

TME Lambda Calcul

On souhaite implémenter quelques opérations sur les λ -expressions vues en cours. Nous représenterons les variables par des chaînes de caractères. Un λ -terme (ou simplement terme) sera donc une valeur du type

```
type var = string

type lambda =
  | Occ of var
  | App of lambda*lambda
  | Abs of int*lambda
```

Un λ -terme de la forme `App t_1 t_2` s'appelle une application (noté $t_1 t_2$), un λ -terme de la forme `Abs x t` s'appelle une abstraction (noté $\lambda x.t$) et un terme de la forme `Occ x` se nomme une occurrence de la variable x . Ce sont donc des arbres dont les feuilles sont des occurrences de variable et dont les noeuds sont des applications ou des abstractions.

Exercice 1

Une occurrence `Occ x` d'une variable x est dite liée dans un lambda terme t si elle apparait comme feuille d'un sous arbre s de t de la forme `Abs x t` . Elle est dite libre sinon. Un λ -terme est dit clos s'il ne contient aucune variable libre.

Pour pouvoir accéder aux sous termes d'un terme nous devons les désigner par un chemin depuis la racine du terme. Nous utiliserons le type

```
type lambda_kind =
  | InFun
  | Inapp
  | Inarg
```

ainsi la valeur `Infun` désignera le sous-arbre t d'un terme $\lambda x.t$, `Inapp` (resp `Inarg`) désignera le sous-arbre t_1 (resp t_2) d'un terme $(t_1 t_2)$. Un chemin dans l'arbre sera alors identifié par une liste de valeurs du type `lambda_kind`. Bien sûr la liste vide désignera le terme en entier.

Question 1.1

Écrire la fonction `chemin_valide` qui prend en arguments un terme t et un chemin c pour retourner vrai si et seulement si c désigne bien un sous terme de t .

Question 1.2

Écrire la fonction `sous_terme` qui prend un terme et un chemin en arguments pour retourner le sous terme correspondant. On pourra déclencher l'exception `No_such_term` si le chemin ne désigne pas un sous terme.

Question 1.3

Écrire la fonction `est_occurrence` qui prend un terme, un chemin dans le terme et une variable en arguments pour retourner vrai si le sous terme est une occurrence de la variable et faux sinon.

Question 1.4

Écrire la fonction `est_libre` (resp `est_liee`) qui prend un terme, un chemin dans le terme et une variable en arguments et retourne vrai si le sous terme correspondant est une occurrence libre (resp liée) de la variable et faux sinon. On pourra déclencher l'exception `Not_an_occurrence` lorsque le sous terme n'est pas une occurrence de la variable et utiliser un argument supplémentaire qui est la liste des variables liées par le chemin.

Question 1.5

Écrire directement la fonction qui donne les variables libres d'un λ -terme.

Exercice 2

Deux λ -termes t_1 et t_2 sont dits α -équivalents si on obtient t_2 à partir d'un renommage des variables de t_1 . On veut pouvoir se donner quelques outils pour tester si deux λ -termes sont α -équivalents.

Une substitution de variables est une fonction (partielle) de l'ensemble des variables vers lui-même. Nous utiliserons des listes d'associations dont les clefs sont des variables et dont les valeurs sont aussi des variables pour gérer les substitutions de variables.

Nous dirons qu'une substitution σ lie une variable v si v apparaît en tant que clé dans la liste représentant σ . C'est à dire si $\sigma(v)$ est défini. La liste vide code donc la substitution qui ne lie aucune variable.

Question 2.1

Écrire la fonction qui étant donnés deux termes t_1 et t_2 et une substitution σ teste si t_1 et t_2 sont égaux modulo σ . On pourra déclencher l'exception `Not_bound` si t_1 contient une variable non liée par σ .

Question 2.2

On étend une substitution σ à un λ -terme en une substitution σ_λ définie par :

- l'occurrence de $\sigma(v)$ si le terme est une occurrence de v ,
- l'application de $\sigma_\lambda(t_1)$ sur $\sigma_\lambda(t_2)$ si le terme est l'application de t_1 sur t_2 .
- l'abstraction par $\sigma(v)$ de $\sigma_\lambda(t)$ si le terme est l'abstraction par v de t .

Écrire la fonction `sigma_1`.

Question 2.3

Étant donnés t_1 et t_2 deux termes et une substitution σ tels que t_1 soit égal à t_2 modulo σ . À quelle condition $\lambda v_1.t_1$ et $\lambda v_2.t_2$ sont-ils α -équivalents ?

Question 2.4

Écrire la fonction qui étant donnée une substitution σ et deux variables v_1 et v_2 retourne la substitution σ augmentée de la liaison de v_1 à v_2 . On déclenche une exception `Already_bound` si v_1 est déjà renommée en autre chose que v_2 par σ .

Question 2.5

En déduire une fonction qui étant données deux substitutions σ_1 et σ_2 retourne une substitution liant à la fois les variables liées par σ_1 et σ_2 .

Question 2.6

On se donne des termes f_1 , f_2 , a_1 et a_2 ainsi que deux substitutions σ_f et σ_a . On suppose que f_1 et f_2 sont égaux modulo σ_f et que a_1 et a_2 sont égaux modulo σ_a . À quelle condition les applications $(f_1 a_1)$ et $(f_2 a_2)$ sont-elles α -équivalentes ?

Question 2.7

En déduire une fonction qui étant donnés deux termes t_1 et t_2 retourne une substitution σ telle que t_1 et t_2 soient égaux modulo σ et déclenche une exception sinon.

Question 2.8

Modifier les fonctions précédentes pour que substitutions ne prennent en compte que les variables libres des termes.