

Balancing expressiveness and inexpressiveness in view design

MICHAEL BENEDIKT AND PIERRE BOURHIS AND LOUIS JACHET AND EFTHYMIA TSAMOURA

We study the design of data publishing mechanisms that allow a collection of autonomous distributed data sources 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 restrictions. 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 expressiveness and inexpressiveness requirements, for views in a restricted declarative language (conjunctive queries), and for arbitrary views.

ACM Reference Format:

Michael Benedikt and Pierre Bourhis and Louis Jachiet and Efthymia Tsamoura. 2019. Balancing expressiveness and inexpressiveness in view design. *ACM Trans. Datab. Syst.* 1, 1, Article 1 (January 2019), 45 pages. <https://doi.org/10.1145/3365834>

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 data sources.

But the benefits of data publishing come with a corresponding risk of revealing too much. For example, there 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.

Author's address: Michael Benedikt and Pierre Bourhis and Louis Jachiet and Efthymia Tsamoura.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.

0362-5915/2019/1-ART1 \$15.00

<https://doi.org/10.1145/3365834>

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. And the focus has been on *analysis of a given set of views*: we have in front of us some view definitions, and we want to see what properties they satisfy. This is the case in particular for the recent works [7–9]. 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. [12, 21]. 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 cannot 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 cannot be computed from the views on any source data. Variants of data-independent privacy have been studied in [7–9, 27]. Expressiveness restrictions with a similar flavor have also been studied in the context of ontologies [11]. But the question of whether there are expressiveness restrictions for views that do not require the specification of a particular set of secrets, 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.

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. [13, 16, 17]). The guarantees are probabilistic, sometimes alternatively or additionally with computational restrictions on an external party. Recent efforts [24] have considered a family of mechanisms that look at database queries, with the utility of a mechanism defined (as in our case) using the notion of query determinacy. But randomness still plays a central role in the mechanism and in the definition of privacy.

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 to both the difference in the mechanisms we consider and the requirements we impose, our contribution has a very different flavor from prior lines of work. It also differs from work on secure querying over distributed data [6]. There, the goal is to support ad hoc querying of traditional database queries in the presence of privacy restrictions; but they allow the use of encryption as a query language primitive.

Example 1.1. A health study hosted by a government agency holds information about certain treatments, with an abstraction of the data being a database with schema `Treatment(pid, tinfo, tdate)`, where `pid` is a national insurance number.

Demographic information about patients is stored by another agency, in a table `Patient(pid, age, address)`. The agencies are completely autonomous, perhaps even in

distinct administrative regions. But they want to co-operate to support certain queries over the data that legitimate researchers might wish; for example about the relationships between treatment and age:

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

Of course, the parties could agree to an encryption scheme on the patient identifiers, and then expose encrypted versions of their local data. 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 Treatment table on pid and tinfo, and the other revealing the projection of the Patient table on pid and age.

If this intuition is correct, it would imply that nothing the parties can do with views based on 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

$$p = \exists \text{tdate}. \text{Treatment}(\text{pid}, \text{tinfo}, \text{tdate})$$

Our results will validate this obvious answer – in this example, the two projections of the Patient table described above – are the CQ views that reveal minimal information, and one cannot support the disclosure of the query Q that is joining on attribute pid, while protecting the information concerning the exact pids that are joined. Further we will show that even using encryption – indeed, even using any deterministic function – the parties cannot reveal less information while supporting exact answering of the query Q . ◀

Example 1.2. Consider a scenario with one source storing a geography-based linking relationship $\text{geolink}(x, y)$ between individuals x and y , and another storing a social media linking relationship $\text{soclink}(x, y)$ between individuals x and y . A third source provides information about individual income, which we abstract into a relation $\text{HighIncome}(x)$ identifying individuals x with high income.

Consider a query Q asking if there are high-income individuals who are linked geographically and via social media:

$$Q = \exists x, y. \text{geolink}(x, y) \wedge \text{soclink}(x, y) \wedge \text{HighIncome}(x) \wedge \text{HighIncome}(y)$$

One can ask what is the minimal information we can reveal in views from the three sources in order to allow the query Q to be answered. In this example, intuitively, it appears difficult to answer the query without revealing all information from the sources. We will see that our results enable us to formalize the notion of minimal information and validate this intuition. ◀

Our goal here is to look at the problem of designing independent views over multiple relational data sources that satisfy both expressiveness and inexpressiveness requirements. Our expressiveness requirement (or usefulness) will be phrased in terms of the ability to answer, using only the information exposed by the views, a relational query – the “utility query” that the parties want to support – where answering is the traditional deterministic notion used in database theory and knowledge representation.

For expressiveness limitations we will consider two different criteria: the first one asks to find useful views that *minimizes the information* within the useful views based on a particular class of queries.

Example 1.3. Recall the query $\exists \text{pid, address, tdate. Treatment}(\text{pid, tinfo, tdate}) \wedge \text{Patient}(\text{pid, age, address})$ from Example 1.1, and the views that return the projection of the Treatment table on pid and tinfo in one source, and the projection of the Patient table on pid and age in the other source. These views will be considered *useful for the query Q*, since they are sufficient to answer the query. We will also show that they reveal the minimal information among all useful views.

In Example 1.2, the views that reveal everything from each source – the “identity views” — are clearly sufficient to answer the query Q . An external party can just take the view images, which are just the sources themselves, and simply “join them” just as in the definition of Q , to answer the queries. Thus, the identity views are “useful for Q ” according to our definitions. Conversely, it will follow from our results that, in this example, the identity views actually reveal the minimal information among those which support the answering of Q . That is, any useful set of views for Q must reveal the entire content of each data source. \triangleleft

We can also consider a second expressiveness limitation specified by *non-disclosure* of a set of *secret queries*.

Example 1.4. We return again to Example 1.2. Suppose that we are tasked with finding views that are useful for Q but where our inexpressiveness condition is to avoid disclosing whether there are high-income individuals who are linked via a chain of three social media links.

In that example we have already conjectured that any views useful for the utility query Q must disclose the entire content of each source, and thus in particular they must disclose the content of the relations soclink and HighIncome. Thus it is intuitively clear that any useful views must disclose the secret.

In comparison, suppose that in Example 1.1 we want to keep secret the set of pairs of individuals having the same address. Here the views that we proposed project out the address information so, intuitively, it is possible to keep this information private.

In contrast to these two examples, we will show that in some cases the parties can obtain combinations of expressiveness and inexpressiveness requirements by using counter-intuitive view combinations. If the reader who wants to get an idea of how complicated such views may need to be, they can do so by looking ahead a bit in this paper. In Example 1.5, we will see that to achieve a balance we may require disjunction in the views, even though our secret query and utility query do not have disjunction. In Example 4.14, the optimal views will require either unsafe disjunction (different free variables in the different disjuncts) or the use of negation. And in the proof of Theorem 6.10 in Section 6 we present a scenario in which one needs to go beyond any standard relational query language to satisfy both an expressiveness and an inexpressiveness restriction. \triangleleft

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, in the form of database integrity constraints, on these problems.

Example 1.5. In Example 1.2 suppose background knowledge tells us that the geographic linking relationship is known to be symmetric. We formalize this as a background theory consisting of the following *integrity constraint*:

$$\forall x, y. \text{geolink}(x, y) \leftrightarrow \text{geolink}(y, x)$$

It is now possible to support the utility query while revealing less information. We can change the view on the soclink source to reveal

$$\text{soclink}(x, y) \vee \text{soclink}(y, x)$$

It is easy to see that an external party can still answer Q simply by joining the views. Just as clearly, less information is being revealed by this view, since an external party with an access to this views will not be able to distinguish the direction of a social link.

Of course, the reader will observe that we used disjunction in the view above: it is not a conjunctive query. We can show that the “obvious views” from Example 1.3 are still minimally informative within the set of *conjunctive query views*, even in the presence of this integrity constraint. Thus, we see that – at least in this example – the presence of integrity constraints impacts whether a set of views is minimally informative within the set of all views, but not within the set of CQ views.

Let us now consider the impact of this integrity constraint in terms of disclosure of secrets. An external observer can still learn whether there are high earners that are connected by three social links. They cannot learn this on *all* databases, since they may be unsure about the direction of a social link. But there are *some* valid instances where an external party with the access to the views will be able to detect this; this is true in an instance with a high-earner having a social link to themselves.

In fact, we will see that in this example the set of Boolean CQs that are disclosed is not impacted by the presence of these constraints. We therefore see a distinction between distributed views with minimal information and a narrower definition of minimality in terms of the Boolean CQs that are revealed. ◀

Contributions. We will not be able to present a full picture of the view design problem in this work. We will deal only with the case of utility and secret queries given as conjunctive queries, the analogs of SQL basic SELECTs, and we will study only a restricted class of source integrity constraints. But we believe our results suffice to show that our formulation captures an important trade-off in schema design, and that query language expressiveness issues come to the fore. In technical terms, we make the following contributions:

- We formalize the idea of balancing expressiveness and inexpressiveness in distributed views, via the notions of useful distributed views as well as “minimal information” requirements among these views.
- We show that there are useful views with minimal information among the set of CQ views, and also among the set of all views, but that these may differ.
- In contrast to the above, we show that to obtain useful views with minimal information we do not need to go that far beyond CQ views: it suffices to use views defined either using unsafe disjunction of CQ views, or in relational algebra.

- We show that the above results extend to the presence of background knowledge that is local to each source.
- We examine the impact of background knowledge that relates multiple sources, looking at the simplest kind of relationship, the replication of a relation across sources. We show that we may no longer have useful views with minimal information, but we can get useful views that are minimal in terms of the secrets they disclose. In the process we show that replication can be exploited to allow a view designer to reveal certain queries while not disclosing others.

A diverse set of techniques are applied to study these problems. For investigating the use of CQ views to satisfy both expressiveness and inexpressiveness restrictions, we employ an analysis of the chase proofs that witness determinacy. For studying the use of arbitrary views, both in the absence of background knowledge and in the presence of only local background knowledge, we use a Myhill-Nerode style characterization of when two local sources are interchangeable in terms of their impact on a utility CQ, and then we relate this characterization to certain partial symmetries of the utility queries (“shuffles”). Our analysis of the impact of replication constraints relies on a product construction, which is the key to allowing us to generate useful views that disclose the minimal number of secrets.

Organization. Section 2 gives database and logic preliminaries, and then goes on to formalize our expressiveness and inexpressiveness requirements. 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 examines how the situation changes when background knowledge local to a source is present; this is the case of the integrity constraints in Example 1.5. Section 6 consider background knowledge that can connect relations across distinct sources. We close with a discussion of how to interpret the results, future directions, and open questions in Section 7.

2 PRELIMINARIES

2.1 Basic definitions

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

Databases and queries. A *schema* consists of a finite set of relation names, each with an associated arity (a non-negative number). An *instance* of a schema is an assignment of each relation name R in the schema of arity n to a collection (finite or infinite) of n -tuples of elements. We say that the collection is the *interpretation* of R in the instance. Given an instance \mathcal{I} and a relation name R , let $\mathcal{I}(R)$ be the n -tuples assigned to R in \mathcal{I} . The *active domain* of an instance \mathcal{I} , denoted $\text{adom}(\mathcal{I})$, is the set of elements occurring in some tuple of some $\mathcal{I}(R)$. A *fact* over an instance \mathcal{I} consists of a relation name R of arity n and an n -tuple \mathbf{t} of values $t_1 \dots t_n$ from the active domain of \mathcal{I} . We write such a fact as $R(\mathbf{t})$, and also say that it is a *fact over R* or an *R -fact*. An instance \mathcal{I} is determined by the set of facts over \mathcal{I} , so it 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. Fix once and for all a set of *variables* $x_1 \dots$ and *constants* $c_1 \dots$. An *atom* over a schema is an expression $R(u_1 \dots u_n)$ where R is an

n -ary relation name over the schema and each u_i is either a variable or a constant. An atom $R(u_1 \dots u_n)$ is an *atom over R* , and similarly we talk of “facts over R ”. A *position* is a pair (R, i) consisting of a relation name R of some arity n and a number i between 1 and n . A position thus represents an argument of a fact or atom over R . We will also talk about a position of relation name R , meaning a pair (R, i) as above.

A *Conjunctive Query* (CQ) over a schema S is a formula of the form $\exists y. \bigwedge_i A_i$ where A_i are atoms over the schema. We will often omit “over schema” when it is clear from context or unimportant, and just talk about a CQ. A Boolean CQ (BCQ) is a CQ with no free variables. Following the notation used in several places in the literature (e.g. [2]) 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). UCQs can be extended to *relational algebra*, the standard algebraic presentation of first-order relational queries: queries are built up from symbols for each relation name by union, difference, selection, and product [1].

For a logical formula ρ with free variables x_1, \dots, x_n and an instance I , a *variable binding* σ for ρ in I is a mapping taking each x_i to an element of $\text{adom}(I)$. 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, \dots, x_n we may identify a variable binding with a k -tuple \mathbf{t} . In particular, when a certain ordering is assumed we will write $\rho(\mathbf{t})$ to mean that t_i is substituted for x_i in ρ .

A *homomorphism* between instances I_1 and I_2 is a function f from $\text{adom}(I_1)$ to $\text{adom}(I_2)$ such that for all relation names R and values $c_1 \dots c_m$ in $\text{adom}(I_1)$, $R(c_1, \dots, c_m) \in I_1$ implies $R(f(c_1), \dots, f(c_m)) \in I_2$.

The *canonical database* of a CQ Q , denoted $\text{canondb}(Q)$, is the instance whose facts are the atoms of Q , where each variable v corresponds to an element c_v . The notion of homomorphism from a CQ Q to an instance I is just a homomorphism from $\text{canondb}(Q)$ to I . A homomorphism from a CQ Q to a CQ Q' is just a homomorphism between their canonical databases, where we additionally require that the mapping be the identity on any free variables or constants. The *output* of a CQ Q on an instance I , denoted $Q(I)$ consists of the restrictions to free variables of Q of the homomorphisms of Q to 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 I consists of n -tuples. We write $I, \mathbf{t} \models Q$ for an n -tuple \mathbf{t} , if $\mathbf{t} \in Q(I)$. We analogously define $I, \sigma \models Q$ for a variable binding σ , and in this case we also say I, σ *satisfies* Q . A CQ with n free variables thus defines an n -ary query, and similarly for a UCQ, and more generally any logical formula with n free variables. We sometimes refer to a homomorphism of a BCQ into an instance as a *match*.

