Probabilistic Ontologies in Datalog+/-

Fabrizio Riguzzi, Elena Bellodi, and Evelina Lamma

ENDIF – University of Ferrara, Via Saragat 1, I-44122, Ferrara, Italy
{fabrizio.riguzzi,elena.bellodi,evelina.lamma}@unife.it

Abstract. In logic programming the distribution semantics is one of the most popular approaches for dealing with uncertain information. In this paper we apply the distribution semantics to the Datalog+/- language that is grounded in logic programming and allows tractable ontology querying. In the resulting semantics, called DISPONTE, formulas of a probabilistic ontology can be annotated with an epistemic or a statistical probability. The epistemic probability represents a degree of confidence in the formula, while the statistical probability considers the populations to which the formula is applied. The probability of a query is defined in terms of finite set of finite explanations for the query. We also compare the DISPONTE approach for Datalog+/- ontologies with that of Probabilistic Datalog+/- where an ontology is composed of a Datalog+/- theory whose formulas are associated to an assignment of values for the random variables of a companion Markov Logic Network.

1 Introduction

Representing uncertain information is fundamental for ensuring that the Semantic Web achieves its full potential [36, 29, 25]. Ontologies are a decisive component of the Semantic Web and recently Datalog was extended for modeling ontologies [5, 6]. Answering conjunctive queries in the resulting language, Datalog+/-, has a polynomial data complexity, thus allowing tractable query answering.

Probabilistic Datalog+/- [15, 14] has been proposed for representing uncertainty in Datalog+/-. In this approach an ontology is composed of a Datalog+/- theory and a Markov Logic Network (MLN) [30] and each Datalog+/- formula is associated to an assignment of values to (a subset of) the random variables that are modeled by the MLN. This assignment, called *scenario*, controls the activation of the formulas: they hold only in worlds where the scenario is satisfied.

In the field of logic programming, the distribution semantics [35] has emerged as one of the most effective approaches for integrating logic and probability and underlies many languages such as PRISM [35], ICL [28], Logic Programs with Annotated Disjunctions [37] and ProbLog [11]. In this semantics the clauses of a probabilistic logic program contain alternative choices annotated with probabilities. Each grounding of a probabilistic clause represents a random variable that can assume a value from the finite set of alternatives. In order to compute the probability of a query, its explanations have to be found, where an explanation is a set of choices that ensure the entailment of the query. The set of explanations must be covering, i.e., it must represent all possible ways of entailing the query. The probability is computed from a covering set of explanations by solving a disjoint sum problem, either using an iterative splitting algorithm [28] or Binary Decision Diagrams [19, 31].

In Bellodi et al. [1] we have applied the distribution semantics to ontology languages based on Description Logic. We called the approach DISPONTE for "DIstribution Semantics for Probabilistic ONTologiEs" (Spanish for "get ready"). In this paper we apply DISPONTE to Datalog+/-. The idea is to annotate formulas of a theory with a probability and assume that each formula is independent of the others. Moreover, we extend [1] by allowing two types of probabilistic annotations, an epistemic type, that represents a degree of belief in the formula as a whole, and a statistical type, that considers the populations to which the formula is applied. While in the first case the choice is whether to include or not a formula in an explanation, in the latter case the choice is whether to include instantiations of the formula for specific individuals. The probability of a query is again computed from a covering set of explanations by solving the disjoint sum problem. We call the resulting language DISPONTE Datalog+/-.

The paper is organized as follows. Section 2 provides some preliminaries on Datalog+/-. Section 3 presents DISPONTE Datalog+/-. Section 4 describes related work and Section 5 concludes the paper.

2 Datalog+/-

Datalog+/- extends Datalog by allowing existential quantifiers, the equality predicate and the truth constant false in rule heads. Datalog+/- can be used for representing lightweight ontologies and is able to express the DL-Lite family of ontology languages [5]. By suitably restricting the language syntax, Datalog+/- achieves tractability [4].

In order to describe Datalog+/-, let us assume (i) an infinite set of data constants Δ, (ii) an infinite set of labeled nulls Δ_N (used as "fresh" Skolem terms), and (iii) an infinite set of variables Δ_V. Different constants represent different values (unique name assumption), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ vectors of variables $X_1, \ldots, X_k$ with $k \geq 0$. A relational schema $\mathcal{R}$ is a finite set of relation names (or predicates). A term t is a constant, null or variable. An atomic formula (or atom) has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. A database D for $\mathcal{R}$ is a possibly infinite set of atoms with predicates from $\mathcal{R}$ and arguments from $\Delta \cup \Delta_N$. A conjunctive query (CQ) over $\mathcal{R}$ has the form $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms having as arguments variables $\mathbf{X}$ and $\mathbf{Y}$ and constants (but no nulls). A Boolean CQ (BCQ) over $\mathcal{R}$ is a CQ having head predicate q of arity 0 (i.e., no variables in $\mathbf{X}$).

We often write a BCQ as the set of all its atoms, having constants and variables as arguments, and omitting the quantifiers. Answers to CQs and BCQs are defined via homomorphisms, which are mappings $\mu : \Delta \cup \Delta_N \cup \Delta_V \rightarrow \Delta \cup$

$\Delta_N \cup \Delta_V$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to term vectors, atoms, sets of atoms, and conjunctions of atoms. The set of all answers to a CQ $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu : \mathbf{X} \cup \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The answer to a BCQ $q = \exists \mathbf{Y} \Phi(\mathbf{Y})$ over a database D, denoted $q(D)$, is Yes, denoted $D \models q$, iff there exists a homomorphism $\mu : \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{Y})) \subseteq D$, i.e., if $q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency* (or TGD) F is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of F, respectively. Such F is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, there exists an extension h' of h such that $h'(\Psi(\mathbf{X}, \mathbf{Z})) \subseteq D$. We usually omit the universal quantifiers in TGDs. A TGD is *guarded* iff it contains an atom in its body that involves all variables appearing in the body.

