CSU499 Project
Progress Report

Global Value Numbering

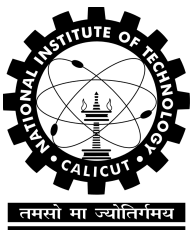
Submitted in partial fulfilment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

Kartik Singhal
B090566CS

Under the guidance of
Dr Vineeth K Paleri



Department of Computer Science and Engineering
National Institute of Technology Calicut
Calicut, Kerala, India - 673 601

Winter Semester 2013

Abstract

Value numbering is a compiler optimization technique to detect redundancy in expressions. We study it in detail; review and compare known algorithms; and aim to implement one of the best among them, and in the process, improve upon the algorithm if possible. We study GCC as an implementation platform for compiler research, identify the chosen algorithm for implementation and select two algorithms and a benchmark suite for performance comparison.

Contents

1 Problem Definition **1**

2 Introduction **2**
2.1 Background and Recent Research 2
2.1.1 Global Value Numbering 2
2.1.2 Literature Survey 2
2.2 Motivation 3

3 Work Done **4**
3.1 Previous Work (Monsoon Semester 2012) 4
3.2 Progress 4
3.2.1 Choice of Algorithm for implementation 4
3.2.2 Preparation for Performance Comparison 5
3.2.3 Study of GCC APIs for Pass Implementation 5

4 Future Work **6**

5 Conclusion **7**

References **8**

Chapter 1

Problem Definition

To study global value numbering, a compiler optimization, in detail; to review and compare known algorithms; to implement one of the best among them; and in the process, improve upon the algorithm if possible.

Chapter 2

Introduction

2.1 Background and Recent Research

2.1.1 Global Value Numbering

Global value numbering (GVN) is a program analysis that categorizes expressions in the program that compute the same static value[1]. This information can be used to remove redundant computations.

2.1.2 Literature Survey

Kildall in 1973[2] introduced the most precise global analysis algorithm for program optimization by using the concept of an optimizing function for generalization. Optimizing functions for constant propagation and common subexpression elimination were described but the algorithm had an exponential cost.

Alpern, Wegman and Zadeck (AWZ) introduced an efficient algorithm in 1988[3] which uses a value graph to represent symbolic execution of a program. It represents the values of variables after a join using a selection function ϕ, similar to the selection function in static single assignment (SSA) form, and treats these functions as uninterpreted, hence remains incomplete.

The polynomial time algorithm introduced by Ruthing, Knoop and Steffen (RKS) in 1999[4] extends the algorithm by AWZ. It employs a normalization process using some rewrite rules for terms involving ϕ functions, until congruence classes reach a fixed point. This results in discovery of more equivalences and is optimal for acyclic programs but remains incomplete.

Karthik Gargi proposed balanced algorithms in 2002[5] which extend AWZ to perform forward propagation and re-association and to consider back edges in SSA graph. This discovers more equivalences but is still incomplete.

Gulwani and Necula in 2004[6] proposed a polynomial time algorithm which is optimal if only equalities of bounded size are considered.

We are interested in a sufficiently efficient but complete algorithm for global analysis problem.

2.2 Motivation

A compiler is a fairly large software program and forms an excellent software engineering case study. Optimizing compilers are hard to build especially when software engineering practices encourage generic programming, which is good for code reuse but bad for run time performance. Study of compiler optimizations provides a good blend of theory (for generality and correctness) and practice (for validation and efficiency). Global Value Numbering, specifically, is an interesting global dataflow analysis for study.

Chapter 3

Work Done

3.1 Previous Work (Monsoon Semester 2012)

In the first phase of the project we started out with basic literature survey on Global Value Numbering to solidify essential concepts and study the approaches taken by researchers in the past.

Next, a study of available options among compiler infrastructures was done and GCC[7] identified as the implementation platform of choice owing to its open-source nature, industry recognition and popularity.

A high level study of structure of GCC, including its intermediate representation (IR) forms (viz. GENERIC[8], GIMPLE[9] and RTL[10]) and pipeline flow in GCC was done next.

Lastly, some practical experience was gained with a naïve implementation of constant propagation optimization as a GCC plugin.

