Programmation fonctionnelle – TD1

S5 – 2023/2024

1 Encodage des pairs [★]

Étant donné les abbréviations suivantes pour des lambda-termes :

$$
\begin{array}{lcl}
\mathit{true} & \equiv & \lambda x.\lambda y.\, x \\
\mathit{false} & \equiv & \lambda x.\lambda y.\, y \\
\mathit{pair} & \equiv & \lambda f.\lambda s.\lambda b.\, b\, f\, s \\
\mathit{first} & \equiv & \lambda p.\, p\ \mathit{true} \\
\mathit{second} & \equiv & \lambda p.\, p\ \mathit{false}
\end{array}
$$

vérifiez que :

1. $\mathit{first}\,(\mathit{pair}\, v\, w)$ se réduit à v, et
2. $\mathit{second}\,(\mathit{pair}\, v\, w)$ se réduit à w.

2 Encodage des opérateurs booléens [★★]

Définissez les lambda-termes *and*, *or* et *xor* qui encodent les opérateurs booléens habituels. Par exemple *and true false* devra se réduire à *false*, *and true true* devra se réduire à *true*, etc. Vous pouvez utiliser les lambda-termes *ite* et *not* définis en cours.

3 Auto-application [★]

Soit le terme

$$\mathrm{L} \equiv (\lambda x.\, x\, x)\,(\lambda x.\, x\, x)$$

1. Quelle est la forme normale de L ?
2. Concernant l'évaluation du terme

   $$(\lambda x.\lambda y.\, y)\ \mathrm{L}\ v$$

   quelles observations peut-on faire ?

4 Combinateur de point fixe [★★]

Soit le terme

$$\mathrm{Y} \equiv \lambda f.\, (\lambda x.\, f\,(x\,x))\,(\lambda x.\, f(x\,x))$$

Vérifiez que $\mathrm{Y}\,g$ et $g\,(\mathrm{Y}\,g)$ peuvent se réduire à un même terme (on dit qu'ils sont β-équivalents).

5 Une fonction récursive [★★★]

De manière générale $\mathrm{Y}\,g$ est donc β-équivalent à $g\,(g\,(g\,\ldots(\mathrm{Y}\,g)))$, ce qui ne signifie pas que le premier terme se réduise forcément au second, mais seulement que les deux termes se réduisent à un même terme. Le combinateur Y permet ainsi de coder des fonctions récursives.

1. Nous allons utiliser le combinateur Y pour coder une fonction *even* qui prend un entier naturel et retourne *true* si et seulement si cet entier est pair. On suppose que chaque entier naturel n est représenté par un lambda-terme noté $\lfloor n \rfloor$ (pour cette question, la nature exacte de ce codage n'est pas importante). On suppose aussi l'existence de deux lambda-termes *iszero* et *pred* tels que
   — $\mathit{iszero}\,\lfloor 0 \rfloor$ se réduit à *true*
   — $\mathit{iszero}\,\lfloor n \rfloor$ se réduit à *false* pour $n \neq 0$
   — $\mathit{pred}\,\lfloor 0 \rfloor$ se réduit à $\lfloor 0 \rfloor$
   — $\mathit{pred}\,\lfloor n \rfloor$ se réduit à $\lfloor n-1 \rfloor$ pour $n \neq 0$
   À l'aide des lambda-termes ci-dessus, ainsi que des lambda-termes *ite* et *not* définis en cours, définissez une fonction g telle que $\mathrm{Y}\,g$ correspond à *even*, c.-à-d. que $\mathrm{Y}\,g\,\lfloor n \rfloor$ se réduit à *true* si n est pair et à *false* sinon.
2. On considère à présent la nature de l'encodage des entiers : $\lfloor n \rfloor$ est le lambda-terme $\lambda s.\lambda z.\, s^n\, z$. Par exemple l'entier 0 est codé par $\lambda s.\lambda z.\, z$, et l'entier 3 est codé par $\lambda s.\lambda z.\, s\,(s\,(s\,z))$. Utilisez ce codage pour définir *even* sans passer par le combinateur Y.