International Journal of Approximate Reasoning ••• (••••) •••–•••

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

First-order under-approximations of consistent query answers ☆

Floris Geerts [a], Fabian Pijcke [b], Jef Wijsen [b,*]

[a] University of Antwerp, Dept. of Mathematics and Computer Science, Middelheimlaan 1, B-2020 Antwerpen, Belgium
[b] Université de Mons, 20 Place du Parc, B-7000 Mons, Belgium

ARTICLE INFO

Article history:
Received 19 February 2016
Received in revised form 23 September 2016
Accepted 19 October 2016
Available online xxxx

Keywords:
Conjunctive queries
Consistent query answering
Primary key

ABSTRACT

Consistent Query Answering (CQA) is a principled approach for answering queries on inconsistent databases. The consistent answer to a query q on an inconsistent database **db** is the intersection of the answers to q on all repairs, where a repair is any consistent database that is maximally close to **db**. Unfortunately, computing consistent answers under primary key constraints has already exponential data complexity for very simple conjunctive queries, and is therefore completely impracticable.

In this paper, we propose a new framework for divulging an inconsistent database to end users, which adopts two postulates. The first postulate complies with CQA and states that inconsistencies should never be divulged to end users. Therefore, end users should only get consistent query answers. The second postulate states that only those queries can be answered whose consistent answers can be obtained with low data complexity (i.e., by a polynomial-time algorithm or even a first-order logic query). User queries that exhibit a higher data complexity will be rejected.

A significant problem in this framework is as follows: given a rejected query, find other queries, called under-approximations, that are accepted and whose consistent answers are contained in those of the rejected query. We provide solutions to this problem for the special case where the constraints are primary keys and the queries are self-join-free conjunctive queries.

1. Introduction

Inconsistent, incomplete and uncertain data is widespread in the internet and social media era. This has given rise to a new paradigm for query answering, called *Consistent Query Answering* (CQA) [2]. This paradigm starts with the notion of *repair*, which is a new consistent database that minimally differs from the original inconsistent database. In general, an inconsistent database can have many repairs. In this respect, database repairing is different from data cleaning which aims at a unique cleaned database.

In this paper, we assume that the only constraints are primary keys, one per relation. A repair of an inconsistent database **db** is a maximal subset of **db** that satisfies all primary key constraints. Primary keys will be underlined. For example, the database of Fig. 1 stores ages and cities of residence of male and female persons. For simplicity, assume that persons have

☆ A short version of this article was published in the conference proceedings of SUM 2015 [1].
* Corresponding author.

E-mail addresses: floris.geerts@ua.ac.be (F. Geerts), fabian.pijcke@umons.ac.be (F. Pijcke), jef.wijsen@umons.ac.be (J. Wijsen).

http://dx.doi.org/10.1016/j.ijar.2016.10.005
0888-613X/

M	$\underline{N}$	A	C
	Ed	48	Mons
	Ed	48	Paris
	Dirk	29	Mons

F	$\underline{N}$	A	C
	An	37	Mons
	Iris	37	Paris

Fig. 1. Example database with primary key violations.

unique names (attribute N). Every person has exactly one age (attribute A) and city (attribute C). However, distinct tuples may agree on the primary key N, because there can be uncertainty about ages and cities. In the database of Fig. 1, there is uncertainty about the city of Ed (it can be Mons or Paris). The database can be repaired in two ways: delete either $M(\underline{\mathrm{Ed}}, 48, \mathrm{Mons})$ or $M(\underline{\mathrm{Ed}}, 48, \mathrm{Paris})$. A maximal set of tuples that agree on their primary key will be called a *block*; in Fig. 1, blocks are separated by dashed lines.

When database repairing results in multiple repairs, CQA shifts from standard semantics to certainty semantics. Given a query, the *consistent answer* (also called *certain answer*) is defined as the intersection of the answers on all repairs. That is, for a query q on an inconsistent database **db**, CQA replaces the standard query answer $q(\mathbf{db})$ with the consistent answer, defined by the following intersection:

$$\bigcap \left\{ q(\mathbf{r}) \mid \mathbf{r} \text{ is a repair of } \mathbf{db} \right\}. \tag{1}$$

Thus, the certainty semantics exclusively returns answers that hold true in every repair. Given a query q, we will denote by $\lfloor q \rfloor$ the query that maps a database to the consistent answer defined by (1).

A practical obstacle to CQA is that the shift to certainty semantics involves a significant increase in complexity. When we refer to complexity in this paper, we mean data complexity, i.e., the complexity in terms of the size of the database (for a fixed query) [3, p. 422]. It is known for long [4] that there exist conjunctive queries q that join two relations such that the data complexity of $\lfloor q \rfloor$ is already **coNP**-hard. If this happens, CQA is completely impracticable.

This paper investigates ways to circumvent the high data complexity of CQA in a realistic setting, which is based on the following assumptions:

- If a query returns an answer to a user, then every tuple in that answer should belong to the consistent answer. In Libkin's terminology [5], query answers must not contain *false positives*, i.e., tuples that do not belong to the consistent answer.
- The only queries that can be executed in practice are those with data complexity in **FP** or, even better, in **FO**. Here, **FO** refers to the descriptive complexity class that captures all queries expressible in relational calculus [6]. **FP** is the class of function problems solvable in polynomial time.

Therefore, if the data complexity of a query $\lfloor q \rfloor$ is not in **FP**, then the best we can go for is an approximation without false positives (also called under-approximation), computable in polynomial time. The term *strategy* will be used for queries that compute such approximations. Intuitively, a strategy can be regarded as a two-step process in which one starts by issuing a number of well-behaved queries $\lfloor q_i \rfloor$, for $i \in \{1, \ldots, \ell\}$, which can then be subject to a post-processing step. In this paper, well-behaved queries are those that are accepted by a query interface, e.g., self-join-free conjunctive queries q_i such that $\lfloor q_i \rfloor$ is in **FO**, and post-processing is formalized as queries built-up from the $\lfloor q_i \rfloor$'s.

We next illustrate our setting by an example. Consider the following scenario with two persons, called *Bob* and *Alice*. The person called *Bob* owns a database that is publicly accessible only via a query interface which restricts the syntax of the queries that can be asked. Our main results concern the case where the interface is restricted to self-join-free conjunctive queries. The database schema including all primary key constraints is publicly available. However, *Bob* is aware that his database contains many mistakes which should not be divulged. Therefore, whenever some end user asks a query q, *Bob* will actually execute the query $\lfloor q \rfloor$. That is, end users will get exclusively consistent answers. But, for feasibility reasons, *Bob* will reject any query q for which the data complexity of $\lfloor q \rfloor$ is too high. In this paper, we assume that *Bob* considers that data complexity is too high when it is not in **FO**. The person called *Alice* interrogates *Bob*'s database, and she will be happy to get exclusively consistent answers. Unfortunately, her query q will be rejected by *Bob* if the data complexity of $\lfloor q \rfloor$ is too high (i.e., not in **FO**). If this happens, *Alice* has to change strategy. Instead of asking q, she can ask a finite number of queries $q_1, q_2, \ldots, q_\ell$ such that for every $i \in \{1, \ldots, \ell\}$, the data complexity of $\lfloor q_i \rfloor$ is in **FO**, and hence the query q_i will be accepted by *Bob*. No restriction is imposed on the number ℓ of queries that can be asked. The best *Alice* can hope for is that she can compute herself the answer to $\lfloor q \rfloor$ (or even to q) from *Bob*'s answers to $\lfloor q_1 \rfloor, \ldots, \lfloor q_\ell \rfloor$ by means of some post-processing. The question addressed in this paper is: Given that *Alice* wants to answer q, what queries should she ask to *Bob*?

Here is a concrete example. Assume *Bob* owns the database of Fig. 1. Interested in stable couples,[1] *Alice* submits the query q_1 which asks "Get pairs of ages of men and women living in the same city":

$$q_1 = \left\{ y, w \mid \exists x \exists u \exists z \left(M(\underline{x}, y, z) \wedge F(\underline{u}, w, z) \right) \right\}.$$

[1] According to [7], marital stability is higher when the wife is 5+ years younger than her husband.

Please cite this article in press as: F. Geerts et al., First-order under-approximations of consistent query answers, Int. J. Approx. Reason. (2016), http://dx.doi.org/10.1016/j.ijar.2016.10.005

The consistent answer is $\{(48,37),(29,37)\}$. However, the query $\lfloor q_1 \rfloor$ that returns the consistent answer is known to have **coNP**-hard data complexity [8]. Therefore, *Bob* will reject q_1. *Alice* changes strategy and submits the query q_2 which asks "Get pairs of ages and city of men and women living in the same city":

$$q_2 = \{y, w, z \mid \exists x \exists u \left(M(\underline{x}, y, z) \wedge F(\underline{u}, w, z)\right)\}. \tag{2}$$

Since the data complexity of $\lfloor q_2 \rfloor$ is known to be in **FO** [8], *Bob* will execute $\lfloor q_2 \rfloor$. The query q_2 returns {(29, 37, Mons), (48, 37, Mons)} on one repair, and {(29, 37, Mons), (48, 37, Paris)} on the other repair, so the consistent answer is $\{(29, 37, \text{Mons})\}$. This in turn allows *Alice* to derive a consistent answer to the original query: since $(29, 37, \text{Mons})$ belongs to the answer to $\lfloor q_2 \rfloor$, it is correct to conclude that $(29, 37)$ belongs to the answer to $\lfloor q_1 \rfloor$. An interesting question is whether *Alice* has a better strategy that divulges even more answers to $\lfloor q_1 \rfloor$.

The technical contributions of this paper are as follows. We first show that the following problem is undecidable: Given a relational calculus query q, is $\lfloor q \rfloor$ in **FO**? In view of this undecidability result, we then limit our attention to strategies that are first-order combinations (using disjunction and existential quantification) of queries $\lfloor q \rfloor$ that are known to be in **FO**. We show how to build optimal strategies under such syntax restrictions.

This paper is organized as follows. Section 2 discusses related work. Section 3 provides some mathematical definitions. Section 4 introduces our new framework for studying consistent query answering under primary key constraints, and introduces the problem OPTSTRATEGY. Intuitively, OPTSTRATEGY asks, given a query q, to find a new query q' that gets the largest subset of consistent answers while still obeying the restrictions imposed by our framework. Section 5 provides ways to solve OPTSTRATEGY in restricted settings. Section 6 studies a novel query containment problem that is intimately related to the simplification of strategies. Finally, Section 7 concludes the paper.

2. Related work

Consistent query answering (CQA) was proposed in [2] as a principled approach to handle data quality problems that arise from violations of integrity constraints. We refer to the textbooks [9] and [10] for comprehensive overviews of these domains.

Fuxman and Miller [11] were the first ones to focus on CQA under the restrictions that consistency is only with respect to primary keys and that queries are self-join-free conjunctive queries. A survey on consistent query answering to conjunctive queries under primary key constraints is given in [12]. Some recent results not covered by this survey can be found in [8,13].

Instead of returning the query answers true in every repair, one could return the query answers true in, e.g., a majority of repairs. This leads to the counting variant of CQA, which has been studied in [14,15]. As observed in [16], the counting variant of CQA under primary key constraints is closely related to query answering in block-independent-disjoint (BID) probabilistic databases [17,18]. Counting the fraction of repairs that satisfy a query has also been studied by Greco et al. [19]. The constraints in that work are functional dependencies, and the repairs are obtained by updates. Greco et al. present an approach for computing approximate probabilistic answers in polynomial time.

In the past, the paradigm of CQA has been implemented in expressive formalisms, such as Disjunctive Logic Programming [20] and Binary Integer Programming (BIP) [21]. In these formalisms, it is relatively easy to express an algorithm that computes consistent answers to conjunctive queries under primary key constraints. The drawback is that these algorithms may, in the worst case, take exponential time in cases where, in theory, certain answers are computable in polynomial time or expressible in first-order logic. In the latter case, the consistent answer can be computed by a single SQL query using standard database technology, including query optimization. In [10, page 38], the author mentions that logic programs for CQA cannot compete with solutions in first-order logic when they exist. Likewise, in an experimental comparison of EQUIP [21] and ConQuer [22], the authors of the former system found that BIP never outperformed solutions in SQL.

For (unions of) conjunctive queries, under-approximations of consistent answers can be obtained by executing the queries on the intersection of all repairs (instead of intersecting query answers). This is called the *Intersection ABox Repair* (IAR) semantics [23,24]. In our setting, the intersection of all repairs can be computed in **FO**, by selecting from the database all blocks that contain only one tuple.

Our work can also be regarded as querying "consistent views," in the sense that *Bob* returns exclusively consistent answers. It has been observed long ago [25] that consistent views are not closed under relational calculus. In other words, the position of the $\lfloor \cdot \rfloor$ construct in a query does matter. For example, for the database of Fig. 1, the query $\{x \mid \exists y \exists z \lfloor M(\underline{x}, y, z) \rfloor\}$ returns only Dirk, while $\lfloor\{x \mid \exists y \exists z\, M(\underline{x}, y, z)\}\rfloor$ returns both Ed and Dirk. Bertossi and Li [26] have used views to protect the secrecy of data in a database. In our setting, the query answers that are to be hidden from end users are those that are not true in every repair.

3. Preliminaries

We assume disjoint sets of *variables* and *constants*. A *term* is a constant or a variable. If $\vec{t}$ is a sequence of terms, then $\mathsf{vars}(\vec{t})$ denotes the set of variables that occur in $\vec{t}$. A *valuation* over a set U of variables is a total mapping θ from U to the set of constants. At several places, it is implicitly understood that such a valuation θ is extended to be the identity on constants and on variables not in U.

Atoms and key-equal facts. Each *relation name* R of arity n, $n \geq 1$, has a unique *primary key* which is a set $\{1, 2, \dots, k\}$ where $1 \leq k \leq n$. We say that R has *signature* $[n, k]$ if R has arity n and primary key $\{1, 2, \dots, k\}$. We say that R is *all-key* if $n = k$. For all positive integers n, k such that $1 \leq k \leq n$, we assume denumerably many relation names with signature $[n, k]$.

