Locally Consistent Transformations and Query Answering in Data Exchange

Marcelo Arenas
University of Toronto
marenas@cs.toronto.edu

Pablo Barceló
University of Toronto
pablo@cs.toronto.edu

Ronald Fagin
IBM Almaden Research Center
fagin@almaden.ibm.com

Leonid Libkin
University of Toronto
libkin@cs.toronto.edu

ABSTRACT

Data exchange is the problem of taking data structured under a source schema and creating an instance of a target schema. Given a source instance, there may be many solutions – target instances that satisfy the constraints of the data exchange problem. Previous work has identified two classes of desirable solutions: canonical universal solutions, and their cores. Query answering in data exchange amounts to rewriting a query over the target schema to another query that, over a materialized target instance, gives the result that is semantically consistent with the source. A basic question is then whether there exists a transformation sending a source instance into a solution over which target queries can be answered.

We show that the answer is negative for many data exchange transformations that have structural properties similar to canonical universal solutions and cores. Namely, we prove that many such transformations preserve the *local* structure of the data. Using this notion, we further show that every target query rewritable over such a transformation cannot distinguish tuples whose neighborhoods in the source are similar. This gives us a first tool that helps check whether a query is rewritable. We also show that these results are robust: they hold for an extension of relational calculus with grouping and aggregates, and for two different semantics of query answering.

1. Introduction

Data exchange is the problem of materializing an instance that adheres to a target schema, given an instance of a source schema and a specification of the relationship between the source and the target. This is a very old problem [27] that arises in many tasks where data must be transferred between independent applications that do not have the same data format. The need for data exchange has steadily increased over the years. With the proliferation of web data in various formats and with the emergence of e-business applications that need to communicate data yet remain autonomous, data exchange is even more important.

A data exchange setting is a triple ($\mathbf{S}$, $\mathbf{T}$, Σ_{st}), where $\mathbf{S}$ is the source schema, $\mathbf{T}$ is the target schema, and Σ_{st} is a set of source-to-target dependencies that express the relationship between $\mathbf{S}$ and $\mathbf{T}$ (some papers also add a set Σ_t of dependencies that express constraints on $\mathbf{T}$, but here, we will mostly consider data exchange settings with no target constraints). Such a setting gives rise to the following *data exchange problem*: given an instance I over the source schema $\mathbf{S}$, find an instance J over the target schema $\mathbf{T}$ such that I together with J satisfy the source-to-target dependencies Σ_{st} (when target dependencies are used, J must also satisfy them). Such an instance J is called a *solution* for I in the data exchange setting. In general, there may be many different solutions for a given source instance I. For a data exchange system, the two key issues are:

1. Which solution should be materialized?

2. How should queries be answered over the target?

Papers [9, 10] started a systematic investigation of these issues for data exchange settings in which $\mathbf{S}$ and $\mathbf{T}$ are relational schemas. They isolated a class of solutions, called *universal solutions*, possessing good properties that justify selecting them as the best solutions in data exchange. Specifically, universal solutions have homomorphisms into every possible solution; in particular, they have homomorphisms into each other, and thus are homomorphically equivalent. Universal solutions are the most general among all solutions and, in a precise sense, they represent the entire space of solutions. It was shown in [9] that under fairly general conditions, universal solutions exist, and a *canonical* universal solution

PODS 2004, June 14-16, 2004, Paris, France.

can be found in polynomial time, based on the classical chase procedure [4, 26].

Since universal solutions need not be unique, this raises the question of which universal solution to materialize. The answer proposed in [10] is based on using *minimality* as a key criterion for what constitutes the "best" universal solution. Although universal solutions come in different sizes, all of them share a unique (up to isomorphism) common "part", which is nothing else but the *core* of each of them, when they are viewed as relational structures [10]. By definition, the core of a structure is the smallest substructure that is also a homomorphic image of the structure. The concept of the core originated in graph theory, where a number of its properties have been established [16]. It was shown in [10] that if the source-to-target dependencies Σ_{st} are tuple-generating dependencies (tgds), then the core of the universal solutions for I is itself a solution for I (they also allow the possibility of having certain sets Σ_t of target dependencies). Hence, the core of the universal solutions for I is the *smallest* universal solution for I, and thus an ideal candidate for the "best" solution, at least in terms of the space required to materialize it. Furthermore, in a number of cases of interest, they show that there is a polynomial-time algorithm for generating the core.

We now turn to discussing query answering, and the related issue of query rewriting [24, 14]. Given a source instance and a data exchange setting, what is the meaning of the "answer" to a query Q over the target schema? Since there may be multiple solutions to the data exchange problem, the standard approach is to define the answer to be the set of *certain answers* [20, 1], that is, those tuples that appear in $Q(J)$ for every solution J. The goal of query answering in data exchange is to find these certain answers based on just *one* materialized target instance.

If Q is a union of conjunctive queries, and J is an arbitrary universal solution, then [9] showed that the certain answers are given exactly by the set of all tuples in $Q(J)$ that are formed entirely of elements from the source. Such nice behavior fails when we go beyond unions of conjunctive queries: it was shown in [9] that there is a Boolean conjunctive query Q with inequalities such that $Q(J)$ does not give the certain answers, no matter which universal solution J is selected, but for some other first-order query Q' (a *rewriting* of Q), the certain answers for Q are given by $Q'(J)$, where J is the canonical universal solution. Unfortunately, query rewritability is not a general phenomenon either, as [9] constructed a Boolean conjunctive query Q with inequalities for which there is no such rewriting Q'.

But the following basic question remains unanswered: is there a transformation $\mathcal{F}$ that maps each source instance I into a solution $\mathcal{F}(I)$ and a rewriting Q' such that the certain answers are given by $Q'(\mathcal{F}(I))$? Of course we want to forbid cheating solutions (like encoding the answer to a Boolean query with a self-loop). But what is a natural condition then to impose on a transformation? Such a condition must ensure a certain degree of "uniformity" of $\mathcal{F}$ (that is, it should not be tailored to deal with a specific query), and be satisfied by the transformations commonly used in data exchange such as $\mathcal{F}_{\text{univ}}$ that maps the source instance I onto the[1] canonical universal solution, or $\mathcal{F}_{\text{core}}$ that maps I onto the core of the universal solutions.

The condition we impose on a transformation $\mathcal{F}$ is that it must be *locally consistent*, that is, points with similar neighborhoods in the source have similar neighborhoods in the target. We make this notion of "locally consistent" precise (in fact, there are two closely related but incomparable properties based on the exact definition of "similarity"), and prove that, in an appropriate data exchange setting, $\mathcal{F}_{\text{univ}}$ and $\mathcal{F}_{\text{core}}$ possess both properties.

One of our main results is that the failure of the canonical universal solution to support rewriting is not because there is a "better" choice of solution. Specifically, we show that if the transformation that produces the solution is locally consistent, then there are first-order queries that are not rewritable. This implies that neither the canonical universal solution, nor the core, nor any other "uniformly" generated solution supports rewriting for arbitrary first-order queries. We prove this by showing that queries rewritable over locally consistent transformations cannot distinguish points that have isomorphic neighborhoods in the source instance. Unlike ad hoc techniques employed in [10, 5], this criterion gives us easy ways of showing that a query is not rewritable.

The notion of local consistency introduced in this paper is a new one; although it is inspired by standard notions of locality from logic [12, 15, 11], it is different from them since this is the first notion of locality that applies to transformations that invent new values.

We also prove two extensions of the main results. The first one concerns the underlying query language; we shows that all the results continue to hold if instead of first-order queries, we use an extension with grouping and aggregate functions, that is, essentially the `select-from-where-groupby-having` fragment of SQL. Second, we look at an alternative semantics (proposed in [10]) for query answering in data exchange that, instead of taking certain answers (those tuples that appear in $Q(J)$ for every solution J), takes tuples that appear in $Q(J)$ for every *universal* solution J. This is reasonable, because the universal solutions are the desirable solutions in data exchange. We prove that the main results of the paper remain true under this semantics.

Organization Basic notions related to data exchange, universal solutions, cores, and neighborhoods are pre-

[1] We refer to "the" canonical universal solution, although in the scenario of [9], it is not unique. We shall define it in this paper in a way where it is uniquely determined.

sented in Section 2. In Section 3 we study structural properties of data exchange transformations. We first present a rule-based language that allows us to code many such transformations, and to prove local consistency for programs in that language. We derive results for $\mathcal{F}_{\rm univ}$ and $\mathcal{F}_{\rm core}$ as corollaries. We also briefly consider extensions with target dependencies.

