Programación Funcional

Curso en la Maestría en Ciencias de la Computación, UNAM. Por Carlos Velarde.

Objetivos

Analizar los principios teóricos y las técnicas fundamentales de la programación funcional.

Introducción

En la programación funcional los programas se componen de definiciones de funciones matemáticas mediante ecuaciones, la computación corresponde a la aplicación de funciones a argumentos respectivos. De esta manera un programa resulta ser un objeto matemático, lo que facilita razonar matemáticamente acerca del mismo, por ejemplo para demostrar que realiza su tarea correctamente o para transformarlo algebraicamente en otro que le sea equivalente, quizá menos simple pero más eficiente.

Los lenguajes funcionales también son atractivos porque permiten la construcción rápida de prototipos de *software*. La experiencia indica que una vez que se dominan las técnicas de programación funcional, éstas también pueden ser usadas para mejorar el diseño de programas imperativos.

Temario

I. Conceptos fundamentales

Expresiones y su evaluación, funciones y tipos. Definiciones simples, por casos y por cazamiento de patrones.

Tipos de datos básicos y compuestos. Variables de tipo. Funciones de orden superior.

II. Listas

Notación y operaciones sobre listas. Iteración.

III. Recursión e inducción

Sobre números naturales, listas y otras estructuras. Funciones de plegado.

IV. Evaluación

Evaluación perezosa. Definiciones corecursivas. Listas infinitas (*streams*). Estructuras cíclicas.

V. Conceptos avanzados

Tipos recursivos. Polimorfismo y clases de tipos. Inferencia de tipos.

VI. Árboles y Gráficas

Árboles binarios. Árboles generales. Gráficas dirigidas. Algoritmos de búsqueda.

VII. Otros conceptos de interés

Cálculo λ. Mónadas. Reconocimiento de patrones. Estructuras de datos puramente funcionales. Cálculo de programas. Complejidad temporal y espacial.

Método de trabajo

1. Exposición de los temas, principalmente por el profesor pero también por los alumnos. Como libros de texto se usarán [1] y [2], otros de consulta recomendable son [3]-[7].

2. Exámenes consistentes en problemas similares a los presentados en los libros de texto.

3. Como lenguaje de programación se usará *Haskell* [8, 9]. Los documentos citados se pueden encontrar en `http://www.haskell.org` , lo mismo que compiladores e intérpretes del lenguaje (se recomienda el intérprete *Hugs*). También hay diversos tutoriales en:

   `http://www.haskell.org/haskellwiki/Books_and_tutorials` .

4. A cada alumno se le asignará como proyecto una exposición basada en alguno de los artículos [10]-[26], en la que debe presentar e ilustrar los programas que aparezcan en la publicación y, de ser posible, aplicaciones diversas que se le ocurran.

Calificación

Proyecto: 40 %, tareas: 20 %, exámenes: 20 %, exposición de temas en clase: 20 %.

Referencias y bibliografía

[1] R. Bird. *Introduction to Functional Programming using Haskell*. Series in Computer Science. Prentice-Hall, 2nd edition, 1998.

[2] K. Doets and J. van Eijck. *The Haskell Road to Logic, Maths and Programming*. King's College Publications, 2004.

[3] G. Hutton. *Programming in Haskell*. Cambridge University Press, 2006.

[4] P. Hudak. *The Haskell School of Expression: Learning Functional Programming through Multimedia*. Cambridge University Press, New York, 2000. Material auxiliar en `http://www.haskell.org/soe/`.

[5] S. Thompson. *The Craft of Functional Programming*. Addison-Wesley, 2nd edition, 1999.

[6] R. Bird and P. Wadler. *Introduction to Functional Programming*. Prentice Hall International, 1988.

[7] Chris Okasaki. *Purely Functional Data Structures*. Cambridge University Press, 1999.

[8] P. Hudak and J.H. Fasel. A gentle introduction to Haskell. *ACM SIGPLAN Notices*, 27(5):T1–T53, May 1992.

[9] Simon Peyton Jones, editor. *Haskell 98 Language and Libraries. The Revised Report*. Cambridge University Press, 2003.

[10] Richard S. Bird. Functional pearl: A program to solve sudoku. *J. Funct. Program.*, 16(6):671–679, 2006.

[11] Richard S. Bird and Sharon A. Curtis. Functional pearl: Finding celebrities: A lesson in functional programming. *J. Funct. Program.*, 16(1):13–20, 2006.

[12] Manuel Chakravarty (editor) et al. The haskell 98 foreign function interface. An addendum to the haskell 98 report. Technical report, 2003. `http://haskell.org`.

[13] Jeroen Fokker. Functional parsers. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, number 925 in Lecture Notes in Computer Science, pages 1–23. Springer–Verlag, 1995.

[14] J. Gibbons. Calculating functional programs. In R. Backhouse, R. Crole, and J. Gibbons, editors, *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*, number 2297 in Lecture Notes in Computer Science, pages 149–201. Springer–Verlag, 2002.

[15] Jeremy Gibbons. A pointless derivation of radix sort. *J. Funct. Program.*, 9(3):339–346, 1999.

[16] Jeremy Gibbons, David Lester, and Richard S. Bird. Functional pearl: Enumerating the rationals. *J. Funct. Program.*, 16(3):281–291, 2006.

[17] John Hughes. The design of a pretty-printing library. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, number 925 in Lecture Notes in Computer Science, pages 53–96. Springer–Verlag, 1995.

[18] Graham Hutton and Erik Meijer. Monadic parsing in haskell. *J. Funct. Program.*, 8(4):437–444, 1998.

[19] Mark P. Jones. Functional programming with overloading and higher–order polymorphism. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, number 925 in Lecture Notes in Computer Science, pages 97–136. Springer–Verlag, 1995.

[20] Mark P. Jones. Composing fractals. *J. Funct. Program.*, 14(6):715–725, 2004.

[21] Simon Peyton Jones. Tackling the awkward squad: monadic input/output, concurrency, exceptions, and foreign-language calls in haskell. Technical report, 2005. `http://research.microsoft.com/users/simonpj`.

[22] Stefan Kahrs. Red-black trees with types. *J. Funct. Program.*, 11(4):425–432, 2001.

[23] John Launchbury. Graph algorithms with a functional flavour. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, number 925 in Lecture Notes in Computer Science, pages 308–331. Springer–Verlag, 1995.

[24] M. Douglas McIlroy. Power series, power serious. *J. Funct. Program.*, 9(3):325–337, 1999.

[25] L.C. Paulson and C. Lawrence. Foundations of functional programming. Technical report, Computer Laboratory, University of Cambridge, 1995.
`http://www.cl.cam.ac.uk/users/lcp/papers/Notes/Founds-FP.pdf`.

[26] Philip Wadler. Monads for functional programming. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, number 925 in Lecture Notes in Computer Science, pages 24–52. Springer–Verlag, 1995.