Query Answering in Datalog+/– Ontologies under Group Preferences and Probabilistic Uncertainty

Thomas Lukasiewicz, Maria Vanina Martinez,
Gerardo I. Simari, and Oana Tifrea-Marciuska

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

Abstract. In the recent years, the Web has been changing more and more towards the so-called Social Semantic Web. Rather than being based on the link structure between Web pages, the ranking of search results in the Social Semantic Web needs to be based on something new. We believe that it can be based on user preferences and underlying ontological knowledge. Modeling uncertainty is also playing an increasingly important role in these domains, since uncertainty can arise due to many uncontrollable factors. In this paper, we thus propose an extension of the Datalog+/– ontology language with a model for representing preferences of groups of users and a model for representing the (probabilistic) uncertainty in the domain. Assuming that more probable answers are more preferable, this raises the question how to rank query results, since the preferences of single users may be in conflict with the probability-based preferences and also with each other. We thus propose preference merging and aggregation operators, respectively, and study their semantic and computational properties. Based on these operators, we provide algorithms for answering k-rank queries for DAQs (disjunctions of atomic queries), which generalize top-k queries based on the iterative computation of classical skyline answers, and show that, under certain reasonable conditions, they run in polynomial time in the data complexity.

1 Introduction

In the recent years, the Web has been shifting more and more away from data on linked Web pages towards less interlinked data in social networks relative to underlying ontologies, called the *Social Semantic Web*. This requires new technologies for search and query answering, where the ranking of search results is not based on the link structure between Web pages anymore, but on the information in the Social Semantic Web, in particular, the preferences of the users and the underlying ontological knowledge.

Modeling the preferences of a group of users is also an important research topic in its own right. With the growth of social media, people post their preferences and expect to get personalized information. Moreover, people use social networks as a tool to organize events, where it is required to combine the individual preferences and suggest items obtained from aggregated user preferences. For example, if there is a movie night of friends, family trip, or dinner with working colleagues, one has to decide which is the ideal movie or location for the group, given the preferences of each member. In addition, collaborative search [18] has recently started to play an important role.

Q.Z. Sheng and J. Kjeldskov (Eds.): ICWE 2013 Workshops, LNCS 8295, pp. 192–206, 2013.

Modeling group preferences comes with two challenges. The first one is to define a group preference semantics that solves the possible *disagreement* among users. For example, people (even friends) often have different tastes on how they prefer to spend their holidays. Therefore, a system should return results in such a way that each individual benefits from the result. The second challenge is to allow for efficient algorithms, e.g., to compute efficiently the answers to queries under aggregated group preferences [2].

There are many studies that are addressing the area of group modeling, which is indirectly related to the area of social choice (group decision making, i.e., optimal decisions for a user given the opinion of a collection of users), studied in mathematics, economics, politics, and sociology [21,23]. Other areas related to social choice are meta-search [16], collaborative filtering [14], and multi-agent systems [24]. Group preferences have especially been studied in the area of recommender systems [2,20], which focus on quantitative preferences. However, in many real-world scenarios, the ordering of preferences is incomplete. This appears due to privacy issues or an incomplete elicitation process (for example, because users may not want to be asked too many questions). Furthermore, it is often difficult to determine the appropriate numerical preferences and weights that maximize the utility of a decision [5]. For example, it is difficult for a user to determine a numerical value (e.g., 0.7 or 0.9) to rate a certain activity. Therefore, there is a growing interest in formalisms for representing and reasoning with qualitative incomplete preferences [22,13,1]. These approaches, however, do not have underlying ontologies, which are an important ingredient of the Semantic and the Social Semantic Web, and which also provide useful information for the ranking of query results.

The presence of uncertainty in the Web in general is undeniable [12,15,19,9]. Different sources of uncertainty that must be dealt with in answering queries in the Social Semantic Web are, for example, information integration (as in travel sites that query multiple sources to find touristic tours), automatic processing of Web data (analyzing an HTML document often involves uncertainty), as well as inherently uncertain data (such as user comments or tight relationships between users).

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 preferences of groups of users and uncertainty management mechanisms. We do this by developing an extension of the Datalog+/– family of ontology languages [7] with a preference model over the consequences of the ontology, as well as a probabilistic model that assigns probabilities to them. The preference and the probabilistic model are assumed to model the preferences of a group of users and the uncertainty in the domain, respectively.

The main contributions of this paper can be summarized as follows.

