Programación Declarativa

Arsenio Cornejo Jordán
Dept. de Matemática
Universidad Nacional de Panamá

mayo 28, 2014

Temas

1 Programación Imperativa

2 Programación Funcional

3 Programación Lógica

4 Programación Declarativa

Programación en Scilab

```
k = 3;
y = k + 1;
```

describe las siguientes instrucciones:

```
k = 3;  //Almacena el número 3
// en la variable k.
s = s + k;  //Suma el número almacenado
//en la variable k
//al almacenado en la variable s,
// y almacena el resultado
// en la variable s.
```

Definición de funciones en Scilab (Prog. Imperativa)

```
function z = suma(lista)
long=length(lista);
k=1;
suma = 0;
while k <= long  //ciclo de instrucciones
    suma =suma + lista(k);
    k = k+1;
end
z = suma;
endfunction
```

Se ejecuta el programa y, al ejecutar el comando:
$-->$ suma([3,5,6])
resulta la respuesta esperada: ans = 14

Temas

1 Programación Imperativa

2 Programación Funcional

3 Programación Lógica

4 Programación Declarativa

Consideremos el problema de sumar los números de una lista dada en el lenguaje funcional Haskell (GHCi, version 7.6.3).
Programa en Haskell:

Programa en Haskell:

$$suma\,(x : xs) = x + suma \quad xs$$
$$suma\,[\quad] = 0$$

Aplicación: suma [3, 6, 5]. Resultado: 14
Podemos verificar el resultado empleando el Modelo de Sustitución:

$$\begin{aligned}
suma[3,6,5] &=> 3 + (suma[6,5]) && (1)\\
&=> 3 + (6 + suma[5]) && (2)\\
&=> 3 + (6 + (5 + (suma[\quad]))) && (3)\\
&=> 3 + (6 + (5 + 0)) && (4)\\
&=> 3 + (6 + 5) && (5)\\
&=> 3 + 11 && (6)\\
&=> 14 && (7)
\end{aligned}$$

Consideremos la función

```
aplicaDosVeces:: (a -> a) -> a -> a
aplicaDosVeces f x = f (f x)
```

Al ejecutarla en el sistema con

```
aplicaDosVeces (+5) 4\\
```

resulta
14
Por otro lado,

```
aplicaDosVeces (++ " ARRIBA") "HASKELL"\\
```

resulta en
`"HASKELL ARRIBA ARRIBA"`
El programa muestra otra característica importante: Haskell es polimorfo en el sentido de que una misma función puede aplicarse a datos de distintos tipos.

Temas

1 Programación Imperativa

2 Programación Funcional

3 Programación Lógica

4 Programación Declarativa

El término **programación lógica** se refiere a lenguajes de programación que se han diseñado tomando en cuenta una lógica dada; es decir:

- El lenguaje de programación representa un lenguaje lógico.
- El control de la computación, es decir, la ejecución del programa correspondiente, se basa en algún mecanismo deductivo asociado a la lógica representada por el lenguaje.

Prolog

Al inicio de la década de los 70, surgió, en Francia el lenguaje Prolog *PROgrammation en LOGic* ; su creación se atribuye a los franceses Alan Colmerauer y Philippe Roussel. El desarrollo posterior ocurrió gracias a científicos como John Alan Robinson (1928–) y Robert A. Kowalski (1941- -).

Como ocurre frecuentemente con la llamada “matemàtica aplicada”, la programación lógica se originó en investigaciones de la lógica; en particular, las investigaciones realizadas por el lógico francés Jacques Herbrand (1908-1931), estimularon la aplicación de la lógica a la creación de lenguajes de programación y a la demostración automática de teoremas.

