Saikat Chakraborty

saikatc.info

Phone: +1 (434) 242-1306 E-mail : saikatc@microsoft.com

About me

I am a senior researcher at the Research in Software Engineering (RiSE) group at Microsoft Research, USA. My research area is *Programming Language Processing (PLP)* – a coalescence between Programming Languages and Machine Learning. I have been working towards empowering AI models generate provably correct code. One of my major research thrust is making Proof-Oriented-Programming more accessible to programmers - allowing them to simultaneously write code and proofs of correctness. In this problem formulation, I focus in two orthogonal sub-problems, i.e., Specification of Correctness Formulation, and Proof Automation. In addition, I am interested in building General Purpose LLM for Code and Source Code Vulnerability Detection.

Work Experience

September 2022 -Till date	**Senior Researcher** at **Microsoft Research**, USA Research in Software Engineering (RiSE) group.
January 2019 -Summer 2022	**Research Assistant** Arise Lab, Columbia University, New York, NY.
Summer 2021	**Software Engineer Intern** at **Facebook Inc.** Probability (bigcode) team - Prediction of Code review metric from code diffs.
Summer 2019	**Software Engineer Intern** at **Google LLC.**, Sunnyvale, CA. BinEval team. - Identification of embedded malicious code in cloud documents.
Summer 2017 & 2018	**Research Intern Fujitsu Laboratories of America, Sunnyvale, CA** AI-based fault localization for automated program repair.

Selected Projects

Thrust-1: Proof-Oriented Programming - Proof Automation

FStarData-Set [1]	We curate a benchmark of F* code consisting 900K lines of open-source programs and proofs. We use AI to automate proof-oriented programs based on this dataset. Fine-tuned smaller language models show promising results, matching or outperforming larger models like GPT-4, with type-based retrieval techniques enhancing performance.
Loopy [2]	This paper explores using Large Language Models (LLMs) to tackle the complex task of finding inductive loop invariants for formal program verification. By curating a dataset and designing prompts to leverage LLMs, the study demonstrates improved automated verification capabilities compared to traditional symbolic methods.
iRank [5]	In this work, we propose a *re-ranking* approach for the loop invariants generated by LLMs. We have designed a ranker that can distinguish between correct inductive invariants and incorrect attempts based on the problem definition.

Thrust-2: Proof-Oriented Programming - Specification Formulation

NL2Post-Cond [3]	In this work, we investigate using Large Language Models (LLMs) to transform informal natural language descriptions of code into formal, checkable postconditions. Our method, "nl2postcond", effectively generates correct postconditions, helping identify bugs. We validated this by catching 64 real-world bugs from the Defects4J dataset.
TiCoder[4]	We introduce TiCoder, an interactive workflow that uses tests to clarify user intent for more accurate code generation by LLMs. In a study with 15 programmers, TiCoder improved AI-generated code evaluation accuracy and reduced cognitive load. TiCoder achieved a 38.43% average accuracy improvement within a few interactions, while also automatically generating unit tests.

Thrust-3: General Purpose LLM for Programming

NatGen[10]	A new pre-training objective "Naturalizing" of source code, exploiting code's bimodal, dual-channel (formal & natural channels) nature. Learning to generate equivalent, but more natural code, at scale, over large corpora of open-source code, without explicit manual supervision, helps the model learn to both ingest & generate code.
PLBART [14]	A large scale pretrained model for multiple programming languages. PLBART is trained on several hundred millions source code in Java and Python and technical natural languages from stackoverflow.

Thrust-4: Source Code Vulnerability Detection

CausalVul [6]	We introduced causality into deep learning-based vulnerability detection, by applying the causal learning algorithms, we remove the use of spurious features from prediction.
Concord[8]	We introduce CONCORD, a self-supervised pre-training approach where we emphasize benign code clones and distance deviant ones, significantly enhancing DL models' efficiency in identifying program equivalence and detecting vulnerabilities in code.
DISCO[11]	A source code understanding pretrained model that learns to reason about the functional properties of the code.
ReVeal [19]	An empirical study for understanding the feasibility of Deep Learning Based Vulnerability Detection for detecting real world vulnerabilities. We identified major challenges in using DL-based systems for Vulnerability detection, and proposed prospective solution.

