Software Support For Programming-in-the-Large
Final Report

1997

BAA: 90-21
Topic: Advanced Environments
ARPA Order: 8856
ONR Contract: N00014-92-J-1937
Period: September 15, 1992 to June 30, 1997
Total Support: $1,054,008

Thomas W. Reps, Principal Investigator
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
1. Summary of the Completed Project

The goal of this project was to create enhanced tools to support the development of complex software systems. The objective was to create tools that provide powerful language-specific program-manipulation operations for aiding programmers in maintaining, enhancing, and reusing code. The main activities of the project were devoted to:

Program slicing

A particularly important aspect of the project was to explore how "program slicing" could serve as the basis for such program-manipulation operations. The slice of a program with respect to a set of program elements $S$ is a projection of the program that includes only program elements that might affect (either directly or transitively) the values of the variables used at members of $S$. Slicing allows one to find semantically meaningful decompositions of programs—where the decompositions consist of elements that are not textually contiguous. Program slicing is a fundamental operation that can aid in solving many software-engineering problems. For instance, it has applications to program understanding, maintenance, debugging, testing, differencing, specialization, reuse, and merging.

The first phase of the work (1988–94, which was funded by a previous ARPA contract—A.O. 6378—as well as the first two years of A.O. 8856) focused on developing the theoretical foundations of program slicing. The later phases of the project focused on the construction and evaluation of a slicing tool for C, which has been used to investigate the practicality of slicing. Activities included:

- Improving the underlying technology for program slicing (and related operations).
- Implementing program slicers.
- Developing methods for using slicing in software-engineering tools.
- Building slicing-based program-manipulation tools.

This work was reported in publications [8], [13], [14], [15], [16], [23], [26], [27], [37], [38], [39], [40], [41], [43], [46], [47], and [54].

A new approach to program analysis

We discovered that a number of different program-analysis problems could be solved by transforming them to graph-reachability problems. Some of the program-analysis problems that are amenable to this treatment include program slicing, certain dataflow-analysis problems, and the problem of approximating the possible "shapes" that heap-allocated structures in a program can take on. Relationships between graph reachability and other approaches to program analysis were explored. Some techniques that go beyond pure graph reachability were also developed. This work was reported in publications [4], [5], [10], [12], [20], [25], [28], [29], [30], [32], [33], [48], and [52].

Techniques for pointer analysis and shape analysis

Techniques for program manipulation require knowledge of what items pointer variables point to, as well as a characterization of the "shapes" of the structures that pointer variables point to. Our work on this subject was reported in publications [2], [3], [19], [21], [24], [49], and [50].

Incremental computation

Motivated by the goal of speeding up response times of interactive systems, we explored techniques for dealing with "incremental changes" to the inputs of computations. This work was
reported in publications [1], [6], [7], [9], [34], [55], and [56].

Development of the CAPITL program-development environment

CAPITL consists of:

- A shared, object-oriented, versioned database.
- An embedded logic-based data-manipulation language.
- A graphical user interface.

With each software object the database stores a rich set of attributes that describe its syntax, intended semantics, and relationship to other objects. CAPITL is implemented in POL, a data model and deductive query language with elements of persistent, object-oriented and logic-based programming languages. POL is implemented in and tightly coupled with C++. This work was reported in publications [31], [36], [44], and [53].

Publications [10], [32], and [47] are included with this report.

The following papers were written by the Principal Investigator, the co-Investigators, and their students during the period of ARPA Order 8856 (ONR Contract N00014-92-J-1937). Most of the papers are available over the World Wide Web at either http://www.cs.wisc.edu/ reps/ or http://www.cs.wisc.edu/wpis/html.

Books

Journal Publications

Invited Papers

Book Chapters
Reprinted from *ACM Transactions on Programming Languages and Systems* 12, 1 (January 1990), 26-60.
Reprinted from *ACM Transactions on Programming Languages and Systems* 12, 1 (January 1990), 26-60.


Conference Publications


\textbf{Software}


\textbf{Patents}


\textbf{Pending Submissions}


Submitted for journal publication.

**Ph.D. Dissertations**


**Other Publications and Reports**


[52] Reps, T., Sagiv, M., and Horwitz S., Interprocedural dataflow analysis via graph reachability. TR 94-14, Datalogisk Institut, University of Copenhagen, Copenhagen, Denmark, April 1994.


