Preference-Based Query Answering in Datalog+/– Ontologies

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

Department of Computer Science, University of Oxford, UK
firstname.lastname@cs.ox.ac.uk

Abstract. The study of preferences has a long tradition in many disciplines, but it has only relatively recently entered the realm of data management through their application in answering queries to relational databases. The current revolution in data availability through the Web and, perhaps most importantly in the last few years, social media sites and applications, puts ontology languages at the forefront of data and information management technologies. In this paper, we propose the first (to our knowledge) integration of ontology languages with preferences as in relational databases by developing PrefDatalog+/–, an extension of the Datalog+/– family of languages with preference management formalisms closely related to those previously studied for relational databases. We focus on two kinds of answers to queries that are relevant to this setting, skyline and k-rank (a generalization of top-k queries), and develop algorithms for computing these answers to DAQs (disjunctions of atomic queries) as well as preliminary algorithms for CQs (conjunctive queries) based on related work in the databases and ontologies literature. We show that DAQ answering in PrefDatalog+/– can be done in polynomial time in the data complexity, as in relational databases, as long as query answering can also be done in polynomial time (in the data complexity) in the underlying classical ontology.

1 Introduction and Related Work

In the past two decades, there has been a rapid development in the research and development of technologies to make an effective use of Web data. For instance, ontology languages, along with associated query answering algorithms, have been extensively studied with computational tractability in mind in order to guarantee their applicability on large scale knowledge bases. However, there remains much work to be done in central areas, such as search, where the state of the art is still centered in receiving keywords from a user and returning a set of links to Web documents. The main drawback to this paradigm is that the quality of the set of answers is much more important to users than the quantity. The evolution that seeks to address these issues is called *semantic search*, and is focused on identifying objects on the Web instead of documents and keywords. The other revolution that has recently taken place in the Web is often referred to as the *Social Web* (as an evolution of the Semantic Web [2]) or Web 2.0, and is centered around a system composed of platforms that users adopt to share information and collaborate in a social environment. This makes the main tenets of semantic search

ever more relevant for virtually all activities on the Web, since the fundamentally human component of these systems makes each user's *personal preferences* have a much more prevalent role than what was observed prior to this paradigm shift. The current challenge for Web search is thus inherently linked to leveraging the social components of Web content in order to develop some form of semantic search and query answering on the Web as a whole.

Preferences have long been studied as an important part of people's daily lives, and have thus received much attention in many areas of study such as philosophy (as early as Aristotle), choice theory (for instance, in rational decision making), and certain social sciences (such as certain areas dealing with voting, auctions, and other mechanisms for social choice). Though these and many other avenues of study have naturally led to the study of preferences in computer science, the most relevant to our work is that of their incorporation into query answering mechanisms. To date (and to our knowledge), the state of the art in this respect is centered around relational databases. The seminal work in preference-based query answering was that of [9], in which the authors extend the SQL language to incorporate user preferences, showing that the resulting formalism could be translated into the domain relational calculus. Preference formulas were introduced in [6], where a logical formalism is proposed that allows an embedding of preference specifications into SQL through a *winnow* operator that is parameterized by a preference formula; the winnow operator is a generalization of the *skyline* operator, first introduced in [3]. Perhaps closest in nature to our approach is that of preference Datalog programs [8], which are a restriction of preference logic programs [7] 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); the authors develop algorithms based on optimizations of the basic bottom-up evaluation approach to query answering based on the set of input preferences. Also, conditional preference bases [10] are closely related since they consist of a DL knowledge base (in $\mathcal{SHOIN}(\mathbf{D})$ or $\mathcal{SHIF}(\mathbf{D})$) and a preference model built from variable-strength conditional preferences. For a recent survey of preference-based query answering formalisms, see [11].

