On the Computational Complexity of Consistent Query Answers

Jan Chomicki*
Dept. CSE
University at Buffalo
chomicki@cse.buffalo.edu

Jerzy Marcinkowski
Instytut Informatyki
Wroclaw University, Poland
jma@ii.uni.wroc.pl

Keywords: databases, computational complexity, integrity constraints.

1 Introduction

It is nowadays common to build databases integrating information from multiple, autonomous, distributed data sources. The problem of *data integration* is nevertheless very complex [9, 10]. In this paper, we consider a specific issue arising in data integration; how to obtain reliable, consistent information from *inconsistent databases* – databases that do not have to satisfy given integrity constraints. Such databases occur in a natural way in data integration, since there is typically no global monitor that could guarantee that the integrated database satisfies the constraints. The data sources are independent and even if they separately satisfy the constraints, the integrated database may fail to do so. For example, different data sources may contain different, locally unique addresses for the same person, leading to the violation of the global uniqueness constraint for people's addresses. Inconsistent databases occur also in other contexts. For instance, integrity constraints may fail to be enforced for efficiency reasons, or because the inconsistencies are temporary. Or, there may be not enough information to resolve inconsistencies, while the database may have to continue being used for real-time decision support.

To formalize the notion of consistent information obtained from a (possibly inconsistent) database in response to a user query, we proposed in [1] the notion of a *consistent query answer*. A consistent answer is, intuitively, true regardless of the way the database is fixed to remove constraint violations. Thus answer consistency serves as an indication of its reliability. The different ways of fixing an

*Contact author. Address: Dept. CSE, 201 Bell Hall, Univ. at Buffalo, Buffalo, NY 14260-2000. Fax: (716) 645-3464. Phone: (716) 645-3180, ext.103.

inconsistent database are formalized using the notion of *repair*: another database that is consistent and minimally differs from the original database.

Example 1 *Consider the following instance of a relation Person*

Name	*City*	*Street*
Brown	Amherst	115 Klein
Brown	Amherst	120 Maple
Green	Clarence	4000 Transit

and the functional dependency Name $\rightarrow$ City Street. Clearly, the above instance does not satisfy the dependency. There are two repairs: one is obtained by removing the first tuple, the other by removing the second. The consistent answer to the query $Person(n, c, s)$ *is just the tuple* (Green,Clarence,4000 Transit)*. On the other hand, the query* $\exists s[Person(n, c, s)]$ *has two consistent answers:* (Brown,Amherst) *and* (Green,Clarence)*. Similarly, the query*

$$Person(\text{Brown}, \text{Amherst}, \text{115 Klein}) \vee Person(\text{Brown}, \text{Amherst}, \text{120 Maple})$$

has true *as the consistent answer. Notice that for the last two queries the approach based on removing all inconsistent tuples and evaluating the original query using the remaining tuples gives different, less informative results.*

In [1], in addition to a formal definition of a consistent query answer, a computational mechanism for obtaining such answers was presented in the context of first-order queries. In [3], the same problem was studied for scalar aggregation queries. In [1] some cases were identified where consistent query answers are tractable (in PTIME). In the present paper, we provide a *complete classification* of the computational complexity of computing consistent query answers to first-order queries. We consider *functional dependencies* (FDs) and their generalization: *denial constraints*. Denial constraints allow an arbitrary number of literals per constraint and arbitrary built-in predicates. They also relax the typedness restriction of FDs. Denial constraints are particularly useful for databases with interpreted data, e.g., numbers. Their implication problem was studied in [5].

Example 2 *The constraint that* no employee can have a salary greater than that of her manager *is the denial constraint*

$$\forall n, s, m, s', m'. \neg[Emp(n, s, m) \wedge Emp(m, s', m') \wedge s > s'].$$

The results of [1] imply that for binary denial constraints consistent answers can be computed in PTIME for queries that are conjunctions of literals. In the present paper we strengthen that result to arbitrary quantifier-free queries and arbitrary denial constraints. We also identify a class of restricted existentially quantified queries (consisting of single literals), for which consistent query answers can be computed in PTIME. In general, we show how the complexity depends on the *type* of the constraints considered, their *number*, and the *size* of the query. Related work is discussed in depth in [1, 3, 4]. Other papers that adopt our notion of consistent query answer include [2, 8, 7].

