Answering Ontological Ranking Queries Based on Subjective Reports

Thomas Lukasiewicz[1], Maria Vanina Martinez[1], Cristian Molinaro[2], Livia Predoiu[1], and Gerardo I. Simari[1]

[1] Department of Computer Science, University of Oxford, UK
{thomas.lukasiewicz,vanina.martinez,livia.predoiu, gerardo.simari}@cs.ox.ac.uk
[2] DIMES, Università della Calabria, Italy
cmolinaro@dimes.unical.it

Abstract. The use of preferences in query answering, both in traditional databases and in ontology-based data access, has recently received much attention, due to its many real-world applications. In this paper, we tackle the problem of query answering in Datalog+/– ontologies subject to the querying user's preferences and a collection of subjective reports (i.e., scores for a list of features) of other users, who have their own preferences as well. All these pieces of information are combined to rank the query results. We first focus on the problem of ranking atoms in a database by leveraging reports and customizing their content according to the user's preferences. Then, we extend this approach to deal with ontological query answering using provenance information. Though the general problem is shown to have an exponential-time data complexity upper bound, we propose a special case that has polynomial time data complexity.

1 Introduction

The use of preferences in query answering, both in traditional databases and in ontology-based data access, has recently received much attention due to its many real-world applications. In particular, in recent times, there has been a huge change in the way data is created and consumed, and users have largely moved to the Social Web, a system of platforms used to socially interact by sharing data and collaborating on tasks.

In this paper, we tackle the problem of preference-based query answering in Datalog+/– ontologies assuming that the user must rely on subjective reports to get a complete picture and make a decision. This kind of situation arises all the time on the Web; for instance, when searching for a hotel, users provide some basic information and receive a list of answers to choose from, each associated with a set of subjective reports (often called reviews) written by other users to tell everyone about their experience. The main problem with this setup, however, is that users are often overwhelmed and frustrated, because they cannot decide which reviews to focus on and which ones to ignore, since it is likely that, for instance, a very negative (or positive) review may have been produced on the basis of a feature that is completely irrelevant to the querying user.

We study a formalization of this process and its incorporation into preference-based query answering in Datalog+/– ontologies, proposing a framework for user-tailored query answers on the basis of the ontology reports and users' preferences.

The following are the main contributions of this paper:

- We present an approach to preference-based query answering in Datalog+/– ontologies, given a collection of subjective reports containing scores for a list of features.
- We first propose a basic approach to rank atoms in the ontology's database, which combines reports, their authors' preferences among the features, and the querying user's preferences among the features.
- We then extend our framework to ontologies with dependencies and propose a method for dealing with query answering by leveraging provenance information for propagating reports from the database atoms to newly inferred ones.
- As we are using a kind of "how"-provenance modeled using semirings, we can map the general semiring to more specific ones that capture different ways in which the information contained in reports can be leveraged, Though the general case is exponential, we explore one such mapping for which ranking query answers can be done in polynomial time data complexity.

The rest of this paper is organized as follows. In Section 2, we provide some preliminaries on Datalog+/– and the used preference models. Section 3 then defines subjective reports and proposes a basic framework to deal with simple (no dependencies) Datalog+/– ontologies. Section 4 considers ontologies with dependencies and addresses query answering. Section 5 discusses related work, and Section 6 concludes.

2 Preliminaries

First, we briefly recall some basics on Datalog+/– [8], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple-generating dependencies (TGDs), negative constraints, the chase, and ontologies in Datalog+/–.

We assume (i) an infinite universe of *(data) constants* dom (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 $dom \cup \Delta_N$, with every symbol in Δ_N following all symbols in dom. 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*) 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 dom.

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. Satisfaction of TGDs are defined via *homomorphisms*, which are mappings $\mu\colon dom \cup \Delta_N \cup \mathcal{V} \rightarrow dom \cup \Delta_N \cup \mathcal{V}$ such that (i)

$c \in dom$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in dom \cup \Delta_N$, and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. A TGD 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. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. 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 *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 without 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. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples a such that $a \in Q(B)$ 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 [2], even when the schema and TGDs are fixed [7]. Decidability of query answering for the guarded case follows from a bounded tree-width property. The data complexity of query answering in this case is P-complete. We refer to [8] for their details.

The chase algorithm for a database D and a set of TGDs Σ consists of an exhaustive application of the TGDs [8] in a breadth-first (level-saturating) fashion, which outputs a (possibly infinite) chase for D and Σ. The (possibly infinite) chase relative to TGDs is a *universal model*, i.e., there exists a homomorphism from $chase(D, \Sigma)$ onto every $B \in mods(D, \Sigma)$ [8]. 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 EGDs Σ_E, and a set of negative constraints Σ_{NC}. In order to ensure decidability and tractability of query answering, we assume that EGDs are *separable*, which means that the interaction between TGDs and EGDs is controlled—this condition can be ensured by the syntactic criterion of *non-conflicting keys*; for details on these conditions, we refer the reader to [8]. Finally, we say that KB is *guarded* iff Σ_T is guarded. The following example illustrates a simple Datalog+/– ontology, used in the sequel as a running example.

