Type-safe Composition of Object Modules*

Guruduth Banavar, Gary Lindstrom, Douglas Orr
Department of Computer Science, University of Utah
Salt Lake City, Utah 84112 USA

Abstract

We describe a facility that enables routine type-checking during the linkage of external declarations and definitions of separately compiled programs in ANSI C. The primary advantage of our server-style type-checked linkage facility is the ability to program the composition of object modules via a suite of strongly typed module combination operators. Such programmability enables one to easily incorporate programmer-defined data format conversion stubs at link-time. In addition, our linkage facility is able to automatically generate safe coercion stubs for compatible encapsulated data.

1 Introduction

It is widely agreed that strong typing increases the reliability and efficiency of software. However, compilers for statically typed languages such as C and C++ in traditional non-integrated programming environments guarantee complete type-safety only within a compilation unit, but not across such units*. Longstanding and widely available linkers compose separately compiled units by matching symbols purely by name equivalence with no regard to their types. Such “common denominator” linkers accommodate object modules from various source languages by simply ignoring the static semantics of the language. Moreover, commonly used object file formats are not designed to incorporate source language type information in an easily accessible manner.

In this paper, we present a technique to perform type checking of object modules as a routine link-time activity. Our technique is characterized by (i) the design of specific language type systems into a system-wide linker, (ii) programmed link-time control over individual symbols of object modules, and (iii)

*C++ style name-mangling does not accomplish complete type-safety across compilation units; see Section 6.