Programación Funcional

Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Uruguay

Edición 2024

Objetivo del curso

- Este es un curso introductorio de **programación funcional** (PF).
- Veremos conceptos y técnicas de programación relativas a este **paradigma**.
- Nos vamos a focalizar en un abordaje específico a la PF que corresponde a lenguajes **fuertemente tipados** y con **evaluación perezosa** ("lazy evaluation").
- Usaremos el lenguaje Haskell.

Equipo docente

- Alberto Pardo (responsable)
- Marcos Viera (responsable)
- Luis Sierra
- Germán Ferrari

Modalidad del curso

- Esta edición del curso va a ser dictada en forma **híbrida**.
- **Teórico**: Los temas serán introducidos a través de **videos** que iremos dejando disponibles en la plataforma EVA.
- Tendremos dos horarios de **Práctico-Consulta**:
  - Jueves de 18 a 20hs por Zoom
  - Viernes de 9 a 11hs de forma presencial (salón 311)

  En ellos veremos algunos ejercicios de práctico y se atenderán consultas de teórico y de práctico.
- Dada la modalidad del curso se **recomienda fuertemente** el uso de los **foros** de EVA para realizar consultas.

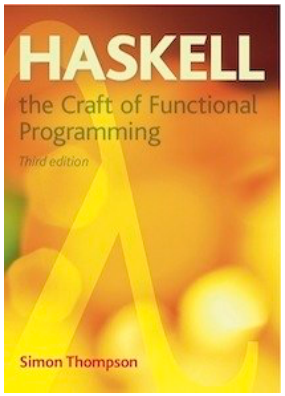
Material del curso

- A medida que avance el curso estarán disponibles en el EVA:
  - El video que presenta cada tema.
  - Las slides correspondientes usadas en el video.
  - Los prácticos.

Material del curso

- A medida que avance el curso estarán disponibles en el EVA:
  - El video que presenta cada tema.
  - Las slides correspondientes usadas en el video.
  - Los prácticos.

- **Bibliografía** Nos basaremos en el siguiente libro:



Haskell: The craft of functional programming
Simon Thompson
Third Edition
Pearson Education Limited
2011

El link al libro se encuentra en el EVA.

A lo largo del curso se indicarán las partes del libro que se van tratando.

Evaluación

- Una **tarea de programación** en las últimas semanas del curso. Puede ser realizada en grupos de hasta 2 estudiantes.

- Una **prueba** al final del curso.

  Habrá **dos fechas** para la prueba.
  - Una en el período de parciales de Noviembre-Diciembre.
  - Otra en el período de exámenes de Diciembre.

- Adicionalmente, a lo largo del curso se propondrán **ejercicios de autoevaluación** a través de la plataforma EVA, cuya resolución es **obligatoria**.

Haskell

- Haskell (`http://www.haskell.org`) es un lenguaje funcional puro con evaluación perezosa.
- Debe su nombre al lógico Haskell B. Curry.
- Es fuertemente tipado; tiene tipado estático e inferencia de tipos.
- Fue definido en 1990 por un comité internacional con el fin de establecer un estándar.

GHC

Usaremos el compilador GHC (Glasgow Haskell Compiler) y su intérprete GHCi (`https://www.haskell.org/ghc`)

Principales Características de la Programación Funcional

Un paradigma diferente

- PF es un **paradigma de programación** de la misma forma que lo son la programación imperativa, la programación orientada a objetos, o la programación lógica.

- **Potentes mecanismos de abstracción y modularización**. Por ejemplo: tipos de datos algebraicos, pattern-matching, funciones de alto orden, etc.

- **Alto nivel de abstracción y una sintaxis minimal**. En PF los programas suelen ser más concisos que en otros lenguajes.

Programas

Un **programa funcional** está formado esencialmente por una lista de definiciones de funciones que se expresan en una notación semejante a la usada en matemática.

