On the Representation and Querying of Sets of Possible Worlds *

Serge Abiteboul † Paris Kanellakis ‡ Gösta Grahne §

July 1989

Abstract

We represent *a set of possible worlds* using an incomplete information database. The representation techniques that we study range from the very simple *Codd-table* (a relation over constants and uniquely occurring variables called nulls) to much more complex mechanisms involving *views of conditioned-tables* (programs applied to Codd-tables augmented by equality and inequality conditions). (1) We provide matching upper and lower bounds on the data-complexity of testing *containment*, *membership*, and *uniqueness* for sets of possible worlds. We fully classify these problems with respect to our representations. (2) We investigate the data-complexity of querying incomplete information databases for both *possible* and *certain* facts. For each fixed positive existential query on conditioned-tables we present a polynomial time algorithm solving the possible fact problem. We match this upper bound by two NP-completeness lower bounds, when the fixed query contains either negation or recursion and is applied to Codd-tables. Finally, we show that the certain fact problem is coNP-complete, even for a fixed first order query applied to a Codd-table.

1 Introduction

In order to extend the applicability of relational databases, one must provide a mechanism for representing incomplete information [Cod] (i.e., for representing sets of possible worlds). The most common such mechanism is *null values*. The basic idea is to replace some of the occurrences of constants in the relations of the database by variables, and thus model many possible worlds instead of a single one. Although null values are an algebraic tool, they do have close analogs in logical databases, e.g., [R, V2]. There already is a large volume of

*A preliminary version of this paper appeared in the proceedings of the ACM SIGMOD International Conference on the Management of Data, pp. 34–48, 1987.

†INRIA Rocquencourt, France. Work supported by the Projet de Recherche Coordonnée BD3.

‡Brown University, USA. Work performed while visiting INRIA Rocquencourt; partly supported by an IBM Faculty Development Award and by NSF grant IRI-8617344.

§University of Helsinki, Finland. Work performed while visiting INRIA Rocquencourt.

interesting work on querying incomplete information databases. The focus of most of this work has been a search for the "correct" semantics for query programs applied to incomplete information databases, e.g., [AG, Cod, G, I, IL, L, R, V2, Z]. We refer to [IL] for a detailed treatment of the topic.

On complete information databases, efficient evaluation is a fundamental property of many database query languages, such as, relational calculus ([U]) and *DATALOG* ([CH2]). More formally, all these languages express queries on complete information databases whose *data-complexity* is in *PTIME* [CH1, V1]. Data-complexity is defined to be the complexity of evaluating the answer as a function of the database size and *not* of the query program size, which is assumed to be a fixed parameter. This approach to computational complexity restricts the analysis by assuming fixed relation arities, i.e., fixed tuple widths. Clearly it leads to a reasonable measure, given that the number of tuples in a typical database dominates (by orders of magnitude) the tuple width and the size of an application program.

The subject of our paper is *a comprehensive data-complexity analysis of problems related to representing and querying databases with null values*. We only consider programs that express queries with *PTIME* data-complexity on complete information databases, since we believe that these are the only "feasible" programs in common database query languages. There has been some previous work on the data-complexity of querying incomplete information databases. The most significant contribution is [V2], which investigates the computational complexity of evaluating certain answers for a wide range of second-order queries on incomplete information databases. Our results refine and extend both [V2] and [IL].

A less realistic alternative (which we do not pursue here) is to let the query program size be part of the input size. Then the complexity of evaluation increases exponentially [Cos, V1]. This increase is due to a certain incompleteness of relational algebra with respect to the algebra of polynomials [Cos]. Such problems were first noted in [HLY, MSY], as part of the study of nulls in weak universal instances. Data-complexity avoids these anomalies, by factoring out the query program representation and maintaining only the combinatorics of the uncertainty in the database.

Section 2 contains a detailed definition of the framework. We now outline, in two parts, our results about representation (contained in Sections 3 and 4) and about querying (contained in Section 5). Section 6 contains conclusions and open questions.

1.1 Representation

Our representations form a hierarchy: from single possible worlds; to our simplest case of uncertainty, which is the Codd-table or table for short; to intermediate cases of uncertainty, which are Codd-tables with additional conditions; and finally to the most general case of uncertainty, views of Codd-tables with additional conditions. Let us describe this hierarchy using the example of Figure 1.

A Codd-table (table) is a relation with constants and variables, where no variable occurs twice. An inequality-table (i-table) is a table with a conjunction of inequalities;

0	1	x
y	z	1
2	0	v

table T_a

0	1	x
x	z	1
2	0	z

$e-table\ T_b$

$\boxed{x \neq 0, y \neq z}$

0	1	x
y	z	1
2	0	v

$i-table\ T_c$

$\boxed{x \neq z}$

0	1	x
x	z	1
2	0	z

$g-table\ T_d$

$\boxed{x \neq 1, y \neq 2}$

0	1	$z = z$
0	x	$y = 0$
y	x	$x \neq y$

$c-table\ T_e$

I_a

0	1	2
2	0	1
2	0	0

I_b

0	1	2
3	0	1
2	0	5

I_c

0	1	2
2	0	1
2	0	0

I_d

0	1
3	2

I_e

Figure 1: Representations of sets of instances and examples of corresponding instances

these are listed on top of the table. An equality-table (e-table) is a table with a conjunction of equalities; we do not list these on top but incorporate them directly in the table, this is standard practice. Combining the two we get a global-table (g-table) which is an e-table together with a conjunction of inequalities; these are listed on top of the e-table. A conditioned-table (c-table) is an extension of a g-table with one more column. This column contains the local conditions; where a local condition is a conjunction of inequalities and equalities attached to a tuple. The set of possible worlds represented by a c-table naturally results from instantiating the variables with constants and by satisfying the conditions. Finally, we have a view of a set of possible worlds, by applying a given query program to every possible world described by a c-table.

Remark: Tables are isomorphic to a very restricted form of the "logical databases" in [V2]. Our e-tables have also been called "V-tables" and "naive-tables" [AG, G, IL]. Our g-tables are similar constructs to the "logical databases" of [V2]. The c-tables are like the "C-tables" of [IL] augmented by one conjunction of equalities and inequalities, that is the global condition. The local conditions of both "C-tables" and our c-tables are conjunctions of equalities and inequalities. □

A central computational problem is the *containment* problem: "is a set of possible worlds a subset of another set of possible worlds?" A special case of this problem is the *membership* problem: "is a complete information database one of a set of possible worlds?" The (superficially) dual question about representations is the *uniqueness* problem: "is a set of possible worlds the same as a complete information database?"

Tables have an important computational property, that makes our choosing them as the simplest representation of uncertainty more meaningful. From a reduction to *bipartite*

	instance	table	e-table	i-table	g-table	c-table	view
instance	P	P	NP 3.1.2	NP 3.1.3	NP	NP	NP 3.1.4
table	P	P	NP	Π_2^p 4.2.1	Π_2^p	Π_2^p	Π_2^p 4.2.2
e-table	P	P	NP	Π_2^p	Π_2^p	Π_2^p	Π_2^p
i-table	P	P	NP	Π_2^p	Π_2^p	Π_2^p	Π_2^p
g-table	P	P	NP	Π_2^p	Π_2^p	Π_2^p	Π_2^p
c-table	$coNP$ 3.2.3	$coNP$	Π_2^p 4.2.3	Π_2^p	Π_2^p	Π_2^p	Π_2^p
view	$coNP$ 3.2.4	$coNP$ 4.2.4	Π_2^p 4.2.5	Π_2^p	Π_2^p	Π_2^p	Π_2^p

Figure 2: The complexity of the containment problem

matching it follows that membership is in polynomial-time for sets of worlds represented by tables (Theorem 3.1). This is a similarity with complete information databases and a distinction from the other, more complex, representations of uncertainty used here. It will be evident from our results that tables have "good" computational properties.

We determine the computational complexity of membership (Theorem 3.1), uniqueness (Theorem 3.2) and containment (Theorems 3.1, 3.2, 4.1, 4.2) with respect to our hierarchy. For this complete classification we use homomorphism techniques from database theory and logspace-reductions from computational complexity. We use the standard complexity classes *PTIME* (polynomial time) and $NP=\Sigma_1^p$, $coNP=\Pi_1^p$, Σ_2^p, Π_2^p (from the first two levels of the polynomial time hierarchy [S], [GJ]).

Let us explain our results using Figure 2. For the containment problem we have 49 cases. These cases depend on a choice among 7 kinds of representation for each dimension of this figure. The vertical dimension is the set of worlds tested for containment in the set of worlds of the horizontal dimension. The seven kinds are as follows:

- a complete information database (instance in Figure 2)
- x-tables, where x is in {(Codd),e,i,g,c} (x-table in Figure 2)
- a program applied to x-tables (view in Figure 2)
  (here the upper bounds are the same for all x's and the lower bounds hold for tables)

In every one of the cases of Figure 2 we provide the upper bounds; these are the areas for *PTIME* (or *P*), *NP*, *coNP* and Π_2^p. These upper bounds follow from Proposition 2.1 parts (1,2,3), Theorem 3.1 part (1), Theorem 3.2 part (1), and Theorem 4.1 parts (1,2,3). Note the differences between membership and uniqueness.

All cases not in the *PTIME* area are shown complete in their respective classes. They are all the cases "strictly" in *NP*, *coNP*, Π_2^p. For this it suffices to show hardness for the cases in Figure 2 that include references to theorems. It is easy to see that the other hardness results follow trivially. These lower bounds are Theorem 3.1 parts (2,3,4), Theorem 3.2 parts (3,4), Theorem 4.2 parts (1,2,3,4,5).

Perhaps, the most interesting lower bound is Theorem 4.2 part (1): "containment is Π_2^p-complete, even if the subset possible worlds are represented by a table and the superset possible worlds are represented by an i-table". We highlight that the highest complexity is reached with a very small amount of expressibility.

In a sense, our lower bounds are syntactically tight. For all the lower bounds with views we use positive existential queries applied to tables. There is one exception. As shown in Theorem 3.2 part (2), for positive existential queries on e-tables the uniqueness problem is in polynomial-time. Thus, the exception is the query for Theorem 3.2 part (4), which is positive existential with $\neq$ applied to a table.