2 Basic Notions

In this paper we assume that we have a fixed database schema containing only one relation schema R with the set of attributes U. We will denote elements of U by $A, B, \ldots$, subsets of U by $X, Y, \ldots$, and the union of X and Y by XY. We also have two fixed, disjoint infinite database domains: D (uninterpreted constants) and N (numbers). We assume that elements of the domains with different names are different. The database instances can be seen as first order structures that share the domains D and N. Every attribute in U is typed, thus all the instances of R can contain only elements either of D or of N in a single attribute. Since each instance is finite, it has a finite active domain which is a subset of $D \cup N$. As usual, we allow the standard built-in predicates over N $(=, \neq, <, >, \leq, \geq)$ that have infinite, fixed extensions.

Integrity constraints are typed, closed first-order formulas over the vocabulary consisting of R and the built-in predicates over N.

Definition 1 *Given a database instance r of R and a set of integrity constraints F, we say that r is* consistent *if $r \vDash F$ in the standard model-theoretic sense;* inconsistent *otherwise.*

We consider the following classes of integrity constraints:

- *denial constraints*: formulas of the form $\forall \bar{x}_1, \ldots \bar{x}_k . \neg[R(\bar{x}_1) \wedge \cdots \wedge R(\bar{x}_m) \wedge \phi(\bar{x}_1, \ldots, \bar{x}_m)]$ where $\bar{x}_1, \ldots, \bar{x}_m$ are tuples of variables and constants, and ϕ is a conjunction of atomic formulas referring to built-in predicates;
- *functional dependencies* (FDs) $X \rightarrow Y$ over the set U (*key* FDs if X is a key of R).

Clearly, functional dependencies are a special case of denial constraints.

Definition 2 *For the instances r, r', r'' , $r' \leq_r r''$ if $r - r' \subseteq r - r''$.* □

Definition 3 *Given a set of integrity constraints F and database instances r and r', we say that r' is a* repair *of r w.r.t. F if $r' \vDash F$ and r' is $\leq_r$-minimal in the class of database instances that satisfy F.* □

We denote by $Repairs_F(r)$ the set of repairs of r w.r.t. F. For any set of denial constraints, all the repairs are obtained by deleting tuples from the table.

Definition 4 *[1] Given a set of integrity constraints F and a database instance r, we say that a (ground) tuple $\bar{t}$ is a* consistent answer *to a query $Q(\bar{x})$ w.r.t. F in r, and we write $r \models_F Q(\bar{t})$ if for every $r' \in Repairs_F(r)$, $r' \vDash Q(\bar{t})$. If Q is a sentence, then true (false) is a* consistent answer *to Q w.r.t. F in r, and we write $r \models_F Q$ $(r \models_F \neg Q)$, if for every $r' \in Repairs_F(r)$, $r' \vDash Q$ $(r' \nvDash Q)$.* □

3 Data Complexity of Consistent Query Answers

Assume a class of databases $\mathcal{D}$, a class of queries $\mathcal{L}$ and a class of integrity constraints $\mathcal{IC}$ are given. We study here the *data complexity* [6, 11] of consistent query answers, i.e., the complexity of (deciding the membership of) the sets $D_{F,\phi} = \{r : r \in \mathcal{D} \wedge r \models_F \phi\}$ for a fixed sentence $\phi \in \mathcal{L}$ and a fixed finite set $F \in \mathcal{IC}$ of integrity constraints.

Proposition 1 *[4] For any set of denial constraints F and sentence ϕ, $D_{F,\phi}$ is in co-NP.*

It is easy to see that even under a single key FD, there may be exponentially many repairs and thus the approach to computing consistent query answers by generating and examining all repairs is not feasible.

Example 3 *Consider the functional dependency $A \rightarrow B$ and the following family of relation instances r_n, $n > 0$, each of which has $2n$ tuples (represented as columns) and 2^n repairs:*

r_n							
A	a_1	a_1	a_2	a_2	$\cdots$	a_n	a_n
B	b_0	b_1	b_0	b_1	$\cdots$	b_0	b_1

Given a set of denial constraints F and an instance r, all the repairs of r with respect to F can be succinctly represented as the *conflict hypergraph*. This is a generalization of the *conflict graph* defined in [3] for FDs only.

