International Journal of Approximate Reasoning 163 (2023) 109040

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Lifted inference for statistical statements in probabilistic answer set programming

Damiano Azzolini [a,*], Fabrizio Riguzzi [b]

[a] Dipartimento di Scienze dell'Ambiente e della Prevenzione, Università di Ferrara, Italy
[b] Dipartimento di Matematica e Informatica, Università di Ferrara, Italy

ARTICLE INFO

Dataset link: https://github.com/damianoazzolini/pasta

Keywords:
Statistical statements
Probabilistic answer set programming
Lifted inference

ABSTRACT

In 1990, Halpern proposed the distinction between Type 1 and Type 2 statements: the former express statistical information about a domain of interest while the latter define a degree of belief. An example of Type 1 statement is "30% of the elements of a domain share the same property" while an example of Type 2 statement is "the element x has the property y with probability p". Recently, Type 1 statements were given an interpretation in terms of probabilistic answer set programs under the credal semantics in the PASTA framework. The algorithm proposed for inference requires the enumeration of all the answer sets of a given program, and so it is impractical for domains of not trivial size. The field of lifted inference aims to identify programs where inference can be computed without grounding the program. In this paper, we identify some classes of PASTA programs for which we apply lifted inference and develop compact formulas to compute the probability bounds of a query without the need to generate all the possible answer sets.

1. Introduction

Answer Set Programming (ASP) [1] is an expressive formalism to represent complex combinatorial problems and it is based on the concept of answer set (or, equivalently, stable model). However, like most logic languages, it cannot represent uncertain data. During the last few years, several proposals have been put forward to overcome this limitation: some of them associate a weight to rules [2] while others attach a probability to facts [3]. In particular, [3] follows the Distribution Semantics (DS) [4–6]. This semantics only allows the representation of what Halpern calls Type 2 statements [7], or statements about a degree of belief, such as "the probability that a particular element of the domain has the property y is x". These are often represented with probabilistic facts [8]. With the DS, it is not possible to handle Halpern's Type 1 statements or expressions of statistical information such as "x% of a given population have the property y".

Recently, the authors of [9] proposed the PASTA language, which allows the representation of Type 1 statements (called probabilistic conditionals) with ASP constructs (disjunctive rules and aggregates) in Probabilistic Answer Set Programming under the credal semantics (PASP) [10], that associates a probability range to a query (a conjunction of ground atoms). Furthermore, they proposed an algorithm, also called PASTA, to compute in an exact way the probability bounds of a query which is based on the

* Corresponding author.
E-mail addresses: damiano.azzolini@unife.it (D. Azzolini), fabrizio.riguzzi@unife.it (F. Riguzzi).

https://doi.org/10.1016/j.ijar.2023.109040
Received 28 January 2023; Received in revised form 15 September 2023; Accepted 22 September 2023

Available online 6 October 2023

enumeration of the possible worlds. This algorithm does not scale with the domain size, since the number of worlds to generate is exponential in the number of probabilistic facts.

The field of lifted inference [11,12] aims to identify some classes of programs where the inference task can be executed on a lifted level, i.e., without grounding all the variables of the program. In this paper, we identify some classes of PASTA programs with a single probabilistic conditional and develop lifted formulas to compute the probability for queries in these. Our approach is based on the identification of the worlds that contribute to the same probability bounds, that we call *indistinguishable*, by analyzing the results of the aggregates.

The paper is structured as follows: in Section 2 we provide some background knowledge. Section 3 introduces some formulas to perform lifted inference on PASTA programs that are empirically evaluated in Section 4. In Section 5 we survey related works and Section 6 concludes the paper.

2. Background

In this section, we introduce some background knowledge about the syntax and semantics of probabilistic logic programs and lifted inference.

2.1. Syntax

We recall here the basic concepts of Logic Programming [13] and Answer Set Programming (ASP) [1]. A rule is of the form:

$$h_1; \dots; h_n \ :- \ b_1, \dots, b_m.,$$

where $h_1; \dots; h_n$ is a disjunction of atoms called *head* while $b_1, \dots, b_m$ is a conjunction of literals is called *body*. If $n > 1$, the rule is called *disjunctive*, if $n = 0$, *constraint*, and if $m = 0$ and $n = 1$, *fact*. A rule is *ground* if it does not contain variables. We allow the b_js to be *aggregates* [14,15]. Their syntax is $\#f\{e_1, \dots, e_l\} \ c_1 g_1$, where f is an aggregate function symbol, c_1 is an arithmetic comparison operator, and g_1 is a constant or variable called guard. Here we consider only the $\#count$ aggregate with the syntax $\#count\{V : a(V)\} = N$ or $\#count\{V : C(V), A(V)\} = N$, where V is a vector of variables.

The authors of [9] propose the PASTA language, where Type 1 statements (also called probabilistic conditionals) can be defined using the syntax:

$$(C|A)[lb, ub]$$

where C is an atom called *consequent*, A is a conjunction of literals called *antecedent*, and $lb, \ ub \in [0, 1]$, $lb \leq ub$.[1] The meaning of the aforementioned formula is that the fraction of A's that are also C's is between lb and ub. These conditionals are converted into three answer set rules:

I) $C; not\_C \ :- \ A.$
II) $:- \ \#count\{X : C, A\} = V0, \#count\{X : A\} = V1, 10 \cdot V0 < 10 \cdot lb \cdot V1.$
III) $:- \ \#count\{X : C, A\} = V0, \#count\{X : A\} = V1, 10 \cdot V0 > 10 \cdot ub \cdot V1.$

where X is a vector of elements containing all the variables in C and A. The multiplication by 10 in II) and III) is due to the impossibility of using floating point numbers is ASP systems. Any other power of 10 can be chosen, but we stick here with 10 for ease of notation. Rule I) indicates that C can be true or false (represented by not_C) if A is true. This denotes the possibility that C may or may not (not_C) hold. The other two rules specify the fraction of elements that have the property C satisfied. Note that I) can be equivalently expressed with a choice rule of the form $\{C\} \ :- \ A$, but we maintain the syntax with the disjunction in the head to better identify whether the property C holds in the answer sets in the examples, and that the two aggregates in II) and III) can sometimes be merged into a unique aggregate to speed up the computations, but we keep them separated for clarity. II) and III) can be omitted from the program when $lb = 0$ and $ub = 1$, respectively. Note that, since II) and III) are ASP constraints, their body must be false, so the comparison operators are complemented.

The Distribution Semantics [6] (DS) allows the definition of probabilistic facts that can be expressed, using the ProbLog [8] syntax, with $\Pi :: f$ where f is an atom and $\Pi \in [0, 1]$. Following the ASP syntax, we use the notation $P :: f(l..u)$ to indicate a set of probabilistic facts $P :: f(l), P :: f(l+1), \dots, P :: f(u)$, where $l, u \in \mathbb{N}, \ l < u$. In this paper we consider only ground probabilistic facts.

2.2. Semantics

The *Herbrand base* of an answer set program $\mathcal{P}$ (denoted with $B_\mathcal{P}$) is the set of all ground atoms that can be constructed using the symbols in the program. If a set I is a subset of the Herbrand base, it is called an *interpretation*. Since we allow aggregates, we need to distinguish between *local* and *global* variables appearing in rules. The latter occur in at least one literal not involved in aggregations

[1] In this paper, the symbol | is used exclusively in the context of a conditional. Note that the same symbol can be found in the literature [16] used to represent disjunctive heads.

while the former are local to the aggregate they appear in [15]. We only consider local variables in aggregates. The grounding of an aggregate with only local variables can be obtained by substituting the local variables with ground terms in all possible ways. A ground atom a is true (false) in I if $a \in I$ ($a \notin I$). Similarly, an aggregate is true (false) in I if the evaluation of the aggregate function in I satisfies (does not satisfy) the guards. The grounding of a rule can be obtained by first substituting global variables with ground terms and then by substituting the local variables of every aggregate. An interpretation I *satisfies* a ground rule if at least one of the h_i is true in I when the conjunction of all the b_j is true in I. An interpretation satisfying all the groundings of all the rules of a program is called a *model*. From a ground program $\mathcal{P}_g$ and an interpretation I we can construct the *reduct* [14] of $\mathcal{P}_g$ with respect to I as the program obtained by removing from $\mathcal{P}_g$ the rules in which a b_j is false in I. An interpretation I is an *answer set* (also called *stable model*) for $\mathcal{P}_g$ if I is a minimal model (under set inclusion) of the reduct of $\mathcal{P}_g$. We use the notation $AS(\mathcal{P})$ to represent the set of all the answer sets of a program $\mathcal{P}$. We also consider cautious and brave consequences, i.e., the intersection and the union of all the stable models, respectively. In some circumstances, we may be interested in projecting the answer sets on a set of atoms [17]. That is, given a set of ground atoms V, the set of the projected solutions are represented by $AS_V(\mathcal{P}) = \{A \cap V \mid A \in AS(\mathcal{P})\}$.

All the probabilistic facts are considered independent. A *world* w is an answer set program obtained by including or not every probabilistic fact and its probability is given by

$$P(w) = \prod_{f_i^\top \in w} \Pi_i \cdot \prod_{f_i^\bot \in w} (1 - \Pi_i)$$

where $f_i^\top$ indicates that f_i is included in the world while $f_i^\bot$ indicates that f_i is excluded from the world. In traditional Probabilistic Logic Programming languages such as ProbLog [8] and Logic Programs with Annotated Disjunctions [18], each world is assumed to have a single stable model and the probability of a query q is given by the sum of the probabilities of the worlds where the query is true, i.e.,

$$P(q) = \sum_{w \models q} P(w).$$

Probabilistic Answer Set Programming under the credal semantics (PASP) [3] relaxes this and requires that every world has *at least one* stable model. With this semantics, a query q in PASP is associated with a lower $\underline{P}(q)$ and upper $\overline{P}(q)$ probability, computed as:

$$\overline{P}(q) = \sum_{w_i \mid \exists m \in AS(w_i),\ m \models q} P(w_i),\ \underline{P}(q) = \sum_{w_i \mid \forall m \in AS(w_i),\ m \models q} P(w_i).$$

That is, the upper probability of a query q, $\overline{P}(q)$, is the sum of the probabilities of the worlds where the query is true in at least one answer set, while the lower probability $\underline{P}(q)$, is the sum of the probabilities of the worlds where the query is true in every answer set. Note again that every world must have at least one answer set. This is needed since, if a world w has no answer sets, $P(w)$ neither contributes to the probability of the query nor to the probability of the negation of the query, causing a loss of probability, i.e., $\underline{P}(q) + \overline{P}(not\ q) < 1$ (using negation as failure). If two worlds have the same probability and contribute to the same probability bounds we call them *indistinguishable*. Overall, the size of the domain is fixed by the syntax of the probabilistic and other logical statements, but not by the syntax of the conditionals.

