Preference-Based Query Answering in Probabilistic Datalog+/– Ontologies

Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari

Department of Computer Science, University of Oxford, UK
{thomas.lukasiewicz,vanina.martinez,gerardo.simari}@cs.ox.ac.uk

Abstract. The incorporation of preferences into information systems, such as databases, has recently seen a surge in interest, mainly fueled by the revolution in Web data availability. Modeling the preferences of a user on the Web has also increasingly become appealing to many companies since the explosion of popularity of social media. The other surge in interest is in modeling uncertainty in these domains, since uncertainty can arise due to many uncontrollable factors. In this paper, we propose an extension of the Datalog+/– family of ontology languages with two models: one representing user preferences and one representing the (probabilistic) uncertainty with which inferences are made. Assuming that more probable answers are in general more preferable, this raises the question of how to rank answers to a user's queries, since the preference model may be in conflict with the preferences induced by the probabilistic model—the need thus arises for preference combination operators. We propose two specific operators and study their properties in terms of behavior and running time. Finally, we provide an algorithm for ranking answers based on the iteration of the well-known skyline answers to a query and show that, under certain conditions, it runs in polynomial time in the data complexity.

1 Introduction

There has recently been a marked push in the research and development of technology surrounding the Web and, perhaps most centrally, its vast repositories of data. Ontology and query languages are examples of such technology, used to share, integrate, and query large-scale and less structured data and knowledge bases, such as those occurring on the Web. One of the central issues in this domain is that Web search is still centered around the paradigm of receiving keywords from a user and returning a list of links to Web documents that are considered to be pertinent. *Semantic search*, on the other hand, has been proposed as an evolution of this paradigm that identifies objects, rather than whole documents, as candidates to answering the users' queries. At the same time, we have recently been witnessing another revolution in the rapid growth of what is generally referred to as the *Social Web* (which is often also implicitly connected to the Semantic Web [1]); the Social Web is centered around a (mostly) loosely coupled set of platforms that people make use of with the objective of sharing, viewing, and searching for information (in the form of pictures, text documents of varying lengths,

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, pp. 501–518, 2013.

videos, etc.) in a social and sometimes collaborative environment. The use of semantic search in the Social Web is of central importance, due to the missing link structure between Web pages, which is well-known from ranking (such as PageRank) in standard Web search. In addition, the fundamentally human component of these systems makes each user's *personal preferences* have a much more prevalent role than what was observed before this paradigm shift. Finally, the presence of uncertainty in the Web in general is undeniable [2–5]: information integration (as in travel sites that query multiple sources to find hotels and flights), automatic processing of Web data (analyzing an HTML document often involves uncertainty), as well as inherently uncertain data (such as user comments) are all examples of uncertainty that must be dealt with in answering queries in the Social Web. The current challenge for Web search is therefore inherently linked to: (1) leveraging the social components of Web content towards the development of some form of semantic search and query answering on the Web as a whole, and (2) dealing with the presence of uncertainty in a principled way throughout the process.

In this paper, we develop a novel integration of ontology languages with both preference and uncertainty management mechanisms by developing an extension of the Datalog+/– family of ontology languages [6] with a preference model over the consequences of the ontology, as well as a probabilistic model that assigns probabilities to them—note that all previous work on extending Datalog+/– with uncertainty (cf. [7] and references therein) does not deal with preferences. The former is assumed to model a user's (or group of users') preferences, while the latter is assumed to model the uncertainty in the domain. The specific mechanisms by which these models are given are left unspecified, since our focus is on the study of how to rank answers to a query when the two models may be in disagreement regarding the ranking: assuming that higher-probability consequences are more preferable than lower-probability ones, it is clear that such situations can arise. The main contributions of this paper are:

(i) the introduction of the PP-Datalog+/– framework, to our knowledge the first extension of ontology languages with both preferences as in relational databases and probabilistic uncertainty;

(ii) the formalization of *preference combination* operators, which take a preference relation in the form of a strict partial order and a score-based preference relation (a weak order) and produce a new preference relation satisfying certain basic properties;

(iii) the development of two specific preference combination algorithms, called CombinePrefs and CombinePrefsRank, which are shown to satisfy the requirements of a preference combination operator as well as several other desirable properties, among which are several postulates proposed in the literature for a less general case; and

(iv) an algorithm for answering k-rank queries, a generalization of top-k queries based on the iterative computation of classical skyline answers, for disjunctions of atomic queries (DAQs), along with proofs of correctness and running time showing that answering DAQs in PP-Datalog+/– can be done in polynomial time in the data complexity modulo the cost of computing probabilities.

The rest of this paper is organized as follows. In Section 2, we present preliminary concepts on classical Datalog+/–. Section 3 introduces the general preference and probabilistic models that are used in this work, motivates the need to combine the user preferences with those induced by probability values by means of preference combination

operators, and then goes on to present the syntax and semantics of PP-Datalog+/–. In Section 4, we present algorithms for the implementation of two preference combination operators, and study their semantic and computational properties. Section 5 presents skyline and k-rank queries and a basic algorithm to answer them, and proves its correctness and running time, showing that under certain conditions k-rank queries can be answered in polynomial time in the data complexity. Finally, Sections 6 and 7 discuss related work and conclusions, respectively.

2 Preliminaries on Datalog+/–

First, we briefly recall some basics on Datalog+/– [6], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple- and equality-generating dependencies (TGDs and EGDs, respectively), negative constraints, the chase procedure, and ontologies in Datalog+/–.

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, dependencies, and constraints). 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 $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is an n-ary predicate, and $t_1, ..., t_n$ are terms.

A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. 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 (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but no nulls. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \Delta_N \cup \mathcal{V} \to \Delta \cup \Delta_N \cup \mathcal{V}$ 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 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 D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ (without nulls), called the *body* and the *head* of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. All sets of TGDs are finite here. Since TGDs can be reduced to TGDs with only single atoms

in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard atom* (or *guard*) of σ.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [8], even when the schema and TGDs are fixed [9]. Decidability and tractability in the data complexity of query answering for the guarded case follows from a bounded tree-width property.

Negative constraints (or simply *constraints*) γ are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ (called the *body* of γ) is a conjunction of atoms (without nulls). Under the standard semantics of query answering of BCQs in Datalog+/– with TGDs, adding negative constraints is computationally easy, as for each constraint $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, we only have to check that the BCQ $\Phi(\mathbf{X})$ evaluates to false in D under Σ; if one of these checks fails, then the answer to the original BCQ Q is true, otherwise the constraints can simply be ignored when answering the BCQ Q.

