

Planificación de
Análisis de Lenguajes de Programación

Código/s: R-322



Identificación y características del Espacio Curricular

Carrera/s:	Licenciatura en Ciencias de la Computación		
Plan de Estudios:	2010, TO2024	Carácter:	Obligatoria
Bloque/Campo:		Área:	Algoritmos y Lenguajes
Régimen de cursado:	Cuatrimestral		
Cuatrimestre:	6º [LCC], 6º [LCC]		
Carga horaria:	120 hs. / 8 hs. semanales	Formato curricular:	Asignatura
Escuela:	Ciencias Exactas y Naturales	Departamento:	Ciencias de la Computación
Docente responsable: MANZINO, Cecilia			

Programa Sintético

Conceptos Preliminares. Modelo de Computación del Paradigma Funcional. Técnicas Formales. Aplicación de Conceptos: Listas. Nociones básicas de semántica. Técnicas de Diseño Funcional. Lambda Cálculo. Mónadas.

Espacios Curriculares Relacionados

Previos Aprobados:	R-223 - Lógica, R-312 - Estructuras de Datos y Algoritmos II
Simultaneos Recomendados:	
Posteriores:	R-422 - Compiladores

Vigencia desde 2024

Firma Profesor

Fecha

Firma Aprob. Escuela

Fecha

Con el aval del Consejo Asesor:

Fundamentación

La asignatura se ubica en el 1º cuatrimestre de 3º año de la carrera de Lic. en Cs. de la Computación. La misma abarca los fundamentos de los lenguajes de programación desde un enfoque formal. Estudiar los lenguajes de programación de manera formal es fundamental en las ciencias de la computación.

Para describir formalmente un lenguaje se ven conceptos como sintaxis, semántica y sistemas de tipos. Además de la base teórica se estudia cómo aplicar estos conceptos en la creación de lenguajes, usando un paradigma funcional.

Conocer cómo se construye un lenguaje ayuda a las y los estudiantes a entender cualquier lenguaje de programación.

Muchos de los conceptos abordados en la materia se ven con mayor profundidad en la materia Compiladores que corresponde al 4º año de la carrera.

Resultados del aprendizaje

Al finalizar el cursado los/las estudiantes serán capaces de:

RA1 Entender a los lenguajes de programación como la composición de programas más sencillos, desde una perspectiva integral, como ser sintaxis, semántica, sistema de tipos, pretty-printers.

RA2 Implementar analizadores sintácticos, evaluadores, inferidores de tipos y pretty-printers de lenguajes de programación usando un paradigma funcional.

RA3 Comprender el concepto de programación como una actividad rigurosa de abstracción.

RA4 Utilizar con seguridad conceptos del paradigma funcional avanzados, como ser mónadas y funtores.

RA5 Conocer distintas propiedades de los lenguajes de programación que garantizan la corrección de los programas y aplicar técnicas formales para demostrar las mismas.

RA6 Diseñar un plan de trabajo colaborativo y factible, identificando tareas individuales y colectivas, llevándolas a cabo con responsabilidad.

Competencias / Ejes transversales y Resultados del Aprendizaje

Competencia/Eje transversal al que tributa	Nivel	Resultados del Aprendizaje
CGT2-Concepción, diseño y desarrollo de proyectos de informática	Medio	RA1, RA3 y RA5
CGT3-Gestión, planificación, ejecución y control de proyectos de informática	Bajo	RA2, RA4 y RA5
CGT5-Generación de desarrollos tecnológicos y/o innovaciones tecnológicas	Medio	RA2 y RA4
CGS1-Fundamentos para el desempeño en equipos de trabajo	Bajo	RA6
CGS6-Fundamentos para la acción emprendedora	Bajo	RA6

Programa Analítico

Unidad 1

1.1. Nociones básicas de sintaxis. Definición de sintaxis concreta y abstracta.

1.2. Lenguajes formales.

1.3. Sintaxis concreta. Gramáticas Libres de Contexto. Lenguaje generado por una gramática. Árboles de parseo. Gramáticas ambiguas. Desambiguar gramáticas.