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. A CQ Q is *minimal* if there is no homomorphism from Q to a strict subquery of Q .

A *view* over a schema S consists of an n -ary relation name V and a corresponding n -ary query Q_V over relation names from S . Given a collection of views \mathcal{V} and an instance I , the *view-image* of I , denoted $\mathcal{V}(I)$ is the instance that interprets each

$V \in \mathcal{V}$ by $Q_V(I)$. 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 local schemas are pairwise disjoint. In Example 1.1, our distributed schema consisted of two sources, one containing Treatment and the other containing Patient. In Example 1.2, our distributed schema consisted of three sources: one containing the list of high-income individuals, one containing the geographical links and one containing social media relationships. We often identify a d-schema with the schema formed by unioning the local schemas. A *distributed instance* (d-instance) is an instance of the distributed schema, where here, in line with the convention above, we mean the union of the local schemas. For a source s , an s -instance is an instance of the local schema \mathcal{S}^s . Given a d-instance \mathcal{D} , we denote by \mathcal{D}^s the restriction of \mathcal{D} to relation names in s . If d-schema \mathcal{S} is partitioned into d-schemas \mathcal{S}_1 and \mathcal{S}_2 , and we have d-instances \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} . A query over a d-schema is simply a query over the schema formed by taking the union over of relations in all local schemas.

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 the queries defining each view cannot span multiple sources. A CQ-based d-view is a d-view where the set of views on each source is composed of CQs. We can similarly talk about relational algebra-based d-views, etc.

Tuple-generating dependencies and entailment under constraints. We want to consider the impact on disclosure problems of background knowledge. In databases, background knowledge is often modeled using *integrity constraints*, logical sentences that are assumed to hold on sources. One common class of integrity constraints are *Tuple Generating Dependencies* (TGDs), which we now review. These are 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 an instance \mathcal{I} and a variable binding σ (or \mathcal{I}, σ *satisfying* ρ , written $\mathcal{I} \models \rho$) is the standard one in first-order logic. It is consistent with the notion we gave for CQs above. 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. Note that a dependency τ is satisfied in \mathcal{I} if there is no active trigger for τ in \mathcal{I} .

Let Σ be a set of TGDs, \mathcal{I} be a finite 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 BCQs Q and Q' , we similarly write $Q \wedge \Sigma \models Q'$ to mean that every instance satisfying Q and Σ satisfies Q' . We also say that Q *entails* Q' *under* Σ . The terminology extends to CQs by talking about every instance and variable binding. If Σ is absent we just say Q entails Q' . Entailment between CQs is easily seen to be equivalent to existence of a homomorphism from Q' to Q . Given a set of TGDs Σ and another TGD σ_0 , we similarly write $\Sigma \models \sigma_0$ or Σ entails σ_0 to mean that any instance satisfying Σ also satisfies σ_0 . The terminology extends in the obvious way to other kinds of logical sentences that have a semantics on instances.

The chase. The results in Section 3 will make use of the characterization of logical entailment between CQs in the presence of TGDs in terms of the chase procedure [18, 25] which we review here. The chase modifies an instance by a sequence of *chase steps* until all dependencies are satisfied. Let \mathcal{I} be an instance, and consider a TGD $\tau = \forall \mathbf{x}.\lambda \rightarrow \exists \mathbf{y}.\rho$. Let h be 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 fresh labeled null that does not occur in \mathcal{I} .

For Σ a set of TGDs and \mathcal{I} an instance a *chase sequence* for Σ and \mathcal{I} is a (possibly infinite) sequence $\mathcal{I}_0, \mathcal{I}_1, \dots$ such that $\mathcal{I} = \mathcal{I}_0$ and, for each \mathcal{I}_i with $i > 0$ 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 $\text{Chase}_\Sigma(\mathcal{I})$ 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}_∞ for Σ and \mathcal{I} and check if Q holds [18]. We can similarly test if $Q \wedge \Sigma \models Q'$ by chasing $\text{canondb}(Q)$ with Σ and then checking Q' . Checking if a set of dependencies Σ has terminating chase is undecidable [15]. *Weak acyclicity* [18] 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 [14, 29].

2.2 Problem formalization

We now give the key definitions in the paper, capturing our expressiveness requirements (“useful”) and expressiveness limitations (“minimally informative” and “non-disclosing”). Our expressiveness requirement is via the concept of determinacy [28], formalizing the idea that on any instance there is sufficient information in the views to recapture the query.

Definition 2.1. Two d-instances \mathcal{D} and \mathcal{D}' are *indistinguishable* by a d-view \mathcal{V} (or just \mathcal{V} -indistinguishable) if $V(\mathcal{D}) = V(\mathcal{D}')$ for each view $V \in \mathcal{V}$.

Since each view $V \in \mathcal{V}^s$ is defined only using relation names occurring in \mathcal{V}^s , we can equivalently say that \mathcal{D}_1 and \mathcal{D}_2 are \mathcal{V} -indistinguishable if $V(\mathcal{D}_1^s) = V(\mathcal{D}_2^s)$ for each $V \in \mathcal{V}^s$ for each source s .

Definition 2.2. A d-view \mathcal{V} *determines* a query Q at a d-instance \mathcal{D} if $Q(\mathcal{D}') = Q(\mathcal{D})$ 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 2.3 (Minimally informative useful views). Given a class of views C and a query Q , we say that a d-view \mathcal{V} is a *minimally informative useful d-view for Q within C* if \mathcal{V} is useful for Q and, for any other d-view \mathcal{V}' useful for Q based on views in 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 2.4 (Non-disclosure). A *non-disclosure function* specifies, for each query p , a 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 , then we say that \mathcal{V} is *non-disclosing for p* (relative to the given non-disclosure function).

When we can find minimally informative useful d-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 minimally informative views, assuming they exist:

PROPOSITION 2.5. *Suppose \mathcal{V} is a minimally informative useful d-view for Q within C , and there is a d-view based on views in 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 non-disclosure functions that are determinacy-compatible. But in our examples, our complexity results, and in Section 6, we will focus on a specific non-disclosure function, whose intuition is that an external party “never infers any answers”.

Definition 2.6 (UN Non-disclosure). 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, the d-view \mathcal{V} is said to be *UN disclosing for p* .

This non-disclosure function is clearly determinacy-compatible, so Proposition 2.5 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 informative.

Variations: background knowledge, and finite instances. All of these definitions can be additionally parameterized by background knowledge Σ , consisting of integrity constraints in some logic. We will refer to an arbitrary finite set of such sentences as a *theory* or a *set of integrity constraints*. 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. The advantage of dealing with the unrestricted variant of the problem is that for views and queries in relational algebra the determinacy

problem is semi-decidable, and is equivalent to the standard notion of rewritability in relational algebra [28]. In contrast, the finite version is not semi-decidable even for conjunctive queries and views [20]. The use of the unrestricted version will also allow us to make use of the chase construction, which will be convenient in Section 3. However, it will turn out that the main results in the paper do not depend on this design choice, and hold for either version of the definition: see Section 7 for further discussion.

Main problem. We focus on the problem of determining whether minimally informative useful d-views exist for a given query Q and class C , and characterizing such views when they do exist. When minimally informative 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 to p as the *secret query*. We investigate this problem both in the presence and the absence of integrity constraints.

Example 2.7. In Example 1.2 our scenario involved the utility query $\exists x, y. \text{geolink}(x, y) \wedge \text{soclink}(x, y) \wedge \text{HighIncome}(x) \wedge \text{HighIncome}(y)$. Our results will validate the intuition stated there, that the minimally informative useful d-view, in the absence of constraints, is one which simply reveals the entire database! Our results will also allow us to formalize the intuition concerning the impact of the symmetry constraint in Example 1.5: to reveal minimal information, we no longer return the entire instance, but utilize a disjunction on the soclink source. From these observations, we will easily conclude that even in the presence of the symmetry constraint, any view that is useful for this query must reveal the answer to the secret query asking high-income individuals who are linked via a chain of three social media links. \triangleleft

Discussion of utility and non-disclosure definitions. Our formalization of utility for 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), without limiting the computation. 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 simple views.

Our formalization of minimally informative views 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 minimally informative useful views on protecting information, and it is also based on information-theoretic and exact criteria. We exemplify our general definition of non-disclosure function with UN non-disclosure, which has been studied in prior work under several different names [7–9, 27]. 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 [17].

The reader may have noted that UN non-disclosure only deals with protecting *positive information* about a secret query p . The attacker can still infer that the query is *not* true on an instance. One can modify the definition in the obvious way to deal with negative information. One still obtains a valid non-disclosure function, and all of our results up until those in Section 6 will still hold for it, other than those referring to complexity. But notice that this enhancement of UN non-disclosure is impossible to achieve in the absence of integrity constraints, or even if there are integrity constraints but only TGDs:

the empty instance is always a counterexample! This variant thus becomes interesting only when more general integrity constraints are allowed. The analysis of a given set of views for this variant of UN non-disclosure is investigated in [8].

On the one hand the UN non-disclosure guarantee, along with its variant with negative information above, is weak in that p is considered safe for \mathcal{V} (UN non-disclosed) if an attacker can never infer something about p with absolute certainty. Given a distribution on source instances, the information in the views may still increase the likelihood that p holds. On the other hand, UN non-disclosure and its variant above are quite strong in that they must hold on *every* valid source instance. As with the positive/negative distinction discussed a paragraph above, we can attempt to address both of these issues by strengthening or weakening the definition to include a more fine-grained quantitative analysis. For example we can say that a secret p is not disclosed if, for each number k , there are at most a quarter of the instances of size k where an external party can be sure that p is true. This is once again a valid non-disclosure function, and all of our expressiveness results for such functions will apply to it. However, we do not know how to analyze such complex functions. And even to see whether these quantitative notions of non-disclosure properties hold in particular examples is a challenging problem, outside the scope of this work.

Thus, although we do not by any means claim that UN non-disclosure 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. We also make use of it as a tool to understand minimal information.

Restrictions and simplifications. Although the utility and non-disclosure definitions above make sense for any queries, *for simplicity, in the remainder of this paper we will assume that Q and p are BCQs without constants.* We will thus abuse notation by referring to a BCQ without constants as simply a BCQ. Our results generalize easily in the presence of 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 $\text{SVars}(s, Q)$ the variables of Q that appear in an atom from source s . We also denote by $\text{SJVars}(s, Q)$ the “source-join variables of s ” in Q : the variables in $\text{SVars}(s, Q)$ and that also occur in an atom of another source.

Definition 2.8. The *canonical view of Q* for source s , $\text{CanView}^s(Q)$, has a view definition formed by conjoining all s -source atoms in Q and then existentially quantifying all bound variables of Q in $\text{SVars}(s, Q) \setminus \text{SJVars}(s, Q)$. The *canonical d-view of Q* is formed by taking the canonical view for each source.

In Example 1.1, the canonical d-view is what we referred to as “the obvious view design”. It would mean that one source has a view exposing $\exists \text{tdate}.\text{Treatment}(\text{pid}, \text{tinfo}, \text{tdate})$ – since pid is a source-join variable while tinfo is a free variable of Q . The other source should expose a view revealing $\exists \text{address}.\text{Patient}(\text{pid}, \text{age}, \text{address})$, since address is neither a free variable nor shared across sources.

The critical instance. In the definition of UN non-disclosure of a query by 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 [26].

Definition 2.9. The *critical instance* of a schema S is the instance whose active domain consists of a single element $*$ and whose facts are $R(*, \dots, *)$ for all relation names R in S .

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

THEOREM 2.10. [8, 9] *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. The same holds when these definitions are relativized to a theory consisting of TGDs.*

Note that this theorem is the only significant result that we take from [8, 9], and that it will be used only for understanding examples and in Section 6.

3 BALANCING EXPRESSIVENESS AND INEXPRESSIVENESS WITH CQ VIEWS

We now begin our analysis of balancing expressiveness and inexpressiveness. Returning to Example 1.1, recall the intuition that the canonical d-view of Q is the “least informative d-view” that supports the ability to answer Q . We would thus expect the canonical d-view to be a minimally information useful d-view as defined in Section 2. We start by proving such a result, but with two restrictions: the utility query Q must be a minimal CQ, and we only consider views specified by CQs:

THEOREM 3.1. [Minimally informative useful CQ views] *For every minimal BCQ Q , the canonical d-view of Q is minimally informative within CQ-based d-views. That is, if any CQ-based d-view \mathcal{V} determines a Q , then \mathcal{V} determines each canonical view $\text{CanView}^s(Q)$ of Q .*

We present a proof of this result, making use of the chase construction reviewed in Section 2. We note that this result will also follow from a more general theorem presented in Subsection 5.2, which will rely on techniques presented later concerning arbitrary views. All proofs make use of the following property of minimal CQs, which is easy to verify:

LEMMA 3.2. *If Q is a minimal CQ, then there is no homomorphism h from Q into itself that maps two different variables occurring in Q to the same variable.*

We will begin the chase-based 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 BCQ and \mathcal{V} be an arbitrary set of CQ-views. 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 name 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 name 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 name by its primed counterpart.

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

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

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

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

For any CQ views \mathcal{V} , let $\Sigma_{\mathcal{V}}$ denote the set of TGDs consisting of the forward and inverse view definitions for views in \mathcal{V} , as well as the forward and inverse view definitions but for primed copies of the base predicates.

Example 3.3. 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 $\Sigma_{\mathcal{V}}$ contains the TGDs:

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

◀

It is easy to see that, for any CQ views \mathcal{V} and any CQ Q , determinacy of Q by \mathcal{V} can be expressed as:

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

This is a containment of CQs under TGDs, and the chase algorithm checks this and is complete for such containments (see Section 2). Intuitively, the chase algorithm produces an infinite instance $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$, and then checks for a match of Q' . Note that, because it produces an infinite model, the chase algorithm might not terminate. But it always terminates when $Q \wedge \Sigma_{\mathcal{V}} \models Q'$. We summarize these observations with the following proposition:

PROPOSITION 3.4. *For each query Q and set of views \mathcal{V} , the following three statements are equivalent:*

- $Q \wedge \Sigma_{\mathcal{V}} \models Q'$;
- \mathcal{V} determines Q ;
- *there exists a homomorphism of $\text{canondb}(Q')$ into $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$ that is the identity on values c_v where v is a free variable of Q .*

The result is implicit in [28], and proofs can be found in [10, 19].

