Balancing expressiveness and inexpressiveness in view design: full version

Michael Benedikt and Pierre Bourhis and Louis Jachiet and Efthymia Tsamoura

Abstract

We study the design of data publishing mechanisms that allow a collection of autonomous distributed datasources to collaborate to support queries. A common mechanism for data publishing is via *views*: functions that expose derived data to users, usually specified as declarative queries. Our autonomy assumption is that the views must be on individual sources, but with the intention of supporting integrated queries. In deciding what data to expose to users, two considerations must be balanced. The views must be sufficiently expressive to support queries that users want to ask – the *utility* of the publishing mechanism. But there may also be some expressiveness restriction. Here we consider two restrictions, a *minimal information* requirement, saying that the views should reveal as little as possible while supporting the utility query, and a *non-disclosure* requirement, formalizing the need to prevent external users from computing information that data owners do not want revealed. We investigate the problem of designing views that satisfy both an expressiveness and an inexpressiveness requirement, for views in a restricted declarative language (conjunctive queries), and for arbitrary views.

1 Introduction

The value of data is increased when data owners make their data available through publicly-accessible interfaces. The value is magnified even further when multiple data owners publish information from related datasets; this allows users to answer queries that require linking information across datasources.

But the benefits of data publishing come with a corresponding risk of revealing too much. For example, here may be information that a data owner wishes to protect, and a user may be able to infer this information either from the published data in isolation, or from the data published by all parties as a whole. There is thus a need to provide data publishing mechanisms that are simultaneously expressive enough to be useful – they enable users to answer appropriate queries – while satisfying some expressiveness restriction.

In data publishing much of the focus has been on disclosure via the familiar mechanism of *views* – declarative queries whose output is made available to users as a table. In this context the competing requirements on a publishing mechanism have been primarily studied in isolation. There is extensive work on analysis of the utility of a set of views: namely given a query, can it be answered using the views, see, e.g. (Halevy 2001; Calvanese et al. 2012). There has also been research concerning analysis of whether a given set of views obeys some expressiveness *restriction*. The negation of answerability is clearly too weak a restriction, since it just guarantees that on some instance the query answer can not be computed from the view images. A relevant query-based notion of expressiveness restriction is *data-independent privacy*: given the views and a set of "secret" queries, check that the secret query answers can not be computed from the views on any source data. Variants of this problem have been studied in (Nash and Deutsch 2007; Benedikt et al. 2016; Benedikt, Cuenca Grau, and Kostylev 2018; Benedikt et al. 2019). Expressiveness restrictions with a similar flavor have also been studied in the context of ontologies (Bonatti and Sauro 2013). But the question of whether there is a query-independent expressiveness restriction appropriate for views based on traditional database queries, as well as the question of how one obtains views that satisfy both expressiveness and inexpressiveness requirements, has not been considered in the context of traditional queries and views, to the best of our knowledge.

In the multi-agent systems community there is work concerned with protocols allowing a set of agents to learn a distribution on source data without revealing certain local information (e.g. (Fernández-Duque and Goranko 2016)). The notion of security is "exact and information-theoretic", as in this work: they deal with getting the exact answer to a query using a deterministic algorithm, but without restricting the computational power used in answering. But the setting, unlike in this work, is propositional.

A larger body of work comes from privacy research, considering the design of mechanisms achieving a mix of expressiveness ("utility") and inexpressiveness ("privacy") goals. But the focus is on probabilistic transformations or, more generally, probabilistic protocols (see e.g. (Chaum, Crépeau, and Damgard 1988; Dwork 2006; Dwork and Roth 2014)). The guarantees are probabilistic, sometimes alternatively or additionally with computational restrictions on an external party.

In contrast, we consider the question of designing views that use *traditional database queries*, with no randomization, so that conflicting requirements of expressiveness and inexpressiveness are satisfied. Both our expressiveness and inexpressiveness requirements will be given in terms of *exact information-theoretic criteria*: they will be defined in terms of what queries can be answered exactly (as opposed to probabilistically) by a party with unlimited computation power. Due both to the difference in the mechanisms we consider and the requirements we impose, our contribution has a very different flavor from prior lines of work.

Example 1. *A health study hosted by a government agency holds information about certain treatments, with an abstraction of the data being a database with schema* $\mathsf{Trtmnt}(\mathsf{pid}, \mathsf{tinfo})$*, where* pid *might be a national insurance number.*

Demographic information about patients is stored by another agency, in a table $\mathsf{Patient}(\mathsf{pid}, \mathsf{age}, \mathsf{address})$. *The agencies are completely autonomous, perhaps even in distinct administrative regions. But they want to cooperate to support certain queries over the data that legitimate researchers might wish; for example about the relationships between treatment and age:*

$$Q = \exists \mathsf{pid}, \mathsf{address}, \mathsf{tdate}.\mathsf{Trtmnt}(\mathsf{pid}, \mathsf{tinfo}, \mathsf{tdate}) \\ \wedge \mathsf{Patient}(\mathsf{pid}, \mathsf{age}, \mathsf{address})$$

Of course, the two parties could agree to an encryption schema on the patient identifiers, and then expose encrypted versions of their local schemas. But this would require both strong co-operation of the parties, and the use of views beyond traditional database queries.

But assuming that the parties are restricted to using traditional queries, what is the most restrictive thing that they can do while supporting the ability to answer Q? Intuitively, the most restrictive views would correspond to one party revealing the projection of the Trtmnt *table on* pid *and* tinfo *and the other revealing the projection of the* Patient *table.*

If this intuition is correct, it would imply that nothing the parties can do with traditional queries can avoid revealing which patients had particular treatments. That is, they have no choice but to allow an external party to learn the answer to query

$$p = \exists \mathsf{tdate}.\mathsf{Trtmnt}(\mathsf{pid}, \mathsf{tinfo}, \mathsf{tdate})$$

Our results will validate this obvious answer – in this example, the projections described above are the CQ views that reveal minimal information, and one can not support the disclosure of the join query while protecting a query on these join attributes. Further we will show that even using encryption – indeed, even using any deterministic function – the parties can not not reveal less information while supporting exact answering of the query Q. In contrast, we will also show that in some cases the parties can obtain combinations of expressiveness and inexpressiveness requirements by using counter-intuitive view combinations.

Our goal here is to look at the problem of designing independent views over multiple relational datasources that satisfy both expressiveness and inexpressiveness requirements. Our expressiveness requirement (usefulness) will be phrased in terms of the ability to answer a relational query, where answering is the traditional deterministic notion used in database theory and KR. For expressiveness limitations we require the views to be useful but to *minimize information* within a class of views. We also consider an expressiveness limitation specified by *non-disclosure* of a set of *secret queries*. Our contributions include formalizing these notions, characterizing when minimal information views exist and what form they take, and determining when views exist that satisfy utility and expressiveness limitations. We look at these problems both for views given in the standard view language of conjunctive queries, and for arbitrary views. We also consider the impact of background knowledge on these problems.

Organization. Section 2 gives database and logic preliminaries, and then goes on to formalize our expressiveness and inexpressiveness restrictions. Section 3 deals with the variant of the problem where the only views considered are conjunctive query views, while Section 4 shows how the situation changes when arbitrary views can be utilized. Section 5 gives some extensions to the situation when background knowledge about the sources is present. We close with conclusions in Section 6. In this extended version we include details of most proofs in the appendix.

2 Preliminaries

2.1 Basic definitions

The bulk of this subsection reviews standard definitions from databases and KR. But it includes two notions, DCQs and distributed schemas, that are less standard.

Databases and queries. A *schema* consists of a finite set of relations, each with an associated arity (a non-negative number). An *instance* of a schema is an assignment of each relation in the schema of arity n to a collection (finite or infinite) of n-tuples of elements. Given an instance $\mathcal{I}$ and a relation R, we let $\mathcal{I}(R)$ be the n-tuples assigned to R in $\mathcal{I}$. The *active domain* of an instance $\mathcal{I}$, denoted $\mathsf{adom}(\mathcal{I})$, is the set of elements occurring in some tuple of some $\mathcal{I}(R)$. A *fact* consists of a relation R of arity n and an n-tuple $\mathbf{t}$. We write such a fact as $R(\mathbf{t})$. An instance can equivalently be thought of as a set of facts.

An n-ary *query* is a function from instances of a fixed schema $\mathcal{S}$ to some set. We refer to $\mathcal{S}$ as the *input schema* of the query. A *Conjunctive Query* (CQ) is a formula of the form $\exists \mathbf{y} \bigwedge_i A_i$ where A_i are atoms from the schema. A Boolean CQ (BCQ) is a CQ with no free variables. Following the notation used in several places in the literature (e.g. (Arenas, Barceló, and Reutter 2011)) we define a *union of CQs* (UCQ) to be a disjunction of CQs satisfying the *safety condition* where the free variables of each disjunct are the same. Disjunctions of CQs where the safety condition is dropped will play a key role in this paper. The terminology for such queries is less standardized, but we refer to them as *disjunctions of CQs* (DCQs). Relational algebra is the standard algebraic presentation of first-order relational queries: queries are build up from relations by union, difference, selection, and product. (Abiteboul, Hull, and Vianu 1995).

For a logical formula ρ with free variables $x_1, \ldots, x_n$ and an instance $\mathcal{I}$, a *variable binding* σ for ρ in $\mathcal{I}$ is a mapping taking each x_i to an element of $\mathsf{adom}(\mathcal{I})$. We can apply σ to ρ to get a new formula $\sigma(\rho)$ where each x is replaced by $\sigma(x)$. Assuming an ordering of the free variables as $x_1, \ldots, x_n$ we may identify a k-tuple $\mathbf{t}$, writing $\rho(\mathbf{t})$ to mean that t_i is substituted for x_i in ρ.

A *homomorphism* between instances $\mathcal{I}_1$ and $\mathcal{I}_2$ is a function f from $\mathsf{adom}(\mathcal{I}_1)$ to $\mathsf{adom}(\mathcal{I}_2)$ such that $R(c_1, \ldots, c_m) \in \mathcal{I}_1$ implies $R(f(c_1), \ldots, f(c_m)) \in \mathcal{I}_2$. The notion of homomorphism from a CQ to an instance and from a CQ to a CQ is defined similarly. In the case of CQ-to-CQ homomorphisms we additionally require that the mapping be the identity on any free variables or constants. The *output* of a CQ Q on an instance $\mathcal{I}$, denoted $Q(\mathcal{I})$ consists of the restrictions to free variables of Q of the homomorphisms of Q to $\mathcal{I}$. The output of a UCQ is defined similarly. We can choose an ordering of the free variables of Q, and can then say that the output of Q on $\mathcal{I}$ consists of n-tuples. We write $\mathcal{I}, \mathbf{t} \models Q$ for an n-tuple $\mathbf{t}$, if $\mathbf{t} \in Q(\mathcal{I})$. We analogously define $\mathcal{I}, \sigma \models Q$ for a variable binding σ. We sometimes refer to a homomorphism of a BCQ into an instance as a *match*. For a logical formula $\rho(\mathbf{x})$ and a tuple of elements $\mathbf{t}$, $\rho(\mathbf{t})$ denotes the formula where each x_i is substituted with t_i.

A CQ Q_0 is a *subquery* of a CQ Q if the atoms of Q_0 are a subset of the atoms of Q and a variable of Q_0 is free in Q_0 if and only if it is free in Q. A *strict subquery* of Q is a subquery of Q that is not Q itself; Q is *minimal* if there is no homomorphism from Q to a strict subquery of Q.

A *view* over a schema $\mathcal{S}$ consists of an n-ary relation V and a corresponding n-ary query Q_{V} over relations from $\mathcal{S}$. Given a collection of views $\mathcal{V}$ and instance $\mathcal{I}$, the *view-image* of $\mathcal{I}$, denoted $\mathcal{V}(\mathcal{I})$ is the instance that interprets each $\mathsf{V} \in \mathcal{V}$ by $Q_{\mathsf{V}}(\mathcal{I})$. We thus talk about *CQ views*, *UCQ views*, etc: views defined by formulas within a class.

Distributed data and views. A *distributed schema* (d-schema) $\mathcal{S}$ consists of a finite set of *sources* Srcs, with each source s associated with a *local schema* $\mathcal{S}_s$. We assume that the relations in distinct local schemas are pairwise disjoint. In Example 1 our distributed schema consisted of two sources, one containing Trtmnt and the other containing $\mathsf{Patient}$. A *distributed instance* (d-instance) is an instance of a distributed schema. For a source s, an s-instance is an instance of the local schema s. Given a d-instance $\mathcal{D}$, we denote by $\mathcal{D}_s$ the restriction of $\mathcal{D}$ to relations in s. If d-schema $\mathcal{S}$ is the disjoint union of $\mathcal{S}_1$ and $\mathcal{S}_2$, and we have sources $\mathcal{I}_1$ for $\mathcal{S}_1$ and $\mathcal{I}_2$ for $\mathcal{S}_2$, then we use $(\mathcal{I}_1, \mathcal{I}_2)$ to denote the union of $\mathcal{I}_1$ and $\mathcal{I}_2$, which is an instance of $\mathcal{S}$.

For a given d-schema a *distributed view* (d-view) $\mathcal{V}$ is an assignment to each source s of a finite set $\mathcal{V}_s$ of views over its local schema. Note that here is our "autonomy" assumption on the instances: we are free to design views on each local source, but views can not cross sources. We can similary talk about CQ-based d-views, relational algebra-based d-views, etc.

Existential Rules. Semantic relationships between relations can be described using *existential rules*, logical sentences of the form $\forall \mathbf{x}.\lambda \rightarrow \exists \mathbf{y}.\rho$, where λ and ρ are conjunctions of relational atoms. The notion of a formula ρ *holding* in $\mathcal{I}$ (or $\mathcal{I}$ *satisfying* ρ, written $\mathcal{I} \models \rho$) is the standard one in first-order logic.

2.2 Problem formalization

We now give the key definitions in the paper, capturing our expressiveness requirements ("usefulness") and expressiveness limitations ("minimal usefulness' and "non-disclosure").

Definition 1. *Two d-instances $\mathcal{D}$ and $\mathcal{D}'$ are* indistinguishable *by a d-view $\mathcal{V}$ (or just $\mathcal{V}$-indistinguishable) if $V(\mathcal{D}) = V(\mathcal{D}')$ holds, for each view $V \in \mathcal{V}$.*

Since each view $V \in \mathcal{V}_s$ is defined over relations occurring only in $\mathcal{V}_s$, we can equivalently say that $\mathcal{D}$ and $\mathcal{D}'$ are $\mathcal{V}$-indistinguishable if $V(\mathcal{D}_s) = V(\mathcal{D}'_s)$ holds, for each $V \in \mathcal{V}_s$.

