ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA250972

An Algebraic View of MAP Inference in Probabilistic Answer Set Programs

Damiano Azzolini[a,*], Giuseppe Mazzotta[b], Francesco Ricca[b] and Fabrizio Riguzzi[a]

[a]University of Ferrara, Ferrara, Italy
[b]University of Calabria, Rende, Italy

Abstract. Maximum-a-Posteriori (MAP) inference is a crucial problem in Artificial Intelligence, which requires both marginalization and maximization, and asks for the most probable value for a given set of variables such that an evidence holds. Several languages within the Statistical Relational Artificial Intelligence landscape support the encoding of MAP. Here, we focus on Probabilistic Answer Set Programming, consider the credal and smProbLog semantics, and introduce a three-level algebraic model counting representation for MAP. We implemented our approach on top of a state-of-the-art solver and compared it with existing solutions, showing the competitive performance of our proposal, even against less general tools.

1 Introduction

Maximum-a-Posteriori (MAP) is a crucial problem in Statistical Relational Artificial Intelligence [10, 29, 32, 36, 40, 41], adopted in problems involving robotics [39], cosmology [15], reinforcement learning [1], and medicine [42], to name a few. In the context of Probabilistic Logic Programming (PLP) [44], the MAP problem can be formulated as follows [10]: given a probabilistic logical theory, where some facts are designated as probabilistic facts and others as MAP facts, the goal is to identify the optimal subset of the latter that maximizes the probability of a given query. This task is more challenging than standard inference as it involves both marginalization and optimization. Specifically, it requires computing the probability of the query for each possible subset of MAP facts, although explicit enumeration can be avoided in practice. Moreover, the MAP task has different complexity and outcomes depending on the considered framework and semantics.

Among the different semantics for probabilistic logic, the Distribution Semantics (DS) [46] is one of the most adopted, thanks to its intuitiveness and the availability of well-maintained practical tools [3, 20]. Here, a logic program is extended with facts that represent uncertain information, often called probabilistic facts [20]. However, the DS requires that every world, i.e., every logic program obtained by fixing the truth values of the probabilistic facts, have a two-valued Well-founded model [27]. This excludes, for example, answer set programs, since they are usually non-stratified and have multiple (stable) models [28]. To overcome this, the credal semantics (CS) was recently proposed [17], which generalizes the DS to worlds with multiple models. A program within this formalism is often called a probabilistic answer set program (PASP). In PASP, the query is no longer associated with a sharp probability value but with a probability range whose lower and upper bounds depend on whether the query is present in *all* or *some* answer sets of the considered worlds. This happens because it refrains from taking a specific probability distribution over the answer sets of a world. If, instead, a uniform distribution over the answer sets of a world is assumed, the resulting semantics corresponds to smProbLog [47].

In this paper, we cast MAP inference in PASP under both the credal and smProbLog semantics as an Algebraic Model Counting (AMC) [31] task. This approach aligns with a well-established line of research in Statistical Relational Artificial Intelligence [43], where various inference tasks are encoded within the AMC framework, which is a generalization of Weighted Model Counting. In particular, some tasks, such as MAP inference in PLP [10], inference under smProbLog [30], and credal semantics [5], require two nested levels of AMC [30]. In contrast, the task addressed in this paper demands *three* nested levels of AMC, referred to as 3AMC, similar to the setting encountered in solving decision-theoretic problems in PASP [8]. In addition, we implemented the proposed approach and conducted experiments on a wide range of instances, demonstrating its effectiveness compared to an existing solution for MAP inference in PASP under the credal semantics. Finally, we compare our tool with specialized solvers that are applicable only when the underlying logic program is stratified, a case where each world admits a single model, thus addressing a more restricted setting than ours. Despite its broader applicability, our approach remains competitive also in this setting. The paper is structured as follows: Section 2 introduces the needed background, Section 3 proposes a 3AMC encoding for MAP, Section 4 analyzes the experimental evaluation, Section 5 positions our paper within the existing literature, and Section 6 concludes.

2 Background

Answer Set Programming. We consider answer set programs (ASP) [13] composed of a set of rules of the form $h_1; \ldots; h_m :- b_1, \ldots, b_n$ where each h_i is an atom and each b_j is a literal (i.e., an atom a or its negation $not\ a$, also called positive and negative literal, respectively). The disjunction of the atoms h_i is called *head* while the conjunction of the literals b_j is called *body*. A rule with a single atom in the head is called *normal* and, if it also does not have body literals, it is called *fact*. A normal program is a program that contains only normal rules. An atom/literal/rule/program with no variables is called *ground*.

The semantics of an ASP is based on the concept of *stable*

* Corresponding Author. Email: damiano.azzolini@unife.it.

model [28]. The Herbrand base of a program $\mathcal{P}$ ($\mathcal{B}_\mathcal{P}$) is the set of ground atoms that can be built from predicates appearing in $\mathcal{P}$ and constants in the program. The *reduct* of a ground program $\mathcal{P}$ w.r.t. an *interpretation* I (i.e., a set of ground atoms $I \subseteq \mathcal{B}_\mathcal{P}$) is the set rules in $\mathcal{P}$ with the body true in I. An interpretation I is called *answer set* (or *stable model*) if it is a minimal model under set inclusion of the reduct of $\mathcal{P}$ w.r.t. I. We denote by $AS(\mathcal{P})$ the set of answer sets of $\mathcal{P}$ and with $|AS(\mathcal{P})|$ its cardinality. The *cautious* (resp. *brave*) consequence for a program $\mathcal{P}$ is the intersection (resp. union) of the answer sets of $\mathcal{P}$. We denote them by $CC(\mathcal{P})$ and $BC(\mathcal{P})$, respectively. The dependency graph of a ground program $\mathcal{P}$, $G_\mathcal{P}$, is a labeled directed graph whose nodes are atoms in $\mathcal{B}_\mathcal{P}$ containing a positive (resp. negative) edge from an atom b to an atom h if there exists a rule r with head h such that b belongs to the set of positive (resp. negative) literals in the body. A normal program $\mathcal{P}$ is said to be *stratified* if $G_\mathcal{P}$ does not contain loops involving negative edges. In this case, such a program has a unique stable model which also coincides with the Well-founded model [27]. A program $\mathcal{P}$ is head-cycle-free (HCF) [11] if it does not contain any rule r having two atoms in the head that are involved in a positive loop (i.e., loop made of positive edges only) in $G_\mathcal{P}$. HCF programs can be transformed, in polynomial time, into normal programs by *shifting* disjunctions [21].

Example 1. *The following program $\mathcal{P}$ has two facts and two rules.*

```
a. b.
p:- a.
q ; nq :- b.
```

It has two answer sets: $AS(\mathcal{P}) = \{\{a, p, b, nq\}, \{a, p, b, q\}\}$. The cautious consequences are $CC(\mathcal{P}) = \{a, p, b\}$ while the brave consequences are $BC(\mathcal{P}) = \{a, p, b, q, nq\}$.

