A Semantics For Probabilistic Answer Set Programs With Incomplete Stochastic Knowledge

David Tuckey, Krysia Broda and Alessandra Russo

Imperial College London, London UK

Abstract

Some probabilistic answer set programs (PASP) semantics assign probabilities to sets of answer sets and implicitly assume these answer sets to be equiprobable. While this is a common choice in probability theory, it leads to unnatural behaviours with PASPs. We argue that the user should have a level of control over what assumption is used to obtain a probability distribution when the stochastic knowledge is incomplete. To this end, we introduce the Incomplete Knowledge Semantics (IKS) for probabilistic answer set programs. We take inspiration from the field of decision making under ignorance. Given a cost function, represented by a user-defined ordering over answer sets through weak constraints, we use the notion of Ordered Weighted Averaging (OWA) operator to distribute the probability over a set of answer sets accordingly to the user's level of optimism. The more optimistic (or pessimistic) a user is, the more (or less) probability is assigned to the more optimal answer sets. We present an implementation and showcase the behaviour of this semantics on simple examples. We also highlight the impact that different OWA operators have on weight learning, showing that the equiprobability assumption is not always the best option.

Keywords

Answer Set Programming, Probabilistic, Weight learning

1. Introduction

Probabilistic logic programming is a seamless method to incorporate structured knowledge together with probabilistic inference. It provides an easy formalism to represent complex relational models [1, 2, 3] and is now the foundation of a class of neural-symbolic models [4, 5]. In the family of probabilistic logic programs (PLP) presented in this paper, a PLP can be defined by a logic program and a set of independent probabilistic facts which help assign probabilities to sets of models (also referred to as possible worlds). When the logic program in a PLP is stratified, meaning it only accepts one minimum model, each set of models mentioned previously is a singleton and we give the probability of the set to its only member. This corresponds to the setting of the Distribution Semantics [6]. In this paper we will focus on the case where the logic programs are *not* stratified and can therefore have multiple (minimal) models. We consider PLPs composed of probabilistic facts and answer set programs, thus we talk of probabilistic answer set programming. Two examples of PASPs are given in Listing 1 and Listing 2.

In Listing 1, the PASP includes the independent probabilistic facts *rain* and *wind*, with probability 0.12 and 0.65 of being true, respectively. This program represents four different

ASPOC 2022: 15th Workshop on Answer Set Programming and Other Computing Paradigms, July 31, 2022, Haifa, Israel
dwt17@ic.ac.uk (D. Tuckey); k.broda@ic.ac.uk (K. Broda); a.russo@ic.ac.uk (A. Russo)

```
0.12::rain.
0.65::wind.
sun.
run :- sun, not wind, not walk.
walk :- sun, not rain, not run.
listen(rock) :- walk, not listen(classical).
listen(classical) :- walk, not listen(rock).
```

Listing 1: Run/Walk example: two probabilistic facts and a non-stratified ASP

scenarios: *rain* and *wind* true, *rain* true and *wind* false, *rain* false and *wind* true and finally *rain* and *wind* false. For the first two scenarios, the corresponding ASP (obtained by adding the true facts to the non probabilistic part of Listing 1) have only one answer set. However in the other two cases, the corresponding ASP programs have multiple answer sets. For instance, in the fourth scenario we have the following three answer sets: {*sun, run*}, {*sun, walk, listen(rock)*} and {*sun, walk, listen(classical)*}. This scenario associates a probability of $p_4 = (1-0.12)*(1-0.65) = 0.308$ to these three answer sets. However, to obtain a probability distribution over answer sets, we need to somehow distribute the probability p_4 among them. Unfortunately the program itself does not provide any information about how to proceed. Informally, the stochastic knowledge contained in the program is insufficient to naturally obtain a probability distribution over answer sets.

An arbitrary choice must be introduced to choose how to distribute this probability to the multiple answer sets. A method used by some PASP semantics consists of equally dividing the probability among the answer sets of a specific scenario [4, 7], in the example above we would assign $p_4/3$ to each of the three answer sets. The Credal Semantics however [8, 9] accepts all possible redistribution of probability, meaning that there exist an infinite amount of probability distributions in our example. The equiprobability assumption, seemingly natural, can lead to unexpected behaviours even on simple programs (as we show in Section 6) and doesn't offer any flexibility or information about the underlying uncertainty. In contrast, the Credal Semantics highlights the full range of possible probability distributions but is less intuitive to use in learning settings where the data needs to come in a very specific shape [10]. A PASP can represent an infinity of probability distributions and choosing the appropriate one for a given application has to be context dependent, guided by expert knowledge on the application domain. The same program, depending on the way chosen to distribute the probability in scenarios where there are multiple answer sets, might yield any probability between 0 and 1 for a given query (an example is given in Section 6). As so, tools are required to allow users to make use of their own chosen assumption on how to distribute probabilities when faced with multiple answer sets for a specific scenario.

