On the Representation and Querying of Sets of Possible Worlds

Serge Abiteboul[1]

Paris Kanellakis[2]

Gosta Grahne[3]

Abstract: We represent a *set of possible worlds* using an incomplete information database The representation techniques that we study form a hierarchy, which generalizes relations of constants This hierarchy ranges from the very simple Codd-table, (i e , a relation of constants and distinct variables called nulls, which stand for values present but unknown), to much more complex mechanisms involving views on conditioned-tables, (i e , queries on Codd-tables together with conditions) The views we consider are the queries that have polynomial data-complexity on complete information databases Our conditions are conjunctions of equalities and inequalities

(1) We provide matching upper and lower bounds on the data-complexity of testing *containement*, *membership*, and *uniqueness* for sets of possible worlds and we fully classify these problems with respect to our representation hierarchy The most surprising result in this classification is that it is complete in Π_2^p, whether a set of possible worlds represented by a Codd-table is a subset of a set of possible worlds represented by a Codd-table with one conjuction of inequalities

(2) We investigate the data-complexity of querying incomplete information databases We examine both asking for *certain facts* and for *possible facts* Our approach is algebraic but our bounds also apply to logical databases We show that asking for a certain fact is coNP-complete, even for a fixed first order query on a Codd-table We thus strengthen a lower bound of [16], who showed that this holds for a Codd-table with a conjunction of inequalities For each fixed positive existential query we present a polynomial algorithm solving the bounded possible fact problem of this query on conditioned-tables We show that our approach is, in a sense, the best possible, by deriving two NP-completeness lower bounds for the bounded possible fact problem when the fixed query contains either negation or recursion

[1]INRIA, Rocquencourt, FRANCE

[2]Brown University, Providence, RI, USA Work performed while visiting INRIA, Rocquencourt, FRANCE, and partly supported by an IBM Faculty Development Award

[3]University of Helsinki, Helsinki, FINLAND Work performed while visiting INRIA, Rocquencourt, FRANCE

1. Introduction

A fundamental property of common database query languages, such as, relational calculus, relational algebra, [15], and Horn clause recursive rules or DATALOG [3, 2] is that they can be evaluated efficiently on complete information relational databases This is the result of representing these databases by *relations* of constants and of the important insight that these languages express queries whose *data-complexity* is within PTIME [3, 17], i e , they are QPTIME queries 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 It therefore restricts the analysis by assuming fixed relation arities, i e , fixed tuple widths More significantly data-complexity is a reasonable measure to study computation on databases, given that the number of tuples in a database typically dominates (by orders of magnitude) the tuple width and the size of an application program

In order to extend relational databases to capture more applications one must use some mechanism for representing incomplete information databases [4] The most typical (and notorious) such mechanism are *null values* This is primarily an algebraic addition to relations but it has very close analogs in logical databases, e g , [16, 13] There already is a large volume of interesting work on querying incomplete information databases, for example, in this paper we refer to [4, 11, 18, 9, 10, 16, 13, 7, 1] Since we cannot reasonably survey so broad an area we refer to [10] for a detailed recent treatment of the topic The focus of most of this work has been a search for the "correct" semantics for query programs applied to incomplete information databases There has been much less work on the data-complexity of querying incomplete information databases The most significant contribution there is [16], where the computational complexity of evaluating certain answers to a wide range of second-order queries on incomplete information databases is investigated The representation used there is one of queries on logical databases

Another (less realistic) extension, 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 [17, 5] This increase is due to a certain incompleteness of relational algebra with respect to the algebra of polynomials [5] Such problems were first noted in [8, 12] and have some connections to nulls and weak universal instances Data-complexity has the advantage of avoiding these anomalies, by factoring out the query program representation and maintaining only the combinatorics of the uncertainty in the database

The subject of our paper is *a complete data-complexity analysis of problems related to representing and querying databases with null values* Our results complement and extend both [16] and [10]

Representation

Incomplete information databases are representations of *sets of possible worlds* For these representations we use relations over constants, relations with null values (*Codd-tables*) and relations with null values and conditions (the most general ones are *conditioned-tables*) Sets of possible worlds are also represented using QPTIME queries on the worlds worlds We restrict our attention to QPTIME queries, since we believe they are a natural closure of what is expressible via common database query languages Our representations form a hierarchy from complete information relations on constants (a single possible world), to our simplest case of uncertainty which is a Codd-table, i e , one relation with null values that are distinct unconstrained variables, (this relation represents a "simple" set of possible worlds), to intermediate cases of uncertainty such as a Codd-table with conditions, and finally to the most general case of uncertainty which is a QPTIME query on conditioned-tables

We investigate our representation hierarchy from a data-complexity perspective, i e , we consider the tuple width and the query (when different from the identity) as fixed parameters The central computational problem is the *containement* problem "is a given set of possible worlds a subset of another given set of possible worlds?" A special case of this problem is the *membership* problem "is a given complete database one of a given set of possible worlds?" In the membership problem the complete database is represented by relations with constants, thus it is a singleton set of possible worlds The (superficially) dual question about representations is the *uniqueness* problem "is a given set of possible worlds a singleton set consisting of a given complete database?"

