Modèles de Calcul [Lambda-Calcul]
Initiation, syntaxe et portées

Pascal Fradet, Jean-François Monin, Catherine Parent-Vigouroux

Rappels et notations

Soit V un ensemble infini dénombrable de variables. On définit inductivement l'ensemble Λ des λ-termes par

$$\Lambda := x \mid (\lambda x.\Lambda) \mid (\Lambda\Lambda) \qquad \text{avec } x \in \mathrm{V}.$$

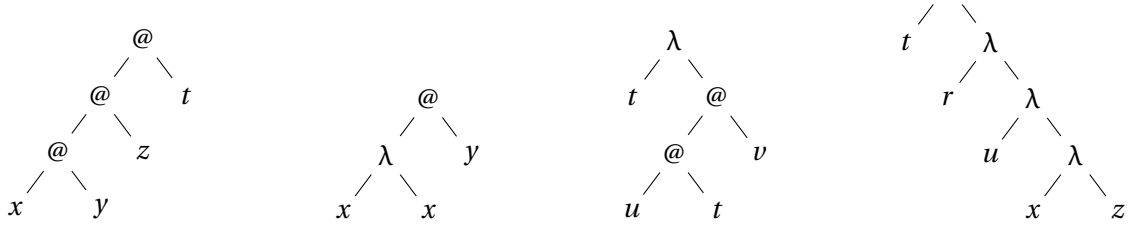
Attention à l'usage des parenthèses : l'application d'une fonction f à son argument x ne se note *pas* $f(x)$ mais $(f\ x)$, ou tout simplement $f\ x$ lorsqu'il n'y a pas d'ambiguïté.

On représente une fonction à deux arguments x et y par $\lambda x.(\lambda y.\mathrm{T})$. En appliquant cette expression à un premier argument U, on obtient $(\lambda x.(\lambda y.\mathrm{T}))\,\mathrm{U}$; puis en l'appliquant à un second argument V, on obtient $((\lambda x.(\lambda y.\mathrm{T}))\,\mathrm{U})\,\mathrm{V}$. On peut alors β-réduire cette expression en deux étapes, ce qui donne $(\lambda y.\mathrm{T}[x := \mathrm{U}])\,\mathrm{V}$, puis $\mathrm{T}[x := \mathrm{U}][y := \mathrm{V}]$, qui est comme attendu le corps T de la fonction dans lequel x et y ont été respectivement substitués par U et V. On adopte alors les abréviations et conventions suivantes :

- $\lambda x.\lambda y.\mathrm{T}$ et $\lambda xy.\mathrm{T}$ sont des notations abrégées pour $\lambda x.(\lambda y.\mathrm{T})$
- l'application associe à gauche : $\mathrm{U\,V\,W}$ est la notation abrégée de $(\mathrm{U\,V})\,\mathrm{W}$; si l'on veut indiquer l'application de U au résultat de l'application de V à W, il faut écrire : $\mathrm{U}\,(\mathrm{V\,W})$
- l'application a priorité sur l'abstraction : $\lambda x.\mathrm{U\,V}$ désigne $\lambda x.(\mathrm{U\,V})$ et non pas $(\lambda x.\mathrm{U})\,\mathrm{V}$.

1 Arbres syntaxiques

1. Donner le λ-terme correspondant à chacun de ces arbres syntaxiques :



2. Pour chacun des termes suivants, donner l'arbre qui lui correspond :

1/ $(\lambda x.(\lambda v.x))$
2/ $(\lambda x.(x\ y))$
3/ $(x\ (y\ t))$
4/ $(\lambda v.(\lambda t.(\lambda r.((u\ x)\ z))))$
5/ $((x\ y)\ (((z\ t)\ t)\ v))$
6/ $(((\lambda v.(((u\ t)\ v)\ r))\ (y\ z))\ (\lambda x.x))$

3. Simplifier au maximum le parenthèsage des termes de cet exercice.

2 Abréviations

Donner la notation complètement parenthésée et non abrégée des λ-termes suivants :