In this paper, we propose a novel integration of ontology languages with preference management mechanisms by developing an extension of the Datalog+/– family of ontology languages [5] with preference specification languages that are closely related to those previously studied in the relational databases literature. The main contributions of this paper are: (i) the introduction of the PrefDatalog+/– framework, the first (to our knowledge) combination of ontology languages with preferences as in relational databases; (ii) the development of algorithms for answering both skyline and k-rank queries (the latter is a generalization of top-k queries) for disjunctions of atomic queries (based on a novel augmented chase procedure) and conjunctive queries (based on the classical chase procedure and leveraging algorithms from related work in relational databases), along with proofs of correctness and running times showing that answering disjunctions of atomic queries (DAQs) in PrefDatalog+/– can be done in polynomial time in the data complexity, i.e., within the same complexity bounds as in relational databases, as long as query answering in the underlying classical ontology can also be

done in polynomial time in the data complexity; and (iii) a complexity result showing that CQ answering is intractable in general.

The rest of this paper is organized as follows. In Section 2, we present preliminary concepts on classical Datalog+/– and preference specification formalisms from relational databases. Section 3 introduces the syntax and semantics of PrefDatalog+/–, along with the two kinds of queries that we focus on in this work. Section 4 presents algorithms for skyline and k-rank queries to PrefDatalog+/– ontologies, focusing especially on DAQs; though an intractability result is proved for CQs, we also discuss how algorithms from relational databases can be leveraged for these queries as well. Finally, in Section 5, we conclude and discuss future work. Proofs of all stated results can be found in the full version of this paper.

2 Preliminary Concepts

We now briefly recall some necessary background concepts.

2.1 Datalog+/–

First, we present some basics on Datalog+/– [5], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple-generating dependencies (TGDs), negative constraints, the chase, 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 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. 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 a database 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 of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, 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 [1], even when the schema and TGDs are fixed [4]. 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.

Negative constraints (or simply *constraints*) γ are first-order formulas $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, 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 false, otherwise the constraints can simply be ignored when answering the BCQ Q.

The final component of the language corresponds to *equality generating dependencies*; since they are not directly used here, we refer to [5] for their details. We usually omit the universal quantifiers in TGDs and negative constraints, 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*.

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 $\mathit{chase}^0(D, \Sigma)$, 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 $\mathit{chase}^k(D, \Sigma)$, 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) $I_1, \ldots, I_n \subseteq \mathit{chase}^{k-1}(D, \Sigma)$ and (ii) the highest level of an atom in every I_i is $k-1$; then, perform every corresponding TGD application on $\mathit{chase}^{k-1}(D, \Sigma)$, choosing the applied TGDs and homomorphisms in a (fixed) linear and lexicographic order (resp.), and assigning to every new atom the *level* k. The *chase* of D relative to Σ, denoted $chase(D, \Sigma)$, is defined as the limit of $\mathit{chase}^k(D, \Sigma)$ 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 $B \in mods(D, \Sigma)$ [5]. 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 $\Sigma = \Sigma_T \cup \Sigma_{NC}$, consists of a database D, a set of TGDs Σ_T, and a set of negative constraints Σ_{NC}. We say KB is *guarded* (resp., *linear*) iff Σ_T is guarded (resp., linear). Example 1 (used in the sequel as a running example) illustrates a simple Datalog+/– ontology.

Example 1. Let $O = \langle D, \Sigma \rangle$ be a simple ontology describing gift ideas for friends:

$\Sigma = \{$ *scifi_book*$(T, A) \rightarrow$ *book*(T, A), *fant_book*$(T, A) \rightarrow$ *book*(T, A),
mmorpg$(X) \rightarrow$ *vidGame*(X), *actAdv*$(X) \rightarrow$ *vidGame*(X),
book$(T, A) \rightarrow$ *educ*(T), *puzzle*$(N) \rightarrow$ *educ*(N) $\}$,

and $D = \{$*scifi_book*$(b_1, asimov)$, *scifi_book*$(b_2, asimov)$, *fant_book*$(b_3.tolkien)$, *puzzle*(p_1), *mmorpg*(v_1), *actAdv*$(v_2)\}$.

This ontology considers two kinds of book and video game genres, and states that books and puzzles are categorized under an "educational" label. Database D provides some instances for each of these genres. ∎

Finally, we would like to recall that different fragments of Datalog+/– have been shown to generalize the $\mathcal{EL}$ and *F-Logic Lite* description logics, as well as the *DL-Lite* family of tractable description logics [5].

