PROGRAM ANALYSIS UNDER RELAXED MEMORY CONCURRENCY

SANJANA SINGH

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI
AUGUST 2023

PROGRAM ANALYSIS UNDER RELAXED MEMORY CONCURRENCY

by

SANJANA SINGH

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of
Doctor of Philosophy

to the

Indian Institute of Technology Delhi
AUGUST 2023

Certificate

This is to certify that the thesis titled **PROGRAM ANALYSIS UNDER RELAXED MEMORY CONCURRENCY** being submitted by **Ms. SANJANA SINGH** for the award of **Doctor of Philosophy** in **Computer Science and Engineering** is a record of bona fide work carried out by her under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Delhi. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Subodh Sharma
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi- 110016

Acknowledgements

I am deeply grateful to everyone who has contributed to the completion of this work.

First and foremost, I express my heartfelt gratitude to my advisor, Subodh Sharma, for his unwavering guidance and support throughout the entire journey. His encouragement and belief in my abilities have not only shaped me as a researcher but also as a confident and level-headed individual. Subodh is a humble, highly motivated and ever curious person. Working with him taught me to enjoy the process and stay committed despite occasional disappointments. I am thankful for his valuable feedback and for keeping me focused on the right track. One of the qualities I truly admire in my advisor is his willingness to give his students the credit they deserve, which has helped us gain better exposure and recognition. Moreover, under Subodh's guidance, I have developed into a more confident and presentable professional. I am grateful to him for his belief in me and for his patience throughout my Ph.D. journey.

I extend my sincere thanks to my committee members, Sanjiva Prasad, Sorav Bansal, and S. Krishna, for for taking the time out to periodically review my ongoing work and provide valuable insights and feedback that significantly contributed to its refinement. I am indebted to S. Arun Kumar for his mentorship and continuous availability for technical and life advice. Guidance of S. Arun Kumar and Subodh Sharma has played a significant role in improving my presentation and communication skills.

My heartfelt thanks go to the staff at the CSE department for their unwavering support, especially Rekha Rathee, Hemant Prasad, S. S. Negi, and Monika Rajan.

I am also grateful to Sandeep K. Singh and Sanjay Goel for their belief in my potential

and encouraging me to pursue a Ph.D. Their support during my undergraduate and graduate years laid a strong foundation for my research journey. I would also like to thank Sapna Gupta who taught me Computer Science in school and developed in me a fascination for the field.

To my dear friends and colleagues, Divyanjali Sharma, Madhukar Yerraguntla, Shailja Pandey, and Vishal Sharma, thank you for making my Ph.D. tenure enjoyable and for being there for me both professionally and emotionally.

A special thanks to my dearest friend and partner Lav Singh, who has been a constant source of strength and belief in my abilities. I am grateful to him for pushing me to apply for Ph.D. at IIT Delhi. He has taught me to loosen up and approach life and work with an ease. His presence in my life has made me a better person.

I was also blessed with the joy of motherhood during my Ph.D. tenure. Balancing the responsibilities of being a young mother and a researcher has taught me valuable lessons in adapting to circumstances without losing sight of my ultimate objectives. My three year old is the source of my inspiration and strength because of whom I have developed enhanced levels of patience and honed my multitasking skills.

I owe my heartfelt appreciation to my parents, Sadhana Singh and Surender Singh, supporting me during my Ph.D. journey as a counselor, guide, and even a nanny for my kid. Their belief in me throughout my life gave me the confidence to take on challenges. I also thank my parents-in-law, Usha Singh and Ram Sharan Singh, for their blessings.

To my siblings, Samyukta Singh and Suraj Pratap Singh, and sibling-in-law, Aditi Singh, thank you for your constant well-wishes and emotional support.

Finally, I acknowledge that a Ph.D. journey is not a solo endeavor; it is made possible by the support and sacrifices of many individuals. I thank everyone in my life for enabling me to embark on this journey with ease.

Sanjana Singh

Abstract

Concurrency is ubiquitous. Concurrent processing improves the speed of execution, and offers better resource utilization. However, the analysis of concurrent programs is remarkably more challenging than sequential counterparts. Concurrency renders an inherently complex program structure and unintuitive control flow. The complexity is further escalated with synchronization primitives used to control access to shared resources. Moreover, the reachable state graph grows exponentially with an increase in the concurrent processing elements resulting in the state space explosion problem.

Relaxed memory models allow out-of-order execution of program instructions that further ravel the program structure and significantly expand the reachable state graph. As a result, the challenge in analyzing concurrent programs becomes more acute under relaxed memory concurrency. Consequently, it is hard to develop concurrent programs that efficiently utilize the permitted ordering relaxations, and equally hard to determine the feasible program outcomes under relaxed memory concurrency. Both the problems may become exacting even for expert programmers.

This work proposes techniques for the ease of development of efficient programs that produce expected outcomes on relaxed memory models. The techniques focus on

verification efficiency, targeting legacy and developed programs, and *developer productivity* and *runtime efficiency*, targeting programs under development.