In Section 4, we study query rewritability. We show that a query rewritable over any locally consistent transformation cannot distinguish constants whose neighborhoods in the source are isomorphic. We show that this property gives us easy non-rewritability results. We also establish a connection between rewritability over the core, and rewritability over the canonical universal solution.

In Section 5, we present extensions of these results to languages with grouping and aggregation, and to the semantics based on universal solutions. Summary and concluding remarks are given in Section 6. All proofs will appear in the full version.

2. Preliminaries

A *schema* $\mathbf{R}$ is a finite sequence $\langle R_1, \ldots, R_k \rangle$ of relation symbols, with each R_i having a fixed arity n_i. An *instance* I *of* $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a finite n_i-ary relation $I(R_i)$. The *domain* $\mathrm{dom}(I)$ of instance I is the set of all elements that occur in any of the relations $I(R_i)$.[2] An instance J of $\mathbf{R}$ is a *subinstance* of I if $\mathrm{dom}(J) \subseteq \mathrm{dom}(I)$ and $J(R_i) \subseteq I(R_i)$, for every i. If one of the inclusions is proper, we refer to J as a *proper subinstance* of I. If $\mathbf{R}$ is a schema, then a *dependency over* $\mathbf{R}$ is a sentence in some logical formalism over $\mathbf{R}$, typically FO (first-order logic).

2.1 Data exchange setting

Let $\mathbf{S} = \langle S_1, \ldots, S_n \rangle$ and $\mathbf{T} = \langle T_1, \ldots, T_m \rangle$ be two schemas with no relation symbols in common. We refer to $\mathbf{S}$ as the *source* schema and to the S_i's as the source relation symbols. We refer to $\mathbf{T}$ as the *target* schema and to the T_j's as the target relation symbols. We denote by $\langle \mathbf{S}, \mathbf{T} \rangle$ the schema $\langle S_1, \ldots, S_n, T_1, \ldots, T_m \rangle$. Instances over $\mathbf{S}$ will be called source instances, while instances over $\mathbf{T}$ will be called target instances. If I is a source instance and J is a target instance, then (I, J) denotes an instance K over $\langle \mathbf{S}, \mathbf{T} \rangle$ such that $K(S_i) = I(S_i)$ and $K(T_j) = J(T_j)$, for $i \in [1, n]$ and $j \in [1, m]$.

[2] An instance is a special case of an $\mathbf{R}$-structure $\mathbf{A}$ defined as $(A, R_1^{\mathbf{A}}, \ldots, R_k^{\mathbf{A}})$, where A is a set (the *universe*), and $R_i^{\mathbf{A}} \subseteq A^{n_i}$ for each i. Thus, in the case of arbitrary structures, the universe may contain elements that are not present in any of the relations.

A *source-to-target dependency* (std) is a sentence of the form

$$\forall \bar{x} \big(\varphi_{\mathbf{S}}(\bar{x}) \rightarrow \exists \bar{y}\, \psi_{\mathbf{T}}(\bar{x}, \bar{y}) \big),$$

where $\varphi_{\mathbf{S}}(\bar{x})$ is a formula over $\mathbf{S}$ in some logical formalism (typically FO) and $\psi_{\mathbf{T}}(\bar{x}, \bar{y})$ is a conjunction of FO atomic formulae over $\mathbf{T}$.

Definition 2.1. **(Data Exchange Setting)** *A* data exchange setting *is a triple* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$*, where* $\mathbf{S}$ *is a source schema,* $\mathbf{T}$ *is a target schema, and* Σ_{st} *is a set of source-to-target dependencies. The* data exchange problem *associated with* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *is the following: given a source instance* I*, find a target instance* J *such that* (I, J) *satisfies* Σ_{st}*. Such a* J *is called a* solution *for* I*, or simply a* solution *if* I *is clear from the context.*

We denote by Const an infinite set of all values that may occur in source instances, and, following the data exchange terminology [9, 10], we call those values *constants*. In addition, we also assume an infinite set Var of elements, disjoint from Const. Elements of Var are called *nulls* [9, 10], and they are used to help populate target instances. That is, the domain of a target instance comes from Const $\cup$ Var.

If I is an instance with values in Const $\cup$ Var, then Const(I) denotes the set of all constants occurring in relations in I, and Var(I) denotes the set of nulls occurring in relations in I. From now on, we assume that there is a way to distinguish constants from nulls. For example, we may assume that the target schema $\mathbf{T}$ contains an auxiliary predicate C whose interpretation is $\mathrm{dom}(I) \cap$ Const.

Papers [9, 10] identified two important subclasses of data exchange, inspired by the *local-as-view (LAV)* and *global-as-view (GAV)* classes of data integration problems [23]:

- *LAV setting*: each dependency in Σ_{st} is of the form $S(\bar{x}) \rightarrow \exists \bar{y}\, \psi_{\mathbf{T}}(\bar{x}, \bar{y})$, where S is some relation symbol in the source schema $\mathbf{S}$, and, as before, $\psi_{\mathbf{T}}(\bar{x}, \bar{y})$ is a conjunction of atomic formulae over $\mathbf{T}$.
- *GAV setting*: each dependency in Σ_{st} is of the form $\varphi_{\mathbf{S}}(\bar{x}) \rightarrow T(\bar{x})$, where T is a relation symbol in the target schema $\mathbf{T}$. If $\varphi_{\mathbf{S}}(\bar{x})$ is a conjunctive query, we speak of the GAV(CQ) setting.[3]

Observe that the general data exchange can be seen as a composition of LAV and GAV.

[3] In [9, 10], the formula $\varphi_{\mathbf{S}}(\bar{x})$ was restricted to being the conjunction of atomic formulae over $\mathbf{S}$, that is, to the GAV(CQ) setting.

Example 2.2. Consider a LAV data exchange setting in which $\mathbf{S} = \langle M(\cdot,\cdot),\ N(\cdot,\cdot)\rangle$, $\mathbf{T} = \langle P(\cdot,\cdot,\cdot),\ Q(\cdot,\cdot)\rangle$ and Σ_{st} contains the following stds:

$$\begin{array}{lcl} M(x,y) & \rightarrow & \exists w \exists z (P(x,y,z) \wedge Q(w,z)), \\ N(x,y) & \rightarrow & \exists z P(x,y,z). \end{array}$$

Suppose we are given a source instance $I = \{M(a,b),\ N(a,b)\}$.[4] Since the stds in Σ_{st} do not completely specify the target, there are multiple solutions that are consistent with the specification. One solution is:

$$J = \{P(a,b,n_1), P(a,b,n_2), Q(n_3,n_1)\},$$

where n_1, n_2, n_3 are values in Var. Another solution, but with no nulls, is $J' = \{P(a,b,a),\ Q(b,a)\}$. $\square$

2.2 Universal solutions and cores

Let J and J' be two instances over the target schema $\mathbf{T}$ with values in $\mathsf{Const} \cup \mathsf{Var}$. A *homomorphism* $h : J \rightarrow J'$ is a mapping from $\mathsf{Const}(J) \cup \mathsf{Var}(J)$ to $\mathsf{Const}(J') \cup \mathsf{Var}(J')$ such that $h(c) = c$ for all $c \in \mathsf{Const}(J)$, and $\bar{t} \in J(R)$ implies[5] $h(\bar{t}) \in J'(R)$ for all $R \in \mathbf{T}$. Furthermore, we say that J and J' are *homomorphically equivalent* if there are homomorphisms $h : J \rightarrow J'$ and $h' : J' \rightarrow J$.

Definition 2.3. **(Universal solution)** *If I is a source instance in a data exchange setting, then a* universal solution for *I is a solution J such that for every solution J' for I, there exists a homomorphism $h : J \rightarrow J'$.*

Example 2.4. The solution J' in Example 2.2 is not universal, since there is no homomorphism from J' to J. On the other hand, it can be shown that J is a universal solution. $\square$

It was shown in [9] that universal solutions possess good properties that justify selecting them (as opposed to arbitrary solutions) for the semantics of the data exchange problem. A universal solution is more general than an arbitrary solution because, by definition, it can be homomorphically mapped into that solution. Moreover, all universal solutions are homomorphically equivalent. Furthermore, results of [9] imply that for the data exchange setting considered in this paper, universal solutions always exist.

To deal with the problem of computing universal solutions, [9] proposes to compute a special kind of universal solution, called a *canonical universal solution*. The algorithm presented in [9] is based on applying the chase, but we shall define canonical universal solutions directly, in Section 3, when we introduce a general class of programs that define data transformations that invent new values.