Definition 5 *The* conflict hypergraph $\mathcal{G}_{F,r}$ *is a hypergraph whose set of vertices is the set of tuples in r and whose set of edges consists of all the sets $\{\bar{t}_1, \bar{t}_2, \ldots \bar{t}_l\}$ such that $\bar{t}_1, \bar{t}_2, \ldots \bar{t}_l \in r$, and there is a constraint*

$$\forall \bar{x}_1, \bar{x}_2, \ldots \bar{x}_l \neg [R(\bar{x}_1) \wedge R(\bar{x}_2) \wedge \ldots \wedge R(\bar{x}_l) \wedge \phi(\bar{x}_1, \bar{x}_2, \ldots \bar{x}_l)]$$

in F such that $\bar{t}_1, \bar{t}_2, \ldots \bar{t}_l$ violate together this constraint, which means that there exists a substitution ρ such that $\rho(\bar{x}_1) = \bar{t}_1, \rho(\bar{x}_2) = \bar{t}_2, \ldots \rho(\bar{x}_l) = \bar{t}_l$ and that $\phi(\bar{t}_1, \bar{t}_2, \ldots \bar{t}_l)$ is true.

By an *independent set* in a hypergraph we mean a subset of its set of vertices which does not contain any edge.

Proposition 2 *Each repair of r w.r.t. F corresponds to a maximal independent set in $\mathcal{G}_{F,r}$.*

3.1 Positive results

Theorem 1 *For every set F of denial constraints and quantifier-free sentence Φ, $D_{F,\Phi}$ is in PTIME.*

Proof. We assume the sentence is in CNF, i.e., of the form $\Phi = \Phi_1 \wedge \Phi_2 \wedge \ldots \Phi_l$, where each Φ_i is a disjunction of ground literals. Φ is true in every repair of r if and only if each of the clauses Φ_i is true in every repair. So it is enough to provide a polynomial algorithm which will check if for a given ground clause *true* is a consistent answer.

It is easier to think that we are checking if for a ground clause *true* is **not** a consistent answer. This means that we are checking, whether there exists a repair r' in which $\neg\Phi_i$ is true for some i. But $\neg\Phi_i$ is of the form $R(\bar{t}_1) \wedge R(\bar{t}_2) \wedge \ldots \wedge R(\bar{t}_m) \wedge \neg R(\bar{t}_{m+1}) \ldots \neg R(\bar{t}_n)$, where $\bar{t}_j$ are tuples of constants. Thus it is enough to check two conditions:

1. whether for every j, $m+1 \leq j \leq n$, $\bar{t}_j \notin r$ or there exists an edge $E_j \in \mathcal{G}_{F,r}$ such that $\bar{t}_j \in E_j$, and
2. there is no edge $E \in \mathcal{G}_{F,r}$ such that $E \subseteq r'$ where

$$r' = \{\bar{t}_1, \ldots, \bar{t}_m\} \cup \bigcup_{m+1 \leq j \leq n, \bar{t}_j \in r} (E_j - \{\bar{t}_j\}).$$

If the conditions are satisfied, then a repair in which $\neg\Phi_i$ is true can be built by adding to r' new tuples from r until the set is maximal independent. The conditions can be checked by a nondeterministic algorithm that needs $n-m$ nondeterministic steps, a number which is independent of the size of the database, and in each of its nondeterministic steps selects one possibility from a set whose size is polynomial in the size of the database. So there is an equivalent PTIME deterministic algorithm. ■

Note that the above result holds also for constraints and queries involving more than one relation. The notion of conflict hypergraph needs to be appropriately generalized in this case.