In this paper we propose a novel method that allows users to encode the arbitrary assumptions they want to use to compute the end probability. As such the user controls which probability distribution, out of the potentially infinite set of probability distributions described by the program, will be used to answer their queries. We take inspiration from the field of *decision making under ignorance* which deals with this exact problem: how to assign a probability

distribution over a set of elements based on a user-defined preference. We propose a novel semantics, called the *Incomplete Knowledge Semantics* (IKS), that applies the notion of Ordered Weighted Averaging (OWA) operators [11], which are user-defined distributions over an ordered set of elements, to assign probabilities to multiple answer sets. The IKS uses weak constraints added to the PASP to order the answer sets in a given scenario, thus declaring the ordering for the OWA operator to work on. We show on simple examples that this new semantics allows the exploration of the different probability distributions that can come from a single program in an intuitive way, of which the semantics based on equiprobability are a special case.

The paper makes the following contributions:

- Define a novel semantics for PASPs by using user-defined OWA operators in place of a single arbitrary assumption.
- Present an implementation[1] of the IKS which allows the user to declare OWA operators using a single number in $[0, 1]$.
- Demonstrate the behaviour of this semantics, using simple examples, when predicting marginal distribution and learning parameters.

We start with explaining in Section 2 the field of decision making under ignorance and introducing the notion of OWA operator. We then present in Section 3 the Incomplete Knowledge Semantics, and explain in Section 4 how we approached the problem of generating OWA operator in our implementation. In Section 5, we explain the differences between our proposed IKS and other probabilistic answer set programming semantics, and show, in Sections 6 and 7, the different behaviours of the IKS both in predicting marginal likelihood and parameter learning. We conclude the paper with a summary and directions for future work.

2. Decision making under ignorance

A decision making problem can be represented as shown in equation (1) (from [11]), where a decision maker is faced with m choices A_i but the "state of nature" could be any of the n worlds S_j. C_{ij} corresponds to the reward of choosing A_i in the state of nature S_j.

$$\begin{array}{c} \\ A_1 \\ \vdots \\ A_i \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{cccccc} S_1 & \dots & S_j & \dots & S_n \end{array} \\ \begin{pmatrix} C_{11} & \dots & C_{1j} & \dots & C_{1n} \\ \vdots & & \vdots & & \vdots \\ C_{i1} & \dots & C_{ij} & \dots & C_{in} \\ \vdots & & \vdots & & \vdots \\ C_{m1} & \dots & C_{mj} & \dots & C_{mn} \end{pmatrix} \end{array} \tag{1}$$

A decision making problem consists in choosing the best action to maximise the reward. In the framework of decision making under uncertainty, we are given a probability distribution over the worlds S_j. A solution is to calculate the expected reward for each action A_i using $\sum_j C_{ij} P(S_j)$ and select the action with the highest expected reward.

[1] The library and binary are available at https://github.com/David-Tuc/IKS

In the context of decision making under ignorance, no distribution over the possible worlds is given. Thus the decision maker needs to assume a specific decision attitude in order to associate to each action the equivalent of an expected reward. For example, with a *pessimistic attitude* the decision maker associates to each action A_i its worst possible outcome C_{ij} and then chooses the action with the best one (the best worse outcome). In the same way, a decision maker with an *optimistic attitude* associates the best possible outcome to each action, and then selects the best action. Decision strategies can also be more nuanced and in between the optimistic and pessimistic attitudes, where the outcome associated to an action is a function of its different rewards. Mathematically, the decision maker defines an aggregate function $f : \mathbb{R}^n \to \mathbb{R}$, calculates $V_{A_i} = f(C_{i1}, ..., C_{in})$ for each action A_i and then selects the best action $A^* = argmax_{A_i} V_{A_i}$.

A formulation for aggregate functions has been given by Yager [11] in the form of Ordered Weighted Averaging (OWA) operators.

Definition 2.1 (OWA operator). *An OWA operator is a function $f : \mathbb{R}^n \to \mathbb{R}$ with n parameters $\boldsymbol{w}_i$ such as $\sum_i \boldsymbol{w}_i = 1$ and $\forall i, \boldsymbol{w}_i \in [0, 1]$. Given $\boldsymbol{u} \in \mathbb{R}^n$, $f(\boldsymbol{u}) = \sum_{i=1}^{n} \Phi(\boldsymbol{u})_i \boldsymbol{w}_i$ where $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ is a function that orders a vector in decreasing order.*

For example, given $\mathbf{u} = (10, 30, 20, 15)$ and $\mathbf{w} = [0.5, 0.3, 0.05, 0.15]$, the result of the corresponding OWA operator is $f(10, 30, 20, 15) = 30*0.5+20*0.3+15*0.05+10*0.15 = 23.25$. Here the numbers $10, 30, 20, 15$ are the rewards C_{ij} for a certain choice A_i. Given the correspondence between an OWA operator f and its weight vector $\mathbf{w}$, we will refer to OWA operators by their weight vector.

OWA operators are a simple way of computing a preference over a set of worlds of which we know only an ordering : $\mathbf{w}$ is the importance to give to each world ordered by their rewards $\mathbf{u}$. Some particular decision strategies have specific OWA operators: the pessimistic attitude, optimistic attitude and normative attitude are respectively represented using the following $\mathbf{w}_{pess}$, $\mathbf{w}_{opt}$ and $\mathbf{w}_{eq}$:

$$\mathbf{w}_{pess} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{w}_{opt} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \mathbf{w}_{eq} = \begin{pmatrix} \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{pmatrix}$$

The pessimistic attitude attributes all the weight to the last world in the ordering, the optimistic attitude assigns it to the first world in the ordering and the normative attitude attributes the same importance to all worlds.

