Cover Letter

Arijit Shaw
Ph.D. candidate
Chennai Mathematical Institute

My research focuses on *developing*, *understanding*, and *applying* a broad class of automated reasoning systems to solve problems in science and engineering. Automated reasoning has significantly strengthened computer science by enabling end-to-end trustworthy systems and solutions. This progress is particularly important in an era where large language models are transforming science and society, yet still lack the ability to reliably reason and produce formally trustworthy results. As a researcher at

Modern automated reasoning systems, however, exhibit several important gaps. (i) Existing reasoning tools remain limited when dealing with systems that incorporate randomness or operate under uncertainty. Such settings naturally demand *quantitative* reasoning frameworks capable of providing principled guarantees that account for the underlying uncertainty. (ii) These systems implement algorithms with worst-case exponential runtime, yet perform remarkably well on many practical instances. Our theoretical and empirical understanding of this "algorithmic luck" is still limited, which makes it difficult to systematically improve performance on hard instances. (iii) Many real-world problems could, in principle, be solved with existing highly optimized engines, but they often require slight extensions or modifications to the queries that current reasoning tools support.

My research addresses each of these gaps through three interconnected thrusts: (i) Advancing SMT beyond its traditional qualitative boundaries and develop new methodologies that enable quantitative reasoning within the SMT framework. I have developed an extensive toolkit for quantitative automated reasoning over SMT and QBF, enabling tasks such as model counting, projected counting, volume estimation, and quantitative synthesis [7, 9, 11, 12]. (ii) Designing systematic analysis frameworks for SAT solvers and model counters, leveraging causal reasoning and benchmark-wise analysis to explain and predict solver behavior. This work moves beyond empirical "rules of thumb" and towards principled, quantitative models of solver performance [10, 13]. (iii) Applying and extending these techniques to domains such as probabilistic model checking, network reliability, and cyber-physical systems, often by slightly generalizing the query interfaces supported by existing engines so that off-the-shelf tools can address new classes of real-world problems.

A particularly timely instantiation of these gaps, and a direction I am eager to pursue, is logic-based machine learning, especially inductive logic programming (ILP), where the goal is to learn symbolic rules from data using solver-backed search. The LOL group's focus on scaling ILP with SAT/ASP/SMT/MaxSAT aligns naturally with my experience building quantitative SMT backends and studying solver behavior. ILP workloads are fundamentally about navigating large hypothesis spaces under hard constraints, where incremental solving, constraint learning, and optimization (for example, MaxSAT trade-offs between accuracy and simplicity) are decisive. I am excited to bring my tool-building and analysis perspective to ILP, treating rule induction as a first-class automated reasoning workload, and to develop solver-centric methods that make learned rules both scalable and reliable in practice.

My work has appeared in leading formal methods and AI venues, including SAT, DAC, KR×2, and AAAI. I received the Best Student Paper Award at KR and have been awarded multiple medals in the SAT Competition.

Thrust 1: Developing Quantitative Automated Reasoning Systems

Satisfiability Modulo Theories (SMT) is a major workhorse in automated reasoning. I focus on *solution-space quantification* for SMT: going beyond the classical yes/no question of satisfiability to ask *how many* solutions a formula has, *how large* the solution space is, or *how* the solutions are distributed. The overarching goal is to obtain

algorithms that come with provable guarantees but also scale to realistic benchmarks.

Volume computation for real arithmetic formulas. The first setting concerns quantitative reasoning over *purely continuous domains*, specifically SMT formulas over linear real arithmetic (LRA), where the quantitative query becomes the *volume* of the feasible region. Prior work by Ge [2024] decomposes the region into *disjoint* polytopes, approximates each polytope's volume, and sums the results; however, enforcing disjointness can cause a prohibitive blow-up in the number of polytopes. In [12], we address this bottleneck by decomposing the region into *non-disjoint* polytopes and aggregating their volumes via a principled union-of-polytopes formulation. This reduces the combinatorial explosion in the decomposition while retaining accuracy guarantees. Our tool solves 8× more instances than the previous state-of-the-art LRA volume computation framework. The central conceptual shift is to abandon disjointness as a requirement and instead treat volume computation as a union-of-sets quantification problem.

