Flexible Query Answering with the powerset-AI Operator and Star-Based Ranking

Lena Wiese(✉)

Institute of Computer Science, University of Göttingen,
Goldschmidtstraße 7, 37077 Göttingen, Germany
lena.wiese@uni-goettingen.de

Abstract. Query generalization is one option to implement flexible query answering. In this paper, we introduce a generalization operator (called powerset-AI) that extends conventional Anti-Instantiation (AI). We analyze structural modifications imposed by the generalization to obtain syntactic similarity measures (based on the star feature) that rank generalized queries with regard to their closeness to the original query.

1 Introduction

Flexible query answering supports users in the search for data in databases or information systems. Users might be unaware of the exact structure of the data and hence an exact formulation of the query conditions is often difficult. However, when using a conventional database system it tries to exactly answer the user query. In case the database system is not able to find an exactly matching answer, the query *fails* and the database system returns an empty answer. This is undesirable for the user, because he has to find alternative ways to express his query in order to receive some information. *Flexible* query answering systems offer intelligent procedures to revise user queries and are able to return answers that are related to the user's original query intent. However, one problem of flexible query answering is *overabundance* of related answers: too many answers are returned that might even be irrelevant for the user.

In this paper, we focus on query generalization of logical queries with the Anti-Instantiation (AI) operator. We extend the basic AI operator used in prior work by defining a novel *powerset-AI* operator. Queries resulting from powerset-AI retain more equality conditions than the conventional AI operator. To address the overabundance problem, we propose a ranking based on the amount of equality conditions retained. In this way, the higher-ranked queries can be answered by the database system with priority while the lower-ranked ones are assumed to be less important for the user. In this paper we make the following contributions:

– We extend the conventional AI operator by introducing our novel powerset-AI operator that allows to obtain a greater set of syntactically and semantically distinct generalized queries than conventional AI.

H. Christiansen et al. (Eds.): FQAS 2017, LNAI 10333, pp. 36–48, 2017.
DOI: 10.1007/978-3-319-59692-1_4

– We apply the star feature (counting equality conditions between query conjuncts) of the original query and a generalized query to obtain a value for the closeness of the generalized formula to the original formula.

After surveying related work in Sect. 2, we introduce theoretical background in Sect. 3 and present the powerset-AI operator in Sect. 4. Syntactic similarity of queries is analyzed in Sects. 6 and 7 concludes the paper.

2 Related Work

The CoopQA system [1,2] applies three generalization operators to a conjunctive query (which – among others – can already be found in the seminal paper of Michalski [3]): **Dropping Condition** (DC) removes one conjunct from a query; **Anti-Instantiation** (AI) replaces a constant (or a variable occurring at least twice) in Q with a new variable y; **Goal Replacement** (GR) takes a rule from Σ, finds a substitution θ that maps the rule's body to some conjuncts in the query and replaces these conjuncts by the head (with θ applied).

The operators DC, AI and GR have been applied to obtain neighborhood proposals [4] or related answers guided by explanations [5] in multi-agent communication. Other approaches propose some form of semantic query generalization where terms a generalized according to their meaning; these methods can be combined with generalization operators, including ranked data tables [6], fuzzy logic [7], taxonomies [8] or clustering [9]. More generally, the term "cooperative database system" was for example used in [10] for a system called "CoBase" that relies on several type abstraction hierarchies (TAH) to relax queries and hence to return a wider range of answers. In a similar manner, Halder and Cortesi [11] employ abstraction of domains and define optimality of answers with respect to some user-defined relevancy constraints. The approach by Pivert et al. using fuzzy sets [12] analyzes cooperative query answering based on semantic proximity. Other related systems are Flex [13], Carmin [14] and Ishmael [15] that introduce and analyze dedicated generalization operators.

Hurtado et al. [16] relax RDF queries based on an ontology. Similarly, [17] introduce the two operators APPROX and RELAX in the query language SPARQL. [18] improve upon existing work of RDF relaxation by finding related answers without generating all relaxed queries. Query relaxation has also been investigated for XML queries. For example, [19] analyze expressiveness of generalization operators for (a fragment of) XPath queries. Similarly, [20] apply structural generalization of XML queries; the authors also provide a recent survey of related work on approximate XML query answering.

3 Background

In this paper we focus on flexible query answering for conjunctive queries. Throughout this article we assume a logical language $\mathscr{L}$ consisting of a finite set of predicate symbols (for example denoted *Ill*, *Treat* or P), a possibly infinite set

dom of constant symbols (for example denoted Mary or a), and an infinite set of variables (for example denoted x or y). A term is either a constant or a variable. The capital letter X denotes a vector of variables; if the order of variables in X does not matter, we identify X with the set of its variables and apply set operators – for example we write $y \in X$. We use the standard logical connectors conjunction $\wedge$, disjunction $\vee$, negation $\neg$ and material implication $\rightarrow$ and universal $\forall$ as well as existential $\exists$ quantifiers. An atom is a formula consisting of a single predicate symbol; a literal is an atom (a "positive literal") or a negation of an atom (a "negative literal"); a ground formula is one without variables; the existential (universal) closure of a formula ϕ is written as $\exists\phi$ ($\forall\phi$) and denotes the closed formula where all free variables are bound the respective quantifier.