- We introduce GPP-Datalog+/–, which combines the Datalog+/– ontology language with both group preferences (a generalization of preference handling in relational databases) and probabilistic uncertainty. To our knowledge, this is the first combination of ontology languages with group preferences and probabilistic uncertainty.
- We present operators for merging single-user and score-based (probability-based) preferences (in the form of a strict partial and a weak order, respectively), to

produce a new single-user preference relation satisfying certain basic properties. We also present several ways to compute group preferences as an aggregation of sets of single-user preferences, based on social choice theory [17].
- Based on an algorithm for the above preference merging and aggregation, we give algorithms for answering k-rank queries for DAQs (disjunctions of atomic queries), which generalize top-k queries based on the iterative computation of classical skyline answers. We show that answering DAQs in GPP-Datalog+/– is possible 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 recall some basics on Datalog+/–. Section 3 introduces the syntax and the semantics of GPP-Datalog+/–, in particular, the general group preference model and the probabilistic model, along with preference merging and aggregation operations. In Section 4, we present algorithms for k-rank query answering, along with correctness and data tractability results. Section 5 summarizes the main results of this paper and gives an outlook on future research.

2 Preliminaries

We first recall some basics on Datalog+/– [7], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple- and equality-generating dependencies (TGDs and EGDs, respectively), 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 [3], even when the schema and TGDs are fixed [6]. 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 of the form $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$, called the *body* of γ, denoted $body(\gamma)$, 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 of the form $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of σ, denoted $body(\sigma)$, is a conjunction of atoms (without nulls), 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 exists 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* [7]. 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 the EGDs do not have any impact on 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 [7] for an extended chase with also EGD chase rules).

TGD Chase Rule. Let D be a database, and σ be 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, 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 $\mathit{chase}^k(D, \Sigma)$ 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 $B \in mods(D, \Sigma)$ [7]. 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* $O = (D, \Sigma)$, 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 O is *guarded* iff Σ_T is guarded. The following example shows a simple Datalog+/– ontology.

Example 1. Let $O = (D, \Sigma)$ be an ontology describing travel activities:

$$\begin{aligned}\Sigma \;=\; &\{\mathit{museum}(X) \to \mathit{SS}(X),\ \mathit{park}(A) \to \mathit{SS}(A),\ \mathit{SS}(A) \to \mathit{act}(A),\\ &\ \mathit{relax}(X) \to \mathit{act}(X),\ \mathit{adv}(X) \to \mathit{act}(X),\ \mathit{sport}(X) \to \mathit{act}(X)\};\\ D \;=\; &\{\mathit{sport}(s_1),\ \mathit{sport}(s_2),\ \mathit{relax}(r_1),\ \mathit{relax}(r_2),\ \mathit{adv}(a_1),\\ &\ \mathit{adv}(a_2),\ \mathit{museum}(m_1),\ \mathit{museum}(m_2),\ \mathit{park}(p_1)\}.\end{aligned}$$

This ontology models a very simple travel itinerary domain, which may be used as the underlying model in an online travel agency. Activities can be either sightseeing (e.g., visiting museums or parks), relaxing (e.g., sauna), adventure (e.g., bungee jumping), or sport. The database D provides some instances for each kind of activities. ∎

3 GPP-Datalog+/–

In this section, we introduce the GPP-Datalog+/– language, an extension of Datalog+/– with both a group preference model and a probabilistic model. 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 $\mathcal{R}_{Pref} \subseteq \mathcal{R}_{Ont}$, $\Delta_{Pref} \subseteq \Delta_{Ont}$, $\mathcal{V}_{Pref} \subseteq \mathcal{V}_{Ont}$. We denote the corresponding *Herbrand bases* (the sets of all possible ground atoms) 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 $\mathcal{H}_{Ont}$.

Group Preference Model. 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 relations that is especially compatible with our approach is the preference formula framework of [8]. A *user preference model* U induces a preference relation over a subset of $\mathcal{H}_{Ont}$, denoted $\succ_U$; in general, we treat $\succ_U$ as a set of ordered pairs. A preference relation $\succ$ is *score-based* iff it is induced by an assignment of a numeric score to each element in such a way that $a_1 \succ a_2$ iff $score(a_1) > score(a_2)$.

In this work, we assume the existence of a *group preference model*, where we intuitively have a group of n users, and each user has an associated preference model.

Definition 1. A *group preference model* $\mathcal{U} = (U_1, \ldots, U_n)$ for $n \geqslant 1$ users is a collection of n user preference models.

Example 2. In the running example, a user may specify a preference relation over the *act* atoms, such as, e.g., shown in Fig. 1 (where we assume the transitive closure of the graphs). For instance, the preferences reflect that the user u_1 is a more sporty person and prefers sport and adventure activities above all other activities. His most preferred activity is $act(s_1)$, i.e., he prefers to practice sport s_1 over all other activities. The group preference model consists of the preferences models of the users u_1, u_2, and u_3. ∎

Probabilistic Model. 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 adopted here was first introduced in [11,10].