Consideremos el siguiente programa en Prolog (SWI-Prolog, Versión 6.6.5):

```
mujer(maria).
mujer(luisa).
mujer(juana).
mujer(alicia)
toca_guitarra(luisa).
toca_guitarra(maria).
oye_musica(X) :- toca_guitarra(X).
```

$$\forall x(toca\_guitarra(x) \Rightarrow oye\_musica(x))$$

Consultas:

```
1 ?- mujer(maria).
true.

2 ?- mujer(ana).
false.

3 ?- mujer(X).\\
maria ;
luisa ;
alicia.

oye_musica(luisa).
true.

oye_musica(alicia).
false.
```

Consideremos ahora un programa que permite calcular longitudes de listas:

```
long([],0).
long([X|T], M):-long(T,N), M is N+1.
```

Consultas:

```
?- long([3, 2, 1],X)
X = 3
?- long([3, 2, a,  1],Y)
Y = 4.
```

Temas

1 Programación Imperativa

2 Programación Funcional

3 Programación Lógica

4 Programación Declarativa

La frase “Programación Declarativa” se aplica al uso de lenguajes de programación que permiten:

- Crear funciones al ejecutarse el programa. Una función dada puede crear otra función o emplear alguna función como argumento.
- Tienen su origen en la lógica matemática.
- Facilitan el análisis de programas escritos en otros lenguajes.

La impòrtancia del último punto ya fue señalada por John Backus, en 1977; Backus señaló la necesidad de disponer de lenguajes de programación que facilitasen el análisis formal de programas.

Referencias I


Backus, J. (1978).
Can programming be liberated from the von neumann style?: A functional style and its algebra of programs.
Commun. ACM, 21(8):613–641.


Bratko, I. (2011).
PROLOG Programming for Artificial Intelligence, 4th Edition.
Pearson Education.


Church, A. (1936).
An unsolvable problem of elementary number theory.
American journal of mathematics, pages 345–363.


Colmerauer, A. (1993).
The birth of Prolog.
In *III, CACM Vol.33, No7*, pages 37–52.

Referencias II

Hudak, P. (1989).
Conception, evolution, and application of functional programming languages.
ACM Comput. Surv., 21(3):359–411.

Kowalski, R. A. (1974).
Predicate logic as programming language.
In *IFIP Congress*, pages 569–574.

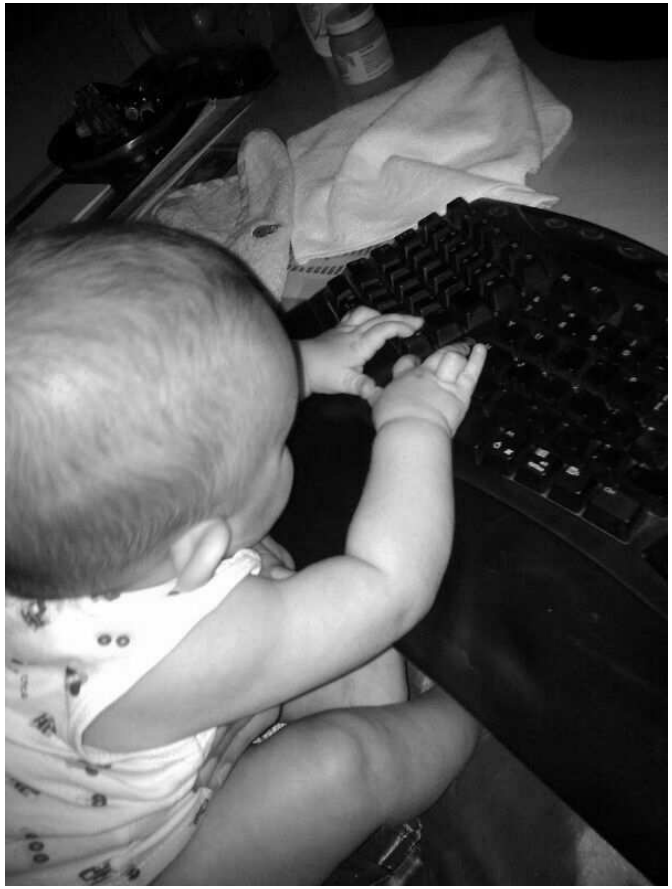
Kowalski, R. A. (1988).
The early years of logic programming.
Commun. ACM, 31(1):38–43.

Landin, P. J. (1966).
The next 700 programming languages.
Communications of the ACM, 9(3):157–166.

Referencias III

Robinson, J. A. (1965).
A machine-oriented logic based on the resolution principle.
J. ACM, 12(1):23–41.

Wadler, P. (1992).
The essence of functional programming.
In *Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 1–14. ACM.



Muchas gracias