January 2008

Conjunctive Queries and Mappings With Unequalities

Grigoris Karvounarakis
University of Pennsylvania

Val Tannen
University of Pennsylvania, val@cis.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-08-37

Conjunctive Queries and Mappings With Unequalities

Abstract
We study conjunctive queries with *unequalities* ($x \neq y$) and we identify cases when query containment can still be characterized by the existence of homomorphisms. We also identify a class of GLAV-like database schema mappings with unequalities, for which the chase theorem holds, and thus data exchange has the same complexity as for GLAV mappings. Finally, we define a notion of consistency and provide an algorithm to check whether a set of mappings is consistent.

Comments
University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-08-37

Conjunctive queries and mappings with unequalities

Grigoris Karvounarakis Val Tannen
Computer and Information Science Department
University of Pennsylvania
{gkarvoun,val}@cis.upenn.edu

Abstract

We study conjunctive queries with *unequalities* ($x \neq y$) and we identify cases when query containment can still be characterized by the existence of homomorphisms. We also identify a class of GLAV-like database schema mappings with unequalities, for which the chase theorem holds, and thus data exchange has the same complexity as for GLAV mappings. Finally, we define a notion of consistency and provide an algorithm to check whether a set of mappings is consistent.

1 Introduction

Containment of queries in general or under dependencies is an important problem that has been studied extensively in database research. Statements of conjunctive query containment are themselves related to very general database dependencies and to schema mappings (GLAV). The applications include *query optimization*, *data integration*, and *data exchange*.

The *chase* procedure gives an algorithm for testing containment of conjunctive queries under certain classes of depedencies [10, 11]. Moreover, for GLAV schema mappings the chase can also be used to solve the problem of data exchange [11].

The results we just mentioned assume conjunctions of *positive* atoms in the queries, dependencies and mappings and the corresponding algorithms have reasonable complexity (for fixed schemas). However, it is known that the introduction of negation or inequalities (i.e., $<, \leq$) makes these problems harder [7, 18].

In this paper, we explore some extensions of conjunctive queries, dependencies and GLAV mappings for which the containment test and the chase procedure can be performed without increased complexity. The negative atoms whose addition we study are *unequalities*, i.e., atoms of the form $t_1 \neq t_2$. In particular, we show that with certain limited use of unequalities in queries and mappings we can use same or slight extensions (without an increase in complexity) of the techniques used for conjunctive queries and GLAV mappings to answer such questions.

For dependencies/mappings this is the case of the class of unequality generating dependencies (*unegds*), which are dependencies whose body is the body of a conjunctive query and their head is a (conjunction of) unequalities. Using such dependencies, one can specify some interesting constraints; e.g. in a schema containing employees and managers of a company, one can use an unegd to ensure that no employee is also listed as a manager:

$$\forall x, y \; Employee(x) \wedge Manager(y) \rightarrow x \neq y$$

Moreover, in a *collaborative data sharing system* (CDSS) [14], which includes data exchange in a peer-to-peer setting it may be desirable to express the fact that the contents of two different peers do not overlap (e.g. such information may be useful for optimization):

$$\forall x, y \; P1(x) \wedge P2(y) \rightarrow x \neq y$$

In such settings, participants can independently add mappings between their schema and other peers, when they join. When the mappings involve unequalities (or just constants) this can lead to set of mappings that

cannot be satisfied (i.e., no data exchange solution exists, regardless of what data each peer contributes). We define the notion of consistency of a set of tgds, egds and unegds and propose an algorithm to test it. Related questions were studied in the context of *data quality* and *data cleaning* by [5, 6].

2 Tableaux with unequalities

We extend the definition of a valuation to accomodate unequalities as follows:

Definition 2.1 *If T is a tableau with unequalities and I an instance, a* $\neq$-valuation *for T in I is a function $\beta : var(T) \longrightarrow \mathbb{D}$. We extend β to map any constant to itself. Moreover, we say that that the $\neq$-valuation β* satisfies *T if*

- *for any atom $R(\mathbf{e}) \in T$ the relation R^I contains $\beta(\mathbf{e})$, and*
- *for any atom $e = e' \in T$ we have $\beta(e) = \beta(e')$.*
- *for any atom $e \neq e' \in T$ we have $\beta(e) \neq \beta(e')$.*

Essentially, the difference between valuations and $\neq$-valuations above is the last condition, which defines how unequality atoms can be satisfied. In the rest of this paper, we use the term "valuation" to refer both to common valuations and to $\neq$-valuations (and it is clear from the context whether they have to satisfy unequality atoms or not).

Observe that the introduction of unequalities in a tableau gives rise to another way in which a tableau can be unsatisfiable (compared to tableaux without unequalities, which can only be unsatisfieable if they equate distinct constants):

Proposition 2.2 *A tableau T with unequalities is unsatisfiable iff:*

- *either $T \vdash c_1 = c_2$ for distinct constants c_1, c_2*
- *or $T \vdash x \neq y$ and $T \vdash x = y$, where x, y can be variables or constants.*

The definition of a homomorphism of queries was extended in [10] to that of a $\neq$-homomorphism of queries with unequalities as follows:

Definition 2.3 *Given two conjunctive queries $q = \langle \mathbf{u}, T \rangle$ and $q' = \langle \mathbf{u}', T' \rangle$ a* $\neq$-homomorphism *$h : q' \longrightarrow q$ is a mapping $h : var(T') \longrightarrow var(T) \cup \mathbb{D}$, extended to map any constant to itself, such that:*

- *for any atom $R(\mathbf{e}') \in T'$ we have $T \vdash R(h(\mathbf{e}'))$,*
- *for any atom $e'_1 = e'_2 \in T'$ we have $T \vdash h(e'_1) = h(e'_2)$, and*
- *for any atom $e'_1 \neq e'_2 \in T'$ we have $T \vdash h(e'_1) \neq h(e'_2)$, and*
- *$T \vdash h(\mathbf{u}') = \mathbf{u}$*

In the rest of this paper, we use the term "homomorphism" to refer both to common homomorphisms and to $\neq$-homomorphisms (and it is clear from the context whether they have to satisfy unequality atoms or not).

To every **satisfiable tableau** T we associate a database instance $Inst(T)$ as follows. Let $var(T)$ be the set of variables in T, and $adom(T)$ be the *active domain* of T, i.e., the set of constants occuring in it. Then, T (or, more precisely, the set of equality atoms in T) determines an equivalence relation on $var(T) \cup adom(T)$. Specifically, e is equivalent to e' iff $T \vdash e = e'$. Let us denote by $\hat{e}$ the equivalence class of the variable or constant e. Because T is satisfiable, such an equivalence class can contain at most one constant. We are going to define $Inst(T)$ by interpreting the relation symbols as sets of tuples of such equivalence classes.

