A First Journey into the Complexity of Statistical Statements in Probabilistic Answer Set Programming

Damiano Azzolini[1], Markus Hecher[2]

[1]University of Ferrara, Ferrara, Italy
[2]Massachusetts Institute of Technology, Cambridge, United States

Abstract

Probabilistic Answer Set Programming is an efficient formalism to express uncertain information with an answer set program (PASP). Recently, this formalism has been extended with statistical statements, i.e., statements that can encode a certain property of the considered domain, within the PASTA framework. To perform inference, these statements are converted into answer set rules and constraints with aggregates. The complexity of PASP has been studied in depth, with results regarding both membership and completeness. However, a complexity analysis of programs with statements is missing. In this paper, we close this gap by studying the complexity of PASTA statements.

Keywords

Probabilistic Answer Set Programming, Computational Complexity, Statistical Statements

1. Introduction

Representing uncertain information with an interpretable language is currently one of the main goal of Statistical Relational Artificial Intelligence [1]. Among the different languages, we have ProbLog [2], Logic Programs with Annotated Disjunctions [3], and Stochastic Logic Programs [4] while some of the possible semantics are the Distribution Semantics [5] and the Credal Semantics [6]. Answer Set Programming [7] is a logic-based language that has been successfully applied to model combinatorial domains. When the information is uncertain, we can adopt the Credal Semantics, that gives a meaning to Answer Set Programs extended with probabilistic facts [2]. These programs are called *Probabilistic* Answer Set Programs (PASP).

More than 30 years ago, Halpern [8] introduced the so-called "Type 1 statments", also known as "probabilistic conditionals" or "statistical statements", that allow expressing statistical information about a domain of interest. An example of such a statement is "X% of elements of a domain have the property Y". Recently, the authors of [9] introduced the possibility to express Type 1 statements with Probabilistic Answer Set Programming, that are converted into a disjunctive rule and a constraint with aggregates, and provided a framework called PASTA.

Inference in PASP has been proved hard [6, 10, 11]. For example, the inference task (i.e., the computation of the probability of a query) for propositional programs lies in the $\mathrm{PP}^{\mathrm{NP}}$ complexity class for programs allowing disjunctive rules only, where PP represents the class of the decision problems that can be solved in polynomial time with a probabilistic Turing machine [12]. If we add stratified negation the complexity jumps to the $\mathrm{PP}^{\Sigma_2^{\mathrm{p}}}$ class of the polynomial hierarchy [13]. Also in practice, inference in such programs is very expensive [9, 14].

In this paper, we focus on PASP composed by a set of probabilistic facts and a set of statistical statements only, and we study the complexity of inference in such programs. Theoretical results show that, in such simple programs, inference can be efficiently computed.

The paper is structured as follows: Section 2 discusses some background knowledge related to Answer Set Programming, Probabilistic Answer Set Programming and statistical statements, and Computational Complexity. In Section 3 we study the complexity of programs with statistical statements. Section 4 surveys related work and Section 5 concludes the paper.

PLP 2024: 11th Workshop on Probabilistic Logic Programming, October, 2024, Dallas, USA.

2. Background

2.1. Answer Set Programming

Answer Set Programming [7] (ASP) is one of the possible logic-based languages suitable to model combinatorial domains. An answer set program is composed by a set of disjunctive rules of the form

$$h_0; \ldots; h_m :- b_0, \ldots, b_n$$

where $h_0; \ldots; h_m$ is a disjunction of atoms called *head* and $b_0, \ldots, b_n$ is a conjunction of literals called *body*. We use not to denote negation. Head and body may be empty: if the heady is empty (but the body is not), the rule is called *constraint*; conversely, if the head has only one atom and the body is empty, it is called *fact*. An example of disjunctive rule is $a; b :- c$, where $a; b$ is the head and c is the body. Another construct often adopted in answer set programs are *aggregates* [15]. An aggregate has the form $t_0\psi_0\#f\{e_0; \ldots; e_n\}\psi_1 t_1$, where the e_is are aggregate terms of the form $t_0, \ldots, t_k : \varphi$ where each t_0 is a term and φ is a conjunction of literals, f is an aggregation function such as $count$ or sum, ψ_0 and ψ_1 are arithmetic comparison operators, and t_0 and t_1 are constants. An example of aggregate is $\#count\{X : a(X)\} > 3$.