Definition 2. *A d-view $\mathcal{V}$* determines *a query Q* at *a d-instance $\mathcal{D}$ if $Q(\mathcal{D}') = Q(\mathcal{D})$ holds, for each $\mathcal{D}'$ that is $\mathcal{V}$-indistinguishable from $\mathcal{D}$.*

The d-view $\mathcal{V}$ is useful *for Q if $\mathcal{V}$ determines Q on every d-instance (for short, just "$\mathcal{V}$ determines Q").*

Usefulness for a given query Q will be our expressiveness requirement on d-views. Our first inexpressiveness requirement captures the idea that we want to reveal as little as possible:

Definition 3 (Minimally useful views)**.** *Given a class of views $\mathcal{C}$ and a query Q, we say that a d-view $\mathcal{V}$ is* minimally useful for Q within $\mathcal{C}$ *if $\mathcal{V}$ is useful for Q and for any other d-view $\mathcal{V}'$ using views in $\mathcal{C}$, $\mathcal{V}'$ determines the view definition of each view in $\mathcal{V}$.*

We look at another inexpressiveness requirement that requires an external party to not learn about another query.

Definition 4 (Non-disclosure)**.** *A* non-disclosure function *specifies, for each query p, the set of d-views $\mathcal{V}$ that are said to disclose p. We require such a function F to be* determinacy-compatible*: if $\mathcal{V}_2$ discloses p according to F and $\mathcal{V}_1$ determines each view in $\mathcal{V}_2$, then $\mathcal{V}_1$ also discloses p according to F. If $\mathcal{V}$ does not disclose p, we say that $\mathcal{V}$ is non-disclosing for p (relative to the given non-disclosure function).*

When we can find minimal useful views, this tells us something about non-disclosure, since it is easy to see that if we are looking to design views that are useful and non-disclosing, it suffices to consider minimal information views, assuming they exist:

Proposition 1. *Suppose d-view $\mathcal{V}$ is minimally useful for Q within $\mathcal{C}$, and there is a d-view based on views in $\mathcal{C}$ that is useful for Q and non-disclosing for p according to non-disclosure function F. Then $\mathcal{V}$ is useful for Q and non-disclosing for p according to F.*

There are many disclosure functions that are determinacy-compatible. But in our examples, our complexity results, and in Subsection 5.3, we will focus on a specific non-disclosure function, whose intuition is that an external party "never infers any answers".

Definition 5. *A d-view $\mathcal{V}$ is* universal non-inference non-disclosing *(UN non-disclosing) for a CQ p if for each instance $\mathcal{I}$ and each tuple $\mathbf{t}$ with $\mathcal{I}, \mathbf{t} \models p$, $\mathcal{V}$ does not determine $p(\mathbf{t})$ at $\mathcal{I}$. Otherwise $\mathcal{V}$ is said to be* UN disclosing *for p.*

This non-disclosure function is clearly determinacy-compatible, so Proposition 1 will apply to it. Thus we will be able to utilize UN non-disclosure as a means of showing that certain d-views are *not* minimally useful.

Variations: negative disclosure, background knowledge, and finite instances. All of these notions can be additionally parameterized by background knowledge Σ, consisting of integrity constraints in some logic. Given d-instance $\mathcal{D}$ satisfying Σ and a d-view $\mathcal{V}$, $\mathcal{V}$ *determines* a query Q over the d-schema at $\mathcal{D}$ relative to Σ if: for every $\mathcal{D}'$ *satisfying* Σ that is $\mathcal{V}$-indistinguishable from $\mathcal{D}$, $Q(\mathcal{D}') = Q(\mathcal{D})$. We say that $\mathcal{V}$ is *useful* for a query Q relative to Σ if it determines Q on every d-instance *satisfying* Σ. We say $\mathcal{V}$ is UN non-disclosing for query p with respect to Σ if $\mathcal{V}$ does not determine p *on any d-instance satisfying* Σ.

By default, when we say "every instance", we mean all instances, finite or infinite. There are variations of this problem requiring the quantification in both non-disclosure and utility to be over finite instances.

Main problem. We focus on the problem of determining whether minimally useful d-views exist for a given query Q and class $\mathcal{C}$, and characterizing such views when they do exist. When minimal useful d-views do not exist, we consider the problem of obtaining a d-view that is useful for Q and which minimizes the set of secrets p that are UN disclosed for p. We refer to Q as the *utility query*, and p as the *secret query*.

Discussion. Our notion of utility of views is *information-theoretic and exact*: a view is useful if a party with access to the view can compute the exact output of the query (as opposed to the correct output with high probability), with no limit on how difficult the computation may be. The generality of this notion will make our negative results stronger. And it turns out the generality will not limit our positive results, since these will be realized by very simply views.

Our notion of minimal usefulness is likewise natural if one seeks an ordering on sets of views measuring the ability to support exact information-theoretic query answering. Our query-based inexpressiveness notion, non-disclosure, gives a way of seeing the impact of minimal useful views on protecting information, and it is also based on information-theoretic and exact notions. We exemplify our notion of non-disclosure function with UN non-disclosure, which has been studied in prior work under several different names (Benedikt et al. 2016; Benedikt, Cuenca Grau, and Kostylev 2018; Benedikt et al. 2019). We choose the name "non-disclosing" rather than "private" for all our query-based expressiveness restrictions, since they are clearly very different from more traditional probabilistic privacy guarantees (Dwork and Roth 2014). On the one hand the UN non-disclosure guarantee is weak in that p is considered safe for $\mathcal{V}$ (UN non-disclosed) if an attacker can never infer that p is true with absolutely certainty. Given a distribution on source instances, the information in the views may still increase the likelihood that p holds. On the other hand, the notion is quite strong in that it must hold on *every* source instance. Thus, although we do not claim that this captures all intuitively desirable properties of privacy, we do feel that it allows us to explore the ability to create views that simultaneously support the strong ability to answer certain queries in data integration and the strong inability to answer other queries.

Restrictions and simplifications. Although the utility and non-disclosure definitions make sense for any queries, *in this paper we will assume that Q and p are BCQs without constants (abusing notation by dropping "without constants")*. While we restrict to the case of a single utility query and secret query here, *all of our results have easy analogs for a finite set of such queries.*

2.3 Some tools

Throughout the paper we rely on two basic tools.

Canonical views. Recall that we are looking for views that are useful for answering a CQ Q over a d-schema. The "obvious" set of views to try are those obtained by partitioning the atoms of Q among sources, with the free variables of the views including the free variables of Q and the variables occurring in atoms from different sources.

Given a CQ Q over a d-schema, and a source s, we denote by $\mathsf{SVars}(s, Q)$ the variables of Q that appear in an atom from source s. We also denote by $\mathsf{SJVars}(s, Q)$, the "source-join variables of s" in Q: the variables that are in $\mathsf{SVars}(s, Q)$ and which also occur in an atom of another source.

Definition 6. *The* canonical view of *Q for source s, $\mathsf{CanView}^s(Q)$, has a view definition formed by conjoining all s-source atoms in Q and then existentially quantifying all bound variables of Q in $\mathsf{SVars}(s, Q) \setminus \mathsf{SJVars}(s, Q)$. The* canonical d-view of *Q is formed by taking the canonical view for each source.*

In Example 1, the canonical d-view is what we referred to as "the obvious view design". It would mean that one source has a view exposing $\exists \mathsf{tdate}\ \mathsf{Trtmnt}(\mathsf{pid}, \mathsf{tinfo})$ – since pid is a source-join variable while tinfo is a free variable of Q. The other source should expose a view revealing $\exists \mathsf{address}.\mathsf{Patient}(\mathsf{pid}, \mathsf{age}, \mathsf{address})$, since address is neither a free variable nor shared across sources.

The critical instance. In the definition of UN non-disclosure of a set of views, we required that on *any* instance of the sources, a user who has access to the views cannot reconstruct the answer to the query p. An instance that will be helpful in several examples is the following "most problematic" instance.

Definition 7. *The* critical instance *of a schema $\mathcal{S}$ is the instance whose active domain consists of a single element $*$ and whose facts are $R(*, \ldots, *)$ for all relational names R in $\mathcal{S}$.*

Note that every BCQ over the relevant relations holds on the critical instance of the source. The critical instance is the hardest instance for UN non-disclosure in the following sense:

Theorem 1. *(Benedikt et al. 2016; Benedikt, Cuenca Grau, and Kostylev 2018) Consider any CQ views $\mathcal{V}$, and any BCQ p. If p is determined by $\mathcal{V}$ at some instance of the source schema then it is determined by $\mathcal{V}$ at the critical instance.*

3 CQ views

Returning to Example 1, recall the intuition that the canonical d-view of Q are the "least informative views" that support the ability to answer Q. We start our analysis by proving such a result, but with two restrictions: Q must be a minimal CQ, and we only consider views specified by CQs:

Theorem 2. *[Minimally useful CQ views] If CQ-based d-view $\mathcal{V}$ determines a minimal Boolean CQ Q, then $\mathcal{V}$ determines each canonical view $\mathsf{CanView}^s(Q)$ of Q.*

We only sketch the proof here. The first step is to show that the determinacy of a CQ Q by a CQ-based d-view $\mathcal{V}$ leads to a certain homomorphism of Q to itself. This step follows using a characterization of determinacy of CQ queries by CQ views via the well-known chase procedure (See (Benedikt, ten Cate, and Tsamoura 2016) or the "Green-Red chase" in (Gogacz and Marcinkowski 2015)). The second step is to argue that if this homomorphism is a bijection, then it implies that the canonical d-view determines Q, while if the homomorphism is not bijective, we get a contradiction of minimality. This step relies on an analysis of the chase characterization of determinacy factorizes over a d-schema.

Consequences. Combining Theorem 2 and Proposition 1 gives a partial answer to the question of how to obtain useful and non-disclosing views:

Corollary 1. *For any non-disclosure function F, BCQs Q and p, if there is a CQ based d-view that is useful for Q and non-disclosing for p according to F, then the canonical d-view of Q^{min} is such a d-view, where Q^{min} is any minimal CQ equivalent to Q.*

If we consider the specific non-disclosure notion, UN non-disclosure, we can infer a complexity bound from combining these results with prior work on the complexity of checking non-disclosure (Benedikt, Cuenca Grau, and Kostylev 2018):

Corollary 2. *There is a Σ_2^p algorithm taking as input BCQs Q and p and determines whether there is a CQ-based d-view that is useful for Q and UN non-disclosing for p. If Q is assumed minimal than the algorithm can be taken to be in* CoNP.

The following example shows that the requirement that Q is minimal in Theorem 2 is essential.

Example 2. *Consider two sources. The first source comprises the relations R_1, R_2 and R_3, while the second source comprises a single relation T. Consider also the conjunctions of atoms C_1 and C_2 defined as:*

$$\begin{aligned}C_1 = {} & R_1(x, y) \wedge R_1(y, z) \wedge R_1(z, x) \wedge \\ & T(x) \wedge R_2(w) \wedge R_3(q) \\ C_2 = {} & R_1(x', y') \wedge R_1(y', z') \wedge R_1(z', x') \wedge \\ & T(x') \wedge R_2(y') \wedge R_3(z')\end{aligned}$$

The conjunct C_1 states that there is R_1-triangle with one vertex in T, and in addition R_2 and R_3 are nonempty. The second conjunct states that there is an R_1-triangle with one vertex in T, a second in R_2, and a third in R_3. Consider now the query Q defined as

$$\exists x, y, z, x', y', z', q, w.\ C_1 \wedge C_2$$

Note that the conjunction of atoms C_1 in Q is redundant. Indeed, the query $Q^{min} = \exists x', y', z'.C_2$ is equivalent to Q.

The canonical view of Q for the R_i's source is:

$$\exists y, z, y', z', w, q.R_1(x, y) \wedge R_1(y, z) \wedge R_1(z, x)\wedge \\ R_1(x', y') \wedge R_1(y', z') \wedge R_1(z', x')\wedge \\ R_2(w) \wedge R_3(q) \wedge R_2(y') \wedge R_3(z')$$

In contrast, the canonical view of Q^{min} for this source is

$$\exists y', z'.\ R_1(x', y') \wedge R_1(y', z') \wedge R_1(z', x')\wedge \\ R_2(y') \wedge R_3(z')$$

Theorem 2 tells us that the canonical d-view of Q^{min} is minimally useful within the class of CQ views. We claim that the canonical d-view of Q is not *minimally useful for this class, and in fact reveals significantly more than the canonical d-view of Q^{min}. Consider the secret query $p = R_1(t, t)$ stating that there is an R_1 self-loop. The canonical d-view of Q is UN non-disclosing for p. Indeed, given any instance $\mathcal{D}$, consider the instance $\mathcal{D}'$ in which the T source is identical to the one in $\mathcal{D}$, but the R source is replaced by one where each node e in the canonical view for $\mathcal{D}$ is in a triangle with distinct elements for y', z'. Such a $\mathcal{D}'$ does not satisfy p, and is indistinguishable from $\mathcal{D}$ according to the canonical d-view of Q'.*

In contrast, the canonical d-view is UN disclosing for p, since on the critical instance the returned bindings have x as only $*$, and from this we can infer that the witness elements for y and z can only be $*$. By Proposition 1 the canonical d-view of Q can not be minimally useful within the class of CQ views.

4 Arbitrary views

In the previous section we showed that the canonical d-view is minimally useful within the class of CQ views, assuming that the utility query is minimized. We now turn to minimally informative useful views, not restricting to views given by CQ view definitions.

Our goal will be to arrive at a generalization of the notion of canonical d-view that give the minimal information over arbitrary d-views that are useful for a given BCQ Q. That is we want to arrive at an analog of Theorem 2 replacing "CQ views" by "arbitrary views" and "canonical view" by a generalization.

