To appear in the Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Conference on Principles of Database Systems (PODS), Seattle, WA, June 1998.

Complexity of Answering Queries Using Materialized Views

Serge Abiteboul*
INRIA-Rocquencourt
Serge.Abiteboul@inria.fr

Oliver M. Duschka†
Socratix Systems, Inc.
oliver@socratix.com

Abstract

We study the complexity of the problem of answering queries using materialized views. This problem has attracted a lot of attention recently because of its relevance in data integration. Previous work considered only conjunctive view definitions. We examine the consequences of allowing more expressive view definition languages. The languages we consider for view definitions and user queries are: conjunctive queries with inequality, positive queries, datalog, and first-order logic. We show that the complexity of the problem depends on whether views are assumed to store all the tuples that satisfy the view definition, or only a subset of it. Finally, we apply the results to the view consistency and view self-maintainability problems which arise in data warehousing.

1 Introduction

The notion of materialized view is essential in databases [34] and is attracting more and more attention with the popularity of data warehouses [28]. The problem of answering queries using materialized views [24, 6, 10, 5, 43, 30, 26, 36, 12, 14, 11, 25] has been studied intensively. We propose a systematic study of its complexity. We also briefly consider the related problems of view consistency and view self-maintainability [19]. Our results exhibit strong connections with two among the most studied problems in database theory, namely query containment [7, 33, 23, 31, 9, 21, 13, 27] and incomplete information querying, e.g. [20, 2]. Indeed, the works most closely related to our complexity results are perhaps those of van der Meyden [40, 41, 42] and Vardi [38] on (indefinite) database queries.

*Part of the work performed when the author was visiting Stanford University.

†Work performed as part of Ph.D. thesis research at Stanford University.

Our results highlight the basic roles played by negation (and in its weak form inequality) and recursion, and a crucial difference between *open* and *closed world assumption* in the view definition.

The main focus of the paper is the study of the *data complexity* of the problem of *answering queries using materialized views*. More precisely, the problem is for a fixed view definition and a fixed query, given a view instance I and a tuple t, is t a *certain* answer, i.e. is t in the answer to the query on the database no matter which is the database yielding the view instance I. This articulation of the problem highlights the main parameters: (i) What are the database and the view models? (ii) What are the query and the view definition languages? (iii) Is *yielding* assuming an open or a closed world?

In the present paper, we use the relational model for the database and the view model. However, our work strongly suggests moving towards an incomplete information model, e.g. conditional tables [20]. Indeed, we will briefly show how these tables can be used for solving the problem in most solvable cases. For the query and view definition languages, we consider the most popular formal query languages, namely conjunctive queries, conjunctive queries with inequality, positive queries, datalog, and first-order logic. We focus on *certain* answers, i.e. tuples that are in the answer for *any* database yielding this particular view instance.

Not surprisingly, our results indicate that recursion and negation in the view definition lead to undecidability. Somewhat also expectedly, we show that the closed world assumption sharply complicates the problem. For instance, under the open world assumption the certain answers in the conjunctive view definitions/datalog queries case can be computed in polynomial time. On the other hand, already the conjunctive view definitions/conjunctive queries case is co-NP-complete under the closed world assumption. This is an a-posteriori argument for a number of recent works that postulate an open world interpretation of views. Perhaps more unexpectedly, we prove that inequalities (a very weak form of negation) lead to intractability. Even under the open world assumption, adding inequalities to the queries, or disjunction to the view definitions makes the problem co-NP-hard.

2 The problem

In this section, we present the problem. We assume some familiarity with database theory [34, 1]. We start with a database instance D, a view definition $\mathcal{V}$, and a view instance I. The database consists of a set of relations and so does the view. Now, given a query $\mathcal{Q}$, we

would like to compute $\mathcal{Q}(D)$. However, we assume that we ignore D and only have access to I, so we will try to get the best possible estimate of $\mathcal{Q}(D)$ given I.

Let us be more precise. Under the *closed world assumption* (CWA), the view instance I stores *all* the tuples that satisfy the view definitions in $\mathcal{V}$, i.e. $I = \mathcal{V}(D)$. Under the *open world assumption* (OWA), on the other hand, instance I is possibly incomplete and might only store *some* of the tuples that satisfy the view definitions in $\mathcal{V}$, i.e. $I \subseteq \mathcal{V}(D)$. As we can see from the following example, in reasoning about the underlying database, it makes a difference whether we are working under the open or closed world assumption.

Example 2.1 Consider the following view definition where the view consists of two relations:

$$v_1(X) :- p(X, Y)$$
$$v_2(Y) :- p(X, Y)$$

and assume that the view instance consists of $\{v_1(a),\ v_2(b)\}$. Under OWA, we only know that some p tuple has value a as its first component, and some (possibly different) p tuple has value b as its second component. Under CWA, however, we can conclude that all p tuples have value a as their first component and value b as their second component, i.e. p contains exactly the tuple $\langle a, b\rangle$. □

Given some view definition and a view instance, observe that there may be a number of possible databases, i.e. database instances that yield this view instance for this view definition. So, we can think of the database as the *incomplete database* [20] consisting of this set of possible databases. To answer a query, we focus on *certain* answers, i.e. on tuples that are in the answer for each possible database. As seen in Example 2.1, this depends on whether we are assuming an open or a closed world. Indeed, an answer that is certain under OWA is also certain under CWA, but the converse does not hold in general. For instance, in the previous example, the query "is $\langle a, b\rangle$ certainly in p?" is answered positively under CWA and negatively under OWA. In fact, we will show that computing certain answers under CWA is harder than under OWA. The following definition formalizes the concept of certain answer under both assumptions:

Definition 2.1 (**certain answer**) Let $\mathcal{V}$ be a view definition, I be an instance of the view, and $\mathcal{Q}$ a query. A tuple t is a *certain answer under OWA* if t is an element of $\mathcal{Q}(D)$ for each database D with $I \subseteq \mathcal{V}(D)$. A tuple t is a *certain answer under CWA* if t is an element of $\mathcal{Q}(D)$ for each database D with $I = \mathcal{V}(D)$. □

We briefly recall the query languages we consider and the standard notion of complexity we use.

2.1 Query and view languages

A *datalog rule* is an expression of the form:

$$p(\bar{X}) :- p_1(\bar{X}_1), \ldots, p_n(\bar{X}_n)$$

where p, and $p_1, \ldots, p_n$ are predicate names, and $\bar{X}$, $\bar{X}_1, \ldots, \bar{X}_n$ are tuples of variables and constants. Each variable in the head of a rule must also occur in the body of the rule. A *datalog* query is a finite set of datalog rules. The notion of *recursive* datalog query/rule is defined in the standard way. A *conjunctive query* (CQ) is a single non-recursive datalog rule. If the body of a conjunctive query is allowed to contain the inequality predicate ($\neq$), then the query is called a *conjunctive query with inequality* ($CQ^{\neq}$). Every variable in a query with inequality must occur at least once in a relational predicate. A *positive query* (PQ) is a non-recursive datalog query together with one particular predicate defined by the query. The query language $PQ^{\neq}$ is obtained by also allowing $\neq$. Finally, *first-order queries* (FO) are defined in the standard way.