We consider the Stable Model Semantics [16] as semantics for answer set programs. Under this semantics, each program is associated with zero (in this case the program is often termed *unsatisfiable*) or more stable models. A program is called *ground* when there are no variables in it. A non ground program can be transformed into a ground program with a process called *grounding*. The Herbrand base of a ground program P is the set of all possible ground atoms that can be constructed using the symbols defined in P, often indicated with B_P. An interpretation is a subset of B_P. The *reduct* of a ground program P w.r.t. a subset I of B_P (this subset is called *interpretation*), denoted with P^I, is computed by removing from P the rules having at least one literal in the body is false in I. An interpretation I is called *stable model* or *answer set* of a program P if it is a minimal model under set inclusion of P^I. We use $AS(P)$ to denote the set of answer sets of P. Let us clarify these definitions with an example.

Example 1. *The following answer set program has two facts and three rules.*

```
a.
b.

qa:- a.
qb:- b, not nqb.
nqb:- b, not qb.
```

It has two answer sets: $\{a, qa, b, nqb\}$ *and* $\{a, qa, b, qb\}$.

2.2. Probabilistic Answer Set Programming

To describe uncertain scenarios with Answer Set Programming, several semantics have been proposed, such as LPMLN [17], P-log [18], and the credal semantics [11]. Here we focus on the credal semantics, which extends an answer set program with probabilistic facts [2] of the form

$$\Pi :: a.$$

where $\Pi \in]0, 1[$ and a is an atom. Intuitively, Π is the probability that a is included in the program and $1 - \Pi$ that it is excluded. In the remaining part of the paper, we use the acronym PASP to indicate both Probabilistic Answer Set Programming under the credal semantics and a probabilistic answer set program interpreted under the credal semantics. Each PASP with n probabilistic fact defines 2^n world, obtained by considering each possible selection of probabilistic facts. Let us call W the set of all the worlds. The probability of a world $w \in W$ is computed as [5]

$$P(w) = \prod_{a_i \in w} P(a_i) \cdot \prod_{a_i \notin w} 1 - P(a_i)$$

that is, by considering the probabilistic facts a_i present or absent in w. In each world w the set of probabilistic facts is fixed, so each world is an answer set program. Thus, w may have 0 more answer sets but the *credal semantics* requires that it has at least one. In this paper, we will always assume that every PASP has at least one answer set per world. Given a conjunction of ground literals q, often called *query*, its probability $P(q)$ is described by a range $[\underline{\mathrm{P}}(q), \overline{\mathrm{P}}(q)]$ whose bounds depend on the number of answer sets of every world with the query included. A world w contributes to both the lower ($\underline{\mathrm{P}}(q)$) and upper ($\overline{\mathrm{P}}(q)$) bound for a query q if q is present in *every* answer set. Otherwise, if the query is true only in some answer sets, w contributes only to the upper probability bound. If the query is present in none of the answer sets of w, w contributes to none of the bounds. In formulas,

$$P(q) = [\underline{\mathrm{P}}(q), \overline{\mathrm{P}}(q)] = [\sum_{w_i \in W \ s.t. \ \forall m \in AS(w_i), \ m \models q} P(w_i), \sum_{w_i \in W \ s.t. \ \exists m \in AS(w_i), \ m \models q} P(w_i)]$$

Let us discuss a clarifying example.

