ICLP - Formal improvement An Algebraic Approach to Stochastic ASP

Francisco Coelho   Salvador Abreu   Bruno Dinis
Universidade de Évora
Évora, Portugal
{fc, spa, bruno.dinis}@uevora.pt

We address the problem of extending probability from the total choices of an ASP program to the stable models, and from there to general events. Our approach is algebraic in the sense that it relies on an equivalence relation over the set of events and uncertainty is expressed with variables and polynomial expressions. We illustrate our methods with two examples, one of which shows a connection to bayesian networks.

1 Introduction and Motivation

A major limitation of logical representations in real world applications is the implicit assumption that the background knowledge is perfect. This assumption is problematic if data is noisy, which is often the case. Here we aim to explore how answer set programming programs with probabilistic facts can lead to characterizations of probability functions on the program's domain, which is not straightforward in the context of answer set programming, as explained below (see also [4, 21, 2, 16]). Unlike current systems such as ProbLog [6], P-log [2], LP$^{\text{MLN}}$ [12], or cplint [1], that derive a probability distribution from a program, in our system some choices are represented by a parameter that can be later estimated from further information, *e.g.* observations. This approach enables later refinement and scoring of a partial program of a model from additional evidence.

Answer set programming (ASP) [14] is a logic programming paradigm based on the stable model (SM) semantics of normal programs (NPs) that can be implemented using the latest advances in SAT solving technology. Unlike ProLog, ASP is a truly declarative language that supports language constructs such as disjunction in the head of a clause, choice rules, and both hard and weak constraints.

The distribution semantics (DS) [19, 18] is a key approach to extend logical representations with probabilistic reasoning. Let $\mathscr{A}$ be a finite set of atoms. A *pre-total choice* is a subset t^* of $\mathscr{A}$. The *total choice* (TC) associated to t^* is the set $t := t^* \cup \left\{ \overline{a} \;\middle|\; a \in \mathscr{A} \setminus t^* \right\}$ where $\overline{a}$ stands for $\neg a$. Probabilistic facts (PFs) are the most basic DS stochastic primitives and take the form $a{:}p$ where each $a \in \mathscr{A}$ is associated to some $p \in [0,1]$. Each PF then represents a boolean random variable that is true with probability p and false with probability $\overline{p} = 1 - p$.

Let $F = \left\{ a{:}p \;\middle|\; a \in \mathscr{A}, p \in [0,1] \right\}$. For a total choice t over $\mathscr{A}$, define

$$P_t := \left\{ p \;\middle|\; a \in t^* \wedge a{:}p \in F \right\} \cup \left\{ \overline{p} \;\middle|\; a \in t \setminus t^* \wedge a{:}p \in F \right\}$$

and

$$\mathrm{P}(T = t) = \prod_{p \in P_t} p, \tag{1}$$

where T is a random variable whose values are total choices.

Our goal is to extend this probability (which is, indeed, a product of Bernoulli distributions [20]), from total choices, to cover the program domain. We use the term "program" as a set of rules and facts, plain and probabilistic. We can foresee two key applications of this extended probability:

1. Support probabilistic reasoning/tasks on the program domain.
2. Given a dataset and a divergence measure, the program can be scored (by the divergence w.r.t. the *empiric* distribution of the dataset), and weighted or sorted amongst other programs. These are key ingredients in algorithms searching, for example, optimal models of a dataset.

To extend probabilities from total choices we start with the stance that *a program describes an observable system*, that *the stable models are all the possible states* of that system and that *observations (i.e. events) are stochastic* — one observation can be sub-complete (a proper subset of a SM) or super-complete (a proper superset of a SM), and might not determine the real state of the system. From here, probabilities must be extended from total choices (TCs) to SMs and then to any event. This extension process starts with a critical problem, illustrated by the example in section 2, concerning situations where multiple SMs, *ab* and *ac*, result from a single TC, *a*, but there is not enough information (in the program) to assign a single probability to each SM. We propose to address this issue by using algebraic variables to describe that lack of information and then estimate the value of those variables from empirical data. This lack of uniqueness is also addressed in [4] along a different approach, using credal sets.

In another related work [21] epistemic uncertainty (or model uncertainty) is considered as a lack of knowledge about the underlying model, that may be mitigated via further observations. This seems to presuppose a bayesian approach to imperfect knowledge in the sense that having further observations allows to improve/correct the model. Indeed, that approach uses Beta distributions on the total choices in order to be able to learn a distribution on the events . This approach seems to be specially fitted to being able to tell when some probability lies beneath some given value. Our approach seems to be similar in spirit, while remaining algebraic in the way that the extension of probabilities is addressed.

The example in section 2 uses the code available in the project's repository[1], developed with the *Julia* programming language [3], and the *Symbolics* [9], and *DataFrames* [10] libraries.

2 A Simple but Fruitful Example

In this section we consider a somewhat simple case that showcases the problem of extending probabilities from total choices to stable models and then to events. As mentioned before, the main issue arises from the lack of information in the program to assign a single probability to each stable model. This becomes a crucial problem in situations where multiple stable models result from a single total choice. We will come back to this example in section 4.1, after we present our proposal for extending probabilities from total choices to stable models in section 3.

