Research and Teaching Statement

ThanhVu Nguyen

1 Highlights

Work Timeline I joined George Mason University (GMU) as a Assistant Professor in Computer Science (CS) in August 2021. Before GMU, I was a Assistant Professor in CS at the University of Nebraska-Lincoln (UNL) from 2016–2021 and a postdoc at the University of Maryland-College Park from 2014–2016. I obtained my Ph.D. in CS from the University of New Mexico-Albuquerque in 2014. My master's and bachelor's degrees were both from Penn State and in CS. In addition to academic experience, I have worked at Lockheed Martin and the Naval Research Laboratory.

Research and Publications My research is in Software Engineering, Programming Languages, and Formal Methods, focusing on correctness analysis of traditional software and AI/ML systems. I have published over 50 refereed papers at top conferences and journals within my field. According to Google Scholar, my work has 3590 citations (h-index 17 and i10-index 24) as of Sep 8, 2023.

Grants and Awards I have received 8 grants: 4 NSF (3 sole-PI, 1 PI), 1 Defense (as Co-PI), 2 industry (sole-PI), and 1 internal (sole-PI) awards. The total amount is **$2.65M** (my share $1.5M, as PI $1.3M). Most of grants were obtained during my time at GMU (total $1.9M, my/GMU share $1.1M, as PI $1.1M). For my research, I have received an *Amazon Research Award* (2023), the NSF *CAREER* award (2023), the NSF CRII Award (2020), an ACM SIGSOFT ICSE *Most Influential Paper* award (2019), an ACM SIGEVO *Impact Award* (2019), and multiple best paper awards.

Students Mentoring I am advising 3 Ph.D. students, all of whom have RAs. One student just passed the Qualifier exam and is from an underrepresented minority (URM) background. Additionally, I have graduated 1 PhD and 2 Masters students from UNL. Currently, I am mentoring 3 undergraduate students and, and in the past, have mentored 10 undergraduates at UNL. In total I have 20 publications with my students, 9 of which are with an undergrad student.

Teaching At GMU I am teaching a new graduate class CS 695/SWE 699 on AI Safety and Assurance in Fall'23 and in have taught Object-Oriented Software Specification and Construction (SWE 619 twice, SWE 419 once). I have also developed an online SWE619 course with Wiley—the course was launched in Spring'23 and taught every semester. At UNL I have taught classes in Software Engineering; Computer Theory/Automata; Software Testing, Verification, and Analysis; and Software Verification seminar. Overall, my classes are well-received by both graduate and undergraduate students, and I was able to recruit many good students from these classes.

Service I regularly serve in technical program committees of major conferences and as reviewers of main journals in my field (e.g., FSE, ASE, ISSTA, TOSEM, TSE). I also have started to take on more leadership role (e.g., Proceeding Chairs of ICSE '21). I have served on NSF panels six times over the past five consecutive years. At GMU I am serving as the program director of the MS degree in Software Engineering and have been in various committees. I also contribute to the

department in other informal ways, e.g., maintaining the CSRankings database for GMU and the About GMU wiki, and organizing the Virtual Open House websites and events.

2 Research and Funding

2.1 Research Directions

My research is in Software Engineering (SE), Formal Methods (FM), Programming Languages (PL), and their interesection. Currently, my group focuses on Safe AI, program specifications/invariants discovery, automatic program repair (APR), and highly-configurable system analysis.

2.1.1 Safe AI and Deep Neural Network Verification (since 2022, new direction)

Deep Neural Networks (DNNs) have been used to tackle a wide-range of real-world problems, including image recognition, autonomous driving, power grid control, fake news detection, drug synthesis and discovery, and even COVID-19 detection and diagnosis. However, similarly to any software system, DNNs can have "bugs" that cause unexpected results when presented with inputs that are different from those in the training data. Additionally, DNNs are susceptible to attack, as slight changes in inputs can result in misclassification. These issues, which have been observed in many DNNs and demonstrated in the real world, naturally raise the question of how DNNs should be tested, validated, and ultimately *verified* to meet relevant robustness and safety standards.

To address this question, my group has developed a new approach called **NeuralSAT** to verify DNNs. The approach integrates abstraction techniques often used in industrial program analyzers and DNN verification with the Conflict Driven Clause Learning algorithm in modern constraint solvers. Abstraction allows the NeuralSAT to approximate and scale to large complex networks while clause learning allows the tool to properly backtrack when making a wrong choice. While the development of NeuralSAT is still in its early stages, we have already seen promising results that suggest NeuralSAT is scalable (can handle real-world DNNs with hundred of millions of parameters) and works with a wide-variety of network models.