4.1 An equivalence class representation of minimal useful views

Recall that general views are defined by queries, where a query can be any function on instances. An *Equivalence Class Representation of a d-view* (ECR) consists of an equivalence relation $\equiv_s$ for each source s. An ECR is just another way of looking at a d-view defined by a set of arbitrary functions on instances: given a function F, one can define an equivalence relation by identifying two local instances when the values of F are the same. Conversely, given an equivalence relation then one can define a function mapping each instance to its equivalence class. A d-instance $\mathcal{D}$ is indistinguishable from a d-instance $\mathcal{D}'$ by the d-view specified by ECR $\langle \equiv_s\colon\ s \in \mathsf{Srcs}\rangle$ exactly when $\mathcal{D}_s \equiv_s \mathcal{D}'_s$ holds for each s. Determinacy of one d-view by another corresponds exactly to the refinement relation between the corresponding ECRs. We will thus abuse notation by talking about indistinguishability, usefulness, and minimal usefulness of an ECR, referring to the corresponding d-view.

Our first step will be to show that there is an easy-to-define ECR whose corresponding d-view is minimally useful. For a source s, an *s-context* is an instance for each source other than s. Given an s-context C and an s-instance $\mathcal{I}$, we use $(\mathcal{I}, C)$ to denote the d-instance formed by interpreting the s-relations as in $\mathcal{I}$ and the others as in C.

We say two s-instances $\mathcal{I}, \mathcal{I}'$ are *(s, Q)-equivalent* if for any s-context C, $(\mathcal{I}, C) \models Q \Leftrightarrow (\mathcal{I}', C) \models Q$. We say two d-instances $\mathcal{D}$ and $\mathcal{D}'$ are *globally Q-equivalent* if for each source s, the restrictions of $\mathcal{D}$ and $\mathcal{D}'$ over source s, are (s, Q)-equivalent.

Global Q-equivalence is clearly an ECR. Via "swapping one component at a time" we can see that the corresponding d-views is useful for Q. It is also not difficult to see that this d-view is minimally useful within the class of all d-views:

Proposition 2. *The d-view corresponding to global Q-equivalence is minimally useful for Q within the collection of all d-views.*

Note that the result can be seen as an analog of Theorem 2. From it we conclude an analog of Corollary 1:

Proposition 3. *If there is any d-view that is useful for BCQ Q and non-disclosing for BCQ p, then the d-view given by global Q-equivalence is useful for Q and non-disclosing for p.*

From an ECR to a concrete d-view. We now have a d-view that is minimally useful within the set of all d-views, but it is given only as the ECR global Q-equivalence, and it is not clear that there are any views in the usual sense – isomorphism-invariant functions mapping into relations of some fixed schema — that correspond to this ECR. Our next goal is to show that global Q-equivalence is induced by a d-view defined using standard database queries.

A *shuffle* of a CQ is a mapping from its free variables to themselves (not necessarily injective). Given a CQ Q and a shuffle μ, we denote by $\mu(Q)$ the CQ that results after replacing each variable occurring in Q by its μ-image. We call $\mu(Q)$ a *shuffled query*. For example, consider the query $\exists y.R(x_1, x_2, x_2, y) \wedge S(x_2, x_3, x_3, y)$. Then, the query $\exists y.R(x_2, x_1, x_1, y) \wedge S(x_1, x_1, x_1, y)$ is a shuffle of Q.

The *canonical context query for Q at source s*, $\mathsf{CanCtxt}^s(Q)$, is the CQ whose atoms are all the atoms of Q that are *not* in source s, and whose free variables are $\mathsf{SJVars}(s, Q)$.

Definition 8. *For a source s, a BCQ Q and a variable binding σ for Q, a shuffle μ of $\mathsf{CanView}^s(Q)$ is* invariant relative to $\langle \sigma, \mathsf{CanCtxt}^s(Q)\rangle$ *if for any d-instance $\mathcal{I}'$ where $\mathcal{I}', \sigma \models \mathsf{CanCtxt}^s(Q)$, we have $\mathcal{I}', \sigma \models \mu(\mathsf{CanCtxt}^s(Q))$.*

Note that we can verify this invariance by finding a homomorphism from $\sigma(\mu(\mathsf{CanCtxt}^s(Q)))$ to $\sigma(\mathsf{CanCtxt}^s(Q))$.

Invariance talks about every binding σ. We would like to abstract to bindings satisfying a set of equalities. A *type* for $\mathsf{SJVars}(s, Q)$ is a set of equalities between variables in $\mathsf{SJVars}(s, Q)$. The notion of a variable binding satisfying a type is the standard one. For a type τ, we can talk about a mapping μ being *invariant relative to* $\langle \tau, \mathsf{CanCtxt}^s(Q)\rangle$: the invariance condition holds for all bindings σ satisfying τ.

For a source s and a CQ Q, let $\tau_1, \ldots, \tau_n$ be all the equality types over the variables in $\mathsf{SJVars}(s)$ and $\mathbf{x}$ be the variables in $\mathsf{SJVars}(s)$.

Definition 9. *The* invariant shuffle views *of Q for source s is the set of views $V_{\tau_1}, \ldots, V_{\tau_n}$ where each V_{τ_i} is defined as $\tau_i(\mathbf{x}) \wedge \bigvee_\mu \mu(\mathsf{CanView}^s(Q))$, where $\mathbf{x}$ are the source-join variables of Q for source s, and where the disjunction is over shuffles invariant relative to τ_i.*

Note that since the domain of μ is finite, there are only finitely many mappings on them, and thus there are finitely many disjuncts in each view up to equivalence.

We can show that global Q equivalence corresponds to agreement on these views:

Proposition 4. *For any BCQ Q and any source s, two s-instances are (s, Q)-equivalent if and only if they agree on each invariant shuffle view of Q for s.*

Putting together Proposition 3 and 4, we obtain:

Theorem 3. *[Views that are minimally useful among all views] The invariant shuffle views are minimally useful for Q within the class of all views.*

This yields a corollary for non-disclosure analogous to Corollary 1:

Corollary 3. *If an arbitrary d-view $\mathcal{V}$ is useful for BCQ Q and non-disclosing for BCQ p, then the d-view containing, for each source s, the invariant shuffle views of Q for s, is useful and non-disclosing. In particular, some DCQ is useful for Q and non-disclosing for p.*

Notice that the invariant shuffle views can be *unsafe*: the variables that are contained in a relational atom within the distinct disjuncts may be distinct. Of course, they can be implemented easily by using a wildcard to represent elements outside the active domain. However, we can convert unsafe views to an "information-equivalent" set of safe relational algebra views.

Proposition 5. *For every view defined by a DCQ (possibly unsafe), there is a finite set of relational algebra-based views $\mathcal{V}'$ that induces the same ECR. Applying this to the invariant shuffle views for a CQ Q, we can find a relation algebra-based d-view that is minimally useful for Q within the class of all views.*

The intuition behind the proposition is to construct separate views for different subsets of the variables that occur as a CQ disjunct. A view with a given set of variables S will assert that some CQ disjunct with variables S holds and that no disjunct corresponding to a subset of S holds.

Example 3. *Consider a d-schema with two sources, one containing a ternary relation R and the other containing a unary relation S. Consider the utility query Q:*

$$\exists x_1, x_2, y.R(x_1, x_2, y) \wedge R(y, x_2, x_1) \wedge S(x_1) \wedge S(x_2)$$

The canonical view for the R source $\mathsf{CanView}^R(Q)$ is $\exists y.\ R(x_1, x_2, y) \wedge R(y, x_2, x_1)$.

Observe that Q is a minimal CQ, and hence by Theorem 2 the canonical d-view of Q is minimally useful among the CQ-based d-views. We will argue that this d-view is not minimally useful for Q among all d-views, by arguing that it discloses more secrets than the shuffle views disclose.

Consider the secret query $p = \exists x.R(x, x, x)$. We show that canonical view is UN disclosing for p. Consider the critical instance of the R-source. An external party will know that the instance contains $\{R(*, *, y_0), R(y_0, *, *)\}$ for some y_0. On the other hand, if $y_0 \neq *$, then the d-view would reveal $x_1 = y_0, x_2 = *$. So y_0 must be $*$, and therefore p is disclosed. By Corollary 1, we know that no CQ-based d-views can be UN non-disclosing for p and useful for Q.

The shuffle views of Q are always useful for Q. We will show that they are UN non-disclosing for p. Let us start by deriving the invariant shuffle views for the R source. There are two types, τ_1 in which $x_1 = x_2$, and τ_2 in which the variables are not identified.

For a binding satisfying τ_1, the canonical view of Q for the R source is equivalent to

$$\exists y.R(x_1, x_1, y) \wedge R(y, x_1, x_1)$$

Since there is only one free variable in it, there is only one invariant shuffle, the identity. Thus

$$\mathsf{V}_{\tau_1} = \exists y. R(x_1, x_1, y) \wedge R(y, x_1, x_1)$$

For bindings satisfying τ_2 there are several shuffles invariant for $\mathsf{CanCtxt}^R(Q) = S(x_1) \wedge S(x_2)$: the identity, the shuffle which swaps x_1 and x_2, the shuffle in which x_1 and x_2 both go to x_1, and the shuffle in which both x_1 and x_2 go to x_2. Thus we get the view V_{τ_2} defined as $x_1 \neq x_2$ conjoined with:

$$\begin{aligned}
&\exists y. R(x_1, x_2, y) \wedge R(y, x_2, x_1) \vee \\
&\exists y. R(x_1, x_1, y) \wedge R(y, x_1, x_1) \vee \\
&\exists y. R(x_2, x_2, y) \wedge R(y, x_2, x_2) \vee \\
&\exists y. R(x_2, x_1, y) \wedge R(y, x_1, x_2)
\end{aligned}$$

This last view is unsafe, but via Proposition 5 we can convert it into a safe relational algebra view that yields the same ECR, $\mathsf{V}^{\mathsf{safe}}_{\tau_2}$ defined as $x_1 \neq x_2$ conjoined with:

$$\begin{aligned}
&\neg(\exists y. R(x_1, x_1, y) \wedge R(y, x_1, x_1)) \wedge \\
&\neg(\exists y. R(x_2, x_2, y) \wedge R(y, x_2, x_2)) \wedge \\
&(\exists y. R(x_1, x_2, y) \wedge R(y, x_2, x_1) \vee \\
&\exists y. R(x_2, x_1, y) \wedge R(y, x_1, x_2))
\end{aligned}$$

We now argue that the shuffle views are UN non-disclosing for p. This is because in any d-instance we can replace each fact $R(x_0, x_0, x_0)$ by facts $R(x_0, x_0, c)$ and $R(c, x_0, x_0)$ for a fresh c, obtaining an indistinguishable instance where p does not hold. Hence by Proposition 1, the canonical views of Q can not be minimally useful within the class of all views, or even within the class of relational algebra views.

5 Background knowledge

We now look at the impact of a background knowledge on the sources. We start with the case of a background theory Σ in which each sentence is *local*, referencing relations on a single source.

5.1 Extension of results on CQ views to local background knowledge

It is easy to show that we can not generalize the prior results for CQ views to arbitrary local background knowledge. Intuitively using such knowledge we can encode design problems for arbitrary views using CQ views; see the appendix for details. Thus we restrict to *local constraints Σ that are existential rules*. We show that the results on CQ views extend to this setting. We must now consider utility queries Q that are *minimal with respect to* Σ, meaning that there is no strict subquery equivalent to Q under Σ. By modifying the chase-based approach used to prove Theorem 2, we show:

Theorem 4. *[Minimally useful CQ views w.r.t. local rules] Let Σ be a set of local existential rules, Q a CQ minimal with respect to Σ. Then the canonical d-view of Q is minimally useful within the class of CQ views relative to Σ.*

The analog of Corollary 1 follows from the theorem:

Corollary 4. *If any CQ based d-view is useful for Σ-minimal Q and non-disclosing for p relative to Σ, then the canonical d-views of Q is useful for Q and non-disclosing for p.*

Consequences for decidability. Theorem 4 shows that even in the presence of arbitrary local existential rules Σ it suffices to minimize the utility under Σ and check the canonical d-view for non-disclosure under Σ. For arbitrary existential rules, even CQ minimization is undecidable. But for well-behaved classes of rules (e.g. those with terminating chase (Cuenca Grau et al. 2013; Baget et al. 2014), or frontier-guarded rules (Baget et al. 2011)) we can perform both minimization and UN non-disclosure checking effectively (Benedikt et al. 2019).

5.2 Extension of results on arbitrary views to the presence of local constraints

We can also revise the results on arbitrary d-views to account for local existential rules Σ. The notion of a shuffle being Σ-invariant is defined in the obvious way, restricting to instances that satisfy the constraints in Σ. The Σ-invariant shuffle views are also defined analogously; they are DCQ views, but can be replaced by the appropriate relational algebra views. Following the prior template, we can show:

Theorem 5. *[Minimally useful views w.r.t. local existential rules] For any set of local existential rules Σ, the Σ-invariant shuffle views of Q are minimally useful within the class of all views, relative to Σ.*

The result has effective consequences for "tame" rules (e.g. with terminating chase) with no non-trivial invariant shuffles. In such cases, the Σ-invariant shuffle views degenerate to the canonical d-view, and we can check whether the canonical d-view is non-disclosing effectively (Benedikt et al. 2016; 2019).

5.3 Non-local background knowledge

The simplest kind of non-local constraint is the replication of a table between sources. Unlike local constraints, these require some communication among the sources to enforce. Thus we can consider a replication constraint to be a restricted form of source-to-source communication.

We will see that several new phenomena arise in the presence of these constraints. Recall that with only local constraints, we have useful d-views with minimal information. We can not guarantee the existence of such views in the presence of replication:

Proposition 6. *There is a schema with replication constraint Σ and a BCQ Q where no d-view is minimally useful for Q within the class of all views w.r.t. Σ.*

Our proof of Proposition 6 uses a schema with unary relations R, S, T. There are two sources: R and T are in different sources, and S is replicated between the two sources. Let Q be equal to $\exists x.R(x) \wedge S(x) \wedge T(x)$.

Identify the active domain of our instances with integers, and consider the functions $F_1(x) = x + 1$ and $F_2(x) = 2 - x$. Let Str_1 be the function that applies F_1 to a relation element-wise, and similarly define Str_2 using F_2. Notice that F_1 maps 0 to 1 and 1 to 2, while F_2 maps 0 to 2 and 1 to 1.