Theorem 2 *Let F consist of a single FD. Then for each sentence Q of the form $\exists\bar{t}[R(\bar{t}) \wedge \phi(\bar{t})]$ (where ϕ is quantifier-free and only built-in predicates occur there), there exists a sentence Q' such that for every database instance r, $r \models_F Q$ iff $r \models Q'$. Consequently, $D_{F,Q}$ is in PTIME.*

Proof. The FD is $A_1 \ldots A_l \rightarrow A_{l+1}, \ldots A_{l+m}$, where $l+m$ is not greater than the arity k of R. Let $\bar{x}$ be a vector of distinct variables of length l, $\bar{y}$ and $\bar{y}_1$ vectors of distinct variables of length m, and $\bar{z}$, $\bar{z}_1$ and $\bar{z}_2$ vectors of distinct variables of length $k-(l+m)$. Then, the query Q' is as follows:

$$\exists\bar{x}, \bar{y}, \bar{z}\forall\bar{y}_1, \bar{z}_1\exists\bar{z}_2[R(\bar{x}, \bar{y}, \bar{z}) \wedge \phi(\bar{x}, \bar{y}, \bar{z}) \wedge [R(\bar{x}, \bar{y}_1, \bar{z}_1) \Rightarrow [R(\bar{x}, \bar{y}_1, \bar{z}_2) \wedge \phi(\bar{x}, \bar{y}_1, \bar{z}_2)]]].$$

■

We show now that the above results are the strongest possible, since relaxing any of the restrictions leads to co-NP-completeness. This is the case even though we limit ourselves to *key* FDs.

3.2 One key dependency, two query literals

Theorem 3 *There exist a key FD f and a query $Q \equiv \exists x, y, z[R(x, y, c) \wedge R(z, y, c')]$, for which $D_{\{f\},Q}$ is co-NP-data-complete.*

Proof. Reduction from MONOTONE 3-SAT. The FD is $A \rightarrow BC$. Let $\Phi = \phi_1 \wedge \ldots \phi_m \wedge \psi_{m+1} \ldots \wedge \psi_l$ be a conjunction of clauses, such that all occurrences of variables in ϕ_i are positive and all occurrences of variables in ψ_i are negative. We build a database with the facts $R(i, p, c)$ if the variable p occurs in the clause ψ_i and $R(i, p, c')$ if the variable p occurs in the clause ϕ_i. Now, there is an assignment which satisfies Φ if and only if there exists a repair of the database in which Q is false. To show the $\Rightarrow$ implication, select for each clause ϕ_i one variable p_i which occurs in this clause and whose value is 1 and for each clause ψ_i one variable p_i which occurs in ψ_i and whose value is 0. The set of facts $\{R(i, p_i, c) : i \leq m\} \cup \{R(i, p_i, c') : m + 1 \leq i \leq l\}$ is a repair in which the query Q is false. The $\Leftarrow$ implication is even simpler. ■

3.3 Two key dependencies, one query literal

By a *bipartite edge-colored graph* we mean a tuple $\mathcal{G} = \langle V, E, B, G\rangle$ such that $\langle V, E\rangle$ is an undirected bipartite graph and $E = B \cup G$ for some given disjoint sets B, G (so we think that each of the edges of $\mathcal{G}$ has one of the two colors).

Definition 6 *Let $\mathcal{G} = \langle V, E, B, G\rangle$ be a bipartite edge-colored graph, and let $F \subset E$. We say that F is maximal $\mathcal{V}$-free if:*

1. *F is a maximal (w.r.t. inclusion) subset of E with the property that neither $F(x, y) \wedge F(x, z)$ nor $F(x, y) \wedge F(z, y)$ holds for any x, y, z.*
2. *$F \cap B = \emptyset$.*

We say that $\mathcal{G}$ has the max-$\mathcal{V}$-free property if there exists F which is maximal $\mathcal{V}$-free.

Lemma 1 *Max-$\mathcal{V}$-free is an NP-complete property of bipartite edge-colored graphs.*

Proof. Reduction from 3-COLORABILITY. Let $\mathcal{H} = \langle U, D\rangle$ be some undirected graph. This is how we define the bipartite edge-colored graph $\mathcal{G}_\mathcal{H}$:

1. $V = \{v_\varepsilon, v'_\varepsilon : v \in U, \varepsilon \in \{m, n, r, g, b\}\}$, which means that there are 10 nodes in the graph $\mathcal{G}$ for each node of $\mathcal{H}$;
2. $G(v_m, v'_r), G(v_m, v'_b), G(v_n, v'_b), G(v_n, v'_g)$ and $G(v_r, v'_m)$,$G(v_b, v'_m)$,$G(v_b, v'_n)$,$G(v_g, v'_n)$ hold for each $v \in U$;
3. $B(v_\epsilon, v'_\varepsilon)$ holds for each $v \in U$ and each pair $\epsilon, \varepsilon \in \{r, g, b\}$ such that $\epsilon \neq \varepsilon$;

4. $B(v_\varepsilon, u'_\varepsilon)$ holds for each $\varepsilon \in \{r, g, b\}$ and each pair $u, v \in U$ such that $D(u, v)$.