A *materialized view*, also called *view instance*, is the stored result of previously executed queries. A *view definition* $\mathcal{V}$ therefore consists of a set of queries defining a finite set of predicates. So, for a query language $\mathcal{L}$, we write $\mathcal{V} \subseteq \mathcal{L}$ to denote the fact that each predicate in the view is defined using a query in $\mathcal{L}$.

2.2 Data complexity

We will be interested in the *data complexity* of the problem of computing certain answers under the open and closed world assumption. The *data complexity* is the complexity of the problem as a function of the size of the view instance. We will also refer to the *query* and *combined complexity* of the problem. The *query complexity* is the complexity of the problem as a function of the size of the view definition $\mathcal{V}$ and the query $\mathcal{Q}$. The *combined complexity* is the complexity of the problem as a function of these two arguments plus the size of the view instance. (These three notions are due to [37].) In the remaining of the paper, when we discuss complexity, we always mean data complexity unless specified otherwise.

In Section 3, we prove that the problem is in co-NP for a wide range of cases. We also highlight some connections with conditional table querying. In Section 4, we examine the complexity of the problem of computing certain answers under OWA and in Section 5 under CWA. In Section 6, we consider view self-maintainability and view consistency.

3 Using conditional tables

In this section, we briefly sketch a solution to the problem for the open and closed world assumption, when the view definition is in $PQ^{\neq}$ and the query is in $datalog^{\neq}$. We also present an effective procedure based on conditional tables [20] which were introduced to represent incomplete information. Indeed, a main purpose of this section is to highlight the strong links between our problem and that of querying incomplete databases.

First we see next that, for $PQ^{\neq}$ views and $datalog^{\neq}$ queries, the problem is in co-NP. So within these limits, it will suffice in the following of the paper to prove co-NP-hardness to establish co-NP-completeness.

Theorem 3.1 *For $\mathcal{V} \subseteq PQ^{\neq}$, $\mathcal{Q} \in datalog^{\neq}$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA or CWA, is in co-NP.*

Proof. We prove the claim first for OWA. Assume that t is not a certain answer. Then there is a database D with $I \subseteq \mathcal{V}(D)$ and t is not in $\mathcal{Q}(D)$. Let n be the total number of tuples in I and let

k be the maximal length of conjuncts in the view definitions. Each tuple in I can be generated by at most k tuples in D. Therefore, there is a database $D' \subseteq D$ with at most nk tuples such that still $I \subseteq \mathcal{V}(D')$. Because t is not in $\mathcal{Q}(D)$ and $\mathcal{Q}$ is monotone, t is also not in $\mathcal{Q}(D')$. It follows that there is a database D' whose size is polynomially bounded in the size of I and $\mathcal{V}$ such that $I \subseteq \mathcal{V}(D')$, and t is not in $\mathcal{Q}(D')$. Moreover, checking that $I \subseteq \mathcal{V}(D')$ and that t is not in $\mathcal{Q}(D')$ can be done in polynomial time.

For CWA, the proof is essentially the same with $I = \mathcal{V}(D)$ in place of $I \subseteq \mathcal{V}(D)$. □

We next turn to an effective way of computing the certain answers. The intuition is to represent, given a view instance, the set of possible databases or more precisely a sufficient set of possible databases by a conditional table, and then query the conditional table using the techniques introduced by Imielinski and Lipski [20]. Due to space limitations, we refer to [20, 1] for a definition of conditional tables. Intuitively, a conditional table is a database instance which might have variables as entries in its tuples. There is also a global condition [1] on the set of variables and for each tuple, a local condition controlling the actual presence of the tuple. A *possible database* for a table T is obtained by choosing a valuation satisfying the global condition, keeping only those tuples with a true local condition and valuating the variables in those tuples.

The following result shows how the problem of querying materialized views can be reduced to the problem of querying conditional tables, thereby highlighting the strong connection between materialized views and incomplete databases. Due to space limitations, we do not give a proof of the result. The construction used in the proof is illustrated by an example.

Theorem 3.2 *Let $\mathcal{V} \subseteq PQ^{\neq}$ and let I be a view instance. Then one can construct a conditional table (with global condition) T_{owa}, resp. T_{cwa}, such that for each $datalog^{\neq}$ query $\mathcal{Q}$, the certain answers to $\mathcal{Q}$ using view instance I under OWA, resp. CWA, are exactly the certain answers to $\mathcal{Q}$ given the incomplete database specified by T_{owa}, resp. T_{cwa}.*

The previous theorem provides an algorithm of evaluating a query on a database given some materialized view in co-NP time: compute the corresponding conditional table and then evaluate the query on the table using the techniques in [20].

Example 3.1 Suppose the view is specified by:

$$v(0, Y) \ :- \ p(0, Y)$$
$$v(X, Y) :- \ p(X, Z), \ p(Z, Y)$$

and the view instance consists of $\{v(0,1), v(1,1)\}$. Then there are two different ways to obtain the first tuple and only one for the second. This yields the following conditional table for p (the global condition is *true*):

0	1	$w = 1$
0	z	$w \neq 1$
z	1	$w \neq 1$
1	u	$true$
u	1	$true$

This is the table needed for OWA. For CWA, we have to introduce the constraints that the view does not hold for any other tuple. One finds the following (complete) table for p:

0	1	$true$
1	1	$true$

□

	— query —				
views	CQ	$CQ^{\neq}$	PQ	$datalog$	FO
CQ	PTIME	co-NP	PTIME	PTIME	undec.
$CQ^{\neq}$	PTIME	co-NP	PTIME	PTIME	undec.
PQ	co-NP	co-NP	co-NP	co-NP	undec.
$datalog$	co-NP	undec.	co-NP	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

Figure 1: *Data complexity of the problem of computing certain answers under the open world assumption.*

We briefly discuss some aspects of the construction of the conditional table. Consider the simplest case, i.e. a conjunctive query view under OWA. Intuitively, the table is constructed by "skolemizing" the variables in the conjunctive query in a standard manner such as [16]. Now, we obtain a conditional table. This is not quite a representation of the possible databases since a possible database may contain additional tuples. But with respect to *certain* answers, we can simply query this conditional table in the style of [20] and get the desired certain answers.

For disjunctions in the view definition, we use tuple local conditions as done in the example (with $w = 1$ and $w \neq 1$). Finally for CWA, this is done by evaluating the view definition $\mathcal{V}$ on the conditional table corresponding to OWA and adding as a constraint that each tuple it generates is indeed in the view instance I. In the example, the conditions simplify dramatically, but in general, this may result in rather gory tables. Note that, more generally, one could similarly introduce any *total dependency* [17, 39] on the database by *chasing* [4, 3] the conditional table as in [20, 18]. Observe also that from a practical viewpoint, this raises the issue of obtaining practical restrictions that prevent the conditions from becoming too complicated.

4 Open world assumption

Figure 1 gives an overview of the complexity of computing certain answers under OWA. Under OWA, the problem of computing certain answers is closely related to the query containment problem. Therefore, decidability and undecidability results carry over in both directions. As shown in Theorem 4.1, if the problems are decidable, then their *query complexity* is the same.

