Schema Mapping Discovery from Data Instances

6

GEORG GOTTLOB

University of Oxford, Oxford, United Kingdom

AND

PIERRE SENELLART

Institut Télécom; Télécom ParisTech; CNRS LTCI, Paris, France

Abstract. We introduce a theoretical framework for discovering relationships between two database instances over distinct and unknown schemata. This framework is grounded in the context of *data exchange*. We formalize the problem of understanding the relationship between two instances as that of obtaining a schema mapping so that a *minimum repair* of this mapping provides a perfect description of the target instance given the source instance. We show that this definition yields "intuitive" results when applied on database instances derived from each other by basic operations. We study the complexity of decision problems related to this optimality notion in the context of different logical languages and show that, even in very restricted cases, the problem is of high complexity.

Categories and Subject Descriptors: F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; H.2.5 [**Database Management**]: Heterogeneous Databases

General Terms: Languages, Theory

Additional Key Words and Phrases: Schema mapping, instance, complexity, match, data exchange

ACM Reference Format:

Gottlob, G., and Senellart, P. 2010. Schema mapping discovery from data instances. J. ACM 57, 2, Article 6, (January 2010), 37 pages.
DOI = 10.1145/1667053.1667055 http://doi.acm.org/10.1145/1667053.1667055

G. Gottlob's work was supported by EPSRC grant EP/E010865/1 "Schema Mappings and Automated Services for Data Integration and Exchange". G. Gottlob also gratefully acknowledges a Royal Society Wolfson Research Merit Award, which allowed him to host Pierre Senellart. P. Senellart's work was supported by the European Research Council grant Webdam (under FP7), grant agreement 226513.

Authors' addresses: G. Gottlob, Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, England, United Kingdom, e-mail: georg.gottlob@comlab.ox.ac.uk; P. Senellart, Télécom ParisTech, Département Informatique et Réseaux, 46 rue Barrault, 75634 Paris Cedex 13, France, e-mail: pierre.senellart@telecom-paristech.fr.

DOI 10.1145/1667053.1667055 http://doi.acm.org/10.1145/1667053.1667055

1. *Introduction*

Main Problem Addressed. This article deals with the automatic discovery of relational schema mappings based on existing data. In particular, we deal with the following main problem, and with closely related questions. Given a relational schema **S**, called the *source schema*, and a differently structured *target schema* **T**, and given an instance I of **S**, and an instance J of **T**, where we assume that J consists of an adequate "translation" of I to fit the target schema **T**, find an optimal translation function, that is, a schema mapping that maps instances of **S** to instances of **T**, taking I to J. This main problem actually consists of two important subproblems: (i) determining an appropriate formal framework in which schema mappings can be expressed and that allow one to numerically assess the quality of a schema mapping, and (ii) understanding the computational fundamentals of the automatic synthesis of schema mappings in that framework, in particular, complexity issues and algorithms. We do not provide a direct algorithm for deriving schema mappings from database instances, but we discuss a theoretical framework and complexity analysis that can be a first step toward it.

Importance of the Problem. Schema mapping discovery has been recognized as an issue of central relevance in various application and research contexts, for example, in the area of data exchange [Fagin et al. 2003; Kolaitis 2005], data integration [Lenzerini 2002; Haas et al. 2005], metadata management [Bernstein 2003], and data extraction from the hidden Web, in particular, automated wrapper generation.

Schemata and dependencies, in a data exchange context, form *metadata* that need to be managed in a systematic and formal way. Bernstein [2003] argues for the definition, in such a setting, of *operators* on this metadata. Thus, Fagin et al. [2004; 2007], respectively propose ways to define the composition and inverse operators on schema mappings. Another operator of importance, which is actually closely related to the research proposed in the present paper, is the *match* operator [Bernstein 2003]: given two schemata and instances of these schemata, how to derive an appropriate set of dependencies between these schemata. More precisely, given two relational databases schemata **S** and **T** and instances I and J of these schemata, the problem is to find a *schema mapping*, that is, a finite set Σ of formulas in a given language $\mathcal{L}$, such that $(I, J) \models \Sigma$, or such that (I, J) *approximates* in some sense a model of Σ. This problem is related to the techniques used for *automatic schema matching* in data exchange. Current methods of automated schema mapping generation, such as those described in Rahm and Bernstein [2001] and Haas et al. [2005], heavily rely on semantic meta-information about the schemata (names of concepts and relations, concrete data types, etc.). However, in many practical contexts such semantic meta-information is either not available or would require too much or too expensive human interaction. In those cases, the mere values of I and J constitutes the only information from which a mapping ought to be constructed. This important case, which has been barely studied, is addressed in the present article. Obviously, such a study is also a first step towards schema matching systems that use both schema-based and data-based information.

In automated wrapper generation (e.g., from Web sources), the absence of suitable meta-information is a similar problem. Let us take a concrete example, namely that of extracting information from research publications databases on the Web.

Consider for instance the list of publications by J. D. Ullman provided by DBLP[1] and Google Scholar.[2] A structural information extraction wrapper such as RoadRunner [Crescenzi et al. 2001] can be applied on both pages (or set of pages obtained by following the *Next* links) to obtain relational data from these pages, *without any metadata*. The set of papers presented by both sources is not exactly the same, their organization is different (e.g., grouping by dates in DBLP), some data is present in some source and not in the other (page numbers in DBLP, direct links to electronic versions of articles in Google Scholar), but both sources essentially present information about the same data. Using the mutual redundancy of these sources to detect the most appropriate schema mapping between them would enable us to wrap from one format of output to another. If the structure of one source is known, then this schema mapping would give us the core structure of the other one, in a fully automatic way.

Results. After stating in Section 2 some relevant definitions, in Section 3 of this article, we present a novel formal framework for defining and studying the automatic discovery of schema mappings. In this framework, schema mappings are—not surprisingly—expressed as source-to-target tuple-generating dependencies (tgds). It is well known that tgds are suited for this task. However, we also introduce a cost function, that tells us how well a tgd does its job of translating the given source instance I into the given target instance J. This cost function takes into account (i) the size of the tgd, (ii) the number of *repairs* that have to be applied to the tgd in order for it to be valid and to perfectly explain all facts of J.

Of course, defining a cost function may be seen as a somewhat arbitrary choice. However, in Section 4, we give formal evidence of the appropriateness of the cost function, showing that it enjoys nice properties when I and J are derived from each other with elementary relational operations.

We study in Section 5 the computational complexity of the relevant problems. In particular, we show that computing the cost of a schema mapping lies at the third level of the polynomial hierarchy, while either fullness or acyclicity conditions reduce this complexity by one level. The problem is thus **NP**-complete for full acyclic tgds, and it remains **NP**-complete even in a very simple case where there is only one relation of arity 3 in the source schema, and one relation of arity 1 in the target schema. To see that, we use a lemma on the complexity of the Vertex-Cover problem in r-partite r-uniform hypergraphs, which is interesting in itself.

We finally discuss in Section 6 an extension and variants of this approach. We show in particular how the cost definition can be extended to a schema mapping expressed as an arbitrary first-order formula and discuss the complexity of the relevant problems. We also examine other choices for the cost function, which may seem simpler at first and closer to the existing notion of *repair* of a database in the literature [Arenas et al. 1999], but which are not appropriate to our context since they do not share the same "niceness" properties established in Section 4.

Related Work. We are not aware of any work with the same focus on a theoretical and systematic analysis of the complexity of deriving a schema mapping, although,

[1] `http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/u/Ullman:Jeffrey_D=.html`.

[2] `http://scholar.google.com/scholar?q=author%3A%22jd+ullman%22`.

in spirit, the problem that we deal with here is similar to the one that inductive logic programming (ILP) [Lavrač and Džeroski 1994] aims to solve. An approach to the complexity of ILP is presented in Gottlob et al. [1997]; the main differences with the work discussed here is the use of negative examples, the existence of a background knowledge, and the restriction to Horn clauses instead of arbitrary tgds.

The notion of *repair* appears in the context of inconsistent databases [Arenas et al. 1999] (with respect to some integrity constraint). In this work, consistent query answers are defined as the common answers to a query on minimal repairs of the database. Repairs use the addition or deletion of tuples to the database, something close to what is discussed in Section 6 and that we show inappropriate to our context. Besides, the focus is different: Arenas et al. [1999] suppose the integrity constraint fixed, while we are looking for an optimal schema mapping without any *a priori*.

Finally, note that the work presented here appeared in an abridged form in Senellart and Gottlob [2008]. Most proofs appear here for the first time, as well as the formal statement of Theorem 4.1, the distinction between complexity for a language and the language of its repairs, and the various **DP**-hardness results.

2. *Preliminaries*

We assume some countably infinite sets $\mathcal{C}$ of constants (denoted a, b, 0, 1, etc.) and $\mathcal{V}$ of variables (denoted x, y, z, etc.). We use the notation $\mathbf{x}$ to represent a vector of variables $x_1 \ldots x_n$. Constants appearing in formulas are here identified, as usual, with the domain elements they are interpreted by.

A (relational) schema is a finite set of pairs (R, n) where R is a relation name and $n \geqslant 1$ the arity of the relation. An instance I of a relational schema $\mathbf{S}$ consists, for every $(R, n) \in \mathbf{S}$, of a *finite* relation over $\mathcal{C}^n$. We occasionally denote R^I the interpretation of the relation name R in the instance I (if $|\mathbf{S}| = 1$, we shall make the confusion $R^I = I$). In the following, we assume that the schemata are implicitly given whenever we are given an instance.

A *language* $\mathcal{L}$ is a subset of the set of formulas of first-order logic with equality and constants, and without function symbols (with its usual semantics). Given a language $\mathcal{L}$, a *schema mapping* in $\mathcal{L}$ is a finite set of formulas in $\mathcal{L}$. We are particularly interested in the following languages, given instances I, J with schemata $\mathbf{S}$, $\mathbf{T}$:

Relational Calculus. $\mathcal{L}_{\text{rc}}$ is the set of first-order formulas without constants or equalities, with relations symbols in $\mathbf{S} \cup \mathbf{T}$.

Source-to-Target Tuple-Generating Dependencies. $\mathcal{L}_{\text{tgd}} \subset \mathcal{L}_{\text{rc}}$ is the set of formulas of the form

$$\forall \mathbf{x}\ \varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}),$$

where: (i) $\varphi(\mathbf{x})$ is a (possibly empty) conjunction of positive relation atoms, with relation symbols in $\mathbf{S}$; (ii) $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of positive relation atoms, with relation symbols in $\mathbf{T}$; (iii) all variables of $\mathbf{x}$ appear in $\varphi(\mathbf{x})$. As for $\mathcal{L}_{\text{rc}}$, we assume that no constants or equality relations appear in the formulas. This choice is discussed in more detail in Section 3, when repairs of a tgd are introduced.

Acyclic tgds. $\mathcal{L}_{\text{acyc}} \subset \mathcal{L}_{\text{tgd}}$ is the set of tgds such that (i) the hypergraph of the relations on the left-hand side is acyclic [Beeri et al. 1981]; and (ii) the hypergraph of the relations on the right-hand side, considering only existentially quantified variables, is also acyclic. More precisely, the tgd is said to be acyclic if there are

two forests F and F' (called the *join forests*) with each relation atom of $\varphi(\mathbf{x})$ (respectively, of $\psi(\mathbf{x}, \mathbf{y})$) a node of F (respectively, F'), such that for all variables $v \in \mathbf{x}$ (respectively, $v \in \mathbf{y}$), the subgraph of F (respectively, F') induced by the atoms of $\varphi(\mathbf{x})$ (respectively, $\psi(\mathbf{x}, \mathbf{y})$) that contain v is connected. Consider for example the tgds:

$$\theta_1 = \forall x\ R(x) \rightarrow \exists y\ R(x, y)$$
$$\theta_2 = \forall x \forall y \forall z\ R(x, y) \wedge R(y, z) \wedge R(z, x) \rightarrow R'(x, y, z)$$

The tgd θ_1 is acyclic, while θ_2 is not. Other equivalent definitions of acyclic hypergraphs are given in Beeri et al. [1981]. Note that this notion of acyclicity is not related to the notion of weakly acyclic sets of tgds that has been much studied in the context of data exchange [Fagin et al. 2003; Kolaitis 2005].

