RESEARCH STATEMENT

KULDEEP S. MEEL
Associate Professor, University of Toronto
Website: https://www.cs.toronto.edu/~meel/

OVERVIEW

My primary research interest is in automated reasoning, which lies at the intersection of formal methods and artificial intelligence. My research program focuses on the development of scalable automated symbolic reasoning techniques and their integration with statistical reasoning methods to enable the development of provably robust, reliable and trustworthy computing systems. The current generation of symbolic reasoning techniques excel at the qualitative tasks (i.e., when the answer is Yes or No); such techniques sufficed for traditional hardware and software systems whose design sought to achieve deterministic behavior. In contrast, modern computing systems crucially rely on the statistical methods to account for the uncertainty in the environment, and to reason about behavior of these systems, there is need to look beyond qualitative symbolic reasoning techniques. My research group develops the **next generation of automated reasoning** techniques that can perform **higher-order tasks** such as quantitative measurement, sampling of representative behavior, and automated synthesis of systems.

From a core technical perspective, my research program builds on the *SAT revolution*, which refers to algorithmic advances in combinatorial solving techniques for the fundamental problem of satisfiability (SAT), i.e., whether it is possible to satisfy a given set of constraints. The SAT revolution offers the opportunity to develop scalable techniques for problems that lie *beyond SAT* from complexity perspective and, therefore, stand to benefit from the availability of powerful SAT engines. My research group seeks to enable a *Beyond SAT revolution* via design of scalable techniques for three fundamental problems that lie beyond SAT: *constrained counting, constrained sampling,* and *automated synthesis*. To achieve scalability, we pursue a tight end-to-end integration among three research thrusts: solver-aware algorithmic frameworks, data-driven system design, and real-world applications.

The first research thrust focuses on the development of **solver-aware algorithmic frameworks** that enable leveraging the success of SAT solvers, and are backed by rigorous theoretical guarantees. This approach has necessitated the development of several new theoretical concepts along with radically new and unconventional SAT solver-aware algorithmic constructs whose design is influenced by data-driven analysis of the behavior of solvers. The second research thrust focuses on **data-driven system design** to achieve a tighter integration with algorithmic frameworks. The modern SAT solvers are designed to be general purpose but the algorithmic frameworks for *Beyond SAT* present opportunities for tighter integration between solvers and algorithms so as to allow the solvers to exploit the structure of the queries. We exploit such opportunities via novel data-driven paradigms and hardware accelerators, and the resulting tight coupling leads to dramatic scalability gains. The third research thrust focuses on the deployment of our algorithms frameworks in **real-world applications** to achieve a virtuous feedback cycle informing our data-driven algorithmic and system design. Through close collaboration with industry and interdisciplinary efforts, we have utilized our techniques in compelling real-world applications such as quantitative verification of neural networks [29, 66, 60] , hardware and software testing [7, 22, 35, 48, 47, 63] quantitative information flow analysis [19], resilience of critical infrastructure [17, 25], and probabilistic inference [54]. As an example, we obtained *the first theoretically sound estimates of resilience* in power transmission networks of ten medium sized cities in US. Furthermore, our **award winning open source tools** have been employed by other researchers in domains such as probabilistic databases, phylogeny inference, cryptanalysis, and network verification.

Recognition: Prominent recognition of my research includes (i) NRF Fellowship for AI, accompanied with 1.9 million USD funding, (ii) ACP Early Career Research Award, (iii) *AI's 10 to Watch* by IEEE Intelligent Systems, and (iv) Early Career Spotlight at IJCAI 2022. Our CP 2013 paper has been chosen as one of the 25 years across 25 years to celebrate 25th anniversary of CP conference. Our CAV-23 received Distinguished Paper Award. Our ICCAD-21 and DATE-23 papers received best paper award nominations. Our PODS-21 paper received ACM SIGMOD 2022 Research Highlight Award, 2023 CACM Research Highlights Award, and publication in ACM TODS as "Best of PODS-21". Our CAV-20 paper received invite from FMSD journal issue dedicated to the best papers from CAV 2020. The CP 2018 paper was invited to IJCAI-19 Sister Conference Best Paper Track while CP 2015 paper received the Best Student Paper Award. I co-founded the annual workshop on counting and sampling in 2020, and I served as program co-chair for SAT 2022. I have also co-organized workshops at Simons Institute and Dagstuhl. I have co-presented tutorials at UAI-16, AAAI-17, IJCAI-18, AAAI-20, IJCAI-20, AAAI-22, and

IJCAI-22.

My work straddles theory and practice, and draws upon ideas from randomized algorithms, statistical inference, formal methods, distribution testing, and software engineering. As such, I have established strong collaborations across broad areas of computer science, reflected by publication record in automated reasoning (CAV: $8\times$, CP: $8\times$, SAT: $7\times$, TACAS:$3\times$, KR), databases (PODS: $6\times$), artificial intelligence (AAAI: $21\times$, IJCAI:$13\times$, NeurIPS: $6\times$), design automation (DAC; $2\times$, ICCAD: $2\times$, DATE: $2\times$), software engineering and security (ICSE/FSE/CCS).

