ECE251: Programming Languages & Translators Fall 2010

Lecture 29 — November 29, 2010

Patrick Lam *version 1*

Scripting Languages

The references for today's lecture are Chapter 13 of the textbook and "Scripting: Higher Level Programming for the 21st Century", by Ousterhout[1].

The plan for today is to introduce the concept of scripting languages, present common characteristics of these languages, and discuss a few examples of application domains where scripting languages are useful.

What are some examples of scripting languages?

Use cases for scripting languages. Scripting languages are good for coordinating pre-existing components (i.e. acting as "glue languages"). Note that components might not be programs, but rather different parts of a larger tool, e.g. a browser, or a VLSI tool, or a GUI toolkit.

Scripting languages tend to be slower at serious computation, but their ease-of-use typically makes up for that. You would usually use them in conjunction with more conventional languages, as we see below.

Here is an example of a task which a scripting language might carry out:

1. Copy a fresh version of the skeleton to a working directory. (It is slightly different from the version which I posted initially).
2. Copy the parser to be graded into that directory.
3. Invoke ANTLR to compile the parser.
4. Recompile the parser.
5. Run the testHarness script on my test suite with the recompiled parser.

Scripting languages coordinate components. What are the components here?

[1]`http://home.pacbell.net/ouster/scripting.html`.

Common characteristics of scripting languages. Let's think about scripting languages and identify some of their commonalities.

- Support interactive and batch use: e.g. Python, Ruby and others let you type in commands from the keyboard (or from a script). Scala and Prolog do this too.
- Short on boilerplate: e.g. you can just write `print "Hello, world!\n"` in Perl, Python or Ruby. Also, some of these languages don't require variable declarations.
- Flexible dynamic typing: e.g. usually not much static type checking; some dynamic type checks and some type conversions.
- Easy access to system facilities: file I/O and other system calls are easy.
- Good string manipulation and pattern-matching: think Perl, where munging strings is built-in.
- High-level data types and easy access to them: Perl also has associative arrays as part of the base language.

Application Domains for Scripting Languages

Here's a brief survey of domains where scripting languages get used.

- as shell (command) languages (bash, etc.) and glue languages;
- for text processing and report generation (sed, awk, perl);
- for mathematical and statistical languages (domain-specific languages) (e.g. APL, Maple, Mathematica, Matlab, S, R); and
- as extension languages (JavaScript, Emacs Lisp, Tcl, Visual Basic).