Modularity Meets Inheritance

Gilad Bracha
Gary Lindstrom

UUCS-91-017

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

October 13, 1991

Abstract

We "unbundle" several roles of classes in existing languages, by providing a suite of operators independently controlling such effects as combination, modification, encapsulation, name resolution, and sharing, all on the single notion of *module*.

All module operators are forms of inheritance. Thus, inheritance not only is not in conflict with modularity in our system, but is its foundation.

This allows a previously unobtainable spectrum of features to be combined in a cohesive manner, including multiple inheritance, mixins, encapsulation and strong typing.

We demonstrate our approach in a language (called *Jigsaw*, as in the tool, not the puzzle!). Our language is modular in two senses: it manipulates modules, and it is highly modular in its own conception, permitting various module combinators to be included, omitted, or newly constructed in various realizations. We discuss two pragmatic avenues for the exploitation of this approach:

1. Adding modules to languages without modularity constructs.

2. Embedding selected new modularity capabilities within existing object-oriented languages (which we are undertaking as a "proof of concept" in the case of *Modula-3* [5]).[1]

[1] This research was sponsored by (i) the Defense Advanced Research Projects Agency (DOD), monitored by the Department of the Navy, Office of the Chief of Naval Research, under Grant number N00014-91-J-4046, and (ii) the National Science Foundation under Grant No. CCR89-20971. The opinions and conclusions contained in this document are those of the authors and should not be interpreted as representing official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the National Science Foundation, or the US Government.