Example 1. *Consider $\mathscr{A} = \{a, b, c\}$ and the following program*

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

The standard form *of this program results from replacing annotated facts, such as $a{:}0.3$, by the associated disjunctions, $a \vee \neg a$. The stable models of the annotated program are the same as the ones from the standard form: $\overline{a}, ab$ and ac, where $\overline{a}$ stands for $\neg a$ (see fig. 1). While it is straightforward to*

[1] https://git.xdi.uevora.pt/fc/sasp

assume $\mathrm{P}(\overline{a}) = 0.7$*, there is no obvious explicit way to assign values to* $\mathrm{P}(ab)$ *and* $\mathrm{P}(ac)$*. For instance, we can use a parameter* θ *as in*

$$\mathrm{P}(ab) = 0.3\theta,$$
$$\mathrm{P}(ac) = 0.3(1-\theta)$$

to express our knowledge that ab, ac *are events related in a certain way and, simultaneously, our uncertainty about that relation. The parameter* $\theta = \theta_{s,t}$ *depends on both the stable model* s *and the total choice* t*. This uncertainty can then be addressed with the help of adequate distributions, such as empirical distributions from a dataset.*

If an ASP program is intended to describe some system then:

1. With a probability set for the stable models, we want to extend it to all the events of the program domain.
2. In the case where some statistical knowledge is available, for example, in the form of a distribution, we consider it as "external" knowledge about the parameters, that doesn't affect the extension procedure described below.
3. Statistical knowledge can be used to estimate parameters and to "score" the program.
4. If that program is only but one of many possible candidates then that score can be used, *e.g.* as fitness, by algorithms searching (optimal) programs of a dataset of observations.
5. If observations are not consistent with the program, then we ought to conclude that the program is wrong and must be changed accordingly.

Currently, we are addressing the problem of extending a probability function (possibly using parameters such as θ above), defined on the SMs of a program, to all the events of that program. This extension must satisfy the Kolmogorov axioms of probability so that probabilistic reasoning is consistent with the ASP program and follow our interpretation of stable models as the states of an observable system.

As sets, the SMs can have non-empty intersection. But, as states of a system, we assume that SMs are disjoint events, in the following sense:

Assumption 1. *Stable models are disjoint events: For any set* X *of stable models,*

$$\mathrm{P}(X) = \sum_{s \in X} \mathrm{P}(s) \tag{3}$$

Consider the stable models ab, ac from example 1, that result from the clause $b \vee c \leftarrow a$ and the total choice $\{a\}$. Since we intend to associate each stable model with a state of the system, ab and ac should be *disjoint* events. So $b \vee c$ is interpreted as an *exclusive disjunction* and, from that particular clause, no further relation between b and c is assumed. This does not prevent that other clauses may be added that entail further dependencies between b and c, which in turn may change the stable models.

By not making distribution assumptions on the clauses of the program we can state such properties on the semantics of the program, as we've done in assumption 1.

3 Extending Probabilities

The diagram in fig. 1 illustrates the problem of extending probabilities from total choices to stable models and then to general events in an *edge-wise* process, where the value in a node is defined from the values in its neighbors. This quickly leads to coherence problems concerning probability, with no clear systematic

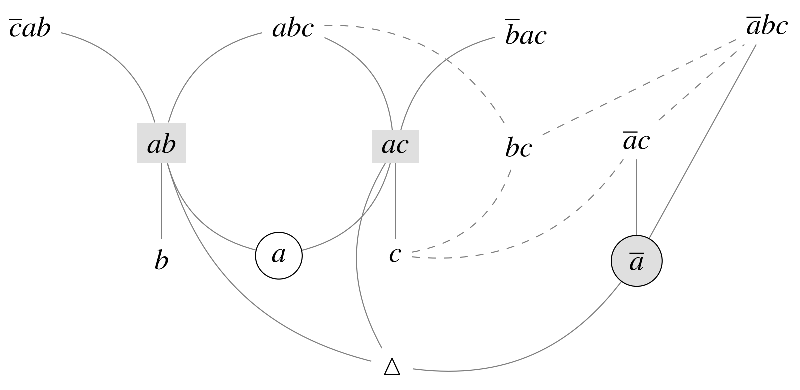


Figure 1: Some events related to the stable models of example 1. The circle nodes are total choices and shaded nodes are stable models. Solid lines represent relations with the SMs and dashed lines relations between other events. The set of events contained in all stable models, denoted by $\triangle$, is empty in this example.

approach. Notice that bc is not directly related with any stable model therefore propagating values through edges would assign a hard to justify ($\neq 0$) value to bc. Instead, we propose to base the extension in the relation an event has with the stable models.

3.1 An Equivalence Relation

Given an ASP program, we consider a set of *atoms* $\mathscr{A}$, the set $\mathscr{L}$ of the *literals* over $\mathscr{A}$, and the set of *events* $\mathscr{E}$ such that $e \in \mathscr{E} \iff e \subseteq \mathscr{L}$. We also consider $\mathscr{W}$ the set of *worlds* (consistent events), a set of *total choices* $\mathscr{T}$ such that for every $a \in \mathscr{A}$ we have $a \in t$ or $\neg a \in t$, and $\mathscr{S}$ the set of *stable models* such that $\mathscr{S} \subset \mathscr{W}$. At last, the set of stable models entailed by the total choice t is denoted by $\langle t \rangle$.

Our path to extend probabilities starts with a perspective of stable models as playing a role similar to *prime factors*. The stable models of a program are the irreducible events entailed from that program and any event must be considered under its relation with the stable models.

From example 1, consider the SMs $\overline{a}, ab, ac$ and events a, abc and c. While a is related with (contained in) with both ab, ac, event c is related only with ac. So, a and c are related with different SMs. On the other hand, both ab, ac are related with abc. So a and abc are related with the same stable models.

