# A Brief Discussion about the Credal Semantics for Probabilistic Answer Set Programs

Damiano Azzolini[1]

[1] *Dipartimento di Scienze dell'Ambiente e della Prevenzione – University of Ferrara*

Abstract

Among the different logic-based programming languages, Answer Set Programming has emerged as an effective paradigm to solve complex combinatorial tasks. Since most of the real-world data are uncertain, several semantics have been proposed to extend Answer Set Programming to manage uncertainty, where rules are associated with a weight, or a probability, expressing a degree of belief about the truth value of certain atoms. In this paper, we focus on one of these semantics, the Credal Semantics, highlight some of the differences with other proposals, and discuss some possible future works.

**Keywords**

Probabilistic Answer Set Programming, Inference, Uncertainty

## 1. Introduction

Logic-based languages [1], such as Prolog [2] and Answer Set Programming (ASP) [3], are powerful formalisms to represent relational data and reason on them. ASP is particularly effective in modelling complex combinatorial domains, thanks to several syntactic constructs such as constraints and aggregates [4]. While with modern ASP solvers such as clingo [5] and WASP [6] it is possible to associate weights to rules to guide the search of optimal answer sets, they cannot natively model uncertain data described as probabilities attached to some of the atoms.

In the context of Logic Programming, probabilistic reasoning has attracted a lot of research interest in the past years, with several possible semantics [7, 8, 9, 10] and several probabilistic reasoning tools [7, 11, 12]. To represent uncertainty in ASP, there are some proposals such as P-log [13], LP$^{\mathrm{MLN}}$ [14], and the Credal Semantics (CS) [15, 16]. In the first part of the paper, we sketch the differences between the LP$^{\mathrm{MLN}}$ and the Credal Semantics. Then, we discuss an approach based on normalization to perform inference in programs where some worlds are unsatisfiable, not allowed by the Credal Semantics, and compare it with two other proposals (the L-Credal Semantics [17] and the SMProbLog semantics [18]) developed in the context of argumentation. Through the paper, we also highlight several directions for future works.

The paper is structured as follows: in Section 2 we introduce the basic concepts involving Probabilistic Logic Programming. In Section 3 we discuss the differences between the LP$^{\mathrm{MLN}}$ and the Credal Semantics and in Section 4 we focus on the inference task. Section 5 describes

*PLP 2023: The Tenth Workshop on Probabilistic Logic Programming*
damiano.azzolini@unife.it (D. Azzolini)
0000-0002-7133-2673 (D. Azzolini)

a possible solution to handle inconsistent worlds, Section 6 presents some related works, and Section 7 concludes the paper.

## 2. Probabilistic Logic Programming under the Distribution Semantics

Probabilistic Logic Programming under the Distribution Semantics (DS) [9] allows the definition of probabilistic facts, i.e., facts that are true with a certain probability. By considering the ProbLog [7] syntax, these can be defined with $\Pi :: f_i$ where $f_i$ is an atom and $\Pi \in [0,1]$ is its probability. Probabilistic facts identify *worlds*, i.e., logic programs obtained by fixing the truth values of the probabilistic facts[1]. With $n$ probabilistic facts there are $2^n$ (since every fact can be true or false) worlds. The probability of a world $\tau$ can be computed with the formula

$$P(\tau) = \prod_{f_i \in \tau} \Pi_i \cdot \prod_{\neg f_i \in \tau} (1 - \Pi_i). \tag{1}$$

The probability of a *query* $q$, a conjunction of ground atoms, can be computed as the sum of the probabilities of the worlds where the query is true, in formula: $P(q) = \sum_{\tau \models q} P(\tau)$.

**Example 1.** *Consider the following ProbLog program:*

```
0.4::pressure(carl).
0.3::high_sugar(carl).
faint(X):- pressure(X).
faint(X):- high_sugar(X).
```

*The first two lines contain probabilistic facts: the first states that carl has a pressure problem with probability 0.4 and the second states that he has a high blood sugar value with probability 0.3. The last two rules state that a person $X$ faints if it has a pressure problem or a high sugar value. We have $2^2 = 4$ possible worlds: $\{\}$ with an associated probability of $0.6 \cdot 0.7 = 0.42$, $\{pressure(carl)\}$ with an associated probability of $0.4 \cdot 0.7 = 0.28$, $\{high\_sugar(carl)\}$ with an associated probability of $0.6 \cdot 0.3 = 0.18$, and $\{pressure(carl)\ high\_sugar(carl)\}$ with an associated probability of $0.4 \cdot 0.3 = 0.12$. If we consider the query $q = faint(carl)$, we have $P(q) = 0.12 + 0.18 + 0.28 = 0.58$.*

Probabilistic facts are considered independent, but this is usually not a limitation on their expressivity since it is possible to encode any Bayesian network with ProbLog [19]. Note that probabilistic rules can be represented by adding a new probabilistic fact in the body of a rule with all the variables appearing in the head and body.