Table 1. Health records

Ill	*PatientID*	*Diagnoses*
	Pete	Cough
	Mary	Flu
	Mary	Bronchitis
	Lisa	Asthma
	Lisa	Cough
	Tom	Cough

Treat	*PatientID*	*Prescription*
	Pete	Antitussive
	Mary	Inhalation
	Lisa	Inhalation
	Tom	Antitussive
	Tom	Throat Lozenge

A query formula Q is a conjunction of literals with some variables X occurring freely (that is, not bound by variables); that is, $Q(X) = L_{i_1} \wedge \ldots \wedge L_{i_n}$. By abuse of notation, we will also write $L_{ij} \in Q$ when L_{ij} is a conjunct in formula Q. A query $Q(X)$ is sent to a knowledge base Σ (a set of logical formulas) and then evaluated in Σ by a function *ans* that returns a set of answers containing instantiations of the free variables (in other words, a set of formulas that are logically implied by Σ); as we focus on the generalization of queries, we assume the *ans* function and an appropriate notion of logical truth given. A special case of a knowledge base can be a relational database with database tables; as for example in [21] we apply a closed world assumption that makes all information not contained in the database false. As already established in [22] we apply a notion of generalization based on a consequence operator $\models$ as follows.

Definition 1 (Deductive generalization wrt. knowledge base [22]). *Let Σ be a knowledge base, $\phi(X)$ be a formula with a tuple X of free variables, and $\psi(X, Y)$ be a formula with an additional tuple Y of free variables disjoint from X. The formula $\psi(X, Y)$ is a* deductive generalization *of $\phi(X)$, if it holds in Σ that the less general ϕ implies the more general ψ where for the free variables X (the ones that occur in ϕ and possibly in ψ) the universal closure and for free variables Y (the ones that occur in ψ only) the existential closure is taken:*

$$\Sigma \models \forall X \exists Y \, (\phi(X) \rightarrow \psi(X, Y))$$

The three operators Dropping Condition (DC), Anti-Instantiation (AI) and Goal Replacement (GR) used in [2] satisfy the definition of deductive generalization (Definition 1), provided the consequence operator ($\models$) satisfies some reasonable properties. For example, the semantic consequence $\models$ should be implemented by syntactic entailment that satisfies *generalized modus ponens* (for GR), *conjunction elimination* (for DC) and *existential introduction* (for AI).

Example 1. As a running example, we consider a hospital information system that stores illnesses and treatments of patients (see Table 1). The example query $Q(x_1, x_2, x_3) = Ill(x_1, Flu) \wedge Ill(x_1, Cough)$ asks for all the patients x_1 that suffer from both flu and cough. This query fails with the given database tables as there is no patient with both flu and cough. However, the querying user might instead be interested in the patient called Mary who is ill with both flu and bronchitis. Query generalization (and in particular Anti-Instantiation) will enable a flexible query answering system to find this informative answer.

4 The powerset-AI Operator

Conventional Anti-Instantiation chooses one occurrence of a term (that is, a constant or variable) of a query and introduces a new variable y for. In this way the conditions of the query are relaxed. In particular, AI covers these cases:

- turning constants into variables: $P(a)$ is converted to $P(y)$ (see [3])
- breaking joins: $P(x) \wedge S(x)$ is converted to $P(x) \wedge S(y)$ (introduced in [22])

For each constant a all occurrences must be anti-instantiated; the same applies to variables x – however, with the exception that if x only occurs twice, one occurrence of x need not be anti-instantiated due to equivalence. Operator 1 lists the steps of conventional AI.

Operator 1. Anti-instantiation (AI)

Input: Query $Q(X) = L_1 \wedge \ldots \wedge L_n$ of length n
Output: Generalized query $Q^{gen}(X, Y)$ with Y containing one new variable y

1: From $Q(X)$ choose a term t such that t is

   - either a variable occurring in $Q(X)$ at least twice
   - or a constant

2: Choose one literal L_j where t occurs
3: Let L'_j be the literal with one occurrence of t replaced with y
4: **return** $Q^{gen}(X, Y) = L_1 \wedge \ldots \wedge L_{j-1} \wedge L'_j \wedge L_{j+1} \wedge \ldots \wedge L_n$

Example 2. For $Q(x_1) = Ill(x_1, Flu) \wedge Ill(x_1, Cough)$ an example generalization with AI is $Q^{AI}(x_1, y) = Ill(x_1, Flu) \wedge Ill(x_1, y)$. It results in an non-empty (and hence informative) answer: $Ill(Mary, Flu) \wedge Ill(Mary, Bronchitis)$.