If R is a relation name with signature $[n, k]$, then $R(t_1, \dots, t_n)$ is called an *R-atom* (or simply atom), where each t_i is a term ($1 \leq i \leq n$). Such an atom is commonly written as $R(\underline{\vec{x}}, \vec{y})$ where the primary-key value $\vec{x} = t_1, \dots, t_k$ is underlined and $\vec{y} = t_{k+1}, \dots, t_n$. An *$R$-fact* (or simply fact) is an R-atom in which no variable occurs. Two facts $R_1(\underline{\vec{a}_1}, \vec{b}_1)$ and $R_2(\underline{\vec{a}_2}, \vec{b}_2)$ are *key-equal* if $R_1 = R_2$ and $\vec{a}_1 = \vec{a}_2$.

We will use letters F, G, H for atoms. For an atom $F = R(\underline{\vec{x}}, \vec{y})$, we denote by $\mathsf{key}(F)$ the set of variables that occur in $\vec{x}$, and by $\mathsf{vars}(F)$ the set of variables that occur in F, that is, $\mathsf{key}(F) = \mathsf{vars}(\vec{x})$ and $\mathsf{vars}(F) = \mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{y})$.

Uncertain databases, blocks, and repairs. A *database schema* is a finite set of relation names. All constructs that follow are defined relative to a fixed database schema.

A *database* is a finite set **db** of facts using only the relation names of the schema. We often refer to databases as "uncertain databases" to stress that such databases can violate primary key constraints.

A *block* of **db** is a maximal set of key-equal facts of **db**. The term R-block refers to a block of R-facts, i.e., facts with relation name R. An uncertain database **db** is *consistent* if no two distinct facts are key-equal (i.e., if every block of **db** is a singleton). A *repair* of **db** is a maximal (with respect to set containment) consistent subset of **db**. We write $\mathsf{rset}(\mathbf{db})$ for the set of repairs of **db**.

Queries and consistent query answering. We assume that the reader is familiar with *relational calculus* [3, Chapter 5] and with the notion of *queries* [27, Definition 2.7]. By **FO**, we denote the descriptive complexity class that contains the queries expressible in relational calculus.

For every m-ary ($m \geq 0$) relational calculus query q, we define $\lfloor q \rfloor$ as the m-ary query that maps every database **db** to $\bigcap \left\{ q(\mathbf{r}) \mid \mathbf{r} \in \mathsf{rset}(\mathbf{db}) \right\}$. Clearly, if **db** is a consistent database, then $\lfloor q \rfloor(\mathbf{db}) = q(\mathbf{db})$.

Given two m-ary queries q_1 and q_2, we say that q_1 is *contained in* q_2, denoted by $q_1 \sqsubseteq q_2$ if for every database **db**, $q_1(\mathbf{db}) \subseteq q_2(\mathbf{db})$. We write $q_1 \sqsubset q_2$, if $q_1 \sqsubseteq q_2$ and $q_2 \not\sqsubseteq q_1$. We say that q_1 and q_2 are *equivalent*, denoted by $q_1 \equiv q_2$, if $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

A 0-ary query is called Boolean. If q is a Boolean query, then q maps any database to either $\{\langle\rangle\}$ or $\{\}$, corresponding to **true** and **false** respectively.

A *conjunctive query* is a relational calculus query of the form $\left\{ \vec{z} \mid \exists \vec{y}\, B \right\}$ where B is a conjunction of atoms. The conjunction B and the query are said to be *self-join-free* if no relation name occurs more than once in B. We write $\mathsf{vars}(B)$ for the set of variables that occur in B. By a slight abuse of notation, we denote by B also the set of conjuncts that occur in B. For example, if $B_1 = R(\vec{x}) \wedge R(\vec{y})$ and $B_2 = R(\vec{x}) \wedge R(\vec{y}) \wedge R(\vec{z})$, then we may write $B_1 \subseteq B_2$. Finally, if q is a self-join-free conjunctive query with an R-atom, then, by an abuse of notation, we write R to mean the R-atom of q.

If q is a conjunctive query, $\vec{x} = \langle x_1, \dots, x_\ell \rangle$ is a sequence of distinct variables in $\mathsf{vars}(q)$, and $\vec{c} = \langle c_1, \dots, c_\ell \rangle$ is a sequence of constants, then we denote by $q_{[\vec{x} \mapsto \vec{c}]}$ the query obtained from q by replacing each occurrence of each x_i with c_i.

Significantly, the following example shows that $\lfloor q \rfloor$ may not be expressible in relational calculus, even if q is a self-join-free conjunctive query.

Example 1. Let $q_1 = \left\{ \langle\rangle \mid \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z) \right) \right\}$. The query q_1 is self-join-free conjunctive. It follows from [8] that $\lfloor q_1 \rfloor$ is not in **FO** (i.e., not expressible in relational calculus).

Let $q_2 = \left\{ \langle\rangle \mid \exists x \exists y \left(R(\underline{x}, y) \wedge S(\underline{y}, b) \right) \right\}$, where b is a constant. Then, $\lfloor q_2 \rfloor$ is equivalent to the following relational calculus query:

$$\exists x \exists y \left(R(\underline{x}, y) \wedge \forall y \left(R(\underline{x}, y) \rightarrow \left(S(\underline{y}, b) \wedge \forall z \left(S(\underline{y}, z) \rightarrow z = b \right) \right) \right) \right). \quad \square$$

4. A framework for divulging inconsistent databases

In this section, we formalize the setting that was described and illustrated in Section 1. The setting is captured by the language called CQAFO, which consists of first-order quantification and Boolean combinations of atomic formulas of the form $\lfloor q \rfloor$, where q is any relational calculus query. The atomic formulas $\lfloor q \rfloor$ capture that the database owner *Bob* only returns consistent answers. Subsequently, the end user *Alice*, who interrogates *Bob*'s database, can do some post-processing on *Bob*'s outputs. In our setting, we assume that *Alice* uses first-order quantification and Boolean combinations of *Bob*'s consistent answers to the atomic formulas $\lfloor q \rfloor$.

Example 2. The scenario in Section 1 is captured by the CQAFO query

$$\left\{ y, w \mid \exists Z \left\lfloor \exists x \exists u \left(M(\underline{x}, y, Z) \wedge F(\underline{u}, w, Z) \right) \right\rfloor \right\}.$$

The formula within $\lfloor \cdot \rfloor$ is the query (2). The quantification $\exists Z$ corresponds to *Alice* projecting away the cities column returned by *Bob*. For readability, we will often use upper case letters for variables that are quantified outside the range of $\lfloor \cdot \rfloor$. $\square$

Example 3. The following query allows *Alice* to find the names of men with more than two cities in the database:

$$\{x \mid \lfloor \exists y \exists z\, M(\underline{x}, y, z) \rfloor \wedge \neg \exists Z\, \lfloor \exists y\, M(\underline{x}, y, Z) \rfloor\}.$$

To understand this query, it may be helpful to notice that $\{x, Z \mid \lfloor \exists y\, M(\underline{x}, y, Z) \rfloor\}$ returns tuple (n, c) whenever c is the only city of residence recorded for the person named n. Interestingly, even though *Alice* gets only consistent answers, she can still infer that the database contains inconsistencies. In particular, since the foregoing query returns Ed on the example database of Fig. 1, *Alice* can infer that there is uncertainty about the city of Ed. □

4.1. The language CQAFO

We next describe the syntax and semantics of the language CQAFO used for postprocessing.

Syntax The following are formulas in CQAFO:

- if q is a relational calculus query, then $\lfloor q \rfloor$ is a CQAFO formula with the same free variables as q;
- if φ_1 and φ_2 are CQAFO formulas, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, and $\neg\varphi_1$ are CQAFO formulas;
- if φ is a CQAFO formula, then $\exists Y\, \varphi$ and $\forall Y\, \varphi$ are CQAFO formulas.

If φ is a CQAFO formula, then $\mathsf{free}(\varphi)$ denotes the set of free variables of φ (i.e., the variables not bound by a quantifier). If $\vec{x}$ is a tuple containing the free variables of φ, we write $\varphi(\vec{x})$.

A CQAFO query is an expression of the form $\{\vec{t} \mid \varphi\}$, where $\vec{t}$ is a sequence of terms containing each variable of $\mathsf{free}(\varphi)$. If $\vec{t}$ contains no constants and no double occurrences of the same variable, then such query is also denoted $\varphi(\vec{t})$.

Semantics Let **db** be an uncertain database. Let $\varphi(\vec{x})$ be a CQAFO formula, and $\vec{a}$ be a sequence of constants (of same length as $\vec{x}$). We inductively define $\mathbf{db} \models \varphi(\vec{a})$.

- If $\varphi(\vec{x}) = \lfloor q(\vec{x}) \rfloor$ for some relational calculus query $q(\vec{x})$, then $\mathbf{db} \models \varphi(\vec{a})$ if for every repair $\mathbf{r}$ of **db**, $\mathbf{r} \models q(\vec{a})$[2];
- $\mathbf{db} \models \neg\varphi(\vec{a})$ if $\mathbf{db} \not\models \varphi(\vec{a})$;
- $\mathbf{db} \models \varphi_1 \wedge \varphi_2$ if $\mathbf{db} \models \varphi_1$ and $\mathbf{db} \models \varphi_2$;
- $\mathbf{db} \models \varphi_1 \vee \varphi_2$ if $\mathbf{db} \models \varphi_1$ or $\mathbf{db} \models \varphi_2$;
- if $\psi(\vec{x}) = \exists Y\, \varphi(Y, \vec{x})$, then $\mathbf{db} \models \psi(\vec{a})$ if $\mathbf{db} \models \varphi(a', \vec{a})$ for some a';
- if $\psi(\vec{x}) = \forall Y\, \varphi(Y, \vec{x})$, then $\mathbf{db} \models \psi(\vec{a})$ if $\mathbf{db} \models \varphi(a', \vec{a})$ for all a'.

Let $Q = \{\vec{t} \mid \varphi(\vec{x})\}$ be a CQAFO query. The answer $Q(\mathbf{db})$ is the smallest set containing $\theta(\vec{t})$ for every valuation θ over $\mathsf{vars}(\vec{t})$ such that for some $\vec{a}$, $\theta(\vec{x}) = \vec{a}$ and $\mathbf{db} \models \varphi(\vec{a})$. By definition, we have $\mathsf{vars}(\vec{t}) = \mathsf{vars}(\vec{x})$, but $\vec{t}$, unlike $\vec{x}$, can contain constants and multiple occurrences of the same variable. If $\vec{t}$ contains no variables, then Q is Boolean.

Domain independence is a desirable property of queries that emerges in CQAFO in the same way as in relational calculus [3, p. 79]. For example, consider the CQAFO query $Q_0 = \{x \mid \lfloor \exists y \exists z\, M(\underline{x}, y, z) \rfloor \vee \lfloor F(\underline{\text{'Iris'}}, \text{'37'}, \text{'Paris'}) \rfloor\}$ on the example database of Fig. 1. Since $F(\underline{\text{'Iris'}}, \text{'37'}, \text{'Paris'})$ holds true in every repair, the query is true for any value of x. The query Q_0 is thus not domain independent. Nevertheless, domain independence will not be an issue in this paper, because we will only deal with syntactic fragments of CQAFO that guarantee domain independence.

4.2. Restrictions on data complexity

The language CQAFO of Section 4.1 captures our first postulate which states that the database owner *Bob* returns exclusively consistent answers. But we do not prohibit that end user *Alice* does some post-processing on *Bob*'s answers. In this section, we will add our second postulate which states that *Bob* rejects queries q if the data complexity of $\lfloor q \rfloor$ is not in **FO**. Unfortunately, *Bob* has to face the following undecidability result.

Theorem 1. *The following problem is undecidable. Given a relational calculus query q, is $\lfloor q \rfloor$ in* **FO**?

Proof. Let $q_1 = \{\langle\rangle \mid \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z) \wedge \varphi\right)\}$ where φ is a closed relational calculus formula, i.e., a formula with no free variables, such that all relation names in φ are all-key. Observe that this implies that the relation names in φ are distinct from R and S. We show hereinafter that $\lfloor q \rfloor$ is in **FO** if and only if φ is unsatisfiable. The desired result then follows by [3, Theorem 6.3.1], which states that (finite) satisfiability of relational calculus queries is undecidable.

Obviously, if φ is unsatisfiable, then $\lfloor q_1 \rfloor \equiv \mathbf{false}$, and hence $\lfloor q_1 \rfloor$ is in **FO**.

We show next that if φ is satisfiable, then $\lfloor q_1 \rfloor$ is not in **FO**. Assume that φ is satisfiable. Let $q_0 = \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z)\right)$ and consider the following two problems:

[2] $\mathbf{r} \models q(\vec{a})$ is defined in the standard way.

- CERTAIN0: Given a database **db**, determine whether every repair of **db** satisfies q_0.
- CERTAIN1: Given a database **db**, determine whether every repair of **db** satisfies q_1.

We show a polynomial-time many-one reduction from CERTAIN0 to CERTAIN1. Let $\mathbf{db}_0$ be a database that is input to CERTAIN0. Let **S** be the database schema that contains the relation names occurring in φ. An algorithm can consider systematically every finite database $\mathbf{db}'$ over **S** and test $\mathbf{db}' \models \varphi$, until a database $\mathbf{db}'$ is found such that $\mathbf{db}' \models \varphi$. The algorithm terminates because φ is satisfiable. Since the computation of $\mathbf{db}'$ does not depend on $\mathbf{db}_0$, it takes $\mathcal{O}(1)$ time. Since all relation names in $\mathbf{db}'$ are all-key, we have that $\mathbf{db}'$ is consistent. Clearly, q_0 is true in every repair of $\mathbf{db}_0$ if and only if q_1 is true in every repair of $\mathbf{db}_0 \cup \mathbf{db}'$. This follows from the fact that the relation names in φ are distinct from R and S. So we have established a polynomial-time many-one reduction from CERTAIN0 to CERTAIN1. Since CERTAIN0 is **coNP**-hard [8], it follows that CERTAIN1 is **coNP**-hard. Since $\mathbf{FO} \subsetneq \mathbf{coNP}$ [6], it follows that CERTAIN1 is not in **FO**. □

By Theorem 1, there exists no algorithm that allows *Bob* to decide whether he has to accept or reject a relational calculus query. In general, little is known about the complexity of $\lfloor q \rfloor$ for relational calculus queries q. One of the stronger known results is the following.