## 3. The $\text{LP}^{\text{MLN}}$ and the Credal Semantics

In this section, we focus on two of the possible semantics to represent uncertainty in answer set programs, namely the $\text{LP}^{\text{MLN}}$ [14, 20] and the Credal Semantics [16].

[1]In the following, we use the Greek letter $\tau$ to indicate worlds, since these are also called total choices, to avoid using the letter $w$, adopted to represent weights in the context of the $\text{LP}^{\text{MLN}}$ semantics.

### 3.1. The $\mathrm{LP}^{\mathrm{MLN}}$ Semantics

The authors of [14, 20] introduced the $\mathrm{LP}^{\mathrm{MLN}}$ language, with the goal to unify Markov Logic Networks (MLN) [21] and logic programs under the stable model semantics [22]. Programs in this language are composed of a set of weighted rules of the form $w : R$ where $R$ is a logic rule and $w$ is either a real number (soft rule) or the symbol $\alpha$ (hard rule), that denotes infinite weight. In the next part of the paper, we omit the $\alpha$ symbol before hard rules for programs following the $\mathrm{LP}^{\mathrm{MLN}}$ semantics. If we consider a ground $\mathrm{LP}^{\mathrm{MLN}}$ program $\mathcal{P}^{\mathrm{L}}$, the weight of an interpretation $I$, $W(I)$, is computed as

$$W(I) = \begin{cases} exp(\sum_{w:R\in\mathcal{P}^{\mathrm{L}}_{\mathrm{I}}} w) & \text{if } \mathrm{I} \in AS(\mathcal{P}^{\mathrm{L}}_{\mathrm{I}}) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{P}^{\mathrm{L}}_{\mathrm{I}}$ is the set of rules $R$ of $\mathcal{P}^{\mathrm{L}}$ such that $I \models R$ and $AS(\mathcal{P}^{\mathrm{L}}_{\mathrm{I}})$ is the set of its answer sets, and its probability as

$$P(I) = \lim_{\alpha\to\infty} \frac{W(I)}{\sum_{J\in AS(\mathcal{P}^{\mathrm{L}}_{\mathrm{I}})} W(J)}.$$

The above definition is general. However, if we consider only stable models that satisfy all the hard rules, the limit in the above formula can be ignored and the weight of an interpretation, $W(I)$, can be computed by considering only the set of soft rules, as discussed in [20].

The possible worlds of a MLN program are the stable models of $\mathrm{LP}^{\mathrm{MLN}}$ programs, and the probability of a query is a sharp probability value computed by considering a probability distribution over these models. If the probability of an interpretation is not 0, it is called a *probabilistic* stable model. If we consider a query $q$, its probability can be computed as the sum of the probabilities of the probabilistic stable models of the programs where the query is true [20], i.e.,

$$P^L(q) = \sum_{I\models q} P(I). \tag{2}$$

### 3.2. The Credal Semantics

An alternative semantics to represent uncertainty within Answer Set Programming is the Credal Semantics (CS) [16], that has been already applied in multiple scenarios [23, 24, 25]. The CS allows the definition of probabilistic facts and assigns a probability range to a query $q$. We call these programs "probabilistic answer set programs". Differently from the $\mathrm{LP}^{\mathrm{MLN}}$ semantics, the CS is similar to the Distribution Semantics [9], since it considers the possible worlds as generated by the possible choices of the truth values for the probabilistic facts and define a probability over these. The probability of a query $q$, $P^C(q)$, is specified via a probability range $[\underline{\mathrm{P}}(q), \overline{\mathrm{P}}(q)]$ where

$$\overline{\mathrm{P}}(q) = \sum_{\tau_i|\exists m\in AS(\tau_i),\ m\models q} P(\tau_i),\ \underline{\mathrm{P}}(q) = \sum_{\tau_i|\forall m\in AS(\tau_i),\ m\models q} P(\tau_i). \tag{3}$$

One crucial point is that the Credal Semantics requires that every world has at *least one* stable model, i.e., it is satisfiable. We call the programs satisfying this requirement *consistent*. This may be strict since it is not satisfied for many of the possible programs that can be written. Moreover, here we assume that probabilistic facts cannot appear as head atoms of any rule, a property called *disjoint condition* [9], since the violation of this may lead to some issues, as discussed in [26].

While the CS can directly represent ProbLog [7] programs, the LP$^{\text{MLN}}$ semantics requires a conversion for the probabilistic facts: for every probabilistic fact $\Pi_i :: f_i$, we need to add $ln(\Pi_i) : f_i$ and $ln(1 - \Pi_i) :\leftarrow f_i$ to the program, where $ln(\Pi_i)$ and $ln(1 - \Pi_i)$ are the weigths associated to the two rules. The other rules of a ProbLog program are considered as hard rules. With this encoding, every well-defined ProbLog program and its LP$^{\text{MLN}}$ counterpart have the same probability distribution over the interpretations [14].