The reason one wants to compute a specific solution for the data exchange problem is to be able to evaluate queries over the target schema. It was noted in [9] that universal solutions need not be isomorphic, and thus any decision to choose one is somewhat arbitrary. To deal with this problem, [10] proposed to use the *core* of the universal solutions.

Definition 2.5. **(Core)** *A subinstance J of an instance I is called a* core *of I if there is a homomorphism from I to J, but there is no homomorphism from I to a proper subinstance of J.*

It is known [16] that every instance has a unique core (up to isomorphism). It is shown in [10] that if the source-to-target dependencies are tuple-generating dependencies (tgds), then every universal solution has the same core (up to isomorphism), and this core is itself a universal solution. Further, it is shown in [10] that under the assumptions in this paper, the core can be computed in polynomial time.

Example 2.6. In Example 2.2, $J^* = \{P(a,b,n_1),\ Q(n_3,n_1)\}$ is the core of the universal solutions. $\square$

2.3 Neighborhoods and locality

The *Gaifman graph* $\mathcal{G}(I)$ of an instance I of $\mathbf{R}$ is the graph whose nodes are the elements of $\mathrm{dom}(I)$, and such that there exists an edge between a and b in $\mathcal{G}(I)$ iff a and b belong to the same tuple of a relation $I(R)$, for some $R \in \mathbf{R}$. For example, if I is an undirected graph, then $\mathcal{G}(I)$ is I itself.

The distance between two elements a and b in I, denoted by $d_I(a,b)$ (or $d(a,b)$, if I is understood), is the distance between them in $\mathcal{G}(I)$. We define $d(\bar{a}, b)$ as the minimum value of $d(a,b)$ where a is an element of $\bar{a}$.

Given a tuple $\bar{a} = (a_1, \ldots, a_m) \in \mathrm{dom}(I)^m$, we define the instance $N_d^I(\bar{a})$, called the *d-neighborhood of $\bar{a}$ in I*, as the restriction of I to the elements at distance at most d from $\bar{a}$, with the members of $\bar{a}$ treated as distinguished elements. That is, if two neighborhoods $N_d^I(\bar{a})$ and $N_d^I(\bar{b})$ are isomorphic (written as $N_d^I(\bar{a}) \cong N_d^I(\bar{b})$), then there is an isomorphism $f : N_d^I(\bar{a}) \rightarrow N_d^I(\bar{b})$ such that $f(a_i) = b_i$, for $1 \leq i \leq m$.

The notion of neighborhoods allows one to define *locality* of logics. A formula $\varphi(\bar{x})$ in some logical formalism is *local* if there exists a number d such that $N_d^I(\bar{a}) \cong N_d^I(\bar{b})$ implies that $I \models \varphi(\bar{a})$ iff $I \models \varphi(\bar{b})$, for every instance I. It is known [12] that all FO formulae are local. This was generalized to logics that extend FO with counting [25] and aggregate functions [18].

[4] It is often convenient to define instances by simply listing the tuples attached to the corresponding relation symbols.

[5] If $\bar{t} = (t_1, \ldots, t_k)$, then by $h(\bar{t})$ we mean $(h(t_1), \ldots, h(t_k))$.

3. Structural Properties of Data Exchange Transformations

In this section we show that data exchange transformations preserve the local character of the data. As a first step towards proving those results, we formulate a rule-based language for specifying transformations such as $\mathcal{F}_{\text{univ}}$, that maps the source instance I onto the canonical universal solution. This language is similar in spirit to languages with oid invention [19, 28] but its rules are nonrecursive. Based on the types of logical formulae used in rules, we establish different results on locality of transformations, and then derive, as corollaries, exact characterizations of locality for various data exchange settings.

3.1 Data exchange programs

A *data exchange program* is a quadruple $\Pi = (\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathcal{R})$, where $\mathbf{S}$ ("source"), $\mathbf{A}$ ("auxiliary") and $\mathbf{T}$("target") are pairwise disjoint relational schemas and $\mathcal{R}$ is a sequence $\langle r_1, \ldots, r_n\rangle$ of rules such that each rule is of the form

$$R_1(\bar{x}_1, \bar{y}_1), \ldots, R_k(\bar{x}_k, \bar{y}_k) \quad :- \quad \varphi(\bar{x}), \qquad (1)$$

where each R_i is either in $\mathbf{A}$ or in $\mathbf{T}$, where $\varphi(\bar{x})$ is an FO formula over $\langle \mathbf{S}, \mathbf{A}\rangle$, where variables in the $\bar{x}_i$'s are among those in $\bar{x}$, and variables in the $\bar{y}_i$'s are not in $\bar{x}$. We refer to $R_1(\bar{x}_1, \bar{y}_1), \ldots, R_k(\bar{x}_k, \bar{y}_k)$ as the *head* of the rule, and to $\varphi(\bar{x})$ as the *body* of the rule.

Furthermore, we require that the program be *stratified*. That is, if $\mathbf{A}_i$ is the set of relation symbols from $\mathbf{A}$ used in rules $r_1, \ldots, r_i$, then the formula φ in the body of rule r_{i+1} is over the schema $\langle \mathbf{S}, \mathbf{A}_i\rangle$.

Example 3.1. Consider $\mathbf{S} = \langle S(\cdot, \cdot)\rangle$, $\mathbf{A} = \langle R(\cdot, \cdot), N(\cdot)\rangle$, $\mathbf{T} = \langle T(\cdot, \cdot)\rangle$, and rules r_1 and r_2 defined as follows:

$$R(x, z),\ R(z, y),\ N(z) :- S(x, y) \qquad (r_1)$$

$$T(x, y) :- \exists z (R(x, z) \wedge R(z, y) \wedge N(x) \wedge N(y)) \quad (r_2)$$

If $\mathcal{R} = \langle r_1, r_2\rangle$, then $(\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathcal{R})$ is a data exchange program. Notice also that if $\mathcal{R}' = \langle r_2, r_1\rangle$, then $(\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathcal{R}')$ is not a data exchange program (because it is not stratified). $\square$

Given a data exchange program $\Pi = (\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathcal{R})$, we define the transformation $\mathcal{F}_\Pi : \text{inst}(\mathbf{S}) \to \text{inst}(\mathbf{T})$ that associates a target schema instance with each source schema instance. For that, we show inductively how to define a mapping $\mathcal{F}^i_\Pi : \text{inst}(\langle \mathbf{S}, \mathbf{A}, \mathbf{T}\rangle) \to \text{inst}(\langle \mathbf{S}, \mathbf{A}, \mathbf{T}\rangle)$ given by the first i rules of the program. Suppose we are given an instance I of $\mathbf{S}$, and $J = \mathcal{F}^{i-1}_\Pi(I)$, where $1 \le i \le n$ (if $i = 1$, then $J(S) = I(S)$ for every $S \in \mathbf{S}$, and $J(P) = \emptyset$ for every $P \in \langle \mathbf{A}, \mathbf{T}\rangle$). Let the ith rule be given by (1), let $\bar{u}$ be the tuple of variables in $\bar{x}$ that are used in the head of the rule, and let $\bar{v}$ be the tuple of variables in the head of the rule that are not in $\bar{x}$.

For each tuple $\bar{a}$ of length $|\bar{u}|$ over $\text{dom}(J)$, find all the tuples $\bar{b}_1, \ldots, \bar{b}_m$ such that $J \models \varphi(\bar{a}, \bar{b}_j)$, for $1 \le j \le m$. Then choose m tuples of length $|\bar{v}|$ of fresh distinct null values $\bar{n}_1, \ldots, \bar{n}_m$ over Var. To construct relation R_l, for $l \le k$, in $\mathcal{F}^i_\Pi(I)$, add tuples $(\pi_{\bar{x}_l}(\bar{a}), \pi_{\bar{y}_l}(\bar{n}_j))$, for $1 \le j \le m$, to the relation $J(R_l)$. Here $\pi_{\bar{x}_l}(\bar{a})$ refers to the components of $\bar{a}$ that occur in the positions of $\bar{x}_l$.

Example 3.2. Consider $\Pi = (\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathcal{R})$ as defined in Example 3.1, and an instance $I = \{S(a, b), S(b, a)\}$. Initially, $\mathcal{F}^0_\Pi(I) = I$. We next deal with the rule (r_1). The only variable that occurs in the head but not in the body of (r_1) is z, and hence, to compute $\mathcal{F}^1_\Pi(I)$, we must invent nulls corresponding to that variable.