Example 1 *(Bird 4.).* Consider the following example taken from [9]:

```
0.4::bird(1..4).
(fly(X) | bird(X))[0.6,1].
```

The first line states that there are 4 individuals indexed with 1, ..., 4, each with probability 0.4 of being a bird. The conditional imposes that at least 60% of the birds fly (and at most 100%). Consider the query $q = fly(1)$. After translating the conditional into three answer set rules as described before, we get the following probabilistic answer set program under the credal semantics:

```
0.4::bird(1..4).
fly(X);not_fly(X) :-  bird(X).
:- #count{X: bird(X)} = H,
   #count{X: fly(X), bird(X)} = FH,
   100*FH < 60*H.
```

Rule III) is omitted since the upper bound is 1. Note again that the disjunctive rule can be equivalently expressed with the rule $\{fly(X)\}$:− $bird(X)$, and this should be preferred [19] since disjunctions may increase the computational complexity [20], but we stick in this example with the disjunctive rule for ease of explanation. This program has $2^4 = 16$ worlds, listed in Table 1. For example, the world where all the individuals are birds (w_0 in Table 1) has a probability of 0.4^4 and it has the following 5 stable models (obtained by including all the four probabilistic facts):

```
bird(1) bird(2) bird(3) bird(4) fly(1) fly(2) fly(3) fly(4)
bird(1) bird(2) bird(3) bird(4) fly(1) not_fly(2) fly(3) fly(4)
bird(1) bird(2) bird(3) bird(4) fly(1) fly(2) not_fly(3) fly(4)
bird(1) bird(2) bird(3) bird(4) fly(1) fly(2) fly(3) not_fly(4)
bird(1) bird(2) bird(3) bird(4) not_fly(1) fly(2) fly(3) fly(4)
```

Table 1
Worlds for Example 1. The column LP/UP indicates whether the considered world contributes to the lower (LP) or only to the upper (UP) probability. A dash in this column indicates that the world does not contribute to the probability bounds. $b(i)$, $i \in \{1,..,4\}$ stands for $bird(i)$.

id	$b(1)$	$b(2)$	$b(3)$	$b(4)$	LP/UP	Probability	id	$b(1)$	$b(2)$	$b(3)$	$b(4)$	LP/UP	Probability
w_0	1	1	1	1	UP	0.0256	w_8	0	1	1	1	-	0.0384
w_1	1	1	0	1	UP	0.0384	w_9	0	1	1	0	-	0.0576
w_2	1	1	0	0	LP, UP	0.0576	w_{10}	0	1	0	1	-	0.0576
w_3	1	1	1	0	UP	0.0384	w_{11}	0	1	0	0	-	0.0864
w_4	1	0	1	1	UP	0.0384	w_{12}	0	0	1	1	-	0.0576
w_5	1	0	1	0	LP, UP	0.0576	w_{13}	0	0	1	0	-	0.0864
w_6	1	0	0	1	LP, UP	0.0576	w_{14}	0	0	0	1	-	0.0864
w_7	1	0	0	0	LP, UP	0.0864	w_{15}	0	0	0	0	-	0.1296

The query is only true in the first four, so this world only contributes to the upper probability. By extending this consideration to all the worlds, we get $\underline{P}(q) = 0.2592$ and $\overline{P}(q) = 0.4$. Note that, even if we change the probability associated with the probabilistic facts, the number of answer sets, and thus the number of worlds, will be the same. Also, the worlds that contribute only to the upper or both to the lower and upper probability will be the same, but with a different contribution in terms of probability (due to the different probability associated with the probabilistic facts). For instance, if the $birds/1$ facts are probabilistic with an associated probability of 0.3, we have the same answer sets, the worlds with the same probabilistic facts true as before contribute to the same bounds, but the lower and upper probability for the same query are now respectively $\underline{P}(q) = 0.2352$ and $\overline{P}(q) = 0.3$.

The authors of [21] studied the complexity of three inference tasks in the context of probabilistic answer set programming: cautious reasoning, most probable explanation, and maximum a posteriori. In particular, cautious reasoning requires checking whether $\underline{P}(q \mid e_1, \ldots, e_n) \geq \gamma$, where q and all the e_i, $i \in [i, \ldots, n]$, are ground literals, and $\gamma \in [0, 1]$. The complexity of the three aforementioned tasks highly depends on the syntactical constructs allowed in programs, namely disjunction, negation, and aggregates: for cautious reasoning in bounded arity programs (i.e., non-ground programs were the arity of each predicate is bounded by a constant) with negation, disjunction in the head, and aggregates (sum, count, and max) it is $PP^{\Sigma_3^{PP}}$, where PP is composed by the languages that can be decided in a polynomial number of steps by a probabilistic Turing machine with an error less than 0.5 for all the input instances.

2.3. The PASTA algorithm

The inference algorithm for PASTA programs proposed in [9] works as follows: after translating the conditionals as previously described, every probabilistic fact $\Pi :: f$ is converted into a choice rule $\{f\}$. Then, the software enumerates all the answer sets projected [17] on the query and on the probabilistic facts. If we consider again Example 1 with query $fly(1)$, the probabilistic facts $0.4 :: bird(1..4)$ are translated into $\{bird(1..4)\}$ and the probability is stored internally. This is because the probability associated with probabilistic facts does not influence the generation of the answer sets. The whole converted program has 20 projected (on the probabilistic facts $bird/1$ and query $fly(1)$) answer sets in total (against the 32 total, non-projected, answer sets). The world w_0 in Table 1, where all the probabilistic facts are true, is identified by two answer sets

```
bird(1) bird(2) bird(3) bird(4) fly(1)
bird(1) bird(2) bird(3) bird(4)
```

The first states that there is at least one answer set with the query true while the second that there is at least one answer set with the query false in the considered world. So, this world only contributes to the upper probability. Differently, for the world w_2, with only $bird(1)$ and $bird(2)$ included, we have a single answer set

```
bird(1) bird(2) fly(1)
```

meaning that there is at least one answer set with the query true and no answer sets with the query false or, in other words, all the answer sets for this world have the query true. So this world contributes to both the lower and upper probability. Again, for the world w_8, we have

```
bird(2) bird(3) bird(4)
```

so no answer sets for this world have the query true, so it does not contribute to the probability bounds. All the projected answer sets are computed with a single call to the ASP solver and then analyzed one by one to reconstruct which world they represent. This approach cannot scale, since it requires the enumeration of 2^{n+1} answer sets in the worst case (every world has more than one answer set and the query is true in some of these and false in others) and 2^n in the best case (where every world has exactly one answer set), where n is the number of probabilistic facts. However, there are statistical statements that allow the computation of the probability of a query in a lifted way, i.e., without enumerating all the answer sets. Table 1 already provides some hints, since some of the worlds have the same probability and contribute to the same probability bounds, such as w_1, w_3, w_4 and w_2, w_5, and w_6. The goal of this paper is to propose a set of equations to compactly compute the probability of a query in PASTA programs. The authors of [9] only discussed an exact algorithm based on enumeration. However, as we will show later, some configurations of statistical statements do not need such an enumeration, but a closed formula suffices, making exact inference applicable to larger domains.

2.4. Lifted inference

The goal of lifted inference is to perform inference on a lifted level, i.e., to answer queries by considering the elements of a domain grouped according to certain criteria [11]. This allows to greatly speed up the computations, since individuals are not considered individually. Lifted inference approaches have been successfully applied in several scenarios with multiple techniques available [12,22–25]. Consider the following example of probabilistic logic program inspired from [22].

Example 2. The following probabilistic logic program has a non-ground probabilistic fact stating that an individual is sick ($sick/1$) with probability p or ill ($ill/1$) if she has a friend that is sick.

```
p :: sick(Y).
ill(X) :- friends(X, Y), sick(Y).
```

By considering non-lifted inference, the program of Example 2 should be grounded and then the probability of a query, say $ill(a)$, can be computed after encoding it into a compact form, such as Binary Decision Diagrams or Sentential Decision Diagrams [8,26]. As the size of the domain increases, inference becomes intractable. However, we can note that the probability of $ill(a)$ can be computed by knowing only the number k of friends of a, without grounding the program, with the formula $P(ill(a)) = 1 - (1 - p)^k$, since the body has an existentially quantified variable.

Lifted inference techniques have been applied to first-order languages. In the context of Probabilistic Logic Programming, the authors of [27] defined hierarchical probabilistic logic programs, where clauses are organized in a hierarchical way such that they can be represented with an arithmetic circuit, where inference is much cheaper. In [28], the authors introduced the classes of liftable PLP programs, a restriction of Probabilistic Logic Programming (under the distribution semantics) where inference can be performed at a lifted level. An overall survey can be found in [22]. The class of first order formulas restricted to two logical variables, denoted with FO^2 has been proved domain liftable [29], i.e., (weighted) model counting can be performed in a polynomial way with respect to the size of the domain. Despite all these approaches, to the best of our knowledge, none of them consider Probabilistic Answer Set Programming with aggregates, and therefore PASTA statements. In the following section, we identify some classes of PASTA programs and provide lifted formulas for the computation of the probability of a query.

3. Lifted inference for statistical statements

As usual in lifted inference, let us restrict the type of theories. We consider programs with only one conditional. Note that, in general, it is not always possible to simply group the worlds by probability and count their number, since worlds with the same probability may contribute differently to the lower and upper probability bounds. This is due to the presence of constraints with aggregates. We suppose that the probabilistic facts have the predicates $a/1$, $b/1$, and $b/2$, and their arguments are increasing integer numbers, such as $a(1)$, $a(2)$, $b(1)$, $b(1,1)$, and so on. Most of the following formulas involve the upper probability, but these can be straightforwardly extended to the lower probability, since the worlds to consider are the same, the only difference is in the contribution. We also use the terms random variable and probabilistic fact interchangeably. Furthermore, we suppose that every world has at least one stable model, as required by the credal semantics. This requirement already imposes some constraint on the structure of the conditionals, as stated in the following theorem.

Theorem 1. *For a program with a single conditional of the form* $(c(X)|a(X))[lb, ub]$ *where* $a/1$ *is defined by probabilistic facts, if* $lb > 0$ *and* $ub < 1$ *there are some worlds without answer sets, so the program fails to have a credal semantics.*

Proof of Theorem 1. The conversion of the conditional introduces two constraints: i) $:- 10 \cdot n_{ca} < lb \cdot n_a$ and ii) $:- 10 \cdot n_{ca} > ub \cdot n_a$ where n_{ca} and n_a are respectively the results of the aggregates $\#count\{X : c(X), a(X)\}$ and $\#count\{X : a(X)\}$. For a world w, constraint i) imposes that the value of n_{ca} cannot be 0 when the value of n_a is greater than 0. At the same time, ii) requires that n_{ca} cannot be equal to n_a. If we ignore the two constraints, every world including only one probabilistic fact $a(k)$ has only two answer sets: {a(k)} and {c(k) a(k)}. However, the first is removed by constraint i), since n_{ca} is 0, and the second by constraint ii), since here both n_{ca} and n_a are 1. Thus, both answer sets are removed, and the world has no answer sets, so it is unsatisfiable and the conditional fails to have a credal semantics. □

Moreover, the following theorem also holds:

Theorem 2. *For a program with a single conditional of the form* $(c(X)|a(X))[0, up]$ *where* $a/1$ *is defined by probabilistic facts and a ground query* $c(j)$, $\underline{P}(c(j)) = 0$.