We use the standard notions of substitutions and most general unifiers. More specifically, 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 2. Let M be a probabilistic model. Then, a *(probabilistic) annotation* λ relative to M is a (finite) set of expressions of the form $A = x$, where (i) A is an atom over $\mathcal{R}_M$, $\mathcal{V}_M$, and Δ_M, and (ii) $x \in \{0, 1\}$. A probabilistic annotation is *valid* iff for any two different $A = x$, $B = y \in \lambda$, there exists no substitution that unifies A and B.

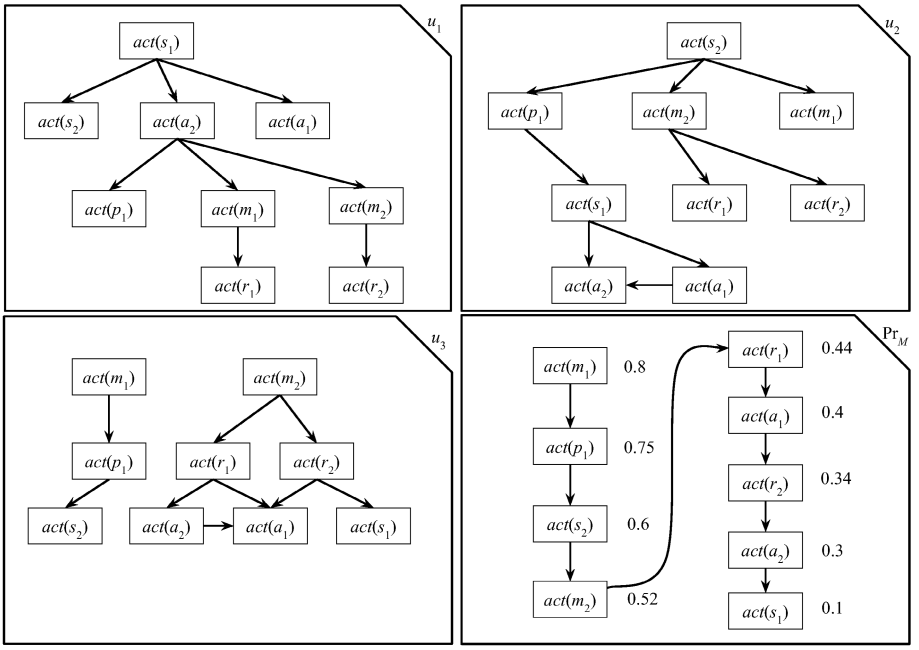

Fig. 1. Group preferences for Example 1

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 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 $A = x_i \in \lambda$ 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 that an online trip booking system consults a probabilistic model that assigns probabilities specifying how likely it is that certain events happen. This uncertainty could arise for instance from the fact that the system is aggregating information from multiple sources, which may contain conflicting information, as well as uncertainty due to other factors. Such a system could inform the user of the probability of an activity being recommended at the time of the query by taking into account reviews, crowds, season, etc. 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: $act(X)\colon \{recommended(X, d) = 1\}$, where $recommended(X, d)$ denotes the probabilistic event that activity X is suitable for the specified date d. Fig. 1 gives an example of such a probability assignment, along with the preference relation as a graph that is induced by these values, assuming that higher probabilities are more preferable. The probabilistic model Pr_M assigns to $act(m_1)$ the highest probability, while it assigns to $act(s_1)$ the lowest. ∎

Preference Merging and Aggregation. There are two challenges encountered in GPP-Datalog+/– ontologies, as seen in Fig. 1. The first challenge is that each user preference model yields a certain precedence relation that might be in disagreement with the one induced by the probabilistic model. The second challenge is that the user preference models may be in disagreement with each other. To address the first challenge, we introduce the notion of *preference merging operators*, which take two preference relations and produce a third one satisfying two basic properties as stated below.

Definition 3. Let $\succ_U$ be an SPO and $\succ_M$ be a score-based preference relation. A *preference merging operator* $\otimes(\succ_U, \succ_M)$ yields a relation $\succ^*$ such that (i) $\succ^*$ is an SPO, and (ii) if $a_1 \succ_U a_2$ and $a_1 \succ_M a_2$, then $a_1 \succ^* a_2$.

The two properties required by Definition 3 are the minimal required to produce a "reasonable" merging of the two relations. To address the second challenge, we define *preference aggregation operators*, which take a set of preference models and produce a new preference model. We next define aggregations of single-user preference models.

Definition 4. Let $\mathcal{U} = (U_1, \ldots, U_n)$ be a group preference model, where every U_i is an SPO. A *preference aggregation operator* $\biguplus$ on $\mathcal{U}$ yields an SPO $\succ^*$,

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

Definition 5. A *GPP-Datalog+/– ontology* has the form $KB = (O, \mathcal{U}, M, \otimes, \biguplus)$, where O is a Datalog+/– ontology, $\mathcal{U} = (U_1, \ldots, U_n)$ is a group preference model with $n \geqslant 1$, M is a probabilistic model (with Herbrand bases $\mathcal{H}_{Ont}$, $\mathcal{H}_{Pref}$, and $\mathcal{H}_M$, respectively, such that $\mathcal{H}_{Pref} \subseteq \mathcal{H}_{Ont}$), $\otimes$ is a preference merging operator, and $\biguplus$ is the preference aggregation operator. We say that KB is a *guarded* iff O is guarded.