Query answering under TGDs is defined as follows. For a set of TGDs T on $\mathcal{R}$, and a database D for $\mathcal{R}$, the set of models of D given T, denoted $mods(D, T)$, is the set of all (possibly infinite) databases B such that $D \subseteq B$ and every $F \in T$ is satisfied in B. The set of answers to a CQ q on D given T, denoted $ans(q, D, T)$, is the set of all tuples $\mathbf{t}$ such that $\mathbf{t} \in q(B)$ for all $B \in mods(D, T)$. The answer to a BCQ q over D given T is Yes, denoted $D \cup T \models q$, iff $B \models q$ for all $B \in mods(D, T)$.

A Datalog+/- theory may contain also *negative constraints* (or NC), which are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \to \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (not necessarily guarded). The universal quantifiers are usually left implicit.

Equality-generating dependencies (or EGDs) are the third component of a Datalog+/- theory. An EGD F is a first-order formula of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \to X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of F, is a conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. We call $X_i = X_j$ the *head* of F. Such F is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. We usually omit the universal quantifiers in EGDs.

The *chase* is a bottom-up procedure for deriving atoms entailed by a database and a Datalog+/- theory. The chase works on a database through the so-called TGD and EGD chase rules. Given a relational database D for a schema $\mathcal{R}$, and a TGD F on $\mathcal{R}$ of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$, F is *applicable to* D if there is a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D. Let F be applicable and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, where z_j is a "fresh" null, i.e., $z_j \in \Delta_N, z_j \notin D$, and z_j lexicographically follows all other labeled nulls already introduced. The result of the application of the TGD chase rule for F is the addition to D of all the atomic formulas in $h_1(\Psi(\mathbf{X}, \mathbf{Z}))$ that are not already in D.

The EGD chase rule is defined as follows. An EGD F on $\mathcal{R}$ of the form $\Phi(\mathbf{X}) \rightarrow X_i = X_j$ is *applicable to* a database D for $\mathcal{R}$ iff there exists a homomorphism $h : \Phi(\mathbf{X}) \rightarrow D$ such that $h(X_i)$ and $h(X_j)$ are different and not both constants. If $h(X_i)$ and $h(X_j)$ are different constants in Δ, then there is a *hard violation* of F. Otherwise, the result of the application of F to D is the database $h(D)$ obtained from D by replacing every occurrence of a non-constant element $e \in \{h(X_i), h(X_j)\}$ in D by the other element e' (if e and e' are both nulls, then e precedes e' in the lexicographic order).

The chase algorithm consists of an exhaustive application of the TGD and EGD chase rules that may lead to an infinite result. The chase rules are applied iteratively, in each iteration (1) a single TGD is applied once and then (2) the EGDs are applied until a fix point is reached. EGDs are assumed to be separable [7]. Intuitively, separability holds whenever: (i) if there is a hard violation of an EGD in the chase, then there is also one on the database w.r.t. the set of EGDs alone (i.e., without considering the TGDs); and (ii) if there is no hard violation, then the answers to a BCQ w.r.t. the entire set of dependencies equals those w.r.t. the TGDs alone (i.e., without the EGDs).

The two problems of CQ and BCQ evaluation under TGDs and EGDs are LOGSPACE-equivalent [6]. Moreover, query answering under TGDs is equivalent to query answering under TGDs with only single atoms in their heads [4]. Henceforth, we focus only on the BCQ evaluation problem and we assume that every TGD has a single atom in its head. A BCQ q on a database D, a set T_T of TGDs and a set T_E of EGDs can be answered by performing the chase and checking whether the query is entailed by the extended database that is obtained. In this case we write $D \cup T_T \cup T_E \models q$.

Example 1. Let us consider the following ontology for a real estate information extraction system, a slight modification of the one presented in Gottlob et al. [15]:

$F_1 = ann(X, label), ann(X, price), visible(X) \rightarrow priceElem(X)$
If X is annotated as a label, as a price and is visible, then it is a price element.

$F_2 = ann(X, label), ann(X, priceRange), visible(X) \rightarrow priceElem(X)$
If X is annotated as a label, as a price range, and is visible, then it is a price element.

$F_3 = priceElem(E), group(E, X) \rightarrow forSale(X)$
If E is a price element and is grouped with X, then X is for sale.

$F_4 = forSale(X) \rightarrow \exists P price(X, P)$
If X is for sale, then there exists a price for X.

$F_5 = hasCode(X, C), codeLoc(C, L) \rightarrow loc(X, L)$
If X has postal code C, and C's location is L, then X's location is L.

$F_6 = hasCode(X, C) \rightarrow \exists L codeLoc(C, L), loc(X, L)$
If X has postal code C, then there exists L such that C has location L and so does X.

$F_7 = loc(X, L1), loc(X, L2) \rightarrow L1 = L2$
If X has the locations $L1$ and $L2$, then $L1$ and $L2$ are the same.

$F_8 = loc(X, L) \rightarrow advertised(X)$

If X has a location L then X is advertised.

Suppose we are given the database

$$\begin{array}{l} codeLoc(ox1, central), codeLoc(ox1, south), codeLoc(ox2, summertown) \\ hasCode(prop1, ox2), ann(e1, price), ann(e1, label), visible(e1), \\ group(e1, prop1) \end{array}$$

The atomic BCQs $priceElem(e1)$, $forSale(prop1)$ and $advertised(prop1)$ evaluate to true, while the CQ $loc(prop1, L)$ has answers $q(L) = \{summertown\}$. In fact, even if $loc(prop1, z_1)$ with $z_1 \in \Delta_N$ is entailed by formula F_5, formula F_7 imposes that $summertown = z_1$. If F_7 were absent then $q(L) = \{summertown, z_1\}$.