Proof of Theorem 2. The conversion of the conditional includes into the answer set program a constraint that prevents the query being true in $100 \cdot up\%$ of the answer sets. Thus, the query can never be true in all the answer sets, thus none of the worlds contribute to the lower probability. □

3.1. Conditionals with one variable

Let us start with the simplest program:

```
p::a(l..u).
(c(X)|a(X))[lb,ub].
```

We are interested in computing the lower and upper probabilities for a query $c(j)$, $j \in [l,u]$. We can identify two possible scenarios: the first, where $p = 0.5$, and the second, with $p \neq 0.5$. After discussing these two cases, we expand the discussion to the case where there are different clusters of probabilities, that is, groups of probabilistic facts with the same probability. Let us denote with n the number of probabilistic facts, i.e., $n = u - l + 1$.

If all the random variables have probability 0.5, the probability of every world is 0.5^n, since, for each random variable x, $P(x) = 1 - P(x) = 0.5$. In this case, we do not need to compute the probability and the answer sets for every world: we can consider only one world for each number of random variables true and compute the answer sets for it. Furthermore, we just need to consider $2^n/2$ worlds, since only $2^n/2$ worlds have the corresponding $a(j)$ random variable set to true. There are $\binom{n-1}{k}$ worlds with k a's true (we consider $n-1$ since we already fixed $a(j)$), so

$$\overline{\mathrm{P}}(q) = 0.5^n \left(\sum_{i=0}^{n-1} \binom{n-1}{i} \cdot \overline{\delta}_{i+1}(q) \right) \tag{1}$$

where

$$\overline{\delta}_i(q) = \begin{cases} 1 & \text{if q is true in at least one answer set for the world} \\ & \text{with } i \text{ probabilistic facts true} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The equation for the lower probability is analogous, but with $\overline{\delta}_i(q)$ replaced by $\underline{\delta}_i(q)$ which is 1 if q is true in every answer set for the world with i probabilistic facts true, 0 otherwise. Overall, with this formula, with n random variables we need to compute $n-1$ worlds and, for each world, the brave and cautious consequences for the upper and lower probability respectively. If we consider Example 1 where all the probabilities are set to 0.5, with $q = fly(1)$ and $n = 4$, Equation (1) requires computing

$$\overline{\mathrm{P}}(q) = 0.5^4 \cdot (1 \cdot \overline{\delta}_1(q) + 3 \cdot \overline{\delta}_2(q) + 3 \cdot \overline{\delta}_3(q) + 1 \cdot \overline{\delta}_4(q))$$

where all the $\overline{\delta}_i(q) = 1$, $i \in 1, \ldots, 4$, resulting in 0.5. For the lower probability, only $\underline{\delta}_1(q)$ and $\underline{\delta}_2(q)$ are 1, resulting in $\underline{\mathrm{P}}(q) = 0.25$.

Consider now the case where the probability of the probabilistic facts is not 0.5, as in Example 1. Here, for each world, we may get a different probability value, so Equation (1) does not hold, but we can easily generalize it with:

$$\overline{\mathrm{P}}(q) = \sum_{i=0}^{n-1} \binom{n-1}{i} \cdot p^{i+1} \cdot (1-p)^{n-1-i} \cdot \overline{\delta}_{i+1}(q). \tag{3}$$

For the upper probability of the query $fly(1)$ of Example 1 we get:

$$1 \cdot 0.4^1 \cdot 0.6^3 \cdot \overline{\delta}_1(q) + 3 \cdot 0.4^2 \cdot 0.6^2 \cdot \overline{\delta}_2(q) + 3 \cdot 0.4^3 \cdot 0.6^1 \cdot \overline{\delta}_3(q) + 1 \cdot 0.4^4 \cdot 0.6^0 \cdot \overline{\delta}_4(q)$$

resulting in 0.4. For the lower probability we get 0.2592 since the worlds with 3 and 4 birds do not contribute. Equation (3) derives from the following theorem.

Theorem 3. *Let P be a program with a single conditional of the form $(c(X)|a(X))[lb,ub]$ and a set of probabilistic facts for $a/1$ with an associated probability p. For a ground query $c(j)$, all the worlds that include the probabilistic fact $a(j)$ and have the same number of probabilistic facts true are indistinguishable.*

Proof of Theorem 3. Consider two worlds, w_a and w_b with the same number n of probabilistic facts true. Consider the constraint with the aggregates $\#count\{X : a(X)\}$ and $\#count\{X : c(X), a(X)\}$. The result of the first aggregate is fixed for the two worlds, since the number of probabilistic facts is the same and fixed. The second counts the number of $c(l)$ with the corresponding $a(l)$ true. There can be one or zero $c(l)$ for every $a(l)$, so at most n c's in an answer set and at most 2^n possible answer sets for both worlds. Some of these violate the constraint with the lb and ub bounds and are discarded. lb and ub are the same for both worlds, so the remaining number of answer sets for every number of c's is the same. Thus, the two aggregates have the same value in the two worlds and the number of answer sets for every possible number of c's is the same for w_a and w_b so they contribute to the same probability bounds with the same value and thus they are indistinguishable. □

In both cases, with the algorithm of [9] we need to compute at least 2^n answer sets (possibly 2^{n-1} by considering only the ones with $a(j)$ for the query $c(j)$) with a single call to the ASP solver, while with the lifted algorithm we need to call the ASP solver $2 \cdot (n-1)$ times to compute both brave and cautious consequences. Thus, a linear number of answer sets instead of exponential. In Section 4 we will empirically assess this improvement by running some experiments.

Let us focus now on a more interesting scenario, where there are 2 or more probability clusters (i.e., some random variables have the same probability). In the previous case, there was only one probability cluster. In the worst case, there will be C probability clusters, one for each random variable (all with different probabilities), so the approach will eventually require the computation of all the worlds, with an exponential complexity in the number of random variables. Consider an extension of the program shown before, where there are two clusters with probability $p0$ and $p1$:

```
p0::a(l0..u0).
p1::a(l1..u1).
(c(X)|a(X))[lb,ub].
```

Example 3. We can modify Example 1 and consider 5 individuals and 2 probability clusters, 0.2, with 3 variables, and 0.3, with 2 variables:

```
0.2::bird(1..3).
0.3::bird(4..5).
```

Consider the query $q = fly(1)$. We can already halve the number of worlds to consider and remove $bird(1)$ from the first cluster, as previously discussed. We can describe the worlds using the product of different binomial coefficients. For the world with 5 birds (all bird facts true), one is fixed and must be true ($bird(1)$). For the others, we need to select the birds for all the clusters. This can be represented with $\binom{2}{2} \cdot \binom{2}{2} = 1$. If we consider 4 birds, we can select two birds from the first cluster and one from the other, or vice versa, obtaining 4 possible combinations: $\binom{2}{2} \cdot \binom{2}{1} + \binom{2}{1} \cdot \binom{2}{2} = 4$. The remaining combinations are $\binom{2}{2} \cdot \binom{2}{0} + \binom{2}{1} \cdot \binom{2}{1} + \binom{2}{0} \cdot \binom{2}{2} = 6$ for 3 birds, $\binom{2}{1} \cdot \binom{2}{0} + \binom{2}{0} \cdot \binom{2}{1} = 4$ for 2 birds, and $\binom{2}{0} \cdot \binom{2}{0} = 1$ for 1 bird ($bird(1)$). In total, as expected, we consider $1+4+6+4+1 = 16 = 2^{5-1}$ worlds. Let us now call *representative product* the product of the binomials in every term of the sum. We do not need to compute the probabilities for all the worlds since some of them are indistinguishable (i.e., have the same probability), but we just need to consider, for every number of probabilistic facts selected, a number of worlds equal to the number of representative products. We have: 1 world with 4 birds + 1 (we need to compute its probability), 4 worlds with 3 birds + 1 (we need to compute the probability only for two worlds), 6 worlds with 2 birds + 1 (we need to compute the probability of only three worlds), 4 worlds with 1 bird + 1 (we need to compute the probability only for two worlds), and 1 world with 0 birds + 1 (we need to compute its probability). Overall, we calculate $1 + 2 + 3 + 2 + 1 = 9$ (instead of $2^4 = 16$) worlds and their respective cautious and brave consequences.

The problem now translates to the generation of the k_is for binomials $\binom{n_i}{k_i}$ in every representative product, where n_i is the number of random variables in the cluster with probability p_i. Recall from combinatorics the following definition [30]:

Definition 1 *(Composition).* A composition is a way of writing an integer n as a sum of m integers > 0 where the order of the addends is significant. If the numbers are ≥ 0 (i.e., 0 is admitted), it is called *weak* composition. To count the number of weak compositions we need to bound the number of possible terms. For example, if $n = 3$, its compositions are 3, $2+1$, $1+1+1$, $1+2$, while some of its weak compositions of size 3 are $3+0+0$, $2+1+0$, ...

Weak compositions of length m (the number of clusters) are needed to generate the k_is for the binomials. Among all the weak compositions, we need to select the ones that are compatible with the number of random variables in the clusters. We indicate the weak compositions with the tuple $(k_1, \ldots, k_m)$, where $k_l \geq 0$ and $\sum_l k_l = i$ (where i is the number of random variables currently considered) and each k_l represents the number of elements selected for the cluster l. If we have 3 random variables and 2 probability clusters, one with 1 random variable and the other with 2, the weak composition $(3, 0)$ cannot be considered since the first cluster has 2 elements, while the weak composition $(2, 1)$ can. We call the weak compositions where $\forall l,\ k_l \leq n_l$, *admissible*, where n_l is the number of variables in the cluster l. Thus, we are interested only in admissible weak compositions.

If we consider again the *bird* example, with 10 total birds with all the same probability and query $fly(1)$, there are $\binom{9}{5} = 126$ worlds with 6 birds but we just need to consider one. If instead we have two probability clusters with 5 birds each (suppose that $bird(1)$ involved in the query is in the first cluster) we still have

$$\binom{4}{0} \cdot \binom{5}{5} + \binom{4}{1} \cdot \binom{5}{4} + \binom{4}{2} \cdot \binom{5}{3} + \binom{4}{3} \cdot \binom{5}{2} + \binom{4}{4} \cdot \binom{5}{1} = 126$$

worlds with 6 birds but 5 to consider (instead of 1), one for each representative product, and compute the cautious and brave consequences for these. Similarly for the other numbers of birds. Note that the n_ls for every representative product sum to the number of birds that can be chosen (9) and the k_ls sum to the number of birds considered (5 here). If n is the total number of random variables (note again that one is fixed to true as previously discussed), k_{il} is the l-th element of the weak composition representing the cluster l for the current number of variables i considered, and there are C clusters, we get:

$$\overline{\mathrm{P}}(q) = \underbrace{\sum_{i=0}^{n-1}}_{\mathrm{A}} \underbrace{\overline{\delta}_i(q)}_{\mathrm{B}} \cdot \underbrace{\sum_{\substack{k_{i1}+k_{i2}+\cdots+k_{iC}=i \\ k_{il} \geq 0 \\ k_{il} \leq n_l}}}_{\mathrm{C}} \underbrace{\prod_{l=1}^{C}}_{\mathrm{D}} \underbrace{\binom{n_l}{k_{il}}}_{\mathrm{E}} \cdot \underbrace{p_l^{k_{il}} \cdot (1-p_l)^{n_l - k_{il}}}_{\mathrm{F}} \cdot \underbrace{p_{\mathrm{fix}}}_{\mathrm{G}}. \tag{4}$$

If $C = 1$ (only one cluster) we obtain Equation (3). Equation (4) has many terms, let us analyze them one by one: A is the sum over all the number of random variables i; B is as in Equation (2); C is the sum over all the possible admissible weak compositions, where C is the (number of clusters); D is the product over all the clusters; E is the number of elements selected for the current cluster; F is the contribution to the probability of the query of the current cluster; G is the probability of the fixed probabilistic fact. The formula for the lower probability is analogous, but with $\overline{\delta}_i(q)$ replaced with $\underline{\delta}_i(q)$ discussed above. The correctness of Equation (4) is proven in the following theorem.