Full tgds. $\mathcal{L}_{\text{full}} \subset \mathcal{L}_{\text{tgd}}$ is the set of tgds without an existential qualifier on the right-hand side, that is, of the form

$$\forall \mathbf{x}\ \varphi(\mathbf{x}) \rightarrow \psi(\mathbf{x}).$$

Acyclic Full tgds. $\mathcal{L}_{\text{facyc}} = \mathcal{L}_{\text{acyc}} \cap \mathcal{L}_{\text{full}}$ is the set of full tgds such that the hypergraph of the relations on the left-hand side is acyclic.

We focus here on source-to-target tuple-generating dependencies (either arbitrary or with one of the restrictions mentioned above). Arbitrary tgds (and, in a lesser way, full tgds) have been at the basis of most works[3] in the data exchange setting [Fagin et al. 2003; Kolaitis 2005]. As we shall see in Section 5, acyclic full tgds have nice complexity results. We show in Section 6 how this work can be extended to arbitrary formulas of the relational calculus.

3. *Cost and Optimality of a tgd*

We first introduce the two basic notions of *validity* and *explanation* that are at the basis of our framework.

Definition 3.1. A schema mapping Σ is *valid* with respect to a pair of instances (I, J) if $(I, J) \models \Sigma$.

Definition 3.2. A (ground) fact in a schema $\mathbf{S}$ is a tuple $R(c_1 \ldots c_n)$ where $c_1 \ldots c_n \in \mathcal{C}$ and R is a relation of $\mathbf{S}$ with arity n.

A schema mapping Σ *explains* a ground fact f in the target schema with respect to a source instance I if, for all instances K of the target schema such that $(I, K) \models \Sigma$, $f \in K$.

A schema mapping *fully explains* a target instance J with respect to a source instance I if it explains all facts of J with respect to I.

We have quite an asymmetric point of view about the pair of instances here; we do not require a full explanation of I by the facts of J, for instance. This asymmetry is quite common in the context of data exchange. For source-to-target tgds, note that Σ fully explains J with respect to I if and only if J is included in the result of the chase [Fagin et al. 2003] of I by Σ.

[3] The other important class of dependencies, namely equality generating dependencies, is less appropriate to this context.

Example 3.3. Let us consider the following database instances I and J, on schemata $\{(R, 1)\}$ and $\{(R', 2)\}$.

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

We can imagine a number of schema mappings that more or less express the relation between I and J:

$$\Sigma_0 = \varnothing$$
$$\Sigma_1 = \{\forall x\ R(x) \rightarrow R'(x, x)\}$$
$$\Sigma_2 = \{\forall x\ R(x) \rightarrow \exists y\ R'(x, y)\}$$
$$\Sigma_3 = \{\forall x \forall y\ R(x) \wedge R(y) \rightarrow R'(x, y)\}$$
$$\Sigma_4 = \{\exists x \exists y\ R'(x, y)\}$$
$$\Sigma_5 = \Sigma_1 \cup \Sigma_2$$

Σ_0 and Σ_4 seem pretty poor, here, as they fail to explain any facts of J, while there seems to be a definite relation (albeit with some noise) between I and J. Σ_3 explains most of the facts of J, but is far from being valid, since it also explains a large number of incorrect facts such as $R'(a, b)$ or $R'(b, d)$. Σ_1, Σ_2, and Σ_5 are more interesting. Σ_1 explains three facts of J, but also incorrectly predicts $R'(c, c)$. Σ_2 fails to explain any facts of J, but explains most of them at least partially, in the sense that they are explained by a fact with an existentially quantified variable (a *skolem*); in addition, it is valid with respect to (I, J). Σ_5 combines the interesting features of Σ_1 and Σ_2, with the downside of being larger. None of the schema mappings explain the last fact of J.

As there seems to be some noise in the operation that produced J from I, it is hard to say with certainty which schema mapping is optimal here, in the sense that it reflects most closely the relation between I and J. At any rate, however, Σ_1, Σ_2, and Σ_5 seem far better candidates than the other ones.

To define in a formal way our notion of optimality of a schema mapping, the basic idea is to get the simultaneous optimal for all three factors of interest (validity, explanation of the target instance, conciseness) by minimizing the size of: the original formula, plus all the local corrections that have to be done for the formula to be valid and to fully explain the target instance. This is close in spirit to the notion of Kolmogorov complexity and Kolmogorov optimal [Li and Vitányi 1997] (though we do not consider a Turing-complete language for expressing either the formula or the corrections, but much more limited languages).

Definition 3.4. Given a schema mapping of tgds $\Sigma \subset \mathcal{L}_{\text{tgd}}$, we define the set of *repairs* of Σ, denoted $\mathsf{repairs}(\Sigma)$, as a set of finite sets of formulas, such that $\Sigma' \in \mathsf{repairs}(\Sigma)$ if it can be obtained from Σ by a finite sequence of the following operations:

Adding to the left-hand side of a tgd θ of Σ, with θ of the form $\forall \mathbf{x}\ \varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y})$, a conjunction $\tau(\mathbf{x})$ of the form: $\bigwedge_i x_i \alpha_i c_i$ where α_i is either $=$ or $\neq$, x_i is a variable from $\mathbf{x}$ and c_i is a constant.

Adding to the right-hand side of a tgd θ of Σ, with θ as above, a formula $\tau'(\mathbf{x}, \mathbf{y})$ of the form:

$$\bigwedge_i \left(\left(\bigwedge_j x_{ij} = c'_{ij} \right) \rightarrow y_i = c_i \right)$$

where x_{ij} are variables from $\mathbf{x}$, y_i variables from $\mathbf{y}$, and c'_{ij} and c_i constants.

Adding to Σ a ground fact $R(c_1 \ldots c_n)$ where R is a relation of arity n, and $c_1 \ldots c_n$ are constants.

The language of repairs $\mathcal{L}^*$ of a language $\mathcal{L}$ is the language consisting of all formulas which can be obtained from formulas of $\mathcal{L}$ with these operations (along with all ground facts over the target schema).

In a repair of a tgd $\forall \mathbf{x}\ \varphi(\mathbf{x}) \wedge \tau(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}) \wedge \tau'(\mathbf{x}, \mathbf{y})$, the term $\tau(\mathbf{x})$ is responsible for correcting cases when the tgd is not valid, by adding additional constraints on the universal quantifier, whereas $\tau'(\mathbf{x}, \mathbf{y})$ specifies the right-hand side of J, by giving the explicit value of each existentially quantified variable, in terms of the universally quantified variables.

Example 3.5. Consider the databases instances and schema mappings from Example 3.3. Two examples of repairs of $\Sigma_2 = \{\forall x\ R(x) \rightarrow \exists y\ R'(x, y)\}$ are the following:

$$\Sigma'_2 = \{\forall x\ R(x) \wedge x \neq e \rightarrow \exists y\ R'(x, y), \quad R(d, f)\}$$
$$\Sigma''_2 = \{\forall x\ R(x) \rightarrow \exists y\ R'(x, y) \wedge (x = a \rightarrow y = a) \wedge (x = b \rightarrow y = b) \wedge (x = c \rightarrow y = a) \wedge (x = d \rightarrow y = d), \quad R(g, h)\}$$

Σ'_2 is not especially interesting with respect to I and J, since the condition $R(x) \wedge x \neq e$ is never verified in I and $R(d, f)$ is not a fact in J. On the other hand, Σ''_2 has nice properties: it is still valid with respect to (I, J) (this was already the case with Σ_2) and it fully explains J given I (this was not true for Σ_2). Σ''_2 is then somehow a "perfect" repair of Σ_2 that corrects all problems with the original schema mapping. It is however quite long, since all conditions are enumerated one by one. In the following, we will define the quality (or *cost*) of a schema mapping in terms of the size of its minimal "perfect" repair.

An interesting property of repairs is that they are reversible: Because all operations add constants to a language where constants do not exist, it is possible to compute (in linear time) the original schema mapping from a repair. Indeed, constants are only used for repairing formulas; in other words, we consider that the relations that we need to find between the source and target instances are to be expressed with constant-free formulas, in order to abstract them as much as possible. Clearly, this is a simplifying assumption that could be lifted in future works. Note that this extension is not straightforward, however: It is not clear how to distinguish between constants that are rightfully part of the optimal schema mapping description and constants that are just used to correct some noise or missing data.

The notion of size of a schema mapping is easily defined as follows; we could also use a more classical definition in function of the number of symbols of a formula, without much difference in the theory.

Definition 3.6. The *size* of a first-order formula $\varphi \in \mathcal{L}^*$, denoted $\mathsf{size}(\varphi)$ is computed as follows:

The size of φ is the number of occurrences of variables and constants in φ (we stress that each variable and constant is counted as many times as it occurs in φ); occurrences of variables as arguments of quantifiers do not count.

If φ is a ground fact $R(c_1 \cdots c_n)$, then the size of φ is computed as if φ were the formula

$$\exists x_1 \cdots \exists x_n \; R(x_1 \cdots x_n) \wedge x_1 = c_1 \wedge \cdots \wedge x_n = c_n.$$

Therefore, $\mathsf{size}(\varphi) = 3n$.

The size of a schema mapping is the sum of the size of its elements.

The refinement on ground facts is performed so that such facts are not "too cheap": the cost of $R(c_1 \cdots c_n)$ is the same as that of the corresponding repair of $\exists x_1 \cdots \exists x_n \; R(x_1 \cdots x_n)$, as will be illustrated in Example 3.9. This is not a major assumption, however, and it does not impact the results of this paper in a significant way. We are now ready to define the cost of a schema mapping, in terms of the size of its repairs:

Definition 3.7. The *cost* of a schema mapping Σ, with respect to a pair of instances (I, J), is defined by:

$$\mathsf{cost}_{(I,J)}(\Sigma) = \min_{\substack{\Sigma' \in \mathsf{repairs}(\Sigma) \\ (I,J) \models \Sigma' \text{ and } \Sigma' \text{ fully explains } J}} \mathsf{size}(\Sigma').$$

Note that $\mathsf{cost}_{(I,J)}(\Sigma)$ may not be defined if the minimizing set is empty.

A schema mapping $\Sigma \subset \mathcal{L}$ is *optimal* in the language $\mathcal{L}$, with respect to a pair of instances (I, J), if:

$$\mathsf{cost}_{(I,J)}(\Sigma) = \min_{\substack{\Sigma' \subset \mathcal{L} \\ \Sigma' \text{ finite}}} \mathsf{cost}_{(I,J)}(\Sigma').$$

It is indeed possible that $\mathsf{cost}_{(I,J)}(\Sigma)$ is not defined. This is for instance the case for $\mathbf{T} = \{(R', 1)\}$, $\Sigma = \{\exists x \; R'(x)\}$ and $J = \varnothing$. However, this case is easily recognizable; in other cases, we have a linear bound on the cost of a schema mapping:

PROPOSITION 3.8. *There is an algorithm that is polynomial in the size of the target instance to check whether the cost of a schema mapping in $\mathcal{L}_{\text{tgd}}$ is defined with respect to a pair of instances. If it is defined, the cost is bounded by a linear function of the size of the data and the schema mapping itself.*

PROOF. Let Σ be a schema mapping, I and J instances of the source and target schema. Then the cost of Σ is undefined if and only if it contains a tgd θ without a left-hand side (i.e., of the form $\exists \mathbf{y} \; \psi(\mathbf{y})$) and which is not valid in J, which can be tested in polynomial time in the size of J.