Stateless model checkers mitigate the state space explosion problem by exploring a reduced state graph comprising representative program executions from each *equivalence class*; where, an equivalence class is a set of *equivalent* executions that is formed based on an *equivalence relation*. This work proposes a novel equivalence relation on the program executions called *view-equivalence* that is at least as coarse as any existing equivalence relation. Consequently, view-equivalence induces the smallest set of equivalence classes, positively reducing the analysis effort. This work also proposes a stateless model checker called `ViEqui`, which partitions program executions on the proposed view-equivalence relation. The fewer equivalence classes under view-equivalence help `ViEqui` achieve faster analysis and better scalability in comparison to the existing stateless model checkers on finer equivalence relations.

As the second contribution, this work explores stateless model checking under relaxed memory consistency specifications or *memory models*. Various program outcomes feasible under the memory model associated with languages (such as `C/C++`) may never manifest on underlying architectures, being disallowed by the architecture's stronger implicit ordering. Existing model checking techniques operate under the memory consistency specification of a language or of an architecture. This work proposes a stateless model checker called `MoCA` that analyzes programs with `C/C++` memory model under a class of architecture memory models called *multi-copy atomics*. `MoCA` performs a precise stateless model checking for program outcomes valid under the `C/C++` specification and feasible under multi-copy atomicity. The precise analysis of

MoCA reduces the cognitive load on developers of sorting the feasible from infeasible outcomes while also reducing the model checking effort by restricting the set of outcomes to analyze.

The C/C++ language is known to have intricate memory consistency semantics. The work with MoCA uncovers the complexity of the semantics that a C/C++ developer is exposed to. With a focus on reducing development effort, the final element of this study proposes the first `fence` synthesis technique for C/C++ memory model. The technique takes a buggy C/C++ program with a correctness specification and presents an automated fix of the program with additional synchronization on concurrent elements through `fences`. This work proposes a technique called FenSying that fixes the input program with minimal necessary synchronization overhead.

In summary, this work proposes analysis techniques for developing correct and efficient concurrent programs for relaxed memory models. The focus of this work is on verification efficiency (through ViEqui and MoCA), developer productivity (through MoCA and FenSying), and runtime efficiency (through FenSying). Each technique is accompanied by an effective tool support.

सार

समवर्तीता सर्वव्यापी है. समवर्ती प्रसंस्करण से निष्पादन की गति में सुधार होता है, और बेहतर संसाधन उपयोग प्रदान करता है। हालाँकि, समवर्ती कार्यक्रमों का विश्लेषण है अनुक्रमिक समकक्षों की तुलना में उल्लेखनीय रूप से अधिक चुनौतीपूर्ण। समवर्ती एक प्रस्तुत करता है स्वाभाविक रूप से जटिल कार्यक्रम संरचना और सहज ज्ञान युक्त नियंत्रण प्रवाह। जटिलता साझा तक पहुंच को नियंत्रित करने के लिए उपयोग किए जाने वाले सिंक्रोनाइज़ेशन प्रिमिटिव के साथ इसे और बढ़ाया गया है संसाधन। इसके अलावा, पहुंच योग्य स्थिति का ग्राफ वृद्धि के साथ तेजी से बढ़ता है समवर्ती प्रसंस्करण तत्वों के परिणामस्वरूप राज्य अंतरिक्ष विस्फोट की समस्या उत्पन्न होती है।

रिलेक्सड मेमोरी मॉडल प्रोग्राम निर्देशों के आउट-ऑफ़-ऑर्डर निष्पादन की अनुमति देते हैं जो कि प्रदान करते हैं- वे प्रोग्राम संरचना को स्पष्ट करते हैं और पहुंच योग्य स्थिति ग्राफ़ का महत्वपूर्ण रूप से विस्तार करते हैं। परिणामस्वरूप, समवर्ती कार्यक्रमों के विश्लेषण में चुनौती और अधिक गंभीर हो जाती है आरामदेह स्मृति संगामिति के अंतर्गत। परिणामस्वरूप, समवर्ती विकास करना कठिन है ऐसे प्रोग्राम जो अनुमत आदेश संबंधी छूटों का कुशलतापूर्वक उपयोग करते हैं, और समान रूप से कठिन भी रिलेक्सड स्मृति संगामिति के तहत व्यवहार्य कार्यक्रम परिणामों को निर्धारित करने के लिए। दोनों विशेषज्ञ प्रोग्रामर के लिए भी समस्याएँ विकट हो सकती हैं।

यह कार्य कुशल कार्यक्रमों के विकास में आसानी के लिए तकनीकों का प्रस्ताव करता है रिलेक्सड मेमोरी मॉडल पर अपेक्षित परिणाम उत्पन्न करें। तकनीकें सत्यापन दक्षता, विरासत और विकसित

कार्यक्रमों को लक्षित करने और डेवलपर समर्थक पर ध्यान केंद्रित करती हैं। लचीलापन और रनटाइम दक्षता, विकास के तहत कार्यक्रमों को लक्षित करना।

