Preface

The importance of typed languages for building robust software systems is, by now, an undisputed fact. Years of research have led to languages with richly expressive, yet easy to use, type systems for high-level programming languages. Types provide not only a conceptual framework for language designers, but also afford positive benefits to the programmer, principally the ability to express and enforce levels of abstraction within a program.

Early compilers for typed languages followed closely the methods used for their untyped counterparts. The role of types was limited to the earliest stages of compilation, and they were thereafter ignored during the remainder of the translation process. More recently, however, implementors have come to recognize the importance of types during compilation and even for object code. Several advantages of types in compilation have been noted to date:

- They support self-checking by the compiler. By tracking types during compilation it is possible for an internal type checker to detect translation errors at an early stage, greatly facilitating compiler development.
- They support certification of object code. By extending types to the generated object code, it becomes possible for a code user to ensure the basic integrity of that code by checking its type consistency before execution.
- They support optimized data representations and calling conventions, even in the presence of modularity. By passing types at compile-, link-, and even run-time, it is possible to avoid compromises of data representation imposed by untyped compilation techniques.
- They support checked integration of program components. By attaching type information to modules, a linker can ensure the integrity of a composite system by checking compliance with interface requirements.

Types in Compilation (TIC) is a recurring workshop devoted to the application of types in the implementation of programming languages. This volume consists of a selection of papers from the TIC 2000 Workshop held in Montreal, Canada in September 2000. The papers published herein were chosen from submissions solicited after the meeting by a rigorous refereeing process comparable in depth and scope to the selection criteria for an archival journal. Each paper was reviewed by at least three referees chosen from the TIC 2000 program committee (named below), with final publication decisions made by the program chair. This volume represents the result of that review and revision process.

March 2001 Robert Harper

Organization

TIC 2000 was held in cooperation with ACM SIGPLAN in Montreal, Canada on September 21, 2000 as part of the Colloquium on Principles, Logics, and Implementations of High-Level Programming Languages (PLI 2000). The TIC organizers are grateful to ACM SIGPLAN and to PLI for their assistance and support.

Organizing Committee

Craig Chambers, University of Washington
Karl Crary (chair), Carnegie Mellon University
Robert Harper, Carnegie Mellon University
Xavier Leroy, INRIA Rocquencourt
Robert Muller, Boston College
Atsushi Ohori, Kyoto University
Simon Peyton Jones, Microsoft Corporation

Program Committee

Dominic Duggan, Stevens Institute of Technology
Robert Harper (chair), Carnegie Mellon University
Trevor Jim, AT&T
Andrew Kennedy, Microsoft Corporation
Atsushi Ohori, Kyoto University
Franklyn Turbak, Wellesley College