Abstraction Algorithms for Probabilistic Relational Models

Paulius Dilkas

4th February 2019

1 Introduction

While the word 'abstraction' can be associated with many different processes that occur in our perception of the world, reasoning about it, or representation in language, science, or art, it can be broadly thought of as the process (and result) of omitting detail [15]. 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.

Abstraction is an important tool in human cognition and a well-studied subject in cognitive science. For example, Gentner and Hoyos [8] 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 artificial intelligence (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 [20]. For instance, abstraction plays a key part in modern approaches to planning, where compound tasks are used to abstract away the details of how the tasks can be implemented [7]. More specifically, abstraction is essential in developing explainable AI [1], where it has been used to create interpretable abstractions of observed behaviour [17] 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 [21].

Model checking and verification benefits 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 such a case, an abstract model could be developed. Depending on how it is created, sometimes properties of the system can be verified using the abstraction [5], 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 [4, 13].

Note that few of the covered works on abstraction ever mention probabilities or uncertainty—concepts crucial to AI. 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 [22]. Meanwhile, in the probabilistic programming [11] 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 [16]. However, only recently has the general case of probabilistic program abstraction been formalised [14]. Abstraction in the context of probabilities is an underexplored field, awaiting significant developments in both its theory and practice.

2 Proposed Research

My proposed research is in abstraction for probabilistic relational models (PRMs)—a collection of representations combining elements of first-order logic with probabilistic graphical models [9, 12]. A prominent example is a Markov logic network [18], which is simply a collection of statements expressed in first-order

logic, with a weight attached to each statement. While some theoretical groundwork for abstraction in these models 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 [10] and recent work on abstraction for structural equation models [19]. 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, $a \implies b$ and $b \implies c$ can be simplified to $a \implies c$.
- 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. 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 an abstraction w.r.t. 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.

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?).

3 Conclusion

As abstraction for expressive probabilistic models such as PRMs and probabilistic programs has only been defined quite recently [2, 14], 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. 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.

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. *arXiv preprint arXiv:1810.02434*, 2018.

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

[4] 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.

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

[6] 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.

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

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

[9] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*, volume 1. MIT press Cambridge, 2007.

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

[11] 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.

[12] David Heckerman, Christopher Meek, and Daphne Koller. Probabilistic models for relational data. Technical report, Technical Report MSR-TR-2004-30, Microsoft Research, 2004.

[13] 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.

[14] Steven Holtzen, Todd D. Millstein, and Guy Van den Broeck. Probabilistic program abstractions. In Elidan et al. [6].

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

[16] 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.

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

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

[19] 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. [6].

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

[21] 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.

[22] 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.