Answering BCQs q over databases and ontologies containing NCs can be performed by first checking whether the BCQ $\Phi(\mathbf{X})$ evaluates to false for each NC of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$. If one of these checks fails, then the answer to the original BCQ q is positive, otherwise the negative constraints can be simply ignored when answering the original BCQ q.

A guarded Datalog+/- ontology is a quadruple (D, T_T, T_C, T_E) consisting of a database D, a finite set of guarded TGDs T_T, a finite set of negative constraints T_C and a finite set of EGDs T_E that are separable from T_T. The data complexity (i.e., the complexity where both the query and the theory are fixed) of evaluating BCQs relative to a guarded Datalog+/- theory is polynomial [4].

In the case in which the EGDs are key dependencies and the TGDs are inclusion dependencies, Calì et al. [8] proposed a backward chaining algorithm for answering BCQ. A *key dependency* κ is a set of EGDs of the form

$$\{r(\mathbf{X}, Y_1, \ldots, Y_m), r(\mathbf{X}, Y'_1, ..., Y'_m) \rightarrow Y_i = Y'_i\}_{1 \leq i \leq m}$$

A TGD of the form $r_1(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} r_2(\mathbf{X}, \mathbf{Z})$, where r_1 and r_2 are predicate names and no variable appears more than once in the body nor in the head, is called an *inclusion dependency*. The key dependencies must not interact with the inclusion dependencies, similarly to the semantic separability condition mentioned above for TGDs and EGDs. In this case once it is known that no hard violation occurs, queries can be answered by considering the inclusion dependencies only, ignoring the key dependencies. A necessary and sufficient syntactic condition for non interaction is based on the construction of CD-graphs [8].

3 The DISPONTE Semantics for Probabilistic Ontologies

The distribution semantics [35] is one of the most widely used semantics for probabilistic logic programming. In this semantics a probabilistic logic program defines a probability distribution over a set of normal logic programs (called *worlds*). The distribution is extended to a joint distribution over worlds and a query and the probability of the query is obtained from this distribution by marginalization.

In this section we discuss how we applied this approach to give a semantics to a probabilistic version of Datalog+/- that we call DISPONTE Datalog+/-.

A *probabilistic DISPONTE Datalog+/- ontology* (or simply *probabilistic ontology*) consists of a database D and a set of *certain formulas*, that take the form of a Datalog+/- TGD, NC or EGD, of *epistemic probabilistic formulas* of the form

$$p_i ::_e F_i \tag{1}$$

where p_i is a real number in $[0, 1]$ and F_i is a TGD, NC or EGD, and of *statistical probabilistic formulas* of the form

$$p_i ::_s F_i \tag{2}$$

where p_i is a real number in $[0, 1]$ and F_i is a TGD.

Let us call T_T the set of all TGD formulas (certain, epistemic or statistical), T_C the set of NC formulas (certain or epistemic) and T_E the set of EGD formulas (certain or epistemic). Thus a probabilistic ontology O is a quadruple $O = (D, T_T, T_C, T_E)$. Let us indicate with T the union $T_T \cup T_C \cup T_E$.

In formulas of the form (1), p_i is interpreted as an epistemic probability, i.e., as the degree of our belief in formula F_i, while in formulas of the form (2), p_i is interpreted as a statistical probability, i.e., as information regarding random individuals from certain populations. These two types of statements can be related to the work [16]: an epistemic statement is a Type 2 statement and a statistical statement is a Type 1 statement according to Halpern's terminology.

For example, an epistemic probabilistic concept inclusion TGD of the form

$$p ::_e c(X) \rightarrow d(X) \tag{3}$$

represents the fact that we believe in the truth of $c \subseteq d$, where c and d are interpreted as sets of individuals, with probability p. A statistical probabilistic concept inclusion TGD of the form

$$p ::_s c(X) \rightarrow d(X) \tag{4}$$

instead means that a random individual of class c has probability p of belonging to d, thus representing the statistical information that a fraction p of the individuals of c belongs to d. In this way the overlap between c and d is quantified. The difference between the two formulas is that, if two individuals belong to class c, the probability that they both belong to d according to (3) is p, since p represents the truth of the formula *as a whole*, while according to (4) is $p \times p$, since are now considered instantiations of the formula for specific individuals, *each one* having the same probability p of beloging to class d.

The idea of DISPONTE Datalog+/- is to associate independent Boolean random variables to (instantiations of) the formulas. By assigning values to every random variable we obtain a *world*, the set of logic formulas whose random variable is assigned to 1.

To clarify what we mean by instantiations, we now define substitutions. Given a formula F, a *substitution* θ is a set of couples X/x where X is a variable universally quantified in the outermost quantifier in F and $x \in \Delta \cup \Delta_N$. The application of θ to F, indicated by $F\theta$, is obtained by replacing X with x in F and by removing X from the external quantification for every couple X/x in θ. An *instantiation* of a formula F is the result of applying a substitution to F.

To obtain a world w of a probabilistic ontology O, we include every certain formula in w. For each axiom of the form (1), we decide whether or not to include it in w. For each axiom of the form (2), we generate all the substitutions for the variables universally quantified in the outermost quantifier.

There may be an infinite number of instantiations. For each instantiated formula, we decide whether or not to include it in w. In this way we obtain a Datalog+/- theory which can be assigned a semantics as seen in Section 2.

To formally define the semantics of a probabilistic ontology we follow the approach of Poole [28]. An *atomic choice* in this context is a triple (F_i, θ_j, k) where F_i is the i-th formula, θ_j is a substitution and $k \in \{0, 1\}$. k indicates whether $F_i\theta_j$ is chosen to be included in a world ($k = 1$) or not ($k = 0$). If F_i is obtained from a certain formula, then $\theta_j = \emptyset$ and $k = 1$. If F_i is obtained from a formula of the form (1), then $\theta_j = \emptyset$. If F_i is obtained from a formula of the form (2), then θ_j instantiates the variables universally quantified in the outermost quantifier.