**Example 2.** *Example 1 is converted into:*

```
-0.9162 pressure(carl).
-0.5108 :- pressure(carl).
-1.2039 high_sugar(carl).
-0.3567 :- high_sugar(carl).
faint(X):- pressure(X).
faint(X):- high_sugar(X).
```

*where* $ln(0.4) = -0.9162$*,* $ln(0.6) = -0.5108$*,* $ln(0.3) = -1.2039$*, and* $ln(0.7) = -0.3567$*. There are 4 probabilistic answer sets:* $\{\}$ *with weight 0.42,* $\{pressure(carl)\ faint(carl)\}$ *with weight 0.28,* $\{high\_sugar(carl)\ faint(carl)\}$ *with weight 0.18, and* $\{pressure(carl)\ high\_sugar(carl)\ faint(carl$ *with weight 0.12, which are exactly the worlds of Example 1. The query is true in the last three, so its probability is 0.58. If we consider the CS, the computation of the probability follows the one of Example 1, since every world has exactly one answer set, and the lower and upper probabilities coincide.*

Thus, in the case of a ProbLog program satisfying the disjoint condition, the CS and the LP$^{\text{MLN}}$ semantics (after the conversion) define the same probability distribution, as stated in Theorem 1.

**Theorem 1.** *Consider a ProbLog program* $\mathcal{P}$ *and a query* $q$*. If every world of* $\mathcal{P}$ *has exactly one model and the disjoint property is satisfied, the probability of* $q$ *computed by considering the Credal Semantics (*$P^C(q) = [\underline{\mathrm{P}}(q), \overline{\mathrm{P}}(q)]$*) and the* LP$^{\text{MLN}}$ *semantics (*$P^L(q)$*), after converting the probabilistic facts as previously discussed, coincide, i.e.,* $\underline{\mathrm{P}}(q) = \overline{\mathrm{P}}(q) = P^L(q)$*. Thus, in this case, the two semantics define the same probability distribution.*

**Proof 1.** *The number of answer sets considered for the* LP$^{\text{MLN}}$ *semantics is the same as the number of worlds (and so answer sets) for the CS. For the Credal Semantics, we get a sharp probability value for the query, as with the DS. For the* LP$^{\text{MLN}}$ *semantics, as described in [14] and reported in Example 2, every probabilistic fact is converted into two rules, with the correspondent weights, which ensure the same distribution of a ProbLog program, that also follows the DS.*

By adopting the just explained conversion, Theorem 1 does not hold for a general consistent probabilistic answer set program, where some worlds may have more than one answer set. To see this, consider the following example.

**Example 3.** *Consider the following program:*

```
0.4::bird(1..4).
{fly(X)} :- bird(X).
:- #count{X:fly(X),bird(X)} = FB, #count{X:bird(X)} = B, 10*FB<6*B.
```

*with query $q = fly(1)$. The last constraint imposes that at least 60% of the birds fly [24]. The weights for the* $\mathrm{LP}^{\mathrm{MLN}}$ *semantics are respectively -0.91629 and -0.51082 for the probabilistic facts true and false.* $P^C(q) = [0.2592, 0.4]$ *while* $P^L(q) = 0.45459$. *So, by adopting the* $\mathrm{LP}^{\mathrm{MLN}}$ *with the conversion proposed in [14], we possibly get a surprising result. Here, $fly(1)$ is only influenced by $bird(1)$, whose probability is 0.4, so we may expect to get at most 0.4 as the probability of the query. However, we get 0.4524, which is greater than 0.4. This since for the* $\mathrm{LP}^{\mathrm{MLN}}$ *the probability is distributed over the answer sets, rather than over the worlds.*

So, it is not guaranteed that $P^L(q) \in [\underline{\mathrm{P}}(q), \overline{\mathrm{P}}(q)]$, and the following holds.

**Theorem 2.** *For a probabilistic answer set program satisfying the disjoint property and a query $q$, the conversion of the probabilistic facts proposed in [14], the probability $P^L(q)$ may exceed the upper bound $\overline{\mathrm{P}}(q)$ obtained by considering the Credal Semantics.*

Consider a simplification of Example 3 with only one probabilistic fact $bird(1)$ with probability 0.4, two deterministic facts, $bird(2)$ and $bird(3)$, and query $fly(1)$ (the constraint remains the same). We have 2 worlds, $\tau_1$ and $\tau_2$, 5 answer sets in total, 4 where $bird(1)$ is present ($\tau_1$) and 1 where is not ($\tau_2$). In $\tau_1$, the query is true in only 3 answer sets, while for $\tau_2$ we do not have a contribution to the probability. $P^L(q) = 3 \cdot 0.4/(4 \cdot 0.4 + 1 \cdot (1 - 0.4)) = 0.5454$ while $\overline{\mathrm{P}}(q) = 0.4$. By parameterizing this formula with $n^T$ (the number of answer sets where the query is true in $\tau_1$), $n^F$ (the number of answer sets where the query is false in $\tau_1$), $n$ (the total number of answer sets of $\tau_2$), we can compute the probability value such that $P^L(q) = \overline{\mathrm{P}}(q)$. By solving the equation, we get that when $p = (n^T - n)/(n^T + n^F - n)$, $P^L(q) = \overline{\mathrm{P}}(q)$. This also shows that the $P^L(q)$ may be less than $\overline{\mathrm{P}}(q)$. In this example, if $p = 2/3$, $P^L(q) = \overline{\mathrm{P}}(q)$.

