NOMBRE DE LA ASIGNATURA

Año 2017

Carrera/ Plan: (Dejar lo que corresponda)
Licenciatura en Informática Plan 2003-07/Plan 2012

Año: 4to
Régimen de Cursada: *Semestral*
Carácter (Obligatoria/Optativa): Optativa
Correlativas: Conceptos y Paradigmas de Lenguajes de Programación
Profesor/es: Gabriel Baum
Hs. semanales : 6

FUNDAMENTACIÓN

Consistentemente con una tradición de las ciencias de la computación, que se remonta a sus mismos orígenes, el curso de Programación Funcional presenta a la disciplina de diseñar y construir programas como una actividad rigurosa y formalizada, esto es, la noción básica de un programa como objeto matemático sobre el cual es posible razonar con toda precisión y utilizando argumentos lógicos, algebraicos y matemáticos.

La formación de los estudiantes en esta disciplina, intrínsecamente formal, posibilita la integración de conceptos matemáticos y lógicos, adquiridos en años previos, como por ejemplo propiedad, demostración, inducción, etc. con conceptos específicos de las ciencias de la computación (algoritmo, ejecución, etc.), en un ciclo de aprendizaje que incluye asignaturas como computabilidad, lenguajes formales, complejidad algorítmica, que potencian el desarrollo del pensamiento riguroso y formal en los alumnos. Por otra parte, permite la comparación entre diferentes paradigmas de pensamiento algorítmico, enriqueciendo el acervo de herramientas para analizar y resolver problemas.

OBJETIVOS GENERALES

Desarrollar los conceptos y metodologías de programación asociadas con el paradigma funcional. Analizar técnicas y herramientas de Programación Funcional. Realizar trabajos experimentales de resolución de algoritmos con PF.

CONTENIDOS MINIMOS (de acuerdo al Plan de Estudios)

- Conceptos básicos.
- Modelo de Computación del Paradigma Funcional.
- Técnicas Formales
- Aplicación de conceptos: Listas.
- Sistemas de Tipos.
- Técnicas de Diseño Funcional.

- Lambda Cálculo

PROGRAMA ANALÍTICO

1. **Conceptos Preliminares**

* Revisión de la noción de programación y el concepto de programa.

* Propiedades deseables de los programas. Razonamiento y demostración de dichas propiedades.

* Dificultades del modelo clásico de programación para el razonamiento sobre programas.

* Descripción del modelo de programación funcional.

* Características principales de los lenguajes funcionales:
  - o transparencia referencial,
  - o alto orden y currificación,
  - o sistemas de tipos.

2. **Modelo de Computación del Paradigma Funcional**

* Valores y expresiones. Las funciones como valores.

* Mecanismos de definición de expresiones y valores. Ecuaciones orientadas para definir funciones. Sintaxis.

* Visión denotacional y operacional de las expresiones. Modelos de computación mediante reducción. Semántica.

* Ordenes de reducción: reducción aplicativa y reducción normal.

* Sistemas de Tipos. Tipos básicos. Constructores de tipos. Polimorfismo. Sintaxis para valores de cada tipo (caracteres, tuplas, listas, strings, funciones)

* Funciones parciales y totales.

* Funciones de alto orden. Currificación.

3. **Técnicas Formales**

* Demostración de propiedades

  o Noción de propiedad y de demostración. Diferentes formas de garantizar propiedades: por construcción, por chequeo automático, por demostración manual.

  o Algunas propiedades interesantes de los programas: corrección, terminación, equivalencia de programas.

* Inducción/Recursión.
  - o Definición inductiva de conjuntos.
  - o Definición recursiva de funciones sobre esos conjuntos.
  - o Demostraciones inductivas sobre dichas funciones.
  - o Ejemplos: programas, expresiones aritméticas, listas.

4. **Aplicación de Conceptos: Listas**

* Listas por comprensión. Definición y ejemplos. Semántica de listas por comprensión mediante reducción.

* Listas como tipo inductivo. Funciones básicas sobre listas (append, head, tail, take, drop, reverse, sort, elem, etc.).

* Funciones de alto orden sobre listas. Patrón de recorrido: map. Patrón de selección: filter. Patrón de recursión: foldr.

* Demostración de propiedades sobre listas y funciones sobre listas.

5. **Sistemas de Tipos.**

* Nociones básicas. Sistemas de tipado fuerte. Ventajas y limitaciones de los lenguajes de programación con tipos.

* Lenguaje de tipos. Asignación de tipos a expresiones. Propiedades interesantes de esta asignación. Algoritmo de inferencia.

