Zugzwang
Logic and Artificial Intelligence

Francisco Coelho
fc@uevora.pt

November 21, 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) 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) is a key approach to extend logical representations with probabilistic reasoning. Probabilistic Facts (PF) 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* $c = \{p :: a, \ldots\}$ such that

$$\mathrm{P}(C = x) = \prod_{a \in c} p \prod_{a \notin c} (1 - p). \tag{1}$$

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, illustrated by the following example. The specification

$$\begin{aligned} 0.3 :: a,& \\ b \vee c \leftarrow a.& \end{aligned} \tag{2}$$

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

$$\begin{aligned} P(ab) &= 0.3x, \\ P(ac) &= 0.3(1-x). \end{aligned}$$

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

In summary, if an ASP specification is intended to describe some observable system then:

1. The observations can be used to estimate the value of the parameters (such as x 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 x), defined on the stable sets of a specification, to all the events of the specification. This extension must, of course, respect the axioms of probability so that probabilistic reasoning is consistent with the ASP specification.

2 Extending Probabilities

Given an ASP specification, we consider the *atoms* $a \in \mathcal{A}$ and *literals*, $z \in \mathcal{L}$, *events* $e \in \mathcal{E} \iff e \subseteq \mathcal{L}$ and *worlds* $w \in \mathcal{W}$ (consistent events), *total choices* $c \in \mathcal{C} \iff c = a \vee \neg a$ and *stable models* $s \in \mathcal{S}$.

Our path, traced by equations (1) and (3 — 8), starts with the probability of total choices, $\mathrm{P}(C = c)$, expands it to stable models, $\mathrm{P}(S = s)$, and then to worlds $\mathrm{P}(W = w)$ and events $\mathrm{P}(E = e)$.

1. **Total Choices.** This case is given by $\mathrm{P}(C = c)$, from equation 1. Each total choice $C = c$ (together with the facts and rules) entails some stable models, $s \in S_c$, and each stable model $S = s$ contains a single total choice $c_s \subseteq s$.

2. **Stable Models.** Given a stable model $s \in \mathcal{S}$, and variables/values $x_{s,c} \in [0, 1]$,

$$\mathrm{P}(S = s \mid C = c) = \begin{cases} x_{s,c} & \text{if } s \in S_c, \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

   such that $\sum_{s \in S_c} x_{s,c} = 1$.

3. **Worlds.** Each world $W = w$ either:

   (a) Is a *stable model*. Then

$$\mathrm{P}(W = w \mid C = c) = \mathrm{P}(S = s \mid C = c)\,. \tag{4}$$

   (b) *Contains* some stable models. Then

$$\mathrm{P}(W = w \mid C = c) = \prod_{s \subset w} \mathrm{P}(S = s \mid C = c)\,. \tag{5}$$

(c) *Is contained* in some stable models. Then

$$\mathrm{P}(W = w \mid C = c) = \sum_{s \supset w} \mathrm{P}(S = s \mid C = c)\,. \tag{6}$$

(d) Neither contains nor is contained by a stable model. Then

$$\mathrm{P}(W = w) = 0. \tag{7}$$

4. **Events.** For each event $E = e$,

$$\mathrm{P}(E = e \mid C = c) = \begin{cases} \mathrm{P}(W = e \mid C = c) & e \in \mathcal{W}, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Since stable model are minimal, there is no proper chain $s_1 \subset w \subset s_2$ so each world folds into exactly one ot the four cases of point 3 above.

Equation (3) expresses the lack of knowledge about the probability assignment when a single total choice entails more than one stable model. In this case, how to distribute the respective probability? Our answer to this problem consists in assigning an unknown probability, $x_{s,c}$, conditional on the total choice, c, to each stable model s. This approach allow the expression of an unknown quantity and future estimation, given observed data.

The stable model case, in equation (4), identifies the probability of a stable model *as a world* with its probability as defined previously in equation (3), as a stable model.

Equation 5 results from conditional independence of the stable models $s \subset w$. Conditional independence of stable worlds asserts a least informed strategy that we make explicit:

Assumption 1. *Stable models are conditionally independent, given their total choices.*

Consider the stable models ab, ac from the example above. They result from the clause $b \vee c \leftarrow a$ and the total choice a. These formulas alone impose no relation between b and c (given a), so none should be assumed. Dependence relations are discussed in Subsection (2.1).