Selected Publications

[1] [**ICSE'25**] Towards Neural Synthesis for SMT-Assisted Proof-Oriented Programming, S. Chakraborty, G. Ebner, S. Bhat, S. Fakhoury, S. Fatima, S. Lahiri, N. Swamy. Accepted to be published in 47th International Conference in Software Engineering (ICSE) 2025.

[2] [**FMCAD'24**] Finding inductive loop invariants using large language models, A. Kamath, A. Senthilnathan, S. Chakraborty, P. Deligiannis, S. K. Lahiri, A. Lal, A. Rastogi, S. Roy, R. Sharma. Accepted to be published in Formal Methods in Computer-Aided Design 2024.

[3] [**FSE'24**] Formalizing Natural Language Intent into Program Specifications via Large Language Models, M. Endres, S. Fakhoury, S. Chakraborty, SK. Lahiri, accepted to be published at The ACM International Conference on the Foundations of Software Engineering (FSE) 2024 .

[4] [**TSE'24**] LLM-based Test-driven Interactive Code Generation: User Study and Empirical Evaluation, S. Fakhoury, A. Naik, G. Sakkas, S. Chakraborty, SK. Lahiri, accepted to be published at The IEEE Transaction on Software Engineering.

[5] [**EMNLP'23 (findings)**] Ranking LLM-Generated Loop Invariants for Program Verification, S. Chakraborty, S. K Lahiri, S. Fakhoury, M. Musuvathi, A. Lal, A. Rastogi, A. Senthilnathan, R. Sharma, N. Swamy, Published at the findings of The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP) 2023.

[6] [**ICSE'24**] Towards Causal Deep Learning for Vulnerability Detection, Md. Rahman, I. Ceka, C. Mao, S. Chakraborty, B. Ray, W. Le, Published at 46th International Conference on Software Engineering (ICSE) 2024.

[7] [**ESEC/FSE'23**] GrACE: Generation using Associated Code Edits, P. Gupta, A. Khare, Y. Bajpai, S. Chakraborty, S. Gulwani, A. Kanade, A. Radhakrishna, G. Soares, A. Tiwari, Published at The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) 2023.

[8] [**ISSTA'23**] CONCORD: Clone-aware Contrastive Learning for Source Code, Y. Ding, S. Chakraborty, L. Buratti, S. Pujar, A. Morari, G. Kaiser, B. Ray,32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA) 2023. [**Won ACM Sigsoft Distinguished Paper Award**].

[9] [**ICSE-NIER'23**] On ML-Based Program Translation: Perils and Promises, A. Malyala, K. Zhou, B. Ray, S. Chakraborty Published at the IEEE/ACM International Conference on Software Engineering - New Ideas and Emerging Results (NIER) track (ICSE-NIER) 2023.

[10] [**ESEC/FSE'22**] NatGen: Generative pre-training by "Naturalizing" source code, S. Chakraborty, T. Ahmed, Y. Ding, P. Devanbu, B. Ray, Published at The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) 2022 **(Acceptance rate : 22%)**.

[11] [**ACL'22**] Towards Learning (Dis)-Similarity of Source Code from Program Contrasts, Y. Ding, L. Buratti, S. Pujar, A. Morari, B. Ray, S. Chakraborty, Published at 60th Annual Meeting of the Association for Computational Linguistics **(Acceptance rate : 20.75%)**.

[12] [**ASE'21**] On Multi-Modal Learning of Editing Source Code, S. Chakraborty, B. Ray, Published in The 36th IEEE/ACM International Conference on Automated Software Engineering. **(Acceptance rate : 28%)**.