1.2 Querying

The view mechanism for specifying sets of possible worlds is a natural step towards querying an incomplete information database. In this context, a first question is the *possibility* problem: "given a set of tuples and given a set of possible worlds (specified as a view) is there a possible world where these tuples are all true?" A second question is the *certainty* problem: "given a set of tuples and given a set of possible worlds (specified as a view) are these tuples all true in every possible world?" Note that, certainty implies possibility and that, certainty and $\neg$(certainty) are different from possibility and $\neg$(possibility).

Unbounded Possibility: There are similarities between the possibility and the membership problems, because the size of the given set of tuples for possibility can be of the same order of magnitude as a possible world. The difference of course is that membership requires the exact equality with a possible world. If we do not restrict the size of the given set of tuples we have the *unbounded* possibility problem (Theorem 5.1) which is clearly computationally related to membership (Theorem 3.1).

Bounded Possibility: If we restrict the size of the set of given tuples for possibility we have the *bounded* possibility problem. This problem seems more meaningful than unbounded possibility, because intuitively it corresponds to the practical question: "is this (small) list of facts even possible?" Our analysis of the bounded possibility problem complements the literature, where much more attention has been given to the certainty problem. Note that, for certainty the unbounded and bounded versions of the problem are polynomial-time equivalent (Proposition 2.1). Our algorithm for bounded possibility uses the algebraic completeness of conditioned-tables demonstrated in [IL]. In Theorem 5.2 part (1), we show that the data-complexity of positive existential queries on conditioned-tables is in polynomial-time. Our lower bounds on possibility are also new and illustrate the effect both of "negation" and of "recursion" on data-complexity. Namely, we extend positive existential queries in two standard ways. One extension is the first order queries and the other is the *DATALOG* queries. Both extensions lead to *NP*-completeness even if the conditioned-tables are tables, see Theorem 5.2 parts (2,3).

Certainty: There are two main observations in the literature on certainty. In its various forms, the first observation follows from central results of [IL] (based on "C-tables") and [R, V2] (based on "logical databases"). Namely, under particular syntactic restrictions on conditioned-tables and using positive queries, the certainty question can be handled exactly as if one had a complete information database. In our framework the syntactic restrictions are g-tables and the positive queries are the *DATALOG* queries. This leads to Theorem 5.3 part (1), which we only list for completeness of presentation, since it is due to [IL, V2]. The second observation deals with the negative effects of the many possible instantiations of the null values. In [V2] the certainty question for a fixed first order query on an i-table is shown *coNP*-complete. We extend this result to a first order query on a table, see Theorem 5.3 part (2).

2 Definitions and Notation

2.1 Complete Information Databases

Let $\mathcal{D}$ be a countably infinite set of *constants.* A *relation* R of *arity* (a) is some *finite* subset of the $\mathcal{D}^a$, where a is a nonnegative integer. A *complete information database* or *instance* I of *arity* $(a_1, ..., a_n)$ is a n-vector of relations $(R_1, ..., R_n)$, such that, relation R_i has arity (a_i) $i = 1, ..., n$. A tuple belonging to a relation is called a *fact.* We assume a fixed binary encoding for facts and instances. When we say that fact t is in instance I we assume that the relation of I, where t belongs, is also specified.

A *query* q of *arity* $(a_1, ..., a_n) \rightarrow (b_1, ..., b_m)$ is a function from instances to instances of appropriate arities. Thus, a query q and an instance I define another instance $q(I)$ called the q *view* of I. One example of a query of arity $(a_1, ..., a_n) \rightarrow (a_1, ..., a_n)$ is the *identity* function of this arity; we sometimes use the symbol - to denote it, e.g., we use *MEMB*(-) instead of *MEMB*(identity). Given a query q we say that the *data-complexity* of q is the complexity of the formal language:

$$L_q = \{(t, I) \mid \text{fact } t \text{ is in instance } q(I)\ \}$$

The family of queries examined in this paper is *QPTIME*, namely the *computable database queries* of [CH1] that have *PTIME* data-complexity. (For the detailed definition we refer to [CH1]). In addition to their low data-complexity, these queries satisfy an important *genericity* property: "for all bijections ρ on $\mathcal{D}$ we have $q(\rho(I))=\rho(q(I))$, where ρ is extended to instances in the natural fashion". This condition says that isomorphic inputs are mapped to isomorphic outputs and therefore the representation details, e.g., the encoding of facts, should not matter.

We refer to three commonly used subfamilies. The *positive existential* queries are the simplest, most practical, and most investigated queries. They are expressed exactly using relational expressions with operators *project, natural join, union, renaming, positive select,* see [U]. They can be extended through "negation" to the *first order* queries, see [CH1, U], or (incomparably) through "recursion" to the *DATALOG* queries, see [CH2]. (1) Positive existential queries are denoted here using first order formulas with equality, but without universal quantification or negation, i.e., $\neq$ cannot be used. (2) First order queries are denoted here using formulas of first order logic with equality, i.e., $\neq$ can be used. (3) *DATALOG* queries are denoted here using fixpoints of positive existential queries, i.e., we only use "pure" *DATALOG* queries without $\neq$.

2.2 Incomplete Information Databases

An *incomplete information database* is a set of instances. A central issue for such sets of instances is their representation, which we now describe. First, assume that the set of variables $\mathcal{V}$ and the set of constants $\mathcal{D}$ are disjoint.

A *table* (short for Codd-table) T of *arity* (a) is the result of replacing some occurrences of constants in a relation of arity (a) by *distinct* variables, i.e., each variable occurs at most once. A *tuple* t of a table T is a row of T. (See Figure 1(a).)

A *condition* is a *conjunct* of *equality atoms* of the form $x = y, x = c$ and of *inequality atoms* of the form $x \neq y, x \neq c$, where the x's and y's are variables and the c's are constants. Note that we only use conjuncts of atoms and that the boolean *true* and *false* can be respectively encoded as atoms $x{=}x$ and $x \neq x$. Conditions may be associated with table T in two ways: (1) a *global* condition Φ_T is associated with the entire table T; (2) a *local* condition Φ_t is associated with one tuple t of table T. Note that, conditions associated with table T and tuple t may contain variables not appearing in T or t.

A *valuation* σ is a function from variables and constants to constants, such that $\sigma(c) = c$ for each constant c. A valuation σ naturally extends to a tuple t producing a fact $\sigma(t)$ and to a table T producing a relation $\sigma(T)$. If formulas Φ_T, Φ_t are conditions we say that σ *satisfies* Φ_T, Φ_t if its assignment of constants to variables makes these formulas true.

A *c-table* (short for conditioned-table) is a table T together with an associated global condition Φ_T and an associated local condition Φ_t for each tuple t of T. (See Figure 1 (e).) By convention, if we omit listing a condition then it is atom *true*. A *g-table* (short for global table) is a c-table without local conditions. (See Figure 1 (d).) An *i-table* (short for inequality table) is a g-table, whose global condition consists entirely of inequality atoms. (See Figure 1 (c).) An *e-table* (short for equality table) is a g-table, whose global condition consists entirely of equality atoms. (See Figure 1 (b).)

Definition $\mathcal{I}$: A given c-table represents a set of instances $\mathcal{I}$. If this c-table consists of a table T of arity (a), a global condition Φ_T, and local conditions Φ_t for each tuple t in T then it represents the following set of instances of arity (a):

$\mathcal{I}$={ R | there is a valuation σ satisfying Φ_T such that relation R consists exactly of those facts $\sigma(t)$ for which σ satisfies Φ_t }.

Example 2.1 Consider the valuation σ: $\sigma x = 2, \sigma y = 3, \sigma z = 0, \sigma v = 5$. Let α be in $\{a, b, c, d, e\}$ and T_α, I_α be like in Figure 1. Then $I_\alpha = \sigma T_\alpha$. □

Specializing the above definition we have: (1) for a table T, all valuations are satisfying and $\mathcal{I}$={ R | R=$\sigma(T)$ for some σ }, (2) for a g-table (T, Φ_T), $\mathcal{I}$={ R | $R = \sigma$(T) for some σ satisfying Φ_T }. For a c-table, $\mathcal{I}$ is the *empty set* iff the global condition is unsatisfiable. This can be checked in *PTIME* because a global condition is a conjunction. For a c-table, $\mathcal{I}$ consists of a relation with only the *empty fact* of arity (a) iff there are satisfying valuations for the global condition but these valuations do not satisfy any local condition. This can also be checked in *PTIME*, because all one has to do is check a formula in disjunctive normal form for unsatisfiability.

The above definitions easily generalize to n-vectors of c-tables, as opposed to 1-vectors, and $\mathcal{I}$'s of arity $(a_1, ..., a_n)$, as opposed to arity (a). For this generalization, we assume that the sets of variables appearing in each table $T_1, ..., T_n$ are *pairwise disjoint*; relationships between these variables can be established through other variables in the global and local conditions. We close with our most general representation of a set of instances, which is a *set of views* of $\mathcal{I}$ through q.

Definition $q(\mathcal{I})$: Let $\mathcal{I}$ be defined using an n-vector of c-tables of arity $(a_1, ..., a_n)$ and let q be a *QPTIME* query of arity $(a_1, ..., a_n) \rightarrow (b_1, ..., b_m)$, then $q(\mathcal{I})$ is the following set of instances of arity $(b_1, ..., b_m)$:

$$q(\mathcal{I}) = \{q(I) \mid I \text{ instance in } \mathcal{I} \}.$$

We sometimes use the notation *rep*(T) for the set of instances represented by table T, *rep*(T,Φ_T) for the set of instances represented by g-table (T,Φ_T) etc.

2.3 The Problems

We list some basic computational questions about incomplete information databases. All of these questions can be answered in $PTIME$ for complete information databases.

Our notation matches the definition of data-complexity. Tables and conditions are the parts of the inputs that contribute to asymptotic growth, i.e., they are unbounded, for this we use capital letters (e.g., T, Φ_T, Φ_t). We also use capital letters for sets of facts (e.g., R, I, $\mathcal{I}$, P) which can be of unbounded size. In our framework queries, and therefore arities and tuple width, are *fixed* parameters, for this we use small letters (e.g., q, a, b, t).

$CONT(q_0, q)$ (the containment problem)
parameter: q_0, q
input: c-tables representing $\mathcal{I}_0$; c-tables representing $\mathcal{I}$
question: $q_0(\mathcal{I}_0) \subseteq q(\mathcal{I})$?