To see this, we shall prove that in all other cases, there is a linear bound on $\mathsf{cost}_{(I,J)}(\Sigma)$. Indeed, every tgd with a left-hand side can be canceled by adding an $x = c$ term on the left-hand side, where x is a universally quantified variable and c a constant which does not appear in I. Moreover, all facts of J can be added to the repair of the tgd as ground facts. We then have the following bound on the cost of Σ:

$$\mathsf{cost}_{(I,J)}(\Sigma) \leqslant \mathsf{size}(\Sigma) + 2|\Sigma| + 3r|J|$$

where r is the maximum arity of a target relation. □

This linear bound is interesting, since we can imagine to use local search algorithms to find the tgd with minimum cost, *as soon as we are able to compute in an efficient way the cost of a tgd*. We shall see in Section 5, unfortunately, that even for a very restricted language, computing the cost of a tgd is **NP**-complete.

Example 3.9. Let us go back to the instances and schema mappings of Example 3.3, compute their respective cost, and see which one is optimal.

Consider first $\Sigma_1 = \{\forall x\ R(x) \rightarrow R'(x, x)\}$ and let us find its perfect repair of minimal size. Since Σ_1 incorrectly predicts $R'(c, c)$, we need to add a condition on the left-hand side; "$\wedge\, x \neq c$" is enough, only invalidates this fact, and there is no correction whose size would be smaller. With this addition, Σ_1 becomes valid, but does not fully explain all facts of the target instance. The only way to fix this is to add the ground facts $R'(c, a)$ and $R'(g, h)$ to Σ_1, to obtain a repair

$$\Sigma_1' = \{\forall x\ R(x) \wedge x \neq c \rightarrow R'(x, x), R'(c, a), R'(g, h)\}$$

that is valid and fully explains the target instance. The discussion above shows that Σ_1' is of minimal size. This means $\mathsf{cost}_{(I,J)}(\Sigma_1) = \mathsf{size}(\Sigma_1') = (3{+}2){+}2{\cdot}3{\cdot}2 = 17$.

Consider now $\Sigma_4 = \{\exists x \exists y\ R'(x, y)\}$. Since the formula has no left-hand side, we only need to consider the last two repair operations: correcting existential qualifiers and adding ground facts. Adding one of the fact of J to Σ_4 yields an increase in the size of the formula of $3 \times 2 = 6$. On the other hand, we can obtain one of the fact of J by repairing the existential qualifiers with, say, "$\wedge\, x = a \wedge y = a$", for an increase in size of 4. The cost of Σ_4 is thus the sum of 2 (the original size of the formula), 4 (the size of the repair of the existential qualifiers, to predict one of the facts of J), 4×6 (the size of the remaining facts of J added as ground facts), that is, 30.

We obtain in this way (the computation is similar for the other schema mappings):

$$\begin{aligned}
\mathsf{cost}_{(I,J)}(\Sigma_0) &= 3 \cdot 2|J| = 30\\
\mathsf{cost}_{(I,J)}(\Sigma_1) &= 3 + 2 + 2 \cdot 3 \cdot 2 = 17\\
\mathsf{cost}_{(I,J)}(\Sigma_2) &= 3 + 4 \cdot 4 + 3 \cdot 2 = 25\\
\mathsf{cost}_{(I,J)}(\Sigma_3) &= 4 + 4 + 4 \cdot 3 \cdot 2 = 32\\
\mathsf{cost}_{(I,J)}(\Sigma_4) &= 2 + 4 + 4 \cdot 3 \cdot 2 = 30\\
\mathsf{cost}_{(I,J)}(\Sigma_5) &= 3 + 2 + 3 + 4 + 3 \cdot 2 = 18
\end{aligned}$$

The fact that Σ_4 has the same cost as Σ_0 is no coincidence, this is due to the choice we made for the cost of a ground fact. It appears that Σ_1 is the best of these schema mappings, with Σ_5 almost as good, and Σ_2 a little behind. We can show that Σ_1 is actually the optimal schema mapping, that is, there are no other

schema mappings with lesser cost. To see this, note first that g and h occur in J without occurring in R. This means $R'(g, h)$ is necessarily added as ground fact in the repair of any schema mapping. Therefore, disregarding this ground fact, only repaired schema mappings of size less than $size(\Sigma_1) - 6 = 11$ are of potential interest. We can then enumerate all such repaired schema mappings and show that none is valid and fully explains J.

At least on this simple example, our measure of cost seems reasonable. We will further justify it in Section 4.

The following decision problems arise naturally once given this notion of optimality. Each of them is defined for a given language $\mathcal{L}$ ($\mathcal{L}$ can for instance be $\mathcal{L}_{\text{tgd}}$ or $\mathcal{L}_{\text{acyc}}{}^*$), and we shall investigate their complexity in Section 5.

VALIDITY	Given instances I, J, and $\Sigma \subset \mathcal{L}$, is Σ valid with respect to (I, J)?
EXPLANATION	Given instances I, J, and $\Sigma \subset \mathcal{L}$, does Σ fully explain J with respect to I?
ZERO-REPAIR	Given instances I, J, and $\Sigma \subset \mathcal{L}$, is $\mathsf{cost}_{(I,J)}(\Sigma)$ equal to $\mathsf{size}(\Sigma)$?
COST	Given instances I, J, $\Sigma \subset \mathcal{L}$ and an integer $K \geqslant 0$, is $\mathsf{cost}_{(I,J)}(\Sigma)$ less than or equal to K?
EXISTENCE-COST	Given instances I, J and an integer $K \geqslant 0$, does there exist $\Sigma \subset \mathcal{L}$ such that $\mathsf{cost}_{(I,J)}(\Sigma)$ is less than or equal to K?
OPTIMALITY	Given instances I, J, and $\Sigma \subset \mathcal{L}$, is it true that Σ is optimal with respect to (I, J)?

4. *Justification*

In this section, we justify the definitions of the previous section by observing that, when instances I and J are derived from each other by elementary operators of the relational algebra, the optimal schema mapping, in $\mathcal{L}_{\text{tgd}}$, is the one that "naturally" describes this operator.

Let r, r' be instances of relations. We consider the following elementary operators of the relational algebra:

Projection	$\pi_i(r)$ denotes the projection of r along its ith attribute.
Selection	$\sigma_\varphi(r)$, where φ is a *conjunction* of equalities and negated equalities between an attribute of r and a constant, denotes the selection of r according to φ. Note that we allow neither identities between attributes of r (this is the role of the *join* operation), nor disjunctions (they may be expressed using a combination of selection and union).
Union	$r \cup r'$ is the union of r and r'.
Intersection	$r \cap r'$ is the intersection of r and r'.
Product	$r \times r'$ is the cross product of r and r'.
Join	$r \ 1_\varphi \ r'$ is the join of r and r' according to φ, where φ is an equality between an attribute of r and an attribute of r'; φ is omitted when the context makes it clear.

The relationship between a database instance I and the instance J obtained from I using one of these operators can often be (partially) expressed in a natural way by a tgd or a set of tgds, where I is the source instance and J the target instance (and similarly when the source instance is expressed as the result of applying some operator to the target instance). For instance $\forall x\ R_1(x) \wedge R_2(x) \rightarrow R'(x)$ is naturally associated with the intersection operator. The only case when the relationship between I and J has no natural expression as a tgd is for the reciprocal of the union operator: If $I = R_1^J \cup R_2^J$, the natural formula for describing this relation is $\forall x\ R(x) \rightarrow R_1(x) \vee R_2(x)$, which is not a tgd since we do not allow disjunction. In some cases, as tgds are not powerful enough to express the relationship, or as some information is lost, the correspondence is only partial. For instance, $\forall x\ R(x) \rightarrow R'(x)$ is the most natural tgd for the operation $J = \sigma_\varphi(I)$, but the tgd does not fully describe the selection.

We now state that, using the notion of optimality of a schema mapping with respect to a pair of instances described in the previous section, and with some simple restrictions on the considered instances, the optimal schema mapping for a pair of instances obtained from each other with an operator of the relational algebra is precisely the schema mapping that is naturally associated with the operator. This justifies the choice of this notion of optimality, at least in these elementary contexts. We shall see in Section 6 other choices for the cost function, that might seem more natural at first, but that fail to satisfy the same property. For clarity's sake, we first state this result in an informal way and illustrate it on the example of the join operator, before presenting it formally.

THEOREM 4.1 (INFORMALLY STATED). *For any elementary operator γ of the relational algebra, the tgd naturally associated with this operator (when it exists) is optimal with respect to $(I, \gamma(I))$ (or $(\gamma(J), J)$, depending on the considered case), if some basic assumptions are fulfilled: the instances are not of trivial size and there is no simpler relation between attributes than the one described by the operator.*

Let us see what this means, and prove this result, for the join operator. Suppose $J = R_1^I\ 1\ R_2^I$ (with R_1 and R_2 two binary relation symbols), and let us add the basic assumptions that $\pi_1(J) \neq \pi_2(J)$, $\pi_1(J) \neq \pi_3(J)$, $\pi_2(J) \neq \pi_3(J)$. We have:

$$\mathsf{cost}_{(I,J)}(\{\forall x \forall y \forall z\ R_1(x, y) \wedge R_2(y, z) \rightarrow R'(x, y, z)\}) = 7$$

since this tgd is valid and explains all facts of J. The cost of the empty schema mapping, $9|J|$, is greater since J is not empty. The only remaining relevant schema mappings with lesser size (of 5) have a single relation symbol R_1 or R_2 on the left-hand-side. But this means that they either predict two identical columns in J (this is incorrect, and has to be fixed in a repair of the schema mapping, whose additional cost is at least 2), or use an existential quantifier on the right-hand size, which also has to be repaired.

Now consider the case where $I = {R'_1}^J\ 1\ {R'_2}^J$, and let us suppose all three attributes $\pi_i(I)$ disjoint.

$$\begin{aligned}\mathsf{cost}_{(I,J)}(\{\forall x \forall y \forall z\ R(x, y, z) \rightarrow R'_1(x, y) \wedge R'_2(y, z)\}) \\ = 7 + 6\left|\left\{(x, y) \in {R'_1}^J \middle| \forall z\ (y, z) \notin {R'_2}^J\right\}\right| \\ + 6\left|\left\{(y, z) \in {R'_2}^J \middle| \forall x\ (x, y) \notin {R'_1}^J\right\}\right|.\end{aligned}$$

$\mathsf{cost}_{(I,J)}(\varnothing) = 6|J|$ is greater than that as soon as I is not empty. As we assumed all three attributes of I disjoint, we can eliminate a number of schema mappings that do not produce any correct facts. The only remaining ones only have $R'_1(w_1, w_2)$ or $R'_2(w_2, w_3)$ terms on the right-hand size with those three variables either existentially quantified or appearing, respectively in the first, second or third position of a $R(w_1, w_2, w_3)$ atom on the left-hand side. None of these schema mappings can explain the facts that the schema mapping above does not explain, and existential quantifiers have to be accounted for in repairs.

We can now state the result of Theorem 4.1 more formally:

THEOREM 4.2 (FORMALLY STATED). *The tgds presented in the last column of Table I are optimal with respect to (I, J), when I and J are as described (in order not to clutter the table, universal quantifiers on the front of tgds are not shown).*

PROOF. First, observe that the size of a ground fact of arity n is the same as the size of any maximal repair of the tgd without a left-hand side $\exists x_1 \cdots \exists x_n\ R(x'_1 \cdots x'_n)$. This means that we do not need to consider such tgds.