Theorem 4.1 *Let $\mathcal{L}_1, \mathcal{L}_2 \in \{CQ, CQ^{\neq}, PQ, datalog, FO\}$ be a view definition language and query language respectively. Then the problem of computing certain answers under OWA of a query $\mathcal{Q} \in \mathcal{L}_2$ given a view definition $\mathcal{V} \subseteq \mathcal{L}_1$ and a view instance is decidable if and only if the containment problem of a query in $\mathcal{L}_1$ in a query in $\mathcal{L}_2$ is decidable. Moreover, if the problems are decidable then the combined complexity of the view problem and the query complexity of the containment problem are identical, so the data complexity of the problem of computing certain answers under OWA is at most the query complexity of the query containment problem.*

Proof. The claim is established by giving reductions between the two problems in both directions. We first consider the reduction from the problem of computing certain answers under OWA to the query containment problem. Let $\mathcal{V} = \{v_1, \ldots, v_k\} \subseteq \mathcal{L}_1$ be a view definition, $\mathcal{Q} \in \mathcal{L}_2$ a query, I a view instance, and t a tuple of the same arity as the head of $\mathcal{Q}$. Let $\mathcal{Q}'$ be a query consisting of the rules of definition $\mathcal{V}$ together with the rule[1]

$$q'(t) :- v_1(t_{11}),\ \ldots,\ v_1(t_{1n_1}), \ldots, v_k(t_{k1}),\ \ldots,\ v_k(t_{kn_k})$$

where I is the instance with $I(v_1) = \{t_{11}, \ldots, t_{1n_1}\}$, ..., $I(v_k) = \{t_{k1}, \ldots, t_{kn_k}\}$. If $\mathcal{L}_1$ is CQ or $CQ^{\neq}$, then the view definitions in $\mathcal{V}$ can be substituted in for the view literals in this new rule. This yields just one conjunctive query. In all cases, $\mathcal{Q}'$ is in $\mathcal{L}_1$. We show that tuple t is a certain answer of $\mathcal{Q}$ given $\mathcal{V}$ and I if and only if $\mathcal{Q}'$ is contained in $\mathcal{Q}$.

"$\Rightarrow$": Assume that t is a certain answer under OWA. Let D be a database. If $I \not\subseteq \mathcal{V}(D)$, then $\mathcal{Q}'(D) = \{\}$, and therefore $\mathcal{Q}'(D)$ is trivially contained in $\mathcal{Q}(D)$. If $I \subseteq \mathcal{V}(D)$, then $\mathcal{Q}'(D) = \{t\}$ and $t \in \mathcal{Q}(D)$. Again, $\mathcal{Q}'(D)$ is contained in $\mathcal{Q}(D)$.

"$\Leftarrow$": Assume that $\mathcal{Q}'$ is contained in $\mathcal{Q}$. Let D be a database with $I \subseteq \mathcal{V}(D)$. Then $\mathcal{Q}'(D) = \{t\}$, and therefore $t \in \mathcal{Q}(D)$. Hence, t is a certain answer.

The remaining of the proof consists of a reduction from the query containment problem to the problem of computing certain answers under OWA. Let $\mathcal{Q}_1 \in \mathcal{L}_1$ and $\mathcal{Q}_2 \in \mathcal{L}_2$ be two queries. Let p be a new predicate, and let q_1 and q_2 be the answer predicates of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ respectively. Consider as view definition the rules of $\mathcal{Q}_1$ together with the additional rule

$$v(c) :- q_1(X),\ p(X)$$

and the instance $I = \{v(c)\}$. Let the query $\mathcal{Q}$ be defined by all the rules of $\mathcal{Q}_2$ together with the following rule:

$$q(c) :- q_2(X),\ p(X).$$

Again, if $\mathcal{L}_1$ or $\mathcal{L}_2$ are CQ or $CQ^{\neq}$, then the definition of $\mathcal{V}$ and query $\mathcal{Q}$ respectively can be transformed into a conjunctive query. Therefore, $\mathcal{V} \subseteq \mathcal{L}_1$ and $\mathcal{Q} \in \mathcal{L}_2$. We show that $\mathcal{Q}_1$ is contained in $\mathcal{Q}_2$ if and only if $\langle c \rangle$ is a certain answer of $\mathcal{Q}$ given $\mathcal{V}$ and I.

"$\Rightarrow$": Suppose that $\langle c \rangle$ is not a certain answer. Then there exists a database D with $I \subseteq \mathcal{V}(D)$ and $\mathcal{Q}(D)$ does not contain $\langle c \rangle$. It follows that $\mathcal{Q}_1(D)$ contains a tuple that $\mathcal{Q}_2(D)$ does not contain. Therefore, $\mathcal{Q}_1$ is not contained in $\mathcal{Q}_2$.

"$\Leftarrow$": Assume that $\mathcal{Q}_1$ is not contained in $\mathcal{Q}_2$. Then there exists a database D such that $\mathcal{Q}_1(D)$ contains a tuple t that is not contained in $\mathcal{Q}_2(D)$. Database D can be assumed to have $p(D) = \{t\}$. Then $\mathcal{V}(D) = I$ and $\mathcal{Q}(D) = \{\}$. Therefore, $\langle c \rangle$ is not a certain answer. □

The previous theorem involves query complexity. However, we are primarily concerned by data complexity, and query complexity results can be misleading. For example, the query complexity of the containment problem of a conjunctive query in a datalog query is EXPTIME-complete, whereas the containment problem of a conjunctive query in a conjunctive query with inequality is considerably easier, namely Π_2^p-complete [42]. In comparison, the data complexity of computing certain answers under OWA for conjunctive view definitions and datalog queries is polynomial, whereas it is presumably harder, namely co-NP-complete, for conjunctive view definitions and conjunctive queries with inequality.

[1] In the case of FO, we use the first-order formula corresponding to this rule.

4.1 Conjunctive view definitions

In this section we consider the complexity of the problem of computing certain answers under OWA in the case of conjunctive view definitions. We will consider queries of different expressive power.

4.1.1 Polynomial cases

The main tool for proving polynomial time bounds is the notion of maximally-contained query plans. We recall the relevant definitions here.

The input of a datalog query $\mathcal{Q}$ consists of a database D storing instances of all EDB predicates in $\mathcal{Q}$. Given such a database D, the output of $\mathcal{Q}$, denoted $\mathcal{Q}(D)$, is an instance of the answer predicate q as determined by, for example, naive evaluation [35]. A datalog query $\mathcal{Q}'$ is *contained* in a datalog query $\mathcal{Q}$ if, for all databases D, $\mathcal{Q}'(D)$ is contained in $\mathcal{Q}(D)$.

A datalog query $\mathcal{P}$ is a *query plan* if all EDB predicates in $\mathcal{P}$ are view literals. The *expansion* $\mathcal{P}^{exp}$ of a datalog query plan $\mathcal{P}$ is obtained from $\mathcal{P}$ by replacing all view literals with their definitions. Existentially quantified variables in view definitions are replaced by new variables in the expansion. A query plan $\mathcal{P}$ is *maximally-contained* in a datalog query $\mathcal{Q}$ w.r.t. a view definition $\mathcal{V}$ if $\mathcal{P}^{exp} \subseteq \mathcal{Q}$, and for each query plan $\mathcal{P}'$ with $(\mathcal{P}')^{exp} \subseteq \mathcal{Q}$, it is the case that $\mathcal{P}'$ is also contained in $\mathcal{P}$. Intuitively, a maximally-contained query plan is the best of all datalog query plans in using the information available from the view instances. As shown in [10], it is easy to construct these maximally-contained query plans in the case of conjunctive view definitions.

