Enumerating Consistent Metaquery Instantiations

Fabrizio Angiulli
ICAR-CNR
Via P. Bucci 41/C
87036 Rende (CS), Italy
E-mail: angiulli@icar.cnr.it

Metaquerying is a data mining technique by which hidden dependencies among several database relations can be discovered in the form of Datalog-like rules, and this technique has already been successfully applied to several real-world application domains. Unfortunately, recent papers have shown that performing metaquerying turns out to be in general quite demanding from the computational viewpoint. The aim of this paper is to illustrate techniques by which metaquerying can be answered as efficiently as possible. Therefore, we first provide some new results regarding the computation of the number of substitutions for a given metaquery. In particular, an important source of complexity of implementing metaquerying relies in the exponential number of variable substitutions potentially to be analyzed to compute results, many of which turn out to be actually redundant. Redundancy checks are therefore illustrated and exploited below in order to minimize the computational cost to be paid to implement metaquerying. Metaquerying result construction algorithms are then given.

Keywords: Data mining, metaquerying

1. Introduction

It is frequent nowadays for organizations to possess huge amounts of information stored in databases to be exploited to extract useful knowledge for business and management support. The data mining research area is supposed to provide tools for the discovery of valuable knowledge in such huge data resources [12]. Among data mining methods that have been in recent years defined and exploited, the metaquerying technique [21,16,7,2,3,8,6] seems to be a particularly promising one for mining relational and deductive databases.

Metaqueries serve as a generic description of the class of patterns to be discovered. Notably, differently from many other mining techniques, patterns discovered using metaqueries can link information from several tables in databases. Moreover, such patterns are first-order (metaquery results take the form of Datalog-like rules), while most machine-learning systems can only learn propositional patterns and work on a single relation. Metaqueries can be specified by human experts or alternatively, they can be automatically generated from the database schema.

Intuitively, a metaquery has the form

$$T \leftarrow L_1, ..., L_m$$

where T and L_i are literal schemes $Q_i(Y_1, ..., Y_{n_i})$. The predicate variable Q_i can be instantiated to a predicate symbol representing a relation in the database. The instantiation must be done in a way which is consistent with the variable names.

For example (taken from [21]), let P,Q, and R be variables for predicates, then the metaquery

$$R(X, Z) \leftarrow P(X, Y), Q(Y, Z)$$

specifies that the patterns to be discovered are relationships of the form

$$r(X, Z) \leftarrow p(X, Y), q(Y, Z)$$

where p, q, and r are specific database relations. One possible result of this metaquery is the pattern (say, with confidence 0.93):

$$speaks(X, Z) \leftarrow citizen(X, Y),\ language(Y, Z)$$

This means that out of all pairs (X, Z) that satisfy the body of the above rule, 93% satisfy the left-hand side.

Thus, an answer to a metaquery is a rule accompanied by indices that indicate its plausibility degree. In [7], for example, each rule in the answer is supplied with *support* and the *confidence*. Index thresholds are provided by the user. As with other mining techniques, indices are used to avoid presenting negligible information to the user and to cut off the search space by early detection of low index values.

Unfortunately metaquerying brings, along with a vast applicability and usefulness, also a very demanding computational behaviour. The computational complexity characterization of metaquerying problems has been provided in [2,3]. In that paper, the authors show that, from the computational viewpoint, metaquerying is, in most cases, a very demanding task (complexity figures lying in the general case, between NP and NP^{PP}) and, therefore, any metaquery computation may require significant resources.

And, in particular, the number of instantiantions to be analyzed to construct the result of evaluating a metaquery is generally very large. In the following we prove that the problem of computing the number of such instantiations is #P-hard in general. Notably, however, several of such instantiations are, so as to say, redundant for the purposes of the evaluation, as they are essentially the same instantiation. It is therefore interesting to be able to discard those redundant instantiation in the first place. We point out that eliminating redundant consistent instantiations has a two-fold usefulness. First of all, it speeds-up result construction as, after generating and storing consistent instantiations in main memory, the metaquery system must compute values of indexes scored on the database by using the expensive join operator. Thus, filtering out redundant consistent instantiations will minimize the number of such evaluations. Second, it avoids presenting to the user many times the same kind of information.

In order to obtain such evaluation speed-up, this paper reports a number of results that allow to efficiently discard redundant instantiations of the metaquery to be answered. Such results form the basis upon which a metaquerying evaluation algorithm can be designed, which is also illustrated in the following.

In more detail, such results use the idea of exploiting a form of isomorphism among variable substitutions which is conducive to the definition of redundant instantiations. Such notion of isomorphism is then proved to reduce to the well-known graph isomorphism problem. Therefore, state-of-the-art algorithms devised to solve this latter problem can be adopted for the metaquery evaluation purposes.

The plan of the paper is, therefore, as follows.

Section 2 provides the definitions of metaquery and of metaquery instantiation, and defines metaquery indexes.

In Section 3, the computational complexity of the problem of counting the number of instantiations of a given metaquery is taken into account. Indeed, any algorithm which can enumerate all metaquery instantiations should be able to count them as efficiently as well. In particular, it is shown that the problems of computing the number of instantiations of a metaquery scoring enough value for an index and that of computing the number of instantiations of a metaquery that are consistent w.r.t. the database schema at hand, are #P-complete or #P-hard depending on the considered index.

In Section 4, the concepts of redundancy of two instantiations, metaquery variant, and redundancy of two metaquery variants are introduced. Intuitively, two instantiations are redundant if the are syntactically equivalent, while a variant of a metaquery $\mathcal{Q}$ is the metaquery obtained from $\mathcal{Q}$ by permutation of ordinary variables of $\mathcal{Q}$ and possibly adding new ordinary variables. Furthermore, two metaquery variants are defined to be redundant if the sets of their instantiations are equivalent via redundancy. Interestingly, it is proved that the set of all non redundant instantiations of a given metaquery is equivalent (via redundancy) to the union of the sets of all non redundant instantiations of its non redundant variants. This result has a practical relevance, as it suggests that in order to compute the consistent instantiations of a metaquery, it is better to compute first all its non redundant metaquery variants and then to find the consistent instantiations of these variants.

Section 5 addresses the problem of deciding if two metaquery variants are redundant. In particular, it is shown that this problem reduces to the graph isomorphism problem. This result makes redundancy checking between metaquery variants effective by using well-known algorithms for solving graph-isomorphism. Furthermore, the concept of autoset is introduced and it is shown how to exploit it in order to speed-up the above checking.

In Section 6, the problem of enumerating the set of all non redundant variants of a given metaquery is considered. In particular, it is shown how to enumerate efficiently certain subsets, as, for example, all the renaming of a metaquery – where a renaming is a particular type of metaquery variant.

In Section 7, the algorithm for enumerating all the non redundant instantiations of a given metaquery is presented and correctness of the algorithm is formally proved.

Finally, Section 8 reports conclusions.

2. Metaquerying

2.1. Metaqueries

We provide first the definition of relation and database schema and then formally define metaqueries.

A *relation schema* $R = (p, A_1, D_1, \ldots, A_n, D_n)$, of *arity* n, consists of a *relation name* p and of a sequence of *attribute names* $A_1, \ldots, A_n$ with associated countable sets of constants, or *domains*, $D_1, \ldots, D_n$, also denoted by $\mathbf{dom}(A_1), \ldots, \mathbf{dom}(A_n)$, respectively. A *relation* r on a relation schema $R = (p, A_1, D_1, \ldots, A_n, D_n)$ is a finite subset of $\mathbf{dom}(A_1) \times \ldots \times \mathbf{dom}(A_n)$.

A *database schema* DBS is a set of relation schemas. A *database* $DB = \{r_1, \ldots, r_n\}$ on a database schema $DBS = \{R_1, \ldots, R_n\}$ is a set of relations r_i on R_i ($1 \leq i \leq n$). We say that a database schema (a database resp.) is *untyped* if the attributes occurring in the database schema (in its associated database schema resp.) have the same domain. Otherwise, we say that it is *typed*.

For example, Figure 1 shows the database DB_1 on the database schema

$$DBS_1 = \{UsCaSch, CaTeSch, UsPTSch\}$$

where

$$UsCaSch = (\,UsCa, \text{user}, \mathbf{Users}, carrier, \mathbf{Companies})$$

$$CaTeSch = (\,CaTe, \text{carrier}, \mathbf{Companies}, technology, \mathbf{Technologies})$$

$$UsPTSch = (\,UsPT, \text{user}, \mathbf{Users}, phone\ type, \mathbf{Technologies})$$

and, **Users**, **Companies**, and **Technologies** are the following domains:

$$\mathbf{Users} = \mathbf{String}$$

$$\mathbf{Companies} = \{\text{Tim, Vodafone, Tre}\}$$

$$\mathbf{Technologies} = \{ETACS, GSM900, GSM1800, GSM1900, UMTS\}$$

Let $R = (p, A_1, D_1, \ldots, A_n, D_n)$ be a relation schema. An *atom* L *on* R is an expression of the form $p(X_1, \ldots, X_n)$, such that X_i is either an ordinary variable or a constant, for $i = 1, \ldots, n$. The domain $\mathbf{dom}(X_i, L)$ of X_i in L, is $\mathbf{dom}(A_i)$, for $i = 1, \ldots, n$.

Assume a database DB and a database schema DBS have been fixed. As stated above, a metaquery $\mathcal{Q}$ is a second-order template stating the form of the pattern to be discovered [21]. The template has the form

$$T \leftarrow L_1, ..., L_m \qquad (1)$$

where T and L_i are literal schemes. Each literal scheme L_i has the form $Q(Y_1, ..., Y_n)$ where Q is either a *predicate* (second order) *variable* or a relation symbol, and each Y_j ($1 \leq j \leq n$) is either an *ordinary* (first order) *variable* or a constant symbol. If Q is a predicate variable, then $Q(Y_1, ..., Y_n)$ is called a *relation pattern* of arity n, otherwise it is called an *atom* of arity n. The right-hand-side $L_1, ..., L_m$ is called the *body* of the metaquery, while T is called the *head* of the metaquery. A metaquery is called *pure* if each relation pattern with the same predicate variable has same arity.

Intuitively, given a database DB, answering a metaquery $\mathcal{Q}$ on DB amounts to finding all substitutions σ of relation patterns appearing in $\mathcal{Q}$ by atoms having as predicate names relations in DB, such that the Horn rule $\sigma(\mathcal{Q})$ (obtained by applying σ to $\mathcal{Q}$) encodes a dependency between the atoms in its head and body, that holds in DB with a certain degree of plausibility. The plausibility here is defined in terms of *indexes* which we will define shortly.

Let $\mathcal{Q}$ be a metaquery and DB a database on the schema DBS. Let $var(\mathcal{Q})$, $pv(\mathcal{Q})$, $ls(\mathcal{Q})$, and $rep(\mathcal{Q})$

$UsCa$

user	carrier
Tom S.	Vodafone
Tom S.	Tim
Laura D.	Vodafone

$CaTe$

carrier	technology
Tim	ETACS
Tim	GSM 900
Tim	GSM 1800
Vodafone	GSM 900
Vodafone	GSM 1800
Tre	UMTS

$UsPT$

user	phone type
Tom S.	GSM 900
Tom S.	GSM 1800
Laura D.	GSM 900

Fig. 1. The relations $UsCa$, $CaTe$, and $UsPT$ of DB_1

denote the set of variables (both predicate and ordinary), the set of predicate variables, the set of literal schemes, and the set of relation patterns occurring in $\mathcal{Q}$, respectively. Moreover, let $rel(DB)$ denote the set of relation names of DBS and $ato(DB)$ the set of atoms on a relation schema of DBS.

2.2. *Instantiations*

Semantics is defined via *metaquery instantiations*: an instantiation type specifies how relation patterns can be instantiated, turning a metaquery to an ordinary Horn rule over the given database. Next, we recall the three different types of metaquery instantiations defined in [2].

Definition 2.1 (Instantiation) Let $\mathcal{Q}$ be a metaquery and DB a database. An *instantiation* (on $\mathcal{Q}$ and DB) is a mapping $\sigma : rep(\mathcal{Q}) \rightarrow ato(DB)$, whose restriction $\sigma' : pv(\mathcal{Q}) \rightarrow rel(DB)$ is functional.

The condition above says that predicate names of $\mathcal{Q}$ are *consistently* substituted with relation names from DB.

Definition 2.2 (Type-0 instantiation) Let $\mathcal{Q}$ be a pure metaquery. An instantiation σ is *type-0* if for any relation pattern L and atom A, $\sigma(L) = A$ implies that L and A have the same list of arguments.

That is, under type-0 semantics, each predicate variable is always matched to a relation with the same arity and ordinary variables are left untouched. As an example, consider the database DB_1 shown in Figure 1 and the metaquery

$$R(X, Z) \leftarrow P(X, Y), Q(Y, Z) \qquad (2)$$

A possible type-0 instantiation for $\mathcal{Q}$ is

$$\begin{aligned}\sigma = \{&\langle R(X, Z), \text{UsPT}(X, Z)\rangle,\\ &\langle P(X, Y), \text{UsCa}(X, Y)\rangle,\\ &\langle Q(Y, Z), \text{CaTe}(Y, Z)\rangle\}\end{aligned}$$

which yields the following Horn rule when applied to $\mathcal{Q}$:

$$\text{UsPT}(X, Z) \leftarrow \text{UsCa}(X, Y), \text{CaTe}(Y, Z)$$

Definition 2.3 (Type-1 instantiation) Let $\mathcal{Q}$ be a pure metaquery. An instantiation σ is *type-1* if for any relation pattern L and atom A, $\sigma(L) = A$ implies that the arguments of A are obtained from arguments of L by permutation.

user	phone type	model
Tom S.	GSM 900	Nokia 6150
Tom S.	GSM 1800	Nokia 6150
Laura D.	GSM 900	Bosch 607

Fig. 2. The new extension for the relation *UsPT*

With type-1 instantiations, variable ordering within relation patterns does not matter. As an example, under this semantics, the metaquery (2) detects relationships among DB_1 relations in a way that is independent of variable ordering in relation patterns, and, among others, both the following Horn rules can be obtained.

$$\text{UsPT}(X, Z) \leftarrow \text{UsCa}(X, Y), \text{CaTe}(Y, Z)$$

$$\text{UsPT}(X, Z) \leftarrow \text{UsCa}(Y, X), \text{CaTe}(Y, Z)$$

The third type of instantiation takes a step further by allowing a relation pattern of arity k to be matched with an atom of arity k', with $k' \geq k$, padding "remaining" arguments to free variables:

Definition 2.4 (Type-2 instantiation) Let $\mathcal{Q}$ be a metaquery. An instantiation σ is *type-2* if for any relation pattern L and atom A, $\sigma(L) = A$ implies the following:

- the arity k' of A is greater-than or equal-to the arity of L;
- k of the arguments of A coincide with the k arguments of L, possibly occurring in different positions;
- the remaining $k' - k$ arguments of A are variables not occurring elsewhere in the instantiated rule.

With type-2 instantiations we can express interesting patterns disregarding how many extra attributes a physical relation has. Should the relation *UsPT* be defined with an additional attribute, as in Figure 2.2, the metaquery (2) can be instantiated, using a type-2 instantiation, to

$$\text{UsPT}(X, Z, T) \leftarrow \text{UsCa}(Y, X), \text{CaTe}(Y, Z)$$

Note that a type-0 instantiation can be viewed as a type-1 instantiation where the chosen permutation of relations's attributes is the identity, whereas a type-1 instantiation can be seen also as a type-2 instantiation, where the arity of the atoms matches the arity of the relation patterns they are substituted for. Note, moreover, that type-2 instantiations may apply to any metaquery, while type-0 and type-1 instantiations require pure metaqueries.

Next we define the concept of consistent instantiation w.r.t. the data base schema.

Definition 2.5 (Consistent instantiation) Let DB be a database on a database schema DBS, let $\mathcal{Q}$ be a metaquery, let $T \in \{0, 1, 2\}$, and let σ be a type-T instantiation of $\mathcal{Q}$ on DB:

- Let X be an ordinary variable occurring into $\mathcal{Q}$. We say that $\sigma(\mathcal{Q})$ is *consistent* w.r.t. X and DBS if, for each pair X', X'' of distinct occurrences of X into $\sigma(\mathcal{Q})$, respectively in the atoms L' and L'', we have that $\mathbf{dom}(X', L') = \mathbf{dom}(X'', L'')$.
- Let c be a constant occurring into $\mathcal{Q}$. We say that $\sigma(\mathcal{Q})$ is *consistent* w.r.t. c and DBS if, for each occurrence of c into $\sigma(\mathcal{Q})$ in an atom L, we have that $c \in \mathbf{dom}(c, L)$.
- We say that σ is *consistent* w.r.t. DBS if, for each ordinary variable or constant V of $\mathcal{Q}$, $\sigma(\mathcal{Q})$ is consistent w.r.t. V and DBS.

For example, consider the metaquery $Q = R(X, Z) \leftarrow P(X, Y), Q(Y, Z)$, the two following type-1 instantiations of $\mathcal{Q}$ on DB_1:

$$\begin{aligned}\sigma_1 = \{&\langle R(X, Z), UsCa(Z, X)\rangle,\\ &\langle P(X, Y), CaTe(X, Y)\rangle,\\ &\langle Q(Y, Z), UsPT(Y, Z)\rangle\}\\ \sigma_2 = \{&\langle R(X, Z), UsCa(Z, X)\rangle,\\ &\langle P(X, Y), CaTe(X, Y)\rangle,\\ &\langle Q(Y, Z), UsPT(Z, Y)\rangle\}\end{aligned}$$

and the metaqueries $\sigma_1(\mathcal{Q})$:

$$UsCa(Z, X) \leftarrow CaTe(X, Y), UsPT(Y, Z)$$

and $\sigma_2(\mathcal{Q})$:

$$UsCa(Z, X) \leftarrow CaTe(X, Y), UsPT(Z, Y).$$

Then σ_1 is not consistent w.r.t. DBS_1 as $\mathbf{dom}(Z, UsCa(Z, X)) = \mathbf{Users}$ and $\mathbf{dom}(Z, UsPT(Y, Z)) = \mathbf{Technologies}$, while σ_2 is consistent w.r.t. DBS_1.

2.3. Metaquery indexes

In data mining applications one is generally interested in discovering patterns of data that are "well-supported" in the analyzed data sets. In other words, it is not typically required for a discovered pattern to be true for the entire given data set, but for a significant subset thereof. This idea is embedded in the definition of *metaquery indexes*. In the literature, several metaquery index definitions are found, such as *support*, *confidence*, *base* and *strength* [7,8,16]. In fact, support is similar to base and confidence is similar to strength [7]. Therefore, in the analysis that follows, we shall use support, confidence and another index we find useful. For any set F, let $|F|$ denote its size.