Example 2. *The following program has two probabilistic facts, a and b.*

```
0.4::a.
0.3::b.

qa:- a.
qb:- b, not nqb.
nqb:- b, not qb.

q:- qa.
q:- qb.
```

It has $2^2 = 4$ worlds: w_0, where neither a nor b are present, whose probability is $(1-0.4)\cdot(1-0.3) = 0.6 \cdot 0.7 = 0.42$ and $AS(w_0) = \{\{\}\}$; w_1, where a is present and b is absent, whose probability is $0.4 \cdot (1-0.3) = 0.4 \cdot 0.7 = 0.28$ and $AS(w_1) = \{\{a, qa, q\}\}$; w_2, where a is absent and b is present, whose probability is $(1-0.4) \cdot 0.3 = 0.6 \cdot 0.3 = 0.18$ and $AS(w_2) = \{\{b, qb, q\}, \{b, nqb\}\}$; and w_3, where both a and b are present, whose probability is $0.4 \cdot 0.3 = 0.4 \cdot 0.3 = 0.12$ and $AS(w_3) = \{\{a, qa, b, qb, q\}, \{a, qa, b, nqb, q\}\}$. If we are interested in computing the probability of q, we have that w_0 contributes to none of the bounds since q is present in none of its answer sets, w_1 contributes to both the lower and upper bound since q is present in every answer set (there is only one with q in it), w_2 contributes to only the upper bound since q is present in only one of the two answer sets, and w_3 contributes to both the lower and upper bound since q is present in every answer set (both have q in it). Overall, $[\underline{\mathrm{P}}(q), \overline{\mathrm{P}}(q)] = [P(w_1) + P(w_3), P(w_1) + P(w_2) + P(w_3)] = [0.28 + 0.12, 0.28 + 0.18 + 0.12] = [0.4, 0.58]$.

Halpern's statistical statements [8], also known as probabilistic conditionals, allow representing statistical information about a domain, such as "X% of domain elements have the property Y". The authors of [9, 19] introduced the possibility of encoding such statements within a PASP with the syntax:

$$(C \mid A)[\Pi_l, \Pi_u]$$

where C and A are (conjunction of) atoms and $\Pi_l, \Pi_u \in [0, 1], \Pi_l \leq \Pi_u$. We call C consequent and A antecedent. The meaning is "the fraction of As that have the property C is between Π_l and Π_u". To perform inference, a statement is converted into a disjunctive rule with the structure $C; not\_C :- A$, describing that the property C may or may not hold for each element A, and a constraint with counting aggregates imposing the specified probability bound, i.e., imposing

$$\Pi_l \leq \frac{\#count\{\mathbf{X} : A(\mathbf{X}), C(\mathbf{X})\}}{\#count\{\mathbf{X} : A(\mathbf{X})\}} \leq \Pi_u.$$

Let us clarify it by discussing the following example from [9].

Example 3 (from [9]). *The following program has 3 probabilistic facts and one statistical statement.*

```
0.4::bird(1). 0.4::bird(2). 0.4::bird(3).
(fly(X) | bird(X))[0.6,1].
```

The probabilistic facts indicate that the animals indexed with 1 up to 4 are birds with a probability of 0.4. The statement describes that at least 60% of the birds fly (and at most 100%, all of them). To perform inference, this program is converted into;

```
0.4::bird(1). 0.4::bird(2). 0.4::bird(3).
fly(X) ; not_fly(X) :- bird(X).
:- #count{X:fly(X),bird(X)} = FB, #count{X:bird(X)} = B, 10*FB < 6*B.
```

If we want to compute the probability of the query $fly(1)$, we get 0.336 for the lower and 0.4 for the upper probability.

We can convert the program above into a program without counting aggregates, by enumerating all the possible results of the aggregates. For examples, the program shown in Example 3 can be converted into:

```
0.4::bird(1). 0.4::bird(2). 0.4::bird(3).
fly(X) ; not_fly(X) :- bird(X).
b1 :- bird(1).
b1 :- bird(2).
b1 :- bird(3).
:- b1, not fb1 .
b2 :- bird(1), bird(2).
b2 :- bird(1), bird(3).
b2 :- bird(2), bird(3).
:- b2, not fb2 .
b3 :- bird (1), bird (2), bird (3).
:- b3, not fb2.
fb1 :- fly(1).
fb1 :- fly(2).
fb1 :- fly(3).
fb2 :- fly(1), fly(2).
fb2 :- fly(1), fly(3).
fb2 :- fly(2), fly(3).
```

In the remaining part of the paper we refer to "*PASTA programs*" by programs comprising a set of probabilistic facts as well as a set of disjunctive rules and constraints with counting aggregates.