Our contribution in this area is a complete classification of containement (and thus membership and uniqueness) with respect to our hierarchy For this 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=Σ_1^P, coNP=Π_1^P, Σ_2^P, Π_2^P of the polynomial-time hierarchy [14], [6] The most surprising result here is that it is complete in Π_2^P, whether a set of possible worlds represented by a Codd-table is a subset of a set of possible worlds represented by a Codd-table with only one conjuction of inequalities, (Theorem 4 2) What is surprising is that containement in our framework is always in Π_2^P and the highest complexity is reached with a minimal amount of expressibility As will be noted below the simplest form of uncertainty (Codd-tables) are fairly well behaved computationally Theorem 4 2 indicates that the addition of a conjunction of inequalitites (not even equalities) is sufficient for Π_p^P-hardness, and its proof has some combinatorial difficulty Our other results for containement (Theorems 3 1, 3 3, 4 1, 4 3) are combinatorially simpler than Theorem 4 2 However, our lower bounds are syntactically tight In the reductions we use positive existential queries (the project, natural join, union, rename, and positive select queries) and Codd-tables There is the following exception

An interesting observation is the breakdown of duality between the membership and uniqueness problems due to the particularities of our representation, (Theorem 3 1 vs Theorem 3 3) In fact the query required for showing coNP-hardness is positive existential with $\neq$ For positive existential queries the uniqueness problem is in polynomial-time This is an illustration of the power of $\neq$

We would like to make some remarks on our simplest algebraic mechanism, namely Codd-tables From a reduction to *bipartite matching* [6] it follows that membership is in polynomial-time for sets of worlds represented by Codd-tables, (Theorem 3 1) This distinguishes Codd-tables from Codd-tables with global conjunctive conditions and makes our classification much more meaningful Codd-tables with one global conjunction of equalities and inequalities are our g-tables These g-tables are similar constructs (modulo isomorphisms) with the logical databases of [16] In fact Codd-tables implicitly assume that all constants are distinct, but it is the additional equalities and inequalities that give the logical databases of [16] their expressive power Thus Codd-tables are isomorphic to a syntactically restricted form of logical database

We use the term e-table for a g-table with only equalities and the term i-table for a g-table with only inequalities Our e-tables have also been described as "V-tables" and "naive-tables" [10, 1] We use the term conditioned-table for the most general tabular representation that we employ These are g-tables with local conditions, i e , conditions attached to the tuples They are like the "C-tables" of [10]augmented by one conjuction of equalitites and inequalities, that is the global condition Wlog, the local conditions of both "C-tables" and our conditioned-tables are conjunctions of equalities and inequalities Conditioned-tables are less general than the constructs used in [7, 1], where global conditions are disjuncts of conjuncts

Our representation hierarchy and classification are illustrated in Figures 1 and 2 respectively

First let us explain Figure 1 A Codd-table (table for short) is a relation with constants and variables, where no variable occurs twice An i-table is a table with a conjunction of inequalities, these are listed on the right of the table An e-table is a table with a conjunction of equalities, we do not list these on the right but incorporate them directly in the table (this is standard practice) Thus a g-table is an e-table together with a conjunction of inequalities, these are listed on the right of the e-table Finally a conditioned-table (c-table for short) 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 The sets of possible worlds represented in this fashion naturally result from instantiating the variables with constants and satisfying the conditions We also allow views of such sets of possible worlds

Now let us explain Figure 2 For the containement problem we have five cases depending on whether x-tables are used x=c,g,e,i,(nil), these are the five cases of Figure 2 For each one of these cases there are nine subcases depending on the two given sets of possible worlds, each one could be of three kinds

(i) a complete database (marked instance on Figure 2),

(ii) an identity view of x-tables (marked x-table on Figure 2),

(iii) a view of x-tables (marked view on Figure 2)

In every one of the five cases of Figure 2 we provide the upper bounds, these are the lines enclosing subcases in PTIME (shaded), NP (solid), coNP (dashed), and Π_2^P (each whole case) All the subcases "strictly" in NP, coNP, Π_2^P are shown complete in their respective classes For this it suffices to show hardness for the ones on Figure 2 that include references to the relevant theorems

Querying

The view mechanism for specifying sets of possible worlds is a natural step towards querying our incomplete information database A first question is the *possibility* problem "given a set of tuples and given a set of possible worlds, is there a possible world where these tuples are all true?" The second question is the *¬certainty* problem "given a set of tuples and a set of possible worlds, is there a possible world where these tuples are not all true?" Its negation is the *certainty* problem "given a set of tuples and given a set of possible worlds are these tuples all true in every possible world?" Note that certainty implies possibility Also certainty and ¬(certainty) are different from possibility and ¬(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 3 2), which is clearly computationaly related to membership, (Theorem 3 1) If we restrict the size of the set of given tuples 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?" For certainty the unbounded and bounded versions of the problem are polynomial-time equivalent (Proposition 2 1) Bounded certainty corresponds to the practical question "is this (small) list of facts certainly true?"