Quantification in hybrid discrete-continuous domains. The second class of problems arises in *mixed discrete–continuous domains*, such as network reliability under power-flow models or probabilistic model checking, where constraints mix discrete and continuous variables but the quantitative query concerns a *projection* onto the discrete part. In [11], we adapt hashing-based counting. The result is a theory-aware reduction that preserves scalability while focusing the hashing on the discrete projection. The key ideas are to design hash functions that are sound and complete for the projected discrete domain and to employ solvers that efficiently handle the resulting hash constraints in the presence of continuous variables. This yields up to a 5× performance improvement over the current state-of-the-art techniques for hybrid projected counting.

Scalable bit-vector model counting. One of the most widely used SMT theories is the discrete theory of bit-vectors. Hadarean et al. observed that, in many cases, eager bit-blasting is the most effective approach for SMT bit-vector problems. Prior approaches to counting bit-vector formulas attempted to exploit structural properties of the original SMT formula. In [9], we proposed a tool that instead leverages eager *equi-cardinal* bit-blasting, followed by aggressive Boolean simplification and the use of high-performance Boolean model counters. With a careful integration of these three components, we obtain a 4× performance improvement over the prior state of the art for bit-vector model counting. Conceptually, this shows that for QF_BV, a well-engineered *blast-simplify-count* pipeline can dominate more theory-aware but structurally rigid approaches.

From model counting to function counting. Finally, when specifications describe functions from inputs to outputs, for example, relational specifications of controllers or hardware components, the natural quantitative question becomes: *How many distinct functions realize this specification?* The problem is particularly challenging, as synthesizing even one function is a hard problem. We design an algorithm, SkolemFC, that approximates the number of functions without synthesizing even one function. SkolemFC counts Skolem functions using only $O(n \log n)$ SAT calls, and shows a performance at the scale of synthesizing one function. This allows us to quantify the design space of implementations compatible with a given specification and provides a new quantitative lens on program and circuit synthesis.

All of the resulting algorithms come with (ε, δ)-style probabilistic accuracy guarantees. These contributions are realized in four open-source tools: `csb` for bit-vector model counting (including projected, weighted, and sampling variants), `pact` for hybrid discrete-projected counting, `ttc` for LRA volume computation, and `skolemfc` for Skolem-function counting. Across standard benchmark suites, these tools achieve 5×–10× improvements over prior state-of-the-art quantitative reasoning frameworks.

Thrust 2: Understanding Reasoning Engines

Modern automated reasoning engines like SAT solvers and model counters implement algorithms with worst-case exponential time complexity. Yet, on large classes of benchmarks from verification, synthesis, and AI, they routinely exhibit remarkable scalability. In this thrust, my goal is to move from empirical "rules of thumb" to a principled, quantitative understanding of why some instances are easy while others remain stubbornly hard, and how this structure should inform solver design.

Causality to understand SAT solvers. In [13], we take a data-driven, causal perspective on SAT solving. Rather than relying solely on expert intuition, we instrument modern CDCL solvers to generate rich observational traces and then learn a structured graph capturing causal relationships between core components. Using tools from causal

reasoning, our system, *CausalSAT*, (i) provides an extensible measurement and logging framework for SAT solvers, (ii) quantitatively fits empirical data from practical benchmark families, and (iii) formally validates several long-standing "rules of thumb" about restarts and clause management. Beyond explaining existing heuristics, the resulting causal graph suggests new, testable design changes and opens the door to learned heuristics that respect causal structure rather than treating the solver as a black box.

