An Implementation of Narrowing Strategies*

Sergio Antoy[1] Michael Hanus[2] Bart Massey[1] Frank Steiner[2]

[1] Department of Computer Science, Portland State University,
P.O. Box 751, Portland, OR 97207, U.S.A.
{antoy,bart}@cs.pdx.edu
[2] Institut für Informatik, Christian-Albrechts-Universität Kiel,
Olshausenstr. 40, D-24098 Kiel, Germany
{mh,fst}@informatik.uni-kiel.de

Abstract

This paper describes an implementation of narrowing for overlapping inductively sequential rewrite systems. Narrowing is the essential computational engine of functional logic languages. An implementation of narrowing translates a program consisting of rewrite rules into executable code. This executable code currently falls into two categories: Prolog predicates or instructions for an abstract machine. Although these approaches are relatively simple, in both cases, several layers of interpretation separate the functional logic program from the hardware intended to execute it. Obviously, this situation does not lead to efficient executions.

In this paper we investigate a different approach. We translate a functional logic program into an imperative program. Our target language is Java, but we make limited use of specific object-oriented features, such as inheritance and dynamic polymorphism. Replacing Java with a lower-level target language, such as C or machine code, would be a simple task.

A central idea of our approach is the explicit representation and processing of narrowing computations as data objects. This enables the implementation of operationally complete strategies (i.e., without backtracking) or techniques for search control (e.g., encapsulated search). Thanks to the use of an intermediate and portable representation of programs, our implementation is general enough to be used as a common back-end for a wide variety of functional logic languages.

* This research has been partially supported by the DAAD/NSF under grant INT-9981317 and the German Research Council (DFG) under grant Ha 2457/1-2.