Research Summary

1 Solver-Aware Algorithmic Frameworks

We focus on *three fundamental problems* that lie beyond SAT: *constrained counting, constrained sampling,* and *automated synthesis*. In constrained counting, the task is to count the total weight, subject to a given weight function, of the set of solutions of input constraints. In constrained sampling, the task is to sample randomly, subject to a given weight function, from the set of solutions of input constraints. In the case of automated synthesis, given a set of constraints capturing the functional specification between inputs and outputs, the task is to construct a program (represented as a circuit) whose outputs meet the desired specification. I discuss below our solver-aware algorithmic frameworks, the ensuing deployment in real-world applications, and data-driven system design.

1.1 Constrained Counting

We introduced a novel universal-hashing based paradigm, ApproxMC, wherein the problem of constrained counting over a n-dimensional space is reduced to solving a small number of queries to a combinatorial solver wherein each query is the underlying set of constraints augmented with randomly generated parity constraints [3, 6, 10, 16, 33, 45, 48, 51]. The core idea of ApproxMC is to use pairwise independent hash functions, expressed as randomly generated XOR constraints, to randomly partition the solution space of the original formula into “small” enough cells. The sizes of sufficiently many randomly chosen cells are then determined using calls to a SAT solver, and a scaled median of these sizes is used to estimate the desired total weight. Our CP 2013 paper, which introduced ApproxMC, was selected as **one of the 25 papers across 25 years for CP anniversary volume** in 2019.

Our framework seeks to build on the success of modern SAT solvers, and therefore, has led to novel and unconventional algorithmic constructs. For example, prior work used independence among different SAT queries as a central component, which was preferred owing to the ease of theoretical analysis. In contrast, we have focused on the introduction of dependence among different SAT-solver queries to maximize the sharing of information among different calls by an underlying incremental SAT solver [16]. The use of hash functions that allow dependent SAT-solver queries also allows a linear search to be replaced by a logarithmic search. The sizes of XOR constraints in the formulas that are fed to a modern SAT solver have a significant impact on the running time of the solver. By exploiting the connection between definability of formulas and variance of the distribution of solutions in a cell defined by 3-wise independent hash functions, we introduced an algorithmic technique, MIS, that reduced the size of required XOR constraints dramatically and improved the runtime performance of ApproxMC by orders of magnitude. The corresponding publication [14] received **Best Student Paper Award** at CP 2015. We further exploited a beautiful connection between concentration measures of XOR-based hash functions and isoperimetric inequalities on Boolean hypercubes to resolve the long-standing open problem of the design of logarithmic size XORs that provide the desired theoretical guarantees and achieve practical scalability [52].

A fundamental contribution of our approach is its generalizability. In particular, if the set of constraints is expressed in Disjunctive Normal Form (DNF), ApproxMC directly yields a Fully Polynomial Randomized Approximation Scheme (FPRAS) for counting – the only known FPRAS for DNF that does not involve Monte Carlo steps [16]. The resulting approach has also been demonstrated to be superior in performance to the classical Monte Carlo technique [20]. The corresponding publication received an invitation to **Sister Conferences Best Paper Track** at IJCAI-19. Furthermore, we demonstrated ApproxMC can be lifted to count the minimal unsatisfiable subsets of a given formula, which is an important metric of diagnosis of faults in a system. The corresponding publication [49] received an invitation to the FMSD special issue dedicated to the **“best papers from CAV 2020”**.

DATA STREAMS AND COUNTING We established a deep and natural connection between constrained counting and F_0 estimation [69]. Given a stream of sets $a = a_1, a_2, \ldots, a_m$ wherein each set $a_i \subseteq \Omega$ the zeroth frequency moment, denoted as F_0, of a is the number of distinct elements appearing in a, i.e., $| \cup_i a_i|$. F_0 computation is a fundamental problem in data streaming with a rich history of prior work. In particular, we design a recipe to transform streaming algorithms for F_0 estimation to those for constrained counting. Such a transformation yields new (ε, δ)-approximate algorithms for constrained counting, which are different from currently known algorithms, and vice versa. The corresponding publication [69] at PODS-21 received an invitation as **"Best of PODS 2021"** by ACM TODS. In a folow-up work, we designed the first algorithm with $\mathrm{poly}(\log |\Omega|, m)$-space and time complexity for the case where each a_i allows *efficient* (i.e., $\mathrm{poly}(\log |\Omega|, m)$-time complexity) access to membership, sampling, and counting [68]. In particular, our result resolved the *open problem* of the existence of $\mathrm{poly}(\log |\Omega|, m)$-space and time complexity algorithms for Klee's measure problem in discrete settings. This work received praise from **Donald Knuth** who wrote: "Indeed, ever since I saw it, a few days ago, I've been unable to resist trying to explain the ideas to just about everybody I meet. Therefore I'm pretty sure that something like this will eventually become a standard textbook topic".

Practical Impact

The open-source version of ApproxMC can handle formulas with up to a million variables, representing a dramatic jump in the capabilities of prior state-of-the-art approximate counting tools, which could scale to only hundreds of variables. Our entry to the 2020 model counting competition **won first place**.