We are now ready for the proof of Theorem 3.1:

PROOF. Assume that \mathcal{V} determines Q and thus, from Proposition 3.4 there exists a homomorphism h_1 from $\text{canondb}(Q')$ into $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$, or more precisely,

from $\text{canondb}(Q')$ into the instance consisting of the primed facts of $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$. We can equivalently think of h_1 as having domain Q' , and we will use this perspective in the proof.

We now observe that the building of $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$ can be done independently on the different sources. Indeed with our autonomy assumption, we have that $Q \wedge \Sigma_{\mathcal{V}} = \bigwedge_s \text{CanView}^s(Q) \wedge \Sigma_{\mathcal{V}^s}$, where we recall that \mathcal{V}^s denotes the views in \mathcal{V} that apply to source s .

The chase process also decomposes by sources:

$$\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q)) = \bigcup_s \text{Chase}_{\Sigma_{\mathcal{V}^s}}(\text{canondb}(\text{CanView}^s(Q)))$$

Thus, the active domains of $\text{Chase}_{\Sigma_{\mathcal{V}^s}}(\text{canondb}(\text{CanView}^s(Q)))$ for the distinct sources s are all disjoint except for the values corresponding to source-join variables. As a consequence, for each value corresponding to a source-join variable c_v , $h_1(c_v)$ must be a value shared between different sources and thus h_1 must map source-join variables to values corresponding to source-join variables.

Looking ahead to our goal, we want to show that \mathcal{V} determines each canonical view $\text{CanView}^s(Q)$. Fix a source s_0 . Applying Proposition 3.4 again, we see that our determinacy goal is equivalent to finding a homomorphism of $\text{CanView}^{s_0}(Q')$ into $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{CanView}^{s_0}(Q))$, that is the identity on the source-join variables. The restriction of h_1 to variables in $\text{CanView}^{s_0}(Q)$ is thus *almost* enough to give us our desired conclusion, except that it may not be the identity on source-join variables. In order to fix this issue we will need to translate between homomorphisms like h_1 and a homomorphism from Q to itself. This will allow us to make use of the assumption that Q is a minimal CQ.

Let $\mathcal{D} = \text{canondb}(Q') \cup \text{canondb}(Q) \cup \mathcal{V}(\text{canondb}(Q))$. Note that the constraints of $\Sigma_{\mathcal{V}}$ hold in \mathcal{D} and thus we can assume $\text{Chase}_{\Sigma_{\mathcal{V}}}(\mathcal{D}) = \mathcal{D}$. Observe also that $\text{canondb}(Q) \subseteq \mathcal{D}$, and we can extend this to a homomorphism as we chase. This means there is a homomorphism h_2 from $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$ into $\text{Chase}_{\Sigma_{\mathcal{V}}}(\mathcal{D})$ which is the identity on $\text{canondb}(Q)$ and, in particular, is the identity on values corresponding to source-join variables. If we look only at the instance consisting of primed atoms we see that h_2 is a homomorphism from the primed atoms of $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$ to the primed atoms of $\text{Chase}_{\Sigma_{\mathcal{V}}}(\mathcal{D}) = \mathcal{D}$ and the primed atoms of \mathcal{D} are exactly $\text{canondb}(Q')$. In summary, we have a homomorphism h_2 from the primed atoms of $\text{Chase}_{\Sigma_{\mathcal{V}}}(\text{canondb}(Q))$ to $\text{canondb}(Q')$.

We let $h_Q = h_2 \circ h_1$, this is a homomorphism from $\text{canondb}(Q')$ to itself. Figure 1 gives a graphical representation of the situation. We now consider two cases.

Case 1: h_Q is not injective. By unpriming and using the canonical homomorphism from Q to $\text{canondb}(Q)$ and its reverse, we obtain a non-injective homomorphism from Q to itself. But now, by Lemma 3.2, we have a contradiction of minimality.

Case 2: h_Q is injective. Since h_Q is a total function on a finite domain, it must be bijective. What we need to prove, by Proposition 3.4, is the existence of a homomorphism h from $\text{canondb}(\text{CanView}^{s_0}(Q'))$ into $\text{Chase}_{\Sigma_{\mathcal{V}^s}}(\text{canondb}(\text{CanView}^s(Q)))$, where the homomorphism must be the identity on values corresponding to free variables of $\text{CanView}^s(Q')$ (i.e. the source-join variables of Q').

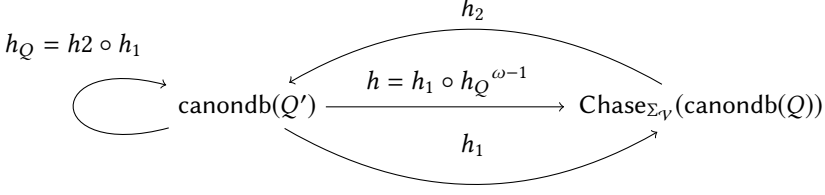


Fig. 1. Representation of the homomorphisms introduced for proof of Proposition 3.4

Since h_Q is bijective, for some number $\omega > 0$, h_Q^ω is the identity. We then look at $h = h_1 \circ (h_Q)^\omega$. This is a homomorphism from $\text{canondb}(Q')$ to $\text{Chase}_{\Sigma_V}(\text{canondb}(Q))$ (see Figure 1). We already know that $h_2 \circ h = h_Q^\omega$ is the identity but since h_2 is already the identity on source-join variables, it must be that h also is. Therefore h restricted to s_0 is the homomorphism we were looking for. \square

Consequences. Combining Theorem 3.1 and Proposition 2.5 gives a partial answer to the question of how to obtain useful and non-disclosing views:

COROLLARY 3.5. *For any non-disclosure function F , and for any BCQs Q and p , if there is some 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 (Theorem 44 of [9]):

COROLLARY 3.6. *There is a Σ_2^P algorithm taking as input BCQs Q and p and determining whether there is a CQ-based d -view that is useful for Q and UN non-disclosing for p . If Q is assumed minimal, then the problem is in CoNP.*

We will not investigate the algorithmic consequences more thoroughly, since they are not the focus of the paper. We note that determining whether the canonical views of Q are UN non-disclosing for a query p can be decomposed into UN non-disclosure questions on each source: whether or not the canonical view of p for a given source s is UN disclosed relative to the canonical view of Q for s . This reduces the complexity to Σ_2^P in the maximum size of the canonical views for Q and p over all sources, which could be considerably smaller than the size of Q and p itself.

The following example shows that the requirement that Q is minimal (that is, has no redundant conjuncts) in Theorem 3.1 is essential.

Example 3.7. Consider two sources. The first source comprises the relation names R_1 , R_2 and R_3 , while the second source comprises a single relation name T . Consider also the conjunctions of atoms C_1 and C_2 defined as:

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

The conjunction C_1 states that there is an element in T that is the source of an R edge. The conjunction C_2 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 BCQ Q defined as

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

Note that the conjunction of atoms C_1 in Q is redundant. Indeed, the BCQ

$$Q^{\min} = \exists x', y', z'. C_2$$

is equivalent to Q .

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

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

But 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 3.1 tells us that the canonical d-view of Q^{\min} is a minimally informative useful d-view within the class of CQ views. We claim that the canonical d-view of Q is *not* minimally informative for this class, and in fact reveals significantly more than the canonical d-view of Q^{\min} . Consider the secret query $p = \exists t. R_1(t, t)$ stating that there is an R_1 self-loop. The canonical d-view of Q^{\min} 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^{\min} .

In contrast, the canonical d-view of Q is UN disclosing for p . Consider \mathcal{D}_0 , the critical instance for the source schema (see Section 2). On \mathcal{D}_0 the returned bindings have x as only $*$, and from this we can infer that the witness elements for y' and z' can only be $*$, and hence the d-view discloses p with \mathcal{D}_0 as the witness.

By Proposition 2.5 the canonical d-view of Q cannot be minimally informative within the class of CQ views. \triangleleft

4 ARBITRARY VIEWS

In Section 3 we showed that the canonical d-view is a minimally informative useful view within the class of CQ views, assuming that the utility query is minimized as a CQ. We now turn to minimally informative useful d-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 gives the minimal information over arbitrary useful d-views for a given BCQ Q . That is, we want to arrive at an analog of Theorem 3.1 replacing “CQ views” by “arbitrary views” and “canonical view” by a generalization.

4.1 An equivalence class representation of minimally informative 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, one can define a function mapping each instance to its equivalence class. A d-instance \mathcal{D}_1 is indistinguishable from a d-instance \mathcal{D}_2 by the d-view specified by ECR $\langle \equiv^s: s \in \text{Srcs} \rangle$ when $\mathcal{D}_1^s \equiv^s \mathcal{D}_2^s$ holds for each source s . Determinacy of one d-view by another d-view corresponds exactly to the refinement relationship between the corresponding ECRs. We will thus abuse notation by talking about indistinguishability, usefulness, and minimal informativeness 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 informative. 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 I , we use (I, C) to denote the d-instance formed by interpreting the s -relation names as in I and the others as in C .

We say two s -instances I, I' are (s, Q) -equivalent if for any s -context C ,

$$(I, C) \models Q \Leftrightarrow (I', C) \models Q$$

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. It is also not difficult to see that the corresponding d-view is a minimally informative useful d-view for Q within the class of all views:

PROPOSITION 4.1. *The d-view corresponding to global Q -equivalence is a minimally informative useful d-view for Q within the collection of all views.*

PROOF. Fixing BCQ Q , we show that the d-view corresponding to global Q -equivalence is useful for Q . Consider two d-instances \mathcal{D} and \mathcal{D}' that are globally Q -equivalent, and assume \mathcal{D} satisfies Q . We will transform \mathcal{D} into \mathcal{D}' by “swapping one source at a time”, replacing the s component of \mathcal{D} with the s component of \mathcal{D}' . Each swap preserves satisfaction of Q because the s components of \mathcal{D} and \mathcal{D}' are (s, Q) -equivalent. Thus \mathcal{D}' satisfies Q as well.

To show that the d-view is minimally informative, consider a d-view \mathcal{V} that is useful for Q , and suppose we have \mathcal{D}_1 which agrees with \mathcal{D}_2 on \mathcal{V} . Fixing any source s we will show that \mathcal{D}_1^s and \mathcal{D}_2^s are (s, Q) equivalent. This will show that \mathcal{D}_1 and \mathcal{D}_2 are globally Q -equivalent, which will imply minimal informativeness. To see this last statement fix any context C , and note that (\mathcal{D}_1^s, C) and (\mathcal{D}_2^s, C) agree on \mathcal{V} , and hence agree on Q . This completes the proof. \square

Note that the result can be seen as an analog of Theorem 3.1. From the result above we conclude an analog of Corollary 3.5:

PROPOSITION 4.2. *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 useful d-view that is minimally informative 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 producing 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 from 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* , $\text{CanCtxt}^s(Q)$, is the CQ whose atoms are all the atoms of Q that are *not* in source s , and whose free variables are $\text{SJVars}(s, Q)$.

Definition 4.3. For a source s , a BCQ Q and a variable binding σ for Q , a shuffle μ of $\text{CanView}^s(Q)$ is *invariant relative to* $\langle \sigma, \text{CanCtxt}^s(Q) \rangle$ if for any d-instance \mathcal{D} such that $\mathcal{D}, \sigma \models \text{CanCtxt}^s(Q)$, we have $\mathcal{D}, \sigma \models \mu(\text{CanCtxt}^s(Q))$. When the source and the utility query is clear, we refer to μ simply as an *invariant shuffle* for brevity.

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

Invariance concerns every binding σ . We would like to abstract to bindings satisfying a set of equalities. A *type* for $\text{SJVars}(s, Q)$ is a set of equalities between variables in $\text{SJVars}(s, Q)$. 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* $\langle \tau, \text{CanCtxt}^s(Q) \rangle$: the invariance condition holds for all bindings σ satisfying τ .

Example 4.4. We illustrate the notion of invariant shuffle with an example that will become important later.

Consider a d-schema with two sources, one containing a ternary relation name R and the other containing a unary relation name 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 source-join variables in each source are x_1 and x_2 . Let τ be the type that states $x_1 \neq x_2$. Let us consider the invariant shuffles for the R -source and τ . The canonical context query of Q for this source is $S(x_1) \wedge S(x_2)$.

The identity is always an invariant shuffle. We claim that the mapping μ_0 taking x_1 and x_2 both to x_1 is an invariant shuffle for τ . Suppose we have a d -instance \mathcal{D} and a binding σ to the source join variables that is consistent with τ and such that $\mathcal{D}, \sigma \models S(x_1) \wedge S(x_2)$. Applying μ_0 to $S(x_1) \wedge S(x_2)$ gives $S(x_1)$, and clearly we have $\mathcal{D}, \sigma \models S(x_1)$. Thus the shuffle μ_0 is invariant. Similarly we can see that the mapping sending x_1 and x_2 to x_2 is invariant, as well as the mapping that swap x_1 and x_2 .

Summing up, an invariant shuffle for a source s and type τ is a symmetry of the query formed from the utility query by specializing to type τ and restricting to the atoms lying outside of s . In this example, the utility query outside of the R -source is extremely symmetric, so it has the maximal set of invariant shuffles. \triangleleft

For a source s and a CQ Q , let τ_1, \dots, τ_n be all the equality types over the variables in $\text{SJVars}(s, Q)$.

Definition 4.5. The set of *invariant shuffle views* of Q for source s consist of views $V_{\tau_1}, \dots, V_{\tau_n}$ where each V_{τ_i} is defined as $\tau_i(\mathbf{x}) \wedge \bigvee_{\mu} \mu(\text{CanView}^s(Q))$, 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 4.6. *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 4.7. Given two s -instances I_1 and I_2 , we say that I_1 and I_2 are *invariant shuffle equivalent* (relative to Q) if: Whenever I_1, σ satisfies $\text{CanView}^s(Q)(\mathbf{x})$ then there is some shuffle μ invariant relative to $\langle \sigma, \text{CanCtxt}^s(Q) \rangle$ such that I_2, σ satisfies $\mu(\text{CanView}^s(Q))(\mathbf{x})$, and similarly with the role of I_1 and I_2 reversed.