Suppose that $\mathcal{H}$ is 3-colorable. We fix a coloring of $\mathcal{H}$ and construct the set F. For each $v \in U$: if the color of v is Red, then the edges $G(v_m, v'_b), G(v_n, v'_g)$ and $G(v_b, v'_m), G(v_g, v'_n)$ are in F. If color of v is Green, then the edges $G(v_m, v'_r), G(v_n, v'_b)$ and $G(v_r, v'_m), G(v_b, v'_n)$ are in F, and if the color of v is Blue, then the edges $G(v_m, v'_r), G(v_n, v'_g)$ and $G(v_r, v'_m), G(v_g, v'_n)$ are in F. It is easy to see that the set F constructed in this way is maximal $\mathcal{V}$-free.

For the other direction, suppose that a maximal $\mathcal{V}$-free set F exists in $\mathcal{G}_\mathcal{H}$. Then, for each $v \in U$ there is at least one node among v_r, v_g, v_b which does not belong to any G-edge in F. Let v_ϵ be this node. Also, there is at least one such node (say, v'_ε) among v'_r, v'_g, v'_b. Now, it follows easily from the construction of $\mathcal{G}_\mathcal{H}$ that if F is maximal $\mathcal{V}$-free then $\epsilon = \varepsilon$. Let this ϵ be color of v in $\mathcal{G}$. It is easy to check that the coloring defined in this way is a legal 3-coloring of $\mathcal{G}$. ■

Theorem 4 *There is a set F of 2 key dependencies and a query $Q \equiv \exists x, y[R(x, y, b)]$, for which $D_{F,Q}$ is co-NP-data-complete.*

Proof. The 2 dependencies are $A \to BC$ and $B \to AC$. For a given bipartite edge-colored graph $\mathcal{G} = \langle V, E, B, G \rangle$ we build a database with the tuples (x, y, g) if $G(x, y)$ holds in $\mathcal{G}$ and (x, y, b) if $B(x, y)$ holds in $\mathcal{G}$. Now the theorem follows from Lemma 1 since a repair in which the query $\exists x, y\ R(x, y, b)$ is not true exists if and only if $\mathcal{G}$ has the max-$\mathcal{V}$-free property. ■

3.4 One denial constraint

By an *edge-colored graph* we mean a tuple $\mathcal{G} = \langle V, E, P, G, B \rangle$ such that $\langle V, E \rangle$ is a (directed) graph and $E = P \cup G \cup B$ for some given pairwise disjoint sets P, G, B (which we interpret as colors). We say that the edge colored graph $\mathcal{G}$ has the $\mathcal{Y}$ property if there are $x, y, z, t \in E$ such that $E(x, y), E(y, z), E(y, t)$ hold and the edges $E(y, z)$ and $E(y, t)$ are of different colors.

Definition 7 *We say that the edge-colored graph $\langle V, E, P, G, B \rangle$ has the max-$\mathcal{Y}$-free property if there exists a subset F of E such that $F \cap P = \emptyset$ and :*

1. *$\langle V, F, P \cap F, G \cap F, B \cap F \rangle$ does not have the $\mathcal{Y}$-property;*

2. *F is a maximal (w.r.t. inclusion) subset of E satisfying the first condition;*

Lemma 2 *Max-$\mathcal{Y}$-free is an NP-complete property of edge-colored graphs.*

Proof. By a reduction of 3SAT. Let $\Phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_l$ be conjunction of clauses. Let $p_1, p_2, \ldots p_n$ be all the variables in Φ. This is how we define the *edge-colored graph* $\mathcal{G}_\Phi$:

1. $V = \{a_i, b_i, c_i, d_i : 1 \le i \le n\} \cup \{e_i, f_i, g_i : 1 \le i \le l\}$, which means that there are 3 nodes in the new graph for each clause in Φ and 4 nodes for each variable.

