Investigación de Lenguajes - ML

Gabriel Falcones Angel Sampin

23 de octubre de 2013

1. Introducción

ML es un lenguaje de programación de tipo funcional de tipificado estático, desarrollado en 1973 por Robert Milner en la Universidad de Edimburgo. Su nombre es un acrónimo de Meta Lenguaje, desarrollado para realizar demostraciones automáticas de teoremas en el sistema LCF.

Difiere de otros lenguajes de programación funcional como Haskell, en que también permite la programación de tipo imperativa, al poseer identificadores, que tienen la apariencia de nombres de variables en lenguages imperativos.

ML ha producido varios dialectos usados en la actualidad, entre ellos destacan Standard ML y OCaml. Además ha sido influencia para otros lenguajes como Haskell,Miranda, Cyclone, Clojure e incluso C++.

Las fortalezas de ML son en su mayoría aplicadas en la manipulación y diseño de lenguajes(compiladores y analizadores), pero es un lenguaje de propósito general aplicado también en sistemas financieros, programacion cliente/servidor, bioinformática, etc

2. Características

ML, al ser un lenguage funcional, fue modelado de manera similar a las funciones matemáticas. En vez de usar variables y asignaciones, se enfoca en la aplicación de funciones, condicionales y de la recursión para el control de la ejecución, y formas funcionales para construir funciones complejas.

Utiliza una sintaxis más relacionada a los lenguajes imperativos que a otros lenguajes funcionales como LISP. ML incluye análisis estático de tipos, inferencia de tipos,polimorfismo parametrizado, manejo de excepciones, una variedad de estructuras de datos, y tipos de datos abstractos.

ML posee también declaración de tipos, pero debido a la característica de inferencia de tipos, por lo general no se lo utiliza.

Considere la siguiente función:

```
fun square(x) = x * x;
```

ML determina el tipo de dato tanto del parámetro como del valor que retorna la función gracias al operador *. Debido a que es un operador aritmético, se asume que tanto el parámetro x y el valor que retorna la función son numéricos, por defecto asume que son tipo int. Ml no permite la coercion de parámetros: no es posible siquiera usar un entero cuando se espera un float, sin hacer el casting explicitamente. Tampoco permite sobrecarga de funciones.
Si la función anterior fuese usada de la siguiente manera:

```
square(2.75);
```

Esto causaría un error, porque ML no coerciona valores reales a valores enteros. Si quisieramos que la función square acepte valores reales, deberíamos escribirla de la siguiente manera:

```
fun square(x:real) = x * x;
```

Y debido a que ML no soporta la sobrecarga de funciones, esta versión no podría coexistir con la versión int anterior.

Es posible usar el llamado de patrones (pattern matching) para realizar multiples definiciones de funciones. Las diferentes definiciones de una función que dependen de la forma del parámetro son separadas por un símbolo **OR** (|). Por ejemplo, usando pattern matching, la función factorial puede ser escrita de la siguiente manera:

```
fun fact(0) = 1
| fact(1) = 1
| fact(n:int) = n * fact(n - 1);
```

Si **fact** es llamada con el parámetro 0, la primera definición es usada, si el parámetro es 1, utiliza la segunda definición, para cualquier otro caso, utilizará la tercera definición.

ML soporta Currificación(Currying), que es el proceso de reemplazar una función que acepta múltiples parámetros con otra función que acepta solo un parámetro y retorna otra función que toma los otros parámetros.
Por ejemplo:

```
fun add a  = fn b => a + b;
```

Esta función toma un parámetro entero (a), y devuelve una función que también toma un parámetro entero (b). Una llamada a esta función no necesita las comas (,) para separar los parámetros, de la siguiente manera:

```
add 4 5  (* El llamado a esta funcion devuelve 8 *)
```

Esta misma función **add** puede escribirse de una manera más sencilla:

```
(* Syntactic sugar para una funcion que utiliza Currying *)
fun add a b = a + b;
```

A diferencia de Haskell que utiliza lazy evaluation, ML utiliza eager evaluation(o greedy evaluation), lo que significa que todas las subexpresiones son siempre evaluadas.

3. Historia

ML se refiere como lenguaje funcional impuro, ya que este permite los efectos secundarios, y a la programcaion imperativa El lenguaje de programacion ML, conocido como SML, inspirado en algunos conceptos fundamentales de la informática; pone a disposición:

- Los árboles y otros tipos de datos recursivos declarado y pueden utilizarse sin mención de los punteros;
- Las funciones son valores de ML estándar, las funciones pueden tener funciones como argumentos y devuelven como resultados;
- Tipo de polimorfismo (Milner, 1978) hace posible para declarar funciones cuyo tipo depende del contexto. Por ejemplo, la misma función puede ser usada para invertir una lista de enteros y una lista de cadenas.

En la actualidad es un lenguaje de propósito general

4. Tutorial de Instalación

- Vaya al directorio C :\ y descomprimir la distribución ML Moscú mediante la ejecución

  unzip win32 mos20bin.zip

  NOTA : Debe utilizar una versión de descompresión hizo preserva archivo largo nombres, : como Info- Zip o Winzip .

  Esto crea un directorio C: \ mOsml con subdirectorios

  mOsml / readme , install.txt bin / mOsml , mosmlc , mosmllex , mosmlyac , camlrunm , y las bibliotecas cargadas dinámicamente copyrght / avisos de copyright doc / manual.pdf , mosmlref.pdf , mosmllib.pdf , ... documentación mosmllib / HTML de la biblioteca ML Moscú examples / algunos programas de ejemplo lib / bytecode y base de las unidades de la biblioteca Herramientas / mosmldep , Makefile.stub

- Añadir C: \ mOsml \ bin a la variable PATH, y establecer el entorno variable ‘ mosmllib ’ a C: \ mOsml \ lib .

  - En el caso de Windows 95/98/ME Si su archivo AUTOEXEC.BAT contención algo como:
    SET PATH = c:\ dos ; ... C:\ mOsml\ bin , establecer MOSMLLIB = C:\ mOsml\ lib
    El sistema se debe reiniciar para que el nuevo entorno variable.
  - En el caso de Windows NT/2000/XP , seleccione Inicio — Configuración — Panel de control — Sistema — Avanzada — Variables de entorno Variables de sistema — — NuevoIntroduzca MOSMLLIB como nombre de variable y C: mOsml lib como valor de la variable .Haga doble clic en Path y añada C: mOsml bin a la variable valor .

- Inicio Moscú ML mediante la introducción de la línea de comandos : mOsml

5. Ejemplo: Hola mundo y otras funciones

```
fun hello =
  "Hello world!";
```

```
(* Funcion que recibe un entero y devuelve su factorial *)
fun fac n =
  if n = 0 then 1
  else n * fac (n - 1)
```

```
 la suma de cada uno de sus valores *)
fun suma lista =
  case lista of
    [] => 0
   | x::xs' => x + suma xs'
```

Referencias

[1] Wikipedia: MetaLenguaje. http://http://es.wikipedia.org/wiki/meta*lenguaje*.

[2] Robert W. Sebesta. *Concepts of Programming Languages*. Pearson, 10th edition, 2012.