Understanding model counters. Unlike SAT solving, where a single paradigm largely dominates, CNF model counting features several competing approaches, including hashing-based approximate counters, and knowledge-compilation-based counters. Their comparative strengths on realistic benchmarks were poorly understood. In [10], we undertake a systematic empirical study of state-of-the-art counters over diverse benchmark families and show that there are regimes in which each paradigm excels while others consistently fail. We identify predictive structural parameters, such as component structure, independent-support size, and graph-theoretic measures like treewidth, that explain when each technique is effective. These predictors clarify when a given approach should be preferred and provide a foundation for model-counter portfolios and automatic algorithm selection guided by structural properties of the input.

The lens of competition. Since 2024, I have been working as a co-organizer of the Model Counting Competition, which has given me a unique vantage point on both solvers and benchmarks. As organizers, we routinely validate submitted tools and, in the process, have uncovered and helped fix bugs in multiple counters, directly improving the robustness of widely used systems. We have also introduced new tracks that support complex-valued weights, motivated by applications in quantum simulation and related domains. These tracks have, in turn, encouraged tool developers to extend their counters to handle complex weights and newer formalisms, broadening the applicability of model counting. More broadly, the competition provides a continuous stream of realistic benchmarks and performance data, which I leverage to test hypotheses about solver behavior and to drive the next generation of reasoning engines.

Thrust 3: Applying Quantitative Reasoning Engines to Real-World Systems

The ultimate test of automated reasoning is its ability to address hard problems arising in practice. Many such problems involve systems operating under uncertainty, where the central questions are inherently quantitative (e.g., probabilities of failure or reliability of large infrastructures). In this thrust, I apply the quantitative reasoning engines developed in Thrusts 1 and 2 to two such domains: probabilistic model checking and network reliability.

Scalable probabilistic model checking via SMT-based counting. For systems that exhibit probabilistic behavior, the model checking question becomes quantitative: what is the probability that a given temporal property holds? Modern probabilistic model checkers perform extremely well on many benchmarks, yet, as Holtzen et al. highlight, there exist surprisingly simple models on which they struggle due to state-space explosion. In [8], we show that such cases can be reduced to SMT-based counting problems: we encode the relevant execution paths as an SMT formula and decompose the overall task into a set of structurally simple model-counting subproblems. By leveraging the counting engines developed in Thrust 1 as backends, our tool analyzes up to a 32× many parallel processes a collection of challenging benchmarks.

Approximate network reliability for continental-scale DAGs. Network reliability is a key challenge in modern civil and electrical engineering, with applications ranging from power distribution networks to gas transmission systems. Paredes observe that many such networks can be modeled as directed acyclic graphs (DAGs), and that weighted model counting is the best approach to compute reliability. However, Feng and Guo showed that reliability on DAGs admits an FPRAS, revealing a complexity gap. Building on these insights, in a line of ongoing work we develop a highly efficient FPRAS implementation specialized to DAG-structured networks, using our quantitative reasoning engines as core subroutines. Our implementation scales to continental-scale power transmission networks with up to 80,000 vertices, bringing theoretically grounded approximation schemes closer to practical engineering workflows.

Future Directions and Outlook

Driven by real-world applications, my work so far has built foundational tools for SMT solution-space quantification and helped set the *virtuous cycle* of theory, tools, and applications [1] in motion. As these tools improve, I expect them to both unlock new applications and pose new foundational questions. Looking ahead, I plan to build on my three thrusts by following.

Further pushing the boundaries of quantitative reasoning engines. Counting and sampling are two sides of the same coin: approximate counters often yield samplers, and high-quality samplers can in turn support estimation. While much of my thesis focuses on counting and volume computation, many downstream tasks, such as test generation for fairness, robustness, or rare-event analysis, require drawing representative samples from complex SMT-defined spaces. A key future direction is to develop *theory-aware samplers* that combine hashing-based techniques, Markov chain methods, and SMT reasoning to provide both practical performance and rigorous guarantees on diversity, coverage, and bias.