The conventional AI operator only replaces a single occurrence of a term. We now introduce a novel AI operator that replaces several occurrences of a term at once. In doing so, different equality conditions can be retained – a property that is impossible with the conventional AI operator. We call the operator *powerset-AI* because first the powerset of all equality conditions is computed and then an element of this powerset (that is, a subset of equalities) is chosen for anti-instantiation. We first provide an example showing limits of conventional AI.

Example 3. Suppose a user is interested in four (possibly different) patients being treated with an inhalation. The user hence submits the query

$$\begin{aligned} Q'(x_1, x_2, x_3) = \; & \mathit{Treat}(x_1, \mathit{Inhalation}) \wedge \mathit{Treat}(x_2, \mathit{Inhalation}) \\ & \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}). \end{aligned}$$

Demanding that the x_i are pairwise distinct ($x_i \neq x_j$ for $i, j \in \{1, \ldots, 4\}$), the database cannot find an answer to this query and hence the query fails. Conventional Anti-Instantiation can replace only a single occurrence of *Inhalation* by a new variable y_1; for example

$$\begin{aligned} Q'_1(x_1, x_2, x_3, y_1) = \; & \mathit{Treat}(x_1, y_1) \wedge \mathit{Treat}(x_2, \mathit{Inhalation}) \\ & \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}) \end{aligned}$$

Again demanding that the x_i are pairwise distinct, the query Q_1 is still failing. A second application of conventional Anti-Instantiation can replace another occurrence of *Inhalation* by a new variable y_2; for example

$$\begin{aligned} Q'_2(x_1, x_2, x_3, y_1, y_2) = \; & \mathit{Treat}(x_1, y_1) \wedge \mathit{Treat}(x_2, y_2) \\ & \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}) \end{aligned}$$

In this case an answer can be found; however, no equality between y_1 and y_2 is expressed. For example, one answer for our table is $\mathit{Treat}(\mathit{Pete}, \mathit{Antitussive}) \wedge \mathit{Treat}(\mathit{Tom}, \mathit{Throat\ Lozenge}) \wedge \mathit{Treat}(\mathit{Mary}, \mathit{Inhalation}) \wedge \mathit{Treat}(\mathit{Lisa}, \mathit{Inhalation})$.

The powerset-AI operator we propose keeps the equality condition by replacing several occurrences of a term with the same new variable.

Example 4. Continuing the above example, we can simultaneously replace the first two occurrences of *Inhalation* with the same new variable y:

$$\begin{aligned} Q'_3(x_1, x_2, x_3, y) = \; & \mathit{Treat}(x_1, y) \wedge \mathit{Treat}(x_2, y) \\ & \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}) \end{aligned}$$

In this way the equality condition between the first and second occurrence is retained. With this query we can retrieve the answer $\mathit{Treat}(\mathit{Pete}, \mathit{Antitussive}) \wedge \mathit{Treat}(\mathit{Tom}, \mathit{Antitussive}) \wedge \mathit{Treat}(\mathit{Mary}, \mathit{Inhalation}) \wedge \mathit{Treat}(\mathit{Lisa}, \mathit{Inhalation})$ without retrieving any less relevant answers as in the previous example.

We now formalize the approach of powerset-AI. First we need the notion of occurrences of a term t.

Definition 2 (Occurrences of a term). *If a term t occurs k times in a query then the set of occurrences in the (left-to-right) order of appearance in the query is defined as $\mathcal{O}_t = \{t.1, \ldots, t.k\}$.*

For the set of occurrences $\mathcal{O}_t$ the powerset $\rho(\mathcal{O}_t)$ contains all $2^{|\mathcal{O}_t|}$ subsets. When anti-instantiating a *constant*, all non-empty subsets of $\rho(\mathcal{O}_t)$ can be chosen leading to semantically different generalized queries. In other words, we consider the sets $\mathcal{P}_t = \{\mathcal{S} \in \rho(\mathcal{O}_t) \mid 1 \leq |\mathcal{S}| \leq |\mathcal{O}_t|\}$ for anti-instantiation with a new variable y.

When anti-instantiating a *variable* with at least two occurrences, the situation is different. For such variables it suffices to consider only the sets of occurrences of size up to half of the total amount of occurrences. More formally, we consider $\mathcal{P}_t = \{\mathcal{S} \in \rho(\mathcal{O}_t) \mid 1 \leq |\mathcal{S}| \leq \lfloor \frac{|\mathcal{O}_t|}{2} \rfloor\}$ for anti-instantiation with a new variable y. The reason for this is that all anti-instantiations considering larger sets of occurrences (that is, sets $\{\mathcal{S} \in \rho(\mathcal{O}_t) \mid \lceil \frac{|\mathcal{O}_t|}{2} \rceil \leq |\mathcal{S}| \leq |\mathcal{O}_t|\}$) lead to queries that are equivalent (more precisely, identical up to variable renaming) to one considering one set from $\mathcal{P}_t$. Operator 2 lists the steps of powerset-AI.