$MEMB(q)$ (the membership problem)
parameter: q
input: instance I_0; c-tables representing $\mathcal{I}$
question: is I_0 in the set $q(\mathcal{I})$?

$UNIQ(q_0)$ (the uniqueness problem)
parameter: q_0
input: c-tables representing $\mathcal{I}_0$; instance I
question: is $q_0(\mathcal{I}_0)$ the singleton set $\{I\}$?

$POSS(k, q)$ (the possibility problem)
parameter: k, q
input: c-tables representing $\mathcal{I}$; set of facts P of size k
question: $\exists$I$\in q(\mathcal{I})$ such that all facts of P are facts of I ?
$POSS(*, q)$ is the same question where k is no longer a parameter.

$CERT(k, q)$ (the certainty problem)
parameter: k, q
input: c-tables representing $\mathcal{I}$; set of facts P of size k
question: $\forall$I$\in q(\mathcal{I})$ are all facts of P facts of I?
$CERT(*, q)$ is the same question where k is no longer a parameter.

Remark: Note that the membership problem is a special case of the containment problem, i.e., a containment where q_0 is the identity and $\mathcal{I}_0$ is completely specified. The uniqueness problem can be reduced to a membership ($I \in q_0(\mathcal{I}_0)$) together with a particular containment ($q_0(\mathcal{I}_0) \subseteq \{I\}$). On the other hand, the possibility and certainty problems cannot be reduced to the containment problem.

The crucial difference between complete and incomplete information is the large number of possible valuations for the latter case. Because of the finite number of variables in a set of c-tables only a finite number of valuations are nonisomorphic. However, the number of such valuations may grow exponentially in the input size. By simple reasoning about all valuations and guessing particular valuations, we have some easy upper bounds.

Proposition 2.1: For any *QPTIME* queries q_0, q we the following:
(1) $CONT(q_0, q)$ is in Π_2^p; (2) $MEMB(q)$ is in NP; (3) $UNIQ(q_0)$ is in $coNP$;
(4) $POSS(*, q)$ is in NP; (5) $CERT(*, q)$ is in $coNP$; and
(6) $CERT(*, q)$ is polynomial time equivalent to $CERT(1, q)$.

Proof: First we consider only a finite set of valuations. Let Δ and X be, respectively, the set of constants and variables that appear in the c-table inputs of *CONT*. Let Δ' be a set of constants distinct from Δ, with the same cardinality as X. The following observation is obvious:

"For every valuation σ there is a valuation σ' with values in $\Delta \cup \Delta'$ and a bijection ρ on the set of all constants, such that, σ and $\rho \circ \sigma'$ satisfy the same conditions of the input c-tables and if applied to them produce the same instances."

(1) $CONT(q_0, q)$ can be reduced to: for each valuation σ_0 with values in $\Delta \cup \Delta'$, there exists a valuation σ with values in $\Delta \cup \Delta'$ such that a polynomial time computable condition is true; $\forall\exists$ quantification. The reduction easily follows from the previous observation and from the fact that *QPTIME* queries satisfy the genericity condition: "for all bijections ρ on $\mathcal{D}$ we have $q(\rho(I))$=$\rho(q(I))$" (see Section 2.1).

In cases (2-3-4-5), the same argument also allows to restrict our attention to valuations with values in $\Delta \cup \Delta'$. For (2) and (4), we guess the right valuation; $\exists$ quantification. For (3) and (5), we reason about all valuations; $\forall$ quantification. To see one case in more details, consider (2). $MEMB(q)$ can be reduced to: there exists a valuation σ with values in $\Delta \cup \Delta'$ such that a polynomial time computable condition is true; $\exists$ quantification.

(6) In order to answer $CERT(k, q)$, all we have to do is repeat $CERT(1, q)$ k times. Note that this last argument does not hold for $POSS(k, q)$, because $POSS(1, k)$ might return "yes", but each "yes" might refer to a different possible instance. $\Box$

3 Membership and Uniqueness

In this section we determine the computational complexity of the membership (Theorem 3.1) and uniqueness (Theorem 3.2) problems. Tables have a polynomial time membership problem. This is like instances and unlike e-tables, i-tables, or views of tables. Despite some symmetry in their definitions membership and uniqueness are quite different. Also, note the role of $\neq$ in Theorem 3.2.

Theorem 3.1: Let $\mathcal{I}$ be as in the definition of $MEMB$, then:
(1) $MEMB(\text{-})$ is in $PTIME$ if $\mathcal{I}$ is represented by tables.
(2) $MEMB(\text{-})$ is NP-complete, even if $\mathcal{I}$ is represented by a single e-table.
(3) $MEMB(\text{-})$ is NP-complete, even if $\mathcal{I}$ is represented by a single i-table.
(4) $\exists$ positive existential query q such that $MEMB(q)$ is NP-complete, even if $\mathcal{I}$ is represented by a single table.

Proof: The upper bound is derived by a reduction to the problem of bipartite graph matching, which is known to be in *PTIME*. Critical use is made of the fact that all occurrences of variables are distinct symbols. Given that the membership problem in general is in NP (by Proposition 2.1) the rest of the proof consists of reductions of NP-hard problems to the *MEMB* problem.

(1) Reduction to bipartite graph matching: Given undirected graph $G = (V, E)$, with nodes V and edges E, a matching E' in G is a subset of E such that no two edges of E' are incident on the same node. The (*bipartite*) *matching problem* is: given (bipartite) graph G find a maximum cardinality matching in it.

We prove the result for a single table. The case of a vector of tables is similar. The following algorithm tests whether, given I_0 and T, I_0 is in $rep(T)$.

input: an instance $I_0 = \{u_i \mid i \in [1..n]\}$ and a table $T = \{v_j \mid j \in [1..m]\}$ of the same arity.
output: yes iff I_0 is in $rep(T)$.
begin
a. let $V = \{a_i \mid i \in [1..n]\} \cup \{b_j \mid j \in [1..m]\}$ be the union of two disjoint sets of nodes;
b. let $E = \{(a_i, b_j) \mid u_i \in I_0, v_j \in T$ and there is a valuation σ s.t. $u_i = \sigma(v_j)\}$;
c. if for some j, b_j is not connected to any a_i then **return no**;
d. compute a maximum cardinality matching E' for the bipartite graph (V, E);

	(a)				(b)

I_0

1	1	2
3	2	3
1	4	5
1	2	3

T

x_1	1	x_2
x_3	2	3
1	x_4	x_5
1	2	3
1	2	x_6

G

a_1	b_1
a_1	b_3
a_2	b_2
a_3	b_3
a_4	b_2
a_4	b_3
a_4	b_4
a_4	b_5

Figure 3: Example for 3.1.1

e. if $\#(E')= n$ then **return yes** else **return no**;
end

For I_0 and T given in Figure 3.a, the construction of the graph is illustrated in Figure 3.b. G in Figure 3.b is a binary relation that contains the edges (a_i, b_j) of the bipartite graph. The algorithm is clearly in *PTIME* so it suffices to show that it is correct.

First suppose that I_0 is in $rep(T)$. Then there is a valuation σ such that $I_0 = \sigma(T)$. For each i in $[1..n]$, let $j(i)$ be such that $\sigma(v_{j(i)}) = u_i$. Such $j(i)$, distinct for each i, exist since $I_0 \subseteq \sigma(T)$. Since $\sigma(T) = I_0$, step (d) is clearly reached. Let $E' = \{(a_i, b_{j(i)}) \mid i \in [1..n]\}$. Clearly E' is a matching, and since $\#(E') = n$, E' is maximum cardinality and *yes* is returned.

Conversely, suppose that *yes* is returned. Consider a maximum cardinality matching E'. Let j be in $[1..m]$. If (a_i, b_j) is in E' for some i, let $i(j) = i$. Otherwise, let $i(j)$ be such that $(a_{i(j)}, b_j)$ is in E. Such an $i(j)$ exists because of step (c). Let σ be a valuation such that $\sigma(v_j) = u_{i(j)}$ for each j. Such a valuation exists since for each j, $(a_{i(j)}, b_j) \in E$ and all variable occurrences are distinct. By construction, $\sigma(T) \subseteq I_0$. Since $\#(E')=n$, for each i, there exists j such that $(a_i, b_j) \in E'$, so $u_i = \sigma(v_j)$. Hence $I_0 \subseteq \sigma(T)$. Thus $\sigma(T) = I_0$.

(2) Reduction of graph 3-colorability: A known NP-complete problem [GJ] is the *graph 3-colorability* problem: given an undirected graph, does there exist a function from its nodes to the set of colors $\{1, 2, 3\}$ such that no edge has nodes mapped to the same color.

For each instance $G = (V, E)$ of the graph 3-colorability problem, an e-table T (here the equalities are directly put in the table) and an instance I_0 of arity 2 are constructed in the following way. Pick an arbitrary orientation of the edges of G, let $\{x_a \mid a \in V\}$ be a set of distinct variables, and then,

a. let $T = \{ij \mid i, j \in 1, 2, 3, i \neq j\} \cup \{x_a x_b \mid (a, b) \in E\}$,

b. let $I_0 = \{ij \mid i, j \in 1, 2, 3, i \neq j\}$.

For example, consider the instance of the 3-colorability problem given in Figure 4.a. The nodes are $1 \ldots 5$ and the edges are listed in a binary relation, each with an arbitrary orientation. The corresponding instance of the membership problem for e-tables is exhibited in Figure 4.c. It is easy to see that: G is 3-colorable iff I_0 is in $rep(T)$.

(3) Reduction of graph 3-colorability. For each instance $G = (V, E)$ of the graph 3-colorability problem, an i-table (T, Φ_T) and an instance I_0 of arity 1 are constructed in the following way. Pick an arbitrary orientation of the edges of G, and $\{x_a \mid a \in V\}$ a set of distinct variables, and then,

a. let $T = \{1, 2, 3\} \cup \{x_a \mid a \in V\}$, with $\Phi_T = \{x_a \neq x_b \mid (a, b) \in E\}$,

b. let $I_0 = \{1, 2, 3\}$.

For example, consider the instance of the 3-colorability problem given in Figure 4.a; the corresponding instance of the membership problem for i-tables is exhibited in Figure 4.b. It is easy to see that using the above reduction: G is 3-colorable iff I_0 is in $rep(T, \Phi_T)$.

