CSci 658-01: Software Language Engineering
Metaprogramming

H. Conrad Cunningham

22 February 2018

Contents

Professor of Computer and Information Science
University of Mississippi
211 Weir Hall
P.O. Box 1848
University, MS 38677
(662) 915-5358

Advisory: The HTML version of this document requires use of a browser that supports the display of MathML. A good choice as of February 2018 is a recent version of Firefox from Mozilla.

Definition

Basically, *metaprogramming* is writing code that writes code.

Metaprogramming: the writing of computer programs that can treat computer programs as their data. A program can read, generate, analyze, and/or transform other programs, and even modify itself while running [Adapted from Wikipedia and other sources, 2019]

We often do metaprogramming in our tasks but just not call it that.

- Our web applications may generate HTML, JavaScript, and CSS code to enable the display of data in a web browser.
- Our Java programs may use `instanceof` to check the type of objects or otherwise manipulate itself with the Java reflection package.

- Our programs may use macros to define new features in terms of existing features.

Under the above definition, much of our study of domain-specific languages is metaprogramming.

- The `pic` little language processor takes a program expressed in an external textual language that describes a picture and generates output expressed in another language that gives instructions to a display program.
- Several of the State Machine DSL processors read and parse a program written in in a special-purpose textual language, represent the program internally in a semantic model, and then "execute" the model on inputs.
- Other of the State Machine processors use the semantic model to generate a program in another language such as C or the Graphviz dot language for graphs.
- The Computer Configuration and Email Message internal DSLs use the host language itself to encode special-purpose languages. The processors can then read and parse descriptions written in these special-purpose languages and manipulate the resulting data structures similarly to the external DSLs.
- The Survey and Lair Configuration DSLs manipulate the structure of the processing program itself to implement the special-purpose language.

The latter are examples of *reflexive metaprogramming*.

Reflexive metaprogramming: the writing of computer programs that manipulate themselves as data.

This manipulation may be at "compile time" involving a phase of transformations in the code before the final program is generated. Or it may be at runtime, involving manipulation of the program's metamodel or generation of new code that is dynamically executed within the program.

The Survey DSL is a Ruby internal DSL. It takes advantage of Ruby's metaprogramming facilities such as the abilities to trap calls to undefined methods, to dynamically add methods or variables to existing objects at runtime, and to execute dynamically generated strings as Ruby code. It also uses Ruby's closures (first-class functions) and flexible syntax – although these are not technically metaprogramming features.

The Lair Configuration programs use the metaprogramming features of Lua in similar ways.

Consider relatively common languages and their metaprogramming features.

1. Java is a statically typed, compiled language. What are metaprogramming features available in Java?

dynamic class loaders, reflection API, annotation processing, dynamic method invocation (JVM feature), JVM bytecode manipulation (mostly with external tools), etc.

2. Lua is a dynamically typed, interpreted language. What are the metaprogramming features available in Lua?

   metatables, metamethods, manipulation of environments, debug library (introspection/reflection features), `loadfile` and `loadstring` functions to dynamically execute code, extensions in C, etc.