Semantics. The semantics of GPP-Datalog+/– ontologies $KB = (O, \mathcal{U}, M, \otimes, \biguplus)$, with $\mathcal{U} = (U_1, \ldots, U_n)$, arises as a direct combination of the semantics of Datalog+/–, the group preference model, and the probabilistic model. As for the probabilistic model, we have that $\Pr_{KB}(a) = \sum_{\lambda \in scn(M),\, O_\lambda \models a} \Pr_M(\lambda)$, and we refer to the score-based preference relation induced by $\Pr_{KB}$ as the *probabilistic preference relation* associated with KB, denoted $\succ_M$. Then, $KB \models a_1 \succ^* a_2$ iff $O \models a_1, a_2$ and $a_1 \succ^* a_2$, where $\succ^* = \biguplus(\otimes(\succ_{U_1}, \succ_M), \ldots, \otimes(\succ_{U_n}, \succ_M))$. Intuitively, the consequences of KB are computed in terms of the classical consequences of O, and for each user i, a preference merging operator establishes a new preference over atoms in $\mathcal{H}_{Ont}$ that takes into account both $\succ_{U_i}$ and $\succ_M$; the final preference relation over pairs of atoms in $\mathcal{H}_{Ont}$ is then defined via the preference aggregation operator $\biguplus$. In the next section, we study different methods for answering k-rank queries under this aggregation operation.

4 Query Answering in GPP-Datalog+/–

In this section, we concentrate on skyline queries [4], 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; we refer to these as k-rank answers. We focus on a special kind of classical queries, called disjunctive atomic queries

(DAQs), which are disjunctions of atoms. We analyze two approaches to computing an answer to a k-rank query that are suitable for partially ordered sets of preferences. In the first one, called *collapse to single user (CSU)*, we reduce the group modeling problem to a single-user problem by creating a single virtual user that is constructed by aggregating the preferences of the individuals from the group – the k-rank is computed over the collapsed preference relation. In the second approach, called *voting-based aggregation*, we first compute the k-rankings according to each individual user and then apply aggregation techniques based on voting strategies (originally developed for quantitative preferences [17]) to aggregate the answers and obtain a single k-ranking.

We now define skyline and k-rank answers to a DAQ in GPP-Datalog+/– ontologies.

Definition 6. Let $KB = (O, \mathcal{U}, M, \otimes, \biguplus)$ be a GPP-Datalog+/– ontology, where $\mathcal{U} = (U_1, \ldots, U_n)$, 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 $\succ^* = \biguplus(\otimes(\succ_{U_1}, \succ_M), \ldots, \otimes(\succ_{U_n}, \succ_M))$ 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 $Q(\mathbf{X})$. For transitive relations, a k-rank answer to Q (where $k \geqslant 0$) is a sequence $S = \langle \theta_1, \ldots \theta_{k'} \rangle$ of maximal length of ground substitutions for the variables in $\mathbf{X}$, 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 skyline answers to Q remain.

In the sequel, we adopt the following preference merging operator $\otimes_t$ (or simply $\otimes$) to combine individual preferences with those based on scores. Given a relation $\succ_U$, score-based relation $\succ_M$, and value $t \in [0, 1]$ (that allows the user to choose how much influence the probabilistic model has on the output preference relation), the operator works by iterating through all pairs (a, b) of elements in $\succ_U$ and, if (i) $\succ_M$ disagrees with $\succ_U$, (ii) the difference in score is greater than t, and (iii) changing (a, b) to (b, a) does not introduce a cycle in the associated graph then the pair is inserted in reverse order into the output; otherwise, the output contains the same pair as $\succ_U$. Finally, the operator outputs the transitive closure of this relation. The following result shows that the $\otimes_t$ is indeed a preference merging operator.

Proposition 1. *Let $\succ_U$ be an SPO, $\succ_M$ be a score-based preference relation, and $t \in [0, 1]$. Then, $\otimes_t$ as defined above is a preference merging operator.*

We now explore two different approaches to a preference aggregation operator $\biguplus$.

4.1 Collapse to Single User

Under the CSU strategy, the preference relation for all users, along with the probabilistic preference relation, are taken into account in the generation of a new preference relation that encodes the dominant preferences. This single-user preference relation is then used to compute the answers to queries. In the following, for $a, b \in \mathcal{H}_{Ont}$ and SPOs $U_1, \ldots, U_n$, let $\#(a, b) = |\{(a, b) \mid a \succ_{U_i} b \text{ with } 1 \leqslant i \leqslant n\}|$.