We show:

PROPOSITION 4.8. *For any source s , invariant shuffle equivalence is identical to (s, Q) -equivalence.*

PROOF. First, suppose I_1, I_2 are local instances for source s that are invariant shuffle equivalent, and suppose we have a match of Q in (I_1, C) via $h^{1,C}$. We want to show that there is a match in (I_2, C) .

We know that the variables in $\text{SJVars}(s, Q)$ are mapped by $h^{1,C}$ into I_1 . Let h_0 be the restriction of $h^{1,C}$ to the variables of $\text{SJVars}(s, Q)$. Then I_1, h_0 satisfies $\text{CanView}^s(Q)$. Thus by shuffle equivalence there is some shuffle μ invariant relative to $\langle h_0, \text{CanView}^s(Q) \rangle$, such that $I_2, h_0 \models \mu(\text{CanView}^s(Q))$, with witness h_2 extending h_0 . We also know that C, h_0 satisfies $\text{CanCtxt}^s(Q)$, since $h^{1,C}$ witnesses this as well. Applying the definition of shuffle invariance, C, h_0 satisfies $\mu(\text{CanCtxt}^s(Q))$. 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 $h^{1,C}$ agree on their common variables. Define $h^{2,C}$ by mapping the variables in $\text{SVars}(s, Q)$ as in h_2 , and those variables outside of $\text{SJVars}(s, Q)$ as in $h^{\mu,C}$. Since these are two compatible homomorphisms, $h^{2,C}$ witnesses that $(I_2, C) \models Q$. 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 I_1, I_2 are globally Q -equivalent, and I_1, σ satisfies $\text{CanView}^s(Q)(\mathbf{x})$. We will show that there is a shuffle μ , invariant relative to $\langle \sigma, \text{CanCtxt}^s(Q) \rangle$, such that $I_2, \sigma \models \mu(\text{CanView}^s(Q))(\mathbf{x})$. Let C_1 be the canonical database of $\sigma(\text{CanCtxt}^s(Q))$. That is, for each source s' other than s , we have a fact for each s' atom of Q , where each variable x of $\text{SJVars}(s, Q)$ is replaced by $\sigma(x)$ and each variable x not in $\text{SJVars}(s, Q)$ is replaced by a fresh element c_x .

Q clearly holds in (I_1, C_1) . So by global Q -equivalence, Q holds in (I_2, C_1) via some homomorphism h . Note that, by the design of C_1 , the only elements that are shared between I_2 -facts and C_1 -facts lie in the range of σ . Thus h must map the variables in $\text{SJVars}(s, Q)$ to the image of σ . The binding σ may not be injective, but we let σ^{-1} be a “right 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 $x \in \text{SJVars}(s, Q)$ to $\sigma^{-1}(h(x))$. So $\sigma(\mu(x)) = h(x)$.

We first claim that μ is invariant relative to $\langle \sigma, \text{CanCtxt}^s(Q) \rangle$. We show this by arguing that h is a homomorphism from $\sigma(\mu(\text{CanCtxt}^s(Q)))$ to $\sigma(\text{CanCtxt}^s(Q))$. By

definition of μ , we have for each atom $A(x_1 \dots x_m, y_1 \dots y_n)$ of $\text{CanCtxt}^s(Q)(\mathbf{x})$,

$$\begin{aligned} & A(h(x_1) \dots h(x_m), h(y_1) \dots h(y_n)) = \\ & A(\sigma(\mu(x_1)) \dots \sigma(\mu(x_m)), h(y_1) \dots h(y_n)) \end{aligned}$$

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

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

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

We next claim that $\mathcal{I}_2, \sigma \models \mu(\text{CanView}^s(Q))$. The witness will be the extension h' of σ that maps all variables in $\text{SVars}(s, Q) - \text{SJVars}(s, Q)$ via h . Consider an atomic formula $A(x_1 \dots x_m, y_1 \dots y_n)$ of $\text{CanView}^s(Q)$, where \mathbf{x} are free variables of $\text{CanView}^s(Q)$. Therefore $A(\mu(x_1) \dots \mu(x_m), y_1 \dots y_n)$ is a generic atom of $\mu(\text{CanView}^s(Q))$. To argue that h' is a homomorphism that witnesses $\mathcal{I}_2, \sigma \models \mu(\text{CanView}^s(Q))$, 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) . \square

To complete the proof of Proposition 4.6, we show:

PROPOSITION 4.9. *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 $\mathcal{I}_1, \sigma \models \text{CanView}^s(Q)$, and let τ be the type of σ . Since the identity is invariant relative to τ , we have \mathcal{I}_1, σ satisfies V_τ , and thus \mathcal{I}_2, σ must satisfy it. Therefore there is μ that is invariant relative to τ such that $\mathcal{I}_2, \sigma \models \mu(\text{CanView}^s(Q))$. Since τ is of type σ , we have μ is invariant relative to $\langle \sigma, \text{CanCtxt}^s(Q) \rangle$. 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, $\mathcal{I}_1, \sigma \models \tau \wedge \mu(\text{CanView}^s(Q))$ 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 $\mathcal{I}_1, \sigma' \models \text{CanView}^s(Q)$ by definition. Thus by invariant shuffle equivalence, there is μ' invariant for $\sigma', \text{CanCtxt}^s(Q)$ such that $\mathcal{I}_2, \sigma' \models \mu'(\text{CanView}^s(Q))$.

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 $\tau, \text{CanCtxt}^s(Q)$.*

PROOF. Suppose σ_0 satisfies τ , and $\mathcal{I}_0, \sigma_0 \models \text{CanCtxt}^s(Q)$. 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 now make the following chain of conclusions, with the rationale appearing to the right of each statement:

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

The last line shows that μ'' is invariant as required. \square

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

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

PROOF. 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 . We note that $\mathcal{I}, \sigma_0 \models \mu(R)$ if and only if $\mathcal{I}, \sigma_1 \models R$. This follows just from unwinding the definitions.

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 just above, we see that the following are equivalent:

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

which gives the proof of the claim. \square

Combining the two claims, we conclude that \mathcal{I}_2, σ satisfies V_τ as required, which completes the proof of Proposition 4.9. \square

Putting together Proposition 4.2 and 4.6, we obtain:

THEOREM 4.10. *[Minimally informative d-views for the class of all views] The invariant shuffle views are minimally informative for Q within the class of all views.*

This yields a corollary for non-disclosure analogous to Corollary 3.5:

COROLLARY 4.11. *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 .*

In Example 1.1 there are no nontrivial shuffles, so the canonical d-view is minimally informative within the class of all views. In general, the invariant shuffle views can be *unsafe*: different disjuncts may contain distinct variables. Of course, they can be implemented easily by using a wildcard to represent elements outside the active domain. Further, we can convert each of these unsafe views to an “information-equivalent” set of relational algebra views:

PROPOSITION 4.12. *For every view defined by a (possibly unsafe) DCQ, 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 relational algebra-based d -view that is minimally informative 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.

PROOF. 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 $\text{Vars}(\varphi_i)$ be the set of variables within it, 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:

$$\bigvee_{\{\varphi_i \mid \text{Vars}(\varphi_i) = S\}} \varphi_i \wedge \neg \left(\bigvee_{\{\varphi_j \mid \text{Vars}(\varphi_j) \subsetneq S\}} \varphi_j \right)$$

Example 4.13. 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.

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

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

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

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

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. \triangleleft

Returning to the general case, we claim that the set of views V_S determines V and vice versa. In one direction, suppose I_1 and I_2 agree on each V_S , and $I_1 \models V(\mathbf{t})$. Choose i with $I_1 \models \varphi_i(\mathbf{t})$ such that $\text{Vars}(\varphi_i)$ is minimal. Let \mathbf{t}' be the subtuple of \mathbf{t} corresponding to the variables of φ_i . Then, by minimality of $\text{Vars}(\varphi_i)$, $I_1 \models V_{\text{Vars}(\varphi_i)}(\mathbf{t}')$ and hence $I_2 \models V_{\text{Vars}(\varphi_i)}(\mathbf{t}')$. From this we see that $I_2 \models \varphi_j(\mathbf{t}')$ with $\text{Vars}(\varphi_j) = \text{Vars}(\varphi_i)$ and hence $I_2 \models V(\mathbf{t})$.

In the other direction, suppose I_1 and I_2 agree on V , and $I_1 \models V_S(\mathbf{t})$. Fix φ_i with variables from S such that $I_1 \models \varphi_i(\mathbf{t})$. We need to show $I_2 \models V_S(\mathbf{t})$. We work by induction and assume that V determines $V_{S'}$ for each proper subset S' of S . We know that $I_1 \models V(\mathbf{t})$. Hence $I_2 \models V(\mathbf{t})$, and thus there is some j such that $I_2 \models \varphi_j(\mathbf{t})$.

First, consider the case where S consists of all free variables in V . In this case the free variables of φ_j are either the same as S , or are a proper subset of S , since it cannot be a superset. If φ_j contains all the variables of φ_i , then φ_j witnesses the left conjunct

in V_S holding for \mathbf{t} . The induction hypothesis implies that the right conjunct is satisfied. We can conclude that $\mathcal{I}_2 \models V_S(\mathbf{t})$ as required. If the free variables of φ_j are 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 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 . Since \mathcal{I}_1 with \mathbf{t} satisfies some positive literal in V_S , we know $\mathcal{I}_1 \models V(\mathbf{t}')$, and therefore $\mathcal{I}_2 \models V(\mathbf{t}')$. Thus there is a proper subtuple \mathbf{t}'' of \mathbf{t}' and a disjunct φ_k such that $\mathcal{I}_2 \models \varphi_i(\mathbf{t}'')$. As above, we can choose \mathbf{t}'' minimal. By our choice of the elements in $\mathbf{t}' - \mathbf{t}$, there is \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. \square

Example 4.14. Let us recall the d-schema of Example 4.4, which contained a ternary relation name R and the other containing a unary relation name S . Recall that the utility query Q was:

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

The canonical view for the R source $\text{CanView}^R(Q)$ 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 3.1 the canonical d-view of Q is minimally informative among the CQ-based d-views. We will argue that this d-view is not minimally informative 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 the canonical d-view of Q is UN disclosing for p . Consider the critical instance of the R -source, which we recall from Definition 2.9. 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 canonical d-view would reveal $x_1 = y_0, x_2 = *$. So y_0 must be $*$, and therefore p is disclosed. By Corollary 3.5, no CQ-based d-view 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 the corresponding shuffle view for τ_1 is:

$$V_{\tau_1} = \exists y. R(x_1, x_1, y) \wedge R(y, x_1, x_1)$$

For bindings satisfying τ_2 we saw in Example 4.4 that there are several shuffles invariant for $\text{CanCtxt}^R(Q) = S(x_1) \wedge S(x_2)$: 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 4.12 we can convert it into a safe relational algebra view that yields the same ECR, $V_{\tau_2}^{\text{safe}}$ defined as $x_1 \neq x_2$ conjoined with:

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

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 2.5, the canonical views of Q cannot be minimally informative within the class of all views, or even within the class of relational algebra views. \blacktriangleleft

4.2 The importance of using negation or unsafe queries

We now consider in a bit more detail which query language features are needed to obtain minimally informative useful d-views.

Let us return to the utility query from Example 4.14:

$$Q = \exists x_1, x_2, y_0. R(x_1, x_2, y_0) \wedge R(y_0, x_2, x_1) \wedge S(x_1) \wedge S(x_2)$$

with R and S in distinct sources.

Consider the secret query from the same example $p = \exists x. R(x, x, x)$.

We saw from Example 4.14 that there is a d-view that is useful for Q and UN non-disclosing for p . Indeed, we have one example of such a d-view which is an unsafe DCQ, and another makes use of negation. We show that this is essential: in this example, *we must give up either safety or positivity in order to get a useful and UN non-disclosing d-view*. Since we know that minimally informative d-views are always minimal with respect to disclosure, this example will also prove that *we may need to use either unsafe queries or negation to construct minimally informative d-views*.

We formalize what we mean by safe and positive views. We recall that a query Q is *homomorphism-invariant* if $\mu(Q(I)) \subseteq Q(I')$ for all pairs of instances I and I' and all homomorphisms μ from I to I' . A query Q is *adom-based* when $\text{adom}(Q(I)) \subseteq \text{adom}(I)$. Note that all queries defined by CQs and UCQs are homomorphism-invariant and adom-based. As with monotonicity, there are expressive extensions of UCQs that are also homomorphism-invariant. Datalog [1] is again a well-known example of such a language, although we will not need to discuss the details of specific languages that are homomorphism-invariant. A set of views is homomorphism-invariant or adom-based exactly when all its defining queries are.

We are now ready to state the result formalizing that we cannot avoid either unsafe queries or negation:

PROPOSITION 4.15. *Any set of useful views for Q which is homomorphism-invariant and adom-based must UN-disclose p .*

Thus in particular, we can protect more information using relational algebra views than by views using only monotone relational algebra operators, even when allowing recursion.

PROOF. Let \mathcal{V} be useful for Q , homomorphism-invariant and adom-based and \mathcal{V}_R be the views on the R sources. Let \mathcal{D}_Q be the canonical database of Q .

We first claim that the active domain of the view image of \mathcal{D}_Q under \mathcal{V}_R must contain each of the elements y_0, x_1, x_2 :

- If we suppose that x_1 does not belong to $\text{adom}(\mathcal{V}_R(\mathcal{D}_Q))$ we can apply a function h mapping x_1 to another value on the source R . By homomorphism-invariance, \mathcal{V}_R is not impacted and, since \mathcal{D}_Q and $h(\mathcal{D}_Q)$ agree on the source S , we have $\mathcal{V}(\mathcal{D}_Q) = \mathcal{V}(h(\mathcal{D}_Q))$. But this contradicts utility of \mathcal{V} for Q as Q holds in \mathcal{D}_Q and not in $h(\mathcal{D}_Q)$. Therefore x_1 needs to appear in $\text{adom}(\mathcal{V}_R(\mathcal{D}_Q))$.
- With the same argument as for x_1 , x_2 must also appear in $\text{adom}(\mathcal{V}_R(\mathcal{D}_Q))$.
- Consider the mapping m on the domain of \mathcal{D}_Q that swaps x_1 and y_0 . Thus it produces the canonical database of $\mathcal{D}_{Q'}$.

$$\exists x_1, x_2, y_0. R(y_0, x_2, x_1) \wedge R(x_1, x_2, y_0) \wedge S(y_0) \wedge S(x_2)$$

This is an isomorphism, hence a homomorphism mapping \mathcal{D}_Q to $\mathcal{D}_{Q'}$. The views \mathcal{V} are homomorphism-invariant. Thus, because x_1 appears in the active domain of $\mathcal{V}(\mathcal{D}_{Q'})$, y_0 should also appear in the active domain of $\mathcal{V}(\mathcal{D}_Q)$.