Projection. Suppose $J = \pi_1(I)$ with $I \neq \varnothing$. Then:

$$\mathsf{cost}_{(I,J)}(\{\forall x \forall y\ R(x, y) \rightarrow R'(x)\}) = 3$$

since it is valid and fully explains all facts of J. We then only need to consider schema mappings of size strictly lesser than 3. The only relevant one is the empty schema mapping and $\mathsf{cost}_{(I,J)}(\varnothing) = 3|J| = 3|I|$, which is greater or equal to 3 as soon as $I \neq \varnothing$.

Consider now the case when $I = \pi_1(J)$ with $\pi_1(J) \cap \pi_2(J) = \varnothing$, $|\pi_1(J)| \geqslant 2$. We have:

$$\begin{aligned}&\mathsf{cost}_{(I,J)}(\{\forall x\ R(x) \rightarrow \exists x\ R'(x, y)\})\\ &\quad = 3 + 4|I| + 3 \cdot 2(|J| - |I|)\\ &\quad = 3 - 2|I| + 6|J|\end{aligned}$$

(this is a valid schema mapping, but it fails to explain all facts of J; $|I|$ facts can be explained by repairing the existential quantifier, all others must be given as ground facts). As we assume that attributes of J have disjoint domains, all schema mappings with a $R'(w_1, w_2)$ where w_2 is not existentially quantified do not explain any facts of J. The only remaining schema mapping of interest is then $\varnothing$, whose cost is $6|J|$, which is greater than $3 - 2|I| + 6|J|$ as soon as $|I| \geqslant 2$.

Selection. If $J = \sigma_\varphi(I)$, with $|\sigma_\varphi(I)| \geqslant \frac{\mathsf{size}(\varphi)+2}{3}$ (in the common case where φ is a single equality or negated equality, $\mathsf{size}(\varphi) = 2$ and this condition amounts to $|J| \geqslant 2$), we have:

$$\begin{aligned}&\mathsf{cost}_{(I,J)}(\{\forall x\ R(x) \rightarrow R'(x)\})\\ &\quad \leqslant \mathsf{size}(\forall x\ R(x) \wedge \varphi(x) \rightarrow R'(x))\\ &\quad = 2 + \mathsf{size}(\varphi).\end{aligned}$$

The only other schema mapping that might have a lesser cost is $\varnothing$ and $\mathsf{cost}_{(I,J)}(\varnothing) = 3|J| = 3|\sigma_\varphi(I)|$.

TABLE I. OPTIMAL TGDS FOR ELEMENTARY OPERATIONS OF THE RELATIONAL ALGEBRA.

	S	T	Condition on I	I and J	Optimal schema mapping
Projection	$(R, 2)$ $(R, 1)$	$(R', 1)$ $(R', 2)$	I non-empty $\pi_1(J) \cap \pi_2(J) = \varnothing$, $\lvert\pi_1(J)\rvert \geqslant 2$	$J = \pi_1(I)$ $I = \pi_1(J)$	$R(x, y) \rightarrow R'(x)$ $R(x) \rightarrow \exists y\ R'(x, y)$
Selection	$(R, 1)$ $(R, 1)$	$(R', 1)$ $(R', 1)$	$\lvert\sigma_\varphi(I)\rvert \geqslant \frac{\mathsf{size}(\varphi)+2}{3}$ $\sigma_\varphi(J)$ non-empty	$J = \sigma_\varphi(I)$ $I = \sigma_\varphi(J)$	$R(x) \rightarrow R'(x)$ $R(x) \rightarrow R'(x)$
Union	$(R_1, 1), (R_2, 2)$	$(R', 1)$	${R_1}^I \subsetneq {R_1}^I \cup {R_2}^I$, ${R_2}^I \subsetneq {R_1}^I \cup {R_2}^I$	$J = R_1^I \cup R_2^I$	$R_1(x) \rightarrow R'(x)$, $R_2(x) \rightarrow R'(x)$
Intersection	$(R_1, 1), (R_2, 1)$ $(R, 1)$	$(R', 1)$ $(R'_1, 1), (R'_2, 1)$	$R_1^I \not\subseteq R_2^I$, $R_2^I \not\subseteq R_1^I$, $R_1^I \cap R_2^I \neq \varnothing$ ${R'_1}^J \cap {R'_2}^J$ non-empty	$J = R_1^I \cap R_2^I$ $I = {R'_1}^J \cap {R'_2}^J$	$R_1(x) \wedge R_2(x) \rightarrow R'(x)$ $R(x) \rightarrow R'_1(x) \wedge R'_2(x)$
Product	$(R_1, 1), (R_2, 1)$ $(R, 2)$	$(R', 2)$ $(R'_1, 1), (R'_2, 1)$	R_1^I and R_2^I non-empty ${R'_1}^J$ and ${R'_2}^J$ non-empty	$J = R_1^I \times R_2^I$ $I = {R'_1}^J \times {R'_2}^J$	$R_1(x) \wedge R_2(y) \rightarrow R'(x, y)$ $R(x, y) \rightarrow R'_1(x) \wedge R'_2(y)$
Join	$(R_1, 2), (R_2, 2)$ $(R, 3)$	$(R', 3)$ $(R'_1, 2), (R'_2, 2)$	$\pi_i(R_1^I \ 1 \ R_2^I) \neq \pi_j(R_1^I \ 1 \ R_2^I)\ (i \neq j)$ $\pi_i({R'_1}^J \ 1 \ {R'_2}^J) \cap \pi_j({R'_1}^J \ 1 \ {R'_2}^J) = \varnothing\ (i \neq j)$ ${R'_1}^J \ 1 \ {R'_2}^J$ non-empty	$J = R_1^I \ 1 \ R_2^I$ $I = {R'_1}^J \ 1 \ {R'_2}^J$	$R_1(x, y) \wedge R_2(y, z) \rightarrow R'(x, y, z)$ $R(x, y, z) \rightarrow R'_1(x, y) \wedge R'_2(y, z)$

Now, suppose $I = \sigma_\varphi(J)$ with $\sigma_\varphi(J) \neq \varnothing$. Observe that

$$\mathsf{cost}_{(I,J)}(\{\forall x\ R(x) \rightarrow R'(x)\}) = 2 + 3(|J| - |I|)$$

is lesser than $\mathsf{cost}_{(I,J)}(\varnothing) = 3|J|$ as soon as $|I| \neq \varnothing$.

Union. If $J = R_1^I \cup R_2^I$ with both relations strictly included in their union,

$$\mathsf{cost}_{(I,J)}(\{\forall x\ R_1(x) \rightarrow R'(x), \forall x\ R_2(x) \rightarrow R'(x)\}) = 4$$

while the cost of each of these tgds alone is greater than 5. We also have:

$$\begin{aligned}&\mathsf{cost}_{(I,J)}(\{\forall x\ R_1(x) \wedge R_2(x) \rightarrow R'(x)\})\\ &\quad = 3 + 3\big(\big|R_1^I \cup R_2^I\big| - \big|R_1^I \cap R_2^I\big|\big) \geqslant 9.\end{aligned}$$

Finally, the cost of the empty schema mapping is:

$$3\big|R_1^I \cup R_2^I\big| \geqslant 6.$$

Intersection. Suppose $J = R_1^I \cap R_2^I$ with neither of these relations containing the other one;

$$\mathsf{cost}_{(I,J)}(\{\forall x\ R_1(x) \wedge R_2(x) \rightarrow R'(x)\}) = 3.$$

Neither $\{\forall x\ R_1(x) \rightarrow R'(x)\}$, nor $\{\forall x\ R_2(x) \rightarrow R'(x)\}$, nor the empty schema mapping, have a lesser cost as soon as both R_1^I and R_2^I contain facts not in the other one, and the intersection is not empty.

Consider now the case where $I = {R'_1}^I \cap {R'_2}^I$. Then:

$$\begin{aligned}&\mathsf{cost}_{(I,J)}(\{\forall x\ R(x) \rightarrow R'_1(x) \wedge R'_2(x)\})\\ &\quad = 3 + 3\big(\big|{R'_1}^J\big| + \big|{R'_2}^J\big| - 2|I|\big)\end{aligned}$$

while

$$\begin{aligned}&\mathsf{cost}_{(I,J)}(\{\forall x\ R(x) \rightarrow R'_1(x)\})\\ &\quad = 2 + 3\big(\big|{R'_1}^J\big| + \big|{R'_2}^J\big| - |I|\big)\\ &\mathsf{cost}_{(I,J)}(\{\forall x\ R(x) \rightarrow R'_2(x)\})\\ &\quad = 2 + 3\big(\big|{R'_1}^J\big| + \big|{R'_2}^J\big| - |I|\big)\\ &\mathsf{cost}_{(I,J)}(\varnothing) = 3\big(\big|{R'_1}^J\big| + \big|{R'_2}^J\big|\big).\end{aligned}$$

The first schema mapping has a lower cost than the other ones as soon as $I \neq \varnothing$.

Product. In both cases, the cost of the schema mapping from Table I is 4 (it is valid and explains all facts of J) and, unless one of the instance is empty, no other schema mapping of lesser size is valid and explains all facts of J.

Join. This has been shown above. □

These results could also be extended to the cases where we have relations of greater arity, but we would then require strong constraints, as the one we imposed

for reciprocal join (that all attributes are disjoint), so as not to have any "hidden" relation between the different attributes. A weaker assumption that could be made is to use a notion of *Kolmogorov randomness* [Li and Vitányi 1997]: A database instance selected at random cannot have a description of length lower than its size, thanks to a simple counting argument. We can use such random instances to get a contradiction when we obtain a schema mapping that uses hidden relations between attributes of relations in the instance to have a lower cost than the natural schema mapping. Using such random instances is not straightforward to formalize: in the case of a join, for instance, we probably do not want to consider instances for which the result of the join is empty, which almost always happens if we consider truly random instances.

One might also want to extend this result to arbitrary relational expressions. However, this is not direct. The first issue is that arbitrary relational algebra expressions might not be minimal, or that several different algebra expressions of the same size might be equivalent (think of projections pushed inside joins, for instance). Depending on the way these expressions are transformed into tgds, this nonminimality may be reflected on the schema mapping itself, and we definitely do not want to derive a nonminimal schema mapping. Secondly, composing several basic operations of the relational algebra will yield in most cases formulas that are not expressible as tgds (possibly requiring second-order formulas [Fagin et al. 2004]). Finally, we need to be sure that there are no hidden dependencies inside the data that will make another schema mapping optimal, as discussed above. The conditions for that were relatively simple in the case of the basic operations of the relational algebra but would be much more involved in the case of arbitrary expressions.

5. *Complexity Study*

We now proceed to a study of the computational complexity of the different problems identified in Section 3, for the different subsets of $\mathcal{L}_{\text{tgd}}$ that we presented in Section 2. We focus here on *combined complexity* (when K and Σ are part of the input to the problem in addition to I and J), since we are precisely reasoning about the schema mappings themselves, but we also present at the end of the section *data complexity* results. We first describe general relationships between the different problems, before giving a full characterization of their complexity for $\mathcal{L}_{\text{facyc}}$, the language of full acyclic tgds. We also consider the more general cases of $\mathcal{L}_{\text{tgd}}$, $\mathcal{L}_{\text{full}}$, and $\mathcal{L}_{\text{acyc}}$. EXISTENCE-COST and OPTIMALITY will be discussed separately.