Impacts and Awards The preliminary results and prototype of NeuralSAT are the foundation for my recent **NSF CAREER** (2023–2028) and **Amazon Research Award**'23. The NeuralSAT framework is also used in two ongoing NSF proposals involving AI safety. NeuralSAT ranked *fourth* in the recent VNN-COMP'23 (the annual neural network verification competition). It is also very encouraging that while we have not published any results or source code of NeuralSAT, words of mouth about its incredible performance have made it known among the DNN verification community. We were invited and have written a "blog" about the project for SIGBED (Special Interest Group from Embedded Systems). I believe NeuralSAT has good opportunities go beyond academic research and can be adopted in industry.

2.1.2 Highly-Configurable and Build System Analysis (since 2015, during postdoc)

A *highly-configurable* system is a software system with an very large number of configuration options. The main challenge in this line of research is that the number of configurations in real-world systems is so large that it is impossible to analyze all of them.

To tackle this, I have developed iGen and GenTree, a lightweight ML-based, dynamic analysis technique to automatically discover program "interactions" that show how configuration option settings affect a program's functionality, performance, etc. This work uses a "guess and check" approach that first guesses an interaction and then checks if the guess is correct by running the program on configurations satisfying the guess. Experimental results show we often need to sample only a small number of configurations to discover very accurate interactions of the systems.

Recently, I have worked with Meta on **Cybolic**, a project that integrates dynamic analysis with symbolic execution techniques to capture *build conditions*. These conditions are interactions over build options and help developers determine how software are configured and built.

Impacts and Awards In addition to multiple publications at top venues, I have received an **NSF CRII** award on combine static and dynamic techniques to analyze highly-configurable systems such as the Linux kernels with an extremely large number of 2^{13000} configurations. I have also received a UNL Seed award in 2020 on this line of research. Moreover, the recent Cybolic work, while is still in progress, has already been used internally at Meta Whatsapp to find build failures. I have received a Facebook grant to improve Cybolic and a recent **NSF Formal Methods in the Field (FMiT) Track 2** (2023–2025) award to make Cybolic more scalable and usable in the industry.

2.1.3 Invariant Generation and Program Repair (since 2008, foundational research)

Specification Discovery/Invariant Generation Software bugs are caused by programs violating required *specifications*. However, software developers often perceive a "specification burden" and do not write down desired or required specifications. Thus, they cannot exploit powerful verification techniques and tools developed to check for specification violations.

To address this problem, I have developed the **DIG** framework to automatically discover program specifications, or more technically, *program invariants*. DIG uses a dynamic or *data-driven* analysis to learn useful program invariants in programs. To support expressive invariants, e.g., nonlinear polynomial properties often required scientific or cyber-physical systems, DIG interprets nonlinear polynomial formulae as convex geometric objects in high-dimensional space, such as hyperplanes and polyhedra. To avoid spurious results, DIG uses the guess-and-check approach to infer invariants: it encodes the semantics of programs as symbolic constraints and use them to generate candidate invariants and also to check or refute, and iteratively refine, those invariants.

DIG has been used as the foundation to analyze many important classes of invariants, e.g., nonlinear numerical relations, array and heap properties, termination and temporal properties, algebraic specifications for modelling library calls, and interactions among configuration options. DIG's invariants are also used for security analysis, e.g., to check the correctness of AES (Advanced Encryption System) and detect potential side-channel complexity attacks.

Impacts and Awards The original DIG work received the Distinguish Paper Award at ICSE'12 and since then my invariant work has produced more than 15 refereed, top-tier conference and journal papers. GammaTech, a company that developes program analysis tools for the industry, has integrated DIG in their Mnemosyne project, raising opportunities for additional industrial adoptions. DIG's nonlinear benchmarks are included in SV-COMP (the annual international competition on software verification). In summary, program analysis through dynamic invariant generation is the "bread and butter" of my work that I can always rely on for research and funding. For example, my **NSF Medium grant (2021–2024)** uses DIG to infer and repair temporal properties.

Automatic Program Repair (APR) Finding program violations or bugs is only half of the debugging process. We also need to fix the bugs! My early APR work on GenProg uses a genetic algorithm to satisfy multiple program repair objectives, e.g., it must pass a given testsuite, is similar to the original buggy program, and (preferably) easy to understand. GenProg is the first APR work to show that computers can automatically repair real-world large programs (million lines of code).

