Lenguajes de Programación
Auxiliar No2

Auxiliares: Kenji Maillard

20/03/2020

1 Resumen

- Test con `test` y `test/exn`

```
(test (mifuncíon args) resulto)
(test/exn (mifuncíon args) "mi error")
```

- Definir nuevos tipos con `deftype`:

```
(deftype typename
   (constructor1 arg1 ... argn)
   ...)
```

- Analisis de estructuras con `match`:

```
(match val
   [patrón resulto]
   ...)
```

  Unos patrones importantes:

patrón	**accepta**
`var`	todo
`'simbolo`	exactamente el simbolo
`(constructor arg1 .. argn)`	el constructor con sus argumentos
`(list arg1 .. argn)`	una lista con n elementos
`(? pred patrón)`	lo mismo que `patrón` si `(pred val)` es `#t`

Si tiene una duda sobre una función, puedes usar el Help Desk (menú Help en DrRacket).
Recuerde enunciar el contrato y escribir tests.

2 Ejercicios

1. *Árbol binario.* Un árbol binario se puede describir con la BNF sigiente:

   ⟨*bt*⟩ ::= `(` `leaf` ⟨*val*⟩ `)`
   | `(` `node` ⟨*bt*⟩ ⟨*val*⟩ ⟨*bt*⟩ `)`

   (a) Como en el curso, defina el típo `BinTree` de los árboles binarios utilizando `deftype`.

   (b) Defina las funciones `map-bt` y `fold-bt` sobre árboles binarios.

   (c) La función `BinTree?` definida por `deftype` retorna `#t` si su argumento es un `leaf` o `node`. Utilize `fold-bt` para definir una función `BinTree-full?` que verifica si su argumento es un `BinTree` valido (en particular si los sub-árboles también son validos).

   (d) Generalize la función `BinTree-full?` para que también tenga predicados para los valores en los `node` y `leaf`.

2. *Manipulacíon de AST.* Consideramos el lenguaje arítmetico definido por la BNF:

   ⟨*Expr*⟩ ::= ⟨*num*⟩
   | `{` `+` ⟨*Expr*⟩ ⟨*Expr*⟩ `}`
   | `{` `-` ⟨*Expr*⟩ ⟨*Expr*⟩ `}`
   | `{` `*` ⟨*Expr*⟩ ⟨*Expr*⟩ `}`
   | `{` `/` ⟨*Expr*⟩ ⟨*Expr*⟩ `}`

   (a) Defina el típo `Expr` utilizando un unicó nodo `binop` para representar las operaciones binarias.

   (b) Defina una funcíon `parse` tomando una s-expresíon como argumento y produciendo un `Expr`.

   (c) Cual es la relacíon entre el típo `Expr` y el típo `BinTree` ?

   (d) Optimizacíon: define una funcíon `opt` tomando una `Expr` y optimizando todas las occurencias de `{+ 0 x}`, `{+ x 0}`, `{* 1 x}`, `{* x 1}` con `x`

3. *Booleanos.*

   (a) Modifique el lenguaje `Expr` para que tenga elementos booleanos. Recuerde incluir las operaciones tipicas como `and` y `or`.

   (b) Define una funcíon `eval` calculando el valor de una `Expr`.

   (c) Agregue las operaciones `=`, `<` y `>` para comparar numeros.

   (d) Una expresíon puede tener típo `'num` o `'bool`. Qué son los tipos de las expresiones siguientes ?

   ```
   42   {or #t {and #f #f}}   {+ 72 {- {* 8 3} 2}}   {/ 7 0}   {+ #t 6}
   {and {< {+ 2 3} 7} #f}   {= #t #f}   {or {> 3 {- 5 4}} {+ 1 2}}
   ```

   (e) Analisis de típo: define una funcíon `infer-type` tomando una `Expr` y retornando el típo `'num` o `'bool` de la expresíon.

4. *Identificadores enlazados, identificadores libres.*

t

```
{with {x 3}
  {with {y 7}
    {with {z 2}
      {- {+ x y} z}}}}
```

u

```
{with {x y}
  {with {y x}
    {+ y x}}}
```

v

```
{+ {with {x 3} x}
   {- {with {y 2} x}}
      {with {x 7} x}}
```

(a) En los programas **t, u , v** utilizando el lenguaje del curso, cuales tienen identificadores libres ?

(b) En cada programa, dar el alcance de `y`.

(c) En cada programa, dar las ocuriencias enlazadas de `x` y su sitio de enlace.