We define an ECR $\mathcal{V}_1$ on each source by relating two instances $\mathcal{I}$ and $\mathcal{I}'$ of the source exactly when $\mathcal{I}'$ can be obtained from applying Str_1 on $\mathcal{I}$ some number of times (applying it to both relations of the source) or vice versa. That is $\mathcal{V}_1$ is the equivalence closure of Str_1. Let $\mathcal{V}_2$ be defined analogously using Str_2. We argue that both $\mathcal{V}_1$ and $\mathcal{V}_2$ are useful. We focus on $\mathcal{V}_1$ since the proof for $\mathcal{V}_2$ is analogous. Usefulness follows from the following claim. If we have two valid d-instances $(\mathcal{I}_1, \mathcal{I}_2)$ and $(\mathcal{I}_1', \mathcal{I}_2')$ with all local instances non-empty then 1. We cannot have $\mathcal{I}_1' = \mathsf{Str}_1^i(\mathcal{I}_1)$, $\mathcal{I}_2' = \mathsf{Str}_1^j(\mathcal{I}_2)$ with $i \neq j$; and similarly for Str_2. 2. We cannot have $\mathcal{I}_1' = \mathsf{Str}_1^i(\mathcal{I}_1)$ and $\mathcal{I}_2 = \mathsf{Str}_2^j(\mathcal{I}_2')$; and similarly for Str_2.

Now suppose $\mathcal{V}$ were a set of views minimally useful for Q. We must have $\mathcal{V}_1$ and $\mathcal{V}_2$ determine $\mathcal{V}$. Thus in particular if we have two local instances that agree on *either* $\mathcal{V}_1$ or $\mathcal{V}_2$, then they agree on $\mathcal{V}$.

Consider a d-instance $\mathcal{D}$ with $R = \{0\}$, $S = \{0, 1\}$, $T = \{0\}$, and a d-instance $\mathcal{D}'$ with $R = \{1\}$, $S = \{1, 2\}$ $T = \{2\}$. These are both valid instances (i.e. the replication constraint is respected). But $\mathcal{D}'$ is obtained from $\mathcal{D}$ by applying Str_1 on the source with R, and by applying Str_2 on the other source.

Thus $\mathcal{D}'$ and $\mathcal{D}$ are indistinguishable by $\mathcal{V}$, but Q has a match in $\mathcal{D}$ but not in $\mathcal{D}'$. This contradicts the assumption that $\mathcal{V}$ is useful for Q.

Given Proposition 6 we will focus on obtaining useful views that minimize the queries that are UN non-disclosed for the remainder of the subsection. We can use the technique in the proof of Proposition 6 to show a more promising new phenomenon: there may be d-views that are useful for CQ Q and UN non-disclosing for CQ p, but they are much more intricate than any query related to the canonical d-view of Q. In fact we can show that with a fully-replicated relation in the utility query we can can get useful and UN non-disclosing views whenever this is not ruled out for trivial reasons:

Theorem 6. *If BCQ Q contains a relation of non-zero arity replicated across all sources then there is a d-view that is useful for Q and UN non-disclosing for BCQ p if and only if there is no homomorphism of p to Q.*

Further we can use the same d-view for every such p without a homomorphism into a given Q! Thus even though we do not have minimally useful d-views, we have d-views that are optimal from the perspective of UN non-disclosure and utility for a fixed Q.

We sketch the idea of the proof of Theorem 6. Given utility query Q and local instance $\mathcal{I}$ we can form the "product instance" of $\mathcal{I}$ and Q. The elements of a product instance will be pairs (x, c) where x is a variable of Q and c an element of $\mathcal{I}$, and there will be atom $R((x_1, c_1), \ldots, (x_n, c_n))$ in the product exactly when there are corresponding atoms in $\mathsf{CanView}^s(Q)$ and $\mathcal{I}$. Thus we will have homomorphisms from this instance to both $\mathcal{I}$ and $\mathsf{CanView}^s(Q)$. We use ECRs that make an s-instance $\mathcal{I}$ equivalent to all instances formed by iterating this product construction of $\mathcal{I}$ with $\mathsf{CanView}^s(Q)$. The d-views corresponding to these ECRs will be UN non-disclosing because in the product we will have a fresh copy of any partial match of Q, and so the only way for the secret query p to hold in the product will be if it has a homomorphism into Q, which is forbidden by hypothesis. The replication constraint ensures that if we have two d-instances that are equivalent, where the interpretation of the replicated relation is non-empty, the number of iterations of the product construction is the same on each source. Using this fact we can ensure that the d-views are useful.

The views used in Theorem 6 are not isomorphism-invariant: like the views from Proposition 6, the product construction can be seen as applying some value transformation on the elements of each instance. We can show — in sharp contrast to the situation with only local constraints – that with replication it may be essential to use d-views based on queries that are not isomorphism-invariant.

Proposition 7. *There is a d-schema with a replication constraint, along with BCQs Q and p such that there is a d-view useful for Q and UN non-disclosing for p, but there is no such d-view based on queries returning values in the active domain and commuting with isomorphisms.*

Proposition 7 shows that in the presence of replication, we can do much more with arbitrary d-views than we can even with relational algebra-based d-views.

6 Discussion and outlook

We have studied the ability to design views that satisfy diverse goals: expressiveness requirements in terms of full disclosure of a specified set of queries in the context of data integration, and inexpressiveness restrictions in terms of either minimal utility or minimizing disclosure of queries. Our main results characterize information-theoretically minimal views that support the querying of a given CQ.

We consider only a limited setting; e.g. CQs for the utility query. Our hope is that the work can serve as a basis for further exploration of the trade-offs in using query-based mechanisms in a variety of settings.

References

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

Arenas, M.; Barceló, P.; and Reutter, J. L. 2011. Query languages for data exchange: Beyond unions of conjunctive queries. *Theory Comput. Syst.* 49(2):489–564.

Baget, J.-F.; Mugnier, M.-L.; Rudolph, S.; and Thomazo, M. 2011. Walking the complexity lines for generalized guarded existential rules. In *IJCAI*.

Baget, J.; Garreau, F.; Mugnier, M.; and Rocher, S. 2014. Extending acyclicity notions for existential rules. In *ECAI*.

Benedikt, M.; Bourhis, P.; ten Cate, B.; and Puppis, G. 2016. Querying visible and invisible information. In *LICS*.

Benedikt, M.; Bourhis, P.; Jachiet, L.; and Thomazo, M. 2019. Reasoning about disclosure in data integration in the presence of source constraints. In *IJCAI*.

Benedikt, M.; Cuenca Grau, B.; and Kostylev, E. V. 2018. Logical foundations of information disclosure in ontology-based data integration. *Artif. Intell.* 262:52–95.

Benedikt, M.; ten Cate, B.; and Tsamoura, E. 2016. Generating plans from proofs. In *TODS*.

Bonatti, P. A., and Sauro, L. 2013. A confidentiality model for ontologies. In *ISWC*.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2012. View-based Query Answering in Description Logics: Semantics and Complexity. *J. Comput. Syst. Sci.* 78(1):26–46.

Chaum, D.; Crépeau, C.; and Damgard, I. 1988. Multiparty unconditionally secure protocols. In *STOC*.

Cuenca Grau, B.; Horrocks, I.; Krötzsch, M.; Kupke, C.; Magka, D.; Motik, B.; and Wang, Z. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *JAIR* 47:741–808.

Deutsch, A.; Nash, A.; and Remmel, J. 2008. The chase revisited. In *PODS*.

Dwork, C., and Roth, A. 2014. The algorithmic foundations of differential privacy. *Found. & Trends in Th. Comp. Sci.* 9(3&4):211–407.

Dwork, C. 2006. Differential privacy. In *ICALP*.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theoretical Computer Science* 336(1):89–124.

Fernández-Duque, D., and Goranko, V. 2016. Secure aggregation of distributed information. *Discrete Appl. Math.* 198(C):118–135.

Gogacz, T., and Marcinkowski, J. 2015. The hunt for a red spider: Conjunctive query determinacy is undecidable. In *LICS*.

Halevy, A. Y. 2001. Answering queries using views: A survey. *VLDB J.* 10(4):270–294.

Nash, A., and Deutsch, A. 2007. Privacy in GLAV information integration. In *ICDT*.

Nash, A.; Segoufin, L.; and Vianu, V. 2010. Views and queries: Determinacy and rewriting. *TODS* 35(3).

Onet, A. 2013. The chase procedure and its applications in data exchange. In *DEIS*, 1–37.

APPENDIX

Additional definitions used in the appendix

Let Σ be a set of existential rules, $\mathcal{I}$ be a instance, and Q be a BCQ. We write $\mathcal{I} \wedge \Sigma \models Q$ to mean that every instance containing $\mathcal{I}$ and satisfying Σ also satisfies Q. For two CQs Q and Q', we similarly write $Q \wedge \Sigma \models Q'$ to mean that every instance satisfying Q and Σ satisfies Q'.

The chase. In the appendix we make heavy use of the standard chase algorithm for existential rules (Fagin et al. 2005). The chase modifies an instance by a sequence of *chase steps* until all dependencies are satisfied. Let $\mathcal{I}$ be an instance, τ be a rule of the form

$$\forall \mathbf{x}\ \lambda(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \rho(\mathbf{x}, \mathbf{y}) \tag{1}$$

A *trigger* for τ in $\mathcal{I}$ is a homomorphism h of $\lambda(\mathbf{x})$ into $\mathcal{I}$. Moreover, a trigger h for τ is *active* if no extension of h to a homomorphism of $\rho(\mathbf{x}, \mathbf{y})$ into $\mathcal{I}$ exists. If h is a trigger for τ in $\mathcal{I}$, performing a chase step for τ and h to $\mathcal{I}$ extends $\mathcal{I}$ with each facts of the conjunction $h'(\rho(\mathbf{x}, \mathbf{y}))$, where h' is a substitution such that $h'(x_i) = h(x_i)$ for each variable $x_i \in \mathbf{x}$, and $h'(y_j)$, for each $y_j \in \mathbf{y}$, is a value that does not occur in $\mathcal{I}$ (a "fresh labelled null").

For Σ a set of rules and $\mathcal{I}$ an instance a *chase sequence* for Σ and $\mathcal{I}$ is a (possibly infinite) sequence $\mathcal{I}_0, \mathcal{I}_1, \ldots$ such that $\mathcal{I} = \mathcal{I}_0$ and, for each $i > 0$, instance $\mathcal{I}_i$ if it exists is obtained from $\mathcal{I}_{i-1}$ by applying a successful chase step to a dependency $\tau \in \Sigma$ and an active trigger h for τ in $\mathcal{I}_{i-1}$. The sequence must be *fair*: for each $\tau \in \Sigma$, each $i \geq 0$, and each active trigger h for τ in $\mathcal{I}_i$, some $j > i$ must exist such that h is not an active trigger for τ in $\mathcal{I}_j$ The *result* of a chase sequence is the (possibly infinite) instance $\mathcal{I}_\infty = \bigcup_{i \geq 0} \mathcal{I}_i$. We use $\mathsf{Chase}_\Sigma(\mathcal{I})$ to denote the result of any chase sequence for Σ on $\mathcal{I}$.

A finite chase sequence is *terminating*. A set of dependencies Σ *has terminating chase* if, for each finite, instance $\mathcal{I}$, each chase sequence for Σ and $\mathcal{I}$ is terminating. For such Σ, the chase provides an effective approach to testing if $\mathcal{I} \wedge \Sigma \models Q$: we compute (any) chase $\mathcal{I}_\infty$ for Σ and $\mathcal{I}$ and check if Q holds (Fagin et al. 2005). We can similarly test if $Q \wedge \Sigma \models Q'$ by chasing $\mathsf{canondb}(Q)$ with Σ and then checking Q'. Checking if a set of dependencies Σ has terminating chase is undecidable (Deutsch, Nash, and Remmel 2008). *Weak acyclicity* (Fagin et al. 2005) was the first sufficient polynomial-time condition for checking if Σ has terminating chase. Stronger sufficient (not necessarily polynomial-time) conditions have been proposed subsequently (Cuenca Grau et al. 2013; Onet 2013).

Proofs for Section 2: basic properties of the definitions

Proof of Proposition 1

Recall the statement:

Suppose d-view $\mathcal{V}$ is minimally useful for Q within $\mathcal{C}$, and there is a d-view based on queries in $\mathcal{C}$ that is useful for Q and non-disclosing for p according to non-disclosure function F. Then $\mathcal{V}$ is useful for Q and non-disclosing for p according to F.

Proof. By assumption $\mathcal{V}$ is useful for Q. Suppose $\mathcal{V}'$ based on views in $\mathcal{C}$ is useful for Q and non-disclosing for p according to F. Since $\mathcal{V}$ is minimally useful in $\mathcal{C}$, $\mathcal{V}'$ determines the view definition of each view in $\mathcal{V}$. If $\mathcal{V}$ disclosed p according to F, then $\mathcal{V}'$ would disclose p according to F, since F is a non-disclosure function, hence is determinacy-compatible. □

Proofs for Section 3: balancing expressiveness and inexpressiveness inside the class of views defined by conjunctive queries

Proof of Theorem 2

Recall the statement:

If CQ views $\mathcal{V}$ determine a minimal Boolean CQ Q, then $\mathcal{V}$ determines each canonical view $\mathsf{CanView}^s(Q)$ of Q.

We will make use of the following property of minimal queries, which is easy to verify:

Lemma 1. *If Q is minimal, then there does not exist a homomorphism h from Q into itself that maps two different variables occurring in Q to the same variable.*

We will begin the proof by showing that the determinacy of a CQ Q by a set of CQ views $\mathcal{V}$ leads to the existence of a certain homomorphism of Q to itself.

Let Q be a Boolean conjunctive query and $\mathcal{V}$ be an arbitrary set of conjunctive query views. We will need to review an algorithm for checking determinacy. We fix a signature for our queries and views, which we refer to as the *original signature*. From it we derive a *primed signature*, containing a relation R' for each R in the original signature. Given a formula φ in the original signature, we let φ' be formed by replacing every relation R in φ by R'. We use a similar notation for a set of facts S in the original signature. In particular, for a conjunctive query Q in the original signature, Q' refers to the conjunctive query obtained by replacing every relation symbol by its primed counterpart.

Given a view $\mathsf{V}(\mathbf{x})$ defined by conjunctive query $\exists \mathbf{y}\ \varphi(\mathbf{x},\mathbf{y})$, the corresponding *forward view definition* (for V) is the rule:

$$\varphi(\mathbf{x},\mathbf{y}) \rightarrow \mathsf{V}(\mathbf{x})$$

while the *inverse view definition* for V is the rule:

$$\mathsf{V}(\mathbf{x}) \rightarrow \exists \mathbf{y}.\varphi(\mathbf{x},\mathbf{y})$$

We let $\mathsf{ForView}(\mathcal{V})$ and $\mathsf{BackView}(\mathcal{V})$, denote the forward and inverse view definitions for views in $\mathcal{V}$, and $\mathsf{ForView}'(\mathcal{V})$ and $\mathsf{BackView}'(\mathcal{V})$ the same axioms but for primed copies of the base predicates.