Definition 1. *The* stable core (SC) *of the event $e \in \mathscr{E}$ is*

$$[\![e]\!] := \{s \in \mathscr{S} \mid s \subseteq e \vee e \subseteq s\}. \tag{4}$$

where $\mathscr{S}$ is the set of stable models.

We now define an equivalence relation so that two events are related if either both are inconsistent or both are consistent and, in the latter case, with the same stable core.

Definition 2. *For a given program, let $u, v \in \mathscr{E}$. The equivalence relation $\sim$ is defined by*

$$u \sim v :\iff u, v \notin \mathscr{W} \vee \left(u, v \in \mathscr{W} \wedge [\![u]\!] = [\![v]\!]\right). \tag{5}$$

Observe that the minimality of stable models implies that, in definition 1, either e is a stable model or at least one of $\exists s\,(s \subseteq e), \exists s\,(e \subseteq s)$ is false. This equivalence relation defines a partition on the set

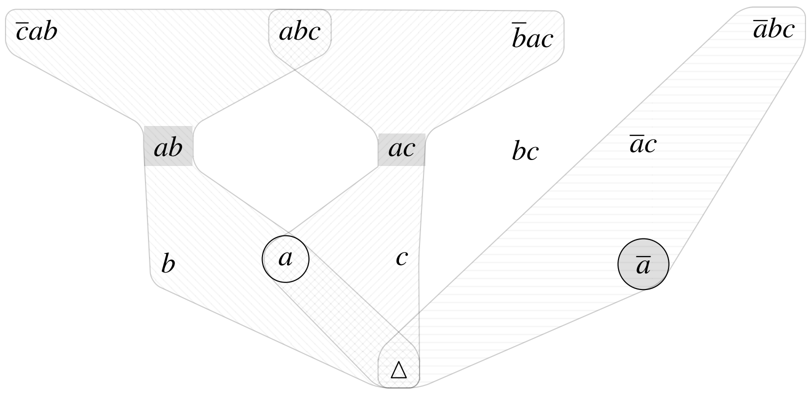


Figure 2: Classes (of consistent events) related to the stable models of example 1 are defined through intersections and inclusions. In this picture we can see, for example, the classes $\{\overline{c}ab, ab, b\}$ and $\{a, abc\}$. Different fillings correspond to different classes and, as before, the circle nodes are total choices and shaded nodes are stable models. Notice that bc is not in a "filled" area.

of events, where each class holds a unique relation with the stable models. In particular we denote each class by:

$$[e]_\sim = \begin{cases} \bot := \mathscr{E} \setminus \mathscr{W} & \text{if } e \in \mathscr{E} \setminus \mathscr{W}, \\ \left\{ u \in \mathscr{W} \mid \llbracket u \rrbracket = \llbracket e \rrbracket \right\} & \text{if } e \in \mathscr{W}. \end{cases} \tag{6}$$

The combinations of the stable models, together with the set of inconsistent events $\bot$, form a set of representatives. Consider again example 1. As previously mentioned, the stable models are the elements of $\mathscr{S} = \{\overline{a}, ab, ac\}$ so the quotient set of this relation is

$$[\mathscr{E}]_\sim = \left\{ \begin{array}{lll} \bot, & \Diamond, & [\overline{a}]_\sim, \\ [ab]_\sim, & [ac]_\sim, & [\overline{a}, ab]_\sim, \\ [\overline{a}, ac]_\sim, & [ab, ac]_\sim, & [\overline{a}, ab, ac]_\sim \end{array} \right\}, \tag{7}$$

where $\Diamond$ denotes, with abuse of notation, both the class of *independent* events e such that $\llbracket e \rrbracket = \emptyset$ and its core and $\triangle$ is the set of events contained in all SMs. We have:

Core, $\llbracket e \rrbracket$	**Class**, $[e]_\sim$	**Size**, $\#[e]_\sim$
$\bot$	$\overline{a}a, \dots$	37
$\Diamond$	$\overline{b}, \overline{c}, bc, \overline{b}a, \overline{b}c, \overline{bc}, \overline{c}a, \overline{c}b, \overline{bc}a$	9
$\overline{a}$	$\overline{a}, \overline{a}b, \overline{a}c, \overline{ab}, \overline{ac}, \overline{a}bc, \overline{ac}b, \overline{ab}c, \overline{abc}$	9
ab	$b, ab, \overline{c}ab$	3
ac	$c, ac, \overline{b}ac$	3
$\overline{a}, ab$	$\emptyset$	0
$\overline{a}, ac$	$\emptyset$	0
ab, ac	a, abc	2
$\overline{a}, ab, ac$	$\triangle$	1
$[\mathscr{E}]_\sim$	$\mathscr{E}$	64

Since all events within an equivalence class are in relation with a specific set of stable models, *measures, including probability, should be constant within classes*:

$$\forall u \in [e]_\sim \left(\mu(u) = \mu(e)\right).$$

In general, we have *much more* stable models than literals but their combinations are still *much less* than events. Nevertheless, the equivalence classes allow us to propagate probabilities from total choices to events, as explained in the next subsection.

In this specific case, instead of dealing with $64 = 2^6$ events, we consider only the $9 = 2^3 + 1$ classes, well defined in terms of combinations of the stable models.

3.2 From Total Choices to Events

Our path to set a distribution on $\mathscr{E}$ starts with the more general problem of extending *measures*, since extending *probabilities* easily follows by means of a suitable normalization (done in (15) and (16)), and has two phases:

1. Extension of the probabilities, *as measures*, from the total choices to events.
2. Normalization of the measures on events, recovering a probability.