OWA operators can be interpreted as human induced probability distribution over possible worlds, where $\mathbf{w}_i$ is the probability of the world with the i-th best outcome. We will make use of this interpretation to define the Incomplete Knowledge Semantics (IKS). While in decision making the ordering is given by a notion of cost or reward (C_{ij}), we will use weak constraints to obtain such ordering and we assume these to be domain specific and human defined. For a given PASP, the IKS uses OWA operators in each of the scenarios for which it obtains multiple answer sets, which are the possible worlds obtained from a specific scenario.

3. Incomplete Knowledge Semantics

We assume the syntax and semantics of Answer Set programs as defined by the ASP-CORE 2 [12]. We consider a probabilistic answer set program P to be composed of an ASP P_{ASP} and a set of independent annotated disjunctions P_{disj}. An annotated disjunction (AD) is of the form

$$p^i_1 :: t^i_1; ...; p^i_{n^i} :: t^i_{n^i} \text{ :- } b^i_1, ..., b^i_{m^i}$$

where $t^i_1, ..., t^i_{n^i}$ are atoms, $p^i_j \in [0, 1]$ with $\Sigma^{n^i}_{j=1} p^i_j \leq 1$, the body $b^i_1, .., b^i_{m^i}$ are literals and i is the index of the AD in P_{disj}[2]. Informally, when the body condition of an AD is fulfilled, its head represents a random variable that take value t^i_j with probability p^i_j. For the following definition, we will note $P(t^i_j) = p^i_j$.

Definition 3.1 (Total Choice). *Let $k = |P_{disj}|$ be the number of ADs, a **total choice** $C = \{t^1_a, t^2_b ... t^k_c\} \in TC_P$ is a set containing one ground atom from each AD's head. Each total choice C corresponds to a possible world in the probabilistic sense and has an associated probability $P(C) = \Pi^k_{i=1} P(C_i)$ as ADs are independent.*

Each total choice corresponds to a deterministic setting, where an outcome has been chosen for each random variable represented by the ADs. There exist one total choice for each possible combination of outcome of all the ADs. This deterministic setting (the total choice) is given the joint probability of observing all the random variable in their given state, which is the product of the probabilities of each individual outcome as the random variables represented by the ADs are assumed independent. We consider the answer sets of the ASPs $P_{ASP} \bigcup C$ (we add to the ASP the atoms in C as facts) for each total choice C. We denote the set of answer sets of $P_{ASP} \bigcup C$ with $AS(C)$ and informally refer to its elements as the answer sets for the total choice C. We also denote the full set of answer sets of P as $AS(P) = \bigcup_{C \in TC_P} AS(C)$. Similarly to Cozman and Mauá [8] we say that a program is **consistent** if it has at least one answer set for each total choice, i.e. $AS(C) \neq \emptyset$. We define a probability distribution over the sets $AS(C)$ of answer sets of P as $P(AS(C)) = P(C)$. Thus if all the total choices have only one single answer set, it directly yields a probability distribution over the answer sets, which coincides with the Distribution Semantics [6]. However if some total choices have multiple answer sets, we must redistribute the probability of the total choice to the answer sets. We make use of *weak constraints* which define a relation of dominance between answer sets, the answer set dominating all others being the most optimal. Given a set of weak constraints WC, we note $A \rhd_{WC} B$ if answer set A dominates answer set B under the weak constraints WC.

Definition 3.2 (Ordering of Answer Sets). *Given weak constraints WC and a set of answer sets AS of cardinality n, we define an ordering $\Phi_{WC}(AS)$ such as $\forall i = 1..n$, $\forall j = 1..i$, $\Phi_{WC}(AS)_i \not\rhd_{WC} \Phi_{WC}(AS)_j$. That is an answer set in position i does not dominate any of the preceding answer sets in the ordering.*

[2] In practice, for each AD where $\Sigma^{n^i}_{j=1} p^i_j < 1$ we add an extra annotated atom $p^i_{n^i+1} :: t^i_{n^i+1}$ to it where $p^i_{n^i+1} = 1 - \Sigma^{n^i}_{j=1} p^i_j$ and t_{n^i+1} is an atom that does not appear in P. Also, if the i-th AD in P_{disj} has a non-empty body, we add to P_{ASP} the clauses t^i_j :- $b^i_1, .., b^i_{m^i}, disj(i, j)$. for $j = 1...n^i$ and replace the AD in P_{disj} with $p^i_1 :: disj(i, 1); ...; p^i_{n^i} :: disj(i, n^i)$. If the ADs are not grounded, we first take all their groundings before doing the previous step.

