ECE251: Programming Languages & Translators Fall 2010

Lecture 24 — November 16, 2010

Patrick Lam *version 1*

We've now seen all of the compiler bits—scanners, parsers, symbol tables, type checkers, code generation—that I intend to discuss in this class. The project touches all these areas.

Programming Paradigms

Because this class is called "Programming Languages and Translators", I'm going to talk about programming languages for the next 1.5 weeks, before finishing with more discussion of domain-specific languages.

The three kinds of programming languages that I'm going to talk about are *functional*, *logic* and *scripting* languages. They each excel at some tasks that the object-oriented and imperative paradigms (i.e. Java, C) you've seen so far aren't good at.

Here is a brief summary of each of these paradigms.

- *Functional* languages make it easy to pass around bits of code, or functions, as first-class values. Programs in these languages often rely heavily on recursion, and look a lot more like "math" than imperative code.
- *Logic* programming languages allow the programmer to specify a collection of facts and then to query the knowledge base for consequences of these facts. These languages were traditionally used in AI, but one such language, Datalog, has been finding widespread applications recently in a number of different domains.
- *Scripting* languages primarily connect other programs together. They are almost always interpreted and don't do any computational heavy lifting, but usually have good support for string munging.

Functional Languages

We are going to study functional languages in the context of Scala. You can also find a general description of functional languages in Chapter 10 of the text.

Scala is a relatively new (since 2001) functional language that compiles to Java bytecode. I've installed it on `ece251.patricklam.ca` and you can use it to solve the final exam practice problems that I'll post. Another functional language, which you are highly likely to encounter, is JavaScript; some JavaScript programming problems are best solved by understanding functional programming.

A good reference for this part of the class is "Scala by Example". `www.scala-lang.org/docu/files/ScalaByExample.pdf`