Theorem 4. *Let P be a program with a single conditional of the form $(c(X)|a(X))[lb, ub]$ and a set of probabilistic facts for $a/1$ with (possibly) different probabilities. Then, the probability of a query $c(j)$ can be computed with Equation (4) by considering only the worlds that include $a(j)$.*

Proof of Theorem 4. We prove the correctness of Equation (4) by induction. Consider the base case, where there is only 1 cluster, i.e., all the probabilistic facts have the same probability. In this case, we fallback to Equation (3), whose correctness is proved by means of Theorem 3. Consider now the case of $C + 1$ clusters and suppose that the formula holds for C clusters with n total probabilistic facts (of which one is fixed). That is, for C clusters, the formula considers the contribution of all the 2^{n-1} worlds. Suppose there are v probabilistic facts in the $(C + 1)$-th cluster with an associated probability of p_{C+1}. By introducing the new cluster we need to consider the contribution of 2^v more worlds. We need to multiply the formula for C clusters by $\sum_{r=0}^{v} \binom{v}{r} \cdot p_{C+1}^r \cdot (1 - p_{C+1})^{v-r}$ and consider $\overline{\delta}_{i+r}(q)$ instead of $\overline{\delta}_i(q)$. This is equivalent to extend the summation A up to $n - 1 + v$, the summation C over $k_{i1} + k_{i2} + \cdots + k_{iC} + k_{iC+1}$, and the product D up to $C + 1$, and we obtain again Equation (4). The worlds are grouped by the number of probabilistic facts, and we know from Theorem 3 that these contribute to the same probability bounds. □

Example 4 *(Computation of the probability of Example 3).* Let us apply Equation (4) to Example 3 with query $q = fly(1)$. For the upper probability we have $n = 5$, $n_1 = 2$, $n_2 = 2$, $C = 2$, $p_{\text{fix}} = p_1 = 0.2$, $p_2 = 0.3$. For the summation A we have five terms. D is always over 2 terms ($l = 1$ and $l = 2$). For example, for $i = 1$ there are two weak compositions and both are admissible, so C is over 2 terms: $(0, 1)$ and $(1, 0)$. For $i = 1$ and $l = 1$, C, D, E, F, and G results in:

$$\binom{2}{0}\binom{2}{1} \cdot p_1^0 \cdot (1 - p_1)^2 \cdot p_2^1 \cdot (1 - p_2)^1 \cdot p_{\text{fix}} + \binom{2}{1}\binom{2}{0} \cdot p_1^1 \cdot (1 - p_1)^1 \cdot p_2^0 \cdot (1 - p_2)^2 \cdot p_{\text{fix}}$$

which is $2 \cdot 0.8^2 \cdot 0.3 \cdot 0.7 \cdot 0.2 + 2 \cdot 0.2 \cdot 0.8 \cdot 0.7^2 \cdot 0.2 = 0.08512$. Overall, we get the following summations for i from 0 to 4: $0.0627 + 0.08512 + 0.04232 + 0.00912 + 0.00072 = 0.2$, which is the value of the upper probability since $\overline{\delta}_i(q) = 1$ for $i \in \{0, \ldots, 4\}$. For the lower probability we only have the first two terms of the sum, so $0.0627 + 0.08512 = 0.14782$. Note that for $i = 3$, there are 4 weak compositions but only $(2, 1)$ and $(1, 2)$ are admissible, so the sum is only over 2 elements instead of 4. Similarly, with $i = 4$, where out of 5 possible weak compositions, only $(2, 2)$ is admissible.

The number of probabilistic facts and clusters is fixed. The variable part is the number of admissible weak compositions for the number of probabilistic facts selected. If all the clusters have the same number of elements, we can consider the following lemma [30]:

Lemma 1. *Let $w(n, j, k)$ be the number of weak compositions of n into k parts, where each part must be less than j. Then,*

$$w(n, j, k) = \sum_{r+sj=n} (-1)^s \binom{k + r - 1}{r} \binom{k}{s}$$

for every pair $(r, s) \in \mathbb{N}^2$ such that $r + sj = n$.

So, in this case, the sum in C of Equation (4) will be over $w(n, j, k)$ terms.

3.2. Variable in the antecedent not appearing in the consequent

Consider conditionals of the form

```
(c(X)|a(X),b(Y))[lb,ub].
```

where $a/1$ and $b/1$ are defined by probabilistic facts with the same probability. We would like to remove the dependency of this conditional from $b/1$, since the variable Y in $b(Y)$ does not appear elsewhere. Thus, the goal is to obtain a conditional

```
(c1(X)|a(X))[l1,u1].
```

such that $\overline{P}(c1(j)) = \overline{P}(c(j))$ and $\underline{P}(c1(j)) = \underline{P}(c(j))$, since the latter is liftable (Equation (3)). Let us start from the constraints. We note that $\#count\{X : a(X)\}$ counts the number of a's, while $\#count\{X, Y : a(X), b(Y)\}$ counts the pairs of a's and b's. If n_a is the number of a's and n_b is the number of b's, the former aggregate results in n_a while the latter in $n_a \cdot n_b$. Similarly for $\#count\{X : c1(X), a(X)\}$ and $\#count\{X, Y : c1(X), a(X), b(Y)\}$, where the former counts the number n_{c1} of $c1$'s (since $c1(i)$ is true if $a(i)$ is true, but not necessarily vice versa) while the latter the product of the count n_{c1} of $c1$'s and the count of b's. Thus, if we consider the lower bound of the

original conditional we have $:- 10 \cdot n_a \cdot n_b < lb \cdot n_c \cdot n_b$. We can simplify and remove the dependency on $b/1$ (by supposing that n_b is greater than 0) and we obtain $:- 10 \cdot n_a < lb \cdot n_c$, which is exactly the constraint for the lower bound for $(c1(X)|a(X))[l1, u1]$. Thus, $l1 = lb$ and $u1 = ub$. Consider now the computation of the upper probability. The first thing to note is that the conditional with both $a/1$ and $b/1$ requires that at least one of the $b/1$ is true, but the combinations of the $a/1$ that make $c/1$ true are the same as the conditional without $b/1$. Thus, $\overline{\mathrm{P}}(c1(j)) > \overline{\mathrm{P}}(c(j))$. To compute the difference of the two probabilities we need to subtract from $\overline{\mathrm{P}}(c1(j))$ the probability for all the worlds where all the $b/1$ are false in the initial conditional. Call $k(a, b)$ the set containing all the worlds obtained by selecting the a's in all possible ways while all the b's are false. We have:

$$\overline{\mathrm{P}}(c(j)) = \overline{\mathrm{P}}(c1(j)) - \sum_{w \in k(a,b)} P(w).$$

$k(a, b)$ contains an exponential (2^n) number of elements. However, we can avoid computing all of these, since they can be grouped in clusters where each cluster contains a fixed number of a's true (as previously discussed). Call $p_k = \prod_l 1 - \overline{\mathrm{P}}(b(l))$. We need to enumerate all the worlds and compute their contribution to the probability for the query $c(j)$. We can leverage Equation (3) to avoid this enumeration and compute $\overline{\mathrm{P}}(c1(j))$. Lastly, $\overline{\mathrm{P}}(c1(j))$ must be multiplied by p_k, to account for the fact that all the $b(i)$ are false in the original conditional. Overall, we get:

$$\overline{\mathrm{P}}(c(j)) = \overline{\mathrm{P}}(c1(j)) - \overline{\mathrm{P}}(c1(j)) \cdot p_k = \overline{\mathrm{P}}(c1(j)) \cdot (1 - p_k).$$

Thus, the dependency from the $b/1$ can be removed. The same considerations hold for the lower probability and can be as well extended to consider multiple probability clusters, as in Equation (4).

To clarify, consider the following program:

```
0.4::a(1..3).
0.4::b(1..3).
(c(X)|a(X),b(Y))[0.4,1].
```

with query $c(1)$, that has probability in the range $[0.112896, 0.3136]$. If we remove the dependency from $b/1$, we get the range $[0.144, 0.4]$. If we consider only the a's, there are $2^3 = 8$ worlds. Half of them (4) can be ignored since $a(1)$ is false. The remaining have probability 0.064 (one world with 3 a's true), 0.096 (2 worlds with 2 a's true), and 0.144 (1 world with a single a true). Then, $p_k = (1 - 0.4)^3$. $\underline{\mathrm{P}}(c(1)) = 0.144 \cdot (1 - p_k) = 0.112896$ which is exactly the value of the lower probability. For the upper probability, $\overline{\mathrm{P}}(c(1)) = 0.4 \cdot (1 - p_k) = 0.3136$.

3.3. A binary predicate in the antecedent

Consider now conditionals of the form

```
(c(X)|a(X),b(X,Y))[lb,ub].
```

with the query $c(i)$. We have two possible cases: there is exactly one $b(j, \_)$ for every $a(j)$ or there are multiple $b(j, \_)$ for a given $a(j)$. Let us focus on the first case and suppose that all the probabilistic facts have the same probability. Note that if there are some $b(k, \_)$ such that $a(k)$ is not present in the program, these can be removed, and it is proven in the following theorem.

Theorem 5. *Let P be a program with a single conditional of the form $(c(X)|a(X), b(X, Y))[lb, ub]$ and a set of probabilistic facts for $a/1$ and $b/2$. If for a given $b(k, \_)$ there is not a corresponding $a(k)$ or for a given $a(k)$ there is not at least one $b(k, \_)$, then, for the former case, all the probabilistic facts of the form $b(k, \_)$, and for the latter case all the probabilistic facts $a(k)$, can be ignored and removed from the program.*

Proof of Theorem 5. After applying the translation for conditionals described in Section 2, we get the aggregates $C_1 = \#count\{X, Y : a(X), b(X, Y)\}$ and $C_2 = \#count\{X, Y : c(X), a(X), b(X, Y)\}$. C_1 counts the tuples (X, Y) such that both $a(X)$ and $b(X, Y)$ are true. Thus, for (X, Y), if $a(X)$ is not paired with a $b(X, Y)$ (whether it is set to false in the current world or not present in the program) or vice versa, the tuple is not considered, so does not contribute to the count. Similarly for C_2, where we also have an additional atom to consider. Moreover, the antecedent $a(X), b(X, Y)$, after applying the translation for conditionals described in Section 2 is considered as the body of a clause with head $c(X)$. To make $c(X)$ true both $a(X)$ and $b(X, Y)$ must be true. If there are (is) no $b(X, Y)$ ($a(X)$) for a given $a(X)$ ($b(X, Y)$) in the program, there is not a contribution to neither C_1 nor C_2, since the two will be always unpaired, so the $b(X, Y)$ ($a(X)$) can be removed from the program. Note that, if for a particular world, $a(X)$ is true while $b(X, Y)$ is false, or vice versa, $a(X)/b(X, Y)$ cannot be removed since it is only set to false in the current world. □

Let us denote with $\delta_w(q)$ the contribution, either to the lower and the upper, only to the upper, or to none of the two bounds, of a world w for a query q. The first thing to note is that the worlds with the same probability are no more indistinguishable, i.e., they may contribute differently to the lower and upper probability, proven by the following theorem.