Definition 2.6 (Fraction) Let R and S be two sets of relations. Then the fraction of R in S is:

$$R \uparrow S = \frac{|\pi_{att(R)}(\mathbf{J}(R) \bowtie \mathbf{J}(S))|}{|\mathbf{J}(R)|}$$

where $att(S)$ denotes the set of attributes and $\mathbf{J}(S)$ denotes the natural join of the relations in S. In particular, in every case where $|\pi_{att(R)}(\mathbf{J}(R) \bowtie \mathbf{J}(S))| = 0$, $R \uparrow S$ is defined equal to 0.

Definition 2.7 (Confidence, Cover and Support) Let r be a Horn rule and let DB be a database. Let $h(r)$ and $b(r)$ denote the sets of relations of DB occurring in the head and in the body of r, respectively. Then (i) the *confidence* of r is *cnf*$(r) = b(r) \uparrow h(r)$, (ii) the *cover* of r is *cvr*$(r) = h(r) \uparrow b(r)$, (iii) the *support* of r is *sup*$(r) = \max_{a \in b(r)}(\{a\} \uparrow b(r))$. In the following, the set of indexes $\{$*cnf*, *cvr*, *sup*$\}$ will be denoted $\mathbf{I}$.

Given a rule r, the indices $cnf(r)$ and $sup(r)$ are equivalent to the notions of *confidence* and *support* defined in [7]. A detailed discussion of support and confidence for metaqueries can be found in [7,8]. The notion of *cover* has been introduced in [2], and it serves the purpose of measuring how many tuples are there that satisfy the head, satisfy also the body.

3. On the Complexity of Answering a Metaquery

In general, given a metaquery $\mathcal{Q}$ and a database DB on a database schema DBS, we are interested in obtaining all the instantiations σ of $\mathcal{Q}$ on DB such that the Horn rule $\sigma(\mathcal{Q})$ scores an adequate value of cover, confidence and support. That is, we search for the set of instantiations of a metaquery, defined next.

Definition 3.1 (Set of instantiations) Let $T \in \{0, 1, 2\}$, let $\mathcal{Q}$ be a metaquery, let DB be a database, let $I \in \mathbf{I}$ and let k be a rational number in $[0, 1)$. We define $\Sigma(\mathcal{Q}, DB, I, k, T)$ as the set of type-T instantiations σ on $\mathcal{Q}$ and DB such that $I(\sigma(\mathcal{Q})) > k$ on DB.

When the database schema DBS is typed, it make sense to search for the set of consistent instantiations of a metaquery, defined as follows.

Definition 3.2 (Set of consistent instantiations) Let $\mathcal{Q}$ be a metaquery, let DBS be a database schema, and let $T \in \{0, 1, 2\}$. We denote by $\Sigma_c(\mathcal{Q}, DBS, T)$ the set consisting of by all the type-T instantiations σ of $\mathcal{Q}$ such that $\sigma(\mathcal{Q})$ is consistent w.r.t. DBS.

Given the definitions above, the following interesting complexity problems are then obtained:

Problem 3.3 Let $T \in \{0, 1, 2\}$ be an instantiation type and let $I \in \mathbf{I}$ be an index. Let DBS denote a database schema, let DB denote a database on DBS, $\mathcal{Q}$ denote a metaquery, and $k \in [0, 1)$ rational threshold value. Then:

- The complexity of the metaquery problem

$$\langle DB, \mathcal{Q}, I, k, T\rangle,$$

  is the complexity, measured in the size of DB, $\mathcal{Q}$ and k, of deciding if $\Sigma(\mathcal{Q}, DB, I, k, T)$ is nonempty (also defined in [2,3]).
- The complexity of the metaquery problem

$$\langle DB, \mathcal{Q}, T\rangle,$$

  is the complexity, measured in the size of DB and DBS, of deciding if $\Sigma_c(\mathcal{Q}, DBS, T)$ is nonempty (also defined in [8]).
- The complexity of the metaquery problem

$$\#\langle DB, \mathcal{Q}, I, k, T\rangle,$$

  is the complexity, measured in the size of DB, $\mathcal{Q}$ and k, of computing the cardinality of the set $\Sigma(\mathcal{Q}, DB, I, k, T)$.
- The complexity of the metaquery problem

$$\#\langle DBS, \mathcal{Q}, T\rangle,$$

  is the complexity, measured in the size of DB and DBS, of computing the cardinality of the set $\Sigma_c(\mathcal{Q}, DBS, T)$.

It is proved in [2,3] that the complexity of $\langle DB, \mathcal{Q}, I, k, T\rangle$ is NP-complete for $I \in \{cvr, sup\}$ and is NP^{PP}-complete for $I = cnf$. Thus, the problem $\#\langle DB, \mathcal{Q}, I, k, T\rangle$ is NP-hard. Furthermore, it is stated in [8] that the problem $\langle DBS, \mathcal{Q}, T\rangle$ is NP-complete if DBS is typed, while it is solvable in quadratic time and logarithmic space if DBS is untyped. Thus, when DBS is typed, also the problem $\#\langle DBS, \mathcal{Q}, T\rangle$ is NP-hard. Next, we provide more strict bounds to the computational complexity of the counting problems $\#\langle DB, \mathcal{Q}, I, k, T\rangle$ and $\#\langle DBS, \mathcal{Q}, T\rangle$.

First, we briefly recall some definitions of computational complexity theory related to counting and the definition of conjunctive query. For more on complexity theory the reader is referred to [13].

A *counting Turing machine* (CTM) is a nondeterministic Turing machine whose output for a given input string is the number of accepting computations for that input. The class #P is the set of all functions that are computable by polynomial-time CTMs [20]. A problem is defined #P-hard and #P-complete in the usual manner. The following form of reduction is used to prove #P-hardness: a *parsimonious transformation* is a polynomial transformation f from problem X to problem Y such that, if $\#(X, x)$ is defined to be the number of solutions that instance x has in the problem X, then $\#(X, x) = \#(Y, f(x))$.

A *conjunctive query* q is a set of atoms $q = \{p_1(\mathbf{X_1}), \ldots, p_n(\mathbf{X_n})\}$, where $\mathbf{X_1}, \ldots, \mathbf{X_n}$ is a list of variables and/or constants. Let DB be a database. The problem of satisfying a conjunctive query (the Boolean Conjunctive Query satisfaction problem, or BCQ) is the problem of deciding whether there exists a substitution ρ for the variables in $\mathbf{X_1}, \ldots, \mathbf{X_n}$ *satisfying* q, i.e. such that, for each $i = 1, \ldots, n$, $\rho(\mathbf{X_i})$ is a tuple occurring in the relation p_i of DB. The set $\rho(q) = \{p_i(\rho(\mathbf{X_i})) \mid 1 \leq i \leq n\}$ is called *ground instance* of q. The problem of counting the number of substitutions satisfying a conjunctive query is known to be #P-complete.

Theorem 3.4 *The complexity of*

1. $\#\langle DB, \mathcal{Q}, I, k, T\rangle$*, and*
2. $\#\langle DBS, \mathcal{Q}, T\rangle$

is #P-hard.

Proof. We show a transformation from BCQ to $\langle DB, \mathcal{Q}, I, k, T\rangle$ and a transformation from BCQ to $\langle BDS, \mathcal{Q}, T\rangle$ preserving the number of solutions. Let $q = \{p_1(\mathbf{X_1}), \ldots, p_n(\mathbf{X_n})\}$ be a conjunctive query on the database DB. W.l.o.g. we can assume that $p_1, \ldots, p_n$ are distinct relation names occurring in DB, that $\mathbf{X} = \mathbf{X_1} \cup \ldots \cup \mathbf{X_n}$ contains only variables, and that each $\mathbf{X_i}$ $(1 \leq i \leq n)$ does not contain duplicate variables. Let $R_i = (p_i, A_{i,1}, \ldots, A_{i,r_i})$ be the relation schema of the relation p_i, let d_i be the number of tuples in the relation p_i, and let $\mathbf{X_i} = X_{i,1}, \ldots, X_{i,r_i}$, for $i = 1, \ldots, n$.

Let $Y_1, \ldots, Y_m$ the distinct variables occurring in $\mathbf{X}$. For each $Y \in \mathbf{X}$, let $\mathcal{C}_Y$ denote the set

$$\bigcup_{X_{i,j}=Y} \pi_{A_{i,j}}(p_i),$$

i.e. the set of the values that the variable Y can assume on the database DB (we assume that duplicate values are eliminated from $\mathcal{C}_Y$).

(1) Now we build a metaquery $\mathcal{Q}^{BCQ}$ and a database DB^{BCQ} associated to q and DB. Let $e_0 = n$, $e_1 = e_0 + |\mathcal{C}_{Y_1}|, \ldots, e_m = e_{m-1} + |\mathcal{C}_{Y_m}|$ be integer numbers. Let $f_i : \mathcal{C}_{Y_i} \mapsto [e_{i-1}+1, e_i]$ be a bijection between the set of values $\mathcal{C}_{Y_i}$ to the set of integers in the interval $[e_{i-1}+1, e_i]$, for $i = 1, \ldots, m$.

The database DB^{BCQ} consists of the relations $t_{i,k} = \{\langle i, f_{X_{i,1}}(u_{i,k,1}), \ldots, f_{X_{i,r_i}}(u_{i,k,r_i})\rangle\}$, for $i = 1, \ldots, n$, $k = 1, \ldots, d_i$, where $\langle u_{i,k,1}, \ldots, u_{i,k,r_i}\rangle$ is the kth tuple of the relation p_i of DB, and of the relations $g_i = \{\langle e_{i-1}+1\rangle, \ldots, \langle e_i\rangle\}$, for $i = 1, \ldots, m$.

The metaquery $\mathcal{Q}^{BCQ}$ is

$$Q_1(1, \mathbf{X_1}) \leftarrow Q_1(1, \mathbf{X_1}), Q_2(2, \mathbf{X_2}), \ldots,$$
$$Q_n(n, \mathbf{X_n}), g_1(Y_1), \ldots, g_m(Y_m)$$

Let ρ be a substitution satisfying q, and let $\rho(q) = \{p_1(\mathbf{x_1}), \ldots, p_n(\mathbf{x_n})\}$ be a ground instance of q. Consider the type-0 (hence also type-1 and type-2) instantiation of $\mathcal{Q}^{BCQ}$ on DB^{BCQ}

$$\sigma^\rho = \{\langle Q_1(1, \mathbf{X_1}), t_{1,j_1}(1, \mathbf{X_1})\rangle, \ldots,$$
$$\langle Q_n(\mathbf{X_n}), t_{n,j_n}(n, \mathbf{X_n})\rangle\}$$

where $j_1, \ldots, j_n$ are such that $\mathbf{x_i} = \langle u_{i,j_i,1}, \ldots, u_{i,j_i,r_i}\rangle$, for $i = 1, \ldots, n$ (that is $\mathbf{x_i}$ is the j_ith tuple of the relation p_i of DB). As ρ is a substitution satisfying q, and the unique tuple of the relation t_{i,j_i} is obtained from $\mathbf{x_i}$ through the bijections $f_1, \ldots, f_m$, then the relation

$$t_{1,j_1}(1, \mathbf{X_1}) \bowtie \ldots \bowtie t_{n,j_n}(n, \mathbf{X_n}) \bowtie$$
$$\bowtie g_1(Y_1) \bowtie \ldots \bowtie g_m(Y_m)$$

is nonempty, and $cnf(\sigma^\rho(\mathcal{Q}^{BCQ}))$, $cvr(\sigma^\rho(\mathcal{Q}^{BCQ}))$, and $sup(\sigma^\rho(\mathcal{Q}^{BCQ}))$ are equal to 1 on DB^{BCQ}.

Vice versa, consider an instantiation σ of $\mathcal{Q}^{BCQ}$ on DB^{BCQ}. Because of the presence of the constant i in $Q_i(i, \mathbf{X_i})$ $(1 \leq i \leq n)$, σ maps $Q_i(i, \mathbf{X_i})$ to a relation t_{i,j_i} $(1 \leq j_i \leq d_i)$ of DB^{BCQ}. Furthermore, as $g(Y_i)$ belongs to $\mathcal{Q}^{BCQ}$, to be $cnf(\sigma^\rho(\mathcal{Q}^{BCQ}))$, $cvr(\sigma^\rho(\mathcal{Q}^{BCQ}))$, and $sup(\sigma^\rho(\mathcal{Q}^{BCQ}))$ non zero, it is the case that σ maps each $\mathbf{X_i}$ into itself, i.e. that none of the $\mathbf{X_i}$ arguments is permuted. Thus, $\mathbf{x_i} = \langle f^{-1}_{X_{i,1}}(u_{i,j_i,1}), \ldots, f^{-1}_{X_{i,r_i}}(u_{i,j_i,r_i})\rangle$ is a tuple of the relation p_i of DB $(1 \leq i \leq n)$ and the substitution ρ^σ for the variables $Y_1, \ldots, Y_m$ such that $\rho^\sigma(q) = \{p_1(\mathbf{x_1}, \ldots, \mathbf{x_n})\}$ satisfies q.

(2) Now we build a metaquery $\mathcal{Q}^{BCQ}$ and a database scheme DBS^{BCQ} associated to q and DB. Let $\mathcal{S}_i$ denote a new distinct type having domain $\{i\}$, for $i = 1, \ldots, n+m$, and let $\mathcal{T}_{i,c}$ denote a new distinct type such that $\{1, \ldots, n+m\}$ are not in the domain of $\mathcal{T}_{i,c}$, for $i = 1, \ldots, m$, $c \in \mathcal{C}_{Y_i}$.

The database scheme DBS^{BCQ} consists of the relation schemes

- $T_{i,j} = (t_i, B_{i,j,0}, B_{i,j,1}, \ldots, B_{i,j,r_i})$:
  with $\mathbf{typ}(B_{i,j,0}) = \mathcal{S}_i$ and $\mathbf{typ}(B_{i,j,k}) = \mathcal{T}_{i,u_{i,j,k}}$, where $\langle u_{i,j,1}, \ldots, u_{i,j,r_i}\rangle$ denotes the jth tuple of the relation p_i of DB, for $i = 1, \ldots, n$, $j = 1, \ldots, d_i$, $k = 1, \ldots, r_i$.
- $G_{i,c} = (g_{i,c}, C_i, C_{i,c})$:
  with $\mathbf{typ}(C_i) = \mathcal{S}_{n+i}$ and $\mathbf{typ}(C_{i,c}) = \mathcal{T}_{i,c}$, for $i = 1, \ldots, m$, $c \in C_{Y_i}$

The metaquery $\mathcal{Q}^{BCQ}$ is

$$Q_1(1, \mathbf{X_1}) \leftarrow Q_1(1, \mathbf{X_1}), \ldots, Q_n(n, \mathbf{X_n}),$$
$$G_1(n+1, Y_1), \ldots, G_m(n+m, Y_m)$$

Let ρ be a substitution satisfying q, and let $\rho(q) = \{p_1(\mathbf{x_1}), \ldots, p_n(\mathbf{x_n})\}$ be a ground instance of q. Consider the type-0 (hence also type-1 and type-2) consistent instantiation of $\mathcal{Q}^{BCQ}$ on DBS^{BCQ}

$$\sigma^\rho = \{\langle Q_1(1, \mathbf{X_1}), t_{1,j_1}(1, \mathbf{X_1})\rangle, \ldots,$$
$$\langle Q_n(\mathbf{X_n}), t_{n,j_n}(n, \mathbf{X_n})\rangle,$$
$$\langle G_1(n+1, Y_1), g_{1,c}(n+1, C_{1,c})\rangle, \ldots,$$
$$\langle G_m(n+m, Y_m), g_m(n+m, Y_m)\rangle\}$$

where $j_1, \ldots, j_n$ are such that $\mathbf{x_i} = \langle u_{i,j_i,1}, \ldots, u_{i,j_i,r_i}\rangle$, for $i = 1, \ldots, n$ (that is $\mathbf{x_i}$ is the j_ith tuple of the relation p_i of DB). As ρ is a substitution satisfying q, then σ^ρ is a consistent instantiation of $\mathcal{Q}^{BCQ}$ on DBS^{BCQ}.

Vice versa, consider a consistent instantiation σ of $\mathcal{Q}^{BCQ}$ on DBS^{BCQ}. Because of the presence of the constant i in $Q_i(i, \mathbf{X_i})$ and $n+j$ in $G_j(n+j, Y_j)$ $(1 \leq i \leq n, 1 \leq j \leq m)$, σ maps $Q_i(i, \mathbf{X_i})$ to a relation t_{i,k_i} $(1 \leq k_i \leq d_i)$ and $G_j(n+j, Y_j)$ to a relation $g_{j,c_j}(n+j, Y_j)$ $(c_j \in \mathcal{C}_{Y_j})$ of DB^{BCQ}. Consequently, it is the case that σ maps each $\mathbf{X_i}$ into itself, i.e. that none of the $\mathbf{X_i}$ arguments is permuted. Thus, $\mathbf{x_i} = \langle c_{j_1}, \ldots, c_{j_n}\rangle$

is a tuple of the relation p_i of DB ($1 \leq i \leq n$) and the substitution ρ^σ for the variables $Y_1, \ldots, Y_m$ such that $\rho^\sigma(q) = \{p_1(\mathbf{x_1}, \ldots, \mathbf{x_n})\}$ satisfies q. □

Theorem 3.5 *The complexity of*

1. $\#\langle DB, \mathcal{Q}, I, k, T\rangle$*, with* $I \in \{cvr, sup\}$ *and*
2. $\#\langle DBS, \mathcal{Q}, T\rangle$*, with* $I \in \mathbf{I}$

is #P-complete.

Proof. Theorem 3.4 proves that problems (1) and (2) are #P-hard. In order to prove #P-completeness of these problems, it remains to show their #P-membership. The latter property follows immediately from the fact that the existential versions of the problems for (1) and (2), i.e. the problems $\langle DB, \mathcal{Q}, I, k, T\rangle$ and $\langle DBS, \mathcal{Q}, T\rangle$ respectively, are in NP. □

4. Redundance of instantiations

In [7,8] the process of answering a metaquery is divided into two stages: the *instantiation stage* and the *filtration stage*.

During the instantiation stage the set of consistent instantiations of the metaquery are determined. The instantiations stage is similar to solving a Constraint Satisfaction Problem (CSP) [11] where we are looking for all the solutions of the CSP. In the filtration stage, the instantiated metaqueries, resulting from the instantiation stage, having low values of support, cover or confidence are filtered out.

