Report from Dagstuhl Seminar 23241

Scalable Analysis of Probabilistic Models and Programs

Sebastian Junges[*1], Joost-Pieter Katoen[*2], Scott Sanner[*3], Guy Van den Broeck[*4], and Bahare Salmani[†5]

1 **Radboud University Nijmegen, NL.** sebastian.junges@ru.nl
2 **RWTH Aachen, DE.** katoen@cs.rwth-aachen.de
3 **University of Toronto, CA.** ssanner@gmail.com
4 **UCLA, US.** guyvdb@cs.ucla.edu
5 **RWTH Aachen, DE.** salmani@cs.rwth-aachen.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23241 "Scalable Analysis of Probabilistic Models and Programs". The seminar brought together researchers from probabilistic graphical models, verification of probabilistic programming languages, and probabilistic planning. The communities bring vastly different perspectives on the methods and goals of inference under uncertainty. In this seminar, we worked towards a common understanding of how the different angles yield subtle differences in the problem statements and how the different methods provide different strengths and weaknesses. The report describes the different areas, the activities during the seminar including hot topics that were vividly discussed, and an overview of the technical talks.

Seminar June 11–16, 2023 – https://www.dagstuhl.de/23241
2012 ACM Subject Classification Computing methodologies → Planning under uncertainty; Computing methodologies → Probabilistic reasoning; Theory of computation → Automated reasoning
Keywords and phrases model counting, probabilistic inference, probabilistic model checking, probabilistic planning, probabilistic programs
Digital Object Identifier 10.4230/DagRep.13.6.1

1 Executive Summary

Sebastian Junges
Joost-Pieter Katoen
Scott Sanner
Guy Van den Broeck

In this Dagstuhl Seminar, we brought together researchers from three different communities that all bring their own perspective on the role of probabilities in programs and models. To facilitate future collaborations, we saw a need to define common terminology. Therefore, we split this seminar into two parts: On the first two days, we surveyed the research areas (see Chapter 3) and on the second half, we had in-depth sessions. In the first half, we particularly discussed the following exemplary questions:

[*] Editor / Organizer
[†] Editorial Assistant / Collector

- *What are probabilistic circuits and why do they allow tractable inference?*
- *What is probabilistic model checking and what are temporal specifications?*
- *What are probabilistic programs and how are their semantics defined?*
- *How does one reason over probabilistic programs on the source-code level?*

We also spent time to identify common interests in problems, which led to some hot topics mentioned below. The second half of the seminar featured 30-minute technical talks (see Section 4) that provided in-depth discussions on recent research and informal discussions based on the technical talks and the hot topics.

Hot discussion topics. Our discussions led to a list of seven hot topics, summarized below, that were the basis for informal discussions later in the week.

- Unbounded executions in probabilistic programs, their use-cases, analysis techniques, and the downsides.
- Algebraic decision diagrams versus probabilistic circuits and their benefits for reasoning about reachability in graphs.
- The expressiveness and tractability borders between different model types.
- Adequate planning horizons in different scenarios and their influence on the effectiveness of various approaches.
- Inferring symbolic plans and policies via reinforcement learning and with logical constraints, in particular in the context of providing verifiable and explainable controllers.
- The limits of Boolean synthesis in the context of model counting.
- Learning models from data across communities, including perspectives on inference and active automata learning.
- Probabilistic models as distribution transformers and the verification of distributional properties.

In-depth technical sessions. We wanted to highlight the many in-depth discussions that happened mostly 24/7. These discussions were both on the hot topics listed above, as well as very technical, in-depth discussions. We believe that part of the success of this seminar were the various connections that were established on very technical levels. It became clear that the prevalent approaches for very similar problems are vastly different and that there was a common interest to learn about these methodologies.

Challenges. While we are proud of what we achieved, the different backgrounds required a significant effort in order to understand the problems that the different subcommunities find most pressing. As organizers, we would have loved to see time and room to also discuss application areas, but it was hard to find cross-community overlapping interests in those.

2 Table of Contents

3 Introduction

Sebastian Junges, Joost-Pieter Katoen, Scott Sanner, Guy Van den Broeck

Models and model-based reasoning allow reasoning about symbolic knowledge about a system. It is often convenient to represent such systems with probabilistic behavior. Such probabilistic behavior is a natural way to treat uncertainty or to abstract from behavior that appears probabilistic but is a consequence of complex interactions. Reasoning about these systems requires treating the probabilistic aspects as first-class citizens. Famously, humans are bad at reasoning under probabilistic uncertainty and thus the availability of scalable engines that support humans in understanding probabilistic systems and making decisions is essential. Naturally, many disciplines in and outside computer science investigate methods that lead to such engines.

A key challenge in the creation of such reasoning engines is a concise and clear modelling language. Historically many of the tools we had for reasoning and inference about the world were built on top of deterministic programming languages that pose a challenge for the representation of stochastic systems. Probabilistic programming – the notion that programs execute stochastically – and the analysis of such programs have caused a major shift and inference for probabilistic programming languages has already enabled various applications: They steer autonomous robots, are at the heart of the NET-VISA system adopted by the UN to detect seismological activities [2], are used in cognitive science [36], planning in AI, describe security mechanisms and naturally code up randomised algorithms. Probabilistic programming is a rapidly emerging field [18]. For almost all programming languages, there is a probabilistic variant by now, and main industrial players (e.g., Meta and Microsoft) have spent serious research efforts.

A recent trend goes towards thinking about any stochastic model as given by a probabilistic program (PP), which can be thought of as a computational specification of a probability distribution. Probabilistic programs can be used to describe complex distributions and the standard inference task is to understand this distribution. PPs can explicitly represent conditioning as part of a model by syntactic constructs that enable conditioning. PPs with such observation statements involve solving the inverse problem of inferring (the likelihood of) inputs matching a given evidence (aka: observation).