Probabilistic Answer Set Programming. A probabilistic answer set program (PASP) is a tuple $(\mathcal{P}, \mathcal{F})$ where $\mathcal{P}$ is an answer set program and $\mathcal{F}$ is a set of ground *probabilistic facts* of the form $\pi_i :: f_i$ with f_i atom and $\pi_i \in [0, 1]$. A *world* is built by deciding, for each probabilistic fact, whether to include it or not in $\mathcal{P}$, thus there are $2^{|\mathcal{F}|}$ of them. Probabilistic facts are considered independent, and the probability of a world w is computed as

$$P(w) = \prod_{f_i \in w} \pi_i \cdot \prod_{f_i \notin w} (1 - \pi_i). \tag{1}$$

The credal semantics (CS) [17] assigns a meaning to PASP by associating a probability range to *queries* (i.e., conjunctions of ground literals) described by a lower and an upper bound. For a query q, the *lower probability* $\underline{\mathrm{P}}(q)$ is the sum of the probability of the worlds where the query is present in *every* answer set (i.e., it is a cautious consequence). Conversely, the *upper probability* $\overline{\mathrm{P}}(q)$ is the sum of the probability of the worlds where the query is present in *at least one* answer set (i.e., it is a brave consequence). In formulas,

$$\underline{\mathrm{P}}(q) = \sum_{w_i | CC(w_i) \models q} P(w_i),\ \overline{\mathrm{P}}(q) = \sum_{w_i | BC(w_i) \models q} P(w_i). \tag{2}$$

The conditional probability of q given a conjunction of ground atoms e (called evidence) is also defined by a lower and an upper bound: $\underline{\mathrm{P}}(q \mid e) = \frac{\underline{\mathrm{P}}(q,e)}{\underline{\mathrm{P}}(q,e)+\overline{\mathrm{P}}(not\ q,e)}$ $\overline{\mathrm{P}}(q \mid e) = \frac{\overline{\mathrm{P}}(q,e)}{\overline{\mathrm{P}}(q,e)+\underline{\mathrm{P}}(not\ q,e)}$. The CS also requires that all worlds have at least one answer set.

At a high level, the idea behind the probability range is the following: once a world is fixed, we obtain an ASP. This ASP has one or more answer sets. If it has multiple answer sets, there is uncertainty on how to distribute the probability mass of the world on these. With the CS, we do not make any choice and instead consider a "brave" scenario (upper probability) and a "cautious" scenario (lower probability). If *all* the answer sets contain the query, no matter how we distribute the probability mass, all counts to $P(q)$, so we have a contribution to the lower and upper probability. If only *some* (and not all) of the answer sets entail the query, not all possible mass placements will count to $P(q)$, so we do not have a contribution to the lower probability, but rather only to the upper probability.

Example 2. *Consider the following simple propositional program.*

```
0.23::a. 0.48::b. 0.86::c. 0.74::d.
q:- a, b.
q ; nq :- c.
q ; nq :- d.
```

Suppose we are interested in computing the probability of query q. The program has $2^4 = 16$ worlds, listed in Table 1. For example, the world including only a and b has a single answer set, $\{q, a, b\}$, so it contributes to the lower and upper probability, whereas the world that only includes c has two answer sets, $\{q, c\}$ and $\{nq, c\}$, one of the two does not entail the query, so this world contributes to the upper probability only. By repeating this process for every world, we obtain $\underline{\mathrm{P}}(q) = 0.1104$ and $\overline{\mathrm{P}}(q) = 0.9676$. If we observe that b and c are true (evidence), we get $\underline{\mathrm{P}}(q \mid b, c) = 0.23$ and $\overline{\mathrm{P}}(q \mid b, c) = 1$.

Another semantics for PASP, which we consider in this paper, is the smProbLog semantics [47]. The setting is similar to the one of the credal semantics, i.e., an answer set program extended with probabilistic facts. However, instead of considering lower and upper probability bounds for a query, so checking whether the query holds bravely or cautiously in a given world, it distributes the probability of the query uniformly among the different answer sets for a world. More formally, given a world w, $\forall A \in AS(w), P_w(A) = P(w)/|AS(w)|$. Then, the probability of a query q is the sum of the probabilities of the answer sets where q holds, i.e.,

$$P_{sm}(q) = \sum_{w_i} \sum_{A \in AS(w_i) | q \in A} P_{w_i}(A).$$

The conditional probability of a query q given evidence e can be computed with the usual formula $P_{sm}(q \mid e) = P_{sm}(q, e)/P_{sm}(e)$. Since some world may have no answer sets and the above ratio may be undefined, the authors of [47] introduced a third truth value, "inconsistent". However, for uniformity with the credal semantics, here we assume that every world has at least one model.

Example 3. *Consider the same program of Example 2, this time interpreted under the smProbLog semantics. Here, for example, the world w where only c is present has two answer sets, $A_0 = \{q, c\}$ and $A_1 = \{nq, c\}$, each with a probability of $P(w)/|AS(W)| = (1 - 0.23) \cdot (1 - 0.48) \cdot (0.86) \cdot (1 - 0.74)/2 \approx 0.044$. If we repeat this process for all worlds and sum the probabilities of the answer sets where the query q holds, we obtain $P_{sm}(q) \approx 0.539$. If we consider the conditional probability $P_{sm}(q \mid b, c)$, we have $P(q, b, c) = 0.253872$, $P(b, c) = 0.4128$, so $P_{sm}(q \mid b, c) = 0.615$.*

Maximum a Posteriori Inference in PASP. Let us introduce the definition of MAP in PASP under the credal semantics, by following [7], and then extend it to the smProbLog semantics.

Definition 1. *Given a PASP $(\mathcal{P}, \mathcal{F})$, a set $Q \subseteq \mathcal{F}$ (the elements of Q are called query probabilistic facts), and a ground atom e called evidence,*

- *the cautious MAP problem consists in finding a truth assignment q to query probabilistic facts in Q such that $\underline{\mathrm{P}}(Q = q \mid e)$ is maximized, i.e., solving:*

$$\begin{aligned}\underline{\mathrm{MAP}}(Q, e) &= \arg\max_q \underline{\mathrm{P}}(Q = q \mid e) \\ &= \arg\max_q \sum_{w_i \mid q, e \in CC(w_i)} P(w_i);\end{aligned}$$

- *the brave MAP problem consists in finding a truth assignment q to query probabilistic facts in Q such that $\overline{\mathrm{P}}(Q = q \mid e)$ is maximized, i.e., solving:*

$$\begin{aligned}\overline{\mathrm{MAP}}(Q, e) &= \arg\max_q \overline{\mathrm{P}}(Q = q \mid e) \\ &= \arg\max_q \sum_{w_i \mid q, e \in BC(w_i)} P(w_i).\end{aligned}$$

- *the MAP problem under the smProbLog semantics consists in finding a truth assignment q to query probabilistic facts in Q such that $P_{sm}(Q = q \mid e)$ is maximized, i.e., solving:*

$$\begin{aligned}\mathrm{MAP}_{\mathrm{sm}}(Q, e) &= \arg\max_q P_{sm}(Q = q \mid e) \\ &= \arg\max_q \sum_{w_i} \sum_{A \in AS(w_i) \mid q, e \in A} P_{w_i}(A).\end{aligned}$$

The results of $\underline{\mathrm{MAP}}$ *and* $\overline{\mathrm{MAP}}$ *are respectively called* lower *and* upper MAP states.