We briefly go over the standard notations used for the complexity classes that are of interest here [Papadimitriou 1994]. All complexity classes are displayed in **bold** font. **PTIME** is the class of problems solvable in polynomial time by a deterministic Turing machine. A problem is in **NP** (respectively, **coNP**) if it is solvable (respectively, if its complement is solvable) in polynomial time by a nondeterministic Turing machine. **DP** is defined as the class of all languages L that can be written as $L = L_1 \cap L_2$, where $L_1 \in \mathbf{NP}$ and $L_2 \in \mathbf{coNP}$. The *polynomial hierarchy* is defined inductively as follows: $\mathbf{\Sigma_0^P} = \mathbf{\Pi_0^P} = \mathbf{PTIME}$, and for all $i \geqslant 0$, $\mathbf{\Sigma_{i+1}^P} = \mathbf{NP}^{\mathbf{\Sigma_i^P}}$ and $\mathbf{\Pi_{i+1}^P} = \mathbf{coNP}^{\mathbf{\Sigma_i^P}}$ where A^B is the class of problems solvable by a Turing machine in class A with access to an oracle for a B-complete problem. Observe that $\mathbf{\Sigma_1^P} = \mathbf{NP}$ and $\mathbf{\Pi_1^P} = \mathbf{coNP}$.

5.1. GENERAL COMPLEXITY RESULTS. As the complexity of the different decision problems depends on the particular language considered, we add to the problem name a subscript identifying the considered language (say, OPTIMALITY$_{\text{tgd}}$ for the OPTIMALITY problem in $\mathcal{L}_{\text{tgd}}$). We use an asterisk to indicate the problem concerns schema mappings that include repairs (for instance, ZERO-REPAIR$_{\text{full}^*}$ for the ZERO-REPAIR problem in $\mathcal{L}_{\text{full}}{}^*$).

We have the following elementary relationships between these problems, that can be used to derive complexity results for one problem from complexity results for another one.

PROPOSITION 5.1. *For any language $\mathcal{L} \subseteq \mathcal{L}_{\text{tgd}}{}^*$:*

(1) ZERO-REPAIR $=$ VALIDITY $\cap$ EXPLANATION.

(2) *There is a polynomial-time reduction of* VALIDITY *to* ZERO-REPAIR.

(3) *There is a polynomial-time reduction of* ZERO-REPAIR *to* COST.

(4) *Given an algorithm $\mathcal{A}_{\text{ZERO-REPAIR}}$ for* ZERO-REPAIR ***in*** *$\mathcal{L}^*$ and a polynomial-time algorithm for determining if a formula is in $\mathcal{L}$, there are nondeterministic algorithms for* COST *and* EXISTENCE-COST ***in*** *$\mathcal{L}$ that run by using once the algorithm $\mathcal{A}_{\text{ZERO-REPAIR}}$, with an additional polynomial-time cost.*

(5) *Given an algorithm $\mathcal{A}_{\text{COST}}$ for* COST *and a polynomial-time algorithm for determining if a formula is in $\mathcal{L}$, there is a nondeterministic algorithm for the complement of* OPTIMALITY *that runs by using a logarithmic number of times the algorithm $\mathcal{A}_{\text{COST}}$, with an additional polynomial-time cost.*

(6) *If $\mathcal{L} \subseteq \mathcal{L}'$, for any problem among* VALIDITY, EXPLANATION, ZERO-REPAIR *and* COST, *there is a constant-time reduction from the problem in $\mathcal{L}$ to the problem in $\mathcal{L}'$. This is in particular true for reductions from the problem in $\mathcal{L}$ to the problem in $\mathcal{L}^*$.*

PROOF

(1) By definition, $\mathsf{cost}_{(I,J)}(\Sigma) \geqslant \mathsf{size}(\Sigma)$. Because the size of a repaired formula is always greater than the original formula, the only case when the equality occurs is when the original formula is valid and fully explains J.

(2) Let (I, J, Σ) be an instance of VALIDITY. Let Σ' be the union of Σ and of all ground facts of J. Obviously, Σ' fully explains J with respect to I. That means that $\mathsf{cost}_{(I,J)}(\Sigma') = \mathsf{size}(\Sigma')$ if and only if Σ' is valid with respect to (I, J). As the ground facts of Σ' do not change its validity, $\mathsf{cost}_{(I,J)}(\Sigma') = \mathsf{size}(\Sigma')$ if and only if Σ is valid with respect to (I, J).

(3) Just take $K = \mathsf{size}(\Sigma)$, which is computable in linear time.

(4) Consider the nondeterministic algorithm for COST shown as Algorithm 1.

Algorithm 1. COST (NON-DETERMINISTIC) GIVEN $\mathcal{A}_{\text{ZERO-REPAIR}}$ FOR ZERO-REPAIR

Input: Instances I, J, schema mapping Σ, $K \geqslant 0$.
Output: Answer to COST.

(a) Let K' be the minimum between K and the upper bound of Proposition 3.8.

(b) Guess a set of formulas Σ' of total size less than or equal to K'.

(c) Check that Σ' is a repair of Σ. Otherwise, give up this guess.

(d) Apply $\mathcal{A}_{\text{ZERO-REPAIR}}$ on Σ'; if the result is `true`, return `true`.

Algorithm 2. COMPLEMENT OF OPTIMALITY (NON-DETERMINISTIC) GIVEN $\mathcal{A}_{\text{COST}}$ FOR COST

Input: Instances I, J, schema mapping Σ.
Output: Answer to the complement of OPTIMALITY.

(a) Use $\mathcal{A}_{\text{COST}}$ a logarithmic number of times to compute $K = \mathsf{cost}_{(I,J)}(\Sigma)$ (we make use on the linear bound of Proposition 3.8).

(b) Guess a set of formulas Σ' of total size less than K.

(c) Check that Σ' is in $\mathcal{L}$. Otherwise, give up this guess.

(d) Apply $\mathcal{A}_{\text{COST}}$ on $(\Sigma', K-1)$; if the result is `true`, return `true`.

The algorithm for EXISTENCE-COST is very similar, just replace the bound on K with the cost of the empty schema mapping, and step (c) by a check that Σ' is in $\mathcal{L}^*$ (this can be done in polynomial time by hypothesis). Note that the bound of $\mathsf{cost}_{(I,J)}(\varnothing)$ on the guess is critical, since otherwise the guess would be of size K, and thus exponential in the length of the representation of K.

(5) Algorithm 2 is a nondeterministic algorithm for the complement of OPTIMALITY.

(6) Directly results from the fact that a formula of $\mathcal{L}$ is also a formula of $\mathcal{L}'$. Note that this property does not necessarily hold for EXISTENCE-COST and OPTIMALITY, since both of them depend on the *existence* of a formula in the underlying language. □

Note that for each language considered here, there is a linear-time algorithm for determining if a formula is in this language; this is obvious for all except $\mathcal{L}_{\text{acyc}}$ and $\mathcal{L}_{\text{facyc}}$, and an algorithm from Tarjan and Yannakakis [1984] gives a linear-time algorithm for the acyclicity of hypergraphs.

In the next sections, we shall investigate in detail the complexity of the different problems in each of the identified subsets of $\mathcal{L}_{\text{tgd}}$, starting from $\mathcal{L}_{\text{tgd}}$ itself. A summary of all combined complexity results proved in the following, along with their direct consequences, is shown in Table II. The lower part of Table II indicates the references to the propositions used to derive the complexity results shown in the upper part; in the case where there are two references, the first one is for the upper bound, the second one for the lower bound.

5.2. COMBINED COMPLEXITY FOR FULL ACYCLIC TGDS. Let us first investigate the combined complexity of VALIDITY and EXPLANATION in $\mathcal{L}_{\text{facyc}}$. We shall need additional notions on *acyclic joins* from Beeri et al. [1981] and Yannakakis [1981]. Note first that an acyclic full tgd $\forall \mathbf{x}\ \varphi(\mathbf{x}) \rightarrow \psi(\mathbf{x})$ that describes the relation between a pair of instances (I, J) can be seen, in the relational algebra, as a project-join expression over the source instance, $\pi_\psi(1_\varphi(I))$, φ expressing the join (which is, by hypothesis, acyclic) and ψ expressing the projection. Adding repaired formulas, of the form $\forall \mathbf{x}\ (\varphi(\mathbf{x}) \wedge \tau(\mathbf{x})) \rightarrow \psi(\mathbf{x})$, means adding an additional selection: $\pi_\psi(\sigma_\tau(1_\varphi(I)))$.

A *full reducer* of a join expression is a program which removes some tuples to the relations to be joined (by performing semi-joins) so that each relation can then be retrieved as a projection of the full join. Such a full reducer always exists in acyclic databases and can be obtained in polynomial time [Bernstein and Chiu 1981]. The full reducer itself runs in polynomial time. Finally, note that a join tree of an acyclic join can be obtained in linear time [Tarjan and Yannakakis 1984].

TABLE II. COMBINED COMPLEXITY RESULTS AND CORRESPONDING PROOFS.

Combined complexity bounds

	$\mathcal{L}_{\text{tgd}}{}^{*}$, $\mathcal{L}_{\text{tgd}}$	$\mathcal{L}_{\text{full}}{}^{*}$, $\mathcal{L}_{\text{full}}$	$\mathcal{L}_{\text{acyc}^*}$	$\mathcal{L}_{\text{acyc}}$	$\mathcal{L}_{\text{facyc}}{}^{*}$, $\mathcal{L}_{\text{facyc}}$
VALIDITY	$\mathbf{\Pi}_2^P$-complete	**coNP**-complete	**coNP**-complete	**coNP**-complete	**PTIME**
EXPLANATION	**NP**-complete	**NP**-complete	**NP**-complete	**PTIME**	**PTIME**
ZERO-REPAIR	$\mathbf{\Pi}_2^P$-complete	**DP**-complete	**DP**-complete	**coNP**-complete	**PTIME**
COST	$\mathbf{\Sigma}_2^P$, $\mathbf{\Pi}_2^P$-hard	$\mathbf{\Sigma}_2^P$, **DP**-hard	$\mathbf{\Sigma}_2^P$, **DP**-hard	$\mathbf{\Sigma}_2^P$, (co)**NP**-hard	**NP**-complete
EXISTENCE-COST	$\mathbf{\Sigma}_3^P$, **NP**-hard	$\mathbf{\Sigma}_2^P$, **NP**-hard	$\mathbf{\Sigma}_2^P$, **NP**-hard	$\mathbf{\Sigma}_2^P$, **NP**-hard	**NP**-complete
OPTIMALITY	$\mathbf{\Pi}_4^P$, **DP**-hard	$\mathbf{\Pi}_3^P$, **DP**-hard	$\mathbf{\Pi}_3^P$, **DP**-hard	$\mathbf{\Pi}_3^P$, **DP**-hard	$\mathbf{\Pi}_2^P$, **DP**-hard

Proofs

	$\mathcal{L}_{\text{tgd}}{}^{*}$, $\mathcal{L}_{\text{tgd}}$	$\mathcal{L}_{\text{full}}{}^{*}$, $\mathcal{L}_{\text{full}}$	$\mathcal{L}_{\text{acyc}^*}$	$\mathcal{L}_{\text{acyc}}$	$\mathcal{L}_{\text{facyc}}{}^{*}$, $\mathcal{L}_{\text{facyc}}$
VALIDITY	**5.5**(1)	**5.6**(1)	**5.7**(1)	**5.7**(1)	**5.2**
EXPLANATION	**5.5**(2), **5.1**(6)	**5.1**(6), **5.6**(2)	**5.1**(6), **5.7**(2)	**5.7**(4)	**5.2**
ZERO-REPAIR	**5.1**(1), **5.1**(2)	**5.1**(1), **5.6**(3)	**5.1**(1), **5.7**(3)	**5.1**(1)	**5.1**(1)
COST	**5.1**(4), **5.1**(3)	**5.1**(4), **5.1**(3)	**5.1**(4), **5.1**(3)	**5.1**(4), **5.1**(3)+**5.1**(6)	**5.1**(4), **5.4**
EXISTENCE-COST	**5.1**(4), **5.8**	**5.1**(4), **5.8**	**5.1**(4), **5.8**	**5.1**(4), **5.8**	**5.1**(4), **5.8**
OPTIMALITY	**5.1**(5), **5.8**	**5.1**(5), **5.8**	**5.1**(5), **5.8**	**5.1**(5), **5.8**	**5.1**(5), **5.8**

"**5.5**(2), **5.1**(6)" means the complexity upper bound can be obtained by item (2) of Proposition 5.5, while the lower bound is from (6) of Proposition 5.1.