2.2 Preference Relations via Preference Formulas

Though the PrefDatalog+/– formalism introduced in Section 3 allows essentially any preference framework to be used, in this paper we adopt a formalism slightly generalizing the *preference formulas* approach that was proposed in [6] in order to develop the concrete approaches to query answering discussed in Section 4.

In the following, let Δ_{Pref} be a finite set of constants, $\mathcal{R}_{Pref}$ be a finite set of predicate names, and $\mathcal{V}_{Pref}$ be an infinite set of variables; we denote with $\mathcal{H}_{Pref}$ the Herbrand base relative to these sets, respectively. A *preference relation* is any binary relation $\succ \subseteq \mathcal{H}_{Pref} \times \mathcal{H}_{Pref}$. Note that, slightly generalizing the framework in [6], preference relations can be defined between atoms corresponding to different predicates, and that formulas are not "iff" statements.

Definition 1. Let a_1 and a_2 be atoms over $\mathcal{R}_{Pref}$, $\mathcal{V}_{Pref}$, and Δ_{Pref}. A *preference formula pf* is of the form "$a_1 \succ a_2$ *if* $C(a_1, a_2)$", where $C(a_1, a_2)$ is a first-order formula. We call $C(a_1, a_2)$ the *condition* of *pf*, denoted *cond*(*pf*).

The smallest (with respect to set inclusion) preference relation defined via a set P of preference formulas is denoted with $\succ_P$.

Example 2. Following the same topic of Example 1, the following formulas might represent a specific friend's preferences for children's gifts:

C_1: *educ*$(X) \succ$ *vidGame*(Y) if $\top$
C_2: *book*$(T_1, A_1) \succ$ *book*(T_2, A_2) if *scifi_book*$(T_1, A_1) \wedge$ *fant_book*(T_2, A_2)
C_3: *book*$(T_1, A_1) \succ$ *book*(T_2, A_2) if $T_1 = b_1 \wedge T_2 = b_2$
C_4: *educ*$(X) \succ$ *educ*(Y) if *puzzle*$(X) \wedge$ *fant_book*(Y)
C_5: *vidGame*$(X) \succ$ *vidGame*(Y) if *actAdv*$(X) \wedge$ *mmorpg*(Y)

For instance, formula C_1 states that educational gifts are preferable to video games (unconditionally), while C_4 states that an educational gift that is a puzzle is preferable to one that is a fantasy book. ∎

3 PrefDatalog+/–: Syntax and Semantics

Recall that we have (as discussed in Section 2), the logical languages for ontologies and preferences. For the former, we have an infinite universe of constants Δ_{Ont}, an infinite set of variables $\mathcal{V}_{Ont}$, and a finite set of predicate names $\mathcal{R}_{Ont}$; analogously for the latter, we have a finite set of constants Δ_{Pref}, an infinite set of variables $\mathcal{V}_{Pref}$, and a finite set of predicate names $\mathcal{R}_{Pref}$. In the following, we assume w.l.o.g. that $\mathcal{R}_{Pref} \subseteq \mathcal{R}_{Ont}$, $\Delta_{Pref} \subseteq \Delta_{Ont}$, and $\mathcal{V}_{Pref} \subseteq \mathcal{V}_{Ont}$. These sets give rise to corresponding *Herbrand bases* consisting of all possible ground atoms that can be formed, which we denote by $\mathcal{H}_{Ont}$ and $\mathcal{H}_{Pref}$, 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.

Definition 2. Let O be a Datalog+/– ontology and P be a set of preference formulas with Herbrand bases $\mathcal{H}_{Ont}$ and $\mathcal{H}_{Pref}$, respectively. A *preference-based Datalog+/– ontology* (PrefDatalog+/– ontology, or knowledge base) is of the form $KB = (O, P)$, where $\mathcal{H}_{Pref} \subseteq \mathcal{H}_{Ont}$. If O is a guarded Datalog+/– ontology, we say that KB is a guarded PrefDatalog+/– ontology.

Semantics. The semantics of PrefDatalog+/– arises as a direct combination of the semantics of Datalog+/– and that of preference formulas. Let $KB = (O, P)$: $KB \models a_1 \succ_P a_2$ iff (i) $O \models a_1$ and $O \models a_2$; and (ii) $\models \bigvee_{pf_i \in P} cond(pf_i)(a_1, a_2)$. Intuitively, the consequences of a knowledge base $KB = (O, P)$ are computed in terms of the chase for the classical Datalog+/– ontology O, and the formulas in P describe the preference relation over pairs of atoms in $\mathcal{H}_{Ont}$. The following is a simple example.