Operator 2. powerset Anti-instantiation (powerset-AI)

Input: Query $Q(X) = L_1 \wedge \ldots \wedge L_n$ of length n
Output: Generalized query $Q^{gen}(X, Y)$ with Y containing one new variable y

1: From $Q(X)$ choose a term t such that t is

   - either a variable occurring in $Q(X)$ at least twice
   - or a constant

2: Let $\mathcal{O}_t$ be the set of all occurrences of t and $\rho(\mathcal{O}_t)$ the powerset
3: If t is a constant, compute the set $\mathcal{P}_t = \{\mathcal{S} \in \rho(\mathcal{O}_t) \mid 1 \leq |\mathcal{S}| \leq |\mathcal{O}_t|\}$
4: If t is a variable, compute the set $\mathcal{P}_t = \{\mathcal{S} \in \rho(\mathcal{O}_t) \mid 1 \leq |\mathcal{S}| \leq \left\lfloor \frac{|\mathcal{O}_t|}{2} \right\rfloor\}$
5: Choose one set $\mathcal{S} \in \mathcal{P}_t$ of occurrences
6: Let $L_{i_1}, \ldots, L_{i_m}$ denote the literals of $Q(X)$ containing the occurrences in $\mathcal{S}$
7: Let $L'_{i_1}, \ldots, L'_{i_m}$ denote the literals with each occurrence of t in $\mathcal{S}$ replaced with the new variable y
8: Let $L_{i_{m+1}}, \ldots, L_{i_n}$ denote the literals of $Q(X)$ apart from $L_{i_1}, \ldots, L_{i_m}$
9: **return** $Q^{gen}(X, Y) = L'_{i_1} \wedge \ldots \wedge L'_{i_m} \wedge L_{i_{m+1}} \wedge \ldots \wedge L_{i_n}$

Example 5. Similar to the above example, we can replace occurrences of a variable contained in the original query with the same new variable y. Suppose a user is interested in a patient x with four specified diseases. The user hence submits the query

$$Q''(x) = Ill(x, Flu) \wedge Ill(x, Bronchitis) \wedge Ill(x, Asthma) \wedge Ill(x, Cough)$$

Replacing three occurrences of x by y leads to a query equivalent to one obtained by replacing one occurrence of x by y:

$$Q_1''(x, y) = Ill(x, Flu) \wedge Ill(y, Bronchitis) \wedge Ill(y, Asthma) \wedge Ill(y, Cough)$$

is identical to

$$Q_2''(x, y) = Ill(y, Flu) \wedge Ill(x, Bronchitis) \wedge Ill(x, Asthma) \wedge Ill(x, Cough)$$

only with the roles of x and y reversed. Moreover, a query that replaces two occurrences of x by y, e.g.

$$Q_3''(x, y) = Ill(x, Flu) \wedge Ill(x, Bronchitis) \wedge Ill(y, Asthma) \wedge Ill(y, Cough)$$

leads to a non-failing query and the answer $Ill(Mary, Flu) \wedge Ill(Mary, Bronchitis) \wedge Ill(Lisa, Asthma) \wedge Ill(Lisa, Cough)$ can be returned to the user.

We now show that the powerset-AI operator complies with Definition 1 independent of any specific knowledge base Σ.

Proposition 1. *powerset-AI is a deductive generalization operator.*

Proof. As in Operator 2, let $L_{i_1} \wedge \ldots \wedge L_{i_m}$ be the subquery containing occurrences of term t. Let $L'_{i_1} \wedge \ldots \wedge L'_{i_m}$ be the subquery with the occurrences of term t replaced by y. It holds that $\models \forall X \exists y \, (L_{i_1} \wedge \ldots \wedge L_{i_m} \rightarrow L'_{i_1} \wedge \ldots \wedge L'_{i_m})$. The same applies to the whole query: $\models \forall X \exists y \, (Q(X) \rightarrow Q^{gen}(X, Y))$.

Note that by choosing different sets $\mathcal{S}$ in a powerset-AI step a set of different output queries $Q^{gen}(X, Y)$ can be produced. Moreover, multiple powerset-AI steps can be executed in sequence by choosing a different t for each step. The question we want to analyze next is how to obtain a ranking on all these output queries based on their structural differences.