Given weak constraints WC and for a given answer set as_i in an ordered set of answer set $\Phi_{WC}(AS)$, we define $Eq(as_i, AS, WC) = \{j | as_j \in \Phi_{WC}(AS), as_i \not\succ_{WC} as_j$ and $as_j \not\succ_{WC} as_i\}$ the indexes of the answer sets in $\Phi_{WC}(AS)$ which are not dominated by or dominate as_i, we will informally say that they share the same optimality as as_i. Note that $i \in Eq(as_i, AS, WC)$ and that the answer sets whose index are in $Eq(as_i, AS, WC)$ might be placed interchangeably in an ordering of AS as they do not dominate each other. We now make use of this notion of ordering of answer sets in tandem with OWA operators.

Definition 3.3 (Incomplete Knowledge Semantics). *Given a consistent PASP P with weak constraints WC and a user defined family of OWA operators $\forall n \in \mathbb{N}\ (\boldsymbol{w}^n \in \mathbb{R}^n)_n$. The incomplete knowledge semantics of P is a probability distribution over $AS(P)$ defined as follow:*

$$\forall C \in TC_P, P(as_i) = \frac{\sum_{j \in Eq(as_i, AS(C), WC)} \boldsymbol{w}^n_j}{|Eq(as_i, AS(C), WC)|} * P(AS(C)) \tag{2}$$

with $as_i = \Phi_{WC}(AS(C))_i$ the i-th answer set in the ordering of $AS(C)$ and $n = |AS(C)|$.

Definition 3.3 defines the IKS by attributing to each answer set in a total choice C a fraction of $P(C)$. This is done by ordering the answer sets from the most optimal to least optimal using the weak constraints and using a defined OWA operator to assign to each answer set its own probability, that we then multiply by $P(C) = P(AS(C))$. However as the ordering Φ_{WC} might not be unique, we average in Equation 2 over the answer sets that share the same optimality.

Example Let us consider the ASP program P given in Listing 1 augmented with the user defined weak constraint $:\sim$ run. [-1]. This makes the answer sets that contain the atom *run* have a lower weight (and be more optimal). There are four total choices $C_1 ... C_4$ and we give the details of the answer sets for each in Table 1. C_1 and C_2 both have only one answer set, so we assign directly their probabilities to the element in the singleton. For C_3 and C_4, we need to distribute the probability of the total choice to the answer sets using OWA operators. Studying the case of C_4, the answer set $\{$*sun, run*$\}$ is the most optimal while the two others have the same weight. As such, they will both always get the same probability, whatever the OWA operator. We give examples of redistribution of the probability $P(C_4) = 0.308$ to the three answer sets given different OWA operators in Table 2.

A query in the IKS has the form Q/O, where Q is a set of literals and O is the family of OWA operators to use. In our implementation, O is a real number in $[0, 1]$ (or a keyword as explained in the next section). The probability of Q/O is the sum of the probabilities of the answer sets satisfying Q, the probability being calculated using the OWA operators defined O. We denote this probability as $P(Q/O)$. Queries may also include evidence $Q|E/O$, in which case $P(Q|E/O) = P(Q, E/O)/P(E/O)$.

Table 1
The four total choices and their corresponding answer sets for the program in Listing 1. The facts added to the two ADs to complete them are omitted.

Total choice	Probability	Answer sets
C_1	0.078	{{*rain*, *wind*, *sun*}}
C_2	0.042	{{*rain*, *run*, *sun*}}
C_3	0.572	{{*wind*, *walk*, *sun*, *listen(rock)*}, {*wind*, *walk*, *sun*, *listen(classical)*}}
C_4	0.308	{{*sun, run*}, {*sun, walk, listen(rock)*}, {*sun, walk, listen(classical)*}}

Table 2
Example of distribution of the probability $P(C_4) = 0.308$ to its three answer sets for the program in Listing 1 using the weak constraint ":∼ run. [-1]". The first answer set is more optimal than the two others. The latter have the same optimality so will always be assigned the same probability.

OWA operator	$P(\{$*sun, run*$\})$	$P(\{$*sun, walk, listen(rock)*$\})$	$P(\{$*sun, walk, listen(classical)*$\})$
$[0.333, 0.333, 0.333]$	0.1026	0.1026	0.1026
$[0.5, 0.5, 0]$	0.154	0.077	0.077
$[1, 0, 0]$	0.308	0	0
$[0, 0, 1]$	0	0.154	0.154
$[0.25, 0.5, 0.25]$	0.077	0.1155	0.1155

4. Computing Ordered Weighted Averaging operators

In order to compute the probabilities in the IKS, one needs to define OWA operators f_n for all the possible amounts of worlds n by setting the weights $\mathbf{w}^n$. Indeed, an OWA operator is required for each possible number of answer sets for a total choice. While this is simple for specific decision strategies like the optimistic, pessimistic and normative attitudes, it is also possible to generate more complex operators depending on a "level of optimism". Yager [11] defines the *optimism* metric for OWA operators as:

$$Opt(\mathbf{w}^n) = \sum_{i=1}^{n} \mathbf{w}_i^n * \frac{n-i}{n-1} \tag{3}$$

For any OWA operator $\mathbf{w}$, $Opt(\mathbf{w}) \in [0, 1]$. We have that $Opt(\mathbf{w}_{pess}) = 0$, $Opt(\mathbf{w}_{opt}) = 1$ and $Opt(\mathbf{w}_{eq}) = 0.5$. It is a criteria that measures how "optimistic" an OWA operator is. A more optimistic OWA operator ($Opt(\mathbf{w})$ close to 1) will assign more weights to the best worlds, while a less optimistic operator ($Opt(\mathbf{w})$ close to 0) will assign more weights to the worse worlds. It indicates at a high level the decision strategy adopted by the user. Given an optimism level, it is possible to generate an OWA operator such as it reflects this decision strategy.