Example 4. *Consider again Example 2, this time with b and d considered as query probabilistic facts (denoted by prepending the functor map). We have the following program.*

```
map 0.48::b. map 0.74::d.
0.23::a. 0.86::c.
q:- a, b.
q ; nq :- c.
q ; nq :- d.
```

Worlds are listed in Table 1. Suppose that we are interested in the lower MAP state for q. Here, four worlds contribute to the lower probability: w_{10} and w_{11} with state $l_0 = \{b\}$ and w_{14} and w_{15} with state $l_1 = \{b, d\}$. The probability associated with l_0 is $P(w_{10}) + P(w_{11}) = 0.00401856 + 0.02468544 = 0.028704$, while the probability associated with l_1 is $P(w_{14}) + P(w_{15}) = 0.01143744 + 0.07025856 = 0.0816956$. State l_1 has the largest probability, so it is the lower MAP state. Consider now the upper MAP state. Here, all combinations of query probabilistic facts result in a state with probability > 0. The probability associated with $u_o = \{\}$ is $P(w_1) + P(w_3) = 0.116272$, with $u_1 = \{d\}$ is $\sum_{i=4}^{7} P(w_i) = 0.3848$, with $u_2 = \{b\}$ is $\sum_{i=9}^{11} P(w_i) = 0.11134656$, and with $u_3 = \{b, d\}$ is $\sum_{i=12}^{15} P(w_i) = 0.3552$. u_1 is associated with the largest probability, so it is the upper MAP state.

If instead we consider the smProbLog semantics, the probability associated with the states $u_0, \ldots, u_3$ is $P(w_1)/2 + P(w_3)/2 = 0.058136$, $\sum_{i=4}^{7} P(w_i)/2 = 0.1924$, $\sum_{i=9}^{11} P(w_i)/2 = 0.07002528$, and $\sum_{i=12}^{15} P(w_i)/2 = 0.218448$, respectively. Thus, under this semantics, the MAP state is u_3. This also shows that $\underline{\mathrm{MAP}}$, $\overline{\mathrm{MAP}}$, *and* $\mathrm{MAP}_{\mathrm{sm}}$ *may not coincide.*

The MAP task under the credal semantics has been proved hard. The authors of [16, 36] considered the decision problem of asking whether $\underline{\mathrm{P}}(Q \mid e) > \gamma$ and showed that for propositional programs it is $\mathrm{NP}^{\mathrm{PP}}$-complete, regardless the considered fragment. To the best of our knowledge, there are no theoretical studies on the complexity of such a task under the smProbLog semantics. However, the next section implicitly provides a membership result in $\mathrm{NP}^{\mathrm{PP}}$.

Table 1. Worlds for Example 4.

id	b	d	a	c	AS	Probability
w_0	0	0	0	0	$\{\{\}\}$	0.01457456
w_1	0	0	0	1	$\{\{c, q\}, \{c, nq\}\}$	0.08952944
w_2	0	0	1	0	$\{\{a\}\}$	0.00435344
w_3	0	0	1	1	$\{\{a, c, q\}, \{a, c, nq\}\}$	0.02674256
w_4	0	1	0	0	$\{\{d, q\}, \{d, nq\}\}$	0.04148144
w_5	0	1	0	1	$\{\{c, d, q\}, \{c, d, nq\}\}$	0.25481456
w_6	0	1	1	0	$\{\{a, d, q\}, \{a, d, nq\}\}$	0.01239056
w_7	0	1	1	1	$\{\{a, c, d, q\}, \{a, c, d, nq\}\}$	0.07611344
w_8	1	0	0	0	$\{\{b\}\}$	0.01345344
w_9	1	0	0	1	$\{\{b, c, q\}, \{b, c, nq\}\}$	0.08264256
w_{10}	1	0	1	0	$\{\{a, b, q\}\}$	0.00401856
w_{11}	1	0	1	1	$\{\{a, b, c, q\}\}$	0.02468544
w_{12}	1	1	0	0	$\{\{b, d, q\}, \{b, d, nq\}\}$	0.03829056
w_{13}	1	1	0	1	$\{\{b, c, d, q\}, \{b, c, d, nq\}\}$	0.23521344
w_{14}	1	1	1	0	$\{\{a, b, d, q\}\}$	0.01143744
w_{15}	1	1	1	1	$\{\{a, b, c, d, q\}\}$	0.07025856

Example 5. *Suppose we visit a doctor, since we have headache (evidence), unsure whether this is due to a cold or just an allergy (query probabilistic facts). These two cause the same symptoms. If someone has a allergy, they may have a headache or fever. If someone has a cold, with probability 0.9 they have a fever and with probability 0.1 they have a headache.*

```
map 0.7::allergy. map 0.8::cold.
0.9::cold_fever. 0.1::cold_headache.
headache ; fever :- allergy.
fever :- cold, cold_fever.
headache :- cold, cold_allergy.
```

Here, the doctor may be interested in knowing the most probable cause for the headache, so asking the MAP state for headache *with query probabilistic facts* allergy *and* cold.

Algebraic Model Counting. Algebraic Model Counting [31] (AMC) is a lingua franca to express many tasks in (Statistical Relational) Artificial Intelligence. It generalizes the well-known task of Weighted Model Counting, where the goal is to find the weight of a propositional formula, computed as the sum of the weight associated with models, where the weight of a model is computed as the product of the weights of the literals in it. AMC, instead of a summation of products, generalizes this to a commutative semiring. The authors of [30] realized that some interesting tasks cannot be expressed with AMC. Thus, they proposed the Second-Level Algebraic Model Counting (2AMC) [30] framework. However, 2AMC is still not sufficient to solve, for example, decision theory problems under the credal semantics [8]. To overcome this, Three-Level Algebraic Model Counting (3AMC) was recently proposed [8]. We now formally define it.

Definition 2 (3AMC)**.** *Given a propositional theory Π whose variables are partitioned into (V_i, V_m, V_o) (we use the subscript/superscript i, m, and o to denote the innermost, middle, and outer layer, respectively), three commutative semirings $\mathcal{R}_i = (R^i, \oplus^i, \otimes^i, n^i_\oplus, n^i_\otimes)$, $\mathcal{R}_m = (R^m, \oplus^m, \otimes^m, n^m_\oplus, n^m_\otimes)$, and $\mathcal{R}_o = (R^o, \oplus^o, \otimes^o, n^o_\oplus, n^o_\otimes)$, three weight functions, w_i, w_m, and w_o,*

and two transformation functions f_{im} and f_{mo} mapping the values of R^i to R^m and the values of R^m to R^o, respectively, let $T = (\Pi, V_i, V_m, V_o, \mathcal{R}_i, \mathcal{R}_m, \mathcal{R}_o, w_i, w_m, w_o, f_{im}, f_{mo})$, $\mu(V)$ be the set of assignments to variables in V and $\varphi(\Pi \mid I)$ be the set of assignments to the remaining variables in Π given that the variables in I are set to true. 3AMC on T is defined as:

$$\begin{aligned} 3AMC(T) = &\bigoplus\nolimits^{o}_{I_o \in \mu(V_o)} \bigotimes\nolimits^{o}_{a \in I_o} w_o(a) \otimes^o \\ &f_{mo}(\bigoplus\nolimits^{m}_{I_m \in \varphi(\Pi \mid I_o)} \bigotimes\nolimits^{m}_{b \in I_m} w_m(b) \otimes^m \\ &f_{im}(\bigoplus\nolimits^{i}_{I_i \in \varphi(\Pi \mid I_o \cup I_m)} \bigotimes\nolimits^{i}_{c \in I_i} w_i(c))). \end{aligned} \tag{3}$$

We call a tuple $A_j = (V_j, \mathcal{R}_j, w_j)$ a *layer* (3AMC has 3 layers). 3AMC requires that the transformation functions must be monoid homomorphism for the multiplication ($\otimes$) of the semirings they consider. That is, multiplying and then applying the transformation function must yield the same result as first applying the transformation function and then multiplying the results.