Since $I \models S(a, b)$, we must choose a null n_1, and add tuples $R(a, n_1), R(n_1, b)$ and $N(n_1)$. Furthermore, since $I \models S(b, a)$, we must pick up a fresh null n_2 (that is, $n_1 \neq n_2$), and add tuples $R(b, n_2), R(n_2, a)$ and $N(n_2)$. Hence, $\mathcal{F}^1_\Pi(I)$ expands I with

$$\{R(a, n_1), R(n_1, b), N(n_1), R(b, n_2), R(n_2, a), N(n_2)\}.$$

In (r_2), there are no variables present in the head that are not free variables of the body, and hence computing $\mathcal{F}^2_\Pi(I)$ amount to evaluating the query given by the body of (r_2) over $\mathcal{F}^1_\Pi(I)$, and adding the result to $\mathcal{F}^1_\Pi(I)$. Thus, $\mathcal{F}^2_\Pi(I)$ is the expansion of $\mathcal{F}^1_\Pi$ with $\{T(n_1, n_2), T(n_2, n_1)\}$. $\square$

Finally, $\mathcal{F}_\Pi(I)$ is defined to be the restriction of $\mathcal{F}^n_\Pi(I)$ to the predicates in $\mathbf{T}$. In Example 3.2, $\mathcal{F}_\Pi(I)$ is the restriction of $\mathcal{F}^2_\Pi$ to T, that is, $\{T(n_1, n_2), T(n_2, n_1)\}$.

Next, we connect data exchange problems with data exchange settings defined earlier. Given a data exchange setting $\mathcal{DE} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, define a data exchange program $\Pi_{\mathcal{DE}} = (\mathbf{S}, \emptyset, \mathbf{T}, \mathcal{R})$, where, for each std

$$\varphi_{\mathbf{S}}(\bar{x}) \to \exists \bar{y}\, (R_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge R_k(\bar{x}_k, \bar{y}_k))$$

in Σ_{st}, we add a rule

$$R_1(\bar{x}_1, \bar{y}_1), \ldots, R_k(\bar{x}_k, \bar{y}_k) :- \varphi_{\mathbf{S}}(\bar{x})$$

to $\mathcal{R}$. In fact, these data exchange programs $\Pi_{\mathcal{DE}}$ are exactly the data exchange programs without auxiliary relation symbols, that is, with the auxiliary schema $\mathbf{A}$ empty. Notice that in the absence of the auxiliary schema $\mathbf{A}$, the order of the rules in $\mathcal{R}$ could be arbitrary (although, as we shall discuss shortly, the order of the rules in $\mathcal{R}$ may affect the result of applying the corresponding transformation).

Definition 3.3. **(Canonical Universal Solution)** *The* canonical universal solution *of instance I in data exchange setting $\mathcal{DE} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ is $\mathcal{F}_{\Pi_{\mathcal{DE}}}(I)$. If the data exchange setting $\mathcal{DE}$ is understood, we shall denote this transformation $\mathcal{F}_{\Pi_{\mathcal{DE}}}$ by $\mathcal{F}_{\text{univ}}$.*

This definition differs slightly from the one given in [9], where a canonical universal solution is not determined uniquely as it is obtained by using the classical chase procedure [4, 26]. Since the result of the chase depends on the order in which the chase steps are applied, there may be multiple nonisomorphic canonical universal solutions under the definition in [9] (even when there are no target constraints). Our definition uses a deterministic procedure that constructs a unique canonical universal solution (since, effectively, our approach fixes the order in which the chase steps are applied). It can easily be shown that $\mathcal{F}_{\text{univ}}(I)$, for every instance I, is a universal solution.

In the definition of $\Pi_{\mathcal{DE}}$, we did not use any auxiliary relations from $\mathbf{A}$. One may then ask if auxiliary relations are ever necessary. The next result says that they are. Thus, with auxiliary relations, one can define transformations that do not arise in any data exchange setting $\mathcal{DE}$. Therefore, in what follows, we prefer to prove results for the more expressive data exchange programs rather than the more restrictive transformation $\mathcal{F}_{\text{univ}}$.

Proposition 3.4. *There is a data exchange program that is not equivalent to any data exchange program with no auxiliary relations.*

Finally, we define the transformation $\mathcal{F}_{\text{core}}$ such that $\mathcal{F}_{\text{core}}(I)$ is the core of $\mathcal{F}_{\text{univ}}(I)$.

3.2 Locally consistent transformations

In this section we introduce the notions of local consistency of transformations from $\text{inst}(\mathbf{S})$ to $\text{inst}(\mathbf{T})$. The first notion says that neighborhoods around elements common to the input and output instances are preserved. Informally, if $a, b \in \text{dom}(I)$ are present in the domain of the resulting instance J of $\mathbf{T}$, then the isomorphism of sufficiently large neighborhoods of a and b in I guarantees that their neighborhoods are isomorphic in J as well. Formally, we define this as follows.

Definition 3.5. **(Local Consistency)** *A mapping $\mathcal{F} : inst(\mathbf{S}) \to inst(\mathbf{T})$ is* locally consistent *if for every $m, d \geq 0$ there exists $d' \geq 0$ such that, for every instance I of $\mathbf{S}$ and m-tuples $\bar{a}, \bar{b} \in dom(I)^m$, if $N^I_{d'}(\bar{a}) \cong N^I_{d'}(\bar{b})$, then*

1) $\bar{a} \in dom(\mathcal{F}(I))^m \ \Leftrightarrow\ \bar{b} \in dom(\mathcal{F}(I))^m$, and

2) $N^{\mathcal{F}(I)}_d(\bar{a}) \cong N^{\mathcal{F}(I)}_d(\bar{b})$.

We next present a sufficient condition for a mapping $\mathcal{F}_\Pi$ associated with a data exchange program Π to be locally consistent. This condition will guarantee local consistency for the LAV setting of data exchange.

We say that a formula $\varphi(\bar{x})$ is *r-bounded* if for every structure I such that $I \models \varphi(a_1, \ldots, a_n)$, it is the case that $d_I(a_i, a_j) \leq r$ for every $i, j \leq n$. A data exchange program Π is r-bounded if every formula in the body of every rule is r-bounded.

Lemma 3.6. *The transformation $\mathcal{F}_\Pi$ of every r-bounded data exchange program is locally consistent.* $\square$

Theorem 3.7. *In the LAV setting, both the canonical universal solution transformation $\mathcal{F}_{\text{univ}}$ and the core transformation $\mathcal{F}_{\text{core}}$ are locally consistent.* $\square$

The result for the canonical universal solution is an immediate consequence of Lemma 3.6, since in a LAV setting $\mathcal{DE}$, the bodies of rules in $\Pi_{\mathcal{DE}}$ are atomic predicates (since they are left-hand sides of the stds), which are 1-bounded. The result for the core requires a separate proof, which will be given in the full version.

One can also show that local consistency for the core transformation depends crucially on the requirement of the data exchange setting that constants be preserved. That is, if homomorphisms are not required to be identity on Const (which is the usual setting in the graph-theoretic literature on the core [16]), then one can find examples of graphs for which the core transformation is not locally consistent.

Theorem 3.7 does not extend to the GAV setting, even when restricted to conjunctive queries.

Proposition 3.8.

(a) There are GAV(CQ) settings such that Σ_{st} contains either one dependency of the form $\varphi_{\mathbf{S}}(x, y, z) \to T(x, y, z)$, or multiple dependencies of the form $\varphi_{\mathbf{S}}(x, y) \to T(x, y)$, and the corresponding transformations $\mathcal{F}_{\text{univ}}$ and $\mathcal{F}_{\text{core}}$ are not *locally consistent.*

(b) If, in the GAV(CQ) setting, Σ_{st} contains only one dependency of the form $\varphi_{\mathbf{S}}(x, y) \to T(x, y)$, then $\mathcal{F}_{\text{univ}}$ and $\mathcal{F}_{\text{core}}$ are locally consistent.

Since local consistency is a nontrivial property of FO-definable mappings, it follows that the question of whether $\mathcal{F}_\Pi$ is locally consistent is undecidable, even in the GAV setting (this easily follows from Trakhtenbrot's theorem; cf. [6]).

3.3 Local consistency under logical equivalence

We have seen that mappings that arise in the LAV setting are locally consistent, and that local consistency may fail even in some simple GAV settings. To overcome the failure of local consistency, we introduce a notion of locality based on logical equivalence (in particular, FO-equivalence) rather than isomorphism of neighborhoods, and we prove that in general, the canonical universal solution transformation $\mathcal{F}_{\text{univ}}$ and the core transformation $\mathcal{F}_{\text{core}}$ are locally consistent under FO-equivalence.