Definition 7. Let $U_1, \ldots, U_n$ be $n \geqslant 1$ SPOs. Then, $\biguplus(U_1, \ldots, U_n)$ is a set of pairs of ground atoms $a, b \in \mathcal{H}_{Ont}$ such that: (i) $\#(a, b) > \#(b, a)$, and (ii) there does not exist

Algorithm 1: AggPrefsCSU$(\succ_M, \succ_{U_1}, \ldots, \succ_{U_n}, t)$
Input: SPOs $(\succ_{U_1}, \ldots, \succ_{U_n})$, score-based $\succ_M$ over $\mathcal{H}_{Ont}$, and $t = (t_1, \ldots, t_n) \in [0,1]^n$.
Output: Preference relation $\succ^* \subseteq \mathcal{H}_{Ont} \times \mathcal{H}_{Ont}$.

1. Initialize G as an empty graph;
2. Add as nodes in G all elements appearing in the preference relations $\succ_{U_i}$;
3. For every user $i \in \{1, \ldots, n\}$ do
4. Initialize *currUserG* as the graph corresponding to $\succ_{U_i}$;
5. For every pair $(a, b) \in \succ_{U_i}$ do
6. if $(score(b) - score(a) > t_i)$ and changing (a, b) to (b, a) in *currUserG* does not introduce a cycle then
7. remove edge (a, b) from *currUserG* and add edge (b, a);
8. *currUserG*:= *transitiveClosure*(*currUserG*);
9. For every edge (s, t) in *currUserG* do
10. if there is no edge (s, t) in G then
11. add edge (s, t) to G and label it with 1;
12. if there is an edge (s, t) in G and it is labeled with $n \geqslant 1$ then
13. increase the label of edge (s, t) in G by 1;
14. if there is an edge (t, s) in G and it is labeled with 1 then
15. remove edge (s, t) from G;
16. if there is an edge (t, s) in G and it is labeled with $n > 1$ then
17. decrease the label of edge (t, s) in G by 1;
18. return *inducedPreferenceRelation*(*removeCycles*(*transitiveClosure*(G))).

Fig. 2. An algorithm for combining the relations in a group preference model with a probabilistic preference relation

$c, d \in \mathcal{H}_{Ont}$ such that $\#(c, d) > \#(d, c)$ and the graph associated with $\biguplus(U_1, \ldots, U_n) \cup \{(c, d)\}$ is cycle-free.

Intuitively, $\biguplus$ compares the numbers of users in $\mathcal{U}$ that prefer a over b with those that prefer b over a. The following algorithm computes the $\biguplus$ operator; below, we provide an algorithm that uses this operator for answering k-rank queries to GPP-Datalog+/– ontologies in polynomial time in the data complexity (modulo the cost of computing probabilities with respect to the probabilistic model M).

Algorithm AggPrefsCSU**.** The algorithm in Fig. 2 implements a preference aggregation operator using a vector of values $t = (t_1, \ldots, t_n) \in [0,1]^n$, where t_i defines how much influence user i wishes to assign to the probabilistic model. The output is a new preference relation consisting of the collapsed preferences of all the users. A graph is used as an intermediate data structure representing the collapsed preferences; the nodes of this graph are all the atoms that appear in the preference relations, while the edges are labeled with an integer representing the number of users that have this edge in their individual preference relation. The algorithm iterates through all the users i and, by inspecting all pairs of elements (a, b) in their preference relations, builds the graph output by the $\otimes_t$ operator (lines 6–8), called *currUserG*. Then, before continuing with the next user, the algorithm looks at all the edges in *currUserG* and updates the general graph G by incrementing or decrementing the edge labels and introducing or removing

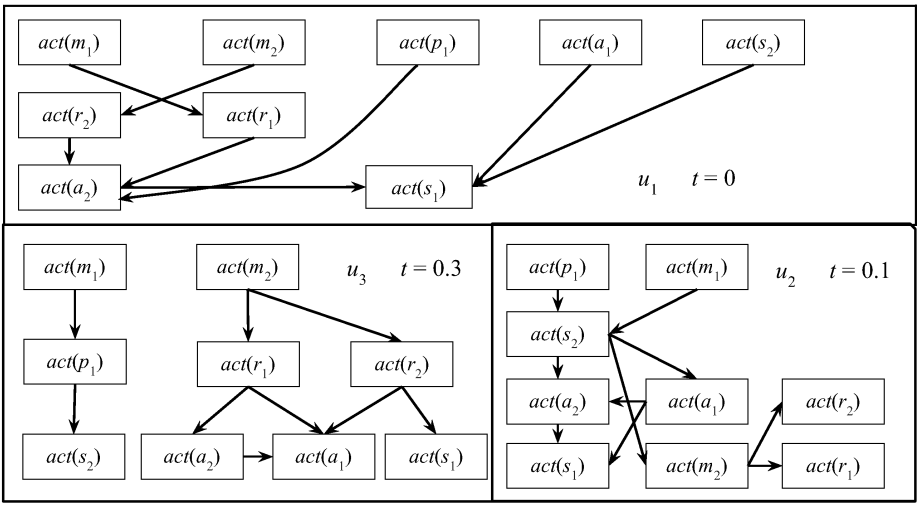

Fig. 3. The merged preference relation obtained for each user. The three graphs show the merging of each individual preference with the probabilistic preference.