A *composite choice* κ is a consistent set of atomic choices, i.e., $(F_i, \theta_j, k) \in \kappa, (F_i, \theta_j, m) \in \kappa \Rightarrow k = m$ (only one decision for each formula). The probability of composite choice κ is $P(\kappa) = \prod_{(F_i,\theta_j,1)\in\kappa} p_i \prod_{(F_i,\theta_j,0)\in\kappa}(1 - p_i)$. A *selection* σ is a total composite choice, i.e., it contains an atomic choice (F_i, θ_j, k) for every instantiation $F_i\theta_j$ of formulas of the theory. Since the domain is infinite, every selection is, too. Let us indicate with $\mathcal{S}_O$ the set of all selections. $\mathcal{S}_O$ is infinite as well. A selection σ identifies a theory w_σ called a *world* in this way: $w_\sigma = \{F_i\theta_j | (F_i, \theta_j, 1) \in \sigma\}$. Let us indicate with $\mathcal{W}_O$ the set of all worlds. A composite choice κ identifies a set of worlds $\omega_\kappa = \{w_\sigma | \sigma \in \mathcal{S}_O, \sigma \supseteq \kappa\}$. We define the set of worlds identified by a set of composite choices K as $\omega_K = \bigcup_{\kappa\in K} \omega_\kappa$.

A composite choice κ is an *explanation* for a BCG query q if q is entailed by the database and every world of ω_κ. A set of composite choices K is *covering* with respect to q if every world w_σ in which q is entailed is such that $w_\sigma \in \omega_K$. Two composite choices κ_1 and κ_2 are *incompatible* if their union is inconsistent. A set K of composite choices is *mutually incompatible* if for all $\kappa_1 \in K, \kappa_2 \in K, \kappa_1 \neq \kappa_2 \Rightarrow \kappa_1$ and κ_2 are incompatible.

Kolmogorov defined probability functions (or measures) as real-valued functions over an algebra Ω of subsets of a set $\mathcal{W}$ called the *sample space*. The set Ω is an algebra of $\mathcal{W}$ iff (1) $\mathcal{W} \in \Omega$, (2) Ω is closed under complementation, i.e., $\omega \in \Omega \rightarrow (\mathcal{W} \setminus \omega) \in \Omega$ and (3) Ω is closed under finite union, i.e., $\omega_1 \in \Omega, \omega_2 \in \Omega \rightarrow (\omega_1 \cup \omega_2) \in \Omega$. The elements of Ω are called *measurable sets*. Not every subset of $\mathcal{W}$ need be present in Ω.

Given a sample space $\mathcal{W}$ and an algebra Ω of subsets of $\mathcal{W}$, a probability measure is a function $\mu : \Omega \rightarrow R$ that satisfies the following axioms: (1) $\mu(\omega) \geq 0$ for all $\omega \in \Omega$, (2) $\mu(\mathcal{W}) = 1$, (3) $\omega_1 \cap \omega_2 = \emptyset \rightarrow \mu(\omega_1 \cup \omega_2) = \mu(\omega_1) + \mu(\omega_2)$ for all $\omega_1 \in \Omega, \omega_2 \in \Omega$. [28] proved the following results:

– Given a finite set K of finite composite choices, there exists a finite set K' of mutually incompatible finite composite choices such that $\omega_K = \omega_{K'}$;
– If K_1 and K_2 are both mutually incompatible finite sets of finite composite choices such that $\omega_{K_1} = \omega_{K_2}$ then $\sum_{\kappa \in K_1} P(\kappa) = \sum_{\kappa \in K_2} P(\kappa)$.

These results also hold for the probabilistic ontologies we consider so we can define a unique probability measure $\mu : \Omega_O \to [0,1]$ where Ω_O is defined as the set of sets of worlds identified by finite sets of finite composite choices: $\Omega_O = \{\omega_K | K \text{ is a finite set of finite composite choices}\}$. It is easy to see that Ω_O is an algebra over $\mathcal{W}_O$.

Then μ is defined by $\mu(\omega_K) = \sum_{\kappa \in K'} P(\kappa)$ where K' is a finite mutually incompatible set of finite composite choices such that $\omega_K = \omega_{K'}$. $\langle \mathcal{W}_T, \Omega_T, \mu \rangle$ is a probability space according to Kolmogorov's definition.

The probability of a BCQ query $q = \exists \mathbf{Y} \Phi(\mathbf{Y})$, where $\Phi(\mathbf{Y})$ is a conjunction of atoms having as arguments variables $\mathbf{Y}$ and constants (but no nulls) is $P(q) = \mu(\{w | w \in \mathcal{W}_O \wedge D \cup w \models q\})$. If q has a finite set K of finite explanations such that K is covering then $\{w | w \in \mathcal{W}_O \wedge D \cup w \models q\} \in \Omega_O$ and $P(q)$ is well-defined.

Let $H(Q)$ be the set of all homomorphisms of the form $h : \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\{w | w \in \mathcal{W}_O \wedge D \cup w \models h(\Phi(\mathbf{Y}))\}$ is not empty. Then

$$\{w | w \in \mathcal{W}_O \wedge D \cup w \models q\} = \bigcup_{h \in H(q)} \{w | w \in \mathcal{W}_O \wedge D \cup w \models h(\Phi(\mathbf{Y}))\}.$$

[28] also proposed an algorithm, called *splitting algorithm*, to obtain a set of mutually incompatible composite choices from any set of composite choices.