The *quantifier rank* of an FO formula is the maximum depth of quantifier nesting in it. If I and J are instances of the same schema, we write $I \equiv_k J$ if I and J satisfy the same FO sentences of quantifier rank up to k. In the new notion of local consistency, we require that $\equiv_{k'}$-equivalent neighborhoods be sent to $\equiv_k$-equivalent neighborhoods. Formally, we define it as follows.

Definition 3.9. **(Local Consistency under FO-equivalence)** *A mapping $\mathcal{F} : inst(\mathbf{S}) \to inst(\mathbf{T})$ is* locally consistent under FO-equivalence *if for every $m, d, k \geq 0$ there exist $d', k' \geq 0$ such that, for every instance I of $\mathbf{S}$ and m-tuples $\bar{a}, \bar{b} \in dom(I)^m$, if $N^I_{d'}(\bar{a}) \equiv_{k'} N^I_{d'}(\bar{b})$, then*

1) $\bar{a} \in dom(\mathcal{F}(I))^m \;\Leftrightarrow\; \bar{b} \in dom(\mathcal{F}(I))^m$, and

2) $N^{\mathcal{F}(I)}_d(\bar{a}) \equiv_k N^{\mathcal{F}(I)}_d(\bar{b})$.

Lemma 3.10. *The transformation $\mathcal{F}_\Pi$ of every data exchange program is locally consistent under FO-equivalence.* □

Theorem 3.11. *For an arbitrary data exchange setting, both the canonical universal solution transformation $\mathcal{F}_{\text{univ}}$ and the core transformation $\mathcal{F}_{\text{core}}$ are locally consistent under FO-equivalence.* □

The result for the canonical universal solution is an immediate consequence of Lemma 3.10. The result for the core is considerably harder and relies on the machinery developed for the proof of a later theorem (Theorem 4.9).

Note that the definitions of local consistency and local consistency under FO-equivalence are incomparable: the latter makes a weaker assumption and arrives at a weaker conclusion. Nevertheless, either definition works for our applications in query rewriting, because the statement we need there makes the stronger assumption (isomorphism of neighborhoods) but needs only the weaker conclusion (FO equivalence of neighborhoods).

3.4 Adding target dependencies

Papers [9, 10] considered an extension of the data exchange setting in which dependencies exist for the target schema as well. A solution is then required to satisfy those target dependencies.

Based on familiar classes of dependencies (cf. [7, 4]), we define extensions of the data exchange setting with tuple-generating dependencies (tgds) as well as equality-generating dependencies (egds). The tgds over $\mathbf{T}$ are of the form $\forall \bar{x}(\varphi_{\mathbf{T}}(\bar{x}) \to \exists \bar{y}\, \psi_{\mathbf{T}}(\bar{x}, \bar{y}))$, where $\varphi_{\mathbf{T}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x}, \bar{y})$ are conjunctions of FO atomic formulae. The egds over $\mathbf{T}$ are of the form $\forall \bar{x}(\varphi_{\mathbf{T}}(\bar{x}) \to (x_1 = x_2))$, where $\varphi_{\mathbf{T}}(\bar{x})$ is a conjunction of atomic FO formulae, with free variables $\bar{x}$, and where x_1, x_2 are in $\bar{x}$. If, furthermore, the data exchange setting is restricted to LAV or GAV, we shall speak of LAV+tgd settings, LAV+egd settings, and so on. The next proposition covers all four possible settings: LAV+tgd, GAV+tgd, LAV+egd, and GAV+egd.

Proposition 3.12.

(a) The transformations $\mathcal{F}_{\text{univ}}$ and $\mathcal{F}_{\text{core}}$ of LAV+tgd (or GAV+tgd) settings are not necessarily locally consistent (under FO-equivalence), even if the target schema contains only one dependency.

(b) The transformations $\mathcal{F}_{\text{univ}}$ and $\mathcal{F}_{\text{core}}$ of GAV+egd settings are locally consistent under FO-equivalence.

(c) The transformations $\mathcal{F}_{\text{univ}}$ and $\mathcal{F}_{\text{core}}$ of LAV+egd settings are not necessarily locally consistent (under FO-equivalence), even if the target schema contains only key dependencies.

4. Query Rewriting and Locality

In this section, we study query rewriting in data exchange. We use local consistency to show that rewritable queries have a certain kind of locality property. This property gives an easily applicable tool for proving nonexistence of rewritings over the canonical universal solution and the core.

4.1 The query rewriting problem

Suppose we have a data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$, and a query Q over the *target* schema $\mathbf{T}$. What does it mean to answer Q? Since there are many possible solutions to the data exchange problem, the standard approach is to define the semantics of Q in terms of *certain answers*: that is, for an instance I of $\mathbf{S}$,

$$\underline{certain}(Q, I) \;=\; \bigcap_{J \text{ is a solution for } I} Q(J).$$

Thus, a tuple $\bar{a}$ is in $\underline{certain}(Q, I)$ if it belongs to $Q(J)$ for all solutions J for I.

But how can one find this set $\underline{certain}(Q, I)$, given that there are potentially infinitely many solutions? The approach proposed in [9, 10] is to look for some specific transformations $\mathcal{F} : \text{inst}(\mathbf{S}) \to \text{inst}(\mathbf{T})$, and find conditions under which $\underline{certain}(Q, I)$ equals $Q'(\mathcal{F}(I))$. Then Q is rewritable over $\mathcal{F}$ by Q'. More formally, we have the following definition.

Definition 4.1. **(Query Rewriting)** *Given a data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$, a mapping $\mathcal{F}$:*

inst($\mathbf{S}$) $\to$ *inst*($\mathbf{T}$) *and an* m*-ary query* Q *over* $\mathbf{T}$*, we say that* Q *is* rewritable over $\mathcal{F}$ *if there exists an* m*-ary FO query* Q' *over* $\mathbf{T}$ *such that*

$$\underline{certain}(Q, I) = Q'(\mathcal{F}(I))$$

for every instance I *of* $\mathbf{S}$.

We shall refer to a query as being rewritable over the canonical universal solution if it is rewritable over $\mathcal{F}_{\text{univ}}$, and rewritable over the core if it is rewritable over $\mathcal{F}_{\text{core}}$. We now note that rewritability is undecidable in general.

Proposition 4.2. *Given a data exchange setting* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *and an FO query* Q *over* $\mathbf{T}$*, it is undecidable whether* Q *is rewritable over the canonical universal solution, or over the core.* $\square$

In some cases, we can establish that a class of queries is or is not rewritable. For example, it is known that for every FO sentence, its asymptotic probability is either 0 or 1 (this is the zero-one law for FO [8]).

Proposition 4.3. *In a data exchange setting, every Boolean query whose asymptotic probability is* 0 *is rewritable, by false, over both the canonical universal solution and over the core.* $\square$

However, such partial results do not give us any *techniques* for proving that queries are *not* rewritable. We shall now exhibit such techniques, based on the notions of locality from the previous section.

4.2 Local source-dependency and rewritability

In this section, we prove that queries rewritable over locally consistent transformations are guaranteed to satisfy a strong locality criterion on their own, and use these results to show that certain queries are not rewritable over the canonical universal solution or over the core.

Definition 4.4. **(Locally source-dependent queries)** *Given a data exchange setting* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *and a query* Q *over* $\mathbf{T}$*, we say that* Q *is* locally source-dependent *if there is* $d \geq 0$ *such that for every instance* I *of* $\mathbf{S}$ *and for every* $\bar{a}, \bar{b} \in dom(I)^m$*, if* $N_d^I(\bar{a}) \cong N_d^I(\bar{b})$ *then*

$$\big(\bar{a} \in \underline{certain}(Q, I) \quad \Leftrightarrow \quad \bar{b} \in \underline{certain}(Q, I)\big).$$

We next show that this notion applies to all queries rewritable over locally consistent transformations.