I also have developed theoretical work showing the equivalence of the seemingly different problems of program repair/synthesis and reachability/verification. I also have extended traditional symbolic execution techniques to repair corrupted data structures. Recently, my group have applied APR to domain-specific languages and have developed powerful techniques to localize, analyze, and fix errors in specifications and models written in the Alloy language.

Impacts and Awards GenProg has received multiple awards, e.g., best papers at ICSE, GECCO, ICST, and the ACM SIGEVO "Humie" gold medal. In 2019, GenProg was recognized with a **10-year IEEE/ACM Most Influential Paper award at ICSE** and a **10-year ACM SIGEVO Most Impact Award at GECCO** for showing that real-world programs can be repaired automatically and creating the now widely popular APR subfield in Software Engineering. Our APR work also has been used in the industry, including GammaTech and MIT Lincoln Lab. In particular, Facebook has adopted ideas in GenProg to find locate and repair bugs in their Android and iOS apps.

In total, my APR research has been cited more than 2000 times as of June'23. This line of research has been funded by the mentioned NSF Medium award (2021–2024) and an **Army Research Office (ARO)** grant (2018–2021). More recently, we have also patented our work on domain specific APR and are applying external funding for this line of work.

2.2 Funding

Tab. 1: Summary of Funded Grants. I also receive supplementary REU funding each year (not shown here).

	Award	Role	Dates	Total	My Share
GMU	NSF FMIT Track 2	Sole-PI	2023–2025	$97,242	$97,242
	Amz Research Award	Sole-PI	2023	$50,000	$50,000
	NSF CAREER	Sole-PI	2023–2028	$510,509	$510,509
	Facebook	Sole-PI	2021	$30,000	$30,000
	NSF CISE Collab. Medium	PI	2021–2024	$1,199,871	$399,879
UNL	UNL Seed Award	Sole-PI	2021	$10,000	$10,000
	NSF CRII	Sole-PI	2020–2022	$174,975	$174,975
	Army Research Office (ARO)	Co-PI	2019–2021	$549,990	$158,850

Tab. 1 gives an overview of my funding. In total, I have been awarded 8 grants, including 4 NSF, 1 Defense (ARO), 2 industry (Facebook and Amazon), and 1 internal (UNL) awards. I am either the sole-PI or PI for all grants except the ARO one in which I am the Co-PI. It is worth noting that the majority of these awards were obtained during my time at GMU.

In total, I have received $\approx 2.7M$ in funding, of which my share is $\approx 1.5M$ ($\approx 1.3M$ as PI). Among grants obtained at GMU, the total is $\approx 1.9M$, of which my and GMU share is $1.1M$ (all as sole PI or PI). More detailed information about these awards and their relevance to my research are described in my CV and the Impacts and Awards subsections in §2.1 of this document.

2.3 Future Plan

My research spans over several fields, including SE, FM, PL, and most recently AI/ML (e.g., verifying AI safety). It appears that I develop a new major research interest and direction every seven years. While this has its disadvantage, as I may not be as well-known in a specific field, it has the advantage that I can effectively apply my research to new areas and problems (e.g., leveraging program analyses to tackle AI challenges).

My recent interest in formal AI analysis has been fruitful and excited (e.g., the power and promise of NeuralSAT have made me ponder about commercial opportunities). Given my 7-year trend, I anticipate to dedicate 5-6 more years on this topic, and I look forward to making more impacts in there. Moreover, I will adapt advancements developed for AI verification to traditional software engineering problems. For example, I can apply scalable verification and analysis techniques in NeuralSAT to traditional software systems, especially highly-configurable ones with large search spaces. In summary, I will continue to focus on my current research topics and also adapt them to new areas and problems.

3 Advising and Mentoring

3.1 Students

Graduate Students Currently I am supporting three PhD students with RAs (including summer). Didier Ishimwe is a 4th-year minority student who transferred to GMU with me from UNL. He works on complexity analysis using invariant generation and recently advanced to PhD candidacy. He plans to defend by Fall'24. Hai Duong is a 1st-year PhD student whom I recruited from Vietnam. He is working on AI safety and verification and is the main developer of the mentioned NeuralSAT project. Finally, Linhan Li is a 2nd-year Ph.D. student who transferred from UNL with me. He is assisting Hai with NeuralSAT.

Graduated I have graduated a Ph.D. student, Guolong Zheng, at UNL in Spring'22. Guolong has published 9 papers (5 full papers, 4 short papers) with me and is currently working at A10Networks. I have also graduated two Masters students at UNL: Alexey Malyshev, a Fullbright fellow, graduated in Spring'22 and is currently working at Oracle, and Mitch Girrald, whom I co-advise with Matt Dwyer at UVA, graduated from UNL in Fall'22 and continued to do PhD at UVA.