(4) Reduction of graph 3-colorability: For reasons of clarity of exposition we use databases with two relations. A simple modification of the proof suffices to show the result for databases with one relation. Namely, by increasing the arity of the largest relation and using constants it is possible to simulate many relations by one.

For each input $G = (V, E)$ to the graph 3-colorability problem, we construct two tables $T(R)$ of arity (5) and $T(S)$ of arity (2), an instance I_0 of arity (3,1), and a positive existential query q of arity $(5, 2) \to (3, 1)$. In q the relation symbols R and S are names for the arity (5) and the arity (2) relations, that are instances of the two tables. (Without loss of generality assume that G has no self-loops and that E is a binary relation, where we list each edge once with an arbitrary orientation).

Let $V = \{a_i \mid i \in [1..n]\}$ and $E = \{(b_j, c_j) \mid j \in [1..m]\}$. Let $\{x_j \mid j \in [1..m]\}$ and $\{y_j \mid j \in [1..m]\}$ be two disjoint sets of distinct variables. Then $< T(R), T(S) >$, I_0 and $q =< q_1, q_2 >$ are constructed as follows.

a. $T(R) = \{t_j \mid j \in [1..m]\}$ where t_j is the tuple $b_j x_j c_j y_j j$.

b. $T(S) = \{ij \mid i, j \in \{1, 2, 3\}, i \neq j\}$.

c. I_0 consists of $R_0 = \{ajk \mid a \in \{b_j, c_j\} \cap \{b_k, c_k\}$ where each (b, c) pair is an edge in E $\}$ and $S_0 = \{j \mid j \in [1..m]\}$.

d. $q_1 = \{xzz' \mid \exists y([\exists vw(R(xyvwz) \vee R(vwxyz))] \wedge [\exists vw(R(xyvwz') \vee R(vwxyz'))])\}$
$q_2 = \{z \mid \exists xyvw(R(xyvwz) \wedge S(yw))\}$.

Intuitively, for each tuple in R, the second-column (respectively, fourth-column) contains the color of the vertex in the first-column (respectively, third-column). Query q_2

(a)

1	2
2	3
3	4
4	1
3	5

(b)

T
$\boxed{\Phi_T}$
1
2
3
x_1
x_2
x_3
x_4
x_5

I_0
1
2
3

$\Phi_T = \{x_1 \neq x_2,$
$x_2 \neq x_3, x_3 \neq x_4,$
$x_4 \neq x_1, x_3 \neq x_5\}$

(c)

T	
1	2
1	3
2	1
2	3
3	1
3	2
x_1	x_2
x_2	x_3
x_3	x_4
x_4	x_1
x_3	x_5

I_0	
1	2
1	3
2	1
2	3
3	1
3	2

(d)

T

$T(R)$				
1	x_1	2	y_1	1
2	x_2	3	y_2	2
3	x_3	4	y_3	3
4	x_4	1	y_4	4
3	x_5	5	y_5	5

$T(S)$	
1	2
1	3
2	1
2	3
3	1
3	2

I_0

R_0		
1	1	1
1	1	4
1	4	1
1	4	4
2	1	1
2	1	2
2	2	1
2	2	2
	...	
4	3	3
4	3	4
4	4	3
4	4	4
5	5	5

S_0
1
2
3
4
5

Figure 4: (a) Example graph for 3-colorability, (b) Example for 3.1.3, (c) Example for 3.1.2, (d) Example for 3.1.4

checks whether this is assignment of the three colors $\{1,2,3\}$, and q_1 checks whether a vertex is assigned the same color in all places of the relation.

For example, consider the instance of the 3-colorability problem given in Figure 4.a; the corresponding instance of the view-membership problem is exhibited in Figure 4.d. Suppose that f is a 3-coloring of G. Consider the valuation σ defined by: for each j, $\sigma(x_j) = f(b_j)$ and $\sigma(y_j) = f(c_j)$. It is easily seen that $I_0 = q(\sigma(T))$. Indeed, it is straightforward to argue that: G is 3-colorable iff I_0 is in $q(rep(T))$. □

Theorem 3.2: Let $\mathcal{I}_0$ be as in the definition of $UNIQ$, then:
(1) *UNIQ*(-) is in *PTIME* if $\mathcal{I}_0$ is represented by g-tables.
(2) *UNIQ*(q_0) is in *PTIME* if q_0 is pos. exist. and $\mathcal{I}_0$ is represented by e-tables.
(3) *UNIQ*(-) is $coNP$-complete, even if $\mathcal{I}_0$ is represented by a single c-table.
(4) $\exists$ positive existential with $\neq$ query q_0 such that *UNIQ*(q_0) is $coNP$-complete, even if $\mathcal{I}_0$ is represented by a single table.

Proof:
(1) Assume that: if it follows from the global condition that a variable equals a constant, then the variable is replaced by that constant in the table. This simplification can be performed in *PTIME*. Then we have that: $rep(T, \Phi_T) = I$ iff (a) Φ_T is satisfiable, and (b) $T = I$. Both (a) and (b) can be tested in *PTIME*.

(2) For this we use the technique of [IL] to get a representation of all possible worlds resulting from the query q_0. This representation is constructed and because of lack of negation can be tested trivially for uniqueness. More precisely, use the algorithm:

begin
a. compute c-table $(T, \Phi_T, \{\Phi_t\})$ equivalent to $q_0(T_0)$ as in [IL] (see $*$ below);
b. for each t in T, let T_t be the e-table defined by $T_t = I \vee \{t\}$;
c. for each t, compute a disjunctive normal form equivalent to Φ_t, say $\Phi_{t,1} \vee \Phi_{t,2}$...;
d. $\Phi_{t,i}$ is conjunction of equalities and can be incorporated in T_t to obtain e-table $T_{t,i}$;
e. consider the set $\{T_{t,i} \mid t \in T, i\}$ of e-tables and **return yes** iff
(α) for each u in I and each valuation σ_0, u is in $q_0(\sigma_0(T_0))$ and
(β) for each t and i, $rep(T_{t,i}) = \{I\}$;
end

($*$) First note that, in step (a), the global condition Φ_T = true, and that the local conditions $\{\Phi_t\}$ contain no negation since q_0 is positive existential. In this step the local conditions are kept as formulas with both or's and and's. These formulas are put in normal form in step (c).

To prove (2), we show that the algorithm is correct, and is in *PTIME*.

Proof of Correctness: Suppose (α) and (β) hold. Let σ be a valuation. By (α), we have that $I \subseteq q_0(\sigma(T_0))$. Since $rep(q_0(T_0)) = rep(T, \Phi_T, \{\Phi_t\})$, $q_0(\sigma(T_0)) = \sigma'(T, \Phi_T, \{\Phi_t\})$ for some valuation σ'. Let u be in $q_0(\sigma(T_0)) = \sigma'(T, \Phi_T, \{\Phi_t\})$. Then $u = \sigma'(t)$ for some t in T such that $\sigma'(\Phi_t)$ is true. Thus $u{=}\sigma'(t)$ for some t in T, and i, such that $\sigma'(\Phi_{t,i})$ is true. Clearly, u is in $\sigma'(T_{t,i})$. By (β), u is in I. Hence $q_0(\sigma(T_0)) \subseteq I$ so $I = q_0(\sigma(T_0))$.

$3CNF \wedge \{c_i\}$	$3DNF \vee \{c_i\}$
$c_1 = x_1 \vee x_2 \vee x_3,$	$c_1 = x_1 \wedge x_2 \wedge x_3$
$c_2 = x_1 \vee \neg x_2 \vee x_4,$	$c_2 = x_1 \wedge \neg x_2 \wedge x_4,$
$c_3 = x_1 \vee x_4 \vee x_5,$	$c_3 = x_1 \wedge x_4 \wedge x_5$
$c_4 = x_2 \vee \neg x_1 \vee x_5,$	$c_4 = x_2 \wedge \neg x_1 \wedge x_5,$
$c_5 = \neg x_1 \vee \neg x_2 \vee \neg x_5$	$c_5 = \neg x_1 \wedge \neg x_2 \wedge \neg x_5$

Figure 5: Example formulas. For ∀∃3CNF $X = \{x_1, x_2\}, Y = \{x_3, x_4, x_5\}$

Now suppose that $rep(q_0(T_0)) = \{I\}$. Clearly (α) holds. Suppose that for some t and i, $rep(T_{t,i})$ is not $\{I\}$. Let σ be a valuation such that $\sigma(T_{t,i})$ is not I. By this and by step (b), $\sigma(\Phi_{t,i})$ is true, and $\sigma(t)$ is not in I. Thus $\sigma(\Phi_t)$ is true. But then, $\sigma(t)$ is in I, because $\{I\} = rep(q_0(T_0)) = rep(T, \Phi_T, \{\Phi_t\})$, a contradiction. Thus (β) holds.

Proof of Efficiency: By [IL], (a) can be done in *PTIME.* Clearly, (b) and (d) can be done in *PTIME.* Using the c-table construction of [IL], the local conditions generated by the bounded size q_0 are of bounded size. Thus (c) can also be done in *PTIME.* Since there is no global condition and no negation in the local conditions, (α) can be tested in *PTIME* by the algorithm in [IL] for certain answers. Finally, (β) can be tested in *PTIME* by Theorem 3.2 part(1). □

(3) Reduction of 3DNF tautology: We first state the $coNP$-complete problem of 3DNF tautology (see Figure 5 for example 3CNF/3DNF formulas).
input: A set X of variables and $H = \{c_{ik} \mid i \in [1..n], k \in [1..3]\}$ such that for each i,k, c_{ik} is x or $\neg x$ for some x in X.
question: is $\vee_i\ [\wedge_k c_{ik}]$ true for each truth assignment of X?

For each instance H of the 3DNF tautology problem, a c-table $(T_0, \Phi_{T0}, \{\Phi_t\})$, and an instance I with a unary relation are constructed. Let $X = \{x_j \mid j \in [1..m]\}$, and let $\{u_j \mid j \in [1..m]\}$ be a set of distinct variables.

a. $T_0 = \{t(i) \mid i \in [1..n]\}$ where $t(i)$ has local condition $\Phi_{t(i)}$. For each i in $[1..n]$, unary tuple $t(i) = 1$ and $\Phi_{t(i)} = \delta_{i,1} \wedge \delta_{i,2} \wedge \delta_{i,3}$ where for each k in $\{1, 2, 3\}$, $\delta_{i,k} \equiv (u_j = 1)$ if $c_{ik} = x_j$, and $\delta_{i,k} \equiv (u_j \neq 1)$ if $c_{ik} = \neg x_j$.

b. Φ_{T0}= true.

c. $I = \{1\}$.