Theorem 2 *([8]). The following problem is decidable in polynomial time. Given a self-join-free conjunctive query q, is $\lfloor q \rfloor$ in* **FO***? Moreover, if $\lfloor q \rfloor$ is in* **FO***, then a relational calculus query equivalent to $\lfloor q \rfloor$ can be effectively constructed.*

In view of Theorems 1 and 2, the following scenario is the best we can go for with the current state of art.

1. The database owner *Bob* only accepts self-join-free conjunctive queries q such that $\lfloor q \rfloor$ is in **FO**. Thus, *Bob* rejects every query that is not self-join-free conjunctive, and rejects a self-join-free conjunctive query q if $\lfloor q \rfloor$ is not in **FO**.
2. As before, *Alice* can do some first-order post-processing on the answers obtained from *Bob*.

Under these restrictions, we focus on the following problem: given that *Alice* wants to answer a self-join-free conjunctive query q on a database owned by *Bob*, develop a *strategy* for *Alice* to get a subset (the greater, the better) of the consistent answer to q. Our framework applies to Boolean queries by representing **true** and **false** by $\{\langle\rangle\}$ and $\{\}$ respectively. A formal definition follows.

4.3. Strategies

Strategies for a query q are defined next as relational calculus queries that can be expressed in CQAFO and that are contained in $\lfloor q \rfloor$.

Definition 1. Let q be a self-join-free conjunctive query. A *strategy for* q is a CQAFO query φ such that $\varphi \sqsubseteq \lfloor q \rfloor$ and for every atomic formula $\lfloor q' \rfloor$ in φ, we have that q' is a self-join-free conjunctive query such that $\lfloor q' \rfloor$ is in **FO**.

A strategy φ for q is *optimal* if for every strategy ψ for q, we have $\psi \sqsubseteq \varphi$. The problem OPTSTRATEGY takes in a self-join-free conjunctive query q and asks to determine an optimal strategy for q. □

Some observations are in place.

- If the input to OPTSTRATEGY is a self-join-free conjunctive q such that $\lfloor q \rfloor$ is in **FO**, then the CQAFO query $\lfloor q \rfloor$ is itself an optimal strategy.
- Every strategy φ is in **FO**, because all atomic formulas $\lfloor q' \rfloor$ are required to be in **FO**. Therefore, if *Alice* wants to answer a query q such that $\lfloor q \rfloor$ is not in **FO**, then there is no strategy φ such that $\varphi \equiv \lfloor q \rfloor$.
- There is no fundamental reason why the input query to OPTSTRATEGY is required to be a self-join-free conjunctive query. However, developing strategies for more expressive queries is left as an open question.

In the remainder of this paper, we will not investigate the problem OPTSTRATEGY in its most general form. Instead, we will confine our investigation to strategies that can be expressed and effectively constructed in a syntactic fragment of CQAFO. We will explain how such strategies can be constructed, but leave open the computational complexity of the construction.

5. How to construct good strategies?

Let q be a self-join-free conjunctive query. In this section, we investigate ways for constructing good (if not optimal) strategies for q of a particular syntax. In Section 5.1, we take the most simple approach: take the union of queries $\lfloor q_i \rfloor$ contained in $\lfloor q \rfloor$, where q_i is self-join-free conjunctive and $\lfloor q_i \rfloor$ is in **FO**. We then show that the strategies obtained in this way cannot be optimal. Therefore, an enhanced approach is developed in Section 5.2.

5.1. Post-processing by unions only

Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\vec{z})$. In this section, we look at strategies of the form

$$\bigcup_{i=1}^{\ell} \lfloor q_i \rfloor, \tag{3}$$

where each q_i is of the form $\{\vec{z}_i \mid \exists \vec{y}_i\, B_i\}$ in which $\vec{z}_i$ has the same length as $\vec{z}$ and B_i is a self-join-free conjunction of atoms. Speaking strictly syntactically, $\lfloor\{\vec{z}_i \mid \exists \vec{y}_i B_i\}\rfloor$ is not a CQAFO query, as it is not of the form $\{\vec{t} \mid \varphi\}$ for some CQAFO formula φ as defined in Section 4.1. However, it can be easily verified that $\lfloor\{\vec{z}_i \mid \exists \vec{y}_i B_i\}\rfloor \equiv \{\vec{z}_i \mid \lfloor \exists \vec{y}_i B_i \rfloor\}$, and the latter query is a CQAFO query.

We use union (with its standard semantics) instead of disjunction to avoid notational difficulties. For example, the union

$$\{x, a \mid \lfloor R(\underline{x}, a)\rfloor\} \cup \{x, y \mid \lfloor S(\underline{x}, y)\rfloor\},$$

where a is a constant, is semantically clear, and is equivalent to

$$\{x, y \mid \lfloor R(\underline{x}, y) \wedge y = a\rfloor \vee \lfloor S(\underline{x}, y)\rfloor\},$$

in which equality is used. It would be wrong to write $\{x, y \mid \lfloor R(\underline{x}, a)\rfloor \vee \lfloor S(\underline{x}, y)\rfloor\}$, an expression that is not domain independent [3, p. 79], because if some fact $R(\underline{c}, a)$ holds true in every repair, then $\lfloor R(\underline{x}, a)\rfloor \vee \lfloor S(\underline{x}, y)\rfloor$ is true when c is assigned to x, no matter what value is assigned to y. On the other hand, a CQAFO formula of the form (3) is domain independent if each $\lfloor q_i \rfloor$ is domain independent.

Furthermore, a formula of the form (3) is a strategy if for every $i \in \{1, \ldots, \ell\}$, $\lfloor q_i \rfloor$ is in **FO** and $\lfloor q_i \rfloor \sqsubseteq \lfloor q \rfloor$. The latter condition is equivalent to $q_i \sqsubseteq q$ as is shown next.

Lemma 1. *Let q and q' be self-join-free m-ary conjunctive queries. Then, $q \sqsubseteq q'$ if and only if $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$.*

Proof. Let $q = \{\vec{z} \mid \exists \vec{y}\, B\}$ and $q' = \{\vec{z}_0 \mid \exists \vec{y}_0\, B'\}$, where $\vec{z}$ and $\vec{z}_0$ both have the same length m.

$\boxed{\Longrightarrow}$ Straightforward. $\boxed{\Longleftarrow}$ Assume $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$. Let μ be an injective mapping with domain $\mathsf{vars}(B)$ that maps each variable to a fresh constant not occurring elsewhere. Since μ is injective, its inverse μ^{-1} is well defined. Let $\mathbf{db} = \mu(B)$. Clearly, $\mathbf{db}$ is consistent and $q(\mathbf{db}) = \{\mu(\vec{z})\} = \lfloor q \rfloor(\mathbf{db})$. From $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$, it follows $\mu(\vec{z}) \in \lfloor q' \rfloor(\mathbf{db}) = q'(\mathbf{db})$. Then, there exists a valuation θ over $\mathsf{vars}(B')$ such that $\theta(B') \subseteq \mathbf{db}$ and $\theta(\vec{z}_0) = \mu(\vec{z})$. Then $\mu^{-1} \circ \theta(B') \subseteq B$ and $\mu^{-1} \circ \theta(\vec{z}_0) = \vec{z}$. Since $\mu^{-1} \circ \theta$ is a homomorphism from q' to q, it follows $q \sqsubseteq q'$ by the Homomorphism Theorem [3, Theorem 6.2.3]. □

Lemma 1 does not hold for conjunctive queries with self-joins, as shown next.

Example 4. Let $q = \{\langle\rangle \mid R(\underline{a}, b) \wedge R(\underline{a}, c)\}$. For every uncertain database $\mathbf{db}$, we have $\lfloor q \rfloor(\mathbf{db}) = \{\}$. Let q' be a query such that $q \not\sqsubseteq q'$ (such query obviously exists). Then, $\lfloor q \rfloor \sqsubseteq \lfloor q' \rfloor$ and $q \not\sqsubseteq q'$. □

Lemma 1 allows us to construct strategies of the form (3), as follows. Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\vec{z})$. For some positive integer ℓ, generate self-join-free conjunctive queries $q_1, \ldots, q_\ell$ such that for each $i \in \{1, \ldots, \ell\}$, $q_i \sqsubseteq q$ and $\lfloor q_i \rfloor$ is in **FO**. The condition $q_i \sqsubseteq q$ is decidable by [3, Theorem 6.2.3]; the condition that $\lfloor q_i \rfloor$ is in **FO** is decidable by Theorem 2. Then by Lemma 1, $\bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$ is a strategy for q.

Unfortunately, Theorem 3 given hereinafter states that there are cases where no strategy of the form (3) is optimal. We first generalize Lemma 1 to unions.

Lemma 2. *Let $q_0, q_1, \ldots, q_\ell$ be self-join-free m-ary conjunctive queries. Then, $\lfloor q_0 \rfloor \sqsubseteq \bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$ if and only if for some $i \in \{1, \ldots, \ell\}$, $q_0 \sqsubseteq q_i$.*

Proof. $\boxed{\Longleftarrow}$ Straightforward. $\boxed{\Longrightarrow}$ Assume $\lfloor q_0 \rfloor \sqsubseteq \bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$. Let $q_0 = \{\vec{z}_0 \mid \exists \vec{y}_0\, B_0\}$, where B_0 is self-join-free. Let μ be an injective mapping with domain $\mathsf{vars}(B_0)$ that maps each variable to a fresh constant not occurring elsewhere. Since μ is injective, its inverse μ^{-1} is well defined. Let $\mathbf{db} = \mu(B_0)$. Clearly, $\mathbf{db}$ is consistent and $q_0(\mathbf{db}) = \{\mu(\vec{z}_0)\} = \lfloor q_0 \rfloor(\mathbf{db})$. From $\lfloor q_0 \rfloor \sqsubseteq \bigcup_{i=1}^{\ell} \lfloor q_i \rfloor$, it follows that we can assume $i \in \{1, \ldots, \ell\}$ such that $\mu(\vec{z}_0) \in \lfloor q_i \rfloor(\mathbf{db}) = q_i(\mathbf{db})$. Let $q_i = \{\vec{z}_i \mid \exists \vec{y}_i\, B_i\}$. Then, there exists a valuation θ over $\mathsf{vars}(B_i)$ such that $\theta(B_i) \subseteq \mathbf{db}$ and $\theta(\vec{z}_i) = \mu(\vec{z}_0)$. Then $\mu^{-1} \circ \theta(B_i) \subseteq B_0$ and $\mu^{-1} \circ \theta(\vec{z}_i) = \vec{z}_0$. Since $\mu^{-1} \circ \theta$ is a homomorphism from q_i to q_0, it follows $q_0 \sqsubseteq q_i$. □

Theorem 3. *There exists a self-join-free conjunctive query q such that for every strategy φ of the form (3) for q, there exists another strategy ψ of the form (3) for q such that $\varphi \sqsubset \psi$.*

Proof. Let $q = \{\langle\rangle \mid \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z)\right)\}$. Then $\lfloor q \rfloor$ is not in **FO** by Theorem 7 in the current paper (which is subsumed by Theorem 1 in [8]). For every constant c, let q_c be the query defined by $q_c := \{\langle\rangle \mid \exists y \exists z \left(R(\underline{c}, z) \wedge S(\underline{y}, z)\right)\}$. For every constant c, we have that $\lfloor q_c \rfloor \sqsubseteq \lfloor q \rfloor$ by Lemma 1, and again by Theorem 7, $\lfloor q_c \rfloor$ is in **FO**.

Let φ be a strategy for q of the form (3). Let A be the greatest set of constants such that for all $c \in A$, there exists some $i \in \{1, \ldots, \ell\}$ such that $q_i \equiv q_c$. Let b be a constant such that $b \notin A$. Clearly $\varphi \sqsubseteq \varphi \cup \lfloor q_b \rfloor \sqsubseteq \lfloor q \rfloor$. It suffices to show that $\varphi \sqsubset \varphi \cup \lfloor q_b \rfloor$, meaning that φ is not optimal.

Assume towards a contradiction that $\lfloor q_b \rfloor \sqsubseteq \varphi$. By Lemma 2, there exists $i \in \{1, \ldots, \ell\}$ such that $q_b \sqsubseteq q_i \sqsubseteq q$. We can assume (not necessarily distinct) variables s, t, u, v such that q_i is the existential closure of $\left(R(\underline{s}, t) \wedge S(\underline{u}, v)\right)$. From $q_i \sqsubseteq q$, it follows that $t = v$. From $q_b \sqsubseteq q_i$ and $b \notin A$, it follows that s, t, u are pairwise distinct variables. But then $q_i \equiv q$, contradicting that $\lfloor q_i \rfloor$ is in **FO**. We conclude by contradiction that $\varphi \sqsubset \varphi \cup \lfloor q_b \rfloor$. □

5.2. Post-processing by unions and quantification

The proof of Theorem 3 indicates that strategies of the form (3) lack expressiveness because the number of constants in such strategies is bounded. An obvious extension is to look for strategies that replace constants with existentially quantified variables. The following example shows how such extension solves the lack of expressiveness that underlies the proof of Theorem 3.