Example 2. Let us consider the following probabilistic ontology, obtained from the one presented in Example 1 by adding probabilistic annotations:

$$
\begin{aligned}
0.4 ::_s\ & F_1 = ann(X, label), ann(X, price), visible(X) \to priceElem(X) \\
0.5 ::_s\ & F_2 = ann(X, label), ann(X, priceRange), visible(X) \to priceElem(X) \\
0.6 ::_s\ & F_3 = priceElem(E), group(E, X) \to forSale(X) \\
& F_4 = forSale(X) \to \exists P price(X, P) \\
& F_5 = hasCode(X, C), codeLoc(C, L) \to loc(X, L) \\
& F_6 = hasCode(X, C) \to \exists L codeLoc(C, L), loc(X, L) \\
0.8 ::_e\ & F_7 = loc(X, L1), loc(X, L2) \to L1 = L2 \\
0.7 ::_s\ & F_8 = loc(X, L) \to advertised(X)
\end{aligned}
$$

and the database of Example 1:

$$
\begin{aligned}
& codeLoc(ox1, central), codeLoc(ox1, south), codeLoc(ox2, summertown), \\
& hasCode(prop1, ox2), ann(e1, price), ann(e1, label), visible(e1), \\
& group(e1, prop1)
\end{aligned}
$$

A covering set of explanations for the query $q = priceElem(e1)$ is $K = \{\kappa_1\}$ where $\kappa_1 = \{(F_1, \{X/e1\}, 1)\}$. K is also mutually exclusive so $P(Q) = 0.4$.

A covering set of explanations for the query $q = forSale(prop1)$ is $K = \{\kappa_1, \kappa_2\}$ where $\kappa_1 = \{(F_1, \{X/prop1\}, 1), (F_3, \{X/prop1\}, 1)\}$ and $\kappa_2 = \{(F_2, \{X/prop1\}, 1), (F_3, \{X/prop1\}, 1)\}$.

An equivalent mutually exclusive set of explanations obtained by applying the splitting algorithm is $K' = \{\kappa'_1, \kappa'_2\}$ where $\kappa'_1 = \{(F_1, \{X/prop1\}, 1), (F_3, \{X/prop1\}, 1), (F_2, \{X/prop1\}, 0)\}$ and $\kappa'_2 = \{(F_2, \{X/prop1\}, 1), (F_3, \{X/prop1\}\ , 1)\}$ so $P(q) = 0.4 \cdot 0.6 \cdot 0.5 + 0.5 \cdot 0.6 = 0.42$.

A covering set of explanations for the query $q = advertised(prop1)$ is $K = \{\kappa_1, \kappa_2, \kappa_3\}$ with

$$\begin{aligned}
\kappa_1 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 1)\}\\
\kappa_2 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 0)\}\\
\kappa_3 &= \{(F_8, \{X/prop1, L/z_1\}, 1), (F_7, \emptyset, 0)\}
\end{aligned}$$

where $z_1 \in \Delta_N$ and certain formulas have been omitted. A mutually exclusive set of explanations is $K' = \{\kappa'_1, \kappa'_2, \kappa'_3\}$ where

$$\begin{aligned}
\kappa'_1 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 1)\}\\
\kappa'_2 &= \{(F_8, \{X/prop1, L/summertown\}, 1), (F_7, \emptyset, 0), (F_8, \{X/prop1, L/z_1\}, 0)\}\\
\kappa'_3 &= \{(F_8, \{X/prop1, L/z_1\}, 1), (F_7, \emptyset, 0)\}
\end{aligned}$$

so $P(q) = 0.7 \cdot 0.8 + 0.7 \cdot 0.2 \cdot 0.3 + 0.7 \cdot 0.2 = 0.742$.

Example 3. Let us consider the following ontology, inspired by the `people+pets` ontology proposed in Patel-Schneider et al. [27]:

$$\begin{aligned}
F_1 &= hasAnimal(X, Y), pet(Y) \rightarrow petOwner(X)\\
0.6 ::_e F_2 &= cat(X) \rightarrow pet(X)
\end{aligned}$$

and the database $hasAnimal(kevin, fluffy), hasAnimal(kevin, tom), cat(fluffy), cat(tom)$. A covering set of explanations for the query $q = petOwner(kevin)$ is $K = \{\kappa_1\}$ where $\kappa_1 = \{(F_2, \emptyset, 1)\}$ and certain formulas have been omitted. This is also a mutually exclusive set of explanations so $P(q) = 0.6$.

Example 4. If the axiom $0.6 ::_e F_2 = cat(X) \rightarrow pet(X)$ in Example 3 is replaced by $0.6 ::_s F_2 = cat(X) \rightarrow pet(X)$ then the query would have the set of explanations $K = \{\kappa_1, \kappa_2\}$ where $\kappa_1 = \{(F_2, \{X/fluffy\}, 1)\}$ and $\kappa_2 = \{(F_2, \{X/tom\}, 1)\}$ which, after splitting, becomes $K' = \{\kappa'_1, \kappa'_2\}$ with

$$\begin{aligned}
\kappa'_1 &= \{(F_1, \{X/fluffy\}, 1), (F_1, \{X/tom\}, 0)\}\\
\kappa'_2 &= \{(F_1, \{X/tom\}, 1)\}
\end{aligned}$$

and certain formulas have been omitted, so $P(q) = 0.6 \cdot 0.4 + 0.6 = 0.84$.

Example 5. Let us consider a slightly different ontology:

$$\begin{aligned}
0.5 ::_s F_1 &= hasAnimal(X, Y), pet(Y) \rightarrow petOwner(X)\\
0.6 ::_s F_2 &= cat(X) \rightarrow pet(X)
\end{aligned}$$

and the database of Example 3. A covering set of explanations for the query $q = \mathit{petOwner}(\mathit{kevin})$ is $K = \{\kappa_1, \kappa_2\}$ where:

$$\begin{aligned}
\kappa_1 &= \{(F_1, \{X/\mathit{kevin}\}, 1), (F_2, \{X/\mathit{fluffy}\}, 1)\} \\
\kappa_2 &= \{(F_1, \{X/\mathit{kevin}\}, 1), (F_2, \{X/\mathit{tom}\}, 1)\}
\end{aligned}$$

An equivalent mutually exclusive set of explanations is $K' = \{\kappa'_1, \kappa'_2\}$ where:

$$\begin{aligned}
\kappa'_1 &= \{(F_1, \{X/\mathit{kevin}\}, 1), (F_2, \{X/\mathit{fluffy}\}, 1), (F_2, \{X/\mathit{tom}\}, 0)\} \\
\kappa'_2 &= \{(F_1, \{X/\mathit{kevin}\}, 1), (F_2, \{X/\mathit{tom}\}, 1)\}
\end{aligned}$$

so $P(q) = 0.5 \cdot 0.6 \cdot 0.4 + 0.5 \cdot 0.6 = 0.42$.