Theorem 4.2 shows that maximally-contained query plans compute exactly the certain answers under OWA.

Theorem 4.2 *For $\mathcal{V} \subseteq CQ$, $\mathcal{Q} \in datalog$, and query plan $\mathcal{P}$ that is maximally-contained in $\mathcal{Q}$ with respect to $\mathcal{V}$, $\mathcal{P}$ computes exactly the certain answers of $\mathcal{Q}$ under OWA for each view instance I.*

Proof. Assume for the sake of contradiction that there is an instance I of the view such that $\mathcal{P}$ fails to compute a certain answer t of $\mathcal{Q}$ under OWA. Let $\mathcal{P}'$ be the query plan that consists of all the rules of $\mathcal{P}$, together with two additional rules r_1 and r_2:

$$\begin{aligned} r_1: &\quad q'(X) :- q(X) \\ r_2: &\quad q'(t) \ :- \ v_1(t_{11}),\ \ldots,\ v_1(t_{1n_1}),\ \ldots, \\ &\qquad\qquad v_k(t_{k1}),\ \ldots,\ v_k(t_{kn_k}) \end{aligned}$$

where q is the answer predicate of $\mathcal{P}$, and I is the instance with $I(v_1) = \{t_{11}, \ldots, t_{1n_1}\}$, ..., $I(v_k) = \{t_{k1}, \ldots, t_{kn_k}\}$. We are going to show that $(\mathcal{P}')^{exp}$ is contained in $\mathcal{Q}$. Since $\mathcal{P}'$ is not contained in $\mathcal{P}$, this contradicts the maximal containment of $\mathcal{P}$ in $\mathcal{Q}$. Therefore, there cannot be a certain answer t under OWA that $\mathcal{P}$ fails to compute.

In order to see that $(\mathcal{P}')^{exp}$ is contained in $\mathcal{Q}$, we have to show that $\mathcal{P}'(\mathcal{V}(D))$ is contained in $\mathcal{Q}(D)$ for each database D. Let D be an arbitrary database. Because $\mathcal{P}^{exp}$ is known to be contained in $\mathcal{Q}$, it suffices to show that $r_2(\mathcal{V}(D))$ is contained in $\mathcal{Q}(D)$. If I is not contained in $\mathcal{V}(D)$, then $r_2(\mathcal{V}(D))$ is the empty set, which is trivially contained in $\mathcal{Q}(D)$. So let us assume that I is contained in $\mathcal{V}(D)$. Then $r_2(\mathcal{V}(D)) = \{t\}$. Because t is a certain answer under OWA, it follows by definition that t is an element of $\mathcal{Q}(D)$. Therefore, $r_2(\mathcal{V}(D))$ is contained in $\mathcal{Q}(D)$. □

As shown in [10] for all $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in datalog$, corresponding maximally-contained datalog query plans can be constructed. Because the data complexity of evaluating datalog queries is polynomial [37], this implies that the problem of computing certain answers under OWA can be done in polynomial time.

Corollary 4.1 *For $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in datalog$, the problem of computing certain answers under OWA can be done in polynomial time.*

4.1.2 Inequalities in the view definition

We next show (Theorem 4.3) that adding inequalities just to the view definition doesn't add any expressive power. The certain answers are exactly the same as if the inequalities in the view definition were omitted. This means that the maximally-contained datalog query constructed from the query and the view definition but disregarding the inequality constraints computes exactly the certain answers. Therefore, the problem remains polynomial.

Theorem 4.3 *Let $\mathcal{V} \subseteq CQ^{\neq}$ and $\mathcal{Q} \in datalog$. Define $\mathcal{V}^-$ to be the same view definition as $\mathcal{V}$ but with the inequality constraints deleted. Then a tuple t is a certain answer under the open world assumption given $\mathcal{V}$, $\mathcal{Q}$ and a view instance I if and only if t is a certain answer under OWA given $\mathcal{V}^-$, $\mathcal{Q}$ and I.*

Proof. "⇒": Assume that t is a certain answer under OWA given $\mathcal{V}$, $\mathcal{Q}$ and I. Let D be a database with $I \subseteq \mathcal{V}^-(D)$. If also $I \subseteq \mathcal{V}(D)$, then it follows immediately that t is in $\mathcal{Q}(D)$. Otherwise, there is a view definition v in $\mathcal{V}$ and a tuple $s \in I$ such that $s \in v^-(D)$, but $s \notin v(D)$. Let $C \neq C'$ be an inequality constraint in v that disabled the derivation of s in $v(D)$. Because we can assume that s is in $v(D')$ for some database D', at least one of C or C' must be an existentially quantified variable X. Add tuples to D that correspond to the tuples that generate s in $v^-(D)$, but with the constant that X binds to replaced by a new constant. These new tuples then satisfy the inequality constraint $C \neq C'$. By repeating this process for each such inequality constraint $C \neq C'$ and each such tuple s, we arrive at a database D'' with $I \subseteq \mathcal{V}(D'')$. Because t is a certain answer given $\mathcal{V}$, it follows that t is in $\mathcal{Q}(D'')$. Therefore, there are tuples $t_1, \ldots, t_k \in D''$ that derive t. If any t_i contains one of the new constants, replace it by the tuple $t_i' \in D$ that it was originally derived from. Because t doesn't contain any new constants, and because $\mathcal{Q}$ cannot test for inequality, it follows that t is also derived from $t_1', \ldots, t_k'$. Hence t is in $\mathcal{Q}(D)$.

"⇐": Assume that t is a certain answer under OWA given $\mathcal{V}^-$, $\mathcal{Q}$ and I. Let D be a database with $I \subseteq \mathcal{V}(D)$. Because $\mathcal{V}$ is contained in $\mathcal{V}^-$, it follows that $I \subseteq \mathcal{V}^-(D)$, and therefore t is in $\mathcal{Q}(D)$. □

4.1.3 Inequalities in the query

On the other hand, we see next (Theorem 4.4) that adding inequalities to queries does add expressive power. A single inequality in a conjunctive query, even combined with purely conjunctive view definitions, suffices to make the problem co-NP-hard. Van der Meyden proved a similar result [40], namely co-NP hardness for the case $\mathcal{V} \subseteq CQ^<$ and $\mathcal{Q} \in CQ^<$. Our theorem strengthens this result to $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in CQ^{\neq}$.

Theorem 4.4 *For $\mathcal{V} \subseteq CQ$, $\mathcal{Q} \in CQ^{\neq}$, the problem of determining whether, given a view instance, a tuple is a certain answer under OWA is co-NP-hard.*

Proof. Let φ be a CNF formula with variables $x_1, \ldots, x_n$ and conjuncts $c_1, \ldots, c_m$. Consider the conjunctive view definition and view instance:

$$\begin{array}{lll} v_1(X,Y,Z) & :- & p(X,Y,Z) \\ v_2(X) & :- & r(X,Y) \\ v_3(Y) & :- & p(X,Y,Z),\ r(X,Z) \end{array}$$

$$\begin{array}{rl} I(v_1) = & \{\langle i,j,1\rangle \mid x_i \text{ occurs in } c_j\} \\ & \cup\, \{\langle i,j,0\rangle \mid \bar{x}_i \text{ occurs in } c_j\} \\ I(v_2) = & \{\langle 1\rangle, \ldots, \langle n\rangle\} \\ I(v_3) = & \{\langle 1\rangle, \ldots, \langle m\rangle\} \end{array}$$

and the query: $q(c) :- r(X,Y),\ r(X,Y'),\ Y \neq Y'$.