The "extension" phase, traced by eq. (1) and eqs. (8) to (14), starts with the measure (probability) of total choices, $\mu(t) = \mathrm{P}(T = t)$, expands it to stable models, $\mu(s)$, and then, within the equivalence relation from eq. (5), to (general) events, $\mu(e)$, including (consistent) worlds.

Total Choices. Using eq. (1), this case is given by

$$\mu_{\mathrm{TC}}(t) := \mathrm{P}(T = t) = \prod_{p \in P_t} p. \tag{8}$$

Stable Models. Recall that each total choice t, together with the rules and the other facts of a program, defines the set $\langle t \rangle$ of stable models associated with that choice. Given a total choice t, a stable model s, and variables or values $\theta_{s,t} \in [0,1]$ such that $\sum_{s \in \langle t \rangle} \theta_{s,t} = 1$, we define

$$\mu(s,t) := \begin{cases} \theta_{s,t} & \text{if } s \in \langle t \rangle \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Classes. Each class is either the inconsistent class, $\perp$, or is represented by some set of stable models.

Inconsistent Class. The inconsistent class contains events that are logically inconsistent, thus should never be observed and have measure zero:

$$\mu(\perp, t) := 0.^{2} \tag{10}$$

Independent Class. A world that neither contains nor is contained in a stable model corresponds to a non-state, according to the program. So the respective measure is also set to zero:

$$\mu(\Diamond, t) := 0. \tag{11}$$

[2] Notice that this measure being equal to zero is actually independent of the total choice.

Other Classes. The extension must be constant within a class, its value should result from the elements in the stable core, and respects assumption 1 (stable models are disjoint):

$$\mu\left([e]_\sim, t\right) := \mu\left(\llbracket e \rrbracket, t\right) = \sum_{s \in \llbracket e \rrbracket} \mu(s,t) \tag{12}$$

and

$$\mu\left([e]_\sim\right) := \sum_{t \in \mathcal{T}} \mu\left([e]_\sim, t\right) \mu_{\mathrm{TC}}(t)\,. \tag{13}$$

Events. Each (general) event e is in the class defined by its stable core, $\llbracket e \rrbracket$. So, denoting by $\#X$ the number of elements in X, we set:

$$\mu(e,t) := \begin{cases} \frac{\mu\left([e]_\sim, t\right)}{\#[e]_\sim} & \text{if } \#[e]_\sim > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

and

$$\mu(e) := \sum_{t \in \mathcal{T}} \mu(e,t)\, \mu_{\mathrm{TC}}(t)\,. \tag{15}$$

The $\theta_{s,t}$ parameters in equation (9) express the *program's* lack of knowledge about the measure assignment, when a single total choice entails more than one stable model. In that case, how to distribute the respective measures? Our proposal to address this problem consists in assigning an unknown measure, $\theta_{s,t}$, conditional on the total choice, t, to each stable model s. This approach allows the expression of an unknown quantity and future estimation, given observed data.

Equation (12) results from assumption 1 and states that the measure of a class $[e]_\sim$ is the sum over it's stable core, $\llbracket e \rrbracket$, and (13) *marginalizes* the TCs on (12).

The *normalizing factor* is:

$$Z := \sum_{e \in \mathcal{E}} \mu(e) = \sum_{[e]_\sim \in [\mathcal{E}]_\sim} \mu\left([e]_\sim\right),$$

and now equation (15) provides a straightforward way to define the *probability of observation of a single event*:

$$\mathrm{P}(E = e) := \frac{\mu(e)}{Z}. \tag{16}$$

Equation (15) together with external statistical knowledge, can be used to learn about the *initial* probabilities of the atoms, that should not (and by proposition 1 can't) be confused with the explicit μ_{TC} set in the program.

It is now straightforward to check that $\mathrm{P}(E)$ satisfies the Kolmogorov axioms of probability.

Since total choices are also events, one can ask, for an arbitrary total choices t, if $\mathrm{P}(T = t) = \mathrm{P}(E = t)$ or, equivalently, if $\mu_{\mathrm{TC}}(t) = \mu(t)$. However, it is easy to see that, in general, that cannot be true. While the domain of the random variable T is the set of total choices, for E the domain is much larger, including all the events. Except for trivial programs, where the SMs are the TCs, some events other than total choices have non-zero probability.

Proposition 1. *In a program with a stable model that is not a total choice there is at least one $t \in \mathcal{T}$ such that:*

$$\mathrm{P}(T = t) \neq \mathrm{P}(E = t)\,. \tag{17}$$

Proof. Suppose towards a contradiction that $\mathrm{P}(T = t) = \mathrm{P}(E = t)$ for all $t \in \mathscr{T}$. Then

$$\sum_{t \in \mathscr{T}} \mathrm{P}(E = t) = \sum_{t \in \mathscr{T}} \mathrm{P}(T = t) = 1.$$

Hence $\mathrm{P}(E = x) = 0$ for all $x \in \mathscr{E} \setminus \mathscr{T}$, in contradiction with the fact that for at least one $s \in \mathscr{S} \setminus \mathscr{T}$ one has $\mathrm{P}(E = s) > 0$. $\square$

The essential conclusion of proposition 1 is that we are dealing with *two distributions*: one, on the TCs, explicit in the annotations of the programs and another one, on the events, and entailed by the explicit annotations *and the structure of the stable models*.

4 Developed Examples

Here we apply the methods from section 3 to example 1 and to a well known bayesian network: the Earthquake, Burglar, Alarm problem.

4.1 The SBF Example

We continue with the program from eq. (2).