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

$$D = \{id_1 : \textit{apthotel}(h_1), \quad id_2 : \textit{hotel}(h_2), \quad id_3 : \textit{bb}(bb_1), \quad id_4 : \textit{hostel}(hs_1)\},$$

$$\begin{aligned}\Sigma = \{&\sigma_1 : \textit{hotel}(H) \rightarrow \textit{accom}(H), && \sigma_2 : \textit{apartment}(A) \rightarrow \textit{accom}(A),\\ &\sigma_3 : \textit{bb}(B) \rightarrow \textit{accom}(B), && \sigma_4 : \textit{apthotel}(A) \rightarrow \textit{hotel}(A),\\ &\sigma_5 : \textit{apthotel}(A) \rightarrow \textit{apartment}(A), && \sigma_6 : \textit{hostel}(H) \rightarrow \textit{accom}(H)\}.\end{aligned}$$

This ontology models a very simple accommodation booking domain, which can be used as the underlying model in an online system. Accommodations can be either hotels, apartments, B&Bs, hostels, or aparthotel. Moreover, an aparthotel is both a hotel and an apartment. Database D provides some example instances. ■

Preference Models. A *preference relation* $\succ$ over a set S is a strict partial order (SPO) over S, i.e., an irreflexive and transitive binary relation over S—we consider these to be the minimal requirements for a useful preference relation. If $a \succ b$, we say that a is *preferred to* b. The *indifference relation* $\sim$ induced by $\succ$ is defined as follows: for any $a, b \in S, a \sim b$ iff $a \not\succ b$ and $b \not\succ a$.

A *stratification* of S relative to $\succ$ is an ordered sequence $\langle S_1, \ldots, S_k \rangle$, where each S_i is a maximal subset of S such that for every $a \in S_i$ there is no $b \in \bigcup_{j=i}^{k} S_j$ with $b \succ a$. Intuitively, S_1 contains the most preferred elements in S relative to $\succ$, i.e., those elements $a \in S$ for which there is no $b \in S$ such that $b \succ a$. Then, S_2 contains the most preferred elements of $S - S_1$, and so on so forth. Notice that a stratification always exists and is unique—moreover, it is a partition of S. Each S_i is called a *stratum*.

3 A Framework for Ranking Atoms based on Subjective Reports

In this section, we propose a framework that allows us to produce a ranking of atoms in a knowledgebase by leveraging subjective reports. For now, we restrict our attention to the case where the set of dependencies Σ is empty—we generalize our framework to deal with dependencies and ontological query answering in the next section.

More specifically, we consider the following setting. We are given a Datalog+/– ontology $KB = (D, \Sigma)$, with $\Sigma = \emptyset$, where ground atoms in D are associated with *reports* provided by *observers*. A report is an evaluation for an entity of interest that specifies a score for different *features*, which are the dimensions along which entities can be evaluated. Also, each observer expresses a preference relation over the features. Finally, a user looking at the ontology also has her own preference relation over the features and wishes to rank the atoms in the ontology on the basis of the reports, her preferences, as well as the observers' preferences.

Example 2. Consider $KB = (D, \emptyset)$ where D is the database from Example 1—recall that predicate symbols refer to different types of accommodations. Accommodations might be rated relative to location, cleanliness, price, breakfast, and internet; in this case, these would be the features of interest. Observers can leave reports for the accommodations in the knowledgebase, with each report specifying one score per feature. Each observer has also a preference relation over the features specifying their importance from the observer's point of view. A user, who also has her own preference relation over the features, wishes to rank the atoms in the knowledgebase (that is, all the atoms in D), taking into account all the elements discussed above. ∎

Features and Reports. We assume the existence of an ordered sequence of *features* $\langle f_1, \ldots, f_m \rangle$, each of which has a domain $dom(f_i) = [0, 1] \cup \{-\}$. We use $\mathcal{F}$ to denote the set of features $\{f_1, \ldots, f_m\}$.[3] The values in a feature domain are possible scores that can be given for the feature, with "$-$" meaning that no score is given. The binary relation $>$ over $dom(f_i)$ is defined in the standard way with the additional requirement that $v > -$ for every $v \in [0, 1]$.

[3] Throughout this section, we consider a single set of features shared by all predicate symbols; this limitation is removed in the following section.

A *report* is an element of $dom(f_1) \times \cdots \times dom(f_m)$. We use $Reports$ to denote the set of all possible reports, i.e., $Reports = dom(f_1) \times \cdots \times dom(f_m)$. A report given by a certain observer for a ground atom A specifies m "scores" representing an evaluation of A for each feature.

Multiple reports, coming from different observers, can be associated with the same ground atom—each report expresses the rating given to the ground atom by a specific observer. Thus, we assume we have N observers and N partial functions $\Gamma_i : D \rightarrow Reports$, called *report functions*, for $1 \leqslant i \leqslant N$. Given an observer $1 \leqslant i \leqslant N$ and a ground atom $A \in D$, if $\Gamma_i(A)$ is defined, it denotes the report given by observer i to A; if undefined, this means that observer i has no report for A (note that we assume that an observer can have at most one report for a given ground atom).