We examine the bounded possibility problem in some detail, (Theorem 5 2) This complements the literature, where much more attention (perhaps unjustifiably) has been given to the certainty problem Our algorithm for bounded possibility uses the algebraic completeness of conditioned-tables demonstarted in [10] We show that the data-complexity of bounded possibility, given a query on conditioned-tables, is polynomial, provided that the query is positive existential 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 ways, always remaining within QPTIME One extension is the first order queries (relational calculus, relational algebra) and the other is the DATALOG queries (Horn clause recursive rules) Both extensions lead to NP-completeness even if the conditioned-tables are Codd-tables The proofs are of some interest, because of the syntactic simplicity of Codd-tables and the queries used

There are two main observations in the literature on certainty The first is an algorithmic observation In its various forms this observation follows from central results of [10] (based on "C-tables") and [16, 13] (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, the positive queries are the DATALOG queries This leads to Theorem 5 1 1, which we only list for completeness of presentation, since it is due to [10, 16] There are some differences between certain answers from logical databases, which might involve variables, and certain answers from conditioned-tables, which have only constants These differences do not affect our analysis The second observation deals with the negative effects of the many possible instantiations of the null values In [16] the certainty question for a fixed first order query on a i-table is shown coNP-complete, both negation and the inequalities are used We stengthen this result to a first order query on a Codd-table (Theorem 5 1 2)

Let us briefly describe what is not covered by our framework The null values used here are values present but unknown, sometimes constrained through explicit conditions Thus we do not cover null values, whose presence is also unknown [18] Our approach is a "closed world" approach and consistent with [16, 13, 11, 10, 9, 7, 1] An alternative approach to incomplete information is an "open world" approach, such as, weak universal instances The complexity results of [8] are motivated by this latter "open world" approach Our queries are QPTIME, and not higher order [16] Thus our bounds are all in the class Π_2^P of the polynomial-time hierarchy [14, 6], (see Proposition 2 1) We do not have explicit operators in the query language for "certainty" and "possibility", [11]

Outline

The detailed definitions are in Section 2, and an effort has been made to minimize notation We now describe our results and justify why they are tight from a syntactic perspective

In Section 3 we study the problem of membership, (Theorem 3 1), the problem of uniqueness, (Theorem 3 3), and the problem of unbounbed possibility (Theorem 3 2) There is an apparent relationship between Theorems 3 1 and 3 2, and an apparent difference between Theorems 3 1 and 3 3 The upper bounds 3 1 1, 3 2 1, 3 3 1, 3 3 3, indicate the "nice" computational character of Codd-tables and the particularities of the uniqueness problem Let us now argue why our results are syntactically tight For the lower bounds 3 1 2, 3 2 2, we necessarily use an e-table, (see 3 1 1, 3 2 1) For 3 1 3, 3 2 3, we necessarily use an i-table for the same reasons For 3 3 2 we necessarily use a c-table, (see 3 3 1) For the lower bounds with views 3 1 4, 3 2 4, we use positive existential queries on Codd-tables, our most restricted class of queries The exception is the query for 3 3 4, which necessarily is positive existential with $\neq$, (see 3 3 3)

In Section 4 we complete the study of the containement problem This generalizes membership and uniqueness Our bounds again are matching upper (Theorem 4 1) and lower bounds (Theorems 4 2 and 4 3) Using the previous section together with this section, we exhaustively examine all possibilities for the containement problem It is easy to see that our theorems completely cover all the cases of Figure 2

Our upper bounds 4 1 1, 4 1 2, 4 1 3, use homomorphism arguments and are further indications of the computational properties of Codd-tables Let us now argue why our results are syntactically tight Our lower bounds for views 4 3 2, 4 3 3 use only positive existential queries and Codd-tables Our lower bound for views 4 3 1 necessarily uses one e-table as superset, (see 4 1 1) Finally Theorem 4 2 is the hardest technically and necessarily uses one i-table as a superset, (see 4 1 3, 4 1 2) Theorem 4 2 is that "containement is Π_2^p-complete, even if the subset possible worlds are represented by a Codd-table and the superset possible worlds are represented by an i-table"

In Section 5 we address the certainty problem (Theorem 5 1) and the bounded possibility problem (Theorem 5 2) The upper bound 5 1 1 is old, the lower bound 5 1 2 is new The upper bound 5 2 1 matches the lower bounds 5 2.2, 5 2 3 Section 6 has our conclusions and open questions (In the theorems the shorthand rep stands for represented)

2. Definitions and Notation

Complete Information Databases

Let the *domain* be the countably infinite set of *constants* {0,1,2, ,c, } A *relation* R of *arity* (a) is some *finite* subset of the $(\text{domain})^a$, where $0 \leq a$ integer A member of a relation is therefore a tuple t of constants (or *fact*) 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_n has arity (a_i) $i=1, ,n$ The relation R above is thus an instance of arity (a) A *query* q of *arity* $(a_1, ,a_n) \rightarrow (b_1, ,b_m)$ is a function from instances to instances of appropriate arities 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, when its arity is clear from the context we will also use the symbol - to denote an identity query Another example of queries are *boolean* queries, where m=1 and $b_1=0$ The output of boolean queries is either the *empty set* (with which we encode *false*) or the nonempty relation of arity (0) consisting of the *empty fact* (with which we encode *true*) We assume a fixed encoding for facts and instances With some abuse of notation, when we say that fact t is in instance I we presume that the relation of I, where t belongs, is also specified Given a query q we say that the *data-complexity* of q is the complexity of the formal language

{ (t,I) | fact t is in instance q(I) }

The family of queries QPTIME characterizes all efficient computations on instances, it consists of those queries whose data complexity is in PTIME [3] All the queries examined in this paper are in QPTIME This family contains many subfamilies of independent interest In particular, we refer to three of these subfamilies

(1) The *positive existential* queries These are the simplest, most practical, and most investigated queries [15] They can be expressed exactly using relational expressions with operators *project, natural join, union, renaming, positive select* We will express them here using first order formulas with equality, but without universal quantification or negation In the conventional fashion, the relation symbols R, will denote relations R, , which are the finite interpretations of these symbols Because negation is not allowed $\neq$ cannot be used The positive existential queries are further extended by the following two incomparable subfamilies through "negation" and "recursion"