It follows from the results referred to in the previous section, that the former problem is as hard as the latter, at least when support and cover are considered. But, as the instantiation stage does not involve the use of the (expensive) join operator, the preprocessing stage can apport significant time savings to the overall process of answering a metaquery.

Furthermore, we note that the output of the filtration stage is sometimes, so as to say, redundant. That is, the set $\Sigma_c(\mathcal{Q}, DBS, T)$ could contain instantiations that are essentially a syntactical variant of other instantiations contained in the same set. Thus, in order to speed-up result construction is therefore interesting to be able to discard those redundant instantiations in the first place.

Now we give an overview of the section contents.

Section 4.2 defines *redundance between instantiations*. Intuitively, two instantiations σ and σ' are to be considered redundant if they are syntactically equivalent, i.e. if there exists a *variable mapping* h such that $h(\sigma)$ equals σ'. Hence, two redundant instantiations are essentially the same instantiation. Furthermore, two sets of instantiations Σ and Σ' are redundant iff for each $\sigma \in \Sigma$ there exists $\sigma' \in \Sigma'$ such that σ and σ' are redundant, and vice versa.

Section 4.3 introduces the concept of metaquery *variant* and the concept of *redundance between variants*. A variant $\mathcal{Q}'$ of a metaquery $\mathcal{Q}$, is a metaquery obtained from $\mathcal{Q}$ by permutation of ordinary variables of $\mathcal{Q}$ and possibly adding new ordinary variables, while $\mathcal{Q}$ and $\mathcal{Q}'$ are defined to be redundant if, for each database, index, and threshold, the sets Σ and Σ' of the instantiations of $\mathcal{Q}$ and $\mathcal{Q}'$ respectively, are redundant.

After defining variants and the associated redundance concept, Theorem 4.16 states that $\mathcal{Q}$ and $\mathcal{Q}'$ are redundant iff $\mathcal{Q}$ and $\mathcal{Q}'$ are syntactically equivalent, i.e. iff there exists a variable mapping h and a *reordering* π of literal schemes such that $\mathcal{Q}$ equals $h(\pi(\mathcal{Q}'))$.

Finally, Theorem 4.20 proves that the set of all non redundant instantiations of a given metaquery is equivalent (via redundancy) to the union of the sets of all non redundant instantiations of its non redundant variants. Theorem 4.20 explains the reasons for defining metaquery variants. Indeed, it has a practical relevance, as it suggests that in order to compute the consistent instantiations of a metaquery, it is better to compute first all its non redundant metaquery variants and then to find the consistent instantiations of these variants.

We introduce next the notion of redundancy between instantiations with some examples and then formalize it.

4.1. Preliminary definitions

First we define the notion of *equality* of two, respectively, literal schemes, substitutions, sequences of literal schemes and metaqueries.

Definition 4.1 (Equality of two relation patterns) Let $P(\mathbf{X})$ and $Q(\mathbf{Y})$ be two relation patterns. We say that $P(\mathbf{X})$ and $Q(\mathbf{Y})$ are *equal* under the type-0 semantics, written $P(\mathbf{X}) =_0 Q(\mathbf{Y})$, iff they are syntactically identical. We say that $P(\mathbf{X})$ and $Q(\mathbf{Y})$ are *equal* under the type-T semantics, with $T \in \{1, 2\}$, written $P(\mathbf{X}) =_T Q(\mathbf{Y})$, iff $P(\mathbf{X}) =_0 Q(\Pi(\mathbf{Y}))$, where $\Pi : \mathbf{Y} \rightarrow \mathbf{Y}$ is a permutation of the arguments of $Q(\mathbf{Y})$.

For example, $R(A, B) =_T R(A, B)$ for $T \in \{0, 1, 2\}$, but $R(A, B) \neq_0 R(B, A)$, while $R(A, B) =_T R(B, A)$ holds for $T \in \{1, 2\}$.

Definition 4.2 (Equality of two atoms) Let $p(\mathbf{X})$ and $q(\mathbf{Y})$ be two atoms. We say that $p(\mathbf{X})$ and $q(\mathbf{Y})$ are *equal* under the type-T semantics, written $p(\mathbf{X}) =_T q(\mathbf{Y})$, iff they are syntactically identical.

Definition 4.3 (Equality of two substitutions) Let σ and σ' be two type-T substitutions. We say that σ and σ' are *equal* under type-T semantics, written $\sigma =_T \sigma'$, iff for each pair $\langle R, A\rangle \in \sigma$ there exists a pair $\langle R', A'\rangle \in \sigma'$ such that $R =_T R'$ and $A =_0 A'$, and vice versa.

Definition 4.4 (Equality of two sequences) Let $S = L_1, \ldots, L_m$ and $S' = L'_1, \ldots, L'_m$ bet two sequences of literal schemes, and let $T \in \{0, 1, 2\}$. We say that S and S' are *equal* under type-T semantics, written $S =_T S'$, iff $L_i =_T L'_i$, for each $i = 1, \ldots, m$.

For example, $R(A, B), S(B, C) =_T R(A, B), S(B, C)$ for $T \in \{0, 1, 2\}$ as the two sequences are syntactically identical, but $R(B, A), S(B, C) \neq_0 R(A, B)$, S(B,C), while $R(B, A), S(B, C) =_T R(A, B), S(B, C)$ hold for $T \in \{1, 2\}$. On the contrary, $R(A, B), (B, C) \neq_T S(B, C), R(A, B)$ for each $T \in \{0, 1, 2\}$, as the literal schemes $R(A, B)$ and $S(B, C)$ are situated into two different positions in the two sequences.

Definition 4.5 (Equality of two metaqueries) Let $\mathcal{Q} = L_1 \leftarrow L_2, \ldots, L_m$ and $\mathcal{Q}' = L'_1 \leftarrow L'_2, \ldots, L'_m$ be two metaqueries, and let $T \in \{0, 1, 2\}$. We say that $\mathcal{Q}$ and $\mathcal{Q}'$ are *equal* under type-T semantics, written $\mathcal{Q} =_T \mathcal{Q}'$, iff $L_1, \ldots, L_m =_T L'_1, \ldots, L'_m$, for each $i = 1, \ldots, m$.

4.2. Redundance of instantiations

We start with an example. Consider the metaquery $\mathcal{Q} = P(X, Y) \leftarrow Q(Y, Z), R(Z, X)$ and the database DB_1. Then the type-1 instantiations

$$\begin{aligned}\sigma_1 = \{&\langle P(X, Y), UsPT(Y, X)\rangle,\\ &\langle Q(Y, Z), UsCa(Y, Z)\rangle,\\ &\langle R(Z, X), CaTe(Z, X)\rangle\}\\ \sigma_2 = \{&\langle P(X, Y), UsPT(X, Y)\rangle,\\ &\langle Q(Y, Z), CaTe(Z, Y)\rangle,\\ &\langle R(Z, X), UsCa(X, Z)\rangle\}\end{aligned}$$

belong to $\Sigma_c(\mathcal{Q}, DBS_1, 1)$, and to $\Sigma(\mathcal{Q}, DB, I, 0, 1)$ for each $I \in \mathbf{I}$. Consider now the metaqueries $\sigma_1(\mathcal{Q})$:

$$UsPT(Y, X) \leftarrow UsCa(Y, Z), CaTe(Z, X)$$

and $\sigma_2(\mathcal{Q})$:

$$UsPT(X, Y) \leftarrow CaTe(Z, Y), UsCa(X, Z).$$

Despite being syntactically different, $\sigma_1(\mathcal{Q})$ and $\sigma_2(\mathcal{Q})$ represent the same metaquery. Thus, the simultaneous presence of both σ_1 and σ_2 in the sets $\Sigma_c(\mathcal{Q}, DBS_1, 1)$ and $\Sigma(\mathcal{Q}, DB_1, I, 0, 1)$ makes these sets, in some sense, redundant. If we look carefully at the two above instantiations, we can say that they are to be considered redundant as they are essentially the same instantiation. The following definition formalizes the above intuition.

Definition 4.6 (Variable mapping) Let $\mathcal{Q}$ and $\mathcal{Q}'$ be two metaqueries. A *variable mapping* h of $\mathcal{Q}$ into $\mathcal{Q}'$ is a bijection $h : var(\mathcal{Q}) \rightarrow var(\mathcal{Q}')$. We denote by $h(\mathcal{Q})$ the metaquery obtained substituting each variable X of $\mathcal{Q}$ with $h(X)$.

Definition 4.7 (Redundant instantiations) Let $\mathcal{Q}$ and $\mathcal{Q}'$ be two metaqueries, and let σ and σ' be two type-T instantiations on $\mathcal{Q}$ and $\mathcal{Q}'$ respectively. We say that σ and σ' are *redundant* under type-T semantics, written $\sigma \equiv_T \sigma'$, if there exists a variable mapping h of $\mathcal{Q}$ into $\mathcal{Q}'$, which maps the head of $\mathcal{Q}$ into the head of $\mathcal{Q}'$, such that $h(\sigma) =_T \sigma'$, where $h(\sigma) = \{\langle h(R), h(A)\rangle \mid \langle R, A\rangle \in \sigma\}$.

Resuming our example, let $h(Y) = X$, $h(X) = Y$, $h(Z) = Z$, $h(Q) = R$, and $h(R) = Q$, then $\sigma_1 \equiv_1 \sigma_2$ as

$$\begin{aligned}&\langle h(P(X, Y)), h(UsPT(Y, X))\rangle =\\ &\quad\langle P(Y, X), UsPT(X, Y)\rangle =_1\\ &\quad\quad\langle P(X, Y), UsPT(X, Y)\rangle,\\ &\langle h(Q(Y, Z)), h(UsCa(Y, Z))\rangle =\\ &\quad\langle R(X, Z), UsCa(X, Z)\rangle =_1\\ &\quad\quad\langle R(Z, X), UsCa(X, Z)\rangle,\\ &\langle h(R(Z, X)), h(CaTe(Z, X))\rangle =\\ &\quad\langle Q(Z, Y), CaTe(Z, Y)\rangle =_1\\ &\quad\quad\langle Q(Y, Z), CaTe(Z, Y)\rangle.\end{aligned}$$

Definition 4.8 (Redundant sets of instantiations) Let $\mathcal{Q}$ and $\mathcal{Q}'$ be two metaqueries and let Σ and Σ' be two sets of instantiations of type-T on $\mathcal{Q}$ and $\mathcal{Q}'$ respectively. We say that Σ and Σ' are *redundant*, written $\Sigma \equiv_T \Sigma'$, if for each $\sigma \in \Sigma$ there exists $\sigma' \in \Sigma'$ such that $\sigma \equiv_T \sigma'$, and vice versa. We write $\Sigma \doteq_T \Sigma'$, if for each $\sigma \in \Sigma$ there exists one and only one $\sigma' \in \Sigma'$ such that $\sigma \equiv_T \sigma'$, and for each $\sigma' \in \Sigma'$ there exists one and only one $\sigma \in \Sigma$ such that $\sigma' \equiv_T \sigma$.

Definition 4.9 (Minimal set of instantiations) Let $T \in \{0,1,2\}$, let $\mathcal{Q}$ be a metaquery, let DB be a database, let $I \in \mathbf{I}$ and let k be a rational number in $[0,1)$. We define $\widehat{\Sigma}(\mathcal{Q}, DB, I, k, T)$ as the smallest subset of $\Sigma(\mathcal{Q}, DB, I, k, T)$ such that $\Sigma(\mathcal{Q}, DB, I, k, T) \equiv_T \widehat{\Sigma}(\mathcal{Q}, DB, I, k, T)$.

4.3. *Variants and redundance of variants*

Now we define the concept of metaquery variant, the notion of redundance between variants, and then we explain how variants are related to the property of redundance between instantiations.

Definition 4.10 (Metaquery variants) Let $\mathcal{Q}$ and $\mathcal{Q}'$ be two metaqueries, we say that:

- $\mathcal{Q}'$ is a $1,0$–*variant* of $\mathcal{Q}$ if $\mathcal{Q} =_1 \mathcal{Q}'$;
- $\mathcal{Q}'$ is a $2,1$–*variant* of $\mathcal{Q}$ if there exists a metaquery $\mathcal{Q}''$ such that $\mathcal{Q}'' =_0 \mathcal{Q}'$, where $\mathcal{Q}''$ is obtained from $\mathcal{Q}$ augmenting the sequence of ordinary variables of every relational pattern of $\mathcal{Q}$ with a possibly empty sequence of padding variables;
- $\mathcal{Q}'$ is a $2,0$–*variant* of $\mathcal{Q}$ if there exists a metaquery $\mathcal{Q}''$ such that $\mathcal{Q}''$ is a $2,1$-variant of $\mathcal{Q}$ and $\mathcal{Q}'$ is a $1,0$-variant of $\mathcal{Q}''$.

For example, consider the metaqueries:

$$\mathcal{Q}_1 = P(X,Y) \leftarrow Q(X,Y,Z), R(Z)$$
$$\mathcal{Q}_2 = P(Y,X) \leftarrow Q(X,Z,Y), R(Z)$$
$$\mathcal{Q}_3 = P(X,Y,A) \leftarrow Q(X,Y,Z), R(Z,B)$$
$$\mathcal{Q}_4 = P(A,X,Y) \leftarrow Q(Z,X,Y), R(Z,B)$$

Then:

- $\mathcal{Q}_2$ is a $1,0$-variant of $\mathcal{Q}_1$ as both $P(X,Y)$ and $Q(X,Y,Z)$ differ from $P(Y,X)$ and $Q(X,Z,Y)$ only for a permutation of the ordinary variables,
- $\mathcal{Q}_3$ is a $2,1$-variant of $\mathcal{Q}_1$ as it is obtained augmenting $P(X,Y)$ with A and $R(Z)$ with B,
- $\mathcal{Q}_4$ is a $1,0$-variant of $\mathcal{Q}_3$ as is obtained permuting the arguments of the literal schemes of $\mathcal{Q}_3$, and
- $\mathcal{Q}_4$ is a $2,0$-variant of $\mathcal{Q}_1$ as it is a $1,0$-variant of $\mathcal{Q}_3$ which is a $2,1$-variant of $\mathcal{Q}_1$.

Definition 4.11 (Redundant variants) Let $\mathcal{Q}$ be a metaquery, let $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$, and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two T_1,T_2-variants of $\mathcal{Q}$. We say that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are *redundant* under type-T_2 instantiations, written $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$, iff for each database DB, index $I \in \mathbf{I}$ and threshold value k, $\Sigma(\mathcal{Q}_1, DB, I, k, T_2) \equiv_{T_2} \Sigma(\mathcal{Q}_2, DB, I, k, T_2)$.

Definition 4.12 (Isolated variable) Let $\mathcal{Q}$ be a metaquery, and let X be an ordinary variable of $\mathcal{Q}$. We say that X is *isolated* if it occurs into only one literal scheme of $\mathcal{Q}$.

For example, consider the metaquery $\mathcal{Q} = P(X,Y) \leftarrow Q(X,Z), R(W)$. Then, the ordinary variables Z and W are isolated.

Definition 4.13 (General metaquery) Let $\mathcal{Q}$ be a metaquery. We say that $\mathcal{Q}$ is *general* under the type-0 and type-1 semantics if $\mathcal{Q}$ does not contain atoms. We say that $\mathcal{Q}$ is *general* under the type-2 semantics if $\mathcal{Q}$ does not contain neither atoms nor isolated ordinary variables.

From now we shall deal only with general metaqueries. Note that, under the type-2 semantics, isolated ordinary variables occurring into literal schemes do not affect the pattern specified by the metaquery. We will show in Section 7 how metaqueries containing atoms can be managed.

Problem 4.14 (Redundance of variants) Given a general metaquery $\mathcal{Q}$ and two T_1,T_2-variants $\mathcal{Q}_1$ and $\mathcal{Q}_2$ of $\mathcal{Q}$, $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$, is it true that $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$?

Definition 4.15 (Reordering) Let $\mathcal{Q} = H \leftarrow B_1, \ldots, B_m$ be a metaquery. A *reordering* π of $\mathcal{Q}$ is a bijection $\pi : ls(\mathcal{Q}) \rightarrow ls(\mathcal{Q})$ which maps the head of $\mathcal{Q}$ into itself. We denote by $\pi(\mathcal{Q})$ the metaquery $H \leftarrow \pi(B_1), \ldots, \pi(B_m)$[1].

The following theorem states an important result regarding general metaqueries, i.e. that two variants of a general metaquery are redundant if and only if they are syntactically equivalent.

Theorem 4.16 *Let $\mathcal{Q}$ be a general metaquery, let $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$, and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two T_1,T_2-variants of $\mathcal{Q}$. Then $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$ iff there exist a variable mapping of $\mathcal{Q}_2$ into $\mathcal{Q}_1$ and a reordering π of $\mathcal{Q}_2$ such that $\mathcal{Q}_1 =_{T_2} h(\pi(\mathcal{Q}_2))$.*

[1]Recall that $ls(\mathcal{Q})$ denotes the set of literal schemes occurring in $\mathcal{Q}$.

Proof. ($\Rightarrow$) Let $\mathcal{Q}_1$ be

$$P_1(\mathbf{X_1}) \leftarrow P_2(\mathbf{X_2}), \ldots, P_m(\mathbf{X_m}),$$

and let $\mathcal{Q}_2$ be

$$P_1(\mathbf{Y_1}) \leftarrow P_2(\mathbf{Y_2}), \ldots, P_m(\mathbf{Y_m}).$$

Let g be a bijection defined on the set of variables occurring into the metaquery $\mathcal{Q}$, such that, for each variable V of $\mathcal{Q}$, $g(V)$ is a new constant associated to V. Let L be a literal scheme of $\mathcal{Q}$, then $g(L)$ denotes the ground atom obtained from L substituting each ordinary and predicate variable V of L with $g(V)$.