[13] [**EMNLP'21 (findings)**] Retrieval Augmented Code Generation and Summarization, MDR. Parvez, WU. Ahmad, S. Chakraborty, B. Ray, K. Chang, Findings of The 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP-findings), 2021. **(Acceptance rate : 38%)**.

[14] [**NAACl'21**] A Unified Pre-training for Program Understanding and Generation, WU. Ahmad[§], S. Chakraborty[§], B. Ray, K. Chang, Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2021. [§] Co-first authors. **(Acceptance rate : 26%)**.

[15] [**TSE'20**] CODIT: Code Edits with Tree Based Machine Translation, S. Chakraborty, Y. Ding, M. Allamanis, B. Ray, in IEEE Transactions on Software Engineering, 2020. **(Impact factor : 3.331)**.

[16] [**ACL'20**] A Transformer-based Approach for Source Code Summarization (short paper), WU. Ahmad, S. Chakraborty, B. Ray, K. Chang, 58th Annual Meeting of the Association for Computational Linguistics (ACL) 2020. **(Acceptance rate : 17.6%)**.

[17] [**SCAM'19**] Toward Optimal Selection of Information Retrieval Models for Software Engineering Tasks, MM. Rahman, S Chakraborty, G. Kaiser, B. Ray, 19th International Working Conference on Source Code Analysis and Manipulation (SCAM) 2019. **(Acceptance rate : 39.6%)**.

[18] [**ACL'18**] Building Language Models for Text with Named Entities, R. Parvez, S. Chakraborty, B. Ray, K. Chang, 56th Annual Meeting of the Association for Computational Linguistics (ACL) 2018. **(Acceptance rate : 24.9%)**.

[19] [**TSE'21**] Deep Learning based Vulnerability Detection: Are We There Yet? S. Chakraborty, R. Krishna, Y. Ding, B. Ray, accepted to be published in IEEE Transaction of Software Engineering. **(Impact factor : 3.331%)**.

Education

August 2022	**Ph.D. in Computer Science** **Columbia University, New York, NY, USA** **Area:** Artificial Intelligence for Software Engineering. **Expertise:** Source Code Analysis, Deep Learning, Natural Language Processing, Neural Machine Translation. **Thesis:** Learning to Edit Code. **Advisor:** Dr. Baishakhi Ray.

Selected Talks

1. Detecting Vulnerabilities in Source Code at Vulnerability Detection and Security Research, National Security Agency, (NSA) (November 2021).
2. PLBART : Unified Pre-training for Program Understanding and Generation at Decal Lab - UC Davis (April 2021), NAACL 2021 (June 2021), IBM Research (June 2021), Facebook BigCode team (August 2021).
3. Programming Language Processing - Learning to Edit Code at Programming Systems Lab Research Seminar, Department of Computer Scicence, UC Berkeley (May 2021).
4. Machine Learning for Source Code Analysis at Open University, UK and Toshiba, UK (March 2021).

Service Experience

Reviewer	TSE, TOSEM, IEEE Software. NeurIPS, EMNLP, ICLR (OpenReview) ACL, NAACL, EACL (ACL Rolling Review)
PC member	International Conference of Software Engineering **(ICSE)** - 2023, 2024, 2025 Foundation of Software Engineering **(FSE)** - 2023, 2025 Automated Software Engineering **(ASE)** - 2023
Organizing	**Virtualization Chair** - ESEC/FSE 2023. **Review Process Chair** - ASE 2024.
Session Chair	**ESEC/FSE 2021.** Program Repair, Code Recommendation. **ESEC/FSE 2022.** Program Repair/Synthesis, Human Computer Interaction.
Leadership	**Secretary,** Association of Bangladeshi Students at UVa. (2017-2018). **Founding Vice President**, Engineering Students' Association of Bangladesh . **Organizer**, International Engineering Innovation Summit Bangladesh, 2015.