We can show that tuple $\langle c\rangle$ is a certain answer under OWA if and only if formula φ is *not* satisfiable. Because the problem of testing a CNF formula for satisfiability is NP-complete [8], this implies the claim.

"⇒": Assume that φ is satisfiable. Then there is an assignment ν from $x_1, \ldots, x_n$ to $true$ and $false$ such that each conjunct of φ contains at least one variable x_i with $\nu(x_i) = true$ or one negated variable $\bar{x}_i$ with $\nu(x_i) = false$. Consider the database D with

$$\begin{array}{l} p(D) = \{\langle i,j,1\rangle \mid x_i \text{ occurs in } c_j\} \\ \qquad\quad \cup\, \{\langle i,j,0\rangle \mid \bar{x}_i \text{ occurs in } c_j\} \\ r(D) = \{\langle i,d_i\rangle \mid i \in \{1,\ldots,n\}, \\ \qquad\qquad d_i = \left\{ \begin{array}{lcl} 1 & : & \nu(x_i) = true \\ 0 & : & \nu(x_i) = false \end{array} \right. \} \end{array}$$

Instance I is contained in $\mathcal{V}(D)$, and $\mathcal{Q}(D)$ does not derive $\langle c\rangle$. Therefore, $\langle c\rangle$ is not a certain answer.

"⇐": Assume that $\langle c\rangle$ is not a certain answer. Then there exists a database D with $I \subseteq \mathcal{V}(D)$ such that $\mathcal{Q}(D)$ is the empty set. This means that for $i = 1, \ldots, n$, database D contains exactly one r tuple $\langle i, d_i\rangle$. Consider the assignment ν with $\nu(x_i) = true$ if D contains the r tuple $\langle i, 1\rangle$, and with $\nu(x_i) = false$ otherwise. Let c_j be one of the conjuncts. Because $\langle j\rangle$ is contained in $I(v_3)$, there must be a p tuple $\langle i, j, d_i\rangle$ and an r tuple $\langle i, d_i\rangle$. If $d_i = 1$, then c_j contains a variable x_i with $\nu(x_i) = true$. If $d_i = 0$, then c_j contains a negated variable $\bar{x}_i$ with $\nu(x_i) = false$. Since ν satisfies each c_j, φ is satisfiable. □

By Theorem 4.2, we know that maximally-contained queries compute exactly the certain answers under OWA. Because evaluating datalog queries has polynomial data complexity [37], it follows that in general there are *no* datalog queries that are maximally-contained in a conjunctive query with inequality.

4.1.4 First-order queries

We saw that even adding recursion to positive queries leaves the data complexity of the problem of computing certain answers under OWA still polynomial. On the other hand, adding negation makes the problem undecidable for both OWA and CWA, as the following theorem shows.

Theorem 4.5 *For $\mathcal{V} \subseteq CQ$, $\mathcal{Q} \in FO$, the problem of determining, given a view definition together with a view instance, whether a tuple is a certain answer under the open or closed world assumption is undecidable.*

Proof. Let φ be a first-order formula. Consider the query

$$q(c) :- \neg\varphi.$$

Clearly, $\langle c\rangle$ is a certain answer if and only if φ is not satisfiable. Testing whether a first-order formula admits a finite model is undecidable (see [15]). This implies the claim. □

4.2 Positive view definitions

In the previous section, we proved that adding inequalities to the query results in co-NP-completeness of the problem of computing certain answers under OWA. The following theorem shows that allowing disjunction in the view definition has the same effect on the data complexity. The same result was proved by van der Meyden in [41] while studying indefinite databases. We include the theorem for the sake of completeness.

Theorem 4.6 *[41] For $\mathcal{V} \subseteq PQ$, $\mathcal{Q} \in CQ$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA is co-NP-hard.*

4.3 Datalog view definitions

Theorem 3.1 established that the problem can be solved in co-NP for $\mathcal{V} \subseteq PQ^{\neq}$ and $\mathcal{Q} \in datalog^{\neq}$. Here we examine the effect on the complexity of the problem of computing certain answers if we allow datalog as view definition language. For positive queries, the problem stays in co-NP as was shown by van der Meyden in [41]. However, Theorem 4.7 and Corollary 4.2 respectively establish that the problem becomes undecidable for conjunctive queries with inequality and datalog queries.

4.3.1 Inequalities

In the case of conjunctive view definitions, adding inequalities to the query increased the complexity of the problem of computing certain answers under OWA from polynomial to co-NP. With datalog view definitions, adding inequalities to the query raises the problem from co-NP complexity to undecidability. In [40], van der Meyden showed undecidability for the case of $\mathcal{V} \subseteq datalog$ and $\mathcal{Q} \in PQ^{\neq}$. The following theorem proves that the problem is already undecidable for *conjunctive* queries with inequality.

Theorem 4.7 *For $\mathcal{V} \subseteq datalog$, $\mathcal{Q} \in CQ^{\neq}$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA is undecidable.*

Proof. The proof is by reduction of the Post Correspondence Problem [29] to the problem in the claim.

Let $w_1, \ldots, w_n, w'_1, \ldots, w'_n$ be words over alphabet $\{a, b\}$. Consider the following datalog query that defines view v:

$$\begin{aligned}
v(0,0) \quad &:- s(e,e,e)\\
v(X,Y) &:- v(X_0,Y_0),\\
&\qquad s(X_0,X_1,\alpha_1),\ldots,s(X_{k-1},X,\alpha_k),\\
&\qquad s(Y_0,Y_1,\beta_1),\ldots,s(Y_{l-1},Y,\beta_l)\\
&\qquad \text{where } w_i = \alpha_1\ldots\alpha_k \text{ and } w'_i = \beta_1\ldots\beta_l;\\
&\qquad \text{one rule for each } i \in \{1,\ldots,n\}.\\
s(X,Y,Z) &:- p(X,X,Y),\ p(X,Y,Z)
\end{aligned}$$

and query $\mathcal{Q}$ defined by:

$$q(c) :- p(X,Y,Z),\ p(X,Y,Z'),\ Z \neq Z'$$

Let the view instance I be defined by $I(v) = \{\langle e,e\rangle\}$ and $I(s) = \{\}$. We will show that there exists a solution to the instance of the Post Correspondence Problem given by $w_1, \ldots, w_n, w'_1, \ldots, w'_n$ if and only if $\langle c\rangle$ is *not* a certain answer under OWA. The result then follows from the undecidability of the Post Correspondence Problem [29].

