Abstraction in First-Order Probabilistic Inference (500000)

Paulius Dilkas

9th March 2019

1 Introduction

Logical approaches to reasoning have dominated the fields of artificial intelligence (AI) and computing for many decades. They have resulted in expert systems that can use reasoning to make predictions and diagnose problems [19], logic programming languages that allow the user to declaratively describe the problem and trust the inference algorithm to find the answer efficiently [24], and automated (interactive) theorem proving and proof checking software that assists mathematicians in constructing correct proofs [31]. Probabilistic methods, on the other hand, have particularly flourished with the arrival of big data (and access to more data in general), transforming areas such as natural language processing and pattern recognition [9].

While probabilistic models are great at handling uncertainty, their simplistic representations can be hard to interpret. On the other hand, systems based on logic have rich representations, but cannot handle uncertainty. *First-order probabilistic inference* (FOPI) (also known as *statistical relational AI*) attempts to bridge the gap between the two and suggests a range of representations capable of handling probabilities as well as (parts of) first-order logic [9]. The models can be constructed manually or learned from data, and the process of computing the probability of a query—usually in the form of a random variable, possibly conditioned on other random variables—is called *inference*.

Most of these models are based either on adding probabilities to programming languages or on adding richer representational structure to probabilistic graphical models (PGMs). The former kind is called *probabilistic programming* [18] and is an active area of research with many implementations. A well-known example is *ProbLog* [29]—a language that extends the logic programming language Prolog by attaching a probability to every clause in the program. A prominent example of the latter is a *Markov logic network* (MLN) [30], which is simply a collection of first-order statements (formulas), with a weight attached to each statement.

In the last decade, we have seen applications of FOPI in a wide range of areas, ranging from toy problems with probabilities one might find in a textbook [12] all the way to genetics [34] and cancer research [8]. For instance, FOPI models have been integrated into recommendation systems [39] and stream mining software [4], and used to predict the remaining lifetime of hardware components [38] as well as patterns of criminal and terrorist activity [10].

1.1 Key Idea

Abstraction can be broadly defined as the process (and result) of omitting detail [23]. Sometimes the omitted information is irrelevant in answering the questions we are interested in, and sometimes an abstraction provides a simplified (and approximately correct) view of a situation that originally was too complex to be reasoned about. While areas such as planning and verification have benefited from abstraction in various ways [33], research into abstraction in FOPI has just begun and awaits significant contributions [2, 21, 22].

Our goal is to develop new types of abstractions, find efficient algorithms for constructing an abstraction from a given model, and investigate how abstraction can be integrated into both inference and learning algorithms. Simplification and abstraction can benefit us in both efficiency and interpretability, i.e, simpler models are likely to result in faster inference, while at the same time being easier to understand by the user. Finally, the quest for abstraction algorithms is likely to lead to a better theoretical understanding of what properties can be preserved by an abstraction, what error bounds can be established when abstraction approximates the answer, and upper and lower bounds on the complexity of performing abstraction and providing the desired guarantees.

2 State of the Art

2.1 Inference

Recall that a ProbLog program is a set of clauses, where each clause has an associated probability. The ProbLog inference rule [28, 35] for calculating the probability of an arbitrary query Q being true is

$$P(Q) = \sum_{F \models Q} \prod_{f \in F} P(f) \prod_{f \notin F} 1 - P(f).$$

Here, we are summing over all instantiations of variables (called *possible worlds*) that satisfy the query, where F denotes a set of clauses that are evaluated as true. For each world, we calculate its probability by multiplying probabilities associated with clauses or their negations. With this definition, the inference problem becomes an instance of weighted model counting [28].

Weighted model counting (WMC) is an extension of model counting, which is an extension of the *Boolean satisfiability problem* (SAT) [5]. SAT asks whether one can assign values to variables so that a given formula evaluates to true. Model counting asks to count the number of ways that can be done. Weighted model counting further extends this problem by assigning a weight to each possible world (in whatever way is appropriate for the problem) and asks for the sum of the weights corresponding to all possible worlds where the formula (or query) is true. In the case of ProbLog, the weight of a world is the product of probabilities of all literals (whether evaluated/instantiated to true or false) [28]. The WMC instance is then compiled to some type of logical circuit for efficient inference. *Knowledge compilation* [11] is the state-of-the-art inference technique for PGMs as well as many FOPI models [28].

Inference for MLNs works in a similar way. We still sum over all possible worlds where the query is true, but the probability of a world x is now defined as

$$P(x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right),$$

where Z is the normalising constant more commonly known as the *partition function*, F is the number of formulas in the MLN, w_i is the weight of the ith formula, and $n_i(x)$ is the number of ways that formula i can be grounded in order to satisfy world x. Here, *grounding* a formula refers to replacing each variable with a value so that the formula evaluates to true.

