14th of August, 2023

Andre Thevapalan
Jesse Heyninck

References

1. Gerhard Brewka & Thomas Eiter (1999): *Preferred Answer Sets for Extended Logic Programs*. In: Artif. Intell. 109(1-2), pp. 297-356, doi:10.1016/S0004-3702(99)00015-6.
2. Martin Caminada (2017): *Argumentation semantics as formal discussion*. In: Journal of Applied Logics 4(8), pp. 2457-2492.
3. Martin WA Caminada, Wolfgang Dvořák & Srdjan Vesic (2014): *Preferred semantics as socratic discussion*. In: Journal of Logic and Computation 26(4), pp. 1257-1292, doi:10.1093/logcom/exu005.
4. Stefania Costantini (2006): *On the existence of stable models of non-stratified logic programs*. In: Theory Pract. Log. Program. 6(1-2), pp. 169-212, doi:10.1017/S1471068405002589.
5. Stefania Costantini, Benedetto Intrigila & Alessandro Provetti (2003): *Coherence of updates in answer set programming*. In: Proc. of the IJCAI-2003 Workshop on Nonmonotonic Reasoning, Action and Change, pp. 66-72.
6. Martin Gebser, Roland Kaminski, Benjamin Kaufmann & Torsten Schaub (2019): *Multi-shot ASP solving with clingo*. In: Theory Pract. Log. Program. 19(1), pp. 27-82, doi:10.1017/S1471068418000054.
7. Michael Gelfond & Vladimir Lifschitz (1991): *Classical Negation in Logic Programs and Disjunctive Databases*. In: New Gener. Comput. 9(3/4), pp. 365-386, doi:10.1007/BF03037169.
8. Tuomo Lehtonen, Johannes Peter Wallner & Matti Järvisalo (2021): *Declarative Algorithms and Complexity Results for Assumption-Based Argumentation*. In: J. Artif. Intell. Res. 71, pp. 265-318, doi:10.1613/jair.1.12479.
9. Javier Romero (2017): *asprin: Answer Set Programming with Preferences*. In: In Bernhard Mitschang, Norbert Ritter, Holger Schwarz, Meike Klettke, Andreas Thor, Oliver Kopp & Matthias Wieland, editors: Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Workshopband, LNI P-266, GI, pp. 159-162. Available at dl.gi.de/20.500.12116/910.
10. Claudia Schulz & Francesca Toni (2018): *On the responsibility for undecisiveness in preferred and stable labellings in abstract argumentation*. In: Artif. Intell. 262, pp. 301-335, doi:10.1016/j.artint.2018.07.001.
11. Andre Thevapalan, Jesse Heyninck & Gabriele Kern-Isberner (2021): *Establish Coherence in Logic Programs Modelling Expert Knowledge via Argumentation*. In: Joaquín Arias, Fabio Aurelio D'Asaro, Abeer Dyoub, Gopal Gupta, Markus Hecher, Emily LeBlanc, Rafael Peñaloza, Elmer Salazar, Ari Saptawijaya, Felix Weitkämper & Jessica Zangari, editors: Proceedings of the International Conference on Logic Programming 2021 Workshops co-located with the 37th International Conference on Logic Programming (ICLP 2021), Porto, Portugal (virtual), September 20th-21st, 2021, CEUR Workshop Proceedings 2970, CEUR-WS.org. Available at ceur-ws.org/Vol-2970/causalpaper4.pdf.

On the Development of PASTA: Inference in Probabilistic Answer Set Programming under the Credal Semantics

Damiano Azzolini (*University of Ferrara*)

Abstract

PASTA is a novel framework to perform inference, both exact and approximate, in Probabilistic Answer Set Programming under the Credal Semantics and to solve other related tasks, such as abduction, decision theory, maximum a posteriori inference, and parameter learning. Moreover, it also allows the expression of statistical statements. This paper shows the overall structure of the PASTA tool as well as some of the main options.

Description

Answer Set Programming (ASP) is a logic-based language useful in describing complex combinatorial domains. Most of ASP research focuses on deterministic programs. There are some semantics that extend ASP and allow uncertainty on data, mainly LPMLN [15], P-log [7], and the credal semantics [9]. The PASTA tool considers the last.

A probabilistic answer set program under the credal semantics is an answer set program extended with probabilistic facts [10], i.e., constructs of the form $\Pi :: a.$ where a is an atom and $\Pi \in [0, 1]$ is its probability value. The intuitive meaning is that a is true with probability Π. We assume that probabilistic facts cannot appear as head of rules. The following is an example of a program:
0.4:: a.
0.7:: b.
qr : - a.
qr ; nqr : - b.