Equality-generating dependencies (or *EGDs*) σ, are first-order formulas $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of σ, denoted $body(\sigma)$, is a (without nulls) conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there is a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. Adding EGDs over databases with TGDs along with negative constraints does not increase the complexity of BCQ query answering as long as they are *non-conflicting* [6]. Intuitively, this ensures that, if the chase (see below) fails (due to strong violations of EGDs), then it already fails on the database D, and if it does not fail, then whenever "new" atoms (from the logical point of view) are created in the chase by the application of the EGD chase rule, atoms that are logically equivalent to the new ones are guaranteed to be generated also in the absence of the EGDs, guaranteeing that EGDs do not influence the chase with respect to query answering.

We usually omit the universal quantifiers in TGDs, negative constraints, and EGDs, and we implicitly assume that all sets of dependencies and/or constraints are finite.

The Chase. The *chase* was first introduced to enable checking implication of dependencies, and later also for checking query containment. By "chase", we refer both to the chase procedure and to its output. The TGD chase works on a database via so-called TGD *chase rules* (see [6] for an extended chase with also EGD chase rules).

TGD Chase Rule. Let D be a database, and σ a TGD of the form $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$. Then, σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D. Let σ be applicable to D, 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 does not occur in D, and z_j lexicographically follows all other nulls already introduced. The *application of* σ on D adds to D the atom $h_1(\Psi(\mathbf{X}, \mathbf{Z}))$ if not already in D.

The chase algorithm for a database D and a set of TGDs Σ consists of an exhaustive application of the TGD chase rule in a breadth-first (level-saturating) fashion, which outputs a (possibly infinite) chase for D and Σ. Formally, the *chase of level up to* 0 of D relative to Σ, denoted $chase^0(D, \Sigma)$, is defined as D, assigning to every atom in D the *(derivation) level* 0. For every $k \geqslant 1$, the *chase of level up to* k of D relative to Σ, denoted $chase^k(D, \Sigma)$, is constructed as follows: let $I_1, \ldots, I_n$ be all possible images of bodies of TGDs in Σ relative to some homomorphism such that (i) $I_1, \ldots, I_n \subseteq chase^{k-1}(D, \Sigma)$ and (ii) the highest level of an atom in every I_i is $k-1$; then, perform every corresponding TGD application on $chase^{k-1}(D, \Sigma)$, choosing the applied TGDs and homomorphisms in a (fixed) linear and lexicographic order, respectively, and assigning to every new atom the *(derivation) level* k. The *chase* of D relative to Σ, denoted $chase(D, \Sigma)$, is defined as the limit of $chase^k(D, \Sigma)$ for $k \to \infty$.

The (possibly infinite) chase relative to TGDs is a *universal model*, i.e., there exists a homomorphism from $chase(D, \Sigma)$ onto every $B \in mods(D, \Sigma)$ [6]. This implies that BCQs Q over D and Σ can be evaluated on the chase for D and Σ, i.e., $D \cup \Sigma \models Q$ is equivalent to $chase(D, \Sigma) \models Q$. For guarded TGDs Σ, such BCQs Q can be evaluated on an initial fragment of $chase(D, \Sigma)$ of constant depth $k \cdot |Q|$, which is possible in polynomial time in the data complexity.

Datalog+/– Ontologies. A *Datalog+/– ontology* $KB = (D, \Sigma)$, where $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$, consists of a database D, a set of TGDs Σ_T, a set of non-conflicting EGDs Σ_E, and a set of negative constraints Σ_{NC}. We say KB is *guarded* (resp., *linear*) iff Σ_T is guarded (resp., linear). Example 1 illustrates a simple Datalog+/– ontology, which is used in the sequel as a running example.

Example 1. Consider the following simple ontology $O = (D, \Sigma)$, where:

$$\begin{aligned}\Sigma = \{&\mathit{room}(R, H) \to \mathit{accom}(R), \quad \mathit{bed}(B, H) \to \mathit{accom}(B),\\ &\mathit{apartment}(A) \to \mathit{accom}(A), \ \mathit{hotel}(H) \to \exists R\, \mathit{room}(R, H),\\ &\mathit{hostel}(H) \to \exists B\, \mathit{bed}(B, H), \ \mathit{bb}(B) \to \exists R\, \mathit{room}(R, B)\}\end{aligned}$$

and $D = \{\mathit{hotel}(h_1), \mathit{hotel}(h_2), \mathit{hostel}(hs_1), \mathit{bb}(bb_1), \mathit{apartment}(a), \mathit{bed}(b_1, hs_1), \mathit{bed}(b_2, hs_1), \mathit{room}(r_1, h_1), \mathit{room}(r_2, h_1), \mathit{room}(r_3, h_2), \mathit{room}(r_4, bb_1)\}$.

This ontology models a very simple accommodation booking domain, which could be used as the underlying model in an online system. Accommodations can be either rooms in hotels or bed and breakfasts, beds in hostels, or apartments. The database D provides some instances for each kind of accommodation. ∎

3 PP-Datalog+/–: Syntax and Semantics

In this section, we introduce the PP-Datalog+/– language, an extension of Datalog+/– with both a preference model and a probabilistic model; this formalism is based on the one first presented in [10], which extends Datalog+/– with preferences (but not probabilities). To this end, we assume the following sets giving rise to the logical languages for ontologies, preferences, and probability models: Δ_{Ont}, Δ_{Pref}, and Δ_M are finite sets of constants, $\mathcal{R}_{Ont}$, $\mathcal{R}_{Pref}$, and $\mathcal{R}_M$ are finite sets of predicate names such that

$\mathcal{R}_M \cap \mathcal{R}_{Ont} = \emptyset$, and $\mathcal{V}_{Ont}$, $\mathcal{V}_{Pref}$, and $\mathcal{V}_M$ are infinite sets of variables. In the following, we assume w.l.o.g. that $\mathcal{R}_{Pref} \subseteq \mathcal{R}_{Ont}$, $\Delta_{Pref} \subseteq \Delta_{Ont}$, $\mathcal{V}_{Pref} \subseteq \mathcal{V}_{Ont}$. These sets give rise to corresponding *Herbrand bases* consisting of all possible ground atoms that can be formed, which we denote with $\mathcal{H}_{Ont}$, $\mathcal{H}_{Pref}$, and $\mathcal{H}_M$, respectively. Clearly, we have $\mathcal{H}_{Pref} \subseteq \mathcal{H}_{Ont}$, meaning that preference relations are defined over a subset of the possible ground atoms.

Preference Models. A *preference relation* is any binary relation $\succ \subseteq \mathcal{H}_{Pref} \times \mathcal{H}_{Pref}$. In this paper, we are interested in strict partial orders (SPOs), which are irreflexive and transitive relations—we consider these to be the minimal requirements for a preference relation to be useful in the applications that we envision. One way of specifying such a relation that is especially compatible with our approach is the preference formula framework of [11]. In this work, we assume the existence of a general *preference model* P specifying a preference relation over a subset of $\mathcal{H}_{Ont}$, denoted $\succ_P$; in general, we treat $\succ_P$ as a set of ordered pairs. When a preference relation $\succ$ is induced by an assignment of a numeric score to each element in such a way that $a_1 \succ a_2$ iff $score(a_1) > score(a_2)$, we say that $\succ$ is *score-based*.