We now consider the critical instance. For this schema, the critical instance is $\bar{I}_0 = \{R(*, *, *), S(*)\}$. Consider an instance I' that is equivalent to \bar{I}_0 on \mathcal{V} . Let I'_R be the restriction of I' to the R -source. Since \mathcal{V} is adom-based, $\text{adom}(\mathcal{V}_R(I_0)) = \{*\}$ and the same holds on I' .

I' must satisfy Q by usefulness of \mathcal{V} and thus there is a homomorphism h of \mathcal{D}_Q into I' . Let x'_1, x'_2 , and y'_0 be the images of x_1, x_2 and y_0 respectively under h . Then by homomorphism invariance, $\{x'_1, x'_2, y'_0\}$ appear in $\text{adom}(\mathcal{V}_R(I'))$ but $\text{adom}(\mathcal{V}_R(I')) = \{*\}$. Thus all of these must be the same as $*$ and thus p is true in I' . This shows that \mathcal{V} is UN-disclosing for \mathcal{V} . \square

5 LOCAL 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 relation names within a single source.

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

We begin our analysis of the impact of constraints by extending our results on *arbitrary* d-views to account for local TGDs Σ . Here we will follow a straightforward extension of the ideas in Section 4.

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. By adapting the approach in Section 4, we can show:

THEOREM 5.1. *[Minimally informative d-views w.r.t. local TGDs] For any set of local TGDs Σ , the Σ -invariant shuffle views of Q provide a minimally informative useful d-view for Q within the class of all views, relative to Σ .*

The result has effective consequences for “tame” TGDs (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. From [7, 8] we know that checking whether a d-view is UN non-disclosing can be decided for such classes. Thus we can check whether the canonical d-view is UN non-disclosing effectively, and hence whether any d-view can be useful and UN non-disclosing.

We will now begin the proof of Theorem 5.1. We start by generalizing the ECR global Q -equivalence to global Q - Σ -equivalence, looking only at contexts that satisfy Σ . Using the same swapping argument as in Proposition 4.1, we see that the d-view corresponding to this ECR is a minimally informative useful d-view for Q within the class of all views, relative to Σ .

Let σ be a mapping of $\text{SJVars}(s)$ into some instance I . A shuffle μ of $\text{CanView}^s(Q)$ is Σ -invariant relative to $\langle \sigma, \text{CanCtx}^s(Q) \rangle$ if whenever $I', \sigma \models \text{CanCtx}^s(Q)$ and I' satisfies Σ then $I', \sigma \models \mu(\text{CanCtx}^s(Q))$. Note that since the TGDs 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 I_1 and I_2 , we say that I_1 and I_2 are Σ -invariant shuffle equivalent if:

- whenever I_1, σ satisfies $\text{CanView}^s(Q)(x)$ there is some shuffle μ which is Σ -invariant relative to $\langle \sigma, \text{CanView}^s(Q) \rangle$, such that $I_2, \sigma \models \mu(\text{CanView}^s(Q))(x)$;
- conversely, whenever I_2, σ satisfies $\text{CanView}^s(Q)(x)$ there is some shuffle μ which is Σ -invariant relative to $\langle \sigma, \text{CanView}^s(Q) \rangle$, such that $I_1, \sigma \models \mu(\text{CanView}^s(Q))(x)$

We can now extend Proposition 4.8, following the same proof:

PROPOSITION 5.2. *Suppose Σ consists of local TGDs. Then for all instances satisfying the TGDs, Σ -invariant shuffle equivalence is identical to global Q - Σ equivalence.*

PROOF. First, suppose I_1, I_2 satisfy all local TGDs and are Σ -invariant shuffle equivalent. Consider a context C that satisfies local TGDs Σ and suppose we have a match of Q in (I_1, C) via $h^{1,C}$. We want to show that there is a match in (I_2, C) . This will be exactly as in the case without constraints.

We know that the variables in $\text{SJVars}(s, Q)$ are mapped by $h^{1,C}$ into I_1 . Let h_0 be the restriction of $h^{1,C}$ to the variables of $\text{SJVars}(s, Q)$. Then I_1, h_0 satisfies $\text{CanView}^s(Q)$. Thus by Σ -invariant shuffle equivalence there is a shuffle μ which is Σ -invariant relative to $\langle h_0, \text{CanView}^s(Q) \rangle$, such that $I_2, h_0 \models \mu(\text{CanView}^s(Q))$, with witness h_2 extending h_0 . We also know that C, h_0 satisfies $\text{CanCtx}^s(Q)$, 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 $\mu(\text{CanCtx}^s(Q))$. 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 $\text{SVars}(s, Q)$ as in h_2 , and those variables outside of $\text{SVars}(s, Q)$ as in $h^{\mu,C}$. Since these are two compatible homomorphisms, $h^{2,C}$ witnesses that $(I_2, C) \models Q$. 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 I_1, I_2 are Q - Σ -equivalent, and I_1, σ satisfies $\text{CanView}^s(Q)(\mathbf{x})$. We will show that there is a shuffle μ , Σ -invariant relative to $\langle \sigma, \text{CanCtxt}^s(Q) \rangle$ such that $I_2, \sigma \models \mu(\text{CanView}^s(Q))(\mathbf{x})$.

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 $\text{SVars}(s, Q)$ is replaced by $\sigma(x)$ and each variable x not in SVars 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 (I_1, C_1) . So by Q - Σ -equivalence, Q holds in (I_2, C_1) via some homomorphism h . As before, the only elements shared between I_2 and C_1 lie in the range of σ . Thus h must map the variables in $\text{SVars}(s, Q)$ 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 $x \in \text{SVars}(s, Q)$ to $v_{h(x)}$. Thus $\sigma(\mu(x)) = h(x)$.

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

$$\begin{aligned} & A(h(x_1) \dots h(x_m), h(y_1) \dots h(y_n)) = \\ & A(\sigma(\mu(x_1)) \dots \sigma(\mu(x_m)), h(y_1) \dots h(y_n)) \end{aligned}$$

$A(h(x_1) \dots h(x_m), h(y_1) \dots h(y_n))$ is in $\text{Chase}_\Sigma(\text{CanCtxt}^s(Q)(\sigma))$ since h is a homomorphism into (I_2, C_1) and thus for facts in $\mu(\text{CanCtxt}^s(Q))(\sigma)$, it must map into C_1 .

Thus we conclude

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

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

We next claim that $I_2, \sigma \models \mu(\text{CanView}^s(Q))$. The witness will again be the extension of σ that maps all variables in $\text{SVars}(s, Q) - \text{SVars}(s, Q)$ via h .

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

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

is a generic atom of $\mu(\text{CanView}^s(Q))$. We know that $A(h(x_1) \dots h(x_m), h(y_1) \dots h(y_n))$ holds in I_2 , since h is a homomorphism into I_2 . Thus

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

holds of σ in 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 I_2 as required. \square

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

The following result is proven verbatim as in the case without background knowledge, just adding in the statement that the instances in question satisfy Σ .

PROPOSITION 5.3. *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 proof of Theorem 5.1.

5.2 Extension of results on CQ views to local background knowledge

We now turn to the impact of local constraints on view design with CQ views. It is easy to show that we cannot 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.

Example 5.4. Consider the schema, utility query Q and secret query p from Example 4.14. Let Σ consist of the view definitions for the views V_{τ_1} and $V_{\tau_2}^{\text{safe}}$ in the example. Relative to Σ , we have CQ view definitions that are useful and UN non-disclosing: these would be views that export the relations V_{τ_1} and $V_{\tau_2}^{\text{safe}}$, along with the view that exports the interpretation of the additional relation name S . However, as observed in Example 4.14, the canonical views of Q are UN disclosing for p . \triangleleft

Thus we restrict to *local constraints Σ that are TGDs*. 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 . We will show the following extension of Theorem 3.1:

THEOREM 5.5. *[Minimally informative CQ views w.r.t. local TGDs] Let Σ be a set of local TGDs, Q a CQ minimal with respect to Σ . Then the canonical d -view of Q is minimally informative useful within the class of CQ views relative to Σ .*

If the constraints have terminating chase, then the same holds when the notion of usefulness is relativized to finite instances.

It is possible to prove Theorem 5.5 through the same technique as Theorem 3.1. But we will give an alternative proof. Notice that if there are no constraints, the requirement of terminating chase is vacuous. Thus this represents an alternate proof of Theorem 3.1, which also covers the case where usefulness is relativized to finite instances.

The alternative approach makes use of the following result about TGDs:

PROPOSITION 5.6. *If we have a set of TGDs Σ , a conjunction of atoms $\phi(\mathbf{x})$ and a disjunction of CQs $\bigvee_i Q_i$ such that Σ entails $\forall \mathbf{x} \phi(\mathbf{x}) \rightarrow \bigvee_i Q_i$. Then, for some i , Σ entails $\forall \mathbf{x} \phi(\mathbf{x}) \rightarrow Q_i$. The same holds when entailment is considered over finite instances only.*

This is proven using a simple product construction. A proof can be found in e.g. Lemma 3.5 of [5].

We now prove Theorem 5.5. We first prove it for our default notion of usefulness, involving quantification over all instances. Suppose \mathcal{V} is a CQ-based d-view that is useful for Q . We show that each view \mathcal{V}_s in \mathcal{V} is determined by the canonical d-view.

Fix a source s . Let $\Sigma_{\mathcal{V},s}$ be the conjunction of Σ , Σ' a copy of Σ on primed relation names, and:

$$\forall \mathbf{x}. \mathcal{V}_s(\mathbf{x}) \leftrightarrow \mathcal{V}'_s(\mathbf{x})$$

where \mathcal{V}'_s is a primed copy of the definitions in \mathcal{V}_s .

We know that \mathcal{V}_s determines Σ -invariant shuffle equivalence on s . Writing this out as an entailment, we obtain:

$$\Sigma_{\mathcal{V},s} \models \forall \mathbf{z}. \text{CanView}_s(Q)(\mathbf{z}) \rightarrow \bigvee_{\mu} \text{CanView}_s(Q')(\mu(\mathbf{z}))$$

In the disjunction above, μ varies over shuffles of s that are Σ -invariant.

Note that this is an entailment that matches the requirements of Proposition 5.6. So we can apply Proposition 5.6 to infer that there is some Σ -invariant shuffle μ such that

$$\Sigma_{\mathcal{V},s} \models \forall \mathbf{z}. \text{CanView}_s(Q)(\mathbf{z}) \rightarrow \text{CanView}_s(Q')(\mu(\mathbf{z}))$$

We consider two cases, as in the proof of Theorem 3.1:

Case 1: μ is not injective. Note that the entailment holds for all instances satisfying $\Sigma_{\mathcal{V},s}$, and in particular to the case where the primed and unprimed copies of the relations are identical. Thus we have

$$\Sigma \models \forall \mathbf{z}. \text{CanView}_s(Q)(\mathbf{z}) \rightarrow \text{CanView}_s(Q)(\mu(\mathbf{z}))$$

Consider the mapping on all variables of Q that applies μ on $\text{SJVars}(Q, s)$ but is the identity otherwise. We can check that this is a homomorphism of Q into a proper subset of itself. This contradicts the minimality of Q with respect to Σ , since the analog of Lemma 3.2 holds in the presence of constraints.

Case 2: μ is bijective. Then some iterate of it is the identity, and thus we can assume that μ is the identity. That is:

$$\Sigma_{\mathcal{V},s} \wedge \text{CanView}_s(Q)(\mathbf{z}) \models \text{CanView}_s(Q')(\mathbf{z})$$

But then \mathcal{V} determines the canonical view of s for Q .

The proof of the statement quantifying over finite instances is identical. Note that since we assume that the constraints have terminating chase, Σ -invariance over finite instances and Σ -invariance over arbitrary instances is the same, since it is a statement that is captured by a conjunctive query containment using Σ . If there is a counterexample to such a statement, it can always be taken to be the result of chasing a canonical database. The remaining components of the argument (e.g. Proposition 5.6) hold for quantification over finite instances.

This completes the argument for Theorem 5.5.

The analog of Corollary 3.5 follows from the theorem:

COROLLARY 5.7. *If any CQ based d-view is useful for Σ -minimal Q and non-disclosing for p relative to Σ , then the canonical d-view of Q is useful for Q and non-disclosing for p relative to Σ .*

Consequences for decidability. Theorem 5.5 shows that even in the presence of arbitrary local TGDs Σ it suffices to minimize the utility query under Σ and check the canonical d-view for non-disclosure under Σ . For arbitrary TGDs, even CQ minimization is undecidable. But for well-behaved classes of TGDs (e.g. those with terminating chase [3, 14], or frontier-guarded TGDs [4]) we can perform both minimization and UN non-disclosure checking effectively [7].

Example 5.8. Now that we have our results on local constraints, let us revisit Example 1.5, in which we had the local constraint Σ :

$$\forall x, y. \text{geolink}(x, y) \leftrightarrow \text{geolink}(y, x)$$

and the utility query Q was:

$$\exists x, y. \text{geolink}(x, y) \wedge \text{soclink}(x, y) \wedge \text{HighIncome}(x) \wedge \text{HighIncome}(y)$$

Q is still a minimal CQ: no atoms can be removed to get an equivalent query, even taking into account the integrity constraints. Thus Theorem 5.5 tells us that the canonical d-view is still minimally informative among the CQ-based d-views.

Theorem 5.1 tells us that the minimally informative d-views are among all views are given by the Σ shuffle-invariant views. We note that there is one non-trivial shuffle that is invariant with respect to the constraint: the shuffle μ_0 on the *geolink* source which swaps x and y . One can compute directly from the definition that μ_0 is invariant.