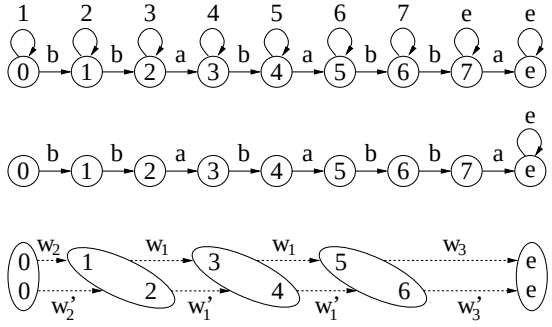

Figure 2: *The instance of the Post Correspondence Problem given by the words $w_1 = ba$, $w_2 = b$, $w_3 = bba$, $w'_1 = ab$, $w'_2 = bb$, and $w'_3 = ba$ has solution "2113" because $w_2w_1w_1w_3 = bbababba = w'_2w'_1w'_1w'_3$. The figure shows a database D with $\langle e,e\rangle \in v(D)$, but $\mathcal{Q}(D) = \{\}$.*

"⇒": Assume that the instance of the Post Correspondence Problem given by the words $w_1, \ldots, w_n, w'_1, \ldots, w'_n$ has a solution i_1, ..., i_k. Then $w_{i_1}\ldots w_{i_k} = w'_{i_1}\ldots w'_{i_k} = \gamma_1\ldots\gamma_m$ for some characters $\gamma_1, \ldots, \gamma_m \in \{a, b\}$. Consider the database D with

$$\begin{aligned}
p(D) = \{&\langle 0,1,\gamma_1\rangle,\ldots,\langle m-2,m-1,\gamma_{m-1}\rangle,\\
&\langle m-1,e,\gamma_m\rangle,\\
&\langle 0,0,1\rangle,\ldots,\langle m-2,m-2,m-1\rangle,\\
&\langle m-1,m-1,e\rangle,\langle e,e,e\rangle\}.
\end{aligned}$$

Clearly, $\mathcal{Q}(D) = \{\}$. Moreover, it is easy to verify that $s(D)$ and $v(D)$ are as follows:

$$s(D) = \{\langle 0, 1, \gamma_1\rangle, \ldots, \langle m-2, m-1, \gamma_{m-1}\rangle, \langle m-1, e, \gamma_m\rangle, \langle e, e, e\rangle\}$$

$$v(D) = \{\langle 0, 0\rangle, \langle |w_{i_1}|, |w'_{i_1}|\rangle, \langle |w_{i_1}| + |w_{i_2}|, |w'_{i_1}| + |w'_{i_2}|\rangle, \ldots, \langle |w_{i_1}| + \ldots + |w_{i_{k-1}}|, |w'_{i_1}| + \ldots + |w'_{i_{k-1}}|\rangle, \langle e, e\rangle\}$$

Since $I \subseteq v(D)$ and $\mathcal{Q}(D) = \{\}$, it follows that $\langle c\rangle$ is not a certain answer.

"$\Leftarrow$": Assume that $\langle c\rangle$ is not a certain answer under OWA. Then there is a database D with $I \subseteq v(D)$ such that $\mathcal{Q}(D) = \{\}$. Because tuple $\langle e, e\rangle$ is in $v(D)$, there must be constants $c_0, c_1, \ldots, c_m$ with $c_0 = 0$ and $c_m = e$ and characters $\gamma_1, \ldots, \gamma_m \in \{a, b\}$ such that

$$\langle c_0, c_1, \gamma_1\rangle,\ \langle c_1, c_2, \gamma_2\rangle, \ldots, \langle c_{m-1}, c_m, \gamma_m\rangle \in s(D). \quad (*)$$

Let $d_0, d_1, \ldots, d_{m'}$ be constants with $d_0 = 0$ and $\delta_1, \ldots, \delta_{m'} \in \{a, b\}$ be characters such that

$$\langle d_0, d_1, \delta_1\rangle,\ \langle d_1, d_2, \delta_2\rangle, \ldots, \langle d_{m'-1}, d_{m'}, \delta_{m'}\rangle \in s(D).$$

We are going to show by induction on m' that for $m' \leq m$, $d_i = c_i$ and $\delta_i = \gamma_i$ for $i = 0, \ldots, m'$. The claim is trivially true for $m' = 0$. For the induction case, let $m' > 0$. We know that $\langle c_{i-1}, c_i, \gamma_i\rangle \in s(D)$ and $\langle d_{i-1}, d_i, \delta_i\rangle \in s(D)$, and that $c_{i-1} = d_{i-1}$. By definition of s, this implies that tuples $\langle c_{i-1}, c_{i-1}, c_i\rangle$, $\langle c_{i-1}, c_{i-1}, d_i\rangle$, $\langle c_{i-1}, c_i, \gamma_i\rangle$, and $\langle c_{i-1}, d_i, \delta_i\rangle$ are all in $p(D)$. Because $\mathcal{Q}(D) = \{\}$, it follows that $d_i = c_i$ and $\delta_i = \gamma_i$.

Assume for the sake of contradiction that $m' > m$. Then there is a tuple $\langle d_m, d_{m+1}, \gamma_{m+1}\rangle \in s(D)$, and therefore $\langle d_m, d_m, d_{m+1}\rangle$, $\langle d_m, d_{m+1}, \gamma_{m+1}\rangle \in p(D)$. Because $\langle e, e, e\rangle \in s(D)$, it follows that $\langle e, e, e\rangle \in p(D)$. Since $d_m = c_m = e$ this implies that $d_{m+1} = e$ and $\gamma_{m+1} = e$, which contradicts the fact that $\gamma_{m+1} \in \{a, b\}$. Hence, $m' = m$.

We proved that there is exactly one chain of the form in (*). Because $\langle e, e\rangle \in v(D)$, there is a sequence $i_1 \ldots i_k$ with $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that $w_{i_1} \ldots w_{i_k} = \gamma_1 \ldots \gamma_m$ and $w'_{i_1} \ldots w'_{i_k} = \gamma_1 \ldots \gamma_m$. Therefore, $i_1, \ldots, i_k$ is a solution to the instance of the Post Correspondence Problem given by $w_1, \ldots, w_n, w'_1, \ldots, w'_n$. □

Theorem 4.7 has an interesting consequence for the query containment problem of a recursive datalog query in a nonrecursive datalog query with inequality. It shows that the technique in [9] to prove decidability of a datalog query in a nonrecursive datalog query does not carry to datalog with inequality. Indeed, it is an easy corollary of Theorems 4.1 and 4.7 that the problem is undecidable.

4.3.2 Datalog queries

As we saw, there is a close relationship between the problem of computing certain answers under OWA and query containment. Not surprisingly it is therefore the case that the problem becomes undecidable for datalog view definitions and datalog queries.

Corollary 4.2 *For $\mathcal{V} \subseteq datalog$, $\mathcal{Q} \in datalog$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA is undecidable.*

Proof. The containment problem of a datalog query in another datalog query is undecidable [32]. Therefore, the claim follows directly from Theorem 4.1. □

4.4 First-order view definitions

Theorem 4.5 showed that adding negation in queries leads to undecidability. The following theorem now shows that the same is true for adding negation to view definitions.

Theorem 4.8 *For $\mathcal{V} \in FO$, $\mathcal{Q} \in CQ$, the problem of determining, given a view instance, whether a tuple is a certain answer under the open or the closed world assumption is undecidable.*

Proof. Let φ be a first-order formula, and p a new predicate. Consider the view definition

$$v(c) :- \varphi(X) \vee p(X)$$

together with the instance $I = \{v(c)\}$ and the query $\mathcal{Q}$ defined by:

$$q(c) :- p(X)$$

We will show that $\langle c\rangle$ is a certain answer under the open or closed world assumption if and only if formula φ is not satisfiable. By Trahtenbrot's theorem, testing whether a first-order formula admits a finite model is undecidable (see [15]). This implies the claim.