Algorithm 3. RESULT TO A PROJECT-JOIN EXPRESSION ON AN ACYCLIC DATABASE (AFTER YANNAKAKIS [1981])

Input: An acyclic join expression φ, a project expression ψ, an instance I.
Output: $\pi_\psi(1_\varphi(I))$.

(a) Compute a full reducer of the relation instances, and apply it.
(b) Compute a join tree T of the acyclic expression. Each node of the tree initially contains the corresponding reduced relation instance.
(c) For each subtree of T with root r, compute recursively for each child r' of r the join of r with r', and project to the union of the variables appearing in ψ and the common variables of r and r'. Remove r' and replace node r with this result.

Yannakakis [1981] proposes then Algorithm 3 for computing the result to a project-join expression on an acyclic database, that we reuse with slight modifications in our next proposition.

An important property of this algorithm is that, at all time, the size of the relation stored in node r of T is bounded by the original (reduced) size of r times the size of the final output. This means in particular that this algorithm computes in polynomial time the result to the project-join expression. Actually, the same algorithm can be applied when repaired formulas are considered, since the only selection performed is a conjunction of constraints (equality and negated equality) on a given variable: These selections can be pushed inside the join.

PROPOSITION 5.2. $\text{VALIDITY}_{\text{facyc}^*}$ *and* $\text{EXPLANATION}_{\text{facyc}^*}$ *are in* ***PTIME***.

PROOF. We first consider VALIDITY and then EXPLANATION.

(1) First check that the ground facts of Σ are valid. Then, we have to apply Algorithm 3 on each φ of Σ which is not a ground fact to check whether its output is included in J. This, however, may not lead to a polynomial-time algorithm, since Algorithm 3 is polynomial in the size of the join expression, the input, *and the output*. The solution is to take care, at each join step, that the output remains in the bound given in the discussion of Algorithm 3. Assume that at some point during a run of the algorithm, the size of a relation stored at node r goes over the bound of the original size of r multiplied by the size of J; this means that the final output of the algorithm is larger than J, that is, it is not included in J and $(I, J) \not\models \varphi$. Otherwise, the computation is polynomial in the size of φ, I, and J, we can let the algorithm terminate and check then if the output is included in J.

(2) For each fact f of J, proceed as follows. If f appears as a ground fact in Σ, it is fully explained in Σ. Otherwise, for each formula of Σ that is not a ground fact, we apply a variant of the algorithm presented above to decide whether f is in the output of the original algorithm (once again, by pushing selections inside joins), as described in Corollary 4.1 of Yannakakis [1981]. □

ZERO-REPAIR is then tractable in $\mathcal{L}_{\text{facyc}}$. One might hope that this tractability extends to COST. Unfortunately, we now show the **NP**-hardness of $\text{COST}_{\text{facyc}}$, even for a very simple schema mapping. For this purpose, we shall first need a quite general result on the minimal size of a vertex cover in a r-partite r-uniform hypergraph (for $r \geqslant 3$).

A hypergraph is r-partite if the set of vertices can be decomposed into an r-partition, such that no two vertices of the same partitioning subset are in a same

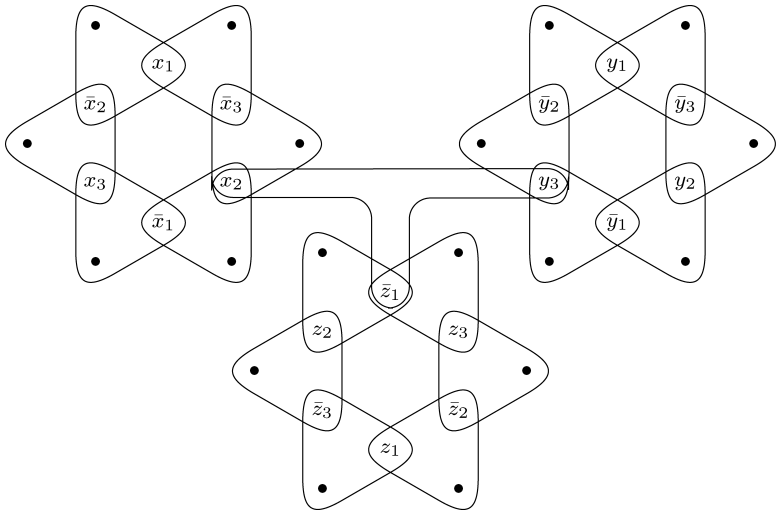

FIG. 1. Example tripartite hypergraph corresponding to the 3SAT instance $\neg z \vee x \vee y$.

hyperedge. It is r-uniform if all hyperedges have a cardinality of r. A vertex cover of a hypergraph is a subset X of the set of vertices, such that for every hyperedge e, at least one of the elements of e is in X. In regular graphs, VERTEX-COVER (determining whether there is a vertex cover of size $\leqslant K$) is one of the most well-known and useful **NP**-complete problems [Garey and Johnson 1979]. This obviously implies that VERTEX-COVER is **NP**-hard in general r-uniform hypergraphs. Note that a 2-partite 2-uniform hypergraph is just a bipartite graph, and VERTEX-COVER in bipartite graphs is **PTIME**, thanks to Kőnig's theorem [Diestel 2005; Kőnig 1936] which states that the maximal number of matchings in a bipartite graph is the minimum size of a vertex cover.

LEMMA 5.3. *The problem of, given an r-partite r-uniform hypergraph $\mathcal{H}$ and a constant K, determining whether there exists a vertex cover in $\mathcal{H}$ of size less than or equal to K is **NP**-complete for $r \geqslant 3$.*

PROOF. This problem is clearly in **NP**: Just guess a set of vertices of size less than or equal to K and check in polynomial time whether it is a vertex cover. For the hardness part, we prove the case $r = 3$; there is an obvious reduction from this case to the same problem for other values of r. We use a reduction from 3SAT.

Note that this result appears in Ilie et al. [2002], but the reduction presented there is not exhaustive (in particular, nothing is said about interdependencies between clauses, or the fact that the hypergraph is tripartite) and it is not clear whether the proof was indeed led to completion. We use here a proof inspired by the proof that 3-DIMENSIONAL-MATCHING is **NP**-hard in Garey and Johnson [1979].

Let $\varphi = \bigwedge_{i=1}^{n} c_i$ be an instance of 3SAT, where the c_i's are 3-clauses over some set $\mathbf{x}$ of variables. We build a tripartite 3-uniform hypergraph $\mathcal{H} = (V, E)$ (with vertex partition $V = V_1 \cup V_2 \cup V_3$) in the following way (see Figure 1 for an illustration when $\varphi = \neg z \vee x \vee y$). For each variable $x \in \mathbf{x}$, we add 12 nodes and 6 hyperedges to $\mathcal{H}$. 6 out of the 12 nodes are anonymous nodes that only appear in one hyperedge; they are denoted by $\bullet$. The other nodes are denoted $x_1, x_2, x_3, \bar{x}_1, \bar{x}_2, \bar{x}_3$. Intuitively, all x_i's are in a minimum covering if and only if a valuation satisfying

TABLE III. LOCAL EDGES USED IN THE PROOF OF LEMMA 5.3.

V_1	V_2	V_3
x_1	•	$\bar{x}_3$
•	x_2	$\bar{x}_3$
$\bar{x}_1$	x_2	•
$\bar{x}_1$	•	x_3
•	$\bar{x}_2$	x_3
x_1	$\bar{x}_2$	•

φ maps x_i to true (similarly with the $\bar{x}_i$'s and false). For each i, x_i and $\bar{x}_i$ belong to V_i. The so-called *local* hyperedges are shown in Table III. Then, for each clause c_i, we add a single *global* hyperedge which contains the vertices corresponding to the variables appearing in c_i, while taking into account their position in the clause and whether they are negated. For instance, if $c_i = \neg z \vee x \vee y$, we add a hyperedge $(\bar{z}_1, x_2, y_3)$. This ensures that the hypergraph remains tripartite.

This reduction is polynomial. Let m be the cardinality of $\mathbf{x}$. We now show that φ is satisfiable if and only if there is a vertex cover in $\mathcal{H}$ of size less than or equal to $3m$ (or, equivalently, if there is a minimum vertex cover of size less than or equal to $3m$).

Suppose first that φ is satisfiable, and let ν be a valuation of $\mathbf{x}$ which satisfies φ. Let us consider the following set S of vertices of $\mathcal{H}$: For each $x \in \mathbf{x}$, we add to S, x_1, x_2 and x_3 if $\nu(x)$ is true, $\bar{x}_1$, $\bar{x}_2$ and $\bar{x}_3$ otherwise. S is of cardinality $3m$. Observe that S covers all local hyperedges and, since ν satisfies φ, all global hyperedges.

Suppose now that there is a minimum vertex cover S of size less than or equal to $3m$. Since anonymous vertices only appear in a single hyperedge, we can always assume that S does not contain any anonymous vertex (they can always be replaced by another vertex of the hyperedge). Let S_i be, for each $1 \leqslant i \leqslant m$, the subset of S containing only the vertices corresponding to the ith variable of $\mathbf{x}$. It is easy to see that $|S_i| \geqslant 3$ for all i, for all local hyperedges to be covered, which means that $|S_i| = 3$ since $|\bigcup S_i| \leqslant 3m$. S_i forms a vertex cover of the local subhypergraph corresponding to the ith variable of $\mathbf{x}$ (let us call it x) and must cover the hyperedges of this sub-hypergraph. But there are only two vertex covers of this subhypergraph of cardinality 3: Either S_i contains all x_k's, or it contains all $\bar{x}_k$'s. We consider the valuation ν of the variables in $\mathbf{x}$ which maps x to true in the first case, to false in the second. Then, since S is a vertex cover of $\mathcal{H}$, ν satisfies all the clauses of φ. □

We now use this lemma to prove the **NP**-hardness of $\textsc{Cost}_{\text{facyc}}$.

PROPOSITION 5.4. $\textsc{Cost}_{\text{facyc}}$ *is* ***NP****-hard.*

PROOF. We consider first the case where we only allow negated equalities $x \neq c$, and no equalities $x = c$, on the left-hand side of repairs of tgds, with x a universally quantified variable, as the proof is clearer.

We reduce the vertex cover problem in tripartite 3-uniform hypergraphs to $\textsc{Cost}_{\text{facyc}}$. Let $\mathcal{H}$ be a tripartite 3-uniform hypergraph. We consider the following instance of $\textsc{Cost}_{\text{facyc}}$:

$\mathbf{S} = \{(R, 3)\}$ and R^I is the representation of $\mathcal{H}$ as a three-column table, where each row corresponds to an edge, and each column to one of the sets of the tripartition of $\mathcal{H}$;

$\mathbf{T} = \{(R', 1)\}$ and $J = \varnothing$;
$\Sigma = \{\forall x_1 \forall x_2 \forall x_3\ R(x_1, x_2, x_3) \rightarrow R'(x_1)\}$ (this is obviously an acyclic tgd).