Theorem 6. *Let P be a program with a single conditional of the form $(c(X)|a(X), b(X, Y))[lb, ub]$ where the $a/1$ and $b/2$ are defined with probabilistic facts with the same probability. For all the pair of worlds w_i and w_j, $P(w_i) = P(w_j) \not\Longrightarrow \delta_{w_i}(q) = \delta_{w_j}(q)$.*

To see why Theorem 6 holds, consider Example 5.

Example 5. The following program has 6 probabilistic facts.

```
0.4::a(1..3).
0.4::b(1..3,1).
(c(X)|a(X),b(X,Y))[0.4,1].
```

Consider the query $c(1)$. The worlds $w_j = \{a(1), a(2), \textit{not } a(3), b(1,1), b(2,1), \textit{not } b(3,1)\}$ and $w_k = \{a(1), a(2), \textit{not } a(3), b(1,1), \textit{not } b(2,1), b(3,1)\}$ (where with *not* f we indicate that the probabilistic fact f is not included) have the same probability (0.009216) but the former contributes only to upper probability while the latter contributes to both the lower and upper probability.

Thus, we cannot simply group the worlds by the number of random variables true and count them. Let us order the probabilistic facts for $a/1$ and $b/2$ such that $a(j)$ is in position j and $b(j,\_)$ is in position $n/2+j$ (note that n is always even). We represent each world with a binary string where the bit at position j is true if the corresponding probabilistic fact is true, 0 otherwise. For example, the binary strings 110|110 and 110|101 are the representation of w_j and w_k of Example 5 respectively. Consider the query $c(1)$. Then, $a(1)$ and $b(1,1)$ must always be true, so we can ignore them in the binary string representation. So, $w_j = 10|10$ and $w_k = 10|01$. The aggregate $\#count\{X,Y : a(X), b(X,Y)\}$ counts the number of pairs (X,Y) such that $a(X)$ and $b(X,Y)$ are both true. For w_j and w_k, this aggregate results in 1 and 0 respectively. In conclusion, we can count the number of 1s in corresponding positions (i.e., j and $j+n/2$) and group the worlds by this number, since all give different contributions but to the same probability bounds. For example, w_j contributes to the same probability bounds as $w_l = 10|11$, but with a different probability value. Overall, if n is the number of probabilistic facts minus the two that must be fixed, we get (for the upper probability):

$$\overline{\mathrm{P}}(q) = p_k \sum_{i=0}^{n} \sum_{k=0}^{n/2} \overline{\delta}_{ik}(q) \cdot \rho(n,k,i) \cdot p^i \cdot (1-p)^{n-i} \tag{5}$$

where $p_k = P(a(1)) \cdot P(b(1,1))$ and with $\rho(n,k,i)$ we indicate the number of possible worlds such that the number of 1s in corresponding positions in a string of length $n/2$ of the representation of a world (we call this value overlap) equals k and there are i 1s in the string, and $\overline{\delta}_{ik}(q)$ is 1 if the query is true in at least one answer set for a world with i probabilistic facts true and k overlaps, 0 otherwise. For example, if $n_a = n_b = n/2 = 2$, $k = 0$, and $i = 1$, we have $\rho(4,0,1) = 4$ where the elements are $\{10|00, 01|00, 00|10, 00|01\}$. Note that here the length of a binary string representing a world is always even, since for each $a(j)$ there is exactly one $b(j,\_)$. The correctness of Equation (5) derives from the following theorem.

Theorem 7. *Let P be a program with a single conditional of the form $(c(X)|a(X), b(X,Y))[lb, ub]$ where the $a/1$ and $b/2$ are defined with probabilistic facts with the same probability and there is exactly 1 $b(i,\_)$ for every $a(i)$ and vice versa. Then, the worlds with the same number of overlaps contribute to the same probability bounds.*

Proof of Theorem 7. As for the other theorems, focus first on the aggregate $\#count\{X,Y : a(X), b(X,Y)\}$. By hypothesis, there is exactly 1 $b(i,\_)$ for every $a(i)$. Thus, the aggregate counts the pairs $(a(i), b(i,\_))$ true in every world. If there is an $a/1$ or $b/2$ unpaired, this does not contribute to the count. Similarly happens with the aggregate with also $c(X)$ in the conjunction. Since the constraint is the only element that possibly eliminates some of the answer sets for a given world, if two worlds have the same count for both of the aggregates, clearly the constraint is imposed with the same values, so the two worlds will contribute to the same probability bounds, but with a possibly different probability value (depending on the probabilities of the probabilistic facts). □

Let us discuss the value of $\rho(n,k,i)$ of Equation (5). We can see that if the number of possible 1s (i) is less than half of the overlaps (k), its value is 0. Otherwise, we can fix the 1s in the first half of the string and position the remaining. So, for each i, we can place from k to the minimum between $n/2$ and i 1s in the first half of the string and count all the possible ways to place the others in the remaining half. However, this is not correct, since we need to ignore the 1s that may be placed in the second half but increase the number of overlaps. For example, if $n = 6$, $k = 1$, $i = 4$, and the first half of the string is 110, there are $\binom{3}{2}$ possible ways to fix the remaining 1s in the second half: 110, 101, 011. However, 110 must be ignored, since there are 2 overlaps for this combination, and not 1 as required. If we consider the program shown in Example 5 with the query $c(1)$, once we fix $a(1)$ and $b(1,1)$, we have 9 worlds (1 with 2 facts true, 4 with 3 facts true, and 4 with 4 facts true) with 0 overlaps (without considering the $a(1)$ and $b(1,1)$) that contribute to the upper probability while the other 6 (2 with 4 facts true and 4 with 5 facts true) with 1 overlap and 1 with 2 overlaps (1 with 6 facts true) contribute to both the lower and upper probability. Overall, we get $P(c(j)) = [0.112896, 0.16]$. The number of overlaps can be computed with the formula:

$$\rho(n,k,i) = \sum_{n_a=k}^{min(n/2,i-k)} \underbrace{\binom{n/2}{n_a}}_{A} \cdot \underbrace{\binom{n_a}{k}}_{B} \cdot \underbrace{\binom{n/2-n_a}{i-n_a-k}}_{C} \tag{6}$$

where A is the number of ways to place n_a 1s in the first half of the string, which has length $n/2$, B is the number of ways to satisfy the number of overlaps, i.e., to place k 1s in the second half of the string to match the required number of overlaps with n_a 1s in the

first half of the string, and C is the number of ways to place in the remaining $n/2 - n_a$ positions of the second half of the string the remaining $i - n_a - k$ 1s. The value n_a ranges between k, the number of overlaps, and the minimum between $n/2$, half of the length of the string, and $i - k$, the difference between the total number of 1s and the required overlaps.

Finally, in the most general case where there are multiple $b(j,\_)$ for a given $a(j)$, we cannot simply examine the overlaps as defined before, since the variables are not necessary in an even number. A possible idea could consist in still creating a binary string with a length double than the number of $a/1$ and, for each $a(j)$, set the corresponding bit in the position $j + n$ to 0 if none of the $b(j,\_)$ are true, to 1 otherwise, and apply the same process as before. However, as discussed in Example 6, this yields a wrong result.

Example 6. Consider the query $c(1)$ asked in the following program:

```
0.3::a(1..2).
0.3::b(1..2,1..2).
(c(X)|a(X),b(X,Y))[0.4,1].
```

There are $2^{2+2\cdot 2} = 2^6 = 64$ possible worlds. The worlds, for example, $w_0 = \{a(1), a(2), b(1,1),$ *not* $b(1,2), b(2,1), b(2,2)\}$ and $w_1 = \{a(1), a(2), b(1,1), b(1,2), b(2,1),$ *not* $b(2,2)\}$ have the same number of probabilistic facts true (5), the same probability ($0.3^5 \cdot 0.7 = 0.001701$), and the same overlaps (3) as defined for Equation (5), but w_0 contributes to the upper probability while w_1 to both the lower and upper probability. This is because, if we ignore the constraint with the aggregates, for w_0, we get the following answer sets (we hide the probabilistic facts since they are set to be true): $\{cab(3)\ cabc(0)\}$, $\{cab(3)\ c(1)\ cabc(1)\}$, $\{cab(3)\ c(2)\ cabc(2)\}$, and $\{cab(3)\ c(1)\ c(2)\ cabc(3)\}$, where with $cab/1$ we indicate the result of the aggregate counting the pairs $(a(X), b(X,Y))$ while with $cabc/1$ the result of the aggregate counting the tuples $(c(X), a(X), b(X,Y))$. For w_1: $\{cab(3)\ cabc(0)\}$, $\{cab(3)\ c(2)\ cabc(1)\}$, $\{cab(3)\ c(1)\ cabc(2)\}$, and $\{cab(3)\ c(1)\ c(2)\ cabc(3)\}$. By considering the constraint, for w_0 only $\{cab(3)\ c(2)\ cabc(2)\}$ and $\{cab(3)\ c(1)\ c(2)\ cabc(3)\}$ remain: $c(1)$ is true in only one of the two, so we get a contribution only to the upper bound. For w_1 we have $\{cab(3)\ c(1)\ cabc(2)\}$ and $\{cab(3)\ c(1)\ c(2)\ cabc(3)\}$: $c(1)$ is true in both, so we have a contribution to both the probability bounds.