2. $P(a_i, b_i)$ and $P(e_j, f_j)$ hold for each suitable i, j;

3. $G(b_i, d_i)$ and $G(e_j, g_j)$ hold for each suitable i, j;

4. $B(b_i, c_i)$ holds for each suitable i;

5. $G(d_i, e_j)$ holds if p_i occurs positively in ϕ_j;

6. $B(d_i, e_j)$ holds if p_i occurs negatively in ϕ_j;

7. $E = B \cup G \cup P$.

Now suppose that Φ is satisfiable, and that μ is the satisfying assignment. We define the set $F \subset E$ as follows. We keep in F all the G-colored edges from item 3 above. If $\mu(p_i) = 1$ then we keep in F all the G edges leaving d_i (item 5). Otherwise we keep in F all the B edges leaving d_i (item 6). Obviously, $F \cap P = \emptyset$. It is also easy to see that F does not have the $\mathcal{Y}$-property and that it is maximal.

In the opposite direction, notice that if an F, as in Definition 7 does exist, then it must contain all the G-edges from item 2 above - otherwise a P edge could be added without leading to the $\mathcal{Y}$-property. But this means that, for each i, F can either contain some (or all) of the B-edges leaving d_i or some (or all) of the G-edges. In this sense F defines a valuation of variables. Also, if F is maximal, it must contain, for each j, at least one edge leading to e_j. But this means that the defined valuation satisfies Φ. ■

Theorem 5 *There exist a denial constraint f and a query of the form $Q \equiv \exists x, y[R(x, y, p)]$, for which $D_{\{f\},Q}$ is co-NP-data-complete.*

Proof. The denial constraint f is:

$$\forall x, y, z, s, s', s'' \ \neg[R(x, y, s) \wedge R(y, z.s') \wedge R(y, w, s'') \wedge s' \neq s'']$$

For a given edge-colored graph $\mathcal{G} = \langle V, E, P, G, B\rangle$ we build a database with the tuples $R(x, y, g)$ if $G(x, y)$ holds in $\mathcal{G}$, with $R(x, y, p)$ if $P(x, y)$ holds in $\mathcal{G}$ and with $R(x, y, b)$ if $B(x, y)$ holds in $\mathcal{G}$. Now the theorem follows from Lemma 2 since a repair in which the query Q is not true exists iff $\mathcal{G}$ has the max-$\mathcal{Y}$-free property. ■

References

[1] M. Arenas, L. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *ACM Symposium on Principles of Database Systems*, pages 68–79, 1999.

[2] M. Arenas, L. Bertossi, and J. Chomicki. Specifying and Querying Database Repairs Using Logic Programs with Exceptions. In *International Conference on Flexible Query Answering Systems*, pages 27–41. Springer-Verlag, 2000.

[3] M. Arenas, L. Bertossi, and J. Chomicki. Scalar Aggregation in FD-Inconsistent Databases. In *International Conference on Database Theory*, pages 39–53. Springer-Verlag, LNCS 1973, 2001.

[4] M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. Spinrad. Scalar Aggregation in Inconsistent Databases. *Theoretical Computer Science*, 2003. Special issue: selected papers from ICDT 2001, to appear.

[5] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-Generating Dependencies. *Journal of Computer and System Sciences*, 59:94–115, 1999. Preliminary version in ICDT'95.

[6] A. K. Chandra and D. Harel. Computable Queries for Relational Databases. *Journal of Computer and System Sciences*, 21:156–178, 1980.

[7] G. Greco, S. Greco, and E. Zumpano. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. In *International Conference on Logic Programming*, pages 348–364. Springer-Verlag, LNCS 2237, 2001.

[8] S. Greco and E. Zumpano. Querying Inconsistent Databases. In *International Conference on Logic for Programming and Automated Reasoning*, pages 308–325. Springer-Verlag, LNCS 1955, 2000.

[9] R. Hull. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. In *ACM Symposium on Principles of Database Systems*, pages 51–61, 1997. Invited talk.

[10] A. Motro. Multiplex: A Formal Model for Multidatabases and Its Implementation. In *International Workshop on Next Generation Information Technology and Systems*, pages 138–158. Springer-Verlag, LNCS 1649, 1999.

[11] M. Y. Vardi. The Complexity of Relational Query Languages. In *ACM Symposium on Theory of Computing*, pages 137–146, 1982.