Definition 3 (Monoid Homomorphism)**.** *Given two monoids $\mathcal{M}_0 = (D_0, \otimes_0, e_0)$ $\mathcal{M}_1 = (D_1, \otimes_1, e_1)$ and where, for $i \in \{0, 1\}$, D_i are the domains, $\otimes_i$ are binary operations and e_i are the neutral elements for such operations, a function $f : \mathcal{M}_0 \to \mathcal{M}_1$ is a monoid homomorphism if $\forall m_a, m_b \in \mathcal{D}_0$, $f(m_a \otimes_0 m_b) = f(m_a) \otimes_1 f(m_b)$ and $f(e_0) = e_1$.*

A practical algorithm to solve 3AMC is implemented in the aspmc solver [22]. Describing the whole procedure is out of the scope of this paper, since it involves different steps involving grounding, cycle breaking, Clark's completion [23] guided by a tree decomposition of the program, and knowledge compilation (KC) [19] targeting d-DNNF [18]. Let us briefly discuss how tree decomposition works. The primal graph of a program is a graph where the nodes are the atoms appearing in the grounding of the rules and there is an edge between node x and node y if there is a rule in the grounding of the program where both appear. A three decomposition [12] of a graph $G = (V(G), E)$ (where V denotes the set of vertices) is a pair (T, χ) where T is a tree and χ is a labeling such that: i) $\forall v \in V(G), \exists t \in V(T)$ s.t. $v \in \chi(t)$; ii) $\forall \{v_1, v_2\} \in E, \exists t \in V(T)$ s.t., $v_1, v_2 \in \chi(t)$; and iii) $\forall v \in V(G)$, $\{t \in V(T) \mid v \in \chi(t)\}$ is a connected subtree of T. That is, each vertex of the tree is associated with a set of nodes called *bags*, and, for each node $v \in V(G)$, the bags containing v are connected. There can be many tree decompositions for a given graph. The width of a tree decomposition is the size of its largest bag minus 1 and the *treewidth* of a graph is the smallest width among all possible tree decompositions. Although computing the treewidth of a graph is a hard task, there are efficient solvers based on heuristics [2]. Moreover, many NP-complete problems become tractable when the treewidth is bounded.

3 Algebraic Representation of MAP

Early approaches to find the lower/upper MAP states [7] addressed only credal semantics using projected answer set enumeration [24], while we propose a 3AMC-based method. In detail:

- in the innermost layer (call it A_i), we have query probabilistic facts and probabilistic facts fixed, and we need to compute whether the current world contributes to the lower or upper MAP state (i.e., if the evidence is cautiously or bravely entailed) or to the MAP_{sm} state;
- in the middle layer (call it A_m), we have query probabilistic facts fixed and we need to compute the probability of the evidence;
- in the outer layer (call it A_o), we range over the possible combinations of query probabilistic facts.

Let us now formalize the encoding.

Definition 4 (MAP as 3AMC)**.** *Consider a PASP P with query probabilistic facts Q, probabilistic facts F, set of evidence literals e, and Herbrand base $\mathcal{B}_\mathcal{P}$. In general, the evidence e can be composed of a single atom. This is possible by adding a rule with the conjunction of atoms in e in the body and as head a new atom e' that does not appear elsewhere in the program, and considering as evidence e' (as in Example 2). To simplify notation, we consider an evidence atom e.*

Let us first focus on the credal semantics. For the innermost layer A_i we have [5]: as semiring, $\mathcal{R}_i = (\mathbb{N}^2, +, \cdot, (0,0), (1,1))$, as variables $V_i = \mathcal{B}_\mathcal{P} \setminus F$, as weight function.

$$w_i(a) = \begin{cases} (0,1) & \text{if } a \neq e \\ (1,1) & \text{otherwise.} \end{cases}$$

The first component of the tuple counts the number of models where the evidence e is true while the second counts all the models. As transformation function we consider $f_{im}(n_1, n_2) = (v_{lp}, v_{up})$ where $v_{lp} = 1$ if $n_1 = n_2$, 0 otherwise, and $v_{up} = 1$ if $n_1 > 0$, 0 otherwise, i.e.,

$$f_{im}((n_1, n_2)) = \begin{cases} (0,0) & \text{if } n_2 = 0 \\ (0,1) & \text{if } n_2 > n_1 \\ (1,1) & \text{if } n_1 = n_2 > 0 \end{cases}$$

As middle layer A_m we have: as semiring, $\mathcal{R}_m = ([0,1]^2, +, \cdot, (0,0), (1,1))$ (this is the probability semiring [31] extended to two dimensions), as variables $V_m = F \setminus Q$, and as weight function

$$w_m(a) = \begin{cases} (p,p) & \text{if } a = f, \text{ with } p :: f \\ (1-p, 1-p) & \text{if } a = not\ f, \text{ with } p :: f \\ (1,1) & \text{otherwise.} \end{cases}$$

As transformation function f_{mo} we have

$$f_{mo}((l,u)) = (l, u, \{\}, \{\}).$$

Lastly, as outer layer A_o, we have: as semiring $\mathcal{R}_o = (\mathbb{R}^2 \times 2^{|Q|^2}, max^4, times^4, (0,0,Q,Q), (1,1,\{\},\{\}))$, as variables $V_o = Q$, and as weight function

$$w_o(a) = \begin{cases} (p, p, \{a\}, \{a\}) & \text{if a=p with q.p.f. } p :: f \\ (1-p, 1-p, \{a\}, \{a\}) & \text{if a=not p with q.p.f. } p :: f \\ (1,1,\{a\},\{a\}) & \text{otherwise} \end{cases}$$

where q.p.f. stands for query probabilistic fact and where $max^4((v_0, v_1, S_0, S_1), (v_a, v_b, S_a, S_b)) = (v_x, v_y, S_x, S_y)$ with $v_x = v_0$ if $v_0 > v_a$ else $v_x = v_a$, $v_y = v_1$ if $v_1 > v_b$ else $v_y = v_b$, $S_x = S_0$ if $v_0 > v_a$ else $S_x = S_a$, and $S_y = S_1$ if $v_1 > v_b$ else $S_y = S_b$, and $times^4((v_0, v_1, S_0, S_1), (v_a, v_b, S_a, S_b)) = (v_0 \cdot v_a, v_1 \cdot v_b, S_0 \cup S_a, S_1 \cup S_b)$. The first two components store, respectively, the value associated with the lower and upper MAP states, while the last two the values of the query probabilistic facts selected for such states. With small modifications, we can obtain a 3AMC encoding for the smProbLog semantics. Basically, instead of considering both the lower and upper probability, we have a single value.