As $J = \varnothing$, any schema mapping fully explains J. This also means that the only repairs of Σ to be considered are the ones that add a "$x_i \neq c_i$" term to the left-hand side of the single element of Σ. A repair of Σ has to "cancel" somehow with these additions each tuple of R^I. In other words, the cost of Σ is $\mathsf{size}(\Sigma) + 2r$, where r is the minimal number of conjuncts in a formula of the form $\bigwedge x_i \neq c_i$, such that this formula is false for all tuples of R^I. Such a formula expresses a vertex cover in $\mathcal{H}$, and $\mathcal{H}$ has a vertex cover of size less than or equal to K if and only if $\mathsf{cost}_{(I,J)}(\Sigma) \leqslant \mathsf{size}(\Sigma) + 2K$, which concludes the proof in this case.

In the case where we allow arbitrary repairs, including $x = c$ terms on the left-hand side of a tgd, the same proof does not work, since it suffices to choose an arbitrary constant c, which does not appear in I for the term $x_1 = c$ to cancel every tuple of R^I. To fix this, we need to make prohibitive the addition of such a term to Σ in the following way: Let c', and, for $1 \leqslant i \leqslant 3$, $1 \leqslant j \leqslant K$, $c_{i,j}$ be $3K + 1$ fresh constants. We can always assume that K is linear in the size of the input of $\textsc{Cost}_{\text{facyc}}$ (otherwise, just replace K with the upper bound of Proposition 3.8). We consider the following slightly modified instance of $\textsc{Cost}_{\text{facyc}}$:

$\mathbf{S} = \{(R, 3)\}$ and $R^I = A \cup B$, where A is the representation of $\mathcal{H}$ as a three-column table, where each row corresponds to an edge, and each column to one of the set of the tripartition of $\mathcal{H}$, and B is the set:

$$\{R(c_{1j}, c', c') \mid 1 \leqslant j \leqslant K\} \cup$$
$$\{R(c', c_{2j}, c') \mid 1 \leqslant j \leqslant K\} \cup$$
$$\{R(c', c', c_{3j}) \mid 1 \leqslant j \leqslant K\};$$

$\mathbf{T} = \{(R', 3)\}$ and J is the same as B if R is replaced by R';
$\Sigma = \{\forall x_1 \forall x_2 \forall x_3\ R(x_1, x_2, x_3) \rightarrow R'(x_1, x_2, x_3)\}$ (this is obviously an acyclic tgd).

This reduction is polynomial if K is linear in the size of the input, as assumed. Σ fully explains J and the repairs considered in the previous case do not change this.

Let now $x_i = c$ be a term with $1 \leqslant i \leqslant 3$ and c some constant. Whatever the choice of c, the addition of this term to the tgd of Σ cancels at least K tuples of B, and hence, fails to explain at least K tuples of J (the best case is when $c = c'$). Just observe that, as the cost of a ground fact, which is the only way to repair unexplained tuples, is 9, the size of any repair of Σ with such an $x_i = c$ term on the left-hand side is greater than or equal to $\mathsf{size}(\Sigma) + 9K$. We keep the fact that $\mathsf{cost}_{(I,J)}(\Sigma) \leqslant \mathsf{size}(\Sigma) + 2K$ if and only if $\mathcal{H}$ has a vertex cover of size less than or equal to K. □

It is an open issue whether COST is in **PTIME** for the very restricted case when the schema mapping consists of a single full tgd with a single binary relation symbol appearing once in the left-hand side.

5.3. COMBINED COMPLEXITY FOR TGDS. We now look at the complexity of the same problems in the more general case of $\mathcal{L}_{\text{tgd}}$.

PROPOSITION 5.5

(1) VALIDITY$_{\text{tgd}}$ *and* VALIDITY$_{\text{tgd}^*}$ *are* $\mathbf{\Pi_2^P}$*-complete.*
(2) EXPLANATION$_{\text{tgd}^*}$ *is in* ***NP****.*

PROOF

(1) VALIDITY$_{\text{tgd}^*}$ is clearly in $\mathbf{\Pi_2^P}$. First check the validity of the ground facts. For the other formulas of Σ, guess a valuation of the variables of the left-hand side; if the left-hand side is false (can be decided in polynomial time), give up the guess. Otherwise, use the **NP** oracle to decide whether the right-hand side holds; if it does not, return `false`.

For the hardness part, we use a reduction of $\forall\exists$3SAT: the satisfiability of the formula $\forall\mathbf{x}\exists\mathbf{y}\ \varphi(\mathbf{x}, \mathbf{y})$, where φ is a propositional formula in 3-CNF over $\mathbf{x} \cup \mathbf{y}$. This problem is $\mathbf{\Pi_2^P}$-complete [Wrathall 1976; Schaefer and Umans 2002]. Let $\forall\mathbf{x}\exists\mathbf{y}\ \bigwedge_{i=1}^{n} c_i(z_{i1}, z_{i2}, z_{i3})$ be an instance of $\forall\exists$3SAT, where each c_i is a 3-clause, and each z_{ij} is one of the variables of $\mathbf{x}\cup\mathbf{y}$. We consider then the following instance of VALIDITY$_{\text{tgd}}$:

$\mathbf{S} = \{(B, 1)\}$ and $I = \{B(0), B(1)\}$;

$\mathbf{T} = \{(R_1, 3) \ldots (R_8, 3)\}$ and J is such that the R_i^J's are the 8 distinct subsets of $\{0, 1\}^3$ of cardinality 7 (this corresponds to the 8 possible truth tables of a 3-clause);

For each $1 \leqslant i \leqslant n$, let $1 \leqslant k_i \leqslant 8$ be the unique integer such that $c_i(z_{i1}, z_{i2}, z_{i3})$ is true if and only if $(z_{i1}, z_{i2}, z_{i3}) \in R_{k_i}^J$ (with the usual abuse of notation of identifying values of Boolean variables and values in $\{0, 1\}$). We now define Σ as follows:

$$\Sigma = \left\{ \forall\mathbf{x} \bigwedge_{i=1}^{m} B(x_i) \rightarrow \exists\mathbf{y} \bigwedge_{i=1}^{n} R_{k_i}(z_{i1}, z_{i2}, z_{i3}) \right\}.$$

This reduction is clearly polynomial and yields an instance of VALIDITY$_{\text{tgd}}$. Now we have:

$$(I, J) \models \Sigma \iff \left(\forall\mathbf{x} \bigwedge_{i=1}^{m} (x_i = 0 \vee x_i = 1) \rightarrow \exists\mathbf{y} \bigwedge_{i=1}^{n} c_i(z_{i1}, z_{i2}, z_{i3}) \right) \text{ is true}$$
$$\iff \forall\mathbf{x}\exists\mathbf{y}\ \varphi(\mathbf{x}, \mathbf{y}) \text{ is true}$$

(2) Let F be the set of facts of J which are not directly in Σ as ground facts. Guess $|F|$ valuations of the variables on the left- and right-hand sides of the formulas of Σ which are not ground facts. If, for all $1 \leqslant i \leqslant |F|$, there is some $\varphi \in \Sigma$ such that the ith valuation of the left-hand side of φ holds in I (which can be decided in polynomial time) and the ith fact of F appears in the ith valuation of the right-hand side of φ, then return `true`. □

5.4. COMBINED COMPLEXITY FOR FULL TGDS. We then consider the language of full tgds, $\mathcal{L}_{\text{full}}$.

PROPOSITION 5.6

(1) VALIDITY$_{\text{full}}$ *and* VALIDITY$_{\text{full}^*}$ *are* ***coNP****-complete;*
(2) EXPLANATION$_{\text{full}}$ *is* ***NP****-hard.*
(3) ZERO-REPAIR$_{\text{full}}$ *is* ***DP****-hard.*

PROOF

(1) Here is a nondeterministic polynomial-time algorithm for $\text{VALIDITY}_{\text{full}^*}$. First check the validity of the ground facts of Σ. For the other formulas of Σ, guess a valuation of the universally quantified variables. If the left-hand side is true in I and the right-hand side false in J (both of which can be decided in polynomial time), return `false`.

For the hardness part, we use a reduction from the problem of evaluating a boolean conjunctive query over a database, which is **NP**-complete [Chandra and Merlin 1977]. Let D be a relational database of schema $\mathbf{U}$, and $Q = \exists \mathbf{y}\ \varphi(\mathbf{y})$ a boolean conjunctive query over $\mathbf{U}$. We build an instance (I, J, Σ) of $\text{VALIDITY}_{\text{full}}$ in the following way: $\mathbf{S} = \mathbf{U} \cup \{(S, 1)\}$, $I = D \cup \{S(1)\}$, $\mathbf{T} = \{(S', 1)\}$, $J = \varnothing$ and $\Sigma = \{\forall \mathbf{y} \forall t\ \varphi(\mathbf{y}) \wedge S(t) \rightarrow S'(t)\}$. This reduction is polynomial, $\Sigma \subset \mathcal{L}_{\text{full}}$, and Σ is valid with respect to (I, J) if and only if Q is false in D.

(2) There is a straightforward reduction of the problem of deciding whether a tuple is in the result of a project-join expression in the relational algebra, which is a **NP**-complete problem [Yannakakis 1981], to $\text{EXPLANATION}_{\text{full}}$. Alternatively, we can also use a reduction from 3SAT, as the one used next in the proof of the **DP**-hardness of $\text{ZERO-REPAIR}_{\text{full}}$.

(3) Let $L = L_1 \cap L_2 \in \mathbf{DP}$, with $L_1 \in \mathbf{NP}$ and $L_2 \in \mathbf{coNP}$. As 3SAT is **NP**-complete, there is a polynomial-time reduction Θ_1 from L_1 to 3SAT; similarly, we noted in item (1) that the problem of evaluating a Boolean conjunctive query over a database is **NP**-complete, therefore there is a reduction Θ_2 from L_2 to the complement of this problem. Let α be any input. $\alpha \in L$ if and only if $\Theta_1(\alpha)$ (an instance of 3SAT) is satisfiable and $\Theta_2(\alpha)$ (a conjunctive query over a relational database) evaluates to false. Let $\Theta_1(\alpha) = \bigwedge_{i=1}^{n} c_i(z_{i1}, z_{i2}, z_{i3})$, where the c_i's are 3-clauses over some set $\mathbf{x}$ of variables (and $\bigcup_{i,j} z_{ij} = \mathbf{x}$) and $\Theta_2(\alpha) = (D, \mathbf{U}, Q = \exists \mathbf{y}\ \varphi(\mathbf{y}))$, where D is a relational database of schema $\mathbf{U}$ and Q a Boolean conjunctive query over $\mathbf{U}$.

We consider the following instance (I, J, Σ) of $\text{ZERO-REPAIR}_{\text{full}}$:

$\mathbf{S} = \{(R_1, 3) \ldots (R_8, 3), (S_1, 1), (S_2, 1)\} \cup \mathbf{U}$ (we assume without loss of generality that the R_i and S_j do not appear in $\mathbf{U}$) and I such that:

the R_i^I are the 8 distinct subsets of $\{0, 1\}^3$ of cardinality 7;
$S_1^I = S_2^I = \{a\}$;
$\mathbf{U}^I = D$.