Example 3. Consider the accommodation booking domain of Example 2. Suppose the features with respect to which accommodations are rated are $\langle$*location*, *cleanliness*, *price*, *breakfast*, *internet*$\rangle$; in the following, we abbreviate these features as *loc*, *cl*, *pri*, *br*, *net*, respectively. Suppose we have 3 observers and that the Γ_i's are as follows:

	loc	*cl*	*pri*	*br*	*net*
$\Gamma_1(id_1)$	0.8	0	0.4	–	0.6
$\Gamma_1(id_2)$	0.6	0.7	0.4	–	0.2
$\Gamma_1(id_3)$	0.9	1	0.6	0.5	0.1
$\Gamma_1(id_4)$	1	0.1	0.2	0.3	0.4

	loc	*cl*	*pri*	*br*	*net*
$\Gamma_2(id_1)$	0.7	0.5	0.3	0.1	0.5
$\Gamma_2(id_2)$	0.5	0.2	0.2	0.3	0.3
$\Gamma_2(id_3)$	0.4	0.1	0.9	0.5	0.7
$\Gamma_2(id_4)$	0.3	0.7	–	0.4	0.6

	loc	*cl*	*pri*	*br*	*net*
$\Gamma_3(id_1)$	0.5	0	0.4	–	0.1
$\Gamma_3(id_2)$	0.7	–	0.6	0.9	0.4
$\Gamma_3(id_3)$	0.9	0.3	0.3	0.1	0
$\Gamma_3(id_4)$	0.8	0.5	0.6	–	0.2

For instance, Γ_1 says that the first observer assigned the report $\langle 0.8, 0, 0.4, -, 0.6\rangle$ to atom id_1, which means that the score given to id_1 relative to *location* (resp., *cleanliness*, *price*, *internet*) is 0.8 (resp., 0, 0.4, 0.6), while no score is given for *breakfast*. ∎

Comparing Reports. Given a report $r = \langle s_1, \ldots, s_m\rangle$, we use $r[i]$ to refer to s_i. Given two reports r_1, r_2 and a set of features $\mathcal{F}' \subseteq \mathcal{F}$, we write

1. $r_1[\mathcal{F}'] = r_2[\mathcal{F}']$ iff $r_1[i] = r_2[i]$ for every $f_i \in \mathcal{F}'$;
2. $r_1[\mathcal{F}'] \geqslant r_2[\mathcal{F}']$ iff $r_1[i] \geqslant r_2[i]$ for every $f_i \in \mathcal{F}'$;
3. $r_1[\mathcal{F}'] > r_2[\mathcal{F}']$ iff $r_1[\mathcal{F}'] \geqslant r_2[\mathcal{F}']$ and $r_1[j] > r_2[j]$ for some $f_j \in \mathcal{F}'$.

Now, a user inspecting the ontology might have preferences among features—for instance, location and price might be the most preferred. We thus allow a user to specify a preference relation over the features in order to influence the comparison of reports and eventual ranking of atoms. The following definition formalizes the comparison between two reports r_1 and r_2 relative to a preference relation $\succ$ over $\mathcal{F}$. Intuitively, if $\mathcal{F}_1$ is the first stratum of the stratification of $\mathcal{F}$ with respect to $\succ$, and $r_1[\mathcal{F}_1] > r_2[\mathcal{F}_1]$, then $r_1 > r_2$. If $r_1[\mathcal{F}_1] = r_2[\mathcal{F}_1]$, we compare r_1 and r_2 relative to the second stratum $\mathcal{F}_2$ and if $r_1[\mathcal{F}_2] > r_2[\mathcal{F}_2]$, then $r_1 > r_2$. If $r_1[\mathcal{F}_2] = r_2[\mathcal{F}_2]$, then we compare r_1 and r_2 over the third stratum, and so on.

Definition 1. Consider two reports r_1, r_2 and a preference relation $\succ$ over $\mathcal{F}$. Let $\mathcal{F}_1, \ldots, \mathcal{F}_k$ be the stratification of $\mathcal{F}$. We say $r_1 > r_2$ *relative to* $\succ$ iff there exists $1 \leqslant i \leqslant k$ such that: (i) $r_1[\mathcal{F}_i] > r_2[\mathcal{F}_i]$, and (ii) $r_1[\mathcal{F}_j] = r_2[\mathcal{F}_j]$ for $1 \leqslant j < i$.

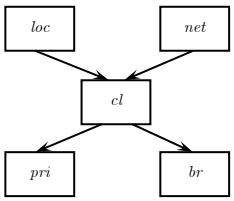

Fig. 1. A user preference relation over features.

Example 4. Consider the reports of Example 3 and let $\mathcal{F}$ be the set of features therein. Suppose a user has the preference relation $\succ_u$ over $\mathcal{F}$ of Figure 1, where an edge from a to b means that $a \succ_u b$. The stratification of $\mathcal{F}$ relative to $\succ_u$ consists of three strata: $\mathcal{F}_1 = \{loc, net\}$, $\mathcal{F}_2 = \{cl\}$, and $\mathcal{F}_3 = \{pri, br\}$. Consider now two reports: $r_1 = \Gamma_1(id_1)$ and $r_2 = \Gamma_1(id_2)$. Then, we have $r_1 > r_2$, since $r_1[\mathcal{F}_1] > r_2[\mathcal{F}_1]$. ∎

Each observer can have a preference relation over the set of features in much the same way as the user looking at the ontology. Thus, we assume the existence of N such preference relations, denoted with $\succ_1, \ldots, \succ_N$ (one for each observer).