The innermost layer remains the same. As transformation function between the innermost layer and the middle layer, $f_{im}^{sm}(n_1, n_2)$, we consider $f_{im}^{sm}(n_1, n_2) = n_1/n_2$. The remaining components can be straightforwardly obtained by considering only one value instead of two. We avoid reporting it to keep the notation uncluttered.

Note that, instead of having two different sets of very similar layers, one for the credal semantics and one for the smProbLog semantics, we can extend the set of layers for the credal semantics to also include the value for the smProbLog semantics. That is, instead of having two elements (lower and upper probability and lower and upper MAP state) in the tuples, we can have three (lower and upper probability for the credal semantics and probability for the smProbLog semantics and lower and upper MAP state for the credal semantics and MAP state for the smProbLog semantics). We prefer to keep the two separated since they attain two different semantics.

Example 6. *Consider Example 4. Here, $\mathcal{B}_\mathcal{P} = \{a, b, c, d, q, nq\}$, $V_i = \{q, nq\}$, $V_m = \{a, c\}$, and $V_o = \{b, d\}$. The query is q. In the outer layer, we iterate over the possible assignments of variables in V_o and, for each possible assignment I_o, we solve a 2AMC task on the variables V_m and V_i considering the variables in I_o fixed. In the middle layer, we iterate over the possible assignments of variables in V_m and, for each assignment $I_o \cup I_m$, we solve an AMC task on the variables V_i considering the assignments in I_o and I_m fixed. So, for example, suppose that we consider the assignment $\{d\}$ in V_o and $\{c\}$ in V_m. If we consider Equation 3, $\varphi(\Pi | I_o \cup I_m) = \{I_i^0, I_i^1\} = \{\{q\}, \{nq\}\}$. We need to compute the weights for each atom in I_i^0 and I_i^1: $w_i(q) = (1, 1)$ and $w_i(nq) = (0, 1)$. Then, the weight of I_i^0 is $(1, 1)$ and of I_i^1 is $(0, 1)$. We sum the two pairwise and obtain $(1, 2)$. Now, we need to apply the transformation function to this value to make it compatible for the middle layer. For the credal semantics, with f_{im} we obtain $(0, 1)$ while for the smProbLog semantics with f_{im}^{sm} we have 0.5. Then, the process continues by considering probabilistic facts (middle layer) and query probabilistic facts (outer layer).*

Let us now prove that the two transformation functions are monoid homomorphism, as required by 3AMC.

Theorem 1. *Consider $\mathcal{M}_i = (\mathbb{N}^2, \cdot, (1, 1))$ (obtained from $\mathcal{R}_i$), $\mathcal{M}_m = ([0, 1]^2, \cdot, (1, 1))$ (obtained from $\mathcal{R}_m$), $\mathcal{M}_o = (\mathbb{R}^2 \times 2^{|Q|^2}, times^4, (1, 1, \{\}, \{\}))$ (obtained from $\mathcal{R}_o$) and the transformation functions f_{im} and f_{mo} as per Definition 4. Then, f_{im} is a monoid homomorphism from $\mathcal{M}_i$ to $\mathcal{M}_m$ and f_{mo} is a monoid homomorphism from $\mathcal{M}_m$ to $\mathcal{M}_o$.*

Proof. (Sketch.) First, focus on f_{im} and consider two tuples (a, b) and (c, d). We have to prove that, $f_{im}((a, b) \cdot (c, d)) = f_{im}((ac, bd)) = f_{im}((a, b)) \cdot f_{im}((c, d))$. This is easy to see by checking the possible outcomes from the right- and left-hand side, which can be $(0, 0)$, $(0, 1)$, or $(1, 1)$, according to the values of a, b, c, and d. The neutral element is preserved as $f_{im}((1, 1)) = (1, 1)$. For f_{mo}, we have $f_{mo}((a, b) \cdot (c, d)) = f_{mo}((ac, bd)) = (ac, bd, \{\}, \{\}) = f_{mo}((ac, bd))$ and for the neutral element $f_{mo}((1, 1)) = (1, 1, \{\}, \{\})$. The proofs for the smProbLog innermost transformation function can be found in [30]. □

4 Experimental Evaluation

In this section we conduct an empirical evaluation aimed at assessing the impact of the proposed techniques w.r.t. previous approaches existing in the literature. The *only* existing tool (other than our proposal) which supports MAP inference in the general case (i.e., *non-stratified* PASP) is PASTA [7], and it is based on projected answer set enumeration [24]. Basically, to compute the lower and upper bounds probability for a query, it enumerates all the worlds and, for each of them, checks whether there is one answer set with the query and one without the query. For MAP, this is extended by grouping answer sets containing the same set of query probabilistic facts.

Concerning instead *stratified* PASP, many other systems are available, namely, cplint [10], ProbLog [20], and aspmc [22] (denoted by aspmc2), which are based on knowledge compilation. For all these tools, we leave every parameter as default. By inspecting the ProbLog source code, we note that it encodes MAP as a decision theory task [48], and so it returns a score, rather than a probability associated with evidence. All the mentioned systems have been compared with the techniques proposed in this paper, denoted by aspmc3.

Due to the distinction among the different tools on the supported program types, in what follows, we study separately the stratified and non-stratified cases. Note that, in the stratified case, the lower and upper MAP states under the credal semantics and the MAP state under the smProbLog semantics coincide (and also coincide with the MAP state obtained with the tools targeting stratified programs). Conversely, the results obtained with the two semantics may not coincide in the non-stratified case (as shown in Example 4). To the best of our knowledge, no implementation for the MAP task under smProbLog semantics is available. All experiments were executed on a machine with Rocky Linux 9.2 running at 3.7 GHz with a time limit of 1 hour and a memory limit of 32 GB. Source code and datasets available at [4].

4.1 Non-stratified Datasets

We consider six different benchmarks. For all, the evidence is the atom qr. For each of the four following benchmarks, an instance of size i has a total number of i probabilistic facts having the signature $aj/0$ with $j \in \{0, \dots, i-1\}$. For compactness, we do not report them in the code snippets below. These can be considered as a generalization of stratified datasets already adopted in related scenarios [10]. Being synthetically generated, this allows us to easily assess the scalability of the systems.

An instance of size i of the dataset $p1$ has $i/2$ rules of the form $qr_j; nqr_j :- a_j$ with j even where a_j can be a positive or negative literal, $i/2$ rules of the form $qr_{k-1} :- a_k$ with k odd and a_k as before, and a rule $qr :- \bigwedge qr_i$ with i even. An example of size 4 is

```
qr0 ; nqr0:- not a0. qr0:- not a1.
qr2 ; nqr2:- not a2. qr2:-  a3.
qr:- qr0,qr2.
```

An instance of size i of the dataset $p2$ has $i/2$ rules of the form $qr_j; nqr_j :- \circ, a_j$ with j even, where $\circ$ can be qr_{k-1} or nqr_{k-1} and a_j as before, $i/2$ rules of the form $qr_k :- \circ, a_k$ with k odd and the rest as before, and i rules $qr :- qr_i$. An example of size 4 is

```
qr0 ; nqr0:- not a0. qr1:- nqr0, not a1.
qr2 ; nqr2:- qr1,  a2. qr3:- qr2,  a3.
qr:- qr0. qr:- qr1. qr:- qr2. qr:- qr3.
```

An instance of size i of the dataset $p3$ has i rules of the form $\bigvee_{j \in \{0, i-1\}} qr_j :- a_i$ and a rule $qr :- qr_j$ for each $j \in \{0, i-1\}$. An example of size 4 is

```
qr0:- not a0. qr0;qr1:- not a1.
```

```
qr0;qr1;qr2:- not a2. qr0;qr1;qr2;qr3:-  a3.
qr:- qr0. qr:- qr1. qr:- qr2. qr:- qr3.
```

An instance of size i of the dataset $p4$ has one rule with head $qr; nqr$ and body composed of a conjunction of random (query) probabilistic facts where each one can be also considered as negated and a rule with head qr and body as before. Note that here some of the (query) probabilistic facts may not be considered in the two rules. An example of size 4 is

```
qr ; nqr:- not a1, a2. qr:- a1, a2.
```

Here, probabilistic facts $a0$ and $a3$ are present in the program but not in the rules. For all, generated instances are of sizes from 5 up to 100.