edges. After the final iteration of the `for` loop in line 3 the edge labels of G correspond to the number of users that have that edge in their preference relation after combining it with the probabilistic one according to t. The final step of the algorithm computes the transitive closure of the graph and eliminates any cycles by applying the procedure *removeCycles* (note that cycles can arise even though all individual relations are cycle-free). We say that this subroutine does not unnecessarily remove edges if there does not exist an edge e in G such that *removeCycles*$(G) \cup \{e\}$ does not contain cycles.

Example 4. Consider again the running example. Fig. 3 shows the result of the individual mergings of the preference relation for each user with the score-based relation using $t = 0$ for u_1, $t = 0.1$ for u_2, and $t = 0.3$ for u_3. Fig. 4 shows the final collapsed graph. Consider the atoms $act(a_1)$ and $act(a_2)$ in this graph. Observe that the user u_2 prefers $act(a_1)$ to $act(a_2)$ after merging with $\succ_M$, but the user u_3 maintains that $act(a_2)$ is preferable (although $act(a_1) \succ_M act(a_2)$, the threshold for u_3 is higher than the difference in probability). Therefore, there is no edge between $act(a_1)$ and $act(a_2)$ in the final graph, since these two results cancel each other out. ∎

The following theorem states several properties satisfied by the output of AggPrefsCSU under certain conditions: (i) the output is an SPO, (ii) if all preference relations (including the probabilistic one) agree on the ordering of a pair of atoms, then the same ordering appears in the output, and (iii) for $t = 0^n$, the output only depends on the ordering given by $\succ_M$ (and not on the actual probabilities).

Theorem 1. *Let* $KB = (O, \mathcal{U}, M, \otimes, \biguplus)$ *be a GPP-Datalog+/– ontology, where* $\mathcal{U} = (U_1, \ldots, U_n)$*, let* Q *be a DAQ,* $k \geqslant 1$*, and* $t = (t_1, \ldots, t_n) \in [0,1]^n$*. Let* $\succ^* = \mathsf{AggPrefsCSU}(\succ_M, \succ_{U_1}, \ldots, \succ_{U_n}, t)$*. Then, (i) if removeCycles preserves transitivity, then*

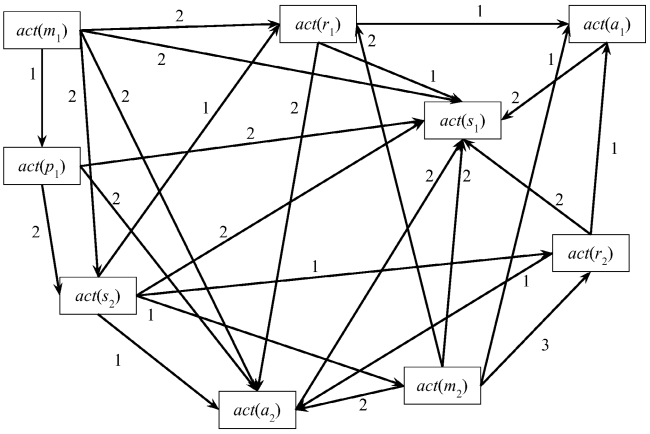

Fig. 4. Collapse to single user graph

$\succ^*$ *is an SPO; (ii) if removeCycles only removes edges* (v_1, v_2) *whenever there does not exist another edge in the cycle labeled with a lower number then, given* $a_1, a_2 \in \mathcal{H}_{Ont}$ *such that for all* $U_i \in \mathcal{U}$*, it holds that* $(a_1, a_2) \in \otimes_{t_i}(\succ_{U_i}, \succ_M)$*, then we have that* $a_1 \succ^* a_2$*; and (iii) if* M' *is a probabilistic model such that* $\succ_M = \succ_{M'}$ *and* $t = 0^n$*, then* $\succ^* = \mathsf{AggPrefsCSU}(\succ_{M'}, \succ_{U_1}, \ldots, \succ_{U_n}, t)$.

