Fondements de la programmation

Exercices 7 lambda-calcul

Un lambda-terme t est une expression syntaxique qui est soit simplement une variable, soit l'abstraction d'un autre lambda-terme selon une variable, soit l'application d'un terme à un autre :

$$t := x \mid \lambda x.t \mid (t\ t)$$

où x est n'importe quel élément d'un ensemble de symboles $\mathcal{V}$ choisi une fois pour toute comme ensemble de variables. On utilise habituellement les lettres de l'alphabet a, b, ..., x, y, z éventuellement indexés par des entiers, x_{42}, comme ensemble de variables. Il est habituel de raccourcir les notations des lambdas en notant $\lambda x_1 \dots x_n.t$ au lieu de $\lambda x_1.\lambda x_2.\dots.\lambda x_n.t$. Par exemple on écrira $\lambda xy.x$ au lieu de $\lambda x.\lambda y.x$. On doit parfois ajouter des parenthèses autour des sous-expressions $\lambda x.t$ pour éviter une ambiguité. Par exemple, on écrit $((\lambda x.x)\ y)$. Par contre on peut omettre nombre de parenthèses d'application en prenant comme convention que les expressions comme $x\ y\ z$ sont parenthésées à gauche : $((x\ y)\ z)$.

L'ensemble des variables libres d'un lambda-terme est défini par induction sur le terme :

$$\begin{aligned} \mathrm{FV}(x) &= \{x\} \\ \mathrm{FV}(\lambda x.t) &= \mathrm{FV}(t) - \{x\} \\ \mathrm{FV}((t\ u)) &= \mathrm{FV}(t) \cup \mathrm{FV}(u) \end{aligned}$$

De même, l'ensemble des variables liées d'un lambda-terme est défini par induction sur le terme :

$$\begin{aligned} \mathrm{BV}(x) &= \emptyset \\ \mathrm{BV}(\lambda x.t) &= \mathrm{BV}(t) \cup \{x\} \\ \mathrm{BV}((t\ u)) &= \mathrm{BV}(t) \cup \mathrm{BV}(u) \end{aligned}$$

Remarques : ce sont les lambdas qui lient les variables ; une variable libre est simplement une variable qui n'est pas liée.

Lorsqu'on écrit un lambda-terme sous forme d'arbre syntaxique, les $\lambda x.\dots$ sont des nœuds unaires, les applications sont des nœuds binaires, habituellement dénotés par le symbole @, et les variables correspondent aux feuilles. On peut alors ajouter à cet arbre syntaxique des arcs orientés. Pour chaque variable liée on dessine un arc allant au lambda qui la lie (le premier lambda en remontant vers la racine).

Avec les arcs lieurs on obtient une arbre appelé arbre du lambda-terme. Un exemple est donné figure 1.

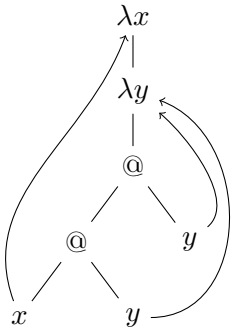

FIGURE 1 – arbre du terme $\lambda xy.((x\ y)\ y)$

Deux termes dont les arbres sont égaux sauf éventuellement pour le nom de leurs variables liées sont dits $\$\alpha\$$-équivalents. On peut toujours renommer n'importe quelle variable liée d'un terme ou d'un sous-terme pour obtenir un terme $\$\alpha\$$-équivalent.

Sur les arbres, la **substitution** d'une variable libre x par un terme u dans un terme t consiste en remplacer dans l'arbre de t chaque feuille x non liée par l'arbre de u. On note $t[x := u]$ le lambda-terme obtenu.

Si on n'y prend pas garde on risque la *capture* de variables : des variables qui étaient libres dans u se retrouvent liées dans $t[x := u]$. On considéra uniquement la substitution sans capture de variables. C'est à dire que l'opération de substitution n'a de sens qu'à $\$\alpha\$$-renommage près. Toute capture d'une variable libre de u par un lieur de t sera évitée par un renommage des variables liées de t en choisissant des noms différents des variables libres de u.

