The Interprocedural Analysis and Automatic Parallelization of Scheme Programs

WILLIAMS LUDWELL HARRISON III* (harrison@uicsrd.csrd.uiuc.edu)

Center for Supercomputing Research and Development
University of Illinois at Urbana-Champaign
305 Talbot Laboratory
104 South Wright Street
Urbana, IL 61801

(Received: March 1, 1989)
(Revised: May 10, 1989)

Keywords: Interprocedural Analysis Abstract Interpretation, Automatic Parallelization, Program Transformation, Parallel Processing, Symbolic Computation, Lisp, Scheme

Abstract. Lisp and its descendants are among the most important and widely used of programming languages. At the same time, parallelism in the architecture of computer systems is becoming commonplace. There is a pressing need to extend the technology of automatic parallelization that has become available to Fortran programmers of parallel machines, to the realm of Lisp programs and symbolic computing. In this paper we present a comprehensive approach to the compilation of Scheme programs for shared-memory multiprocessors. Our strategy has two principal components: interprocedural analysis and program restructuring. We introduce procedure strings and stack configurations as a framework in which to reason about interprocedural side-effects and object lifetimes, and develop a system of interprocedural analysis, using abstract interpretation, that is used in the dependence analysis and memory management of Scheme programs. We introduce the transformations of exit-loop translation and recursion splitting to treat the control structures of iteration and recursion that arise commonly in Scheme programs. We propose an alternative representation for s-expressions that facilitates the parallel creation and access of lists. We have implemented these ideas in a parallelizing Scheme compiler and run-time system, and we complement the theory of our work with "snapshots" of programs during the restructuring process, and some preliminary performance results of the execution of object codes produced by the compiler.

*This work was supported in part by the National Science Foundation under Grant No. NSF MIP-8410110, the U.S. Department of Energy under Grant No. DE-FG02-85ER25001, the Office of Naval Research under Grant No. ONR.N00014-88-K-0686, the U.S. Air Force Office of Scientific Research under Grant No. APOSР-F49620-86-C-0136, and by a donation from the IBM Corporation.
Contents

1 Introduction 185
   1.1 Motivation and Approach .......................... 185
   1.2 The Input Language: Scheme ......................... 187

2 The Interprocedural Analysis of Scheme Programs 189
   2.1 Motivations ...................................... 189
      2.1.1 Side-Effects and Dependence Analysis ........ 190
      2.1.2 Object Lifetimes and Memory Management ...... 190
      2.1.3 Folding Procedural Constants and Merging Contours 192
   2.2 Overview of our Approach .......................... 192
   2.3 Notational Conventions ............................. 194
   2.4 Abstract Interpretation ............................ 195
   2.5 Concrete Semantics ............................... 199
      2.5.1 The Language $\mathcal{L}$ .................... 199
      2.5.2 Procedure Strings ........................... 201
      2.5.3 A Semantics for $\mathcal{L}$ in Terms of Procedure Strings . 203
      2.5.4 Abstraction in the Face of Reflexivity .......... 207
      2.5.5 Modified Domain Definitions for $\mathcal{L}$ ...... 208
      2.5.6 A Modified Semantics for $\mathcal{L}$ ........... 210
   2.6 Optimal Solutions in Terms of Procedure Strings .... 217
      2.6.1 Side-Effects, in Terms of Procedure Strings ... 218
      2.6.2 Stack Allocation, in Terms of Procedure Strings ... 223
      2.6.3 Generalized Hierarchical Allocation and Deallocation 224
      2.6.4 Examples of Side-Effects and Object Lifetimes ... 225
      2.6.5 Some Observations ........................... 227
   2.7 Stack Configurations ............................... 228
   2.8 The Abstraction of Operations Over Procedure Strings ... 231
   2.9 Abstract Semantics ............................... 236
   2.10 Approximate Solutions in Terms of Stack Configurations ... 252
      2.10.1 Side-Effects, in Terms of Stack Configurations ... 253
      2.10.2 Stack Allocation, in Terms of Stack Configurations ... 254
      2.10.3 Generalized Hierarchical Allocation and Deallocation 255
   2.11 A Shift in Perspective (and in Accuracy) ........... 257