PROJ 2

Pitois François et Fernandez Simon

ENS Lyon

17 mai 2017

Fonctions récursives

Gestion des fonctions récursives de l'interpréteur par évalution paresseuse :
Dès que l'interprésteur voit un `let rec`, il le met dans l'environnement en tant que tel.
L'interpréteur ne cherche à évaluer ce binding que quand il applique cette fonction récursive.

Machine ZINC

- Récursion terminale

  ```
  let f x = 1 in
  let g y = f (x+1) in g 2
  ```

  On détecte quand on peut éviter une étape quand on rend la main
- Simplification des fonctions à plusieurs variables

  ```
  let f x y z = ... in f 1 2 3 ;;
  CLOS(CLOS(CLOS(...)))
  ```

  On empile tous les arguments et on y accède en une seule fois

Pattern

```
let pattern = expr in expr ;;
fun pattern -> expr ;;
```

Une pattern peut être :

- nom de variable
- `(pattern, pattern)`
- `pattern : type`
- `pattern -> pattern`

Indice de De Bruijn

Simplification des noms de variables
Éviter les comparaisons de chaines de caractères
Indices dans l'environnement

```
let x = 3 in let y = 4 in x + y
```

devient alors

```
let 3 in let 4 in 0+1
```

Conversion de fouine + blabla vers fouine

Fait en fouine
On utilise des noms de variable réservés pour repérer les zones du programme à modifier.