Example 5. Let $q = \exists x \exists y \exists z \left(R(\underline{x}, z) \wedge S(\underline{y}, z)\right)$ and consider the CQAFO formula φ defined by $\varphi := \exists X \lfloor \exists y \exists z \left(R(\underline{X}, z) \wedge S(\underline{y}, z)\right) \rfloor$. From Lemma 3 and Theorem 7 given hereinafter, it follows that φ is a strategy for q, i.e., $\varphi \sqsubseteq \lfloor q \rfloor$ and $\lfloor \exists y \exists z \left(R(\underline{X}, z) \wedge S(\underline{y}, z)\right) \rfloor$ is in **FO**. Recall from Example 2 that the use of upper case X is for readability. □

Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\vec{z})$. We next investigate strategies of the form

$$\bigcup_{i=1}^{\ell} Q_i, \tag{4}$$

where for each $i \in \{1 \ldots, \ell\}$, Q_i is a CQAFO query of the form

$$\{\vec{z}_i \mid \exists \vec{X}_i \lfloor \exists \vec{y}_i \, B_i \rfloor\}, \tag{5}$$

in which $\vec{z}_i$ has the same length as $\vec{z}$, and B_i is a self-join-free conjunction of atoms. It is understood that $\vec{z}_i$, $\vec{X}_i$, and $\vec{y}_i$ have, pairwise, no variables in common, and that $\mathsf{vars}(\vec{z}_i \vec{X}_i \vec{y}_i) = \mathsf{vars}(B_i)$. For readability, we will use upper case Q to refer to CQAFO queries of the form (5). The main tools for constructing strategies of the form (4) are provided by Theorems 4 and 5.

Theorem 4. *The following problem is decidable in polynomial time. Given a* CQAFO *query Q of the form (5), is Q in* **FO**? *Moreover, if Q is in* **FO**, *then a relational calculus query equivalent to Q can be effectively constructed.*

Proof. Let B be a self-join-free conjunction of atoms, and let

$$Q = \{\vec{z} \mid \exists \vec{X} \lfloor \exists \vec{y} \, B \rfloor\};$$
$$Q' = \{\vec{z}\vec{X} \mid \lfloor \exists \vec{y} \, B \rfloor\}.$$

Obviously, if Q' is in **FO**, then so is Q. We show next that if Q' is not in **FO**, then Q is not in **FO**.

For every variable x, we assume an infinite set of constants, denoted $\mathsf{type}(x)$, such that $x \neq y$ implies $\mathsf{type}(x) \cap \mathsf{type}(y) = \emptyset$. Let **db** be an uncertain database. We say that **db** is *typed relative to* B if for every atom $R(x_1, \ldots, x_n)$ in B, for every $i \in \{1, \ldots, n\}$, if x_i is a variable, then for every fact $R(a_1, \ldots, a_n)$ in **db**, $a_i \in \mathsf{type}(x_i)$ and the constant a_i does not occur in B. Significantly, since B is self-join-free, we can assume without loss of generality that Q and Q' are executed on databases that are typed relative to B.

From the complexity proofs in [8], it follows that if Q' is not in **FO**, then Q' is not in **FO** even if for every variable $v \in \mathsf{vars}(\vec{z}) \cup \mathsf{vars}(\vec{X})$ (i.e., for every free variable v of Q'), $\mathsf{type}(v)$ is a singleton. This means that if Q' is not in **FO**, it is not in **FO** even on uncertain databases **db** such that for every atom $R(x_1, \ldots, x_n)$ in B and $i \in \{1, \ldots, n\}$, if $x_i \in \mathsf{vars}(\vec{z}) \cup \mathsf{vars}(\vec{X})$, then all R-facts of **db** agree on position i. It is then obvious that if Q' is not in **FO**, it must be the case that Q is not in **FO** (because there is only one valuation for $\mathsf{vars}(\vec{z}) \cup \mathsf{vars}(\vec{X})$ that can make $\lfloor \exists \vec{y} \, B \rfloor$ true).

By Theorem 2, it can be decided whether Q' is in **FO**. A relational calculus query equivalent to Q can be straightforwardly obtained from a relational calculus query equivalent to Q'. □

We will be concerned with testing containment between CQAFO queries of the form (5). The following lemma generalizes Lemma 1 by allowing (restricted forms of) existential quantification outside $\lfloor \cdot \rfloor$.

Lemma 3. *Let B_1 and B_2 be self-join-free conjunctions of atoms in the following* CQAFO *queries:*

$$Q_1 = \{\vec{z}_1 \mid \exists \vec{X}_1 \lfloor \exists \vec{y}_1\, B_1 \rfloor\};$$
$$Q_2 = \{\vec{z}_2 \mid \exists \vec{X}_2 \lfloor \exists \vec{y}_2\, B_2 \rfloor\}.$$

Let q_1 and q_2 be the queries obtained from respectively Q_1 and Q_2 by omitting $\lfloor\cdot\rfloor$, that is,

$$q_1 = \{\vec{z}_1 \mid \exists \vec{X}_1 \exists \vec{y}_1\, B_1\};$$
$$q_2 = \{\vec{z}_2 \mid \exists \vec{X}_2 \exists \vec{y}_2\, B_2\}.$$

1. *If $Q_2 \sqsubseteq Q_1$, then $q_2 \sqsubseteq q_1$.*
2. *If X_1 is empty and $q_2 \sqsubseteq q_1$, then $Q_2 \sqsubseteq Q_1$.*

Proof. The proof of 1 is analogous to the proof of the if-direction of Lemma 1.

For 2, assume X_1 is empty and $q_2 \sqsubseteq q_1$. By the Homomorphism Theorem [3, Theorem 6.2.3], there exists a valuation θ over $\mathsf{vars}(B_1)$ such that $\theta(\vec{z}_1) = \vec{z}_2$ and $\theta(B_1) \subseteq B_2$. Let **db** be a database and $\vec{a}$ a sequence of constants such that $\vec{a} \in Q_2(\mathbf{db})$. Then, there exists a valuation γ over $\mathsf{vars}(\vec{z}_2) \cup \mathsf{vars}(\vec{X}_2)$ with $\gamma(\vec{z}_2) = \vec{a}$ such that for every repair **r** of **db**, γ can be extended into a valuation $\Gamma_{\mathbf{r}}$ over $\mathsf{vars}(B_2)$ such that $\Gamma_{\mathbf{r}}(B_2) \subseteq \mathbf{r}$. Let $\mathbf{r}_0$ be an arbitrary repair of **db**. The result $\vec{a} \in q_1(\mathbf{r}_0)$ follows because $\Gamma_{\mathbf{r}_0} \circ \theta$ is a valuation over $\mathsf{vars}(B_1)$ such that $\Gamma_{\mathbf{r}_0} \circ \theta(B_1) \subseteq \mathbf{r}_0$ and $\Gamma_{\mathbf{r}_0} \circ \theta(\vec{z}_1) = \vec{a}$. Since $\mathbf{r}_0$ be an arbitrary repair, from $\vec{a} \in q_1(\mathbf{r}_0)$ and X_1 empty, it follows $\vec{a} \in Q_1(\mathbf{db})$. □

Theorem 5. *Given a self-join-free conjunctive query q_1 and a* CQAFO *query Q_2 of the form (5), it can be decided whether $Q_2 \sqsubseteq \lfloor q_1 \rfloor$.*

Proof. Immediate from Lemma 3 and the decidability of containment for conjunctive queries. □

We point out that Theorem 5 is interesting in its own right. It is well known [3, Corollary 6.3.2] that containment of relational calculus queries is undecidable. A large fragment for which containment is decidable is the class of unions of conjunctive queries. Notice, however, that the queries in the statement of Theorem 5 need not be monotonic (and even not first-order), and that decidability of containment for such queries is not obvious. We next provide an example of such a non-monotonic query.

Example 6. Let $Q = \{x \mid \exists Y \lfloor R(\underline{x}, Y) \rfloor\}$. Let $\mathbf{db} = \{R(\underline{a}, 1)\}$ and $\mathbf{db}' = \{R(\underline{a}, 1), R(\underline{a}, 2)\}$. Then $\mathbf{db} \subseteq \mathbf{db}'$, but $Q(\mathbf{db}) = \{a\}$ is not contained in $Q(\mathbf{db}') = \{\}$. Hence Q is not monotonic. As a note aside, we observe that Q is equivalent to the following relational calculus query:

$$\{x \mid \exists y \big(R(\underline{x}, y) \wedge \forall y' \big(R(\underline{x}, y') \rightarrow y = y'\big)\big)\}. \quad \square$$

Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\vec{z})$. Theorem 5 allows us to build a strategy φ of the form (4) for q as follows. Let A be the set of constants that occur in q. Let φ be the disjunction of all (up to variable renaming) CQAFO formulas Q_i of the form (5) that use exclusively constants from A such that $Q_i \sqsubseteq \lfloor q \rfloor$ and Q_i is in **FO**. Clearly, there are at most finitely many such formulas (up to variable renaming). Containment of Q_i in $\lfloor q \rfloor$ is decidable by Theorem 5. Finally, the condition that Q_i is in **FO** is decidable by Theorem 4. The following theorem remedies the negative result of Theorem 3.

Theorem 6. *For every self-join-free conjunctive query q, there exists a computable strategy φ of the form (4) for q, such that for every strategy ψ of the form (4) for q, $\psi \sqsubseteq \varphi$.*

Proof. Assume that the input to OPTSTRATEGY is a self-join-free conjunctive query $q(\vec{z})$. Let φ be the strategy defined in the paragraph preceding this theorem. Let $Q = \{\vec{z}_0 \mid \exists \vec{X} \lfloor \exists \vec{y}\, B \rfloor\}$ be a query of the form (5) where B is a self-join-free conjunction of atoms such that Q is in **FO** and $Q \sqsubseteq \lfloor q \rfloor$. If all constants that occur in B also occur in q, then Q is already contained in some disjunct of φ (by construction of φ). Assume next that B contains some constants that do not occur in q, and let these constants be $a_1, \ldots, a_m$. For $i \in \{1, \ldots, m\}$, let X_i be a new fresh variable. Let B' be the conjunction obtained from B by replacing each occurrence of each a_i with X_i. Let $Q' = \{\vec{z}_0 \mid \exists \vec{X} \exists X_1 \cdots \exists X_m \lfloor \exists \vec{y} B' \rfloor\}$.

From $Q \sqsubseteq \lfloor q \rfloor$ and Lemma 3, it follows that $\{\vec{z}_0 \mid \exists \vec{X} \exists \vec{y}\, B\} \sqsubseteq q$. By the Homomorphism Theorem [3, Theorem 6.2.3], we can assume a homomorphism θ from q to $\{\vec{z}_0 \mid \exists \vec{X} \exists \vec{y}\, B\}$. Notice that if $\theta(t) = a_i$ for some term t that occurs in q and $i \in \{1, \ldots, m\}$, then it must be the case that t is a variable (because a_i does not occur in q). Let θ' be the substitution obtained from θ such that for every variable v in q and $i \in \{1, \ldots, m\}$,

$$\theta'(v) = \begin{cases} X_i & \text{if } \theta(v) = a_i \\ \theta(v) & \text{otherwise.} \end{cases}$$

Then obviously θ' is a homomorphism from q to $\{\vec{z}_0 \mid \exists\vec{X}\exists X_1 \cdots \exists X_m \exists\vec{y}\, B'\}$. From the Homomorphism Theorem and Lemma 3, it follows $Q' \sqsubseteq \lfloor q \rfloor$. It can be easily seen that $Q \sqsubseteq Q'$. Furthermore, Q' is in **FO** because Q is in **FO** and it can be easily argued that membership in **FO** is preserved if constants are replaced with free variables. Notice here that each variable X_i is free in $\lfloor \exists\vec{y} B' \rfloor$. Since all constants that occur in B' also occur in q, we have that Q' is already contained in some disjunct of φ (by construction of φ).

To conclude, whenever $Q = \{\vec{z}_0 \mid \exists\vec{X} \lfloor \exists\vec{y}\, B \rfloor\}$ is a query of the form (5) where B is a self-join-free conjunction of atoms such that Q is in **FO** and $Q \sqsubseteq \lfloor q \rfloor$, we have that $\varphi \cup Q \sqsubseteq \varphi$. $\square$

So far, we have imposed no restrictions on the size of the computable strategy φ in the statement of Theorem 6. From a practical point of view, it is interesting to construct, among all optimal strategies φ of the form (4), the one with the smallest number ℓ of disjuncts. This problem will be addressed in the next section.

6. Simplifying strategies

In Section 5.2, we considered strategies that are unions of CQAFO queries of the form (5). A natural question is whether such strategies can be simplified. One obvious simplification is to remove any component of the union that is contained in another component, which requires an effective procedure for deciding containment between queries of the from (5). Developing such a procedure turns out to be a challenging problem. In Section 6.1, we illustrate this problem and introduce some simplifying assumptions. We will tackle this problem by using an existing tool, called attack graph, which we recall in Section 6.2, and which we generalize to account for the two queries involved in a containment test (Section 6.3). In Section 6.4, we provide algorithm `ContainedIn` (Function 1) that decides containment of CQAFO queries of the form (5) under some additional restrictions.

6.1. Problem statement and motivation

We consider strategies $Q_1 \cup Q_2 \cup \cdots \cup Q_\ell$ consisting of CQAFO queries Q_i of the form $\{\vec{z}_i \mid \exists\vec{X}_i \lfloor \exists\vec{y}_i\, B_i \rfloor\}$. Clearly, if some Q_i is contained in another Q_j (i.e., $Q_i \sqsubseteq Q_j$ with $i \neq j$), then the presence of Q_i in the strategy is vacuous and Q_i is redundant. That is, an equivalent shorter strategy is obtained by removing Q_i from the union. This raises an important and interesting research question:

Given two CQAFO queries Q_1 and Q_2 of the form (5), decide whether $Q_1 \sqsubseteq Q_2$.

Theorem 5 settles containment of $Q_2 \sqsubseteq \lfloor q_1 \rfloor$. In this containment, the right-hand side $\lfloor q_1 \rfloor$ is restricted to have no quantifier outside the scope of $\lfloor\cdot\rfloor$. The opposite containment $\lfloor q_1 \rfloor \sqsubseteq Q_2$ turns out to be more difficult to handle, as illustrated next.