Un terme de la forme $((\lambda x.t)\ u)$ est appelé un **rédex**. Il se β réduit en une étape en le terme $t[x := u]$. Un rédex peut aussi apparaître dans un sous-terme (une sous-expression). L'idée est celle de l'application d'une fonction à un argument : l'argument se substitue à la variable de la fonction.

Un terme sans rédex est dit en forme normale.

Un terme peut avoir plusieurs rédex comme sous-termes. La $\$\beta\$$-réduction peut alors s'appliquer à n'importe lequel des sous-termes. Le choix du rédex à réduire peut éventuellement être déterminé par une stratégie de réduction.

Partant d'un terme avec rédex si on arrive à une forme normale alors elle est unique, quel que soit le chemin de réduction.

1 Exercices

1.1 Variables libres, variables liées, alpha-équivalence

Question A. Variables libres et liées. Donner l'ensemble des variables libres et l'ensemble des variables liées des termes suivants :

1. $(x\ y)$
2. $(x\ (\lambda y.y))$
3. $\lambda xyz.(x\ (y\ z))$
4. $((\lambda xyz.(x\ y))\ z)$
5. $((\lambda x.(x\ y))\ x)$
6. $((\lambda x.x)\ (\lambda x.x))$

Question B. α-équivalence. Pour chacun des deux derniers termes de l'exercice précédent donner un terme qui lui est α-équivalent et utilise un maximum de noms de variables.

1.2 Substitution, réductions

Question C. Substitutions. Effectuer les substitutions suivantes :

1. $x[x := y]$
2. $(x\ y)[x := \lambda z.z]$
3. $(x\ x)[x := \lambda z.z]$
4. $(x\ x)[y := \lambda z.z]$
5. $((\lambda x.(x\ y))\ x)[x := z]$
6. $((\lambda x.(x\ y))\ y)[y := z]$

Question D. Réductions. Réduire et donner la forme normale des termes suivants :

1. $((\lambda x.x)\ ((\lambda x.x)\ (\lambda x.x)))$
2. $((\lambda x.(x\ x))\ (\lambda z.z))$
3. $((\lambda xy.y)\ f\ g)$
4. $((\lambda xy.(x\ (x\ (x\ y))))\ (\lambda x.z))$
5. $((\lambda x.(x\ x))\ (\lambda x.(x\ x)))$
6. $((\lambda f.(\lambda x.f(x\ x))\ (\lambda x.f(x\ x)))\ g)$

1.3 Entiers de Church

On se donne un ensemble de lambda-termes pour représenter les entiers naturels. On représente le zéro par le terme $\lambda fx.x$, le un par le terme $\lambda fx.(f\ x)$, le deux par le terme $\lambda fx.(f\ (f\ x))$ et plus généralement l'entier n est représenté par le terme noté $\overline{n}$:

$$\overline{n} := \lambda fx.(\underbrace{f\ \ldots\ (f}_{n}\ x)).$$

Cette représentation est appelée entiers de Church du nom du mathématicien Alonzo Church, qui inventa le lambda-calcul et montra que toutes les fonctions calculables sur les entiers peuvent être représentées par des lambda-termes en utilisant la β-réduction pour effectuer le calcul. Nous allons essayer de calculer un peu avec ses entiers.

Question E. Successeur. Donner un lambda-terme succ qui représente l'opération successeur sur les entiers de Church, c'est à dire tel que pour n'importe quel entier naturel n, (succ $\overline{n}$) se β-réduit en $\overline{n+1}$. Tester.

Question F. Addition. Donner un lambda-terme add qui représente l'addition de deux entiers. C'est-à-dire tel que pour n'importe quels entiers n et n, (add $\overline{n}\,\overline{m}$) se β-réduit en $\overline{n+m}$.

Question G. Multiplication. Même question pour la multiplication $\overline{mult}$.

Question H. Test à zéro. Donner un lambda-terme ifzero qui, lorsqu'il est appliqué à un entier $\overline{n}$ et deux termes t_1 et t_2, se β-réduit en t_1 si $n = 0$ et en t_2 sinon.