In addition, we considered a probabilistic extension of the well-known graph coloring problem (*col*) encoded in ASP, which is a standard benchmark adopted in ASP competitions [14, 25], where the edges are considered as (query) probabilistic facts and the evidence is the color of a given node, and the friend-smokers (*smk*) benchmark [47], representing a network of people where the smoking behavior is influenced by friendship and existing pathologies. Here, the evidence encodes whether at least one individual smokes. Note that for every instance on every benchmark, we ensured that every world has at least one model. An instance of size i of *col* has, as for the previous four benchmarks, a total number of probabilistic facts of i. We generated instances starting from 7 nodes and up to 30. An instance of size i of smokers has a total number of probabilistic facts of $5i - 1$. We generated instances starting from size 2 and up to 10. Since aspmc does not support disjunction in the head, we shifted disjunctive rules and represented them with normal rules and negation. This is possible because all the programs are head-cycle free. Furthermore, for aspmc, the negation symbol *not* is replaced with $\backslash+$. For all the six benchmarks described above, we also generated two variations, one with random probabilities and one with a fixed probability of 0.4 associated to probabilistic facts. In addition, for each of these two variations, we also have three sub-variations, with 20%, 50%, and 80% of the probabilistic facts considered as query probabilistic facts (possibly rounded to the preceding integer), denoted with subscript 20, 50, or 80, respectively. Overall, we conducted the experimental evaluation on more than 2500 combined instances.

Here, we aim to empirically answer the following questions: **Q1**) is 3AMC/knowledge compilation a good candidate to solve MAP in PASP and is it better than projected answer set enumeration?; **Q2**) does the probability associated with (query) probabilistic facts influence the execution time?

Non-stratified Datasets Results. Table 2 shows the largest solvable instance for aspmc3 and PASTA. Here, and in the following, with aspmc3 we denote the results obtained by considering the credal semantics. We obtained the same statistics by considering the smProbLog semantics. This was expected, since the 3AMC encodings for the credal and the smProbLog semantics are almost the same; indeed, they mainly differ on the innermost transformation function. For this reason, we avoid reporting statistics about solving the MAP task under smProbLog semantics, and we provide a comparison with PASTA, which supports only the credal semantics. Thus, in the following, *the statistics reported for aspmc3 are representative of both the credal and smProbLog semantics*.

Our 3AMC approach, in most of the datasets, outperforms PASTA, allowing us to positively answer **Q1**. For col, we obtain comparable performance while for $p3$ aspmc3 perform worse (3 instances less). However, for $p2$ and $p4$, aspmc3 can solve four times the instances solved by PASTA, in few seconds. This may be due to the aspmc3 sophisticated pipeline, which is able to find a good and compact representation for the task. It is interesting to note that, almost always, the largest solvable instance for aspmc3 is solved at approximately less than 1/6 of the total execution time available (3600 seconds). After that, we obtain a knowledge compilation error from the c2d tool, which requires too much memory. It is also worth noting that the c2d version available in aspmc is free but closed source. By inspecting it with the bash command `file` we get that it was compiled for 32-bit systems (`ELF 32-bit LSB executable`) and for `GNU/Linux 2.2.5`. Thus, a fresh compilation on modern systems may probably improve its performance. The good performance of aspmc3 was expected, since KC-based approaches has been proven effective in PLP/PASP. Also, we see that the probabilities associated with facts do not seem to influence too much the execution time (**Q2**), even if for $p1_{20}$ and smk_{50}, with random probabilities aspmc can solve up to instance of size 33 and 6, while with fixed probabilities it gets to size 34 and 7, respectively. Conversely, for $p1_{80}$, the largest solvable instance for fixed probabilities is 33 while for random it is 34. Also for PASTA, the probability attached to the facts does not seem to influence the execution time.

Table 2. Largest solvable instance (sz.) and its execution time in seconds (t.) for aspmc3 and PASTA on different datasets (D.).

	Random Prob.				Fixed Prob.			
	aspmc3		PASTA		aspmc3		PASTA	
D.	sz.	t.	sz.	t.	sz.	t.	sz.	t.
col_{20}	24	560	23	2972	24	838	23	2972
col_{50}	24	512	23	2693	24	304	23	2758
col_{80}	24	276	23	2703	24	460	23	2693
$p1_{20}$	33	481	24	3503	34	593	24	3479
$p1_{50}$	34	607	24	3481	34	585	24	3372
$p1_{80}$	34	566	24	3478	33	463	24	3354
$p2_{20}$	100	5	23	2015	100	5	23	1997
$p2_{50}$	100	5	23	2014	100	5	23	2033
$p2_{80}$	100	6	23	2016	100	6	23	1989
$p3_{20}$	20	433	23	1795	20	438	23	1773
$p3_{50}$	20	458	23	1797	20	444	23	1804
$p3_{80}$	20	530	23	1789	20	456	23	1795
$p4_{20}$	100	5	24	3358	100	4	24	2930
$p4_{50}$	100	24	24	3437	100	23	24	2917
$p4_{80}$	100	39	24	3319	100	36	24	3026
smk_{20}	7	324	4	120	7	275	4	116
smk_{50}	6	40	4	104	7	658	4	119
smk_{80}	7	751	4	120	7	434	4	106

To better inspect the behavior of aspmc3, we also analyzed the statistics of treewidth, number of bags, and number of vertices for the largest solvable instances (Table 3). We find that instances with random and fixed probabilities have comparable results. For all datasets, the treewidth, number of vertices, and number of bags remain almost the same between random and fixed probabilities and among different percentages of query probabilistic facts. This is expected as these structural properties are independent of the probabilities.

Overall, encoding MAP as a 3AMC problem and solving it via knowledge compilation has a high impact on execution time, greatly increasing the scalability.

4.2 Stratified Datasets

We also generated three benchmarks with stratified datasets, to compare our approach based on 3AMC against cplint, ProbLog, and aspmc with 2AMC. Note that these tools only support MAP in *stratified* programs, so our approach based on 3AMC generalizes these. Here, we refrain from comparing against PASTA, since it is not com-

Table 3. Tree decomposition statistics for aspmc. The columns tw, #b, and #v contain: the treewidth, the number of bags, and the number of vertices for the tree decomposition obtained on the largest solvable instance (sz).