Example 2. Following the topic of Example 1, a preference relation might be specified by a user over the *accom* atoms; an example of such a relation is shown in Fig. 3 (top left) (we assume the transitive closure of the graph depicted). For instance, these preferences reflect that the user prefers the room in the bed and breakfast above all else; next, the apartment and rooms r_1 and r_3 are all incomparable (perhaps the user cannot decide between the cost and locations of the apartment and hotels h_1 and h_2). Towards the least preferred options are the two beds in the hostel, and room r_2 in hotel h_1. ∎

Probabilistic Models. For modeling uncertainty, we assume the existence of a probabilistic model M that represents a probability distribution $\Pr_M$ over some set $X = \{X_1, \ldots, X_n\}$ of Boolean variables such that there is a 1-to-1 mapping from X to the set of all ground atoms over $\mathcal{R}_M$ and Δ_M. Examples of the type of probabilistic models that we assume in this work are Markov logic and Bayesian networks. The probabilistic extension of Datalog+/– adopted here was first introduced in [12]; probabilistic query answering (without preferences) in a version of this model using Markov logic was also studied in [7].

Substitutions. A substitution is a mapping from variables to variables or constants. Two sets S and T *unify* via a substitution θ iff $\theta S = \theta T$, where θA denotes the application of θ to all variables in all elements of A (here, θ is a unifier). A *most general unifier* (mgu) is a unifier θ such that for all other unifiers ω, there is a substitution σ such that $\omega = \sigma \circ \theta$.

Definition 1. Let M be a probabilistic model. Then, a *(probabilistic) annotation* λ relative to M is a (finite) set of expressions of the form $\langle A_i = x_i \rangle$, where: (i) A_i is an atom over $\mathcal{R}_M$, $\mathcal{V}_M$, and Δ_M; and (ii) $x_i \in \{0, 1\}$. A probabilistic annotation is *valid* iff for any two different expressions $\langle A = x \rangle, \langle B = y \rangle \in \lambda$, there does not exist a substitution that unifies A and B.

Intuitively, a probabilistic annotation is used to describe the class of events in which the random variables in a probabilistic model M are compatible with the settings of the

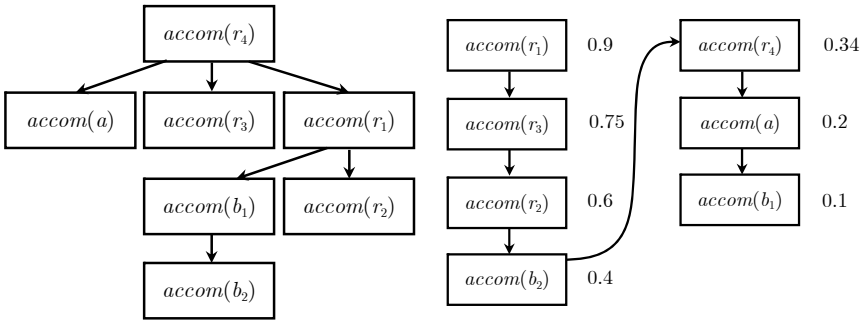


Fig. 1. User preferences for Example 1 (left), and those induced by probability values (right)

random variables described by λ, i.e., each X_i has the value x_i. A *probabilistic scenario* is a valid probabilistic annotation λ for which $|\lambda| = |X|$ and all $\langle A = x_i \rangle \in \lambda$ are such that A is ground. We use $scn(M)$ to denote the set of scenarios in M.

Example 3. Continuing with the running example, suppose the online booking system consults a probabilistic model that assigns probabilities specifying how likely it is that certain events happen. In this case, suppose that the system in question is aggregating information from multiple sources, which may contain conflicting information, as well as uncertainty due to other factors such as language. Thus, it would be useful to inform the user of the probability of accommodations being available for the dates being searched; the following is an example of how the ontology from Example 1 may be extended by replacing the atoms in the database with formulas of the form: $room(R, H) : \{available(R, H, D) = 1\}$, where $available(R, H, D)$ denotes the probabilistic event that the room R in the hotel (or hostel) H is available on the date D (which is instantiated through the query or by adding the information to the probabilistic model with the date specified by the user). Fig. 1 (right) gives an example of such a probability assignment (derived as explained in the semantics section below), along with the preference relation in graph form that is induced by these values assuming that higher probabilities are more preferable. ■

Preference Combination Operators. As seen in Fig. 1, the particular challenge encountered in PP-Datalog+/– ontologies is that the preference model yields a certain precedence relation that might be in disagreement with the one induced by the probabilistic model. To address this situation, we make use of *preference combination operators*, which take two preference relations and produce a third one satisfying two basic properties.

Definition 2. Let $\succ_P$ be an SPO and $\succ_M$ be a score-based preference relation. A *preference combination operator* $\otimes(\succ_M, \succ_P)$ yields a relation $\succ^*$ such that: (i) $\succ^*$ is an SPO, and (ii) if $a_1 \succ_P a_2$ and $a_1 \succ_M a_2$, then $a_1 \succ^* a_2$.

The two properties required by Definition 2 are the minimal required to produce a "reasonable" combination of the two relations; as we show in Section 4 below, particular implementations may satisfy further desirable properties.

We are now ready to define PP-Datalog+/– ontologies.

Definition 3. A *PP-Datalog+/– ontology* (or *PP-KB*) is of the form $KB = (O, P, M, \otimes)$, where O is a Datalog+/– ontology, P is a preference model, M is a probabilistic model with Herbrand bases $\mathcal{H}_{Ont}$, $\mathcal{H}_{Pref}$, and $\mathcal{H}_M$, respectively, such that $\mathcal{H}_{Pref} \subseteq \mathcal{H}_{Ont}$, and $\otimes$ is a preference combination operator. If O is a guarded Datalog+/– ontology, then KB is a *guarded* PP-Datalog+/– ontology.

In the following, we discuss the semantics of PP-Datalog+/– ontologies, and formally define the kinds of queries that we address in this paper.

Semantics. Let $KB = (O, P, M, \otimes)$ be PP-Datalog+/– ontology; the semantics of PP-Datalog+/– arises as a direct combination of the semantics of Datalog+/– and that of the preference and probabilistic models. Relative to the probabilistic model, we have that $\Pr_{KB}(a) = p$ iff $p = \sum_{\lambda \in scn(M),\, O_\lambda \models a} \Pr_M(\lambda)$. We refer to the score-based preference relation induced by $\Pr_M$ as the *probabilistic preference relation* associated with KB, and denote it with $\succ_M$. Let $\succ^* = \otimes(\succ_M, \succ_P)$; we say that $KB \models a_1 \succ^* a_2$ iff (i) $O \models a_1$ and $O \models a_2$; and (ii) $a_1 \succ^* a_2$. Intuitively, the consequences of a knowledge base $KB = (O, P, M, \otimes)$ are computed in terms of the classical consequences of the Datalog+/– ontology O, and the preference combination operator yields a preference relation over pairs of atoms in $\mathcal{H}_{Ont}$.