Example 7. Consider the following two Boolean queries:

$$q_2 = \exists u \exists v \exists w \left(R(\underline{u}, w) \wedge S(\underline{v}, w)\right);$$
$$Q_2 = \exists U \left\lfloor \exists v \exists w \left(R(\underline{U}, w) \wedge S(\underline{v}, w)\right) \right\rfloor,$$

and consider a database (call it **db**) with the following tables, where for readability, columns are named by variables, and blocks are separated by dashed lines.

R	$\underline{u}$	w
	a	1
	b	2

S	$\underline{v}$	w
	c	1
	c	2

The database **db** has two repairs, each satisfying q_2, hence $\mathbf{db} \models \lfloor q_2 \rfloor$. However, $\mathbf{db} \not\models Q_2$, because the two repairs of **db** use different values for u (a and b) to make the query true. So it is correct to conclude $\lfloor q_2 \rfloor \not\sqsubseteq Q_2$.

Consider furthermore the following query q_1:

$$q_1 = \exists x \exists y \left(R(\underline{x}, y) \wedge S(\underline{x}, y)\right).$$

By means of the Homomorphism Theorem [3, p. 117], it can be verified that $q_1 \sqsubseteq q_2$, hence $\lfloor q_1 \rfloor \sqsubseteq \lfloor q_2 \rfloor$ by Lemma 1. It takes some effort to see that if a database satisfies $\lfloor q_1 \rfloor$, then it must contain two singleton blocks of the form $\{R(\underline{d}, e)\}$ and $\{S(\underline{d}, e)\}$, as follows.

R	$\underline{x}$	y
	d	e
		$\vdots$

S	$\underline{x}$	y
	d	e
		$\vdots$

Such database will necessarily satisfy Q_2, hence $\lfloor q_1 \rfloor \sqsubseteq Q_2$. $\square$

It turns out that the containment problem for queries of the form (5) is quite challenging. To ease the technical treatment, we make the following simplifications:

- We will only deal with Boolean conjunctive queries (i.e., henceforth, all variables are assumed to be quantified). By Proposition 1, the restriction to Boolean queries does not compromise generality. At some places, it will be convenient (and unambiguous) to denote a Boolean conjunctive query by its set of atoms. For example, $q_1 = \{\langle\rangle \mid \exists x \exists y \exists z \left(R(\underline{x}, z) \land S(\underline{y}, z)\right)\}$ can be denoted by the set $\{R(\underline{x}, z), S(\underline{y}, z)\}$.
- Let q be a self-join-free conjunction of atoms. Let $\vec{X}$ be a sequence of distinct variables such that $\mathsf{vars}(\vec{X}) \subseteq \mathsf{vars}(q)$. We write $\exists \vec{X} \lfloor q \rfloor$ for the query

$$\exists \vec{X} \left\lfloor \exists \vec{u}\, q \right\rfloor,$$

  where $\mathsf{vars}(\vec{u}) = \mathsf{vars}(q) \setminus \mathsf{vars}(\vec{X})$. That is, we only show the quantifiers that are outside the scope of $\lfloor\cdot\rfloor$.
- Our results concerning containment $\exists \vec{X}_1 \lfloor q_1 \rfloor \sqsubseteq \exists \vec{X}_1 \lfloor q_2 \rfloor$ will often require a homomorphism from q_2 to q_1 (which is tantamount to requiring $q_1 \sqsubseteq q_2$, by the Homomorphism Theorem [3, p. 117]). This requirement is reasonable, because if no such homomorphism exists, then $\exists \vec{X}_1 \lfloor q_1 \rfloor \not\sqsubseteq \exists \vec{X}_2 \lfloor q_2 \rfloor$ by Lemma 3. For completeness, we recall here that a *homomorphism* from q_2 to q_1 is a mapping h with domain $\mathsf{vars}(q_2)$ such that for every atom $R(s_1, \ldots, s_\ell)$ in q_2, we have that $R(h(s_1), \ldots, h(s_\ell))$ belongs to q_1.

Proposition 1. *Let Q_2 and Q_1 be two* CQAFO *queries of the form (5). One can compute in polynomial time two Boolean* CQAFO *queries Q_2' and Q_1', both of the form (5), such that $Q_1 \sqsubseteq Q_2$ if and only if $Q_1' \sqsubseteq Q_2'$.*

Proof. We can assume self-join-free conjunctions of atoms, B_1 and B_2, such that:

$$Q_1 = \left\{\vec{z}_1 \mid \exists \vec{X}_1 \lfloor \exists \vec{y}_1\, B_1 \rfloor\right\};$$
$$Q_2 = \left\{\vec{z}_2 \mid \exists \vec{X}_2 \lfloor \exists \vec{y}_2\, B_2 \rfloor\right\}.$$

Let q_1 and q_2 be the queries obtained from respectively Q_1 and Q_2 by omitting $\lfloor\cdot\rfloor$, that is,

$$q_1 = \left\{\vec{z}_1 \mid \exists \vec{X}_1 \exists \vec{y}_1\, B_1\right\};$$
$$q_2 = \left\{\vec{z}_2 \mid \exists \vec{X}_2 \exists \vec{y}_2\, B_2\right\}.$$

If $q_1 \not\sqsubseteq q_2$, then $Q_1 \not\sqsubseteq Q_2$ by Lemma 3. In this case, pick two distinct key-equal facts A and B and let $Q_1' = A$ and $Q_2' = B$. Clearly, $Q_1' \not\sqsubseteq Q_2'$. Notice that the test $q_1 \sqsubseteq q_2$ can be performed in polynomial time in the absence of self-joins.

Assume next $q_1 \sqsubseteq q_2$. By the Homomorphism Theorem [3, Theorem 6.2.3], we can assume a valuation θ over $\mathsf{vars}(B_2)$ such that $\theta(B_2) \subseteq B_1$ and $\theta(\vec{z}_2) = \vec{z}_1$. Let μ be a valuation over $\mathsf{vars}(\vec{z}_1)$ that maps distinct variables to distinct fresh constants. Let $Q_1' := \left\{\mu(\vec{z}_1) \mid \exists \vec{X}_1 \lfloor \exists \vec{y}_1\, \mu(B_1) \rfloor\right\}$, the query obtained from Q_1 by replacing each occurrence of each variable $z_1 \in \mathsf{vars}(\vec{z}_1)$ with $\mu(z_1)$. Intuitively, Q_1' is the Boolean query obtained from Q_1 by treating free variables as constants. Since B_1 is self-join-free, it can be seen that $Q_1 \sqsubseteq Q_2$ if and only if $Q_1' \sqsubseteq Q_2$.

For example, for $Q_1 = \left\{z \mid \exists X \lfloor \exists y \left(R(\underline{X}, y, b) \land S(\underline{X}, y, z)\right) \rfloor\right\}$ with b a constant and $R \neq S$, we would have that $Q_1' = \left\{c \mid \exists X \lfloor \exists y \left(R(\underline{X}, y, b) \land S(\underline{X}, y, c)\right) \rfloor\right\}$, where c is a fresh constant. Notice that the above construction would make no sense in the presence of self-joins. In particular, if $R = S$, then any answer to Q_1' would be empty (because $b \neq c$).

Since the answer to Q_1' is either empty or the singleton $\{\mu(\vec{z}_1)\}$, the containment $Q_1' \sqsubseteq Q_2$ holds if Q_2 returns $\{\mu(\vec{z}_1)\}$ whenever Q_1' does. Let Q_2' be the query obtained from Q_2 by replacing each occurrence of each variable $z_2 \in \mathsf{vars}(\vec{z}_2)$ with $\mu \circ \theta(z_2)$. That is, the free tuple in Q_2' is equal to $\mu(\vec{z}_1)$. It is now obvious that $Q_1' \sqsubseteq Q_2$ if and only if $Q_1' \sqsubseteq Q_2'$. This concludes the proof. $\square$

To sum up, we start with two Boolean conjunctive queries q_1 and q_2 such that $q_1 \sqsubseteq q_2$ (and hence $\lfloor q_1 \rfloor \sqsubseteq \lfloor q_2 \rfloor$ by Lemma 1), and we want to know which existential quantification can be moved "outside the scope of $\lfloor\cdot\rfloor$" while preserving the containment $\lfloor q_1 \rfloor \sqsubseteq \lfloor q_2 \rfloor$. For the left-hand side (i.e., $\lfloor q_1 \rfloor$), this is easy because, by Lemma 3, $\exists \vec{X}_1 \lfloor q_1 \rfloor \sqsubseteq \lfloor q_2 \rfloor$ if and only if $\lfloor q_1 \rfloor \sqsubseteq \lfloor q_2 \rfloor$. For the right-hand side (i.e., $\lfloor q_2 \rfloor$), our major result will be an algorithm for deciding the containment $\lfloor q_1 \rfloor \sqsubseteq \exists \vec{X}_2 \lfloor q_2 \rfloor$ (Theorem 9), albeit by imposing some further restrictions on q_1. We leave the design of a general containment test for future work.

To explain how the containment test works, we first recall the notion of attack graph which is defined relative to a single query (Section 6.2) and then introduce a new notion of attack that takes into account two queries q_1 and q_2 related by a homomorphism (Section 6.3).

Please cite this article in press as: F. Geerts et al., First-order under-approximations of consistent query answers, Int. J. Approx. Reason. (2016), http://dx.doi.org/10.1016/j.ijar.2016.10.005

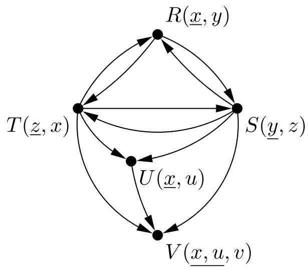

Fig. 2. Attack graph of the query in Example 8.

6.2. Attack graphs

The construct of *attack graph* is the main tool for determining the complexity of $\lfloor q \rfloor$. Attack graphs were first introduced in [28] for studying first-order expressibility of $\lfloor q \rfloor$ for self-join-free conjunctive queries q.

Let q be a self-join-free Boolean conjunctive query (denoted by its set of atoms). We define $\mathcal{K}(q)$ as the following set of functional dependencies:

$$\mathcal{K}(q) := \{\mathsf{key}(F) \rightarrow \mathsf{vars}(F) \mid F \in q\}.$$

For every atom $F \in q$, we define $F^{+,q}$ as the following set of variables:

$$F^{+,q} := \{x \in \mathsf{vars}(q) \mid \mathcal{K}(q \setminus \{F\}) \models \mathsf{key}(F) \rightarrow x\}.$$

Here, the symbol $\models$ denotes standard logical entailment. The *attack graph of* q is a directed graph whose vertices are the atoms of q. There is a directed edge from F to G ($F \neq G$) if there exists a sequence

$$F_0 \overset{z_1}{\frown} F_1 \overset{z_2}{\frown} F_2 \ldots \overset{z_n}{\frown} F_n \tag{6}$$

where

- $F_0, \ldots, F_n$ are atoms of q;
- $F_0 = F$ and $F_n = G$; and
- for all $i \in \{1, \ldots, n\}$, $z_i \in \mathsf{vars}(F_{i-1}) \cap \mathsf{vars}(F_i)$ and $z_i \notin F^{+,q}$.

A directed edge from F to G in the attack graph of q is also called an *attack from* F *to* G, denoted by $F \overset{q}{\rightsquigarrow} G$. The sequence (6) is called a *witness* for the attack $F \overset{q}{\rightsquigarrow} G$. If $F \overset{q}{\rightsquigarrow} G$, then we also say that F *attacks* G (or that G is attacked by F).

Example 8. Let $q = \{R(\underline{x}, y), S(\underline{y}, z), T(\underline{z}, x), U(\underline{x}, u), V(\underline{x, u}, v)\}$. We have $R^{+,q} = \{x, u, v\}$. A witness for $R \overset{q}{\rightsquigarrow} T$ is $R \overset{y}{\frown} S \overset{z}{\frown} T$. Note that, by an abuse of notation, we write R to mean the R-atom of q. The complete attack graph is shown in Fig. 2. □

Equipped with the notion of attack graph, we can now present a theorem that explains the decidability result of Theorem 2.

Theorem 7 *([8]). For every self-join-free Boolean conjunctive query q, the query $\lfloor q \rfloor$ is in* **FO** *if and only if the attack graph of q is acyclic.*

The attacks defined so far are from an atom to another atom. Attacks from an atom to a variable are defined as follows: $F \overset{q}{\rightsquigarrow} x$ if $F \overset{q \cup \{N(\underline{x})\}}{\rightsquigarrow} N(\underline{x})$, where N is a new relation name with signature $[1, 1]$. That is, $F \overset{q}{\rightsquigarrow} x$ if there is an attack from F to the "dummy" atom $N(\underline{x})$ in the attack graph of $q \cup \{N(\underline{x})\}$. The following lemma gives an important semantic property of unattacked variables.

Lemma 4. *Let q be a self-join-free Boolean conjunctive query. Let $x \in \mathsf{vars}(q)$ such that for every atom F of q, $F \overset{q}{\not\rightsquigarrow} x$. Then for every database* **db** *such that* **db** $\models \lfloor q \rfloor$*, there exists a constant c such that* **db** $\models \lfloor q_{[x \mapsto c]} \rfloor$.

Proof. Let $q' = q \cup \{N(\underline{x})\}$ where N is a fresh relation name. The attack graph of q' can be obtained from the attack graph of q by adding the isolated vertex $N(\underline{x})$. The desired result then follows form Lemma 9 in [8]. □

The proof of the following lemma is analogous to the proof of Lemma C.1 in [29]. Intuitively, it states that no new attacks emerge if we replace a variable with a constant in a Boolean self-join-free conjunctive query.

Lemma 5. *Let q be a self-join-free Boolean conjunctive query. Let c be a constant and let $q' = q_{[x \mapsto c]}$. For every $F \in q$, let F' be the atom in q' with the same relation name as F. For all $F, G \in q$, if $F' \overset{q'}{\rightsquigarrow} G'$, then $F \overset{q}{\rightsquigarrow} G$.*

Let q be a self-join-free Boolean conjunctive query such that the attack graph of q is acyclic. To avoid non-determinism in some definitions and results to follow, assume a lexicographic order on the atoms of q. We write $\mathsf{head}(q)$ to denote the first (in lexicographic order) atom of q that has no incoming attacks in the attack graph of q.

