Zugzwang
Logic and Artificial Intelligence

Francisco Coelho fc@uevora.pt
Bruno Dinis bruno.dinis@uevora.pt

December 6, 2022

Abstract

A major limitation of logical representations is the implicit assumption that the Background Knowledge (BK) is perfect. This assumption is problematic if data is noisy, which is often the case. Here we aim to explore how ASP specifications with probabilistic facts can lead to characterizations of probability functions on the specification's domain.

1 Introduction and Motivation

Answer Set Programming (ASP) [3] is a logic programming paradigm based on the Stable Model semantics of Normal Logic Programs (NP) that can be implemented using the latest advances in SAT solving technology. ASP is a truly declarative language that supports language constructs such as disjunction in the head of a clause, choice rules, and hard and weak constraints.

The Distribution Semantics (DS) [4] is a key approach to extend logical representations with probabilistic reasoning. Probabilistic Facts (PF) [4] are the most basic stochastic DS primitive and they take the form of logical facts, a, labelled with a probability, such as $p :: a$; Each probabilistic fact represents a boolean random variable that is true with probability p and false with probability $1 - p$. A (consistent) combination of the PFs defines a *total choice* $\theta = \{p :: a, \ldots\}$ such that

$$P(\theta) = \prod_{a \in \theta} p \prod_{a \notin \theta} (1 - p). \tag{1}$$

Our goal is to extend this probability, from total choices, to cover the specification domain. We can foresee two key applications of this extended probability:

1. Support any probabilistic reasoning/task on the specification domain.
2. Also, given a dataset and a divergence measure, now the specification can be scored (by the divergence w.r.t. the *empiric* distribution of the dataset), and sorted amongst other specifications. This is a key ingredient in algorithms searching, for example, an *optimal specification* of the dataset.

This goal faces a critical problem concerning situations where *multiple* standard models result from a given total choice[1], illustrated by the following example. The specification

$$0.3 :: a,$$
$$b \vee c \leftarrow a.$$

has three stable models, $\{\neg a\}, \{a, b\}$ and $\{a, c\}$. While it is straightforward to set $P(\neg a) = 0.7$, there is *no further information* to assign values to $P(a, b)$ and $P(a, c)$. At best, we can use a parameter α such that

$$P(a, b) = 0.3\alpha,$$
$$P(a, c) = 0.3(1 - \alpha).$$

This uncertainty in inherent to the specification, but can be mitigated with the help of a dataset: the parameter α can be estimated from the empirical distribution.

In summary, if an ASP specification is intended to describe some system that can be observed then:

1. The observations can be used to estimate the value of the parameters (such as α above and others entailed from further clauses).

2. With a probability set for the stable models, we want to extend it to all the samples (*i.e.* consistent sets of literals) of the specification.

3. This extended probability can then be related to the *empirical distribution*, using a probability divergence, such as Kullback-Leibler; and the divergence value used as a *performance* measure of the specification with respect to the observations.

4. If that specification is only but one of many possible candidates then that performance measure can be used, *e.g.* as fitness, by algorithms searching (optimal) specifications of a dataset of observations.

Currently, we are on the step two above: Extending a probability function (with parameters such as α), defined on the stable sets of a specification, to all the samples of the specification. This extension must, of course, respect the axioms of probability so that probabilistic reasoning is consistent with the ASP specification.

2 Work Plan

A team of two researchers and a undergraduate, master, or Ph.D. student, working over six months with adequate resources, should be able to advance substantial contributions and produce an intermediate progress report for a workshop, a final comprehensive paper for a conference, or start a Ph.D. project with greater reach and depth, describing:

- The formalization of the methods outlined above, including the parameter estimation from observations and the probability distribution over the specification samples.

- Application and evaluation of this approach, using tools such as s(casp), or the Potassco suit to a range of problems from the simple *Burglar, Earthquake, Alarm* to measuring a specification accuracy on a given dataset, or finding an optimal specification for a given dataset given some background knowledge.

While the theoretical work for this project has yet to be completed, there are some relevant tasks that, with different levels of ambition, can be started right now:

1. *Extract Probability Annotations.* For example, convert the annotated specification `0.3::a. b ; c :- a.` to `a ; -a. b ; c :- a`. This is a simple, syntactical task that can be implemented either with `prolog` or using `python` and the API provided by the Potassco suite.

2. *Extend Probability to Stable Models.* Application of the method outlined before, where the probability of total choices is extended to standard models using parameters, which are next estimated with a dataset.

3. *Relate Samples, Stable Models and Total Choices.* Determine which stable models, or total choices, contain and which are contained in a given sample.

4. *Propagate Probability to Samples.* Use of the relation above to assign a probability to an arbitrary event, using an aggregation operation, such as max or $\prod$, from the relevant stable models.

5. *Process Evaluation on Well-known Cases.* Assessment of the implemented prototype using well-known problems such as the "Alarm-Burglar-Earthquake".

References

References

[1] Fabio Gagliardi Cozman and Denis Deratani Mauá. "The joy of probabilistic answer set programming: semantics, complexity, expressivity, inference". In: *International Journal of Approximate Reasoning* 125 (2020), pp. 218–239.

[2] Andrew Cropper et al. "Inductive logic programming at 30". In: *Machine Learning* 111.1 (2022), pp. 147–172.

[3] Martin Gebser et al. "Answer set solving in practice". In: *Synthesis lectures on artificial intelligence and machine learning* 6.3 (2012), pp. 1–238.

[4] Fabrizio Riguzzi. *Foundations of probabilistic logic programming: Languages, semantics, inference and learning.* CRC Press, 2022.

[5] Victor Verreet et al. "Inference and learning with model uncertainty in probabilistic logic programs". In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 36. 9. 2022, pp. 10060–10069.