However, the analysis of PPs is and remains hard. Elementary questions such as "does a program terminate almost surely on a given input?" are undecidable. Lifting existing inference techniques to the level of programs is difficult, and reasoning about loops is harder than for classical programs. Techniques to analyse PPs in a symbolic and fully automated manner are currently a vibrant research topic and receive lots of attention in the various research fields, most notably: *probabilistic graphical models*, *probabilistic model checking/program analysis*, as well as *planning in AI*. This Dagstuhl Seminar brought together members of these communities to taxonomize existing research domains and progress in terms of a probabilistic programming lingua franca, identify areas for cross-pollination of ideas across fields, and understand major representational, inferential, and domain-specific challenges that the community (and groups of researchers from this seminar) may collectively tackle beyond the seminar.

The Research Areas

Probabilistic Graphical Models. These models cover Bayesian networks, undirected Markov networks, and extensions thereof dealing with e.g., relational data. They have numerous applications in machine learning, computer vision, natural language processing, and computational biology. Graphical models bring together graph theory and probability theory, and provide a flexible framework for modeling large collections of random variables with complex interactions. Although exact inference is PP-complete in general, efficient algorithms exist for specific structures (e.g., join tree) and dedicated symbolic data structures have been developed to make inference efficient. Parameter and structure learning techniques enable synthesising values in conditional probability tables and full graph topologies. Recent work in exact discrete inference with probabilistic graphical models focuses on tractable models, where marginal probabilities, or the mode of the distribution, can all be computed efficiently in the size of the model. In particular, probabilistic circuits in the form of sum-product networks, arithmetic circuits, and other data structures, have received considerable attention in recent years [13]. Such new probabilistic models provide an opportunity to re-imagine the types of analysis that are possible, for example towards information-theoretic queries [44]. Another important frontier is to discover larger classes of distributions where the probabilistic inference analysis can be performed efficiently [46]. Probabilistic graphical models, either classical ones, or more modern tractable models, are often the target representation of compilers that start with a higher-level language – for example a probabilistic program [17, 24, 35], or even a quantum program [25].

Verification of Probabilistic Models: Model Checking and Beyond. Probabilistic model checking [4, 3, 27] is a verification technique that takes a probabilistic model and a (temporal) specification and asks whether the model satisfies the specification. The (underlying) models are typically Markov chains, Markov decision processes, or stochastic games, state-based models describing how the system evolves over time. These models are pivotal in various disciplines that involve process analysis such as performance evaluation, and sequential decision making, e.g., in reinforcement learning and robotics. To overcome the state-space explosion problem that limits the scalability of PMC algorithms, additional structure must be exploited. This structure can be found in the symbolic descriptions of models and are often defined using programs. The scalability of PMC is then boosted using symbolic data structures, but also clever model reduction techniques [32, 19, 43], and iterative abstraction techniques [22, 28, 6]. Specifications range from the more classical reachability queries and temporal logics to cost-bounded [20, 7], conditional [5] and multi-objective queries [11, 12]. Mature tools such as PRISM [31] and Storm [21] are applicable to finite-state probabilistic programs and are not limited to just the verification question. They can compute how much a specification is satisfied, extract strategies that satisfy the specification, and handle unknown probabilities. Model checkers are used often as back-ends to find plans: in the context of robotics, e.g., in [33] or to synthesise probabilistic programs [14, 1]. Beyond model checking, automated verification techniques for infinite-state Markov models such as bounded model checking, termination analysis (e.g., using deep neural networks), loop analysis, and k-induction have recently been made.

Probabilistic Planning in AI. (Classical) planning aims to find a policy or strategy to solve a decision making process in problems that can range from navigational path planning [34] to supply chain logistics [37] to the management of epidemic outbreaks [45]. Just as in model checking (MC), one is interested in plans that achieve a goal (in MC, to find a bug) or more generally to minimize a cumulative cost function or bound thereon. Contrary to MC, one is

less interested in proving the absence of a solution (or bug). Despite these minor differences, there has been successful cross-fertilization partially initiated at earlier Dagstuhl Seminars. The need for planning under uncertainty has led to probabilistic planning methods that explicitly take this uncertainty into account. These plans (or policies) are typically computed within the Markov Decision Process (MDP) framework, and probabilistic planners are able to successfully find strategies in large MDPs using techniques such as lifting [41, 16] on Monte Carlo Tree Search with dead-end detection [29] as witnessed by International Probabilistic Planning Competitions (IPCs) [15, 42]. The probabilistic planning and probabilistic model checking communities have diverged in their symbolic representations with RDDL [40] being the quasi-standard in probabilistic planning for the object-oriented specification of concurrently evolving stochastic systems (which was in turn inspired by a lifted perspective of dynamic Bayesian networks). Furthermore, the planning community has long embraced partially observable settings as evidenced by partially observed MDP tracks of the IPC [15], whereas such extensions have only recently gained traction in the model checking community. Finally, it is important to note that probabilistic planning typically focuses on a class of algorithms particularly tailored for reachability analysis, namely highly scalable heuristic search techniques and with specialized domain analysis to support them [29].

The Seminar Topics

In *Scalable Analysis of Probabilistic Models and Programs*, the aim is the development of methods, algorithms, and tools that reason in and about probabilistic uncertainty. Such reasoning is interesting in a variety of scientific areas both inside and outside of computer science. But how can we fundamentally boost the reasoning engines and make them more applicable beyond our own communities?

Joint Context. In the planning, and verification communities, program-like descriptions of models have a rich tradition (RDDL [40], Prism [31], Modest [8], JANI [10]). Compared to more general programming languages, these models typically have a variety of tailored but intricate constructs. In recent years, probabilistic programs have been heavily studied as a natural extension of probabilistic graphical models. These probabilistic programs are easier to learn, but a naive user may not obtain the necessary performance out of their inference engines. The holy grail are engines that are efficient in reasoning about easy-to-use probabilistic programs. While goals and perspectives differ across the research communities, there are plenty of similarities in automated analysis techniques in the aforementioned three research fields. For instance, symbolic techniques such as binary decision diagrams (BDDs) and satisfiability solver (SAT/SMT) techniques are commonly used and model counting has made substantial progress in recent years.

