Object-Oriented Programming in Scheme
with First-Class Modules and Operator-Based Inheritance

Guruduth Banavar*
Gary Lindstrom
Department of Computer Science
University of Utah, Salt Lake City, UT 84112

Abstract

We characterize object-oriented programming as structuring and manipulating a uniform space of first-class values representing modules, a distillation of the notion of classes. Operators over modules individually achieve effects such as encapsulation, sharing, and static binding. A variety of idioms of O-O programming find convenient expression within this model, including several forms of single and multiple inheritance, abstract classes, class variables, inheritance hierarchy combination, and reflection. We show that this programming style simplifies O-O programming via enhanced uniformity, and supports a flexible model of object-orientation that provides an attractive alternative to meta-programming. Finally, we show that these notions of O-O programming are language independent, by implementing a Modular Scheme prototype as a completion of a generic O-O framework for modularity.

Paper Category: Research. Topic Area: Language design and implementation.

1 Introduction

Class-based object-oriented programming is usually thought of as creating a graph structured inheritance hierarchy of classes, instantiating some of these classes, and computing with these instances. Instances are typically “first-class” values in the language, i.e. they can be created, stored, accessed, and passed into and out of functions. Classes, on the other hand, are usually not first-class values, and inheritance is often considered an operational and static structuring activity.

Some dynamic languages like CLOS [18] and Smalltalk [15] permit access to classes at run-time, usually as objects of other (meta-)classes. However, even in dynamic O-O languages, there is often a disparity between the manner in which classes and other values are manipulated. Classes are often not on an equal footing with other values; for example, classes are not passed into and out of functions or stored and retrieved as attributes of other classes. When a more equitable status for classes is desired, meta-programming is resorted to. A meta-level architecture assumes the role of capturing and exposing the properties of classes, objects, and their interactions via a collection

*Primary contact author. E-mail: banavar@cs.utah.edu, Phone: +1-801-581-8378, fax: +1-801-581-5843.