In our implementation we make use of the method described by Chen et al. [13] adopting the normal distribution as a base. To generate an OWA operator $\mathbf{w}$ of size n with an optimism of o,

Table 3
Example of OWA operators for 9 worlds/answer sets. $\mathbf{w}_i$ are generated with an optimism of i using the algorithm from Chen et al. [13]. We also include $\mathbf{w}_{eq}$.

OWA operators	
$\mathbf{w}_0$	$[0, 0, 0, 0, 0, 0, 0, 0, 1]$
$\mathbf{w}_{0.1}$	$[0.002, 0.005, 0.01, 0.019, 0.032, 0.052, 0.083, 0.149, 0.648]$
$\mathbf{w}_{0.3}$	$[0.008, 0.023, 0.046, 0.078, 0.117, 0.156, 0.187, 0.201, 0.184]$
$\mathbf{w}_{0.5}$	$[0.051, 0.085, 0.124, 0.156, 0.168, 0.156, 0.124, 0.085, 0.051]$
$\mathbf{w}_{0.7}$	$[0.184, 0.201, 0.187, 0.156, 0.117, 0.078, 0.046, 0.023, 0.008]$
$\mathbf{w}_{0.9}$	$[0.648, 0.149, 0.083, 0.052, 0.032, 0.019, 0.01, 0.005, 0.002]$
$\mathbf{w}_1$	$[1, 0, 0, 0, 0, 0, 0, 0, 0]$
$\mathbf{w}_{eq}$	$[0.111, 0.111, 0.111, 0.111, 0.111, 0.111, 0.111, 0.111, 0.111]$

we first generate a vector of size n using the following formula:

$$\mathbf{v}_i = \frac{e^{-\frac{(i-\mu)}{2\sigma^2}}}{\sum_{j=1}^{n} e^{-\frac{(j-\mu)}{2\sigma^2}}} \tag{4}$$

With $\mu = \frac{1+n}{2}$ and $\sigma = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(i-\mu)^2}$. This intermediate operator has an optimism of 0.5 so we need to modify it to match our target optimism of o. We generate a specific adjustment matrix $\mathbf{U}$ such as $\mathbf{w} = \mathbf{U}\mathbf{v}$ gives $Opt(\mathbf{w}) = o$. For details on how to generate the adjustment matrix $\mathbf{U}$ refer to Algorithm 1 in Chen et al. [13]. We give examples of generated OWA operators using this algorithm in Table 3. As for $o = 0.5$ we do not need to modify $\mathbf{v}$, we have that $\mathbf{v} = \mathbf{w}_{0.5}$ in Table 3.

The method chosen to generate the OWA operator has an impact on the shape of the latter, as different operators can have the same optimism value. As an example, both $\mathbf{w}_{0.5}$ and $\mathbf{w}_{eq}$ given in Table 3 have an optimism value of 0.5. We use this specific generation method for computational reasons when n is big and also because the shape of the operator allows us to select a range of answer sets in total choices.

5. Related Works

We compare here the IKS with other semantics for probabilistic answer set programs. But firstly, one can note that when the logic program is stratified, the IKS is equivalent to the Distribution Semantics [6]. The IKS uses probabilities on independent facts and takes its syntax from Problog [14]. It is similar in its mechanism to P-log [7], though P-log doesn't expressly define

total choices, however it gives a single probability distribution by distributing the probability evenly when necessary. Neural-ASP [4] uses a very similar semantics to P-log by assuming equiprobability in each total choice, but has the particularity of not always defining a full proper probability distribution: the sum of probabilities of all the answer sets does not need to sum to one. The IKS allows the user to decide the assumption to redistribute probabilities instead of using equiprobability by default. The equiprobability assumption is a special cases of the IKS when we restrict ourselves to use $\mathbf{w}_{eq}$ exclusively. On the other hand, the Credal Semantics [8, 9] avoids the issue of using assumptions by accepting all possible probability distributions, giving a lower and upper probability to queries. Compared to the IKS, these bounds are the minimum and maximum probabilities that can be obtained using different weak constraints and OWA operators.

Other semantics for PASP differ in the way the probabilities are declared. PASP [15] defines probabilities directly on atoms in the program without assuming their independence. PrASP [16] defines probabilities on first-order-logic sentences that are translated to ASP. When the logic program is not stratified, both accept an infinite number of probability distributions as correct under their semantics, but both systems define a specific optimization algorithm to obtain a particular distribution. Finally, LP^{MLN} [17] defines weights on ASP clauses that are used to compute probabilities over answer sets by a normalization procedure. This semantics does not use the notion of total choices, however two answer sets that satisfy the same set of rules will be given an equal probability. Compared to these other semantics for PASPs, IKS focuses on allowing the user the degree of freedom to explore the different probability distributions that can be obtained from the program.