	Random Prob.				Fixed Prob.			
D.	sz.	tw.	#b.	#v.	sz.	tw.	#b.	#v.
col_{20}	24	20	262	292	24	20	262	290
col_{50}	24	15	263	290	24	13	256	285
col_{80}	24	19	261	287	24	19	251	284
$p1_{20}$	33	28	28	133	34	29	103	136
$p1_{50}$	34	18	33	136	34	18	35	136
$p1_{80}$	34	24	28	136	33	26	100	133
$p2_{20}$	100	83	38	697	100	83	46	697
$p2_{50}$	100	53	58	697	100	55	55	697
$p2_{80}$	100	83	36	697	100	84	36	697
$p3_{20}$	20	40	421	460	20	40	420	460
$p3_{50}$	20	40	421	460	20	40	417	460
$p3_{80}$	20	40	421	460	20	40	418	460
$p4_{20}$	100	15	13	53	100	15	13	53
$p4_{50}$	100	34	47	96	100	34	47	96
$p4_{80}$	100	57	57	118	100	57	57	118
smk_{20}	7	41	86	690	7	42	70	688
smk_{50}	6	32	70	510	7	41	84	692
smk_{80}	7	41	73	688	7	41	71	688

petitive with aspmc3, as also shown in the previous set of experiments, and cannot scale on programs with over 24 probabilistic facts. Now, we want to answer the following question: **Q3**) how much do we pay in terms of execution time for a larger expressivity?

We considered three datasets: $q1$ where there is a single rule of the form $q :- \bigwedge_{i=1}^{n} a_i$ where a_i can be a positive or negative literal for i odd, omitted otherwise and a_i is a (query) probabilistic fact and n is the total number of them (which also represent the size of the instance). $q2$ where there is a rule $q :- a_i$ for each possible (query) probabilistic fact, with the same setting as $q1$. These two are a variation of the ones adopted for non-stratified datasets. The last one, $cmpl$, contains a set of instances representing complete directed graphs of size $k \in \{2, \dots, 20\}$ where each node is represented with an integer starting from 0 and the edges are represented by $edge/2$ probabilistic facts. There are two rules in these programs $path(A, B) :- edge(A, B)$ and $path(A, B) :- edge(A, C), path(C, B)$. encoding the reachability problem, still largely adopted in ASP competitions [14, 25]. The query is $path(0, k-1)$. Note that the total number of probabilistic facts for an instance of size k is $k \cdot (k-1)/2$. For these experiments, we fixed the probabilities to 0.4 and, as before, we generated instances with 20%, 50%, and 80% of total probabilistic facts considered as query probabilistic facts.

Stratified Datasets Results. Figure 1 shows that the tools based on AMC (aspmc2 and our proposal, aspmc3) have analogous performance: this shows that adding an additional layer of AMC does not seem to influence the execution time on those datasets. The tool cplint is one that can solve most instances, while ProbLog is the worst performing one. Interestingly, aspmc3, although more general than ProbLog, outperforms it. This allows us to answer **Q3** by saying that, despite being more general, aspmc3 is still competitive with specialized tools for a special case of the problem at hand.

5 Related Works

The CS requires at least one model per world. The dPASP tool [26] supports inference in PASP via answer set enumeration, but does not support MAP inference. Worlds without models are allowed by the L-credal semantics [37, 45] that considers a third truth value (undefined) for atoms. Handling the MAP task under such a semantics is an interesting future work. Other semantics exist to express probabilities and weights with ASP, such as LPMLN [33] and P-log [9], and the relationship between these two and weak constraints in ASP (which allow expressing preferences among answer sets) has been studied in [34]. If we focus on the LPMLN semantics, the author of [35] discussed the MAP task and considered it as "the system finds a most probable stable model [...] (MAP estimate)". However, this is not how MAP is usually defined in Statistical Relational Artificial Intelligence (and so in our definition), rather it is the MPE task. Our MAP is a substantially harder problem that involves both marginalization and maximization, while their definition of MAP involves maximization only. Unfortunately, it is well known [32] that there is some ambiguity in the adopted terms, since some works call MPE as MAP and MAP as marginal MAP.

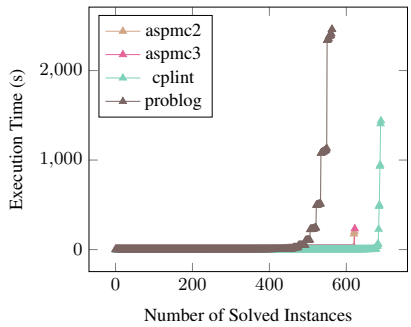

Figure 1. Cactus plot for stratified datasets.

Tasks such as abduction [6] and decision theory [8, 48] have been studied in PLP/PASP and are related to MAP, although not equivalent. The former requires finding a subset of so-called "abducibles facts" (which are not associated with a probability, differently from query probabilistic facts) such that the probability of a query is maximized, and ties between different subsets are broken according to a minimality criterion, such as subset or cardinality minimality. The latter, instead, associates a utility (that is, a real number) to some (possibly all) atoms in the program and requires finding a subset of so-called "decision atoms" (which, again, are not associated with a probability value) that maximizes the expected utility of the program.

6 Conclusions

In this paper, we considered the MAP task in Probabilistic Answer Set Programming under the credal semantics and the smProbLog semantics, and proposed a Three-Level Algebraic Model Counting encoding. We implemented it into the aspmc solver and ran an experimental evaluation against a general solver for non-stratified programs and specific solvers for stratified programs. Empirical results show the greater scalability of our novel approach for non-stratified datasets and competitive performance against specialized tools handling stratified programs only.

Future work might include supporting disjunctive programs as well as considering alternative approaches such as ASP(Q) [38], and completing a study of the complexity (e.g., checking whether MAP under smProbLog is $\mathrm{NP}^{\mathrm{PP}}$-complete).

Acknowledgements

This work has been partially supported by Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR

Missione 4 - Next Generation EU (NGEU), by the Italian Ministry of Industrial Development (MISE) under project EI-TWIN n. F/310168/05/X56 CUP B29J24000680005, and by the Italian Ministry of Research (MUR) under projects: PNRR FAIR - Spoke 9 - WP 9.1 CUP H23C22000860006, Tech4You CUP H23C22000370006, and PRIN PINPOINT CUP H23C22000280006. DA and FR are members of the Gruppo Nazionale Calcolo Scientifico – Istituto Nazionale di Alta Matematica (GNCS-INdAM). We thank Rafael Kiesel for helping us with the implementation on top of aspmc.

References

[1] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[2] M. Abseher, N. Musliu, and S. Woltran. htd – a free, open-source framework for (customized) tree decompositions and beyond. In *Integration of AI and OR Techniques in Constraint Programming*, 2017. doi: 10.1007/978-3-319-59776-8_30.

[3] M. Alberti, E. Bellodi, G. Cota, F. Riguzzi, and R. Zese. cplint on SWISH: Probabilistic logical inference with a web browser. *Intelligenza Artificiale*, (1), 2017. doi: 10.3233/IA-170105.

[4] D. Azzolini. Source code and datasets for the article: An algebraic view of map inference in probabilistic answer set programs, July 2025. URL https://doi.org/10.5281/zenodo.16356712.