To properly define an equation to compute the probability of a query for these conditionals we need to consider, as in Equation (4), the possible weak compositions, but in this case the clusters are on the number of random variables $a(1), \ldots, a(n), b(1,\_), \ldots, b(n,\_)$, that we denote with increasing indexes, from 1 to C. Note that $n_m = 1$ for each $a(m)$ and C is twice the number of atoms for $a/1$ since there is at least one $b(i,\_)$ for every $a(i)$. For each index $1, \ldots, C$, we have $n_1, \ldots, n_C$ possible random variables. For example, in the program of Example 6, we can associate index 1 to $a(1)$ with $n_{a(1)} = 1$, 2 to $a(2)$ with $n_{a(2)} = 1$, 3 to $b(1,\_)$ with $n_{b(1,\_)} = 2$ (since we have $b(1,1)$ and $b(1,2)$), and 4 to $b(2,\_)$ with $n_{b(2,\_)} = 2$ ($b(2,1)$ and $b(2,2)$). However, this is not sufficient because, in addition to this, we also need to consider the overlaps and the number of corresponding $a(i)$ and $b(i,\_)$ for a query $c(i)$. Without loss of generality, for a query $q = c(1)$ we get:

$$\overline{\mathrm{P}}(q) = \sum_{i=2}^{n} \sum_{o=1}^{n/2} \sum_{b=1}^{n_{b(1,\_)}} \sum_{\substack{K_i \in wc(i,o,b) \\ K_i = (k_{i1}+k_{i2}+\cdots+k_{iC})}} \overline{\delta}_{K_i}(q) \cdot p^i \cdot (1-p)^{n-i} \cdot \prod_{l=1}^{C} \binom{n_l}{k_{il}}. \tag{7}$$

$wc(i,o,b)$ is the set of all the weak compositions $(k_{i1}, k_{i2}, \ldots, k_{iC})$ satisfying $k_{i1} + k_{i2} + \cdots + k_{iC} = i$ (number of probabilistic facts true), $k_{il} \geq 0$, $k_{il} \leq n_l$, $k_{i1}, k_{i(C/2)} > 0$ (this since we suppose the query is $c(1)$, thus the first element of the weak composition contains the number of $a(1)$ while the one at the half of the tuple is the number of $b(1,\_)$; in other words, this constraint imposes that $a(1)$ and at least one $b(1,\_)$ should be true), and with o overlaps and $n_{b(1,\_)} = b$, and $\overline{\delta}_{K_i}(q)$ is similar to the one of Equation (2) and is the result of asking the query on a program with k_{i1} random variables with index 1 set to true, k_{i2} random variables with index 2 set to true, and so on.

Theorem 8. *If there is 1 $b(j,\_)$ for every $a(j)$, Equation (7) is equivalent to Equation (5).*

Proof of Theorem 8. For a query $c(1)$, we can already fix $a(1)$ and $b(1,\_)$, and remove these variables from the summations, which contribute with a factor $p_k = p^2$ where p is the probability of the facts. We can now subtract 2 (fixed values for $a(1)$ and $b(1,\_)$) from n, and thus change the range of the first summation from 0 to n (instead of from 2 to $n-2$). Similarly, since we removed the two facts, the summation over the variable o is now between 0 and $n/2$, so the same as $\sum_{k=0}^{n/2}$ in Equation (5). Moreover, we do not need to sum over the number of $n_{b(1,\_)}$ since this value is always 1. We are left to prove $\sum_{K_i \in wc(i,o,1)} \prod_{l=1}^{C} \binom{n_l}{k_{il}} \stackrel{?}{=} \rho(n,k,i)$. All the n_l are 1 and k_{il} can be either 0 or 1 (by hypothesis, there is 1 $b(i,\_)$ for every $a(i)$). So, the binomials are either $\binom{1}{0}$ or $\binom{1}{1}$ and, in both cases, they equal 1, so we have a product of 1s that clearly results in 1. The summation is over all the possible overlaps, by definition, and counts 1 for each of them, which is exactly what $\rho(n,k,i)$ does, and thus also the deltas are over the same worlds. □

The following theorem proves the correctness of Equation (7).

Theorem 9. *Let P be a program with a single conditional of the form $(c(X)|a(X), b(X,Y))[lb, ub]$ where the $a/1$ and $b/2$ are defined by probabilistic facts with the same probability. Then, the probability of a query $c(j)$ can be computed with Equation (7).*

Proof of Theorem 9. To prove the correctness of Equation (7) we still need to consider the aggregates and their values. In Theorem 8, we proved that the overlaps identify indistinguishable worlds when there is a one-to-one correspondence between $a(i)$ and $b(i,\_)$. Now, this correspondence does not hold. Thus, we also need to account for the possible numbers of $b(j,\_)$ for every $a(j)$, since they influence the value of the aggregates. However, since we are interested in the probability of a query $c(i)$, we only need to know the number of $b(i,\_)$, since when the number of $b(i,\_)$ is fixed, the maximum number of $c(i)$ is fixed as well. The variation of the other $b(j,\_)$ with $j \neq i$ influences the result of the aggregates but does not influence the generation of the $c(i)$. So, we can further group the weak compositions by also the number of $b(i,\_)$, since these have the same number of $c(i)$ true and equally contribute to the probability. □

If the probabilistic facts also have different probabilities, Equation (7) can be combined with Equation (4), obtaining an even more complex equations that considers both the probabilities and the number of random variables. Let us apply Equation (7) to compute the probability of Example 6.

Example 7. Probability of Example 6. Let us rewrite here the program of Example 6.

```
0.3::a(1..2).
0.3::b(1..2,1..2).
(c(X)|a(X),b(X,Y))[0.4,1].
```

Suppose the query is $c(1)$. For $i = 2$, there is only one weak composition that satisfies all the constraints: (1, 0, 1, 0). This has 1 overlap and $n_{b(1,\_)} = 1$. There are $\binom{1}{1} \cdot \binom{1}{0} \cdot \binom{2}{1} \cdot \binom{2}{0} = 2$ possible worlds with this structure, $\{a(1), b(1,1)\}$ and $\{a(1), b(1,2)\}$, and both have the same associated probability $0.3^2 \cdot 0.7^4 = 0.021609$ and contribute to the lower and upper bound. So, we get, for the first iteration, $2 \cdot 0.021609 = 0.043218$. For $i = 3$, there are 3 admissible weak compositions: (1, 1, 1, 0), (1, 0, 1, 1), and (1, 0, 2, 0). For the first there are $\binom{1}{1} \cdot \binom{1}{1} \cdot \binom{2}{1} \cdot \binom{2}{0} = 2$ possible worlds with this structure, $\binom{1}{1} \cdot \binom{1}{0} \cdot \binom{2}{1} \cdot \binom{2}{1} = 4$ for the second and $\binom{1}{1} \cdot \binom{1}{0} \cdot \binom{2}{2} \cdot \binom{2}{0} = 1$ for the third. These $2 + 4 + 1 = 7$ worlds have probability $0.3^3 \cdot 0.7^3 = 0.009261$, so we get $7 \cdot 0.009261 = 0.064827$ and contribute to the lower and upper probability. The first two have the same number of $b(1,\_)$, so can be grouped together. For $i = 4$ there are 4 admissible weak compositions: (1, 1, 2, 0), (1, 1, 1, 1), (1, 0, 2, 1), and (1, 0, 1, 2). All the four are different, since all have different combinations of overlaps and $n_{b(1,\_)}$. In total, there are $1 + 4 + 2 + 2 = 9$ worlds with an associated probability of $0.3^4 \cdot 0.7^2 = 0.003969$. The first, third, and fourth contribute to the lower and upper probability, while the second only to the upper probability. For $i = 5$ there are 3 admissible weak compositions: (1, 1, 2, 1), (1, 1, 1, 2), and (1, 0, 2, 2). All the three are different, since all have different combinations of overlaps and $n_{b(1,\_)}$. We have $2 + 2 + 1 = 5$ worlds with probability 0.001701. The first and the third contribute to the lower and upper probability while the second only to the upper probability. For $i = 6$, there is only one weak composition that satisfies all the constraints: (1, 1, 2, 2). There is only one world with this structure and it contributes to the upper probability with $0.3^6 = 0.000729$. To sum up, $P(c(1)) = [0.132993, 0.153]$. Overall, there are 24 worlds that contribute to the probability, but we only considered 11 of these (and thus called the ASP solver 11 times instead of 24).

3.4. Two variables in the consequent and two facts with respectively one and two variables in the antecedent

Consider now this type of conditionals:

```
(c(X,Y)|a(X),b(X,Y))[lb,ub].
```

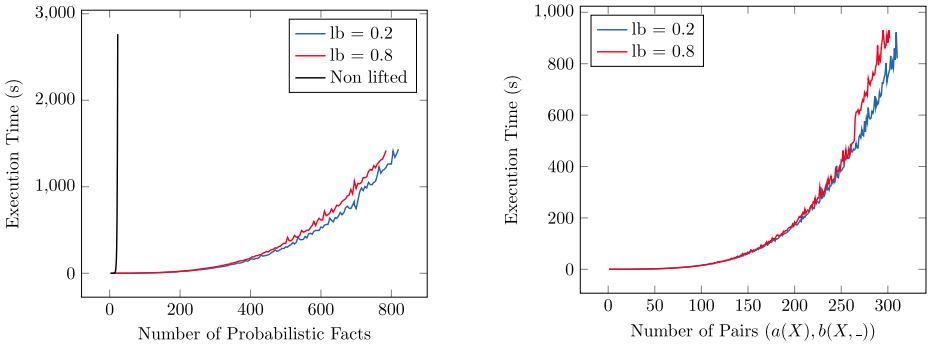
and query $c(i,j)$. As before, if there are $a(X)$ or $b(X,Y)$ unpaired, these can be removed from the program. Also here we can identify multiple cases. When there is exactly one $b(k,\_)$ for every $a(k)$, the probability of a query can be computed with Equation (5). This is because the result of the aggregates is always the same. When there are multiple $b(k,\_)$ for a given $a(k)$, we have two sub-cases: a single $b(i,j)$, $i \neq k$ and possibly multiple $b(k,\_)$, or multiple $b(i,\_)$ and multiple $b(k,\_)$. In the former, Equation (5) can still be applicable, since the $c(i,j)$ are in the same number as before. In the general case, Equation (5) cannot be directly applied, because we have an additional value for $c/2$ due to the variable (Y) to consider in the tuple $(c(X,Y), a(X), b(X,Y))$. For example, if we have a world with $a(1)$, $b(1,1)$, and $b(1,2)$, we have 2 answer sets for a conditional $c_x = (c(X)|a(X), b(X,Y))[0,1]$ ($\{\}$ and $\{c(1)\}$, by showing only $c/1$) while 4 for the conditional $c_{xy} = (c(X,Y)|a(X), b(X,Y))[0,1]$ ($\{\}$, $\{c(1,1)\}$, $\{c(1,2)\}$, and $\{c(1,1)\ c(1,2)\}$, by showing only $c/2$). We can already see that, depending on the value of lb and ub, Equation (5) is an upper bound for the probability of the query. For example, with $lb = 20\%$, the aforementioned world for c_x contributes to the lower and upper probability for $c(1)$ (the empty answer set is removed) while only to the upper probability for the query $c(1,1)$ in c_{xy} (also here the empty answer set is removed, but we still remain with three answer sets where the query is true only in two of them). In general, the following theorem holds:

Theorem 10. *If $c_x = (c(X)|a(X), b(X,Y))[lb, ub]$ and $c_{xy} = (c(X,Y)|a(X), b(X,Y))[lb, ub]$, then the lower and upper probability for the query $c(1)$ asked in a program with only c_x, where $a/1$ and $b/2$ are defined by probabilistic facts, are upper bounds for the lower and upper probability for the query $c(1,1)$ asked in a program with only c_{xy} and the same probabilistic facts for $a/1$ and $b/2$.*

Proof of Theorem 10. For c_{xy}, the choice rule $\{c(X,Y)\} :- a(X), b(X,Y)$ allows the generation of a $c(X,Y)$ for every pair $(a(X), b(X,Y))$. Similarly for c_x with the choice rule $\{c(X)\} :- a(X), b(X,Y)$, but here, since only X is present in $c/1$, multiple pairs $(a(X), b(X,Y))$ with the same value of X are considered only once. Thus, when there is exactly one $b(i,j)$ for $a(i)$ for the query

Table 2
Summary of the considered conditionals together with the corresponding theorems and equations to compute the probability of a query.