6. Behaviour of the IKS

We implement the IKS using the OWA operator generation algorithm mentioned in Section 4. The user selects the family of OWA operators to use by setting their optimism value. We show here how the different hyper-parameters (weak constraints and optimism) influence the downstream probabilities.

Example 1 We highlight here the fact that assuming equiprobability between answer sets in the same total choice is not always the most natural assumption. We consider the program in Listing 1. In the total choice C_4 there are three answer sets (see Table 1). Assigning $p(C_4)/3 = 0.102$ to each answer set seems to be the easiest option, assuming equiprobability between all three. But looking at the program, it would be more natural to assume equiprobability on the disjunctive events: *run*/*walk* and *listen(rock)*/*listen(classical)*. In this case, we would assign 0.154 to $\{$*run*, *sun*$\}$ and 0.077 to each of the two other answer sets.

The IKS avoids the issue of making assumptions in place of the user when faced with this kind of situation. Focusing again on C_4 and adding the weak constraint ":$\sim$ run. [-1]" to the program. The answer set with the atom *sun* is more optimal than the two others, which are equally optimal. An optimism of 1 gives all the probability to the $\{$*run*, *sun*$\}$ answer set while an optimism of 0 gives it none. Consider now an optimism of 0.5. Because we generate a Gaussian shaped OWA operator its weights are close to $[0.25, 0.5, 0.25]$, making that $P(\{$*run*, *sun*$\}) = 0.074$. Using

an optimism of 0.697 allows to have $P(\{run, sun\}) = 0.154$ and be in the case described above where equiprobability was assumed on the individual disjunctive events. Finally if instead of a generated OWA operator we use $\mathbf{w}_{eq}$, we obtain the equiprobability case where all three answer sets are assigned 0.102.

We give other examples of possible redistributions in Table 2. By doing the same process in all the total choices, we can compute the probability of queries over the entire program: $P(\{run\}/0) = 0.572$, $P(\{run\}/1) = 0.880$, $P(\{run\}/0.697) = 0.725$, $P(\{run\}/\mathbf{w}_{eq}) = 0.674$. Depending on the hyperparameters (OWA operator, optimism, weak constraint) chosen for running this program, we can obtain very different probabilities.

While the goal of the IKS is not to allow the user to match one or another equiprobability assumption, these methods are subsumed by the IKS which offers more granularity. Choosing a level of optimism is a simple way to define a rather complex decision strategies across all total choices.

Example 2 We now give a more complex setting which shows a case where, given the same program, the probability of a query might range anywhere between 0 and 1 depending on the assumption used to distribute the probabilities. This highlights the importance of choosing the appropriate assumption given the application domain. Consider a probabilistic version of the graph colouring problem where edges exist with a given probability. Each node can be coloured in one of three colours and adjacent nodes (connected by an edge) cannot be of the same colour. One application is radio frequency assignments, nodes are emitters, edges represent possible interference between two emitters and the colours are different radio frequencies. Consider that some emitters may be turned off some of the time, and that we consider worlds where more emitters are running to be "better". We formalise this problem in Listing 2 as a PASP. Nodes 1 to 4 are always on and nodes 5 to 8 can be on or off. The colours assigned are red, yellow or blue. The weak constraint ranks as more optimal the answer sets with more nodes present. In this program, a total choice corresponds to a combination of links between nodes being active. In each total choice there are multiple answer sets, each corresponding to a different coloring and differing as well in the amount of nodes present.

Take as an example the following query Q :"what is the probability that emitters 1 and 3 share the same frequency/colour ?". Depending on the context, the user might want to adopt a more optimistic or pessimistic decision attitude when querying the program given in Listing 2: if the user believes more emitters are functioning the optimism level chosen will be closer to 1. With an optimism of 1, the predicted probability $P(Q/1)$ is 0.289 while with an optimism of 0 we have $P(Q/0) = 0.333$. Finally using $\mathbf{w}_{eq}$ we obtain $P(Q/\mathbf{w}_{eq}) = 0.317$.

The weak constraints chosen also have an important impact on the predicted probability. For example, consider replacing the weak constraint in Listing 2 with

```
:~ colour(1,red), colour(3,red). [-1]
:~ colour(1,yellow), colour(3,yellow). [-1]
:~ colour(1,blue), colour(3,blue). [-1]
```

They order the answer sets such as the most optimal are the ones satisfying the query Q. Thus we obtain with an optimism of 0 $P(Q/0) = 0$ and with an optimism of 1 $P(Q/1) = 1$, which are the lower and upper bound for our query under the Credal Semantics. This is due to the fact

```
on(1..4).
sometime(5..8).
node(X) :- on(X).
{node(X)} :- sometime(X).
0.1::link(1,2).
0.7::link(1,5).
0.2::link(2,4).
0.1::link(4,6).
0.4::link(3,6).
0.4::link(2,7).
0.8::link(5,8).
0.95::link(8,3).
0.1::link(7,6).
:~ node(X). [-1, X]
1{colour(V, red); colour(V, yellow); colour(V, blue)}1 :- node(V).
:- link(V,U), colour(V,C), colour(U,C).
```

Listing 2: Graph colouring example

that when the optimism is 0, we give the probability of each total choice to the answer sets that do not satisfy Q, so Q is false in all the answer sets with a non zero probability. Inversely when the optimism chosen is 1, the probability of each total choice is exclusively given to answer sets which satisfy Q, making that the only answer sets with non zero probability are satisfying Q. Modifying the weak constraints have of course no effect when using $\mathbf{w}_{eq}$ as the OWA operator.