Now for any relation symbol R we define

$$\hat{\mathbf{e}} \in R^{Inst(T)} \quad \text{iff} \quad T \vdash R(\mathbf{e})$$

If $\hat{\mathbf{e}} = \hat{\mathbf{e}}'$ then $T \vdash R(\mathbf{e})$ iff $T \vdash R(\mathbf{e}')$ hence this definition does not depend on the choice of representatives from the equivalence classes.

Conversely, to every database instance I we associate a tableau $Tab(I)$ as follows. The variables of the tableau are the elements of $adom(I)$ and the tableau consists of all the relational atoms $R(\mathbf{a})$ such that $R^I(\mathbf{a})$ holds in I. Moreover, for every pair of distinct constants a, b in the instance, the tableau contains the unequality $a \neq b$. Thus, this tableau contains $|I|$ relational and $|adom(I)|^2$ unequality atoms.

3 Containment of CQs with unequalities

3.1 Cases in which the homomorphism criterion holds

The homomorphism theorem [1] gives the existence of a homomorphism as the decidable criterion for conjunctive query containment. In this section we show that the homomorphism criterion holds if the contained query has unequalities but the containing does not. In the next section, we will show that it fails to hold if the containing query has unequalities, unless the contained one is "complete" wrt unequalities .

Theorem 3.1 *1. Let $q \in CQ^{\neq}$ and $q' \in CQ$. Then:*

$$q \sqsubseteq q' \text{ iff there is a homomorphism from } q' \text{ to } q$$

2. Let $q \in CQ^{\neq}$ s.t. q is "complete" wrt equalities and unequalities (see lemma 3.3 and footnote 1) and $q' \in CQ^{\neq}$. Then:

$$q \sqsubseteq q' \text{ iff there is a homomorphism from } q' \text{ to } q$$

Proof

1. Let $q = \langle\, \mathbf{u}, T \,\rangle$ and $q' = \langle\, \mathbf{u}', T' \,\rangle$.

   ($\Rightarrow$) First assume $q \sqsubseteq q'$. It is easy to see that $\hat{\mathbf{u}} \in q(Inst(T))$. For the sake of completeness, we show that $\beta(x) = \hat{x}$ is a valuation that satisfies T over $Inst(T)$, s.t. $\beta(\mathbf{u}) = \hat{\mathbf{u}}$ (by definition).

   - Relational atoms:
     $R(\mathbf{e}) \in T$
     $\Rightarrow \hat{\mathbf{e}} \in R^{Inst(T)}$ by the definition of $Inst(T)$
     $\Rightarrow \beta(\mathbf{e}) \in R^{Inst(T)}$ by the definition of β
   - Equality atoms:
     $e_1 = e_2 \in T$
     $\Rightarrow e_1 \equiv e_2$ by the definition of the equivalence classes of $adom(Inst(T))$
     $\Rightarrow \hat{e_1} = \hat{e_2}$ by the definition of $Inst(T)$
     $\Rightarrow \beta(e_1) = \beta(e_2)$ by the definition of β
   - Non-equality atoms:
     Let $e_1 \neq e_2 \in T$ (1). Suppose, bwoc, that $\hat{e_1} = \hat{e_2}$. By the definition of $Inst(T)$, this means that $T \vdash e_1 = e_2$, which, together with (1), means that T is unsatisfiable. Thus, $\hat{e_1} \neq \hat{e_2}$, i.e., $\beta(e_1) \neq \beta(e_2)$.

Thus, $\hat{\mathbf{u}} \in q(Inst(T))$. Since $q \sqsubseteq q'$, this implies $\hat{\mathbf{u}} \in q'(Inst(T))$. Finally, by lemma 3.3(part 1) there is a homomorphism from q' to q.

($\Leftarrow$) Conversely, assume that $h : q' \longrightarrow q$ is a homomorphism. We wish to show that for any instance and any $\mathbf{a} \in adom(I)$ (this is sufficient because conjunctive queries are domain-independent) we have $\mathbf{a} \in q(I) \Rightarrow \mathbf{a} \in q'(I)$.

By lemma 3.3 (part 2) if $\mathbf{a} \in q(I)$ then there is a homomorphism $g : q \longrightarrow \langle\, \mathbf{a}, Tab(I) \,\rangle$. But homomorphisms compose so $g \circ h : q' \longrightarrow \langle\, \mathbf{a}, Tab(I) \,\rangle$ is also a homomorphism. Again by lemma 3.3 (part 2) we have $\mathbf{a} \in q'(I)$.

2. ($\Rightarrow$) First assume $q \sqsubseteq q'$. As in the proof of (part 1), it is easy to see that $\hat{\mathbf{u}} \in q(Inst(T))$. Since $q \sqsubseteq q'$, this implies $\hat{\mathbf{u}} \in q'(Inst(T))$. Finally, by lemma 3.3 (part 2) there is a homomorphism from q' to q.

   ($\Leftarrow$) Conversely, assume that $h : q' \longrightarrow q$ is a homomorphism. We wish to show that for any instance and any $\mathbf{a} \in adom(I)$ (this is sufficient because conjunctive queries are domain-independent) we have $\mathbf{a} \in q(I) \Rightarrow \mathbf{a} \in q'(I)$.

   By lemma 3.3 (part 2) if $\mathbf{a} \in q(I)$ then there is a homomorphism $g : q \longrightarrow \langle\, \mathbf{a}, Tab(I) \,\rangle$. But homomorphisms compose so $g \circ h : q' \longrightarrow \langle\, \mathbf{a}, Tab(I) \,\rangle$ is also a homomorphism. Again by lemma 3.3 (part 2) we have $\mathbf{a} \in q'(I)$.

These results can also be proved as a corollary of Lemma 3.5 in [17].

□

Corollary 3.2 *1. $q \sqsubseteq q'$ where $q \in CQ^{\neq}$ and $q' \in CQ$ is NP-complete.*

2. $q \sqsubseteq q'$ where $q \in CQ^{\neq}$ is "complete" wrt equalities and unequalities and $q' \in CQ^{\neq}$ is NP-complete.

Proof For both parts, an NP algorithm is implied by the homomorphism theorem. The problem is known to be NP-hard, even for a pair of CQs without unequalities.

□

The proof of theorem 3.1 follows from the following lemma, which also contains the definition of "completeness".

Lemma 3.3 *1. Let $q = \langle\, \mathbf{u}, T \,\rangle \in CQ^{\neq}$, where T is satisfiable, and $q' = \langle\, \mathbf{u}, T' \,\rangle \in CQ$. Then, there is a homomorphism from q' to q iff $\hat{\mathbf{u}} \in q'(Inst(T))$.*

2. Let $q = \langle\, \mathbf{u}, T \,\rangle \in CQ^{\neq}$, where T is satisfiable and $\forall x, y \in var(T) \cup Const$, either $T \vdash x = y$ or $T \vdash x \neq y$[1]. Let $q' = \langle\, \mathbf{u}, T' \,\rangle \in CQ^{\neq}$. Then, there is a homomorphism from q' to q iff $\hat{\mathbf{u}} \in q'(Inst(T))$.