Challenges. While (general-purpose) probabilistic programming languages that extend (classical) programming languages are already a success for a significant class of applications, their analysis remains challenging: In particular, we highlight *the presence of discrete variables, conditional program flow, non-deterministic behavior, and unbounded loops.* Within the AI, planning, programming languages and verification community, various efforts aim to improve the analysis of probabilistic programs, all from their own perspective.

Despite this strong link between analysis in probabilistic planning and reachability analysis in PMC, the research in general, and the development of new algorithms, has happened largely independently in each community. However, first cross-fertilizations between verification

and inference [38] and vice versa [23], and probabilistic planning and PMC [9, 30] have been established and show the potential of overcoming research community boundaries. Beyond standard inference, methods still seem orthogonal but may be unified: e.g., probabilistic program sketching approaches [1] seem orthogonal to structure learning techniques [26] for Bayesian networks and parameter synthesis in Markov models have potential for parameter synthesis in graphical models [39].

References

1 Roman Andriushchenko, Milan Ceska, Sebastian Junges, Joost-Pieter Katoen, and Simon Stupinský. PAYNT: A tool for inductive synthesis of probabilistic programs. In *CAV (1)*, volume 12759 of *Lecture Notes in Computer Science*, pages 856–869. Springer, 2021.

2 Nimar S. Arora, Stuart Russell, and Erik Sudderth. NET-VISA: Network processing vertically integrated seismic analysis. *Bulletin of the Seismological Society of America*, 103(2A):709–729, 2013.

3 Christel Baier, Luca de Alfaro, Vojtech Forejt, and Marta Kwiatkowska. Model checking probabilistic systems. In *Handbook of Model Checking*, pages 963–999. Springer, 2018.

4 Christel Baier, Holger Hermanns, and Joost-Pieter Katoen. The 10, 000 facets of MDP model checking. In *Computing and Software Science*, volume 10000 of *Lecture Notes in Computer Science*, pages 420–451. Springer, 2019.

5 Christel Baier, Joachim Klein, Sascha Klüppelholz, and Sascha Wunderlich. Maximizing the conditional expected reward for reaching the goal. In *TACAS (2)*, volume 10206 of *Lecture Notes in Computer Science*, pages 269–285, 2017.

6 Kevin Batz, Sebastian Junges, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Philipp Schröer. Pric3: Property directed reachability for mdps. In *CAV (2)*, volume 12225 of *Lecture Notes in Computer Science*, pages 512–538. Springer, 2020.

7 Frantisek Blahoudek, Tomás Brázdil, Petr Novotný, Melkior Ornik, Pranay Thangeda, and Ufuk Topcu. Qualitative controller synthesis for consumption Markov decision processes. In *CAV (2)*, volume 12225 of *Lecture Notes in Computer Science*, pages 421–447. Springer, 2020.

8 Henrik C. Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. MODEST: A compositional modeling formalism for hard and softly timed systems. *IEEE Trans. Software Eng.*, 32(10):812–830, 2006.

9 Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Vojtech Forejt, Jan Kretínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Verification of Markov decision processes using learning algorithms. In *ATVA*, volume 8837 of *Lecture Notes in Computer Science*, pages 98–114. Springer, 2014.

10 Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. JANI: quantitative model and tool interaction. In *TACAS (2)*, volume 10206 of *Lecture Notes in Computer Science*, pages 151–168, 2017.

11 Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov decision processes with multiple objectives. In *STACS*, volume 3884 of *Lecture Notes in Computer Science*, pages 325–336. Springer, 2006.

12 Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3-4):129–163, 2014.

13 YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. oct 2020.

14 Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. Profeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Aspects Comput.*, 30(1):45–75, 2018.

15 A. Coles, A. Coles, A. García Olaya, S. Jiménez, C. Linares López, S. Sanner, and S. Yoon. A survey of the seventh international planning competition. *Artificial Intelligence Magazine (AI Magazine)*, 33(1):83–88, 2012.

16 Hao Cui, Thomas Keller, and Roni Khardon. Stochastic planning with lifted symbolic trajectory optimization. In *ICAPS*, pages 119–127. AAAI Press, 2019.

17 Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 15:358–401, 5 2015.

18 Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. In *FOSE*, pages 167–181. ACM, 2014.

19 Henri Hansen, Marta Z. Kwiatkowska, and Hongyang Qu. Partial order reduction for model checking Markov decision processes under unconditional fairness. In *QEST*, pages 203–212. IEEE Computer Society, 2011.

20 Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Multi-cost bounded tradeoff analysis in MDP. *J. Autom. Reason.*, 64(7):1483–1522, 2020.

21 Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *CoRR*, abs/2002.07080, 2020.

22 Holger Hermanns, Björn Wachter, and Lijun Zhang. Probabilistic CEGAR. In *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 162–175. Springer, 2008.

23 Steven Holtzen, Sebastian Junges, Marcell Vazquez-Chanlatte, Todd D. Millstein, Sanjit A. Seshia, and Guy Van den Broeck. Model checking finite-horizon Markov chains with probabilistic inference. In *CAV (2)*, volume 12760 of *Lecture Notes in Computer Science*, pages 577–601. Springer, 2021.

24 Steven Holtzen, Guy Van den Broeck, and Todd Millstein. Scaling exact inference for discrete probabilistic programs. *Proc. ACM Program. Lang. (OOPSLA)*, nov 2020.

25 Yipeng Huang, Steven Holtzen, Todd Millstein, Guy Van den Broeck, and Margaret R. Martonosi. Logical abstractions for noisy variational quantum algorithm simulation. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2021.

26 Tommi S. Jaakkola, David A. Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *AISTATS*, volume 9 of *JMLR Proceedings*, pages 358–365. JMLR.org, 2010.