Undergraduate Students I enjoy involving and mentoring undergraduate and minority students in my research. Currently, I am mentoring three undergraduate students, two of whom are at GMU, and one female student at Boston University. I am also mentoring two undergraduate volunteers from Vietnam, who are seeking research experience to apply to PhD programs in the US.

At UNL, I worked with over 10 undergraduate research assistants. Among them, a majority were first-generation students, two were female, two were freshmen, and six had received departmental or university awards for their work with me. My current PhD student Linhan Li was my former undergraduate student at UNL. Another student, KimHao, has worked with me since his freshman year, and has produced *nine* publications with me (from top venues including ICSE, OOPSLA, TSE). KimHao's research has earned him two summer internships at Facebook and numerous awards, including UNL Outstanding Senior award in Spring'23 and the Outstanding Undergraduate Research award in Spring'21. He recently graduated with a BS from UNL and is working at Jump

Trading, where he has a remarkable $500,000 annual salary (the main reason why I cannot convince him to do PhD with me).

3.2 Future Plan

I am committed to creating a welcoming and diverse environment within my research group, where students from all backgrounds feel valued and supported. I am actively recruiting PhD students, particularly those from underrepresented minority groups and countries with limited representation in US PhD programs. For example, Didier and I are working to introduce research opportunities to students from Rwanda, which is Didier's home country. Hai and I are also working together to mentor and recruit talented students from top universities in Vietnam and also to raise awareness of GMU in the country. I will continue placing a strong emphasis on creating opportunities for undergraduate students. I consistently include budget to support undergraduate students in my proposals, and I have successfully obtained NSF REU funding to increase undergraduate research in my group.

4 Teaching

4.1 Courses Taught

At GMU I have taught the required graduate Software Engineering course SWE619: OO Software Specification and Construction (twice, $\approx$ 32 students per class) and its undergraduate version SWE419 (once, 16 students). For this Fall'23, I am offering a new graduate special topic course CS695/SWE699: AI safety and assurance (35 students registered). In Fall'22 I have collaborated with with Wiley publishing to design and develop an *online version* of SWE619. The course was launched in Spring'23 and has been taught by various faculty members every semester.

At UNL I have taught Automata (twice, $\approx$40 undergraduate students), Software Engineering 2 (once, $\approx$30 undergraduate students), Software Verification seminar (SV, 5 times, $\approx$8 graduate students), Software Testing, Verification, and Analysis (TVA, twice, $\approx$70 undergraduate and graduate students), and Compilers (twice, $\approx$20 undergraduate and graduate students).

4.2 Philosophy

My *teaching goal* is to have students finish the class understanding the subject and being able to apply it to real life. I use several *strategies* to achieve this:

First, I put extra efforts in creating **motivating, real-world examples** to help students appreciate and remember difficult CS concepts. For example, for *fuzz testing* in 619/419/TVA, I tell the story of how Barton Miller conceived the idea in 1988 when he noticed that thunderstorm generates noise on the telephone line, which in turn creates bad inputs to remote Unix commands, causing them to crash. I also use this real-bug report in LinuxMint in 2020 to motivate fuzzing *"A few weeks ago, my kids wanted to hack my Linux desktop, so they typed and clicked everywhere, while I was standing behind them looking at them play... when the screensaver core dumped and they actually hacked their way in! wow, those little hackers..."*.

To help students understand concepts they have learned, I design my **programming assignments (PAs)** to be challenging yet interesting. For example, my TVA students enjoyed using their *genetic algorithm* (GA) implementations to crack long, secret passphrases. In TVA 20, I provide

the run time of my own GA reference implementation so that students know what to expect for the assignment. A student mentioned that she became a bit obsessed with tweaking and optimizing the parameters of her GA to outperform mine. My upcoming AI safety and assurance class will have a PA requiring students to implement symbolic testing to find *adversarial inputs* that can fool deep neural networks. These PAs help students become familiar with existing powerful verification and analysis tools and know how to adapt and apply them to their work.

I introduce students to existing **program analysis tools**, such as Microsoft Z3, a constraint solver that is invoked *billions of times daily* at Amazon AWS for privacy and security checking, and INFER, a static analyzer developed by Facebook to check for problems in Instagram and Messenger apps. I also talk about my research and demonstrate how DIG can automatically find useful and missing specifications (for my upcoming AI safety class I will use NEURALSAT as the verification framework). Students expressed that they were not aware of these tools but were amazed by their capabilities, and they plan to apply these tools in their development environment.