Example 4. *Suppose we have a set of views $\mathcal{V}$ consisting only of view $V(x)$ given by CQ $\exists y, z.R(x,y) \wedge S(x,z)$. Then $\mathsf{ForView}(\mathcal{V})$ contains the rule:*

$$R(x,y) \wedge S(x,z) \rightarrow V(x)$$

while $\mathsf{BackView}'(\mathcal{V})$ contains the rule

$$V(x) \rightarrow \exists y, z.R'(x,y) \wedge S'(x,z)$$

Letting $\Sigma_{Q,\mathcal{V}}$ to be the axioms above, we can easily see that determinacy can be expressed as:

$$Q \wedge \Sigma_{Q,\mathcal{V}} \models Q'$$

This is a containment of CQs under existential rules, and Algorithm 1 checks this via the chase procedure, which is complete for such containments (see Section 2).

Intuitively, the algorithm iteratively chases with the axioms above in rounds, checking for a match of Q' after each round. The correctness is implicit in (Nash, Segoufin, and Vianu 2010); see also (Benedikt, ten Cate, and Tsamoura 2016; Gogacz and Marcinkowski 2015).

Theorem 7. *Algorithm 1 returns true if and only if $\mathcal{V}$ determines Q.*

Given an arbitrary input query Q, we use a superscript to distinguish among the different sets computed during each iteration of Algorithm 1, e.g., $F_0^i(Q,\mathcal{V})$ denotes instance F_0 during the i-th iteration of Algorithm 1 when run on Q with $\mathcal{V}$. For any $Q,\mathcal{V}$ such that the algorithm returns **true**, and any $j \in 0 \dots 5$, let $F_j^\infty(Q,\mathcal{V})$ denote the content of $F_j(Q,\mathcal{V})$ at the point where the algorithm terminates; this will be $F_j^l(Q,\mathcal{V})$ for some l. Note that in line 11 of Algorithm 1 we add to each $F_0^l(Q,\mathcal{V})$ only the subset of ground atoms from $F_4^l(Q,\mathcal{V})$ that belong to the original schema, since the other atoms will be re-derived in the next round.

For any fact $F = R'(\mathbf{c})$ in the primed signature, let $\mathsf{UnPrime}(F)$ be the corresponding fact $R(\mathbf{c})$. For a set of facts S, we let

$$\mathsf{UnPrime}(S) = \{\mathsf{UnPrime}(F) : F \in S \text{ in the primed signature }\}$$

The following lemma states the easy fact that iterating the "chase and backchase" steps in lines 3 to 4 gives a homomorphic pre-image of what we started with:

Figure 1: $\mathsf{Determinacy}(Q, \mathcal{V})$

```
1: $F_0 := \mathsf{canondb}(Q)$
2: **while true do**   % Next 2 lines create view facts from $F_0$ and then primed facts based on views
3:     $F_1 := \mathsf{Chase}_{\mathsf{ForView}(\mathcal{V})}(F_0)$
4:     $F_2 := \mathsf{Chase}_{\mathsf{BackView}'(\mathcal{V})}(F_1)$
5:     **if** $\exists h : Q' \to F_2$ mapping each free variable $v$ of $Q'$ into $c_v \in \mathsf{adom}(\mathsf{canondb}(Q))$ **then return true**
6:     **end if**
7:     $F_3 := \mathsf{Chase}_{\mathsf{ForView}'(\mathcal{V})}(F_2)$   %These 2 lines create view facts from primed facts and then original facts based on views
8:     $F_4 := \mathsf{Chase}_{\mathsf{BackView}(\mathcal{V})}(F_3)$
9:     $F_5 =$ restrict $F_4$ to original schema
10:    **if** $F_0 \neq F_5$ **then**   % no fixpoint, so repeat with expanded original facts
11:        $F_0 := F_0 \cup F_5$,
12:    **else**
13:        **return false**
14:    **end if**
15: **end while**
```

Lemma 2. *For each $l \geq 1$, there is a homomorphism μ_l : $\mathsf{UnPrime}(F_2^l(Q, \mathcal{V})) \to F_0^l(Q, \mathcal{V})$ that preserves all elements of facts in $F_0^l(Q, \mathcal{V})$.*

Proof. Let $\mathcal{V}_i \subseteq \mathcal{V}$ be the set of all views providing data from the i-th datasource and let $\mathsf{V}_{i,1} \ldots \mathsf{V}_{i,j_i}$ enumerate the view predicate names in $\mathcal{V}_i$. , where $\mathsf{V}_{i,j}$ has definition $\mathbf{B}_{i,j}$. Consider the forward view definition, forward transfer as well as, for each $\mathsf{V}_{i,j}$, the inverse view definition associated going from $\mathsf{V}_{i,j}$ to the primed relations. That is, the axioms:

$$\mathbf{B}_{i,j}(\mathbf{x}, \mathbf{y}) \to \mathsf{V}_{i,j}(\mathbf{x})$$
$$\mathsf{V}_{i,j}(\mathbf{x}) \to \exists \mathbf{y}\; \mathbf{B}'_{i,j}(\mathbf{x}, \mathbf{y})$$

where $\mathbf{B}_{i,j}(\mathbf{x})$ is some conjunction of atoms from the i-th datasource and $\{\mathbf{y}\}$ is disjoint from $\{\mathbf{x}\}$.

Each null λ appearing in $F_2^l(Q, \mathcal{V})$ is generated by a chase step associated with Line 4 of Algorithm 1, triggered by a unique fact $\mathsf{V}_{i,j}(\sigma)$, where σ is a binding of $\mathbf{x}$, with λ corresponding to some variable y_λ in $\mathbf{B}'_{i,j}(\mathbf{x}, \mathbf{y})$. For each fact $\mathcal{V}'_{i,j}(\sigma)$, we know we can extend to a binding σ' for $\mathbf{y}$ such that $\mathbf{B}_{i,j}(\sigma')$ is in $F_0^l(Q, \mathcal{V})$. Our homomorphism will map λ to $\sigma'(v_\lambda)$. □

An analogous reasoning applies to lines 7 to 8:

Lemma 3. *For any CQ Q, each $l \geq 1$ there exists a homomorphism μ_l : $F_5^l(Q, \mathcal{V}) \to \mathsf{UnPrime}(F_2^l(Q, \mathcal{V}))$ that preserves all elements occurring in $\mathsf{UnPrime}(F_2^l(Q, \mathcal{V}))$.*

Using the two lemmas above, we can prove by induction on l:

Lemma 4. *For any CQ Q, CQ views $\mathcal{V}$, and number l, there exists a homomorphism $\mathsf{UnPrime}(F_2^l(Q, \mathcal{V})) \to \mathsf{canondb}(Q)$ that is the identity on elements c_v. By priming each relation, we can get $\mu : F_2^l(Q, \mathcal{V}) \to \mathsf{canondb}(Q')$.*

The following simple proposition relates the sets produced by the algorithm on input Q with the sets produced by the same algorithm on one of the canonical views $\mathsf{CanView}^s(Q)$.

Lemma 5. *For any $l \geq 1$ and any source s,*
$F_0^l(\mathsf{CanView}^s(Q), \mathcal{V})$ is the same as the set of atoms in source s within $F_0^l(Q, \mathcal{V})$.

We are now ready for the proof of Theorem 2:

Proof. Assume that $\mathcal{V}$ does not determine $\mathsf{CanView}^s(Q)$, for some datasource s, and that $\mathcal{V}$ determines Q. From the latter assumption and Theorem 7 we conclude that there exists a homomorphism h_1 from Q' into $F_2^\infty(Q, \mathcal{V})$.

By composing h_1 with the ν produced by Lemma 4 we obtain a homomorphism h_Q from Q' into itself. By unpriming, we can change h_Q to a homomorphism from Q to itself, and we will sometimes abuse notation by considering h_Q in either way. Note that h_1, and also h_Q, maps each variable x in $\mathsf{SJVars}(s, Q)$ to some element of the form c_v, since each such x appears in atoms over distinct sources, and only elements of the form c_v appear in such atoms within $F_2^\infty(Q, \mathcal{V})$.

We consider two cases.

Case 1: h_Q is injective. Thus h_Q is both an injection on Q' and an injection from source-join variables to the constant corresponding to a source-join variable. Then $(h_Q)^{-1}$ composed with h_1 is an injective homomorphism from Q' into $F_2^\infty(Q, \mathcal{V})$ that maps each x in $\mathsf{SJVars}(s, Q)$ to c_x. By Lemma 5 we can identify the source s atoms of $F_2^\infty(Q, \mathcal{V})$ with the atoms of $F_2^\infty(\mathsf{CanView}^s(Q), \mathcal{V})$. Applying this identification and abusing notation as described above we can see $(h_Q)^{-1}$ as a homomorphism of Q into $F_2^\infty(\mathsf{CanView}^s(Q), \mathcal{V})$. that maps each v in $\mathsf{SJVars}(s, Q)$ to c_v. But then Algorithm 1 would have returned true for $\mathsf{CanView}^s(Q)$. By Theorem 7, this is a contradiction of the fact that $\mathcal{V}$ does not determine $\mathsf{CanView}^s(Q)$.

Case 2: h_Q is not injective. But now, by Lemma 1, we have a contradiction of minimality. □

Proof of Corollary 2, and further optimizations

Recall the statement of the corollary:

There is a Σ_2^p algorithm taking as input BCQs Q and p without constants, that determines whether there is a CQ based d-view that is useful for Q and UN non-disclosing for p. If Q is assumed minimal than the algorithm can be taken to be in CoNP.

Proof. We know it suffices to find a minimal query Q^{min} and check that the canonical views of Q^{min} are UN non-disclosing for p. This means guessing a homomorphic image Q^{min} of Q and checking that Q^{min} is UN non-disclosing for p. (Benedikt, Cuenca Grau, and Kostylev 2018) showed that there is a CoNP procedure that takes as input views and a query p, and checks whether or not the views are UN non-disclosing for p. Composing the guess with that check gives the Σ_2^p bounds. When Q is minimal, we can just use the CoNP check of (Benedikt, Cuenca Grau, and Kostylev 2018). □

In fact, we can do slightly better. The canonical views are a way of decomposing a utility query. But it turns our that we can also apply the same construction underlying the canonical views as a way of decomposing a secret query. We start with the following observation, which states that in analyzing secrecy of a set of CQ views, we can break up the secret query into its canonical views and analyze them one at a time:

Proposition 8. *For any Boolean CQ p and CQ-based d-view $\mathcal{V}$, $\mathcal{V}$ is UN non-disclosing for p if and only if $\mathcal{V}$ is UN non-disclosing for* $\mathsf{CanView}^s(p)$ *for some s.*

Proof. In one direction, suppose there is s such that $\mathcal{V}$ is UN non-disclosing for $\mathsf{CanView}^s(p)$. We show that $\mathcal{V}$ is UN non-disclosing for p by showing that the critical instance $\mathcal{D}_0$ for the d-schema is $\mathcal{V}$-indistinguishable from some other d-instance where p fails. We know that there is some s-instance $\mathcal{I}_s$ that is $\mathcal{V}^s$ indistinguishable from $\mathcal{D}_s$ but in which $\mathsf{CanView}^s(p)(*)$ does not hold. Let $\mathcal{D}'$ be formed from taking $\mathcal{I}_s$ on source s and taking the critical instance on the other sources. choosing the other components arbitrarily. Clearly $\mathcal{D}'$ is $\mathcal{V}$-indistinguishable from $\mathcal{D}$. If p held on $\mathcal{D}'$, the only possible witness would be the critical tuple, since this is the only binding. But the critical tuple fails the conjuncts on source s.

In the other direction, suppose $\mathcal{V}$ is UN disclosing for $\mathsf{CanView}^s(p)$ for some s. We will show that $\mathcal{V}$ is UN disclosing for p. We know that for each s, letting $\mathcal{I}_s$ be the critical instance for source s, if we apply the disjunctive chase procedure from (Benedikt et al. 2019; 2016) to $\mathcal{I}_s$ then $\mathsf{CanView}^s(* \ldots *)$ holds. But when we apply the disjunctive chase procedure to the critical instance for the d-schema, this is the same as applying it to each component. Thus p holds with $* \ldots *$ as a witness, so $\mathcal{V}$ is UN disclosing for p. □

The result above is about a fixed set of CQ views. But using Corollary 1 we can lift it to an observation about decomposing the secret query in searching for the existence of useful and UN non-disclosing views:

Proposition 9. *For any Boolean CQs Q and p, there is a CQ-based d-view that is useful for Q and UN non-disclosing for p if and only if for some s, the canonical views of Q are UN non-disclosing for* $\mathsf{CanView}^s(p)$.

Proof. If there are CQ views that are useful for Q and UN non-disclosing for p, then the canonical views of Q are such a set of views, by Corollary 1. Thus by the previous proposition, they are UN non-disclosing for some $\mathsf{CanView}^s(p)$.

In the other direction, if the canonical views of Q are UN non-disclosing for $\mathsf{CanView}^s(p)$ for some s, then by the proposition above they are UN non-disclosing for p, and thus these views serve as a witness. □

Proposition 9 implies that to test for useful and UN non-disclosing views, we need only take each source s and test whether $\mathsf{CanView}^s(Q)$ for source s is UN non-disclosing for the canonical view of $\mathsf{CanView}^s(p)$ for source s. A witness to failure of such a test requires first a deterministic computation that consists of chasing forward and backward with the view definitions, then a series of guesses of homomorphism of $\mathsf{CanView}^s(Q)$, followed by the guess of a homomorphism of $\mathsf{CanView}^s(p)$, thus CoNP in the maximum cardinality over all s of $\mathsf{CanView}^s(p)$ and $\mathsf{CanView}^s(Q)$.

Proofs for Section 4: balancing expressiveness and inexpressiveness within the class of arbitrary views

Proof of Proposition 2

Recall the statement:

> The set of views corresponding to global Q-equivalence is minimally useful for Q within the collection of all views.

Proof. To see that Q-equivalence is useful, consider two d-instances $\mathcal{D}$ and $\mathcal{D}'$ that are globally Q-equivalent. Source-by-source, we can modify $\mathcal{D}$ on a source s to be the same as $\mathcal{D}'$ on source s, and the results of Q is not changed, by s, Q-equivalence.