27 Joost-Pieter Katoen. The probabilistic model checking landscape. In *LICS*, pages 31–45. ACM, 2016.

28 Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. A game-based abstraction-refinement framework for Markov decision processes. *Formal Methods Syst. Des.*, 36(3):246–280, 2010.

29 Thomas Keller and Patrick Eyerich. PROST: Probabilistic planning based on UCT. In *International Conference on Automated Planning and Scheduling*, pages 119–127, 2012.

30 Michaela Klauck, Marcel Steinmetz, Jörg Hoffmann, and Holger Hermanns. Bridging the gap between probabilistic model checking and probabilistic planning: Survey, compilations, and empirical comparison. *J. Artif. Intell. Res.*, 68:247–310, 2020.

31 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

32 Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Assume-guarantee verification for probabilistic systems. In *TACAS*, volume 6015 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 2010.

33 Bruno Lacerda, Fatma Faruq, David Parker, and Nick Hawes. Probabilistic planning with formal performance guarantees for mobile service robots. *Int. J. Robotics Res.*, 38(9), 2019.

34 Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic A*: An anytime, replanning algorithm. In *Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'05, page 262–271. AAAI Press, 2005.

35 Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. *Advances in Neural Information Processing Systems*, 22:1249–1257, 2009.

36 Desmond C. Ong, Harold Soh, Jamil Zaki, and Noah D. Goodman. Applying probabilistic programming to affective computing. *IEEE Trans. Affect. Comput.*, 12(2):306–317, 2021.

37 David Pardoe and Peter Stone. Predictive planning for supply chain management. In *ICAPS*, pages 21–30. AAAI, 2006.

38 Bahare Salmani and Joost-Pieter Katoen. Bayesian inference by symbolic model checking. In *QEST*, volume 12289 of *Lecture Notes in Computer Science*, pages 115–133. Springer, 2020.

39 Bahare Salmani and Joost-Pieter Katoen. Fine-tuning the odds in Bayesian networks. In *ECSQARU*, volume 12897 of *Lecture Notes in Computer Science*, pages 268–283. Springer, 2021.

40 S. Sanner. Relational Dynamic Influence Diagram Language (RDDL): Language description. Unpublished Manuscript, Australian National University, 2010.

41 S. Sanner and C. Boutilier. Practical solution techniques for first-order MDPs. *Artificial Intelligence Journal (AIJ)*, pages 748–788, April 2009. Recipient of the 2014 Artificial Intelligence Journal (AIJ) Prominent Paper Award.

42 M. Vallati, L. Chrpa, M. Grzes, T. L. McCluskey, M. Roberts, and S. Sanner. The 2014 international planning competition: Progress and trends. *Artificial Intelligence Magazine (AI Magazine)*, 36(3):90–98, 2015.

43 Tom van Dijk and Jaco van de Pol. Multi-core symbolic bisimulation minimisation. *Int. J. Softw. Tools Technol. Transf.*, 20(2):157–177, 2018.

44 Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, dec 2021.

45 Shan Xue. *Scheduling and Online Planning in Stochastic Diffusion Networks*. PhD thesis, Oregon State University, 2020.

46 Honghua Zhang, Brendan Juba, and Guy Van den Broeck. Probabilistic generating circuits. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, jul 2021.

4 Overview of Talks

Below, we provide abstracts of the technical talks presented at the seminar, ordered lexicographically by the presenter.

4.1 Markov Decision Processes as Distribution Transformers: Decidability and Strategy Synthesis for Safety Objectives

S. Akshay (Indian Institute of Technology Bombay – Mumbai, IN)

Joint work of S. Akshay, Krishnendu Chatterjee, Tobias Meggendorfer, Djordje Zikelic

Markov decision processes can be viewed as transformers of probability distributions. This view is useful to reason about trajectories of distributions, but even basic reachability and safety problems turn out to be computationally intractable (Skolem-hard!). The issue is further complicated by the question of how much memory is allowed: even for simple examples, strategies for safety objectives over distributions can require infinite memory and randomization.

In light of this, we ask what one can do to tackle these problems in theory and in practice. After taking a look at some theoretical insights, we adopt an over-approximation route to approach these questions. Inspired by the success of invariant synthesis in program verification, we develop a framework for template-based synthesis of certificates as affine distributional and inductive invariants for safety objectives in MDPs. We show the effectiveness of our approach as well as explore limitations and future perspectives.

4.2 A Framework for Transforming Specifications in Reinforcement Learning

Suguman Bansal (Georgia Institute of Technology – Atlanta, US)

Joint work of Suguman Bansal, Rajeev Alur, Kishor Jothimurugan, Osbert Bastani
Main reference Rajeev Alur, Suguman Bansal, Osbert Bastani, Kishor Jothimurugan: "A Framework for Transforming Specifications in Reinforcement Learning", in Proc. of the Principles of Systems Design – Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday, Lecture Notes in Computer Science, Vol. 13660, pp. 604–624, Springer, 2022.
URL https://doi.org//10.1007/978-3-031-22337-2_29

Reinforcement Learning (RL) algorithms are designed to learn an optimal policy when the transition probabilities of the MDP are unknown but require the user to associate local rewards with transitions. The appeal of high-level temporal logic specifications has motivated research to develop RL algorithms for the synthesis of policies from specifications. To understand the techniques, and nuanced variations in their theoretical guarantees, in the growing body of resulting literature, we develop a formal framework for defining transformations among RL tasks with different forms of objectives. We define the notion of sampling-based reduction to transform a given MDP into another one that can be simulated even when the transition probabilities of the original MDP are unknown. We formalize the notions of preservation of optimal policies, convergence, and robustness of such reductions. We then use our framework

to restate known results, establish new results to fill in some gaps, and identify open problems. In particular, we show that certain kinds of reductions from LTL specifications to reward-based ones do not exist, and prove the non-existence of RL algorithms with PAC-MDP guarantees for safety specifications.