It is important to distinguish between the contribution of the optimism value and that of the weak constraints: the weak constraints define the ranking of the answer sets, meaning that they define a relative cost between possible answer sets. The optimism value defines which answer sets in the ordering are favored by the user.

7. The impact of optimism on learning

We have shown the possibility to manipulate the probabilities using the IKS. We show here the impact on parameter learning. It consists of learning the probability of some probabilistic facts in the program. We reuse the program from Listing 1 and generate custom datasets. The data is defined as follows: let $D = \{d_i\}$ be a collection of n data points of the form $d_i = (Q/O)$ where Q is a query and $O \in [0, 1]$ is the optimism to use for that particular data point. O might also be a keyword that designates a special OWA operator like $\mathbf{w}_{eq}$. We train the program by maximising the log-likelihood $L = \sum_i log(P(d_i))$. We implement this using gradient ascent, pre-computing the OWA operators and answer sets to have a differentiable graph. We initialize randomly the probabilities of the probabilistic facts in the program, compute L over the dataset and differentiate the loss with regards to the learnable probabilities to make a step (by mini-batch).

Table 4
Probability learned on the datasets D_1 and D_2 with our two different settings: setting 1 uses an optimism of 0.697 for all examples while setting 2 uses $\mathbf{w}_{eq}$ for all examples. The probability to learn are 0.12 for *wind* and 0.65 for *Rain*. Reported are the means with standard deviations over 10 runs.

Atom	Wind	Rain
Setting 1		
D_1	$0.118 \pm 5.3e^{-5}$	$0.651 \pm 1.8e^{-4}$
D_2	$0.106 \pm 4.75e^{-5}$	$0.721 \pm 1.9e^{-4}$
Setting 2		
D_1	$0.134 \pm 6.11e^{-5}$	$0.581 \pm 2.8e^{-4}$
D_2	$0.117 \pm 5.14e^{-5}$	$0.663 \pm 3.1e^{-4}$

Data generation We generate synthetic datasets based on the program from Listing 1. We consider that the observed atoms are the following: *run*, *walk*, *listen(rock)*, *listen(classical)*. Then there are four possible labels: {}, {*run*}, {*walk*, *listen(rock)*} and {*walk*, *listen(classical)*}. We generate the datasets by sampling these four labels at the frequency described by the following mechanism. We first sample the atoms *wind* and *rain* using their respective probabilities 0.65 and 0.12. If both are true then the label is {} and if only *rain* is true then the label is {*run*}. If only *wind* is true then we have two possibilities, we sample another random variable which is true with probability $p_{rock} = 0.5$, if true the label is {*walk*, *listen(rock)*} and if false the label is {*walk*, *listen(classical)*}. In the case where both *wind* and *rain* are false, we sample the atom *run* with probability p_{run}, which if true gives the label {*run*}. If false, we again have two options ({*walk*, *listen(rock)*} and {*walk*, *listen(classical)*}) which we sample using $p_{rock} = 0.5$.

We generate two datasets using this process, the first one D_1 using $p_{run} = 0.5$ which represents the case where someone would have chosen to run half of the times if faced with the choice of running and walking. The second dataset D_2 uses $p_{run} = \frac{1}{3}$ which represents the distribution that would be obtained if we were using the program in Listing 1 under a semantics that assumes equiprobability of answer sets.

Learning results We delete from Listing 1 the probabilities of the two probabilistic facts, add the weak constraint ":∼ run. [-1]" and attempt to learn from data. To do so, we initialize randomly the probabilities and use gradient ascent to maximize L for 50 epochs. Before learning, we need to consider which optimism value to use for each example in the dataset. We simplify here by assuming that all examples during a run have the same optimism so use the same OWA operators. We perform two runs for each dataset: one run where the optimism is 0.697 and one where the OWA operator used is $\mathbf{w}_{eq}$. The first setting semantically corresponds to the distribution used to generate D_1 while the second setting corresponds to D_2. We give the results of these experiments in table 4. As we can see, the first setting allows to learn the proper probabilities on D_1 while the second setting works well for D_2. However using setting 1 with D_2 and setting 2 with D_1, we obtain different probabilities from the targets.

This sanity check confirms that the choice of OWA operator impacts the learned probabilities. The IKS allows each data point to have its own OWA operators, avoiding a "one assumption fits all" all solution. In a realistic setting, the optimism, OWA operators and weak constraints

would either be given by expert knowledge during the data collection or would be treated as hyper-parameters to tune during training.