6.3. A new attack notion

We now define a generalized attack notion, which refers to two Boolean conjunctive queries, q_1 and q_2, such that there exists a homomorphism from q_2 to q_1. This new attack notion, denoted by the symbol $\overset{q_2 q_1}{\rightsquigarrow}$, turns out to be a useful tool in the study of the containment problem for queries of the form (5).

Definition 2. Let q_1 and q_2 be self-join-free Boolean conjunctive queries such that there exists a homomorphism (call it h) from q_2 to q_1. Notice that such a homomorphism, if it exists, is unique (because the queries are self-join-free). Let G and H be distinct atoms of q_2. We write

$$G \overset{q_2 q_1}{\rightsquigarrow} H$$

if there exists a sequence

$$G_0 \overset{u_1}{\frown} G_1 \overset{u_2}{\frown} G_2 \dots \overset{u_\ell}{\frown} G_\ell \tag{7}$$

such that

1. $G_0, G_1, \dots, G_\ell$ are atoms of q_2 such that $G_0 = G$ and $G_\ell = H$;
2. for all $i \in \{1, \dots, \ell\}$, $u_i \in \mathsf{vars}(G_{i-1}) \cap \mathsf{vars}(G_i)$;
3. for all $i \in \{1, \dots, \ell\}$, $h(u_i)$ is a variable that does not belong to $\big(h(G_0)\big)^{+,q_1}$.

Let $u \in \mathsf{vars}(q_2)$. We write

$$G \overset{q_2 q_1}{\rightsquigarrow} u$$

if

$$G \overset{q_2' q_1'}{\rightsquigarrow} N(\underline{u})$$

where

1. N is a new relation name with signature $[1, 1]$;
2. $q_2' = q_2 \cup \{N(\underline{u})\}$; and
3. $q_1' = q_1 \cup \{N(h(\underline{u}))\}$.

Notice that if $h(u)$ is a constant, then $G \overset{q_2 q_1}{\not\rightsquigarrow} u$. Note also that for every atom F of q_1, $F^{+,q_1} = F^{+,q_1'}$. □

Intuitively, $G \overset{q_2 q_1}{\rightsquigarrow} H$ if there exists a sequence of the form (7) whose image under the homomorphism h is a witness for $h(G) \overset{q_1}{\rightsquigarrow} h(H)$. The notion $\overset{q_2 q_1}{\rightsquigarrow}$ is a proper generalization of $\overset{q_1}{\rightsquigarrow}$, because for $q_1 = q_2$, the relationship $\overset{q_2 q_1}{\rightsquigarrow}$ is the same as $\overset{q_1}{\rightsquigarrow}$. That is, $F \overset{q_1 q_1}{\rightsquigarrow} F'$ if and only if $F \overset{q_1}{\rightsquigarrow} F'$.

Example 9. Let $q_2 = \{R(\underline{u}, x)\}$ and $q_1 = \{R(\underline{y}, z), S(\underline{z})\}$. Then, $R(\underline{u}, x) \overset{q_2 q_1}{\rightsquigarrow} x$ because $R(\underline{u}, x) \overset{q_2' q_1'}{\rightsquigarrow} N(\underline{x})$, where $q_2' = \{R(\underline{u}, x), N(\underline{x})\}$ and $q_1' = \{R(\underline{y}, z), S(\underline{z}), N(\underline{z})\}$. Indeed, note that $R(\underline{u}, x)$ and $N(\underline{x})$ share the variable x, and $h(x) = z$ does not belong to $R^{+,q_1'} = \{y\}$. □

Example 10. Consider the following two queries:

$$q_2 = \big\{R(\underline{a}, u), S(\underline{u}, x_1), T(\underline{x_2})\big\};$$
$$q_1 = \big\{R(\underline{a}, y), S(\underline{y}, z), T(\underline{z})\big\},$$

and let h be the (unique) homomorphism from q_2 to q_1. Notice that $h(x_1) = h(x_2) = z$. Since $\mathsf{key}(R) = \emptyset$ in q_1, but $\mathsf{key}(S) \neq \emptyset \neq \mathsf{key}(T)$, we have $R^{+,q_1} = \emptyset$. Hence, $R(\underline{a}, u) \overset{q_2 q_1}{\rightsquigarrow} x_1$ and $R(\underline{a}, u) \overset{q_2 q_1}{\rightsquigarrow} u$ trivially hold. Note, however, that $R(\underline{a}, u) \overset{q_2 q_1}{\not\rightsquigarrow} x_2$. This is because the atom $T(\underline{x_2})$ shares no variable with any other atom of q_2. $\square$

6.4. Testing containment

The following theorem expresses a significant relationship between $\overset{q_2 q_1}{\rightsquigarrow}$ and query containment for queries of the form (5). Paraphrasing somewhat, if $\lfloor q_1 \rfloor \sqsubseteq \lfloor q_2 \rfloor$ and $u \in \mathsf{vars}(q_2)$ such that $G \overset{q_2 q_1}{\rightsquigarrow} u$ for some $G \in q_2$, then query containment is lost if the quantification of the variable u is moved outside the scope of $\lfloor \cdot \rfloor$. It is an open question whether the inverse of Theorem 8 also holds.

Theorem 8. *Let q_1 and q_2 be self-join-free Boolean conjunctive queries such that there exists a homomorphism (call it h) from q_2 to q_1. Let $u \in \mathsf{vars}(q_2)$. If $G \overset{q_2 q_1}{\rightsquigarrow} u$ for some $G \in q_2$, then $\lfloor q_1 \rfloor \not\sqsubseteq \exists u \lfloor q_2 \rfloor$.*

Proof. We first fix some notations. Let $G_0 \in q_2$ such that $G_0 \overset{q_2 q_1}{\rightsquigarrow} u$. Let $h(G_0) = F_0$ and $h(u) = w$. Assume that R_0 is the relation name of G_0 (which is necessarily equal to the relation name of F_0). We show that $\lfloor q_1 \rfloor \not\sqsubseteq \exists u \lfloor q_2 \rfloor$ by constructing a database instance $\mathbf{db}$ such that $\mathbf{db} \models \lfloor q_1 \rfloor$ but $\mathbf{db} \not\models \exists u \lfloor q_2 \rfloor$.

To define $\mathbf{db}$, let θ, μ be two valuations over $\mathsf{vars}(q_1)$ such that for every $x \in \mathsf{vars}(q_1)$, $\theta(x) = \mu(x)$ if and only if $x \in F_0{}^{+,q_1}$. Assume that $q_1 = \left\{ \langle\rangle \mid \exists \vec{y}\, B_1 \right\}$. Let $\mathbf{db} = \theta(B_1) \cup \mu(B_1)$. We next show that $\mathbf{db}$ has only two repairs, denoted by $\mathbf{r}$ and $\mathbf{s}$, where

$$\mathbf{r} = \mathbf{db} \setminus \{\mu(F_0)\};$$

$$\mathbf{s} = \mathbf{db} \setminus \{\theta(F_0)\}.$$

To see that these are repairs, we first show that for every $F \in q_1 \setminus \{F_0\}$, the facts $\theta(F)$ and $\mu(F)$ are either equal or not key-equal, i.e., they never constitute two distinct facts of a same block. Indeed, for every $F \in q_1 \setminus \{F_0\}$, two cases are possible:

Case $\mathsf{key}(F) \subseteq F_0{}^{+,q_1}$**.** Then, $\mathsf{vars}(F) \subseteq F_0{}^{+,q_1}$, and thus θ and μ agree on all variables of $\mathsf{vars}(F)$. That is, $\theta(F) = \mu(F)$.

Case $\mathsf{key}(F) \not\subseteq F_0{}^{+,q_1}$**.** Then, by the definition of θ and μ, for some variable $x \in \mathsf{key}(F)$, $\theta(x) \neq \mu(x)$, hence $\theta(F)$ and $\mu(F)$ are not key-equal.

Furthermore, when considering F_0, $\theta(F_0)$ and $\mu(F_0)$ are distinct and key-equal (hence, $\mathbf{r}$ contains $\theta(F_0)$ and $\mathbf{s}$ contains $\mu(F_0)$). The facts $\theta(F_0)$ and $\mu(F_0)$ are key-equal because $\mathsf{key}(F_0) \subseteq F_0{}^{+,q_1}$ is obvious. Further, from $G_0 \overset{q_2 q_1}{\rightsquigarrow} u$, we can assume some variable $y \in \mathsf{vars}(F_0)$ such that $F_0 \overset{q_1}{\rightsquigarrow} y$, hence $y \notin F_0{}^{+,q_1}$. Since θ and μ disagree on y, we have $\theta(F_0) \neq \mu(F_0)$. Clearly, $\mathbf{r}$ and $\mathbf{s}$ are the only repairs of $\mathbf{db}$, since $\{\theta(F_0), \mu(F_0)\}$ is the only block of $\mathbf{db}$ with more than one fact.

It is obvious that $\mathbf{r} \models q_1$ and $\mathbf{s} \models q_1$, hence $\mathbf{db} \models \lfloor q_1 \rfloor$ since $\mathbf{r}$ and $\mathbf{s}$ are the only repairs of $\mathbf{db}$. We now show that $\mathbf{db} \not\models \exists u \lfloor q_2 \rfloor$, or in other words, that there is no constant c such that both $\mathbf{r} \models q_{2[u \mapsto c]}$ and $\mathbf{s} \models q_{2[u \mapsto c]}$. First, we show that if $\mathbf{r} \models q_{2[u \mapsto c]}$ and $\mathbf{s} \models q_{2[u \mapsto c]}$ for some constant c, then it must be the case that either $c = \mu(w)$ or $c = \theta(w)$. Indeed, for every valuation α over $\mathsf{vars}(q_2)$ such that $\alpha(q_2) \subseteq \mathbf{r}$, we have $\alpha(u) \in \{\mu(w), \theta(w)\}$. Likewise, for every valuation β over $\mathsf{vars}(q_2)$ such that $\beta(q_2) \subseteq \mathbf{s}$, we have $\beta(u) \in \{\mu(w), \theta(w)\}$. Second, we show that $\mu(w) \neq \theta(w)$. Indeed, from $G_0 \overset{q_2 q_1}{\rightsquigarrow} u$, it is correct to conclude $w \notin F_0{}^{+,q_1}$. To see this, consider a sequence $G_0 \overset{u_1}{\frown} G_1 \overset{u_2}{\frown} G_2 \ldots \overset{u}{\frown} N(\underline{u})$ witnessing that $G_0 \overset{q_2 q_1}{\rightsquigarrow} u$. Then, $h(u) = w \notin \left(h(G_0)\right)^{+,q_1'} = F_0{}^{+,q_1}$. From the definition of μ and θ, it is correct to conclude that $\mu(w) \neq \theta(w)$. Finally, we show that $\mathbf{r} \not\models q_{2[u \mapsto \mu(w)]}$ and $\mathbf{s} \not\models q_{2[u \mapsto \theta(w)]}$. This suffices to show that $\mathbf{db} \not\models \exists u \lfloor q_2 \rfloor$.

We show $\mathbf{r} \not\models q_{2[u \mapsto \mu(w)]}$ (the proof of $\mathbf{s} \not\models q_{2[u \mapsto \theta(w)]}$ is symmetrical). More specifically, we show that any valuation α over $\mathsf{vars}(q_2)$ such that $\alpha(q_2) \subseteq \mathbf{r}$ satisfies $\alpha(u) = \theta(w)$. Hence, $\alpha(u) \neq \mu(w)$ for any such valuation α and it is correct to infer that $\mathbf{r} \not\models q_{2[u \mapsto \mu(w)]}$.

It is easily verified that from $G_0 \overset{q_2 q_1}{\rightsquigarrow} u$, it follows that for some $\ell \geq 0$, there exists a sequence

$$G_0 \overset{u_1}{\frown} G_1 \overset{u_2}{\frown} G_2 \ldots \overset{u_\ell}{\frown} G_\ell \tag{8}$$

such that

1. $G_0, G_1, \ldots, G_\ell$ are atoms of q_2;
2. $u \in \mathsf{vars}(G_\ell)$;
3. for all $i \in \{1, \ldots, \ell\}$, $u_i \in \mathsf{vars}(G_{i-1}) \cap \mathsf{vars}(G_i)$; and
4. for all $i \in \{1, \ldots, \ell\}$, $h(u_i)$ is a variable such that $\mu(h(u_i)) \neq \theta(h(u_i))$.

Function ContainedIn(q_1,q_2,u) **is**
  Data: self-join-free Boolean conjunctive queries q_2 and q_1 such that $|q_2| = |q_1|$, there exists a homomorphism from q_2 to q_1, and $\lfloor q_1 \rfloor$ is in **FO**; a variable u
  Result: Is $\lfloor q_1 \rfloor \sqsubseteq \exists u \lfloor q_2 \rfloor$?
  if $u \notin \mathsf{vars}(q_2)$ **then**
    return *true*
  else
    let h be the (unique) homomorphism from q_2 to q_1
    let $F_0 := \mathsf{head}(q_1)$
    let G_0 be the (unique) atom of q_2 such that $h(G_0) = F_0$
    if $G_0 \overset{q_2 q_1}{\rightsquigarrow} u$ **then**
      return *false*
    else
      let $\hat{q}_1 := q_1 \setminus \{F_0\}$
      let $\hat{q}_2 := q_2 \setminus \{G_0\}$
      let α be an arbitrary valuation over $\mathsf{vars}(F_0)$
      return ContainedIn($\alpha(\hat{q}_1)$,$\hat{q}_2$,u)
    end
  end
end

Function 1. ContainedIn.