4.3 Approximate Weighted Model Counting using Universal Hashing

Supratik Chakraborty (Indian Institute of Technology Bombay – Mumbai, IN)

Joint work of Supratik Chakraborty, Daniel Fremont, Kuldeep Meel, Sanjit Seshia, Moshe Vardi
Main reference Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, Moshe Y. Vardi: "Distribution-Aware Sampling and Weighted Model Counting for SAT", in Proc. of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada, pp. 1722–1730, AAAI Press, 2014.
URL https://doi.org//10.1609/aaai.v28i1.8990

Given a system of propositional constraints, (unweighted) model counting concerns counting the number of satisfying assignments of the constraints. If every assignment is associated with a non-negative weight, the problem of finding the total weight of all satisfying assignments is called weighted model counting. This is a fundamental problem in computer science, with diverse applications in probabilistic inference, quantitative information flow, and partition function estimation among others. Unfortunately, model counting (even the unweighted variant) is computationally intractable – Valiant showed that this is #P-complete. Hence, it is unlikely that efficient algorithms exist for solving this problem. This has spurred a lot of interest in approximate weighted model counting. While there has been a lot of theoretical work in this domain that has yielded algorithms with strong guarantees of approximation, and also a lot of work that uses heuristics to scale to large problem instances without providing strong approximation guarantees, marrying scalability in practice with strong approximation guarantees is significantly hard. In this talk, we present a technique based on universal hash functions for solving the weighted model counting with PAC-style guarantees, yielding an approximate counter that scales well to large problem instances. The core idea of our technique is to partition the set of all assignments into cells of "small enough" and "almost equal" weights using universal hash functions, and then to count the total weight of solutions in a randomly chosen cell. The resulting weight is then multiplied by the total number of cells in the partition to obtain an estimate of the overall weighted model count.

We define a parameter called the "tilt" of weights of assignments, and show how the above idea leads to an algorithm that makes polynomially (in tilt, number of variables, and PAC approximation parameters) many calls to an NP oracle to yield an estimate of the weighted model count with PAC guarantees. Experiments show that this algorithm scales very well in practice when the tilt is bounded by a small constant. We then discuss an extension of our algorithm to deal with problem instances where the tilt may be large, and where assignment weights are the product of literal weights. The extended algorithm requires solving linear (in the number of variables) pseudo-boolean constraints. In practice, pseudo-boolean satisfiability solvers (including those that reduce to propositional satisfiability) are not as efficient in practice as propositional satisfiability solvers on large problem instances. Therefore, weighted model counting using universal hashing doesn't scale as well in practice when the tilt of weights is large, compared to the case of small tilt. Future advances in pseudo-boolean satisfiability solving are likely to directly impact the ability of weighted model counters to solve problem instances with large tilt.

4.4 Tractable Inference with Probabilistic Circuits

YooJung Choi (Arizona State University – Tempe, US)

Probabilistic circuits (PCs) are a family of models that guarantee exact inference of various probabilistic queries in polynomial (often linear) time. In this talk, we introduce the syntax and semantics of probabilistic circuits and study the structural properties that enable linear-time inference of marginal and MAP queries. We then discuss how we can perform inference on other probabilistic models such as Bayesian networks and probabilistic programs by compiling to circuits, in particular by reducing probabilistic inference to the task of weighted model counting/integration which can be performed tractably on certain circuits. Lastly, we showcase some recent works in complex reasoning using PCs. For instance, by representing queries as pipelines of atomic circuit operations, we show how we can systematically derive tractability conditions and inference algorithms for various information-theoretic entropies and divergences. This talk is based on the joint tutorial with Antonio Vergari, Robert Peharz, and Guy Van den Broeck.

4.5 Bit Blasting Probabilistic Programs

Poorva Garg (UCLA, US), Steven Holtzen (Northeastern University – Boston, US), and Guy Van den Broeck (UCLA, US)

Joint work of Poorva Garg, Steven Holtzen, Todd Millstein, Guy Van den Broeck

Probabilistic programming languages (PPLs) have emerged as a prominent area of research due to their ability to democratize probabilistic modeling. One of the key tasks in building a PPL is to design a generalizable probabilistic inference algorithm. Weighted model counting (WMC) is a popular exact inference algorithm for discrete probabilistic programs with much success. Can we extend the advantages of WMC to a wider class of probabilistic programs with both discrete and continuous distributions? Discretization of continuous distribution seems to be an obvious choice. However, it either leads to exhaustive enumeration or compromises on accuracy. LexBit (Language for Exact Bit blasting) is a non-trivial core language, with discrete and continuous constructs, that does not suffer from the limitations of discretization. It bit blasts exactly and scalably. We bit blast the continuous distributions outside this language using linear piece-wise distributions. Once all the continuous distributions in the probabilistic program are bit blasted, we harness the power of existing discrete PPLs to perform exact inference on the new discrete probabilistic program. Case studies and experiments on existing benchmarks show that this approach of bit blasting is competitive with existing probabilistic inference algorithms.

4.6 Compositional Probabilistic Model Checking with String Diagrams of MDPs

Ichiro Hasuo (National Institute of Informatics – Tokyo, JP)

Joint work of Kazuki Watanabe, Clovis Eberhart, Kazuyuki Asada, Ichiro Hasuo
Main reference Kazuki Watanabe, Clovis Eberhart, Kazuyuki Asada, Ichiro Hasuo: "Compositional Probabilistic Model Checking with String Diagrams of MDPs", in Proc. of the Computer Aided Verification – 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part III, Lecture Notes in Computer Science, Vol. 13966, pp. 40–61, Springer, 2023.
URL https://doi.org//10.1007/978-3-031-37709-9_3

We present a compositional model checking algorithm for Markov decision processes, in which they are composed in the categorical graphical language of string diagrams. The algorithm computes optimal expected rewards. Our theoretical development of the algorithm is supported by category theory, while what we call decomposition equalities for expected rewards act as a key enabler. Experimental evaluation demonstrates its performance advantages.