5 The Star Feature and Similarities

Features are properties that can be attributed to objects; in our case objects are queries. The similarity of two objects can then be determined by evaluating (for example, counting) the feature commonalities and differences. Here we focus on the so-called star feature. We borrow the definition of a star of a chosen literal from [23]. The star contains all predicate symbols of other literals that share a term with the chosen literal. In this way, the star can express connections between different literals. In particular, two occurrences of the same variable (inside different literals) correspond to a join condition of an equality join; and using the same constants corresponds to a join followed by a selection with the constant as the specific value required for the join attribute. Hence, losing one such shared variable or constant in a generalization corresponds to "breaking joins" (see [22]) which should be penalized with a lower similarity. Moreover, losing a literal with many connections is worse than losing a literal with few

connections to other literals. The star is the appropriate feature for this: stars of other literals are less affected by breaking a join with a literal with less connections. More formally, we denote $Terms(L_i, Q)$ the set of terms (that is, constants and variables) of literal L_i in a conjunctive query Q; moreover, let $Pred(Q)$ be the set of predicate symbols occurring in Q. For a given literal we use the following definition of its star (cf. [23]) resulting in a multiset of predicate symbols:

Definition 3. (Star of a literal [23]). *For a literal L_i in a given query Q we define the* star *of L_i to be a multiset of predicate symbols as follows*

$$\begin{aligned} Star(L_i, Q) = \{P \mid\ & \textit{there is a literal } L_j \in Q, i \neq j, \\ & \textit{such that } L_j = P(t_1, \ldots t_k) \textit{ and} \\ & Terms(L_j, Q) \cap Terms(L_i, Q) \neq \emptyset\} \subseteq Pred(Q) \end{aligned}$$

We compare the star feature of literals in the original query Q and a query Q^{gen} (which is obtained by one or more applications of the powerset-AI operator). We see that for each literal L_i of the original query Q the amount of connections to other literals is always greater or equal to the amount of connections of the corresponding literal L'_i in the anti-instantiated query. Hence, $Star(L'_i, Q^{gen}) \subseteq Star(L_i, Q)$.

Example 6. We compare the star of the first literal of

$$\begin{aligned} Q'(x_1, x_2, x_3) = & \, Treat(x_1,\ Inhalation)\ \wedge\ Treat\ (x_2,\ Inhalation) \\ & \wedge\ Treat\ (x_3,\ Inhalation)\ \wedge\ Treat\ (x_4,\ Inhalation) \end{aligned}$$

to the stars of its generalizations in the queries

$$\begin{aligned} Q'_1(x_1, x_2, x_3, y_1) = & \, Treat(x_1, y_1)\ \wedge\ Treat(x_2,\ Inhalation) \\ & \wedge\ Treat(x_3,\ Inhalation)\ \wedge\ Treat(x_4,\ Inhalation) \\ Q'_2(x_1, x_2, x_3, y_1, y_2) = & \, Treat(x_1, y_1)\ \wedge\ Treat(x_2, y_2) \\ & \wedge\ Treat(x_3,\ Inhalation)\ \wedge\ Treat(x_4,\ Inhalation) \\ Q'_3(x_1, x_2, x_3, y) = & \, Treat(x_1, y)\ \wedge\ Treat(x_2, y) \\ & \wedge\ Treat(x_3,\ Inhalation)\ \wedge\ Treat(x_4,\ Inhalation) \end{aligned}$$

We obtain $Star(Treat(x_1, Inhalation), Q') = \{Treat, Treat, Treat\}$ for the original query, $Star(Treat(x_1, y_1), Q'_1) = \emptyset$, $Star(Treat(x_1, y_1), Q'_2) = \emptyset$, as well as $Star(Treat(x_1, y), Q'_3) = \{Treat\}$.

One way to judge the relation of two objects is determining a similarity between them.

Definition 4 (Similarity Measure). *For a set of objects $\mathcal{O}$, a function $sim : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$ is called a similarity on $\mathcal{O}$, if for all $a, b \in \mathcal{O}$ it holds that*

1. ***(Non-negativity)*** $sim(a, b) \geq 0$
2. ***(Maximality)*** $sim(a, a) \geq sim(a, b)$

Based on feature sets of two objects a and b, similarity between these two objects can be calculated by means of different similarity measures. That is, if A is a feature set of a and B is the corresponding feature set of b, then $A \cap B$ is the set of their common features, $A \setminus B$ is the set of features that are only attributed to A, and $B \setminus A$ is the set of features that are only attributed to B. In Tversky's seminal paper [24], a function f is applied to each set such that it is mapped to a numerical value. Typically the cardinality function is used. We will also follow this approach in this paper. That is, we obtain the cardinalities of each set: $\mathfrak{l} = |A \cap B|$, $\mathfrak{m} = |A \setminus B|$, and $\mathfrak{n} = |B \setminus A|$ and use them as input to specific similarity measures. In this paper, we focus on the ratio model (in particular, one of its special cases called Jaccard index).

