An Algebraic Approach to Stochastic ASP

Francisco Coelho * fc@uevora.pt
Bruno Dinis † bruno.dinis@uevora.pt
Salvador Abreu ‡ spa@uevora.pt

July 18, 2023

Abstract

(*rewrite*) 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 program specifications with probabilistic facts can lead to characterizations of probability functions (*Why is this important? Is this what `others in sota' are trying do to?*) on the specification's domain.

1 Introduction and Motivation

(*Define and/or give references to all necessary concepts used in the paper*) (*state of the art; references*)

Answer set program (ASP) 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 hard and weak constraints.

(*references*) The distribution semantics (DS) is a key approach to extend logical representations with probabilistic reasoning. Probabilistic facts (PFs) are the most basic DS stochastic primitives and take the form of logical facts, a, labelled with probabilities, p, such as $p :: a$; Each PF represents a boolean

*Universidade de Évora, NOVALINCS, High Performance Computing Chair
†Universidade de Évora, CIMA, CMAFcIO
‡Universidade de Évora, NOVALINCS

random variable that is true with probability p and false with probability $\overline{p} = 1 - p$. A (consistent) combination of the PFs defines a total choice (TC) $t = \{p :: a, \ldots\}$ such that changed total choice c to t everywhere.

$$\mathrm{P}(T = t) = \prod_{a \in t} p \prod_{a \notin t} \overline{p}. \tag{1}$$

Our goal is to extend this probability, from TCs, to cover the *specification* domain. We use the term ``specification'' as set of rules and facts, plain and probabilistic, to decouple it from any computational semantics, implied, at least implicitly, by the term ``program''. We can foresee at least two key applications of this extended probability:

1. Support probabilistic reasoning/tasks on the specification domain.

2. Also, given a dataset and a divergence measure, the specification can be scored (by the divergence w.r.t. the *empiric* distribution of the dataset), and weighted or sorted amongst other specifications. These are key ingredients in algorithms searching, for example, optimal specifications of a dataset.

Our idea to extend probabilities starts with the stance that a specification describes an *observable system* and that observed events must be related with the SMs of that specification. From here, probabilities must be extended from total choices to SMs and then from SMs to any event.

Extending probability from TCs to SMs faces 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 specification) 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.

In a related work, [8], 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, the approach in that work uses Beta distributions in order to be able to learn the full distribution. 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. If so, we should mention this in the introduction.*) (*Also remark that our apporach remains algebraic in the way that we address the problems concerning the extension of probabilities.*)

(*cite ``SymPy: symbolic computing in Python'' --- why here? but cite ``The joy of probabilistic answer set programming: semantics, complexity, expressivity, inference'' and relate with our work.*)

(*Discuss the least informed strategy and the corolary that stable models should be conditionally independent on the total choice.*)

(*Give an outline of the paper.*)

2 A simple but fruitful example

(*Write an introduction to the section*)

Example 1. *Consider the following specification*

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

This specification has three stable models, $\overline{a}, ab$ and ac (see fig. 1). While it is straightforward to set $P(\overline{a}) = 0.7$, there is no further information to assign values to $P(ab)$ and $P(ac)$. Assuming that the stable models (SMs) are (probabilistically) independent, we can use a parameter θ such that

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

While uncertainty is inherent to the specification it can be mitigated with the help of a dataset: the parameter θ can be estimated from a empirical distribution (*or we can have a distribution of θ*). (*point to examples of this in following sections.*)

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

1. Observations can be used to estimate the value of the parameters (such as θ above and others entailed from further clauses).
2. (*What about the case where we already know a distribution of θ?*)
3. With a probability set for the stable models, we want to extend it to all the events of the specification domain.
4. This extended probability can then be related to the *empirical distribution*, using a probability divergence, such as Kullback-Leibler (KL); and the divergence value used as a *performance* measure of the specification with respect to the observations.

5. 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.

