.x :: \text{number} \rightarrow \text{number} \rightarrow \text{number}$
l'expression est déjà β -réduite
10. $(\lambda x.\lambda y.x \ 3) :: \text{number} \rightarrow \text{number}$
 $(\lambda x.\lambda y.x \ 3) \xrightarrow{x} \lambda y.3$
11. $((\lambda x.\lambda y.(y \ (3 + x)) \ 4) \ \lambda z.(z * 2)) :: \text{number}$
 $(\lambda x.\lambda y.(y \ (3 + x))) \ 4 \ (\lambda z.(z * 2)) \xrightarrow{x} (\lambda y.(y \ (3 + 4)) \ \lambda z.(z * 2))$
 $\xrightarrow{y} (\lambda z.(z * 2) \ (3 + 4)) \rightsquigarrow (\lambda z.(z * 2) \ 7) \xrightarrow{z} (7 * 2) \rightsquigarrow 14$
12. $(\lambda x.(\lambda y.(y * x) \ x) \ 5) :: \text{number}$
 $(\lambda x.(\lambda y.(y * x) \ x) \ 5) \xrightarrow{x} (\lambda y.(y * 5) \ 5) \xrightarrow{y} (5 * 5) \rightsquigarrow 25$
2. — $(\lambda x.(x + 5) \ (\lambda z.z \ 2)) \xrightarrow{z} (\lambda x.(x + 5) \ 2) \xrightarrow{x} (2 + 5) \rightsquigarrow 7$

$$— (\lambda x.(x + 5) \ (\lambda z.z \ 2)) \xrightarrow{x} ((\lambda z.z \ 2) + 5) \xrightarrow{z} 2 + 5 \rightsquigarrow 7$$

Solution de l'exercice 2

On peut expliquer, en guise d'ouverture, qu'on peut faire un codage similaire avec les entiers.

1. En remarquant que $\mathbf{T}(Z) \rightsquigarrow \lambda y.Z$ et que $\mathbf{F}(Z) \rightsquigarrow \lambda y.y$ on peut obtenir les dérivations suivantes assez rapidement.

$$\begin{aligned} ((\mathbf{AND} \ \mathbf{T}) \ \mathbf{T}) &\rightsquigarrow ((\lambda x.\lambda y.((x \ y) \ \mathbf{F}) \ \mathbf{T}) \ \mathbf{T}) \\ &\xrightarrow{x} (\lambda y.((\mathbf{T} \ y) \ \mathbf{F}) \ \mathbf{T}) \\ &\xrightarrow{y} ((\mathbf{T} \ \mathbf{T}) \ \mathbf{F}) \\ &= ((\lambda x.\lambda y.x \ \lambda x.\lambda y.x) \ \mathbf{F}) \\ &\xrightarrow{x} (\lambda y.\lambda x.\lambda y.x \ \mathbf{F}) \\ &\xrightarrow{y} (\lambda x.\lambda y.x) \\ &= \mathbf{T} \\ ((\mathbf{AND} \ \mathbf{T}) \ \mathbf{F}) &= ((\lambda x.\lambda y.x \ \lambda x.\lambda y.y) \ \mathbf{F}) \\ &\xrightarrow{x} (\lambda y.\lambda x.\lambda y.y \ \mathbf{F}) \\ &\xrightarrow{y} (\lambda x.\lambda y.y) \\ &= \mathbf{F} \\ ((\mathbf{AND} \ \mathbf{F}) \ \mathbf{T}) &= ((\lambda x.\lambda y.y \ \lambda x.\lambda y.x) \ \mathbf{F}) \\ &\xrightarrow{x} (\lambda y.y \ \mathbf{F}) \\ &\xrightarrow{y} \mathbf{F} \\ ((\mathbf{AND} \ \mathbf{F}) \ \mathbf{F}) &= ((\lambda x.\lambda y.y \ \lambda x.\lambda y.y) \ \mathbf{F}) \\ &\xrightarrow{x} (\lambda y.y) \ \mathbf{F} \\ &\xrightarrow{y} \mathbf{F} \end{aligned}$$

2. On va évaluer les quatre cas $((\mathbf{XXX} \ \mathbf{F}) \ \mathbf{F})$, $((\mathbf{XXX} \ \mathbf{T}) \ \mathbf{F})$, $((\mathbf{XXX} \ \mathbf{F}) \ \mathbf{T})$ et $((\mathbf{XXX} \ \mathbf{T}) \ \mathbf{T})$ pour remarquer qu'il s'agit d'une autre expression du « et ».
3. Il y aura bien réduction mais le résultat ne sera pas nécessairement la représentation d'un booléen, par exemple $((\mathbf{AND} \ \lambda x.x) \ \lambda x.(x \ x))$ se réduit en $(\mathbf{F} \ \mathbf{F})$ puis en $\lambda y.y$ qui ne représente pas un booléen : *garbage in, garbage out* comme on dit.
4. On peut proposer $\mathbf{NOT} = \lambda x.((x \ \mathbf{F}) \ \mathbf{T})$, l'intuition étant que pour inverser les valeurs booléennes, on échange les deux derniers arguments passés à \mathbf{AND} .
5. On peut proposer, parmi plusieurs solutions possibles, $\mathbf{OR} = \lambda x.\lambda y.((y \ y) \ x)$ ou $\mathbf{OR} = \lambda x.\lambda y.((x \ \mathbf{T}) \ y)$. L'expression $\lambda x.\lambda y.((x \ y) \ \mathbf{T})$ représente quant à elle l'implication.
6. On va montrer que $((\mathbf{IF} \ \mathbf{T}) \ M) \ N$ se réduit en M et que $((\mathbf{IF} \ \mathbf{F}) \ M) \ N$ se réduit en N ce qui capture l'idée de l'instruction conditionnelle. On donne le premier cas, le second étant similaire.

$$\begin{aligned} (((\mathbf{IF} \ \mathbf{T}) \ M) \ N) &= (((\lambda x.x \ \mathbf{T}) \ M) \ N) \\ &\xrightarrow{x} ((\mathbf{T} \ M) \ N) \\ &\rightsquigarrow M \end{aligned}$$