Definition 5 (Tversky's Ratio Model [24], Jaccard Index). *A similarity measure sim between two objects a and b can be represented by the ratio of features common to both a and b and the joint features of a and b using a non-negative scale f and two non-negative scalars α and β:*

$$sim_T(a, b) = \frac{f(A \cap B)}{f(A \cap B) + \alpha \cdot f(A \setminus B) + \beta \cdot f(B \setminus A)}$$

The Jaccard index *is a special form of the ratio model where $\alpha = \beta = 1$ and f is the cardinality $|\cdot|$:*

$$sim_{jacc}(a, b) = \frac{|A \cap B|}{|A \cap B| + |A \setminus B| + |B \setminus A|} = \frac{|A \cap B|}{|A \cup B|} = \frac{\mathfrak{l}}{\mathfrak{l} + \mathfrak{m} + \mathfrak{n}}$$

The Jaccard index satisfies the property of monotonicity (see [24]): whenever a formula has more features in common with one formula than with another formula and they differ on less features, then the similarity between the first two is higher than the similarity between the first and the last.

Definition 6 (Monotonicity [24]). *For three objects a, b and c, if $A \cap B \supseteq A \cap C$, $A \setminus B \subseteq A \setminus C$, and $B \setminus A \subseteq C \setminus A$, then $sim(a, b) \geq sim(a, c)$. When the inclusions are proper, the inequality is strict.*

6 Similarity for powerset-AI

Based on the star feature, we want to calculate the similarity between the original query Q and a query Q^{gen} (which is obtained by one or more applications of the powerset-AI operator). We calculate the Jaccard index for the feature sets by taking the cardinalities of their commonalities and differences: if A is a feature set of Q and B is the corresponding feature set of Q^{gen}, then $A \cap B$ is the set of their common features, $A \setminus B$ is the set of features that are lost during generalization, and $B \setminus A$ is the set of features that have been added during generalization.

Note that the star feature is a multiset and we define the operators $\cup$, $\cap$ and $\setminus$ to be multiset operations. To obtain similarity $sim(Q, Q^{gen})$ between the original query Q and a query Q^{gen} we proceed as follows:

- For each literal L_i in Q and the corresponding L'_i in Q^{gen} compute the star features $A = Star(L_i, Q)$ and $B = Star(L'_i, Q^{gen})$.
- Compute the literal similarities $sim_{jacc}(L_i, L'_i)$ for each such pair of literals.
- Compute the average by summing all literal similarities and dividing by the total amount n of literals in the query.

Example 7. We compute the stars of all literals of

$$\begin{aligned} Q'(x_1, x_2, x_3) = \mathit{Treat}(x_1, \mathit{Inhalation}) \wedge \mathit{Treat}(x_2, \mathit{Inhalation}) \\ \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}) \end{aligned}$$

and the stars of its generalizations in the query

$$\begin{aligned} Q'_1(x_1, x_2, x_3, y_1) = \mathit{Treat}(x_1, y_1) \wedge \mathit{Treat}(x_2, \mathit{Inhalation}) \\ \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}) \end{aligned}$$

to compute the literal similarities. We obtain for all literal pairs $sim_{jacc}(L_1, L'_1) = 0$, $sim_{jacc}(L_2, L'_2) = \frac{2}{3}$, $sim_{jacc}(L_3, L'_3) = \frac{2}{3}$, $sim_{jacc}(L_4, L'_4) = \frac{2}{3}$. By summing all literal similarities and dividing by 4 (the total amount of literals), we obtain the query similarity $sim(Q', Q'_1) = 0.5$ Similarly, for the query

$$\begin{aligned} Q'_2(x_1, x_2, x_3, y_1, y_2) = \mathit{Treat}(x_1, y_1) \wedge \mathit{Treat}(x_2, y_2) \\ \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}) \end{aligned}$$

we obtain $sim_{jacc}(L_1, L'_1) = 0$, $sim_{jacc}(L_2, L'_2) = 0$, $sim_{jacc}(L_3, L'_3) = \frac{1}{3}$, and $sim_{jacc}(L_4, L'_4) = \frac{1}{3}$. By summing all literal similarities and dividing by 4 (the total amount of literals), we obtain the query similarity $sim(Q', Q'_2) = \frac{1}{6}$. Lastly, for

$$\begin{aligned} Q'_3(x_1, x_2, x_3, y) = \mathit{Treat}(x_1, y) \wedge \mathit{Treat}(x_2, y) \\ \wedge \mathit{Treat}(x_3, \mathit{Inhalation}) \wedge \mathit{Treat}(x_4, \mathit{Inhalation}) \end{aligned}$$

we obtain $sim_{jacc}(L_1, L'_1) = \frac{1}{3}$, $sim_{jacc}(L_2, L'_2) = \frac{1}{3}$, $sim_{jacc}(L_3, L'_3) = \frac{1}{3}$, and $sim_{jacc}(L_4, L'_4) = \frac{1}{3}$. By summing all literal similarities and dividing by 4 (the total amount of literals), we obtain the query similarity $sim(Q', Q'_3) = \frac{1}{3}$.