2.3. Computational Complexity

We assume familiarity with computational complexity and use complexity classes as usual, see, e.g., [20, 21]. In particular, we use the complexity class PP for *probabilistic polynomial-time problems* [22] and we rely on oracle notations as usual. A complete canonical problem for PP is to decide whether a propositional formula has at least $q \in \mathbb{N}$ satisfying assignments. More formally, the goal is to evaluate whether the formula $\psi = \#_{\geq q} X.\varphi(X)$ evaluates to true, where $\varphi(X)$ is a 3-CNF formula over variables in X. This is the case if there are at least q satisfying assignments of φ. Analogously, evaluating formulas of the form $\psi = \#_{\geq q} X.\forall Y.\varphi(X, Y)$ with φ being a Boolean formula in 3-DNF over variables in $X \cup Y$, is $\mathrm{PP}^{\mathrm{NP}}$-complete. Intuitively, this requires witnessing at least q assignments over X, where any extension of these assignments over variables Y, ensures that φ evaluates to true. For a detailed introduction over this formalism and a survey on complexity results for probabilistic reasoning, see [10].

3. The Complexity of PASTA Inference

We obtain the following complexity results, which underline the simplicity of inference for PASTA programs. As in [10], we focus here on the complexity of cautious reasoning, i.e., given a set of ground literals Q (query), a set of ground atoms E (evidence), and a real number γ, deciding whether $\underline{\mathrm{P}}(Q \mid E) \geq \gamma$. As also noted in [10], the computation of the upper probability reduces to the computation of the lower probability on the complement set of Q. To lighten the notation, we simply refer to the cautious reasoning task as inference task.

Theorem 1 (Complexity)**.** *Inference in PASTA programs under credal semantics is PP^{NP}-complete.*

Proof. Hardness: We reduce from $\#_{\geq q} X.\forall Y.\varphi(X,Y)$ with φ being a Boolean formula in 3-DNF over variables in $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. We construct a PASTA program Π, where for every variable x occurring in X, we add probabilistic facts

```
0.5::t(x).
```

We care about satisfied worlds, so we guess the state of satisfiability for every DNF term c

```
sat(c) ; nsat(c).     sat ; nsat.
```

We guess the truth assignments for every universally quantified variable $y \in Y$:

```
t(y) ; f(y).
```

For every DNF term c, every positive variable v_i in c and every negative occurrence $\neg v_j$ add:

```
pos(c, v_i). neg(c, v_j).
```

We have satisfiability if and only if at least one DNF term is satisfied:

```
:- #count {C : sat(C)}=T, #count {1: sat}=S, T<S.
:- #count {C : nsat(C)}=T, #count {1: nsat}=S, S*|φ| >T.
```

For the satisfied DNF terms c we add the following constraint:

```
:- #count {V : t(V), pos(c,V); V : not t(V), neg(c,V)}=T,
   #count {1 : sat(c)}=S, T<3S.
```

For the unsatisfied DNF terms c we do the following

```
:- #count {V : t(V), pos(c,V); V : not t(V), neg(c,V)}=T,
   #count {1 : nsat(c)}=S, T=3S.
```

It is easy to see that $\#_{\geq q}\{v_1, \ldots, v_n\}.\varphi$ holds, i.e., there are at least q satisfying assignments of φ if and only if $\underline{\mathrm{P}}(\{sat\}) \geq \frac{q}{2^n}$.

Note that since φ is in 3-CNF, all constraint sizes are constant. Indeed, the aggregate body sizes are at most 3 and we could even slightly adapt the proof to only use ground rules (i.e., without first-order variables).

Membership: Follows directly from previous work [10, Table 1], which shows PP$^{\mathrm{NP}}$ membership for aggregates and default negation. Indeed, disjunction in PASTA programs can be directly translated to default negation [23], as by definition there can not be a cyclic dependency between disjunctive head atoms. □

Note that disjunction or, alternatively, default negation is crucial in PASTA programs. Indeed, if we did not allow disjunction, such programs can only capture co-NP decision complexity.