Total choices. The total choices, and respective stable models, are

Total choice	**Stable models**	$\mu_{\mathbf{TC}}(t)$
a	ab, ac	0.3
$\overline{a}$	$\overline{a}$	$\overline{0.3} = 0.7$

Stable models. The $\theta_{s,t}$ parameters in this example are

$\theta_{s,t}$	$\overline{a}$	a
$\overline{a}$	1	0
ab	0	θ
ac	0	$\overline{\theta}$

with $\theta \in [0, 1]$.

Classes. Following the definitions in eqs. (4) to (6) and (10) to (12) we get the following quotient set (ignoring $\perp$ and $\Diamond$), and measures /[1]:

$[\![e]\!]$	$\mu(s, \overline{a})$	$\mu(s, a)$	$\mu\left([e]_\sim, \overline{a}\right)$	$\mu\left([e]_\sim, a\right)$	$\mu\left([e]_\sim\right)$
	$\overline{a}, ab, ac$	$\overline{a}, ab, ac$	$\mu_{\mathrm{TC}} = 0.7$	$\mu_{\mathrm{TC}} = 0.3$	
$\overline{a}$	$\boxed{1}, 0, 0$	$\boxed{0}, \theta, \overline{\theta}$	1	0	0.7
ab	$1, \boxed{0}, 0$	$0, \boxed{\theta}, \overline{\theta}$	0	θ	0.3θ
ac	$1, 0, \boxed{0}$	$0, \theta, \boxed{\overline{\theta}}$	0	$\overline{\theta}$	$0.3\overline{\theta}$
$\overline{a}, ab$	$\boxed{1}, \boxed{0}, 0$	$\boxed{0}, \boxed{\theta}, \overline{\theta}$	1	θ	$0.7 + 0.3\theta$
$\overline{a}, ac$	$\boxed{1}, 0, \boxed{0}$	$\boxed{0}, \theta, \boxed{\overline{\theta}}$	1	$\overline{\theta}$	$0.7 + 0.3\overline{\theta}$
ab, ac	$1, \boxed{0}, \boxed{0}$	$0, \boxed{\theta}, \boxed{\overline{\theta}}$	0	$\theta + \overline{\theta} = 1$	0.3
$\overline{a}, ab, ac$	$\boxed{1}, \boxed{0}, \boxed{0}$	$\boxed{0}, \boxed{\theta}, \boxed{\overline{\theta}}$	1	$\theta + \overline{\theta} = 1$	1

[1] /These are **equations** and must become **tables** to float to the top of the page

$[\![e]\!]$	$\#\{S \in [e]_\sim\}$	$\mathrm{P}(S \in [e]_\sim)$	$\mathrm{P}(E \in [e]_\sim)$
$\bot$	0	0	0
$\Diamond$	24	$\frac{24}{1000}$	0
$\overline{a}$	647	$\frac{647}{1000}$	$\frac{7}{23}$
ab	66	$\frac{66}{1000}$	$\frac{3}{23}\theta$
ac	231	$\frac{231}{1000}$	$\frac{3}{23}\overline{\theta}$
$\overline{a}, ab$	0	0	0
$\overline{a}, ac$	0	0	0
ab, ac	7	$\frac{7}{1000}$	$\frac{3}{23}$
$\overline{a}, ab, ac$	25	$\frac{25}{1000}$	$\frac{10}{23}$
	$n = 1000$		

Table 1: *Experiment 1.* Results from an experiment where $n = 1000$ samples where generated following the *Model+Noise* procedure with parameters $\alpha = 0.1, \beta = 0.3, \gamma = 0.2$. The *empirical* distribution is represented by the random variable S while the *prior*, as before, is denoted by E.

Prior Distributions. Following the above values (in rational form), and considering the inconsistent and independent classes (resp. $\bot, \Diamond$):

$[\![e]\!]$	$\#[e]_\sim$	$\mu([e]_\sim)$	$\mu(e)$	$\mathrm{P}(E = e)$	$\mathrm{P}(E \in [e]_\sim)$
$\bot$	37	0	0	0	0
$\Diamond$	9	0	0	0	0
$\overline{a}$	9	$\frac{7}{10}$	$\frac{7}{90}$	$\frac{7}{207}$	$\frac{7}{23}$
ab	3	$\frac{3}{10}\theta$	$\frac{1}{10}\theta$	$\frac{1}{23}\theta$	$\frac{3}{23}\theta$
ac	3	$\frac{3}{10}\overline{\theta}$	$\frac{1}{10}\overline{\theta}$	$\frac{1}{23}\overline{\theta}$	$\frac{3}{23}\overline{\theta}$
$\overline{a}, ab$	0	$\frac{7+3\theta}{10}$	0	0	0
$\overline{a}, ac$	0	$\frac{7+3\overline{\theta}}{10}$	0	0	0
ab, ac	2	$\frac{3}{10}$	$\frac{3}{20}$	$\frac{3}{46}$	$\frac{3}{23}$
$\overline{a}, ab, ac$	1	1	1	$\frac{10}{23}$	$\frac{10}{23}$
			$Z = \frac{23}{10}$		

So the prior distributions, denoted by the random variable E, of events and classes are:

$[\![e]\!]$	$\bot$	$\Diamond$	$\overline{a}$	ab	ac	$\overline{a}, ab$	$\overline{a}, ac$	ab, ac	$\overline{a}, ab, ac$
$\mathrm{P}(E = e)$	0	0	$\frac{7}{207}$	$\frac{1}{23}\theta$	$\frac{1}{23}\overline{\theta}$	0	0	$\frac{3}{46}$	$\frac{10}{23}$
$\mathrm{P}(E \in [e]_\sim)$	0	0	$\frac{7}{23}$	$\frac{3}{23}\theta$	$\frac{3}{23}\overline{\theta}$	0	0	$\frac{3}{23}$	$\frac{10}{23}$

(18)

Testing the Prior Distributions.