Consider the database $DB^{\mathcal{Q}_1} = \{g(P_i(\mathbf{X_i})) \mid 1 \leq i \leq m\}$, i.e. the database obtained grounding the literal schemes occurring into $\mathcal{Q}_1$ through the mapping g. Then $\sigma_1 = \{\langle P_i(\mathbf{X_i}), g(P_i)(\mathbf{X_i})\rangle \mid 1 \leq i \leq m\}$ belongs to $\Sigma(\mathcal{Q}_1, DB^{\mathcal{Q}_1}, I, 0, T_2)$, for each $I \in \mathbf{I}$. Indeed, the relation

$$r = g(P_1)(\mathbf{X_1}) \bowtie g(P_2)(\mathbf{X_2}) \bowtie \ldots \bowtie g(P_m)(\mathbf{X_m})$$

is not empty on the database $DB^{\mathcal{Q}_1}$ by construction, and $cnf(\sigma_1(\mathcal{Q}_1))$ is

$$\frac{|\pi_{att(\mathbf{X_2} \cup \ldots \cup \mathbf{X_m})}(r)|}{|g(P_2)(\mathbf{X_2}) \bowtie \ldots \bowtie g(P_m)(\mathbf{X_m})|} > 0,$$

$cvr(\sigma_1(\mathcal{Q}_1))$ is

$$\frac{|\pi_{att(\mathbf{X_1})}(r)|}{|g(P_1)(\mathbf{X_1})|} > 0,$$

and $sup(\sigma_1(\mathcal{Q}_1))$ is

$$\max_{2 \leq i \leq m} \left\{ \frac{|\pi_{att(\mathbf{X_i})}(r)|}{|g(P_i)(\mathbf{X_i})|} \right\} > 0$$

on $DB^{\mathcal{Q}_1}$. $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$ implies that there exists an instantiation $\sigma_2 \in \Sigma(\mathcal{Q}_2, DB^{\mathcal{Q}_1}, I, 0, T_2)$ and a variable mapping h such that $\sigma_1 =_{T_2} h(\sigma_2)$. Let $\sigma_2 = \{\langle P_j(\mathbf{Y_j}), r_j(\mathbf{Y'_j})\rangle \mid 1 \leq j \leq m\}$, where r_j is a relation of $DB^{\mathcal{Q}_1}$, and $\mathbf{Y'_j}$ is obtained by permutation of the variables and constants in $\mathbf{Y_j}$, possibly augmented with a set of padding variables, for each $j = 1, \ldots, m$. Then there exists a permutation ρ[2] of $\{1, \ldots, m\}$ such that

$$\left\{ \begin{array}{l} P_{\rho(j)}(\mathbf{X}_{\rho(\mathbf{j})}) =_{T_2} h(P_j(\mathbf{Y_j})) \\ g(P_{\rho(j)})(\mathbf{X}_{\rho(\mathbf{j})}) =_0 r_j(h(\mathbf{Y'_j})) \end{array} \right. , \; j = 1, \ldots, m$$

[2]Recall that $\rho(1) = 1$ by definition of redundant instantiations.

Hence, it is immediate to build from the permutation ρ a reordering π of $\mathcal{Q}_2$ such that $\mathcal{Q}_1 =_{T_2} h(\pi(\mathcal{Q}_2))$.
($\Leftarrow$) The inverse implication is immediate. □

For example, consider the general metaquery $\mathcal{Q} = P(X,Y) \leftarrow Q(Y,Z), R(Z,X)$ and the following two redundant $2,0$-variants of $\mathcal{Q}$: $\mathcal{Q}_1 = P(X,Y) \leftarrow Q(Y,Z,A), R(Z,X,B)$ and $\mathcal{Q}_2 = P(Y,X) \leftarrow Q(Z,Y,A), R(X,Z,B)$. Then the database $DB^{\mathcal{Q}_1}$ is $\{p(x,y), q(y,z,a), r(z,x,b)\}$ and

$$\begin{aligned} \sigma_1 = \{&\langle P(X,Y), p(X,Y)\rangle, \\ &\langle Q(Y,Z,A), q(Y,Z,A)\rangle, \\ &\langle R(Z,X,B), r(Z,X,B)\rangle\} \end{aligned}$$

certainly belongs to $\Sigma(\mathcal{Q}_1, DB^{\mathcal{Q}_1}, I, k, 0)$, for each $I \in \mathbf{I}$ and $k \in [0,1)$. Consider the instantiation

$$\begin{aligned} \sigma_2 = \{&\langle P(Y,X), p(X,Y)\rangle, \\ &\langle R(X,Z,B), q(X,Z,B)\rangle, \\ &\langle Q(Z,Y,A), r(Z,Y,A)\rangle\} \end{aligned}$$

of $\mathcal{Q}_2$ belonging to $\Sigma(\mathcal{Q}_2, DB^{\mathcal{Q}_1}, I, k, 0)$, for each $I \in \mathbf{I}$ and $k \in [0,1)$, and redundant to σ_1, as $\sigma_1 =_0 h(\sigma_2)$, where h is such that $h(P) = P$, $h(Q) = R$, $h(R) = Q$, $h(X) = Y$, $h(Y) = X$, $h(Z) = Z$, $h(A) = B$, and $h(B) = A$. Now, we can build a reordering π such that $\mathcal{Q}_1 =_0 h(\pi(\mathcal{Q}_2))$. In particular, π is such that $\pi(P(X,Y)) = P(X,Y)$, $\pi(Q(Z,Y,A)) = R(X,Z,B)$, and $\pi(R(X,Z,B)) = Q(Z,Y,A)$.

Vice versa, consider the general metaquery $\mathcal{Q} = P(X,Y) \leftarrow Q(Y,Z), R(Z,W)$ and the following two $1,0$-variants of $\mathcal{Q}$: $\mathcal{Q}_1 = \mathcal{Q}$ and $\mathcal{Q}_2 = P(Y,X) \leftarrow Q(Z,Y), R(W,Z)$. It is easy to verify that there does not exist a variable mapping h and a reordering π such that $\mathcal{Q}_1 =_0 h(\pi(\mathcal{Q}_2))$. Nevertheless, if we release the constraint that π must map the head of $\mathcal{Q}_2$ into itself, we can obtain the above equality as follows: $h(P) = R$, $h(R) = P$, $h(Q) = Q$, $h(X) = W$, $h(W) = X$, $h(Y) = Z$, $h(Z) = Y$, $\pi(P(Y,X)) = R(W,Z)$, $\pi(Q(Z,Y)) = Q(Z,Y)$, and $\pi(R(W,Z)) = P(Y,X)$. Now, we verify that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are not redundant under type-0 semantics. Let DB^{sup} be the database $\{p(x',y'), q(y,z), r(z,w)\}$. Then, for each $k \in [0,1)$, the instantiation

$$\begin{aligned} \sigma_1 = \{&\langle P(X,Y), p(X,Y)\rangle, \langle Q(Y,Z), q(Y,Z)\rangle, \\ &\langle R(Z,W), r(Z,W)\rangle\} \end{aligned}$$

belongs to $\Sigma(\mathcal{Q}_1, DB^{sup}, sup, k, 0)$, while the unique instantiation of $\Sigma(\mathcal{Q}_2, DB^{sup}, sup, k, 0)$ having non zero support is

$$\sigma_2 = \{\langle P(Y,X), p(Y,X)\rangle, \langle Q(Z,Y), r(Z,Y)\rangle, \langle R(W,Z), q(W,Z)\rangle\}$$

but $\sigma_2 \not\equiv_0 \sigma_1$. On the contrary, the instantiation

$$\sigma_2' = \{\langle R(W,Z), p(W,Z)\rangle, \langle Q(Z,Y), r(Z,Y)\rangle, \langle P(Y,X), q(Y,Z)\rangle\}$$

of $\mathcal{Q}_2$ on DB^{sup} is such that $sup(\sigma_2'(\mathcal{Q}_2))$ on DB^{sup} is zero.

Let $\mathcal{Q}$ be a general metaquery, let $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$, and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two T_1,T_2-variants of $\mathcal{Q}$ such that $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$. We say that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are *redundant via h and* π, to denote a reordering π of $\mathcal{Q}_2$ and a variable mapping of $\mathcal{Q}_2$ into $\mathcal{Q}_1$, such that $\mathcal{Q}_1 =_{T_2} h(\pi(\mathcal{Q}_2))$ (which exist by Theorem 4.16).

Definition 4.17 (Sets of variants of a metaquery) Let $\mathcal{Q}$ be a metaquery, and let $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$. Then

– $\bigvee_{T_1,T_2} \mathcal{Q}$ denotes the set of all the T_1,T_2-variants of $\mathcal{Q}$;

– $\bigvee^{\equiv}_{T_1,T_2} \mathcal{Q}$ denotes the set of all the T_1,T_2-variants of $\mathcal{Q}$ that are not redundant under the type-T_2 semantics.

The following theorem explains how variants are related to the property of redundancy between instantiations, and establishes an alternative way to calculate the set $\widehat{\Sigma}(\mathcal{Q}, DB, I, k, T)$.

Two technical definitions are needed in the theorem and provided next.

Definition 4.18 (Restriction of a literal scheme) Let $P(\mathbf{X})$ be a literal scheme and let S be a set of ordinary variables. Then the *restriction* $rest(P(\mathbf{X}), S)$ of $P(\mathbf{X})$ on S, is the literal scheme $P(\mathbf{Y})$ obtained by deleting from $\mathbf{X}$ all the variables which do not appear into S.

For example, the restriction of $P(X, Z, a, Y)$ on the set of ordinary variables $\{X, Y\}$ is the literal scheme $P(X, a, Y)$.

Definition 4.19 (Restriction of an instantiation) Let σ be a type-T instantiation, $T \in \{0,1,2\}$, and let S be a set of ordinary variables. Then the *restriction* $rest(\sigma, S)$ of σ on S is the type-T instantiation σ' such that $\sigma' = \{\langle rest(P(\mathbf{X}), S), p(\mathbf{Y})\rangle \mid \langle P(\mathbf{X}), p(\mathbf{Y})\rangle \in \sigma\}$. Let Σ be a set of instantiations. Then the *restriction* $rest(\Sigma, S)$ of Σ on S is the set $\{rest(\sigma) \mid \sigma \in \Sigma\}$.

Theorem 4.20 *Let $\mathcal{Q}$ be a general metaquery, DB be a database, $I \in \mathbf{I}$, $k \in [0,1)$, $T \in \{1,2\}$, and let*

$$\mathcal{M} = \bigcup_{\mathcal{Q}' \in \bigvee_{T,0} \mathcal{Q}} \Sigma(\mathcal{Q}', DB, I, k, 0)\text{, and}$$

$$\widehat{\mathcal{M}} = \bigcup_{\mathcal{Q}' \in \bigvee^{\equiv}_{T,0} \mathcal{Q}} \widehat{\Sigma}(\mathcal{Q}', DB, I, k, 0).$$

Then

1. $\Sigma(\mathcal{Q}, DB, I, k, T) \equiv_T rest\,(\mathcal{M}, ov(\mathcal{Q}))$*, and*
2. $\widehat{\Sigma}(\mathcal{Q}, DB, I, k, T) \doteq_T rest(\widehat{\mathcal{M}}, ov(\mathcal{Q}))$*.*

Proof. Let $\mathcal{Q} = P_1(\mathbf{X_1}) \leftarrow P_2(\mathbf{X_2}), \ldots, P_m(\mathbf{X_m})$.

(1) Consider the inclusion $\Sigma(\mathcal{Q}, DB, I, k, T) \supseteq rest(\mathcal{M}, ov(\mathcal{Q}))$. Let $\mathcal{Q}'$ be a $T,0$-variant of $\mathcal{Q}$ and let σ'' be a type-0 instantiation of $\mathcal{Q}'$ belonging to $\Sigma(\mathcal{Q}', DB, I, k, 0)$. Then $\mathcal{Q}'$

$$P_1(\Pi_1(\mathbf{X_1} \cup \mathbf{A_1})) \leftarrow P_2(\Pi_2(\mathbf{X_2} \cup \mathbf{A_2})), \ldots, P_m(\Pi_m(\mathbf{X_m} \cup \mathbf{A_m}))$$

where $\mathbf{A_1}, \ldots, \mathbf{A_m}$ are possibly empty sets of padding ordinary variables and $\Pi_1, \ldots, \Pi_m$ are permutations of the sets $\mathbf{X_1} \cup \mathbf{A_1}, \ldots, \mathbf{X_m} \cup \mathbf{A_m}$ respectively, while $\sigma'' = \{\langle P_j(\Pi_j(\mathbf{X_j} \cup \mathbf{A_j})), p_j(\mathbf{Y_j})\rangle \mid 1 \le j \le m\}$. Now consider the instantiation $\sigma' = rest(\sigma'', ov(\mathcal{Q})) = \{\langle P_j(\Pi_j(\mathbf{X_j})), p_j(\mathbf{Y_j})\rangle \mid 1 \le j \le m\}$. Then σ' is a type-T instantiation of $\mathcal{Q}$. As $\sigma'(\mathcal{Q})$ is identical to $\sigma''(\mathcal{Q}')$, then $\sigma' \in \Sigma(\mathcal{Q}, DB, I, k, T)$.

Consider now the inclusion $rest(\mathcal{M}, ov(\mathcal{Q})) \supseteq \Sigma(\mathcal{Q}, DB, I, k, T)$. Let σ be an instantiation belonging to $\Sigma(\mathcal{Q}, DB, I, k, T)$. Now we show that there exists a $T,0$-variant $\mathcal{Q}'$ of $\mathcal{Q}$ such that σ belongs to $rest(\Sigma(\mathcal{Q}', DB, I, k, 0), ov(\mathcal{Q}))$. Let $\sigma = \{\langle P_j(\mathbf{X_j}), p_j(\mathbf{Y_j})\rangle \mid 1 \le j \le m\}$, then, by definition of instantiation, there exist possibly empty sets $\mathbf{A_1}, \ldots, \mathbf{A_m}$ of padding variables such that $\mathbf{Y_j} = \mathbf{X_j} \cup \mathbf{A_j}$, for $j = 1, \ldots, m$, and permutations $\Pi_1, \ldots, \Pi_m$ of the sets $\mathbf{X_1} \cup \mathbf{A_1}, \ldots, \mathbf{X_m} \cup \mathbf{A_m}$ respectively, such that the sequence of ordinary variables and constants $\Pi_j(\mathbf{X_j} \cup \mathbf{A_j})$ coincides with the sequence $\mathbf{Y_j}$, for each $j = 1, \ldots, m$. Consider the metaquery

$$\mathcal{Q}' = P_1(\Pi_1(\mathbf{X_1} \cup \mathbf{A_1})) \leftarrow P_2(\Pi(\mathbf{X_2} \cup \mathbf{A_2})), \ldots, P_m(\Pi_m(\mathbf{X_m} \cup \mathbf{A_m}))$$

then $\mathcal{Q}'$ is a $T,0$-variant of $\mathcal{Q}$, and $\sigma'' = \{\langle P_j(\Pi_j(\mathbf{X_j} \cup \mathbf{A_j})), p_j(\mathbf{Y_j})\rangle \mid 1 \le j \le m\}$ is a type-0 instantiation belonging to $\Sigma(\mathcal{Q}', DB, I, k, 0)$, as $\sigma''(\mathcal{Q}')$ is identical to $\sigma(\mathcal{Q})$. Clearly, $\sigma' = rest(\sigma'', ov(\mathcal{Q})) =$

$\{\langle P_j(\Pi_j(\mathbf{X_j})), p_j(\mathbf{Y_j})\rangle \mid 1 \le j \le m\}$ is such that $\sigma =_T \sigma'$.

(2) First we note that $\mathcal{M} \equiv_T \widehat{\mathcal{M}}$ by construction. Then, it follows that:

$$\widehat{\Sigma}(\mathcal{Q}, DB, I, k, T) \equiv_T \Sigma(\mathcal{Q}, DB, I, k, T) \equiv_T$$
$$rest(\mathcal{M}, ov(\mathcal{Q})) \equiv_T rest(\widehat{\mathcal{M}}, ov(\mathcal{Q})).$$

Now we prove that $\widehat{\mathcal{M}}$ is the smallest subset of $\mathcal{M}$ such that $\mathcal{M} \equiv_T \widehat{\mathcal{M}}$, or equivalently that for each pair $\mathcal{Q}'$, $\mathcal{Q}''$ of distinct metaqueries belonging to $\overset{\equiv}{\underset{T,0}{\bigvee}} \mathcal{Q}$, and for each pair σ', σ'' of instantiations such that $\sigma' \in \widehat{\Sigma}(\mathcal{Q}', DB, I, k, 0)$ and $\sigma'' \in \widehat{\Sigma}(\mathcal{Q}'', DB, I, k, 0)$, it holds that $\sigma' \not\equiv_0 \sigma''$. By contradiction, assume that there exist $\mathcal{Q}'$, $\mathcal{Q}''$, σ', and σ'' as defined above, such that $\sigma' \equiv_0 \sigma''$. Let

$$\mathcal{Q}' = P_1(\Pi'_1(\mathbf{X_1} \cup \mathbf{A_1})) \leftarrow$$
$$P_2(\Pi'_2(\mathbf{X_1} \cup \mathbf{A_2})), \ldots, P_m(\Pi'_m(\mathbf{X_m} \cup \mathbf{A_m}))$$
$$\mathcal{Q}'' = P_1(\Pi''_1(\mathbf{X_1} \cup \mathbf{B_1})) \leftarrow$$
$$P_2(\Pi''_2(\mathbf{X_1} \cup \mathbf{B_2})), \ldots, P_m(\Pi''_m(\mathbf{X_m} \cup \mathbf{B_m}))$$

then $\sigma' \equiv_0 \sigma''$ implies that there exists a variable mapping h and a permutation ρ of $\{1, \ldots, m\}$, which maps the head of $\mathcal{Q}'$ into the head of $\mathcal{Q}''$ by definition of redundant instantiations, such that

$$P_j(\Pi'_j(\mathbf{X_j} \cup \mathbf{A_j})) =_0 h(P_{\rho(j)}(\Pi''_{\rho(j)}(\mathbf{X}_{\rho(\mathbf{j})} \cup \mathbf{B}_{\rho(\mathbf{j})})))$$

and thus a reordering π such that $\mathcal{Q}' =_0 h(\pi(\mathcal{Q}''))$, i.e. $\mathcal{Q}'$ and $\mathcal{Q}''$ are redundant under type-0 semantics, a contradiction. Thus $\widehat{\mathcal{M}}$ is the smallest subset of $\mathcal{M}$ such that $\widehat{\mathcal{M}} \equiv_T \mathcal{M}$.

In order to conclude the proof, we need the following technical result.