Theorem 2 (Complexity without disjunction)**.** *Inference in PASTA programs under credal semantics without disjunctions is co-NP-complete.*

Proof. Hardness: We reduce from $\forall X.\varphi(X)$, where $\varphi(X)$ is in 3-DNF over variables in $X = \{x_1, \ldots, x_n\}$. We construct a PASTA program Π, where for every variable x occurring in X, we add probabilistic facts

```
0.5::t(x).
```

For every 3-DNF term c we assume an arbitrary but fixed order among its literals $l_1 \wedge l_2 \wedge l_3$. Then, for the i-th variable v_i such that $l_i = v_i$ (positive occurrence in c) and for the j-th variable v_j such that $l_j = \neg v_j$, we add the facts:

```
pos_i(c, v_i). neg_j(c, v_j).  sat.
```

For every 3-DNF term c, we add the following constraint which counts all satisfied 3-DNF terms (going over all valid 2^3 cases):

```
:- #count {
    C: t(V1), t(V2), t(V3), pos1(c,V1), pos2(c,V2), pos3(c,V3);
    C: not t(V1), t(V2), t(V3), neg1(c,V1), pos2(c,V2), pos3(c,V3);
    C: t(V1), not t(V2), t(V3), pos1(c,V1), neg2(c,V2), pos3(c,V3);
    C: t(V1), t(V2), not t(V3), pos1(c,V1), pos2(c,V2), neg3(c,V3);
    C: not t(V1), not t(V2), t(V3), neg1(c,V1), neg2(c,V2), pos3(c,V3);
    C: not t(V1), t(V2), not t(V3), neg1(c,V1), pos2(c,V2), neg3(c,V3);
    C: t(V1), not t(V2), not t(V3), pos1(c,V1), neg2(c,V2), neg3(c,V3);
    C: not t(V1), not t(V2), not t(V3), neg1(c,V1), neg2(c,V2), neg3(c,V3)
}=T, #count {1 : sat}=S, T<S.
```

By the credal semantics, a query can only be answered positively, if there is no world without any answer set. Consequently, we have that $\forall X.\varphi(X)$ evaluates to true if and only if $\underline{\mathrm{P}}(\{sat\}) \geq 1$.

Membership: In order to answer a query $\underline{\mathrm{P}}(A \mid B) \geq p$ for a given PASTA program Π, we need to ensure that there is no world without an answer set. This is the co-problem of finding some world without an answer set. In order to find such a world, we just guess any subset of facts and check in polynomial time whether the guess fulfills all aggregate constraints. Observe that for the guess the query $\underline{\mathrm{P}}(A \mid B) \geq p$ can then be answered in polynomial time. Indeed, we do so by using the known relationship [10, Eqs (4)] $\underline{\mathrm{P}}(A \mid B) = \frac{\underline{\mathrm{P}}(A \cap B)}{\underline{\mathrm{P}}(A \cap B) + \overline{\mathrm{P}}(\overline{A} \cap B)}$. It is easy to see that thereby $\underline{\mathrm{P}}(X)$ can be computed by just multiplying probabilities of the literals in X. Consequently, we can answer $\underline{\mathrm{P}}(A \mid B) \geq p$ in polynomial time (without the need to go over the potentially exponential number of worlds individually). Analogously this works for answering queries of the form $\overline{\mathrm{P}}(A \mid B) \geq p$ instead of $\underline{\mathrm{P}}(A \mid B) \geq p$, as we have $\underline{\mathrm{P}}(A \mid B) = \overline{\mathrm{P}}(\overline{A} \mid B)$. □

4. Related Work

Different semantics have been proposed in Probabilistic Answer Set Programming. Here we focused on the credal semantics. However, there are alternatives such as LPMLN [24], P-log [18], smProbLog [25], and PrASP [26]. As discussed through the paper, the complexity of PASP under the credal semantics has been studied in depth in [6, 10, 11]. The inference task goes up several levels of the polynomial hierarchy depending on the allowed syntactic constructs such as disjunctive rules, negation (either stratified or non-stratified), and aggregates. However, this seems the price to pay to use a rich and expressive language. Statistical statements are also discussed in [27], where the author provides a semantics based on the cross entropy minimization. The approach considered in PASTA is different, since it does not focus on a single model but it considers multiple models and lower and upper probability bounds.