Skyline and k-Rank Queries. In this paper, we are interested in skyline queries [13], a well-known class of queries that can be issued over preference-based formalisms, and the iterated computation of skyline answers that allows us to assign a *rank* to every atom. In the following, we focus on a specific kind of classical queries, called disjunctive atomic queries (DAQs), which are disjunctions of atoms.

Definition 4. Let $KB = (O, P, M, \otimes)$ be a PP-Datalog+/– ontology where $\mathcal{H}_{Ont}$ is the Herbrand base of O, $\succ^* = \otimes(\succ_M, \succ_P)$, $k \geqslant 0$, and $Q(\mathbf{X}) = q_1(\mathbf{X}_1) \vee \cdots \vee q_n(\mathbf{X}_n)$ be a DAQ. Then, a *skyline answer* to Q relative to $\succ^*$ is any θq_i entailed by O such that no θ' exists with $O \models \theta' q_j$ and $\theta' q_j \succ^* \theta q_i$, where θ and θ' are ground substitutions for the variables in $Q(\mathbf{X})$.

To obtain an ordered list of answers to queries under transitive relations, we adopt the *iterated skyline* approach proposed in [11], which simply assigns a *rank* to each answer by iterating the following procedure: (1) assign the rank k to the skyline answers; and (2) remove from consideration all answers of rank k, increment k by one, and go back to (1). As a generalization of classical top-k queries, we adopt here *k-rank queries*; answers to such queries are simply k-tuples of answers sorted by rank. In the following, we use *rank*$(a, \succ_M)$ to denote the rank assigned to atom a by relation $\succ_M$.

Intuitively, for DAQs, both kinds of answers can be seen as atomic consequences of O that satisfy the query: the skyline answers can be seen as sets of atoms that are not dominated by any other such atom, while k-rank answers are k-tuples sorted according to the preference relation. We refer to these as answers in *atom form*.

In the next section, we present algorithms for implementing two particular preference combination operators that, as we show through a series of properties, produce a new preference relation $\succ^*$ that is useful in answering k-rank queries that adequately

Algorithm 1: CombinePrefs$(\succ_M, \succ_P, t)$
Input: SPO $\succ_P$, score-based preference relation $\succ_M$ over $\mathcal{H}_{Ont}$, and $t \in [0, 1]$.
Output: Preference relation $\succ_t^* \subseteq \mathcal{H}_{Ont} \times \mathcal{H}_{Ont}$.

1. Initialize $\succ_t^*$ as a copy of $\succ_P$;
2. For every pair $(a_i, a_j) \in \succ_P$ do begin
3. if $score(a_j) - score(a_i) > t$ and changing (a_i, a_j) to (a_j, a_i) in $\succ_t^*$ does not introduce a cycle in its associated graph then
4. change (a_i, a_j) to (a_j, a_i) in $\succ_t^*$;
5. End;
6. Return *transitiveClosure*$(\succ_t^*)$.

Fig. 2. An algorithm for combining an arbitrary SPO with a weak order in the form of a score-based preference relation.

reflect both the initial preferences expressed by the user as well as the fact that higher probability answers are more desirable.

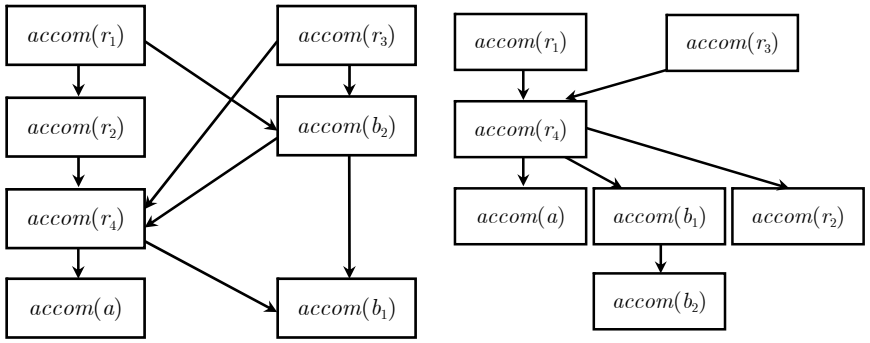
4 Two Preference Combination Operators

In this section, we study two particular instantiations of a preference combination operator, and study their semantic properties. Later, in Section 5, we show an algorithm that uses these operators for answering k-rank queries to PP-Datalog+/– ontologies in polynomial time in the data complexity (modulo the cost of computing probabilities with respect to the model M).

4.1 A General Preference Combination Operator

Algorithm CombinePrefs in Fig. 2 implements a preference combination operator using a value $t \in [0, 1]$ that allows the user to choose how much influence the probabilistic model has on the output preference relation; we use $\succ_t^*$ to denote the output relation. The algorithm iterates through all pairs of elements in $\succ_P$ and, if (i) $\succ_M$ disagrees with $\succ_P$, (ii) the difference in score is greater than t, and (iii) inserting the pair according to $\succ_M$ doesn't produce a cycle, then the pair is inserted in reverse order into the output; otherwise, the output contains the same pair as $\succ_P$. Finally, the algorithm outputs the transitive closure of this relation. Note that if $t = 0$, then the probability-induced relation has precedence; at the other end of the spectrum, if $t = 1$, we have that CombinePrefs$(\succ_M, \succ_P, t) = \succ_P$ (these properties are presented formally in Theorem 4). The following is an example of how the algorithm works.

Example 4. Let us return to the running example; Fig. 3 (top left) shows the result of running the CombinePrefs algorithm over the two preference relations and $t = 0$, while Fig. 3 (top right) corresponds to $t = 0.3$—note that, for clarity, the transitive closures are not shown in these figures. Fig. 3 (bottom) shows the result of computing the rank via the iterative computation of the skyline answers for the three preference relations. As we can see, the user's original preferences ($\succ_P$) are preserved to a greater



Rank	$\succ_P$	$\succ_0^*$	$\succ_{0.3}^*$
1	$[X/r_4]$	$[X/r_1]$, $[X/r_3]$	$[X/r_1]$, $[X/r_3]$
2	$[X/a]$, $[X/r_3]$, $[X/r_2]$	$[X/r_2]$, $[X/b_2]$	$[X/r_4]$
3	$[X/b_1]$,$[X/r_2]$	$[X/r_4]$	$[X/a]$, $[X/b_1]$, $[X/r_2]$
4	$[X/b_2]$	$[X/a]$, $[X/b_1]$	$[X/b_2]$

