Siddharth Bhat

Skills

I'm an expert in formal verification, optimizing compilers, and AI4maths.

Functional Programming & Formal Verification: I have been working on functional programming since I was a teenager, and I was hired by Tweag, a research organization to implement a new Haskell to WebAssembly compiler (Asterius), and is in the top two contributors to Asterius, which was eventually merged into the Glasgow Haskell Compiler.

I am in the top 20 contributors to the Lean theorem prover overall, and one the top two contributors of the Lean bitvector theory. I'm an author of the Lean4 metaprogramming book and has written a large part of the necessary theorems for proving
Lean's bitblaster (`bv_decide`) correct.

Compilers & HPC: I have deep experience on high performance computing and loop optimization: Polly is a research loop optimizer for the LLVM project, a production-grade compiler optimization framework. Polly was the first to create a framework for loop analysis and optimisation that brought mathematical research ideas in polyhedral compilation to real-world ideas. Technical successors of Polly that use many of its innovations are used to accelerate large machine learning models today. I have contributed 121 patches to Polly, making me among the top three contributors to the Polly loop optimizer. Thanks to my work, the Polly loop optimizer gained the ability to perform high-performance GPU code generation for realistic climate models, including the dynamical core of COSMO, Switzerland's official climate model. I mentored students working on the Polly project as a Google Summer of Code mentor in 2016.

AI for Maths: My long term vision is to build scalable systems for mathematical theorem proving, as the bitter lesson teaches us that only search scales. As an intern at Microsoft Research (2023), I worked on RL approaches for proof repair in the F* proof assistant ("Towards neural synthesis for SMT-assisted proof-oriented programming" @ ICSE 25, best paper).

Education

PhD **University of Cambridge**.
(2024 - Ongoing)

PhD **University of Edinburgh (*moved to Cambridge*)**.
(2022 - 2024)

Master by Research **International Institute of Information Technology Hyderabad India**.
(2020 - 2021)

Undergraduate **International Institute of Information Technology Hyderabad India**.
2015 - 2020

Publications

Certified Decision Procedures for Width-Independent Bitvector Predicates: **Siddharth Bhat**, Léo Stefanesco, Chris Hughes, Tobias Grosser. OOPSLA 2025

Interactive Bit Vector Reasoning using Verified Bitblasting: Henrik Böving, **Siddharth Bhat**, Alex Keizer, Luisa Cicolini, Leon Frenot, Abdalrhman Mohamed, Léo Stefanesco, Harun Khan, Josh Clune, Clark Barrett, Tobias Grosser. OOPSLA 2025

X03, 64 Storey's Way, Churchill College – CB30DS – Cambridge, United Kingdom
✉ *siddu.druid@gmail.com* • *pixel-druid.com*

Verifying Peephole Rewriting in SSA Compiler IRs: **Siddharth Bhat**, Alex Keizer, Chris Hughes, Andres Goens, Tobias Grosser. ITP 2024

Verifying Wu's Method can Boost Symbolic AI to Rival Silver Medalists and AlphaGeometry to Outperform Gold Medalists at IMO Geometry: Shiven Sinha, Ameya Prabhu, Ponnurangam Kumaraguru, **Siddharth Bhat**, Matthias Bethge. NeurIPS 2024 Workshop MATH-AI

Verifying Peephole Rewriting in SSA Compiler IRs: **Siddharth Bhat**, Alex Keizer, Chris Hughes, Andres Goens, Tobias Grosser. ITP 2024

Towards Neural Synthesis for SMT-Assisted Proof-Oriented Programming: Saikat Chakraborty, Gabriel Ebner, **Siddharth Bhat**, Sarah Fakhoury, Sakina Fatima, Shuvendu Lahiri, Nikhil Swamy. ICSE 2024

Rewriting Optimization Problems into Disciplined Convex Programming Form: Ramon Fernandez Mir, **Siddharth Bhat**, Andres Goens, Tobias Grosser. CICM 2024

Guided Equality Saturation: Thomas Koehler, Andres Goens, **Siddharth Bhat**, Tobias Grosser, Phil Trinder, Michel Steuwer. POPL 2024

Lambda the Ultimate SSA: **Siddharth Bhat**, Tobias Grosser. CGO 2022

QSSA: An SSA based IR for Quantum Computing: Anurudh Peduri, **Siddharth Bhat**, Tobias Grosser. CC 2021

Optimizing Geometric Multigrid Computation using a DSL Approach: Vinay Vasista, Kumudha KN, **Siddharth Bhat**, Uday Bondhugula. Supercomputing (SC), Nov 2017

Word Embeddings as Tuples of Feature Probabilities: **Siddharth Bhat**, Alok Debnath, Souvik Banerjee, Manish Shrivastava Representation Learning for NLP, 2020

Internship Experience

Sep-Nov '24 **Amazon Web Services, Automated Reasoning Group**, *Austin*.
Deciding memory (non)interference in `lnsym`, a Lean-based ARM symbolic simulator

Jul-Sep '23 **Microsoft Research**, *Redmond*.
Retrieval Augmented theorem proving for the Fstar proof assistant.

