Pranav Garg

Amazon India, Brigade Gateway, 8th Floor, 26/1, Dr. Rajkumar Road, Malleshwaram, Bengaluru, India 560055
+91.705.403.1116 • prangarg@amazon.com • http://pranav-garg.com/

Research Interests

Machine learning, causal inference, formal methods, security.

Employment

Amazon
Research Scientist, Core Machine Learning Team, India — Aug 2015 - present
Worked on the R&D of a machine learning system that characterizes network traffic to detect malware in EC2. Also working on estimating causal effects of customer e-commerce purchase actions on long-term downstream revenue.

Education

University of Illinois at Urbana-Champaign — 2015
Ph.D., Department of Computer Science
Thesis: Learning-based Inductive Invariant Synthesis
Advisor: Madhusudan Parthasarathy

Indian Institute of Technology Kanpur, India — 2009
B.Tech., Department of Computer Science and Engineering

Research Experience

University of Illinois at Urbana-Champaign
Research Assistant, Department of Computer Science — 2009 - 2015
Machine learning based automatic program verification including learning based inductive program invariant synthesis.

Microsoft Research India — Bangalore, India
Research Intern — Spring 2013
Designed and developed new deductive verifiers for concurrent programs. Further, developed static program analysis techniques for proving absence of security bugs, like buffer overflows, in concurrent Linux device drivers by automated learning of precise, adequate inductive invariants.
Mentor: Akash Lal

NEC Laboratories America — Princeton, NJ
Research Intern — Summer 2011
Designed and developed a hybrid unit test generation technique that combines random test generation with symbolic execution to significantly improve the code coverage provided by a feedback-directed random unit testing framework.
Mentor: Franjo Ivančić

École Polytechnique Fédérale de Lausanne (EPFL) — Lausanne, Switzerland
Undergraduate Research Intern — Summer 2008
Full-blown functional verification of a deadlock immunity system nucleus using automated theorem provers. The

verified nucleus helps guarantee deadlock immunity in large Java systems with modest runtime and verification overheads.
Advisor: George Candea

Awards

Gold medal in the invariant synthesis track at SyGuS-COMP 2015 and SyGuS-COMP 2016.

Travel grant for the following conferences: MVD 2010, POPL 2011, ISCA 2011, PLDI 2013 and CAV 2014.

Indian National Physics Olympiad 2005.

National Talent Search Scholarship awarded by the Government of India 2003-2008.

Publications

Thesis

[1] Pranav Garg. Learning-based Inductive Invariant Synthesis. Ph.D. Dissertation. University of Illinois at Urbana-Champaign (UIUC). July 2015.

Refereed Journal Articles

[2] Pranav Garg, Christof Löding, P. Madhusudan, and Daniel Neider. Quantified data automata for linear data structures: a register automaton model with applications to learning invariants of programs manipulating arrays and lists. (Invited Paper). *Formal Methods in System Design (FMSD) 47(1)*, Special Issue on Computer Aided Verification, 2015.

[3] Rajeev Alur, Rastislav Bodik, Eric Dallal, Dana Fisman, Pranav Garg, Garvit Juniwal, Hadas Kress-Gazit, P. Madhusudan, Milo M. K. Martin, Mukund Raghothaman, Shamwaditya Saha, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. Syntax-Guided Synthesis. (Invited Paper). *Dependable Software Systems Engineering*, 2015.

Refereed Conference Publications

[4] Daniel Neider, P. Madhusudan, Pranav Garg, Shambwaditya Saha and Daejun Park. Invariant Synthesis for Incomplete Verification Engines. In *24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* **(TACAS)**, 2018.

[5] Alex Gyori, Pranav Garg, Edgar Pek and P. Madhusudan. Efficient Incrementalized Runtime Checking of Linear Measures on Lists. In *10th IEEE International Conference on Software Testing, Verification and Validation* **(ICST)**, 2017.

[6] Pranav Garg, Daniel Neider, P. Madhusudan, Dan Roth. Learning invariants using decision trees and implication counterexamples. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* **(POPL)**, 2016.

[7] Shambwaditya Saha, Pranav Garg, P. Madhusudan. Alchemist: Learning Guarded Affine Functions. In *Proceedings of the 27th International Conference on Computer Aided Verification* **(CAV)**, 2015.

[8] Ankush Desai, Pranav Garg, and P. Madhusudan. Natural Proofs for Asynchronous Programs using Almost-Synchronous Invariants. In *Proceedings of the 28th ACM International Conference on Object Oriented Programming Systems Languages & Applications* **(OOPSLA)**, 2014.

[9] Pranav Garg, Christof Löding, P. Madhusudan, and Daniel Neider. ICE: A Robust Framework for Learning Invariants. In *Proceedings of the 26th International Conference on Computer Aided Verification* **(CAV)**, 2014. **Invited for submission to Journal of the ACM (JACM)**.