* Mecanismos de definición de tipos nuevos y de funciones sobre ellos. Tipos algebraicos. Ejemplos: enumeraciones, listas, árboles binarios, árboles generales.

6. **Técnicas de Diseño Funcional.**

* Transformación y Síntesis de Programas.
  - o Motivación. Obtención de programas a partir de especificaciones. Mejoramiento de eficiencia, con corrección por construcción.
  - o Transformación de expresiones que utilizan listas por comprensión en expresiones que utilizan map, filter y concat.
  - o Transformación y síntesis de programas. Técnicas y ejemplos.
* Combinadores y Transformadores.
  - o Noción de combinador y transformador.
  - o Estructuración de bibliotecas de funciones basadas en combinadores y transformadores. Ejemplos: parsing, pretty-printing, etc.

7. **Lambda Cálculo**

* Definición del lenguaje. Sintaxis. Definición de sustitución.
* Modelo de computación. Nociones de alfa, beta y eta reducción. Semántica operacional.
* Lambda cálculo como modelo teórico de los lenguajes funcionales.

8. **Temas Adicionales (opcionales)**

* Sistemas inductivos. Ordenes parciales completos (CPOs).
*Lambda Cálculo tipado.

BIBLIOGRAFÏA

BIBLIOGRAFÍA OBLIGATORIA

- R. S. Bird.

Introduction to Functional Programming using Haskell.
Prentice-Hall, 1998.

- R. S. Bird and P. Wadler.

Introduction to Functional Programming.
Prentice-Hall, 1988.

BIBLIOGRAFÍA COMPLEMENTARIA

- A. J. T. Davie.

An Introduction to Functional Programming Systems Using Haskell.
Cambridge University Press, 1992.

- M. J. C. Gordon.

Programming language theory and its implementation.
Prentice-Hall - C.A.R. Hoare Series Editor, 1988.

- M. P. Jones.

Hugs 1.4 - the Haskell user's Gofer System. User manual.
Technical report, Department of Computer Science, University of Nottingham, 1997.

- S. L. Peyton Jones.

The implementation of functional programming languages.
Prentice-Hall - C.A.R. Hoare Series Editor, 1987.

- S. L. Peyton Jones, John Hughes, et al.

Report on the programming language Haskell'98.
Technical report, Yale University, February 1999.
http://www.haskell.org/onlinereport

- Thompson. Simon. The craft of Functional Programming.
- L.C. Paulson. ML for the working programming.
- Richard Bird and Oege de Moor. Algebra of Programming.

METODOLOGÍA DE ENSEÑANZA

De acuerdo a los objetivos y metas expuestos anteriormente tanto en las clases teóricas como los prácticos se enfatizan los aspectos relacionados con la concepción de la programación como una actividad rigurosa y formalizada, esto es, la noción básica de un programa como objeto matemático sobre el cual es posible razonar con toda precisión y utilizando argumentos lógicos, algebraicos y

matemáticos. Conceptualmente, el curso puede dividirse en dos partes: la primera consiste en los conceptos y técnicas fundamentales, y la segunda en el estudio y práctica de las construcciones avanzadas y aplicaciones.
En este sentido, en las primeras clases del curso se repasan los conceptos matemáticos de propiedad , demostración, prueba formal, inducción, en paralelo con la introducción de los conceptos y notaciones propios de la programación funcional. En ese contexto se discuten las similitudes y las diferencias del paradigma funcional respecto del procedural, y se introducen los conceptos característicos de la programación funcional, en particular, transparencia referencial, alto orden y sistemas de tipos.
Una vez establecidos los conceptos fundamentales se aborda el modelo computacional de la programación funcional, introduciendo formalmente los conceptos de valores y expresiones y la semántica denotacional y algebraica (ecuacional) de los programas funcionales. Sin embargo, se pone especial énfasis en la semántica operacional de los programas, trabajando tanto en las teorías como en los prácticos los conceptos sustitución, reducción, formas normales, etc. El módulo culmina con el estudio de los sistemas de tipos, específicamente el sistema de tipos HM, introduciendo los conceptos de tipos primitivos o básicos, constructores, polimorfismo. Finalmente, en ese contexto,se introducen los conceptos de función total y parcial, y de función de alto orden y currificación.
La primera parte del curso finaliza con el estudio de técnicas formales para demostración de propiedades de programas y sobre inducción y recursión como técnicas para definición, construcción y demostración.
La segunda parte del curso consiste en el estudio de las estructuras de datos y las técnicas de diseño típicas de la programación funcional. Respecto de las primeras, se estudian las formas de definición y los operadores de alto orden de las estructuras fundamentales (listas), así como la demostración de propiedades. Se introducen luego los conceptos relacionados a tipos de datos y se estudian los mecanismos de construcción y utilización de tipos avanzados (por ejemplo, árboles binarios y generales). En esta parte del curso se hace especial énfasis en el desarrollo de programas en los trabajos prácticos.
Finalmente, se introducen los conceptos de transformación y síntesis de programas, persiguiendo el objetivo fundamental de integrar el conjunto de conocimientos estudiados anteriormente e introduciendo la noción de que es prácticamente factible desarrollar programas de manera sistemática y rigurosa partiendo de una especificación formal, asegurando la corrección del producto obtenido respecto de su descripción inicial.
Dependiendo del desarrollo del curso, esto es, del rendimiento y nivel observado en las clases y exámenes parciales, se estudian los rudimentos del Lambda Calculo, presentándolo como el modelo teórico que fundamenta la programación funcional.
Opcionalmente, para los estudiantes que demuestren interés en los aspectos fundamentales de la materia, se ofrecen estudios dirigidos sobre temas avanzados, por ejemplo, sistemas inductivos y órdenes parciales completos (CPOs) y lambda cálculo tipado y con sustituciones explicitas.

