Partial Redundancy Elimination for Multi-threaded Programs

Mohamed A. El-Zawawy and Hamada A. Nayeel

†College of Computer and Information Sciences, Al-Imam M. I-S. I. University, Riyadh, Kingdom of Saudi Arabia
‡Department of Mathematics, Faculty of Science, Cairo University, Giza 12613, Egypt
¶Faculty of Computers and Informatics, Benha University, Benha, Egypt

Summary
Multi-threaded programs have many applications which are widely used such as operating systems. Analyzing multi-threaded programs differs from sequential ones; the main feature is that many threads execute at the same time. The effect of all other running threads must be taken in account. Partial redundancy elimination is among the most powerful compiler optimizations: it performs loop-invariant code motion and common subexpression elimination. We present a system with optimization component which performs partial redundancy elimination for multi-threaded programs.

Key words:
Partial redundancy elimination, Type systems, Multi-threaded programs, Operational semantics, Compiler optimization.

1. Introduction

There are many methods for compiler optimizations; a powerful one of them is partial redundancy elimination (PRE). PRE eliminates redundant computations on some but not necessarily all paths of programs. PRE is a complex optimization as it consists of loop invariant code motion and common subexpression elimination. PRE was established by Morel and Renvoise [17] where they introduce a more general problem (as a system of Boolean equations). Xue and Cai formulated a speculative PRE as a maximum flow problem [27]. Xue and Knoop proved that the classic PRE is a maximum flow problem [28]. Saabas and Ustalu use the type-systems framework to approach this problem [22]. Some optimizations have been added to PRE such as strength reduction [11] and global value numbering [3]. All methods mentioned above are established to operate on sequential programs.

In the present paper, we achieve partial redundancy elimination for multi-threaded programs which are widely used. Operating system is an example of a system software that depends on multi-threading. You can write your document in a word processor while running an audio file, downloading a file from the internet, and/or scanning for viruses (each of these tasks is considered a thread of computations). Web browser as an example can explore your e-mail, while downloading a file in the background.

The key feature of multi-threaded programs is that many threads can be executed at the same time. Consequently, when executing a thread there is an effect that comes from executing other threads. In general, when analyzing multi-threaded programs, the effect of all threads at the same time must be taken in account. Hence, analyzing multi-threaded programs completely differs from sequential ones. Deducing and stating properties of programs can be done using type systems as well as program analysis. Program analysis has algorithmic manner while type systems are more declarative and easy to understand with type derivations that provide human-friendly format of justifications. We present a type system for optimizing multi-threaded programs. Our type system depends on a new analysis, namely modified analysis, and a function called concurrent modified, rather than on anticipability analysis and conditional partial availability analysis used for the while language.

Organization of this paper is as follow. In section 2 we introduce an operational semantics for the language we study. Section 3 presents the concepts of modified analysis and concurrent modified function. Also the soundness of modified analysis, the anticipability analysis, and conditional partial availability analysis for multi-threaded language are discussed in this section. In section 4, we present the type system including the optimization component and prove its soundness. Section 5 and 6 outline related and future work, respectively.

2. Motivation

In this section we introduce the language we study (FWhile), a motivating example, and a natural semantics of FWhile.

2.1 FWhile Language

We assume that our reader is familiar with data flow analysis. We introduce a motivating example to show the importance and obstacles of applying PRE on multi-threaded programs. We use a simple language which we call FWhile. The basic building blocks of FWhile are literals $l \in \text{Lit}$, statements $s \in \text{Stmt}$, arithmetic expressions $a \in \text{AExp}$, and Boolean expressions $b \in \text{BExp}$. These blocks are defined over a set of
Mohamed A. El-Zawawy received: PhD in Computer Science from the University of Birmingham in 2007, M.Sc. in Computational Sciences in 2002 from Cairo University, and a BSc. in Computer Science in 1999 from Cairo University. Dr El-Zawawy is an assistant professor of Computer Science at Faculty of Science, Cairo University Since 2007. Currently, Dr. El-Zawawy is on a sabbatical from Cairo University to College of Computer and Information Sciences, Al-Imam M. I.-S. I. University, Riyadh, Kingdom of Saudi Arabia. During the year 2009, Dr. El-Zawawy held the position of an extra-ordinary senior research at the Institute of Cybernetics, Tallinn University of Technology, Estonia. Dr. El-Zawawy worked as a teaching assistant at Cairo University from 1999 to 2003 and latter at Birmingham University from 2003 to 2007. Dr. El-Zawawy is interested in static analysis, shape analysis, type systems, and semantics of programming languages.

Hamada A. Nayel received the B.Sc. in mathematics from Benha University in 2003. Diploma in computer science and information systems from Institute of Statistical Studies and Research (ISSR), Cairo University in 2005. He worked as a demonstrator in faculty of Science from 2003 to 2009. Since 2009, he works at faculty of Computers and Informatics, Benha University. Mr. Nayel is interested in semantics of programming languages, type systems, static analysis, and shape analysis.