Algorithm k-Rank-CSU**.** We now present an algorithm to compute k-rank answers according to Definitions 6 and 7; we also analyze its correctness as well as running time when used in conjunction with the AggPrefsCSU algorithm. Fig. 5 shows the pseudocode for the algorithm, which begins by computing for every user the combination of the two preference relations in the GPP-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 may add superfluous elements.

Example 5. Consider the running example, with $Q = act(X)$, $k = 5$, and $(t_1, t_2, t_3) = (0, 0.1, 0.3)$. One possible k-rank answer to Q (in atom form) as computed by the algorithm is: $\langle act(m_1), act(p_1), act(s_2), act(m_2), act(r_2)\rangle$. ∎

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

Theorem 2. *Let* $KB = (O, \mathcal{U}, M, \otimes, \biguplus)$ *be a GPP-Datalog+/– ontology,* Q *be a DAQ, and* $k \geqslant 0$*. If* O *is a guarded Datalog+/– ontology and the removeCycles subroutine does not unnecessarily remove any edges, then Algorithm* k-Rank-CSU *correctly computes* k*-rank answers to* Q *in* $O(poly(|D|) \cdot S + C)$ *time in the data complexity, where* S *is the cost of computing score*$(a) = \Pr_{KB}(a)$ *for any atom* a *such that* $O \models a$*, and* C *is the cost of removeCycles.*

Algorithm 2: k-Rank-CSU$(KB = (O, \mathcal{U}, M, \otimes, \uplus), Q, k, t)$
Input: Guarded GPP-Datalog+/– ontology KB, DAQ $Q(\mathbf{X})$, $t \in [0,1]^n$, and $k \geqslant 0$.
Output: k-rank answer $\langle a_1, \ldots, a_{k'} \rangle$ to Q, with $k' \leqslant k$.

1. Initialize Res as an empty vector of ground atoms;
2. Set $\succ^*$:= AggPrefsCSU$(\succ_M, \succ_{U_1}, \ldots, \succ_{U_n}, t)$;
3. C:= *computeChase*(O, Q);
4. i:= k;
5. While $i > 0$ and $C \neq \emptyset$ do
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. Return $truncate(Res, k)$.

Fig. 5. An algorithm for computing a k-rank answer to DAQ Q using the CSU strategy

Note that the running time depends on the cost of the *removeCycles* subroutine. Though cycles can be removed in polynomial time, depending on the properties that we wish the output of this subroutine to satisfy, the actual cost may vary considerably.

4.2 Voting-Based Preference Aggregation

As an alternative to the approach described in the previous section, we now briefly discuss specific strategies that can be used to combine the answers to k-rank queries computed individually for each user based on a small set of well-known voting mechanisms from the social choice literature. Recall that this is essentially different from the CSU approach above, where a single k-ranking is computed from a preference relation distilled from all the users' individual preferences. We consider the following voting mechanisms: *plurality voting*, where each user votes for their top-preferred items, the items' frequency for all the users are summed up, and the items with highest number of votes win; the *least misery* strategy first removes from consideration the elements that are the least preferred by each user, and then applies plurality voting — the idea behind it is that a group is as happy as its least happy member; in the *average without misery* strategy, the least misery approach is generalized by removing the t least liked elements for each member (instead of just one); and the *fairness strategy*, which is often applied when people try to fairly divide a set of items — one person chooses first, then another, until everyone has made one choice, and next, everybody chooses a second item, often starting with the person who had to choose last on the previous round; an advantage of this strategy is that the top items from all individuals are always selected.

Integrating the voting-based aggregation strategies presented above into an algorithm for computing k-rank answers can be done by leveraging the k-Rank-CSU algorithm from the previous section: an answer can then be computed by calling k-Rank-CSU for each individual user (this will only merge the individual preferences of each user with the score-based relation — i.e., this step implements the application of the $\otimes_t$ merging operator). Since rankings are total orders, we can think of them as preference relations; therefore, the aggregation strategy chosen will lead to a specific implementation of the

$\biguplus$ preference aggregation operator: plurality voting will simply tally the number of votes received for each atom from the individual rankings. If a *misery* strategy is used, then it is necessary to identify all the nodes that are undesired for each user and mark them as unavailable before obtaining the individual rankings. On the other hand, if *fairness* is adopted, it is necessary to iterate through the 1-rank answers to the query for each user, and directly build the output k-tuple. The following corollary to Theorem 2 states that k-rank answers to DAQs using voting-based aggregation, as discussed above, can be computed in polynomial time in the data complexity.