(2) The *first order* queries These are the *domain relational calculus* queries of [15] We will express them here using formulas of a first order formulas with equality, in the conventional fashion Since these queries have negation $\neq$ may be used

(3) The DATALOG queries These are the queries most common in *deductive databases* and can be thought of as Horn clause recursive rules [2] For uniformity they

will be expressed here as fixpoints of positive existential queries We assume they do not contain $\neq$

Incomplete Information Databases

An *incomplete information database* is a set of instances A central issue for such sets of instances is their representation A number of algebraic representations have been developed, so that, sets of instances can be queried in a fashion similar to complete information databases, i e , single instances We will use the term table (short for Codd-table [4]) for the simplest algebraic structure used for such a representation Based on tables we define tables with conditions (i e , c-,g-,e-,i- tables), as well as, views of sets of instances We assume that {x,y,z,u,v,w, } is a countably infinite set of *variables*, disjoint from the set of constants

A *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 is a tuple of constants and variables appearing as a row of T

A *condition* is a *conjunct* of *equality atoms* (of the form x=y, x=c) and *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

(i) a *global* condition Φ is associated with the entire table T

(ii) a *local* condition $\Phi(t)$ is associated with one tuple t of table T

Note that conditions associated in table T and its tuple t may contain variables not appearing in T or t We omit explicitly listing the condition *true*, x=x Also the set of variables appearing in a table and its associated conditions is finite because of the finiteness of the table and of the conjuncts

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

A *c-table* (short for conditioned-table) is a table T together with an associated global condition Φ and an associated local condition $\Phi(t)$ for each tuple t of T Recall that, by convention, a missing condition is atom *true* A *g-table* (short for global table) is a c-table without local conditions An *i-table* (short for inequality table) is a g-table, whose global condition consists entirely of inequality atoms An *e-table* (short for equality table) is a g-table, whose local condition consists entirely of equality atoms Clearly a table is also an e-table and an i-table without global condition

Definition 1 A given c-table represents a set of instances $\mathcal{I}$ Let the given c-table consist of, (1) a table T of arity (a), and (2) a global condition Φ, and (3) local conditions $\Phi(t)$, 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, such that, relation R consists exactly of those facts $\sigma(t)$ for which σ satisfies $\Phi(t)$ }

For the important special case of a table T all valuations are satisfying and $\mathcal{I}$={ R | R=$\sigma(T)$ for some σ } Also for a g-table $\mathcal{I}$={ R | R=$\sigma(T)$ for some σ satisfying Φ } Note that, in a g-table, if the global condition is unsatisfiable, (which can be checked in PTIME because a global condition is a conjunction), then $\mathcal{I}$ is the *empty set* If there are satisfying valuations for the global conditions, 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 [6]), then $\mathcal{I}$ consists of a relation with only the *empty fact* of arity (a)

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 the sets of variables appearing in each table $T_1, \ ,T_n$ are pairwise disjoint, relationships between these variables can be established through the conditions

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 $(b_1, \ ,b_m)$
$q(\mathcal{I}) = \{ q(I) \mid I \text{ instance in } \mathcal{I} \}$

Our most general representation of a set of instances is thus a *set of views* of $\mathcal{I}$ through q This is the most general case because of the possibility of using identity queries of any arity Finally, note that our possible instances (worlds) are "closed worlds" since they correspond to valuations of tables all of whose tuples are specified in our representations

The Problems

We now describe some basic computational questions about incomplete information databases All of these questions can be answered in PTIME for complete information databases, because the queries used are in QPTIME

If $q_0(\mathcal{I}_0)$ and $q(\mathcal{I})$ are two sets of instances the first obvious question is whether one set is contained in the other This is the *containement* problem CONT We assume that there are no variables in common in these two representations If $\mathcal{I}_0$ happens to be the singleton set $\{I_0\}$ represented by a given instance I_0 then wlog we may assume that q_0 is an identity query, (because $q_0(I_0)$ may be computed in PTIME) In this case we have the *membership* problem MEMB, i e , is a given instance I_0 a possible instance of $q(\mathcal{I})$ The dual case is where $\mathcal{I}$ is represented by instance I For this dual case we have the *uniqueness* problem UNIQ, i e , is every possible instance of $q_0(\mathcal{I}_0)$ an element of {I}

The three questions above deal with entire instances What about possible or certain occurrences of patterns in a set in instances? If P is a given set of facts of size k we typically ask
Do the facts in P appear together in some possible instance, this is the *possibility* problem POSS
Do the facts in P appear in all possible worlds, this is the *certainty* problem CERT
Since our possible worlds descriptions include views, the POSS and CERT problems involve querying incomplete databases

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, $\Phi(t)$) We also use capital letters for sets of facts, (e g , R, I, $\mathcal{I}$, and P), which can be of unbounded size In our framework queries, and therefore arities, are *fixed* parameters, for this we use small letters, (e g , q,a,b) A single fact and tuple in this frameowrk has fixed width, for this we use small letter t We use * instead of size k if k is unbounded The formal definitions follow

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

MEMB(q)
parameter q
input c-tables representing $\mathcal{I}$, instance I_0
question is I_0 in set $q(\mathcal{I})$?

$UNIQ(q_0)$
parameter q_0
input c-tables representing $\mathcal{I}_0$, instance I
question is $q_0(\mathcal{I}_0)$ singleton set {I} ?

POSS(k,q)

parameter k,q

input c-tables representing $\mathcal{I}$, set of facts P of size k

question $\exists$I in q($\mathcal{I}$), s t , all facts of P are facts of I ?