3. Let q be a conjunctive query, I an instance and $\mathbf{a} \in adom(I)$. Then $\mathbf{a} \in q(I)$ iff there is a homomorphism from q to $\langle\, \mathbf{a}, Tab(I) \,\rangle$.

Proof

1. ($\Rightarrow$) Let $h : T' \longrightarrow T$ be a tableau homomorphism. Define $\beta(x) = \widehat{h(x)}$. We show that β satisfies T' over $Inst(T)$ (note that T' only has relational and equality atoms).

   - Relational atoms:
     $R(\mathbf{e}') \in T'$
     $\Rightarrow T \vdash R(h(\mathbf{e}'))$ by the definition of tableau homomorphism (2.3)
     $\Rightarrow \widehat{h(\mathbf{e}')} \in R^{Inst(T)}$ by the definition of $Inst(T)$
     $\Rightarrow \beta(\mathbf{e}') \in R^{Inst(T)}$

[1] Henceforth called "completeness" condition

- Equality atoms:
  $e'_1 = e'_2 \in T'$
  $\Rightarrow T \vdash h(e'_1) = h(e'_2)$ by the definition of tableau homomorphism (2.3)
  $\Rightarrow \widehat{h(e'_1)} = \widehat{h(e'_2)}$ by the definition of $Inst(T)$
  $\Rightarrow \beta(e'_1) = \beta(e'_2)$

Moreover, $\beta(\mathbf{u}') = \widehat{h(\mathbf{u}')} = \hat{\mathbf{u}}$. It follows that $\hat{\mathbf{u}} \in q'(Inst(T))$.

($\Leftarrow$)

Let $\beta : var(T') \longrightarrow adom(Inst(T))$ be a valuation that satisfies T' over $Inst(T)$, s.t., $\beta(\mathbf{u}') = \hat{\mathbf{u}}$. Define $h(x) =$ pick an element of the eq.class $\beta(x)$. We show that h is a tableau homomorphism from T' to T (again, T' only has relational and equality atoms).

- Relational atoms:
  $R(\mathbf{e}') \in T'$
  $\Rightarrow \beta(\mathbf{e}') \in R^{Inst(T)}$ since β satisfies T' over $Inst(T)$
  $\Rightarrow \exists\ \mathbf{e}_1 \in \beta(\mathbf{e}'), R(\mathbf{e}_1) \in T$ by the definition of $Inst(T)$
  $\Rightarrow \forall\ \mathbf{e}_2 \in \beta(\mathbf{e}'), T \vdash R(\mathbf{e}_2)$ by the definition of $Inst(T)$
  $\Rightarrow T \vdash R(h(\mathbf{e}'))$
- Equality atoms:
  $e'_1 = e'_2 \in T'$
  $\Rightarrow \beta(e'_1) = \beta(e'_2)$ since β satisfies T' over $Inst(T)$
  $\Rightarrow h(e'_1) = h(e'_2)$ they are the same variable in $var(T)$, since we pick the same element from the same eq.class
  $\Rightarrow T \vdash h(e'_1) = h(e'_2)$ otherwise, $T \vdash x \neq x$ for some $x \in var(T) \cup C$, and thus T would be unsatisfiable, which is a contradiction
- Output tuple:
  $\beta(\mathbf{u}') = \hat{\mathbf{u}}$ since β satisfies T' over $Inst(T)$
  $\Rightarrow \forall\ \mathbf{u}_1 \in \beta(\mathbf{u}'), T \vdash \mathbf{u}_1 = \mathbf{u}$ by the definition of $Inst(T)$
  $\Rightarrow T \vdash h(\mathbf{u}') = \mathbf{u}$

2. ($\Rightarrow$) Let $h : T' \longrightarrow T$ be a tableau homomorphism. Define $\beta(x) = \widehat{h(x)}$. We show that β satisfies T' over $Inst(T)$ (note that T' only has relational and equality atoms).

- Relational atoms: as in (part 1).
- Equality atoms: as in (part 1).
- Non-equality atoms:
  $e'_1 \neq e'_2 \in T'$
  $\Rightarrow T \vdash h(e'_1) \neq h(e'_2)$ by the definition of tableau homomorphism (2.3)
  $\Rightarrow \widehat{h(e'_1)} \neq \widehat{h(e'_2)}$ by the definition of $Inst(T)$
  $\Rightarrow \beta(e'_1) \neq \beta(e'_2)$

Moreover, $\beta(\mathbf{u}') = \widehat{h(\mathbf{u}')} = \hat{\mathbf{u}}$. It follows that $\hat{\mathbf{u}} \in q'(Inst(T))$.

($\Leftarrow$)

Let $\beta : var(T') \longrightarrow adom(Inst(T))$ be a valuation that satisfies T' over $Inst(T)$, s.t., $\beta(\mathbf{u}') = \hat{\mathbf{u}}$. Define $h(x) =$ pick an element of the eq.class $\beta(x)$. We show that h is a tableau homomorphism from T' to T (again, T' only has relational and equality atoms).

- Relational atoms: as in (part 1).

- Equality atoms: as in (part 1).
- Non-equality atoms:
  $e'_1 \neq e'_2 \in T'$
  $\Rightarrow \beta(e'_1) \neq \beta(e'_2)$ since β satisfies T' over $Inst(T)$
  $\Rightarrow h(e'_1) \neq h(e'_2)$ since they are picked from different eq. classes
  $\Rightarrow T \vdash h(e'_1) \neq h(e'_2)$ otherwise, by our assumption about "completeness" of T, $T \vdash h(e'_1) = h(e'_2)$, which together with $h(e'_1) \neq h(e'_2)$ implies that T is unsatisfiable, which is a contradiction
- Output tuple: as in (part 1).

3. This part follows from (part 2) once we observe that, by construction, $Tab(I)$ satisfies the "completeness" condition and, moreover, since $Tab(I)$ has no equality atoms, there is an isomorphism between $Inst(Tab(I))$ and I such that $\hat{\mathbf{a}}$ corresponds to $\mathbf{a}$.

□

3.2 ... and another in which it does not

The situation wrt testing for containment changes dramatically if the containing query has unequality atoms. For example, consider the boolean queries (adapted from an example in [16]):

$$q() \ :- \ R(x,y), R(y,z), x \neq z$$
$$q'() \ :- \ R(x,y), x \neq y$$

Clearly, q logically implies q' and thus is contained in it, but there is no homomorphism from q' to q. As a result, the homomorphism theorem does not apply in this case. Instead, the following result was proved in [18], as a special case of the more general theorem about containment of CQs with inequalities ($<$).

Theorem 3.4 (from [18]) *Let $q \in CQ^{\neq}$ and $q' \in CQ^{\neq}$. Then, testing whether $q \sqsubseteq q'$ is Π_2^P-complete.*

In fact, van der Meyden shows that containment is Π_2^P-complete even for monadic conjunctive queries (theorem 7.1 in [18]) A Π_2^Palgorithm that employs the chase is outlined in [7].