A commonly used inference algorithm for MLNs relies on *probabilistic theorem proving* [17, 37] which is an example of a *lifted inference* algorithm, i.e., an algorithm that attempts to work directly with variables without having to consider every possible value [27]. The underlying problem, however, is still WMC, and is solved using a combination of techniques well established in the SAT community, e.g., unit propagation and clause learning [37].

2.2 Abstraction

Abstraction is an important tool in human cognition and a well-studied subject in cognitive science. For example, Gentner and Hoyos [15] investigate how children learn abstract patterns from observing several objects with a common property, while Bransford and Franks [3] show how the idea conveyed by a sentence is abstracted away from the particulars of its syntactic expression.

Abstraction is also well-known in the AI community, where the main goal of abstraction is to reduce the computational complexity of a task, while ensuring that the process of abstraction itself is reasonably efficient [33]. For instance, abstraction plays a key part in modern approaches to planning, where compound tasks are used to abstract away the details of how those tasks can be implemented [14]. More specifically, abstraction is essential in developing explainable AI [1], where it has been used to create interpretable abstractions of observed behaviour [26] and model the domain knowledge of the user as an abstraction of the system, thus producing explanations that are at the level of detail corresponding to the user's knowledge [36].

Model checking and verification benefit from abstraction as well, particularly in the area of software verification, where a complete model of the program might be too big to be handled by even the most efficient methods, in which case an abstract model could be developed. Depending on how it is created, sometimes properties of the system can be verified using the abstraction [7], while other times the abstract model might produce a false positive, i.e., signal about a possible problem where there is none. If the occurrence of a false positive is suspected, parts of the abstraction can be refined to provide the necessary level of detail, while keeping other parts as they were [6, 20].

Probabilistic abstractions have been used in the context of software verification, where probabilities can help the verification algorithm choose which part of the abstract model needs to be refined [40]. Meanwhile, in the probabilistic programming community, abstractions have been used to determine the required number of Monte Carlo samples in order to compute a probability within a required level of precision [25].

However, only recently has the general case of abstraction for FOPI models been formalised, and the work is mostly limited to defining several key properties that an abstraction may have and showing how those properties interact with each other [2]. While the work on probabilistic programming considers specific examples of abstractions [22] and presents an algorithm for performing predicate abstraction [21], significant work is required to achieve the full generality outlined in this proposal.

3 Proposed Research

While some theoretical groundwork for abstraction in FOPI has recently been developed by Belle [2], there are many questions left to be answered:

1. How to efficiently create an abstraction of an already-existing model?
2. When is the correct time to stop? What is the right balance between simplicity and information?
3. What makes one abstraction preferable to another?
4. How to incorporate abstraction steps into learning a model from data?
5. How to provide guarantees about an abstraction? For example, we may want to bound the error of an answer to any query, or to ensure that all answers remain exact for a selected set of queries.

In order to answer these questions and develop the required algorithms and techniques, we can draw inspiration from the theory of abstraction for reasoning in formal systems developed by Giunchiglia and Walsh [16] and recent work on abstraction for structural equation models [32]. In particular, an abstraction is often defined as a transformation of the representation into a different form. One way to create such a transformation is via a composition of atomic operations. For example:

- In some cases, $a \to b$ and $b \to c$ can be simplified to $a \to c$.
- If a statement S is true with high probability, perhaps that probability can be rounded up to 1, eliminating the need to consider the case where S is false.
- If a statement is true for all values of a variable, barring a few exceptions, perhaps the exceptions can be discarded.

Consider a specific query Q. Applying such an abstraction rule may or may not change the answer to Q, depending on whether the removed information is relevant to the query. Even if the answer becomes less precise, it might be an acceptable approximation, given that the error is bounded to a reasonable degree.