Solving real-world problems via quantification. Network reliability, cyber-physical system safe-zone quantification, and fairness and robustness analysis for AI systems can all be encoded as SMT-based counting or volume problems. Each of these domains comes with its own modeling assumptions, data regimes, and scalability constraints. I plan to build collaborations with experts in power systems, control, and machine learning to refine these encodings, validate them on realistic case studies, and identify the right quantitative questions (e.g., worst-case vs. average-case risk, sensitivity to modeling choices) that automated reasoning can most effectively answer.

Quantitative automated reasoning in the AI loop. Recent work has shown the promise of combining large language models (LLMs) with automated reasoning for tasks such as program synthesis, program verification, and mathematical proof generation. In most current workflows, the automated reasoner is queried as a *qualitative* oracle that returns only yes/no answers or counterexamples. I plan to investigate *quantitative* feedback in this loop: for instance, using SMT counters to provide model sizes, coverage metrics, or robustness scores for candidate programs or proofs proposed by an LLM. Such richer signals can guide search, prioritize refinements, and help the LLM reason about uncertainty and partial progress.

Causality-aware AI for automated reasoning. Conversely, language models are increasingly used to design solver heuristics and tuning strategies, sometimes outperforming human-crafted designs. This creates a powerful but opaque combination: we are coupling two complex systems, LLMs and solvers, that we only partially understand. Building on my work on causal analysis of solver behavior, I plan to develop *causality-aware* frameworks that use data-driven causal models to interpret, audit, and robustify AI-generated heuristics. The long-term vision is an automated reasoning ecosystem in which (i) AI designs and adapts strategies, (ii) quantitative and causal analysis explains and validates these strategies, and (iii) the resulting insights feed back into both solver design and AI training.

References

[1] Cook, B. (2022). Automated reasoning's scientific frontiers. [link]. Amazon Science blog post.

[2] Feng, W. and Guo, H. (2024). An FPRAS for Two Terminal Reliability in Directed Acyclic Graphs. In *Proceedings of EATCS International Colloquium on Automata, Languages and Programming (ICALP)*.

[3] Ge, C. (2024). sharpSMT: A Scalable Toolkit for Measuring Solution Spaces of SMT(LA) Formulas. *Frontiers of Computer Science (FCS)*.

[4] Hadarean, L., Bansal, K., Jovanović, D., Barrett, C., and Tinelli, C. (2014). A tale of two solvers: Eager and lazy approaches to bit-vectors. In Biere, A. and Bloem, R., editors, *Proceedings of International Conference on Computer-Aided Verification (CAV)*.

[5] Holtzen, S., Junges, S., Vazquez-Chanlatte, M., Millstein, T., Seshia, S. A., and Van den Broeck, G. (2021). Model checking finite-horizon markov chains with probabilistic inference. In *Proceedings of International Conference on Computer-Aided Verification (CAV)*.

[6] Paredes, R. (2022). *Principled Uncertainty Quantification for Resilient Infrastructure Management*. PhD thesis, Rice University.

[7] Shaw, A., Juba, B., and Meel, K. S. (2024). An approximate skolem function counter. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

[8] Shaw, A., Junges, S., and Meel, K. S. (2026). Covering-based approximate model counting for statistical model checking. In *Under Submission at CAV*.

[9] Shaw, A. and Meel, K. S. (2024a). CSB: A Counting and Sampling Tool for Bitvectors. In *Proc. of SMT Workshop at CAV*.

[10] Shaw, A. and Meel, K. S. (2024b). Model counting in the wild. In *Proceedings of International Conference on Knowledge Representation and Reasoning (KR)*.

[11] Shaw, A. and Meel, K. S. (2025). Approximate smt counting beyond discrete domains. In *Proceedings of Design Automation Conference (DAC)*.

[12] Shaw, A., Sarkar, U., and Meel, K. S. (2025). Efficient volume computation for smt formulas. In *Proceedings of Knowledge Representation and Reasoning (KR)*.

[13] Yang, J., Shaw, A., Baluta, T., Soos, M., and Meel, K. S. (2023). Explaining sat solving using causal reasoning. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT)*.