[5] D. Azzolini and F. Riguzzi. Inference in probabilistic answer set programming under the credal semantics. In *AIxIA 2023 - Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 367–380, 2023. doi: 10.1007/978-3-031-47546-7_25.

[6] D. Azzolini, E. Bellodi, S. Ferilli, F. Riguzzi, and R. Zese. Abduction with probabilistic logic programming under the distribution semantics. *International Journal of Approximate Reasoning*, 2022. doi: 10.1016/j.ijar.2021.11.003.

[7] D. Azzolini, E. Bellodi, and F. Riguzzi. MAP inference in probabilistic answer set programs. In *AIxIA 2022 – Advances in Artificial Intelligence*, pages 413–426, 2023. doi: 10.1007/978-3-031-27181-6_29.

[8] D. Azzolini, E. Bellodi, R. Kiesel, and F. Riguzzi. Solving decision theory problems with probabilistic answer set programming. *Theory and Practice of Logic Programming*, (1):33–63, 2025. doi: 10.1017/S1471068424000474.

[9] C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, (1):57–144, 2009. doi: 10.1017/S1471068408003645.

[10] E. Bellodi, M. Alberti, F. Riguzzi, and R. Zese. MAP inference for probabilistic logic programming. *Theory and Practice of Logic Programming*, (5):641–655, 2020. doi: 10.1017/S1471068420000174.

[11] R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, (1), Mar 1994. doi: 10.1007/BF01530761.

[12] H. L. Bodlaender et al. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–21, 1993.

[13] G. Brewka, T. Eiter, and M. Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[14] F. Calimeri, M. Gebser, M. Maratea, and F. Ricca. Design and results of the fifth answer set programming competition. *Artificial Intelligence*, 2016. doi: https://doi.org/10.1016/j.artint.2015.09.008.

[15] J. Carron and A. Lewis. Maximum a posteriori CMB lensing reconstruction. *Physical Review D*, 96(6):063510, 2017.

[16] F. G. Cozman and D. D. Mauá. The complexity of inferences and explanations in probabilistic logic programming. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 2017. doi: 10.1007/978-3-319-61581-3_40.

[17] F. G. Cozman and D. D. Mauá. The joy of probabilistic answer set programming: Semantics, complexity, expressivity, inference. *International Journal of Approximate Reasoning*, 2020. doi: 10.1016/j.ijar.2020.07.004.

[18] A. Darwiche. Decomposable negation normal form. *Journal of the ACM (JACM)*, 48(4):608–647, 2001.

[19] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 2002. doi: 10.1613/jair.989.

[20] L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, 2007.

[21] J. Dix, G. Gottlob, and V. W. Marek. Reducing disjunctive to non-disjunctive semantics by shift-operations. *Fundam. Informaticae*, (1-2): 87–100, 1996. doi: 10.3233/FI-1996-281205.

[22] T. Eiter, M. Hecher, and R. Kiesel. aspmc: New frontiers of algebraic answer set counting. *Artificial Intelligence*, page 104109, 2024. doi: 10.1016/j.artint.2024.104109.

[23] F. Fages. Consistency of clark's completion and existence of stable models. *Journal of Methods of logic in computer science*, 1(1):51–60, 1994.

[24] M. Gebser, B. Kaufmann, and T. Schaub. Solution enumeration for projected boolean search problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 71–86, 2009. doi: 10.1007/978-3-642-01929-6_7.

[25] M. Gebser, M. Maratea, and F. Ricca. The seventh answer set programming competition: Design and results. *Theory and Practice of Logic Programming*, (2), 2020. doi: 10.1017/S1471068419000061.

[26] R. L. Geh, J. Gonçalves, I. C. Silveira, D. D. Mauá, and F. G. Cozman. dPASP: a probabilistic logic programming environment for neurosymbolic learning and reasoning. In *KR*, volume 21, pages 731–742, 2024.

[27] A. V. Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[28] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *5th International Conference and Symposium on Logic Programming (ICLP/SLP 1988)*, volume 88, pages 1070–1080, 1988.

[29] J. Huang, M. Chavira, A. Darwiche, et al. Solving MAP exactly by searching on compiled arithmetic circuits. In *AAAI*, volume 6, 2006.

[30] R. Kiesel, P. Totis, and A. Kimmig. Efficient knowledge compilation beyond weighted model counting. *Theory and Practice of Logic Programming*, (4):505–522, 2022. doi: 10.1017/S147106842200014X.

[31] A. Kimmig, G. Van den Broeck, and L. De Raedt. Algebraic model counting. *J. of Applied Logic*, July 2017. doi: 10.1016/j.jal.2016.11.031.

[32] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. 2009.

[33] J. Lee and Y. Wang. Weighted rules under the stable model semantics. In *KR*, pages 145–154, 2016.

[34] J. Lee and Z. Yang. LPMLN, weak constraints, and P-log. In *AAAI*, pages 1170–1177, 2017.

[35] J. Lee, S. Talsania, and Y. Wang. Computing LPMLN using ASP and MLN solvers. *Theory and Practice of Logic Programming*, (5-6), 2017. doi: 10.1017/S1471068417000400.

[36] D. D. Mauá and F. G. Cozman. Complexity results for probabilistic answer set programming. *International Journal of Approximate Reasoning*, pages 133–154, 2020. doi: 10.1016/j.ijar.2019.12.003.

[37] D. D. Mauá, F. G. Cozman, and A. Garces. Probabilistic logic programming under the L-stable semantics. In *NMR*, pages 24–33, 2024.

[38] G. Mazzotta, F. Ricca, and M. Truszczynski. Quantifying over optimum answer sets. *Theory and Practice of Logic Programming*, 2024. doi: 10.1017/S1471068423000121.

[39] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In *2009 IEEE International Conference on Robotics and Automation*, pages 1402–1409. IEEE, 2009.

[40] J. D. Park and A. Darwiche. Solving MAP exactly using systematic search. In *UAI*, pages 459–468, 2002.

[41] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

[42] A. Prémaud, Y. Le Meur, J. Debord, J.-C. Szelag, A. Rousseau, G. Hoizey, O. Toupance, and P. Marquet. Maximum a posteriori bayesian estimation of mycophenolic acid pharmacokinetics in renal transplant recipients at different postgrafting periods. *Therapeutic drug monitoring*, 27(3):354–361, 2005.

[43] L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2): 1–189, 2016.

[44] F. Riguzzi. *Foundations of Probabilistic Logic Programming Languages, Semantics, Inference and Learning, Second Edition*. River publishers, 2022.

[45] V. H. N. Rocha and F. Gagliardi Cozman. A credal least undefined stable semantics for probabilistic logic programs and probabilistic argumentation. In *KR*, 2022. doi: 10.24963/kr.2022/31.

[46] T. Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, 1995. doi: 10.7551/mitpress/4298.003.0069.

[47] P. Totis, L. De Raedt, and A. Kimmig. smProbLog: Stable model semantics in ProbLog for probabilistic argumentation. *Theory and Practice of Logic Programming*, pages 1–50, 2023. doi: 10.1017/S147106842300008X.

[48] G. Van den Broeck, I. Thon, M. van Otterlo, and L. De Raedt. DTProbLog: A decision-theoretic probabilistic Prolog. In *AAAI*, pages 1217–1222, 2010.