4.7 Introduction to Probabilistic Programming Inference

Steven Holtzen (Northeastern University – Boston, US)

URL https://www.khoury.northeastern.edu/home/sholtzen/CS7480Fall21/

How do we effectively run probabilistic programs in order to reason automatically about their behavior? In particular, how do we efficiently execute them in order to compute the probability that the program will have a particular behavior as efficiently as possible? In this talk, we go over the foundations of probabilistic program semantics and inference. We built a simple probabilistic programming language from scratch and described how to run it in order to evaluate queries. This talk was based on the introduction to a course taught on probabilistic programming at Northeastern University; the link is in the description.

4.8 Intelligent and Dependable Decision-Making Under Uncertainty

Nils Jansen (Radboud University Nijmegen, NL)

Main reference Nils Jansen: "Intelligent and Dependable Decision-Making Under Uncertainty", in Proc. of the Formal Methods – 25th International Symposium, FM 2023, Lübeck, Germany, March 6-10, 2023, Proceedings, Lecture Notes in Computer Science, Vol. 14000, pp. 26–36, Springer, 2023.
URL https://doi.org//10.1007/978-3-031-27481-7_3

This talk highlights our vision of foundational and application-driven research toward safety and dependability in artificial intelligence (AI). We take a broad stance on AI that combines formal methods, machine learning, and control theory. As part of this research line, we study problems inspired by autonomous systems, planning in robotics, and industrial applications.

We consider reinforcement learning (RL) as a specific machine learning technique for decision-making under uncertainty. RL generally learns to behave optimally via trial and error. Consequently, and despite its massive success in the past years, RL lacks mechanisms

to ensure safe and correct behavior. Formal methods, in particular formal verification, is a research area that provides formal guarantees of a system's correctness and safety based on rigorous methods and precise specifications. Yet, fundamental challenges have obstructed the effective application of verification to reinforcement learning. Our main objective is to devise novel, data-driven verification methods that tightly integrate with RL. In particular, we develop techniques that address real-world challenges to the safety of AI systems in general: Scalability, expressiveness, and robustness against the uncertainty that occurs when operating in the real world. The overall goal is to advance the real-world deployment of reinforcement learning.

The talk is mainly based on the following references: [1, 2, 3, 4, 5].

References

1 Nils Jansen. Intelligent and dependable decision-making under uncertainty. In *FM*, volume 14000 of *Lecture Notes in Computer Science*, pages 26–36. Springer, 2023.
2 Thom S. Badings, Thiago D. Simão, Marnix Suilen, and Nils Jansen. Decision-making under uncertainty: beyond probabilities. *Int. J. Softw. Tools Technol. Transf.*, 25(3):375–391, 2023.
3 Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. Safe reinforcement learning via shielding under partial observability. In *AAAI*, pages 14748–14756. AAAI Press, 2023.
4 Thom S. Badings, Licio Romao, Alessandro Abate, David Parker, Hasan A. Poonawala, Mariëlle Stoelinga, and Nils Jansen. Robust control for dynamical systems with non-gaussian noise via formal abstractions. *J. Artif. Intell. Res.*, 76:341–391, 2023.
5 Steven Carr, Nils Jansen, and Ufuk Topcu. Task-aware verifiable rnn-based policies for partially observable markov decision processes. *J. Artif. Intell. Res.*, 72:819–847, 2021.

4.9 Deductive Verification of Probabilistic Programs: Loops and Recursion

Joost-Pieter Katoen (RWTH Aachen, DE)

Probabilistic programs describe recipes on how to infer conclusions about big data from a mixture of uncertain data and real-world observations. Bayesian networks, a key model in decision-making, are simple instances of such programs. Probabilistic programs are used to steer autonomous robots and self-driving cars, are key to describe security mechanisms, and naturally encode randomised algorithms. Due to their learning ability, they are rapidly encroaching on AI and probabilistic machine learning.

This talk focuses on syntax-based verification of discrete probabilistic programs. We will show how weakest pre-condition style reasoning can be used to determine quantitative program properties such as the probability of divergence, bounds on the expected outcomes of program expressions, or the program's expected run-time. Complementary to Holtzen's talk on straight-line code, we focus primarily on how to treat possibly unbounded loops and recursion.

We will present automated methods such as k-induction and how to find loop invariants in a CEGIS-like fashion. An outlook will be given of some alternative automated techniques for program equivalence and how to exploit model checking for obtaining lower bounds on loops in probabilistic programs.

4.10 Scalable Learning of Probabilistic Circuits

Anji Liu (UCLA, US)

Joint work of Anji Liu, Guy Van den Broeck, Honghua Zhang, Antonio Vergari, YooJung Choi, Stephan Mandt
Main reference Anji Liu, Honghua Zhang, Guy Van den Broeck: "Scaling Up Probabilistic Circuits by Latent Variable Distillation", in Proc. of the The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023, OpenReview.net, 2023.
URL https://openreview.net/pdf?id=067CGykiZTS

Probabilistic Circuits (PCs) are a unified framework for tractable probabilistic models that support efficient computation of various probabilistic queries (e.g., marginal probabilities). One key challenge is to scale PCs to model large and high-dimensional real-world datasets: we observe that as the number of parameters in PCs increases, their performance immediately plateaus. This phenomenon suggests that the existing optimizers fail to exploit the full expressive power of large PCs. We propose to overcome such bottleneck by latent variable distillation: we leverage the less tractable but more expressive deep generative models to provide extra supervision over the latent variables of PCs. Specifically, we extract information from Transformer-based generative models to assign values to latent variables of PCs, providing guidance to PC optimizers. Experiments on both image and language modeling benchmarks (e.g., ImageNet and WikiText-2) show that latent variable distillation substantially boosts the performance of large PCs compared to their counterparts without latent variable distillation. In particular, on the image modeling benchmarks, PCs achieve competitive performance against some of the widely-used deep generative models, including variational autoencoders and flow-based models, opening up new avenues for tractable generative modeling.