POSS(*, q) is the same question where k is no longer a parameter

CERT(k,q)

parameter k,q

input c-tables representing $\mathcal{I}$, set of facts P of size k

question $\forall$I in q($\mathcal{I}$), all facts of P are facts of I ?

CERT(*, q) is the same question where k is no longer a parameter

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 grows 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 queries q_0, q in QPTIME we have 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, (6) CERT(*,q) is polynomially equivalent to CERT(1,q)

For (1) we reason that every valuation for $\mathcal{I}_0$ corresponds to a valuation for $\mathcal{I}$, $\forall\exists$ quantification For (2) and (4) we guess the right valuation, $\exists$ quantification For (3) and (5) we reason about all valuations, $\forall$ quantification In order to answer CERT(k,q) all we have to do is repeat CERT(1,q) k times, this gives us (6) 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

3. Membership, Uniqueness and Unbounded Possibility

We start with a classification for the membership problem Note that tables have a polynomial-time membership problem This is like instances and unlike e-tables, i-tables, and views of tables The reduction for Theorem 3 1 4 is complicated by the requirement for a positive existential query on a single table

Theorem 3.1 Let $\mathcal{I}$ be as in the definition of MEMB, then

(1) MEMB(-) is in PTIME if $\mathcal{I}$ is represented by a vector of tables

(2) MEMB(-) is NP-complete even if $\mathcal{I}$ is represented by a single e-table

(3) MEMB(-) is NP-complete even if $\mathcal{I}$ is represented by a single i-table

(4) $\exists$q positive existential query, s t , MEMB(q) is NP-complete even if $\mathcal{I}$ is represented by a single table

Proof Sketch (1) This upper bound is derived by a reduction to the problem of *bipartite graph matching* [6] 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 (Proposition 2 1) the rest of the proof consists of reductions of NP-hard problems to MEMB

(2) Reduction of *graph 3-colorability* [6] using an arity two e-table and a size six instance

(3) Reduction of graph 3-colorability using an arity one i-table and a size three instance

(4) Reduction of graph 3-colorability using an arity (6) table, an unbounded size instance of arity (3), and the query q of arity (6)→(3) described by the following formula

q = { xyz | ϕ(xyz)$\vee\psi$(xyz) }, where ϕ(xyz) is

$x=0\wedge y=0\wedge\exists x_1 \;\; x_4$ [$R(1x_1x_2x_3x_4z)\wedge R(0000x_2x_4)$]

ψ(xyz) is

$\exists x_1 \;\; x_5$ [$R(1xx_1x_2x_3y)\wedge R(1xx_1x_4x_5z)$] $\vee$

[$R(1xx_1x_2x_3y)\wedge R(1x_4x_5xx_1z)$] $\vee$

$[R(1x_2x_3xx_1y) \wedge R(1xx_1x_4x_5z)] \vee$
$[R(1x_2x_3xx_1y) \wedge R(1x_4x_5xx_1z)]$
(Q E D)

The next theorem ındicates how sımılar unbounded possıbılıty ıs to membershıp, from a computatıonal poınt of vıew The two problems are by defınıtıon dıfferent problems

Theorem 3.2 Let I be as ın the defınıtıon of POSS, then

(1) POSS(*,-) ıs ın PTIME ıf I ıs represented by a vector of tables

(2) POSS(*,-) ıs NP-complete even ıf I ıs represented by a sıngle e-table

(3) POSS(*,-) ıs NP-complete even ıf I ıs represented by a sıngle ı-table

(4) $\exists$q posıtıve exıstentıal query, s t , POSS(*,q) ıs NP-complete even ıf I ıs represented by a sıngle table

Proof Sketch (1) The argument ıs a varıatıon on that of Theorem 3 1 1

(2) Reductıon of 3CNF *satısfıablıty* [6] usıng an arıty three e-table and an unbounded set of facts

(3) Reductıon of 3CNF satısfıabılıty usıng an arıty two ı-table and an unbounded set of facts

(4) Reductıon and query are ıdentıcal wıth those of Theorem 3 1 4 (Q E D)

The last theorem of thıs sectıon deals wıth unıqueness, whıch although dual to membershıp from a defınıtıon poınt of vıew, ıs quıte dıfferent from membershıp Note the role of $\neq$

Theorem 3 3 Let I_0 be as ın the defınıtıon of UNIQ, then

(1) UNIQ(-) ıs ın PTIME ıf I_0 ıs represented by a vector of g-tables

(2) UNIQ(-) ıs coNP-complete even ıf I_0 ıs represented by a sıngle c-table

(3) UNIQ(q) ıs ın PTIME if q ıs posıtıve exıstentıal and I_0 ıs represented by a vector of e-tables

(4) $\exists$q posıtıve exıstentıal wıth $\neq$, s t , UNIQ(q) ıs coNP-complete even ıf I_0 ıs represented by a sıngle table

Proof Sketch (1) For thıs part the proof ıs by ınspectıon of the matrıx representatıon of the g-tables

(2) Reductıon of 3DNF *tautology* [6] usıng an arıty one table and an ınstance of sıze two

(3) For thıs we use [10] to get a representatıon of all possıble worlds resultıng from the query q Thıs representatıon can be constructed and because of lack of negatıon can be tested trıvıally for unıqueness

(4) Reductıon of *graph non 3-colorabılıty* usıng an arıty three table, the arıty one ınstance {0,1}, and the query q of arıty (3)→(1) descrıbed by the followıng formula