Fig. 3. Combinations of user preferences and probability induced preferences with the CombinePrefs algorithm (transitive closure not shown)—top left: $t = 0$; top right: $t = 0.3$. Bottom: answers to query $accom(X)$ according to their rank relative to the original preference relation (Fig. 1, left) and the two combinations.

extent in $\succ_{0.3}^*$ than in $\succ_0^*$: the answers in the former have a difference in rank of at most 1, whereas those in the latter have a difference of 2 (out of a total of 4) in multiple cases. This reflects the greater influence that $\succ_P$ has in the result for $t = 0.3$ as compared to $t = 0$. ∎

Theorem 1. *Let $\succ_P$ be an SPO and $\succ_M$ be a score-based preference relation, and let S be the time required to compute the score of any atom with respect to $\succ_M$. Algorithm* CombinePrefs *runs in time $O(poly(|\succ_P|) \cdot S)$.*

Proof sketch. The `for` loop in line 2 of CombinePrefs, which involves the creation of the output relation except for the transitive closure, is carried out $|\succ_P|$ times; in each iteration, the score of two atoms is computed. Each iteration of the loop can be done in time $O(poly(|\succ_P|) \cdot S)$ and, since $\prec^*$ right before executing line 7 has the same size as $\succ_P$, the transitive closure can also be computed in time $O(poly(|\succ_P|)$. □

For PP-Datalog+/– ontologies, the cost S in Theorem 1 refers to the cost of computing probabilities relative to model M; therefore, this cost could range from polynomial time (such as in approximation algorithms, or tractable models like polytrees [14] or tractable Markov logic [15]) to #P-complete (such as in general Markov logic [16] or Bayesian networks). In particular, since $\succ_P$ describes a preference relation over the consequences of an ontology, we have the following corollary.

Corollary 1. *Let $KB = (O, P, M,$ CombinePrefs$)$ be a guarded PP-Datalog+/– ontology such that $\mathcal{H}_{Ont}$ is the Herbrand base for O, $\succ_P$ be an SPO, and $\succ_M$ be a score-based preference relation defined over $\mathcal{H}_{Ont}$, and Q be a conjunctive query. Then, if*

$\Pr_M(a)$ *can be computed or approximated in polynomial time in the data complexity for any* a *such that* $KB \models a$*, Algorithm* CombinePrefs *runs in polynomial time in the data complexity.*

Proof sketch. For guarded Datalog+/– ontologies, the necessary finite part of the chase required to answer a query is of polynomial size in the data complexity [6], and it is precisely this structure that gives rise to $\succ_P$. By the hypothesis that probabilities are computed in polynomial time and the result in Theorem 1, the statement follows. □

In the following theorem, we show that CombinePrefs satisfies the definition of a preference combination operator (cf. Definition 2).

Theorem 2. *Let* $\succ_P$ *be an SPO,* $\succ_M$ *be a score-based preference relation,* $t \in [0, 1]$*, and* $\succ_t^* = \mathsf{CombinePrefs}(\succ_M, \succ_P, t)$*. Then, for any value of* t*, we have:*

(i) $\succ_t^*$ *is an SPO; and*

(ii) If $a_1 \succ_P a_2$ *and* $a_1 \succ_M a_2$*, then* $a_1 \succ_t^* a_2$*.*

Proof sketch. (i) We must show that $\succ^*$ is antisymmetric, irreflexive, and transitive. Let $\succ'$ be the intermediate result of the CombinePrefs operator just before computing the transitive closure. By hypothesis, $\succ_U$ is irreflexive and, since $\succ'$ cannot contain any new self-edges, this property is preserved in $\succ'$. Similarly, since $\succ_U$ is antisymmetric and CombinePrefs checks for cycles before changing edges, there will never be two edges (a, b) and (b, a)—antisymmetry therefore vacuously holds. Finally, by construction, $\succ^*$ is the transitive closure of $\succ'$; since this operation cannot add cycles, we conclude that $\succ^*$ is an SPO.

(ii) A necessary condition for CombinePrefs to change the order of a pair in $\succ_U$ is that the pair in $\succ_M$ be reversed. Since by hypothesis this is not the case, the statement follows. □

We now study the properties enjoyed by the output of the CombinePrefs algorithm. The following theorem states that the output for $t = 0$ is invariant to changes in the scores assigned as long as the order is the same.

Theorem 3. *Let* $\succ_P$ *be an SPO, and* $\succ_M$ *and* $\succ_{M'}$ *be score-based preference relations. If* $\succ_M = \succ_{M'}$*, then* $\mathsf{CombinePrefs}(\succ_M, \succ_P, 0) = \mathsf{CombinePrefs}(\succ_{M'}, \succ_P, 0)$.

Proof sketch. If $t = 0$, the algorithm checks (in line 3) if pairs (a_i, a_j) in $\succ_P$ are such that $\Pr(a_j) > \Pr(a_i)$. Since by hypothesis, we know that $\succ_M = \succ_{M'}$, it must be the case that $\Pr(a_j) > \Pr(a_i)$ relative to $\succ_M$ iff this is the case relative to $\succ_{M'}$, and thus the outputs in both cases must be identical. □

This is an important property, since it implies that approximation algorithms can be used to compute the probability values that induce the $\succ_M$ relation—as long as the relative order of the atoms is guaranteed to be correct, there is no need to compute the exact values when $t = 0$.

The following theorem shows the behavior of the operator for the extreme values of parameter t.

Theorem 4. *Let $\succ_P$ be an SPO, and $\succ_M$ be a score-based preference relation:*

(i) $rank(a, \mathsf{CombinePrefs}(\succ_M, \succ_P, 0)) = rank(a, \succ_M)$*, for any atom* a*.*

(ii) $\mathsf{CombinePrefs}(\succ_M, \succ_P, 1) = \succ_P$*.*

Proof sketch. (i) When $t = 0$, the output relation is the result of comparing all pairs in $\succ_P$ and replacing them with the order imposed on their elements by $\succ_M$. Thus, given that $\succ_P$ is transitive, the sequence of atoms giving rise to the rank of each atom is reordered in the resulting relation according to the rank of each element in $\succ_M$, and the statement follows.

(ii) Direct consequence of the condition in line 3 of the algorithm: since the difference in probabilities can never be greater than 1, the output relation is a copy of $\succ_P$. □

Finally, the following theorem studies a property that is analogous to the "Sensitivity postulates" presented in [17], [18, pp. 12–14,63–64]. Note, however, that these postulates were designed to combine two score-based preference relations, whereas our setting is more general—the "Faithfulness" postulate in [18] is already guaranteed by our definition of preference combination operator.

