Normas de la asignatura

Modelos Formales de Computación

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Santiago Escobar

Motivación (1/4)

- Un modelo de computación proporciona una sintaxis y una semántica a un paradigma de programación.
  - Paradigma funcional – reescritura y lamda cálculo
  - Paradigma lógico – predicados y resolución
  - Paradigma OO – clases, objetos, métodos y atributos
  - Paradigma metaprogramación - reflexión
  - Paradigma concurrencia – redes de petri, actores, etc.
  - Paradigma restricciones – variables e inecuaciones
  - Paradigma eventos – disparo, condiciones, atomicidad

Motivación (2/4)

- Según Real Academia.
  - Sintaxis = conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje
  - Semántica = el estudio del significado de los signos lingüísticos y de sus combinaciones
- Informalmente
  - Sintaxis = cómo construir expresiones correctas
  - Semántica = dotar de significado a las expresiones bien construidas
- Ejemplos ambigüedad en la semántica
  - Ana cogió su bicicleta (la suya o la de otra persona)
  - Vio un hombre en un barco con un catalejo (el hombre tenía un catalejo o fue visto a través de un catalejo)
  - El pollo está listo para comer (el pollo come o es comido)

Motivación (3/4)

- Expresividad vs programación vs computación
- Ejemplo potencia (C vs Haskell)

$$x^0 = 1$$
$$x^{n+1} = x \cdot x^n$$

```
int power(int x, int n);
{
    int i = 1;
    int result = 1;
    while (i <= n)
    {
        result = result * x;
        i++;
    }
    return(result);
}
```

```
power x 0 = 1
power x n = x * power x (n - 1)
```

Motivación (4/4)

- Ejemplo búsqueda soluciones (Haskell vs Prolog)

```
permute [] = [[]]
permute xs = [x:ps | x <- xs, ps <- permute (delete x xs)]
```

```
permute([],[]).
permute([H|T],L) :- permute(T,U), insert(H,U,L).
insert(X,L,[X|L]).
insert(X,[H|U],[H|L]) :- insert(X,U,L).
```

¿Por qué estudiar semánticas?

- Entender diferentes conceptos y funcionalidades
- Facilitan el "adaptarse" a otros paradigmas
- Lenguajes modernos son multiparadigma
  - Mathematica soporta 12 paradigmas (funcional, declarativo, simbólico, lenguaje natural, reflectivo, OO, etc.)
  - Microsoft F#: imperativa, funcional, OO, concurrente, reflexión
- Permiten elegir el lenguaje correcto para una aplicación
- Diseñar nuevos lenguajes de programación

  (Racket, Google Go, Apple Swift, Kotlin, R, Rust)
- Kotlin (objetos, eventos, lambdas, reflexión, scope, etc.)

¿Por qué estudiar semánticas?

The LLVM Compiler Infrastructure

Site Map:

- Overview
- Features
- Documentation
- Command Guide
- FAQ
- Publications
- LLVM Projects
- Open Projects
- LLVM Users
- Bug tracker
- LLVM Logo

LLVM Overview

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies. Despite its name, LLVM has little to do with traditional virtual machines. The name "LLVM" itself is not an acronym; it is the full name of the project.

LLVM began as a research project at the University of Illinois, w
modern, SSA-based compilation strategy capable of supporting b
compilation of arbitrary programming languages. Since then, LL
umbrella project consisting of a number of subprojects, many of
production by a wide variety of commercial and open source proj
used in academic research. Code in the LLVM project is licensed

Latest LLVM Release!

18 June 2024: LLVM 18.1.8 is now **available for download**! LLVM is publicly available under an open source License. Also, you might want to

runtime verification

kontrol simbolik about services products research blog

CVE® CVE List▾ CNAs▾

Search CVE List Downloads Data Feed

TOTAL CVE Re

NOTICE: Transition to the all-new CVE website at WWW

NOTICE: Support for the legacy CVE do
New CVE List download forma

Take Your Security to the Next Level

Instead of paying researchers to look for bugs, verify your smart contracts are bug-free with **formal verification.**

We explore all possible behaviors of the code, to give you the highest possible assurance.

Ship FAST with continous integration of proofs, and outsource the compute to us using K as a Service (Kaas).

Temario

- Tema 1: Introducción (2h)
- Tema 2: Semánticas (2h)
- Tema 3: Programación imperativa (4h)
- Tema 4: Programación funcional (2h)
- Tema 5: Programación lógica (2h)
- Tema 6: Programación orientada a objetos (4h)
- Tema 7: Concurrencia (4h)

- Laboratorio 1: Haskell (2h)
- Laboratorio 2: Prolog (2h)
- Laboratorio 3: Semántica Funcional (2h)
- Laboratorio 4: Semántica Lógica (2h)
- Laboratorio 5: Semántica Concurrencia (2h)

Evaluación

Descripción	Nº Actos	Peso (%)
(05) Trabajos académicos	3	30
(15) Prueba práctica de laboratorio/campo/informática/aula	5	30
(14) Prueba escrita	2	40

Los contenidos teóricos de la asignatura se evaluarán con un examen con un peso global del 20% de la nota final. La prueba será recuperable.
Por otro lado, los contenidos prácticos de la asignatura se evaluarán con la entrega de prácticas, con un peso global del 30%.
Además se propondrán tres trabajos académicos (basados en algún ejercicio práctico desarrollado a lo largo de la asignatura). Estos trabajos se realizarán fuera del aula y se entregarán al profesor, siendo su nota de un 10% cada uno de los tres trabajos.
Por último, una parte de la evaluación residirá en el seguimiento del trabajo del alumno en el aula (principalmente preguntas del último minuto planteadas durante las clases), con una nota máxima del 20%.
Para los alumnos con dispensa, la evaluación se realizará con una prueba escrita. La prueba será recuperable.

Competencias transversales

(5) Responsabilidad y toma de decisiones

- Actividades desarrolladas relacionadas con la adquisición de la competencia

· Durante la asignatura, se desarrolla un trabajo académico donde deben evaluar y decidir cómo desarrollar una nueva característica de un lenguaje de programación

- Criterios de evaluación

· El trabajo académico se desarrolla durante el curso y se entrega directamente al profesor, quien lo evalúa.

Bibliografía

- *Programming languages and operational semantics: an introduction*. Maribel Fernández.
- *Models of computation: an introduction to computability theory*. Maribel Fernández.
- *Programación funcional con Haskell*. Francisco Gutiérrez López ; Blas Carlos Ruiz Jiménez ; José E Gallardo Ruiz ; Pablo Guerrero García.
- *Introducción a la programación funcional con Haskell*. Richard Bird
- *Programación lógica : teoría y práctica*. Pascual Julián Iranzo.
- *Concepts, Techniques, and Models of Computer Programming*. Peter Van Roy and Seif Haridi.