Ejemplo:

$$square :: Integer \to Integer$$
$$square\ x = x * x$$

$$min\ (x, y) = \textbf{if}\ x \leqslant y\ \textbf{then}\ x\ \textbf{else}\ y$$

$$errorMsg :: Int \to String$$
$$errorMsg\ n = \texttt{"error tipo "} +\!\!+ show\ n$$

$$twice\ f = f\ .\ f$$

$$pot4 = twice\ square$$

Modelo de computación

- En PF el modelo de computación es diferente.
- La computación de resultados se hace mediante la **evaluación de expresiones**, las cuáles pueden contener llamadas a funciones definidas en el propio programa o importadas de otros módulos.
- Durante la **evaluación** las definiciones de las funciones actúan como **reglas de reducción** que permiten **simplificar** las expresiones.

Evaluación de expresiones

Evaluemos *square* $(3 + 4)$.

Recordemos que *square* $x = x * x$.

Evaluación de expresiones

Evaluemos *square* $(3 + 4)$.

Recordemos que *square* $x = x * x$.

$$
\begin{aligned}
& \mathit{square}\ (3 + 4) \\
= & \{\mathit{def}+\} \\
& \mathit{square}\ 7 \\
= & \{\mathit{def}\ \ \mathit{square}\} \\
& 7 * 7 \\
= & \{\mathit{def}*\} \\
& 49
\end{aligned}
$$

Evaluación de expresiones

Evaluemos *square* $(3 + 4)$.

Recordemos que *square* $x = x * x$.

$$
\begin{aligned}
& \mathit{square}\ (3 + 4) \\
= & \{\mathit{def}+\} \\
& \mathit{square}\ 7 \\
= & \{\mathit{def}\ \ \mathit{square}\} \\
& 7 * 7 \\
= & \{\mathit{def}*\} \\
& 49
\end{aligned}
$$

La expresión 49 se dice que está en **forma normal** o **canónica** debido a que no puede reducirse más.

Evaluación de expresiones (2)

Es la única forma de evaluar *square* $(3 + 4)$?

Evaluación de expresiones (2)

Es la única forma de evaluar *square* $(3 + 4)$? **NO**.

Evaluación de expresiones (2)

Es la única forma de evaluar *square* $(3+4)$? **NO**.

Esta es otra forma:

$$
\begin{aligned}
& \mathit{square}\ (3+4) \\
= & \{\mathit{def}\ \mathit{square}\} \\
& (3+4) * (3+4) \\
= & \{\mathit{def} +\} \\
& 7 * (3+4) \\
= & \{\mathit{def} +\} \\
& 7 * 7 \\
= & \{\mathit{def} *\} \\
& 49
\end{aligned}
$$

Ausencia de efectos laterales

Propiedad

Dada una expresión *e* cualquiera se cumple que: todas las secuencias de reducción de *e* que terminen lo van a hacer en un **mismo valor** *v*.

Ausencia de efectos laterales

Propiedad

Dada una expresión *e* cualquiera se cumple que: todas las secuencias de reducción de *e* que terminen lo van a hacer en un **mismo valor** *v*.

- Esta es una propiedad que caracteriza a la PF y la diferencia de otros paradigmas.
- Se debe a la ausencia de efectos laterales, y en particular, a la **inexistencia** de una **sentencia de asignación**.

Ausencia de efectos laterales (2)

- La propiedad anterior **no** es válida en la mayoría de los **lenguajes imperativos**.
- Eso se debe a que, en los lenguajes imperativos, el modelo de computación se basa esencialmente en **modificar** los valores almacenados en memoria.
- Por lo tanto el **orden** en que se ejecutan las sentencias es **crucial**.
- Por ejemplo, en un lenguaje imperativo la expresión

$$n + (n := 1)$$

va a retornar distintos valores dependiendo de si se evalúa primero el sumando de la izquierda o el de la derecha.

Ausencia de efectos laterales (3)

- La ausencia de efectos hace que en PF las variables se comporten como en la matemática: cada nombre está asociado/ligado a un **único** valor.

- En PF las **variables denotan valores**, a diferencia de lo que ocurre en los lenguajes imperativos en que las variables denotan direcciones de memoria cuyo contenido puede ser modificado.

- Al igual que en la matemática, en PF el valor retornado por una función **sólo** depende de los valores dados como entrada. No hay variables globales que puedan modificar el comportamiento de una función.

- La ausencia de efectos facilita el **razonamiento formal** sobre los programas funcionales, por ejemplo, para probar propiedades de los mismos.