Quantitative Verification of Neural Networks Given a system $\mathcal{N}$ and a property $\mathcal{P}$, the classical verification methodology focuses on the qualitative question of detecting whether there exists some input for which the output of $\mathcal{N}$ does not satisfying $\mathcal{P}$. While the qualitative verification methodology has been dominant paradigm in the context of traditional hardware and software systems wherein presence of bug is often viewed unacceptable, the advent of neural networks requires a rethinking given the statistical guarantees on the behavior of these systems. We proposed the notion of quantitative verification to capture how often a property $\mathcal{P}$ is satisfied by a network $\mathcal{N}$ [29, 66]. We demonstrated how the framework can be applied in the context of properties such as robustness, susceptibility to trojan attacks, and fairness. As a concrete example, an analyst can analyze whether a statistically significant number of adversarial examples exist for a given input under user-defined distributions of perturbations. We have designed the first scalable framework, called NPAQ, with formal guarantees of correctness for binarized neural networks [29, 41]. NPAQ reduces the problem of quantitative verification of binarized neural networks to projected model counting, therefore allowing us to rely on the scalability of ApproxMC. The open-source release of NPAQ has allowed the community to focus on designing training methodologies so that the resulting networks can be efficient certified.

Resilience of Critical Infrastructure The availability of critical facilities and utility services, such as power, telecommunications, water, gas, and transportation is crucial to an increasingly connected world. Natural disasters often result in disruption of underlying networks. In a joint project with collaborators from civil and electrical engineering, we introduced a new approach, RelNet, to measure the resilience of underlying systems in events of natural disasters [17, 25]. The approach relies on the representation of the problem of computation of resilience as constrained counting over the configurations of the edges in the underlying graph. The computational engine of RelNet was applied to ten power transmission networks powering small to medium sized cities in the states of Texas, Florida, California, Tennessee, Georgia, and South Carolina. Our approach was successfully applied to obtain *the first theoretically sound estimates of resilience* in these power transmission networks.

Quantitative Information Flow Analysis: Quantitative information flow analysis is a powerful code-analysis technique to detect leakage of secret program information to public program outputs. A specific fragment of the program (e.g., a function, or the whole program) is modeled as an information-theoretic channel from its input to its output. To compute the maximum amount of information that can leak from the program fragment of interest, a constrained counter is used to determine the number of distinct outputs of the fragment (e.g., return values of the function, or the outputs of the program). By counting the number of cases where information is leaked, an information-theoretic estimate of the quantified information flow can be obtained. We have demonstrated that our approximate counting tools provide a scalable and accurate approach for quantitative information leakage [19].

1.2 Constrained Sampling

We exploited the inter-reducibility of constrained counting and sampling and the properties of 3-wise independent hash functions to design the first scalable constrained sampler, UniGen, that can sample solutions of a given set of propositional a set of constraints [2, 4, 7, 51]. Unlike prior approaches pioneered that either required a linear number of calls to a constrained counter or computationally prohibitive steps of inverting n-wise independent hash functions, UniGen requires *a single call* to a constrained counter and employs hash functions that can be inverted reasonably efficiently in practice using state-of-the-art SAT solvers. The algorithm first invokes a constrained counter, viz. ApproxMC, to get an approximate count of the size of the solution space. Having this count enables automatically determining the parameters for using 3-wise independent hash functions, expressed as random XOR constraints. These constraints are used to partition the solution space in such a way that a randomly chosen cell is "small" enough in expectation, and the solutions in it can be enumerated by an existing SAT solver and sampled accordingly. While this yields a provably *almost uniform* sampling, the practical performance is, surprisingly, even better: in particular, for benchmarks where the ideal distribution can be generated using exhaustive enumeration, the distributions from UniGen are statistically indistinguishable from the ideal distributions [4].

DISTRIBUTION TESTING-DRIVEN DEVELOPMENT Given the widespread applications of sampling, scalability remains a major challenge despite recent developments in the community, including UniGen. Consequently, for applications where samplers such as UniGen do not scale, the samplers based on heuristics and lacking theoretical analysis are the last resort. Recognizing the gap between the existence of heuristic-based samplers and lack of testing tools with rigorous guarantees, we initiated Barbarik project that combines distribution testing paradigm with modeling of the behavior of solvers to design a tester that can provably test whether a given sampler's output distribution is uniform or not [32, 43, 61]. We demonstrated that Barbarik is able to successfully *reject* samplers' whose distribution is far from uniform and would *accept* samplers such as UniGen. Next, we pursued a test-driven methodology in the design of CMSGen [63], a sampler based on state of the art SAT solver, whose theoretical analysis is beyond the reach of the currently existing techniques but for which Barbarik returns *accept*. Crucially, CMSGen is able to achieve significant scalability in comparison to existing samplers with theoretical guarantees, and thereby achieving the right trade-off between scalability and sample quality.

Practical Impact

Testing of Highly Configurable Software Systems The widespread and diverse usage has led to the design of highly configurable software systems operating in diverse environments. A fundamental problem is the generation of test configurations that maximize t-wise coverage while respecting constraints that rule out invalid configurations. Building on the scalability of our constrained samplers, we proposed an adaptive weighted sampling approach, called baital, that achieves significantly higher t-wise coverage [47].