Assume $\equiv$ is useful and $\mathcal{D} \equiv \mathcal{D}'$. Fix a context C for source s such that $(\mathcal{D}_s, C) \models Q$. We will argue that $(\mathcal{D}'_s, C) \models Q$. Note that for every source s, identical instances for source s must be $\equiv_s$, since $\equiv_s$ is an equivalence relation. Thus $(\mathcal{D}_s, C) \equiv (\mathcal{D}'_s, C)$. Since $\equiv$ is useful for Q, this means $(\mathcal{D}'_s, C) \models Q$ as required. □

Proof of Proposition 4

Recall the statement:

> For any BCQ Q and any source s, two s-instances are (s, Q)-equivalent if and only if they agree on each invariant shuffle view of Q for s.

The proof will go through an intermediate view, also given by an ECR.

Definition 10. *Given two s-instances $\mathcal{I}_1$ and $\mathcal{I}_2$, we say that $\mathcal{I}_1$ and $\mathcal{I}_2$ are* invariant shuffle equivalent *(relative to Q) if: Whenever $\mathcal{I}_1, \sigma$ satisfies $\mathsf{CanView}^s(Q)(\mathbf{x})$ then there is some shuffle μ invariant relative to $\langle \sigma, \mathsf{CanCtxt}^s(Q)\rangle$, such that $\mathcal{I}_2, \sigma$ satisfies $\mu(\mathsf{CanView}^s(Q))(\mathbf{x})$ and similarly with the role of $\mathcal{I}_1$ and $\mathcal{I}_2$ reversed.*

We show:

Proposition 10. *For any source s invariant shuffle equivalence is identical to s, Q-equivalence.*

Proof. First, suppose $\mathcal{I}_1, \mathcal{I}_2$ are local instances for source s that are invariant shuffle equivalent, and suppose we have a match of Q in $(\mathcal{I}_1, C)$ via $h^{1,C}$. We want to show that there is a match in $(\mathcal{I}_2, C)$.

We know that the variables in $\mathsf{SJVars}(s, Q)$ are mapped by $h^{1,C}$ into $\mathcal{I}_1$. Let h_0 be the restriction of $h^{1,C}$ to the variables of $\mathsf{SJVars}(s, Q)$. Then $\mathcal{I}_1, h_0$ satisfies $\mathsf{CanView}^s(Q)$. Thus by shuffle equivalence there is some shuffle μ invariant relative to $\langle h_0, \mathsf{CanView}^s(Q)\rangle$, such that $\mathcal{I}_2, h_0 \models \mu(\mathsf{CanView}^s(Q))$, with witness h_2 extending h_0. We also know that C, h_0 satisfies $\mathsf{CanCtxt}^{S_0}(Q)$, since $h^{1,C}$ witnesses this as well. Applying the definition of shuffle invariance, C, h_0 satisfies $\mu(\mathsf{CanCtxt}^{S_0}(Q))$. Let $h^{\mu,C}$ be a homomorphism witnessing this. Note that since $h^{\mu,C}$ extends h_0 and h_0 restricts $h^{1,C}$, $h^{\mu,C}$ and $h^{1,C}$ agree on their common variables. Define $h^{2,C}$ by mapping the variables in $\mathsf{SVars}(s, Q)$ as in h_2, and those variables outside of $\mathsf{SJVars}(s, Q)$ as in $h^{\mu,C}$. Since these are two compatible homomorphisms, $h^{2,C}$ witnesses that $(\mathcal{I}_2, C) \models Q$. This completes the argument that invariant shuffle equivalence implies global Q-equivalence.

We now show that global Q-equivalence implies shuffle equivalence. Suppose s-instances $\mathcal{I}_1$, $\mathcal{I}_2$ are globally Q-equivalent, and $\mathcal{I}_1, \sigma$ satisfies $\mathsf{CanView}^s(Q)(\mathbf{x})$, We will show that there is a shuffle μ, invariant relative to $\langle \sigma, \mathsf{CanView}^s(Q)\rangle$, such that $\mathcal{I}_2, \sigma \models \mu(\mathsf{CanView}^s(Q))(\mathbf{x})$.

Let C_1 be the canonical database of $\sigma(\mathsf{CanCtxt}^s(Q))$. That is, for each source s other than s, we have a fact for each atom of s atom of Q, where each variable x of $\mathsf{SJVars}(s, Q)$ is replaced by $\sigma(x)$ and each variable x not in $\mathsf{SJVars}(s, Q)$ is replaced by a fresh element c_x.

Q clearly holds in $(\mathcal{I}_1, C_1)$. So by global Q-equivalence, Q holds in $(\mathcal{I}_2, C_1)$ via some homomorphism h. Note that in $(\mathcal{I}_2, C_1)$ the only elements that are shared between $\mathcal{I}_2$-facts and C_1-facts lie in the range of σ. Thus h must map the variables in $\mathsf{SJVars}(s, Q)$ to the image of σ. The binding σ may not be injective, but we let σ^{-1} be "some inverse" that is, any function from the range of σ to variables such that for any c in the range of σ $\sigma(\sigma^{-1}(c)) = c$. Let μ map any variable $x \in \mathsf{SJVars}(s, Q)$ to $\sigma^{-1}(h(x))$. So $\sigma(\mu(x)) = h(x)$.

We first claim that μ is invariant relative to $\langle \sigma, \mathsf{CanCtxt}^s(Q)\rangle$. We show this by arguing that h is a homomorphism from $\sigma(\mu(\mathsf{CanCtxt}^s(Q)))$ to $\sigma(\mathsf{CanCtxt}^s(Q))$. By definition of μ, we have for each atom $A(x_1 \dots x_m, y_1 \dots y_n)$ of $\mathsf{CanCtxt}^s(Q)(\mathbf{x})$,

$$A(h(x_1) \dots h(x_m), h(y_1) \dots h(y_n)) =$$
$$A(\sigma(\mu(x_1)) \dots \sigma(\mu(x_m)), h(y_1) \dots h(y_n))$$

$A(h(x_1)\dots h(x_m), h(y_1)\dots h(y_n))$ is in $\sigma(\mathsf{CanCtxt}^s(Q))$ since by assumption h is a homomorphism from Q to $(\mathcal{I}_2, C_1)$, C_1 is the canonical database of $\sigma(\mathsf{CanCtxt}^s(Q))$, and $\mathcal{I}_2$ is an s-instance, and hence cannot contain any facts over the relations in $(\mu(\mathsf{CanCtxt}^s(Q)))(\sigma)$. Thus

$$A(\sigma(\mu(x_1))\dots\sigma(\mu(x_m)), h(y_1)\dots h(y_n))$$

lies in $\sigma(\mathsf{CanCtxt}(Q))$ as required.

We next claim that $\mathcal{I}_2, \sigma \models \mu(\mathsf{CanView}^s(Q))$. The witness will be the extension h' of σ that maps all variables in $\mathsf{SVars}(s, Q) - \mathsf{SJVars}(s, Q)$ via h.

Consider an atomic formula $A(x_1\dots x_m, y_1\dots y_n)$ of $\mathsf{CanView}^s(Q)$, where $\mathbf{x}$ are free variables of $\mathsf{CanView}^s(Q)$. Therefore $A(\mu(x_1)\dots\mu(x_m), y_1\dots y_n)$ is a generic atom of $\mu(\mathsf{CanView}^s(Q))$. To argue that h' is a homomorphism that witnesses $\mathcal{I}_2, \sigma \models \mu(\mathsf{CanView}^s(Q))$, we need to argue that

$$A(\sigma(\mu(x_1))\dots\sigma(\mu(x_m)), h(y_1)\dots h(y_n))$$

holds in $\mathcal{I}_2$.

But by the definition of μ, this is equivalent to showing that

$$A(h(x_1)\dots h(x_m), h(y_1)\dots h(y_n))$$

holds in $\mathcal{I}_2$. But this follows since h is a homomorphism of Q into $(\mathcal{I}_2, C_1)$. □

To complete the proof of Proposition 4 we show:

Proposition 11. *For any CQ Q and source s, two s-instances are invariant shuffle equivalent if and only if they agree on each invariant shuffle view of Q for s.*

Proof. We first show that if $\mathcal{I}_1$ and $\mathcal{I}_2$ agree on the invariant shuffle views of Q, they are invariant shuffle equivalent. Suppose $\mathcal{I}_1, \sigma \models \mathsf{CanView}^s(Q)$, and let τ be the type of σ. Since the identity is invariant relative to τ, we have $\mathcal{I}_1, \sigma$ satisfies V_τ, and thus $\mathcal{I}_2, \sigma$ must satisfy it. Therefore there is μ that is invariant relative to τ such that $\mathcal{I}_2, \sigma \models \mu(\mathsf{CanView}^s(Q))$. Since τ is of type σ, we have μ is invariant relative to $\langle \sigma, \mathsf{CanCtxt}^s(Q)\rangle$. Arguing symmetrically for $\mathcal{I}_2$, we see that $\mathcal{I}_1$ and $\mathcal{I}_2$ are invariant shuffle equivalent.

In the other direction, suppose $\mathcal{I}_1$ and $\mathcal{I}_2$ are invariant shuffle equivalent. We will argue that they agree on each invariant shuffle view V_τ.

Towards that end, suppose $\mathcal{I}_1, \sigma \models V_\tau$. That is, $\mathcal{I}_1, \sigma \models \tau \wedge \mu(\mathsf{CanView}^s(Q))$ for some μ that is invariant relative to τ. Let σ' be the pre-image of σ under μ: that is, the variable binding defined by $\sigma'(x) = \sigma(\mu(x))$. Then $\mathcal{I}_1, \sigma' \models \mathsf{CanView}^s(Q)$ by definition. Thus by invariant shuffle equivalence, there is μ' invariant for σ', $\mathsf{CanCtxt}^s(Q)$ such that $\mathcal{I}_2, \sigma' \models \mu'(\mathsf{CanView}^s(Q))$.

Let $\mu'' = \mu'(\mu)$. We will show that μ'' witnesses that $\mathcal{I}_2, \sigma \models V_\tau$. We first verify invariance:

Claim 1. *μ'' is invariant relative to $\tau, \mathsf{CanCtxt}^s(Q)$.*

Proof. Suppose σ_0 satisfies τ, and $\mathcal{I}_0, \sigma_0 \models \mathsf{CanCtxt}^s(Q)$. Let σ_0' be the pre-image of σ' under μ. Note that a shuffle that is invariant for σ must be invariant for σ_0, since σ_0 satisfies all the equalities that σ does. Similarly a shuffle that is invariant for σ' must be invariant for σ_0'. We can see that the following chain of implications:

$$\begin{aligned}
\mathcal{I}_0, \sigma_0 &\models \mu(\mathsf{CanCtxt}^s(Q))\text{by invariance of } \mu \text{ for } \tau\\
\mathcal{I}_0, \sigma_0' &\models \mathsf{CanCtxt}^s(Q)\text{by definition of } \sigma_0'\\
\mathcal{I}_0, \sigma_0' &\models \mu'(\mathsf{CanCtxt}^s(Q))\text{by invariance of } \mu' \text{ for } \sigma_0'\\
\mathcal{I}_0, \sigma_0 &\models \mu(\mathsf{CanCtxt}^s(Q))\text{by definition of } \sigma_0' \text{ again}
\end{aligned}$$

□

We now show that $\mathcal{I}_2, \sigma$ satisfies the corresponding shuffled query.

Claim 2. $\mathcal{I}_2, \sigma \models \mu''(\mathsf{CanView}^s(Q))$.

Proof. We make the following observation. For any instance $\mathcal{I}$, bindings σ_0, CQs R, and shuffles μ_0, let σ_1 be the pre-image of σ_0 under μ_0. Then $\mathcal{I}, \sigma_0 \models \mu(R)$ if and only if $\mathcal{I}, \sigma_1 \models R$.

Let σ'' be the pre-image of σ under μ''. Note that σ'' is also the pre-image of σ' under μ'.

From the observation above, we see that the following are equivalent:

$$\begin{aligned}\mathcal{I}_2, \sigma &\models \mu(\mathsf{CanView}^s(Q))\\ \mathcal{I}_2, \sigma'' &\models \mathsf{CanView}^s(Q)\\ \mathcal{I}_2, \sigma' &\models \mu'(\mathsf{CanView}^s(Q))\end{aligned}$$

which gives the proof of the claim. □

Putting together the two claims, We conclude that $\mathcal{I}_2, \sigma$ satisfies V_τ as required, which completes the proof of Proposition 11. □

Proof of Proposition 5

Recall the statement:

For every view defined by a DCQ (possibly unsafe), there is a set of relational algebra views $\mathcal{V}'$ that induces the same ECR. Applying this to the invariant shuffle views for a CQ Q, we can find a relational algebra-based d-view that is minimally useful for Q within the class of all views.

Putting the conclusion "same ECR" another way: if V is the original DCQ view, then we obtain a finite set of views $\mathcal{V}'$ with the definition of each view in relation algebra, such that $\mathcal{V}'$ determines V and V determines $\mathcal{V}'$.

Clearly if we have this for a single DCQ view V, we obtain it for a finite set of views (and hence for a d-view) by applying the construction to each view in the set.

We consider a DCQ $V(x_1 \dots x_n)$ defined by $\bigvee_i \varphi_i$. For each φ_i let Vars_i be the set of variables within them, and for each subset S of the vars let D_S be the set of i such that φ_i uses variables S.

Given a set of variables $S = x_{j_1} \dots x_{j_k}$ with $D_S \neq \emptyset$, create a view $V_S(x_{j_1} \dots x_{j_k})$ defined by

$$\bigvee_{i \in D_S} \varphi_i(x_{j_1} \dots x_{j_k}) \wedge \neg(\bigvee_{S' \subsetneq D_S, \mathsf{Vars}(\varphi_j) = S'} \varphi_j)$$

Example 5. *We explain the construction of relational algebra views by example. Suppose we have a view V given by a DCQ:*

$$R(x, y, z) \vee P(x, y, z) \vee W(x, y, w) \vee T(x, y)$$

We have three sets S such that $D_S \neq \emptyset$: $S_1 = \{x, y, z\}, S_2 = \{x, y, w\}$ and $S_3 = \{x, y\}$.

Our construction will create views for each of these.