$q = \{v \mid v=0 \vee (v=1 \wedge \exists xyz\, [R(1xy) \wedge R(0xz) \wedge R(0yz)])$
$\vee (v=1 \wedge \exists yz\, [R(0yz) \wedge z \neq 1 \wedge z \neq 2 \wedge z \neq 3])$
(Q E D)

4. Containement

For our upper bounds we use homomorphısms to refıne Proposıtıon 2 1

Theorem 4.1 Let the ınputs I_0, I be as ın the defınıtıon of problem CONT, then

(1) CONT(q_0,-) ıs ın coNP ıf I ıs represented by a vector of tables

(2) CONT(-,-) ıs ın NP ıf I_0 ıs represented by a vector of g-tables and I by a vector of e-tables

(3) CONT(-,-) ıs ın PTIME ıf I_0 ıs represented by a vector of g-tables and I by a vector of tables

Proof Sketch (1) Consıder the negatıon of thıs problem Thıs negatıon ıs ın NP because all one has to do ıs guess a valuatıon dısprovıng the contaınement and do a PTIME computatıon to produce an ınstance dısprovıng the contaınement Fınally use Theorem 3 1 1 sınce I ıs represented by a vector of tables and membershıp then ıs ın PTIME

(2) Fırst ıncorporate the equalıtıes of the condıtıons ın

the representation of I_0 Now think of the variables in this representation as distinct constants, this gives rise to instance I_0 Using a *homomorphism* argument reduce the problem to MEMB(-), where the input instance is I_0, and employ Theorem 3 1 2

(3) Use the same argument as the previous case, but now employ Theorem 3 1 1 since I is represented by a vector of tables and membership then is in PTIME (Q E D)

Our lower bounds together with the other results of Section 3 and this section, exhaustively cover all cases of Figure 2 In the outline (Section 1) we argued why these are syntactically tight lower bounds Theorem 4 2 is quite interesting given 4 1 2 and 4 1 3

Theorem 4.2 Let the inputs I_0, I be as in the definition of problem CONT then CONT(-,-) is Π_2^P-complete even if I is represented by a single i-table and I_0 by a single table

Proof Sketch Reduction from the appropriate version of the *quantified boolean formula* [14] problem ∀∃3CNF Unbounded size tables of arity (4) are used Encoding 3CNF satisfiability in the i-table (for I) is straightforward What is more interesting is using the table (for I_0) to force the assignments to variables The following example captures the intuition for this mechanism

Let us examine a table of arity (3) consisting of tuples {001, 122, 133, $1xx_1$} and

an i-table of arity (3) consisting of tuples {001, 122, 133, vzz_1, uyy_1} where $u \neq v \wedge z \neq 3 \wedge y \neq y_1$

The relations described by the first table are a subset of relations described by the second table, moreover, (i) if $x=x_1$ then u=0 and v=1 in the equal instance of the i-table, (ii) if $x=3\neq x_1$ then u=1 and v=0 in the equal instance of the i-table, (iii) $x\neq 3$, $x\neq x_1$ then u=1,v=0 and u=0,v=1 are both possible in equal instances of the i-table This construction provides the necessary encoding for ∀ quantification (Q E D)

The remaining cases are covered by Theorem 4 3 Its proof involves reduction techniques, which are simpler than those used for Theorem 4 2, and we therefore omit them in this abstract

Theorem 4.3 Let the inputs I_0,I be as in the definition of problem CONT and let I_0 be represented by a single table, then

(1) $\exists q_0$ positive existential query, s t , CONT(q_0,-) is Π_2^P-complete even if I is rep by a single e-table

(2) $\exists q_0$ positive existential query, s t , CONT(q_0,-) is coNP-complete even if I is rep by a single table

(3) $\exists q$ positive existential query, s t , CONT(-,q) is Π_2^P-complete even if I is rep by a single table

5. Certainty vs Bounded Possibility

Much work has already been done in the area of searching for certain answers In particular, when the query is positive and the incomplete database is represented as a g-table [13, 16, 10] The upper bound of Theorem 5 1 1 follows directly from the central results of [16, 10, 13] 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 The lower bound of Theorem 5 1 2 is a refinement of the lower bound in [16] (also, Theorem 5, IBM Res Rep RJ 4874) from an e-table to a table representation

The problem of searching for possible answers of bounded size has received less attention The upper bound of Theorem 5 2 1 is a consequence of the fact that c-tables are *representation systems* in the sense of [10] 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 in Theorems 5 2 2 and 5 2 3 Once again the interest of the lower bounds lies in the syntactic constraints, e g , the query of 5 2 3 uses monadic fixpoints on (unconditioned) tables

Theorem 5 1 Let $\mathcal{I}$ be as in the definition of CERT(*,q), then
(1) [16, 10] If q a DATALOG query and $\mathcal{I}$ is represented by a vector of g-tables then CERT(*,q) is in PTIME
(2) $\exists$q first order query, s t , CERT(*,q) is coNP-complete even if $\mathcal{I}$ is represented by a table