1.4. Sintaxis abstracta. Definición de AST (abstract syntax tree) como conjunto inductivo de términos.

- 1.5. Analizadores sintácticos o parsers. Definición.
- 1.6. Implementación de librería de parsers en Haskell.

Unidad 2

- 2.1. Nociones básicas de semántica. Distintos enfoques.
- 2.2. Semántica operacional.
- 2.3. Relaciones de evaluación de paso chico y paso grande.
- 2.4. Árboles de derivación.
- 2.5. Inducción sobre una derivación.
- 2.6. Formas normales.
- 2.7. Propiedades: determinismo, valores como forma normal, unicidad de formas normales, terminación.
- 2.8. Semántica operacional para lenguaje imperativo
- 2.9. Semántica para lenguaje con efectos colaterales: errores, entrada/salida, estado.

Unidad 3

- 3.1. Lambda Cálculo. Origen. Sintaxis. Convenciones.
- 3.2. Variables libres y ligadas. Substitución. Alfa-conversión.
- 3.3. Semántica de Lambda Cálculo. Beta-reducción. Formas normales. Propiedades. Nociones de reducción y de equivalencia. Estrategia de reducción normal.
- 3.4. Programando con Lambda Cálculo. Representación de tipos de datos con fold. Operador de punto fijo.

Unidad 4

- 4.1. Sistemas de Tipos. Chequeo estático y dinámico.
- 4.2. Reglas de inferencia.
- 4.2. Definición de lenguaje seguro mediante las propiedades de Progreso y Preservación. Términos atascados.
- 4.3. Lambda cálculo simplemente tipado. Estilos a la Curry y a la Church.
- 4.4. Sistema T de Gödel.

Unidad 5

- 5.1. Polimorfismo. Motivación. Variedades de polimorfismo.
- 5.2. Lambda cálculo polimórfico (Sistema F).
- 5.3. Representación de tipos de datos en Sistema F.
- 5.4. Propiedades.

Unidad 6

- 6.1. Abstracciones: funtores y mónadas.
- 6.2. Leyes de funtores. Implementación de funtores en Haskell.
- 6.3. Introducción a mónadas como generalización de la sustitución de variables.
- 6.4. Clase Monad en Haskell. Leyes de las mónadas.
- 6.5. Entrada y salida en lenguajes funcionales puros.
- 6.6. Mónada IO de Haskell. Operaciones. Notación do.
- 6.7. Evaluador monádico simple. Mónadas State, Error, Reader y Writer.

Unidad 7

- 7.1. Lenguajes de dominio específico (DSL). Ventajas y desventajas.
- 7.2. DSLs embebidos (EDSL).
- 7.3. Sintaxis y semántica de EDSLs en Haskell.
- 7.4. Diseño de EDSLs. Shallow y deep embedding.

Modalidades de enseñanza

Los contenidos teóricos del curso se presentan en una clase semanal en el aula, mientras que las y los estudiantes resuelven las prácticas en papel con el apoyo de docentes en una clase semanal en aula y revuelven ejercicios de programación contenidos en las prácticas y los trabajos prácticos en una clase semanal en laboratorio.

Se realizan 4 trabajos prácticos en grupo. En el primer trabajo práctico las y los estudiantes implementan las distintas partes de un lenguaje imperativo simple, mientras que en el segundo y tercer trabajo práctico implementan un lambda cálculo y un lambda cálculo simplemente tipado, respectivamente. Los resultados de aprendizaje RA1, RA2 y RA3 están relacionados con la elaboración de los mismos, donde deben comprender cómo se componen los lenguajes de programación, aprender a implementar cada una de sus partes y comprender a la programación como una actividad rigurosa de abstracción. En el cuarto trabajo práctico implementan un lenguaje imperativo usando mónadas. Los resultados de aprendizaje RA1 a RA4 están vinculados con este trabajo práctico.

El resultado de aprendizaje RA5 está relacionado con las prácticas 2 y 4, donde aplican técnicas formales para probar propiedades sobre los lenguajes.

El resultado de aprendizaje RA6 está vinculado a la realización de los trabajos prácticos.