4 Containment under dependencies with unequalities and the chase

Dependencies are first-order sentences of a particular form (see below). This form is the result of successive generalizations of various definitions of *integrity constraints*, i.e., assertions that database instances are expected to satisfy in order to agree with the semantic intuition of the db designers.

Definition 4.1 *A* tuple-generating dependency *(*tgd*) is a sentence of the form*

$$(d) \qquad \forall \mathbf{x} \ (B \rightarrow \exists \mathbf{y} \, C)$$

where B and C are tableaux, which are "read" as the conjunction of their elements, and where $\mathbf{x} \subseteq var(B)$ and $\mathbf{y} \subseteq var(C)$. Because (d) is a sentence, we must also have $var(B) \subseteq \mathbf{x}$ and $var(C) \subseteq \mathbf{x} \cup \mathbf{y}$.

Definition 4.2 *An* (un)equality-generating dependency *((*un)egd*) is a sentence of the form*

$$(d) \qquad \forall \mathbf{x} \ (B \rightarrow \exists \mathbf{y} \, C)$$

where B is a conjunction of relational atoms and C is a conjunction of (un)egds, and where $\mathbf{x} \subseteq var(B)$ and $\mathbf{y} \subseteq var(C)$. Because (d) is a sentence, we must also have $var(B) \subseteq \mathbf{x}$ and $var(C) \subseteq \mathbf{x} \cup \mathbf{y}$.

The chase procedure can be used to prove containment of CQs under a set of tgds and egds. We now show that it can also be used to prove containment of CQs (and in some cases CQ$^{\neq}$s) under a set of tgds, egds and unegds. In the case of dependencies with unequalities in the body, we show that the chase theorem fails, even for queries without unequalities. However, an extension of the chase (with higher data complexity) proposed by [7], can be used to prove containment of CQs (with or without unequalities) under such dependencies.

4.1 Dependencies with unequalities in the head

To define the chase step for arbitrary queries and dependencies we need to talk about *homomorphisms of tableaux*. This is the same as what was defined earlier as homomorphism of queries, except that we do not need the fourth condition (that relates the output variables).

Definition 4.3 *Consider the dependency* $d \stackrel{\text{def}}{=} \forall \mathbf{x}\ (B \rightarrow \exists \mathbf{y}\, C)$ *where B is a tableau with equalities and C is either a tableau without equalities or unequalities (i.e., a tgd) or a conjunction of equalities (egd) or unequalities (unegd). Let T be a tableau with equalities and unequalities. We say that the* chase *with d is* applicable *to T if there exists a homomorphism $h : B \longrightarrow T$ that cannot be extended to $B \cup C$, i.e., there is* no *homomorphism $h' : B \cup C \longrightarrow T$ that is equal to h on $var(B)$*[2]*. When the chase is applicable, the* result *of one step of chase of T with d is the tableau $T' \stackrel{\text{def}}{=} T \cup C[\mathbf{x} := h(\mathbf{x})]$ and we write $T \stackrel{d}{\rightarrow} T'$.*

In defining the chase result, we assume that $\mathbf{y}$ is disjoint from $var(T)$. If this is not the case we rename the bound variables $\mathbf{y}$. For example, we will certainly need such renaming in a sequence of multiple chase steps with the same dependency.

Recall our earlier construction that associates to every satisfiable tableau T a database instance $Inst(T)$.

Lemma 4.4 *Let d be a tgd, egd or unegd. If the chase with d is not applicable to a satisfiable tableau T then $Inst(T) \models d$.*

Proof We prove the contrapositive. Suppose $Inst(T) \not\models d$. Then, there exists a *valuation* $\beta : \mathbf{x} \longrightarrow adom(Inst(T))$ that satisfies the atoms in B over $Inst(T)$ and such that there is no extension $\beta' : \mathbf{x} \cup \mathbf{y} \longrightarrow adom(Inst(T))$ that additionally satisfies the atoms in C. Let $q' = \langle\, (), B \,\rangle$, then $\texttt{true} \in q'(Inst(T))$. By lemma 3.3.1, since B has no unequalities, there is a homomorphism of tableaux $h : B \longrightarrow T$. This homomorphism cannot be extended to $B \cup C$, because the extension would yield an extension of β that satisfies C. It follows that the chase with d is applicable to T.

□

Definition 4.5 *We define the chase on conjunctive queries using their underlying tableaux: if $q \stackrel{\text{def}}{=} (\mathbf{x}, T)$ and $T \stackrel{d}{\rightarrow} T'$ then $q \stackrel{d}{\rightarrow} q'$ where $q' \stackrel{\text{def}}{=} (\mathbf{x}, T')$.*

Lemma 4.6 *If $q \stackrel{d}{\rightarrow} q'$ then $d \models q \equiv q'$.*

Proof Since the tableau of q is a subset of the tableau of q' we have a trivial homomorphism $q \longrightarrow q'$ hence $q' \sqsubseteq q$.

It remains to prove that if I is an instance such that $I \models d$ then $q(I) \subseteq q'(I)$. Let $d = \forall \mathbf{x}\ (B \rightarrow \exists \mathbf{y}\, C)$, $q = (\mathbf{u}, T)$, and $q' = (\mathbf{u}, T')$. Let $h : B \longrightarrow T$ be the homomorphism used in the chase step hence $T' = T \cup C[\mathbf{x} := h(\mathbf{x})]$. Since the output tuple is the same and $T \subseteq T'$ it suffices to show that any valuation

[2]for *egds* and *unegds* this just means that h does not satisfy the condition in the head of the dependency

in I that satisfies T can be extended to a valuation that satisfies T'. Let $\beta : T \longrightarrow I$ be such a valuation. It follows that $\beta \circ h : B \longrightarrow I$ satisfies B in I and since $I \models d$, $\beta \circ h$ can be extended to a valuation γ that satisfies C in I. Now define $\beta' : C[\mathbf{x} := h(\mathbf{x})] \longrightarrow I$ as follows: if $z \in \mathbf{y}$ then $\beta'(z) \stackrel{\texttt{def}}{=} \gamma(z)$ but if $z \in h(\mathbf{x})$ then $\beta'(z) \stackrel{\texttt{def}}{=} \beta(z)$.