A selection of a truth value for every probabilistic fact defines a world w, i.e., an answer set program, whose probability can be computed as $P(w) = \prod_{f_i \text{ selected}} \Pi_i \cdot \prod_{f_i \text{ not selected}} (1 - \Pi_i)$. The previously shown program has $2^2 = 4$ worlds. The probability of the world, for example, where both a and b are true is $0.4 \cdot 0.7 = 0.28$. Since every world is an answer set program [14], every world has zero or more answer sets. The credal semantics requires that every world has at least one answer set. The probability of a query (conjunction of ground atoms) $P(q)$ is defined by a range, with a lower and upper probability, i.e., $P(q) = [\underline{P}(q), \overline{P}(q)]$. A world contributes to the upper probability if it has at least one answer set where the query is true and it contributes to both the lower and upper probability if the query is true in all the answer sets. That is, the lower probability is the sum of the probabilities of the worlds where the query is true in all the answer sets while the upper probability is the sum of the probabilities of the worlds where the query is true in at least one answer set. In formula

$$\underline{P}(q) = \sum_{w_i | \forall m \in AS(w_i),\, m \models q} P(w_i),\ \overline{P}(q) = \sum_{w_i | \exists m \in AS(w_i),\, m \models q} P(w_i).$$

Consider the query *qr* in the previous program. The world where both a and b are false has one empty answer set, so it does not contribute to the probability. The world where a is false and b is true has two answer sets, $\{\{b\ qr\}, \{b\ nqr\}\}$. The query is true in only one of the two, so we have a contribution only to the upper probability of $0.6 \cdot 0.7 = 0.42$. The world where a is true and b is false has one answer set, $\{\{a\ qr\}\}$. The query is true in all the answer sets, so we have a contribution of $0.4 \cdot 0.3 = 0.12$ to both the lower and upper probability. Lastly, the world where both a and b are true has one answer set, $\{\{a\ b\ qr\}\}$. The query is true in all the answer sets, so we have a contribution of $0.4 \cdot 0.7 = 0.28$ to both the lower and upper probability. Overall, $P(q) = [0.12 + 0.28, 0.42 + 0.12 + 0.28] = [0.4, 0.82]$.

Currently, the PASTA tool performs exact inference [2], whose algorithm is based on projected answer set enumeration [13], approximate inference via sampling [3], abduction [1] (i.e., computing the minimal set, in terms of set inclusion, of abducible facts such that the probability of the query is maximized), MAP/MPE [5] (finding an assignment to a subset of probabilistic facts that maximizes the sum of the probabilities of the identified worlds while evidence is satisfied), decision theory [6] (finding a subset of decision atoms that maximizes a reward), and parameter learning [4] (learning the probability of the probabilistic facts such that the likelihood of the examples is maximized). It allows to express statistical statements of the form [2] $(C|A)[\Pi_l, \Pi_u]$ meaning that the fraction of As that are also Cs is between Π_l and Π_u. Moreover, it adopts normalization [11] to perform inference in probabilistic answer set programs where some worlds have no answer sets. That is, the lower and upper probability bounds are divided by the sum of the probabilities of the worlds with no answer sets.

The system is written in Python, is open source, and is available on GitHub. It can be installed via the Python package installer pip and provides a command line interface with several options to select the task to solve. Moreover, a web interface exposes some of the most relevant options in a more intuitive way. It is also possible to use the system as a Python library. PASTA leverages the clingo ASP solver [12] to compute answer sets. The system is structured in multiple modules: the pasta module parses the command line arguments and selects the subsequent functions and methods to call. The generator module parses the program into an ASP version that can be interpreted by the clingo solver. The interface with clingo is managed by the asp_interface module and the resulting answer sets are analyzed with the models_handler module which also computes the various probabilities. There are some reserved keywords: *abducible*, that denotes abducible facts in the context of abduction (for example, *abducible* a states that the atom a is an abducible); *map*, that denotes facts to be considered for the MAP/MPE task (for example, *map* 0.4 :: a states that a is a probabilistic fact whose probability is 0.4 and that can be selected to be included in the solution set for the MAP/MPE task); and *decision* and *utility*, that mark decision facts and utility attributes [8] for the decision theory task (for example, *decision* a states that a is a decision atom and $utility(f, 2.3)$ states that the reward obtained when f is true is 2.3).

Acknowledgements

This research was partly supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No. 952215.