Finally, I continuously refine my class materials based on **feedback** from students and peers. When moving to an online format during Spring'20 due to COVID, students in Compilers mentioned that it was difficult to understand lectures without seeing diagrams and drawings (e.g., of how garbage collection works on memory layouts). One student showed me how to use a tablet to create an interactive "whiteboard" experience through Zoom. Immediately after that, I started using both a tablet for drawing figures and the *VSCode Preview* feature for demonstrating code and diagrams. The VSCode method allows the student to see live contents "twice", as I am writing it in plain text and as it is rendered and displayed in the preview pane. Students have consistently expressed positive feedback that this teaching method facilitates their learning. Notably, Professor Lisong Su from UNL evaluated my class and was impressed with the approach. In fact, he provided a detailed description of the method, accompanied by screenshots to illustrate its benefits.

4.3 Future Plan

I will continue to improve my teaching and class materials through the strategies and feedback described. In recent years, in addition to class evaluations (which many students ignored), I have included questions directly on final exams such as *"What do you like and dislike about the course?"* and *"Do you have any suggestions for next year's students?"*. I integrate students' responses to these questions to enhance class materials and use their feedback as "words of wisdom" for future students taking the course.

One major effort I plan to undertake in my upcoming AI safety and DNN verification class is transforming the course lectures and programming assignments into an **open-source interactive book**. The book will have a similar Jupyter/Python-based format as the Fuzzing Book for software testing. The dynamic format of the book will make learning more enjoyable, as students can edit code and experiment with its effects directly on the web browser. Students will also have the opportunity to contribute by extending the Markdown text or Python code through GitHub. Many of my collaborators also teach AI safety, making them potential users of the book. This idea is a part of the teaching component in my NSF CAREER proposal and has received enthusiastic support from reviewers. I believe this book will be a valuable resource for students and researchers interested in AI safety and verification.

5 Service

5.1 Research Community

I regularly serve as a technical program committee member and reviewer for many top-tier conferences and journals in my field, including FSE, ASE, ISSTA, PLDI, TSE, and TOSEM. Additionally, I served as the proceedings co-chair of ICSE'21, where I oversaw the publication process of ICSE, the top conference in Software Engineering, as well as all of its research tracks, workshops, and co-located events. This was the most challenging service role I have taken on, as it required coordinating with hundreds of authors, chairs, and editors to ensure timely publication of the proceedings.

I have also served on *seven* consecutive NSF panels from 2019 to 2023. This experience has allowed me to stay updated on the latest research trends and gain expertise in writing winning proposals (my first panel was in 2019, and my first NSF grant followed in 2020). Detailed information about my service roles in the research community can be found in my CV.

5.2 Department

I greatly enjoy the faculty interaction and atmosphere at GMU CS and have been actively involved in serving the department. Currently, I am the program director of the MS degree in Software Engineering. I have also served in the MS/PhD Admission, Executive, and Web communities. In the Admission committee, I take the lead in organizing Virtual Open House events and maintaining websites to recruit admitted students. In the Executive committee, I advocate for increasing benefits for PhD students, especially international students. In the Web committee, I lead the effort in migrating the CS webserver to the university Masonry system, and in general strive to be responsive and ensure that the department website is up-to-date and informative. Particularly, I maintain an internal Award database to track faculty and students' accomplishments and update our CS Award website accordingly.

Outside of the assigned committees, I contribute to the department in several significant ways. I maintain the CSRankings database for GMU, which is used by many faculty candidates and prospective students to evaluate the department's research performance. I also create the About GMU-CS wiki, which highlights the strengths and uniqueness of GMU and its CS department and is used by many faculty for recruitment purposes. One faculty candidate sent me a thank-you note about it, saying, *"In looking up your email address, I came across your About GMU CS page... a very fun and informative read!"*. I also actively participate in faculty recruitment by hosting and meeting candidates. For example, this past Spring'23 semester, I met with 19 candidates in total.

5.3 Future Plan

I will continue to serve my research community, focusing more on organizational and leadership roles (e.g., hosting an SE conference in the DC area). I will also continue my service to GMU CS, particularly in improving and expanding the MS SWE program and student recruitment. I strongly believe that while GMU CS faculty has been doing a great job in research (as evident from CSRankings), we need to attract better PhD students to compete with other top CS programs. To this end, I am collaborating with multiple faculty and students to develop a comprehensive guideline that demystifies the PhD admission process in the US. My goal is to encourage more applications from top domestic and international students, especially those from smaller countries such as Vietnam and Rwanda.