Ranking Atoms. Our goal is to obtain a ranking of the ground atoms in the knowledge-base on the basis of: *(i)* N report functions $\Gamma_1, \ldots, \Gamma_N$ (one for each observer), *(ii)* N preference relations $\succ_1, \ldots, \succ_N$ over $\mathcal{F}$ (one for each observer), and *(iii)* a preference relation $\succ_u$ over $\mathcal{F}$ (the user's preferences). For this, we adopt a two-phase approach:

1. Generate a preference relation over D for each observer i on the basis of Γ_i and $\succ_u$, thereby obtaining a preference relation over ground atoms that takes into account the i-th observer's scores and is "customized" according to the user's preferences.
2. Then, the N preference relations obtained in the previous step are combined into one by taking into account how "relevant" the observers' feature preferences are relative to the user's preferences.

Definition 2. Let Γ_i be a report function and $\succ_u$ the user's preference relation over $\mathcal{F}$. For all $A_1, A_2 \in D$, $A_1 >_i A_2$ iff *(i)* $\Gamma_i(A_1)$ is defined and $\Gamma_i(A_2)$ is undefined; or *(ii)* both $r_1 = \Gamma_i(A_1)$ and $r_2 = \Gamma_i(A_2)$ are defined and $r_1 > r_2$ relative to $\succ_u$.

The following example shows how atoms in the database can be ranked in our running scenario by each observer when the user's preferences are also incorporated.

Example 5. Consider the reports of Example 3 and the user's preference relation $\succ_u$ over the features of Figure 1. Then, for each observer, we can determine an order among ground atoms based on the observer's reports, customizing it relative to the user's feature preferences. Specifically, we get:

$$\begin{array}{lll} \textit{Observer 1:} & id_1 >_1 id_2; \quad id_4 >_1 id_2; & id_4 >_1 id_3; \\ \textit{Observer 2:} & id_1 >_2 id_2; \quad id_3 >_2 id_4; & \\ \textit{Observer 3:} & id_2 >_3 id_1; \quad id_4 >_3 id_1. & \end{array}$$

∎

After obtaining a preference relation $>_i$ over D for each observer i, we need to combine them into a single one. In this second step, we want to take into account how "relevant"

the observers' feature preferences are given the user's feature preferences. Thus, we assume the existence of a *relevance function* ρ which takes as input two preference relations over $\mathcal{F}$ (the user preference relation and the preference relation of an observer) and gives as output a value in $[0, 1]$, measuring how similar the two preference relations are (e.g., various distance measures over fully and partially specified preference structures have been proposed in [13]).

Below, we provide a general definition of an operator that combines a set of preference relations where each is associated with a relevance value.

Definition 3. A *preference aggregation operator* is a function that takes as input a set $\mathcal{P}$ of pairs $\langle >_i, \rho_i \rangle$ where $>_i$ is a preference relation and $\rho_i \in [0, 1]$, and returns a preference relation $>$ such that if $a >_i b$ for every $>_i$ appearing in $\mathcal{P}$ then $a > b$.

For instance, when combining the $>_i$'s we may want to give more importance to the more relevant ones, i.e., those with higher ρ_i value, and give the same importance to $>_i$'s with the same ρ_i value.

Example 6. Consider again the setting from Example 5, and suppose the observers' preference relations over the features are such that the relevance function yields 0.8 for observers 1 and 2, and 0.3 for observer 3. An example of preference aggregation operator can be the one that first combines $>_1$ and $>_2$ using, e.g., Pareto composition, thereby obtaining $id_1 >_{1\otimes 2} id_2$ and $id_4 >_{1\otimes 2} id_2$. Then, it combines $>_{1\otimes 2}$ with $>_3$ with prioritized composition giving more importance to $>_{1\otimes 2}$ (because of the higher relevance of the first two observers) thereby obtaining a final order among atoms $>$ as follows: $id_1 > id_2$, $id_4 > id_2$, and $id_4 > id_1$. ∎

Definition 4. *Given a Datalog+/– ontology $KB = (D, \Sigma)$, N report functions $\Gamma_1, \ldots, \Gamma_N$, N preference relations $\succ_1, \ldots, \succ_N$ over $\mathcal{F}$, a preference relation $\succ_u$ over $\mathcal{F}$, a relevance function ρ, and a preference aggregation operator agg, we define a ranking for KB as $agg(\{\langle >_1, \rho(\succ_1, \succ_u)\rangle, \ldots, \langle >_N, \rho(\succ_N, \succ_u)\rangle\})$.*

Notice that in the previous definition each $>_i$ is obtained as per Definition 2. Once a partial order over the ground atoms is obtained, a total order over this set can easily be derived by computing one of its topological sortings.

Proposition 1. *The worst-case time complexity of computing a ranking for a Datalog+/– ontology (D, Σ) is $km^2 + N(mn^2 + f_\rho + logN) + f^N_{agg}$, where m is the number of features, $n = |D|$, N is the number of observers, $k = |\succ_u|$, f_ρ is the worst-case time complexity of the relevance function, and f^N_{agg} is is the worst-case time complexity of the adopted preference aggregation operator over N preference relations.*

4 Ontological Query Answering based on Subjective Reports

Up to now, we have considered simple knowledgebases $(D, \emptyset)$, without directly addressing ontological query answering. We now generalize our framework to deal with conjunctive query answering over ontologies containing tuple-generating dependencies.