Thus β' extends β and what's left is to check that β' satisfies T'. Clearly β' satisfies the atoms that are also in T. Now suppose, (keeping the notation simple) that $R(h(x), y)$ is an atom in $C[\mathbf{x} := h(\mathbf{x})]$ obtained from the atom $R(x, y)$ in C, where $x \in \mathbf{x}, y \in \mathbf{y}$. We have $(\beta'(h(x)), \beta'(y)) = (\beta(h(x)), \gamma(y)) = (\gamma(x)), \gamma(y)) \in R^I$ because γ satisfies C. For equality and unequality atoms, C only contains variables in $\mathbf{x}$ and it suffices to define $\beta'(x) \stackrel{\texttt{def}}{=} \beta(x)$. For equality, suppose $d : \forall \mathbf{x}\ \phi(\mathbf{x}) \rightarrow x_1 = x_2$. Then, $\beta' \circ h$ is a homomorphism $\phi \longrightarrow I$ and $\beta' \circ h(x_1) = \beta \circ h(x_1) = \beta \circ h(x_2) = \beta' \circ h(x_2)$ and thus $\beta'(h(x_1)) = \beta'(h(x_2))$. The proof for unequalities is identical. Thus, for every $I \models d$ and every valuation β $T \longrightarrow I$, β is also a valuation $T' \longrightarrow I$ and as a result, $\mathbf{u} \in q(I)$ implies $\mathbf{u} \in q'(I)$, i.e., $q(I) \subseteq q'(I)$.

□

Corollary 4.7 *If $q \stackrel{d}{\rightarrow} q'$ and q' is unsatisfiable, then $d \models$ (q is unsatisfiable)*[3].

Definition 4.8 *Let q be a conjunctive query and D a set of dependencies. A* terminating chase sequence *of the query q with dependencies from D is a sequence of chase steps*

$$q \stackrel{d_1}{\rightarrow} q_1 \stackrel{d_2}{\rightarrow} q_2 \cdots \stackrel{d_n}{\rightarrow} q_n$$

such that $d_1, \ldots, d_n \in D$ and such that no chase with dependencies from D is applicable to q_n.

Corollary 4.9 *(of lemma 4.6) If $q \stackrel{d_1}{\rightarrow} q_1 \stackrel{d_2}{\rightarrow} q_2 \cdots \stackrel{d_n}{\rightarrow} q_n$ such that $d_1, \ldots, d_n \in D$ is a terminating chase sequence, then $D \models q \equiv q_n$*

In general, the chase with an arbitrary set of dependencies is not guaranteed to terminate; however, [11] and [7] have identified *weak acyclicity* as a sufficient condition of a set of dependencies, to guarantee termination of any chase sequence with that set. As a result, there is a *decision* procedure for containment of queries under weakly acyclic set of dependencies, employing the chase, as the following theorem shows:

Theorem 4.10 *Let $q \in CQ^{\neq}$, $q' \in CQ$ and let D be a set of* egds, unegds and weakly acyclic tgds. *Let q_n be the last chase result in a terminating chase sequence of q with D. Then*

$$D \models q \sqsubseteq q' \quad \textit{iff} \quad q_n \sqsubseteq q'$$

Proof By corollary 4.9 $D \models q \equiv q_n$, so if $q_n \sqsubseteq q'$ it also follows that $D \models q \sqsubseteq q'$.

For the other implication, let T_n be the tableau underlying q_n and let $I_n \stackrel{\texttt{def}}{=} \mathit{Inst}(T_n)$. If T_n is unsatisfiable, q_n returns the empty set on all inputs, hence it is contained in any query, in particular q'. Assume that T_n is satisfiable. By lemma 4.4 above, $I_n \models D$; hence $I_n \models q \sqsubseteq q'$ or $q(I_n) \subseteq q'(I_n)$, because $D \models q \sqsubseteq q'$.

Now let $q \stackrel{\texttt{def}}{=} (\mathbf{u}, T)$ and $q' \stackrel{\texttt{def}}{=} (\mathbf{u}', T')$. Note that $T \subseteq T_n$ because each chase step just adds atoms. So there is a mapping $\sigma : \mathit{var}(T) \longrightarrow \mathit{adom}(I_n)$ that associates each variable to its equivalence class, $\sigma(x) = \hat{x}$. It follows (by lemma 3.3 (part 1)) that $\hat{\mathbf{u}} = \sigma(\mathbf{u})$ is in $q(I_n)$, and therefore also in $q'(I_n)$. Hence there must exist a mapping $\beta : \mathit{var}(T') \longrightarrow \mathit{adom}(I_n)$ such that $\beta(\mathbf{u}') = \hat{\mathbf{u}}$. It is now straightforward to show that we have a homomorphism $q' \longrightarrow q_n$.

□

[3]Since in the proof above we did not assume anywhere that q' was satisfiable

Corollary 4.11 *Let $q \in CQ^{\neq}$, $q' \in CQ$ and D a set of* egds, unegds and weakly acyclic tgds. *Let q_n be the last chase result in a terminating chase sequence of q with D. Then*

$$D \models q \sqsubseteq q' \quad \textit{iff there is a homomorphism from } q' \textit{ to } q_n$$

The same method can be used when the containing query is complete wrt equalities and unequalities

Corollary 4.12 *Let $q \in CQ^{\neq}$, $q' \in CQ^{\neq}$, D a set of* egds, unegds and weakly acyclic tgds *and q_n be the last chase result in a terminating chase sequence of q with D. If q_n is complete wrt unequalities, then*

$$D \models q \sqsubseteq q' \quad \textit{iff there is a homomorphism from } q' \textit{ to } q_n$$

4.2 Satisfiability of conjunctive queries with unequalities under dependencies

A simple, but practically interesting, application of theorem 4.10 is the following: we say that q is *unsatisfiable under D* if for all $I \models D$, $q(I) = \emptyset$.

Corollary 4.13 *Let $q \in CQ^{\neq}$, $q' \in CQ$ and let D be a set of* egds, unegds and weakly acyclic tgds. *Let q_n be the last chase result in a terminating chase sequence of q with D. Then q is unsatisfiable under D iff $q_n = \perp$.*

Proof Clearly, q is unsatisfiable under D iff $D \models q \sqsubseteq q_\emptyset$, where $q_\emptyset$ is the unsatisfiable/empty query (i.e., for all I, $q_\emptyset(I) = \emptyset$).

By theorem 4.10, $D \models q \sqsubseteq q_\emptyset$ iff $q_n \sqsubseteq q_\emptyset$ iff $q_n = \perp$.

□

In fact, if q is satisfiable under D, we can strengthen the statements of Theorem 4.10 and Corollary 4.11: one only needs to chase q with the tgds and egds in D (i.e., ignoring the unegds), since the containing query doesn't have any unequalities, so any containment mapping from q' to q would have to map relational atoms of q' to relational atoms of the result of the chase (i.e., the existence or not of unequality atoms in the result of the chase does not affect the existence of a homomorphism).