Example 3. Let $KB = (O, P)$ be a PrefDatalog+/– ontology, with ontology O from Example 1 and set of preference formulas P from Example 2. We can see that $KB \models$

educ$(b_1) \succ_P$ *vidGame*(v_1), since $O \models$ *educ*(b_1) and $C_1 \in P$ allows us to conclude that *educ*$(b_1) \succ_P$ *vidGame*(v_1). On the other hand, we have $KB \not\models$ *educ*$(b_1) \succ_P$ *educ*(b_2), since condition (ii) is not satisfied. ∎

Queries. In this paper, we are interested in two kinds of preference-based queries that can be issued over PrefDatalog+/– ontologies: *skyline* and *k-rank*. In order to define answers to such queries, we adopt the following definitions from classical logic. A substitution is a function 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 exists a substitution σ such that $\omega = \sigma \circ \theta$. We consider two kinds of classical queries: conjunctive queries (CQs) and disjunctive atomic queries (disjunctions of atoms – DAQs).

Definition 3. Consider PrefDatalog+/– ontology $KB = (O, P)$, $k \geqslant 0$, and DAQ $Q(\mathbf{X}) = q_1(\mathbf{X}_1) \vee ... \vee q_n(\mathbf{X}_n)$ where the q_i's are atoms and $\mathbf{X}_1 \cup ... \cup \mathbf{X}_n = \mathbf{X}$. The set of *skyline answers* to Q is defined as: $\{\theta q_i \mid O \models \theta q_i$ and $\nexists \theta'$ s.t. $O \models \theta' q_j$ and $\theta' q_j \succ_P \theta q_i$, with $1 \leqslant i, j \leqslant n\}$, where θ, θ' are ground substitutions for the variables in $Q(\mathbf{X})$. A *k-rank answer* to $Q(\mathbf{X})$ is defined for transitive relations $\succ_P$ as a sequence of maximal length of mgu's for $\mathbf{X}$: $S = \langle \theta_1, \ldots, \theta_{k'} \rangle$ such that $O \models \theta_i Q$ for $1 \leqslant i \leqslant k' \leqslant k$, and S is built by subsequently appending the skyline answers to Q, removing these atoms from consideration, and repeating the process until either $S = k$ or no more answers to Q remain.

In Definition. 3, note that k-rank answers are only defined when the preference relation is transitive—this is required for the answers to make sense, since the atoms in each rank are supposed to be preferred to those in the next ranks. This kind of answer can be seen as a generalization of traditional top-k answers [11] that are still defined when $\succ_P$ is not a weak order, and their name arises from the concept of *rank* introduced in [6].

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*.

Example 4. Consider again ontology $KB = (O, P)$ from Example 3, $Q_1(X, Y) =$ *book*(X, Y), and $Q_2(X) =$ *educ*(X); the $\succ_P$ relation is depicted in Figure 1 (dotted lines). The set of skyline answers to query Q_1 is $\{$*book*$(b_1,$ *asimov*$)\}$, while for Q_2 it is $\{$*educ*(p_1), *educ*(b_1), *educ*$(b_2)\}$. A 3-rank answer to Q_1 is: $\langle$*book*$(b_1,$ *asimov*$)$, *book*$(b_2,$ *asimov*$)$, *book*$(b_3,$ *tolkien*$)\rangle$. For Q_2, a 3-rank answer is $\langle$*educ*(p_1), *educ*(b_1), *educ*$(b_2)\rangle$. Finally, $Q_3 =$ *puzzle*$(X) \vee$ *vidGame*(X) yields $\{$*puzzle*(p_1), *vidGame*$(v_2)\}$ as skyline answers, and a 3-rank answer is $\{$*puzzle*(p_1), *vidGame*(v_2), *vidGame*$(v_1)\}$.∎

Finally, we have a result relating k-rank to the traditional top-k queries (cf. [11]).