If we compute the shuffle views for the *HighIncome* source, we see that they are essentially the same as the canonical views. For example, for the type with $x \neq y$ we get the view

$$(\text{HighIncome}(x) \wedge \text{HighIncome}(y)) \vee (\text{HighIncome}(y) \wedge \text{HighIncome}(x))$$

but this is clearly the same as $\text{HighIncome}(x) \wedge \text{HighIncome}(y)$. But if we compute the shuffle views for the *soclink* source, we see that the impact of μ_0 is to give the invariant shuffle view:

$$\text{soclink}(x, y) \vee \text{soclink}(y, x)$$

Thus the minimally informative d-view in the presence of this constraint requires a disjunction on the *soclink* source, matching exactly the intuition we proposed in Example 1.5. \triangleleft

Injective shuffles and CQ Views. We have seen that the canonical d-view may be less informative than the shuffle invariant views. However, the canonical d-view can still be minimal for UN-disclosure even when it is not minimally informative. This was the situation in Example 1.5 in the presence of constraints: by using the power of disjunction in a shuffle, we could hide information, but we could not hide whether a Boolean CQ held. The key in this example was that the shuffles were *injective*: they never map multiple variables to the same variable. One can contrast this to the situation in Example 4.14. The utility query in that example admits a non-injective shuffle, identifying x_1 and x_2 . And in that example the canonical views were not minimal even in terms of UN-disclosure of CQs. We prove that this phenomenon occurs generally:

THEOREM 5.9. *Suppose every invariant shuffle of Q is injective, and that our constraints consist of TGDs. Then the canonical views of Q are UN-disclosure minimal*

PROOF. By Theorem 4.10, the shuffle invariant d-view returns minimal information, and in particular is minimal for UN-disclosure.

We show that if a BC p is UN-disclosed by the canonical d-view then it is UN-disclosed by the invariant shuffle d-view.

We know from Theorem 2.10 that p must be UN-disclosed on the critical instance I_0 . That is, every instance I' that agrees with I_0 on the canonical d-view must satisfy p . Now suppose I' agrees with I_0 on the invariant shuffle views. Because the shuffles are all injective, each invariant shuffle view is safe, and on I_0 returns either empty (for the types that impose some inequalities), or only tuples containing *. Then I' must agree with I_0 on the invariant shuffle views, since when all shuffles are injective, the invariant shuffle views without inequalities are just disjunctions of permutations of the canonical view. Thus the former returns a tuple of * exactly when the latter does. But p is then true on I' , which shows that it is UN-disclosed by the invariant shuffle views. \square

6 NON-LOCAL BACKGROUND KNOWLEDGE

The simplest kind of non-local constraint is the replication of a table between sources. A *replication constraint* is a constraint of the form

$$\forall \mathbf{x}. R_1(\mathbf{x}) \leftrightarrow R_2(\mathbf{x})$$

where R_1 is a relation name associated to source s_1 and R_2 is a relation name associated to a different source s_2 . We will sometimes abuse notation by using a single relation name, say R , and express the same constraint as above by saying “ R is replicated between s_1 and s_2 ” or “ R is shared between s_1 and s_2 ”.

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.

6.1 From minimally informative to UN non-disclosure minimality

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

THEOREM 6.1. *There is a schema with a replication constraint Σ and a BCQ Q where there is no minimally informative useful d-view for Q within the class of all views with respect to Σ .*

Our proof of Theorem 6.1 uses a schema with unary relation names 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 $\exists x. R(x) \wedge S(x) \wedge T(x)$.

We will explain how our views act when the active domain of our instances is over the integers. The proof will easily be seen to extend to arbitrary instances (e.g., by having the views reveal all information outside of the integers).

Consider the function $F_1(x) = 2x + 3$ for x even and $2x + 2$ for x odd, and also the function $F_2(x) = 2x + 4$ for x even and $3x$ for x odd. Let Stretch_1 be the function that applies F_1 to the interpretation of a relation name, element by element. Similarly define Stretch_2 using F_2 . Notice that F_1 maps 0 to 3 and 1 to 4, while F_2 maps 0 to 4 and 1 to 3. Intuitively, these are functions that “stretch the instance” by distorting the values within

the relations. For a number i and any function \mathcal{F} from instances to instance, such as Stretch_1 or Stretch_2 above, and any instance I , we let $(\mathcal{F})^i(I)$ be the instance resulting from applying \mathcal{F} i times to I .

We define a d-view \mathcal{V}_1 via an ECR, relating two instances I and I' of the source exactly when I' can be obtained by applying Stretch_1 on I some number of times (applying it to both relation names of the source) or vice versa. That is, the ECR of \mathcal{V}_1 is the smallest equivalence relation containing each pair $(I, \text{Stretch}_1(I))$. Let \mathcal{V}_2 be defined analogously using Stretch_2 . Towards showing that \mathcal{V}_1 and \mathcal{V}_2 are useful we prove following claim, which captures their key properties.

CLAIM 3. *If we have two d-instances satisfying the replication constraint, (I_1, I_2) and (I'_1, I'_2) , with the instances of the replicated relation names non-empty, then:*

- *If $I'_1 = \text{Stretch}_1^i(I_1)$, $I'_2 = \text{Stretch}_1^j(I_2)$ then $i = j$; and similarly for Stretch_2 .*
- *If $I'_1 = \text{Stretch}_1^i(I_1)$ and $I_2 = \text{Stretch}_1^j(I'_2)$ then $i = j = 0$; and similarly for Stretch_2 .*

PROOF. Let S be the content of the replicated relation name in (I_1, I_2) , while S' is the content of the replicated relation name in (I'_1, I'_2) .

We focus first on Stretch_1 . For the first item, let $c(S) = \min\{|x + 2| \mid x \in S\}$. We can check directly that for any non-empty S , $c(\text{Stretch}_1(S)) > c(S)$. Then $\text{Stretch}_1^i(S) = S' = \text{Stretch}_1^j(S)$ implies that $i = j$ as otherwise $c(\text{Stretch}_1^i(S)) > c(\text{Stretch}_1^j(S))$ when $i > j$ and $c(\text{Stretch}_1^i(S)) < c(\text{Stretch}_1^j(S))$ when $i < j$. For the second item we would have, $\text{Stretch}_1^i(S) = S'$ and $\text{Stretch}_1^j(S') = S$ which means $\text{Stretch}_1^{i+j}(S) = S$ which is only possible for $i + j = 0$.

For Stretch_2 , the proof is the same, but now using the function d defined as $d(S) = \min\{|x + 1.5| \mid x \in S\}$. We can check that $d(S) < d(\text{Stretch}_2(S))$ for any non-empty S . \square

From the claim, usefulness follows easily. Suppose we have (I_1, I_2) satisfying Q , and (I'_1, I'_2) is equivalent to (I_1, I_2) . Since (I_1, I_2) satisfies Q , we know that the interpretation of the replicated relation name in I_1 and I_2 is non-empty, so the claim applies to tell us that $I'_1 = I_1$, $I'_2 = I_2$.

Now suppose \mathcal{V} were a minimally informative useful d-view 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 $R = \{3\}$, $S = \{3, 4\}$, $T = \{4\}$. These are both valid instances (i.e., the replication constraint is respected). But \mathcal{D}' is obtained from \mathcal{D} by applying Stretch_1 on the source with R , and by applying Stretch_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 . This completes the proof of Theorem 6.1.

Given Theorem 6.1, for the remainder of subsection we will focus on obtaining useful views that minimize the set of queries that are UN non-disclosed. We say that a d-view \mathcal{V} is *UN non-disclosure minimal for CQ Q* within a class of views \mathcal{C} if: \mathcal{V} is useful for Q and for any BCQ p , if there is a d-view based on \mathcal{C} which is useful for Q and UN non-disclosing for p , then \mathcal{V} is UN non-disclosing for p . Proposition 2.5 implies that

if \mathcal{V} is minimally informative within C , then it is UN non-disclosure minimal for any BCQ Q within C .

We can use the technique in the proof of Theorem 6.1 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 name in the utility query we can get useful and UN non-disclosing views whenever this is not ruled out for trivial reasons:

THEOREM 6.2. *[UN non-disclosure minimal d-views in the presence of replication]*
If BCQ Q contains a relation name 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 . In particular, there is a view that is UN non-disclosure minimal for Q .

Thus even though we do not have minimally informative useful d-views, we have d-views that are optimal from the perspective of UN non-disclosure and utility for a fixed Q . Note that the condition on p and Q can be restated as saying that Q does not entail p .

We now begin the proof of Theorem 6.2. 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} cannot 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 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 $\text{Pair}(\mathcal{N})$ defined below.

We let \mathcal{N} denote the natural numbers. $\text{Pair}(\mathcal{N})$ is the set of terms built up from elements \mathcal{N} by the pairing function: when x and y are in $\text{Pair}(\mathcal{N})$, then so is (x, y) , the ordered pair consisting of x and y . Note that all elements in $\text{Pair}(\mathcal{N})$ have a finite height. We let height be the function taking such a term and returning the nesting depth of the pairing function. It is defined inductively as $\text{height}(x) = 0$ for $x \in \mathcal{N}$, and $\text{height}((x, y)) = \max(\text{height}(x), \text{height}(y)) + 1$. Note in particular that all elements in $\text{Pair}(\mathcal{N})$ have finite height.

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 name 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 .

We consider the transformation Stretch on a d-instance that maps each local instance to its synchronous product with $\text{canondb}(Q)$. As the name implies, this transformation is very roughly analogous to Stretch_1 and Stretch_2 from the proof of Theorem 6.1. In

each case we are translating the concrete values in an instance, but leaving the relational structure in place.

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

Definition 6.3. We define \equiv^s on s-instances as the symmetric reflexive transitive closure of \mathcal{R} defined as $I \mathcal{R} I'$ when $I = \text{Stretch}(I')$ for I, I' two s-instances.

Definition 6.4. We define \equiv_G on d-instances as the symmetric reflexive transitive closure of \mathcal{R} defined as $\mathcal{D} \mathcal{R} \mathcal{D}'$ when $\mathcal{D}' = \text{Stretch}(\mathcal{D})$ for $\mathcal{D}, \mathcal{D}'$ two d-instances.

Definition 6.5. The $\text{ECR} \equiv$ is defined as $\mathcal{D}_1 \equiv \mathcal{D}_2$ exactly when for each source s $\mathcal{D}_1^s \equiv^s \mathcal{D}_2^s$.

PROPOSITION 6.6. *For two d-instances \mathcal{D} and \mathcal{D}' with $\mathcal{D} \models Q$, $\mathcal{D} \equiv \mathcal{D}'$ 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 $\mathcal{D} \models Q$ and let us show that $\mathcal{D} \equiv_G \mathcal{D}'$.

Since $\mathcal{D}_1 \equiv \mathcal{D}_2$, for each source s , there exists i such that $\mathcal{D}_1^s = \text{Stretch}^i(\mathcal{D}_2^s)$ or $\mathcal{D}_2^s = \text{Stretch}^i(\mathcal{D}_1^s)$, where the notation Stretch^i means iterating Stretch i times. On a d-instance \mathcal{D} where the interpretation of the replicated relation name T is not empty, the minimal height of elements appearing in the first position of T needs to be equal for all sources s . Therefore, if we have distinct sources s and s' , i is a number witnessing the definition above for s and i' the number for s' then we must have $i = i'$. And we cannot have s and s' such that $\mathcal{D}_1^s = \text{Stretch}^i(\mathcal{D}_2^s)$ and $\mathcal{D}_2^{s'} = \text{Stretch}^{i'}(\mathcal{D}_1^{s'})$ with $i, i' > 0$. Therefore, when the replicated predicate is not empty, $\mathcal{D} \equiv \mathcal{D}'$ implies $\mathcal{D} \equiv_G \mathcal{D}'$.

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

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

PROPOSITION 6.7. *The view corresponding to $\text{ECR} \equiv$ 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 $\mathcal{D} \models Q$. By Proposition 6.6, we have that $\mathcal{D} \equiv_G \mathcal{D}'$. Recall that there is a homomorphism from $\text{Stretch}(\mathcal{D})$ to \mathcal{D} . Thus it is clear that if $\text{Stretch}(\mathcal{D}) \models Q$ 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 $\text{canondb}(Q)$. From $\mathcal{D} \equiv_G \mathcal{D}'$ either \mathcal{D} or \mathcal{D}' can be obtained from the other by applying Stretch , and so $\mathcal{D} \models Q$ implies $\mathcal{D}' \models Q$. \square

PROPOSITION 6.8. *The view corresponding to $\text{ECR} \equiv$ is UN non-disclosing for p .*

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

Putting together the results above we complete the proof of Theorem 6.2.

Example 6.9. We give an example of the power of Theorem 6.2, and we highlight the difference from the situation with only local constraints. Suppose we have two sources, with one binary relation name S replicated between the two, and each source having one non-replicated binary relation name, R in one source and T in the other. The utility query Q is $\exists x, y. R(x, y) \wedge S(x, y) \wedge T(x, y)$ and the secret query p is $\exists x. R(x, x)$.

Since p is not entailed by Q , Theorem 6.2 implies that there are views that are useful for Q but UN non-disclosing for p . But it is easy to see that the canonical d-view of Q is UN disclosing for p .

We show that for this particular example there do indeed exist *relational algebra views* that were useful for Q and UN non-disclosing for p . Thus in exploiting replication we can sometimes stay within a standard class of views. We now explain how to use relational algebra views in this example.

Given an instance of the R source \mathcal{I}_R , we say an *R -harmless pair* is any pair of elements (x_1, x_2) of \mathcal{I}_R , where:

- x_1 and x_2 are connected by both R and S edges;
- there is no S self-loop on x_1 ;
- x_2 has no outgoing S edges, and x_2 is the unique element that is a target of an S edge from x_1 with no outgoing S edges.

The *modification* of such a pair (x_1, x_2) is the pair (x_1, x_1) .

Our view on the R -source takes as input an instance \mathcal{I} of the R source, and returns all pairs that are modifications of R -harmless pairs, unioned with pairs that are in $S \cap R$ but are not R -harmless.

A *T -harmless pair* is a pair of elements in the T -source, defined similarly but replacing R with T . A modification of such a pair is as above. Analogously, our view on the T -source returns all pairs that are modifications of T -harmless pairs, unioned with pairs that are in $S \cap T$ but are not harmless. It is clear that these views can be expressed in relational algebra.

We first show that the views are UN non-disclosing for p . Consider an instance $\mathcal{D} = (R, T, S)$ where p holds, with S the shared relation name. We will construct an instance $\mathcal{D}' = (R', T', S')$ with the same view images, but where p does not hold. We let $V_R(\mathcal{I})$ be the content for the view for the R -source on instance \mathcal{I} , and similarly $V_T(\mathcal{I})$.

