

Siddharth Bhat

siddharth.bhat@cl.cam.ac.uk · +44 7442 785227 · pixel-druid.com · Churchill College, Cambridge, UK
github.com/bollu · linkedin.com/in/siddharth-bhat · Google Scholar · UK Global Talent visa (Exceptional Promise)

SUMMARY

ML Performance Engineer at **XTX Markets**; PhD researcher (Cambridge) working on formally verified decision procedures and mechanical theorem proving in Lean, with a focus on compiler verification. Co-developer of Lean's `bv_decide` bitvector solver (**5th at SMT-COMP 2025**, `QF_BV`) and author of sound-and-complete, machine-checked decision procedures for parametric bitvectors. Also a decade of performance-critical C++ inside production compilers and low-level runtimes: #3 contributor to **Polly** (LLVM's polyhedral loop optimizer) and primary author of **Lean's LLVM backend**, lowering an IR through instruction selection and code generation down to native machine code. Comfortable reasoning about the machine underneath the language — cache behaviour, memory models, SIMD, assembly — and optimizing hot code to that level.

Languages: Lean, C++, Haskell, Rust, Rocq/Coq, Python (PyTorch), CUDA, WebAssembly, C

Infrastructure: LLVM, MLIR, polyhedral compilation, SMT solvers, GPU code generation

EXPERIENCE

2026–present XTX Markets, London — *ML Performance Engineer*. Performance engineering for machine-learning systems.

Sep–Nov '24 Amazon Web Services, Automated Reasoning Group, Austin — *Research Intern*. Deciding memory non-interference in `lmsym`, a Lean-based ARM symbolic simulator; large-scale symbolic execution and its performance.

Jul–Sep '23 Microsoft Research, Redmond — *Research Intern*. Retrieval-augmented proof synthesis for the `F*` proof assistant (ICSE 2025 best paper).

May–Jul '19 Tweag.io, Paris — *Research Intern*. Re-implemented portions of the GHC runtime for `Asterius` (Haskell, C, WASM).

Summer '18 ETH Zurich — *Research Intern*. Formal verification of polyhedral compilation; built `PolyIR`, a formal specification of polyhedral programs.

Mar–Dec '17 ETH Zurich, Scalable Parallel Computing — *Research Intern*. CUDA code generation in `Polly` for Fortran loop nests.

May–Jul '16 IISc Bangalore — *Research Intern*. `PolyMage` DSL compiler; tiling patterns and stencil optimization; contributions to `ISL` and `PLUTO`.

PUBLICATIONS

1. Towards Mechanized Floating Point Bitblasting. *Siddharth Bhat*, Abdalrhman Mohamed, Tobias Grosser. *POPL 2027 (under review)*.
2. Sound and Complete Solving for Multi-Width Parametric Bitvectors via Principled Reductions. *Siddharth Bhat*, Léo Stefanescu, Tobias Grosser. *OOPSLA 2026 (accepted with minor revisions)*.
3. Faults in Our Formal Benchmarking: Dataset Defects and Evaluation Failures in Lean Theorem Proving. Pawan Sasanka Ammanamanchi, *Siddharth Bhat*, Stella Biderman. *ICML 2026*.
4. Certified Decision Procedures for Width-Independent Bitvector Predicates. *Siddharth Bhat*, Léo Stefanescu, Chris Hughes, Tobias Grosser. *OOPSLA 2025*.
5. Interactive Bit Vector Reasoning using Verified Bitblasting. Henrik Böving, *Siddharth Bhat*, Alex Keizer, Luisa Cicolini, Abdalrhman Mohamed, Léo Stefanescu, Josh Clune, Clark Barrett, Tobias Grosser. *OOPSLA 2025*.
6. Towards Neural Synthesis for SMT-Assisted Proof-Oriented Programming. Saikat Chakraborty, Gabriel Ebner, *Siddharth Bhat*, Sarah Fakhoury, Sakina Fatima, Shuvendu Lahiri, Nikhil Swamy. *ICSE 2025 (best paper)*.
7. 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*.
8. Verifying Peephole Rewriting in SSA Compiler IRs. *Siddharth Bhat*, Alex Keizer, Chris Hughes, Andres Goens, Tobias Grosser. *ITP 2024*.
9. Guided Equality Saturation. Thomas Koehler, Andres Goens, *Siddharth Bhat*, Tobias Grosser, Phil Trinder, Michel Steuwer. *POPL 2024*.
10. Lambda the Ultimate SSA. *Siddharth Bhat*, Tobias Grosser. *CGO 2022*.

Full list (14 papers) on Google Scholar.

SOFTWARE

Lean 4: co-developed the verified bitblasting theory behind `bv_decide`; primary author of Lean's LLVM backend; top-20 contributor overall, #2 to the bitvector theory.

Polly (LLVM): #3 contributor to LLVM's polyhedral loop optimizer: 121 commits, 6000 LoC, Fortran support and unified-memory CUDA code generation.

Asterius / GHC: #2 contributor to this Haskell→WebAssembly compiler; reimplemented the GHC runtime on WASM, later merged into GHC proper.

lean-mlir: formal semantics for MLIR in Lean 4, with verified peephole rewriting (ITP 2024).

Lean 4 Metaprogramming Book: authored the chapters on tactics and on metaprogramming for embedded DSLs.

TALKS AT LLVM DEVELOPER MEETINGS

How to trust your peephole rewrites: automatically verifying them for arbitrary width (EuroLLVM '25) · `lean-mlir`: verifying peephole optimizations in MLIR (LLVM Dev '24) · (Correctly) extending dominance to MLIR regions (LLVM Dev '23) · MLIR side-effect modelling (LLVM Dev '23) · MLIR for functional programming (EuroLLVM '22).

EDUCATION

PhD, Computer Science, University of Cambridge; transferred from University of Edinburgh

2022–2026 (writing up)

MS / BTech, Computer Science, IIT Hyderabad, India

2015–2021

Awards: Two Renaissance Philanthropy AI for Maths grants (1 of 30 from 280+ applicants; tactic synthesis, with EleutherAI) · Best paper, ICSE 2025 · 5th place, SMT-COMP 2025 (QF_BV) · Google Summer of Code student '15, mentor '16 and '18