EVALUACIÓN

Se realiza una evaluación permanente de los alumnos realizando una evaluación de cada uno de los trabajos prácticos así como da la participación en las clases teóricas y prácticas. Se realizan dos evaluaciones parciales integradoras (EPI) que resultan la base para la aprobación de la cursada, combinada con la evaluación individual a lo largo del curso antes mencionada.

La evaluación final puede consistir en un examen escrito y oral, o bien la realización de un proyecto de desarrollo en grupos de no más de tres estudiantes, el cual debe ser presentado y defendido a través de un coloquio

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Fecha	Contenidos/Actividades
1	**07/3/17**	**Teórica**: Introducción, sintaxis, valores y expresiones, scripts **Práctica**: (no comienza)
2	**14/3/17**	**Teórica**: Tipos, currificación **Práctica**: sintaxis, valores y expresiones, scripts
3	**21/3/17**	**Teórica**: Órdenes de evaluación **Práctica**: Tipos, currificación
4	**28/3/17**	**Teórica**: Demostraciones **Práctica**: Órdenes de evaluación
5	**4/4/17**	**Teórica**: Demostraciones (2da parte) **Práctica**: Órdenes de evaluación (2da parte)
6	**11/4/17**	**Teórica**: Inducción **Práctica**: Demostraciones
7	**18/4/17**	**Teórica**: Inducción **Práctica**: Demostraciones
8	**25/4/17**	**Teórica**: Listas **Práctica**: Inducción
9	**22/4/17**	**Teórica**: Tipos algebráicos y abstractos **Práctica**: Listas
10	**29/4/17**	**Teórica**: Tipos recursivos (árboles) y demostraciones **Práctica**: Tipos algebráicos y abstractos
11	**2/5/17**	**Teórica**: Esquemas de recursión en listas **Práctica**: Tipos recursivos (árboles) y demos
12	**9/5/17**	**Teórica**: Esquemas de recursión en listas

		Práctica: Esquemas de recursión en listas
13	**16/5/17**	**Teórica**: Esquemas de recursión en árboles **Práctica**: Esquemas de recursión en listas
14	**23/5/17**	**Teórica**: Lazy evaluation **Práctica**: Esquemas de recursión en árboles
15	**305/17**	**Teórica**: Recursión de cola y teoremas de dualidad **Práctica**: Lazy evaluation
16	**6/6/17**	**Teórica**: Derivación de programas y técnicas de diseño **Práctica**: Recursión de cola y teoremas de dualidad
17	**13/6/17**	**Teórica**: Derivación de programas y técnicas de diseño **Práctica**: Derivación de programas y técnicas de diseño
18	**20/6/17**	**Teórica**: Lambda-cálculo (definición, binding, sustit./reemplazo) **Práctica**: Derivación de programas y técnicas de diseño
19	**27/6/17**	**Teórica**: Lambda-cálculo (semántica por equivalencia) **Práctica**: Lambda-cálculo
20	**4/7/17**	**Teórica:** Lambda-cálculo (programación)
21	**11/7/17**	**Práctica**: Muestra y discusión 1° EPI

Evaluaciones previstas	Fecha
1° Evaluación Parcial Integradora	01/7/2016
2° Evaluación Parcial Integradora	15/7/2016

Contacto de la cátedra (mail, sitio WEB, plataforma virtual de gestión de cursos):
e-mail: gbaum@lifia.info.unlp.edu.ar o fcional@lifia.info.unlp.edu.ar
pagina web: https://catedras.lifia.info.unlp.edu.ar/funcional/

Firma del/los profesor/es