Lemma 4.21 *Let $\mathcal{Q}$ be a general metaquery, let DB be a database, let $T \in \{1, 2\}$, let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two non redundant $T, 0$-variants of $\mathcal{Q}$ and let σ_1 and σ_2 be two instantiations of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ on DB. Then $rest(\sigma_1, ov(\mathcal{Q})) \not\equiv_0 rest(\sigma_2, ov(\mathcal{Q}))$.*

Proof. Let $\sigma_1 = \{\langle P_j(\mathbf{X_j}), p_j(\mathbf{X_j})\rangle \mid 1 \le j \le m\}$ and $\sigma_2 = \{\langle P_j(\mathbf{Y_j}), q_j(\mathbf{Y_j})\rangle \mid 1 \le j \le m\}$. By contradiction, assume that $rest(\sigma_1, ov(\mathcal{Q})) \equiv_0 rest(\sigma_2, ov(\mathcal{Q}))$. Then there exist a variable mapping h and a permutation ρ of $\{1, \ldots, m\}$ such that

$$rest(P_j(\mathbf{X_j}), ov(\mathcal{Q})) =_0$$
$$rest(P_{\rho(j)}(\mathbf{Y}_{\rho(\mathbf{j})}), ov(\mathcal{Q}))$$
$$\text{i.e. } P_j =_0 P_{\rho(j)}, \text{ and}$$
$$p_j(\mathbf{X_j}) =_0 q_{\rho(j)}(\mathbf{Y}_{\rho(\mathbf{j})}) \text{ i.e. } \mathbf{X_j} =_0 \mathbf{Y}_{\rho(\mathbf{j})}$$

and thus a reordering π such that $\mathcal{Q}_1 =_0 h(\pi(\mathcal{Q}_2))$, a contradiction. □

We can thus resume the proof of Theorem 4.20 by noting that, from Lemma 4.21 it follows that $rest(\widehat{\mathcal{M}}, ov(\mathcal{Q}))$ is the smallest subset of $rest(\mathcal{M}, ov(\mathcal{Q}))$ such that $rest(\widehat{\mathcal{M}}, ov(\mathcal{Q})) \equiv_T rest(\mathcal{M}, ov(\mathcal{Q}))$. □

5. Deciding redundance

Theorem 4.20 establishes an important relationship between the set $\widehat{\Sigma}(\mathcal{Q}, DB, I, k, T)$ and the non redundant $T, 0$-variants of a metaquery. Thus, in order to take advantage of this relationship it is needed to have an effective method to check the redundancy of two variants of a metaquery.

This problem is taken into account in the following. Indeed, Theorems 5.2 and 5.3 show that the problems of deciding if two metaquery variants are redundant and if two instantiations of a metaquery are redundant reduce to the *graph isomorphism* problem. These results make redundancy check effective by using well-known algorithms for solving graph isomorphism.

Section 5.1 introduces the concept of *autoset*. Intuitively, an autoset of a metaquery is a set of variables S such that, for each literal scheme of the metaquery, either all the variables of S occur in it, or none of the variables of S occur in it. Autosets are strictly related to the property of variant redundance, as explained by Theorem 5.7 showing that if two variants are redundant via a variable mapping h, then h is a bijection between autosets, besides being a bijection between variables. Hence, autosets can be exploited to speed-up redundancy check, as explained by Corollary 5.10, by considering a *contraction* of the metaquery instead of the original metaquery, where a contraction is obtained by "collapsing" autosets into a single new variable. Finally, Theorem 5.14 provides a method to compute the autosets of a metaquery in linear time.

Now we show that the well-known graph isomorphism algorithms can be exploited in order to check quite efficiently whether two metaquery variants are redundant.

Problem 5.1 (Graph isomorphism) Given two graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$, we say that they are *isomorphic* if there exists a bijection $\rho : V_2 \to V_1$, that we call *isomorphism*, such that for each edge $(u, v) \in E_2$ there exists an edge $(\rho(u), \rho(v)) \in E_1$.

By little abuse of notation and for simplicity, in the following we will denote by $\rho(\mathcal{G}_2)$ the graph $(\{\rho(v) \mid v \in V_2\}, \{(\rho(v), \rho(u)) \mid (u,v) \in E_2\})$.

Theorem 5.2 *Redundance between variants of a general metaquery reduces to graph isomorphism.*

Proof. We reduce general variants redundance to undirected labelled graph isomorphism, that is to the special case of the graph isomorphism problem in which the two graphs are labelled and the isomorphism must preserve labels. Let $\mathcal{Q}$ be a general metaquery, and let $T \in \{0,1,2\}$. We denote by $\mathcal{G}_T(\mathcal{Q})$ an undirected labelled graph associated to $\mathcal{Q}$ and defined as follows. For each variable and constant V of $\mathcal{Q}$ we introduce in $\mathcal{G}_T(\mathcal{Q})$ a labelled node associated to V and in addition a labelled node associated to each occurrence of V in $\mathcal{Q}$. The labels we employ in the reduction are the following:

- y : denote the nodes associated to predicate variables;
- x : denote the nodes associated to ordinary variables;
- c : denote the nodes associated to constants c (i.e. we have a distinct label c for each constant c in the rule);
- h : denotes the node associated to the predicate variable occurring in the head of the rule;
- p : denote the nodes associated to predicate variables occurring in the body of the rule;
- z : denote the nodes associated to ordinary variables and constants occurring in the rule, under type-1 and type-2 semantics;
- $1, 2, 3, \ldots$: i.e. positive integer numbers, denote the nodes associated to ordinary variables and constants occurring in the rule, under type-0 semantics.

For each variable or constant V of $\mathcal{Q}$ and label l, let $ln(V,l)$ denote a node associated to V having label l. For each variable and constant V of $\mathcal{Q}$, let

$$n(V) = \begin{cases} ln(V,y), & \text{if } V \text{ is a predicate variable} \\ ln(V,x), & \text{if } V \text{ is an ordinary variable} \\ ln(V,V), & \text{if } V \text{ is a constant} \end{cases}$$

The graph $\mathcal{G}_T(\mathcal{Q})$ is built as follows:

- for each variable and constant V of $\mathcal{Q}$, put the node $n(V)$ into the graph $\mathcal{G}_T(\mathcal{Q})$;
- for each literal scheme $P(\mathbf{X})$ in the body of $\mathcal{Q}$, put the node $ln(P,p)$ and the edge $(n(P), ln(P,p))$ into $\mathcal{G}_T(\mathcal{Q})$. If P occurs many times in $\mathcal{Q}$, then the nodes $ln(P,p)$ associated to each occurrence of P are to be considered distinct;
- let $P(\mathbf{X})$ be the head of $\mathcal{Q}$, then put the node $ln(P, h)$ and the edge $(n(P), ln(P,h))$ into $\mathcal{G}_T(\mathcal{Q})$;
- for each literal scheme $P(X_1, \ldots, X_n)$ of $\mathcal{Q}$, let v be $ln(P,p)$ if the literal scheme occurs in the body of $\mathcal{Q}$, and $ln(P,h)$ if it occurs in the head of $\mathcal{Q}$. For each $i = 1, \ldots, n$:
  - ∗ put the node $ln(X_i, i)$, and the edges $(v, ln(X_i, i))$ and $(ln(X_i,i), n(X_i))$ into $\mathcal{G}_T(\mathcal{Q})$, for $T = 0$;
  - ∗ put the edge $(v, n(X_i))$ into $\mathcal{G}_T(\mathcal{Q})$, for $T \in \{1,2\}$.

  Under the type-0 semantics, if X_i occurs more times in $\mathcal{Q}$, then the nodes $ln(X_i, i)$ associated to each occurrence of X_i are to be considered distinct, for $i = 1, \ldots, n$.

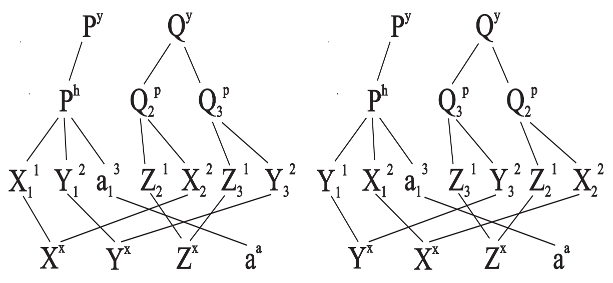
  In the following we denote by $\mathcal{G}_T^{P(X_1,\ldots,X_n)}(\mathcal{Q})$ the subgraph of $\mathcal{G}_T(\mathcal{Q})$ induced (1) by the set of nodes $\{v, ln(X_1,1), \ldots, ln(X_n,n)\}$ under the type-0 semantics, or (2) by the set of nodes $\{v, n(X_1), \ldots, n(X_n)\}$ under the type-1 and type-2 semantics.

We note that labeled graph isomorphism is reducible to graph isomorphism [9].

Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two T_1,T_2-variants, $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$, of a metaquery $\mathcal{Q}$.

Assume that $\mathcal{G}' = \mathcal{G}_{T_2}(\mathcal{Q}_1) = (V_1, E_1)$ and $\mathcal{G}'' = \mathcal{G}_{T_2}(\mathcal{Q}_2) = (V_2, E_2)$ are isomorphic. Then there exists a bijection $\rho : V_1 \to V_2$ such that $\rho(\mathcal{G}') = \mathcal{G}''$. Define $un(ln(U,l))$ as U, then let h the symbol mapping of $\mathcal{Q}_1$ into $\mathcal{Q}_2$ such that $h(V) = un(\rho(ln(V,x)))$ if V is an ordinary variable, and $h(V) = un(\rho(ln(V,y)))$ if V is a predicate variable. By construction, each subgraph $\mathcal{G}_{T_2}^{P(\mathbf{X})}(\mathcal{Q}_1)$ of $\mathcal{G}'$ is mapped by ρ into a subgraph $\mathcal{G}_{T_2}^{Q(\mathbf{Y})}(\mathcal{Q}_2)$ of $\mathcal{G}''$, where $P(\mathbf{X})$ and $Q(\mathbf{Y})$ are literal schemes occurring in $\mathcal{Q}_1$ and $\mathcal{Q}_2$ respectively. It is immediate to build from this mapping a reordering of $\mathcal{Q}_1$ such that $\mathcal{Q}_2 =_{T_2} h(\pi(\mathcal{Q}_1))$ holds.

Indeed, assume that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are redundant under the type-T_2 semantics. Then there exist a symbol mapping h of $\mathcal{Q}_1$ into $\mathcal{Q}_2$ and a reordering of $\mathcal{Q}_1$, such that $\mathcal{Q}_2 =_{T_2} h(\pi(\mathcal{Q}_1))$. Let $\rho : V_1 \to V_2$ be such that $\rho(ln(V,x)) = ln(h(V),x)$ if V is an ordinary variable, and $\rho(ln(V,y)) = ln(h(V),y)$ if V is a predicate variable, for each variable V of $\mathcal{Q}_1$. Furthermore, let ρ such that, for each literal scheme $P(\mathbf{X})$ of $\mathcal{Q}_1$, $\rho(\mathcal{G}_{T_2}^{P(\mathbf{X})}(\mathcal{Q}_1)) = \mathcal{G}_{T_2}^{Q(\mathbf{Y})}(\mathcal{Q}_2)$, where $Q(\mathbf{Y})$ is the literal scheme of $\mathcal{Q}_2$ into which is mapped $P(\mathbf{X})$ via h and π. It is immediate to verify that $\mathcal{G}_2 = \rho(\mathcal{G}_1)$. □



P^y
Q^y
P^h
Q_2^p
Q_3^p
X_1^1 Y_1^2 a_1^3 Z_2^1 X_2^2 Z_3^1 Y_3^2
X^x Y^x Z^x a^a
P^y
Q^y
P^h
Q_3^p
Q_2^p
Y_1^1 X_1^2 a_1^3 Z_3^1 Y_3^2 Z_2^1 X_2^2
Y^x X^x Z^x a^a

Fig. 3. The graphs $\mathcal{G}_0(\mathcal{Q}_1)$ (on the left) and $\mathcal{G}_0(\mathcal{Q}_2)$ (on the right)

For example, let $\mathcal{Q} = P(X, Y, a) \leftarrow Q(Z, X), Q(Z, Y)$ be a general metaquery, and let $\mathcal{Q}_1 = \mathcal{Q}$ and $\mathcal{Q}_2 = P(Y, X, a) \leftarrow Q(Z, X), Q(Z, Y)$ be two $1, 0$-variants of $\mathcal{Q}$. Figure 3 shows the graphs $\mathcal{G}_0(\mathcal{Q}_1)$ and $\mathcal{G}_0(\mathcal{Q}_2)$ associated respectively to $\mathcal{Q}_1$ and $\mathcal{Q}_2$ under the type-0 semantics. There the superscripts denote the labels associated to each node, while the subscripts denote the literal scheme in which the variables appear. It can be verified that the two graphs are isomorphic. The bijection between the nodes of the two graphs so obtained preserves the labelling of the nodes. In particular, let $\rho(X^x) = Y^x$, $\rho(Y^x) = X^x$, $\rho(Y_1^1) = X_1^1$, $\rho(X_1^2) = Y_1^2$, $\rho(Y_3^2) = X_2^2$, $\rho(X_2^2) = Y_3^2$, $\rho(Q_3^p) = Q_2^p$, $\rho(Q_2^p) = Q_3^p$, and $\rho(V) = V$ otherwise, then $\mathcal{G}_0(\mathcal{Q}_1) = \rho(\mathcal{G}_0(\mathcal{Q}_2))$. Let h the variable mapping of $\mathcal{Q}_2$ into $\mathcal{Q}_1$ obtained as follows: $h(X) = un(\rho(X^x)) = un(Y^x) = Y$, $h(Y) = un(\rho(Y^x)) = un(X^x) = X$, $h(Z) = un(\rho(Z^x)) = un(Z^x) = Z$, $un(\rho(P^y)) = un(P^y) = P$, and $un(\rho(Q^y)) = un(Q^y) = Q$. Furthermore, let π the reordering of $\mathcal{Q}_2$ obtained from $\rho(\mathcal{G}_0^{P(Y,X,a)}(\mathcal{Q}_2)) = \mathcal{G}_0^{P(X,Y,a)}(\mathcal{Q}_1)$, $\rho(\mathcal{G}_0^{Q(Z,X)}(\mathcal{Q}_2)) = \mathcal{G}_0^{Q(Z,Y)}(\mathcal{Q}_1)$, and $\rho(\mathcal{G}_0^{Q(Z,Y)}(\mathcal{Q}_2)) = \mathcal{G}_0^{Q(Z,X)}(\mathcal{Q}_1)$, i.e. such that $\pi(P(X, Y, a)) = P(X, Y, a)$, $\pi(Q(Z, X)) = Q(Z, Y)$ and $\pi(Q(Z, Y)) = Q(Z, X)$. Then $\mathcal{Q}_1 =_0 h(\pi(\mathcal{Q}_2))$, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are redundant under the type-0 semantics.

We recall that graph isomorphism belong to NP and it is not known to belong to P, while there is strong theoretical evidence against its NP-completeness [14]. Nevertheless, from a practical point of view, the backtracking algorithms available for solving graph isomorphism perform quite well [19].

Thus Theorem 5.2 is relevant as it states a practical way to check the redundance of two variants. Moreover, it permits to single out interesting subsets of metaqueries for which redundance checking is tractable. Indeed, while the best worst-case bound for a graph isomorphism algorithm actually known is $\exp \sqrt{cn \log n}$ for a graph with n nodes [5], for different classes of graphs, as, for example, for trees [1], planar graphs [15], graphs of bounded degree [17], graphs with colored (i.e. labelled) vertices and bounded color classes [4], polynomial time algorithms are known.

In order to exploit result of Theorem 4.20 we also need an effective method to check redundancy of two instantiations of the same metaquery variant.

Theorem 5.3 *Redundance between instantiations of a general metaquery reduces to graph isomorphism.*

Proof. Let $\mathcal{Q}$ be a general metaquery, let σ a type-T instantiation of $\mathcal{Q}$, and let $\nu(\sigma(\mathcal{Q}))$ be the general metaquery obtained by replacing each atom $p(\mathbf{X})$ occurring into $\sigma(\mathcal{Q})$ with the relation pattern $P(p, \mathbf{X})$

Given two instantiations σ and σ' of a general metaquery $\mathcal{Q}$, then $\sigma \equiv_T \sigma'$ iff $\mathcal{G}_0(\nu(\sigma(\mathcal{Q})))$ and $\mathcal{G}_0(\nu(\sigma'(\mathcal{Q})))$ are isomorphic, can be proved by following the same line of reasoning of Theorem 5.2. □

For example, let $\mathcal{Q}$ be $P(X) \leftarrow Q(X, Y), R(X, Y)$, and let σ and σ' be two type-0 instantiations of $\mathcal{Q}$ such that $\sigma(\mathcal{Q}) = a(X) \leftarrow b(X, Y), c(X, Y)$, and $\sigma'(\mathcal{Q}) = a(X) \leftarrow c(X, Y), b(X, Y)$. Then, $\nu(\sigma(\mathcal{Q})) = P(a, X) \leftarrow P(b, X, Y), P(c, X, Y)$, and $\nu(\sigma'(\mathcal{Q})) = P(a, X) \leftarrow P(c, X, Y), P(b, X, Y)$, while the graphs $\mathcal{G}_0(\nu(\sigma(\mathcal{Q})))$ and $\mathcal{G}_0(\nu(\sigma'(\mathcal{Q})))$ are isomorphic.

5.1. Autosets

Now we introduce the concept of autoset and show that it is strictly related with the property of variant redundancy.

Definition 5.4 Let $\mathcal{Q}$ be a metaquery, and let S be a subset of variables of $\mathcal{Q}$. We denote by $\mathcal{Q}^S$ the set $\{P(\mathbf{X}) \mid S \subseteq \{P\} \cup \mathbf{X}$ and $P(\mathbf{X})$ occurs into $\mathcal{Q}\}$, and by $\overline{\mathcal{Q}^S}$ the set $Q^{\emptyset} - Q^S$, i.e. the set composed by the literal schemes occurring in Q but not in Q^S.

Definition 5.5 (Autoset) Let $\mathcal{Q}$ be a metaquery and let S be a set of variables. We say that S is an *autoset* of $\mathcal{Q}$ if

1. S is a set of ordinary variables and none of the variables of S occur in a literal scheme of $\overline{\mathcal{Q}^S}$, or
2. $S = \{P\}$, where P is a predicate variable of $\mathcal{Q}$.

We call *ordinary* (*predicate* resp.) *autoset* an autoset composed solely by ordinary (predicate resp.) variables.

For example, let $\mathcal{Q}$ be

$$P(X, Y, W) \leftarrow Q(X, Y, Z), P(Z, W).$$

Then the autosets of $\mathcal{Q}$ are $S_1 = \{P\}$, $S_2 = \{Q\}$, $S_3 = \{X\}$, $S_4 = \{Y\}$, $S_5 = \{X, Y\}$, $S_6 = \{W\}$ and $S_7 = \{Z\}$. Indeed, $\mathcal{Q}^{\{X,Y\}} = \{P(X, Y, W), Q(X, Y, Z)\}$, whereas nor X neither Y occur in $\overline{\mathcal{Q}^{\{X,Y\}}} = \{P(Z, W)\}$.

Each variable of a metaquery form itself an autoset. Nevertheless, every variable belong to one and only one maximal autoset, as explained by the following property.