   (*Expand this:*) If observations are not consistent with the models of the specification, then the specification is wrong and must be changed.

Currently, we are addressing the problem of extending a probability function (possibly using parameters such as θ), defined on the SMs of a specification, to all the events of that specification. Of course, this extension must satisfy the Kolmogorov axioms of probability so that probabilistic reasoning is consistent with the ASP specification.

The conditional independence of stable worlds asserts the least informed strategy (*references?*) that we discussed in the introduction and make explicit here:

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

The stable models ab, ac from example 1 result from the clause $b \vee c \leftarrow a$ and the total choice a. These formulas alone imposes no relation between b and c (given a), so none should be assumed. Dependence relations are further discussed in section 5.1.

3 Extending Probabilities

(*Somewhere, we need to shift the language from extending* probabilities *to extending* measures)

The diagram in fig. 1 illustrates the problem of extending probabilities from TCs nodes to SMs and then to general events in a *node-wise* process. This quickly leads to coherence problems (*for example?*) concerning probability, with no clear systematic approach --- Instead, weight extension can be based in the relation an event has with the stable models.

Δ notation introduced in fig. 1.

3.1 An Equivalence Relation

Given an ASP specification, Introduce also the sets mentioned below (*how?*) 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* $t \in \mathcal{T} \iff t = a \vee \neg a$ and *stable models* $s \in \mathcal{S} \subset \mathcal{W}$.

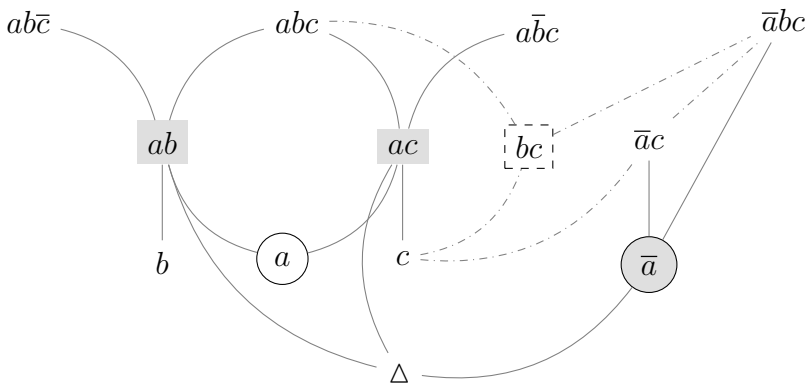


Figure 1: Events related to the stable models of example 1. The circle nodes are total choices and shaded nodes are stable models. The *empty event*, with no literals, is denoted by $\triangle$. Notice that the event bc is not related with any stable model.

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

This focus on the SMs leads to the following definition:

Definition 1. *A* stable structure *is a pair* (A, S) *where* A *is a set of atoms (*can be extracted from S.*) and* S *is a set of consistent events over* A*.*

(*expand this text to explain how the stable models form the basis of the equivalence relation*).

Definition 2. *The* stable core (SC) *of the event* $e \in \mathcal{E}$ *is*

$$[\![e]\!] := \{s \in \mathcal{S} \mid s \subseteq e \vee e \subseteq s\} \tag{3}$$

We now define an equivalence relation, $\sim$, so that two events are related if either both are inconsistent or both are consistent with the same stable core.

Definition 3. *For a given specification, let* $u, v \in \mathcal{E}$*. The equivalence relation* $\sim$ *is defined by*

$$u \sim v :\iff u, v \notin \mathcal{W} \vee \left(u, v \in \mathcal{W} \wedge [\![u]\!] = [\![v]\!]\right). \tag{4}$$