Hardware-Design Verification: Every major Electronic Design Automation company employs simulation-based techniques for functional verification, which heavily depend on the *quality* of test inputs with which the design is simulated. The standard industrial approach is *constrained-random verification*, in which random solutions of carefully crafted constraints are used as test inputs. Prior techniques either fail to provide theoretical guarantees on the quality of test inputs generated or fail to scale beyond toy designs. We extended our hashing-based framework, UniGen, to design the first and only linearly scalable distributed method for test-input generation that comes with rigorous approximation guarantees on the quality of generated stimuli [7].

Ride Sharing Route Prediction For operational and financial efficiency, a central problem for delivery and ride-sharing companies is inference and prediction of routes taken by the drivers. In particular, whenever a customer requests a ride from location A to location B, the ride-sharing app needs to predict the route in order to determine the trip time, fare, and establish trust between the customers, drivers, and the company. In a joint work with Grab Taxiholding Limited , by viewing route distributions as structured probability distributions (SPD), we rely on our compilation-based constrained sampling approaches to efficiently learn and predict the routes taken for a given source and destination, while achieving up to 6× performance improvement.

1.3 Automated Synthesis

Automated synthesis concerns with the automatic synthesis of programs (also represented as circuits) that provably meet the end user's functional requirements. We introduced a novel data-driven approach, manthan, that combines advances in machine learning and formal methods [50, 70]: we rely on machine learning to generate candidate functions and then rely on formal methods to repair the generated candidate functions to synthesize functions that provably meet the end user's functional requirements. manthan consists of three phases: we first exploit the recent advances in constrained sampling to generate samples from the given relational specification, say $R(X, Y)$ over inputs X and outputs Y. Next, we cast the functional synthesis as a classification problem wherein the input variables (X) in samples correspond to features while output variables (Y) correspond to labels. Consequently, the generated samples are fed as training data, and the learned classifier is encoded as a Boolean function. Often, machine learning techniques are capable of generating *almost correct* and our approach relies on advances in SAT and MaxSAT to repair such *almost correct* functions. To this end, we rely on automated reasoning techniques for the diagnosis and repair of candidate functions. The corresponding publications received **Best Paper Award Nomination** at DATE 2023 and ICCAD 2021.

Practical Impact

Program Synthesis The approach of combining formal methods and machine learning has led to dramatic scalability: on the standard suite of 609 instances, the prior state of the art techniques ranged from 210 to 280 instances while manthan can solve 509 instances; therefore, manthan is the only functional synthesis technique to solve more than 300 instances. Motivated by the impressive scalability, we turned our attention to program synthesis tasks commonly represented in Syntax Guided Synthesis (SyGuS) form. We demonstrated that the problem of program synthesis, represented in SyGuS, reduces to functional synthesis when there are no restrictions on the grammar [55]. Our reduction allows us to transform manthan as a state of the art approach for program synthesis tasks over bit-vector theory.

Scalable Interpretable Rule Learning We developed MLIC, a scalable interpretable decision rule learning framework that provides a precise control of accuracy vs. interpretability [21, 30, 39]. The key strength of the framework lies its ability to separate the modeling from the optimization and therefore has applications in a wide variety of interpretable classification formulations, including group-sparsity and having prior knowledge on variable importance. We observed that a major bottleneck to scalability of MLIC is the size (i.e., number of clauses) of optimization query, which would be linear in the number of samples in the training data. To achieve scalability, we introduced a novel partitioning-based incremental optimization solving approach that achieves scalability by invoking optimization solver linearly many times wherein each query is over a fixed size formula. The resulting open-source tool can now handle problems involving up to million samples in training data, and generates highly accurate decision rules.

2 Data-Driven System Design

The modern SAT solvers are designed to be general purpose but the algorithmic frameworks for *Beyond SAT* present opportunities for tighter integration between solvers and algorithms so as to allow the solvers to exploit the structure of the queries. We exploit such opportunities via new paradigms for SAT sovlers, and the resulting tight coupling leads to dramatic scalability gains.

Native Support for Parity (XOR) Constraints The constrained counting and sampling techniques heavily rely on the usage of pairwise independent hash functions encoded as random XOR (parity) constraints. Consequently, the SAT solver is queried with conjunction of CNF and XOR constraints, also known as CNF-XOR formulas. Profiling of counters and samplers revealed that over 99% of the runtime is consumed by the underlying SAT solvers, thereby critically highlighting the need for efficient design of solvers with native support for CNF-XOR formulas. In a long-term collaboration with Mate Soos, we introduced a novel framework, called bird, which put forth an unconventional architecture: bird first blasts the XOR constraints into CNF so that the entire formula is available to the solver in CNF [33, 51]. Furthermore, after every inprocessing step, bird recovers XOR constraints so that CDCL can be performed with on-the-fly Gauss-Jordan Elimination on the recovered XOR constraints. Our specialized data structure operations based on matrix row handling achieve efficient propagation and conflict checking of XOR constraints. Our project on verification of Binarized Neural Networks (BNN) showed that BNNs can be encoded naturally into Pseudo-Boolean (PB) constraints, and highlighted the need for counting tool with native support for PB. Consequently, we developed LinPB, a PB solver with native support for PB-XOR formulas [64], and ApproxMC augmented with LinPB is able to handle significantly larger neural networks.