Theorem 4.14 [4] *Let $q \in CQ^{\neq}$, $q' \in CQ$, D be a set of* egds, unegds and weakly acyclic tgds *and $D^+ \subseteq D$ a set of* tgds and egds *only. Let q_n be the last chase result in a terminating chase sequence of q with D^+. Then $D \models q \sqsubseteq q'$ iff*

- *either q is unsatisfiable under D*
- *or $q_n \sqsubseteq q'$ (and there is a hom. q' to q_n)*

[4] This result is due to Alin Deutsch

4.3 Dependencies with unequalities in the body

First, observe that lemma 4.4 does not hold for dependencies with unequalities in the body. In particular, let $T = R(x, y)$ and $d = \forall x, y\ R(x, y) \wedge x \neq y \rightarrow x = y$. The chase with d is not applicable on T because there is no unequality atom in T to which a $\neq$ homomorphism could map the atom $x \neq y$ in d. Hence, if the lemma above would hold, $Inst(T) \models d$. However, by the definition of $Inst(T)$, $\hat{x} \neq \hat{y}$ and thus there is a valuation $h : var(d) \longrightarrow T$ - namely $h(x) = \hat{x}$, $h(y) = \hat{y}$ - that satisfies the body of d, but not its head. Therefore, $Inst(T) \not\models d$.

For a simple counterexample of theorem 4.10, let:

$$q(x, y) : -R(x, y)$$
$$d \stackrel{\texttt{def}}{=} \forall x, y\ R(x, y) \wedge x \neq y \longrightarrow x = y$$

d is only satisfied by the empty instance $\emptyset$ and instances where $R(x, y)$ are such that $x = y$. It is easy to see that for such instances, $D \models q \sqsubseteq q'$, where $q'(x, x) : -R(x, x)$. Observe that the chase on q is not applicable with d, hence $chase_d(q) = q$. Suppose, bwoc, that theorem 4.10 holds, then $q \sqsubseteq q'$. However, there are instances, e.g., $I = \{R(a, b)\}$, in which $q(I) \not\subseteq q'(I)$, which is a contradiction.

Remark. From the result in [18], we know that containment of queries in $CQ^{\neq}$ is Π_2^P-hard, so obviously containment of queries in $CQ^{\neq}$ under dependencies (whether they have unequalities on the head or body or both or not at all) is Π_2^P-hard. However, it would be also interesting to know the complexity of containment of CQs (without unequalities) under dependencies with unequalities in the body. The corresponding mappings are also interesting: they are produced as quasi-inverses [12] of sets of tgds and egds. Such queries and mappings are studied in [7, 8], where the authors propose techniques to deal with such dependencies, that have however Π_2^P complexity (see below). The lower bound for the case mentioned above seems to be open, as it doesn't follow directly from the lower bound in [18]. Nonetheless, we conjecture that the problem is still Π_2^P-hard.

In [7] the authors propose an extension to the chase technique that accommodates unequalities, as well as negated relational atoms, in both the body and the head of the dependencies. To simplify the presentation we just highlight the application of their theorem for unequalities:

Theorem 4.15 (from [7]) *Let D be a weakly acyclic set of dependencies, $q, q' \in UCQ^{\neq}$ and $d : \forall x, y\ \mathsf{true} \rightarrow x = y \vee x \neq y$. Let $q^D = chase_{D \cup \{d\}}(q)$. Then $D \models q \sqsubseteq q'$ iff $q^D \sqsubseteq q'$.*

Notice that, in the theorem above, the *disjunctive* chase [9] is used (because of the disjunction in d). As a result, q^D is a union of queries with unequalities, and $q^D \sqsubseteq q'$ iff each of these queries is contained in q'. Moreover, it is easy to observe that all queries in q^D satisfy the "completeness" condition in lemma 3.3 (part 2) (otherwise, the chase with d would be applicable), and thus the homomorphism theorem can be used to check containment in q', for each one of them. More precisely:

Corollary 4.16 *$D \models q \sqsubseteq q'$ iff $\forall q_i \in q^D$, there is a homomorphism $q' \longrightarrow q_i$*

Observe that, the corollary above also gives a Π_2^P algorithm for deciding containment between queries in $CQ^{\neq}$: chase q with d and find a homomorphism from q' into every query in the result of the chase.

Since dependencies with unequalities in the body are "unfriendly", even for the fairly simple problem of containment, in the rest of this paper we are going to investigate other problems, such as data exchange and consistency, only for the case of dependencies with unequalities in the head, i.e., *unegds*.

5 Unegds and data exchange

In this section we discuss the problem computing solutions of the data exchange problem, for settings where the mappings also include unegds, as well as query answering of CQs in such settings. In particular, we

want to extend the technique used in [11] in order to accomodate mappings that contain unegds without increase in complexity [5]. A small difficulty arises from the fact that the solution proposed in [11] involves chasing instances to produce other instances that satisfy the mappings. Since traditional relational instances only contain relational atoms (i.e., no unequalities), and the chase with unegds introduces such atoms, we cannot express the output of a chase step as an instance. For this reason, in the case of data exchange with unegds, the output of every step of the chase, as well as the whole data exchange algorithm, is a *pair* (K, C) of a relational instance K and a conjunction of unequalities C, similar to the representation formalism of *g-tables* [2]. Note that, this pair also forms a tableaux with unequalities, as the ones discussed above, and as a result the properties of the chase that were proved in the previous section also apply in this case.

Finally, we say that there is a homomorphism $h : (K, C) \longrightarrow J$, where J is a database instance (no unequalities) iff there is a valuation (as defined in section 2) that satisfies the tableau (K, C) over J.

5.1 Computing Universal Solutions

First, we extend a basic property of a chase step that was proved in [11] for a set of tgds and egds and is used to prove that the chase produces a universal solution, for the case with unegds:

Lemma 5.1 *Let* $(K_1, C_1) \xrightarrow{d,h_d} (K_2, C_2)$ *be a chase step where* $C_2 \neq \perp$[6]. *Let* K *be an instance such that: (i)* $K \models d$ *(ii) there exists a homomorphism* $h_1 : (K_1, C_1) \longrightarrow K$.

Then there exists a homomorphism $h_2 : (K_2, C_2) \longrightarrow K$.

Proof

We have three cases, according to the form of d.

1. $d : \forall \mathbf{x}\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y})$: In this case, $C_2 = C_1$. By the definition of the chase step, $h_d : \phi(\mathbf{x}) \longrightarrow K_1$ is a homomorphism. As a result, $h_1 \circ h_d : \phi(\mathbf{x}) \longrightarrow K$ is also a homomorphism. Since $K \models d$, $h_1 \circ h_d$ can be extended to a homomorphism $h' : \phi(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y}) \longrightarrow K$ s.t. h' is an extension of $h_1 \circ h_d$. Let Λ_y be the labeled null introduced by the chase step for variable y. Define h_2 as follows: $h_2(z) = h_1(z), z \in var(K_1)$ and $h_2(\Lambda_y) = h'(y), y \in \mathbf{y}$. We need to show that h_2 is a homomorphism from (K_2, C_1) to K. This is obvious for facts and unequality atoms in (K_2, C_1) that also were in (K_1, C_1), so we only need to show this for the new (relational) facts introduced by the chase step. Let $T(\mathbf{x}', \mathbf{y}')$ be a fact in $\psi(\mathbf{x}, \mathbf{x})$, where $\mathbf{x}', \mathbf{y}'$ are subsets of the variables in $\mathbf{x}, \mathbf{y}$, respectively. Then, K_2 contains a fact $T(h_d(\mathbf{x}'), \Lambda_{\mathbf{y}'})$. By the definition of h_2, $h_2(T(h_d(\mathbf{x}'), \Lambda_{\mathbf{y}'})) = T(h_2(h_d(\mathbf{x}')), h_2(\Lambda_{\mathbf{y}'})) = T(h'(\mathbf{x}'), h'(\mathbf{y}')) = h'(T(\mathbf{x}', \mathbf{y}'))$. Since h' is a homomorphism $\phi \wedge \psi \rightarrow K$, it maps $T(\mathbf{x}', \mathbf{y}')$ into a fact of K. Thus, we conclude that $h_2(T(\mathbf{x}', \mathbf{y}'))$ is a fact of K, i.e., h_2 is a homomorphism $K_2 \longrightarrow K$.

