

# Evaluateur de $\lambda$ -calcul

Application web pour l'évaluation de  $\lambda$ -calcul avec représentation graphique

Vincent Botbol – Mathieu Chailloux

27 octobre 2013

- $\lambda = \text{"I"}$
- arguments séparés du corps par "."
- plusieurs arguments, curryfiés plus tard
- application associative gauche ( $xxx \Rightarrow (x\ x)\ x$ )

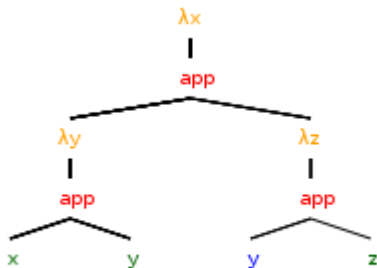
## Exemples :

- $lxy.y\ x$
- $(lx.xx)(lx.xx)$
- $abc$

```
type term =  
| Const of string           (* Constants and  
                             free variables *)  
| App of term * term        (* Applications *)  
| Abstr of string * term     (* Abstractions *)  
| Var of int                (* Bound variables *)
```

Variables liées codées par un entier.

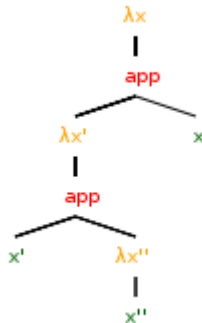
Valeur de cet entier = indice de l'abstraction en remontant l'arbre



Occurences liées :

- $x \iff 1$
- $y \iff 0$
- $z \iff 0$

- Changer le nom des variables liées pour éviter les ambiguïtés.
- Ambiguïté quand il y a 2 abstractions de même argument dans la même branche.

$$\lambda x. (\lambda x. x (\lambda x. x)) x$$


Une seule règle de réduction :

$$(\lambda x.M) N \rightarrow B[N/x]$$

Substitution du paramètre par l'argument grâce aux indices de *Bruijn*.

Dans la formule :  $(\lambda x. \textcolor{red}{M}) \textcolor{blue}{N} \rightarrow B[\textcolor{blue}{N}/x]$

Faut-il évaluer d'abord  $\textcolor{blue}{N}$  (l'argument) ou  $\textcolor{red}{M}$  (le corps de la fonction) ?

2 stratégies ont été implémentées :

- *call-by-name* : On évalue d'abord le corps de la fonction.
- *call-by-value* : On évalue d'abord l'argument.

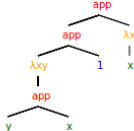
- Réalisée en *OCaml*
- *Js\_of\_ocaml* : outil d'inter-opérabilité d'*OCaml* vers *JavaScript*
- On accède au meilleur des deux mondes



- Visualisation arborescente des termes
- Implémentation d'un algorithme d'agencement des noeuds
- On dessine dans un canvas HTML5 (grâce à *js\_of\_ocaml*)

# Interface Graphique – Utilisation

`(λx.yx)((λx.x) 1) (λx.x)`    Call-by-value    Evaluate



Prev    Next

```
(λx.y x) ((λx.x) 1) λx.x  
(λx.y x) 1 λx.x  
(λy.y 1) λx.x  
(λx.x) 1  
1
```

`http://htmlpreview.github.io/?https://github.com/vincent-botbol/lambda-eval/blob/master/index.html`