Clearly, if H is satisfiable, I is a representative of $(T_0, \Phi_{T0}, \{\Phi_t\})$. It is easily seen that if H is not a tautology, then the empty instance is also a representative. Indeed, we have: H is a tautology iff I is the unique representative of $(T_0, \Phi_{T0}, \{\Phi_t\})$.

(4) Reduction of graph non 3-colorability: For each instance $G = (V, E)$ of the graph non 3-colorability problem, a table T_0 is constructed in the following way:

T_0		
1	1	2
1	2	3
1	3	4
1	4	1
1	3	5
0	1	x_1
0	2	x_2
0	3	x_3
0	4	x_4
0	5	x_5

Figure 6: Example for 3.2.4

$$T_0 = \{1ab \mid (a,b) \in E\} \cup \{0ax_a \mid a \in V\}$$

where $\{x_a \mid a \in V\}$ is a set of distinct variables. Consider the query:
$q_0 = \{1 \mid \exists xyz[R(1xy) \wedge R(0xz) \wedge R(0yz)] \vee \exists yz[R(0yz) \wedge z \neq 1 \wedge z \neq 2 \wedge z \neq 3]\}$

The table corresponding to the graph of Figure 4.a is shown in Figure 6.

We may assume without loss of generality that G is not the empty graph. Consider the valuation σ_0 which assigns the value 4 to each variable. Then $\{1\} = q_0(\sigma_0(T_0))$. Now suppose that f is a 3-coloring of G using colors in $\{1,2,3\}$. Consider the valuation σ defined by: for each a, $\sigma(x_a) = f(a)$. Clearly, $\{\} = q_0(\sigma(T_0))$. Indeed, we have: G is not 3-colorable iff $\{1\}$ is the unique instance in $rep(q_0(T_0)$. □

4 Containment

For our upper bounds (Theorem 4.1) we use homomorphisms to refine Proposition 2.1. Our lower bounds (Theorem 4.2) together with the results of Section 3 exhaustively cover all cases of Figure 2. It is interesting to contrast Theorems 4.2 part (1) and 4.1 parts (2,3).

Theorem 4.1: Let $\mathcal{I}_0$,$\mathcal{I}$ be as in the definition of problem *CONT*, then:
(1) *CONT*(q_0,-) is in *coNP* if $\mathcal{I}$ is represented by tables.
(2) *CONT*(-,-) is in *NP* if $\mathcal{I}_0$ is represented by g-tables and $\mathcal{I}$ by e-tables.
(3) *CONT*(-,-) is in PTIME if $\mathcal{I}_0$ is represented by g-tables and $\mathcal{I}$ by tables.

Proof: **(1)** Consider the negation of the problem. All one has to do is (a) guess a valuation σ_0, (b) compute $I_0 = q_0(\sigma_0(T_0))$, and (c) check that I_0 is not in $\mathcal{I}$. As in the proof of Proposition 2.1, by genericity, only a polynomial number of valuations have to be considered. For each of these guesses,

- since q_0 is a *QPTIME* query, step (b) is in *PTIME*; and
- since $\mathcal{I}$ is represented by a vector of tables, by Theorem 3.1 part (1), (c) is in *PTIME*.

Thus, one has to perform a polynomial number of guesses and for each guess a polynomial time computation. Therefore the negation of the problem is in NP, so the problem is in $coNP$.

To prove **(2)** and **(3)**, it suffices to use the properties of *MEMB* and to prove the following claim:

Claim: Let (T_0, Φ_{T0}) be a g-table and T an e-table. Let X be the set of variables appearing in T_0. Let $\{a_x \mid x \in X\}$ be a set of distinct constants not occurring in T_0 or T. We denote by K_0 the instance obtained by replacing each occurrence of each variable x in T_0 by a_x. Then, $rep(T_0) \subseteq rep(T)$ iff $K_0 \in rep(T)$.

Proof of the claim: (i) First suppose that K_0 is in $rep(T)$. Then $K_0 = \sigma(T)$ for some valuation σ. Let I_0 be in $rep(T_0)$. Then $I_0 = \sigma_0(T_0)$ for some valuation σ_0. Consider the mapping ρ over the set of constants defined by $\rho(a_x) = \sigma_0(x)$ for each x in X, and the identity elsewhere. Now consider the valuation σ' of T defined by $\sigma'(x) = \rho(\sigma(x))$ for each x occurring in T. Then $I_0 = \sigma'(T)$, so $I_0 \in rep(T)$. Thus $rep(T_0) \subseteq rep(T)$. (ii) Now suppose that $rep(T_0) \subseteq rep(T)$. Since K_0 is in $rep(T_0)$ (all equalities and inequalities are satisfied) we have that K_0 is in $rep(T)$. $\Box$

Theorem 4.2: Let $\mathcal{I}_0, \mathcal{I}$ be as in the definition of problem *CONT*, then we have that,
(1) *CONT*(-,-) is Π_2^p-complete even if $\mathcal{I}$ is represented by a single i-table and $\mathcal{I}_0$ is represented by a single table.
(2) $\exists$ positive existential q such that *CONT*(-,q) is Π_2^p-complete even if $\mathcal{I}$ is represented by a single table and $\mathcal{I}_0$ is represented by a single table.
(3) *CONT*(-,-) is Π_2^p-complete even if $\mathcal{I}$ is represented by a single e-table and $\mathcal{I}_0$ be represented by a single c-table.
(4) $\exists$ positive existential q_0 such that *CONT*(q_0,-) is *coNP*-complete even if $\mathcal{I}$ is represented by a single table and $\mathcal{I}_0$ is represented by a single table.
(5) $\exists$ positive existential q_0 such that *CONT*(q_0,-) is Π_2^p-complete even if $\mathcal{I}$ is represented by a single e-table and $\mathcal{I}_0$ is represented by a single table.

Proof: (1) Reduction of $\forall\exists$ 3CNF: We first state the Π_2^p-complete problem used ([S]).
input: two disjoint sets X and Y of variables, and a conjunction H of or-clauses over $X \cup Y$ such that for each c in H, $\#(c) = 3$.
question: does there exist for each truth assignment of X, a truth assignment of Y which makes H true?

Let $H = \{c_k \mid k \in [1..p]\}$ be a conjunction of or-clauses over $X \cup Y$. Suppose that we have $X = \{x_i \mid i \in [1..n]\}$, and $Y = \{x_j \mid j \in [n+1..n+m]\}$. Then T_0 and (T, Φ_T) of arity 4 are constructed as follows.

a. $T_0 = \{0z_iii \mid i \in [1..n]\} \cup \{10ii \mid i \in [1..n]\} \cup \{abc0 \mid a, b, c \in \{0, 1\}, a + b + c \neq 0\}$.

T_0			
0	z_1	1	1
1	0	1	1
0	z_2	2	2
1	0	2	2
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

T			
	$\boxed{\Phi_T}$		
u_1	w_1	1	1
v_1	y_1	1	1
u_2	w_2	2	2
v_2	y_2	2	2
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0
z_{11}	z_{12}	z_{13}	0
z_{21}	z_{22}	z_{23}	0
z_{31}	z_{32}	z_{33}	0
z_{41}	z_{42}	z_{43}	0
z_{51}	z_{52}	z_{53}	0

$$\begin{aligned}\Phi_T = \{&w_1 \neq 5, y_1 \neq 6, w_2 \neq 5, y_2 \neq 6, z_{11} \neq z_{42}, z_{11} \neq z_{51}, z_{21} \neq z_{42},\\ &z_{21} \neq z_{51}, z_{31} \neq z_{42}, z_{31} \neq z_{51}, z_{12} \neq z_{22}, z_{12} \neq z_{52}, z_{41} \neq z_{22},\\ &z_{41} \neq z_{52}, z_{33} \neq z_{53}, z_{43} \neq z_{53}, z_{11} \neq v_1, z_{12} \neq v_2, z_{13} \neq v_3,\\ &z_{21} \neq v_1, z_{22} \neq u_2, z_{23} \neq v_4, z_{31} \neq v_1, z_{32} \neq v_4, z_{33} \neq v_5\\ &z_{41} \neq v_2, z_{42} \neq u_1, z_{43} \neq v_5 z_{51} \neq u_1, z_{52} \neq u_2, z_{53} \neq u_5\}\end{aligned}$$

Figure 7: Example for 4.2.1

b. $T = \{u_i w_i ii \mid i \in [1..n]\} \cup \{v_i y_i ii \mid i \in [1..n]\} \cup \{abc0 \mid a,b,c \in \{0,1\} a+b+c \neq 0\}$
$\cup \{z_{k,1} z_{k,2} z_{k,3} 0 \mid k \in [1..p]\}$.

c. $C = \{w_i \neq 5, y_i \neq 6 \mid i \in [1..n]\}$
$\cup\{z_{k,j} \neq z_{k',j'}$ | for some x, x is the jth member of clause k and $\neg\ x$ is the j'-member of clause k' $\}$
$\cup\{z_{k,j} \neq v_l$ | for some l in [1..n], x_l is the jth member of kth clause$\}$
$\cup\{z_{k,j} \neq u_l$ | for some l in [1..n], $\neg x_l$ is the jth member of kth clause$\}$.

Consider the instance of the ∀∃3CNF problem of Figure 5. The corresponding tables are represented in Figure 7.

Intuitively, for each i in $[1..n]$, x_i assigned to "true" corresponds to $\sigma_0(z_i) = 5, \sigma(y_i) = 5, \sigma(u_i) = 1$ and $\sigma(v_i) = 0$; and x_i assigned to "false" corresponds to $\sigma_0(z_i) = 6, \sigma(w_i) = 6, \sigma(u_i) = 0$ and $\sigma(v_i) = 1$. If $\sigma_0(z_i)$ is neither 5 nor 6, x_i is neither bound to true nor false.