Theorem 4.5. *Let* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *be a data exchange setting, and* Q *a query over* $\mathbf{T}$*. Assume that* Q *is rewritable over a transformation* $\mathcal{F}$*, where* $\mathcal{F}$ *is either locally consistent, or locally consistent under FO-equivalence. Then* Q *is locally source-dependent.*

Proof. Let Q' be a first-order rewriting of Q over $\mathcal{F}$, that is, an m-ary FO query over $\mathbf{T}$ such that for every instance I of $\mathbf{S}$, we have $\underline{certain}(Q, I) = Q'(\mathcal{F}(I))$. Assume that $\mathcal{F}$ is locally consistent (the proof for transformations that are locally consistent under FO-equivalence is similar). By Gaifman's theorem [12], there exists a constant r such that for every instance J of $\mathbf{T}$ and m-tuples $\bar{a}, \bar{b}$ in J, if $N_r^J(\bar{a}) \cong N_r^J(\bar{b})$, then $\bar{a} \in Q'(J)$ if and only if $\bar{b} \in Q'(J)$. Given that $\mathcal{F}$ is locally consistent, there exists $d \geq 0$ such that for every instance I of $\mathbf{S}$ and m-tuples $\bar{a}, \bar{b}$ in I, if $N_d^I(\bar{a}) \cong N_d^I(\bar{b})$, then

1. $\bar{a} \in \operatorname{dom}(\mathcal{F}(I))$ iff $\bar{b} \in \operatorname{dom}(\mathcal{F}(I))$, and
2. $N_r^{\mathcal{F}(I)}(\bar{a}) \cong N_r^{\mathcal{F}(I)}(\bar{b})$.

From this we conclude that Q is locally source-dependent since for every instance I of $\mathbf{S}$ and m-tuples $\bar{a}, \bar{b}$ in I,

$$\begin{aligned} & N_d^I(\bar{a}) \cong N_d^I(\bar{b}) \\ \Rightarrow \quad & N_r^{\mathcal{F}(I)}(\bar{a}) \cong N_r^{\mathcal{F}(I)}(\bar{b}) \\ \Rightarrow \quad & \bar{a} \in Q'(\mathcal{F}(I)) \text{ iff } \bar{b} \in Q'(\mathcal{F}(I)) \\ \Rightarrow \quad & \bar{a} \in \underline{certain}(Q, I) \text{ iff } \bar{b} \in \underline{certain}(Q, I). \end{aligned}$$

$\square$

Corollary 4.6. *In a data exchange setting, a target query rewritable over the canonical universal solution or over the core is locally source-dependent.* $\square$

We now show how this result can be used as a simple tool for proving non-rewritability results, even in very simple data exchange settings. We call a data exchange setting *copying* if $\mathbf{S}$ and $\mathbf{T}$ are two copies of the same schema (that is, $\mathbf{S} = \langle R_1, \ldots, R_l \rangle, \mathbf{T} = \langle R_1', \ldots, R_l' \rangle$, and R_i and R_i' have the same arity), and $\Sigma_{st} = \{R_i(\bar{x}) \to R_i'(\bar{x}) \mid i = 1, \ldots, l\}$. Note that a copying setting is both LAV and GAV.

Theorem 4.7. *There is a copying data exchange setting and an FO-query that is not rewritable over the canonical universal solution, nor over the core.*

Proof. Let $\mathbf{S} = \langle G(\cdot,\cdot), R(\cdot) \rangle, \mathbf{T} = \langle G'(\cdot,\cdot), R'(\cdot) \rangle$ and $\Sigma_{st} = \{G(x, y) \to G'(x, y),\ R(x) \to R'(x)\}$. Define a query $Q(x)$ over the target schema as:

$$R'(x) \ \vee \ \exists y \exists z (R'(y) \wedge G'(y, z) \wedge \neg R'(z)).$$

Assume that Q is FO-rewritable over $\mathcal{F}_{\text{univ}}$ or $\mathcal{F}_{\text{core}}$. Then it is locally source-dependent: there exists $d \geq 0$ such that for every source instance I and every $a, b \in \operatorname{dom}(I)$, we have $a \in \underline{certain}(Q, I)$ iff $b \in \underline{certain}(Q, I)$ whenever $N_d^I(a) \cong N_d^I(b)$.

Define a source instance I as shown in Figure 1: $I(G)$ is the disjoint union of two cycles of length $2d + 2$, and $I(R) = \{c\}$. Then $N_d^I(a) \cong N_d^I(b)$, which implies that that $a \in \underline{certain}(Q, I)$ iff $b \in \underline{certain}(Q, I)$.

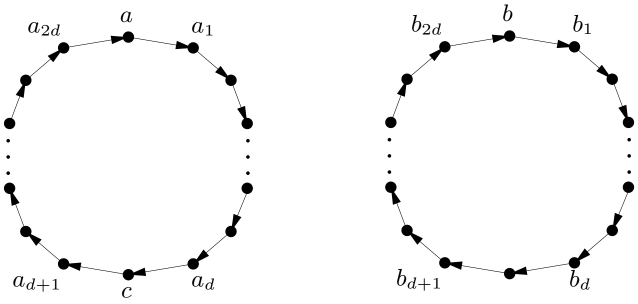


Figure 1: Instance I of Theorem 4.7.

However, it is easy to see that $a \in \underline{certain}(Q, I)$ and $b \notin \underline{certain}(Q, I)$. Indeed, if J is an arbitrary solution for I, then $J \models R'(a) \vee \exists y \exists z (R'(y) \wedge G'(y, z) \wedge \neg R'(z))$ (if J does not satisfy the second disjunct, then $J \models \forall y \forall z (R'(y) \wedge G'(y, z) \rightarrow R'(z))$ and, hence, $J \models R'(a)$ since $R'(c)$ is true in every solution, and a and c are on the same cycle). Furthermore, if J_0 is a target instance such that $J_0(G') = I(G)$ and $J_0(R')$ includes exactly all the points in the cycle containing a, then J_0 is a solution for I. However, $J_0 \not\models Q(b)$, and thus $b \notin \underline{certain}(Q, I)$. This contradiction shows that Q is not rewritable. $\square$

Rewritability over the source Another type of rewriting considered in the literature is rewriting over the source: that is, certain answers to a target query are obtained by applying a rewriting of the query to the source instance. This type of rewriting is common in data integration (e.g., see [5]).

Formally, given a data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and an m-ary query Q over $\mathbf{T}$, we say that Q is *rewritable over the source* if there exists an m-ary FO query Q' over $\mathbf{S}$ such that $\underline{certain}(Q, I) = Q'(I)$ for every instance I of $\mathbf{S}$.

The following corollary is obtained directly from the proof of Theorem 4.5.

Corollary 4.8. *In a data exchange setting, a target query Q rewritable over the source is locally source-dependent.*

Thus, we can also use local source-dependency as a simple tool for proving non-rewritability results over the source.

4.3 Rewritability over the core

We now establish the connection between rewritability over the core and rewritability over canonical universal solution: we show that the former implies the latter.

Theorem 4.9. *Given a data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$, every query Q over the target schema that is rewritable over the core is also rewritable over the canonical universal solution. Moreover, there is a polynomial-time algorithm that, given a rewriting of Q over the core, finds a rewriting of Q over the canonical universal solution.* $\square$

The local consistency of $\mathcal{F}_{\text{core}}$ under FO equivalence, stated in Theorem 3.11, actually follows from several lemmas developed in the proof of this theorem.

The next proposition says that the converse of Theorem 4.9 does not hold.

Proposition 4.10. *There exists an FO query that is rewritable over the canonical universal solution, but not rewritable over the core.* $\square$

5. Extensions

Most results of the previous two sections can be extended in two ways. First, as the underlying language for both data exchange programs and query rewritability one can use an extension of FO with grouping and aggregation, corresponding to basic features of SQL `select-from-where-groupby-having` statements. Second, we show that many results extend for an alternative semantics [10] of queries over the target schema.

5.1 Extended data exchange setting

So far, both data exchange settings and data exchange programs were based on first-order formulae: that is, all stds were of the form $\varphi_{\mathbf{S}}(\bar{x}) \rightarrow \exists \bar{y}\, \psi_{\mathbf{T}}(\bar{x}, \bar{y})$, where $\varphi_{\mathbf{S}}(\bar{x})$ is an FO formula, and all formulae in the bodies of rules were FO as well.

We now show how to extend our main results to the setting where these formulae correspond not to relational calculus but to its extension with grouping and aggregates. Such languages are typically defined as an extension of relational algebra (see [21, 22]), but here instead we adopt the logic approach of [18].

Based on the approach of [13, 18], we define an *aggregate operator* to be a sequence $\mathcal{G} = \langle g_0, g_1, g_2, \ldots, g_\omega \rangle$ of functions, where each g_n, for $0 < n < \omega$, takes an n-element bag of rational numbers, and returns a number in $\mathbb{Q}$. The values g_0 and g_ω are constants associated with the output of $\mathcal{G}$ on the empty bag and on infinite bags, respectively (the latter may occur in the definition of the semantics of terms in the logic).