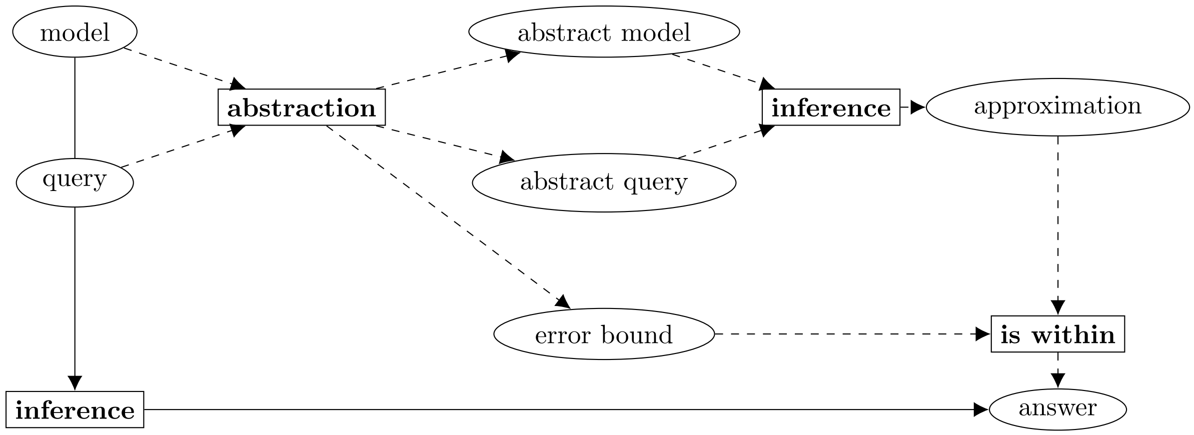

Figure 1: A graphical representation of the role abstraction can play during inference. Solid lines represent inference without abstraction, whereas dashed lines show the workflow with abstraction.

Either way, the abstraction can reduce the search space the inference algorithm has to explore in order to produce an answer.

It becomes clear that it is important to consider creating abstractions with respect to a specific set of queries. Question 5 can then be answered by considering how each abstraction rule affects different types of queries. Sometimes we may get a reasonable numerical upper bound on the error, while other times it may be too time-consuming (or impossible) to bound the error to any reasonable degree, forcing us to reject the abstraction rule altogether.

See Figure 1 for an example of how abstraction can benefit inference. The abstract model and query produced by the abstraction algorithm are likely to make inference faster, and the error bound provides a precision guarantee in case some relevant information is lost.

With this goal in mind, we will develop a comprehensive list of abstraction rules (transformations) and define a way to categorise all queries answerable by a FOPI model such that we could answer the following set of questions for each abstraction rule:

- What types of queries can no longer be answered exactly after applying the abstraction rule?
- What is the error bound? Can it be calculated in constant time?
- What is the complexity of applying the abstraction?

Questions 2 and 3 delve deeper into how an abstraction algorithm could work. If the set of rules is extensive enough, any model might eventually be oversimplified into something trivial. We need to measure two things: the amount of (relevant) information preserved by an abstraction, and the complexity of the model. The two metrics would provide a systematic way to answer both questions, while being easily adaptable to different needs (e.g., how much precision are we willing to sacrifice? What queries do we want to support?).

4 Conclusion

As abstraction for expressive probabilistic models has only been defined quite recently [2, 22], this is the perfect time to explore the possibilities and benefits of an old idea applied to modern models for probabilistic inference and reasoning. Simplification and abstraction can benefit us in both efficiency and interpretability, i.e, simpler models are likely to result in faster inference, while at the same time being easier to understand by the user. Furthermore, establishing a link between abstract and concrete representations could provide a basis for an agent's ability to correctly interpret high-level (abstract) instructions. Finally, the quest for abstraction algorithms is likely to lead to a better theoretical understanding of what properties can be preserved by an abstraction, what error bounds can be established when abstraction approximates the answer, and upper and lower bounds on the complexity of performing abstraction and providing the desired guarantees.

I am also interested in the following projects:

- Explaining and interpretable task planning (230005)
- Autonomous Agents Modelling Other Agents (230003)
- Ethical and responsible decision making (300003)

References

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[2] Vaishak Belle. Abstracting probabilistic relational models. *CoRR*, abs/1810.02434, 2018.

[3] John D Bransford and Jeffery J Franks. The abstraction of linguistic ideas. *Cognitive Psychology*, 2(4):331 – 350, 1971.

[4] Swarup Chandra, Justin Sahs, Latifur Khan, Bhavani M. Thuraisingham, and Charu C. Aggarwal. Stream mining using statistical relational learning. In Ravi Kumar, Hannu Toivonen, Jian Pei, Joshua Zhexue Huang, and Xindong Wu, editors, *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, pages 743–748. IEEE Computer Society, 2014.

[5] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

[6] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In E. Allen Emerson and A. Prasad Sistla, editors, *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2000.

[7] Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. *ACM Trans. Program. Lang. Syst.*, 16(5):1512–1542, 1994.

[8] Joana Côrte-Real, Inês Dutra, and Ricardo Rocha. On applying probabilistic logic programming to breast cancer data. In Nicolas Lachiche and Christel Vrain, editors, *Inductive Logic Programming - 27th International Conference, ILP 2017, Orléans, France, September 4-6, 2017, Revised Selected Papers*, volume 10759 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2017.