Let X,Y,H be an instance of the ∀∃ 3CNF problem that is answered positively. Let

σ_0 be a valuation of T_0. Consider the truth assignment τ_0 of X defined by: $\tau_0(x_i)$ is true iff $\sigma_0(z_i) = 5$. Since the question is answered positively, there is an extension τ of τ_0 which satisfies H. Consider the valuation σ defined by:

i. for each i, if $\sigma_0(z_i) = 5$ then $\sigma(u_i) = 1, \sigma(w_i) = 0, \sigma(v_i) = 0, \sigma(y_i) = 5$ else $\sigma(u_i) = 0, \sigma(w_i) = \sigma_0(z_i), \sigma(v_i) = 1, \sigma(y_i) = 0$;

ii. for all i, k, if τ satisfies the kth member of ith clause, $\sigma(z_{i,k}) = 1$ else $\sigma(z_{i,k}) = 0$.

Note that τ_0 satisfies x_i iff $\sigma_0(z_i) = 5$ iff $\sigma(u_i) = 1$ and $\sigma(v_i) = 0$. Otherwise, $\sigma(u_i) = 0$ and $\sigma(v_i) = 1$. Since τ satisfies H, it is easy to see that $\sigma_0(T_0) = \sigma(T)$. To conclude that $\sigma_0(T_0) \in rep(T, \Phi_T)$, it is straightforward to check that σ satisfies Φ_T. Hence $rep(T_0) \subseteq rep(T, \Phi_T)$.

Conversely, suppose that $rep(T_0) \subseteq rep(T, \Phi_T)$. Let τ_0 be a truth assignment for X. Consider the valuation σ_0 of T_0 defined by $\sigma_0(z_i)$ is 5 if $\tau_0(x_i)$ is true and is 6 otherwise. Since $rep(T_0) \subseteq rep(T, \Phi_T)$, $\sigma_0(T_0) = \sigma(T, \Phi_T)$ for some valuation σ. One can check that σ encodes an extension of τ_0 which satisfies H. Thus we have:

The answer to the instance of $\forall\exists$ 3CNF problem is yes iff $rep(T_0) \subseteq rep(T, \Phi_T)$.

(2) Reduction of $\forall\exists$ 3CNF: The proof is given for databases of more than one relations. By standard modifications one can prove the single relation case.

Given an instance of the $\forall\exists$ 3CNF problem, a vector of tables $T = < T(R), T(S) >$, a vector of tables $T_0 = < T_0(R_0), T_0(S_0) >$, and a query $q = < q_1, q_2 >$ are constructed. Let $H = \{c_k \mid k \in [1..p]\}$ be a set of clauses over $X \cup Y$. Suppose that $X = \{x_i \mid i \in [1..n]\}$, and $Y = \{x_j \mid j \in [n+1..n+m]\}$.

a. $T_0(R_0) = \{iv_i \mid i \in [1..n]\}$ where $\{v_i\}$ is a set of distinct variables.

b. $T_0(S_0) = \{k \mid k \in [1..p]\}$.

c. $T(R) = \{iu_i \mid i \in [1..n]\}$ where $\{u_i\}$ is a set of distinct variables.

d. $T(S) = \{kz_{k,j}i1 \mid x_i \text{ is the } j\text{th member of } k\text{th clause}\}$
$\cup\{kz_{k,j}i0 \mid \neg x_i \text{ is the } j\text{th member of } k\text{th clause}\}$.

e. $q =< q_1, q_2 >$ where $q_1 = \{xy \mid R(x, y)\}$ and

$$q_2 = \{x \mid \exists yz[S(x1yz)] \vee \exists x'x''y[S(x'1y0) \wedge S(x''1y1) \wedge x = 0] \\ \vee \exists x'y[R(y0) \wedge S(x'1y1) \wedge x = 0] \vee \exists x'y[R(y1) \wedge S(x'1y0) \wedge x = 0]\}$$

The previous construction is illustrated in Figure 8 for the instance of Figure 5.

Intuitively, $\sigma_0(v_i) = 1$ corresponds to x_i assigned to true, and $\sigma_0(v_i) = 0$ to x_i assigned to false. Any other value means freedom for x_i.

$T_0(R_0)$	
1	v_1
2	v_2

$T_0(S_0)$
1
2
3
4
5

$T(R)$	
1	u_1
2	u_2

$T(S)$			
1	z_{11}	1	1
1	z_{12}	2	1
1	z_{13}	3	1
2	z_{21}	1	1
2	z_{22}	2	0
2	z_{23}	4	1
3	z_{31}	1	1
3	z_{32}	4	1
3	z_{33}	5	1
4	z_{41}	2	1
4	z_{42}	1	0
4	z_{43}	5	1
5	z_{51}	1	0
5	z_{52}	2	0
5	z_{53}	5	0

Figure 8: Example for 4.2.2

Let X,Y,H be an instance of the problem that is answered positively. Let σ_0 be a valuation of T_0. Consider the truth assignment τ_0 of X, s.t., $\tau_0(x_i)$ is true iff $\sigma_0(v_i) = 1$. Since the answer is yes, there is an extension τ of τ_0 which satisfies H. Consider the valuation σ of T defined by:

i. for each i, $\sigma(u_i) = \sigma_0(v_i)$,

ii. $\sigma(z_{k,j}) = 1$ iff τ satisfies the kth member of jth clause, and $\sigma(z_{k,j}) = 0$ otherwise.

By (i), $q_1(\sigma(T)) = \sigma(T(R)) = \sigma_0(T_0(R_0))$. Let k be in $[1..p]$. Then one of the three disjuncts in c_k is satisfied by τ. Thus $k1ii'$ is in $\sigma(T)$ for some i, i'. Hence k is in $q_2(\sigma(T))$. To conclude that $q_2(\sigma(T)) = \sigma_0(T_0(S_0))$, it suffices to notice that by construction, 0 can not be in $q_2(\sigma(T))$. Hence $rep(T_0) \subseteq q(rep(T))$.

The converse is also similar. So we have that: the answer to the instance of $\forall\exists$ 3CNF problem is yes iff $rep(T_0) \subseteq q(rep(T))$.

(3) This case follows from the proof of case **(5)** below and the technique of [IL]. Namely, the c-table used in the hardness reduction here is the result of applying the query q_0 on the table from case **(5)**. By [IL] this application leads to a c-table describing the same set of worlds and can be done in *PTIME*.

(4) Reduction of 3DNF tautology: The proof is given for databases of more than one relations. By standard modifications one can prove the single relation case.

$T_0(R_0)$				$T_0(S_0)$	
1	1	1		1	u_1
1	2	1		2	u_2
1	3	1		3	u_3
	...			4	u_4
5	1	0		5	u_5
5	2	0			
5	5	0			

Figure 9: Example for 4.2.4

For each instance H of the 3DNF tautology problem, tables $T_0 = < T_0(R_0), T_0(S_0) >$, query q_0, and table T of arity 1 are constructed. Let $H = \{c_i \mid i \in [1..p]\}$, and $X = \{x_j \mid j \in [1..m]\}$. Then T_0, q_0, and T are constructed as follows:

a. $T_0(R_0) = \{ij1 \mid x_j$ is member of clause i $\} \cup \{ij0 \mid \neg x_j$ is member of clause i $\}$.

b. $T_0(S_0) = \{ju_j \mid j \in [1..m]\}$.

c. $q_0 = \{x \mid \exists yz((R_0(xyz) \wedge S_0(yz)) \vee x = 0)\}$.

d. $T = \{z_1, ..., z_p\}$.

The construction for the formula of Figure 5 is illustrated in Figure 9.

Suppose that H is not a tautology. Then there is a truth assignment τ such that τ falsifies H. Let σ_0 be the valuation defined by $\sigma_0(u_i) = 0$ if τ satisfies x_i, and $\sigma_0(u_i) = 1$ otherwise. Since τ falsifies H, and H is in DNF, τ falsifies c_i for each i. Let i be in $[1..p]$. Since τ falsifies c_i, one member of that clause is not satisfied by τ. Suppose that that member is x_j. (The case $\neg x_j$ is treated in a similar way.) Since τ falsifies x_j, $\sigma_0(u_j) = 1$. Since x_j is member of c_i, $ij1 \in T_0(R_0)$. Therefore $ij1 \in \sigma_0(T_0(R_0))$, and $j1 \in \sigma_0(T_0(S_0))$. Hence $i \in q_0(\sigma_0(T_0))$. Thus $q_0(\sigma_0(T_0)) = \{0, 1, ..., p\}$, and $q_0(\sigma_0(T_0)) \notin rep(T)$. The converse is similar and one can show that: H is a tautology iff $q_0(rep(T_0)) \subseteq rep(T)$.

(5) Reduction of $\forall\exists$ 3CNF: Again for clarity we use many relation databases. Given an instance of the $\forall\exists$ 3CNF problem, a vector of e-tables $T = < T(R), T(S) >$, a vector of tables $T_0 = < T_0(R_0), T_0(S_0) >$, and a query $q_0 =< q_{01}, q_{02} >$ are constructed. Let $H = \{c_k \mid k \in [1..p]\}$ be a set of clauses over $X \cup Y$. Suppose that $X = \{x_i \mid i \in [1..n]\}$, and $Y = \{x_j \mid j \in [n+1..n+m]\}$.

a. $T_0(R_0) = \{ijk \mid i \in [1..p], j, k \in [0..1]\}$.

b. $T_0(S_0) = \{iy_iz_i \mid i \in [1..n]\}$.

c. $q_{01} = \{xyz \mid R_0(xyz)\}$.

$T_0(R_0)$		
1	0	0
1	0	1
1	1	0
1	1	1
	...	
5	0	0
5	0	1
5	1	0
5	1	1

$T_0(S_0)$		
1	y_1	z_1
2	y_2	z_2

$T(R)$		
1	1	0
1	0	1
	...	
5	1	0
5	0	1
1	u_1	1
1	u_2	1
1	u_3	1
	...	
5	u_1	0
5	u_2	0
5	u_5	0
1	z_1	z_1
	...	
5	z_5	z_5

$T(S)$	
1	u_1
1	0
2	u_2
2	0

Figure 10: Example for 4.2.5

d. $q_{02} = \{xw \mid \exists yz((S_0(xyy) \wedge w = 1) \vee (S_0(xyz) \wedge w = 0))\}$.

e. $T(R) = \{iu_j1 \mid x_j \text{ is member of clause } i\ \} \cup \{iu_j0 \mid \neg x_j \text{ is member of clause } i\ \}$
$\cup\{i10 \mid i \in [1..p]\} \cup \{i01 \mid i \in [1..p]\} \cup \{iz_iz_i \mid i \in [1..p]\}$.

f. $T(S) = \{iu_i \mid i \in [1..n]\} \cup \{i0 \mid i \in [1..n]\}$.