Proposition 5.6 *Let $\mathcal{Q}$ be a metaquery, let X be an ordinary variable of $\mathcal{Q}$, and let $S \supseteq \{X\}$ be an autoset of $\mathcal{Q}$. Then, $\mathcal{Q}^{\{X\}} = \mathcal{Q}^S$.*

Proof. It follows from the definition of autoset, that $\mathcal{Q}^{\{X\}} \supseteq \mathcal{Q}^S$, as each literal scheme in $\mathcal{Q}^S$ contains all the variables in the set S, hence also the variable X. Furthermore, as $X \in S$ then X cannot belong to none of the literal schemes in the set $\overline{\mathcal{Q}^S}$, thus $\mathcal{Q}^S \supseteq \mathcal{Q}^{\{X\}}$, i.e. $\mathcal{Q}^{\{X\}} = \mathcal{Q}^S$. □

Thus the maximal autosets of a metaquery $\mathcal{Q}$ form a partition of the variables of $\mathcal{Q}$.

For example, let $\mathcal{Q}$ be

$$P(X, Y, U, V, W) \leftarrow Q(X, Y, W), Q(U, V, W),$$

then the maximal autosets of $\mathcal{Q}$ are $S_1 = \{P\}$, $S_2 = \{Q\}$, $S_3 = \{X, Y\}$, $S_4 = \{U, V\}$, $S_5 = \{W\}$.

The following theorem explains why autosets are strictly related to the property of variant redundance. Indeed, it proves that, if $\mathcal{Q}_1 =_{T_2} h(\pi(\mathcal{Q}))$ holds, then the variable mapping h is a bijection between autosets, besides being a bijection between variables.

Theorem 5.7 *Let $\mathcal{Q}$ be a general metaquery, let $T_1, T_2 \in \{0, 1, 2\}$, $T_1 > T_2$, and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two T_1, T_2-variants of $\mathcal{Q}$ such that $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$ via h and π. Then for each maximal autoset S_1 of $\mathcal{Q}_1$ there exists a maximal autoset S_2 of $\mathcal{Q}_2$ such that $S_1 = h(S_2)$.*

Proof. Let S_2 be a maximal autoset of $\mathcal{Q}_2$. We note that the variables in S_2 occur in each literal scheme of $\mathcal{Q}_2^{S_2}$, thus the variables in $h(S_2)$ certainly occur in each literal scheme of $h(\mathcal{Q}_2^{S_2})$. Suppose that there exists $X \in h(S_2)$ such that X occurs in a literal scheme of $\mathcal{Q}_1$ which is not in $h(\mathcal{Q}_2^{S_2})$. If follows from the hypothesis that h is not a bijection, a contradiction. Hence $h(S_2)$ is an autoset of $\mathcal{Q}_1$, and there exists a maximal autoset S_1 of $\mathcal{Q}_1$ such that $S_1 \supseteq h(S_2)$, and $\mathcal{Q}_1^{S_1} = h(\mathcal{Q}_2^{S_2})$.

We note that the variables in $h^{-1}(S_1)$ occur in each literal scheme of $\mathcal{Q}_2^{S_2}$. Suppose that there exists $Y \in h^{-1}(S_1)$ such that Y occurs in a literal scheme of $\mathcal{Q}_2$ which is not in $\mathcal{Q}_2^{S_2}$. If follows from the hypothesis that h is not a bijection a contradiction. Hence, $h^{-1}(S_1)$ is an autoset of $\mathcal{Q}_2$, and there exists a a maximal autoset S_2' of $\mathcal{Q}_2$ such that $S_2' \supseteq h^{-1}(S_1)$, and $\mathcal{Q}_1^{S_1} = h(\mathcal{Q}_2^{S_2'})$.

As we are dealing with maximal autosets, then it is the case that S_2' is S_2. Thus, $S_1 = h(S_2)$. □

For example, let $\mathcal{Q}$:

$$P(X, Y, U, V, W) \leftarrow Q(X, Y, W), Q(U, V, W)$$

be a general metaquery, and let $\mathcal{Q}_1 = \mathcal{Q}$ and $\mathcal{Q}_2 = P(U, V, X, Y, W) \leftarrow Q(X, Y, W), Q(U, V, W)$ be two redundant under type-0 semantics $1, 0$-variants of $\mathcal{Q}$ via π and h. In this case π is such that $\pi(P(U, V, X, Y, W)) = P(X, Y, U, V, W)$, $\pi(Q(X, Y, W)) = Q(U, V, W)$, and $\pi(Q(U, V, W)) = Q(X, Y, W)$, while $h(P) = P$, $h(Q) = Q$, $h(W) = W$, $h(X) = U$, $h(U) = X$, $h(Y) = V$ and $h(V) = Y$. That is, according to Theorem 5.7, h maps the maximal autosets $\{P\}$, $\{Q\}$, $\{W\}$ into themselves, and the maximal autoset $\{X, Y\}$ into $\{U, V\}$ and vice versa.

We will show in the next section that autosets can be exploited to efficiently avoid the generation of certain redundant variants of a metaquery. Furthermore, by Theorem 5.7 it follows that the check described in Theorem 5.2 can be performed taking advantage of the autosets of the metaquery, as accounted for in the following.

Definition 5.8 (Contraction of a metaquery) Let $\mathcal{Q}$ be a metaquery, let $S_1, \ldots, S_n$ be the maximal ordinary autosets of $\mathcal{Q}$, and let $\Phi_1, \ldots, \Phi_n$ be arbitrary ordinary variables not occurring into $\mathcal{Q}$. Then the *contraction* $contr(\mathcal{Q})$ of $\mathcal{Q}$ is the metaquery obtained from $\mathcal{Q}$ substituting each occurrence of the autoset S_i with the ordinary variable Φ_i, for $i = 1, \ldots, m$.

For example, consider the metaquery $\mathcal{Q}$:

$$P(X, U, Y, V, W) \leftarrow Q(X, Y, W), R(V, W, U),$$

with maximal ordinary autosets $S_1 = \{X, Y\}$, $S_2 = \{U, V\}$, and $S_3 = \{W\}$.

Then the contraction $contr(\mathcal{Q})$ of $\mathcal{Q}$ is the metaquery

$$P(\Phi_1, \Phi_2, \Phi_3) \leftarrow Q(\Phi_1, \Phi_3), R(\Phi_2, \Phi_3).$$

Definition 5.9 (Graph associated to a contraction) Let $\mathcal{Q}$ be a general metaquery, and let $T \in \{0,1,2\}$. Let $S_1,\dots,S_n$ be the maximal ordinary autosets of $\mathcal{Q}$, and let $\Phi_1,\dots,\Phi_n$ be arbitrary ordinary variables not occurring into $\mathcal{Q}$. We denote by $contr\mathcal{G}_T(\mathcal{Q})$ the labelled graph defined as follows:

– $\mathbf{T=0}$: the graph $contr\mathcal{G}_T(\mathcal{Q})$ is defined as $\mathcal{G}_T(\mathcal{Q})$ except that the nodes

$$ln(X_1,x),\dots,ln(X_m,x)$$

associated to the ordinary variables $X_1,\dots,X_n$ of $\mathcal{Q}$ are substituted with the nodes

$$ln(\Phi_1,-|S_1|),\dots,ln(\Phi_n,-|S_n|).$$

In particular, let $S_i = Y_{i,1},\dots,Y_{i,k_i}$, for $i = 1,\dots,n$, then the nodes

$$ln(Y_{i,1},x),\dots,ln(Y_{i,k_i},x)$$

are "grouped" and substituted with a unique new node $ln(\Phi_i,-|S_i|)$, for $i=1,\dots,n$.

– $\mathbf{T\in\{1,2\}}$: the graph $contr\mathcal{G}_T(\mathcal{Q})$ is defined as $\mathcal{G}_T(contr(\mathcal{Q}))$ of Theorem 5.2 except that the nodes

$$ln(\Phi_1,x),\dots,ln(\Phi_n,x)$$

associated to the ordinary variables $\Phi_1,\dots,\Phi_n$ of $contr(\mathcal{Q})$ are substituted with the nodes

$$ln(\Phi_1,-|S_1|),\dots,ln(\Phi_n,-|S_n|).$$

Corollary 5.10 *Let $\mathcal{Q}$ be a general metaquery, let $T_1,T_2 \in \{0,1,2\}$, $T_1 > T_2$, and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ two T_1,T_2-variants of $\mathcal{Q}$. Then $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$ iff $contr\mathcal{G}_{T_2}(\mathcal{Q}_1)$ and $contr\mathcal{G}_{T_2}(\mathcal{Q}_2)$ are isomorphic.*

Proof. If can be shown using the same line of reasoning used to prove Theorem 5.2. □

In the following it is shown that the maximal autosets of a metaquery can be computed in linear time. This is done by reducing the maximal autosets problem, defined next, to the well-known modular decomposition problem.

Problem 5.11 (Maximal autosets) Given a metaquery $\mathcal{Q}$, find the maximal autosets of $\mathcal{Q}$.

Definition 5.12 (Vertex partitioning of a graph) Given a graph $\mathcal{G} = (V,E)$ and an initial partition $\mathcal{P} = \{S_1,\dots,S_n\}$ of V, the *vertex partitioning* of $\mathcal{G}$ is the coarsest partition $\mathcal{P}' = \{M_1,\dots,M_m\}$ of V such that:

1. each M_i is a subset of some S_j, and
2. $v,u \in M_i$ implies $N(v) - M_i = N(u) - M_i$, where $N(x) = \{y \mid (x,y) \in E\}$

where coarsest means that each M_i is maximal w.r.t. the second property, i.e. that there not exist $M_j, M_k \in \mathcal{P}'$, $j \neq k$, such that $(\mathcal{P}' - \{M_j,M_k\}) \cup \{M_j \cup M_k\}$ is a vertex partitioning.

The second condition states that each set M_i must be indistinguishable by any vertex v in $V - M_i$, i.e. v is either adjacent to every vertex in M_i, or it is adjacent to no vertex of M_i. Each M_i is also called a *module*.

Problem 5.13 (Modular decomposition) Given a bipartite graph $\mathcal{G} = (V,U,E)$, find the vertex partitioning of $\mathcal{G}$ with initial partition $\{V,U\}$.

The modular decomposition of a bipartite graph can be determined in linear time [10,18].

Theorem 5.14 *Maximal autosets reduces to modular decomposition.*

Proof. The computation of the maximal autosets of a given metaquery can be reduced to modular decomposition of a bipartite graph.

Let $\mathcal{Q} = P_1(\mathbf{X_1}) \leftarrow P_2(\mathbf{X_2}),\dots,P_m(\mathbf{X_m})$ be a metaquery. For each ordinary variable Y of $\mathcal{Q}$ let $g(Y)$ denote a distinct node associated to Y, and for each literal scheme $P(\mathbf{X})$ of $\mathcal{Q}$ let $g(P(\mathbf{X}))$ denote a distinct node associated to $P(\mathbf{X})$.

Let $\mathcal{G}(\mathcal{Q}) = (V,U,E)$ be the bipartite graph associated to $\mathcal{Q}$ defined as follows:

– $V = \{g(Y) \mid Y \in \mathbf{X_1} \cup \ldots \cup \mathbf{X_m}\}$;
– $U = \{g(P_1(\mathbf{X_1})),\dots,g(P_m(\mathbf{X_m}))\}$;
– $E = \{(g(Y), g(P_i(\mathbf{X_i}))) \mid \forall \text{ variable } Y \in \mathbf{X_i}, 1 \le i \le m\}$.

Let M be the set of the modules of the modular decomposition of $\mathcal{G}(\mathcal{Q})$, then

$$\{g^{-1}(m) \mid m \in M \wedge m \subseteq V\} \cup \{\{P\} \mid P \in pv(\mathcal{Q})\}$$

are the autosets of $\mathcal{Q}$.

Indeed, let $m \subseteq V$ be a module of the modular decomposition of $\mathcal{G}(\mathcal{Q})$. It follows from the definition of modular decomposition that for each pair $v,u \in m$, $N(v) = N(u)$ (note that, as the graph is bipartite, for each $w \in m$, $N(w) - m = N(w)$), i.e. that $\mathcal{Q}^{\{g^{-1}(v)\}} = \mathcal{Q}^{\{g^{-1}(u)\}}$. Furthermore, as m is maximal w.r.t. the above property, then $\{g^{-1}(v) \mid v \in m\}$ is a maximal autoset of $\mathcal{Q}$.

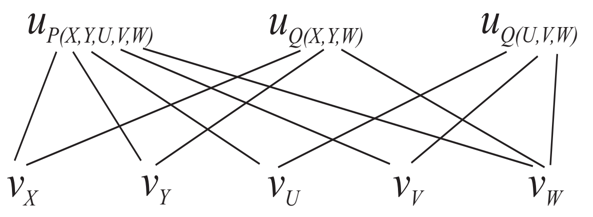


Fig. 4. The bipartite graph associated to $P(X,Y,U,V,W) \leftarrow Q(X,Y,W), Q(U,V,W)$ by Theorem 5.14

Let S be a maximal ordinary autoset of $\mathcal{Q}$. Then, by definition of autoset, $X, Y \in S$ implies that $\mathcal{Q}^{\{X\}} = \mathcal{Q}^{\{Y\}}$, thus $N(g(X)) = N(g(Y))$. As S is maximal, then we can conclude that $m = \{g(X) \mid X \in S\}$ is a module of the modular decomposition of $\mathcal{G}(\mathcal{Q})$. $\square$

For example, the bipartite graph associated to $\mathcal{Q} = P(X,Y,U,V,W) \leftarrow Q(X,Y,W), Q(U,V,W)$ is $\mathcal{G} = (V,U,E)$, with $V = \{v_X, v_Y, v_U, v_V, v_W\}$, $U = \{u_1 = u_{P(X,Y,U,V,W)}, u_2 = u_{Q(X,Y,W)}, u_3 = u_{Q(U,V,W)}\}$, and $E = \{(v_X,u_1),(v_Y,u_1),(v_U,u_1),$ $(v_V,u_1),(v_W,u_1),(v_X,u_2),(v_Y,u_2),(v_W,u_2),(v_U,$ $u_3),(v_V,u_3),(v_W,u_3)\}$. The modular decomposition M of $\mathcal{G}$ is $M = \{\{v_X,v_Y\},\{v_U,v_V\},\{v_W\},\{u_1\},$ $\{u_2\},\{u_3\}\}$. Thus the set of the maximal autosets of $\mathcal{Q}$ is $\{\{X,Y\},\{U,V\},\{W\},\{P\},\{Q\}\}$.

6. Enumerating sets of variants

In this section the problem of enumerating sets of non redundant variants of a metaquery is considered. In particular, first the problem of deciding whether a metaquery admits at least two distinct variants which are redundant (see Section 6.1 in the following) is taken into account, and then it is shown how to efficiently enumerate certain subsets of non redundant variants (see Section 6.2).

We begin by giving the definition of renaming, a particular case of redundant variant.

Definition 6.1 (Renaming) Let $\mathcal{Q}$ be a metaquery, let $T \in \{0,1,2\}$, $T_1 > T_2$, and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two T_1,T_2-variants of $\mathcal{Q}$. We say that $\mathcal{Q}_2$ is a *renaming* of $\mathcal{Q}_1$ under the type-T_2 semantics, if there exists a variable mapping h of $\mathcal{Q}_2$ into $\mathcal{Q}_1$ such that $\mathcal{Q}_1 =_{T_2} h(\mathcal{Q}_2)$. We say that $\mathcal{Q}_2$ is a renaming of $\mathcal{Q}_1$ via h to indicate the variable mapping h.

For example, consider the metaquery $\mathcal{Q}$:

$$P(X,Y,Z) \leftarrow Q(Y,X,W), R(Z,W)$$

and the two $1,0$-variant $\mathcal{Q}_1 = \mathcal{Q}$ and $\mathcal{Q}_2$:

$$P(Y,X,Z) \leftarrow Q(X,Y,W), R(Z,W)$$

of $\mathcal{Q}$. Then, $\mathcal{Q}_2$ is a renaming of $\mathcal{Q}_1$ under type-0 semantics, as $\mathcal{Q}_1 = h(\mathcal{Q}_2)$, where $h(X) = Y$, $h(Y) = X$, and $h(V) = V$ for each other variable V occurring into $\mathcal{Q}$.

Definition 6.2 *Let $\mathcal{Q}$ be a metaquery, and let $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$. Then $\overset{=}{\bigvee\limits_{T_1,T_2}} \mathcal{Q}$ denotes the set of all the T_1,T_2-variants of $\mathcal{Q}$ that are not renamings under the type-T_2 semantics.*

6.1. Deciding proper inclusions between sets of variants

The following chain of inclusions holds

$$\bigvee_{T_1,T_2} \mathcal{Q} \supseteq \overset{=}{\bigvee_{T_1,T_2}} \mathcal{Q} \supseteq \overset{\equiv}{\bigvee_{T_1,T_2}} \mathcal{Q}$$

However, it is important to state if the two inclusions are proper. Indeed, assume that we want to compute the set $\bigvee^{\equiv}_{T_1,T_2} \mathcal{Q}$ and that we know that the first inclusion above is not proper. In this case all we have to do is to enumerate the T_1,T_2-variants of $\mathcal{Q}$.

We begin from the former inclusion.

Problem 6.3 (Self renaming metaquery) Given a metaquery $\mathcal{Q}$, and $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$, check if $\mathcal{Q}$ is T_1,T_2-*self renaming*, that is if there exist two different T_1,T_2-variants $\mathcal{Q}_1$ and $\mathcal{Q}_2$ of $\mathcal{Q}$, such that $\mathcal{Q}_2$ is a renaming of $\mathcal{Q}_1$ under the type-T_2 semantics.

It is easy to verify that a metaquery cannot be $2,1$-self renaming. Indeed, let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two different $2,1$-variants of a metaquery $\mathcal{Q}$. Then, by definition, there exists an integer i, $1 \le i \le m$, where m is the number of literal schemes occurring into $\mathcal{Q}$, such that the i-th relational pattern $P(\mathbf{X})$ of $\mathcal{Q}_1$ and the i-th relation pattern $P(\mathbf{Y})$ of $\mathcal{Q}_2$ are such that $|\mathbf{X}| \neq |\mathbf{Y}|$.

As for $1,0$-self renaming the following result holds.