5. Conclusion

In this paper, we provided an initial study of the complexity of the statistical statements provided by the PASTA framework. These statements are represented with a probabilistic answer set program, interpreted under the credal semantics, by means of disjunctive rules and constraints with counting aggregates involved. Our results show that PASTA programs are quite rich, as we reach PP^{NP}-completeness. However, the hardness proof of this result in Theorem 1 relies on using disjunction (or, alternatively, cyclic default negation), so without this language feature we can only reach co-NP completeness, as shown in Theorem 2. As a future work we plan to extend the complexity analysis to also consider the structure of PASTA programs.

References

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

[2] 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.

[3] J. Vennekens, S. Verbaeten, M. Bruynooghe, Logic programs with annotated disjunctions, in: B. Demoen, V. Lifschitz (Eds.), 20th International Conference on Logic Programming (ICLP 2004), volume 3131 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2004, pp. 431–445. doi:10.1007/978-3-540-27775-0_30.

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

[5] 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.

[6] 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.

[7] 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.

[8] J. Y. Halpern, An analysis of first-order logics of probability, Artificial Intelligence 46 (1990) 311–350. doi:10.1016/0004-3702(90)90019-V.

[9] 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.

[10] D. D. Mauá, F. G. Cozman, Complexity results for probabilistic answer set programming, International Journal of Approximate Reasoning 118 (2020) 133–154. doi:10.1016/j.ijar.2019.12.003.

[11] 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.

[12] J. Gill, Computational complexity of probabilistic turing machines, SIAM Journal on Computing 6 (1977) 675–695. doi:10.1137/0206049.

[13] K. W. Wagner, The complexity of combinatorial problems with succinct input representation, Acta Inf. 23 (1986) 325–356. doi:10.1007/BF00289117.

[14] D. Azzolini, F. Riguzzi, Inference in probabilistic answer set programming under the credal semantics, in: R. Basili, D. Lembo, C. Limongelli, A. Orlandini (Eds.), AIxIA 2023 - Advances in

Artificial Intelligence, volume 14318 of *Lecture Notes in Artificial Intelligence*, Springer, Heidelberg, Germany, 2023, pp. 367–380. doi:10.1007/978-3-031-47546-7_25.

[15] M. Alviano, W. Faber, Aggregates in answer set programming, KI-Künstliche Intelligenz 32 (2018) 119–124. doi:10.1007/s13218-018-0545-9.

[16] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming., in: 5th International Conference and Symposium on Logic Programming (ICLP/SLP 1988), volume 88, MIT Press, USA, 1988, pp. 1070–1080.

[17] J. Lee, Y. Wang, Weighted rules under the stable model semantics, in: C. Baral, J. P. Delgrande, F. Wolter (Eds.), Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, 2016, pp. 145–154.

[18] 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.

[19] D. Azzolini, F. Riguzzi, Lifted inference for statistical statements in probabilistic answer set programming, International Journal of Approximate Reasoning 163 (2023) 109040. doi:10.1016/j.ijar.2023.109040.

[20] R. Fagin, Generalized First-Order Spectra and Polynomial-Time Recognizable Sets, in: 7th Symposia in Applied Mathematics (SIAM 1974), 1974.

[21] C. H. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[22] J. Gill, Computational Complexity of Probabilistic Turing Machines, SIAM J. Comput. 6 (1977) 675–695. URL: https://doi.org/10.1137/0206049. doi:10.1137/0206049.

[23] T. Eiter, G. Gottlob, Expressiveness of stable model semantics for disjunctive logic programs with functions, The Journal of Logic Programming 33 (1997) 167–178.

[24] J. Lee, Z. Yang, LPMLN, weak constraints, and P-log, in: S. Singh, S. Markovitch (Eds.), Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA, AAAI Press, 2017, pp. 1170–1177.

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

[26] 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.

[27] M. Jaeger, Probabilistic reasoning in terminological logics, in: J. Doyle, E. Sandewall, P. Torasso (Eds.), 4th International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1994, pp. 305–316. doi:10.1016/B978-1-4832-1452-8.50124-X.