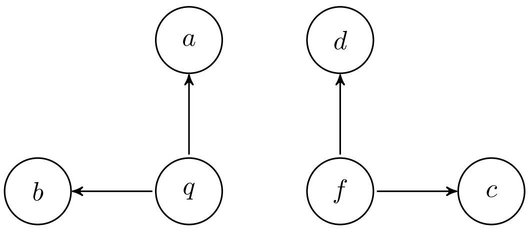
So, in the general case, to obtain the same probability value from both the $\mathrm{LP}^{\mathrm{MLN}}$ and the Credal Semantics, we cannot directly provide a weight to associate to the rules, since this would require knowing the number of answer sets for every world (by considering the CS).

## 4. Inference under the Credal Semantics

One of the major advances for inference in Probabilistic Logic Programming was the conversion of a program into a compact form [7], operation called *knowledge compilation* [27]. In this new form, inference can be performed in a faster way. Clearly, the complexity of the inference task still remains in the #P-complete class [19, 28]. This representation is built starting from the derivation of the query, so probabilistic facts not involved in it can be ignored, since they do not contribute to the probability. Being ASP model driven (even if there are some proposal for a query driven ASP language [29]) probabilistic facts that are not involved in the computation of the probability of a query for a probabilistic answer set program under the Credal Semantics (so every world has at least one answer set) are always considered, but can be ignored. One may argue that the identification of these facts is a task that should be demanded to the programmer.

```
0.2::a. 0.3::b. 0.4::c. 0.5::d.
q ; nq:- a, b.
f:- c, d.
```

(a) Program.



(b) Dependency graph.

**Figure 1:** Program (left) and (part of) its dependency graph (right).

However, this is not always simple, for example when there are multiple clauses that share the same literals. Moreover, in the context of parameter learning, usually we are given a knowledge base of examples and a program, but often only a small part of the overall probabilistic facts is involved in the computation of the probability for a given query. To solve this issue, we can consider the dependency graph of an answer set program $\mathcal{P}$. We can convert the probabilistic answer set program into an answer set program by translating every probabilistic fact $\Pi :: f$ into a choice rule $\{f\}$ [24], ignoring the probability (since it does not influence the generation of the stable models). The nodes of a dependency graph are the ones in the Herbrand base of $\mathcal{P}$, and it contains a directed edge $(A, B)$ if and only if there exists a rule $r$ in the grounding of $\mathcal{P}$ such that such that $A$ is in the head and $B$ is in the body. Here, we do not need to consider the label (positive or negative) of the edges. $A$ depends on $B$ if there exists a path from $A$ to $B$ in the graph built as just described. Computing the dependency graph from a probabilistic answer set program is straightforward: after the conversion into an ASP version, we can obtain the graph by, for example, computing its *aspif* format [30], converting the rules into terms that represent connections with edges, and then finding the subgraphs that are not connected to the node representing the query. This operation takes negligible time with respect to the probability computation. If we want to compute the probability of a query $q$, we just need to consider the probabilistic facts in the subgraph where the query is present. All the other probabilistic facts can be ignored. For example, the probabilistic answer set program shown in Figure 1a has the dependency graph shown in Figure 1b (we kept only some edges for clarity). There are two subgraphs, one with the nodes $a$, $b$, and $q$, and one with $c$, $d$, and $f$. These two are not connected, so they not influence each other. Thus, for the computation of the probability of the query $q$ we can consider only $a$ and $b$.

The generation of the dependency graph and the pruning of unused clauses and facts is possible only if the program is consistent. To see this, consider the following program: `0.2::a.` `q:- a. c:- not c.`. The dependency graph has two subgraphs: one with $a$ and $q$ connected and one with a single node, $c$. If we are interested in the probability of the query $q$, we remove the clause `c:- not c.`. However, if we remove that clause, we get a program that is different to the original one, since the new program has a Credal Semantics while the original one does not have it (all the worlds are unsatisfiable). The answer set version (probabilistic fact replaced with a choice rule) of the previous program is unsatisfiable, so one may argue that it is devoid of meaning. This may be true, but it illustrates one of the possible issues that may arise by considering the dependency graph. Interesting future works could consist in formalizing this approach, precisely identifying the class of programs where this may work, and adopting knowledge compilation techniques to speed up inference, as in [18].

