Programación Funcional

Ingeniería Informática 2005

Práctica #1 (λ-Cálculo)

Ejercicio 1 (**λ-términos**) Implemente un tipo de dato (`term`) que represente los términos del λ-cálculo. Para las variables, utilice el tipo `string`. Los tipos están definidos en el módulo `Tipos`.

```
type variable = string

type term = Var of variable
          | Abs of variable * term
          | App of term * term
```

Defina valores `term` para representar a :

$I \equiv (\lambda x.x)$ $\quad I' \equiv (\lambda y.y)$ $\quad Ky \equiv (\lambda x.y)$ $\quad T \equiv (\lambda x.(\lambda y.x))$

$F \equiv (\lambda x.(\lambda y.y))$ $\quad E1 \equiv (\lambda x.(x\ z))$ $\quad E2 \equiv (\lambda x.a\ x)\ ((\lambda y.b\ y)\ c)$ $\quad E3 \equiv \lambda x.x\ (\lambda y.y\ x)$

$A \equiv (\lambda x.x\ x)$ $\quad E4 \equiv (A\ A)$ $\quad E5 \equiv A\ (y\ z)$ $\quad E6 \equiv (\lambda x.a)\ A$

Para leer y presentar los términos use las siguientes funciones:.

```
val Pretty.term : Tipos.term -> string
val Lector.term : string -> Tipos.term
```

un ejemplo de uso es:

```
let e1 = Lector.term "/x.x z";;
val e1 : Tipos.term = Abs ("x", App (Var "x", Var "z"))
let resultado = Pretty.term e1;;
- : string = "/ x . x z"
```

Opcional Definir la función `Pretty.term` de forma que minimice el número de paréntesis.

Ejercicio 2 (**Variables libres y ligadas**)

Implemente las funciones que calculen el conjunto de variables libres (`fv`) y ligadas (`bv`) de un λ-término:

```
val bv : Tipos.term -> Tipos.variable conjunto
val fv : Tipos.term -> Tipos.variable conjunto
```

Ejercicio 3 (**Substitución**) Implemente la función que realiza la substitución **segura** de una variable por un `term`:

```
val subst: Tipos.term -> Tipos.variable -> term -> term
```

de tal forma que `subst M x N` $\equiv M[N/x]$.

En el módulo `Fresca` está definida la función:

```
fresca : unit -> Tipos.variable
```

que llamada de la forma `fresca ()` devuelve una variable distinta cada vez que se llama.

Ejercicio 4 (**Orden Normal de Reducción**) Implemente las funciones de reducción dadas en teoría:

```
hnf: Tipos.term -> Tipos.term
whnf: Tipos.term -> Tipos.term
nf: Tipos.term -> Tipos.term
wnf: Tipos.term -> Tipos.term
vnf: Tipos.term -> Tipos.term
vwnf: Tipos.term -> Tipos.term
```