The application of such dependencies in Σ can generate new atoms, to which we need to "propagate" reports associated with the ground atoms in D that participated in the creation. Therefore, besides being able to compute answers to queries in the classical manner, we must also provide a way of relating query answers to the information contained in the reports associated with atoms that "contributed" to them—this suggests the need to use an adequate *data provenance representation*. There is a variety of approaches that have been proposed in the formalization of provenance and lineage [12, 6]. In this work, we adopt a special case of the framework from [12], where "how"-provenance is formalized through *semirings of polynomials*; this is a very general and flexible formalism that we adapt to our needs, as discussed next.

A Data Provenance Model. Consider an ontology $KB = (D, \Sigma)$. Recall that every ground atom $A \in D$ has an id, denoted $id(A)$; let $X = \{id(A) \mid A \in D\}$ and $S = \{0, 1\}$, we define a semiring of polynomials $(S[X], +, \times, 0, 1)$, with variables from X and coefficients from S, with operations $+$ and $\times$ being idempotent, associative, and commutative, and $\times$ distributing over $+$. Note that 1 (resp. 0) is the identity element of $\times$ (resp. $+$). The provenance information is modeled as annotations to atoms in $\mathcal{H}$, defined by a function $Ann : \mathcal{H} \mapsto S[X]$, i.e., *Ann* maps ground atoms to polynomials in the semiring. This approach allows us to record the provenance information by means of symbolic expressions over the ids of atoms using the semiring operations, which in turn allows us to model not only information about *which* atoms contributed to the answer's computation but also *how* they contributed.

The Guarded Chase Forest. Note that, as shown in [12], commutative semirings are not enough to correctly model the propagation of provenance annotations that arise from derivations in Datalog; this is the case because derivations in Datalog (and therefore in Datalog+/– ontologies) can be infinite, which requires semirings with infinite sums. For this reason, we focus on guarded Datalog+/– ontologies, and instead of computing the provenance annotations over the chase itself, we compute them over the necessary finite part of the *guarded chase forest* (introduced in [8]). It is sufficient to consider only the finite part of the chase, as for every derivation outside, there is an isomorphic one inside the finite part (which follows from the results in [8]). In the following, we show how the guarded chase forest can be extended to consider provenance annotations.

The guarded chase forest for D and Σ contains *(i)* a node n_A for each atom $A \in D$, having $label(n_A) = A$, and *(ii)* for any two atoms $A, B \in chase(D, \Sigma)$ there are two nodes n_A and n_B with $label(n_A) = A$ and $label(n_B) = B$ along with an arrow from n_A to n_B iff B is obtained from A and possibly other atoms by a one-step application of a TGD $\sigma \in \Sigma$ with A as guard. In [8], it has been shown that whenever homomorphic images of a CQ $Q(\mathbf{X})$ are contained in $chase(D, \Sigma)$, then they are also contained in a finite, initial portion of the guarded chase forest, whose size is determined only by the query and the schema. Furthermore, for guarded Datalog+/– ontologies, the whole derivation of the query atoms is also contained in such a portion of the forest. This means that we can construct the provenance annotations based on this finite portion of the chase forest. For this, we associate with each node n a provenance annotation $ann\_ch(n)$; the forest is computed in the following way: first, for all nodes n labeled with atoms $A \in D$, we have $ann\_ch(n) = id(A)$. Then, each time the chase rule is applied via a TGD $\sigma : \Upsilon(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$ to atoms labeling the nodes of the

guarded chase forest constructed thus far (let these nodes be $n_1, \ldots, n_k$), the annotation of the new node n' labeled with the atom to which $\Psi(\mathbf{X}, \mathbf{Z})$ is mapped is defined as $\mathit{ann\_ch}(n') = \prod_{1 \leqslant i \leqslant k} \mathit{ann\_ch}(n_i)$.

The guarded depth of an atom A in the guarded chase forest for D and Σ, denoted $\mathit{depth}(A)$, is the minimum length of a path from D to a node labeled with A in the forest. Then, the guarded chase of level up to $k > 0$ for D and Σ, denoted $g\_chase^k(KB)$, is the set of all atoms in the guarded chase forest of depth at most k. Given a CQ Q, we denote with $g\_chase^\gamma(KB, Q)$ the part of the chase forest needed to answer Q; as pointed out above, it has been shown that for guarded Datalog+/– ontologies $g\_chase^\gamma(KB, Q)$ is finite and its size (i.e., γ) depends only on Q and KB.

Definition 5. Let $KB = (D, \Sigma)$ be a Datalog+/– ontology and $(S[X], +, \times, 0, 1)$ a commutative doubly idempotent semiring of polynomials. We define the annotation function $\mathit{Ann}\colon \mathcal{H} \to S[X]$ as follows: for each atom $A \in \mathcal{H}$, if $KB \models A$ then $\mathit{Ann}(A) = \sum_{n \in g\_chase^\gamma(KB,A) s.t.\ \mathit{label}(n)=A} \mathit{ann\_ch}(n)$, otherwise $\mathit{Ann}(A) = 0$.

Note that $\mathcal{H}$ can be infinite as can the guarded chase forest; however, the support for function *Ann* is defined as $\mathit{supp}(\mathit{Ann}) = \{A \mid \mathit{Ann}(A) \neq 0\}$, which is clearly finite in $g\_chase^\gamma(KB, Q)$. Note that the annotations depend on the query and the schema.