Theorem 5. *Let $KB = (O, P, M, \mathsf{CombinePrefs})$ be a PP-Datalog+/– ontology where $\mathcal{H}_{Ont}$ is the Herbrand base of O, Q be a DAQ, $k \geqslant 0$, and $\succ_t^* = \mathsf{CombinePrefs}(\succ_M, \succ_P, t)$, with $t \in [0, 1]$. For every atom a that does not belong to any k-rank answer to Q over KB relative to $\succ_t^*$, there exists a probabilistic model M' and $t' \in [0, 1]$ such that a belongs to some k-rank answer to Q over $KB' = (O, P, M', \mathsf{CombinePrefs})$ relative to $\succ_{t'}' = \mathsf{CombinePrefs}(\succ_{M'}, \succ_P, t')$.*

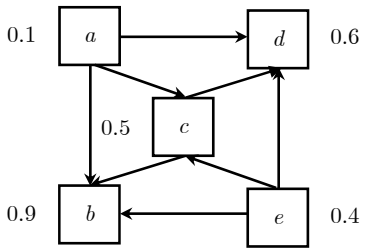
Proof sketch. By hypothesis, there exist atoms $\{a_1, ..., a_n\}$ such that $a_i \succ_t^* a$. Now, let $t' = 0$ and set the probabilities in M' so that $\Pr_{M'}(a) > \Pr_{M'}(a_i)$, for $1 \leqslant i \leqslant n$. By construction, we now have that $a \succ_{t'}' a_i$, and the statement follows. □

In the next section, we study a different kind of combination operator based on a tradeoff between generality and the properties that can be proved about the result.

4.2 An Operator Based on Rank

The following example shows a shortcoming of the CombinePrefs operator in that it fails to exhibit a kind of monotonicity that might be expected when the two input relations disagree on a given pair (a_1, a_2). In particular, if the operator chooses the order imposed by $\succ_M$, then it may not continue to do so for smaller values of t. On the other hand, if the operator chooses the order imposed by $\succ_P$, then it may also fail to do so for greater values of t.

Example 5. Consider the preference relation in Fig. 4 (left), where the directed edges between nodes describe a relation $\succ_P$ and the numbers beside each node define score-based relation $\succ_M$. The table on the right specifies what happens when the two are input to the CombinePrefs algorithm and the edges are inspected in the specified order, for two values of t. Note row 4, where edge (a, c) becomes (c, a) under $t = 0.1$ but stays unaltered under $t = 0.05$, and row 6, where edge (a, b) stays unaltered under $t = 0.05$ but becomes (b, a) under $t = 0.1$. ■



Order	Edge	$t = 0.1$	$t = 0.05$
1	(c, d)	No (t)	Yes
2	(d, e)	No (cycle)	Yes
3	(c, e)	No (t)	Yes
4	(a, c)	Yes	No (cycle)
5	(a, d)	Yes	Yes
6	(a, b)	Yes	No (cycle)
7	(c, b)	Yes	No (cycle)
8	(e, b)	Yes	No (cycle)

Fig. 4. User preferences for Example 5 (left), and the results of applying Algorithm CombinePrefs for two different values of t (right). In the table, "yes" and "no" means whether or not the edge is reversed in the result—in the negative cases, the reason is given in parenthesis.

Algorithm 2: CombinePrefsRank($\succ_M, \succ_P, f$)
Input: SPO $\succ_P$, score-based preference relation $\succ_M$ over $\mathcal{H}_{Ont}$, and integer binary function f such that $f(x, y) \in [x, y]$.
Output: Score-based preference relation $\succ_f^{rank} \subseteq \mathcal{H}_{Ont} \times \mathcal{H}_{Ont}$.

1. Initialize $\succ_f^{rank}$ as an empty preference relation over $\subseteq \mathcal{H}_{Ont} \times \mathcal{H}_{Ont}$;
2. Initialize *ranks* as an empty mapping from preference relations to integers;
3. Compute the ranks of each atom a relative to $\succ_P$ and $\succ_M$, and store them in *ranks*;
4. For every atom a in $dom(ranks)$ do
5. add a to $\succ_f^{rank}$ with score $f(ranks(a, \succ_P), ranks(a, \succ_M))$;
6. Return $\succ_f^{rank}$.

Fig. 5. Algorithm for combining an SPO with a score-based preference relation, based on rank.

As shown in Example 5, the cause of this kind of unpredictability is the potential presence of cycles when merging preference relations. Algorithm 2 (Fig. 5) avoids this issue by combining the two relations into a score-based one based on the rank of each atom relative to each input relation. The algorithm takes as input an integer binary function that combines the two ranks and assigns it to the atom in the output relation—this function can be as simple as `min`, `max`, or `avg`, or a more complex function that takes into account how much the user would like to base the new rank on one relation or the other. A detailed treatment of such functions is outside the scope of this paper.

The following theorem states that CombinePrefsRank satisfies the conditions for being a preference combination operator for certain functions.

Theorem 6. *Let $\succ_P$ be an SPO, $\succ_M$ be a score-based preference relation, f be an integer binary function such that $f(x, y) \in [x, y]$, and $\succ_f^{rank}$= CombinePrefsRank($\succ_M$, $\succ_P$, f). Then, we have:*

(i) $\succ_f^{rank}$ is an SPO; and

(ii) If $a_1 \succ_P a_2$, $a_1 \succ_M a_2$, and $f \in \{min, max, avg\}$, then $a_1 \succ_f^{rank} a_2$.

Proof sketch. (i) Direct consequence, since $\succ_f^{rank}$ is a score-based preference relation.

(ii) Let $r_1^P = \mathit{rank}(a_1, \succ_P)$, $r_2^P = \mathit{rank}(a_1, \succ_P)$, $r_1^M = \mathit{rank}(a_1, \succ_M)$, and $r_2^M = \mathit{rank}(a_1, \succ_M)$. The hypotheses imply that $r_1^P < r_2^P$ and $r_1^M < r_2^M$. Clearly, $f(r_1^P, r_1^M) < f(r_2^P, r_2^M)$ for $f \in \{\mathit{min}, \mathit{max}, \mathit{avg}\}$, and thus $a_1 \succ_f^{\mathit{rank}} a_2$. □

Analyzing the proof of Theorem 6, we can see that the result also holds for variations of the functions considered—as long as the condition $f(r_1^P, r_1^M) < f(r_2^P, r_2^M)$ is guaranteed, the result holds.

Another property of Algorithm CombinePrefsRank that can be shown is the analogous of Theorem 3; since the hypothesis implies that ranks relative to $\succ_M$ and $\succ_{M'}$ are equal, the result clearly holds. Finally, the following theorem discusses properties related to the postulates from [18] for this algorithm:

Theorem 7. *Let $KB = (O, P, M, \mathsf{CombinePrefs})$ be a PP-Datalog+/– ontology where $\mathcal{H}_{Omt}$ is the Herbrand base of O, Q be a DAQ, $k \geqslant 0$, and $\succ_t^* = \mathsf{CombinePrefs}(\succ_M, \succ_P, t)$, with $t \in [0, 1]$. Then:*

(i) Let $f \in \{\mathit{min}, \mathit{max}, \mathit{avg}\}$, M' be a probabilistic model such that for some ground atom a we have $\Pr_{KB'}(a) \leqslant \Pr_{KB}(a)$ and $\Pr_{KB'}(a') = \Pr_{KB}(a')$ for every ground atom $a' \neq a$; let $KB' = (O, P, M', \mathsf{CombinePrefsRank})$. If a does not belong to any k-rank answer to Q over KB relative to $\succ_t^*$, then a does not belong to any k-rank answer to Q over KB' relative to $\succ_t' = \mathsf{CombinePrefsRank}(\succ_{M'}, \succ_P, t)$.