1/ $(x\ y\ z)$
2/ $\lambda v.u\ x\ z$
3/ $\lambda vxy.u$
4/ $\lambda vtr.(u\ x\ z)$
5/ $\lambda x.(x\ (\lambda y.y\ x))$
6/ $(u\ x\ (y\ z)(\lambda v.v\ y))$
7/ $(\lambda xyz.(x\ z\ (y\ z))\ u\ v\ w)$
8/ $(w\ (\lambda xyz.(x\ z\ (y\ z))\ u\ v))$

3 Définitions de fonctions

Rendre à chaque terme sa définition en français.

1/ $\lambda x.\, x$

2/ $\lambda x.\, y$

3/ $\lambda x.\lambda v.\, x$

4/ $\lambda x.\lambda y.\, x\ y$

5/ $\lambda x.\, x\ x$

1/ Fonction à deux arguments, et qui rend le premier d'entre eux.

2/ Fonction identité.

3/ Fonction qui applique son argument à lui-même.

4/ Fonction à deux arguments, qui rend l'application du premier au deuxième.

5/ Fonction à un argument, et qui rend une valeur extérieure ne dépendant pas de cet argument.

4 Variables libres / liées

Pour chaque λ-terme, donner les occurrences libres et liées, puis marquer la portée de chaque variable x.

1/ $\lambda x.\, x\ y$

2/ $\lambda v x y.\, u\ v\ y$

3/ $\lambda x.x\ (\lambda y.\, y\ x)$

4/ $(\lambda x y z.\, x\ z\ (y\ z))\ u\ x\ w$

5/ $u\ x\ (y\ z)\ (\lambda x.\, x\ y)$

5 Redexes et Réduction

Souligner les redexes des λ-termes suivants puis les réduire jusqu'à leur forme normale.

1/ $(\lambda x.\, x\, x)\ (\lambda y.\, x)\ (\lambda y.\, x)$

2/ $(\lambda y.\, x)\ ((\lambda y.\, x)\ (\lambda y.\, x))$

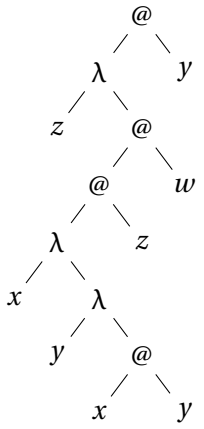
3/ $\lambda y.\, x\, (\lambda y.\, x)\ (\lambda y.\, x)$

4/ $\lambda y.\, (\lambda y.\, x)\ (\lambda y.\, x)\ x$

6 Capture de variable libre

1/ Le λ-terme $(\lambda x y.\, x y)\ y$ comporte un redex. Le réduire, avec ou sans capture de variable libre, et donner la signification intuitive des fonctions obtenues. Laquelle faut-il retenir ?

2/ Repérer les redexes du λ-terme suivant :



Le réduire en utilisant différentes stratégies de réduction et observer ce qui se passe si on oublie de se prémunir contre la capture des variables libres.

7 Évaluations en Coq

Coq est un assistant à la preuve qui embarque un λ-calcul typé avec constantes que nous utiliserons plus tard. Nous avons défini en Coq le λ-calcul pur et non typé comme des expressions de type `lexp`. Les λ-abstractions $\lambda x.$E se notent avec un λ puis un point haut ou avec un \ puis un point haut un tilde, ce qui donne comme possibilités `λx·E`, `\x·E` et `\x~E`. Les variables autorisées sont limitées aux minuscules de l'alphabet $(a, b, c, \ldots, z)$. Le point haut se saisit sur clavier AZERTY par la combinaison : `AltGR + :` Pour le λ, vous pouvez procéder par copié-collé.