The *aggregate logic* $\mathrm{FO}_{\mathrm{aggr}}$ over schema $\mathbf{R}$ is two-sorted: first-sort variables range over domains on instances of $\mathbf{R}$, and second-sort variables range over $\mathbb{Q}$. It extends FO by

- *numerical terms and predicates*: for *every* function $f : \mathbb{Q}^n \to \mathbb{Q}$ and every predicate $P \subseteq \mathbb{Q}^n$, if $t_1(\bar{x}), \ldots, t_n(\bar{x})$ are terms of the second (numerical) sort, then so is $f(t_1(\bar{x}), \ldots, t_n(\bar{x}))$; furthermore, $P(t_1(\bar{x}), \ldots, t_n(\bar{x}))$ is an atomic formula. These have the standard semantics.
- *aggregate terms*: for *every* aggregate operator $\mathcal{G}$, a second-sort term $t(\bar{x}, \bar{y})$ and a formula $\varphi(\bar{x}, \bar{y})$, we have a new second-sort term

$$t'(\bar{x}) \;\; = \;\; \mathsf{Aggr}_{\mathcal{G}}\; \bar{y}\; \big(t(\bar{x}, \bar{y}), \varphi(\bar{x}, \bar{y})\big).$$

The semantics $t'(\bar{a})$ is defined as follows. If there are infinitely many $\bar{b}$ such that $\varphi(\bar{a}, \bar{b})$ holds, then the value of $t'(\bar{a})$ is g_ω. Otherwise, let $\bar{b}_1, \ldots, \bar{b}_m$ enumerate all the $\bar{b}$ such that $\varphi(\bar{a}, \bar{b})$ holds. Then $t'(\bar{a})$ is defined as g_m applied to the bag $\{\!\{t(\bar{a}, \bar{b}_1), \ldots, t(\bar{a}, \bar{b}_m)\}\!\}$.

Example 5.1. Let R be a ternary relation whose tuples are (d, e, s), where d is the department name, e is the employee name, and s is the salary. The query that computes the total salary for each department is given by the following $\mathrm{FO}_{\mathrm{aggr}}$ formula $\varphi(d, v)$:

$$(\exists e \exists s R(d, e, s)) \wedge (v = \mathsf{Aggr}_{\mathcal{G}_{\mathtt{SUM}}}(e, s)(s, R(d, e, s))),$$

where $\mathcal{G}_{\mathtt{SUM}}$ is the sequence $\langle g_0, g_1, g_2 \ldots, g_\omega \rangle$ such that $g_0 = 0$ and $g_n(\{\!\{a_1, \ldots, a_n\}\!\}) = a_1 + \cdots + a_n$ for positive integers n. (The value of g_ω could be arbitrary if we are interested only in values of aggregates terms on finite sets.) $\square$

We define an $\mathrm{FO}_{\mathrm{aggr}}$-*data exchange setting* to be a data exchange setting in which every std is of the form $\varphi_{\mathbf{S}}(\bar{x}) \to \exists \bar{y}\, \psi_{\mathbf{T}}(\bar{x}, \bar{y})$, where $\varphi_{\mathbf{S}}(\bar{x})$ is an $\mathrm{FO}_{\mathrm{aggr}}$ formula with all free variables of the first sort. Likewise, we define an $\mathrm{FO}_{\mathrm{aggr}}$-data exchange program as one in which all formulae in the bodies of rules are $\mathrm{FO}_{\mathrm{aggr}}$ formulae with all free variables of the first sort. Just as in the case of FO, we define the canonical universal solution of an $\mathrm{FO}_{\mathrm{aggr}}$-data exchange setting as the result of an $\mathrm{FO}_{\mathrm{aggr}}$ data exchange program obtained by converting each std $\varphi_{\mathbf{S}}(\bar{x}) \to \exists \bar{y}\, (R_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge R_k(\bar{x}_k, \bar{y}_k))$ into a rule $R_1(\bar{x}_1, \bar{y}_1), \ldots, R_k(\bar{x}_k, \bar{y}_k) \coloneq \varphi_{\mathbf{S}}(\bar{x})$.

Theorem 5.2. *Let* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *be an* $\mathrm{FO}_{\mathrm{aggr}}$*-data exchange setting. Every query over* $\mathbf{T}$ *that is* $\mathrm{FO}_{\mathrm{aggr}}$*-rewritable over the canonical universal solution, or over the core, is locally source-dependent.* $\square$

The proof is based on a modified version of local consistency, in which we use equivalence with respect to a certain counting extension of FO [17, 25].

Since every standard data exchange setting is also an $\mathrm{FO}_{\mathrm{aggr}}$-data exchange setting, we can derive a result stronger than Corollary 4.6.

Corollary 5.3. *In a standard data exchange setting, a target query* $\mathrm{FO}_{\mathrm{aggr}}$*-rewritable over the canonical universal solution or over the core is locally source-dependent.* $\square$

5.2 Universal solutions semantics

We wish to begin by exhibiting counterintuitive behavior of the certain answer semantics in the case of Boolean queries. We first give a clarification of the semantics in this case. Let Q be a Boolean (that is, 0-ary) query over the target schema $\mathbf{T}$ and I a source instance. If we let $\underline{\text{true}}$ denote the set with one 0-ary tuple and $\underline{\text{false}}$ denote the empty set, then $Q(J) = \underline{\text{true}}$ and $Q(J) = \underline{\text{false}}$ each have their usual meanings for Boolean queries Q. Note that $\underline{certain}(Q, I) = \underline{\text{true}}$ means that for every solution J of this instance of the data exchange problem, we have that $Q(J) = \underline{\text{true}}$; moreover, $\underline{certain}(Q, I) = \underline{\text{false}}$ means that there is a solution J such that $Q(J) = \underline{\text{false}}$.

As the next proposition shows, the usual certain answers semantics sometimes exhibits rather counterintuitive behavior.

Proposition 5.4. *Let* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *be a data exchange setting. Then for every Boolean query* Q *over* $\mathbf{T}$*, either* $\underline{certain}(Q, I) = \underline{false}$ *for all instances* I *of* $\mathbf{S}$*, or* $\underline{certain}(\neg Q, I) = \underline{false}$ *for all instances* I *of* $\mathbf{S}$.

Proof: Let Q be a Boolean query over $\mathbf{T}$, and assume that there exists an instance I_0 of $\mathbf{S}$ such that $\underline{certain}(Q, I_0) = \underline{\text{true}}$. Then we show that $\underline{certain}(\neg Q, I) = \underline{\text{false}}$ for every instance I of $\mathbf{S}$.

Let I be an instance of $\mathbf{S}$ and J a solution for I. Then given a solution J_0 for I_0, the instance J' defined as $J'(R) = J(R) \cup J_0(R)$, for every $R \in \mathbf{T}$, is a solution for both I and I_0. Since $\underline{certain}(Q, I_0) = \underline{\text{true}}$, it follows that $Q(J')$ is true, and so there is a solution of I not satisfying $\neg Q$. We conclude that $\underline{certain}(\neg Q, I) = \underline{\text{false}}$. $\square$

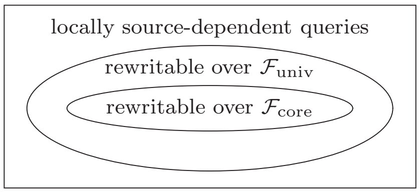
That is, contrary to the expectation that for some instances the result of a query is true and for others it is

Transformation	LAV Setting		GAV Setting		General Setting	
	locally consistent	locally consistent under $\equiv$	locally consistent	locally consistent under $\equiv$	locally consistent	locally consistent under $\equiv$
canonical universal solution $\mathcal{F}_{\text{univ}}$	yes	yes	no	yes	no	yes
core $\mathcal{F}_{\text{core}}$	yes	yes	no	yes	no	yes

Summary of local consistency results

Rewritable over	Under semantics	in logic	
		FO	FO_{aggr}
$\mathcal{F}_{\text{univ}}$	usual	yes	yes
	universal solution	yes	yes
$\mathcal{F}_{\text{core}}$	usual	yes	yes
	universal solution	yes	yes

Is a query locally-source-dependent?



Summary of rewritability results

Figure 2: Summary of the main results

false, in the case of certain answers semantics, for one of Q or $\neg Q$ the result is false in all instances!

This anomaly suggests that we consider a different semantics. It was argued in [10] that since the universal solutions are the preferred solutions in data exchange, it may be more meaningful to consider semantics based on them.

Given a data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$, an m-ary query Q over $\mathbf{T}$, and a source instance I, we define the *universal solutions semantics* of Q as

$$\underline{\text{u-certain}}(Q, I) \;=\; \bigcap_{J \text{ is a universal solution for } I} Q(J).$$

Clearly, $\underline{\text{certain}}(Q, I) \subseteq \underline{\text{u-certain}}(Q, I)$.

