SEMANTICS OF PROGRAM REPRESENTATION GRAPHS

G. RAMALINGAM and THOMAS REPS
University of Wisconsin – Madison

Program representation graphs are a recently introduced intermediate representation form for programs. In this paper, we develop a mathematical semantics for these graphs by interpreting them as data-flow graphs. We also study the relation between this semantics and the standard operational semantics of programs. We show that the semantics of the program representation graphs is more defined than the program semantics and that for states on which a program terminates normally, the PRG semantics is identical to the program semantics.

CR Categories and Subject Descriptors: D.3.4 [**Programming Languages**]: Processors - *compilers, interpreters, optimization*; E.1 [**Data Structures**] *graphs*

General Terms: Theory

Additional Key Words and Phrases: control dependence, flow dependence, program dependence graph, program representation graph, semantics

1. INTRODUCTION

In this paper, we develop a mathematical semantics for program representation graphs (PRGs) and study its relation to the standard (operational) semantics of programs. Program representation graphs are an intermediate representation of programs, introduced by Yang et al. [Yang89] in an algorithm for detecting program components that exhibit identical execution behaviours. They combine features of static-single-assignment forms (SSA forms) [Shap70, Alpe88, Cytr89, Rose88] and program dependence graphs (PDGs) [Kuck81, Ferr87, Horw89]. They have also been used in a new algorithm for merging program variants[Yang89a].

Program dependence graphs have been used as an intermediate program representation in various applications such as vectorization, parallelization [Kuck81], and merging program variants [Horw89]. Horwitz et al. [Horw88] were the first to address the question of whether PDGs were “adequate” as program representations. They showed (for a simplified programming language) that if the program dependence graphs of two programs are isomorphic, the programs are equivalent in the following sense: for any initial state σ, either both programs diverge or both halt with the same final state.

This work was supported in part by a David and Lucile Packard Fellowship for Science and Engineering, by the National Science Foundation under grant DCR-8552602, by the Defense Advanced Research Projects Agency, monitored by the Office of Naval Research under contract N00014-88-K-0590, as well as by grants from IBM, DEC, and Xerox.

Authors’ address: Computer Sciences Department, University of Wisconsin – Madison, 1210 W. Dayton St., Madison, WI 53706.