4.11 How to Make Logics Neurosymbolic

Luc De Raedt (KU Leuven, BE)

Joint work of Luc De Raedt, Giuseppe Marra, Robin Manhaeve, Thomas Winters, Vincent Derkinderen, Sebastijan Dumancic
Main reference Giuseppe Marra, Sebastijan Dumancic, Robin Manhaeve, Luc De Raedt: "From Statistical Relational to Neural Symbolic Artificial Intelligence: a Survey", CoRR, Vol. abs/2108.11451, 2021.
URL https://arxiv.org/abs/2108.11451

Neurosymbolic AI (NeSy) is regarded as the third wave in AI. It aims at combining knowledge representation and reasoning with neural networks. Numerous approaches to NeSy are being developed and there exists an "alphabet soup" of different systems, whose relationships are often unclear. I will discuss the state-of-the-art in NeSy and argue that there are many similarities with statistical relational AI (StarAI).
Taking inspiration from StarAI, and exploiting these similarities, I will argue that Neurosymbolic AI = Logic + Probability + Neural Networks. I will also provide a recipe for developing NeSy approaches: start from a logic, add a probabilistic interpretation, and then turn neural networks into "neural predicates". Probability is interpreted broadly here and is necessary to provide a quantitative and differentiable component to the logic. At the semantic and the computation level, one can then combine logical circuits (ako proof structures) labeled with probability, and neural networks in computation graphs.

I will illustrate the recipe with NeSy systems such as DeepProbLog, a deep probabilistic extension of Prolog, and DeepStochLog, a neural network extension of stochastic definite clause grammars (or stochastic logic programs).

The key references of the talk are as follows: [1, 2, 3].

References

1 Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*, pages 3753–3763, 2018.

2 Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. Deepstochlog: Neural stochastic logic programming. In *AAAI*, pages 10090–10100. AAAI Press, 2022.

3 Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. From statistical relational to neural symbolic artificial intelligence: a survey. *arXiv preprint arXiv:2108.11451*, 2021.

4.12 Tutorial: Probabilistic Model Checking

David Parker (University of Oxford, GB)

Probabilistic model checking is an automated technique for the formal verification of stochastic systems. This tutorial will provide an introduction to some of the key ingredients of this technique, giving a particular focus on the similarities and differences with some of the other fields represented at the seminar, such as probabilistic programming, probabilistic circuits and, probabilistic planning.
I will cover: (i) the types of probabilistic models typically used; (ii) the use of temporal logic to formalise quantitative behavioural specifications, in particular for models such as Markov chains and Markov decision processes; (iii) the solution techniques usually used by probabilistic model checking tools, and the approaches taken to improve scalability and efficiency; and (iv) modelling languages for probabilistic verification. In the final part of the talk, I will discuss how this framework has been extended to support multi-agent systems modelled as stochastic games.

4.13 Mixing formal methods and learning to tackle (too) large MDPs

Jean-Francois Raskin (UL – Brussels, BE)

In a recent series of works, we investigate optimizing strategies in Markov decision processes (MDPs) using Monte Carlo Tree Search (MCTS). We introduce symbolic advice to enhance MCTS, biasing its selection and simulation strategies while maintaining its theoretical guarantees. Efficient implementation of symbolic advice is achieved using QBF and SAT solvers. Additionally, we integrate formal methods and deep learning to produce superior receding horizon policies in large MDPs. Model-checking techniques guide MCTS for high-quality decision sampling, which subsequently trains a neural network to imitate the sampled

policy. This network can guide low-latency MCTS online searches or serve as an independent policy when quick responses are vital. Statistical model checking identifies areas needing extra samples and highlights discrepancies between the neural network and the offline policy. Our methodologies are validated using the Pac-Man and Frozen Lake environments, benchmarks in evaluating reinforcement-learning algorithms. The results outperform standard MCTS and human players.
The main related papers to the talk are as follows: [1, 2, 3].

References

1 Debraj Chakraborty, Damien Busatto-Gaston, Jean-François Raskin, and Guillermo A. Pérez. Formally-sharp dagger for MCTS: lower-latency monte carlo tree search using data aggregation with formal methods. In Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh, editors, *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 – 2 June 2023*, pages 1354–1362. ACM, 2023.

2 Damien Busatto-Gaston, Debraj Chakraborty, and Jean-François Raskin. Monte carlo tree search guided by symbolic advice for mdps. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 40:1–40:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

3 Gilles Geeraerts, Shibashis Guha, and Jean-François Raskin. Safe and optimal scheduling for hard and soft tasks. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 36:1–36:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

4.14 Automatically Finding the Right Probabilities in Bayesian Networks

Bahare Salmani (RWTH Aachen, DE) and Joost-Pieter Katoen (RWTH Aachen, DE)

Main reference Bahare Salmani, Joost-Pieter Katoen: "Automatically Finding the Right Probabilities in Bayesian Networks", J. Artif. Intell. Res., Vol. 77, pp. 1637–1696, 2023.
URL https://doi.org//10.1613/jair.1.14044

Parametric Bayesian networks (pBNs) are extensions of Bayesian networks that allow conditional probability tables (CPTs) to include unknown parameters rather than concrete probabilities. We present in this talk alternative techniques to find the correct values for the parameters with respect to a given constraint. The key is to translate (a) pBNs to parametric Markov chains (pMCs) and (b) pBN constraints to reachability constraints. This allows exploiting the state-of-the-art parameter synthesis techniques for pMCs to target pBN problems including sensitivity analysis, (minimal-change) parameter tuning, and parameter space partitioning. We address pBNs with an arbitrary number of parameterized CPTs and with arbitrary dependencies between the parameters. This lifts the limitations of the existing pBN techniques. Our experimental results indicate that our techniques scale up to 800 unknown parameters for large Bayesian networks with ~ 100 random variables.