2. $d : \forall \mathbf{x}\phi(\mathbf{x}) \rightarrow x_1 = x_2$: Again, $C_2 = C_1$ and $h_1 \circ h_d : \phi(\mathbf{x}) \longrightarrow K$ is a homomorphism. Define $h_2 = h_1$. Since $C_2 = C_1$, h_2 obviously satisfies the unequality atoms in C_2. Moreover, since there are no new facts in K_2, the only way for h_2 to not be a homomorphism from $(K_2, C_1) \longrightarrow K$ is if h_1 maps $h_d(x_1)$ and $h_d(x_2)$ to different constants or labeled nulls of K. However, this is not the case, since $K \models d$ and $h_1 \circ h_d$ is a homomorphism $\phi(\mathbf{x}) \longrightarrow K$ implies that, $h_1 \circ h_d(x_1) = h_1 \circ h_d(x_2)$ or $h_1(h_d(x_1)) = h_1(h_d(x_2))$. Finally, h_2 clearly respects C_2 (by hypothesis, since $h_2 = h_1$ and $C_2 = C_1$)

3. $d : \forall \mathbf{x}\phi(\mathbf{x}) \rightarrow x_1 \neq x_2$: In this case $C_2 = C_1 \wedge h_d(x_1) \neq h_d(x_2)$, $K_2 = K_1$ and $h_1 \circ h_d : \phi(\mathbf{x}) \longrightarrow K$ is a homomorphism. Again define $h_2 = h_1$. Since $K_2 = K_1$, h_2 is obviously satisfies the relational atoms of K_2. Suppose, towards a contradiction, that h_2 does not satisfy some unequality in C_2. Since $h_2 = h_1$ satisfies the unequalities in C_1, this means that h_2 does not satisfy $h_d(x_1) \neq h_d(x_2)$, i.e., $h_2 \circ h_d(x_1) = h_2 \circ h_d(x_2)$, which implies $h_1 \circ h_d(x_1) = h_1 \circ h_d(x_2)$. However, since $K \models d$ and $h_1 \circ h_d : \phi(\mathbf{x}) \longrightarrow K$ is a homomorphism, $h_1 \circ h_d(x_1) \neq h_1 \circ h_d(x_2)$, which is a contradiction.

□

[5] [8, 3] defined extended chase procedures for mappings with negation in both head and body (NDEDs), but its complexity is higher

[6] i.e., C_2 is satisfiable

Then, the data exchange algorithm proposed in [11] can be extended as follows:

Theorem 5.2 *Assume a data exchange setting where Σ_{st} is a set of tgds and Σ_t is a set of tgds, egds and unegds.*

1. *Let $\langle\,(I, true), (J, C)\,\rangle$ be the result of some successful finite chase of $\langle\,(I, true), (\emptyset, true)\,\rangle$ with $\Sigma_{st} \cup \Sigma_t$. Then (J, C) is a universal solution.*
2. *If there exists some failing finite chase of $\langle\,(I, true), (\emptyset, true)\,\rangle$ with $\Sigma_{st} \cup \Sigma_t$, then there is no solution.*

Proof
The following proof is a direct adaptation of the proof of [11] for the case of tgds and egds.

1. By lemma 4.4, we know that $\langle\,I, (J, C)\,\rangle \models \Sigma_{st} \cup \Sigma_t$. Since Σ_t only uses target relation symbols, it follows that $(J, C) \models \Sigma_t$. Let J' be an arbitrary solution, i.e., $\langle\,I, J'\,\rangle \models \Sigma_{st} \cup \Sigma_t$. Since the identity mapping $id : \langle\,I, (\emptyset, \text{true})\,\rangle \longrightarrow \langle\,I, J'\,\rangle$ is a homomorphism, by applying lemma 5.1 inductively on the chase sequence that led to $\langle\,I, (J, C)\,\rangle$, we obtain a homomorphism $h : \langle\,I, (J, C)\,\rangle \longrightarrow \langle\,I, J'\,\rangle$ which is also a homomorphism $(J, C) \longrightarrow J'$. Thus, J is universal.

2. Let $\langle\,I, (J_k, C_k)\,\rangle \xrightarrow{d, h_d} \perp$ be the failing chase step; in order for the chase to fail, d must be an egd or unegd. In both cases, $d \in \Sigma_t$ so $\langle\,I, J\,\rangle \models d\text{iff} J \models d$. There are two cases:

   - $d : \forall \mathbf{x}\ \phi(\mathbf{x}) \rightarrow x_1 = x_2$. There are two ways in which this chase step can fail:
     (a) $h_d(x_1) = c_1 \neq c_2 = h_d(x_2)$, where c_1, c_2 are distinct constants. Suppose, bwoc, that there is a J s.t. $\langle\,I, J\,\rangle \models \Sigma_s t \cup \Sigma_t$. By induction and the lemma above (similarly with the proof of (1)) there is a homomorphism $h' : (J_k, C_k) \rightarrow J$. Since homomorphisms compose, $h' \circ h_d : \phi \rightarrow J$ is a homomorphism. Since $\langle\,I, J\,\rangle \models \Sigma_s t \cup \Sigma_t$ it follows that $J \models d$, hence $h' \circ h_d(x_1) = h' \circ h_d(x_2) \Rightarrow h'(c_1) = h'(c_2)$. Since homomorphisms are the identity on constants, this implies $c_1 = c_2$, which is a contradiction.
     (b) $h_d(x_1) \neq h_d(x_2) \in C_k$. Suppose, bwoc, that there is a J s.t. $\langle\,I, J\,\rangle \models \Sigma_s t \cup \Sigma_t$. By induction and the lemma above (similarly with the proof of (1)) there is a homomorphism $h' : (J_k, C_k) \rightarrow J$. Since $h_d(x_1) \neq h_d(x_2) \in C_k$, this implies that $h'(h_d(x_1) \neq h'(h_d(x_2))$. However, $h' \circ h_d : \phi \rightarrow J$ is a homomorphism and, since $J \models d$, $h' \circ h_d(x_1) = h' \circ h_d(x_2)$, which is a contradiction.
   - $d : \forall \mathbf{x}\ \phi(\mathbf{x}) \rightarrow x_1 \neq x_2$. For this step to fail, it must be the case that $h_d(x_1) = h_d(x_2)$. Suppose, bwoc, that there is a J s.t. $\langle\,I, J\,\rangle \models \Sigma_s t \cup \Sigma_t$. By induction and the lemma above (similarly with the proof of (1)) there is a homomorphism $h' : (J_k, C_k) \rightarrow J$. Since homomorphisms compose, $h' \circ h_d : \phi \rightarrow J$ is a homomorphism. Since $J \models d$, $h' \circ h_d(x_1) \neq h' \circ h_d(x_2)$ or $h'(h_d(x_1)) \neq h'(h_d(x_2))$, which is a contradiction, since h' is a function and $h_d(x_1) = h_d(x_2)$.