I'm not sure about what to say here. todo

$$\mathrm{P}(W = w \mid C = c) = \sum_{s \supset w} \mathrm{P}(S = s \mid C = c)\,.$$

But! $\mathrm{P}(W = w \mid C = c)$ already separates $\mathrm{P}(W)$ into **disjoint**

A world that neither contains nor is contained in a stable model describes a case that, according to the specification, should never be observed. So the respective probability is set to zero, per equation (7).

2.1 Dependence

Dependence relations in the underlying system can be explicitly expressed in the specification.

For example, $b \leftarrow c \wedge d$, where d is an atomic choice, explicitly expressing this dependence between b and c. One would get, for example, the specification

$$0.3 :: a, b \vee c \leftarrow a, 0.2 :: d, b \leftarrow c \wedge d.$$

with the stable models $\overline{ad}, \overline{a}d, a\overline{d}b, a\overline{d}c, adb$.

The interesting case is the subtree of the total choice ac. Notice that no stable model s contains adc because (1) adb is a stable model and (2) no stable model contains adc, because if $adc \subset s$ then $b \in s$ and $adb \subset s$.

Following the case of equations (4) and (7) this sets

$$\begin{cases} \mathrm{P}(W = adc \mid C = ad) = 0, \\ \mathrm{P}(W = adb \mid C = ad) = 1 \end{cases}$$

which concentrates all probability mass from the total choice ad in the adb branch, including the node $W = adbc$. This leads to the following cases:

x	$\mathrm{P}(x \mid C = ad)$
ad	1
adb	1
adc	0
$adbc$	1

so, for $C = ad$, $\mathrm{P}(b) = 2/3, \mathrm{P}(c) = 1/3, \mathrm{P}(b, c) = 1/3 \neq \mathrm{P}(b)\,\mathrm{P}(c)$.

Todo

Prove the four world cases (done), support the product (done) and sum (tbd) options, with the independence assumptions.

3 Developed Example

We continue with the specification from equation 2.

Step 1: Total Choices. The total choices, and respective stable models, are

Total Choice (c)	$\mathrm{P}(C = c)$	Stable Models (s)
a	0.3	ab and ac.
$\overline{a} = \neg a$	$\overline{0.3} = 0.7$	$\overline{a}$.

Step 2: Stable Models. Suppose now that

Stable Models (s)	Total Choice (c)	$\mathrm{P}(S = c \mid C = c)$
$\overline{a}$	1.0	$\overline{a}$.
ab	0.8	a.
ac	$0.2 = \overline{0.8}$	a.

Step 3: Worlds. Following equations 4 — 7 we get:

Occ. (o)	S.M. (s)	Relation	T.C. (c)	$\mathrm{P}(W = w)$
$\emptyset$	all	contained	a, $\overline{a}$	1.0
a	ab, ac	contained	a	$0.8 \times 0.3 + 0.2 \times 0.3 = 0.3$
b	ab	contained	a	$0.8 \times 0.3 = 0.24$
c	ac	contained	a	$0.2 \times 0.3 = 0.06$
$\overline{a}$	$\overline{a}$	stable model	$\overline{a}$	$1.0 \times 0.3 = 0.3$
$\overline{b}$	none	independent	none	0.0
$\overline{c}$	none	…		
ab	ab	stable model	a	0.24
ac	ac	stable model	a	0.06
$a\overline{b}$	none	…		
$a\overline{c}$	none	…		
$\overline{a}b$	$\overline{a}$	contains	$\overline{a}$	1.0
$\overline{a}c$	$\overline{a}$	…		
$\overline{a}\overline{b}$	$\overline{a}$	…		
$\overline{a}\overline{c}$	$\overline{a}$	…		
abc	ab, ac	contains	a	$0.8 \times 0.2 = 0.016$

References

1. Victor Verreet, Vincent Derkinderen, Pedro Zuidberg Dos Martires, Luc De Raedt, Inference and Learning with Model Uncertainty in Probabilistic Logic Programs (2022)

2. Andrew Cropper, Sebastijan Dumancic, Richard Evans, Stephen H. Muggleton, Inductive logic programming at 30 (2021)

3. Fabio Gagliardi Cozman, Denis Deratani Mauá, The joy of Probabilistic Answer Set Programming: Semantics - complexity, expressivity, inference (2020)

4. Fabrizio Riguzzi, Foundations of Probabilistic Logic Programming Languages, Semantics, Inference and Learning. Rivers Publishers (2018)

5. Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub, Answer Set Solving in Practice, Morgan & Claypool Publishers (2013)