References

1. Damiano Azzolini, Elena Bellodi, Stefano Ferilli, Fabrizio Riguzzi, and Riccardo Zese (2022): Abduction with probabilistic logic programming under the distribution semantics. International Journal of Approximate Reasoning 142, pp. 41-63, doi:10.1016/j.ijar.2021.11.003.
2. Damiano Azzolini, Elena Bellodi, and Fabrizio Riguzzi (2022): Statistical Statements in Probabilistic Logic Programming. In Georg Gottlob, Daniela Inclezan, and Marco Maratea, editors: Logic Programming and Non-monotonic Reasoning, Springer International Publishing, Cham, pp. 43-55, doi:10.1007/978-3-031-15707-3_4.
3. Damiano Azzolini, Elena Bellodi, and Fabrizio Riguzzi (2023): Approximate Inference in Probabilistic Answer Set Programming for Statistical Probabilities. In Agostino Dovier, Angelo Montanari, and Andrea Orlandini, editors: AIxIA 2022 - Advances in Artificial Intelligence, Springer International Publishing, Cham, pp. 33-46, doi:10.1007/978-3-031-27181-6_3.
4. Damiano Azzolini, Elena Bellodi, and Fabrizio Riguzzi (2023): Learning the Parameters of Probabilistic Answer Set Programs.
5. Damiano Azzolini, Elena Bellodi, and Fabrizio Riguzzi (2023): MAP Inference in Probabilistic Answer Set Programs. In Agostino Dovier, Angelo Montanari, and Andrea Orlandini, editors: AIxIA 2022 - Advances in Artificial Intelligence, Springer International Publishing, Cham, pp. 413-426, doi:10.1007/978-3-031-27181-6_29.
6. Damiano Azzolini, Elena Bellodi, and Fabrizio Riguzzi (2023): Towards a Representation of Decision Theory Problems with Probabilistic Answer Set Programs.
7. Chitta Baral, Michael Gelfond, and Nelson Rushton (2009): Probabilistic reasoning with answer sets. Theory and Practice of Logic Programming 9(1), pp. 57-144, doi:10.1017/S1471068408003645.
8. Guy Van den Broeck, Ingo Thon, Martijn van Otterlo, and Luc De Raedt (2010): DTProbLog: A Decision-Theoretic Probabilistic Prolog. In Maria Fox and David Poole, editors: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI Press, pp. 1217-1222.
9. Fabio Gagliardi Cozman and Denis Deratani Mauá (2020): The joy of Probabilistic Answer Set Programming: Semantics, complexity, expressivity, inference. International Journal of Approximate Reasoning 125, pp. 218-239, doi:10.1016/j.ijar.2020.07.004.
10. Luc De Raedt, Angelika Kimmig, and Hannu Toivonen (2007): ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In Manuela M. Veloso, editor: 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), 7, AAAI Press, pp. 2462-2467.
11. Daan Fierens, Guy Van den Broeck, Maurice Bruynooghe, and Luc De Raedt (2012): Constraints for probabilistic logic programming. In Daniel M. Roy, Vikash K. Mansinghka, and Noah D. Goodman, editors: Proceedings of the NIPS Probabilistic Programming Workshop.
12. Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub (2019): Multi-shot ASP solving with clingo. Theory and Practice of Logic Programming 19(1), pp. 27-82, doi:10.1017/S1471068418000054.
13. Martin Gebser, Benjamin Kaufmann, and Torsten Schaub (2009): Solution Enumeration for Projected Boolean Search Problems. In Willem-Jan van Hoeve, and John Hooker, editors: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer Berlin Heidelberg, pp. 71-86, doi:10.1007/978-3-642-01929-6_7.
14. Michael Gelfond and Vladimir Lifschitz (1988): The stable model semantics for logic programming. In: 5th International Conference and Symposium on Logic Programming (ICLP/SLP 1988), 88, MIT Press, pp. 1070-1080.
15. Joohyung Lee and Yi Wang (2016): Weighted Rules under the Stable Model Semantics. In Chitta Baral, James P. Delgrande and Frank Wolter, editors: Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016, AAAI Press, pp. 145-154.

Logical English Demonstration

Robert Kowalski (*Department of Computing, Imperial College, London, UK*)
Jacinto Dávila (*Universidad de Los Andes, Merida, Venezuela*)

Abstract:

Logical English (LE) is a natural language syntax for pure Prolog and other logic programming languages, such as ASP and s(CASP). Its main applications until now have been to explore the representation of a wide range of legal texts, helping to identify ambiguities, explore alternative representations of the same text, and compare the logical consequences of the alternatives. The texts include portions of loan agreements, accountancy law, Italian citizenship, EU law on criminal rights, International Swaps and Derivative contracts, and insurance contracts. The current implementation in SWI Prolog can be accessed at `https://logicalenglish.logicalcontracts.com`.

Introduction to Logical English

The basic form of LE is simply syntactic sugar for pure Prolog[2], with predicates written in infix form and declared by means of templates, as in:

```
*a borrower* defaults on *a date*
```

where asterisks delimit the argument places of the predicate. Variables are signalled by the use of one of the determiners "a", "an" or "the". An indefinite determiner, "a" or "an", introduces the first occurrence of a variable in a sentence. All later occurrences of the same variable in the same sentence are prefixed by the definite determiner "the".

LE has only minimal linguistic knowledge of English. Its knowledge of English vocabulary is restricted to the determiners; the logical connectives "if", "and", "or" and "it is not the case that"; the logical pattern "for all cases in which — it is the case that —"; and the logical keyword "that". The keyword "that" identifies the use of a meta-predicate, for representing such "propositional attitudes as prohibition, obligation, belief, desire, fear, notification, etc. Indentation, rather than parentheses, is used to indicate the relative strength of binding of the logical connectives. LE has virtually no knowledge of English grammar. In particular, it does not distinguish between singular and plural nouns and verbs, and it does not know about the relationship between the different tenses of verbs. Despite these restrictions, and because it has the same expressiveness as pure Prolog, it can be used to represent a broad range of knowledge, as shown by its application to the representation of legal texts [3,4,5,6,7]. Here is an example based on the loan agreement in [1]. The SWISH implementation of the example can be found at `https://logicalenglish.logicalcontracts.com`