An NSF Proposal

ROBERT PAIGE
Department of Computer Science, Courant Institute, New York University, 251 Mercer Street,
New York, NY 10012, USA

Abstract. The objectives of this research are to improve software productivity, reliability, and performance of complex systems. The approach combines program transformations, sometimes in reflective ways, to turn very high level perspicuous specifications into efficient implementations. These transformations will be implemented in a meta-transformational system, which itself will be transformed from an executable specification into efficient code. Experiments will be conducted to assess the research objectives in scaled up applications targeted to systems that perform complex program analysis and translation.

The transformations to be used include dominated convergence (for implementing fixed points efficiently), finite differencing (for replacing costly repeated calculations by less expensive incremental counterparts), data structure selection (for simulating associative access on a RAM in real time), and partial evaluation (for eliminating interpretive overhead and simplification). Correctness of these transformations, of user-defined transformations, and of the transformational system itself will be addressed in part. Both the partial evaluator and components of the transformational system that perform inference and conditional rewriting will be derived by transformation from high level specifications. Other transformations will be specified in terms of Datalog-like inference and conditional rewriting rules that should be amenable to various forms of rule induction.

Previously, Cai and Paige in [12] used an ideal model of productivity free from all human factors in order to demonstrate experimentally how a transformation from a low level specification language into C could be used to obtain a five-fold increase in the productivity of efficient algorithm implementation in C in comparison to hand-coded C. However, only small-scale examples were considered. The proposed research includes a plan to expand this model of productivity to involve other specification languages (and their transformation to C), and to conduct experiments to demonstrate how to obtain a similar five-fold improvement in productivity for large-scale examples of C programs that might exceed 100,000 lines.

The proposal lays out extensive evidence to support the approach, which will be evaluated together with its theoretical underpinnings through substantial experiments. If successful, the results are expected to have important scientific and economic impact. They are also expected to make interesting, new pedagogical connections between the areas of programming languages, software engineering, databases, artificial intelligence, and algorithms.

Keywords: program transformation, software productivity, software performance, partial evaluation, language translators, data structure selection

1. Introduction

Program Transformations is about semantics-based analysis and manipulation of programs. Over the past twenty years we have made contributions to the area by developing two distinct tracks: (1) general-purpose problem specification and its transformation to efficient programs, and (2) special-purpose specification of systems that implement the program analysis and transformations used in Track (1). The long term goal is to combine both tracks.
Previously, using the approach of Track (1) we were able to demonstrate effective translations of high level specifications of algorithms into high performance codes limited to small-scale examples. In [12] Cai and Paige developed an ideal model of productivity free from human factors. Within this model they gave experimental evidence that their transformational approach to program development leads to at least a five-fold improvement in productivity of efficient algorithm implementation in C as compared to hand-coded C. However, those experiments only considered small-scale examples of procedureless programs no more than ten pages long. In the proposed grant period, we plan to extend our specification languages and the transformations that implement them in order to specify and develop efficient large-scale systems with a similar improvement in productivity.

Previously, using the approach of Track (2) we were able to demonstrate effective development of large-scale systems limited to inefficient prototypes. The complex translation of a statically typed variant of SETL into C used in the productivity experiments mentioned above was specified in RSL (Rule Specification Language), a high level language implemented by the APTS program transformation system [37]. The translation suffered from two major sources of inefficiency. First, APTS is implemented in SETL2 [49], which runs 30 times slower than C at best. Second, APTS only provides an interpretive implementation for RSL. Hence, the translation of SETL to C was bogged down to 24 lines per minute on a SPARC 2. In the proposed grant period we plan experiments to test the feasibility of a radical new transformational methodology that combines reflective forms of partial evaluation (to eliminate interpretive overhead) and data structure selection (to replace the naive SETL2 runtime) used in Track (1) in order to gain a 300-fold speedup in RSL execution.

By combining improvements to both tracks, we plan to demonstrate a five- to ten-fold improvement in productivity for implementing large-scale systems with more than 100,000 lines of C. Our automated program development methodology is most effective in development of systems with high algorithmic content, which are among the most difficult to construct and maintain by hand. Included in this class of systems are those that implement complex program analysis and transformation, which is the application domain for the research proposed here. An example is the SETL-to-C translator, an RSL specification which our methods are expected to speedup by a factor of 300.

Section 2 of this proposal describes a cohesive research project that should take three to five years to complete for three Ph.D. students and the Principal Investigator. Section 3 details a number of experiments, including a novel, reflective combination of partial evaluation and data structure selection, that are expected to be completed within the three year funding period. Although this proposal only seeks funding for one Ph.D. student, two other student participants will be funded by other sources—one by NYU fellowship and another by an ONR grant that partly overlaps with this proposal.

2. Background and objectives

2.1. Track (1) background

2.1.1. Specification languages and transformations. Within Track (1) we consider an implementation language (e.g., C) that serves as a conventional RAM model of computation.