Bridging Algebraic and Combinatorial Solving Techniques Algebraic Normal Form (ANF) and Conjunctive Normal Form (CNF) are two commonly used normal forms in Boolean algebra. ANF is a *system of polynomial equations* in GF(2), where each monomial is a product of zero or more variables. ANF is widely preferred by cryptologists as it permits natural representation of cryptographic primitives. ANF solvers are based on the computation of Gröbner basis, and often struggle due to memory requirements. On the other hand, modern SAT solvers take in a CNF formulas as input, and use significantly less memory owing to the depth-first nature of CDCL. The modern SAT solvers, however, struggle with algebraic constraints, and therefore, highlighting the contrasting nature of CNF and ANF solvers. Our framework Bosphorus [23] bridges ANF and CNF solving techniques by iteratively applying techniques from ANF and CNF solving to *learn facts* to augment the original problems, and via frequent conversions from ANF to CNF, and vice-versa. The paradigm proposed in Bosphorus allows problems to be encoded in their most natural and comprehensible manner, either in ANF or CNF, and draws from solving techniques to achieve robust runtime performance. The resulting open source library has attracted significant involvement from cryptographic community.

GPU-Enabled Parallel SAT Solving The crucial importance of CPU+GPU architecture to unprecedented advances in machine learning prompted us to investigate its usefulness in the context of parallel SAT solving. To this end, we have designed a novel CDCL-based framework called GpuShareSat, designed to take advantage of CPU+GPU heterogeneous architectures [65]. We focus on the major Achilles heel of parallel SAT solvers: identifying which clauses to import from other threads. We observe that efficient bit-vector-based operations can allow a GPU to efficiently determine the usefulness of a learnt clause to different threads and accordingly notify the thread of the presence of these relevant clauses, which is determined based on the locality of assignments. The resulting solver achieves improvement over the 2020 SAT competition winners. Furthermore, we have remained steadfast in our focus to enable community adoption of our framework, and to this end, the augmentation of our open-source library requires minimal modification to the existing architecture of the CDCL solvers. The project was recognized with an **Amazon Research Award 2021** for automated reasoning.

Machine Learning-enabled SAT Solving The dramatic progress in the SAT solving relied on careful software engineering, which owes much to the design of powerful hand-crafted heuristics by experts. The process of manually tuning of the heuristics continues to the day with leading SAT solver developers spending tens of hours every year in the lead up to the SAT competition. Our ambitious project CrystalBall, envisioned to be a decade-long effort, focuses on building a data-driven framework that seeks to transform the task of design of heuristics into supervised learning by a precise collection of the data about the run of a solver and high-quality labeling relying on the recent advances in proof generation [27]. Since a conflict-driven solver encounters millions of propagations and conflicts for a a typical practical instance, we first designed a high throughput and low latency framework to collect detailed statistics of a run of a SAT solver. To demonstrate the utility of our approach, we first focused on the design of *clause deletion* policies for the modern CDCL solvers. We have showcased that automatically learned heuristics perform as well as manually tuned heuristics by experts over the years [27].

Future Research Plans

My long-term research plan is to continue designing scalable automated symbolic reasoning techniques to enable the development of provably robust, reliable, and trustworthy computing systems. As evident from our progress, addressing these computational challenges would require an influx of ideas from both foundational research and practice. My interdisciplinary and industrial collaborations have provided deeper insights into the structure of the problems, and I will continue to seek such further collaborations. Below are several research themes that will benefit from this agenda.

Achieving Beyond SAT Revolution Our quest to enable a *Beyond SAT* revolution has contributed to the significant scalability of today's state of the art counters, samplers, and synthesis engines. These are still early days for *Beyond SAT* techniques, and there remain immediate and significant problems to work on along the three research thrusts: applications, algorithmic frameworks, and system design. Given the inherent computational intractability of counting, sampling, and synthesis, the progress in algorithmic frameworks and system design relies on exploiting the underlying problem structure found in problem instances arising from real-world applications. In this regard, identifying well-defined challenges is crucial to align the efforts. We seek to achieve the following **three goals in 2026**: (1) quantitative verification of robustness for a binarized neural network with one million neurons, (2) rigorous resilience of critical infrastructures such as power grid of cities of size Los Angeles, and (3) software reliability and coverage analysis for programs with 10K lines of code. Achieving these goals would necessitate algorithmic innovations that can take advantage of the underlying structure in real-world problem instances, develop data-driven solver design, and utilize new architectures.

Probabilistic Programming Probabilistic programming systems have emerged as a powerful paradigm to represent probabilistic models using standard programming language constructs, and the presence of powerful inference techniques allows the developer to perform probabilistic reasoning without necessitating expertise in the design of inference techniques. We have recently initiated investigations into one of the fundamental problems in the context of testing and verification: equivalence checking. Given two probabilistic programs, the problem of equivalence checking is to determine whether they specify identical distributions. The development of algorithmic techniques for equivalence testing of probabilistic programs must address the challenge for appropriate notion for a witness for non-equivalence of two probabilistic programs. While a trace where the output of two deterministic programs differs suffices as a witness for non-equivalence, such is not the case for probabilistic programs. We seek to address the above challenges by proposing a novel algorithmic framework relying on the viewpoint of probabilistic programs as distributions and building on the distribution testing framework we have developed in the context of testing samplers. We will create a modular verification framework to achieve scalability where proofs for equivalence based on the distribution testing framework can be composed together.