Conditional	Theorems	Equations
$(c(X)\|a(X))[lb,ub]$	1, 2, 3, 4	(1), (3), (4)
$(c(X)\|a(X),b(Y))[lb,ub]$		(3)
$(c(X)\|a(X),b(X,Y))[lb,ub]$	5, 6, 7, 8, 9	(5), (7)
$(c(X,Y)\|a(X),b(X,Y))[lb,ub]$	10	(7)



(a) $(c(X)|a(X))[lb,1]$. (b) $(c(X)|a(X),b(X,Y))[lb,1]$.

Fig. 1. Inference times for the conditional $(c(X)|a(X))[lb,1]$ with an increasing number of probabilistic facts $a/1$ (left) and for the conditional $(c(X)|a(X),b(X,Y))[lb,1]$ with an increasing number of pairs $(a(X),b(X,\_))$. For both, we set the probabilities of the probabilistic facts to 0.4, and lb to 0.2 and 0.8.

$c(i,j)$ the computed probability is the same as the one computed with c_x. Otherwise, for a given world in c_{xy} we will have multiple $c(i,k)$ and thus multiple answer sets that may not have $c(i,j)$ included, so possibly changing the contribution of the worlds. Moreover, when there are multiple $b(i,\_)$ for a given $a(i)$, to make the query $c(i)$ true, $a(i)$ and at least one $b(i,k)$ should be true. On the contrary, to make the query $c(i,j)$ true, we need to have both $a(i)$ and $c(i,j)$. □

In this case, we need to proceed as in Equation (7), by considering the different numbers of pairs and the different numbers of $b(i,\_)$ for a query $c(i,\_)$. The only modifications to consider are: fixing the value of $a(i)$ and $b(i,j)$ for a query $c(i,j)$ (so subtract 2 from n and start counting from 0 for the variable i) and, for the binomial coefficient, decrementing the value of $n_{C/2}$ and $k_{iC/2}$ by 1 if $n_{C/2}>0$ and $k_{iC/2}$ is greater than 1 and different from $n_{C/2}$. This is because, if we have, for example, 3 $b(1,\_)$, say $b(1,1)$, $b(1,2)$, and $b(1,3)$, when $k_{iC/2}$ is set to 2, we do not have $\binom{3}{2}=3$ possible combinations, but only $\binom{2}{1}=2$, since $b(1,1)$ must be fixed to true. Moreover, if $k_{iC/2}=1$ and $n_{C/2}>1$, we need to consider only 1 combination, the one with $a(i)$ and $b(i,j)$, and ignore the others.

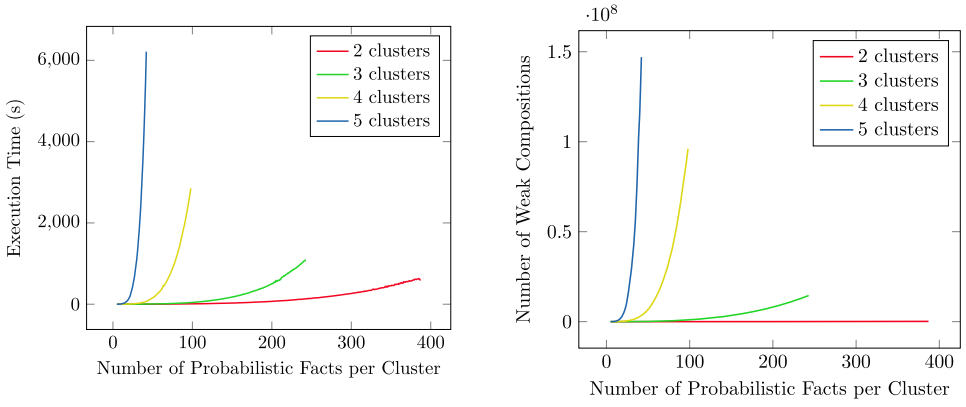
Overall, Table 2 summarizes our contribution in terms of liftable programs. Note that all the results of the previous theorems apply to programs with a single conditional. If we consider programs with multiple conditionals, some of the results of the previous theorems may not apply, due to the possible sharing of parts of the antecedent and/or consequent. We leave the further investigation of these programs as subject for future work.

4. Experiments

To better analyze the actual benefits in terms of execution time and calls to the ASP solver we conducted some experiments on a computer running at 2.40 GHz. The time limit is 8 hours. We implemented the lifted formulas in Python3 and included them in the freely available PASTA framework[2] that uses clingo [31] as underlying ASP solver. A comparison with other ASP solvers may be the subject of a potential future work. Moreover, since it is hard to represent a closed formula that counts the number of weak compositions, we naively generate all the possible compositions and then discard the ones that are not admissible. Thus, for the formulas requiring the computation of the weak compositions, the most significant value to consider is the number of these, not the execution time: a closed formula will clearly allow a faster execution time, but the generated number will always be the same.

In a first experiment, we considered the conditional $(c(X)|a(X))[lb,ub]$ (so Equation (3)) with an increasing number of probabilistic facts $a/1$ with an associated probability of 0.4. The probability value is not significant since it does not influence the execution time. We run two experiments, one with lb set to 0.2 and the other with lb set to 0.8 (ub is always set to 1). Results are shown in Fig. 1a. The execution time seems to grow exponentially, and this may be due to the requirements of the ASP solver to compute the models for programs of increasing size. The results by setting the lower bound to 0.2 and 0.8 are comparable in terms of execution time. We also plot the execution time obtained without lifted inference with the lower bound set to 0.2, to better assess the difference in

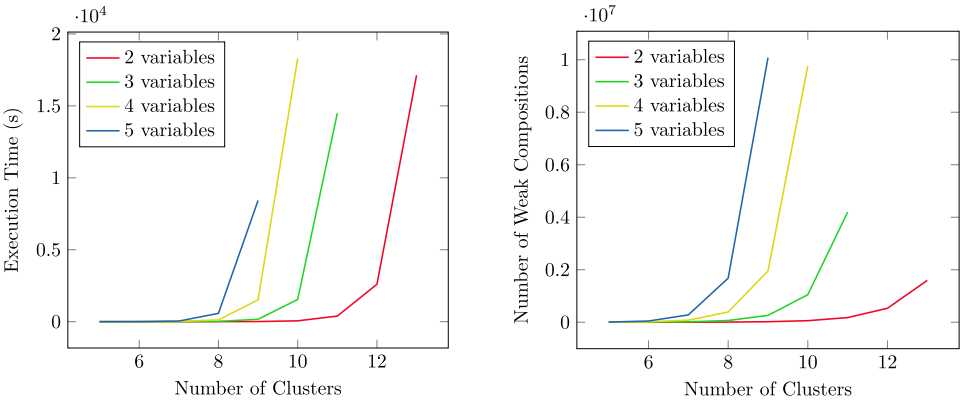
[2] https://github.com/damianoazzolini/pasta.



(a) Execution Time

(b) Number of weak compositions.

Fig. 2. Execution time and number of weak compositions for the conditional $(c(X)|a(X))[0.2, 1]$ by fixing the number of clusters and increasing the number of probabilistic facts per cluster.



(a) Execution Time

(b) Number of weak compositions.

Fig. 3. Execution time and number of weak compositions for the conditional $(c(X)|a(X))[0.2, 1]$ by fixing the number of probabilistic facts per cluster and increasing the number of clusters.

performance. We report the execution time for the non lifted algorithm (PASTA) only for this conditional, since the behavior is the same for all the other conditionals.

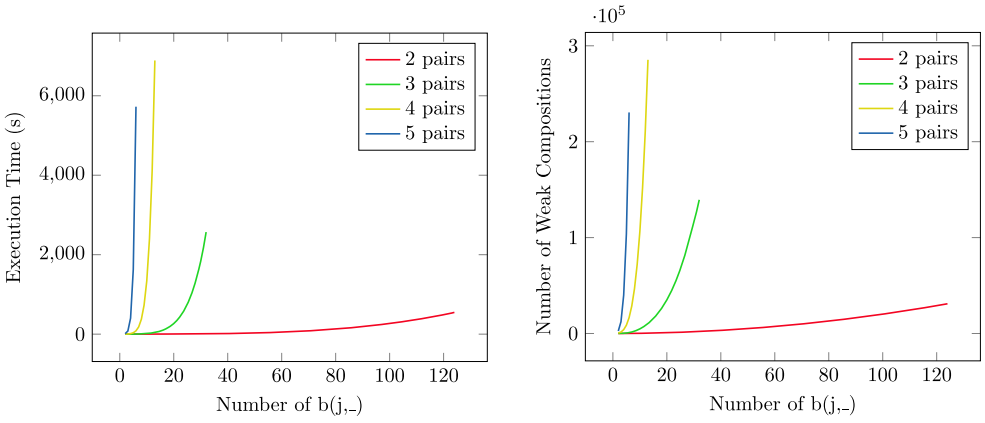
Then, we tested Equation (4), with the conditional $(c(X)|a(X))[0.2, 1]$ with an increasing number of probability clusters and probabilistic facts $a/1$. Fig. 2a shows how the execution time varies by fixing the number of clusters to 2, 3, 4, and 5 and increasing the probabilistic facts, while Fig. 2b shows the number of admissible weak compositions. Both the execution time and the number of weak compositions exponentially increase by increasing the clusters and the number of probabilistic facts. This is particularly evident for the cluster of size 5. Note that on the x axis of Fig. 2b we have the number of probabilistic facts per cluster: to obtain the total number of probabilistic facts, we need to multiply this value by the number of clusters. Fig. 3 shows both the execution time and the number of weak compositions by fixing the probabilistic facts to 2, 3, 4, and 5 for every cluster and increasing the number of clusters. Also here we get an exponential slope, due to the naive generation of all the weak compositions.

We also tested Equation (5) for the conditional $(c(X)|a(X), b(X, Y))[lb, 1]$ with an increasing number of pairs $(a(X), b(X, \_))$ and lb set to 0.2 and 0.8. We set the probabilities of the probabilistic facts to 0.4. Results are shown in Fig. 1b. The results are similar for the lower bounds set to 20% and 80%. As for Fig. 1a, the adoption of a lifted formula allows to perform inference with hundreds of pairs.

Finally, we tested again the conditional $(c(X)|a(X), b(X, Y))[0.2, 1]$ (Section 3.3) with more than one $b(X, \_)$ for every $a(X)$ (Equation (7)). Results are shown in Fig. 4 and Fig. 5. Here, the plots present an exponential slope after a few dozens of variables, especially for the case of more than three pairs.