Example 6. Let us consider the following ontology:

$$\begin{aligned}
F_1 &= \exists \mathit{hasAnimal}(X, Y), \mathit{pet}(Y) \rightarrow \mathit{petOwner}(X) \\
0.6 ::_s F_2 &= \mathit{cat}(X) \rightarrow \mathit{pet}(X) \\
0.4 ::_e F_3 &= \mathit{cat}(\mathit{fluffy}) \\
0.3 ::_e F_4 &= \mathit{cat}(\mathit{tom})
\end{aligned}$$

and the database $\mathit{hasAnimal}(\mathit{kevin}, \mathit{fluffy}), \mathit{hasAnimal}(\mathit{kevin}, \mathit{tom})$. A covering set of explanations for the query axiom $q = \mathit{petOwner}(\mathit{kevin})$ is:
$K = \{\kappa_1, \kappa_2\}$ where

$$\begin{aligned}
\kappa_1 &= \{(F_3, \emptyset, 1), (F_2, \{X/\mathit{fluffy}\}, 1)\} \\
\kappa_2 &= \{(F_4, \emptyset, 1), (F_2, \{X/\mathit{tom}\}, 1)\}
\end{aligned}$$

and certain formulas have been omitted. After splitting K becomes
$K' = \{\kappa'_1, \kappa'_2, \kappa'_3\}$ with:

$$\begin{aligned}
\kappa'_1 &= \{(F_3, \emptyset, 1), (F_2, \{X/\mathit{fluffy}\}, 1), (F_4, \emptyset, 1), (F_2, \{X/\mathit{tom}\}, 0)\} \\
\kappa'_2 &= \{(F_3, \emptyset, 1), (F_2, \{X/\mathit{fluffy}\}, 1), (F_4, \emptyset, 0)\} \\
\kappa'_3 &= \{(F_4, \emptyset, 1), (F_2, \{X/\mathit{tom}\}, 1)\}
\end{aligned}$$

so $P(q) = 0.4 \cdot 0.6 \cdot 0.3 \cdot 0.4 + 0.4 \cdot 0.6 \cdot 0.7 + 0.3 \cdot 0.6 = 0.3768$.

Example 7. Let us consider a further ontology:

$$\begin{aligned}
F_1 &= 0.7 ::_s \mathit{schoolchild}(X) \rightarrow \mathit{european}(X) \\
F_2 &= 0.3 ::_s \mathit{schoolchild}(X) \rightarrow \mathit{onlyChild}(X) \\
F_3 &= 0.6 ::_s \mathit{european}(X) \rightarrow \mathit{goodInMath}(X) \\
F_4 &= 0.5 ::_s \mathit{onlyChild}(X) \rightarrow \mathit{goodInMath}(X)
\end{aligned}$$

and the database $schoolchild(anna)$. A covering set of explanations for the query $q = goodInMath(anna)$ is $K = \{\kappa_1, \kappa_2\}$ where:

$$\kappa_1 = \{(F_1, \{X/anna\}, 1), (F_3, \{X/anna\}, 1)\}$$
$$\kappa_2 = \{(F_2, \{X/anna\}, 1), (F_4, \{X/anna\}, 1)\}$$

After splitting we get $K' = \{\kappa'_1, \kappa'_2, \kappa'_3\}$ where:

$$\begin{aligned}\kappa'_1 &= \{(F_1, \{X/anna\}, 1), (F_3, \{X/anna\}, 1), (F_2, \{X/anna\}, 1),\\ &\quad (F_4, \{X/anna\}, 0)\}\\ \kappa'_2 &= \{(F_1, \{X/anna\}, 1), (F_3, \{X/anna\}, 1), (F_2, \{X/anna\}, 0)\}\\ \kappa'_3 &= \{(F_2, \{X/anna\}, 1), (F_4, \{X/anna\}, 1)\}\end{aligned}$$

So $P(q) = 0.7 \cdot 0.6 \cdot 0.3 \cdot 0.5 + 0.7 \cdot 0.6 \cdot 0.7 + 0.3 \cdot 0.5 = 0.507$.

4 Related Work

This work builds on Bellodi et al. [1] that presented a version of DISPONTE applied to Description Logics and where only epistemic probabilities were present.

Gottlob et al. [15, 14] presented Probabilistic Datalog+/-, a version of Datalog+/- that allows the representation of probabilistic information by combining Markov Logic Networks with Datalog+/-. Each Probabilistic Datalog+/- formula F is annotated with a *probabilistic scenario* λ, an assignment of values to a set of random variables from the MLN associated to the ontology. A full probabilistic scenario assigns a value to all the random variables of the MLN. A probabilistic scenario represents an event that happens when the random variables described by the MLN assume the values indicate in the scenario. Probabilistic formulas then take the form $F : \lambda$.

A Probabilistic Datalog+/- ontology is of the form $\Phi = (O, M)$ where O is a set of annotated formulas and M is an MLN. An annotated formula holds when the event associated with their probabilistic annotation holds.

If a is a ground atom, its probability in a Probabilistic Datalog+/- ontology $\Phi = (O, M)$, denoted $Pr(a)$, is obtained by summing the probabilities according to M of all full scenarios such that the atom is entailed by the annotated formulas that hold in the scenario.