Recursos

Las clases teóricas y de práctica se realizan en aula y las clases de laboratorio en un laboratorio informático. Se utiliza un proyector en las clases teóricas.

El material de estudio, que incluye presentaciones, prácticas y trabajos prácticos se publica en el campus virtual (Comunidades) de la materia. Se utiliza el campus virtual para la entrega de los trabajos prácticos.

Para la comunicación se utiliza un canal de Zulip del DCC.

Actividades de Formación Práctica

Nº	Título	Descripción
1	Práctica 0	Ejercicios sobre sintaxis concreta y abstracta
2	Práctica 1	Ejercicios de parsers.
3	Práctica 2	Ejercicios sobre semántica operacional y demostraciones de propiedades de los lenguajes.
4	Trabajo Práctico 1	Los alumnos implementan un intérprete en Haskell para un lenguaje imperativo sencillo.
5	Práctica 3	Ejercicios de lambda cálculo.
6	Práctica 4	Ejercicios de sistemas de tipos y pruebas sobre la seguridad de los lenguajes.
7	Trabajo Práctico 2	Los alumnos implementan un evaluador de términos de lambda cálculo.
8	Trabajo Práctico 3	Los alumnos extienden la implementación de un intérprete de lambda cálculo simple tipado, incorporando extensiones como números naturales y listas.
9	Práctica 5	Ejercicios de funtores y mónadas
10	Práctica 6	Ejercicios adicionales de mónadas.

11	Trabajo Práctico 4	Los alumnos implementan un intérprete en Haskell para un lenguaje imperativo sencillo mediante el uso de mónadas.
----	--------------------	---

Evaluación

La evaluación durante el cursado se realiza mediante dos exámenes parciales y 4 trabajos prácticos grupales. El primer parcial corresponde a las unidades 1 a 3 y el segundo a las unidades 4 a 6.

Durante la última semana de clases se contempla un examen recuperatorio con carácter sustitutivo de uno de los parciales (aquel donde el alumno hubiera obtenido menor nota).

Para aprobar los trabajos prácticos se realiza una defensa oral, donde se evalúa el contenido y la participación de cada estudiante en el trabajo práctico.

Si el promedio de los parciales es mayor o igual a 6, con nota no inferior a 4 en cada examen y los trabajos prácticos están aprobados la o el alumno alcanza la condición de regular en la materia.

Para aprobar la materia en condición regular el estudiante puede elegir entre hacer un examen teórico-práctico o un trabajo práctico que requiere la implementación de un EDSL (lenguaje de dominio específico) en Haskell. El tema del trabajo práctico es elegido por el estudiante y aprobado por el docente. Un lenguaje de dominio permite a sus usuarios trabajar en un nivel de abstracción ideal dado que los conceptos con los que se trabaja son propios de su dominio de aplicación. Que el dsl sea embebido le permite al estudiante utilizar la infraestructura del lenguaje anfitrión, en este caso de Haskell, disminuyendo así la ardua tarea de construir un lenguaje desde cero.

Las y los estudiantes libres deben realizar un examen final y un trabajo práctico final para aprobar la materia.

Resultado de Aprendizaje	Actividades/Modalidad de Enseñanza	Modalidad de Evaluación
RA1	Clases teóricas. Prácticas 0,1,2,4,5 y 6. Los 4 trabajos prácticos.	Exámenes parciales (con ejercicios donde a través de un lenguaje sencillo puedan definir cada una de sus partes), trabajos prácticos (donde se evalúan los conceptos, pero en lenguajes más grandes), trabajo práctico o examen final.
RA2	Clases teóricas. Prácticas 0,1,2,4,5 y 6. Los 4 trabajos prácticos.	Exámenes (parciales y final) y trabajos prácticos.
RA3	Clases teóricas y prácticas.	Exámenes (parciales y final) y trabajos prácticos.
RA4	Clases teóricas y prácticas 5 y 6.	Exámenes (con ejercicios sobre mónadas y funtores), trabajo práctico 4 y trabajo práctico final (dada su finalidad, generalmente se requiere el uso de varias mónadas para su implementación).
RA5	Clases teóricas y prácticas 2 y 4.	Exámenes parciales y final. Trabajos prácticos 1 y 3 (que incluyen las pruebas de distintas propiedades de los lenguajes implementados).