Par exemple, le λ-terme $\lambda v.u\ x\ z$ s'écrit comme l'expression `lexp` :

`λv·u x z  ou  \v·u x z  ou  \v~u x z`

L'expression abrégée $\lambda v x y.u$ peut s'écrire comme l'expression `lexp` abrégée :

`λv x y·u ou  \v x y·u  ou  \v x y~u`

Nous allons utiliser le système de preuves Coq pour vérifier la bonne construction des λ-termes et les évaluer. Récupérer préalablement le fichier `untypedLC.vo` sous votre répertoire de travail. Pour lancer Coq, sous Linux, taper

```
coqide
```

Ouvrez un nouveau fichier et faites le commencer par la requête

```
Require Import untypedLC.
```

qui importe la définition des expressions `lexp`.

Nota Bene : *Toute phrase Coq se termine par un point.*

Pour donner un nom (ici `app`) au λ-terme $\lambda x.\lambda y.x\ y$, on utilise la commande

```
Definition app := \x·\y·x y.
```

Pour voir ce terme et son type on utilise la commande

```
Print app.
```

Pour définir le λ-terme $(\lambda x.\lambda y.x\ y)\ a\ b$ on peut :

- soit procéder directement : `Definition t1 := (\x·\y·x y) a b.`
- soit sous forme abrégée : `Definition t2 := (\x y·x y) a b.`
- soit utiliser la définition précédente : `Definition t3 := app a b.`

Pour voir la forme dépliée de `t3` (par exemple), utiliser la commande

```
Compute t3.
```

On peut également obtenir (toujours par la commande `Compute`, voir-ci-dessous) des informations sur un terme ou effectuer des réductions au moyen des fonctions suivantes.

- `vlibres` : rend l'ensemble des variables libres d'une `lexp`;
- `vliees` : rend l'ensemble des variables liées d'une `lexp`;
- `pos_redexes` : rend l'ensemble des positions de redexes d'une `lexp`;
- `see_redex` n : rend le redex numéro n d'une `lexp`;
- `red_beta` n : rend une `lexp` après β-réduction de son redex numéro n;
- `red1_cbn` : réduit le redex le plus externe et le plus à gauche (appel par nom);
- `red1_cbv` : réduit le redex le plus interne et le plus à droite (appel par valeur);

- `show_cbn` : montre la réduction en appel par nom de l'expression jusqu'à la forme normale ;
- `show_cbv` : montre la réduction en appel par valeur partant de la droite de l'expression jusqu'à la forme normale.
- `show_cbv_l` : montre la réduction en appel par valeur partant de la gauche de l'expression jusqu'à la forme normale.
- `show_wcbn`, `show_wcbv`, etc. : variantes des réductions précédentes sans réduction sous des λ
- `fnorm` : rend `true` si la `lexp` est sous forme normale
- `eq_lexp` : rend `true` si deux `lexp` sont égales syntaxiquement (modulo renommage) ;
- `equiv_lexp` : rend `true` si deux `lexp` sont équivalentes (i.e., ont la même forme normale).

Pour utiliser ces fonctions, on écrira

```
Compute (fct exp).
```

Par exemple

```
Compute (show_cbn t3).
```

Exercices

1/ Vérifier pour deux expressions de la section 4 vos résultats pour les variables libres et liées.

2/ Vérifier pour deux expressions de la section 5 vos résultats pour l'ensemble des redexes et la réduction.

3/ Observer la réduction (par `pos_redexes`, `see_redex`, `red_beta`, puis par `show_cbn`, `show_cbv`) des deux expressions de la section 6 et l'absence de capture.

4/ Traduire les λ-termes suivants en expressions `lexp`

— $(\lambda x\ y.x)\ (\lambda a.a)\ ((\lambda b.b)(\lambda c.c))$
```
Definition exp1 := ...
```
— $(\lambda x.xx)\ ((\lambda a.a)\ (\lambda b.b))$
```
Definition exp2 := ...
```

Réduire `exp1` et `exp2` en appel par nom (`show_cbn`) et en appel par valeur (`show_cbv`). Noter les points communs et les différences. Quelle est selon vous la meilleure stratégie (cbv ou cbn) ? Essayer également `show_wcbn` et `show_wcbv`.