TP λ-calcul

Le but de ce TP est de mettre en oeuvre le lambda-calcul en OCaml.

Écrire un fichier `.ml` et le déposer sur Moodle dans la zone de dépôt prévue à cet effet.

1 Questions préliminaires

1. Ecrire une fonction `union` qui permet de calculer l'union de deux listes (sans créer de doublons).
   Ex : `union [1;2;3] [1;3;4] = [2;1;3;4]`.
2. Ecrire une fonction `inter` qui permet de calculer l'intersection de deux listes.
   Ex : `inter [1;2;3] [1;3;4] = [1;3]`.

2 Lambda-terme

1. Définir un type `terme` permettant de représenter les termes du lambda-calcul.
2. Définir les termes suivants :
   (a) $I \stackrel{\text{def}}{=} \lambda x.x$
   (b) $A \stackrel{\text{def}}{=} (\lambda x.x)\ a$
   (c) $K \stackrel{\text{def}}{=} \lambda x.\lambda y.x$
   (d) $S \stackrel{\text{def}}{=} \lambda x.(\lambda y.(\lambda z.(x\ z)\ (y\ z)))$
   (e) $\Delta \stackrel{\text{def}}{=} \lambda x.x\ x$
   (f) $\Omega \stackrel{\text{def}}{=} \Delta\ \Delta$
3. Définir une fonction `print_terme t` qui permet d'afficher un terme à l'écran.

3 Variables libres et variables liées

1. Définir une fonction `fv t` qui permet de calculer la liste des variables libres dans un terme (sous forme d'une liste de chaînes de caractères).
2. Définir une fonction `bv t` qui permet de calculer la liste des variables liées dans un terme (sous forme d'une liste de chaînes de caractères).
3. Définir une fonction `vars t` qui permet de calculer la liste des variables d'un terme.
4. Définir une fonction `variable_fraiche l` qui prend en argument une liste de chaînes de caractères et qui renvoie une chaîne de caractères qui n'apparaît pas dans la liste. On pourra utiliser une fonction auxiliaire et un compteur.

4 Substitution et renommage

1. Définir une fonction `subst t x t'` qui réalise la substitution d'une variable (Rappel : On ne substitue que les occurrences libres de la variable dans le λ-terme).
2. Définir une fonction `alpha t x y` qui réalise une α-conversion : C'est-à-dire renomme une variable liée dans un terme.

5 Réduction

1. Ecrire une fonction `est_forme_normale t` qui teste si un terme contient un redex.
2. Ecrire une fonction `etape_NOR t` qui réalise une étape de la stratégie NOR : C'est-à-dire renvoie le terme obtenu par β-conversion en choisissant de réduire le redex le plus à l'extérieur et le plus à gauche. (En cas de problème de capture de variable, la fonction devra appliquer une ou plusieurs α-conversions préalablement à la β-conversion.)
3. Ecrire une fonction `normalise t` qui calcule la forme normale d'un terme (si elle existe).

6 Modélisation

1. Définir les λ-termes pour *VRAI*, *FAUX* et *COND* (conditionnelle). Vérifier que *COND VRAI* u v se réduit en u (en utilisant plusieurs β-conversions) et que *COND FAUX* u v se réduit en v.
2. Définir quelques entiers de Church et les opérations de base sur les entiers : successeur, addition, multiplication. Vérifier sur quelques exemples que ces opérations se comportent correctement.
3. Définir le combinateur de point fixe de Turing et en déduire un modèle de la fonction factorielle. Vérifier que ce modèle est correct en prenant quelques exemples.