Siddharth Bhat

Skills

I'm an expert in formal verification, compilers for imperative and functional programming languages, and AI4Maths.

Formal Verification of Compiler IRs: I am in the top 20 contributors to the Lean theorem prover overall, and one of the top two contributors of the Lean bitvector theory. I'm an author of the Lean4 metaprogramming book and have written a large part of the necessary theorems for proving Lean's bitblaster (`bv_decide`) correct. I've led the `lean-mlir` project, which aims to provide a framework for formally verifying optimizations in MLIR. I have given over five talks at the LLVM developer meeting on formal semantics for LLVM, MLIR, and verifying optimizations in MLIR.

Loop Optimization & HPC: I have a lot of experience in loop optimization, via working on Polly, LLVM's polyhedral loop optimizer. 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 MeteoSwiss, Switzerland's Federal Office of Meteorology and Climatology. I then continued to oversee the Polly project and have mentored students working on the Polly project (Google Summer of Code mentor, 2016)

Compilers for Functional Languages: I am the #2 contributor to Asterius a Haskell → WebAssembly compiler, 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. I am also the primary author of the Lean programming language's LLVM backend, and have written papers on designing MLIR based IRs for quantum programming and Lean's untyped compiler IR.

AI for Maths: I have expertise in integrating the bitter lesson into interactive theorem proving contexts, and believe that neurosymbolic approaches such as AlphaProof will get us to much better proof automation than what we have at present. 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 ("Towards neural synthesis for SMT-assisted proof-oriented programming"@ICSE 2025, best paper). Similarly, I have helped establish 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.

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

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

.

Publications

Certified Decision Procedures for Width-Independent Bitvector Predicates: **Siddharth Bhat (1st)**, 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

Verifying Peephole Rewriting in SSA Compiler IRs: **Siddharth Bhat (1st)**, 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

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 (1st)**, 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 (1st)**, 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 performance. Successfully completed.

Open Source Contributions

Lean4 Co-developed the bitblasting theory for Lean's bitvector automation, wrote the LLVM backend for the compiler.

Rocq Submitted issues, bug-fixes, helped improve developer documentation.

VE-LLVM Collaboration with VE-LLVM, a formal semantics of the LLVM compiler toolchain in Coq

Polly Implementing support for Fortran, added unified memory abilities to the CUDA backend within Polly, a polyhedral loop optimiser for LLVM. (Link to commits)

Symengine.hs GSoC 2016. Haskell bindings to SymEngine, a C++ symbolic manipulation library.

VisPy GSoC 2015. Rewrote scene graph for performance. Added visuals, high level API for easy use of plotting. Implemented auto-resizing with **Cassowary**, a linear optimisation library.

Rust Contributed to the Rust compiler and ecosystem. Found compiler errors, fixed libraries. Was part of *Piston*, group of Rust programmers that experimented with writing game engines.

Haskell Contributed to the Haskell ecosystem. Reported and fixed bugs in *stack*, *stackage*, *diagrams*, *GHC*, etc. (Link to GHC commits).

PLUTO Source to Source C optimiser for loop nests. Improved the PLUTO API that had gone out of sync with master. Discovered bugs in PLUTO for diamond tiling transforms

PolyMage DSL compiler than generates C code. Uses **Polyhedral Compilation** Extended the compiler to add stencils, time iterated-stencils.

PPSSPP C++ open source PSP emulator. Wrote most of the touch handling code. Implemented atomic locks for audio performance.

My Projects

Lean-MLIR Formal semantics for the MLIR compiler framework, defined within the Lean4 proof assistant.

lz An MLIR based compiler backend for the Lean4 proof assistant.

Lean4 Metaprogramming Book A textbook on metaprogramming in Lean4. I wrote the chapters on tactics and metaprogramming for embedded DSLs.

Lean-to Jupyter kernel for the Lean4 proof assistant.

Simplexhc A custom compiler for a subset of Haskell. The goal is to try and apply *polyhedral compilation* ideas to compile a lazy, pure, functional programming language. with LLVM as a backend. Has **64 stars** on github.

Sublime Bookmarks	A plugin for sublime text to quickly jump between pieces of your codebase. **26k downloads** and counting.
Cellular Automata	A collection of Cellular Automata written in Haskell. Uses **Comonads** for abstraction. **130 stars** on Github.
Teleport	A simple tool to switch between projects written in Haskell. Shows how to write "real world Haskell". Published as a **Literal Haskell tutorial**. **90 stars** on github
TIMi	A visual interpreter of the **template instantiation machine** to understand evaluation of lazy functional languages. **51** stars on github.

Miscellaneous

Barvinok	Talk at ETH Zurich: Slides describing the barvinok algorithm to count lattice points in polyhedra
FunctionalConf '19	Talk on implementing embedded probabilistic programming languages in Haskell (Slides)
Haskell Exchange 2020	Talk on optimizing `smallpt-hs` (a port of a raytracer to haskell) to beat `C++` performance (Slides)
FPIndia	Talk on egg: fast and extensible equality saturation. (Slides)
Theory seminar, winter '19	Talk on impossibility of compass-straightedge constructions using field theory.
math.se	Answer on **math.stackexchange. 8312 reputation, top 4% overall**. Abstract algebra and differential/algebraic geometry.

.

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 awarded out of over 280 applicants.