In the example we see that the query Q'_3 where two occurrences of *Inhalation* are replaced by one new variable y is ranked better than the query Q'_2 where the same occurrences are replaced by two new variables y_1 and y_2. We now formally show this fact when replacing multiple occurrences.

Theorem 1. *Let Q be the original query containing term t. Let $\mathcal{S} \in \rho(\mathcal{O}_t)$ be the set of occurrences of t chosen for powerset-AI. Let $Q^{gen}(x, y)$ be a query obtained by anti-instantiating occurrences $\mathcal{S}$ of the term t with one new variable y. Let $Q^{AI}(x, y_1, \ldots, y_{|\mathcal{S}|})$ be a query obtained by anti-instantiating the same occurrences of term t with $|\mathcal{S}|$ new variables $y_1, \ldots, y_{|\mathcal{S}|}$. Then it holds that $sim_{jacc}(Q, Q^{AI}(x, y_1, \ldots, y_{|\mathcal{S}|})) \leq sim_{jacc}(Q, Q^{gen}(x, y))$.*

Proof. As in Operator 2, let $L_{i_1}, \ldots, L_{i_m}$ be the literals containing occurrences of term t that are replaced (Case 1). Let $L_{i_{m+1}}, \ldots, L_{i_n}$ be the literals unaffected by anti-instantiation (Case 2). We analyze the stars of the literals in these cases:

- Case 1: Let $L'_{i_1}, \ldots, L'_{i_m}$ be the literals in $Q^{gen}(x, y)$ with the occurrences of term t replaced by y. Let $L''_{i_1}, \ldots, L''_{i_m}$ be the literals in $Q^{AI}(x, y_1, \ldots, y_{|\mathcal{S}|})$ with the occurrences of term t replaced by $y_1, \ldots, y_{|\mathcal{S}|}$. It holds that for $A = Star(L_{i_j}, Q)$ and $B = Star(L'_{i_j}, Q^{gen})$ and $C = Star(L''_{i_j}, Q^{AI})$ (where $j = 1, \ldots, m$) both A and B definitely contain predicate symbols of literals in $L_{i_1}, \ldots, L_{i_m}$: In Q, a literal L_{i_j} has connections to all other such literals based on t, while in $Q^{gen}(x, y)$ these connections are based on y. C does not necessarily contain them; however, if it contains such a predicate symbol then only due to some other term different from t and y; in this case also A and B necessarily contain this predicate symbol, too.
- Case 2: The literals $L_{i_{m+1}}, \ldots, L_{i_n}$ occur unmodified in both $Q^{gen}(x, y)$ and $Q^{AI}(x, y_1, \ldots, y_{|\mathcal{S}|})$, too. When comparing $A = Star(L_{i_j}, Q)$ and $B = Star(L_{i_j}, Q^{gen})$ and $C = Star(L_{i_j}, Q^{AI})$ (where $j = m+1, \ldots, n$), A, B and C are identical if L_{i_j} does not contain t. If L_{i_j} contains t, then A contains all predicate symbols of literals $L_{i_1}, \ldots, L_{i_m}$ due to connections based on t; in contrast, B and C are identical: they are both reduced by the predicate symbols of literals $L_{i_1}, \ldots, L_{i_m}$ (as long as no other occurrence of t remains in one such literal).

In both cases it follows that $A \cap B \supseteq A \cap C$, $A \setminus B \subseteq A \setminus C$, and $B \setminus A \subseteq C \setminus A$. Due to these facts we can apply the notion of monotonicity according to Definition 6 and the theorem follows.

Hence we conclude that the star feature is an appropriate measure to rank different queries obtained by applying several generalization steps (in particular, powerset-AI) to the same original query.

7 Discussion and Conclusion

We introduced powerset-AI as a novel generalization operator that replaces terms in a given query to obtain relaxed queries that provide related information. In order to restrict query answering to the most relevant queries (and hence avoid overabundance of answers), we proposed a ranking of these relaxed queries based on the star feature. Future work will investigate the application of powerset-AI in combination with other generalization operators (like GR and DC) and their joint behavior as in [2]. Ongoing work covers efficient computation of query relaxation with powerset-AI which includes duplicate checking and computing answers without generating all queries as in [18].