## 5. Handling Inconsistent Worlds

If a world $\tau$ has no answer sets, $P(\tau)$ neither contributes to the probability of the query $q$ nor to the probability of the negation of the query. If this happens, $\underline{\mathrm{P}}(q) + \overline{\mathrm{P}}(not\ q) < 1$, and there would be a loss of probability mass. If so, the correspondent program fails to have a Credal Semantics. In the context of probabilistic answer set programs, meaningful answer set programs, such as the one of Figure 2b, when integrated with probabilistic facts and considered under the Credal Semantics, may fail to have at least one stable model for every world. However, we would like to still compute the probability of a query in these. Clearly, we need to consider in some way the probability lost. To handle this, in the context of Description Logics, the authors of [31] proposed two possible solutions, that we discuss here for a possible adoption for probabilistic answer set programs under the Credal Semantics: try to repair the program or try to normalize the probability. For the first proposal [32, 33], the authors of [32] introduced the *generalized repair semantics* that allows to split the database into a hard and a soft part. The hard part is fixed while the soft part can be modified to try to repair the inconsistency. However, it can be difficult to precisely identify these rules. Consider again the program `0.2::a. q:- a. c:- not c.`. If we convert the probabilistic fact into a choice rule, the program has no answer sets. The inconsistency may be eventually identified using solutions such as the one presented in [34]. The rule that causes the inconsistency is clearly the last one. Here, all the worlds are unsatisfiable, since its answer set version is unsatisfiable. However, when only some of the worlds are unsatisfiable, it is not easy to remove a particular rule to make them satisfiable, without affecting the probability of the worlds.

Thus, we focus on the second solution, which consists in reintegrating back (by normalization) the probability lost, as also adopted in cProbLog [35], an extension of ProbLog with constraints.

**Example 4.** *Consider a graph coloring scenario, a standard answer set program, extended with some probabilities on the edges connecting the nodes, as shown in Figure 2. There are $2^3 = 8$ possible worlds. The last constraint prevents $edge(a, b)$ and $edge(b, c)$ to be true at the same time, so it removes 2 worlds: $\tau_1 = \{edge(a, b)\ edge(b, c)\}$ and $\tau_2 = \{edge(a, b)\ edge(b, c)\ edge(d, c)\}$. However, it seems natural to try to assign a probability to a query such as $green(c)$ by, for example, normalization, instead of trying to find a repair for the program (note again that here it is not clear how to identify the parts that should be repaired).*

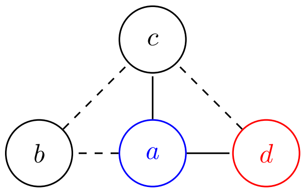
We can assign a probability to programs such as the one described in Figure 2a by dividing both the probability bounds of Equation 3 by a normalization constant $Z$ where

$$Z = 1 - \sum_{\tau_i \mid |AS(\tau_i)|=0} P(\tau_i) = \sum_{\tau_i \mid |AS(\tau_i)|>0} P(\tau_i) \tag{4}$$

That is, $Z$ is 1 minus the probability of the worlds without answer sets. Equivalently, $Z$ is the sum of the probabilities of the worlds with at least one answer set. We ignore the corner case

```
node(a). node(b). node(c). node(d).
red(X) ; green(X) ; blue(X) :- node(X).
e(X,Y) :- edge(X,Y). e(Y,X) :- edge(Y,X).
blue(a). red(d).

:- e(X,Y), red(X), red(Y).
:- e(X,Y), green(X), green(Y).
:- e(X,Y), blue(X), blue(Y).

edge(a,d). edge(a,c).

0.4::edge(a,b). 0.4::edge(b,c). 0.4::edge(c,d).

:- edge(a,b), edge(b,c).
```

(a) Program.



(b) Graph representation.

**Figure 2:** Probabilistic answer set program encoding the graph coloring problem.

where all the worlds are inconsistent. Thus, the formulas for the upper and lower probabilities became

$$\overline{\mathrm{P}}_n(q) = \overline{\mathrm{P}}(q)/Z,\ \underline{\mathrm{P}}_n(q) = \underline{\mathrm{P}}(q)/Z. \tag{5}$$

With the normalization constant, we now have that $\underline{\mathrm{P}}_n(q) + \overline{\mathrm{P}}_n(\neg q) = 1$. The probability of the query $q =$ green(c) can be now computed as: $Z = 1 - (P(\tau_1) + P(\tau_2)) = 0.84$, $\overline{\mathrm{P}}_n(q) = 1$, $\underline{\mathrm{P}}_n(q) = 0.4$. If the query $q$ is a conjunction of atoms that cannot appear together in an answer set, such as $q = edge(a,b), edge(b,c)$ in the previous program (Figure 2a), we get 0 as probability, since none of the worlds $w_i$ with $|AS(\tau_i)| > 0$ contain these two atoms.