These results can be *tested by simulation* in a two-step process, where (1) a "system" is *simulated*, to gather some "observations" and then (2) empirical distributions from those samples are *related* with the prior distributions from eq. (18). Tables 1 and 2 summarize some of those tests, where datasets of $n = 1000$ observations are generated and analyzed.

Simulating a System. Following some criteria, more or less related to the given program, a set of events, that represent observations, is generated. Possible simulation procedures include:

- *Random.* Each sample is a Random Set of Literals (RSL). Additional sub-criteria may require, for example, consistent events, a Random Consistent Event (RCE) simulation.
- *Model+Noise.* Gibbs' sampling [8] tries to replicate the program model and also to add some noise. For example, let $\alpha, \beta, \gamma \in [0,1]$ be some parameters to control the sample generation. The first parameter, α is the "out of model" samples ratio; β represents the choice a or $\overline{a}$ (explicit in the model) and γ is the simulation representation of θ. A single sample is then generated following the probabilistic choices below:

$$\begin{cases} \alpha & \text{by RCE} \\ & \begin{cases} \beta & \overline{a} \\ & \begin{cases} \gamma & ab \\ & ac \end{cases} \end{cases} \end{cases},$$

  where

$$\begin{cases} p & x \\ & y \end{cases}$$

  denotes "*the value of x with probability p, otherwise y*" — notice that y might entail x and *vice-versa*: E.g. some ab can be generated in the RCE.
- *Other Processes.* Besides the two sample generations procedures above, any other processes and variations can be used. For example, requiring that one of $x, \overline{x}$ literals is always in a sample or using specific distributions to guide the sampling of literals or events.

Relating the Empirical and the Prior Distributions. The data from the simulated observations is used to test the prior distribution. Consider the prior, $\mathrm{P}(E)$, and the empirical, $\mathrm{P}(S)$, distributions and the following error function:

$$\mathrm{err}(\theta) := \sum_{e \in \mathscr{E}} \big(\mathrm{P}(E = e) - \mathrm{P}(S = e)\big)^2. \tag{19}$$

Since E depends on θ, one can ask how does the error varies with θ, what is the *optimal* (i.e. minimum) error value

$$\hat{\theta} := \arg\min_{\theta} \mathrm{err}(\theta) \tag{20}$$

and what does it tell us about the program.

In order to illustrate this analysis, consider the experiment summarized in table 1.

1. Equation (19) becomes
$$\mathrm{err}(\theta) = \frac{20869963}{66125000} + \frac{477}{52900}\theta + \frac{18}{529}\theta^2.$$
2. The minimum of $\mathrm{err}(\theta)$ is at $\frac{477}{52900} + 2\frac{18}{529}\theta = 0$. Since this value is negative and $\theta \in [0,1]$, it must be $\hat{\theta} = 0$, and
$$\mathrm{err}\left(\hat{\theta}\right) = \frac{20869963}{66125000} \approx 0.31561.$$

$[\![e]\!]$	$\#\{S_{0.2} \in [e]_\sim\}$	$\#\{S_{0.8} \in [e]_\sim\}$	$\#\{S_{0.5} \in [e]_\sim\}$
$\bot$	0	0	0
$\Diamond$	24	28	23
$\overline{a}$	647	632	614
ab	66	246	165
ac	231	59	169
$\overline{a}, ab$	0	0	0
$\overline{a}, ac$	0	0	0
ab, ac	7	8	4
$\overline{a}, ab, ac$	25	27	25

Table 2: *Experiments 2 and 3.* Results from experiments, each with $n = 1000$ samples generated following the *Model+Noise* procedure, with parameters $\alpha = 0.1, \beta = 0.3, \gamma = 0.8$ (Experiment 2) and $\gamma = 0.5$ (Experiment 3). Empirical distributions are represented by the random variables $S_{0.8}$ and $S_{0.5}$ respectively. Data from experience table 1 is also included, and denoted by $S_{0.2}$, to provide reference.

The parameters α, β, γ of that experiment favour ac over ab. In particular, setting $\gamma = 0.2$ means that in the simulation process, choices between ab and ac favour ac, 4 to 1. For completeness sake, we also describe one experiment that favours ab over ac (setting $\gamma = 0.8$) and one balanced ($\gamma = 0.5$).

For $\gamma = 0.8$, the error function is

$$\begin{aligned}\operatorname{err}(\theta) &= \frac{188207311}{529000000} - \frac{21903}{264500}\theta + \frac{18}{529}\theta^2 \\ &\approx 0.35579 - 0.08281\theta + 0.03403\theta^2\end{aligned}$$

and, with $\theta \in [0,1]$ the minimum is at $-0.08281 + 0.068805\theta = 0$, *i.e.*:

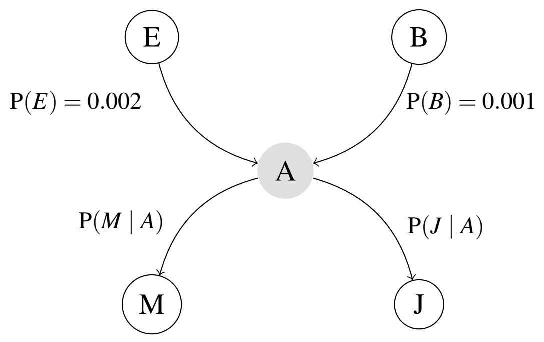
$$\begin{aligned}\hat{\theta} &: \frac{0.08281}{0.06805} \approx 1.21683 \quad > 1. \quad \text{So, } \hat{\theta} = 1, \\ \operatorname{err}\left(\hat{\theta}\right) &\approx 0.30699 \qquad .\end{aligned}$$