Expressiveness Over the longer time horizon, I will focus on expanding expressiveness via lifting techniques to higher-order theories. The success of SAT in the early 2000s inspired the quest to lift satisfiability techniques to more expressive theories such as linear real arithmetic, bitvectors, strings; such constraints allow precise modeling of modern hardware and software. These efforts have yielded a vibrant ecosystem with the availability of state of the art Satisfiability Modulo Theory (SMT) techniques that serve as crucial engines in modern formal methods and artificial intelligence. The availability of SMT solvers in the 2020s is akin to the availability of SAT solvers in the 2010s, and the time is ripe for *Beyond SMT* techniques that focus on problems beyond satisfiability: counting/integration, sampling, and synthesis. Similar to work on *Beyond SAT*, there is a need for the development of algorithmic foundations that can leverage the success of SMT solvers, which would necessitate a data-driven system design of SMT solvers, and the identification of real-world applications to achieve a virtuous feedback cycle of data-driven algorithmic design. The success of *Beyond SMT* program would allow us to broaden the scope of our applications such as scalable quantitative verification techniques for large deep neural networks, ability to augment complex physics-based modeling in critical infrastructure resilience, and allow quantitative reasoning over programs over strings and floating point operations, and the like. Our initial foray into *Beyond SMT* [10, 53] highlights the aforementioned challenges and opportunities, and I am excited to pursue this as a long-term research program in **this decade**.

References

[1] Meel, K. S., M. Y. Vardi, S. Chakraborty, D. J. Fremont, S. A. Seshia, D. Fried, A. Ivrii, and S. Malik. "Constrained Sampling and Counting: Universal Hashing meets SAT Solving". In: *Proc. of Beyond NP Workshop*. 2016.

[2] Chakraborty, S., K. S. Meel, and M. Y. Vardi. "A Scalable and Nearly Uniform Generator of SAT Witnesses". In: *Proc. of CAV*. 2013.

[3] Chakraborty, S., K. S. Meel, and M. Y. Vardi. "A Scalable Approximate Model Counter". In: *Proc. of CP*. 2013.

[4] Chakraborty, S., K. S. Meel, and M. Y. Vardi. "Balancing Scalability and Uniformity in SAT-Witness Generator". In: *Proc. of DAC*. 2014.

[5] Majeti, D., K. S. Meel, R. Barik, and V. Sarkar. "ADHA: Automatic Data layout framework for Heterogeneous Architectures". In: *Proc. of PACT*. 2014.

[6] Chakraborty, S., D. J. Fremont, K. S. Meel, S. A. Seshia, and M. Y. Vardi. "Distribution-Aware Sampling and Weighted Model Counting for SAT". In: *Proc. of AAAI*. 2014.

[7] Chakraborty, S., D. J. Fremont, K. S. Meel, S. A. Seshia, and M. Y. Vardi. "On parallel scalable uniform SAT witness generation". In: *Proc. of TACAS*. 2015.

[8] Chakraborty, S., D. Fried, K. S. Meel, and M. Y. Vardi. “From Weighted to Unweighted Model Counting”. In: *Proc. of IJCAI*. 2015.

[9] Ivrii, A., S. Malik, K. S. Meel, and M. Y. Vardi. “On Computing Minimal Independent Support and Its Applications to Sampling and Counting”. In: *Proc. of CP*. Best Student Paper Award. 2015.

[10] Chakraborty, S., K. S. Meel, R. Mistry, and M. Y. Vardi. “Approximate Probabilistic Inference via Word-Level Counting”. In: *Proc. of AAAI*. 2016.

[11] Majeti, D., K. S. Meel, R. Barik, and V. Sarkar. “Automatic Data Layout Generation and Kernel Mapping for CPU+GPU Architectures”. In: *Proc. of CC*. 2016.

[12] Dudek, J., K. S. Meel, and M. Y. Vardi. “The Hard Problems Are Almost Everywhere For Random CNF-XOR Formulas”. In: *Proc. of IJCAI*. 2017.

[13] Murthy, K., S. R. Paul, K. S. Meel, T. Cogumbreiro, and J. Mellor-Crummey. “Design and Verification of Distributed Phasers”. In: 2016.

[14] Ivrii, A., S. Malik, K. S. Meel, and M. Y. Vardi. “On Computing Minimal Independent Support and Its Applications to Sampling and Counting”. In: *Constraints* 21.1 (2016).

[15] Dudek, J., K. S. Meel, and M. Y. Vardi. “Combining the k-CNF and XOR Phase-Transitions”. In: *Proc. of IJCAI*. 2016.

[16] Chakraborty, S., K. S. Meel, and M. Y. Vardi. “Algorithmic Improvements in Approximate Counting for Probabilistic Inference: From Linear to Logarithmic SAT Calls”. In: *Proc. of IJCAI*. 2016.