RA6	Trabajos prácticos y defensa oral de los mismos (en las defensas de los trabajos prácticos las y los estudiantes no solo muestran la resolución de los mismos sino la manera en que trabajaron).	Defensa de los trabajos prácticos y trabajo práctico final.
-----	--	---

Bibliografía básica

Autores (Apellido, Inicial nombre)	Año de edición	Título de la obra	Editorial o Revista	Ejemplares disponibles o sitio web
Graham Hutton	2007	Programming in Haskell	Cambridge University Press	1
J. R. Hindley and J. P. Seldin	2008	Lambda-Calculus and Combinators	Cambridge University Press	https://www.cambridge.org/ar/universitypress/subjects/computer-science/programming-languages-and-applied-logic/lambda-calculus-and-combinators-introduction-2nd-edition?format=AR
Benjamin Pierce.	2002	Types and Programming Languages.	The Mit Press	3
Philip Wadler	1995	Monads for Functional Programming	Lecture Notes in Computer Science	https://homepages.inf.ed.ac.uk/wadler/papers/marktoberdorf/baastad.pdf
Simon Thompson	1999	The Craft of Functional Programming	Addison Wesley	3
JS. L. Peyton Jones, John Hughes, et al.	1999	Report on the Programming language Haskell '98	Yale University	http://www.haskell.org/onlinereport

Bibliografía complementaria

Autores (Apellido, Inicial nombre)	Año de edición	Título de la obra	Editorial o Revista	Ejemplares disponibles o sitio web
Jean-Yves Girard, Yves Lafont and Paul Taylor	1987	Proofs and Types	Cambridge University Press	https://www.paultaylor.eu/stable/prot.pdf
J. Reynolds	1998	Theories of Programming Languages	Cambridge University Press	1
J.C. Mitchell	1997	Foundations of Programming Languages	Cambridge University Press	1
R. S. Bird.	1998	Introduction to Functional Programming using Haskell.	Prentice-Hall	4

Distribución de la carga horaria

Presenciales

Teóricas		42 Hs.
Prácticas	Formación Experimental	
	Resolución de Problemas vinculados a la Profesión	20 Hs.
	Resolución de Problemas y Ejercicios	50 Hs.
	Actividades de Proyecto y Diseño	
	Formación en la Práctica Profesional	
Evaluaciones		8 Hs.
	Total	120 Hs.

Dedicadas por el alumno fuera de clase

	Preparación Teórica	16 Hs.
	Preparación Práctica	35 Hs.
	Elaboración y redacción de informes, trabajos, presentaciones, etc.	25 Hs.
	Total	76 Hs.

Cronograma de actividades

Semana	Unidad	Tema	Actividad
1	1	Unidades 1.1 a 1.4	Práctica 0
2	1	Unidades 1.5 y 1.6.	Práctica 1
3	2	Unidades 2.1. a 2.7.	Práctica 2 Trabajo Práctico 1
4	2	Unidades 2.8. y 2.9.	Práctica 2 Trabajo Práctico 1
5	3	Unidades 3.1. a 3.3.	Práctica 3
6	3	Unidad 3.4.	Práctica 3 Trabajo Práctico 2
7	3	Parcial 1	Práctica 3 Trabajo Práctico 2
8	4	Unidades 4.1. a 4.3.	Práctica 4
9	4	Unidades 4.4. y 4.5.	Práctica 4 Trabajo Práctico 3
10	5	Unidad 5	Práctica 4 Trabajo Práctico 3
11	6	Unidades 6.1. a 6.3.	Práctica 5
12	6	Unidades 6.5. a 6.7.	Práctica 5
13	6	Parcial 2	Práctica 6 Trabajo Práctico 4
14	6	Recuperatorio	Práctica 6 Trabajo Práctico 4
15	7	Unidad 7	Práctica 6