Proof Sketch (2) Reduction of 3DNF tautology Let $\{C_i\}$ be the given set of clauses and $\{X_j\}$ the given set of variables, then construct a table T with variables $\{v_{i,k}\}$ and tuples the set $\{i v_{i,k} j 1 \mid X_j$ appears in position k of $C_i\} \cup \{i v_{i,k} j 0 \mid \neg X_j$ appears in position k of $C_i\}$ The query asked is a boolean query $q = \{ c \mid \phi \}$ We want the fact c to be certain iff the original 3DNF formula is a tautology, for this ϕ is as follows

$$[\exists x y z v x_1 y_1 z_1 v_1 \quad R(xyzv) \wedge R(x_1 y_1 z_1 v_1) \wedge z = z_1 \wedge y \neq y_1]$$
$$\vee$$
$$[\forall x y z v \exists x_1 y_1 z_1 v_1 \quad R(xyzv) \Rightarrow \{R(x_1 y_1 z_1 v_1) \wedge x = x_1 \wedge ((y_1 = 1 \wedge v_1 = 1) \vee (y_1 \neq 1 \wedge v_1 = 0))\}]$$

(Q E D)

Our final theorem is about bounded possibility

Theorem 5.2 Let $\mathcal{I}$ be as in the definition of POSS(k,q), then
(1) If q is a positive existential query and $\mathcal{I}$ is represented by a vector of c-tables then POSS(k,q) is in PTIME
(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 Sketch (1) Transform the given positive existential view of c-tables into other equivalent c-tables, that are 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 It is then simple to find whether a bounded pattern is possible
(2) Similar to the reduction of Theorem 5 1 2
(3) 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 of arity (2,2,1)$\rightarrow$(1)

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

q with input instance (R_0, R_1, R_2) is the least fixpoint of q_1, which contains R_0 (Q E D)

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 tables to the more complex c-tables In this setting we analysed containement, membership, uniqueness, 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 good characterizations for the MEMB lower bound Theorem 3 1 4 would be interesting These would be positive existential views of Codd-tables whose membership questions are in PTIME

References

1. Abiteboul S , Grahne, G Update Semantics for Incomplete Databases Proceedings of the 11th VLDB, Stockholm, 1985

2 Chandra, A K , Harel, D "Horn Clause Programs and Generalizations" *J Logic Programming 2* (1985), 1-15

3 Chandra, A K , Harel, D "Structure and Complexity of Relational Queries" *JCSS 25*, 1 (1982), 99-128

4 Codd, E F "Extending the Database Relational Model to Capture More Meaning" *ACM Transactions on Database Systems 4*, 4 (December 1979), 397-434

5. Cosmadakis, S S "The Complexity of Evaluating Relational Queries" *Information and Control 58* (1983), 101-112

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

7. Grahne, G Dependency Satisfaction in Databases with Incomplete Information Proceedings of the 10th VLDB, Singapore, 1984

8. Honeyman, P , Ladner, R , Yannakakis, M "Testing the Universal Instance Assumption" *Inform Process Lett 10*, 1 (1980), 14-19

9. Imielinski, T On Algebraic Query Processing in Logical Databases In *Advances in Database Theory, Vol 2*, Gallaire, H and Minker, J , Eds , Plenum Press, 1984

10 Imielinski, T , Lipski, W Jr "Incomplete Information in Relational Databases" *JACM 31*, 4 (October 1984), 761-791

11. Lipski, W Jr "On Databases with Incomplete Information" *JACM 28*, 1 (January 1981), 41-70

12 Maier, D , Sagiv, Y , Yannakakis, M "On the Complexity of Tesing Implications of Functional and Join Dependencies" *JACM 28*, 4 (October 1981), 680-695

13. Reiter, R A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values Tech Rep 83-11, Univ of British Columbia, 1983

