Research Statement

Sanjana Singh
Ph.D.
Indian Institute of Technology Delhi

I hold a Ph.D. degree from the Indian Institute of Technology Delhi (IITD) on "Program Analysis under Relaxed Memory Concurrency". This research was conducted under the guidance of Prof. Subodh Sharma from the Department of Computer Science and Engineering at IITD. My research work can be described by the following keywords: stateless model checking, dynamic and static program analysis, concurrency, weak memory models, sound and optimal analysis, efficient techniques for scalability and performance, as well as effective implementation strategies. For the next phase of my career, I am highly motivated to pursue a post-doctoral position where I can leverage my research background and skill set in order to contribute to the development of pragmatic and impactful solutions.

In this research statement, I provide an overview of my background, highlight my significant contributions, and outline my future research goals.

Technical strengths. I have acquired a comprehensive understanding of memory models, encompassing various aspects such as their axiomatic semantics, permitted program behaviors, and the strengths and weaknesses they exhibit in terms of performance, applicability, and comprehensibility. These models include a range of C11-style language models like C11, RC11, RC20, RA, SRA, WRA, RAR, as well as architecture models such as TSO, PSO, ARMv7, ARMv8, and Power. I am also well-versed in the semantics of both multicopy and non-multicopy atomics.

Moreover, my expertise extends to the domains of verification and program analysis of concurrent programs, which I extensively explored throughout my Ph.D. research. I specifically focused on the development of stateless model checking techniques and utilized static program transformation and SAT solving methodologies. A central theme of my work was the pursuit of efficient techniques, leading me to delve into areas such as coarse trace equivalence, bounding techniques (such as depth and context), and strategies to enhance the precision of analysis.

Tools and development strengths. As part of my Ph.D., I undertook significant software development efforts for each project, resulting in academic tools. Notably, I focused on creating stateless model checkers [7, 8] within runtime-analysis frameworks like Nidhugg [4] and rInspect [5], developed using C++. These model checkers analyze intermediate representations of input programs generated and instrumented using llvm. Additionally, I developed a fence synthesis technique [6] for the C11 memory model, utilizing the CDSChecker model checker [1], Z3 SAT solver, and Nextworkx package for various computations, with its core development in Python. Moreover, I have working knowledge of state-of-the-art tools such as GenMC [2], TruSt [3], HMC [2], and RVF-SMC [9].

Personal strengths. I possess proficiency in both spoken and written English. As an individual, I embody discipline, dedication, and a strong work ethic. My problem-solving abilities are characterized by creativity and an aptitude for designing effective solutions. During my Ph.D., I faced numerous publication rejections, which instilled in me a sense of resilience and perseverance. I adopted a positive approach towards each review, recognizing that they played a crucial role in strengthening the quality and impact of my work.

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. In this process, I have developed enhanced levels of patience and honed my multitasking skills.

Key contributions of Ph.D. dissertation. My Ph.D. work proposes techniques to facilitate the development of efficient programs that produce expected outcomes under relaxed memory consistency models. In particular, I make three key contributions.

1. *Efficient stateless model checker* [8]. This work proposes the coarsest existing equivalence relation on program executions called *view-equivalence*. View-equivalence results in the smallest number of partitions of equivalent executions and hence, reduces the stateless model checking effort. This work also proposes a sound, complete, and optimal stateless model checker to analyze concurrent programs based on view-equivalence partitioning.
2. *Precise stateless model checker for C11 under MCA* [7]. This work proposes a sound and precise stateless model checker that analyzes only those executions of C/C++ programs that can manifest on a class of architecture memory models called *multi-copy atomics* (MCA). The precise analysis results in improved verification performance and reduced development load.
3. *Automated correction under C11* [6]. This work proposes a fence synthesis technique under C11 memory model. The work uses C11 fences to preserve the ordering between program instructions that are otherwise unordered under the relaxed semantics of C11, and produces the most runtime-efficient, bug-free program. This work also proposes a related near-optimal fence synthesis technique that significantly outperforms the optimal technique.

Other projects.

1. Software Source-Code Analysis Related to Plagiarism Dispute. A report developed for the Hon'ble Delhi High Court.
2. Automated Bug Detection and Repair under C11: Insights and Experiences (*under peer review*). The first study of the existing development support for C11 programs. This work proposes a novel set of stress tests the uncover previously unreported strengths and weaknesses. Further, building upon the findings, this work proposes improvements for automated repair under C11.
3. View-equivalence based stateless model checking under RA memory model.
   The RA memory model has gained popularity in recent years as it strikes a good balance between performance and programmability. However, the extension of view-equivalence based stateless model checker for RA is non-trivial as RA does not provide much of the assumed ordering.
4. Lock-aware extension of view-equivalence based stateless model checking.
   Locks increase the challenge for stateless model checkers as different orders of lock acquisitions may lead to different equivalence classes. This work efficiently extends support for locks by recognizing the feasibility of the reads-from relations instead of considering all acquisition orders.
5. Website design for the 2^{nd} Indian SAT+SMT school and the Vertecs[2] research group at IITD.

Future Research Goals.

- During my Ph.D., I focused on the development of sound and optimal analysis techniques which are limited in scalability and performance. Moving forward, I aim to create scalable approaches capable of efficiently handling large programs. These techniques may leverage symbolic execution, integrate machine learning, or utilize parallelization, all while ensuring the quality of results remains satisfactory.
- In addition, I am interested in exploring the application of program analysis techniques to other programming paradigms, including concurrent data structures, database transactions, distributed systems, and neural networks.
- While I have briefly explored program synthesis with my C11 fence synthesis project, I believe there is a vast scope for synthesis under relaxed models that I would like to explore and address. Specifically, there is scope for automated correction of bugs induced by relaxed semantics and for automated transformations that optimize the utilization of relaxed ordering.
- Furthermore, I would like to emphasize that apart from gaining expertise in my Ph.D. topics, I have also honed my problem-solving skills. As a result, I am eager to explore new research areas and expand the breadth of my knowledge and expertise.

References

[1] computersforpeace. model-checker. https://github.com/computersforpeace/model-checker, 2021.

[2] MPI-SWS. genmc. https://github.com/MPI-SWS/genmc, 2021.

[3] MPI-SWS. genmc. https://github.com/MPI-SWS/genmc, 2022.

[4] Nidhugg. Nidhugg. https://github.com/nidhugg/nidhugg, 2021.

[5] rinspect. rinspect. https://github.com/rinspect/rinspect, 2017.

[6] Sanjana Singh, Divyanjali Sharma, Ishita Jaju, and Subodh Sharma. Fence synthesis under the c11 memory model. In *Automated Technology for Verification and Analysis: 20th International Symposium, ATVA 2022, Virtual Event, October 25–28, 2022, Proceedings*, pages 83–99. Springer, 2022.

[7] Sanjana Singh, Divyanjali Sharma, and Subodh Sharma. Dynamic verification of c11 concurrency over multi copy atomics. In *2021 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pages 39–46. IEEE, 2021.

[8] Sanjana Singh and Subodh Sharma. Stateless model checking based on view-equivalence (*under review*). 2023.

[9] ViToSVK. nidhugg. https://github.com/ViToSVK/nidhugg, 2021.