A Note about Idempotent Operators. Though in general the assumption of idempotent $+$ and $\times$ operators is not made in the context of lineage expressions, for our purposes it makes sense to adopt it since: (i) The $+$ operator is used to express the fact that an answer can be derived in more than one way—if two terms in the polynomial are identical, we only need to keep track of one of them since they both represent the same information regarding which reports should be considered (cf. the derivation of atom $\mathit{accom}(h_1)$ in Example 7). (ii) The $\times$ operator is used to keep track of which atoms participated in a single derivation of an answer—if identical atoms appear in an expression, we only need to keep track of the atom's presence since, as before, we only need to know which reports are associated with the atom and not how many times the atom participated in the derivation; thus, we have that $a * a = a$. Therefore, all exponents and coefficients in our lineage expressions will be equal to 1.

Example 7. Consider the ontology $KB = (D, \Sigma)$ obtained from the one of Example 1 by adding the following atoms to the database:

$$\begin{array}{lll} id_5 : \mathit{pub}(p_1), & id_8 : \mathit{locIn}(h_1, \mathit{oxford}, \mathit{st\_giles}), & id_{11} : \mathit{locIn}(p_2, \mathit{oxford}, \mathit{st\_giles}), \\ id_6 : \mathit{pub}(p_2), & id_9 : \mathit{locIn}(p_1, \mathit{oxford}, \mathit{st\_giles}), & id_{12} : \mathit{locIn}(p_3, \mathit{oxford}, \mathit{queen\_st}). \\ id_7 : \mathit{pub}(p_3), & id_{10} : \mathit{locIn}(bb_1, \mathit{oxford}, \mathit{queen\_st}), & \end{array}$$

Then, each ground atom in D is annotated with its id, i.e., $\mathit{Ann}(\mathit{apthotel}(h_1)) = id_1$, $\mathit{Ann}(\mathit{hotel}(h_2)) = id_2$, etc. The following atoms, along with their annotations, are derived from D and Σ:

$$\begin{array}{ll} \mathit{Ann}(\mathit{accom}(h_1)) = id_1 + id_1 = id_1 & \mathit{Ann}(\mathit{accom}(h_2)) = id_2 \\ \mathit{Ann}(\mathit{accom}(bb_1)) = id_3 & \mathit{Ann}(\mathit{accom}(hs_1)) = id_4 \\ \mathit{Ann}(\mathit{hotel}(h_1)) = id_1 & \mathit{Ann}(\mathit{apartment}(h_1)) = id_1 \end{array}$$ ∎

Query answers with lineage. Given the formalism described above, we can now associate useful information about how query answers are computed.

Definition 6. Let $KB = (D, \Sigma)$ be a Datalog+/– ontology and Q be a CQ of the form $q(\mathbf{X}) = \exists \mathbf{Y} \bigwedge_{i=1}^{k} p_i(\mathbf{X_i}, \mathbf{Y_i})$. The *annotation* of a query answer $A \in ans(Q, KB)$ is defined as follows:

$$Ann(A) = \sum_{\theta \in subs(Q,KB) \wedge A = \theta q(\mathbf{X})} \prod_{i=1}^{k} Ann(\theta p_i(\mathbf{X_i}, \mathbf{Y_i}))$$

, where $subs(Q, KB)$ denotes the set of all substitutions θ such that $KB \models \theta p_i(\mathbf{X_i}, \mathbf{Y_i})$ for all $i \in \{1, \ldots, k\}$.

Example 8. Consider again the ontology from Example 7 and the query $myaccom(X) = \exists S, Y\ accom(X) \wedge locIn(X, oxford, S) \wedge pub(Y) \wedge locIn(Y, oxford, S)$, asking for accommodations in Oxford such that there is a pub on the same street. There are two query answers, namely $myaccom(h_1)$ with annotation $(id_1 \times id_8 \times id_5 \times id_9) + (id_1 \times id_8 \times id_6 \times id_{11})$ and $myaccom(bb_1)$ with annotation $(id_3 \times id_{10} \times id_7 \times id_{12})$. ∎

Thus, each query answer A is associated with an annotation computed as follows: $Ann = \sum_{i=1}^{n} \prod_{j=1}^{k} Ann_{ij}$. Each expression $\prod_{j=1}^{k} Ann_{ij}$ $(1 \leqslant i \leqslant n)$ keeps track of one way of deriving A; then, all possible ways are combined with a summation. We define $S\text{-}Ann(Ann) = \{ \prod_{j=1}^{k} Ann_{ij} \mid 1 \leqslant i \leqslant n \}$. We also define the set of *annotated answers* as $\overline{ans}(Q, KB) = \{\langle A_1, Ann(A_1)\rangle, \ldots, \langle A_n, Ann(A_n)\rangle\}$, where $ans(Q, KB) = \{A_1, \ldots, A_n\}$. The set of *possible set of answers* is defined as:

$$\{\{\langle A_1, Ann'_1\rangle, \ldots, \langle A_n, Ann'_n\rangle\} \mid Ann'_i \in S\text{-}Ann(Ann(A_i)) \text{ for } 1 \leqslant i \leqslant n\}$$

Example 9. Returning to Example 8, There are two possible sets of answers, namely:

$$\{\langle myaccom(h_1), (id_1 \times id_8 \times id_5 \times id_9)\rangle, \langle myaccom(bb_1), (id_3 \times id_{10} \times id_7 \times id_{12})\rangle\}$$
$$\{\langle myaccom(h_1), (id_1 \times id_8 \times id_6 \times id_{11})\rangle, \langle myaccom(bb_1), (id_3 \times id_{10} \times id_7 \times id_{12})\rangle\}.$$ ∎

We can now apply the framework proposed in the previous section to each possible set of answers, and then combine the results obtained for all of them. But before that, we need a way of *determining the features* of the query answers and the reports associated with them. Of course, these should be obtained on the basis of the query structure and the available provenance information.