The construction is illustrated in Figure 10 for the $\forall\exists$ 3CNF example of Figure 5.

Intuitively, $x_i \in X$ assigned to true corresponds to a tuple (iaa) in S_0 and two tuples $(i1), (i0)$ in S. An assignment to false is represented by a tuple (iab) with $a \neq b$ in S_0 and a tuple $(i0)$ in S. Now consider R and a clause i. The tuples $(i01)$ and $(i10)$ are in R; one tuple in $\{i00, i11\}$ is obtained if the ith clause is satisfied, and the other one can be provided by (iz_iz_i).

Suppose that $q_0(rep(T_0)) \subseteq rep(T)$. We prove that the corresponding instance of the $\forall\exists$ 3CNF problem is answered positively. Let τ be a truth assignment of $x_1, ..., x_n$. Consider the valuation σ_0 defined by: $\sigma_0(y_i) = \sigma_0(z_i) = 17$ if τ satisfies x_i; and $\sigma_0(y_i) = 15$, $\sigma_0(z_i) = 17$ otherwise. Since $q_0(rep(T_0)) \subseteq rep(T)$, there exists a valuation σ of T such that $q_0(\sigma_0(T_0)) = \sigma(T)$. By inspection of q_{02}, for each i in $[1..n]$, $\sigma(u_i) = 1$ iff τ satisfies x_i. Since q_{01} is the identity, $\sigma(u_i) = 0$ or 1 for each i in $[1..n{+}m]$. Consider the truth assignment τ' defined by: $\tau'(x_i)$ is true iff $\sigma(u_i) = 1$. Clearly, τ' is an extension of τ. We next prove that τ' satisfies H.

Let i be in $[1..p]$. Since q_{01} is the identity, $i00$ and $i11$ are in $q_{01}(\sigma_0(T_0))$, and so in

$\sigma(T(R))$. Thus one of the following two cases arise:

i. "$\sigma(iz_iz_i) = i00$." Then, for some j, and $iu_j1 \in T(R)$, $\sigma(iu_j1) = i11$. Hence x_j is member of the ith clause, and $\sigma(u_j) = 1$, i.e., τ satisfies x_j and the ith clause is satisfied.

ii. "$\sigma(iz_iz_i) = i11$". Then, use the symmetric argument.

Hence τ satisfies H and the instance H,X,Y of the $\forall\exists$ 3CNF problem is answered by yes.

The converse is similar and we thus have: the answer to the instance of $\forall\exists$ 3CNF problem is yes iff $q_0(rep(T_0)) \subseteq rep(T)$. $\square$

5 Possibility and Certainty

The next theorem indicates how similar unbounded possibility is to membership, from a computational point of view.

Theorem 5.1: Let $\mathcal{I}$ be as in the definition of $POSS$, then we have that,
(1) $POSS(*,\text{-})$ is in $PTIME$ if $\mathcal{I}$ is represented by tables.
(2) $POSS(*,\text{-})$ is NP-complete even if $\mathcal{I}$ is represented by a single e-table.
(3) $POSS(*,\text{-})$ is NP-complete even if $\mathcal{I}$ is represented by a single i-table.
(4) $\exists$ positive existential q, s.t., $POSS(*,q)$ is NP-complete even if $\mathcal{I}$ is represented by a single table.

Proof: **(1)** The argument is a variation on that of Theorem 3.1 part (1).

(2) Reduction of 3CNF satisfiability: We first state this problem.
input: Set X of variables and conjunction H of cardinality 3 or-clauses over X.
question: is there a satisfying truth assignment for H?

For each instance H of the 3CNF satisfiability problem, an e-table T, and a set of facts P of width 3 are constructed. Let $H = \{c_i \mid i \in [1..n]\}$, and $X = \{x_j \mid j \in [1..m]\}$. For each variable x_j in X, let u_i, y_j be new variables.

a. $T = \{ju_jy_j \mid j \in [1..m]\} \cup \{jy_ju_j \mid j \in [1..m]\}$
$\cup\{(m+i)(m+i)u_j \mid c_i \text{ contains } x_j \text{ for some } j\}$
$\cup\{(m+i)(m+i)y_j \mid c_i \text{ contains } \neg x_j \text{ for some } j\}$.

b. $P = \{j01 \mid j \in [1..m]\} \cup \{j10 \mid j \in [1..m]\}$
$\cup\{(m+i)(m+i)1 \mid i \in [1..n]\}$.

This reduction is illustrated in Figure 11.b for the example formula of Figure 5.

Suppose that τ is a satisfying truth assignment of H. Consider the valuation σ defined by $\sigma(u_j) = 1$, $\sigma(y_j) = 0$ if $\tau(x_j)$ is true; and $\sigma(u_j) = 0$, $\sigma(y_j) = 1$ otherwise. It is easily seen that $P \subseteq \sigma(T)$. The converse is similar so we have:

(a)

T	
1	x_{11}
1	x_{12}
1	x_{13}
2	x_{21}
2	x_{22}
2	x_{23}
3	x_{31}
3	x_{32}
3	x_{33}
4	x_{41}
4	x_{42}
4	x_{43}
5	x_{51}
5	x_{52}
5	x_{53}

P	
1	1
2	1
3	1
4	1
5	1

$$\begin{aligned}\Phi_T = \{&x_{11} \neq x_{42}, x_{11} \neq x_{51},\\ &x_{21} \neq x_{42}, x_{21} \neq x_{51},\\ &x_{31} \neq x_{42}, x_{31} \neq x_{51},\\ &x_{12} \neq x_{22}, x_{12} \neq x_{52},\\ &x_{41} \neq x_{22}, x_{41} \neq x_{52},\\ &x_{33} \neq x_{53}, x_{43} \neq x_{53}\}\end{aligned}$$

(b)

T		
1	y_1	u_1
1	u_1	y_1
2	y_2	u_2
2	u_2	y_2
3	y_3	u_3
3	u_3	y_3
4	y_4	u_4
4	u_4	y_4
5	y_5	u_5
5	u_5	y_5
6	6	u_1
6	6	u_2
6	6	u_3
7	7	u_1
7	7	y_2
7	7	u_4
8	8	u_1
8	8	u_4
8	8	u_5
9	9	u_2
9	9	y_1
9	9	u_5
10	10	y_1
10	10	y_2
10	10	y_5

P		
1	0	1
1	1	0
2	0	1
2	1	0
3	0	1
3	1	0
4	0	1
4	1	0
5	0	1
5	1	0
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1

Figure 11: (a) Example for 5.1.3, (b) Example for 5.1.2

There is a satisfying truth assignment for H iff $P \subseteq I$ for some I in $rep(T)$.

(3) Reduction of 3CNF satisfiability. For each instance H of the 3CNF satisfiability problem, an i-table (T, Φ_T), and an instance P of arity 2 are constructed. Let $H = \{c_i \mid i \in [1..n]\}$, and $\{x_{i,k} \mid i \in [1..n], k \in \{1,2,3\}\}$ be a set of distinct variables.

a. $T = \{ix_{i,k} \mid i \in [1..n], k \in \{1,2,3\}\}$.

b. $C = \{x_{ik} \neq x_{jl} \mid k$th member of ith clause is some variable x, and l-th member of jth clause is $\neg x\}$.

c. $P = \{i1 \mid i \in [1..n]\}$.

This reduction is illustrated in Figure 11.a for the example formula of Figure 5. Now, suppose that τ is a satisfying truth assignment for H. Consider the valuation σ defined by $\sigma(x_{i,k}) = 1$ if the kth member of the ith clause is satisfied by τ; and $\sigma(x_{i,k}) = 0$ otherwise. It is easily seen that $P \subseteq \sigma(T, \Phi_T)$. The converse is similar. Thus, there is a satisfying truth assignment for H iff $P \subseteq I$ for some I in $rep(T, \Phi_T)$.

(4) Consider the proof of Theorem 3.1 part(4). It can be shown that G is 3-colorable iff there exists K in $q(rep(T))$ such that $I_0 \subseteq K$. □

Our next theorem is about bounded possibility. The upper bound is a consequence of the fact that c-tables are *representation systems* in the sense of [IL] and positive existential queries can be incorporated explicitly in the c-table representation, without any exponential growth. This growth may be unavoidable for first order and *DATALOG* queries as indicated by the lower bounds. Once again the interest of the lower bounds lies in their syntactic simplicity, e.g., the *DATALOG* query of uses monadic fixpoints on tables.

Theorem 5.2: Let $\mathcal{I}$ be as in the definition of *POSS*(k,q), then we have that:
(1) *POSS*(k,q) is in $PTIME$ for q pos. exist. and $\mathcal{I}$ is represented by c-tables.
(2) $\exists q$ first order query, s.t., *POSS*(1,q) is NP-complete even if $\mathcal{I}$ is represented by tables.
(3) $\exists q$ *DATALOG* query, s.t., *POSS*(1,q) is NP-complete even if $\mathcal{I}$ is represented by tables.

Proof: (1) The idea is to transform the given positive existential view of a c-table into another equivalent c-table, that is not bigger than a polynomial of the size of the input. This can be done because of the positivity of the queries and because of their fixed length. One proceeds by structural induction on algebraic expressions composed of: *project, natural join, union, renaming, positive select.* These are equivalent to the positive existential queries and correspond to a subset of the c-table manipulation rules from [IL]. It is straightforward to see that these manipulation rules have the desired behavior. Then one can find whether a bounded pattern is possible by exhaustive search, with the size of the bounded pattern being an exponent in the running time.

(2) Reduction of 3DNF nontautology: Let $H = \{c_i\}$ be the given set of clauses and $\{x_j\}$ the given set of variables. Then a table T is constructed with variables $\{z_{i,k}\}$ and tuples the set:

$\{iz_{i,k}j1 \mid x_j$ appears in position k of $c_i\}$ $\cup\{iz_{i,k}j0 \mid \neg x_j$ appears in position k of c_i $\}$.