"$\Rightarrow$": Suppose that φ is satisfiable. Then there exists a database D such that $\varphi(D)$ is satisfied, and such that $p(D)$ is empty. For this database, $I = v(D)$ and $\mathcal{Q}(D) = \{\}$. Therefore, $\langle c\rangle$ is not a certain answer.

"$\Leftarrow$": Suppose that $\langle c\rangle$ is not certain. Then there is a database D with $I \subseteq \mathcal{V}(D)$ (or with $I = \mathcal{V}(D)$) such that $\langle c\rangle$ is not in $\mathcal{Q}(D)$. Since $p(D)$ is empty, $\varphi(D)$ must be satisfied. Therefore, formula φ is satisfiable. □

5 Closed world assumption

Figure 3 gives an overview of the complexity of the problem of computing certain answers under CWA. Computing certain answers under CWA is harder than under OWA. Whereas the problem is polynomial for $\mathcal{V} \subseteq CQ^{\neq}$ and $\mathcal{Q} \in datalog$ under OWA, the problem is already co-NP-complete for $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in CQ$ under CWA. Moreover, whereas the problem is decidable for $\mathcal{V} \subseteq datalog$ and $\mathcal{Q} \in PQ$ under OWA, the problem is already undecidable for $\mathcal{V} \subseteq datalog$ and $\mathcal{Q} \in CQ$ under CWA.

views	— query — CQ	$CQ^{\neq}$	PQ	$datalog$	FO
CQ	co-NP	co-NP	co-NP	co-NP	undec.
$CQ^{\neq}$	co-NP	co-NP	co-NP	co-NP	undec.
PQ	co-NP	co-NP	co-NP	co-NP	undec.
$datalog$	undec.	undec.	undec.	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

Figure 3: *Data complexity of the problem of computing certain answers under the closed world assumption.*

5.1 Conjunctive view definitions

The following theorem shows that computing certain answers under the closed world assumption is already co-NP-hard in the very simplest case, namely in the case of conjunctive view definitions and conjunctive queries.

Theorem 5.1 *For $\mathcal{V} \subseteq CQ$, $\mathcal{Q} \in CQ$, the problem of determining, given a view instance, whether a tuple is a certain answer under CWA is co-NP-hard.*

Proof. Let $G = (V, E)$ be an arbitrary graph. Consider the view definition:

$$\begin{aligned} v_1(X) &:- color(X, Y) \\ v_2(Y) &:- color(X, Y) \\ v_3(X, Y) &:- edge(X, Y) \end{aligned}$$

and the instance I with $I(v_1) = V$, $I(v_2) = \{red, green, blue\}$ and $I(v_3) = E$. We will show that under CWA the query

$$q(c) :- edge(X, Y),\ color(X, Z),\ color(Y, Z)$$

has the tuple $\langle c \rangle$ as a certain answer if and only if graph G is not 3-colorable. Because testing a graph's 3-colorability is NP-complete [22], this implies the claim.

For each database D with $I = \mathcal{V}(D)$, relation $edge$ contains exactly the edges from E, and relation $color$ relates all vertices in V to either red, $green$, or $blue$.

"$\Rightarrow$": Assume that $\langle c \rangle$ is a certain answer of the query. It follows that for each assignment of the vertices to red, $green$, and $blue$, there is an edge $\langle e_1, e_2 \rangle$ in E such that e_1 and e_2 are assigned to the same color. Therefore, there is not a single assignment of vertices to the three colors red, $green$, and $blue$ such that all adjacent vertices are assigned to different colors. Hence G is not 3-colorable.

"$\Leftarrow$": Assume G is not 3-colorable. Then for each assignment of vertices in V to red, $green$, and $blue$ there exists at least one edge $\langle e_1, e_2 \rangle$ such that e_1 and e_2 are assigned to the same color. It follows that the query will produce $\langle c \rangle$ for each database D with $I = \mathcal{V}(D)$, i.e. the query has $\langle c \rangle$ as a certain answer. □

5.2 Datalog view definitions

The final theorem in this section shows that for datalog view definitions, the problem is undecidable under CWA.

Theorem 5.2 *For $\mathcal{V} \subseteq datalog$, $\mathcal{Q} \in CQ$ the problem of determining, given a view instance, whether a tuple is a certain answer under CWA is undecidable.*

Proof. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two datalog queries with answer predicate q_1 and q_2 respectively. Consider the view definition consisting of the rules of $\mathcal{Q}_1$ and $\mathcal{Q}_2$, and the rules

$$\begin{aligned} v_1(c) &:- r(X) \\ v_1(c) &:- q_1(X),\ p(X) \\ v_2(c) &:- q_2(X),\ p(X) \end{aligned}$$

where p and r are two relations not appearing in $\mathcal{Q}_1$ and $\mathcal{Q}_2$. Consider the instance I with $I(v_1) = \{\langle c \rangle\}$ and $I(v_2) = \{\}$, and the query $\mathcal{Q}$ defined by:

$$q(c) :- r(X)$$

If $\mathcal{Q}_1 \subseteq \mathcal{Q}_2$, then for each database D with $\mathcal{V}(D) = I$,

$$q_1(D) \cap p(D) \subseteq q_2(D) \cap p(D) = I(v_2) = \{\}.$$

Therefore,

$$r(D) = I(v_1) = \{\langle c \rangle\},$$

i.e. $\langle c \rangle$ is a certain answer under CWA.

On the other hand, if $\mathcal{Q}_1 \not\subseteq \mathcal{Q}_2$, then there is a database D such that some tuple t is in $\mathcal{Q}_1(D)$, but not in $\mathcal{Q}_2(D)$. By extending D such that $p(D) = \{t\}$ and $r(D) = \{\}$, we have that $\mathcal{V}(D) = I$. Because $q(D) = \{\}$, $\langle c \rangle$ is not a certain answer under CWA.

We established that $\langle c \rangle$ is a certain answer under CWA if and only if $\mathcal{Q}_1$ is contained in $\mathcal{Q}_2$. The claim now follows from the undecidability of containment of datalog queries [32]. □

6 View consistency and view self-maintainability

In this section, we consider two other important problems on materialized views, view consistency and view self-maintainability. We do it in the context of CWA since both of these problems make more sense in that context than under OWA.

Definition 6.1 (**view consistency**) Let $\mathcal{V}$ be a view definition and I an instance of the view. Then the view is *consistent* if there is a database D such that $I = \mathcal{V}(D)$. □

Definition 6.2 (**view self-maintainability**) Let D be a database. An *update* to D is either a deletion $d(t)$ of a tuple t in D, or an insertion $i(t)$ of some tuple t not in D. Let $\mathcal{V}$ be a view definition and I a consistent view instance. Then the view is *self-maintainable* for an update α if there exists a view instance J such that for each D with $I = \mathcal{V}(D)$, $J = \mathcal{V}(\alpha(D))$. □

views	consistency	self-maintainability
CQ	NP	co-NP
$CQ^{\neq}$	NP	co-NP
PQ	NP	co-NP
$datalog$	undec.	undec.
FO	undec.	undec.

Figure 4: *Data complexity of the view consistency and the view self-maintainability problem.*

The complexity of these problems is shown in Figure 4. The complexity table for self-maintainability is the same as the one for the problem of computing certain answers under CWA in Figure 3 for conjunctive queries. The complexity of the view consistency problem is similar with NP in place of co-NP. Note that the undecidable cases for the view consistency problem are r.e., whereas for computing certain answers and self-maintainability, they are co-r.e.