(ii) Given the setup in part (i), if a belongs to some k-rank answer to Q over KB relative to $\succ_t^*$ and $\Pr_{KB'}(a) \geqslant \Pr_{KB}(a)$, then a also belongs to some k-rank answer to Q over KB' relative to $\succ_t' = \mathsf{CombinePrefsRank}(\succ_{M'}, \succ_P, t)$.

(iii) For every atom a that does not belong to any k-rank answer to Q over KB relative to $\succ_t^*$, there exists a probabilistic model M' and $t' \in [0, 1]$ such that a belongs to some k-rank answer to Q over $KB' = (O, P, M', \mathsf{CombinePrefs})$ relative to $\succ_{t'}' = \mathsf{CombinePrefsRank}(\succ_{M'}, \succ_P, t')$.

Proof sketch. (i) $\Pr_{KB'}(a) \leqslant \Pr_{KB}(a)$ implies that $\mathit{rank}(a, \succ_{M'}) \leqslant \mathit{rank}(a, \succ_M)$, and the result from Theorem 6 can be used to show the result.

(ii) Analogous to (i): $\Pr_{KB'}(a) \geqslant \Pr_{KB}(a)$ implies that $\mathit{rank}(a, \succ_{M'}) \geqslant \mathit{rank}(a, \succ_M)$.

(iii) Using $f = \mathit{min}$, the rest of the proof is analogous to that of Theorem 5. □

We conclude this section with an analysis of the running time of Algorithm CombinePrefsRank.

Theorem 8. *Let $\succ_P$ be an SPO and $\succ_M$ be a score-based preference relation, and let S be the time required to compute the score of any atom with respect to $\succ_M$. Algorithm* CombinePrefsRank *runs in time $O(\mathit{poly}(|\succ_P|) \cdot S)$.*

Proof sketch. Ranks for each atom relative to $\succ_P$ can be computed in polynomial time in the data complexity by a linear scan of each relation; for $\succ_M$, the cost is $O(\mathit{poly}(|\succ_P|) \cdot S)$. Using these values, the output relation can also be built within this time bound. □

5 Answering k-Rank Queries

As discussed above, in this paper, we are interested in computing the rank of the answers to queries by means of the iterated computation of its skyline answers [13].

Algorithm 3: k-Rank$(KB = (O, P, M, \otimes), Q, k)$
Input: Guarded PP-Datalog+/– ontology KB, DAQ $Q(\mathbf{X})$, and $k \geqslant 0$.
Output: k-rank answer $\langle a_1, ..., a_{k'} \rangle$ to Q, with $k' \leqslant k$.

1. Initialize Res as an empty vector of ground atoms;
2. Set $\succ^*$:= $\otimes(\succ_M, \succ_P)$;
3. C:= *computeChase*(O, Q);
4. i:= k;
5. While $i > 0$ do begin
6. $\quad S$:= *computeSkyline*$(C, Q, \succ^*)$;
7. $\quad$ Append S to Res (in arbitrary order);
8. $\quad$ Remove S from C;
9. $\quad i$:= $i - |S|$;
10. End;
11. Return $truncate(Res, k)$.

Fig. 6. An algorithm for computing a k-rank answer to DAQ Q relative to the composition of $\succ_P$ and $\succ_M$

We now present a general algorithm to do so, and then analyze its correctness as well as its running time when used in conjunction with either the CombinePrefs or CombinePrefsRank algorithms.

The k-Rank Algorithm. The algorithm in Fig. 6 begins by computing the combination of the two preference relations in the PP-Datalog+/– ontology and the necessary finite part of the chase relative to Q. The main `while` loop iterates through the process of computing the skyline answers to Q relative to this new relation by using a *computeSkyline* subroutine (which can be implemented by means of a linear-time scan of C), updating the result by appending these answers in arbitrary order, and removing the atoms in the result from C. Once the loop is finished, the algorithm returns the first k results, since the last iteration might add superfluous elements.

Example 6. Consider the running example, with $Q = accom(X)$, $k = 5$, and $t = 0.3$. Fig. 3 (top right) depicts the result of the CombinePrefs algorithm, and Fig. 3 (bottom) (last column) shows the ranks associated with the different answers to Q. One possible k-rank answer to Q (in atom form) as computed by the algorithm is thus: $\langle accom(r_1), accom(r_3), accom(r_4), accom(a), accom(b_1) \rangle$. ■

The following theorem proves the correctness of the k-Rank algorithm, and shows that it runs in polynomial time under certain conditions.

Theorem 9. *Let $KB = (O, P, M, \otimes)$, with $O = (D, \Sigma)$, be a PP-Datalog+/– ontology, Q be a DAQ, and $k \geqslant 0$. Then,*

(i) Algorithm k-Rank correctly computes a k-rank answer to Q over KB; and

(ii) if O is a guarded Datalog+/– ontology and $\otimes$ can be computed in $O(poly(D) \cdot S)$, the running time of k-Rank is $O(poly(D) \cdot S)$, where S is the cost of computing $\Pr_M(a)$ for any atom a such that $O \models a$.

Proof sketch. (i) Correctness is a consequence of the direct application of the definition of k-rank: the `while` loop in line 4 iteratively computes the skyline answers to Q by means of a subroutine, adds these results to the output in arbitrary order, and removes them from consideration. Line 10 ensures that at most k results are returned.

(ii) Since O is guarded, answers to Q can be computed in polynomial time in the data complexity [6]. The $\otimes$ operator is computed in time $O(|\succ_P| \cdot S)$, and *computeSkyline* can be computed in polynomial time by a linear-time scan of the chase structure for Q (of polynomial size by hypothesis), and the results can be removed by another scan. □

As a consequence of Theorem 9 and Corollary 1, we can conclude that if probabilities in M can be computed or approximated in polynomial time (in the data complexity), then so can k-rank answers.

6 Related Work

The study of preferences has received much attention in many areas of study such as philosophy, choice theory, and certain areas of the social sciences (such as social choice). In computer science, the most relevant to our work is their incorporation into query answering mechanisms. To our knowledge, the current state of the art in this respect is centered around relational databases, and no other work to date combines general preferences with those induced by probability assignments.