Each predicate symbol can be associated with a sequence of features. In the following, given a conjunctive query Q, we assume there is an arbitrary but fixed criterion to determine a sequence of features $\mathcal{F}_Q$ for the query answers and assume a way of generating a report for each query answer A on the basis of $Ann(A)$ (in Propositions 2 and 3, we assume that both can be accomplished in polynomial time). A concrete approach is illustrated in the following example.

Example 10. Consider the set of possible answers of Example 9:

$$\{\langle myaccom(h_1), (id_1 \times id_8 \times id_5 \times id_9)\rangle, \quad \langle myaccom(bb_1), (id_3 \times id_{10} \times id_7 \times id_{12})\rangle\}$$

Suppose the features of *accom* are *loc*, *cl*, *pri*, *br*, *net*, while those of *pub* are *food*, *drink*. The features of the query answers might be defined as the union of the aforementioned ones, i.e., *loc*, *cl*, *pri*, *br*, *net*, *food*, *drink*—in general, there can be different

reasonable ways of combining the features of the predicate symbols in the query (in our case, we considered only predicates *accom* and *pub* and took the union of their features). The reports for the query answers may be derived by merging the reports of the *accom*- and *pub*-atoms that contributed to each query answer. For instance, suppose we have two observers and their reports are as follows:

$$
\begin{array}{ll}
\Gamma_1(id_1) = \langle 1, 0.7, 0.5, 1, - \rangle & \Gamma_2(id_1) = \langle 0.3, 0.4, 1, -, - \rangle \\
\Gamma_1(id_5) = \langle 0.5, 1 \rangle & \Gamma_2(id_5) = \langle 0.7, 0.2 \rangle \\
\Gamma_1(id_3) = \langle 0.8, 0.3, 0.2, 0.5, 0.7 \rangle & \Gamma_2(id_3) = \langle 0.4, 0.5, 0.5, 0.6, 0.1 \rangle \\
\Gamma_1(id_7) \text{ not defined} & \Gamma_2(id_7) = \langle 0.1, 0.8 \rangle \\
\ldots & \ldots
\end{array}
$$

Thus, the reports of the first and second observers for the query answers are:

$$
\begin{array}{l}
\Gamma_1(\mathit{myaccom}(h_1)) = \langle 1, 0.7, 0.5, 1, -, 0.5, 1 \rangle \\
\Gamma_1(\mathit{myaccom}(bb_1)) = \langle 0.8, 0.3, 0.2, 0.5, 0.7, -, - \rangle \\
\Gamma_2(\mathit{myaccom}(h_1)) = \langle 0.3, 0.4, 1, -, -, 0.7, 0.2 \rangle \\
\Gamma_2(\mathit{myaccom}(bb_1)) = \langle 0.4, 0.5, 0.5, 0.6, 0.1, 0.1, 0.8 \rangle
\end{array}
$$

At this point, we can apply the framework of the previous section. ∎

Once we get a preference relation for each possible set of answers, they can be combined using a preference aggregation operator, and a total order over this set can easily be derived (e.g., as before, via topological sorting). Thus, eventually, we obtain a ranking over the query answers.

The following proposition provides an upper bound on the (data) complexity of computing query answer rankings. The exponential time upper bound is due to the fact that the number of possible sets of answers can be exponential in the worst case.

Proposition 2. *Computing a query answer ranking can be done in exponential time in the data complexity.*

Alternatively, we can use the annotations in a different way. For instance, we can use a function that maps the id of an atom to a value in the $[0, 1]$ interval such that it compiles all reports (from different observers) associated with the atom into a single score (in Proposition 3 below, we assume that such a function can be computed in polynomial time); this score may represent, for instance, the overall relevance of the atom for the user based on the reports and the relative importance of each dimension. The scores can be combined through the min (resp., max) operator when $\times$ (resp., $+$) is encountered in the annotation. This yields the application of a commutative semiring ($\mathbb{R}^{+}_{\leqslant 1}$, *min*, *max*, 1, 0). This evaluation of reports is called *extensional*.

The following proposition provides an upper bound on the (data) complexity of computing query answer rankings under the extensional evaluation of reports.

Proposition 3. *Computing a query answer ranking under the extensional evaluation of reports can be done in polynomial time in the data complexity.*

The polynomial time complexity of the extensional approach follows from the fact that each of the following tasks can be accomplished in polynomial time: computing the provenance information, mapping atoms' ids to scores, and combining the scores.