4.15 Tutorial on Planning with Probabilistic Programming Languages

Scott Sanner (University of Toronto, CA)

Planning aims to find sequences of actions that achieve a goal or optimize a cost-based objective given an initial starting state. Modern planning methods seek to leverage the structure in symbolic domain specification languages to improve the efficiency of the search process. The Relational Dynamic Influence Diagram Language (RDDL) is a probabilistic programming language that has been developed to compactly model real-world stochastic planning problems, i.e., Markov Decision Processes (MDPs), and specifically factored MDPs with highly structured transition and reward functions. In this tutorial, we will cover the basics of RDDL and present recent language extensions and capabilities through the incremental development and extension of running examples based on real-world domains. We will also introduce a range of planning methodologies that leverage RDDL structure covering Monte Carlo Tree Search (MCTS), mathematical programming, gradient-based optimization, and symbolic methods.

4.16 Deterministic stream-sampling for probabilistic programming: semantics and verification

Alexandra Silva (Cornell University – Ithaca, US)

Joint work of Fredrik Dahlqvist, Alexandra Silva, William Smith
Main reference Fredrik Dahlqvist, Alexandra Silva, William Smith: "Deterministic stream-sampling for probabilistic programming: semantics and verification", in Proc. of the LICS, pp. 1–13, 2023.
URL https://doi.org//10.1109/LICS56636.2023.10175773

Probabilistic programming languages rely fundamentally on some notion of sampling, and this is doubly true for probabilistic programming languages which perform Bayesian inference using Monte Carlo techniques. Verifying samplers – proving that they generate samples from the correct distribution – is crucial to the use of probabilistic programming languages for statistical modelling and inference. However, the typical denotational semantics of probabilistic programs is incompatible with deterministic notions of sampling. This is problematic, considering that most statistical inference is performed using pseudorandom number generators. We present a higher-order probabilistic programming language centred on the notion of samplers and sampler operations. We give this language an operational and denotational semantics in terms of continuous maps between topological spaces. Our language also supports discontinuous operations, such as comparisons between reals, by using the type system to track discontinuities. This feature might be of independent interest, for example in the context of differentiable programming. Using this language, we develop tools for the formal verification of sampler correctness. We present an equational calculus to reason about equivalence of samplers, and a sound calculus to prove semantic correctness of samplers, i.e. that a sampler correctly targets a given measure by construction.

4.17 E-MCTS: Deep Exploration in Model-Based Reinforcement Learning by Planning with Epistemic Uncertainty

Matthijs Spaan (TU Delft, NL)

Joint work of Matthijs Spaan, Wendelin Böhmer, Yaniv Oren
Main reference Yaniv Oren, Matthijs T. J. Spaan, Wendelin Böhmer: "Planning with Uncertainty: Deep Exploration in Model-Based Reinforcement Learning", CoRR, Vol. abs/2210.13455, 2022.
URL https://doi.org//10.48550/arXiv.2210.13455

One of the most well-studied and highly performing planning approaches used in Model-Based Reinforcement Learning (MBRL) is Monte-Carlo Tree Search (MCTS). Key challenges of MCTS-based MBRL methods remain dedicated deep exploration and reliability in the face of the unknown, and both challenges can be alleviated through principled epistemic uncertainty estimation in the predictions of MCTS. We present two main contributions: First, we develop methodology to propagate epistemic uncertainty in MCTS, enabling agents to estimate the epistemic uncertainty in their predictions. Second, we utilize the propagated uncertainty for a novel deep exploration algorithm by explicitly planning to explore. We incorporate our approach into variations of MCTS-based MBRL approaches with learned and provided models, and empirically show deep exploration through successful epistemic uncertainty estimation achieved by our approach. We compare to a non-planning-based deep-exploration baseline, and demonstrate that planning with epistemic MCTS significantly outperforms non-planning based exploration in the investigated setting.

Participants

- S. Akshay
Indian Institute of Technology Bombay – Mumbai, IN
- Suguman Bansal
Georgia Institute of Technology – Atlanta, US
- Kevin Batz
RWTH Aachen, DE
- Milan Ceska
Brno University of Technology, CZ
- Supratik Chakraborty
Indian Institute of Technology Bombay – Mumbai, IN
- YooJung Choi
Arizona State University – Tempe, US
- Cassio de Campos
TU Eindhoven, NL
- Luc De Raedt
KU Leuven, BE
- Rayna Dimitrova
CISPA – Saarbrücken, DE
- Poorva Garg
UCLA, US
- Vibhav Gogate
University of Texas at Dallas – Richardson, US
- Ichiro Hasuo
National Institute of Informatics – Tokyo, JP
- Holger Hermanns
Universität des Saarlandes – Saarbrücken, DE
- Steven Holtzen
Northeastern University – Boston, US
- Manfred Jaeger
Aalborg University, DK
- Nils Jansen
Radboud University Nijmegen, NL
- Sebastian Junges
Radboud University Nijmegen, NL
- Amir Kafshdar Goharshady
HKUST – Kowloon, HK
- Benjamin Kaminski
Universität des Saarlandes – Saarbrücken, DE
- Joost-Pieter Katoen
RWTH Aachen, DE
- Samuel Kolb
KU Leuven, BE
- John Li
Northeastern University – Boston, US
- Anji Liu
UCLA, US
- Christoph Matheja
Technical University of Denmark – Lyngby, DK
- Chih-Hao Luke Ong
Nanyang TU – Singapore, SG
- David Parker
University of Oxford, GB
- Jean-Francois Raskin
UL – Brussels, BE
- Jurriaan Rot
Radboud University Nijmegen, NL
- Bahare Salmani Barzorki
RWTH Aachen, DE
- Scott Sanner
University of Toronto, CA
- Alexandra Silva
Cornell University – Ithaca, US
- Matthijs Spaan
TU Delft, NL
- Guy Van den Broeck
UCLA, US
- Antonio Vergari
University of Edinburgh, GB