July 1-10 '23 **Adjoint School**, *Glasgow*.
Researched Markov categories and their relationship to probabilistic programming.

May-Jul '19 **Intern at Tweag.io**, *Paris, France*.
Re-implemented portions of GHC(Glasgow Haskell Compiler) runtime for `Asterius` (link), a Haskell to WebAssembly compiler. Involved Haskell, C, and WebAssembly.

Summer 2018 **Visiting research intern at ETH Zurich**, *Zurich, Switzerland*.
Investigating formal verification of polyhedral compilation. `PolyIR (Link)` is a formal specification of polyhedral programs.

Summer 2018 **GSoC mentor, Polly Labs**.
Mentoring a project to enable Polly's loop optimisations into Chapel.

Mar-Dec '17 **ETH Zurich, Research Intern at SPCL**, *Zurich, Switzerland*.
Worked on Polly, a polyhedral loop optimizer for LLVM.

May-Jul '16 **Research Intern**, *IISC Bangalore*, Bangalore.
Worked on PolyMage, DSL compiler for optimising loop transforms. Contributed to ISL and PLUTO. Implemented tiling patterns, optimised PolyMage for stencils.

Summer 2016 **Selected for GSoC 2016**, *Google*.
Binding SymEngine, a symbolic math library to Haskell. Had to drop this to intern at IISc, Bangalore. Still maintain the library (symengine.hs)

Summer 2015 **GSoC 2015**, *Google*.
Worked on VisPy, a pure Python graphics library which uses OpenGL internally for performace. Successfully completed.

««««< HEAD

Skills & Interests

I'm an expert in formal verification, the implementation of functional programming languages, compilers, and AI for mathematics. **Formal Verification:** I am in the top 20 contributors to the Lean theorem prover overall, and one the top two contributors of the Lean bitvector theory. I'm an author of the Lean4 metaprogramming book and has written a large part of the necessary theorems for proving Lean's bitblaster (`bv_decide`) correct.
Compilers & HPC: I have deep experience on high performance computing and loop optimization: Polly is the polyhedral loop optimizer for the LLVM project, which is a production-grade compiler optimization framework that powers many compilers, including clang and the official Rust compiler. Polly was the first to create a framework for loop analysis and optimisation that brought mathematical research ideas in polyhedral compilation to real-world ideas. Technical successors of Polly that use many of its innovations are used to accelerate large machine learning models today. I have 121 commits in Polly, making me the #3 contributor to the Polly loop optimizer. Thanks to my work, the Polly loop optimizer gained the ability to perform high-performance GPU code generation for realistic climate models, including the dynamical core of COSMO, Switzerland's official climate model maintained by the MeteoSwiss, Switzerland's Federal Office of Meteorology and Climatology. I then continued to oversee the Polly project and has even mentored students working on the Polly project as a Google Summer of Code mentor in 2016.
Functional Programming: I've been a long-time haskell user (I started as a teenager on the `#haskell` IRC channel) and I was an intern at Tweag, a research organization, in order to implement a new Haskell to WebAssembly compiler (Asterius). I am the #2 to Asterius, where I worked on re-implementing a Haskell-style generational GC on top of the WASM GC. My contributions were merged into Asterius, which was eventually merged upstream into the Glasgow Haskell Compiler.
AI for Maths: My long term vision is to build scalable systems for mathematical theorem proving, as the bitter lesson teaches us that only search scales. Toward this, during an internship at Microsoft Research in 2023, I worked on reinforcement-learning based approaches for correcting incorrect proofs in the F* proof assistant, which was published as "Towards neural synthesis for SMT-assisted proof-oriented programming"@ICSE 2025, with a best paper award. Similarly, I have helped established stronger baselines for AI based geometric theorem proving, by showing that symbolic methods, when used well, go head-to-head with state of the art AI based systems.

Talks & Presentations

Euro LLVM Dev 2025: How to trust your peephole rewrites: automatically verifying them for arbitrary width!. **US LLVM Dev 2024:** lean-mlir: A workbench for formally verifying peephole optimizations in MLIR. **US LLVM Dev 2023:** (Correctly) Extending dominance to MLIR Regions. **US LLVM Dev 2023:** MLIR Side Effect Modelling. **Euro LLVM Dev 2022:** MLIR for Functional Programming. **FPIndia 2021:** Equality Saturation. **Functiona Conf 2019:** Monad-bayes: Probabilistic programming in Haskell.

Awards

https://www.renaissancephilanthropy.org/mathbench-towards-evaluating-natural-language-proofs
One of 30 research groups that was awarded out of over 280 applicants.