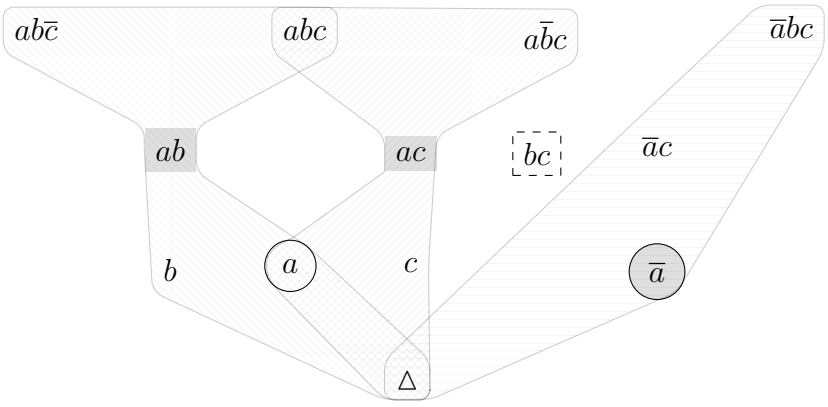


Figure 2: Classes (of consistent events) related to the stable models of example 1 are defined through intersections and inclusions. (*write the caption*)

Observe that the minimality of stable models implies that, in definition 2, either e is a stable model or one of $s \subseteq e, e \subseteq s$ is never true. This relation defines a partition of the events space, where each class holds a unique relation with the stable models. In particular, we denote each class by:

$$[e]_\sim = \begin{cases} \bot := \mathcal{E} \setminus \mathcal{W} & \text{if } e \in \mathcal{E} \setminus \mathcal{W}, \\ \{u \in \mathcal{W} \mid [\![u]\!] = [\![e]\!]\} & \text{if } e \in \mathcal{W}, \end{cases} \tag{5}$$

The subsets of the stable models, together with $\bot$, form a set of representatives. Consider again Example 1. As previously mentioned, the stable models are $\mathcal{S} = \bar{a}, ab, ac$ so the quotient set of this relation is:

$$[\mathcal{E}]_\sim = \left\{\bot, \Diamond, [\bar{a}]_\sim, [ab]_\sim, [ac]_\sim, [\bar{a}, ab]_\sim, [\bar{a}, ac]_\sim, [ab, ac]_\sim, [\bar{a}, ab, ac]_\sim\right\} \tag{6}$$

where $\Diamond$ denotes both the class of *independent* events e such that $[\![e]\!] = \emptyset$ and its core (which is the emptyset). We have:

Core, $[\![e]\!]$	**Class**, $[e]_\sim$	**Size**, $\#[e]_\sim$
$\bot$	$a\bar{a}, \ldots$	37
$\Diamond$	$\bar{b}, \bar{c}, bc, \bar{b}a, \bar{b}c, \bar{b}\bar{c}, \bar{c}a, \bar{c}b, \bar{b}\bar{c}a$	9
$\bar{a}$	$\bar{a}, \bar{a}b, \bar{a}c, \bar{a}\bar{b}, \bar{a}\bar{c}, \bar{a}bc, \bar{a}b\bar{c}, \bar{a}\bar{b}c, \bar{a}\bar{b}\bar{c}$	9
ab	$b, ab, ab\bar{c}$	3
ac	$c, ac, a\bar{b}c$	3
$\bar{a}, ab$	$\emptyset$	0
$\bar{a}, ac$	$\emptyset$	0
ab, ac	a, abc	2
$\bar{a}, ab, ac$	$\triangle$	1
Ω	all events	64

- Since all events within an equivalence class are in relation with a specific set of stable models, *weights, including probability, should be constant within classes*:
$$\forall u \in [e]_{\sim} \left(\mu(u) = \mu(e)\right).$$
- So, instead of dealing with $64 = 2^6$ events, we consider the $9 = 2^3 + 1$ classes, well defined in terms of combinations of the stable models. In general, we have *much more* stable models than literals. Nevertheless, the equivalence classes allow us to propagate probabilities from total choices to events, as explained in the next subsection.