[9] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. A survey of first-order probabilistic models. In Dawn E. Holmes and Lakhmi C. Jain, editors, *Innovations in Bayesian Networks: Theory and Applications*, volume 156 of *Studies in Computational Intelligence*, pages 289–317. Springer, 2008.

[10] Brian Delaney, Andrew S. Fast, William M. Campbell, Clifford J. Weinstein, and David D. Jensen. The application of statistical relational learning to a database of criminal and terrorist activity. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 409–417. SIAM, 2010.

[11] Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2178–2185. IJCAI/AAAI, 2011.

[12] Anton Dries, Angelika Kimmig, Jesse Davis, Vaishak Belle, and Luc De Raedt. Solving probability problems in natural language. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3981–3987. ijcai.org, 2017.

[13] Gal Elidan, Kristian Kersting, and Alexander T. Ihler, editors. *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.

[14] Kutluhan Erol, James A. Hendler, and Dana S. Nau. Complexity results for HTN planning. *Ann. Math. Artif. Intell.*, 18(1):69–93, 1996.

[15] Dedre Gentner and Christian Hoyos. Analogy and abstraction. *Topics in Cognitive Science*, 9(3):672–693, 2017.

[16] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artif. Intell.*, 57(2-3):323–389, 1992.

[17] Vibhav Gogate and Pedro M. Domingos. Probabilistic theorem proving. *Commun. ACM*, 59(7):107–115, 2016.

[18] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. In James D. Herbsleb and Matthew B. Dwyer, editors, *Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31 - June 7, 2014*, pages 167–181. ACM, 2014.

[19] F. Hayes-Roth, D.A. Waterman, and D.B. Lenat. *Building expert systems*. Teknowledge series in knowledge engineering. Addison-Wesley, 1983.

[20] Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Lazy abstraction. In John Launchbury and John C. Mitchell, editors, *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002*, pages 58–70. ACM, 2002.

[21] Steven Holtzen, Guy Van den Broeck, and Todd D. Millstein. Sound abstraction and decomposition of probabilistic programs. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 2004–2013. JMLR.org, 2018.

[22] Steven Holtzen, Todd D. Millstein, and Guy Van den Broeck. Probabilistic program abstractions. In Elidan et al. [13].

[23] Arnon Levy and William Bechtel. Abstraction and the organization of mechanisms. *Philosophy of Science*, 80(2):241–261, 2013.

[24] John W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.

[25] David Monniaux. An abstract monte-carlo method for the analysis of probabilistic programs. In Chris Hankin and Dave Schmidt, editors, *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, pages 93–101. ACM, 2001.

[26] Svetlin Penkov and Subramanian Ramamoorthy. Explaining transition systems through program induction. *CoRR*, abs/1705.08320, 2017.

[27] David Poole. First-order probabilistic inference. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 985–991. Morgan Kaufmann, 2003.

[28] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2016.

[29] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic prolog and its application in link discovery. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2462–2467, 2007.

[30] Matthew Richardson and Pedro M. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[31] John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes).* Elsevier and MIT Press, 2001.

[32] Paul K. Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M. Mooij, Dominik Janzing, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Causal consistency of structural equation models. In Elidan et al. [13].

[33] Lorenza Saitta and Jean-Daniel Zucker. *Abstraction in artificial intelligence and complex systems*, volume 456. Springer, 2013.

[34] Nikita A. Sakhanenko and David J. Galas. Probabilistic logic methods and some applications to biology and medicine. *Journal of Computational Biology*, 19(3):316–336, 2012.

[35] Taisuke Sato. A statistical learning method for logic programs with distribution semantics. In Leon Sterling, editor, *Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995*, pages 715–729. MIT Press, 1995.

[36] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4829–4836. ijcai.org, 2018.

[37] Deepak Venugopal. Advances in inference methods for Markov logic networks. *IEEE Intelligent Informatics Bulletin*, 18(2):13–19, 2017.

[38] Jonas Vlasselaer and Wannes Meert. Statistical relational learning for prognostics. In *Proceedings of the 21st Belgian-Dutch Conference on Machine Learning*, pages 45–50, 2012.

[39] Shuo Yang, Mohammed Korayem, Khalifeh AlJadda, Trey Grainger, and Sriraam Natarajan. Application of statistical relational learning to hybrid recommendation systems. *CoRR*, abs/1607.01050, 2016.

[40] Xin Zhang, Xujie Si, and Mayur Naik. Combining the logical and the probabilistic in program analysis. In Tatiana Shpeisman and Justin Gottschlich, editors, *Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL@PLDI 2017, Barcelona, Spain, June 18, 2017*, pages 27–34. ACM, 2017.