[17] Duenas-Osorio, L., K. S. Meel, R. Paredes, and M. Y. Vardi. “Counting-Based Reliability Estimation for Power-Transmission Grids”. In: *Proc. of AAAI*. 2017.

[18] Meel, K. S., A. A. Shrotri, and M. Y. Vardi. “On Hashing-Based Approaches to Approximate DNF-Counting”. In: *Proc. of FSTTCS*. 2017.

[19] Biondi, F., M. Enescu, A. Heuser, A. Legay, K. S. Meel, and J. Quilbeuf. “Scalable Approximation of Quantitative Information Flow in Programs”. In: *Proc. of VMCAI*. 2018.

[20] Meel, K. S., A. A. Shrotri, and M. Y. Vardi. “Not All FPRASs are Equal: Demystifying FPRASs for DNF-Counting”. In: *Proc. of CP*. Invited to Constraints journal. 2018.

[21] Malioutov, D. and K. S. Meel. “MLIC: A MaxSAT-Based framework for learning interpretable classification rules”. In: *Proc. of CP*. 2018.

[22] Sharma, S., R. Gupta, S. Roy, and K. S. Meel. “Knowledge Compilation meets Uniform Sampling”. In: *Proc. of LPAR*. 2018.

[23] Choo, D., M. Soos, K. M. A. Chai, and K. S. Meel. “Bosphorus: Bridging ANF and CNF Solvers”. In: *Proc. of DATE*. 2019.

[24] Gupta, R., S. Sharma, S. Roy, and K. S. Meel. “WAPS: Weighted and Projected Sampling”. In: *Proc. of TACAS*. 2019.

[25] Paredes, R., L. Duenas-Osorio, K. S. Meel, and M. Y. Vardi. “Principled network reliability approximation: A counting-based approach.” In: *Reliability Engineering and System Safety* (2019).

[26] Narodytska, N., A. Shrotri, K. S. Meel, A. Ignatiev, and J. Marques Silva. “Assessing Heuristic Machine Learning Explanations with Model Counting”. In: *Proc. of SAT*. 2019.

[27] Soos, M., R. Kulkarni, and K. S. Meel. “CrystalBall: Gazing in the Black Box of SAT Solving”. In: *Proc. of SAT*. 2019.

[28] Colnet, A. de and K. S. Meel. “Dual Hashing-based Algorithms for Discrete Integration”. In: *Proc. of CP*. 2019.

[29] Baluta, T., S. Shen, S. Shine, K. S. Meel, and P. Saxena. “Quantitative Verification of Neural Networks And its Security Applications”. In: *Proc. of CCS*. 2019.

[30] Ghosh, B. and K. S. Meel. “IMLI: An Incremental Framework for MaxSAT-Based Learning of Interpretable Classification Rules”. In: *Proc. of AIES*. 2019.

[31] Chakraborty, S., K. S. Meel, and M. Y. Vardi. “On the Hardness of Probabilistic Inference Relaxations”. In: *Proc. of AAAI*. 2019.

[32] Chakraborty, S. and K. S. Meel. “On testing of Uniform Samplers”. In: *Proc. of AAAI*. 2019.

[33] Soos, M. and K. S. Meel. “BIRD: Engineering an Efficient CNF-XOR SAT Solver and its Applications to Approximate Model Counting”. In: *Proc. of AAAI*. 2019.

[34] Pote, Y., S. Joshi, and K. S. Meel. “Phase Transition Behavior of Cardinality and XOR Constraints”. In: *Proc. of IJCAI*. 2019.

[35] Sharma, S., S. Roy, M. Soos, and K. S. Meel. “Ganak: A Scalable Probabilistic Exact Model Counter”. In: *Proc. of IJCAI*. 2019.

[36] Meel, K. S., A. A. Shrotri, and M. Y. Vardi. “Not All FPRASs are Equal: Demystifying FPRASs for DNF-Counting (Extended Abstract)”. In: *Proc. of IJCAI*. Sister Conference Best Papers Track. 2019.

[37] Xie, Y., Z. Xu, M. S. Kankanhalli, K. S. Meel, and H. Soh. “Embedding Symbolic Knowledge into Deep Networks”. In: *Proc. of NeurIPS*. 2019.

[38] Dileep, A., K. S. Meel, and A. F. Sabili. “Induction Models on N”. In: *Proc. of LPAR*. 2020.

[39] Ghosh, B., D. Malioutov, and K. S. Meel. “Classification Rules in Relaxed Logical Form”. In: *Proc. of ECAI*. 2020.

[40] Ciampiconi, L., B. Ghosh, J. Scarlett, and K. S. Meel. “A MaxSAT-based Framework for Group Testing”. In: *Proc. of AAAI*. 2020.

[41] Dudek, J. M., D. Fried, and K. S. Meel. “Taming Discrete Integration via the Boon of Dimensionality”. In: *Proc. of NeurIPS*. 2020.

[42] Bhattacharyya, A., S. Gayen, K. S. Meel, and N. V. Vinodchandran. “Efficient Distance Approximation for Structured High-Dimensional Distributions via Learning”. In: *Proc. of NeurIPS*. 2020.