3.2 From Total Choices to Events

(*Check adaptation*) Our path to set a probability measure on $\mathcal{E}$ has two phases:

1. Extending the probabilities, *as weights*, from the total choices to events.
2. Normalization of the weights.

The ``extension'' phase, traced by equations (1) and (7 --- 13), starts with the weight (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. (4), to (general) events, $\mu(e)$, including (consistent) worlds.

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

$$\mu(t) := \mathrm{P}(T = t) = \prod_{a \in t} p \prod_{a \notin t} \overline{p} \tag{7}$$

Stable Models. Each total choice t, together with the rules and the other facts of a specification, defines a set of stable models associated with that choice, that we denote by $\widehat{t}$ (*put this in the introduction, where core concepts are presented*).

Given a stable model $s \in \mathcal{S}$, a total choice t, and variables/values $\theta_{s,t} \in [0, 1]$,

$$\mu(s, t) := \begin{cases} \theta_{s,t} & \text{if } s \in \widehat{t} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

such that $\sum_{s \in \widehat{t}} \theta_{s,t} = 1$.

Classes. Each class is either the inconsistent class, $\bot$, 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:

$$\mu(\bot, t) := 0. \tag{9}$$

Independent Class. A world that neither contains nor is contained in a stable model describes a case that, according to the specification, shouldn't exist. So the respective weight is set to zero:

$$\mu(\Diamond, t) := 0. \tag{10}$$

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

$$\mu\big([e]_\sim, t\big) := \sum_{k=1}^{n} \mu(s_k, t)\,, \text{ if } [\![e]\!] = \{s_1, \ldots, s_n\}\,. \tag{11}$$

and

$$\mu\big([e]_\sim\big) := \sum_{t \in \mathcal{T}} \mu\big([e]_\sim, t\big)\, \mu(t)\,. \tag{12}$$

Events. Each (general) event e is in the class defined by its stable core, $[\![e]\!]$. So, we set:

$$\mu(e, t) := \frac{\mu\big([e]_\sim, t\big)}{\#[e]_\sim}. \tag{13}$$

and

$$\mu(e) := \sum_{t \in \mathcal{T}} \mu(e, t)\, \mu(t)\,. \tag{14}$$

- *(Remark that $\mu(\bot, t) = 0$ is independent of the total choice.)*
- Consider the event bc. Since $[bc]_\sim = \Diamond$, from eq. (10) we get $\mu(bc) = 0$.
- *(Remark that equation (14), together with observations, can be used to learn about the* initial *probabilities of the atoms, in the specification.)*

The $\theta_{s,t}$ parameters in equation (8) express the *specification's* lack of knowledge about the weight assignment, when a single total choice entails more than one stable model. In that case, how to distribute the respective weights? Our proposal to address this problem consists in assigning an unknown weight, $\theta_{s,t}$, 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 (11) results from conditional independence of stable models.

4 Developed Examples

4.1 The SBF Example

We continue with the specification from Equation (2).

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

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

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

$$\theta_{ab,\overline{a}} = \theta_{ac,\overline{a}} = \theta_{\overline{a},a} = 0 \text{ and } \theta_{\overline{a},\overline{a}} = 1, \theta_{ab,a} = \theta, \theta_{ac,a} = \overline{\theta}$$

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

Classes. Following the definitions in eqs. (3) to (5) and in eqs. (9) to (11) we get the following quotient set (ignoring $\bot$ and $\Diamond$), and weights:

$[\![e]\!]$	$\mu(s_k, t = \overline{a})$	$\mu(s_k, t = a)$	$\mu\big([e]_\sim\big) = \sum_t \mu\big([e]_\sim, t\big)\, \mu(t)$
$\overline{a}$	1		0.7
ab		θ	0.3θ
ac		$\overline{\theta}$	$0.3\overline{\theta}$
$\overline{a}, ab$	$1, 0$	$0, \theta$	$0.7 + 0.3\theta$
$\overline{a}, ac$	$1, 0$	$0, \overline{\theta}$	$0.7 + 0.3\overline{\theta}$
ab, ac		$\theta, \overline{\theta}$	0.3
$\overline{a}, ab, ac$	$1, 0, 0$	$0, \theta, \overline{\theta}$	1