Observe that (4) is equivalent to $h(u_i) \notin h(G_0)^{+,q_1} = F_0{}^{+,q_1}$ (for all $i \in \{1, \dots, \ell\}$). For every $i \in \{1, \dots, \ell\}$, define $w_i := h(u_i)$. Let α be a valuation over $\mathsf{vars}(q_2)$ such that $\alpha(q_2) \subseteq \mathbf{r}$. Based on the sequence (8), we show by induction on increasing i that for $i \in \{0, \dots, \ell\}$, $\alpha(G_i) = \theta(F_i)$. This suffices since if this holds, then $\alpha(G_\ell) = \theta(F_\ell)$ and since $u \in \mathsf{vars}(G_\ell)$, $\alpha(u) = \theta(w)$.

The induction hypothesis trivially holds for $i = 0$. Indeed, as argued above, $\theta(F_0)$ is the only R_0-fact of $\mathbf{r}$.

For the induction step, $i \mapsto i + 1$, the induction hypothesis is that for all $j \in \{0, \dots, i\}$, $\alpha(G_j) = \theta(F_j)$. Clearly, since $u_{i+1} \in \mathsf{vars}(G_i)$, we have that $\alpha(u_{i+1}) = \theta(u_{i+1})$. Then, since $u_{i+1} \in \mathsf{vars}(G_{i+1})$ and $\theta(w_{i+1}) \neq \mu(w_{i+1})$, it must be the case that $\alpha(G_{i+1}) = \theta(F_{i+1})$.

So we obtain $\alpha(G_\ell) = \theta(F_\ell)$, hence $\alpha(u) = \theta(w)$. This concludes the proof. $\square$

As already mentioned, it is an open question whether the inverse of Theorem 8 also holds:

From $\lfloor q_1 \rfloor \sqsubseteq \lfloor q_2 \rfloor$, $u \in \mathsf{vars}(q_2)$, and $G \overset{q_2 q_1}{\not\rightsquigarrow} u$ for all $G \in q_2$, is it correct to conclude $\lfloor q_1 \rfloor \sqsubseteq \exists u \lfloor q_2 \rfloor$?

Theorem 9 provides a positive answer to this question under restrictions on q_1. The theorem is stated in the form of Function 1, which recursively checks whether the variable u has an incoming $\overset{q_2 q_1}{\rightsquigarrow}$-attack. The function will be called once for every atom of q_2. We briefly discuss the restrictions imposed on q_1 by Theorem 9.

- The restriction that q_1 and q_2 have the same cardinality can be easily met, because we can always add "dummy" atoms to a conjunctive query without affecting query containment. For example, if q_1 contains an R-atom with signature $[n, k]$, but q_2 contains no R-atom, then we can add to q_2 the dummy atom $R(\underline{u_1, \dots, u_k}, u_{k+1}, \dots, u_n)$, where each u_i is a fresh variable not occurring elsewhere.
- The restriction that $\lfloor q_1 \rfloor$ is in **FO** is not problematic for the application we have in mind, which, as explained in Section 6.1, is the simplification of strategies, which are unions of queries of the form (5) that are in **FO**. Notice that no such restriction is imposed on $\lfloor q_2 \rfloor$, which can thus be a query not in **FO**.
- The more technical restriction is $F^{+,q_1} \subseteq \mathsf{vars}(F)$. This restriction is met, for example, by the queries $q_{11} = \exists x \exists y \left(R(\underline{x}, y) \wedge S(\underline{x}, y)\right)$ and $q_{12} = \exists x \exists y \exists z \left(R(\underline{x}, y) \wedge S(\underline{y}, z)\right)$, but not by $q_{13} = \exists x \exists y \left(R(\underline{x}, y) \wedge S(\underline{x}, z)\right)$ (because $R^{+,q_{13}} = \{x, z\}$ and $z \notin \mathsf{vars}(R)$). This restriction excludes some queries, but is not overly prohibitive. It is an open question whether Theorem 9 can be proved without relying on this restriction.

Theorem 9. *Let q_1 and q_2 be self-join-free Boolean conjunctive queries, of the same cardinality, such that there exists a homomorphism (call it h) from q_2 to q_1. Assume that $\lfloor q_1 \rfloor$ is in* **FO** *and that for every $F \in q_1$, it is the case that $F^{+,q_1} \subseteq \mathsf{vars}(F)$. Then the following are equivalent for any variable u:*

1. ContainedIn(q_1,q_2,u) *returns true; and*
2. $\lfloor q_1 \rfloor \sqsubseteq \exists u \lfloor q_2 \rfloor$.

Proof. $\boxed{2 \Longrightarrow 1}$ Proof by contraposition. Assume that ContainedIn(q_1,q_2,u) returns false. Then, at some point in the execution of ContainedIn(q_1,q_2,u), the test "**if** $G_0 \overset{q_2 q_1}{\rightsquigarrow} u$" returns true. Let $F_0, F_1, \dots, F_n$ be a topological sort of the attack graph of q_1 where ties are broken lexicographically. For every $i \in \{0, \dots, n\}$, let G_i be the atom of q_2 with the same relation name as F_i (i.e., $h(G_i) = F_i$). Then, there exists $\ell \in \{0, \dots, n\}$ such that $G_\ell \overset{q_2' \alpha(q_1')}{\rightsquigarrow} u$ where

- $q_2' = \{G_\ell, G_{\ell+1}, \dots, G_n\}$,
- $q_1' = \{F_\ell, F_{\ell+1}, \dots, F_n\}$, and
- α is a valuation over $\mathsf{vars}(F_0) \cup \mathsf{vars}(F_1) \cup \dots \cup \mathsf{vars}(F_{\ell-1})$.

We have $\alpha(F_\ell) \overset{\alpha(q_1')}{\rightsquigarrow} h(u)$. From Lemma 5, it follows $F_\ell \overset{q_1}{\rightsquigarrow} h(u)$. It is now easy to see $G_\ell \overset{q_2 q_1}{\rightsquigarrow} u$. By Theorem 8, $\lfloor q_1 \rfloor \not\sqsubseteq \exists u \lfloor q_2 \rfloor$.

$\boxed{1 \Longrightarrow 2}$ We use the following notations:

$$
\begin{array}{lcl}
h & := & \text{(unique) homomorphism from } q_2 \text{ to } q_1; \\
F_0 & := & \mathsf{head}(q_1); \\
G_0 & := & \text{the (unique) atom in } q_2 \text{ such that } h(G_0) = F_0; \\
\hat{q}_1 & := & q_1 \setminus \{F_0\}; \\
\hat{q}_2 & := & q_2 \setminus \{G_0\}.
\end{array}
$$

The initial assumptions are the following:

1. `ContainedIn`(q_1,q_2,u) returns true;
2. **db** is a database such that every repair of **db** satisfies q_1.

The proof runs by structural induction. For the base case (i.e., $u \notin \mathsf{vars}(q_2)$), it is obvious that $\exists u \lfloor q_2 \rfloor \equiv \lfloor q_2 \rfloor$ and the desired result holds because there exists a homomorphism from q_2 to q_1. Assume hereinafter that $u \in \mathsf{vars}(q_2)$.

Since $\lfloor q_1 \rfloor$ is in **FO**, the attack graph of q_1 is acyclic. Let $R_0, R_1, \dots, R_n$ be a topological ordering of the attack graph of q_1, where ties are broken lexicographically.[3] Since $F_0 = \mathsf{head}(q_1)$, the relation name of F_0 is R_0.

We need to show that $\mathbf{db} \models \exists u \lfloor q_2 \rfloor$. Clearly, since $\mathbf{db} \models \lfloor q_1 \rfloor$, there must exist a (not necessarily unique) subset $\mathbf{db}_0$ of **db** such that

1. $\mathbf{db}_0 \models \lfloor q_1 \rfloor$;
2. for every block $\mathcal{B}$ of **db**, either $\mathcal{B} \subseteq \mathbf{db}_0$ or $\mathcal{B} \cap \mathbf{db}_0 = \emptyset$.
3. *Minimality:* for every block $\mathcal{B}$ of $\mathbf{db}_0$, we have $\mathbf{db}_0 \setminus \mathcal{B} \not\models \lfloor q_1 \rfloor$.

In practice, $\mathbf{db}_0$ can be obtained from **db** by repeatedly removing blocks until the further removal of any more block would lead to a database that falsifies $\lfloor q_1 \rfloor$. We will show that $\mathbf{db}_0 \models \exists u \lfloor q_2 \rfloor$, which obviously implies $\mathbf{db} \models \exists u \lfloor q_2 \rfloor$ (because every repair of **db** contains a repair of $\mathbf{db}_0$).

Let the set of R_0-facts in $\mathbf{db}_0$ be $\{A_1, \dots, A_m\}$. For $1 \leq i \leq m$, denote by θ_i the (unique) valuation over $\mathsf{vars}(F_0)$ such that $\theta_i(F_0) = A_i$. We show the following:

Agreement Property: For every $v \in \mathsf{vars}(F_0) \cap F_0^{+,q_1}$, for all $i, j \in \{1, \dots, m\}$, $\theta_i(v) = \theta_j(v)$.

To this extent, let $v \in \mathsf{vars}(F_0) \cap F_0^{+,q_1}$. Then, $F_0 \overset{q_1}{\not\rightsquigarrow} v$. Moreover, since F_0 has no incoming attacks in the attack graph of q_1, we have that for all $F \in q_1$, $F \overset{q_1}{\not\rightsquigarrow} v$. From Lemma 4, it follows that for all $i, j \in \{1, \dots, m\}$, $\theta_i(v) = \theta_j(v)$, which concludes the proof of the *Agreement Property*. Notice that from $\mathsf{key}(F_0) \subseteq F_0^{+,q_1}$ and the *Agreement Property*, it follows that the set $\{A_1, \dots, A_m\}$ is the unique R_0-block of $\mathbf{db}_0$.

It suffices now to show that there exists a constant b (which depends on $\mathbf{db}_0$) such that every repair of $\mathbf{db}_0$ satisfies $q_{2[u \mapsto b]}$. We distinguish two cases, the first case being the easier one.

Case $u \in \mathsf{vars}(G_0)$

In this case, it can be shown that all R_0-facts agree on the position at which u occurs in G_0. Indeed, from $G_0 \overset{q_2 q_1}{\not\rightsquigarrow} u$ (since `ContainedIn`(q_1,q_2,u) returns true), it follows $h(u) \in F_0^{+,q_1}$. From $h(u) \in \mathsf{vars}(F_0)$ and the *Agreement Property*, it follows that for all $i, j \in \{1, \dots, m\}$, $\theta_i(h(u)) = \theta_j(h(u))$. In this case, the desired result holds for $b = \theta_1(h(u))$.

Case $u \notin \mathsf{vars}(G_0)$

Let $\hat{\mathbf{db}}_0 := \mathbf{db}_0 \setminus \{A_1, \dots, A_m\}$. For $i \in \{1, \dots, m\}$, denote by $\hat{\mathbf{db}}_0^i$ a minimal subset of $\hat{\mathbf{db}}_0$ such that $\hat{\mathbf{db}}_0^i \models \lfloor \theta_i(\hat{q}_1) \rfloor$ and every block of $\hat{\mathbf{db}}_0$ is either contained in $\hat{\mathbf{db}}_0^i$ or disjoint with $\hat{\mathbf{db}}_0^i$. That is, $\hat{\mathbf{db}}_0^i$ is obtained from $\hat{\mathbf{db}}_0$ relative to $\theta_i(\hat{q}_1)$ in

[3] By an abuse of notation, we blur the distinction between atoms and their relation names.

Please cite this article in press as: F. Geerts et al., First-order under-approximations of consistent query answers, Int. J. Approx. Reason. (2016), http://dx.doi.org/10.1016/j.ijar.2016.10.005

exactly the same way as $\mathbf{db}_0$ was obtained from $\mathbf{db}$ relative to q_1. In the same way as $\{A_1, \ldots, A_m\}$ was shown to be the only R_0-block of $\mathbf{db}_0$, it can be shown that for each $i \in \{1, \ldots, m\}$, $\hat{\mathbf{db}}_0^{\,i}$ contains only one R_1-block.

It follows from Lemma 5 that $R_1, R_2, \ldots, R_n$ will be a topological sort of the attack graph of $\theta_i(\hat{q}_1)$ (for all $1 \leq i \leq m$). The following hold for any $i \in \{1, \ldots, m\}$:

- from our initial hypothesis that ContainedIn(q_1,q_2,u) returns true, it follows that ContainedIn($\theta_i(\hat{q}_1)$,$\hat{q}_2$,u) returns true; and
- by the induction hypothesis, there exists a constant b_i such that every repair $\hat{\mathbf{r}}$ of $\hat{\mathbf{db}}_0^{\,i}$ satisfies $\hat{q}_{2[u \mapsto b_i]}$.

We show that $\mathbf{db}_0 \models \lfloor q_{2[u \mapsto b_1]} \rfloor$ (i.e., we fix $i = 1$). By symmetry, it will actually follow that for every $i \in \{1, \ldots, m\}$, $\mathbf{db}_0 \models \lfloor q_{2[u \mapsto b_i]} \rfloor$.

Let $\mathbf{r}$ be an arbitrary repair of $\mathbf{db}_0$. We need to show $\mathbf{r} \models q_{2[u \mapsto b_1]}$.

We can assume $\ell \in \{1, \ldots, m\}$ such that $A_\ell \in \mathbf{r}$. Since $\mathbf{r} \models q_1$, there exists a valuation δ over $\mathsf{vars}(q_1)$ such that $\delta(q_1) \subseteq \mathbf{r}$ and $\delta(F_0) = A_\ell$. The latter follows because A_ℓ is the only R_0-fact in $\mathbf{r}$. Let α be the valuation over $\mathsf{vars}(q_2)$ such that for every $x \in \mathsf{vars}(q_2)$, $\alpha(x) = \delta(h(x))$. Obviously, $\alpha(q_2) = \delta(q_1) \subseteq \mathbf{r}$ and $\alpha(G_0) = A_\ell$.

Clearly, $\mathbf{r} \cap \hat{\mathbf{db}}_0^{\,1}$ is a repair of $\hat{\mathbf{db}}_0^{\,1}$. By the induction hypothesis, we can assume a valuation β over $\mathsf{vars}(q_2)$ such that

1. $\beta(\hat{q}_2) \subseteq \mathbf{r} \cap \hat{\mathbf{db}}_0^{\,1}$;
2. $\beta(u) = b_1$; and
3. $\beta(G_0) = A_1$.

