On Semantics of Programming Languages: MSc. Report

Hamada A. Nayel
Department of Computer Science
Faculty of Computers and Artificial Intelligence
Benha University
hamada.ali@fci.bu.edu.eg

Abstract

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. This these focuses on the analysis of multi-threaded programs. The first aim of our work is to implement Partial Redundancy Elimination (PRE) for multi-threaded programs via type systems. PRE is among the most powerful compiler optimization: it performs loop invariant code motion and common subexpression elimination. We present a type system with optimization component which performs PRE for a multi-threaded programs. In addition, we designed a type systems based data race detector. Data race occurs when two threads try to access a shared variable at the same time without a proper synchronization. A detector is a software that determines if the program contains a data-race problem or not. In this work, we developed a detector that has the form of a type system. We present a type system which discovers the data-race problems. The soundness of our type system has been proved.

1 Introduction

There are many methods for compiler optimization; a powerful one of them is 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 [1] where they introduce a more general problem (as a system of boolean equations). Xue and Cai formulated speculative PRE as a maximum flow problem [2]. Xue and Knoop proved that the classic PRE is a maximum flow problem [3]. Saabas and Uustalu use type-systems framework to approach this problem [4]. Some Optimizations have been added to PRE such as strength reduction [5] and global value numbering [6]. All methods mentioned above are established to operate on sequential programs.

2 Methodology

2.1 PRE

We achieve partial redundancy elimination for multi-threaded programs which are widely used [7]. Operating system is an example of 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 [8]. 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.

2.2 Data-race detector

Developing and debugging software that depend on multi-threading is a tricky mission because of ingrained concurrency and indeterminism [7]. There are many bugs occur according to these properties. Detecting and preventing these bugs are important area of research. Bugs have several forms. The most extensively studied one is data-race: two concurrent threads accessing the same shared variable without proper synchronization. Data-race detector is a tool that determines whether a program is a data-race free or not. Two approaches are followed when developing detectors: static approach, and dynamic approach. Static detectors determine whether a program produce a data-race regardless of inputs of the program. Apart from static detectors, dynamic detectors determine whether a program produce a data-race of a given inputs at execution of the program.

Type systems can infer and gather information about programs as well as achieving program analysis. The merits of using type systems are attesting and rationalization of properties of programs directed by their phrase structures. Type systems are actually sufficient frameworks for describing data flow analysis. Type systems are used as a framework for analyzing multi-threaded programs as well as imperative programs.

A static data-race detector has been developed [9]. We introduce a type system that detects data-race problem for multi-threaded programs of a simple language *m-while*. In addition, we proved the soundness of proposed type system. An implementation of a detector based on our type system has been given. This implementation checks any program of *m-while* language for safety.

3 Conclusion

The first contribution is the application of PRE to a multithreaded programming language. Up to our knowledge, this is the first deal with this problem. We use type systems as a tool to solve the problem. We designed a simple type system for optimizing multi-threaded programs. We approach the problem in a simple way; we use usual PRE with simple modifications. We look for variables that have been modified in other threads and exclude the expressions that contain any of the modified variables.

The second contribution is development of a static data-race detector. We use type systems as a frame work to implement this detector. Firstly, we present read type system which computes the variables accessed by read operations. Secondly, we present safe type system. This type system is based on read type system and decides if a program contains data race problems or not. The soundness of these type systems are discussed in this chapter as well. For future work we plan to use type systems as a tool to solve more complicated problems (like deadlock, pointer dangling). We expect type systems to be an amenable and trustable framework to deal with static analyses.

Notification

Although, the author of this thesis changed his area of research, he recommends young researchers to do work in this area. The author did his PhD in Natural Language Processing (NLP) for Biomedical texts [10, 11]. The research areas of the author are: Biomedical NLP [12, 13, 14, 15], NLP for Cyber Security: [16, 17] and NLP for Arabic texts [18, 19, 20].

References

[1] E. Morel and C. Renvoise. Global optimization by suppression of partial redundancies. *Commun. ACM*, 22(2):96–103, February 1979.

[2] Jingling Xue and Qiong Cai. A lifetime optimal algorithm for speculative pre. *ACM Trans. Archit. Code Optim.*, 3(2):115–155, June 2006.

[3] Jingling Xue and Jens Knoop. A fresh look at pre as a maximum flow problem. In Alan Mycroft and Andreas Zeller, editors, *Compiler Construction*, pages 139–154, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[4] Ando Saabas and Tarmo Uustalu. Proof optimization for partial redundancy elimination. *The Journal of Logic and Algebraic Programming*, 78(7):619 – 642, 2009. The 19th Nordic Workshop on Programming Theory (NWPT 2007).