□

Note that the universal "solution" computed by the algorithm above is not an actual instance; instead it is a representation of several possible instances that satisfy the mapping and one can compute many actual solutions from it, by finding assignments of values to the labeled nulls that satisfy the unequalities in C.

5.2 Query Answering

In [11] the authors showed that if J is a universal solution in a data exchange setting, $certain(q, I) = q(J)_{\downarrow}$, where $q(J)_{\downarrow}$ is the result of dropping tuples with labeled nulls from $q(J)$. As we showed above, the solution computed in the case of data exchange involving *unegds* is also universal, according to the definition of [11]. A small subtlety is related with the fact that this universal solution is not an actual instance. However, this is

easily overcome, since, as it is shown in several papers about representation of incomplete information (e.g., [15], [13], [2]), the global condition does not affect query answering in any way; thus, one can essentially ignore the conjunction of unequalities in the universal solution. As a result, the proof of proposition 4.2 in [11] still holds in the case of unegds.

6 Consistency of Mappings

In a P2P setting, different peers specify mappings independently, so it is important to guarantee that when a new mapping is added, it doesn't "break the whole system". We define consistency of a set of mappings Σ to capture this notion. Formally:

Definition 6.1 *A set Σ of constraints/mappings over a schema S is consistent, if there exists an instance J of S s.t. for every relation $R \in S$, $J(R) \neq \emptyset$ and $J \models \Sigma$. Every such J is called a consistent instance for Σ.*

Note that any set of tgds and egds without constants is consistent: let a be a constant in the domain, then it is straightforward to check that the instance containing one tuple in every relation with the value a for every attribute satisfies any such dependencies. The problem is more interesting in the presence of constants and/or unegds. Consider, for example, the set of dependencies:

$\forall x_1, x_2 \; R(x_1, x_2) \rightarrow S(x_1, x_1)$
$\forall x_1, x_2 \; S(x_1, x_2) \rightarrow x_1 \neq x_2$

It is easy to see that no non-empty instance of R, S can satisfy these dependencies.

One can view this problem as a special case of the data exchange problem, where $\Sigma_{st} = \emptyset$, $\Sigma_t = \Sigma$, $I = \emptyset$. An additional restriction is that we are looking for a non-empty solution (otherwise, $\emptyset$ always satisfies any set of mappings, and one can easily see that applying the data exchange algorithm on the setting outlined above would indeed always yield as a canonical universal solution). However, as we will show in the sequel, this problem is easily overcome and the basic techniques used in data exchange can be applied to check whether a set of mappings is consistent. First, we need to define some basic notions:

Definition 6.2 *J is a universal consistent instance for a set of constraints Σ if for every consistent instance J' there is a homomorphism from J into J'.*

Let I_0 be an instance of S s.t. for every $R \in S$, $I_0(R)$ is a single tuple and the values of every attribute of t are distinct labeled nulls (variables),[7] and values of attributes of tuples of different relations are also distinct.

Theorem 6.3 *Let Σ be a set of* egds, unegds and weakly acyclic tgds *over schema S.*

1. *Let (J, C) be the result of some successful finite chase of $(I_0, true)$ with Σ. Then J is a universal consistent instance.*
2. *If there exists some failing finite chase of $(I_0, true)$ with Σ, then Σ is inconsistent.*

Proof

1. By lemma 4.4, we know that $(J, C) \models \Sigma$. Let J' be an arbitrary solution, i.e., $J' \models \Sigma$. We show that there is a homomorphism from J into J' by induction on the length of the chase sequence.

   Basis: There is obviously a homomorphism from I_0 into any instance with non-empty relations, and every consistent instance belongs to this set.

   Inductive step: Follows directly from lemma 5.1.

[7] note that these are distinct from constants that may appear in the dependencies

2. The proof of this part is essentially identical to the proof of theorem 5.2 (part 2).

□

Note that, as in the case of data exchange, the output of the algorithm in the successful case is not an actual instance, but instead a representation of a set of instances. A different proof that this problem (as well as implication for this kind of dependencies) can be solved in PTIME was given in [4].

Acknowledgements We are grateful to Alin Deutsch for several very useful discussions.

References

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

[2] S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. *Theoretical Computer Science*, 78(1):158–187, 1991.

[3] F. Afrati, C. Li, and V. Pavlaki. Data exchange with arithmetic comparisons. In *EDBT*, 2008.

[4] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-generating dependencies. *J. Comput. Syst. Sci.*, 59(1):94–115, 1999.

[5] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, pages 746–755, 2007.

[6] L. Bravo, W. Fan, and S. Ma. Extending dependencies with conditions. In *VLDB*, pages 243–254, 2007.

[7] A. Deutsch, B. Ludäscher, and A. Nash. Rewriting queries using views with access patterns under integrity constraints. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 352–367, 2005.

[8] A. Deutsch, A. Nash, and J. Remmel. Data exchange, data integration, and chase. Technical Report CS2006-0859, University of California San Diego, 2006.

[9] A. Deutsch and V. Tannen. Optimization Properties for Classes of Conjunctive Regular Path Queries. In *DBPL*, pages 21–39, 2001.

[10] A. Deutsch and V. Tannen. XML queries and constraints, containment and reformulation. *Theoretical Computer Science*, 336(1):57–87, 2005.

[11] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 2005.

[12] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Quasi-inverses of schema mappings. In *PODS*, pages 123–132, 2007.

[13] G. Grahne. *The Problem of Incomplete Information in Relational Databases*. Springer-Verlag, LNCS 554, 1991.

[14] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Update exchange with mappings and provenance. In *VLDB*, pages 675–686, 2007.

[15] T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.

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

[17] P. G. Kolaitis, D. L. Martin, and M. N. Thakur. On the complexity of the containment problem for conjunctive queries with built-in predicates. In *PODS*, pages 197–204, 1998.

[18] R. van der Meyden. Recursively indefinite databases. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 364–378, 1990.