Notice that the induction hypothesis gives us the first two items. The last item follows from the construction of $\hat{\mathbf{db}}_0^{\,1}$.

Let γ be the valuation over $\mathsf{vars}(q_2)$ such that for every $x \in \mathsf{vars}(q_2)$,

$$\gamma(x) = \begin{cases} \alpha(x) & \text{if } G_0 \overset{q_2 q_1}{\rightsquigarrow} x \\ \beta(x) & \text{otherwise} \end{cases} \tag{9}$$

From the construction of γ and $G_0 \overset{q_2 q_1}{\not\rightsquigarrow} u$, it follows $\gamma(u) = b_1$. It remains to be shown that $\gamma(q_2) \subseteq \mathbf{r}$. To this extent, let G be an arbitrary atom of q_2. It remains to be shown that $\gamma(G) \in \mathbf{r}$. We distinguish two cases.

Case $G = G_0$**.** Recall that $\alpha(G_0) = A_\ell$, $\beta(G_0) = A_1$, and $A_\ell \in \mathbf{r}$. We show that $\gamma(G_0) = \alpha(G_0) = A_\ell$. To this extent, let w be an arbitrary variable in $\mathsf{vars}(G_0)$. If $G_0 \overset{q_2 q_1}{\rightsquigarrow} w$, then $\gamma(w) = \alpha(w)$ by the construction of γ in (9). Consider next $G_0 \overset{q_2 q_1}{\not\rightsquigarrow} w$. Then it must be the case that $h(w) \in {F_0}^{+,q_1}$ and, by the *Agreement Property*, A_1 and A_ℓ agree on the position at which w occurs in G_0. Then, $\alpha(w) = \beta(w)$.

Case $G \neq G_0$**.** Assume towards a contradiction $G \in q_2$ such that $\gamma(G) \notin \mathbf{r}$. Then, it must be the case that $\alpha(G) \neq \gamma(G) \neq \beta(G)$, because $\alpha(G)$ and $\beta(G)$ belong to $\mathbf{r}$. Then we can assume $y_1, y_2 \in \mathsf{vars}(G)$ such that $\gamma(y_1) = \alpha(y_1) \neq \beta(y_1)$ and $\gamma(y_2) = \beta(y_2) \neq \alpha(y_2)$. We next show a contradiction by proving $\alpha(y_2) = \beta(y_2)$.

Observe that by the construction of γ in (9), from $\gamma(y_1) = \alpha(y_1) \neq \beta(y_1)$, it follows $G_0 \overset{q_2 q_1}{\rightsquigarrow} y_1$. Likewise, from $\gamma(y_2) = \beta(y_2) \neq \alpha(y_2)$, it follows $G_0 \overset{q_2 q_1}{\not\rightsquigarrow} y_2$. We show next $h(y_2) \in {F_0}^{+,q_1}$.

From $G_0 \overset{q_2 q_1}{\rightsquigarrow} y_1$ and $y_1 \in \mathsf{vars}(G)$, it follows $G_0 \overset{q_2 q_1}{\rightsquigarrow} G$, which implies the existence of a sequence of the form (7) with $G_\ell = G$. Then for every variable $v \in \mathsf{vars}(G)$, either $G_0 \overset{q_2 q_1}{\rightsquigarrow} v$ or $h(v) \in {F_0}^{+,q_1}$. Since $y_2 \in \mathsf{vars}(G)$ and $G_0 \overset{q_2 q_1}{\not\rightsquigarrow} y_2$, it must be the case $h(y_2) \in {F_0}^{+,q_1}$.

The statement of Theorem 9 makes the hypothesis that ${F_0}^{+,q_1} \subseteq \mathsf{vars}(F_0)$, hence $h(y_2) \in \mathsf{vars}(F_0)$. Then, by the *Agreement Property*, it is correct to conclude that for all $i, j \in \{1, \ldots, m\}$, $\theta_i(h(y_2)) = \theta_j(h(y_2))$. In the remainder of the proof, we denote by d the constant such that for all $i \in \{1, \ldots, m\}$, $\theta_i(h(y_2)) = d$. Intuitively, this means that all R_0-facts of $\mathbf{db}_0$ contain the constant d at the position at which $h(y_2)$ occurs in F_0. Note incidentally that this does not mean that y_2 occurs in G_0, because the homomorphism h can map distinct variables of q_2 to the same variable in q_1 (i.e., h needs not to be injective).

Let F be the atom such that $h(G) = F$, and let the relation name of F be R. From $G \neq G_0$, it follows $F \neq F_0$ (and $R \neq R_0$). Since y_2 occurs in G, $h(y_2)$ occurs in F. So $h(y_2)$ occurs in both F_0 and F. Let o be the arity of R and let $p \in \{1, \ldots, o\}$ such that y_2 occurs at position p in G (and hence $h(y_2)$ occurs at position p in F). The construction of $\mathbf{db}_0$ ensures that all R-facts of $\mathbf{db}_0$ will contain the same constant d at position p. Indeed, if an R-fact A of $\mathbf{db}$ contains a distinct constant at position p, then the block containing A will be excluded from $\mathbf{db}_0$ (because of the *Minimality* condition). It follows $\alpha(y_2) = d = \beta(y_2)$, a contradiction. We conclude by contradiction that $\gamma(G) \in \mathbf{r}$.

This concludes the proof. □

Please cite this article in press as: F. Geerts et al., First-order under-approximations of consistent query answers, Int. J. Approx. Reason. (2016), http://dx.doi.org/10.1016/j.ijar.2016.10.005

7. Conclusion

We have studied a realistic setting for divulging an inconsistent database to end users. In this setting, users access the database exclusively via syntactically restricted queries, and get exclusively consistent answers computable in **FO** data complexity. If the data complexity is higher, then the query will be rejected, in which case users have to fall back on strategies that obtain a large (the larger, the better) subset of the consistent answer. Such strategies combine answers obtained from several "easier" queries.

Although our setting applies to arbitrary queries and constraints, we searched for strategies when constraints are primary keys, and the database is accessible only via self-join-free conjunctive queries for which consistent query answering is in **FO**. Under these access restrictions, we showed how to construct strategies that combine answers by means of union and quantification. It turns out that the simplification of such strategies raises a novel and challenging query containment problem. By means of a new tool (a generalization of attack graphs), we were able to solve this containment problem under some syntactic restrictions, leaving a general solution for future work. Another interesting open question is whether our strategies can still be improved, e.g., by using negation.

Of practical interest is the development of an academic prototype that allows investigating the real-life applicability and efficiency of the proposed strategies.

References

[1] F. Geerts, F. Pijcke, J. Wijsen, First-order under-approximations of consistent query answers, in: C. Beierle, A. Dekhtyar (Eds.), Scalable Uncertainty Management – Proceedings of the 9th International Conference, SUM 2015, Québec City, QC, Canada, September 16–18, 2015, in: Lecture Notes in Computer Science, vol. 9310, Springer, 2015, pp. 354–367, http://dx.doi.org/10.1007/978-3-319-23540-0_24.

[2] M. Arenas, L.E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: V. Vianu, C.H. Papadimitriou (Eds.), Proceedings of the Eighteenth ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, May 31–June 2, 1999, Philadelphia, PA, USA, ACM Press, 1999, pp. 68–79, http://doi.acm.org/10.1145/303976.303983.

[3] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison–Wesley, 1995.

[4] J. Chomicki, J. Marcinkowski, Minimal-change integrity maintenance using tuple deletions, Inf. Comput. 197 (2005) 90–121.

[5] L. Libkin, SQL's three-valued logic and certain answers, in: M. Arenas, M. Ugarte (Eds.), 18th International Conference on Database Theory, ICDT 2015, March 23–27, 2015, Brussels, Belgium, in: LIPIcs, vol. 31, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015, pp. 94–109, http://dx.doi.org/10.4230/LIPIcs.ICDT.2015.94.

[6] N. Immerman, Descriptive Complexity, Graduate Texts in Computer Science, Springer, 1999, http://dx.doi.org/10.1007/978-1-4612-0539-5.

[7] N.V. Cao, E. Fragnière, J.-A. Gauthier, M. Sapin, E.D. Widmer, Optimizing the marriage market: an application of the linear assignment model, Eur. J. Oper. Res. 202 (2010) 547–553.

[8] P. Koutris, J. Wijsen, The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints, in: T. Milo, D. Calvanese (Eds.), Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31–June 4, 2015, ACM, 2015, pp. 17–29, http://doi.acm.org/10.1145/2745754.2745769.

[9] W. Fan, F. Geerts, Foundations of Data Quality Management, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2012, http://dx.doi.org/10.2200/S00439ED1V01Y201207DTM030.

[10] L.E. Bertossi, Database Repairing and Consistent Query Answering, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2011.

[11] A. Fuxman, R.J. Miller, First-order query rewriting for inconsistent databases, in: T. Eiter, L. Libkin (Eds.), Database Theory, Proceedings of the 10th International Conference, ICDT 2005, Edinburgh, UK, January 5–7, 2005, in: Lecture Notes in Computer Science, vol. 3363, Springer, 2005, pp. 337–351, http://dx.doi.org/10.1007/978-3-540-30570-5_23.

[12] J. Wijsen, A survey of the data complexity of consistent query answering under key constraints, in: C. Beierle, C. Meghini (Eds.), Foundations of Information and Knowledge Systems – Proceedings of the 8th International Symposium, FoIKS 2014, Bordeaux, France, March 3–7, 2014, in: Lecture Notes in Computer Science, vol. 8367, Springer, 2014, pp. 62–78, http://dx.doi.org/10.1007/978-3-319-04939-7_2.

[13] P. Koutris, J. Wijsen, Consistent query answering for primary keys, SIGMOD Rec. 45 (2016) 15–22.

[14] D. Maslowski, J. Wijsen, A dichotomy in the complexity of counting database repairs, J. Comput. Syst. Sci. 79 (2013) 958–983.

[15] D. Maslowski, J. Wijsen, Counting database repairs that satisfy conjunctive queries with self-joins, in: N. Schweikardt, V. Christophides, V. Leroy (Eds.), Proc. 17th International Conference on Database Theory, ICDT, Athens, Greece, March 24–28, 2014, OpenProceedings.org, 2014, pp. 155–164, http://dx.doi.org/10.5441/002/icdt.2014.18.

[16] J. Wijsen, Charting the tractability frontier of certain conjunctive query answering, in: R. Hull, W. Fan (Eds.), Proceedings of the 32nd ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA, June 22–27, 2013, ACM, 2013, pp. 189–200, http://doi.acm.org/10.1145/2463664.2463666.

[17] N.N. Dalvi, C. Ré, D. Suciu, Probabilistic databases: diamonds in the dirt, Commun. ACM 52 (2009) 86–94.

[18] N.N. Dalvi, C. Re, D. Suciu, Queries and materialized views on probabilistic databases, J. Comput. Syst. Sci. 77 (2011) 473–490.

[19] S. Greco, C. Molinaro, Approximate probabilistic query answering over inconsistent databases, in: Q. Li, S. Spaccapietra, E.S.K. Yu, A. Olivé (Eds.), Conceptual Modeling – ER 2008, Proceedings of the 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20–24, 2008, in: Lecture Notes in Computer Science, vol. 5231, Springer, 2008, pp. 311–325, http://dx.doi.org/10.1007/978-3-540-87877-3_23.

[20] G. Greco, S. Greco, E. Zumpano, A logical framework for querying and repairing inconsistent databases, IEEE Trans. Knowl. Data Eng. 15 (2003) 1389–1408.

[21] P.G. Kolaitis, E. Pema, W. Tan, Efficient querying of inconsistent databases with binary integer programming, Proc. VLDB Endow. 6 (2013) 397–408.

[22] A. Fuxman, E. Fazli, R.J. Miller, ConQuer: efficient management of inconsistent databases, in: F. Özcan (Ed.), Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, MD, USA, June 14–16, 2005, ACM, 2005, pp. 155–166, http://doi.acm.org/10.1145/1066157.1066176.

[23] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D.F. Savo, Inconsistency-tolerant query answering in ontology-based data access, J. Web Semant. 33 (2015) 3–29.

[24] M. Bienvenu, R. Rosati, Tractable approximations of consistent query answering for robust ontology-based data access, in: F. Rossi (Ed.), Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013, Beijing, China, August 3–9, 2013, IJCAI/AAAI, 2013, pp. 775–781, http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6904.

Please cite this article in press as: F. Geerts et al., First-order under-approximations of consistent query answers, Int. J. Approx. Reason. (2016), http://dx.doi.org/10.1016/j.ijar.2016.10.005

[25] J. Wijsen, Making more out of an inconsistent database, in: G. Gottlob, A.A. Benczúr, J. Demetrovics (Eds.), Advances in Databases and Information Systems, Proceedings of the 8th East European Conference, ADBIS 2004, Budapest, Hungary, September 22–25, 2004, in: Lecture Notes in Computer Science, vol. 3255, Springer, 2004, pp. 291–305, http://dx.doi.org/10.1007/978-3-540-30204-9_20.
[26] L.E. Bertossi, L. Li, Achieving data privacy through secrecy views and null-based virtual updates, IEEE Trans. Knowl. Data Eng. 25 (2013) 987–1000.
[27] L. Libkin, Elements of Finite Model Theory, Springer, 2004.
[28] J. Wijsen, On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases, in: J. Paredaens, D.V. Gucht (Eds.), Proceedings of the Twenty-Ninth ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems, PODS 2010, June 6–11, 2010, Indianapolis, IN, USA, ACM, 2010, pp. 179–190, http://doi.acm.org/10.1145/1807085.1807111.
[29] J. Wijsen, Certain conjunctive query answering in first-order logic, ACM Trans. Database Syst. 37 (2012) 9.

Please cite this article in press as: F. Geerts et al., First-order under-approximations of consistent query answers, Int. J. Approx. Reason. (2016), http://dx.doi.org/10.1016/j.ijar.2016.10.005