Theorem 6.4 *Let $\mathcal{Q}$ be a general metaquery. Then $\mathcal{Q}$ is $1,0$-self renaming iff there exists a maximal autoset S of $\mathcal{Q}$ such that $|S| \geq 2$.*

Proof. ($\Rightarrow$) It follows from the definition of renaming and from Theorem 5.7 that for each maximal autoset S of $\mathcal{Q}$ we have $h(S) = S$. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two distinct $1,0$-variants of $\mathcal{Q}$ such that $\mathcal{Q}_2$ is a renaming of $\mathcal{Q}_1$ via h under the type-0 semantics. By contradiction, assume that for each maximal autoset S of $\mathcal{Q}$, $|S| = 1$. Then h is the identity and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are the same metaquery, a contradiction.
($\Leftarrow$) Let S be a maximal autoset of $\mathcal{Q}$ such that $|S| \geq 2$. Let $h' : S \rightarrow S$ be a bijection different from the identity, and let h such that $h(X) = h'(X)$ if $X \in S$, and $h(X) = X$ otherwise. Then $h(\mathcal{Q})$ and $\mathcal{Q}$ are two distinct $1,0$-variants of $\mathcal{Q}$ such that $h(\mathcal{Q})$ is a renaming of $\mathcal{Q}$ under the type-0 semantics. □

For example, let $\mathcal{Q} = P(X,Y,Z) \leftarrow Q(Y,Z,W), R(X,W)$ be a general metaquery. $\{Y,Z\}$ is a maximal autoset of $\mathcal{Q}$, thus let $h(Y) = Z$, $h(Z) = Y$, and $h(V) = V$ otherwise. Then $h(\mathcal{Q}) = P(X,Z,Y) \leftarrow Q(Z,Y,W), R(X,W)$ is a renaming of $\mathcal{Q}$ under the type-0 semantics and a $1,0$-variant of $\mathcal{Q}$.

Next, we consider the $2,0$-self renaming problem.

Definition 6.5 (Trivial renaming) Let $\mathcal{Q}$ be a metaquery, and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two distinct $2,0$-variants of $\mathcal{Q}$ such that $\mathcal{Q}_2$ is a renaming under the type-0 semantics of $\mathcal{Q}_1$ via h. We say that $\mathcal{Q}_2$ is a *trivial renaming* of $\mathcal{Q}_1$ under the type-0 semantics, if h is the identity on the variables of $\mathcal{Q}$.

For example, let $\mathcal{Q} = P(X,Y) \leftarrow Q(Y,Z)$, and let $\mathcal{Q}_1 = P(A,B,Y,X) \leftarrow Q(Y,B,A,Z)$ and $\mathcal{Q}_2 = P(B,A,Y,X) \leftarrow Q(Y,A,B,Z)$ be two $2,0$-variants of $\mathcal{Q}$. Consider the variable mapping h of $\mathcal{Q}_2$ into $\mathcal{Q}_1$ such that $h(A) = B$, $h(B) = A$, and $h(V) = V$ otherwise. Then, $\mathcal{Q}_1 =_0 h(\mathcal{Q}_2)$ and $\mathcal{Q}_2$ is a renaming of $\mathcal{Q}_1$ under the type-0 semantics. Furthermore, $\mathcal{Q}_2$ is a trivial renaming as it is the identity on the variables occurring into $\mathcal{Q}$.

Every general metaquery has $2,0$-variants that are trivial renamings. Indeed, let $\mathcal{Q}$ be a general metaquery and let $P(\mathbf{X})$ be a relation pattern of $\mathcal{Q}$. Then the metaqueries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ obtained from $\mathcal{Q}$ substituting the atom $P(\mathbf{X})$ respectively with $P(\mathbf{X},A,B)$ and $P(\mathbf{X},B,A)$, are $2,0$-variants of $\mathcal{Q}$ and also trivial renamings under the type-0 semantics.

Nevertheless, if we restrict to non trivial renamings the following property holds.

Theorem 6.6 *Let $\mathcal{Q}$ be a general metaquery. Then $\mathcal{Q}$ is non trivially $2,0$-self renaming iff there exists a maximal autoset S of $\mathcal{Q}$ such that $|S| \geq 2$.*

Proof. The proof is analogous to that of Theorem 6.4. □

Therefore, Theorems 6.4 and 6.6 permit us to decide if the former inclusion is proper or not. Next, we consider the second inclusion, namely, $\bigvee\limits_{T_1,T_2}^{=} \mathcal{Q} \supseteq \bigvee\limits_{T_1,T_2}^{\equiv} \mathcal{Q}$.

Problem 6.7 (Graph automorphism) Given an undirected graph $\mathcal{G} = (V,E)$, we say that it is *automorphic* if there exists a non trivial isomorphism ρ, i.e. an isomorphism different from the identity, such that such that $\mathcal{G} = \rho(\mathcal{G})$.

Problem 6.8 (Self redundant metaquery) Given a metaquery $\mathcal{Q}$, and $T_1, T_2 \in \{0,1,2\}$, $T_1 > T_2$, check if $\mathcal{Q}$ is T_1,T_2-*self redundant*, that is, if there exist two different T_1,T_2-variants $\mathcal{Q}_1$ and $\mathcal{Q}_2$ of $\mathcal{Q}$, such that (1) $\mathcal{Q}_1 \equiv_{T_2} \mathcal{Q}_2$ and (2) $\mathcal{Q}_2$ is not a renaming of $\mathcal{Q}_1$.

Theorem 6.9 *Self redundant general metaquery is reducible to graph automorphism.*

Proof. Let $\mathcal{Q}$ be a general metaquery. To show that $contr\mathcal{G}_1(\mathcal{Q})$ is automorphic iff $\mathcal{Q}$ is T_1,T_2-self redundant, for each $T_1,T_2 \in \{0,1,2\}$, $T_1 > T_2$, the proof proceeds as in Theorem 5.2. □

Therefore, Theorem 6.9 permit us to decide if the latter inclusion is proper or not.

6.2. *Efficiently enumerating sets of variants*

This section reports two results concerning efficient enumeration of set of variants. First, it is shown that the set $\bigvee_{1,0}^{=} \mathcal{Q}$ is efficiently computable (see the following Theorem 6.10), and then it is shown that the set $\bigvee_{2,0}^{\equiv} \mathcal{Q}$ can be efficiently generated from the set $\bigvee_{1,0}^{\equiv} \mathcal{Q}$ (see Theorem 6.13 in the following).

We begin by showing that the set $\bigvee\limits_{1,0}^{=} \mathcal{Q}$ is efficiently computable.

Theorem 6.10 Let $\mathcal{Q}$:

$$P_1(\mathbf{X_1}) \leftarrow P_2(\mathbf{X_2}), \ldots, P_m(\mathbf{X_m})$$

be a general metaquery, let $S_1, S_2, \ldots, S_n$ be the maximal ordinary autosets of $\mathcal{Q}$, let $j_1, j_2, \ldots, j_n \in \{1,\ldots,m\}$ such that $S_i \subseteq \mathbf{X_{j_i}}$, for each $i = 1,2,\ldots,n$, and let $\Pi : ov(\mathcal{Q}) \rightarrow \{1,\ldots,|ov(\mathcal{Q})|\}$ be a bijection.

Let $\mathcal{M}$ be the set of the $1,0$-variants of $\mathcal{Q}$ obtained by (1) putting the ordinary variables of the i-th maximal autoset S_i into the j_i-th relational pattern of $\mathcal{Q}$ according to the order Π (i.e., if $Y_1, Y_2 \in S_i$ and $\Pi(Y_1) < \Pi(Y_2)$, then Y_1 precedes Y_2), and (2) permuting all the other variable occurrences in the remaining relational patterns. Then

$$\mathcal{M} = \overset{=}{\bigvee_{1,0}} \mathcal{Q}$$

Proof. First, we prove that $\overset{=}{\bigvee_{1,0}} \mathcal{Q} \supseteq \mathcal{M}$. Any metaquery $\mathcal{Q}_1 \in \mathcal{M}$ is a $1,0$-variant of $\mathcal{Q}$ by construction. By absurd, assume that there exists a metaquery $\mathcal{Q}_2 \in \mathcal{M}$ different from $\mathcal{Q}_1$ such that $\mathcal{Q}_2$ is a renaming of $\mathcal{Q}_1$ via h under the type-0 semantics. Let $\mathcal{Q}_1 = P_1(\mathbf{Y_1}) \leftarrow P_2(\mathbf{Y_2}), \ldots, P_m(\mathbf{Y_m})$, and $\mathcal{Q}_2 = P_1(\mathbf{Z_1}) \leftarrow P_2(\mathbf{Z_2}), \ldots, P_m(\mathbf{Z_m})$.

As h maps each maximal autoset into itself, then each ordinary autoset of $\mathcal{Q}$ occupies the same positions among the arguments of each pair $P_i(\mathbf{Y_i})$, $P_i(\mathbf{Z_i})$, for each $i = 1, \ldots, m$ (otherwise it cannot be $P_i(\mathbf{Y_i}) =_0 P_i(\mathbf{Z_i})$). Then, for each maximal autoset S_i of $\mathcal{Q}$, $1 \leq i \leq n$, the two occurrences of S_i into the pair $P_{j_i}(\mathbf{Y_{j_i}})$, $h(P_{j_i}(\mathbf{Z_{j_i}}))$ agree both for position and order. Thus, it is the case that $h(S_i) = S_i$, for each $i = 1, \ldots, n$. We can therefore conclude that $\mathcal{Q}_1 = \mathcal{Q}_2$, a contradiction.

It remains to prove that $\mathcal{M} \supseteq \overset{=}{\bigvee_{1,0}} \mathcal{Q}$. We show that for each $\mathcal{Q}_1 \in \bigvee_{1,0} \mathcal{Q}$ there exists a symbol mapping h such that $h(\mathcal{Q}_1) \in \mathcal{M}$.

Let $\mathbf{d}_i(k_i)$ the k_i-th variable of S_i according to the order defined in the j_i-th literal scheme of $\mathcal{Q}_1$, for each $i = 1, \ldots, n$, $k_i = 1, \ldots, |S_i|$, and let $\mathbf{c}_i(k_i)$ the k_i-th variable of S_i according to the order Π, for each $i = 1, \ldots, n$, $k_i = 1, \ldots, |S_i|$. Let h_i be the bijection from S_i to S_i such that $h_i(\mathbf{d}_i(k_i)) = \mathbf{c}_i(k_i)$, for each $i = 1, \ldots, n$, $k_i = 1, \ldots, |S_i|$.

Consider the symbol mapping h of $\mathcal{Q}$ into $\mathcal{Q}$ such that $h(S_i) = h_i(S_i)$, for each $i = 1, \ldots, n$, and $h(V) = V$ otherwise. It is easy to verify that $h(\mathcal{Q}_1) \in \mathcal{M}$. □

Next, we deal with the size of the set $\overset{=}{\bigvee_{1,0}} \mathcal{Q}$.

Proposition 6.11 *Let $\mathcal{Q}$ be a general metaquery, let a_i be the arity of the i-th relation pattern of $\mathcal{Q}$, for each $i = 1, \ldots, m$, where m is the number of relational patterns occurring in $\mathcal{Q}$, let $S_1, S_2, \ldots, S_n$ be the maximal ordinary autosets of $\mathcal{Q}$, and let $s_j = |S_j|$, for each $j = 1, 2, \ldots, n$. Then*

$$\left| \overset{=}{\bigvee_{1,0}} \mathcal{Q} \right| = \frac{a_1! a_2! \ldots a_k!}{s_1! s_2! \ldots s_n!}$$

Proof. This property follows immediately from Theorem 6.10. □

For example, let $\mathcal{Q} = P(X, Y) \leftarrow Q(X, Y, Z), R(Z)$. The maximal ordinary autosets of $\mathcal{Q}$ are $S_1 = \{X, Y\}$ and $S_2 = \{Z\}$. Let $j_1 = 1$, $j_2 = 3$, and let $\Pi(X) = 1$, $\Pi(Y) = 2$, and $\Pi(Z) = 3$, then $\overset{=}{\bigvee_{1,0}} \mathcal{Q}$ is the following set:

$$\begin{aligned}
\mathcal{Q}_1 &= P(X, Y) \leftarrow Q(X, Y, Z), R(Z) \\
\mathcal{Q}_2 &= P(X, Y) \leftarrow Q(X, Z, Y), R(Z) \\
\mathcal{Q}_3 &= P(X, Y) \leftarrow Q(Y, X, Z), R(Z) \\
\mathcal{Q}_4 &= P(X, Y) \leftarrow Q(Y, Z, X), R(Z) \\
\mathcal{Q}_5 &= P(X, Y) \leftarrow Q(Z, X, Y), R(Z) \\
\mathcal{Q}_6 &= P(X, Y) \leftarrow Q(Z, Y, X), R(Z)
\end{aligned}$$

Consider a $1,0$-variant $\mathcal{Q}'$ of $\mathcal{Q}$ not appearing in the previous set, e.g. $\mathcal{Q}' = P(Y, X) \leftarrow Q(Y, X, Z), R(Z)$. Then, $\mathbf{d_1}(1) = Y$, $\mathbf{d_1}(2) = X$, $\mathbf{d_2}(1) = Z$, $\mathbf{c_1}(1) = X$, $\mathbf{c_1}(2) = Y$, and $\mathbf{c_2}(1) = Z$. Define h as $h(Y) = h(\mathbf{d_1}(1)) = \mathbf{c_1}(1) = X$, $h(X) = h(\mathbf{d_1}(2)) = \mathbf{c_1}(2) = Y$, $h(Z) = h(\mathbf{d_2}(1)) = \mathbf{c_2}(1) = Z$, and $h(P) = P$, $h(Q) = Q$, and $h(R) = R$. Then $h(\mathcal{Q}') = P(X, Y) \leftarrow Q(X, Y, Z), R(Z)$ which corresponds to the metaquery $\mathcal{Q}_1$.

Next we show that the set $\overset{\equiv}{\bigvee_{2,0}} \mathcal{Q}$ is efficiently computable from the set $\overset{\equiv}{\bigvee_{1,0}} \mathcal{Q}$.

Definition 6.12 (Expansion of a metaquery) Let $P(\mathbf{X})$ and $P(\mathbf{Y})$ be two relational patterns. We say that $P(\mathbf{Y})$ is an *expansion* of $P(\mathbf{X})$ if $\mathbf{X} = X_1, \ldots, X_n$ and $\mathbf{Y} = \mathbf{Y_1}, X_1, \mathbf{Y_2}, X_2, \mathbf{Y_3}, \ldots, \mathbf{Y_n}, X_n, \mathbf{Y_{n+1}}$, where each $\mathbf{Y_i}$, $i = 1, \ldots, n+1$, is a possibly empty sequence of padding ordinary variables.

Let $\mathcal{Q}_1 = P_1(\mathbf{X_1}) \leftarrow P_2(\mathbf{X_2}), \ldots, P_m(\mathbf{X_m})$ and $\mathcal{Q}_2 = P_1(\mathbf{Y_1}) \leftarrow P_2(\mathbf{Y_2}), \ldots, P_m(\mathbf{Y_m})$ be two metaqueries. We say that $\mathcal{Q}_2$ is an *expansion* of $\mathcal{Q}_1$ if $P_i(\mathbf{Y_i})$ is an expansion of $P_i(\mathbf{X_i})$, for each $i = 1, \ldots, m$.

Theorem 6.13 *Let $\mathcal{Q}$ be a general metaquery, and let*

$$\mathcal{M} = \bigcup_{\mathcal{Q}' \in \bigvee^{\equiv}_{1,0} \mathcal{Q}} \{\mathcal{Q}'' \mid \mathcal{Q}'' \textit{ is an expansion of } \mathcal{Q}'\}$$

Then $\mathcal{M} = \bigvee^{\equiv}_{2,0} \mathcal{Q}$*.*

Proof. First we prove that $\bigvee^{\equiv}_{2,0} \mathcal{Q} \supseteq \mathcal{M}$. Any metaquery $\mathcal{Q}_1 \in \mathcal{M}$ is a $2,0$-variant of $\mathcal{Q}$ by construction. By absurd, assume that there exists a metaquery $\mathcal{Q}_2 \in \mathcal{M}$ different from $\mathcal{Q}_1$ such that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are redundant via h and π under the type-0 semantics. Let $\mathcal{Q}_1 = P_1(\mathbf{X_1}) \leftarrow P_2(\mathbf{X_2}), \ldots, P_m(\mathbf{X_m})$, and $\mathcal{Q}_2 = P_1(\mathbf{Y_1}) \leftarrow P_2(\mathbf{Y_2}), \ldots, P_m(\mathbf{Y_m})$. Let $\mathbf{A_i}$ ($\mathbf{B_i}$ resp.) be the set of isolated variables occurring into $\mathbf{X_i}$ ($\mathbf{Y_i}$ resp.), for each $i = 1, \ldots, m$.

By definition of general metaquery under type-2 semantics, there are no isolated variables into $\mathcal{Q}$, then h maps each maximal autoset $\mathbf{B_i}$ of $\mathcal{Q}_2$ into a maximal autoset $\mathbf{A_{j_i}}$, for each $i = 1, \ldots, m$. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be an expansion of $\mathcal{Q}'_1$ and $\mathcal{Q}'_2$ respectively, where $\mathcal{Q}'_1$ and $\mathcal{Q}'_2$ belong to $\bigvee_{1,0} \mathcal{Q}$. Then it is the case that $\mathcal{Q}'_1 =_0 h(\pi(\mathcal{Q}'_2))$. Thus $\mathcal{Q}'_1$ and $\mathcal{Q}'_2$ are two redundant $1,0$-variants of $\mathcal{Q}$, a contradiction.

Now we prove that $\mathcal{M} \supseteq \bigvee^{\equiv}_{2,0} \mathcal{Q}$. We show that for each $\mathcal{Q}_1 \in \bigvee_{2,0} \mathcal{Q}$ there exists $\mathcal{Q}_2 \in \mathcal{M}$ such that $\mathcal{Q}_1 \equiv_0 \mathcal{Q}_2$. Let $\mathcal{Q}_1$ be an expansion of $\mathcal{Q}'_1$, with $\mathcal{Q}'_1 \in \bigvee_{1,0} \mathcal{Q}$, and let $\mathcal{Q}'_2$ be the unique $1,0$-variant of $\mathcal{Q}$ such that $\mathcal{Q}'_2 \in \bigvee^{\equiv}_{1,0} \mathcal{Q}$ and $\mathcal{Q}'_1 \equiv_0 \mathcal{Q}'_2$. Then, by definition of $\mathcal{M}$, it there exists an expansion $\mathcal{Q}_2$ of $\mathcal{Q}'_2$ such that $\mathcal{Q}_2 \in \mathcal{M}$ and $\mathcal{Q}_1 \equiv_0 \mathcal{Q}_2$. □

Proposition 6.14 *Let b the maximum arity of a relation into the database, let m be the number of literal schemes occurring into the metaquery, and let a_i, $i = 1, \ldots, m$, be the arity of the i-th literal scheme of the metaquery. Then*

$$\left| \bigvee^{\equiv}_{1,0} \mathcal{Q} \right| \cdot \prod_{i=1}^{m} \left[\sum_{j=a_i}^{b} \binom{j}{a_i} \right]$$

is an upper bound to the cardinality of the subset of $\bigvee^{\equiv}_{2,0} \mathcal{Q}$ including relational patterns of arity at most b.