Example 8. Let us consider the following probabilistic Datalog+/- ontology from [14]

$$\begin{aligned}F_1 &= visible(X) \rightarrow priceElem(X) : \{ann(X, label), ann(X, price)\}\\ F_2 &= visible(X) \rightarrow priceElem(X) : \{ann(X, label), ann(X, priceRange)\}\\ F_3 &= priceElem(E), group(E, X) \rightarrow forSale(X) : \{sale\}\\ F_4 &= forSale(E) \rightarrow \exists P price(X, P)\\ F_5 &= hasCode(X, C), codeLoc(C, L) \rightarrow loc(X, L)\\ F_6 &= hasCode(X, C) \rightarrow \exists L codeLoc(C, L), loc(X, L)\end{aligned}$$

$$F_7 = loc(X, L1), loc(X, L2) \to L1 = L2 : \{uniqueLoc\}$$

and the MLN

$$\begin{array}{ll} 0.3 & ann(X, label) \wedge ann(X, price) \\ 0.4 & ann(X, label) \wedge ann(X, priceRange) \\ 0.8 & sale \\ 1.1 & uniqueLoc \end{array}$$

Suppose that this network is grounded with respect to the only constant $e1$. The resulting ground network has 5 Boolean random variables, each corresponding to a logical atom. Therefore, there are 2^5 full scenarios.
In this theory $P(priceElem(e1)) = 0.492$ and $P(forSale(prop1)) = 0.339$.

DISPONTE differs from Probabilistic Datalog+/- because the probabilistic interactions among the atoms are modeled directly by means of Datalog+/- formulas rather than by a separate entity. The parameters of DISPONTE Datalog+/- are easier to interpret as they are probabilities (statistical or epistemic) while MLN parameters are weights not directly interpretable as probabilities.

Moreover, DISPONTE does not require the prior grounding of the probabilistic atoms, for which the set of constants has to be defined by the user, but allows an on demand grounding on the basis of the terms that are used for inference.

Heinsohn [17] proposed an extension of the description logic $\mathcal{ALC}$ that is able to express statistical information on the terminological knowledge such as partial concept overlapping. Similarly, Koller et al. [21] presented a probabilistic description logic based on Bayesian networks that deals with statistical terminological knowledge. [17, 21] do not allow probabilistic assertional knowledge about concept and role instances. Jaeger [18] allows assertional knowledge about concept and role instances together with statistical terminological knowledge and combines the resulting probability distributions using cross-entropy minimization but does not allow epistemic statements.

Ding et al. [12] proposed a probabilistic extension of OWL that admits a translation into Bayesian networks. The semantics that is proposed assigns a probability distribution $P(i)$ over individuals, i.e. $\sum_i P(i) = 1$, and assigns a probability to a class C as $P(C) = \sum_{i \in C} P(i)$, while we assign a probability measure to sets of worlds. PR-OWL [10, 9] is an upper ontology that provides a framework for building probabilistic ontologies. It allows to use the first-order probabilistic logic MEBN [22] for representing uncertainty in ontologies. The use of a full fledged first-order probabilistic logic distinguishes this work from ours, where we tried to provide a minimal extension to Datalog+/-.

A different approach to the combination of ontologies with probability is taken by Giugno et al. [13] and Lukasiewicz [23, 24], who use probabilistic lexicographic entailment from probabilistic default reasoning. The description logics proposed in these papers allows both terminological probabilistic knowledge as well as assertional probabilistic knowledge about instances of concepts and roles.

PRONTO [20] is one of the systems that allows to perform inference in this semantics. Similarly to Jaeger [18], the terminological knowledge is interpreted statistically while the assertional knowledge is interpreted in an epistemic way by assigning degrees of beliefs to assertions. Moreover it also allows to express default knowledge about concepts that can be overridden in subconcepts and whose semantics is given by Lehmann's lexicographic default entailment. These works are based on Nilsson's probabilistic logic [26] where a probabilistic interpretation Pr defines a probability distribution over the set of interpretations Int. The probability of a logic formula F according to Pr, denoted $Pr(F)$, is the sum of all $Pr(I)$ such that $I \in Int$ and $I \models F$.

A probabilistic knowledge base K in Nilsson's logic is a set of probabilistic formulas of the form $F \geq p$. A probabilistic interpretation Pr satisfies $F \geq p$ iff $Pr(F) \geq p$. Pr satisfies K, or Pr is a model of K, iff Pr satisfies all $F \geq p \in K$. We say $P(F) \geq p$ is a tight logical consequence of K iff p is the infimum of $Pr(F)$ subject to all models Pr of K. Computing tight logical consequences from probabilistic knowledge bases can be done by solving a linear optimization problem.

In fact Nilsson's logic allows weaker conclusions than the distribution semantics. For example, consider a probabilistic ProbLog [11] program composed of $0.4 :: a.$ and $0.5 :: b.$ and a probabilistic knowledge base composed of $a \geq 0.4$ and $b \geq 0.5$. The distribution semantics allows to say that $P(a \vee b) = 0.7$, while with Nilsson's logic the lowest p such that $Pr(a \vee b) \geq p$ holds is 0.5. This is due to the fact that in the distribution semantics the probabilistic atoms are considered as independent, which allows to make stronger conclusions. However, note that this does not restrict expressiveness as one can specify any joint probability distribution over the atoms interpreted as Boolean random variables, possibly introducing new random facts if needed.

Alternative approaches to modeling imperfect and incomplete knowledge in ontologies are based on fuzzy logic. A good survey of these approaches is presented in [25].

5 Conclusions

We have presented the DISPONTE semantics for probabilistic ontologies in Datalog+/- that is inspired by the distribution semantics of probabilistic logic programming and is a minimal extension of the underlying ontology semantics to allow for representing and reasoning with uncertain knowledge.

In the future we plan to develop inference algorithm for DISPONTE Datalog+/-, both bottom-up, on the basis of the chase procedure, and top-down, by applying PITA [34] to non interacting inclusion and key dependencies. Moreover, we will investigate the possibility of annotating also NCs and EGDs with a statistical probability. Finally, we will work towards the automatic learning of DISPONTE Datalog+/- ontologies, exploiting the techniques developed in Riguzzi [32], Riguzzi et al. [33], Bellodi et al. [3, 2].