3.2 Progress

3.2.1 Choice of Algorithm for implementation

We have identified **A Simple Algorithm for Global Value Numbering**[11] as the algorithm of choice for implementation of GVN on GCC framework. Salient features of this new algorithm include its complete nature in terms of number of identifiable redundancies (same as that of Kildall's approach), introduction of the concept of a *Value Expression* to represent a set of equivalent expressions, and the inherent simplicity of the algorithm.

3.2.2 Preparation for Performance Comparison

To compare the efficiency and performance of proposed implementation of the new algorithm, it was necessary to identify few good algorithms and benchmarks to run the tests.

We have identified the following two algorithms for comparison, both of which are already implemented in GCC:

- SCC-Based Value Numbering[12] (`tree-ssa-sccvn.c` in GCC source) – This is the current implementation of value numbering in GCC.
- GVN-PRE[1, 13] (`tree-ssa-pre.c` in GCC source) – This is a value based Partial Redundancy Elimination algorithm, which looks like a good choice to observe the difference in number of redundancies identified by our implementation which does not include partial redundancies.

We have also identified SPEC CPU2006[14] as the benchmark suite for comparison testing based on previous work done in this area. More clarity on which tests among the suite are suitable for GVN is expected to be gained after the implementation.

3.2.3 Study of GCC APIs for Pass Implementation

We have chosen GCC version 4.6.3 for implementation.

A comprehensive study of GCC Application Programming Interface for implementation of optimization passes is under progress using GCC source code, GCC Internals Documentation[15], and slides from GCC Internals Course[16], which is necessary to proceed with the proposed implementation.

Chapter 4

Future Work

Next step is to implement the chosen algorithm as a GCC plugin, test and compare its performance with other algorithms and look for any possible improvements in the algorithm.

Chapter 5

Conclusion

Algorithm for implementation is chosen and studied. Two algorithms and a benchmark for performance comparison are identified. Deeper understanding of GCC APIs for implementation is being gained.

References

[1] VanDrunen, T. J. 2004. Partial redundancy elimination for global value numbering (Doctoral dissertation, Purdue University). http://docs.lib.purdue.edu/dissertations/AAI3154748/

[2] Gary A. Kildall. 1973. A unified approach to global program optimization. In *Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages* (POPL '73). ACM, 194-206. http://doi.acm.org/10.1145/512927.512945

[3] B. Alpern, M. N. Wegman, and F. K. Zadeck. 1988. Detecting equality of variables in programs. In *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (POPL '88). ACM, 1-11. http://doi.acm.org/10.1145/73560.73561

[4] Rüthing, O., Knoop, J., & Steffen, B. 1999. Detecting equalities of variables: Combining efficiency with precision. *Static Analysis*, 848-848. http://dl.acm.org/citation.cfm?id=718137

[5] Karthik Gargi. 2002. A sparse algorithm for predicated global value numbering. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation* (PLDI '02). ACM, 45-56. http://doi.acm.org/10.1145/512529.512536

[6] Gulwani, S., & Necula, G. 2004. A polynomial-time algorithm for global value numbering. *Static Analysis*, 703-1020. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.7271

[7] GCC, the GNU Compiler Collection - http://gcc.gnu.org/

[8] GENERIC - http://gcc.gnu.org/onlinedocs/gccint/GENERIC.html

[9] GIMPLE - http://gcc.gnu.org/onlinedocs/gccint/GIMPLE.html

[10] RTL Representation - http://gcc.gnu.org/onlinedocs/gccint/RTL.html

[11] Saleena Nabeezath, Vineeth Paleri. 2013. A Simple Algorithm for Global Value Numbering. *arXiv:1303.1880* - http://arxiv.org/abs/1303.1880

[12] Simpson, Loren Taylor. 1996. Value-driven redundancy elimination. Diss. Rice University. http://scholarship.rice.edu/handle/1911/16942

[13] GVN-PRE - http://gcc.gnu.org/wiki/GVN-PRE

[14] SPEC CPU2006 - http://www.spec.org/cpu2006/

[15] GCC Internals Documentation (for GCC 4.6.3) - http://gcc.gnu.org/onlinedocs/gcc-4.6.3/gccint/

[16] GCC Internals Course - November 2007 - http://www.airs.com/dnovillo/200711-GCC-Internals/