[5] Robert Kennedy, Fred Chow, Peter Dahl, Shin-Ming Liu, Raymond Lo, and Mark Streich. Strength reduction via ssapre. In Kai Koskimies, editor, *Compiler Construction*, pages 144–158, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[6] Preston Briggs and Keith D. Cooper. Effective partial redundancy elimination. *SIGPLAN Not.*, 29(6):159–170, June 1994.

[7] Hamada A. Nayel. On Semantics of Programming Languages. Master's thesis, Benha University, Egypt, 2012.

[8] Mohamed A El-Zawawy and Hamada A Nayel. Partial redundancy elimination for multi-threaded programs. *arXiv preprint arXiv:1111.0640*, 2011.

[9] Mohamed A El-Zawawy and Hamada A Nayel. Type systems based data race detector. *Computer and Information Science*, 5(4):53–60, 2012.

[10] Hamada A. Nayel. *Biomedical Named Entity Recognition*. PhD thesis, Mangalore University, 2018.

[11] Hamada A Nayel. Biomedical Named Entity Recognition: PhD Synopsis. Mangalore University, 2018.

[12] H. L. Shashirekha and H. A. Nayel. A comparative study of segment representation for biomedical named entity recognition. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1046–1052, Sept 2016.

[13] Hamada A. Nayel, H L Shashirekha, Hiroyuki Shindo, and Yuji Matsumoto. Improving Multi-Word Entity Recognition for Biomedical Texts. *International Journal of Pure and Applied Mathematics*, 118(16):301–3019, 2017.

[14] Hamada A. Nayel and Shashirekha H. L. Integrating Dictionary Feature into A Deep Learning Model for Disease Named Entity Recognition. *CoRR*, abs/1911.01600, 2019.

[15] Hamada Nayel and H L Shashirekha. Improving ner for clinical texts by ensemble approach using segment representations. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 197–204, Kolkata, India, December 2017. NLP Association of India.

[16] Hamada A. Nayel and H. L. Shashirekha. Mangalore University INLI@FIRE2018: Artificial Neural Network and Ensemble based Models for INLI. In Parth Mehta, Paolo Rosso, Prasenjit Majumder, and Mandar Mitra, editors, *Working Notes of FIRE 2018 - Forum for Information Retrieval Evaluation, Gandhinagar, India, December 6-9, 2018.*, volume 2266 of *CEUR Workshop Proceedings*, pages 110–118. CEUR-WS.org, 2018.

[17] Hamada A. Nayel and H. L. Shashirekha. Mangalore-University@INLI-FIRE-2017: Indian Native Language Identification using Support Vector Machines and Ensemble Approach. In Prasenjit Majumder, Mandar Mitra, Parth Mehta, and Jainisha Sankhavara, editors, *Working notes of FIRE 2017 - Forum for Information Retrieval Evaluation, Bangalore, India, December 8-10, 2017.*, volume 2036 of *CEUR Workshop Proceedings*, pages 106–109. CEUR-WS.org, 2017.

[18] Hamada A. Nayel, Walaa Medhat, and Metwally Rashad. BENHA@IDAT: Improving Irony Detection in Arabic Tweets using Ensemble Approach. In Parth Mehta, Paolo Rosso, Prasenjit Majumder, and Mandar Mitra, editors, *Working Notes of FIRE 2019 - Forum for Information Retrieval Evaluation, Kolkata, India, December 12-15, 2019*, volume 2517 of *CEUR Workshop Proceedings*, pages 401–408. CEUR-WS.org, December 2019.

[19] Hamada A. Nayel. NAYEL@APDA: Machine Learning Approach for Author Profiling and Deception Detection in Arabic Texts. In Parth Mehta, Paolo Rosso, Prasenjit Majumder, and Mandar Mitra, editors, *Working Notes of FIRE 2019 - Forum for Information Retrieval Evaluation, Kolkata, India, December 12-15, 2019*, volume 2517 of *CEUR Workshop Proceedings*, pages 92–99. CEUR-WS.org, December 2019.

[20] Hamada A. Nayel and Shashirekha H. L. DEEP at HASOC2019: A machine learning framework for hate speech and offensive language detection. In Parth Mehta, Paolo Rosso, Prasenjit Majumder, and Mandar Mitra, editors, *Working Notes of FIRE 2019 - Forum for Information Retrieval Evaluation, Kolkata, India, December 12-15, 2019*, volume 2517 of *CEUR Workshop Proceedings*, pages 336–343. CEUR-WS.org, 2019.