$V_{S_1}(x, y, z)$ is defined by query:

$$R(x, y, z) \vee P(x, y, z)] \wedge \neg T(x, y)$$

$V_{S_2}(x, y)$ is defined by query:

$$W(x, y) \wedge \neg T(x, y)$$

Finally, $V_{S_3}(x, y)$ is defined by the query $T(x, y)$.

It is not difficult to see that these views determine V and vice versa.

Returning to the general case, we claim that the set of views V_S determines V and vice versa.

In one direction suppose $\mathcal{I}_1$ and $\mathcal{I}_2$ agree on each V_S, and $\mathcal{I}_1 \models V(\mathbf{t})$. Choose i such that $\mathcal{I}_1 \models \varphi_i(\mathbf{t})$ with $S_i = \mathsf{Vars}_i$ minimal. Let $\mathbf{t}'$ be the subtuple of $\mathbf{t}$ corresponding to the variables of φ_i. Then $\mathcal{I}_1 \models V_{S_i}(\mathbf{t}')$ and hence $\mathcal{I}_2 \models V_{S_i}(\mathbf{t}')$. From this we see that $\mathcal{I}_2 \models V_S(\mathbf{t})$.

In the other direction, suppose $\mathcal{I}_1$ and $\mathcal{I}_2$ agree on V, and $\mathcal{I}_1 \models V_S(\mathbf{t})$. Fix φ_i with variables from S such that $\mathcal{I}_1 \models \varphi_i(\mathbf{t})$. We need to show $\mathcal{I}_2 \models V_S(\mathbf{t})$. We can assume by induction that V determines $V_{S'}$ for each S' that is a proper subset of S. First consider the case where S consists of all variables. Then $\mathcal{I}_1 \models V(\mathbf{t})$ hence $\mathcal{I}_2 \models V(\mathbf{t})$, and thus there is some j such that $\mathcal{I}_2 \models \varphi_j(\mathbf{t})$. If φ_j contains all the variables of φ_i. Using the induction hypothesis and φ_j we can conclude that $\mathcal{I}_2 \models V_S(\mathbf{t})$ as required. If φ_j contains a proper subset S' of the variables in S, then we have $\mathcal{I}_2 \models \varphi_k(\mathbf{t}')$ for $\mathbf{t}'$ a proper subtuple of $\mathbf{t}$. Choose φ_k and $\mathbf{t}'$ with this property such that the variables S' involved are minimized. The $\mathcal{I}_2 \models V_{S'}(\mathbf{t}')$ so by the induction hypothesis $\mathcal{I}_1 \models V_{S'}(\mathbf{t}')$, which contradicts the facts that $\mathcal{I}_1 \models V_S(\mathbf{t})$.

Next consider the case where S is a proper subset of the variables. We extend $\mathbf{t}$ to $\mathbf{t}'$ choosing elements outside the active domain of both $\mathcal{I}_1$ and $\mathcal{I}_2$. $\mathcal{I}_1 \models V(\mathbf{t}')$, and $\mathcal{I}_2 \models V(\mathbf{t}')$. Thus we have a proper subtuple $\mathbf{t}''$ of $\mathbf{t}'$ and a disjunct

φ_k such $\mathcal{I}_2 \models \varphi_i(\mathbf{t}'')$. As above, we can choose $\mathbf{t}''$ minimal. By our choice of the elements in $\mathbf{t}' - \mathbf{t}$, we must have $\mathbf{t}''$ a subtuple of $\mathbf{t}$. If $\mathbf{t}'' = \mathbf{t}$, then we can conclude that $\mathcal{I}_2 \models V_S(\mathbf{t})$ as required. If $\mathbf{t}''$ is a proper subtuple, we argue by contradiction of the induction hypothesis as above.

Proofs for Section 5: balancing requirements in the presence of constraints

Minimally useful CQs beyond local existential rules

In the body of the paper we mentioned that we can not generalize results such as Corollary 1 from Section 3 to the presence of arbitrary local constraints. The following example exhibits the problem.

Consider the schema and utility query Q from Example 3. Let Σ consist of the view definitions for the invariant shuffle views in the example. We claim that the canonical views of Q are not minimally useful for Q within the set of CQ views, relative to Σ.

To see this, we consider the secret query p from Example 3.

Relative to Σ we have a distributed view based on CQ view definitions that is useful, and UN non-disclosing, namely the views those simply export each shuffle view. However, as observed in Example 3, the canonical views are-UN disclosing for p. Thus, applying Proposition 1, the canonical views can not be minimally useful among the class of all CQ views relative to Σ.

Proof of Theorem 4

Recall the statement:

Let Σ be any set of existential rules that are local. Suppose that Q is minimal with respect to Σ. If CQ views $\mathcal{V}$ determine CQ Q over all instances satisfying Σ, then $\mathcal{V}$ determines each canonical view $\mathsf{CanView}^s(Q)$ of Q over all instances satisfying Σ.

We proceed as in the case of no constraints. We modify the determinacy algorithm by chasing with the local constraints in each round, giving us the Algorithm 2.

Figure 2: $\mathsf{Determinacy}(Q, \mathcal{V}, \Sigma)$

```
$F_0 := \mathsf{Chase}_\Sigma(\mathsf{canondb}(Q))$
while true do
    $F_1 := \mathsf{Chase}_{\mathsf{ForView}(\mathcal{V})}(F_0)$
    $F_2 := \mathsf{Chase}_{\mathsf{BackView}'(\mathcal{V})}(F_1)$
    $G_2 := \mathsf{Chase}_{\Sigma'}(F_2)$
    if $\exists h : Q' \to G_2$ mapping each free variable $v$ of $Q'$ into $c_v \in \mathsf{adom}(\mathsf{canondb}(Q))$ then
        return true
    end if
    $F_3 := \mathsf{Chase}_{\mathsf{ForView}'(\mathcal{V})}(G_2)$
    $F_4 := \mathsf{Chase}_{\mathsf{BackView}(\mathcal{V})}(F_3)$
    $G_4 := \mathsf{Chase}_\Sigma(F_4)$
    $G_5 :=$ restrict $G_4$ to the original signature
    if $F_0 \neq G_5$ then
        $F_0 := F_0 \cup G_5$
    else
        return false
    end if
end while
```

As with Algorithm 1, it is a straightforward exercise to see that this algorithm correctly checks determinacy:

Theorem 8. *(Benedikt, ten Cate, and Tsamoura 2016) Algorithm 2 returns true if and only if $\mathcal{V}$ determines Q relative to Σ.*

Note that the chase may not terminate, thus this is only a semi-decision procedure.

As before, we have a homomorphism from the output of the algorithm to its input:

Lemma 6. *For any CQ Q, views $\mathcal{V}$ and number l, there exists a homomorphism $\nu : \mathsf{UnPrime}(F_2^l(Q, \mathcal{V})) \to \mathsf{canondb}(Q)$ that is the identity on elements c_v. In particularly, if Algorithm 2 returns **true**, there is a homomorphism from $F_2^\infty(Q, \mathcal{V})$ to $\mathsf{canondb}(Q)$ that is the identity on elements c_v.*

We have the same relation of the output of the algorithm when run on the canonical view to the output when run on Q:

Lemma 7. *For any $l \geq 1$ the source s atoms in $F_0^l(Q, \mathcal{V})$ are the same as the atoms of $F_0^l(\mathsf{CanView}^s(Q), \mathcal{V})$.*

We can now complete the proof of Theorem 4 as in the case without constraints. Locality of constraints ensures that the homomorphism h_1 maps each variable x in $\mathsf{SJVars}(s)$ to some element of the form c_v.

Proof of Theorem 5

Recall the statement:

For any set of local existential rules Σ, the Σ-invariant shuffle view of Q are minimally useful within the class of all views, relative to Σ.

We can generalize the ECR global Q-equivalence to global Q-Σ-equivalence, looking only at contexts that satisfy Σ. Using the same argument we see that the views corresponding to this ECR is minimally useful for Q relative to Σ.

Let σ be a mapping of $\mathsf{SJVars}(s)$ into some instance $\mathcal{I}$. A shuffle μ of $\mathsf{CanView}^s(Q)$ is *Σ-invariant relative to* $\langle \sigma, \mathsf{CanCtxt}^s(Q) \rangle$ if whenever $\mathcal{I}', \sigma \models \mathsf{CanCtxt}^s(Q)$ and $\mathcal{I}'$ satisfies Σ then $\mathcal{I}', \sigma \models \mu(\mathsf{CanCtxt}^s(Q))$. Note that since the rules are local, the notion of a context satisfying them is well-defined. Invariance is decidable whenever query containment for CQs under Σ is decidable; for example, this is the case when Σ is a set of dependencies with terminating chase.

Fixing Σ and two s-instances $\mathcal{I}_1$ and $\mathcal{I}_2$, we say that $\mathcal{I}_1$ and $\mathcal{I}_2$ are *Σ-invariant shuffle equivalent* if whenever $\mathcal{I}_1, \sigma$ satisfies $\mathsf{CanView}^s(Q)(\mathbf{x})$ then there is some shuffle μ which is Σ-invariant relative to $\langle \sigma, \mathsf{CanView}^s(Q) \rangle$, such that:
$\mathcal{I}_2, \sigma \models \mu(\mathsf{CanView}^s(Q))(\mathbf{x})$
and vice versa.

We can now extend Proposition 10, following the same proof:

Proposition 12. *Suppose Σ consists of local existential rules. Then for all instances satisfying the rules Σ-invariant shuffle equivalence is identical to global Q-Σ equivalence.*

Proof. First, suppose $\mathcal{I}_1$, $\mathcal{I}_2$ satisfy all local rules and are Σ-invariant shuffle equivalent. Consider a context C that satisfies local rules Σ and suppose we have a match of Q in $(\mathcal{I}_1, C)$ via $h^{1,C}$. We want to show that there is a match in $(\mathcal{I}_2, C)$. This will be exactly as in the case without constraints.

We know that the variables in $\mathsf{SJVars}(s, Q)$ are mapped by $h^{1,C}$ into $\mathcal{I}_1$. Let h_0 be the restriction of $h^{1,C}$ to the variables of $\mathsf{SJVars}(s, Q)$. Then $\mathcal{I}_1, h_0$ satisfies $\mathsf{CanView}^s(Q)$. Thus by Σ-invariant shuffle equivalence there is a shuffle μ which is Σ-invariant relative to $\langle h_0, \mathsf{CanView}^s(Q) \rangle$, such that $\mathcal{I}_2, h_0 \models \mu(\mathsf{CanView}^s(Q))$, with witness h_2 extending h_0. We also know that C, h_0 satisfies $\mathsf{CanCtxt}^s(Q)$, since $h^{1,C}$ witnesses this as well. Applying the definition of Σ-invariance, and noting that C satisfies Σ by assumption, we infer that C, h_0 satisfies $\mu(\mathsf{CanCtxt}^s(Q))$. Let $h^{\mu,C}$ be a homomorphism witnessing this. Note that since $h^{\mu,C}$ extends h_0 and h_0 restricts $h^{1,C}$, $h^{\mu,C}$ and h_0 agree on their common variables. Define $h^{2,C}$ by mapping the variables in $\mathsf{SVars}(s, Q)$ as in h_2, and those variables outside of $\mathsf{SJVars}(s, Q)$ as in $h^{\mu,C}$. Since these are two compatible homomorphisms, $h^{2,C}$ witnesses that $(\mathcal{I}_2, C) \models Q$. This completes the argument that Σ-invariant shuffle equivalence implies Q-Σ-equivalence.

We now show that global Q-Σ-equivalence implies Σ-invariant shuffle equivalence. Suppose s-instances $\mathcal{I}_1, \mathcal{I}_2$ are Q-Σ-equivalent, and $\mathcal{I}_1, \sigma$ satisfies $\mathsf{CanView}^s(Q)(\mathbf{x})$, We will show that there is a shuffle μ, Σ-invariant relative to $\langle \sigma, \mathsf{CanCtxt}^s(Q) \rangle$ such that $\mathcal{I}_2, \sigma \models \mu(\mathsf{CanView}^s(Q))(\mathbf{x})$.

Let C_1 be the context defined in two steps. We first proceed as in the case without constraints: for each source s other than s, we have a fact for each s atom of Q, where each variable x of $\mathsf{SJVars}(s, Q)$ is replaced by $\sigma(x)$ and each variable x not in SJVars is replaced by a fresh element c_x. In the second step, we perform the chase construction with Σ to get an instance that satisfies the local constraints.

Q clearly holds in $(\mathcal{I}_1, C_1)$. So by Q-Σ-equivalence, Q holds in $(\mathcal{I}_2, C_1)$ via some homomorphism h. As before, the only elements shared between $\mathcal{I}_2$ and C_1 lie in the range of σ. Thus h must map the variables in $\mathsf{SJVars}(s, Q)$ to the image of σ. For each c in the image of σ, choose a variable v_c such that σ maps v_c to c. Let μ map any variable $x \in \mathsf{SJVars}(s, Q)$ to $v_{h(x)}$. Thus $\sigma(\mu(x)) = h(x)$.

We claim that μ is Σ-invariant relative to $\langle \sigma, \mathsf{CanCtxt}^s(Q) \rangle$. We show this by arguing that h is a homomorphism from $\mu(\mathsf{CanCtxt}^s(Q))(\sigma)$ to the chase under Σ of $\mathsf{CanCtxt}^s(Q)(\sigma)$. By definition of μ, we have for each atom $A(x_1 \ldots x_m, y_1 \ldots y_n)$ of $\mathsf{CanCtxt}^s(Q)(\mathbf{x})$,

$$A(h(x_1) \ldots h(x_m), h(y_1) \ldots h(y_n)) \ = \ A(\sigma(\mu(x_1)) \ldots \sigma(\mu(x_m)), h(y_1) \ldots h(y_n))$$

$A(h(x_1) \ldots h(x_m), h(y_1) \ldots h(y_n))$ is in $\mathsf{Chase}_\Sigma(\mathsf{CanCtxt}^s(Q)(\sigma))$ since h is a homomorphism into $(\mathcal{I}_2, C_1)$ and thus for facts in $\mu(\mathsf{CanCtxt}^s(Q))(\sigma)$, it must map into C_1.

Thus we conclude

$$A(\sigma(\mu(x_1)) \dots \sigma(\mu(x_m)), h(y_1) \dots h(y_n)) \in \mathsf{Chase}_\Sigma(\mathsf{CanCtxt}^s(Q)(\sigma))$$

This completes the proof that h is a homomorphism into the chase, and thus the proof that μ is Σ-invariant relative to $\langle \sigma, \mathsf{CanCtxt}^s(Q) \rangle$.