Proof. This property follows immediately from Theorem 6.13. □

For example, consider the metaquery $\mathcal{Q} = P(X,Y) \leftarrow Q(X,Y,Z), R(Z)$, and let $b = 3$. As in this case $\bigvee^{\equiv}_{1,0} \mathcal{Q} = \bigvee^{=}_{1,0} \mathcal{Q}$, then the cardinality is

$$6 \cdot \left[\binom{2}{2} + \binom{3}{2}\right] \cdot \left[\binom{3}{3}\right] \cdot \left[\binom{1}{1} + \binom{2}{1} + \binom{3}{1}\right] =$$
$$6 \cdot (1+3) \cdot 1 \cdot (1+2+3) = 6 \cdot 4 \cdot 6 = 144$$

and the set consists in the metaqueries $L_1 \leftarrow L_2, L_3$ where L_1, L_2 and L_3 are one of the following relational patterns:

L_1	L_2	L_3
	$Q(X,Y,Z)$	$R(Z)$
$P(X,Y)$	$Q(X,Z,Y)$	$R(B,Z)$
$P(A,X,Y)$	$Q(Y,X,Z)$	$R(Z,B)$
$P(X,A,Y)$	$Q(Y,Z,X)$	$R(B,C,Z)$
$P(X,Y,A)$	$Q(Z,X,Y)$	$R(B,Z,C)$
	$Q(Z,Y,X)$	$R(Z,B,C)$

while the subset of $\bigvee_{2,0} \mathcal{Q}$ containing relation patterns of arity at most 3 includes

$$(2! + 3!) \times 3! \times (1! + 2! + 3!) = 8 \times 6 \times 9 = 432$$

metaqueries.

7. The algorithm

In this section we present the algorithm *Instantiation-Stage* that enumerates all the non redundant consistent instantiations of a metaquery. This algorithm solves the *instantiation stage* of the process of answering a metaquery, while for the subsequent *filtration stage*, algorithms presented in [7,8] can be adopted. We note that in [7,8] an algorithm is presented, based on CSP techniques, for the filtration stage that computes all the consistent instantiations of a metaquery, thus also the redundant ones.

The algorithm *Instantiation-Stage* exploits Theorem 4.20 in order to avoid the computation of redundant instantiations. Indeed, by Theorem 4.20 the instantiations of two non redundant variants of a general metaquery are always non redundant. Thus, the algorithm computes in the first place the set of all the non redundant $1,0$-variants of the input metaquery, and then computes all the instantiations of each variant (under the type-0 semantics). First, enumeration of renaming is efficiently avoided using the algorithm described in Theorem 6.10. Then, in order to discard redundant $1,0$-variants, an algorithm for graph isomorphism is used, exploiting Theorem 5.2 and Corollary 5.10. The instantiation of the non redundant $1,0$-variants, obtained as explained above, is done using algorithms for the CSP problem. Theorem 6.13, showing that the non redundant $2,0$-variants of a metaquery are efficiently computable from the set of its non redundant $1,0$-variants, is exploited here. Indeed, during the instantiation, each literal scheme $P(\mathbf{X})$ is instantiated to an atom $p(\mathbf{Y})$ provided that $\mathbf{Y}$ is an expansion of $\mathbf{X}$ (see Theorem 6.13 for further details), thus efficiently avoiding the enumeration of redundant $2,0$-variants. Computed instantiations are then passed to the subsequent filtration stage.

Next we comment on the pseudo-code of the algorithm *Instantiation-Stage* and of the functions *Minimize* and *Instantiates* (see Figures 5, 6, and 7). Figure 5 shows the algorithm, which receives in input a general metaquery $\mathcal{Q}$, a database scheme DBS, and the type of semantics $T \in \{1,2\}$.

First, *Instantiation-Stage* determines the autosets of $\mathcal{Q}$ using an algorithm for the modular decomposition of a bipartite graph (see Theorem 5.14).

Then, it decides if $\mathcal{Q}$ is $T,0$-self redundant (see Theorem 6.9). In any case, the set $\bigvee^{\equiv}_{1,0}\mathcal{Q}$ is calculated as in Theorem 6.10. If $\mathcal{Q}$ is $T,0$-self redundant then the set $\bigvee^{\equiv}_{1,0}\mathcal{Q}$ is minimized w.r.t. the redundance between variants, calling the function *Minimize*, reported in Figure 6, which exploits the results of Theorem 5.2 and Corollary 5.10.

The function *Instantiate*, reported in Figure 7, instantiates the metaqueries of the set $\mathcal{M}$. This function builds a sequence $\sigma_0,\sigma_1,\ldots,\sigma_m$ of *partial instantiations* of $\mathcal{Q}$, where each σ_i is an instantiation defined on a subset of i literal schemes of the metaquery $\mathcal{Q}$. In particular, we say that σ_i is *consistent* if the instantiated part of the metaquery $\sigma_i(\mathcal{Q})$ is consistent w.r.t. the database schema DBS. The function *Instantiate* can be implemented using CSP techniques [11]. For $T = 1$, *Instantiate* searches for all the consistent type-0 instantiations of $\mathcal{Q}$. For $T = 2$, each literal scheme $P(\mathbf{X})$ is mapped into an atom $p(\mathbf{Y})$ provided that $\mathbf{Y}$ is an expansion of $\mathbf{X}$. That is, Theorem 6.13 is exploited in order to efficiently enumerate all the metaqueries in the set $\bigvee^{\equiv}_{2,0}$.

Finally, we note that, some variants of the input metaquery $\mathcal{Q}$ could admit type-0 instantiations that are redundant. For example, consider the metaquery $\mathcal{Q} = P(X) \leftarrow Q(X,Y), R(Y,X)$ and its $1,0$-variant $\mathcal{Q}' = P(X) \leftarrow Q(X,Y), R(X,Y)$. The instantiations σ_1 and σ_2 of $\mathcal{Q}'$ such that $\sigma_1(\mathcal{Q}') = a(X) \leftarrow b(X,Y), c(X,Y)$ and $\sigma_2(\mathcal{Q}') = a(X) \leftarrow c(X,Y), b(X,Y)$ are clearly redundant. On the contrary, the $1,0$-variant $\mathcal{Q}'' = P(X) \leftarrow Q(Y,X), R(X,Y)$ of $\mathcal{Q}$ has no redundant type-0 instantiations. Thus, in order to discard these redundant instantiations, if $\mathcal{Q}'$ is self redundant under type-0 semantics, also the set of its instantiations must be minimized by using again the function *Minimize* (see Figure 5, step 4, the function ν there employed is defined in Theorem 5.3).

The following theorem proves correctness of the algorithm *Instantiation-Stage*.

Theorem 7.1 *Given a general metaquery $\mathcal{Q}$ and $T \in \{0,1,2\}$, the set of instantiations Σ^{alg}, computed by the algorithm Instantiation-Stage, is the smallest subset of $\Sigma_c(\mathcal{Q}, DBS, T)$ such that, for each database DB on the schema DBS, and, for each index $I \in \mathbf{I}$ and threshold $k_I \in (0,1]$, it holds that*

$$\Sigma^{\mathrm{alg}} \supseteq \widehat{\Sigma}(\mathcal{Q}, DB, I, k_I, T).$$

Proof. It follows from the description above done, that the algorithm *Instantiation-Stage* computes the set

$$\Sigma^{\mathrm{alg}} = \bigcup_{\mathcal{Q}' \in \bigvee^{\equiv} \mathcal{Q}} \widehat{\Sigma}_c(\mathcal{Q}', DBS, 0)$$

where $\widehat{\Sigma}_c(\mathcal{Q}, DBS, T)$ denotes the smallest subset of $\Sigma_c(\mathcal{Q}, DBS, T)$ such that $\Sigma_c(\mathcal{Q}, DBS, T) \equiv_T \widehat{\Sigma}_c(\mathcal{Q}, DBS, T)$.

W.l.o.g. we assume that $I(\sigma(\mathcal{Q})) > 0$ only if σ is a consistent instantiation of $\mathcal{Q}$[3], i.e. that $\widehat{\Sigma}(\mathcal{Q}, DBS, T) \supseteq \widehat{\Sigma}(\mathcal{Q}, DB, I, k_I, T)$. Thus, by Theorem 4.20, $\Sigma^{\mathrm{alg}} \supseteq \widehat{\Sigma}(\mathcal{Q}, DB, I, k_I, T)$.

It remains to prove that there exists a database DB^{alg} on the schema DBS, an index I, and a threshold k_I, such that $\Sigma^{\mathrm{alg}} = \widehat{\Sigma}(\mathcal{Q}, DB^{\mathrm{alg}}, I, k_I, T)$. The database DB^{alg} can be obtained by "grounding" the set Σ^{alg}, as described in the following. For each instantiation $\sigma \in \Sigma^{\mathrm{alg}}$, let g^{σ} denote a bijection defined on the set of variables occurring into $\sigma(\mathcal{Q})$, such that, for each variable V of $\sigma(\mathcal{Q})$, $g^{\sigma}(V)$ is a new constant associated to V. Let L be a literal scheme of $\mathcal{Q}$, then $g^{\sigma}(L)$ denotes the ground atom obtained from L by substituting each ordinary and predicate variable V of L with $g^{\sigma}(V)$. Then

$$DB^{\mathrm{alg}} = \bigcup_{\sigma \in \Sigma^{\mathrm{alg}}} \{g^{\sigma}(L) \mid L \text{ occurs into } \sigma(\mathcal{Q})\}.$$

[3]Note that, if this assumption does not hold, then Theorem is still valid for $\Sigma^{\mathrm{alg}} \supseteq \Sigma_c(\mathcal{Q}, DBS, T) \cap \widehat{\Sigma}(\mathcal{Q}, DB, I, k_I, T)$.

```
Instantiation-Stage($\mathcal{Q}, DBS, T$)
1. determine the autosets of $\mathcal{Q}$
2. if $\mathcal{Q}$ is not $T$, 0-self redundant
   then set $\mathcal{M} := \bigvee^{=}_{1,0} \mathcal{Q}$
   else set $\mathcal{M} := Minimize(\bigvee^{=}_{1,0} \mathcal{Q}, 0)$
3. $\Sigma := \emptyset$
4. for each $\mathcal{Q}' \in \mathcal{M}$,
   if $\mathcal{Q}'$ is self redundant under type-0 semantics then set
   $\Sigma := \Sigma \cup \{\sigma \mid \nu(\sigma(\mathcal{Q}')) \in Minimize(\{\nu(\sigma(\mathcal{Q}')) \mid \sigma \in Instantiate(\mathcal{Q}', DBS, T)\})\}$
   else set $\Sigma := \Sigma \cup Instantiate(\mathcal{Q}', DBS, T)$
5. return $\Sigma$ and exit
```

Fig. 5. The algorithm *Instantiation-Stage*

```
$Minimize(\mathcal{M}, T)$
1. set $\Psi := \emptyset$
2. if $\mathcal{M} = \emptyset$ then go to 6
3. take a metaquery $\mathcal{Q}$ belonging to $\mathcal{M}$ and set $\mathcal{M} := \mathcal{M} - \{\mathcal{Q}\}$
4. if there not exists $\langle \mathcal{Q}', \mathcal{G}' \rangle$ in $\Psi$ such that $\mathcal{G}'$ is isomorphic to $contr\mathcal{G}_T(\mathcal{Q})$, then set $\Psi := \Psi \cup \{\langle \mathcal{Q}, contr\mathcal{G}_T(\mathcal{Q}) \rangle\}$
5. go to 2
6. $\widehat{\mathcal{M}} := \emptyset$
7. for each $\langle \mathcal{Q}', \mathcal{G}' \rangle$ in $\Psi$, set $\widehat{\mathcal{M}} := \widehat{\mathcal{M}} \cup \{\mathcal{Q}'\}$
8. return $\widehat{\mathcal{M}}$ and exit
```

Fig. 6. The function $Minimize$

```
$Instantiate(\mathcal{Q}, DBS, T)$
1. set $\sigma_0 := \emptyset$, and $i := 1$
2. $(T = 1)$ if exist a literal scheme $P(\mathbf{X})$ of $\mathcal{Q}$ not in the domain of $\sigma_{i-1}$, and a relation scheme, with name $p$, in $DBS$ such that $\sigma_{i-1} \cup \sigma'$ is consistent, where $\sigma' = \{\langle P(\mathbf{X}), p(\mathbf{X}) \rangle\}$, set $\sigma_i := \sigma_{i-1} \cup \sigma'$ and $i := i + 1$
   $(T = 2)$ if exist a literal scheme $P(\mathbf{X})$ of $\mathcal{Q}$ not in the domain of $\sigma_{i-1}$, a relation scheme, with name $p$, in $DBS$ and an expansion $\mathbf{Y}$ of $\mathbf{X}$ such that $\sigma_{i-1} \cup \sigma'$ is consistent, where $\sigma' = \{\langle P(\mathbf{X}), p(\mathbf{Y}) \rangle\}$, set $\sigma_i := \sigma_{i-1} \cup \sigma'$ and $i := i + 1$
3. if $i < n$ then set $i := i + 1$ and goto 2 else return $\sigma_n$ and exit
```

Fig. 7. The function $Instantiate$

By construction, database DB^{alg} is such that, for each $\sigma \in \Sigma^{\text{alg}}$, $cnf(\sigma(\mathcal{Q})) > 0$ on DB^{alg}, and thus $\Sigma^{\text{alg}} = \widehat{\Sigma}(\mathcal{Q}, DB^{\text{alg}}, cnf, 0, T)$. □

Before concluding this section, we discuss how to use the algorithm *Instantiation-Stage* when a metaquery contains atoms. Given a metaquery $\mathcal{Q}$, let $gen(\mathcal{Q})$ denote the metaquery obtained from $\mathcal{Q}$ by removing the atoms occurring into $\mathcal{Q}$[4]. Then, in order to compute the consistent instantiations of $\mathcal{Q}$, the algorithm *Instantiation-Stage* is executed on the metaquery $gen(\mathcal{Q})$. Types associated to variables occurring into atoms of $\mathcal{Q}$ are also passed to the algorithm *Instantiation-Stage* and exploited by the function *Instantiate* in order to speed-up building of consistent instantiations.

For example, let $\mathcal{Q} = P(X,Y) \leftarrow Q(X,Y), r(Y)$, then $gen(\mathcal{Q}) = P(X,Y) \leftarrow Q(X,Y)$, while the variable Y is constrained to assume the type of the unique attribute of the relation r.

8. Conclusions

This work deals with the problem of determining all the consistent instantiations of a metaquery. In particular, it is presented an algorithm for the *instantiation stage* of the process of answering a metaquery. In order to reduce the computational cost associated to the subsequent *filtration stage*, the algorithm exploits the here introduced notion of redundancy between instantiations to filter out in the first place those redundant instantiations. A number of results concerning this property are presented, and then exploited in an algorithm that outputs the set of instantiations minimal w.r.t. the property of redundancy between instantiations. Algorithms for the instantiation stage presented in literature compute all the instantiations of a metaquery, thus also the redundant ones. The computational complexity analysis of the problem of computing the number of instantiations of a metaquery – that is strictly related to the task of enumerating consistent instantiations, is also dealt with into account, showing that this problem is #P-hard.

Acknowledgements. The author would like to thank Luigi Palopoli for his encouragement and precious comments, and the anonymous referees for their useful suggestions.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms.* Addison Wesley, 1974.

[4]Note that, $gen(\mathcal{Q})$ might be a metaquery with an empty head. This does not affect the execution of the algorithm.

[2] F. Angiulli, R. Ben-Eliyahu-Zohary, G. Ianni, and L. Palopoli. Computational properties of metaquerying problems. In *Proc. of the 9th ACM SIGMOD-SIGACT-SIGART Symp. on Princ. of Database Systems (PODS)*, pages 237–244, Dallas, Texas, 2000.

[3] F. Angiulli, R. Ben-Eliyahu-Zohary, G. Ianni, and L. Palopoli. Computational properties of metaquerying problems. *ACM Transactions on Computational Logic*, 4(3):149–180, 2003.

[4] L. Babai. Monte carlo algorithms in graph isomorphism testing. Technical Report TR-79-10, Dép. Math. et Stat., Univ. de Montréal, 1979.

[5] L. Babai and E.M. Luks. Canonical labeling of graphs. In ACM, editor, *Proceedings of the fifteenth annual ACM Symposium on Theory of Computing, Boston, Massachusetts, April 25–27, 1983*, pages 171–183, New York, NY, USA, 1983. ACM Press.

[6] R. Ben-Eliyahu-Zohary, C. Domshlak, E. Gudes, N. Liusternik, A. Meisels, T. Rosen, and S.E. Shimony. Fleximine - a flexible platform for kdd research and application development. *Annals of Mathematics and Artificial Intelligence*, 39(1-2):175–204, 2003.

[7] R. Ben-Eliyahu-Zohary and E. Gudes. Towards efficient metaquerying. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 800–805, Stockholm, Sweden, 1999.

[8] R. Ben-Eliyahu-Zohary, E. Gudes, and G. Ianni. Metaqueries: Semantics, complexity, and efficient algorithms. *Artificial Intelligence*, 149(1):61–87, 2003.

[9] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *ACM Symp. on Theory of Computing*, pages 77–90, 1977.

[10] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. *Lecture Notes in Computer Science*, 787:68–84, 1994.

[11] R. Dechter. Constraint networks. In Stuart C. Shapiro, editor. *Encyclopedia of Artificial Intelligence*, pages 276–285. John Wiley, 1992.

[12] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.

[13] M. R. Garey and D. S. Johnson. *Computer and Intractability*. W. H. Freeman and Company, New York, 1979.

[14] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187, Los Angeles, Ca., USA, October 1986. IEEE Computer Society Press.

[15] J.E. Hopcroft and J. Wong. Linear time algorithm for isomorphism of planar graphs. In *ACM Symp. on Theory of Computing*, pages 172–184, 1974.

[16] B. Leng and W. Shen. A metapattern-based automated discovery loop for integrated data mining - unsupervised learning of relational patterns. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):898–910, 1996.

[17] E.M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Science*, 25(1):42–65, 1982.

[18] R.M. McConnell and K.P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 536–545, Arlington, VA, January 1994. ACM Press.

[19] B.D. McKay. nauty user's guide (version 1.5). Technical Report TR-CS-90-02, Department of Computer Science, The Australian National University, Canberra 0200 ACT, Australia, March 1990.

[20] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[21] B.G. Mitbander W. Shen, K. Ong and C. Zaniolo. *Metaqueries for Data Mining*, chapter 15, pages 375–398. In Fayyad et al. [12], 1996.