There are two other semantics that manage worlds without answer sets: the L-Credal Semantics [17] and the SMProbLog semantics [18]. The former, considers a variant of the commonly adopted stable model semantics, namely the least undefined stable model semantics: a least stable model (L-stable model, for short) is a partial stable model where the number of undefined atoms is the lowest possible. In this way, the stable models of a program coincide with the L-stable models but when a program does not have stable models, it may have some L-stable models which contain undefined atoms. Thus, with this semantics, we still have a range as probability for a query, but we need to consider three possible values for an atom: true, false, and undefined. A three-valued semantics is also adopted in SMProbLog [18]. Within this framework, differently from the L-Credal Semantics, the probability of a query is a sharp probability value and, to compute it, also the number of stable models for every world are considered. The probability of the inconsistent worlds is associated with the value inconsistent, while the probability of the consistent worlds contributes to the probability of the query. Differently from these two semantics, the normalization approach does not require a three-valued semantics. Among the three, SMProbLog and the normalization approach have a practical implementation[2] while

[2]SMProbLog can be found at https://github.com/PietroTotis/smProblog while an approach based on normalization at https://github.com/damianoazzolini/pasta

the L-Credal Semantics seems to not have one. An interesting future work could be further analyzing the relationship between these two semantics and the normalization approach, with a particular focus also on the computation of the conditional probability. Moreover, consider this program: `0.4::a. :- not a.`. If we ask for the probability of $a$, by using normalization we obtain 1 (the world where $a$ is true has probability 0.4, and the world where it is false is unsatisfiable, so the normalizing factor $Z$ is 0.4), which is different from the probability associated to it.

Efficiently identifying the worlds without answer sets, so programs that admit or not the Credal Semantics, is a complicated task, since it may involve the generation of all the worlds and the test of the satisfiability for each one. This is not scalable, since the worlds are exponential in the number of probabilistic facts. There are other possible approaches to explore. We can identify the possible MUSes [34], i.e., minimal (in terms of cardinality) subsets of variables (probabilistic facts) which makes the ASP program unsatisfiable. However, this should be applied on all the possible worlds and thus still intractable. Moreover, this supposes that the set of atoms and the initial program, considered together, should be unsatisfiable. An alternative solution consists in iteratively sampling a world and check whether it is satisfiable or not. In this way, we may identify whether a program admits or not a Credal Semantics, but not necessarily identify all the worlds that are unsatisfiable. Clearly, the effectiveness of this approach depends both on the number of samples and on the ratio between the number of unsatisfiable and possible worlds. Also the development of effective algorithms to spot inconsistent worlds can be a subject of a future work.

## 6. Related Work

There are several approaches that manage uncertainty with ASP. In the previous sections we discussed some frameworks to represent uncertainty in Answer Set Programming: P-log [13] and LP$^{\mathrm{MLN}}$ [14], that define a probability distribution on stable models, and SMProbLog [18], the Credal Semantics [16] and the L-Credal Semantics [17] that are based on the definition of world of the Distribution Semantics [9]. SMProbLog and the L-Credal Semantics have been discussed in the context of argumentation and can also manage worlds with no answer sets. The authors of cProbLog [35] proposed a formulation and a tool to perform inference in ProbLog programs where some worlds have no models, due to the presence of constraints.

Diff-SAT [36] and PrASP [37] are two other proposals to handle uncertain rules. Differently from the Credal Semantics, PrASP defines a probability distribution over answer sets and it is not based on the Distribution Semantics. A PrASP program is a set of annotated formulas and independence constraints (since it does not impose independence on the random variables). Each formula can be associated with a pair of values $[l, u]$ representing its (imprecise) probability ($l$ and $u$ can coincide). To perform inference, a PrASP program is converted into an ASP program called *spanning program*. Then, the goal is to find a solution to a set of equations that define a parameterized probability distribution over answer sets. Both under PrASP and CS the probability of the query is specified by a range, but they differ in how this range is computed. Similar considerations hold for DiffSAT, where inference is considered as a multi model optimization problem: in case of probabilistic inference, the solution of such a problem approximates the probability distribution over the possible models. Also here, the probability distribution is over answer sets, while the CS considers a probability distribution over probabilistic facts.

PASOCS [38] and PASTA [24] are two frameworks to perform inference, both exact and approximate [23], in probabilistic answer set programs under the Credal Semantics, and thus they require that every world has at least one stable model. They have a similar approach for the computation of the exact probability of a query: the first enumerates all the worlds and then, for each one, computes the answer sets with an answer set solver. PASTA, on the contrary, translates the probabilistic answer set program into an answer set program by converting probabilistic facts into choice rules and then calls an answer set solver only once to compute the projective solutions. Empirically [24], PASTA is slightly faster, but both solvers cannot manage more than 30-32 probabilistic facts.