References

1. Bellodi, E., Lamma, E., Riguzzi, F., Albani, S.: A distribution semantics for probabilistic ontologies. In: International Workshop on Uncertainty Reasoning for the Semantic Web. CEUR Workshop Proceedings, vol. 778. CEUR-WS.org (2011)
2. Bellodi, E., Riguzzi, F.: Learning the structure of probabilistic logic programs. In: International Conference on Inductive Logic Programming (2011)
3. Bellodi, E., Riguzzi, F.: Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intelligent Data Analysis 16(6) (2012)
4. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: International Conference on Principles of Knowledge Representation and Reasoning. pp. 70–80. AAAI Press (2008)
5. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Symposium on Principles of Database Systems. pp. 77–86. ACM (2009)
6. Calì, A., Gottlob, G., Lukasiewicz, T.: Tractable query answering over ontologies with Datalog+/-. In: International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
7. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: IEEE Symposium on Logic in Computer Science. pp. 228–242. IEEE Computer Society (2010)
8. Calì, A., Gottlob, G., Pieris, A.: Tractable query answering over conceptual schemata. In: International Conference on Conceptual Modeling. LNCS, vol. 5829, pp. 175–190. Springer (2009)
9. Carvalho, R.N., Laskey, K.B., Costa, P.C.G.: PR-OWL 2.0 - bridging the gap to OWL semantics. In: International Workshops on Uncertainty Reasoning for the Semantic Web (2010)
10. Costa, P.C.G., Laskey, K.B., Laskey, K.J.: PR-OWL: A bayesian ontology language for the semantic web. In: International Workshops on Uncertainty Reasoning for the Semantic Web. vol. 5327, pp. 88–107. Springer (2008)
11. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: International Joint Conference on Artificial Intelligence. pp. 2462–2467 (2007)
12. Ding, Z., Peng, Y.: A probabilistic extension to ontology language OWL. In: Hawaii International Conference on System Sciences. IEEE (2004)
13. Giugno, R., Lukasiewicz, T.: P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In: European Conference on Logics in Artificial Intelligence. LNCS, vol. 2424, pp. 86–97. Springer (2002)
14. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Answering threshold queries in probabilistic Datalog+/- ontologies. In: International Conference on Scalable Uncertainty Management. LNCS, vol. 6929, pp. 401–414. Springer (2011)
15. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Conjunctive query answering in probabilistic Datalog+/- ontologies. In: International Conference on Web Reasoning and Rule Systems. LNCS, vol. 6902, pp. 77–92. Springer (2011)
16. Halpern, J.Y.: An analysis of first-order logics of probability. Artificial Intelligence 46(3), 311–350 (1990)
17. Heinsohn, J.: Probabilistic description logics. In: Conference on Uncertainty in Artificial Intelligence. pp. 311–318. Morgan Kaufmann (1994)

18. Jaeger, M.: Probabilistic reasoning in terminological logics. In: International Conference on Principles of Knowledge Representation and Reasoning. pp. 305–316 (1994)
19. Kimmig, A., Demoen, B., Raedt, L.D., Costa, V.S., Rocha, R.: On the implementation of the probabilistic logic programming language ProbLog. Theory and Practice of Logic Programming 11(2-3), 235–262 (2011)
20. Klinov, P.: Pronto: A non-monotonic probabilistic description logic reasoner. In: European Semantic Web Conference. LNCS, vol. 5021, pp. 822–826. Springer (2008)
21. Koller, D., Levy, A.Y., Pfeffer, A.: P-classic: A tractable probablistic description logic. In: National Conference on Artificial Intelligence. pp. 390–397 (1997)
22. Laskey, K.B., Costa, P.C.G.: Of starships and klingons: Bayesian logic for the 23rd century. In: Conference in Uncertainty in Artificial Intelligence. pp. 346–353. AUAI Press (2005)
23. Lukasiewicz, T.: Probabilistic default reasoning with conditional constraints. Annals of Mathematics and Artificial Intelligence 34(1-3), 35–88 (2002)
24. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence 172(6-7), 852–883 (2008)
25. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics 6(4), 291–308 (2008)
26. Nilsson, N.J.: Probabilistic logic. Artificial Intelligence 28(1), 71–87 (1986)
27. Patel-Schneider, P, F., Horrocks, I., Bechhofer, S.: Tutorial on OWL (2003), `http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/`
28. Poole, D.: Abducing through negation as failure: stable models within the independent choice logic. Journal of Logic Programming 44(1-3), 5–35 (2000)
29. Predoiu, L., Stuckenschmidt, H.: Probabilistic extensions of semantic web languages - a survey. In: The Semantic Web for Knowledge and Data Management: Technologies and Practices. Idea Group Inc (2008)
30. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
31. Riguzzi, F.: Extended semantics and inference for the Independent Choice Logic. Logic Journal of the IGPL 17(6), 589–629 (2009)
32. Riguzzi, F.: ALLPAD: Approximate learning of logic programs with annotated disjunctions. Machine Learning 70(2-3), 207–223 (Mar 2008)
33. Riguzzi, F., Di Mauro, N.: Applying the information bottleneck to statistical relational learning. Machine Learning 86(1), 89–114 (2012)
34. Riguzzi, F., Swift, T.: The PITA system: Tabling and answer subsumption for reasoning under uncertainty. Theory and Practice of Logic Programming, International Conference on Logic Programming Special Issue 11(4–5), 433–449 (2011)
35. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: International Conference on Logic Programming. pp. 715–729. MIT Press (1995)
36. URW3-XG: Uncertainty reasoning for the World Wide Web, final report (2005)
37. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: International Conference on Logic Programming. LNCS, vol. 3131, pp. 195–209. Springer (2004)