We begin our construction of \mathcal{D}' by describing the interpretation of the shared relation name S' . For each element v in either view image ($V_R(\mathcal{D})$ or $V_T(\mathcal{D})$), we create an S edge from v to a new element n_v . We also include all pairs in either view that are not self-loops.

We now turn to describing the content of R' . It includes all edges in the view $V_R(\mathcal{D})$ that are not self-loops. It also contains an edge from v to n_v if (v, v) is in $V_R(\mathcal{D})$. The content of the non-shared relation name T' is defined analogously.

It is clear that the new instance does not satisfy p . We need to show that it agrees with \mathcal{D} on each view. Note that all of the pairs (v, n_v) such that (v, v) is in $V_R(\mathcal{D})$ are R -harmless. A pair of the form (c, v) where v is one of the original nodes of the instance, cannot be R -harmless, since v has an outgoing S edge. Pairs of the form (c, n_v) where $c \neq v$ are not R -harmless because there is no S or R edge between them. Thus the R -harmless pairs are exactly those of the form (v, n_v) where $(v, v) \in V_R(\mathcal{D})$. But

the view will produce all such pairs (v, v) . We can conclude that $V_R(\mathcal{D})$ and $V_R(\mathcal{D}')$ agree on pairs of the form (v, v) . The next case to consider consists of pairs in the view $V_R(\mathcal{D})$ of the form (c, d) with $d \neq c$. By definition, such pairs are included in both S' and R' . They are not R -harmless in \mathcal{D}' , since d has an outgoing edge to n_d . Thus they are included in $V_R(\mathcal{D}')$. Conversely, if a pair (c, d) with $c \neq d$ does not occur in $V_R(\mathcal{D})$, we can argue that it is not in $V_R(\mathcal{D}')$. This follows since we will not have R' hold of (c, d) . We have thus shown that the new instance agrees with \mathcal{D} on each view.

We next show that the views are useful for Q , by arguing that the BCQ Q is equivalent to the non-emptiness of the intersection of V_R and V_T .

In one direction, we suppose Q has a match (c, d) in an instance \mathcal{D} , and we argue that (c, d) is in the intersection of $V_R(\mathcal{D})$ and $V_T(\mathcal{D})$. Note that such a pair is R -harmless if and only if it is T -harmless because $T \wedge R$ holds of it in \mathcal{D} . Thus we will distinguish pairs that are harmless (meaning R or T -harmless) versus pairs that are not harmless. If (c, d) is a harmless pair, then (c, c) will be in the intersection of V_R and V_T within \mathcal{D} . While if (c, d) is not a harmless pair, then (c, d) will be in the intersection of V_R and V_T .

We now suppose that $V_R(\mathcal{D})$ and $V_T(\mathcal{D})$ intersect, and show that Q must hold on \mathcal{D} . First, suppose that the intersection has a pair (c, d) with $d \neq c$. Then it is clear that (c, d) must be a match of Q . The more interesting case is when there is an element in the intersection of the form (c, c) . As a first subcase, suppose $S(c, c)$ holds within \mathcal{D} . Then there are no R -harmless or T -harmless pairs of the form (c, d) and thus (c, c) could not have been produced in either V_R or V_T as a modification. Hence (c, c) must have gotten into $V_R(\mathcal{D})$ because $\mathcal{D} \models S(c, c) \wedge R(c, c)$, while (c, c) was in view V_T because $\mathcal{D} \models S(c, c) \wedge T(c, c)$. Thus we have a match of Q . The second subcase is where (c, c) does not hold in S within \mathcal{D} . Then (c, c) must have been produced as a modification of an R -harmless pair (c, d) and as a modification of some T -harmless pair (c, d') . But from the uniqueness condition in R -harmlessness and T -harmlessness, we conclude that $d = d'$. Now (c, d) is a match of Q . \triangleleft

6.2 Negative results on UN non-disclosure minimality

Theorem 6.2 shows that we can still find UN non-disclosure minimal d-views even in the presence of replication. We now proceed to show some restrictions on what one can achieve even in terms of this more restricted notion of “minimally expressive” d-view.

Separating the power of general and isomorphism-invariant d-views. The views used in Theorem 6.2 are not isomorphism-invariant: like the views from Theorem 6.1, 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, even to achieve this weaker notion of minimality, it may be essential to use d-views based on queries that are not isomorphism-invariant.

THEOREM 6.10. *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. In particular, we cannot find a UN non-disclosure minimal d-view that is isomorphism-invariant in the above sense.*

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

For example, any query in relational algebra has the isomorphism-invariance property above. We now proceed with the proof of Theorem 6.10. Consider the following Boolean CQs (existentially quantifiers omitted):

$$\begin{aligned} Q &= P(x) \wedge P(y) \wedge S(y) \wedge S(z) \wedge T(z) \wedge T(x) \wedge R(w) \\ p &= S(x) \wedge T(x) \wedge P(x) \end{aligned}$$

It is easy to see that Q does not entail p . Since there is a replicated relation name mentioned in the query, Theorem 6.2 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 whose values lie in the active domain, and where the queries in the view definitions satisfy the isomorphism-invariance property. Our proof will involve two auxiliary propositions about this setting.

PROPOSITION 6.11. *Suppose d-view \mathcal{V} is useful for Q and satisfies the active domain and isomorphism-invariance properties above. Given two instances I_1 and I_2 for the source whose unreplicated relation name is G , if I_1 and I_2 agree on the replicated relation name and agree on each view in \mathcal{V} for this source, then they must agree on G .*

PROOF. Let $c \in G(I_1)$ and form \mathcal{D}_1 by choosing a context for the other unshared relation names with the interpretation of each relation name containing only c . Let \mathcal{D}_2 be formed similarly from 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 useful for Q . Clearly the witness can only be c , thus $c \in G(I_2)$. \square

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

PROPOSITION 6.12. *Let \mathcal{V} be views that are useful for Q , satisfying isomorphism-invariance property and returning only elements in the active domain. Let \mathcal{D} agree with \mathcal{D}_0 on \mathcal{V} . For any unreplicated relation name G , we have $\mathcal{D} \models \forall x ((G(x) \wedge x \neq *) \rightarrow R(x))$.*

PROOF. Let I be the restriction of \mathcal{D} to the G source, and suppose $G(v)$ holds in I for $v \neq *$. Let c be a fresh value and I' be the result of applying an isomorphism swapping v and c . Since $v \neq *$, the isomorphism-invariance property implies that I and I' agree on the views. Because I' and I disagree on G , Proposition 6.11 implies they cannot agree on the replicated relation name. Note that c was fresh (hence in particular was not in the interpretation of R within I), and no other change occurred in R outside of the swap of v and c . Thus the only way that I and I' can disagree on R is because $R(v)$ held in I . \square

With these propositions in hand we are ready to complete the argument for Theorem 6.10. 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 6.12 implies that they are both in R . Since y_0 and z_0 are not in $\text{adorn}(\mathcal{V}_0)$, 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 6.11 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 .

This completes the proof of Theorem 6.10.

Separating the power of Relational Algebra-based d-views and DCQ-based d-views. In Theorem 6.10 we have seen that the UN non-disclosure minimal d-view may not be isomorphism-invariant. In particular, the theorem shows we may be able to find a d-view that balances supporting the querying of a CQ Q while avoiding UN disclosure of CQ p , but we may not be able to do it with an isomorphism-invariant d-view. We will now give another separation result, showing that we may be able to design a d-view that is useful for Q and UN non-disclosing for p based on relational algebra, while not being able to construct such a d-view with DCQ views. This contrasts again with the situation without replication, where the minimal information, and hence the minimal non-disclosure, can always be achieved by DCQ-based views.

We will prove a more general result. Instance I_1 is a *subinstance* of I_2 if I_1 is a subset of I_2 when they are seen as sets of facts. In other words, for every relation name, its interpretation in I_1 is a subset of its interpretation in I_2 . A view V is *monotone* if the output whenever I_1 is a subinstance of I_2 , the output of V on I_1 is a subset of the output on I_2 . A set of views \mathcal{V} is monotone if each view in the set is monotone. Note that UCQs, DCQs, along with their extensions with inequalities, are all monotone. There are a number of powerful query languages that extend UCQs, such as Datalog [1], that also define only monotone queries.

THEOREM 6.13. *There is a d-schema with replication constraints along with BCQs Q and p , such that there is a relational algebra based d-view that is useful for Q and UN non-disclosing for p , but none based on monotone views.*

PROOF. We use the d-schema Q , and p from Example 6.9. We have already shown that there is a relational algebra based d-view that is useful for Q and UN non-disclosing for p , so we focus on showing that this cannot be done with monotone views.

Say that an element d *appears non-trivially* in a k -ary relation S if there is a tuple t in S with $t_i = d$, a t' formed from t by setting t'_i to $d' \neq d$ while $t_j = t_i$ for $j \neq i$, such that t' is not in S . For the output of a CQ or UCQ view, there is no difference between appearing non-trivially and being in the active domain of the view output. But for unsafe views there is a difference, since it is possible that every element appears in the output, but if d appears non-trivially in the output of a DCQ then there must be some disjunct in the DCQ such that d satisfies the disjunct.

For elements c and d let $I_{c,d}$ be the instance where each relation name is interpreted by the single pair (c, d) . We first claim that on this instance each of c, d needs to be

appear non-trivially in the output of some view. We show the claim for d , with the claim for c being symmetric.

Suppose not, and fix e distinct from both c and d . Consider $I_0 = I_{c,d} \cup I_{c,e}$ along with I_1 the instance with only (c, d) in R , both (c, d) and (c, e) in S and in S but only $(c, e) \in T$. If d does not occur non-trivially in the view output in $I_{c,d}$, the views must return the same result on $I_{c,e}$ and $I_{c,d}$. Now I_1 is a subinstance of $I_{c,d}$ on the R source, and a subinstance of $I_{c,e}$ on the S source. Thus by monotonicity of the views, the output of each view on I_1 is contained in the corresponding output on I_0 . On the other hand, since I_0 is a subinstance of I_1 , the view outputs must be identical on I_1 and I_0 . But since Q holds on I_0 and not on I_1 , this contradicts usefulness of the views.

Now consider an instance of the form $I_{c,c}$ for an element c . The secret query p holds, and Q holds. So by UN non-disclosure there must be an instance I' with the same view image as $I_{c,c}$ where p fails and Q holds. Since Q holds, I' must contain $I_{e,f}$ for some $e \neq f$. By the assertions above, there must be a view where e appears non-trivially in its output and also a view where f appears non-trivially. Clearly, for each view output on $I_{c,c}$, only c can appear non-trivially. Since one of e, f must be distinct from c , this is a contradiction of the fact that $I_{c,c}$ and I' must agree on the views.

This completes the proof of Theorem 6.13. \square

Lack of UN disclosure minimality d-views when restricting to CQ views. We now turn to another limitation of Theorem 6.2. It states that there are d-views minimal for UN disclosure even in the presence of replication. But it tells us nothing about whether there are d-views that are minimal for UN disclosure in the presence of replication within the class of CQ-based d-views. Unfortunately, there we can get quite strong negative results.

THEOREM 6.14. *There is a schema with a replication constraint and a utility query Q such that there is no CQ-based d-view \mathcal{V} that is minimal for UN non-disclosure within the class of CQ views. In particular, there is no minimally informative useful d-view for Q within the class of CQ-based views.*

Note that this does not follow from Theorem 6.10, which only concerns UN non-disclosure minimality within the class of all views.

We will prove a stronger statement, along the lines of what is shown in Subsection 4.2. Recall from there that a query Q is homomorphism-invariant when, for all pairs of instances I and I' and all homomorphisms μ from I to I' then $\mu(Q(I)) \subseteq Q(I')$ and adom -based when $\text{adom}(Q(I)) \subseteq \text{adom}(I)$. We will show that views based on such queries cannot be minimally informative.

Our schema has two sources, \mathcal{P} and \mathcal{S} . \mathcal{P} contains a binary relation name P and \mathcal{S} contains a binary relation S . Both sources contain a shared binary relation name T . The utility query is $Q = \exists w, x, y, z. T(x, y) \wedge S(y, z) \wedge T(z, w) \wedge P(w, x)$. It is illustrated in Figure 2.

We consider three secrets:

- $p_1 = \exists x. S(x, x)$,
- $p_2 = \exists x, y. T(x, y) \wedge S(y, x)$,
- $p_3 = \exists x, y, z. T(x, y) \wedge S(y, z) \wedge T(z, x)$.

It is easy to see that, for each of these secrets, there exists a d-view based on CQs that is useful for Q and UN non-disclosing for the secret. Indeed, we have:

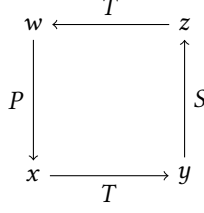


Fig. 2. Query Q counter-example for theorem 6.14

- $Q_S(x, w) = \exists y, z. T(x, y) \wedge S(y, z) \wedge T(z, w)$ and $Q_P(w, x) = P(w, x)$ for p_1 ,
- $Q_S(y, w) = \exists z. S(y, z) \wedge T(z, w)$ and $Q_P(w, y) = \exists y. P(w, x) \wedge T(x, y)$ for p_2
- $Q_S(y, z) = S(y, z)$ and $Q_P(y, z) = \exists w, x. T(y, w) \wedge P(w, x) \wedge T(x, y)$ for p_3 .

Now consider a d-view \mathcal{V} , useful for Q , homomorphism-invariant, and adom-based. We will show that \mathcal{V} is necessarily UN disclosing for one of the secrets among p_1, p_2, p_3 . We now claim that:

LEMMA 6.15. *Any d-view \mathcal{V} that is useful for Q , homomorphism-invariant, and adom-based must necessarily be UN disclosing for one of the secrets among p_1, p_2, p_3 .*

The remainder of this subsection is devoted to the proof of this lemma, which completes the theorem. The proof will follow the technique used in Proposition 4.15.

Let us consider the following d-instance \mathcal{D}_Q shown in Figure 3. This is nothing more than an isomorphic copy of $\text{canondb}(Q)$. Recall that \mathcal{V}^S denotes the component of the d-view \mathcal{V} for source S , and thus $\mathcal{V}^S(\mathcal{D}_Q)$ denotes the view image of the instance \mathcal{D}_Q under this view.