Proposition 1. *Let $KB = (O, P)$ be a PrefDatalog+/– ontology, Q be a DAQ or a CQ, and $k \geqslant 0$. If $\succ_P$ is a weak order, then k-rank answers are equal to top-k answers (modulo ordering of incomparable elements).*

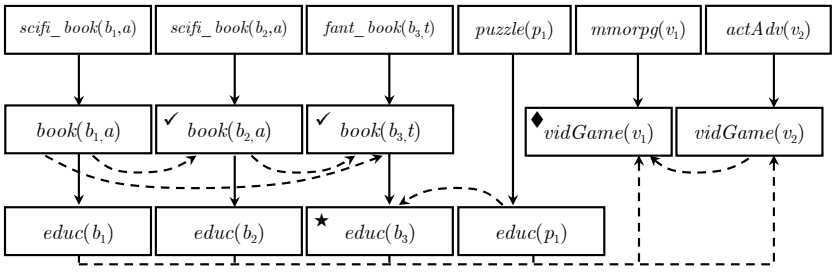

Fig. 1. Preference-augmented chase forest for the running example. Dashed arrows correspond to preference edges, and the marks in the upper left corner represent marked nodes for different queries; author names have been shortened to letters.

Conjunctive Queries. Definition 3 can easily be extended to conjunctive queries—intuitively, substitutions in the set of answers must be for all q_i's in the query instead of just one and, in the case of (non-atomic) conjunctive queries, the substitutions in answers no longer yield single atoms but rather *sets* of atoms; therefore, we must extend the preference specification framework to take into account sets of atoms instead of individual ones. One such approach was proposed in [12], where a mechanism to define a preference relation over tuple sets $\succ_{PS}$: $2^{\mathcal{H}_{Pref}} \times 2^{\mathcal{H}_{Pref}}$ is introduced. Though this is not the focus of this paper, in Section 4.2 we treat their complexity and briefly discuss how methods from the relational databases literature can be applied to answer them.

4 Preference-Based Query Answering

In this section, we propose algorithms for answering queries to PrefDatalog+/– ontologies; we start with DAQs and then move on to the conjunctive case.

4.1 Disjunctive Atomic Queries

We begin by introducing a structure that is used in preference-based query answering. Next, we investigate algorithms for skyline and k-rank query answering for DAQs.

The Preference-Augmented Chase (*prefChase*). In order to compute skyline and k-rank answers to queries over a PrefDatalog+/– ontology $KB = (\langle D, \Sigma \rangle, P)$, we make use of a data structure called the *chase forest* for D, Σ, and query Q [5], which is defined as the directed graph consisting of (i) for every $a \in D$, one node labeled with a, and (ii) for every node labeled with $a \in chase(D, \Sigma)$ and for every atom $b \in chase(D, \Sigma)$ that is obtained from a and possibly other atoms by a one-step application of a TGD $\sigma \in \Sigma$ with a as guard, one node labeled with b along with an edge from the node labeled with a. Here, we propose an *augmented* chase forest, comprised of the necessary finite part of the chase forest relative to a given query that is augmented with an additional kind of edge that we call *preference edges*, which occur between nodes labeled with $a, b \in chase(D, \Sigma)$ iff $a \succ_P b$. Finally, when an edge is introduced *between nodes whose labels satisfy* Q, the node with the incoming edge is *marked*. Figure 1

Algorithm 1: Skyline$(KB = (\langle D, \Sigma \rangle, P), Q)$
Input: PrefDatalog+/– ontology KB and DAQ $Q(\mathbf{X})$.
Output: Set of skyline answers $\{a_1, ..., a_k\}$ to Q.

1. Initialize Res as an empty set of ground atoms;
2. C:= $prefChase(KB, Q)$;
3. For each atom a labeling node v in C do begin
4. If $a \models Q$ and v is unmarked then Res:= $Res \cup \{a\}$;
5. end;
6. return Res.

Fig. 2. An algorithm that computes the set of skyline answers (in atom form) to an atomic query.

shows *prefChase*(KB, Q) for the PrefDatalog+/– ontology from Example 3; for illustrative reasons, markings for three different queries have been included in the figure: *book*(X, Y) (check mark), *educ*(X) (star), and *vidGame*(X) (diamond).