14 Stockmeyer, L "The Polynomial Time Hierarchy" *TCS 3*, 1 (1976), 1-22

15. Ullman, J D *Principles of Database Systems* Computer Science Press, 1983

16 Vardi, M Y Querying Logical Databases Proceedings of the 4th PODS, ACM, 1985

17. Vardi, M Y Complexity of Relational Query Languages Proceedings of the 14th STOC, ACM, 1982

18 Zaniolo, C Database Relations with Null Values Proceedings of the 1rst PODS, ACM, 1982

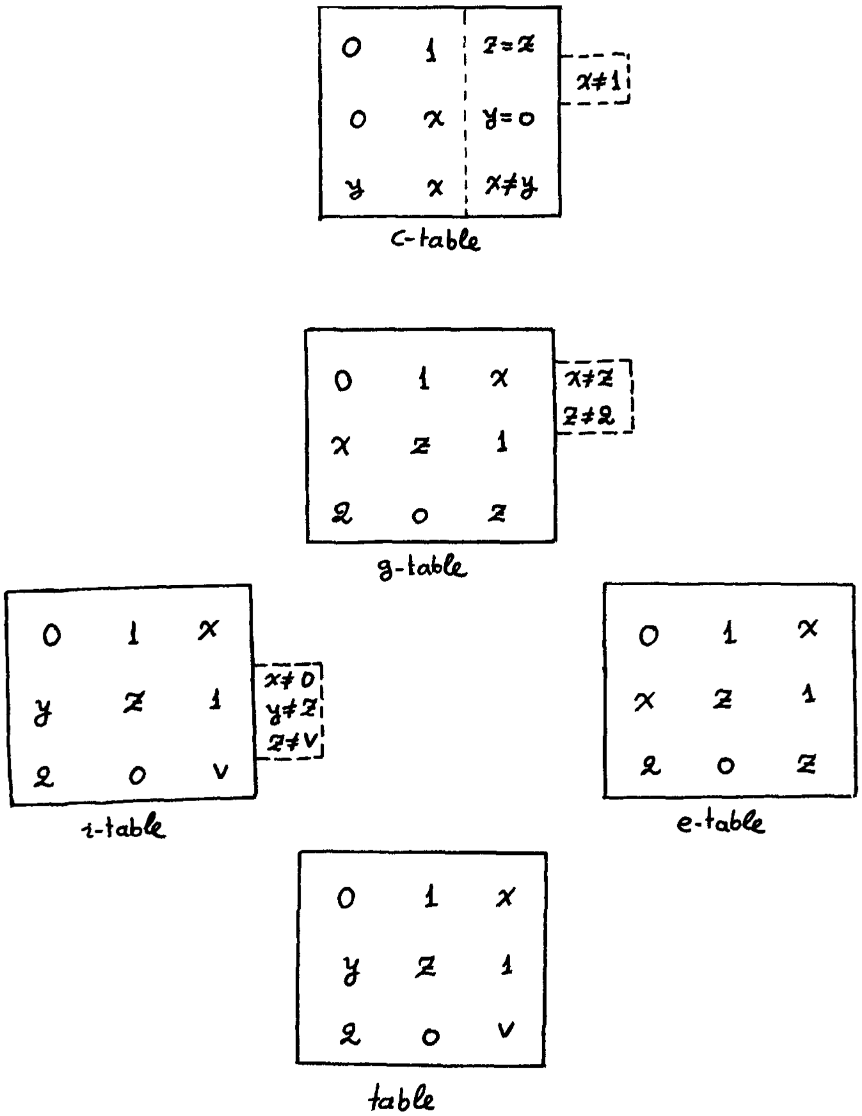

Figure 1 Representations

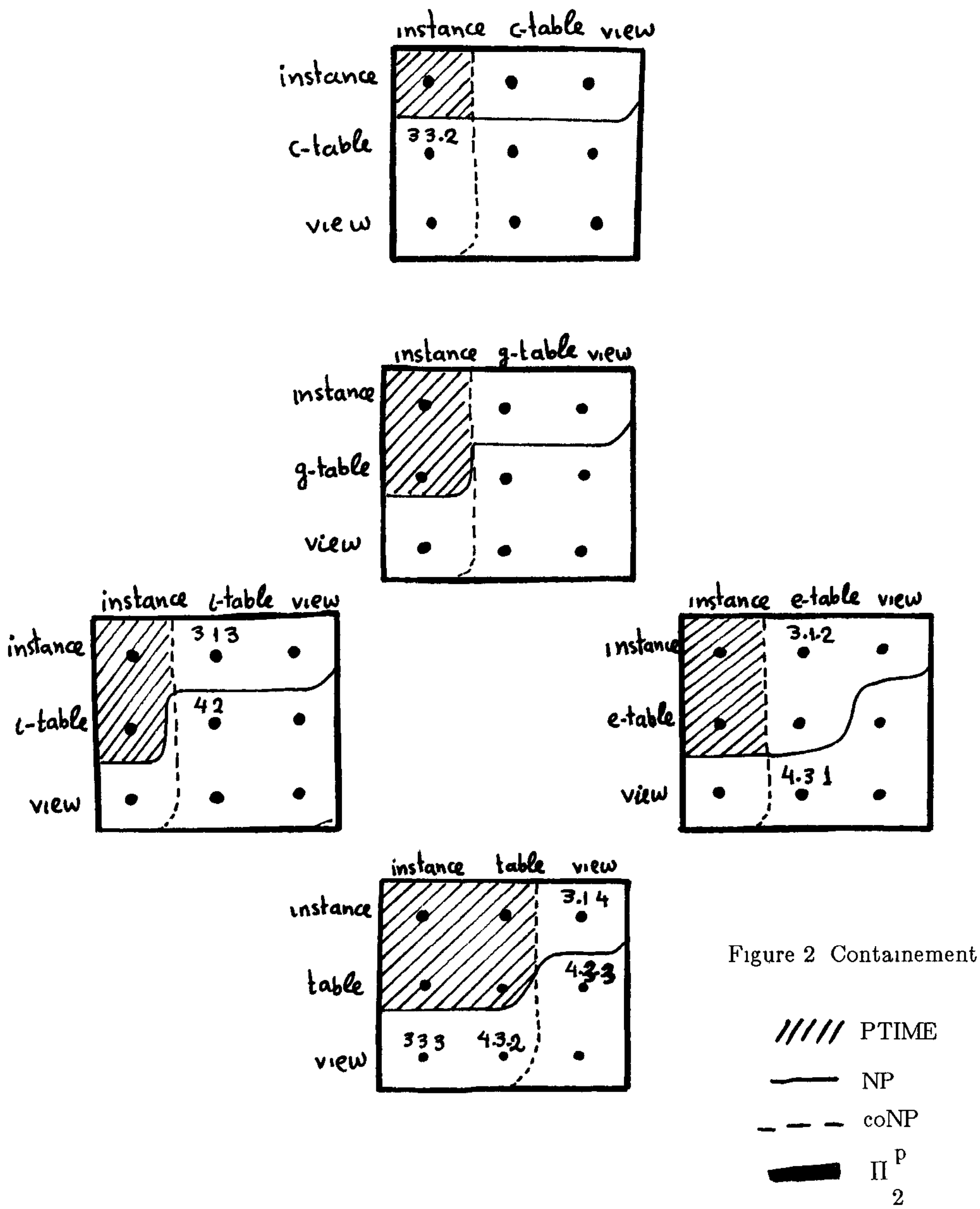

Figure 2 Containement