LIFPF – Programmation fonctionnelle
TD1 – introduction au λ-calcul

Licence informatique UCBL – Printemps 2023–2024

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` $\to$ `number` $\to$ `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 $(\lambda x.x + 3)\ 2 \overset{x}{\rightsquigarrow} 2 + 3 \rightsquigarrow 5$
   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. $(\lambda x.((x\ (\lambda y.(y + 2))) + (x\ (\lambda z.(z * 3)))))\quad ((\lambda u.\lambda w.(w\ u))\ 5)$
   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 » :

— $\mathbf{T} = \lambda x.\lambda y.x$
— $\mathbf{F} = \lambda x.\lambda y.y$
— $\mathbf{AND} = \lambda x.\lambda y.((x\ y)\ \mathbf{F})$, soit $\lambda x.\lambda y.x\ y\ \mathbf{F}$ avec les parenthèses implicites

1. Vérifier les réductions suivantes

$$\begin{aligned} \mathbf{AND}(\mathbf{T})(\mathbf{T}) &\rightsquigarrow \mathbf{T} \\ \mathbf{AND}(\mathbf{T})(\mathbf{F}) &\rightsquigarrow \mathbf{F} \\ \mathbf{AND}(\mathbf{F})(\mathbf{T}) &\rightsquigarrow \mathbf{F} \\ \mathbf{AND}(\mathbf{F})(\mathbf{F}) &\rightsquigarrow \mathbf{F} \end{aligned}$$

2. Quelle est la fonction booléenne associée à $\textbf{XXX} = \lambda x.\lambda y.x\ y\ x$?
3. Qu'est ce qui se passe quand on évalue $\textbf{AND}\ x\ y$ 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 $\textbf{IF} = \lambda x.x$ permet de représenter l'instruction conditionnelle `IF cond THEN M ELSE N`.

Corrections

Solution de l'exercice 1

1. 1. Typage :

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

avec x : `number` et y : `number`.
Évaluation :

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

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 appliquées 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 $\alpha \to \alpha' = \alpha$, *i.e.* en replaçant α à gauche : $(\alpha \to \alpha') \to \alpha' = \alpha$, dans laquelle on peut encore remplacer α à gauche autant de fois qu'on veut.

4. Typage :

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

avec x : `number`, y : `number` $\to$ `number` et z : `number`.
Évaluation :

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

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

5. Typage :

$$\underbrace{\underbrace{(\lambda x.\underbrace{((\ \underbrace{\lambda y.(\underbrace{x+y}_{\texttt{number}})}_{\texttt{number}\to\texttt{number}}\ )\ x)}_{\texttt{number}})}_{\texttt{number}\to\texttt{number}}\ \underbrace{5}_{\texttt{number}}}_{\texttt{number}}$$

Évaluation, 2 stratégies possibles :

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

et

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

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

6. Typage :

$$\underbrace{\underbrace{(\lambda x.(\underbrace{\underbrace{(x\ (\underbrace{\lambda y.(\underbrace{y+2}_{\texttt{number}})}_{\texttt{number}\to\texttt{number}}))}_{\texttt{number}} + \underbrace{(x\ (\underbrace{\lambda z.(\underbrace{z*3}_{\texttt{number}})}_{\texttt{number}\to\texttt{number}}))}_{\texttt{number}}}_{\texttt{number}}))}_{((\texttt{number}\to\texttt{number})\to\texttt{number})\to\texttt{number}}\ \underbrace{((\underbrace{\lambda u.\underbrace{\lambda w.(\underbrace{w\ u}_{\texttt{number}})}_{(\texttt{number}\to\texttt{number})\to\texttt{number}}}_{\texttt{number}\to(\texttt{number}\to\texttt{number})\to\texttt{number}})\ \underbrace{5}_{\texttt{number}})}_{(\texttt{number}\to\texttt{number})\to\texttt{number}}}_{\texttt{number}}$$

avec x : $(\texttt{number} \to \texttt{number}) \to \texttt{number}$, y :, z : $\texttt{number}$, u : $\texttt{number}$, w : $\texttt{number} \to \texttt{number}$
Évaluation :

$$(\lambda x.((x\ (\lambda y.(y+2))) + (x\ (\lambda z.(z*3)))))\quad ((\lambda u.\lambda w.(w\ u))\ 5)$$
$$\overset{u}{\rightsquigarrow} (\lambda x.((x\ (\lambda y.(y+2))) + (x\ (\lambda z.(z*3)))))\quad (\lambda w.(w\ 5))$$
$$\overset{x}{\rightsquigarrow} (((\lambda w_1.(w_1\ 5))\ (\lambda y.(y+2))) + ((\lambda w_2.(w_2\ 5))\ (\lambda z.(z*3))))$$
$$\overset{w_1}{\rightsquigarrow} (((\lambda y.(y+2))\ 5) + ((\lambda w_2.(w_2\ 5))\ (\lambda z.(z*3))))$$
$$\overset{y}{\rightsquigarrow} ((5+2) + ((\lambda w_2.(w_2\ 5))\ (\lambda z.(z*3))))$$
$$\overset{w_2}{\rightsquigarrow} ((5+2) + ((\lambda z.(z*3))\ 5))$$
$$\overset{z}{\rightsquigarrow} ((5+2) + (5*3)) \rightsquigarrow 22$$

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. $(\lambda x.\lambda y.(x\ y))(\lambda z.z) :: \texttt{number} \to \texttt{number}$
   $(\lambda x.\lambda y.(x\ y))(\lambda z.z) \overset{x}{\rightsquigarrow} \lambda y.((\lambda z.z)\ y) \overset{y}{\rightsquigarrow} \lambda y.y$

8. $\lambda x.(x+3)(\lambda y.y)$ n'est pas typable
   $\lambda x.(x+3)(\lambda y.y) \overset{x}{\rightsquigarrow} (\lambda y.y)+3$

9. $(\lambda x.\lambda y.x) :: \texttt{number} \to \texttt{number} \to \texttt{number}$
   l'expression est déjà β-réduite

10. $(\lambda x.\lambda y.x)\ 3 :: \texttt{number} \to \texttt{number}$
    $(\lambda x.\lambda y.x)\ 3 \overset{x}{\rightsquigarrow} \lambda y.3$

11. $(\lambda x.\lambda y.(y\ (3+x)))\ 4\ (\lambda z.(z*2)) :: \texttt{number}$
    $(\lambda x.\lambda y.(y\ (3+x)))\ 4\ (\lambda z.(z*2)) \overset{x}{\rightsquigarrow} \lambda y.(y\ (3+4))\ (\lambda z.(z*2)) \overset{y}{\rightsquigarrow} (\lambda z.(z*2))\ (3+4) \rightsquigarrow (\lambda z.(z*2))\ (7) \overset{z}{\rightsquigarrow} (7*2) \rightsquigarrow 14$

12. $(\lambda x.((\lambda y.(y*x))\ x))\ 5 :: \texttt{number}$
    $(\lambda x.((\lambda y.(y*x))\ x))\ 5 \overset{x}{\rightsquigarrow} (\lambda y.(y*5))\ 5 \overset{y}{\rightsquigarrow} 5*5 \rightsquigarrow 25$

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

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.

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

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 $\mathbf{FF}$ 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 $\mathbf{NOT}$.
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.

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