Theorem 1. *If $KB = (\langle D, \Sigma \rangle, P)$ is a guarded PrefDatalog+/– ontology and Q a DAQ, $prefChase(KB, Q)$ can be computed in time $O(n^2)$ in the data complexity.*

Skyline Queries. Algorithm Skyline (Fig. 2) begins by computing the preference-augmented chase forest relative to the input ontology and query and then simply analyzing the set of marked nodes (i.e., those that are dominated by other tuples relative to the query). As we can see, the node markings have done almost all of the work towards answering the skyline query; the algorithm only needs to go through the structure and find the nodes whose labels satisfy the query and, if unmarked, add them to the output.

Example 5. Consider again the setup in Example 4 and Figure 1. For query Q_1, the only unmarked atom is *book*(b_1, a), and so the skyline answer is the singleton set with this atom. For Q_2, we get all the *educ* atoms except *educ*(b_3). ■

Theorem 2. *Let KB be a PrefDatalog+/– ontology, Q be a DAQ, and $k \geqslant 0$. Then, (i) Algorithm* Skyline *correctly computes the skyline answers to Q, and (ii) Algorithm* Skyline *runs in time $O(k)$ in the data complexity, where k is the time needed to compute prefChase(KB, Q) in the data complexity.*

Corollary 1. *Let $KB = (O, P)$ be a PrefDatalog+/– ontology, Q be a DAQ, and $k \geqslant 0$. If O is a guarded Datalog+/– ontology, then the set of skyline answers to Q can be computed in polynomial time in the data complexity.*

k-Rank Queries. Algorithm k-Rank (Figure 3) also begins by computing the preference-augmented chase forest. The main `while` loop iterates through the process of computing the skyline answers to Q by using the Skyline algorithm, updating the result by appending these answers in arbitrary order, and removing the nodes and edges involved in the result from the chase forest; the loop finally prepares for the next iteration by updating the markings in the forest. Once the loop is finished, the algorithm returns the first k results, since the last iteration might add superfluous elements.

Algorithm 2: k-Rank$(KB = (\langle D, \Sigma \rangle, P), Q, k)$
Input: PrefDatalog+/– ontology KB, DAQ $Q(\mathbf{X})$, $k \geqslant 0$.
Output: k-rank answer $\langle a_1, ..., a_{k'} \rangle$ to Q.

1. Initialize Res as an empty vector of ground atoms;
2. C:= $prefChase(KB, Q)$;
3. i:= k;
4. While $i > 0$ do begin
5. S:= Skyline(KB, Q);
6. Append S to Res (arbitrary order);
7. Remove S from C along with all associated edges;
8. Identify marked nodes that no longer have incoming preference edges and change them to *unmarked*;
9. i:= $i - |S|$;
10. End;
11. Return $truncate(Res, k)$.

Fig. 3. An algorithm for obtaining a k-rank answer (in atom form) to DAQ Q.

Example 6. Consider the setup in Example 4 and the structure in Figure 1. The node $book(b_1, a)$ is the first skyline answer; Algorithm k-Rank then proceeds to update *Res* with $\langle book(b_1, a) \rangle$ and to remove this node (and associated edges) from the forest; according to Line 8, node $book(b_2, a)$ now becomes unmarked. The algorithm continues until the three *book* atoms are processed. ■

Theorem 3. *Let KB be a PrefDatalog+/– ontology, Q be a DAQ, and $k \geqslant 0$. Then, (i) Algorithm k-Rank correctly computes a k-rank answer to Q, and (ii) Algorithm k-Rank runs in time $O(k)$ in the data complexity, where k is the time needed to compute $prefChase(KB, Q)$ in the data complexity.*

Corollary 2. *Let $KB = (O, P)$ be a PrefDatalog+/– ontology, Q be a DAQ, and $k \geqslant 0$. If O is a guarded Datalog+/– ontology, then a k-rank answer to Q can be computed in polynomial time in the data complexity.*

4.2 Conjunctive Queries

As discussed at the end of Section 3, non-atomic conjunctive queries require a preference relation that is defined between sets of tuples—in this section we assume that we have an extension of the preference formulas formalism *PS* that yields such a relation; a detailed treatment of this extension is outside the scope of this paper. Unfortunately, the following result tells us that this change has a direct impact on the complexity of both skyline and k-rank query answering.