References

1. Bakhtyar, M., Dang, N., Inoue, K., Wiese, L.: Implementing inductive concept learning for cooperative query answering. In: Spiliopoulou, M., Schmidt-Thieme, L., Janning, R. (eds.) Data Analysis, Machine Learning and Knowledge Discovery. SCDAKO, pp. 127–134. Springer, Cham (2014). doi:10.1007/978-3-319-01595-8_14
2. Inoue, K., Wiese, L.: Generalizing conjunctive queries for informative answers. In: Christiansen, H., Tré, G., Yazici, A., Zadrozny, S., Andreasen, T., Larsen, H.L. (eds.) FQAS 2011. LNCS, vol. 7022, pp. 1–12. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24764-4_1
3. Michalski, R.S.: A theory and methodology of inductive learning. Artif. Intell. **20**(2), 111–161 (1983)
4. Sakama, C., Inoue, K.: Negotiation by abduction and relaxation. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), IFAAMAS, pp. 1010–1025 (2007)
5. Sá, S., Alcântara, J.: Abduction-based search for cooperative answers. In: Leite, J., Torroni, P., Ågotnes, T., Boella, G., Torre, L. (eds.) CLIMA 2011. LNCS, vol. 6814, pp. 208–224. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22359-4_15
6. Urbanova, L., Vychodil, V., Wiese, L.: Applications of ordinal ranks to flexible query answering. In: Hüllermeier, E., Link, S., Fober, T., Seeger, B. (eds.) SUM 2012. LNCS, vol. 7520, pp. 16–29. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33362-0_2
7. Belohlavek, R., Vychodil, V.: A logic of graded attributes. Arch. Math. Logic **54**(7–8), 785–802 (2015)
8. Wiese, L.: Taxonomy-based fragmentation for anti-instantiation in distributed databases. In: Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing International Workshop on Intelligent Techniques and Architectures for Autonomic Clouds (ITAAC13), pp. 363–368. IEEE Computer Society (2013)
9. Wiese, L.: Clustering-based fragmentation and data replication for flexible query answering in distributed databases. J. Cloud Comput. **3**(1), 18 (2014)
10. Chu, W.W., Yang, H., Chiang, K., Minock, M., Chow, G., Larson, C.: CoBase: a scalable and extensible cooperative information system. JIIS **6**(2/3), 223–259 (1996)
11. Halder, R., Cortesi, A.: Cooperative query answering by abstract interpretation. In: Černá, I., Gyimóthy, T., Hromkovič, J., Jefferey, K., Králović, R., Vukolić, M., Wolf, S. (eds.) SOFSEM 2011. LNCS, vol. 6543, pp. 284–296. Springer, Heidelberg (2011). doi:10.1007/978-3-642-18381-2_24
12. Pivert, O., Jaudoin, H., Brando, C., Hadjali, A.: A method based on query caching and predicate substitution for the treatment of failing database queries. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 436–450. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14274-1_32
13. Motro, A.: Flex: a tolerant and cooperative user interface to databases. IEEE Trans. Knowl. Data Eng. **2**(2), 231–246 (1990)
14. Godfrey, P., Minker, J., Novik, L.: An architecture for a cooperative database system. In: Litwin, W., Risch, T. (eds.) ADB 1994. LNCS, vol. 819, pp. 3–24. Springer, Heidelberg (1994). doi:10.1007/3-540-58183-9_35
15. Godfrey, P.: Minimization in cooperative response to failing database queries. IJCS **6**(2), 95–149 (1997)

16. Hurtado, C.A., Poulovassilis, A., Wood, P.T.: Query relaxation in RDF. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 31–61. Springer, Heidelberg (2008). doi:10.1007/978-3-540-77688-8_2
17. Selmer, P., Poulovassilis, A., Wood, P.T.: Implementing flexible operators for regular path queries. In: Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), CEUR Workshop Proceedings, vol. 1330, pp. 149–156 (2015)
18. Hermann, A., Ferré, S., Ducassé, M.: An interactive guidance process supporting consistent updates of RDFS graphs. In: Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 185–199. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33876-2_18
19. Fazzinga, B., Flesca, S., Furfaro, F.: On the expressiveness of generalization rules for XPath query relaxation. In: ACM International Conference on Proceedings Series Fourteenth Int'l Database Engineering and Applications Symposium (IDEAS), pp. 157–168. ACM(2010)
20. Liu, J., Yan, D.: Answering approximate queries over XML data. IEEE Trans. Fuzzy Syst. **24**(2), 288–305 (2016)
21. Biskup, J., Wiese, L.: A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. Theoret. Comput. Sci. **412**(31), 4044–4072 (2011)
22. Gaasterland, T., Godfrey, P., Minker, J.: Relaxation as a platform for cooperative answering. JIIS **1**(3/4), 293–321 (1992)
23. Ferilli, S., Basile, T.M.A., Biba, M., Mauro, N.D., Esposito, F.: A general similarity framework for horn clause logic. Fundam. Informaticae **90**(1–2), 43–66 (2009)
24. Tversky, A.: Features of similarity. Psychol. Rev. **84**(4), 327–352 (1977)