METAPROGRAMMING

Nada Amin (na482)
Jeremy Yallop (jdy22)
AIMS

➤ This course surveys principled approaches to metaprogramming:
  ➤ writing programs that manipulate programs.
  ➤ Topics include
    ➤ evaluators
    ➤ reflection
    ➤ writing programs that write programs
    ➤ designing domain specific languages and meta-linguistic abstractions
    ➤ synthesis.
PROGRAMMING LANGUAGES

Why?
META-LINGUISTIC ABSTRACTIONS

SICP = Structure and Interpretation of Computer Programs
LECTURES

➤ Week 1. Programs as data, data as programs. Interpreters. Meta-interpreters in Lisp and Prolog.

➤ Week 2. Reification / reflection, reflective towers of interpreters, meta-object protocols.

➤ Week 3. Partial evaluation including Futamura projections, multi-stage programming. Turning interpreters into compilers, collapsing towers of interpreters.


➤ Week 5. Synthesis.

OBJECTIVES

➤ By the end of the course students should be able to:

➤ Mechanically turn an interpreter into a compiler;

➤ design meta-linguistic abstractions to deal with a complex problem;

➤ think beyond traditional paradigms for programming.
SCALA
ASSESSMENT

➤ 3 programming assignments:
  ➤ Reflective Lisp Interpreter
  ➤ Image processing DSL with LMS
    ➤ DSL = Domain-Specific Language
    ➤ LMS = Lightweight Modular Staging
  ➤ Image processing from scratch from Scala to JavaScript
  ➤ 1 open-ended assignment
    ➤ Counts for 10% for Part II and 50% for MPhil
ASSIGNMENT 1
released Thu 18 Oct
submitted Tue 30 Oct
returned Thu 8 Nov

ASSIGNMENT 2
released Thu 1 Nov
submitted Tue 13 Nov
returned Thu 22 Nov

ASSIGNMENT 3
released Thu 15 Nov
submitted Tue 27 Nov

OPEN-ENDED ASSIGNMENT: submitted beginning Lent Term
OUR FAVORITE META-PROGRAMS

➤ NFA to DFA by staging exploration
➤ Probabilistic Monad / Pattern Matching
➤ Interpreter to Compiler (e.g. SQL to C)
➤ Embedded Compiler Transformations
➤ Staged Parser Combinators
➤ Towers of Interpreters for Aspect-Oriented Programming
➤ Reflection by Mirrors
➤ Abstract Interpreters
➤ Automatic Differentiation
SCALA

➤ Implicits / Type classes
  ➤ conversion
  ➤ parameter
➤ Traits
  ➤ mixin composition
  ➤ cake pattern
➤ Sealed case classes
  ➤ Algebraic Data Types (ADTs)
➤ Sugar — apply/unapply
➤ Companion objects
INFLUENTIAL BOOKS
A comprehensive step-by-step guide

Programming in Scala
Third Edition

Updated for Scala 2.12

Martin Odersky
Lex Spoon
Bill Venners

artima

SCALA

➤ https://scala-lang.org/documentation/learn.html

“The evaluator, which determines the meaning of expressions in a programming language, is just another program.”

“Metalinguistic abstraction -- establishing new languages -- plays an important role in all branches of engineering design. It is particularly important to computer programming, because in programming not only can we formulate new languages but we can also implement these languages by constructing evaluators.”
VEHICLES

➤ “Diaspora of Love”

➤ SICP assignment
  Tit for Tat

➤ Alan Kay suggests game
  “Robot Odyssey”
➤ truth-maintenance systems
➤ beyond theorem proving...
PAIP

➤ interpreter
➤ compiler
➤ optimiser
➤ LISP
➤ Prolog
➤ data-driven programming
THE ART OF PROLOG

➤ meta-programming in logic programming
TDD BY EXAMPLE

- bootstrapping a testing framework
EXPERT F#

➤ language-oriented programming
➤ probability monad
XINU OS

implementing concurrency
The Reasoned Schemer
Second Edition

Daniel P. Friedman, William E. Byrd, Oleg Kiselyov, and Jason Hemann

Foreword by Guy Lewis Steele Jr. and Gerald Jay Sussman
Afterword by Robert A. Kowalski
Drawings by Duane Bibby

RELATIONAL PROGRAMMING