Theorem 4. *Let $KB = (\langle D, \Sigma \rangle, PS)$ be a guarded PrefDatalog+/– ontology, where PS describes a preference relation $\succ_{PS}$: $2^{\mathcal{H}_{Pref}} \times 2^{\mathcal{H}_{Pref}}$ such that membership can be tested in polynomial time, and Q be a CQ. If the relational schema and Σ are fixed, deciding if the set of skyline answers or a k-rank answer to Q are non-empty is Σ_2^p-complete.*

The heuristic algorithms presented in [12] for relational databases can be applied to PrefDatalog+/– by computing the *prefChase* relative to the query and thus materializing the necessary part of the ontology into a database. The following result provides a way to leverage tools developed for relational databases for first order rewritable fragments; we use *prefQ-DPM* to denote either Skyline or k-Rank, and *prefQ-RDB* the corresponding algorithm for relational databases.

Theorem 5. *Let $KB = (\langle D, \Sigma \rangle, PS)$ be a PrefDatalog+/– ontology and Q a CQ. If $\langle D, \Sigma \rangle$ belongs to an FO-rewritable fragment of Datalog+/– and all $F \in PS$ are such that cond(F) are conjunctions of atoms, then we can compute PS' and Q' in PTIME such that prefQ-DPM$(KB, Q) =$ prefQ-RDB(D, PS', Q').*

Intuitively, the theorem is a consequence of the fact that for each $F \in PS$, *cond*(F) can be treated like a BCQ and thus rewritten relative to Σ_T. This result is important since it identifies conditions under which preference-based queries can be answered using highly optimized relational database management systems.

5 Summary and Outlook

In this work, we have presented an extension of the Datalog+/– family of ontology languages for preference-based query answering, a topic that has recently become central in dealing with data related to the Social Semantic Web. We focused on two well-known kinds of preference-based queries, namely skyline and k-rank (a generalization of traditional top-k queries), and studied algorithms and complexity of computing their answers when the underlying classical queries are either DAQs or CQs. For DAQs, we proposed an augmented version of the chase forest, and showed how this structure can be used to answer both kinds of preference-based queries in polynomial time in the data complexity whenever classical query answering is PTIME in the underlying Datalog+/– ontology. For CQs, we showed that this desirable property is lost, and note that related work in relational databases can however be applied in order to leverage heuristics—we also provided a result for leveraging an RDBMS for FO-rewritable fragments of Datalog+/–. Current and future work involves implementing and testing the PrefDatalog+/– framework, and exploring specific heuristics and other approximation techniques that can be applied to answer preference-based CQs scalably.

Acknowledgments. A slightly longer version of this paper appears in the proceedings of IJCAI 2013. This work was supported by UK EPSRC grant EP/J008346/1–"PrOQAW", ERC Grant 246858–"DIADEM", and a Yahoo! Research Fellowship.

References

1. C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *Proc. ICALP-1981*, volume 115 of *LNCS*, pages 73–85. Springer, 1981.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Sci. Am.*, 284(5):34–43, 2002.
3. S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proc. ICDE-2001*, pages 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

4. A. Calì, G. Gottlob, and M Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. KR-2008*, pages 70–80. AAAI Press, 2008.
5. A. Calì, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.
6. J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, December 2003.
7. K. Govindarajan, B. Jayaraman, and S. Mantha. Preference logic programming. In *Proc. ICLP-1995*, pages 731–745, 1995.
8. K. Govindarajan, B. Jayaraman, and S. Mantha. Preference queries in deductive databases. *New Generat. Comput.*, 19(1):57–86, 2001.
9. M. Lacroix and P. Lavency. Preferences: Putting more knowledge into queries. In *Proc. VLDB-1997*, volume 87, pages 1–4, 1987.
10. T. Lukasiewicz and J. Schellhase. Variable-strength conditional preferences for ranking objects in ontologies. *J. of Web Sem.*, 5(3), 2007.
11. K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst.*, 36(3):19:1–19:45, August 2011.
12. X. Zhang and J. Chomicki. Preference queries over sets. In *Proc. ICDE-2011*, pages 1019–1030, 2011.