The next example shows that the universal solution semantics avoids the problem shown in Proposition 5.4. Specifically, we show that there exists a Boolean query Q and instances I_1 and I_2 such that $\underline{\text{u-certain}}(Q, I_1) = \underline{\text{true}}$ and $\underline{\text{u-certain}}(\neg Q, I_2) = \underline{\text{true}}$.

Example 5.5. Given a copying data exchange setting with $\mathbf{S} = \langle P(\cdot), R(\cdot)\rangle$, $\mathbf{T} = \langle P'(\cdot), R'(\cdot)\rangle$ and $\Sigma_{st} = \{P(x) \rightarrow P'(x),\ R(x) \rightarrow R'(x)\}$, let Q be a Boolean query over $\mathbf{T}$ defined as $\exists x\,(P'(x) \wedge R'(x))$. Define instances I_1, I_2 of $\mathbf{S}$ as $\{P(a), R(a)\}$ and $\{P(a), R(b)\}$, respectively. Then both $\underline{\text{u-certain}}(Q, I_1)$ and $\underline{\text{u-certain}}(\neg Q, I_2)$ are $\underline{\text{true}}$ (if J is a universal solution for I_2, then there is a homomorphism $h : J \rightarrow \mathcal{F}_{\text{univ}}(I_2) = \{P'(a), R'(b)\}$ and, hence, for every null value c in J it could not be the case that $P'(c)$ and $R'(c)$ are in J).

The universal solutions semantics has other appealing properties. For example, from [10] we know that every existential query (which includes every union of conjunctive queries with inequalities $\neq$) is FO-rewritable over $\mathcal{F}_{\text{core}}$, under the universal solutions semantics. This is not the case for the usual semantics, even when restricted to conjunctive queries with only one inequality [9].

We now show that the main results of the paper are preserved when one considers this new semantics of answering queries over the target.

Given a mapping $\mathcal{F} : \text{inst}(\mathbf{S}) \rightarrow \text{inst}(\mathbf{T})$, we say that Q is FO_{aggr}*-rewritable over* $\mathcal{F}$ *under the universal solutions semantics* if there exists an m-ary FO_{aggr}-query Q' over $\mathbf{T}$ such that $\underline{\text{u-certain}}(Q, I) = Q'(\mathcal{F}(I))$ for every instance I of $\mathbf{S}$.

We say that a query Q over $\mathbf{T}$ is *locally source-dependent under the universal solutions semantics* if there is $d \geq 0$ such that for every instance I of $\mathbf{S}$ and every $\bar{a}, \bar{b} \in \text{dom}(I)^m$, whenever $N_d^I(\bar{a}) \cong N_d^I(\bar{b})$ then

$$\big(\bar{a} \in \underline{\text{u-certain}}(Q, I) \;\;\Leftrightarrow\;\; \bar{b} \in \underline{\text{u-certain}}(Q, I)\big).$$

The next theorem says that Theorem 5.2 extends to the universal solutions semantics.

Theorem 5.6. *Let* $(\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *be an* FO_{aggr}*-data exchange setting. Every query over* $\mathbf{T}$ *that is* FO_{aggr}*-rewritable over the canonical universal solution, or over the core, under the universal solutions semantics, is locally source-dependent under the universal solutions semantics.* □

Thus, Theorem 5.6 can be used a tool for proving non-rewritability under the new semantics.

We conclude with a result that shows the incomparability of rewritability under the usual semantics and the universal solutions semantics.

Proposition 5.7. *Let $\mathcal{F}$ be either $\mathcal{F}_{\mathrm{univ}}$ or $\mathcal{F}_{\mathrm{core}}$.*

1) *There is an FO-query Q that is rewritable (even in FO) over $\mathcal{F}$ under the usual semantics, but is not $\mathrm{FO}_{\mathrm{aggr}}$-rewritable over $\mathcal{F}$ under the universal solutions semantics.*

2) *There is an FO-query Q that is rewritable (even in FO) over $\mathcal{F}$ under the universal solutions semantics, but is not $\mathrm{FO}_{\mathrm{aggr}}$-rewritable over $\mathcal{F}$ under the usual semantics.*

6. Conclusions

Figure 2 summarizes the main results of the paper. The first table shows when the canonical universal solution and core transformations $\mathcal{F}_{\mathrm{univ}}$ and $\mathcal{F}_{\mathrm{core}}$ are locally consistent ("under $\equiv$" means "under FO equivalence", but instead of FO one can use $\mathrm{FO}_{\mathrm{aggr}}$ as well). The second table gives four classes of locally source-dependent queries, based on the logic and transformation they are rewritable over. The final picture shows the relationship between different classes of rewritable queries. Unlike isolated results on rewriting that exist in the literature, our results give easily applicable tools for studying these notions.

In the future, we would like to develop tools for studying data exchange transformation and query rewriting in the presence of target dependencies, and to extend techniques from relational databases to other data formats.

Acknowledgments We thank Michael Benedikt and Phokion Kolaitis for their comments. M. Arenas, P. Barceló, and L. Libkin have been supported by grants from NSERC, PREA, and CITO.

7. References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *PODS 1998*, pages 254–263.

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

[3] S. Abiteboul and P. Kanellakis. Object identity as a query language primitive. *Journal of the ACM* 45(5), pages 798–842, 1998.

[4] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM* 31(4), pages 718–741, 1984.

[5] O. Duschka and A. Levy. Recursive plans for information gathering. In *IJCAI 1997*, pages 778–784.

[6] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer Verlag, 1995.

[7] R. Fagin. Horn clauses and database dependencies. *Journal of the ACM* 29(4), pages 952–985, 1982.

[8] R. Fagin. Probabilities on finite models. *Journal of Symbolic Logic* 41(1), pages 50–58, 1976.

[9] R. Fagin, Ph. Kolaitis, R. Miller and L. Popa. Data exchange: semantics and query answering. In *ICDT 2003*, pages 207–224.

[10] R. Fagin, Ph. Kolaitis and L. Popa. Data exchange: getting to the core. In *PODS 2003*, pages 90–101.

[11] R. Fagin, L. Stockmeyer and M. Vardi. On monadic NP vs monadic co-NP. *Information and Computation* 120(1), pages 78–92, 1995.

[12] H. Gaifman. On local and non-local properties. In *Proceedings of the Herbrand Symposium, Logic Colloquium '81*, North Holland, 1982.

[13] E. Grädel and Y. Gurevich. Metafinite model theory. *Information and Computation* 140(1), pages 26–81, 1998.

[14] A. Halevy. Theory of answering queries using views. *SIGMOD Record* 29(4), pages 40–47, 2000.

[15] W. Hanf. Model-theoretic methods in the study of elementary logic. In J.W. Addison et al, eds, *The Theory of Models*, North Holland, pages 132–145, 1965.

[16] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics* 109, pages 117–126, 1992.

[17] L. Hella. Logical hierarchies in PTIME. *Information and Computation* 129(1), pages 1–19, 1996.

[18] L. Hella, L. Libkin, J. Nurmonen and L. Wong. Logics with aggregate operators. *Journal of the ACM* 48(4), pages 880–907, 2001.

[19] R. Hull and M. Yoshikawa. ILOG: declarative creation and manipulation of object identifiers. In *VLDB 1990*, pages 455–468.

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

[21] A. Klug. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *Journal of the ACM* 29(3), pages 699–717, 1982.

[22] K. Larsen. On grouping in relational algebra. *International Journal of Foundations of Computer Science* 10(3), pages 301–311, 1999.

[23] M. Lenzerini. Data integration: a theoretical perspective. In *PODS 2002*, pages 233–246.

[24] A. Levy, A. Mendelzon, Y. Sagiv and D. Srivastava. Answering queries using views. In *PODS 1995*, pages 95–104.

[25] L. Libkin. Logics with counting and local properties. *ACM Transactions on Computational Logic* 1(1), pages 33–59, 2000.

[26] D. Maier, A. O. Mendelzon and Y. Sagiv. Testing implications of data dependencies. *ACM Transactions on Database Systems* 4(4), pages 455-469, 1979.

[27] N. Shu, B. Housel, R. Taylor, S. Ghosh and V. Lum. EXPRESS: a data extraction, processing, and restructuring system. *ACM Transactions on Database Systems* 2(2), pages 134–174, 1977.

[28] J. Van den Bussche, D. Van Gucht, M. Andries and M. Gyssens. On the completeness of object-creating database transformation languages. *Journal of the ACM* 44(2), pages 272–319, 1997.