स्टेटलेस मॉडल चेकर्स खोज करके राज्य अंतरिक्ष विस्फोट की समस्या को कम करते हैं प्रत्येक समतुल्य से प्रतिनिधि कार्यक्रम निष्पादन को शामिल करते हुए कम किया गया राज्य ग्राफ़- लेंस वर्ग; जहां, एक समतुल्य वर्ग समतुल्य निष्पादन का एक सेट है जो बनता है तुल्यता संबंध पर आधारित. यह कार्य एक नवीन तुल्यता संबंध का प्रस्ताव करता है प्रोग्राम निष्पादन पर दृश्य-समतुल्यता कहा जाता है जो कम से कम उतना ही मोटा है मौजूदा तुल्यता संबंध. नतीजतन, दृश्य-समतुल्यता सबसे छोटे को प्रेरित करती है तुल्यता वर्गों का सेट, विश्लेषण प्रयास को सकारात्मक रूप से कम करता है। यह कार्य भी प्रो- ViEqui नामक एक स्टेटलेस मॉडल चेकर बनाता है, जो प्रोग्राम निष्पादन को विभाजित करता है प्रस्तावित दृश्य-समतुल्य संबंध पर। विचाराधीन कम तुल्यता वर्ग- समतुल्यता ViEqui को तुलना में तेज़ विश्लेषण और बेहतर स्केलेबिलिटी प्राप्त करने में मदद करती है बेहतर तुल्यता संबंधों पर मौजूदा स्टेटलेस मॉडल चेकर्स के लिए।

दूसरे योगदान के रूप में, यह कार्य आराम के तहत स्टेटलेस मॉडल चेकिंग की पड़ताल करता है मेमोरी संगति विनिर्देश या मेमोरी मॉडल। विभिन्न कार्यक्रम परिणाम- भाषाओं (जैसे C/C++) से जुड़े मेमोरी मॉडल के तहत सिबल कभी नहीं हो सकता अंतर्निहित आर्किटेक्चर पर प्रकट, आर्किटेक्चर के मजबूत द्वारा अस्वीकृत किया जा रहा है अंतर्निहित आदेश. मौजूदा मॉडल जाँच तकनीकें मेमोरी के अंतर्गत काम करती हैं किसी भाषा या वास्तुकला की स्थिरता विशिष्टता। यह कार्य प्रस्तावित है MoCA नामक एक स्टेटलेस मॉडल चेकर जो C/C++ मेमोरी वाले प्रोग्रामों का विश्लेषण करता है आर्किटेक्चर मेमोरी मॉडल के एक वर्ग के तहत मॉडल जिसे मल्टी-कॉपी एटॉमिक्स कहा जाता है। मोका के अंतर्गत मान्य कार्यक्रम परिणामों के लिए एक सटीक स्टेटलेस मॉडल जाँच करता है सी/सी++ विशिष्टता और बहु-प्रतिलिपि परमाणुता के तहत व्यवहार्य। MoCA का सटीक विश्लेषण व्यावहारिक से व्यवहार्य को छांटने में डेवलपर्स पर संज्ञानात्मक भार को कम करता है- सेट को सीमित करके मॉडल जाँच प्रयास को कम करते हुए परिणामों को ख़राब करें विश्लेषण करने के लिए परिणाम.

C/C++ भाषा को जटिल मेमोरी संगति शब्दार्थ के लिए जाना जाता है। MoCA के साथ काम करने से C/C++ डेवलपर के शब्दार्थ की जटिलता का पता चलता है से अवगत कराया। विकास प्रयास को कम करने पर ध्यान देने के साथ, इसका अंतिम तत्व अध्ययन C/C++ मेमोरी मॉडल के लिए पहली बाड़ संश्लेषण तकनीक का प्रस्ताव करता है। तकनीक एक त्रुटिहीन C/C++ प्रोग्राम को शुद्धता विनिर्देशन के साथ लेती है और प्रस्तुत करती है समवर्ती तत्वों पर अतिरिक्त सिंक्रनाइज़ेशन के साथ कार्यक्रम का एक स्वचालित समाधान- बाड़ के माध्यम से प्रवेश. यह कार्य फ़ेंसिंग नामक एक तकनीक का प्रस्ताव करता है जो ठीक करती है न्यूनतम आवश्यक सिंक्रोनाइज़ेशन ओवरहेड के साथ इनपुट प्रोग्राम।

संक्षेप में, यह कार्य सही और प्रभावोत्पादक विकास के लिए विश्लेषण तकनीकों का प्रस्ताव करता है। रिलेक्सड मेमोरी मॉडल के लिए वैज्ञानिक समवर्ती कार्यक्रम। इस काम पर फोकस है सत्यापन दक्षता (ViEqui और MoCA के माध्यम से), डेवलपर उत्पादकता (के माध्यम से)। MoCA और FenSying), और रनटाइम दक्षता (FenSying के माध्यम से)। प्रत्येक तकनीक है एक प्रभावी उपकरण समर्थन के साथ।

Contents

List of Figures

List of Tables