The seminal work in preference-based query answering is that of [19], in which the SQL language is extended to incorporate user preferences, showing that the resulting formalism could be translated into the domain relational calculus. In [11], preference formulas are introduced as a logical formalism that allows an embedding of preference specifications into SQL through a *winnow* operator parameterized by a preference formula; the winnow operator is a generalization of the *skyline* operator, first introduced in [13]. Perhaps closest to our approach (except for the probabilistic model) is that of preference Datalog programs [20], which are a restriction of preference logic programs [21] that contain no uninterpreted function symbols and extend classical Datalog with constructs for determining which predicates must be optimized along with the optimization criteria (i.e., the set of preferences). For a recent survey on preference-based query answering, see [22].

With respect to probabilistic preferences, there are several works that have been developed in the last few years. Probabilistic skylines were introduced in [23] (and later also studied in [24]; the related *stochastic* skyline is introduced in [25]), where the authors tackle the problem of computing the probability of an element belonging to the skyline, rather than using the probability values for the purpose of ranking, as proposed in our work. In [17, 26], the authors focus on a more specific version of our problem, since they assume that tuples receive both a score and a probability value, and thus the two preference relations in question are score-based; furthermore, the approach adopted in their work is based on possible worlds, whereas, here, we focus on the use of probabilities solely as vehicles for ranking.

Preference revision is also closely related to our approach. The work of [27] tackles the problem of modeling preference change, studying axioms and postulates for the

revision and contraction of a set of sentences specifying preferences by an input preference, in the style of belief revision theory; addition and subtraction of elements is also studied. The main difference with our work is that the author does not study algorithms nor complexity of deriving a revised relation, but rather focuses on a set of postulates based on those from belief revision and centered on obtaining a relation that incorporates the new preference with minimal change. The work of [28] is also related in that it addresses the problem of *query modification* in preference-based query answering in relational DBs. The focus is, however, on three specific combination mechanisms: union, prioritized, and Pareto composition, and on the study of the preservation of properties of different kinds of relations under these combinations.

Finally, social choice theory [29] is also relevant, since it seeks to combine preferences to produce a new preference relation; methods range from those using score-based relations (e.g., approval voting) to ones using more general ones (e.g., ranked pairs). In particular, the work of [30] studies possibility/impossibility results generalizing properties such as Arrow's theorem to the case in which incomparable elements exist. The study of potential applications of voting methods to query answering, especially in relation to the Social Web, is the topic of ongoing work.

7 Summary and Outlook

In this work, we have presented an extension of the Datalog+/– family of ontology languages for preference-based query answering under uncertainty. This task has recently attained central importance due to its relation with the Social (Semantic) Web. The main focus of this work has been on defining preference combination operators, which produce a preference relation, given a general SPO and a score-based preference relation. We have proposed two specific algorithms for such an operator, and analyzed their semantic and computational properties. Finally, we have studied a basic algorithm for answering k-rank queries, and showed that under certain conditions, k-rank queries can be answered in polynomial time in the data complexity, which is the same complexity as answering traditional preference-based queries in relational DBs. Current and future work involves implementing and testing the PP-Datalog+/– framework, developing other combination operators, and studying specific preference specification mechanisms and probabilistic models.

Acknowledgments. This work was supported by the UK EPSRC grant EP/J008346/1 ("PrOQAW"), ERC grant 246858 ("DIADEM"), and a Yahoo! Research Fellowship.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Sci. Am. 284(5), 34–43 (2002)
2. Jung, J.C., Lutz, C.: Ontology-based access to probabilistic data with OWL QL. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 182–197. Springer, Heidelberg (2012)
3. Lukasiewicz, T., Martinez, M.V., Orsi, G., Simari, G.I.: Heuristic ranking in tightly coupled probabilistic description logics. In: Proc. of UAI, AUAI, pp. 554–563 (2012)

4. Noessner, J., Niepert, M.: ELOG: A probabilistic reasoner for OWL EL. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 281–286. Springer, Heidelberg (2011)
5. Finger, M., Wassermann, R., Cozman, F.G.: Satisfiability in EL with sets of probabilistic ABoxes. In: Proc. of DL (2011)
6. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
7. Gottlob, G., Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Query answering under probabilistic uncertainty in Datalog+/– ontologies. AMAI, 1–36 (2013)
8. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. of ICALP, pp. 73–85 (1981)
9. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. of KR, pp. 70–80. AAAI Press (2008)
10. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Preference-based query answering in Datalog+/– ontologies. In: Proc. of IJCAI (to appear, 2013)
11. Chomicki, J.: Preference formulas in relational queries. TODS 28(4), 427–466 (2003)
12. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Consistent answers in probabilistic Datalog+/– ontologies. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 156–171. Springer, Heidelberg (2012)
13. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. of ICDE, pp. 421–430. IEEE Computer Society (2001)
14. Kim, J., Pearl, J.: A computational model for causal and diagnostic reasoning in inference systems. In: Proc. of IJCAI, pp. 190–193 (1983)
15. Domingos, P., Webb, W.: A tractable first-order probabilistic logic. In: Proc. of AAAI (2012)
16. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. 62(1/2), 107–136 (2006)
17. Zhang, X., Chomicki, J.: Semantics and evaluation of top-k queries in probabilistic databases. Distributed and Parallel Databases 26, 67–126 (2009)
18. Zhang, X.: Probabilities and Sets in Preference Querying. PhD thesis, University at Buffalo, State University of New York (2010)
19. Lacroix, M., Lavency, P.: Preferences: Putting more knowledge into queries. In: Proc. of VLDB, vol. 87, pp. 1–4 (1987)
20. Govindarajan, K., Jayaraman, B., Mantha, S.: Preference queries in deductive databases. New Generation Computing 19(1), 57–86 (2001)
21. Govindarajan, K., Jayaraman, B., Mantha, S., et al.: Preference logic programming. In: Proc. of ICLP, pp. 731–745 (1995)
22. Stefanidis, K., Koutrika, G., Pitoura, E.: A survey on representation, composition and application of preferences in database systems. TODS 36(3), 19:1–19:45 (2011)
23. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: Proc. of VLDB, pp. 15–26 (2007)
24. Atallah, M.J., Qi, Y.: Computing all skyline probabilities for uncertain data. In: Proc. of PODS, pp. 279–287. ACM (2009)
25. Lin, X., Zhang, Y., Zhang, W., Cheema, M.: Stochastic skyline operator. In: Proc. of ICDE, pp. 721–732 (2011)
26. Soliman, M., Ilyas, I., Chen-Chuan Chang, K.: Top-k query processing in uncertain databases. In: Proc. of ICDE, pp. 896–905 (2007)
27. Hansson, S.O.: Changes in preference. Theory and Decision 38, 1–28 (1995)
28. Chomicki, J.: Database querying under changing preferences. AMAI 50, 79–109 (2007)
29. Gaertner, W.: A primer in social choice theory: Revised edition. Oxford Univ. Press (2009)
30. Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Aggregating partially ordered preferences. J. Log. Comput. 19(3), 475–502 (2009)