[10] Pranav Garg, Christof Löding, P. Madhusudan, and Daniel Neider. Learning Universally Quantified Invariants of Linear Data Structures. In *Proceedings of the 25th International Conference on Computer Aided Verification* **(CAV)**, 2013. **Invited for submission to Formal Methods in System Design (FMSD)**.

[11] Pranav Garg, P. Madhusudan and Gennaro Parlato. Quantified Data Automata on Skinny Trees: An Abstract Domain for Lists. In *Proceedings of the 20th Static Analysis Symposium* **(SAS)**, 2013.

[12] Xiaokang Qiu, Pranav Garg, Andrei Stefanescu, and P. Madhusudan. Natural Proofs for Structure, Data, and Separation. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation* **(PLDI)**, 2013.

[13] Pranav Garg, Franjo Ivancic, Gogul Balakrishnan, Naoto Maeda, and Aarti Gupta. Feedback-Directed Unit Test Generation for C/C++ using Concolic Execution. In *Proceedings of the 35th International Conference on Software Engineering* **(ICSE)**, 2013.

[14] Rishi Agarwal, Pranav Garg, and Josep Torrellas. Rebound: Scalable Checkpointing for Coherent Shared Memory. In *Proceedings of the 38th International Symposium on Computer Architecture* **(ISCA)**, 2011.

[15] Pranav Garg and P. Madhusudan. Compositionality Entails Sequentializability. In *Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* **(TACAS)**, 2011.

Refereed Workshop Publications

[15] Ankush Desai, Pranav Garg, and P. Madhusudan. A New Reduction for Event-Driven Distributed Programs. In *7th International Workshop on Exploiting Concurrency Efficiently and Correctly* **(EC2)**, 2014.

Invited Tutorials

[16] Pranav Garg, and P. Madhusudan. Machine Learning based methods for Invariant Synthesis. In *27th International Conference on Computer Aided Verification* **(CAV)**, 2015.

Industry Publications (Internally peer-reviewed)

[17] Pranav Garg, and Vineet Chaoji. Malware Detection in Amazon EC2. In *5th Amazon Machine Learning Conference* **(AMLC)**, 2017.

Articles under Submission

[18] Edgar Pek, Pranav Garg, Muntasir Raihan Rahman, Indranil Gupta, P. Madhusudan and Karl Palmskog. Certified Models for Eventual Consistency of Practical Key-Value Stores. 2018. Under Submission.

Patents

[18] Pranav Garg, Franjo Ivancic, Gogul Balakrishnan, Naoto Maeda, and Aarti Gupta. Feedback-Directed Random Class Unit Test Generation Using Symbolic Execution. US Patent Number 20130091495, Issued April 11, 2013.

[19] Pranav Garg, Vineet Chaoji, and Karthik Gurumoorthy. Jointly Estimating Valid Causal Effects of Multiple R-related Events. Approved for filing with the US Patent Office.

[20] Vineet Chaoji, Pranav Garg. Artificial Intelligence System for Network Traffic Flow-based Detection of Service Usage Policy Violations. Approved for filing with the US Patent Office.

[21] Saurabh Sohoney, Vineet Chaoji, Pranav Garg. Automatically Identifying Dynamic Applications. Approved for filing with the US Patent Office.

Software

ICE-DT — A static verification tool for Boogie (an intermediate verification language from Microsoft with translators from several high-level languages like C and C# into Boogie) programs. ICE-DT automatically verifies programs by learning inductive program invariants using custom decision tree learning algorithms. **Winner of the invariant synthesis track at SyGuS-COMP 2015 and SyGuS-COMP 2016**.

ALCHEMIST — A SyGuS (Syntax-Guided Synthesis) solver that learns linear integer arithmetic programs. ALCHEMIST uses computational geometry and decision tree learning algorithms to synthesize non-convex piece-wise linear functions from logical specifications.

P-ASI — An automatic tool for verifying event-driven, asynchronous programs (written in a DSL P) by generating invariants using model-checking combined with natural proof tactics. P-ASI verified the responsiveness of the USB device-driver stack in Microsoft Windows phone, and is being used at Microsoft.

RANDOOP — Automatic unit test generation tool for C/C++ programs that combines feedback-directed random unit testing with symbolic execution to obtain improved code coverage. The tool was developed as part of an internship at NEC Laboratories America.

QDA-LEARN — An automatic tool that uses Angluin's L* algorithm to generate *likely* quantified invariants from dynamic test runs, for programs over arrays and list structures.