Corollary 1. *Let $KB = (O, \mathcal{U}, M, \otimes, \biguplus)$ be a GPP-Datalog+/– ontology, Q be a DAQ, and $k \geqslant 0$. If O is guarded, and the voting-based aggregation $\biguplus$ can be computed in polynomial time in the data complexity, then a k-rank answer to Q using voting-based aggregation can be computed in $O(poly(|D|) \cdot S)$ time in the data complexity, where S is the cost of computing score$(a) = \Pr_{KB}(a)$ for any atom a such that $O \models a$.*

Here, the computational cost depends on the implementation of the voting strategies, which can clearly be computed in polynomial time in the data complexity for the strategies discussed above, and the computational cost of calling k-Rank-CSU for each user. Since k-Rank-CSU is only ever called with a single user, cycles can never arise; this is why the C factor from Theorem 2 does not appear.

5 Summary and Outlook

In this paper, we have proposed an extension of the Datalog+/– ontology language that allows for dealing with both partially ordered preferences of groups of users and probabilistic uncertainty. We have focused on answering k-rank queries in this context. In detail, we have presented different operators to compute group preferences as a merging and an aggregation of the preferences of single users with probability-based preferences and with each other, respectively. We have then provided algorithms to answer k-rank queries for DAQs (disjunctions of atomic queries) under these group preferences. We have shown that, under certain reasonable conditions, such DAQ answering in Datalog+/– can be done in polynomial time in the data complexity.

Current and future work involves implementing and testing the GPP-Datalog+/– framework. Furthermore, we want to explore which of the merging/aggregation operators is similar to human judgment and thus well-suited as a general default merging/aggregation operator for search and query answering in the Social Semantic Web.

Acknowledgments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J008346/1 "PrOQAW: Probabilistic Ontological Query Answering on the Web", the European Research Council (FP7/2007-2013)/ERC grant 246858 ("DIADEM"), by a Yahoo! Research Fellowship, and by a Google European Doctoral Fellowship.

References

1. Ackerman, M., Choi, S.Y., Coughlin, P., Gottlieb, E., Wood, J.: Elections with partially ordered preferences. Public Choice (2012)

2. Amer-Yahia, S., Roy, S.B., Chawla, A., Das, G., Yu, C.: Group recommendation: Semantics and efficiency. Proc. VLDB Endow. 2(1), 754–765 (2009)
3. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 73–85. Springer, Heidelberg (1981)
4. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. ICDE 2001, pp. 421–430. IEEE Computer Society (2001)
5. Brafman, R.I., Domshlak, C.: Preference handling — An introductory tutorial. AI. Mag. 30(1), 58–86 (2009)
6. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. KR 2008, pp. 70–80. AAAI Press (2008)
7. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
8. Chomicki, J.: Preference formulas in relational queries. ACM Trans. Database Syst. 28(4), 427–466 (2003)
9. Finger, M., Wassermann, R., Cozman, F.G.: Satisfiability in $\mathcal{EL}$ with sets of probabilistic ABoxes. In: Proc. DL 2011. CEUR-WS.org (2011)
10. Gottlob, G., Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Query answering under probabilistic uncertainty in Datalog+/– ontologies. Ann. Math. Artif. Intell. (in press, 2013)
11. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Answering threshold queries in probabilistic Datalog+/– ontologies. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 401–414. Springer, Heidelberg (2011)
12. 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)
13. Lang, J., Pini, M.S., Rossi, F., Salvagnin, D., Venable, K.B., Walsh, T.: Winner determination in voting trees with incomplete preferences and weighted votes. Auton. Agent. Multi-Ag. 25(1), 130–157 (2012)
14. Linden, G., Smith, B., York, J.: Industry report: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Distributed Systems Online 4(1) (2003)
15. Lukasiewicz, T., Martinez, M.V., Orsi, G., Simari, G.I.: Heuristic ranking in tightly coupled probabilistic description logics. In: Proc. UAI 2012, pp. 554–563. AUAI Press (2012)
16. Manoj, M., Jacob, E.: Information retrieval on internet using meta-search engines: A review. Journal of Scientific and Industrial Research 67(10), 739–746 (2008)
17. Masthoff, J.: Group modeling: Selecting a sequence of television items to suit a group of viewers. User Modeling and User-Adapted Interaction 14(1), 37–85 (2004)
18. Morris, M.R.: Collaborative search revisited. In: Proc. CSCW 2013, pp. 1181–1192. ACM Press (2013)
19. 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)
20. Ntoutsi, I., Stefanidis, K., Norvag, K., Kriegel, H.-P.: gRecs: A group recommendation system based on user clustering. In: Lee, S.-G., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part II. LNCS, vol. 7239, pp. 299–303. Springer, Heidelberg (2012)
21. Pattanaik, P.K.: Voting and Collective Choice: Some Aspects of the Theory of Group Decision-making. Cambridge University Press (1971)
22. Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Aggregating partially ordered preferences. J. Log. Comput. 19(3), 475–502 (2009)
23. Taylor, A.D.: Social Choice and the Mathematics of Manipulation. Cambridge University Press (2005)
24. Wooldridge, M.: An Introduction to Multiagent Systems. Wiley (2009)