Theorem 6.1

(i) For $\mathcal{V} \subseteq PQ^{\neq}$, the view consistency problem is in NP*, and the view self-maintainability problem is in co-*NP *(w.r.t. the size of the view).*

(ii) For $\mathcal{V} \subseteq CQ$, the view consistency problem is NP*-hard, and the view self-maintainability problem is co-*NP*-hard (w.r.t. the size of the view).*

(iii) For $\mathcal{V} \subseteq datalog$ or $\mathcal{V} \subseteq FO$, the view consistency problem is undecidable (r.e.), and the view self-maintainability problem as well (co-r.e.).

Due to space limitations, the proof of this result is omitted. It basically involves some simple reductions of these problems from/to the problem of answering queries using materialized views under the closed world assumption.

7 Conclusion

We presented some complexity results with respect to materialized views. A main contribution is (i) the exhibition of deep connections with incomplete databases and (as a consequence) (ii) the point of view that a materialized view should be seen as an incomplete database. This indeed suggests using some model of incomplete information as the view model. We will illustrate briefly this direction with an example. Consider the self-maintainability problem of materialized views. Suppose we have such a view, the database is unavailable and we receive some updates to the database. A known technique is to verify whether the view is self-maintainable. If it is not, we raise our hands and in principle the view becomes unavailable. However, one could consider updating the incomplete database corresponding to the view. We could continue answering queries, and indeed, with such a model, it is possible to have more semantics in our answers, e.g. provide besides *certain* answers, *possible* answers, or indicate whether our answer is *surely* complete or not. We intend to continue the present work in that direction.

Acknowledgments

We would like to thank Moshe Y. Vardi and Alon Y. Levy for pointing us to the work of Ron van der Meyden on the complexity of querying indefinite databases, and Michael R. Genesereth, Pierre Huyn, Werner Nutt, Anand Rajaraman, and Yehoshua Sagiv for discussions on the topic.

References

[1] S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Abiteboul, P. Kanellakis, G. Grahne. On the representation and querying of sets of possible worlds. *TCS*, 78(1):159 – 187, 1991.

[3] A. V. Aho, Y. Sagiv, J. D. Ullman. Efficient optimiziation of a class of relational expressions. *ACM TODS*, 4(4):435–454, 1979.

[4] A. V. Aho, Y. Sagiv, J. D. Ullman. Equivalences among relational expressions. *SIAM J. on Computing*, 8(3):218–246, 1979.

[5] C. Beeri, A. Y. Levy, M.-C. Rousset. Rewriting queries using views in description logics. In *PODS-97*, pp. 99 – 108, 1997.

[6] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, K. Shim. Optimizing queries with materialized views. In *ICDE-95*, pp. 190–200, 1995.

[7] A. K. Chandra, P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *ACM STOC*, pp. 77–90, 1977.

[8] S. A. Cook. The complexity of theorem proving procedures. In *ACM STOCS*, pp. 151 – 158, 1971.

[9] S. Chaudhuri, M. Y. Vardi. On the equivalence of recursive and nonrecursive datalog programs. In *PODS-92*, pp. 55–66.

[10] O. M. Duschka, M. R. Genesereth. Answering recursive queries using views. In *PODS-97*, pp. 109 – 116.

[11] O. M. Duschka, M. R. Genesereth. Query planning in Infomaster. In *ACM SAC*, 1997.

[12] O. M. Duschka, A. Y. Levy. Recursive plans for information gathering. In *IJCAI-97*.

[13] G. Dong, J. Su. Conjunctive query containment with respect to views and constraints. *IPL*, 57(2):95–102, 1996.

[14] O. M. Duschka. Query optimization using local completeness. In *AAAI-97*, pp. 249–255.

[15] H.-D. Ebbinghaus, J. Flum. *Finite model theory*. Springer-Verlag, 1995.

[16] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, Inc., 1972.

[17] R. Fagin, M. Y. Vardi. The theory of data dependencies: A survey. In M. Anshel and W. Gewirtz, editors, *Mathematics of Information Processing: Proceedings of Symposia in Applied Mathematics*, vol. 34, pp. 19 – 71, 1986.

[18] G. Grahne. *Problem of incomplete information in relational databases*. Springer-Verlag, 1991.

[19] N. Huyn. Maintaining data warehouses under limited source access. Ph.D. Thesis STAN-CS-TR-97-1595, Stanford University, 1997.

[20] T. Imielinski, W. Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[21] D. S. Johnson, A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *JCSS*, 28:167–189, 1984.

[22] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pp. 85 – 104, 1972.

[23] A. Klug. On conjunctive queries containing inequalities. *J. ACM*, 35(1):146–160, 1988.

[24] A. Y. Levy, A. O. Mendelzon, D. Srivastava, Y. Sagiv. Answering queries using views. In *PODS-95*.

[25] A. Y. Levy, A. Rajaraman, J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *VLDB-96*, pp. 251–262.

[26] A. Y. Levy, A. Rajaraman, J. D. Ullman. Answering queries using limited external processors. In *PODS-96*.

[27] A. Y. Levy, D. Suciu. Deciding containment for queries with complex objects. In *PODS-97*, pp. 32 – 43.

[28] W. J. Labio, Y. Zhuge, J. L. Wiener, H. Gupta, H. Garcia-Molina, J. Widom. The WHIPS prototype for data warehouse creation and maintenance. In *SIGMOD Conf.*, pp. 557 – 559, 1997.

[29] E. Post. A variant of a recursively unsolvable problem. *Bulletin AMS*, 52:264–268, 1946.

[30] A. Rajaraman, Y. Sagiv, J. D. Ullman. Answering queries using templates with binding patterns. In *PODS-95*.

[31] O. Shmueli. Decidability and expressiveness aspects of logic queries. In *PODS-87*, pp. 237 – 249.

[32] O. Shmueli. Equivalence of datalog queries is undecidable. *J. Logic Programming*, 15:231–241, 1993.

[33] Y. Sagiv, M. Yannakakis. Equivalence among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.

[34] J. D. Ullman. *Principles of Database and Knowledge-base Systems*, vol. 1. Computer Science Press, 1988.

[35] J, D. Ullman. *Principles of Database and Knowledge-base Systems*, vol. 2. Computer Science Press, 1989.

[36] J. D. Ullman. Information integration using logical views. In *ICDT-97*.

[37] M. Y. Vardi. The complexity of relational query languages. In *STOC-82*, pp. 137 – 146.

[38] M. Y. Vardi. Querying logical databases *JCSS*, 33(2):142–160, 1986.

[39] M. Y. Vardi. Fundamentals of dependency theory. In E. Borger, editor, *Trends in Theoretical Computer Science*, pp. 171 – 224. Computer Science Press, 1987.

[40] R. van der Meyden. The complexity of querying indefinite information: Defined relations, recursion and linear order. Technical report, Rutgers University, 1992.

[41] R. van der Meyden. Recursively indefinite databases. *TCS*, 116(1):151–194, 1993.

[42] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *JCSS*, 54(1):113 – 135, 1997.

[43] H. Z. Yang, P.-Å. Larson. Query transformation for PSJ-queries. In *VLDB-87*, pp. 245–254.