For $\gamma = 0.5$, the error function is

$$\begin{aligned}\operatorname{err}(\theta) &= \frac{10217413}{33062500} - \frac{2181}{66125}\theta + \frac{18}{529}\theta^2 \\ &\approx 0.30903 - 0.03298\theta + 0.03402\theta^2\end{aligned}$$

and, with $\theta \in [0,1]$ the minimum is at $-0.03298 + 0.06804\theta = 0$, *i.e.*:

$$\begin{aligned}\hat{\theta} &\approx \frac{0.03298}{0.06804} \approx 0.48471 \approx \frac{1}{2}, \\ \operatorname{err}\left(\hat{\theta}\right) &\approx 0.30104\end{aligned}$$



$P(M \mid A)$

	m	$\neg m$
a	0.9	0.1
$\neg a$	0.05	0.95

$P(J \mid A)$

	j	$\neg j$
a	0.7	0.3
$\neg a$	0.01	0.99

$P(A \mid B \wedge E)$

		a	$\neg a$
b	e	0.95	0.05
b	$\neg e$	0.94	0.06
$\neg b$	e	0.29	0.71
$\neg b$	$\neg e$	0.001	0.999

Figure 3: The Earthquake, Burglary, Alarm model

These experiments show that data can indeed be used to estimate the parameters of the model. However, we observe that the estimated $\hat{\theta}$ has a tendency to over- or under- estimate the θ used to generate the samples. More precisely, in experiment 1 data is generated with $\gamma = 0.2$ (the surrogate of θ) which is under-estimated with $\hat{\theta} = 0$ while in experiment 2, $\gamma = 0.8$ leads the over-estimation $\hat{\theta} = 1$. This suggests that we might need to refine the error estimation process. However, experiment 3 data results from $\gamma = 0.5$ and we've got $\hat{\theta} \approx 0.48471 \approx 0.5$, which is more in line with what is to be expected.

4.2 An Example Involving Bayesian Networks

As it turns out, our framework is suitable to deal with more sophisticated cases, in particular cases involving bayesian networks. In order to illustrate this, in this section we see how the classical example of the Burglary, Earthquake, Alarm [17] works in our setting. This example is a commonly used example in bayesian networks because it illustrates reasoning under uncertainty. The gist of the example is given in fig. 3. It involves a simple network of events and conditional probabilities.

The events are: Burglary (B), Earthquake (E), Alarm (A), Mary calls (M) and John calls (J). The initial events B and E are assumed to be independent events that occur with probabilities $P(B)$ and $P(E)$, respectively. There is an alarm system that can be triggered by either of the initial events B and E. The probability of the alarm going off is a conditional probability given that B and E have occurred. One denotes these probabilities, as per usual, by $P(A \mid B)$, and $P(A \mid E)$. There are two neighbors, Mary and John who have agreed to call if they hear the alarm. The probability that they do actually call is also a conditional probability denoted by $P(M \mid A)$ and $P(J \mid A)$, respectively.

We follow the convention of representing the (upper case) random variable X by the lower case x. Considering the probabilities given in fig. 3 we obtain the following specification:

$$b{:}0.001,$$
$$e{:}0.002,$$

For the table giving the probability $\mathrm{P}(M \mid A)$ we obtain the program:

$$\begin{aligned} p_{m|a}&:0.9, \\ p_{m|\overline{a}}&:0.05, \\ m &\leftarrow a \wedge p_{m|a}, \\ m &\leftarrow \neg a \wedge p_{m|\overline{a}}. \end{aligned}$$

The latter program can be simplified (abusing notation) by writing $m{:}0.9 \leftarrow a$ and $m{:}0.05 \leftarrow \neg a$. Similarly, for the probability $\mathrm{P}(J \mid A)$ we obtain

$$\begin{aligned} j&:0.7 \leftarrow a, \\ j&:0.01 \leftarrow \neg a, \end{aligned}$$

Finally, for the probability $\mathrm{P}(A \mid B \wedge E)$ we obtain

$$\begin{aligned} &a{:}0.95 \leftarrow b, e, \quad a{:}0.94 \leftarrow b, \overline{e}, \\ &a{:}0.29 \leftarrow \overline{b}, e, \quad a{:}0.001 \leftarrow \overline{b}, \overline{e}. \end{aligned}$$

One can then proceed as in the previous subsection and analyze this example. The details of such analysis are not given here since they are analogous, albeit admittedly more cumbersome.

5 Discussion and Future Work

This work is a first venture into expressing probability distributions using algebraic expressions derived from a logical program, in particular an ASP. We would like to point out that there is still much to explore concerning the full expressive power of logic programs and ASP programs. So far, we have not considered recursion, logical variables or functional symbols. Also, there is still little effort to articulate with the related fields, probabilistic logical programming, machine learning, inductive programming, *etc.*

The equivalence relation from definition 2 identifies the $s \subseteq e$ and $e \subseteq s$ cases. Relations that distinguish such cases might enable better relations between the models and processes from the stable models.

The example from section 4.2 shows that the theory, methodology, and tools, from bayesian networks can be adapted to our approach. The connection with Markov Fields [11] is left for future work. An example of a "program selection" application (as mentioned in item 4, section 2) is also left for future work.

