Auxiliar 6
CC41A - CC54A
Lenguajes de Programación

Profesor: Éric Tanter
Aux: Richard Ibarra, Oscar E. A. Callaú
{*etanter,ribarra,oalvarez*}*@dcc.uchile.cl*

`Santiago - Chile, Abr/28/2008`

1. P1

Extendiendo F1WAE, para ello ud. tiene que:

1.1. P1-1

Agregar la siguiente regla de producción a la GA del F1WAE:

$$F1WAE \rightarrow \ if0 \ \ cond : F1WAE \ \ inst\_si : F1WAE \ \ inst\_sino : F1WAE$$

donde si la `cond` es igual a `0` entonces ejecuta las instrucciones de `inst_si` y si es diferente de `0` entonces ejecuta las instrucciones de `inst_sino`

1.2. P1-2

Ahora implemente las siguientes funciones usando su lenguaje F1WAE

- `fact :: Number ->Number`, ej. `(fact 4) ->24`
- `fibb :: Number ->Number`, ej. `(fibb 5) ->(0 + 1 + 1 + 2 + 3)`

Si no puede implementar alguna de estas funciones en su lenguaje, explique porque y que es lo que le falta para hacerlo. Por último indique que tipo de funciones no puede implementar.

2. P2

Explorando la substitución diferida.

2.1. P2-1

Un compañero propone cambiar la definición de la gramática de `DefrdSub` por:

```
(define-type DefrdSub
  [mtSub]
  [aSub (name symbol?) (value F1WAE?) (ds DefrdSub?)])
```

Que repercusiones cree ud. que tiene esta nueva definición?

2.2. P2-2

Agregue el `azucar sintáctico` para que nuestro F1WAE acepte multiples declaraciones dentro de una definición de `with`, ej.

```
{with {
   {x 1}
   {y {+ x 1}}
   {z {+ y x}}
 }
 {+ x {+ y z}}
}
```

3. P3, (propuesto)

Elabore una tabla comparativa de las ventajas y desventajas de tener en tu interprete:

- Funciones de 1er orden (ej. Java, C sin funciones referenciadas)
- Funciones de orden superior (ej. Lenguajes que aceptan retornar una función como salida de otra función)
- Funciones de 1era clase (ej. Scheme, Haskell, ...)