8. Conclusion

We have introduced in this paper a novel semantics for probabilistic answer set programs with incomplete knowledge, where the stochastic information contained might result in a number of different probability distributions. It allows the user to define the set of assumptions to use through the use of ordered weighted averaging operators, a user defined distribution taken from the field of decision making under ignorance. Using weak constraints to define an ordering over answer sets, OWA operators are used to assign probabilities when none are provided by the program itself. We present our implementation which uses a specific OWA operator generation algorithm based on the user's level of optimism. We show that for even simple PASPs, using different OWA operators have a big impact on the downstream probability distribution, and that some simple assumptions (like equiprobability) can lead to unnatural behaviours. Finally the impact of OWA operators is showed when performing parameter estimation, arguing that the choice of weak constraints, OWA operator and optimism can fall withing the realm of model selection. As future work, we will study the possibility to learn the structure of PASPs under the IKS and study the impact of the OWA operators on neural-symbolic settings.

References

[1] I. Thon, N. Landwehr, L. De Raedt, Stochastic relational processes: Efficient inference and applications, Machine Learning 82 (2011) 239–272. doi:10.1007/s10994-010-5213-8.

[2] D. Nitti, T. De Laet, L. De Raedt, Relational object tracking and learning, Proceedings - IEEE International Conference on Robotics and Automation (2014) 935–942. doi:10.1109/ICRA.2014.6906966.

[3] A. Groß, B. Kracher, J. M. Kraus, S. D. Kühlwein, A. S. Pfister, S. Wiese, K. Luckert, O. Pötz, T. Joos, D. Van Daele, L. De Raedt, M. Kühl, H. A. Kestler, Representing dynamic biological networks with multi-scale probabilistic models, Communications Biology 2 (2019) 1–12. doi:10.1038/s42003-018-0268-3.

[4] Z. Yang, A. Ishay, J. Lee, NeurASP: Embracing neural networks into answer set programming, IJCAI International Joint Conference on Artificial Intelligence 2021-Janua (2020) 1755–1762. doi:10.24963/ijcai.2020/243.

[5] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. D. Raedt, DeepProbLog: Neural Probabilistic Logic Programming, CoRR abs/1805.1 (2018). URL: http://arxiv.org/abs/1805.10872. arXiv:1805.10872.

[6] T. Sato, A Statistical Learning Method for Logic Programs with Distribution Semantics, in: ICLP, 1995.

[7] C. Baral, M. Gelfond, N. Rushton, Probabilistic reasoning with answer sets, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, 2004, pp. 21–33.

[8] F. G. Cozman, D. D. Mauá, On the Semantics and Complexity of Probabilistic Logic Programs, CoRR abs/1701.0 (2017). URL: http://arxiv.org/abs/1701.09000. arXiv:1701.09000.

[9] F. G. Cozman, D. D. Mauá, The joy of Probabilistic Answer Set Programming: Semantics, complexity, expressivity, inference, International Journal of Approximate Reasoning (2020) 91–101. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888613X20302012. doi:10.1016/j.ijar.2020.07.004.

[10] D. Tuckey, K. Broda, A. Russo, Towards Structure Learning under the Credal Semantics, CEUR Workshop Proceedings 2678 (2020).

[11] R. R. Yager, Decision Making Under Dempster-Shafer Uncertainties, International Journal of General Systems 20 (1992) 233–245. doi:10.1080/03081079208945033.

[12] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. A. Maratea, F. Ricca, T. Schaub, ASP-Core-2 Input Language Format, Theory and Practice of Logic Programming (2019). doi:10.1017/S1471068419000450. arXiv:1911.04326.

[13] Z. S. Chen, C. Yu, K. S. Chin, L. Martínez, An enhanced ordered weighted averaging operators generation algorithm with applications for multicriteria decision making, Applied Mathematical Modelling 71 (2019) 467–490. URL: https://doi.org/10.1016/j.apm.2019.02.042. doi:10.1016/j.apm.2019.02.042.

[14] D. Fierens, G. Van Den Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, L. De Raedt, Inference and learning in probabilistic logic programs using weighted Boolean formulas, Theory and Practice of Logic Programming 15 (2015) 358–401. URL: http://arxiv.org/abs/1304.6810http://dx.doi.org/10.1017/S1471068414000076. doi:10.1017/S1471068414000076. arXiv:1304.6810.

[15] E. Morais, M. Finger, Probabilistic Answer Set Programming, in: Proceedings - 2013 Brazilian Conference on Intelligent Systems, BRACIS 2013, 2013, pp. 150–156. doi:10.1109/BRACIS.2013.33.

[16] M. Nickles, A. Mileo, Probabilistic Inductive Logic Programming Based on Answer Set Programming, CoRR abs/1405.0 (2014). URL: http://arxiv.org/abs/1405.0720. arXiv:1405.0720.

[17] J. Lee, Y. Wang, A probabilistic extension of the stable model semantics, in: Logical Formalizations of Commonsense Reasoning - Papers from the AAAI Spring Symposium, Technical Report, volume SS-15-04, AI Access Foundation, 2015, pp. 96–102.