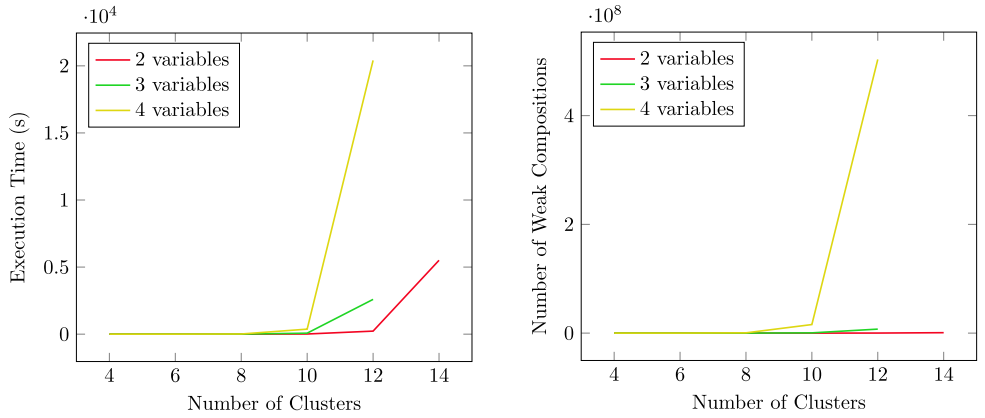
5. Related work

Probabilistic conditionals are also discussed in [32], where the authors proposed an approach based on the maximum entropy principle. Differently from [32], the approach of [9] does not constrain the distribution of the probability on the models.



(a) Execution Time (b) Number of weak compositions.

Fig. 4. Results for the conditional $(c(X)|a(X), b(X,Y))[0.2, 1]$ by fixing the number of pairs $(a(X), b(X,Y))$ and increasing the number of $b(X,Y)$.



(a) Execution Time (b) Number of weak compositions.

Fig. 5. Results for the conditional $(c(X)|a(X), b(X,Y))[0.2, 1]$ by fixing the variables per pair and increasing the number of clusters.

Most lifted inference approaches involve Probabilistic Logic Programming [33,22] or first-order knowledge bases in general [12,34]. To the best of our knowledge, no previous works proposed lifted approaches for (a subset of) Probabilistic Answer Set Programming (under the credal semantics) and statistical statements. A related approach is [23], where the authors introduced the concept of counting formulas associated with random variables, but they still do not consider the ASP syntax and probabilistic conditionals.

There are different possible semantics for probabilistic answer set programming, mainly the credal semantics [3] (that we adopt in this paper), the LP$^{\text{MLN}}$ semantics [2], and the P-log [35] semantics. LP$^{\text{MLN}}$ represent uncertain data using weighted rules, while P-log uses probabilities. The relation between the two has been analyzed in [36]. Both, differently from the credal semantics, assign a sharp probability value to a query. However, an in-depth study of the expressive power of the three different semantics is still missing and may be the subject of a future work. Finally, there exist solvers that allow to perform inference in the LP$^{\text{MLN}}$ and P-log semantics, as [37], as well as on the credal semantics [9], but none of these consider lifted inference.

6. Conclusions

In this paper, we proposed multiple formulas to perform lifted inference in statistical statements expressed as probabilistic answer set programs under the credal semantics. Our results cover some of the possible programs that contain a single conditional that can be written using one atom with one or two arguments in the consequent and one or two atoms with one or two variables in the antecedent. We also provide an implementation of the discussed formulas and empirically assessed the performance. A potential future work could consist in an in-depth analysis of other structures of conditionals and their computational complexity [21], also by adopting different semantics, and the development of lifted formulas for other types of reasoning tasks, such as abduction [38].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Source code and datasets available at: https://github.com/damianoazzolini/pasta.

Acknowledgements

This work has been partially supported by the 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 TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No. 952215, and by the “National Group of Computing Science (GNCS INdAM)”.

References

[1] G. Brewka, T. Eiter, M. Truszczyński, Answer set programming at a glance, Commun. ACM 54 (12) (2011) 92–103, https://doi.org/10.1145/2043174.2043195.
[2] J. Lee, Y. Wang, A probabilistic extension of the stable model semantics, in: AAAI Spring Symposia, 2015.
[3] F.G. Cozman, D.D. Mauá, The structure and complexity of credal semantics, in: A. Hommersom, S.A. Abdallah (Eds.), PLP 2016, in: CEUR Workshop Proceedings, vol. 1661, 2016, pp. 3–14, CEUR-WS.org.
[4] D. Azzolini, F. Riguzzi, E. Lamma, A semantics for hybrid probabilistic logic programs with function symbols, Artif. Intell. 294 (2021) 103452, https://doi.org/10.1016/j.artint.2021.103452.
[5] F. Riguzzi, Foundations of Probabilistic Logic Programming: Languages, Semantics, Inference and Learning, River Publishers, Gistrup, Denmark, 2018.
[6] T. Sato, A statistical learning method for logic programs with distribution semantics, in: L. Sterling (Ed.), ICLP 1995, MIT Press, 1995, pp. 715–729.
[7] J.Y. Halpern, An analysis of first-order logics of probability, Artif. Intell. 46 (3) (1990) 311–350, https://doi.org/10.1016/0004-3702(90)90019-V.
[8] L. De Raedt, A. Kimmig, H. Toivonen, ProbLog: a probabilistic prolog and its application in link discovery, in: M.M. Veloso (Ed.), IJCAI 2007, vol. 7, AAAI Press, 2007, pp. 2462–2467.
[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.
[10] F.G. Cozman, D.D. Mauá, The joy of probabilistic answer set programming: semantics, complexity, expressivity, inference, Int. J. Approx. Reason. 125 (2020) 218–239, https://doi.org/10.1016/j.ijar.2020.07.004.
[11] G. Van den Broeck, K. Kersting, S. Natarajan, D. Poole, An Introduction to Lifted Probabilistic Inference, MIT Press, 2021.
[12] R. de Salvo Braz, E. Amir, D. Roth, Lifted first-order probabilistic inference, in: L.P. Kaelbling, A. Saffiotti (Eds.), 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Professional Book Center, 2005, pp. 1319–1325.
[13] J.W. Lloyd, Foundations of Logic Programming, 2nd edition, Springer, 1987.
[14] W. Faber, N. Leone, G. Pfeifer, Recursive aggregates in disjunctive logic programs: semantics and complexity, in: European Workshop on Logics in Artificial Intelligence, Springer, 2004, pp. 200–212.
[15] M. Alviano, W. Faber, Aggregates in answer set programming, Künstl. Intell. 32 (2) (2018) 119–124, https://doi.org/10.1007/s13218-018-0545-9.
[16] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, ASP-core-2: input language format, Theory Pract. Log. Program. 20 (2) (2020) 294–309, https://doi.org/10.1017/S1471068419000450.
[17] M. Gebser, B. Kaufmann, T. Schaub, Solution enumeration for projected Boolean search problems, in: W.-J. van Hoeve, J.N. Hooker (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 71–86.
[18] J. Vennekens, S. Verbaeten, M. Bruynooghe, Logic programs with annotated disjunctions, in: B. Demoen, V. Lifschitz (Eds.), ICLP 2004, in: LNCS, vol. 3131, Springer, Berlin, Heidelberg, 2004, pp. 431–445.
[19] Potassco, 2022, Potassco user guide.
[20] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: propositional case, Ann. Math. Artif. Intell. 15 (3) (1995) 289–323.
[21] D.D. Mauá, F.G. Cozman, Complexity results for probabilistic answer set programming, Int. J. Approx. Reason. 118 (2020) 133–154, https://doi.org/10.1016/j.ijar.2019.12.003.
[22] F. Riguzzi, E. Bellodi, R. Zese, G. Cota, E. Lamma, A survey of lifted inference approaches for probabilistic logic programming under the distribution semantics, Int. J. Approx. Reason. 80 (2017) 313–333, https://doi.org/10.1016/j.ijar.2016.10.002.
[23] B. Milch, L.S. Zettlemoyer, K. Kersting, M. Haimes, L.P. Kaelbling, Lifted probabilistic inference with counting formulas, in: D. Fox, C.P. Gomes (Eds.), 23rd AAAI Conference on Artificial Intelligence (AAAI 2008), AAAI Press, 2008, pp. 1062–1068.
[24] G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, L. De Raedt, Lifted probabilistic inference by first-order knowledge compilation, in: T. Walsh (Ed.), 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), IJCAI/AAAI, 2011, pp. 2178–2185.
[25] N. Taghipour, D. Fierens, J. Davis, H. Blockeel, Lifted variable elimination: decoupling the operators from the constraint language, J. Artif. Intell. Res. 47 (2013) 393–439.
[26] F. Riguzzi, T. Swift, The PITA system: tabling and answer subsumption for reasoning under uncertainty, Theory Pract. Log. Program. 11 (4–5) (2011) 433–449.
[27] A. Nguembang Fadja, F. Riguzzi, E. Lamma, Learning hierarchical probabilistic logic programs, Mach. Learn. 110 (7) (2021) 1637–1693, https://doi.org/10.1007/s10994-021-06016-4.
[28] A. Nguembang Fadja, F. Riguzzi, Lifted discriminative learning of probabilistic logic programs, Mach. Learn. 108 (7) (2019) 1111–1135, https://doi.org/10.1007/s10994-018-5750-0.
[29] P. Beame, G. Van den Broeck, E. Gribkoff, D. Suciu, Symmetric weighted first-order model counting, in: Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS ’15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 313–328.
[30] R.P. Stanley, Enumerative Combinatorics, 2nd edition, Cambridge Studies in Advanced Mathematics, vol. 1, Cambridge University Press, 2011.
[31] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory Pract. Log. Program. 19 (1) (2019) 27–82, https://doi.org/10.1017/S1471068418000054.
[32] G. Kern-Isberner, M. Thimm, Novel semantical approaches to relational probabilistic conditionals, in: Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, 2010, pp. 382–392.

[33] E. Bellodi, E. Lamma, F. Riguzzi, V.S. Costa, R. Zese, Lifted variable elimination for probabilistic logic programming, Theory Pract. Log. Program. 14 (4–5) (2014) 681–695, https://doi.org/10.1017/S1471068414000283.
[34] G. Van den Broeck, On the completeness of first-order knowledge compilation for lifted probabilistic inference, in: J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F.C.N. Pereira, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, NIPS 2011, vol. 24, 2011, pp. 1386–1394.
[35] C. Baral, M. Gelfond, N. Rushton, Probabilistic reasoning with answer sets, Theory Pract. Log. Program. 9 (1) (2009) 57–144, https://doi.org/10.1017/S1471068408003645.
[36] 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.
[37] S. Hahn, T. Janhunen, R. Kaminski, J. Romero, N. Rühling, T. Schaub, plingo: a system for probabilistic reasoning in clingo based on LPMLN, https://doi.org/10.48550/ARXIV.2206.11515, 2022.
[38] D. Azzolini, E. Bellodi, S. Ferilli, F. Riguzzi, R. Zese, Abduction with probabilistic logic programming under the distribution semantics, Int. J. Approx. Reason. 142 (2022) 41–63, https://doi.org/10.1016/j.ijar.2021.11.003.