5 Related Work

The study of preferences has been carried out in many disciplines; in computer science, the developments that are most relevant to our work is in the incorporation of preferences into query answering mechanisms. To date (and to our knowledge), the state of the art in this respect is centered around relational databases and, recently, in ontological languages for the Semantic Web [15]. The seminal work in preference-based query answering was that of [14], in which the authors extend the SQL language to incorporate user preferences. The preference formula formalism was introduced in [9] as a way to embed a user's preferences into SQL. An important development in this line of research is the well-known *skyline* operator, which was first introduced in [4]. A recent survey of preference-based query answering formalisms is provided in [17]. The main difference is that in this paper we have ontologies, i.e., roughly speaking, we are combining preferences in DBs with ontology-based data access. The problem of evaluating ranked top-k queries in the context of ontology-based access over relational databases was considered in [18]. Studies of preferences related to our approach have also been done in classical logic programming [11] as well as answer set programming frameworks [5].

The present work can be considered as a further development of the PrefDatalog+/– framework presented in [15], where we develop algorithms to answer skyline queries, and their generalization to k-rank queries, over classical Datalog+/– ontologies. The main difference between PrefDatalog+/– and the work presented here is that PrefDatalog+/– assumes that a model of the user's preferences is given at the time the query is issued. On the other hand, we make no such assumption here; instead, we assume that the user only provides some very basic information regarding her preferences over certain features, and has access to a set of reports provided by other users in the past. In a sense, this approach is akin to building an ad hoc model *on the fly* at query time and using it to provide a ranked list of results.

Related to our approach is also the one presented in [10], where preferences are created from provenance annotations of RDF data. There, the usage of different provenance annotation dimensions is considered, each yielding a total preference order represented by a semiring. These preferences are then aggregated according to common social choice theory preference aggregation methods. Another line of related research are constraint-based formalisms for modeling preferences based on semirings, like the one presented in [3].

This work is also related to the study and use of provenance in information systems and, in particular, the Semantic Web and social media [16, 10, 1]. Here, we use a kind of "how"-provenance [12] to study how to propagate report annotations of atoms in an ontology to create preference-based ranked results of ontological queries. Representing the "how"-provenance in this manner allows us to map the semiring into others depending on the manipulations that we wish to perform during the propagation of reports. To our knowledge, this is the first study of a direct application of provenance of reports of this kind found in online reviews to query answering.

6 Summary and Outlook

In this paper, we have studied the problem of preference-based query answering in Datalog+/– ontologies under the assumption that the user's preferences are informed by a set of subjective reports representing opinions of others—such reports model the kind of information found, for instance, in online reviews of products, places, and services.

We first introduced a basic approach to rank atoms in an ontology by combining reports, report authors' preferences, and the user's preferences. Then, we extended our framework to deal with dependencies and query answering using provenance information to keep track of which reports should be considered to evaluate query answers, as well as new information derived from dependencies. Representing the provenance by means of a semiring enabled us to adapt our framework depending on the kind of report propagation that we wish to carry out. One direction for future work involves investigating further semirings for manipulating reports during their propagation.

Much work remains to be done in this line of research, such as expressing general reports that apply to sets of tuples, and exploring the application of existing techniques to gather reports from actual information available in Web reviews. We also plan to experimentally evaluate our framework over synthetic and real-world data.

Acknowledgments. This work was supported by EPSRC grant EP/J008346/1 ("PrOQAW"), an EU (FP7/2007-2013) Marie-Curie Intra-European Fellowship, the ERC grant 246858 ("DIADEM"), and a Yahoo! Research Fellowship.

References

1. Barbier, G., Feng, Z., Gundecha, P., Liu, H.: Provenance Data in Social Media. Morgan and Claypool (2013)
2. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP. pp. 73–85 (1981)
3. Bistarelli, S., Pini, M.S., Rossi, F., Venable, K.B.: Bipolar preference problems: Framework, properties and solving techniques. In: Recent Adv. in Constraints, pp. 78–92. Springer (2007)
4. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. ICDE. pp. 421–430 (2001)
5. Brewka, G.: Preferences, contexts and answer sets. In: Proc. ICLP. p. 22 (2007)
6. Buneman, P.: The providence of provenance. In: Proc. BNCOD. pp. 7–12 (2013)
7. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. KR. pp. 70–80 (2008)
8. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
9. Chomicki, J.: Preference formulas in relational queries. TODS 28(4), 427–466 (2003)
10. Dividino, R., Gröner, G., Scheglmann, S., Thimm, M.: Ranking RDF with provenance via preference aggregation. In: Proc. EKAW'12. pp. 154–163. Springer (2012)
11. Govindarajan, K., Jayaraman, B., Mantha, S.: Preference queries in deductive databases. New Generat. Comput. 19(1), 57–86 (2001)
12. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proc. PODS. pp. 31–40 (2007)
13. Ha, V., Haddawy, P.: Toward case-based preference elicitation: Similarity measures on preference structures. In: Proc. UAI. pp. 193–201 (1998)

14. Lacroix, M., Lavency, P.: Preferences: Putting more knowledge into queries. In: Proc. VLDB. vol. 87, pp. 1–4 (1987)
15. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Preference-based query answering in Datalog+/– ontologies. In: Proc. IJCAI. pp. 1017–1023 (2013)
16. Moreau, L.: The foundations for provenance on the Web. Found. Trends Web Sci. 2(2/3), 99–241 (2010)
17. 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)
18. Straccia, U.: On the top-k retrieval problem for ontology-based access to databases. In: Flexible Approaches in Data, Information and Knowledge Management, pp. 95–114 (2013)