We next claim that $\mathcal{I}_2, \sigma \models \mu(\mathsf{CanView}^s(Q))$. The witness will again be the extension of σ that maps all variables in $\mathsf{SVars}(s,Q) - \mathsf{SJVars}(s,Q)$ via h.

Consider an atomic formula $A(x_1 \dots x_m, y_1 \dots y_n)$ of $\mathsf{CanView}^s(Q)$ where $\mathbf{x}$ are free variables of $\mathsf{CanView}^s(Q)$. That is,

$$A(\mu(x_1) \dots \mu(x_m), \mathbf{y})$$

is a generic atom of $\mu(\mathsf{CanView}^s(Q))$. We know that $A(h(x_1) \dots h(x_m), h(y_1) \dots h(y_n)))$ holds in $\mathcal{I}_2$, since h is a homomorphism into $\mathcal{I}_2$. Thus

$$A(v_{h(x_1)}, \dots, v_{h(x_m)}, h(y_1) \dots h(y_n)))$$

holds of σ in $\mathcal{I}_2$, by definition of v_c. From this we see that

$$A(\mu(x_1) \dots \mu(x_m), h(y_1) \dots h(y_n)))$$

holds of σ in $\mathcal{I}_2$ as required. □

Recall that the Σ-invariant shuffle views of Q for s and types τ are defined analogously to the case without local rules, as $\tau(\mathbf{x}) \wedge \bigvee_\mu \mu(\mathsf{CanView}^s(Q))$ where the disjunction is over Σ-invariant shuffles of τ.

The following result is proven exactly as in the case without background knowledge:

Proposition 13. *For any Boolean CQ Q, and any source s two s-instances are Σ-invariant shuffle equivalent if and only if they agree on each Σ-invariant shuffle view of Q for s.*

Putting the prior results together gives us the extension of Theorem 5.

The power of replication: proof of Theorem 6

Recall the statement:

If BCQ Q contains a relation of non-zero arity replicated across all sources then there are views that are useful for Q and UN non-disclosing for BCQ p if and only if there is no homomorphism of p to Q.

Note that the condition on p and Q can be restated as saying that Q does not logically entail p.

For notational simplicity, we keep the replication constraints implicit, assuming that the replicated predicates are named T in each source and Q refers to this "global" T.

One direction of the theorem is clear: if there is a homomorphism of p to Q and $\mathcal{V}$ are useful for Q, then $\mathcal{V}$ can not be UN non-disclosing for p, since on any instance where Q holds, the views will disclose p.

For the other direction, we show, as in the case without constraints, that there is a single d-view that is *Q-universal*: it works for any p such that there is no homomorphism from p to Q. We provide views that are not isomorphism-invariant, assuming that the active domain of instances is $Pair(\mathcal{N})$ defined below.

$Pair(\mathcal{N})$ is the set that contains $\mathcal{N}$ and is closed under pairing: when x and y are allowed, then so is (x,y), the ordered pair consisting of x and y. Note that all elements in $Pair(\mathcal{N})$ have a finite $\mathsf{pairheight}$, where $\mathsf{pairheight}$ is defined as follows: $\mathsf{pairheight}(x) = 0$ for x an integer of $\mathcal{N}$, and $\mathsf{pairheight}((x,y)) = max(\mathsf{pairheight}(x), \mathsf{pairheight}(y)) + 1$.

Given two instances $\mathcal{I}_1$ and $\mathcal{I}_2$ for the same schema, the synchronous product of $\mathcal{I}_1$ and $\mathcal{I}_2$ is the instance defined as follows:

- elements of the instance are pairs (x,y) with $x \in \mathcal{I}_1, y \in \mathcal{I}_2$.
- for each relation R, we have $R((x_1,y_1) \dots (x_n,y_n))$ holds exactly when $R(x_1, \dots x_n)$ holds in $\mathcal{I}_1$ and $R(y_1, \dots y_n)$ holds in $\mathcal{I}_2$

Note that the projection on the first component is a homomorphism of the product to instance $\mathcal{I}_1$ and projection on the second component is a homomorphism to $\mathcal{I}_2$.

By a *position* we mean a relation S and a number between 1 and the arity of S. That is, a position describes an argument of a relation. We assume that each variable of Q is associated with a unique integer index.

We consider the transformation Str on a d-instance that maps each local instance to its product with $\mathsf{canondb}(Q)$.

Note that $\mathsf{Str}(\mathcal{D})$ is over the same schema as $\mathcal{D}$ and that $\mathsf{Str}(\mathcal{D})$ validates the replication constraint. Furthermore we suppose that the domain of $\mathsf{canondb}(Q)$ is included in $\mathcal{N}$ which ensures that the minimal $\mathsf{pairheight}$ of elements in the relation T of $\mathsf{Str}(\mathcal{D})$ will be the minimal $\mathsf{pairheight}$ of elements in the relation T of $\mathcal{D}$ plus one.

Definition 11. *We define $\equiv_s$ on s-instances as the reflexive transitive closure of $\mathcal{R}$ where $\mathcal{R}$ is defined as $\mathcal{I}\ \mathcal{R}\ \mathcal{I}'$ when $\mathcal{I} = \mathsf{Str}(\mathcal{I}')$ for $\mathcal{I}, \mathcal{I}'$ two s-instances.*

Definition 12. *We define $\equiv_G$ on d-instances as the reflexive transitive closure of $\mathcal{R}$ where $\mathcal{R}$ is defined as $\mathcal{D}\ \mathcal{R}\ \mathcal{D}'$ when $\mathcal{D}' = \mathsf{Str}(\mathcal{D})$ for $\mathcal{D}, \mathcal{D}'$ two d-instances.*

Definition 13. *The ECR $\equiv$ is defined as $\mathcal{D} \equiv \mathcal{D}' \Leftrightarrow \bigwedge_{s \in \mathsf{Srcs}} \mathcal{D}_s \equiv_s \mathcal{D}'_S$.*

Proposition 14. *For two d-instances $\mathcal{D}$ and $\mathcal{D}'$, when $\mathcal{D} \vDash Q$ then $\mathcal{D} \equiv \mathcal{D}'$ if and only if $\mathcal{D} \equiv_G \mathcal{D}'$.*

Proof. Clearly $\mathcal{D} \equiv_G \mathcal{D}'$ implies $\mathcal{D} \equiv \mathcal{D}'$. Now let us suppose that $\mathcal{D} \equiv \mathcal{D}'$ with $\mathcal{D} \vDash Q$ and let us show that $\mathcal{D} \equiv_G \mathcal{D}'$.

Since $\mathcal{D} \equiv \mathcal{D}'$, we have, for each source s, there exists i_s such that $\mathcal{D}_s = \mathsf{Str}^{i_s}(\mathcal{D}'_s)$ or $\mathcal{D}'_s = \mathsf{Str}^{i_s}(\mathcal{D}_s)$, where the notation Str^i means iterating Str i times. On a d-instance $\mathcal{D}$ where the replicated relation T is not empty, the minimal $\mathsf{pairheight}$ of elements appearing in position $T[1]$ needs to be equal for all sources s. Therefore we cannot have distinct sources s and s' where $i_s \neq i_{s'}$. And we cannot have s and s' such that $\mathcal{D}_s = \mathsf{Str}^{i_s}(\mathcal{D}'_s)$ and $\mathcal{D}_{s'} = \mathsf{Str}^{i_{s'}}(\mathcal{D}_{s'})$ with $i_s, i_{s'} > 0$. Therefore, when the replicated predicate is not empty, $\mathcal{D} \equiv \mathcal{D}'$ implies $\mathcal{D} \equiv_G \mathcal{D}'$.

When $\mathcal{D} \vDash Q$ we have that the replicated predicate T is not empty (since it appears in Q) which proves that $\mathcal{D} \equiv_G \mathcal{D}'$. □

We now show that $\equiv$ is the view that we want.

Proposition 15. *The view corresponding to ECR $\equiv$ is useful for Q.*

Proof. Let $\mathcal{D}$ and $\mathcal{D}'$ be two d-instances and let us suppose $\mathcal{D} \equiv \mathcal{D}'$ and $\mathcal{D} \vDash Q$.

By Proposition 14, we have that $\mathcal{D} \equiv_G \mathcal{D}'$. Recall that there is a homomorphism from $\mathsf{Str}(\mathcal{D})$ to $\mathcal{D}$. Thus it is clear that if $\mathsf{Str}(\mathcal{D}) \models Q$ it must be that $\mathcal{D} \models Q$. On the other hand if we have any match h of Q in $\mathcal{D}$, we can extend it to a match of Q in the product by taking any variable x of Q to $(h(x), c_x)$ where c_x is the constant corresponding to x in $\mathsf{canondb}(Q)$. From $\mathcal{D} \equiv_G \mathcal{D}'$ either $\mathcal{D}$ or $\mathcal{D}'$ can be obtained from the other by applying Str, and so $\mathcal{D} \models Q$ implies $\mathcal{D}' \models Q$. □

Proposition 16. *The view corresponding to ECR $\equiv$ is UN non-disclosing for p.*

Proof. Given an instance $\mathcal{D}$, we know that $\mathsf{Str}(\mathcal{D})$ has a homomorphism into $\mathsf{canondb}(Q)$. Therefore there is no homomorphism from p into $\mathsf{Str}(\mathcal{D})$ because if there were, we would have a homomorphism from p into $\mathsf{canondb}(Q)$, a contradiction of the assumption that Q did not entail p. □

Proof of Proposition 7

Recall the statement:

There is a d-schema with a replication constraint, along with Boolean CQs Q and p such that there is a d-view which is useful for Q and UN non-disclosing for p, but there is no d-view whose view definitions return only facts containing values in the active domain and which commute with isomorphisms.

For a query Q_V that always returns tuples containing only values in the active domain of the input, we will use the following restricted version of the isomorphism-invariance property:

If $\mathcal{I}$ and $\mathcal{I}'$ are source instances for Q_V with $\mathcal{I}'$ formed from $\mathcal{I}$ via an isomorphism that is the identity on $\mathsf{adom}(V(I))$, then $\mathcal{I}$ and $h(\mathcal{I})$ agree on Q_V.

For example, any query in relational algebra has this property.

Consider the following Boolean CQs (existentially quantifiers omitted):

$$Q = P(x) \wedge P(y) \wedge S(y) \wedge S(z) \wedge T(z) \wedge T(x) \wedge R(w)$$

$$p = S(x) \wedge T(x) \wedge P(x)$$

It is easy to see that Q does not entail p. Because there is a replicated relation mentioned in the query, Theorem 6 implies that we can get a d-view that is useful for Q and UN non-disclosing for p.

Now, by way of contradiction, fix a d-view $\mathcal{V}$ that is useful for Q, returns only facts containing only values in the active domain, and satisfies the isomorphism-invariance property.

Proposition 17. *Suppose d-view $\mathcal{V}$ is useful for Q. Given two instances $\mathcal{I}_1$ and $\mathcal{I}_2$ for the source whose unreplicated relation is G, if $\mathcal{I}_1$ and $\mathcal{I}_2$ agree on the replicated relation and agree on each view in $\mathcal{V}$ for this source, then they must agree on G.*

Proof. Let $c \in G(\mathcal{I}_1)$ and form $\mathcal{D}_1$ by choosing a context for the other unshared relations with each relation containing only c. Let $\mathcal{D}_2$ be formed similarly from $\mathcal{I}_2$. Then $\mathcal{D}_1$ and $\mathcal{D}_2$ agree on all views in $\mathcal{V}$. There is a match of Q on $\mathcal{D}_1$ so the same must be true of $\mathcal{D}_2$, since $\mathcal{V}$ is usful for Q. Clearly the witness can only be c, thus $c \in G(\mathcal{I}_2)$. □

By way of contradiction, we fix a d-view $\mathcal{V}$ that is useful for Q and UN non-disclosing for p, where each view in $\mathcal{V}$ returns facts containing only elements of the active domain and satisfies the isomorphism-invariance property.

Let $\mathcal{D}_0$ be the critical instance, recalling that this has only a single element $*$ with every relation holding of it. We show that $\mathcal{D}_0$ contradicts that $\mathcal{V}$ is UN non-disclosing for p. Note that $\mathsf{adom}(\mathcal{V}_0) \subseteq \{*\}$ since we assume views return facts containing only elements in the active domain of the input instance.

Proposition 18. *Let $\mathcal{D}$ agree with $\mathcal{D}_0$ on $\mathcal{V}$. For any unreplicated relation G, we have $\mathcal{D} \models \forall x.((G(x) \wedge x \neq *) \rightarrow R(x))$.*

Proof. Let $\mathcal{I}$ be the restriction of $\mathcal{D}$ to the G source, and suppose $G(v)$ holds in $\mathcal{I}$ for $v \neq *$. Let c be a fresh value and $\mathcal{I}'$ be the result of applying an isomorphism swapping v and c. Since $v \neq *$ the isomorphism-invariance property implies that $\mathcal{I}$ and $\mathcal{I}'$ agree on the views. Since $\mathcal{I}'$ and $\mathcal{I}$ disagree on G, Proposition 17 implies they cannot agree on the replicated relation. Note that c was fresh (hence in particular was not in the interpretation of R within $\mathcal{I}$), and no other change occurred in R outside of the swap of v and c, the only way that $\mathcal{I}$ and $\mathcal{I}'$ can disagree on R is because $R(v)$ held in $\mathcal{I}$. □

By usefulness of $\mathcal{V}$ and the fact that $\mathcal{D}_0$ has a match of Q, we know that $\mathcal{D}$ has a match of Q with x_0, y_0, z_0, w_0. We first consider the case where two of x_0, y_0, z_0 are the same. In this case p has a match.

Now suppose the match has all of x_0, y_0, z_0 distinct. At most one of them can be $*$, so assume y_0 and z_0 are not $*$. Proposition 18 implies that they are both in R. Since y_0 and z_0 are not in $\mathsf{adom}(\mathcal{V}_0)$, when we consider the result of swapping y_0 and z_0, it does not impact the views, by the isomorphism-invariance property. Since this swap also does not change R, we can apply Proposition 17 to conclude that all local sources agree on y_0 and z_0. In particular we have $P(y_0) \wedge S(z_0)$, thus we have $P(z_0) \wedge S(y_0)$. But then either y_0 or z_0 gives a match of p.