The authors of [39] introduced the plausibility reasoning task, which sits between cautious reasoning and brave reasoning, by asking whether a query is present in at least $k\%$ of the answer set of a program. This may be an interesting approach to extend to the Credal Semantics, to relax, for example, the lower bound. Let us discuss this for a moment. For example, we may propose the *kp-Credal Semantics* and the *kn-Credal Semantics*, where, instead of computing the cautious consequences for the lower bound, we consider the worlds where the query is true in at least k% (for the kp-Credal Semantics) of the answer sets ($\underline{\mathrm{P}}^{k}_{kp}(q)$), or in n (for the kn-Credal Semantics) answer sets ($\underline{\mathrm{P}}^{n}_{kp}(q)$). However, for the kp-Credal Semantics as proposed, there may be world that contribute to both the bounds, so $\underline{\mathrm{P}}^{k}_{kp}(q) + \overline{\mathrm{P}}(not\ q)$ is not guaranteed to sum to 1 (and may exceed 1). Similarly for the kn-Credal Semantics. If instead of a probability range, we define a sharp probability value for the two semantics, this still does not guarantee that $P(q) + P(not\ q) = 1$. A more in-depth study of this may be the subject of a future work.

## 7. Conclusions

In this paper, we conducted a brief discussion about the Credal Semantics and its main features. First, we sketched the differences between it and the $\mathrm{LP}^{\mathrm{MLN}}$ semantics. Then, we considered the problem of inference and argued that, in some cases, the generation of the dependency graph may help identifying rules and facts not involved in the probability computation. Finally, we discuss some possible approaches to handle worlds where some programs have no answer sets.

## References

[1] J. W. Lloyd, Foundations of Logic Programming, 2nd Edition, Springer, 1987.

[2] L. Sterling, E. Shapiro, The Art of Prolog: Advanced Programming Techniques, Logic programming, MIT Press, 1994.

[3] G. Brewka, T. Eiter, M. Truszczyński, Answer set programming at a glance, Communications of the ACM 54 (2011) 92–103. doi:10.1145/2043174.2043195.

[4] W. Faber, G. Pfeifer, N. Leone, Semantics and complexity of recursive aggregates

in answer set programming, Artificial Intelligence 175 (2011) 278–298. doi:10.1016/j.artint.2010.04.002.

[5] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82. doi:10.1017/S1471068418000054.

[6] M. Alviano, C. Dodaro, N. Leone, F. Ricca, Advances in WASP, in: F. Calimeri, G. Ianni, M. Truszczynski (Eds.), Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings, volume 9345 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 40–54. doi:10.1007/978-3-319-23264-5_5.

[7] L. De Raedt, A. Kimmig, H. Toivonen, ProbLog: A probabilistic Prolog and its application in link discovery, in: M. M. Veloso (Ed.), 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), volume 7, AAAI Press, 2007, pp. 2462–2467.

[8] S. Muggleton, et al., Stochastic logic programs, Advances in inductive logic programming 32 (1996) 254–264.

[9] T. Sato, A statistical learning method for logic programs with distribution semantics, in: L. Sterling (Ed.), Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995, MIT Press, 1995, pp. 715–729. doi:10.7551/mitpress/4298.003.0069.

[10] D. Azzolini, F. Riguzzi, E. Lamma, A semantics for hybrid probabilistic logic programs with function symbols, Artificial Intelligence 294 (2021) 103452. doi:10.1016/j.artint.2021.103452.

[11] F. Riguzzi, MCINTYRE: A Monte Carlo system for probabilistic logic programming, Fundamenta Informaticae 124 (2013) 521–541. doi:10.3233/FI-2013-847.

[12] F. Riguzzi, T. Swift, The PITA system: Tabling and answer subsumption for reasoning under uncertainty, Theory and Practice of Logic Programming 11 (2011) 433–449.

[13] C. Baral, M. Gelfond, N. Rushton, Probabilistic reasoning with answer sets, Theory and Practice of Logic Programming 9 (2009) 57–144. doi:10.1017/S1471068408003645.

[14] J. Lee, Y. Wang, A probabilistic extension of the stable model semantics, in: AAAI Spring Symposia, 2015.

[15] F. G. Cozman, D. D. Mauá, The structure and complexity of credal semantics, in: A. Hommersom, S. A. Abdallah (Eds.), 3rd International Workshop on Probabilistic Logic Programming (PLP 2016), volume 1661 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 3–14.

[16] F. G. Cozman, D. D. Mauá, On the semantics and complexity of probabilistic logic programs, Journal of Artificial Intelligence Research 60 (2017) 221–262. doi:10.1613/jair.5482.

[17] V. H. N. Rocha, F. G. Cozman, A credal least undefined stable semantics for probabilistic logic programs and probabilistic argumentation, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, 2022.

[18] P. Totis, L. De Raedt, A. Kimmig, smProbLog: Stable model semantics in problog for probabilistic argumentation, Cambridge University Press, 2023, pp. 1–50. doi:10.1017/S147106842300008X.