We want the fact 1 to be certainly in the answer to a query q' iff the original 3DNF formula is a tautology. For this, let $q' = \{1 \mid \psi\}$ where ψ is as follows:
$\psi = \exists xyzvx_1y_1v_1[R(xyzv) \wedge R(x_1y_1zv_1) \wedge y \neq y_1] \vee \exists xyzv[R(xyzv) \wedge y \neq 1 \wedge y \neq 0] \vee$
$\exists xyzv[R(xyzv) \wedge \forall y_1z_1v_1\{R(xy_1z_1v_1) \Rightarrow (y_1 = 1 \wedge v_1 = 1) \vee (y_1 = 0 \wedge v_1 = 0)\}]$

Intuitively, ψ states that either $\sigma(T)$ does not represent a truth assignment, or that truth assignment is satisfied by H. The proof of the following claim is straightforward.

Claim: H is a tautology iff 1 is a certain fact in $q'(rep(T))$.

To see that, suppose that H is not a tautology. Let τ be a truth assignment such that τH does not hold. Consider the valuation σ defined by:

- $\sigma z_{i,k} = 1$ if x_j appears in position k of c_i and τx_j is true;
- $\sigma z_{i,k} = 1$ if $\neg x_j$ appears in position k of c_i and τx_j is false;
- $\sigma z_{i,k} = 0$ otherwise.

It is easy to verify that 1 is not in $q'(\sigma T)$. The converse is also straightforward.

Now, for the bounded possibility problem take the query $q = \{1 \mid \neg\psi\}$. We have that: 1 is a possible fact in $q(rep(T))$ iff H is a nontautology.

(3) Reduction of 3CNF satisfiability. We can show that *POSS*(1,*transitive-closure*) is NP-complete for a g-table representation, but it is in $PTIME$ for a table representation. So instead, we use a query:

$$q_1(R) = \{x \mid R(x) \vee \exists yz[R(y) \wedge R(z) \wedge R_1(yx) \wedge R_2(zx)]\}$$

The query q with input instances (R_0, R_1, R_2) is the least fixpoint of q_1, which contains R_0.

For each instance H of 3CNF satisfiability, we now give the construction of a table T. Let $X = \{x_i \mid i \in [1..n]\}$, and $H = \{c_j \mid j \in [1..m]\}$. For each i in $[1..n]$, let t_i, f_i, a_i, and b_i be distinct constants. For each j in $[1..m]$, let h_j be a new constant. Let a be a new constant. Then T is constructed as follows The variables used are $\{x_i \mid i \in [1..n]\}$.

$T(R_0) = \{a\}$.

$T(R_1) = \{at_i \mid i \in [1..n]\} \cup \{af_i \mid i \in [1..n]\} \cup\{aa_i \mid i \in [1..n]\} \cup\{ab_1\} \cup\{b_ib_{i+1} \mid i \in [1..n-1]\} \cup\{t_ih_j \mid$ if x_i is in jth clause $\} \cup\{f_ih_j \mid$ if $\neg x_i$ is in jth clause$\} \cup\{b_n1\}$.

$T(R_2) = \{ax_1\} \cup \{a_ix_{i+1} \mid i \in [1..n-1]\} \cup\{t_ia_i \mid i \in [1..n]\} \cup\{f_ia_i \mid i \in [1..n]\} \cup\{a_ib_i \mid i \in [1..n]\} \cup\{ah_1\} \cup \{h_jh_{j+1} \mid j \in [1..m-1]\} \cup\{h_m1\}$.

Consider the instance of the 3CNF satisfiability problem given in Figure 5. A graphical representation of T is shown in Figure 12. The R_0 unary relation is indicated by all the a-labeled nodes, the R_1 relation is indicated by broken lines, and the R_2 relation is indicated by continuous lines.

A simulation of the possible evaluations of q suffices to show the following equivalence: H is satisfiable iff 1 is a possible answer in q applied to T. Let us argue the two directions.

If H is satisfiable then there is a satisfying assignment τ. Pick a valuation that equates x_i to t_i if $\tau(x_i)$ is true and to f_i if it is false. It is easy to perform the evaluation (using Figure 12) and to see that 1 is in the output.

Let 1 be in the output. In order for this to be the case we make two observations. Call the group of nodes $\{t_i, f_i, a_i, b_i\}$ the i-group. (i) "All the b_i are in the output". This is forced by the R_1 ancestors of 1. (ii) "Exactly one x has to be valuated to the nodes of each i-group." By the pigeonhole principle if this does not happen one group (say the jth) will not have any x valuated to it. But then, one can argue that b_j cannot be in the output, a contradiction.

Based on the two observations (i,ii) above we have that, if 1 is in the output at most one of each pair (t_i, f_i) is in the output and all the x's are valuated in the i-groups. Thus, for 1 to be in the output all the h's must be in the output. Finally, it is easy to see that, the insertion of the h's in the output requires a satisfying assignment to H. $\Box$

Our last theorem is about certainty.

Theorem 5.3: Let $\mathcal{I}$ be as in the definition of $CERT(*,q)$, then we have that,
(1) [IL, V2] $CERT(*,q)$ is in *PTIME* if q is a *DATALOG* query and $\mathcal{I}$ is represented by g-tables.
(2) $\exists q$ first order query, s.t., $CERT(1,q)$ is *coNP*-complete even if $\mathcal{I}$ is represented by a table.
(3) $CERT(1,\text{-})$ is *coNP*-complete even if $\mathcal{I}$ is represented by a c-table.

Proof: **(1)** The upper bound follows directly from the central results of [IL, V2] and is only included here for completeness of presentation. The efficient algorithm corresponds to manipulating the matrix representation of the g-tables (i.e., with equalities incorporated) as if they were complete information databases.

(2) Reduction of 3DNF tautology: Consider the 3DNF formula H, the table T, and the query q' of the proof of Theorem 5.2 part(2). Recall that: H is a tautology iff 1 is a certain fact in $q'(rep(T))$. This lower bound is a refinement of a lower bound in [V2].

(3) Similar to the proof of Theorem 3.2 part (3). $\Box$

6 Conclusions and Open Questions

We have investigated the data complexity of incomplete information databases. We have focused on views of tabular representations, from the very simple Codd-tables to the more complex conditioned-tables. In this setting we analyzed containment, membership, unique-

a			a_5			b_5	1	
	a	t_5		f_5	a			
			x_5					f_1
a			a_4			b_4	h_5	f_2
	a	t_4		f_4	a			f_5
			x_4					t_2
a			a_3			b_3	h_4	f_1
	a	t_3		f_3	a			t_5
			x_3					t_1
a			a_2			b_2	h_3	t_4
	a	t_2		f_2	a			t_5
			x_2					t_1
a			a_1			b_1	h_2	f_2
	a	t_1		f_1	a			t_4
			x_1					t_1
			a			a	h_1	t_2
								t_3

Figure 12: Example for 5.2.3

ness, possibility, and certainty problems.

Many of our lower bounds are in terms of particular hard queries. Are there syntactic characterizations for easy queries in each case? In particular, it would be interesting to have good characterizations for the *MEMB* lower bound of Theorem 3.1, part (4). These would be positive existential views of Codd-tables, whose membership question is in *PTIME*. We believe that they could serve as a starting point for studying sufficient conditions for efficient evaluation in the presence of incomplete information.

The null values used here are values present but unknown, sometimes constrained through explicit conditions. It would be interesting to investigate null values, whose presence is also unknown [Z]. Finally, in our query programs we do not have explicit operators for "certainty" and "possibility", [L]. What is the effect of such "modal" operators on data-complexity?

References

[AG] Abiteboul S., Grahne, G., "Update Semantics for Incomplete Databases", *Proceedings of the 11th International Conference on Very Large Databases*, pp. 1–12, 1985.

[CH1] Chandra, A.K., Harel, D., "Structure and Complexity of Relational Queries", *Journal of Computer and System Sciences*, vol. 25, no. 1, pp. 99–128, 1982.

[CH2] Chandra, A.K., Harel, D., "Horn Clause Programs and Generalizations", *Journal of Logic Programming*, vol. 2, no. 1, pp. 1–15, 1985.

[Cod] Codd, E.F., "Extending the Database Relational Model to Capture More Meaning", *ACM Transactions on Database Systems*, vol. 4, no. 4, pp. 397–434, 1979.

[Cos] Cosmadakis, S.S., "The Complexity of Evaluating Relational Queries", *Information and Control*, vol. 58, pp. 101–112, 1983.

[GJ] Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[G] Grahne, G., "Dependency Satisfaction in Databases with Incomplete Information", *Proceedings of the 10th International Conference on Very Large Databases*, pp. 37–45, 1984.

[HLY] Honeyman, P., Ladner, R., Yannakakis, M., "Testing the Universal Instance Assumption", *Information Processing Letters*, vol. 10, no. 1, pp. 14–19, 1980.

[I] Imielinski, T., "On Algebraic Query Processing in Logical Databases", in *Advances in Database Theory, Vol. 2*, ed. Gallaire, H. and Minker, J. Plenum Press, 1984.

[IL] Imielinski, T., Lipski, W. Jr, "Incomplete Information in Relational Databases", *Journal of the ACM*, vol. 31, no. 4, pp. 761–791, 1984.

[L] Lipski, W. Jr, "On Databases with Incomplete Information", *Journal of the ACM*, vol. 28, no. 1, pp. 41–70, 1981.

[MSY] Maier, D., Sagiv, Y., Yannakakis, M., "On the Complexity of Testing Implications of Functional and Join Dependencies", *Journal of the ACM*, vol. 28, no. 4, pp. 680–695, 1981.

[R] Reiter R., "A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values", *Journal of the ACM*, vol. 33, no. 2, pp. 349–370, 1986.

[S] Stockmeyer, L., "The Polynomial Time Hierarchy", *Theoretical Computer Science*, vol. 3, no. 1, pp. 1–22, 1976.

[U] Ullman, J.D., *Principles of Database and Knowledge Base Systems: Volume I*, Computer Science Press, 1988.

[V1] Vardi, M.Y., "The Complexity of Relational Query Languages", *Proceedings of the 14th ACM SIGACT Symposium on the Theory of Computing* pp. 137–146, 1982.

[V2] Vardi, M.Y., "Querying Logical Databases", *Journal of Computer and System Sciences*, vol. 33, no. 2, pp. 142–160, 1986.

[Z] Zaniolo, C., "Database Relations with Null Values", *Journal of Computer and System Sciences*, vol. 28, no. 1, pp. 142–166, 1984.