Talks

Learning Inductive Invariants using Decision-trees and Implication Counter-examples
Dagstuhl Seminar on "Machine Learning and Formal Methods", Germany. Invited Talk — Aug 2017

Machine Learning Bootcamp
Amazon, Bangalore, India — Jul 2017

Supervised Learning: ML Gurukul Series
Amazon, Bangalore, India — May 2017

Malware Detection in EC2
ML Workshop, core-ML team, Amazon, Seattle, WA. — Apr 2016
AWS Security, Amazon, Herndon, VA. — Jan 2016

Machine Learning based methods for Invariant Synthesis
Invited Tutorial, 27th International Conference on Computer Aided Verification (CAV), San Francisco, CA. — Jul 2015

Learning Invariants for Software Reliability and Security
IBM Research, India. — Apr 2015
Google Research, Mountain View, CA. — Mar 2015
University of California Riverside, CA. — Mar 2015

Black Box Invariant Synthesis

Expeditions in Computer Augmented Program Engineering (ExCAPE) Annual Research Meeting, Berkeley, CA. Mar 2014

ICE: A Robust Learning Framework for Synthesizing Invariants

The Midwest Verification Day (MVD), Chicago, IL. Sep 2013

Learning Universally Quantified Invariants of Linear Data Structures

25th International Conference on Computer Aided Verification (CAV), Saint Petersburg, Russia. Jul 2013

Quantified Data Automata on Skinny Trees: An Abstract Domain for Lists

20th Static Analysis Symposium (SAS), Seattle, WA. Jun 2013

Synthesizing Universally Quantified Invariants of Linear Data Structures

Indian Institute of Science, Bangalore, India. Invited Talk. May 2013

Improving Random Testing via Symbolic Execution

NEC Laboratories, Princeton, NJ. Aug 2011

Rebound: Scalable Checkpointing for Coherent Shared Memory

Illinois-Intel Parallelism Center (I2PC) Distinguished Speaker Series, University of Illinois at Urbana-Champaign, IL. Invited Talk. Sep 2011

38th International Symposium on Computer Architecture (ISCA), San Jose, CA. Jun 2011

NEC Laboratories America, Princeton, NJ. Jun 2011

Compositionality Entails Sequentializability

17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Saarbrücken, Germany. Mar 2011

The Midwest Verification Day (MVD), Iowa City, IA. Sep 2010

Teaching Experience

Instructor

ML Bootcamp and ML Gurukul Series (Amazon) 2017

Designed the module on linear models for supervised learning in the ML Gurukul Series. Also participated in the ML Bootcamp. The target audience for both were software engineers in Amazon.

Teaching Assistant

Software Engineering II CS428 (Undergraduate course, UIUC) Spring 2014

Closely mentored multiple student teams on their course projects, prepared iteration deliverables for them and monitored their progress throughout the semester. Created and graded a final exam, and gave a guest lecture on Software Quality Assurance.

Instructor

ACA Summer School on Data Structures and Algorithms (Workshop, IITK) Summer 2009

Designed the syllabus and gave lectures on data structures and algorithms and on C++ programming. The target audience was engineering undergraduate students, including those from non computer-science backgrounds.

Service

Program Committee Membership

Foundations of Software Technology and Theoretical Computer Science (FSTTCS)	2016
Knowledge Discovery and Data Mining (KDD), Applied Data Science Track	2016
International Symposium on Software Testing and Analysis (ISSTA), Artifact Evaluation	2015

Refereeing and Reviewing

Verification, Model Checking, and Abstract Interpretation (VMCAI)	2017
Programming Language Design and Implementation (PLDI)	2014
Computer Aided Verification (CAV)	2011, 2012, 2013, 2015
Tools and Algorithms for the Construction and Analysis of Systems (TACAS)	2012
Concurrency Theory (CONCUR)	2011

Departmental and University Service

Member of the CS Graduate Academic Council, UIUC. 2014-15
Proposed and reviewed new initiatives to enhance graduate education, organize academic seminars, and enhance communication networks among CS graduate students.

Student representative of the CS Graduate Admissions Committee, UIUC. 2011-12
Evaluated graduate school admission applications and made recommendations to the Admissions Committee.

Programming Experience

Alexa Hackathon, Amazon
Developed a new Alexa skill *Vee*, which is an English language based puzzle game. In the game, the player is asked to quickly speak a sentence that satisfies a few language constraints, such as the sentence starts with x, the sentence has x words, etc., and the game backend checks if the spoken sentence satisfied the contraints.

Software Development Intern
Loop Optimizations in the Gnu C Compiler (GCC), Google Summer of Code Summer 2009
Implemented the loop blocking/strip mining loop optimizations in the GCC compiler. The developed code is part of GCC version 4.6 and onwards.
Mentor: Sebastian Pop, Free Software Foundation

Embedded Software for Industrial Control Systems Summer 2006
Micro-controller programming for designing a PID controller which controlled the fueling system of an IC engine, through a close loop processing of the engine speed error data.