[19] F. Riguzzi, Foundations of Probabilistic Logic Programming: Languages, semantics, infer-

ence and learning, River Publishers, Gistrup, Denmark, 2018.
[20] J. Lee, Y. Wang, Weighted rules under the stable model semantics, in: C. Baral, J. P. Delgrande, F. Wolter (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016, AAAI Press, 2016, pp. 145–154.
[21] M. Richardson, P. Domingos, Markov logic networks, Machine Learning 62 (2006) 107–136.
[22] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming., volume 88, MIT Press, 1988, pp. 1070–1080.
[23] D. Azzolini, E. Bellodi, F. Riguzzi, Approximate inference in probabilistic answer set programming for statistical probabilities, in: A. Dovier, A. Montanari, A. Orlandini (Eds.), AIxIA 2022 – Advances in Artificial Intelligence, Springer International Publishing, Cham, 2023, pp. 33–46. doi:10.1007/978-3-031-27181-6_3.
[24] D. Azzolini, E. Bellodi, F. Riguzzi, Statistical statements in probabilistic logic programming, in: G. Gottlob, D. Inclezan, M. Maratea (Eds.), Logic Programming and Nonmonotonic Reasoning, Springer International Publishing, Cham, 2022, pp. 43–55. doi:10.1007/978-3-031-15707-3_4.
[25] D. Azzolini, E. Bellodi, F. Riguzzi, MAP inference in probabilistic answer set programs, in: A. Dovier, A. Montanari, A. Orlandini (Eds.), AIxIA 2022 – Advances in Artificial Intelligence, Springer International Publishing, Cham, 2023, pp. 413–426. doi:10.1007/978-3-031-27181-6_29.
[26] F. G. Cozman, D. D. Mauá, The joy of probabilistic answer set programming: Semantics, complexity, expressivity, inference, International Journal of Approximate Reasoning 125 (2020) 218–239. doi:10.1016/j.ijar.2020.07.004.
[27] A. Darwiche, P. Marquis, A knowledge compilation map, Journal of Artificial Intelligence Research 17 (2002) 229–264. doi:10.1613/jair.989.
[28] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, Adaptive computation and machine learning, MIT Press, Cambridge, MA, 2009.
[29] J. Arias, M. Carro, E. Salazar, K. Marple, G. Gupta, Constraint answer set programming without grounding, Theory and Practice of Logic Programming 18 (2018) 337–354. doi:10.1017/S1471068418000285.
[30] R. Kaminski, T. Schaub, P. Wanko, A Tutorial on Hybrid Answer Set Solving with clingo, 2017, pp. 167–203. doi:10.1007/978-3-319-61033-7_6.
[31] İ. İ. Ceylan, T. Lukasiewicz, R. Peñaloza, Complexity results for probabilistic datalog$^{\pm}$, in: G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, F. van Harmelen (Eds.), 22nd European Conference on Artificial Intelligence (ECAI 2016), volume 285 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2016, pp. 1414–1422. doi:10.3233/978-1-61499-672-9-1414.
[32] T. Eiter, T. Lukasiewicz, L. Predoiu, Generalized consistent query answering under existential rules, in: Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning, KR'16, AAAI Press, 2016, pp. 359–368. doi:10.5555/3032027.3032070.
[33] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D. F. Savo, Inconsistency-tolerant semantics for description logics, in: P. Hitzler, T. Lukasiewicz (Eds.), Web Reasoning and Rule Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 103–117. doi:10.1007/978-3-

642-15918-3_9.

[34] M. Alviano, C. Dodaro, S. Fiorentino, A. Previti, F. Ricca, Enumeration of minimal models and MUSes in WASP, in: G. Gottlob, D. Inclezan, M. Maratea (Eds.), Logic Programming and Nonmonotonic Reasoning, Springer International Publishing, Cham, 2022, pp. 29–42.

[35] D. Fierens, G. V. den Broeck, M. Bruynooghe, L. D. Raedt, Constraints for probabilistic logic programming, in: D. Roy, V. Mansinghka, N. Goodman (Eds.), Proceedings of the NIPS Probabilistic Programming Workshop, 2012.

[36] M. Nickles, Differentiable SAT/ASP., in: PLP@ ILP, 2018, pp. 62–74.

[37] M. Nickles, A tool for probabilistic reasoning based on logic programming and first-order theories under stable model semantics, in: L. Michael, A. Kakas (Eds.), Logics in Artificial Intelligence, Springer International Publishing, Cham, 2016, pp. 369–384. doi:doi.org/10.1007/978-3-319-48758-8_24.

[38] D. Tuckey, A. Russo, K. Broda, PASOCS: A parallel approximate solver for probabilistic logic programs under the credal semantics, arXiv abs/2105.10908 (2021). doi:10.48550/ARXIV.2105.10908.

[39] J. K. Fichte, M. Hecher, M. A. Nadeem, Plausibility reasoning via projected answer set counting - A hybrid approach, in: L. D. Raedt (Ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, ijcai, 2022, pp. 2620–2626. doi:10.24963/ijcai.2022/363.