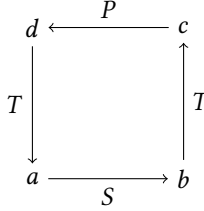


Fig. 3. Instance \mathcal{D}_Q

CLAIM 4. $\text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$ includes either a or d .

PROOF. Let us consider the instance \mathcal{I}_1 depicted in Figure 4, and let us show that

- (1) $\text{adom}(\mathcal{V}^S(\mathcal{I}_1))$ includes a or d if and only if $\text{adom}(\mathcal{V}^S(\mathcal{D}))$ includes a or d , and
- (2) $\text{adom}(\mathcal{V}^S(\mathcal{I}_1))$ includes either a or d .

For the first item since $\mathcal{D}_Q \subseteq \mathcal{I}_1$ we know that $\text{adom}(\mathcal{V}^S(\mathcal{D}_Q)) \subseteq \text{adom}(\mathcal{V}^S(\mathcal{I}_1))$. Consider the homomorphism μ that is the identity on the elements a, b, c and d but sends f to d , e to a , h to c , and g to b .

The image of \mathcal{I}_1 is \mathcal{D}_Q and, since $\mu(\mathcal{I}_1) = \mathcal{D}_Q$, we know from homomorphism-invariance that $\mu(\mathcal{V}^S(\mathcal{I}_1)) \subseteq \mathcal{V}^S(\mathcal{D}_Q)$. If $\text{adom}(\mu(\mathcal{V}^S(\mathcal{I}_1)))$ contains a or d , $\text{adom}(\mu(\mathcal{V}^S(\mathcal{D}_Q)))$

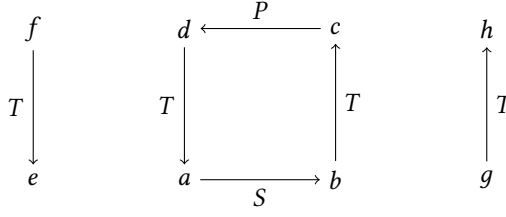


Fig. 4. Instance I_1

contains a or d . Note that this also proves that when e or f in $\text{adom}(\mu(\mathcal{V}^S(I_1)))$, a or d are in $\text{adom}(\mu(\mathcal{V}^S(I_1)))$.

To prove that $\text{adom}(\mathcal{V}^S(I_1))$ includes a or d , let us consider the instance I_2 , depicted in Figure 5. This instance is obtained from I_1 by replacing the S source with its image through the isomorphism ν that exchanges e and a and f and d .

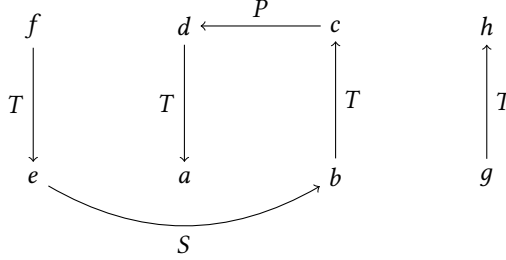


Fig. 5. Instance I_2

If $\{a, d\} \cap \text{adom}(\mathcal{V}^S(I_1)) = \emptyset$, then $\{f, e\} \cap \text{adom}(\mathcal{V}^S(I_1)) = \emptyset$. And if $\{a, d, e, f\} \cap \text{adom}(\mathcal{V}^S(I_1)) = \emptyset$, then $\mathcal{V}^S(I_1) = \nu(\mathcal{V}^S(I_1))$. Since ν is an isomorphism between I_1^S and I_2^S , $\mathcal{V}^S(I_1) = \nu(\mathcal{V}^S(I_1)) = \mathcal{V}^S(\nu(I_1)) = \mathcal{V}^S(I_2)$. And as the \mathcal{P} sources are identical we conclude that $\mathcal{V}(I_1) = \mathcal{V}(I_2)$. This implies that \mathcal{V} is not useful for Q as $Q \models I_1$ but $Q \not\models I_2$, a contradiction. \square

CLAIM 5. $\text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$ contains either b or c .

PROOF. Same proof as above (by symmetry). \square

CLAIM 6. \mathcal{V} is UN-disclosing:

- (1) for p_1 when $\{a, b\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$;
- (2) for p_2 when $\{d, b\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$;
- (3) for p_2 when $\{a, c\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$;
- (4) for p_3 when $\{d, c\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$.

PROOF. Consider the critical d-instance, $I = \{T(*, *), S(*, *), P(*, *)\}$, and take a d-instance \mathcal{E} such that $\mathcal{V}(\mathcal{E}) = \mathcal{V}(I)$. Since \mathcal{V} is useful for Q , there exists x, y, z, w such that $Q = T(x, y) \wedge S(y, z) \wedge T(z, w) \wedge P(w, x)$ holds in \mathcal{E} (the x, y, z and w being not necessarily different).

Let ξ be the homomorphism from \mathcal{D}_Q to \mathcal{E} that maps d to x , a to y , b to z , and c to w . ξ maps \mathcal{D}_Q to a subset of \mathcal{E} and this proves that $\xi(\mathcal{V}^S(\mathcal{D}_Q)) \subseteq \mathcal{V}^S(\mathcal{E}) = \mathcal{V}^S(\mathcal{I})$. But $\text{adom}(\mathcal{V}^S(\mathcal{I})) = \{*\}$. We now do a case analysis, depending on the four cases of the claim.

- Suppose $\{a, b\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$. By homomorphism invariance, $\{\xi(a), \xi(b)\} \subseteq \text{adom}(\mathcal{V}^S(\xi(\mathcal{D}_Q))) = \text{adom}(\mathcal{V}^S(\mathcal{I})) = \{*\}$, and thus $y = \xi(a) = * = \xi(b) = z$ which means that p_1 holds in \mathcal{E} .
- Suppose $\{d, b\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$. Then $x = \xi(d) = \xi(b) = z$, and thus p_2 holds in \mathcal{E} .
- Suppose $\{a, c\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$. Then $y = \xi(a) = \xi(c) = w$, and thus p_2 holds in \mathcal{E} .
- Suppose $\{d, c\} \subseteq \text{adom}(\mathcal{V}^S(\mathcal{D}_Q))$. Then $x = \xi(d) = \xi(c) = w$, and thus p_3 holds in \mathcal{E} .

Since \mathcal{E} was an arbitrary instance agreeing with the views, we have shown that \mathcal{V} is UN-disclosing for one of the secrets p_1, p_2, p_3 . \square

By combining claims 4, 5, and 6, we get the proof of Lemma 6.15.

Theorem 6.14 follows easily from Lemma 6.15. Suppose there were a set of views \mathcal{V} in the class that were minimal for UN non-disclosure. Since for each $i \in \{1, 2, 3\}$, there are d-views based on CQs that are useful for Q and non-disclosing for p_i , \mathcal{V} could not disclose any of p_1, p_2, p_3 . But this contradicts Lemma 6.15.

7 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 note that our main results hold for the variant of the problem where quantification in the utility and non-disclosure definitions is considered over all instances and the one where the quantification is considered over only finite instances. There has been a long-running discussion over which quantification is to be preferred. We adopt the version with unrestricted instances as a default, following quite a number of works in this area (e.g. [22]). This is primarily for technical convenience, since it allows us to employ the chase as a technique without assuming termination, which provides more intuitive proofs in certain cases. In Section 4 we show that the analysis using shuffles can provide alternative – albeit more complex – proofs of many of our main results without relying on infinite instances. However, we note one case where we have not explored the situation when the quantification is over finite instances: Theorem 5.5 in the case of local constraints that do not have terminating chase.

We now look first at the high-level take-aways of our results, and future directions based on these. We then turn to some formal problems left open by our work.

Outlook for practice. Standard approaches to database privacy give guarantees that rely on limiting the computational resources of the attacker, or on modelling disclosure mechanisms probabilistically, with the corresponding guarantees being probabilistic. The framework in this work is radically different in that disclosure mechanisms and

secrets are modelled deterministically. The corresponding utility and privacy guarantees are incomparable to prior work, but they are strong in many senses; e.g. the views must allow the utility query to be answered exactly on every instance. Our results in Section 3 can be considered negative, and a validation of the current approaches: If we use CQ views the only secret queries we can protect in this model are the obvious ones. The main results in Section 4 are nearly as discouraging. There we show, surprisingly, that by exploiting symmetries in the utility query we can protect some secrets by using some non-trivial views. But these are the only secrets we can preserve, and they are hardly natural. Section 5 shows that this pessimistic story line extends to the presence of local constraints.

The results in Section 6 are more encouraging. We show that in the presence of replication that non-trivial trade-offs can be achieved even for very simple utility queries, with views that are quite involved. Like any non-local constraints, replication constraints must rely on some communication mechanism between sources for their enforcement. These results can thus be interpreted as saying that if such a replication mechanism is in place, a large class of queries can be supported without revealing more than the minimal set of secrets – those that are entailed. However the results are limited to UN non-disclosure and it remains to be seen whether they can be extended to non-disclosure functions that are more interesting for practice.

Outlook for foundations of view design. We see the main significance of our results more broadly as making a contribution to understanding view design. One long-term goal would be a framework for specifying properties of views and a corresponding set of algorithms for synthesizing views that meet a given specification. Recent research [23] has indicated that determinacy can play a fundamental role in ordering views by expressiveness, and so it is natural to first gain a better grasp of this ordering. Our notion of minimal information view and our characterization of what such views look like in special cases gives new insight into determinacy. It can be seen as an – admittedly very small – step towards a framework for view synthesis.

Open questions. In this work we consider only a limited setting; e.g. CQs for the utility query and the secret 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.

Even in this restricted setting, our contribution focuses primarily on expressiveness, leaving open many questions of decidability and complexity. In particular we do not know whether the Σ_2^P bound of Corollary 3.6 is tight. Nor do we know whether the analogous question for arbitrary views – whether there is an arbitrary d-view that is useful for a given Q but UN non-disclosing for a given p – is even decidable. Our results reduce this to a non-disclosure question for the shuffle views.

We have restricted to constraints that are TGDs within this paper. For our results on CQ-based views it is routine to extend to Equality-Generating Dependencies (EGDs) as well, since we rely only on some properties of the chase that hold for EGDs as well as TGDs. However, as we have shown in Example 5.4, the results on CQ-based views do not generalize to other common classes of integrity constraints. On the other hand, we believe our results on arbitrary views extend to the presence of any background knowledge. We defer the investigation of the borderline to future work.

REFERENCES

- [1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
- [2] Marcelo Arenas, Pablo Barceló, and Juan L. Reutter. 2011. Query Languages for Data Exchange: Beyond Unions of Conjunctive Queries. *Theory Comput. Syst.* 49, 2 (2011), 489–564.
- [3] Jean-François Baget, Fabien Garreau, Marie-Laure Mugnier, and Swan Rocher. 2014. Extending Acyclicity Notions for Existential Rules. In *ECAI*.
- [4] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. 2011. Walking the Complexity Lines for Generalized Guarded Existential Rules. In *IJCAI*.
- [5] Vince Bárány, Michael Benedikt, and Balder Ten Cate. 2018. Some Model Theory Of Guarded Negation. *The Journal of Symbolic Logic* 83, 4 (2018), 1307–1344.
- [6] John Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel N. Kho, and Jennie Rogers. 2017. SMCQL: Secure Query Processing for Private Data Networks. In *VLDB*.
- [7] Michael Benedikt, Pierre Bourhis, Louis Jachiet, and Michaël Thomazo. 2019. Reasoning about Disclosure in Data Integration in the Presence of Source Constraints. In *IJCAI*.
- [8] Michael Benedikt, Pierre Bourhis, Balder ten Cate, and Gabriele Puppis. 2016. Querying Visible and Invisible Information. In *LICS*.
- [9] Michael Benedikt, Bernardo Cuenca Grau, and Egor V. Kostylev. 2018. Logical foundations of information disclosure in ontology-based data integration. *Artif. Intell.* 262 (2018), 52–95.
- [10] Michael Benedikt, Balder ten Cate, and Efi Tsamoura. 2016. Generating plans from proofs. In *TODS*.
- [11] Piero A. Bonatti and Luigi Sauro. 2013. A Confidentiality Model for Ontologies. In *ISWC*.
- [12] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2012. View-based Query Answering in Description Logics: Semantics and Complexity. *J. Comput. Syst. Sci.* 78, 1 (2012), 26–46.
- [13] David Chaum, Claude Crépeau, and Ivan Damgard. 1988. Multiparty Unconditionally Secure Protocols. In *STOC*.
- [14] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. 2013. Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies. *JAIR* 47 (2013), 741–808.
- [15] A. Deutsch, A. Nash, and J. Remmel. 2008. The Chase Revisited. In *PODS*.
- [16] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*.
- [17] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. & Trends in Th. Comp. Sci.* 9, 3&4 (Aug. 2014), 211–407.
- [18] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. 2005. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science* 336, 1 (2005), 89–124.
- [19] Tomasz Gogacz and Jerzy Marcinkowski. 2015. The Hunt for a Red Spider: Conjunctive Query Determinacy Is Undecidable. In *LICS*.
- [20] Tomasz Gogacz and Jerzy Marcinkowski. 2016. Red Spider Meets a Rainworm: Conjunctive Query Finite Determinacy Is Undecidable. In *PODS*.
- [21] Alon Y. Halevy. 2001. Answering queries using views: A survey. *VLDB J.* 10, 4 (2001), 270–294.
- [22] David S. Johnson and Anthony C. Klug. 1984. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *JCSS* 28, 1 (1984).
- [23] Paraschos Koutiris, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, and Dan Suciu. 2015. Query-Based Data Pricing. *J. ACM* 62, 5 (2015).
- [24] Chao Li, Daniel Yang Li, Gerome Miklau, and Dan Suciu. 2017. A theory of pricing private data. *Commun. ACM* 60, 12 (2017), 79–86.
- [25] D. Maier, A. O. Mendelzon, and Y. Sagiv. 1979. Testing implications of data dependencies. *TODS* 4, 4 (1979), 455–469.
- [26] B. Marnette. 2009. Generalized schema-mappings: from termination to tractability. In *PODS*.
- [27] Alan Nash and Alin Deutsch. 2007. Privacy in GLAV Information Integration. In *ICDT*.
- [28] Alan Nash, Luc Segoufin, and Victor Vianu. 2010. Views and queries: Determinacy and rewriting. *TODS* 35, 3 (2010).
- [29] A. Onet. 2013. The Chase Procedure and its Applications in Data Exchange. In *DEIS*. 1–37.