[43] Meel, K. S., Y. Pote, and S. Chakraborty. “On Testing of Samplers”. In: *Proc. of NeurIPS*. 2020.

[44] Gupta, R., S. Roy, and K. S. Meel. “Phase Transition Behavior in Knowledge Compilation”. In: *Proc. of CP*. 2020.

[45] Agrawal, D., Bhavishya, and K. S. Meel. “On the Sparsity of XORs in Approximate Model Counting”. In: *Proc. of SAT*. 2020.

[46] Shaw, A. and K. S. Meel. “Designing New Phase Selection Heuristics”. In: *Proc. of SAT*. 2020.

[47] Baranov, E., A. Legay, and K. S. Meel. “Baital: An Adaptive Weighted Sampling Approach for Improved t-wise Coverage”. In: *Proc. of FSE*. 2020.

[48] Wang, W., M. Usman, A. Almaawi, K. Wang, K. S. Meel, and S. Khurshid. “A Study of Symmetry Breaking Predicates and Model Counting”. In: *Proc. of TACAS*. 2020.

[49] Bendik, J. and K. S. Meel. “Approximate Counting of Minimal Unsatisfiable Subsets”. In: *Proc. of CAV*. 2020.

[50] Golia, P., S. Roy, and K. S. Meel. “Manthan: A Data-Driven Approach for Boolean Function Synthesis”. In: *Proc. of CAV*. 2020.

[51] Soos, M., S. Gocht, and K. S. Meel. “Tinted, Detached, and Lazy CNF-XOR solving and its Applications to Counting and Sampling”. In: *Proc. of CAV*. 2020.

[52] Meel, K. S. and S. Akshay. “Sparse Hashing for Scalable Approximate Model Counting: Theory and Practice”. In: *Proc. of LICS*. 2020.

[53] Pesant, G., K. S. Meel, and M. Mohammadalitajrishi. “ On the Usefulness of Linear Modular Arithmetic in Constraint Programming”. In: *Proc. of CPAIOR*. 2021.

[54] Agrawal, D., Y. Pote, and K. S. Meel. “Partition Function Estimation: A Quantitative Study”. In: *Proc. of IJCAI*. 2021.

[55] Golia, P., S. Roy, and K. S. Meel. “Program Synthesis as Dependency Quantified Formula Modulo Theory”. In: *Proc. of IJCAI*. 2021.

[56] Yang, S., M. Lupascu, and K. S. Meel. “Predicting Forest Fire Using Remote Sensing Data And Machine Learning”. In: *Proc. of AAAI*. 2021.

[57] Lai, Y., K. S. Meel, and R. Yap. “The Power of Literal Equivalence in Model Counting”. In: *Proc. of AAAI*. 2021.

[58] Bendık, J. and K. S. Meel. “Counting Maximal Satisfiable Subsets”. In: *Proc. of AAAI*. 2021.

[59] Bremen, T. van, V. Derkinderen, S. Sharma, S. Roy, and K. S. Meel. “Symmetric Component Caching for Model Counting on Combinatorial Instances”. In: *Proc. of AAAI*. 2021.

[60] Ghosh, B., D. Basu, and K. S. Meel. “Justicia: A Stochastic SAT Approach to Formally Verify Fairness”. In: *Proc. of AAAI*. 2021.

[61] Pote, Y. and K. S. Meel. “Testing Probabilistic Circuits”. In: *Proc. of NeurIPS*. 2021.

[62] Soos, M. and K. S. Meel. “Gaussian Elimination meets Maximum Satisfiability”. In: *Proc. of KR*. 2021.

[63] Golia, P., M. Soos, S. Chakraborty, and K. S. Meel. “Designing Samplers is Easy: The Boon of Testers”. In: *Proc. of FMCAD*. 2021.

[64] Yang, J. and K. S. Meel. “Engineering an Efficient PB-XOR Solver”. In: *Proc. of CP*. 2021.

[65] Prevot, N., M. Soos, and K. S. Meel. “Leveraging GPUs for Effective Clause Sharing in Parallel SAT Solving”. In: *Proc. of SAT*. 2021.

[66] Baluta, T., Z. L. Chua, K. S. Meel, and P. Saxena. “Scalable Quantitative Verification For Deep Neural Networks”. In: *Proc. of ICSE*. 2021.

[67] Bendik, J. and K. S. Meel. “Counting Minimal Unsatisfiable Subsets”. In: *Proc. of CAV*. 2021.

[68] Meel, K. S., N. Vinodchandran, and S. Chakraborty. “Estimating the Size of Union of Sets in Streaming Models”. In: *Proc. of PODS*. 2021.

[69] Pavan, A., N. V. Vinodchandran, A. Bhattacharya, and K. S. Meel. “Model Counting Meets F_0 Estimation”. In: *Proc. of PODS*. 2021.

[70] Golia, P., F. Slivovsky, S. Roy, and K. S. Meel. “Engineering an Efficient Boolean Functional Synthesis Engine”. In: *Proc. of ICCAD*. 2021.

[71] Meel, K. S., A. A. Shrotri, and M. Y. Vardi. “Not All FPRASs are Equal: Demystifying FPRASs for DNF-Counting”. In: *Constraints* (2019).