Normalization. To get a weight that sums up to one, we compute the *normalization factor*. Since $\mu(\cdot)$ is constant on classes,(*prove that we get a probability.*)

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

that divides the weight function into a normalized weight

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

such that

$$\sum_{e \in \mathcal{E}} \mathrm{P}(e) = 1.$$

For the SBF example,

$[\![e]\!]$	$\#[e]_\sim$	$\mu([e]_\sim)$	$\mu(e)$	$\mathrm{P}(e)$
$\bot$	37	0	0	0
$\Diamond$	9	0	0	0
$\overline{a}$	9	$\frac{7}{10}$	$\frac{7}{90}$	$\frac{7}{792}$
ab	3	$\frac{3\theta}{10}$	$\frac{\theta}{10}$	$\frac{\theta}{88}$
ac	3	$\frac{3\overline{\theta}}{10}$	$\frac{\overline{\theta}}{10}$	$\frac{\overline{\theta}}{88}$
$\overline{a}, ab$	0	$\frac{7+3\theta}{10}$	0	0
$\overline{a}, ac$	0	$\frac{7+3\overline{\theta}}{10}$	0	0
ab, ac	2	$\frac{3}{10}$	$\frac{3}{20}$	$\frac{3}{176}$
$\overline{a}, ab, ac$	1	1	1	$\frac{5}{176}$
		$Z = \frac{44}{5}$		

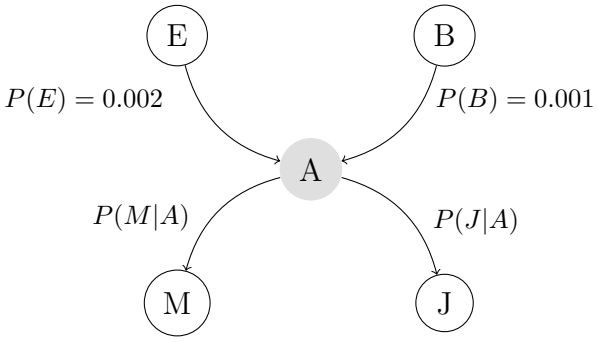
(*Continue this example with a set of observations to estimate θ and try to show some more. For example, that the resulting distribution is not very good when $t = \overline{a}$. Also gather a sample following the specification.*)

4.2 An example involving Bayesian networks

Comentários:

- Há uma macro, \pr{A}, para denotar a função de probabilidade, $\mathrm{P}(A)$ em vez de $P(A)$. Já agora, para a condicional também há um comando, \given: $\mathrm{P}(A \mid B)$.
- E, claro, para factos+probabilidades: $p :: a$.
- A designação dos `pesos' não está consistente: *pj_a* e *a_be*. Fiz uma macro (*hehe*) para sistematizar isto: *pa_bnc*.
- Nos programas, alinhei pelos factos. Isto é, $0.3 :: a$ e $a \leftarrow b$ alinham pelo (fim do) a.

As it turns out, our framework is suitable to deal with more sophisticated cases, ~~for example~~/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 [6] works in our setting. This example is a commonly used example in Bayesian networks because it illustrates reasoning



$P(M|A)$

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

$P(J|A)$

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

$P(A|B \wedge E)$

		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

under uncertainty. The gist of 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 $P(A|B)$, and $P(A|E)$. There are two neighbours, 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 $P(M|A)$ and $P(J|A)$, respectively.

Considering the probabilities given in fig. 3 we obtain the following specification

$$0.001 :: b,$$
$$0.002 :: e,$$

For the table giving the probability $P(M|A)$ we obtain the specification:

$$
\begin{aligned}
0.9 :: & \, pm\_a, \\
0.05 :: & \, pm\_na, \\
m \leftarrow & \, a, pm\_a, \\
\neg m \leftarrow & \, a, \neg pm\_a.
\end{aligned}
$$

This latter specification can be simplified by writing $0.9 :: m \leftarrow a$ and $0.05 :: m \leftarrow \neg a$.