Related with the remark at the end of section 4.1, on the tendency of $\hat{\theta}$ to under- or over- estimate θ, notice that the error function in (19) expresses only one of many possible "distances" between the empirical and prior distributions. Variations include normalizing this function by the size of $\mathcal{E}$ or using the Kullback-Leibler divergence. The key contribution of this function in this work is to find an optimal θ. Moreover, further experiments, not included in this paper, with $\alpha = 0.0$, lead to $\hat{\theta} \approx \gamma$, *i.e.* setting the prior noise to zero leads to full recovering θ from the observations.

We decided to set the measure of inconsistent events to 0 but, maybe, in some cases, we shouldn't. For example, since observations may be affected by noise, one can expect inconsistencies between the literals of an observation to occur.

Acknowledgements

This work is supported by NOVALINCS (UIDB/04516/2020) with the financial support of FCT.IP. The third author acknowledges the support of FCT - Fundação para a Ciência e Tecnologia under the project UIDP/04674/2020, and the research center CIMA – Centro de Investigação em Matemática e Aplicações.

The authors grateful to Lígia Henriques-Rodrigues, Matthias Knorr and Ricardo Gonçalves for valuable comments on a preliminary version of this paper, and Alice Martins for contributions on software.

References

[1] Marco Alberti, Elena Bellodi, Giuseppe Cota, Fabrizio Riguzzi & Riccardo Zese (2017): *cplint on SWISH: Probabilistic logical inference with a web browser*. *Intelligenza Artificiale* 11(1), pp. 47–64.

[2] Chitta Baral, Michael Gelfond & Nelson Rushton (2009): *Probabilistic reasoning with Answer Sets*. *Theory and Practice of Logic Programming* 9(1), pp. 57–144.

[3] Jeff Bezanson, Alan Edelman, Stefan Karpinski & Viral B. Shah (2017): *Julia: A Fresh Approach to Numerical Computing*. *SIAM Review* 59(1), pp. 65–98, doi:10.1137/141000671.

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

[5] Andrew Cropper, Sebastijan Dumančić, Richard Evans & Stephen H Muggleton (2022): *Inductive logic programming at 30*. *Machine Learning* 111(1), pp. 147–172.

[6] Luc De Raedt, Angelika Kimmig, Hannu Toivonen & M Veloso (2007): *ProbLog: A probabilistic Prolog and its application in link discovery*. In: *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence*, IJCAI-INT JOINT CONF ARTIF INTELL, pp. 2462–2467.

[7] Martin Gebser, Roland Kaminski, Benjamin Kaufmann & Torsten Schaub (2012): *Answer set solving in practice*. *Synthesis lectures on artificial intelligence and machine learning* 6(3), pp. 1–238.

[8] Stuart Geman & Donald Geman (1984): *Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images*. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(6), pp. 721–741, doi:10.1109/TPAMI.1984.4767596.

[9] Shashi Gowda, Yingbo Ma, Alessandro Cheli, Maja Gwozdz, Viral B Shah, Alan Edelman & Christopher Rackauckas (2021): *High-performance symbolic-numerics via multiple dispatch*. *arXiv preprint arXiv:2105.03949*.

[10] Bogumił Kamiński (2023): *JuliaData/DataFrames.jl*, doi:10.5281/zenodo.3376177.

[11] Ross Kindermann & J. Laurie Snell (1980): *Markov random fields and their applications*. *Contemporary Mathematics* 1, American Mathematical Society, Providence, RI.

[12] Joohyung Lee & Yi Wang (2016): *Weighted rules under the stable model semantics*. In: *Fifteenth international conference on the principles of knowledge representation and reasoning*.

[13] Joohyung Lee & Zhun Yang (2017): *LPMLN, Weak Constraints, and P-log*. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 31.

[14] Vladimir Lifschitz (2002): *Answer set programming and plan generation*. *Artificial Intelligence* 138(1), pp. 39–54, doi:https://doi.org/10.1016/S0004-3702(02)00186-8.

[15] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman & Anthony Scopatz (2017): *SymPy: symbolic computing in Python*. *PeerJ Computer Science* 3, p. e103, doi:10.7717/peerj-cs.103. Available at `https://doi.org/10.7717/peerj-cs.103`.

[16] Jukka Pajunen & Tomi Janhunen (2021): *Solution enumeration by optimality in Answer Set Programming*. *Theory and Practice of Logic Programming* 21(6), pp. 750–767.

[17] Judea Pearl (1988): *Probabilistic reasoning in intelligent systems: networks of plausible inference*. The Morgan Kaufmann Series in Representation and Reasoning, Morgan Kaufmann, San Mateo, CA.

[18] Fabrizio Riguzzi (2022): *Foundations of Probabilistic Logic Programming: Languages, Semantics, Inference and Learning*, 1 edition. River Publishers, New York, doi:10.1201/9781003338192.

[19] Taisuke Sato (1995): *A Statistical Learning Method for Logic Programs with Distribution Semantics*. In: *International Conference on Logic Programming*.

[20] Jozef L. Teugels (1990): *Some representations of the multivariate Bernoulli and binomial distributions*. *J. Multivariate Anal.* 32(2), pp. 256–268, doi:10.1016/0047-259X(90)90084-U. Available at `https://doi.org/10.1016/0047-259X(90)90084-U`.

[21] Victor Verreet, Vincent Derkinderen, Pedro Zuidberg Dos Martires & Luc De Raedt (2022): *Inference and learning with model uncertainty in probabilistic logic programs*. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 36, pp. 10060–10069.