Similarly, for the probability $P(J|A)$ we obtain

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

Again, this can be simplified by writing $0.7 :: j \leftarrow a$ and $0.01 :: j \leftarrow \neg a$.

Finally, for the probability $P(A|B \wedge E)$ we obtain

$$
\begin{aligned}
0.95 :: & \, a\_be, \\
0.94 :: & \, a\_bne, \\
0.29 :: & \, a\_nbe, \\
0.001 :: & \, a\_nbne, \\
a \leftarrow & \, b, e, a\_be, \\
\neg a \leftarrow & \, b, e, \neg a\_be, \\
a \leftarrow & \, b, e, a\_bne, \\
\neg a \leftarrow & \, b, e, \neg a\_bne, \\
a \leftarrow & \, b, e, a\_nbe, \\
\neg a \leftarrow & \, b, e, \neg a\_nbe, \\
a \leftarrow & \, b, e, a\_nbne, \\
\neg a \leftarrow & \, b, e, \neg a\_nbne.
\end{aligned}
$$

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

5 Discussion

- Changed from $\prod$ to $\sum$ to represent ``either'' instead of ``both'' since the later is not consistent with the ``only one stable model at a time'' assumption.

- (*The `up and down' choice in the equivalence relation and the possibility of describing any probability distribution.*)
- (*Remark that no benchmark was done with other SOTA efforts.*)
- (*The possibility to `import' bayesian theory and tools to this study.*)

5.1 Dependence

Our basic assertion about dependence relations between atoms of the underlying system is that they can be *explicitly expressed in the specification*. And, in that case, they should be.

For example, a dependence relation between b and c can be expressed by $b \leftarrow c \wedge d$, where d is an atomic choice that explicitly expresses the dependence between b and c. One would get, for example, a specification such as

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

with 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 ad. Notice that no stable model s contains adc because (i) adb is a stable model and (ii) if $adc \subset s$ then $b \in s$ so $adb \subset s$.

Following equations (**??**) and (**??**) What are these equations? this entails

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

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

so, for $C = ad$,

$$\begin{aligned} \mathrm{P}(W = b) &= \frac{2}{4} \\ \mathrm{P}(W = c) &= \frac{1}{4} \\ \mathrm{P}(W = bc) &= \frac{1}{4} \\ &\neq \mathrm{P}(W = b)\,\mathrm{P}(W = c) \end{aligned}$$

i.e. the events $W = b$ and $W = c$ are dependent and that dependence results directly from the segment $0.2 :: d, b \leftarrow c \wedge d$ in the specification.

Why does this not contradict Assumption 1?

Todo

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

5.2 Future Work

(*develop this section.*)

- The measure of the inconsistent events doesn't need to be set to 0 and, maybe, in some cases, it shouldn't.
- The physical system might have *latent* variables, possibly also represented in the specification. These variables are never observed, so observations should be concentrated *somewhere else*.
- Comment on the possibility of extending equation (14) with parameters expressing further uncertainties, enabling a tuning of the model's total choices, given observations.

$$\mu(e) := \sum_{c \in \mathcal{T}} \mu(e, c)\, \theta_c.$$

Acknowledgements

This work is supported by NOVA**LINCS** (UIDB/04516/2020) with the financial support of FCT.IP.

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] Aaron Meurer et al. ``SymPy: symbolic computing in Python''. In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[5] Aaron Meurer et al. ``SymPy: symbolic computing in Python''. In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[6] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. The Morgan Kaufmann Series in Representation and Reasoning. Morgan Kaufmann, San Mateo, CA, 1988, pp. xx+552. ISBN: 0-934613-73-7.

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

[8] 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.