$\mathbf{T} = \{(S_1', 1), (S_2', 1)\}$ and $J = \{S_1'(a)\}$;

For each $1 \leqslant i \leqslant n$, let $1 \leqslant k_i \leqslant 8$ be the unique integer such that $c_i(z_{i1}, z_{i2}, z_{i3})$ is true if and only if $(z_{i1}, z_{i2}, z_{i3}) \in R_{k_i}^I$. We then define $\Sigma = \{\psi_1, \psi_2\}$ with:

$$\psi_1 = \forall \mathbf{x} \forall t \bigwedge_{i=1}^{n} R_{k_i}(z_{i1}, z_{i2}, z_{i3}) \wedge S_1(t) \rightarrow S_1'(t),$$

$$\psi_2 = \forall \mathbf{y} \forall t\ \varphi(\mathbf{y}) \wedge S_2(t) \rightarrow S_2'(t).$$

Observe that $(I, J) \models \psi_1$ and that the only tuple to explain in J can only be explained by ψ_1. In other words, (I, J, Σ) is a solution of $\text{ZERO-REPAIR}_{\text{full}}$ if and only if the following two conditions are satisfied: ψ_1 fully explains J with respect to I and $(I, J) \models \psi_2$. But ψ_1 fully explains J with respect to I if and only if φ is satisfiable, while $(I, J) \models \psi_2$ if and only if Q is false in D. The reduction

presented here is polynomial, and $\alpha \in L$ if and only if (I, J, Σ) is a solution of ZERO-REPAIR$_{\text{full}}$. □

5.5. COMBINED COMPLEXITY FOR ACYCLIC TGDS. The last subset of $\mathcal{L}_{\text{tgd}}$ that we consider here is $\mathcal{L}_{\text{acyc}}$.

PROPOSITION 5.7

(1) VALIDITY$_{\text{acyc}}$ *and* VALIDITY$_{\text{acyc}^*}$ *are* ***coNP****-complete.*
(2) EXPLANATION$_{\text{acyc}^*}$ *is* ***NP****-hard.*
(3) ZERO-REPAIR$_{\text{acyc}^*}$ *is* ***DP****-hard.*
(4) EXPLANATION$_{\text{acyc}^*}$ *is in* ***PTIME*** if, for all existentially quantified variables y and for all constants c, there is at most one term $y = c$ appearing in each formula of the schema mapping. *In particular,* EXPLANATION$_{\text{acyc}}$ *is in* ***PTIME****.*

PROOF

(1) — Let us first prove that VALIDITY$_{\text{acyc}^*}$ is in **coNP**. The validity of ground facts of Σ is trivial to check. Let θ be a formula of Σ which is not a ground fact. Recall that θ is of the form

$$\forall \mathbf{x}\ \varphi(\mathbf{x}) \wedge \tau(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \tau'(\mathbf{x}, \mathbf{y})$$

with $\tau(\mathbf{x})$ a conjunction of terms expressing equality or negated equality between a variable of $\mathbf{x}$ and a constant, and $\tau'(\mathbf{x}, \mathbf{y})$ of the form

$$\tau'(\mathbf{x}, \mathbf{y}) = \bigwedge_{i=1}^{n} \left(\left(\bigwedge_{j=1}^{m_i} x_{ij} = c'_{ij} \right) \rightarrow y_i = c_i \right).$$

Guess a valuation ν of the variables of $\mathbf{x}$. If the left-hand side of θ is made false by this valuation, give up this guess. Otherwise, consider the formula

$$\xi = \exists \mathbf{y}\ \psi(\nu(\mathbf{x}), \mathbf{y}) \wedge \tau'(\nu(\mathbf{x}), \mathbf{y})$$

where $\tau'(\nu(\mathbf{x}), \mathbf{y})$ is equivalent to a conjunction of terms of the form $y = c$ with $y \in \mathbf{y}$ and c a constant. We can then check in polynomial time if J satisfies ξ using Algorithm 3.

— To prove that VALIDITY$_{\text{acyc}}$ is **coNP**-hard, we use a reduction of 3SAT. Let $\varphi = \bigwedge_{i=1}^{n} c_i(z_{i1}, z_{i2}, z_{i3})$ be an instance of 3SAT, where the c_i's are 3-clauses over some set $\mathbf{x} = \{x_1 \ldots x_m\}$ of variables (and $\bigcup_{i,j} z_{ij} = \mathbf{x}$). We consider now the following instance of VALIDITY$_{\text{acyc}}$:

$\mathbf{S} = \{(B, 1)\}$ and $I = \{B(0), B(1)\}$.

$\mathbf{T} = \{(A, 3), (F, 1), (R_1, 4) \cdots (R_8, 4)\}$, $F^J = \{0\}$, A^J is the 0 and 1 truth table of the *and* operator, and each R_k^J is the following 4-ary relation of arity 8:

0	0	0	$\bar{\delta}_{k1}$
0	0	1	$\bar{\delta}_{k2}$
0	1	0	$\bar{\delta}_{k3}$
0	1	1	$\bar{\delta}_{k4}$
1	0	0	$\bar{\delta}_{k5}$
1	0	1	$\bar{\delta}_{k6}$
1	1	0	$\bar{\delta}_{k7}$
1	1	1	$\bar{\delta}_{k8}$

where $\bar{\delta}_{kj}$ is 0 if $k = j$ and 1 otherwise.

For all $1 \leqslant i \leqslant n$, let $1 \leqslant k_i \leqslant 8$ be the unique integer such that $c_i(z_{i1}, z_{i2}, z_{i3})$ is false if and only if $(z_{i1}, z_{i2}, z_{i3}, 0) \in R^J_{k_i}$ (with the usual abuse of notation of identifying values of Boolean variables and values in $\{0, 1\}$). We then define:

$$\Sigma = \Big\{ \forall x_1 \cdots \forall x_m \; B(x_1) \wedge \cdots \wedge B(x_m) \rightarrow \exists y_1 \cdots \exists y_n \exists y'_1 \cdots \exists y'_{n-1} \; F(y'_{n-1}) \\ \wedge \bigwedge_{i=1}^{n} R_{k_i}(z_{i1}, z_{i2}, z_{i3}, y_i) \\ \wedge A(y_1, y_2, y'_1) \wedge A(y'_1, y_3, y'_2) \wedge \cdots \wedge A(y'_{n-2}, y_n, y_{n-1}) \Big\}.$$

This transformation is polynomial. We prove that $\Sigma \subset \mathcal{L}_{\text{acyc}}$. First, remember that the acyclicity condition on the right-hand side of the tgd only concerns existentially quantified variables. Second, the y_i's are not sources of cycles since each only occurs in two relation atoms, one of which where they are the only existentially quantified variables. Finally, we can build a join tree for the y'_i's that consists in a simple chain, each relation atom A being a node of this chain.

Now observe that Σ is valid with respect to (I, J) if and only if the following quantified propositional formula is true:

$$\forall \mathbf{x} \; \exists y'_1 \cdots \exists y'_{n-1} \; \neg y'_{n-1} \\ \wedge (c_1(z_{11}, z_{12}, z_{13}) \wedge c_2(z_{21}, z_{22}, z_{23}) \rightarrow y'_1) \\ \wedge (y'_1 \wedge c_3(z_{31}, z_{32}, z_{33}) \rightarrow y'_2) \wedge \cdots \\ \wedge (y'_{n-2} \wedge c_n(z_{n1}, z_{n2}, z_{n3}) \rightarrow y'_{n-1}).$$

This can be rewritten as:

$$\forall \mathbf{x} \bigwedge_{i=1}^{n} c_i(z_{i1}, z_{i2}, z_{i3}) \rightarrow \bot \equiv \neg\varphi.$$

In other words, Σ is valid with respect to (I, J) if and only if the original 3SAT instance is not satisfiable.

(2) We use once more a reduction from 3SAT.

$\mathbf{S} = \{(B, 1)\}$ and $I = \{B(0), B(1)\}$;

$\mathbf{T} = \{(S, n)\}$ and $J = \{S(1, \ldots, 1)\}$;

For all $1 \leqslant i \leqslant n$, let $\tau_{i1} \cdots \tau_{i8}$ be the eight conjunctions of the form $z_{i1} = b_1 \wedge z_{i2} = b_2 \wedge z_{i3} = b_3$ such that each b_k is either the constant 0 or 1, and $c_i(z_{i1}, z_{i2}, z_{i3})$ is true if any of these conjunctions holds. We then define:

$$\Sigma = \Big\{ \forall x_1 \cdots \forall x_m \; B(x_1) \wedge \cdots \wedge B(x_m) \rightarrow \exists y_1 \cdots \exists y_n \; S(y_1 \cdots y_n) \\ \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{8} (\tau_{ij} \rightarrow y_i = 1) \Big\}.$$

Observe that Σ is valid with respect to (I, J). This transformation is polynomial, and Σ fully explains J with respect to I if and only if the original 3SAT instance is satisfiable.

(3) The problem of determining whether, given two instances φ_1 and φ_2 of 3SAT, the first is not satisfiable and the second is satisfiable, is a typical example of a **DP**-complete problem [Papadimitriou 1994]. We can reduce this problem to ZERO-REPAIR$_{\text{acyc}^*}$ in polynomial time, by combining the two reductions from 3SAT to the complement of VALIDITY$_{\text{acyc}^*}$ and from 3SAT to EXPLANATION$_{\text{acyc}^*}$ given respectively in items (1) and (2). The corresponding schemata, database instances, and schema mappings are obtained as the union of the schemata, database instances, and schema mappings of both reductions. One of the two formulas (let us call it ψ_1) comprising the resulting schema mapping Σ cannot contribute to the explanation of the target instance J, while the other (ψ_2) is valid. That means that Σ is valid and fully explains J if and only if ψ_1 is valid (or, after (1), if φ_1 is not satisfiable) and ψ_2 fully explains J (or, after (5.7), if φ_2 is satisfiable).

(4) Let f be a fact of J, we describe a polynomial algorithm for deciding whether f is explained by Σ. First, check if f is in the ground facts of Σ. Otherwise, for each atom $R(z_1 \cdots z_k)$ in the right-hand side of a formula $\theta \in \Sigma$ which is not a ground fact, such that R is the relation name appearing in f, do the following. We keep the same notations as above for the subformulas of θ.

Observe that f is explained by the $R(z_1 \cdots z_k)$ atom of θ if and only if there is a valuation ν of the variables of $\mathbf{x}$ such that $\nu(\varphi(\mathbf{x}) \wedge \tau(\mathbf{x}))$ is true in I and, for all extensions ν' of ν to $\mathbf{x} \cup \mathbf{y}$, $\nu'(\tau'(\mathbf{x}, \mathbf{y}))$ is true and $\nu'(R(z_1 \cdots z_k)) = f$.

This means that all variables of $\mathbf{y}$ appearing in $R(z_1 \cdots z_k)$ must also appear as the right-hand side of an implication of $\tau'(\mathbf{x}, \mathbf{y})$ (otherwise, an extension ν' of ν such that $\nu'(R(z_1 \cdots z_k))$ is not equal to f is possible). By the hypothesis we made, there is exactly one conjunct of $\tau'(\mathbf{x}, \mathbf{y})$ where each variable of $\mathbf{y}$ appears. Let $\rho(\mathbf{x})$ be the conjunction of the left-hand sides of these implications for all variables of $\mathbf{y}$. Then f is explained by the $R(z_1 \cdots z_k)$ atom of θ if and only if the boolean query $\exists \mathbf{x}\ \varphi(\mathbf{x}) \wedge \tau(\mathbf{x}) \wedge \rho(\mathbf{x})$ is true in the instance I. As $\tau(\mathbf{x}) \wedge \rho(\mathbf{x})$ is a simple conjunction, we can first perform the corresponding selection on I, and then use Algorithm 3 to decide if the boolean query matches. □

Note that, to prove the hardness of EXPLANATION$_{\text{acyc}^*}$, we use the repairs themselves to encode the instance of a **NP**-hard problem: although the tgd itself is acyclic, its repairs are not. We could probably get polynomial algorithms for the same problems if we impose some acyclicity condition to repairs of a formula; this, however, would weaken our notion of optimality.

5.6. COMBINED COMPLEXITY OF EXISTENCE-COST AND OPTIMALITY. With the help of Lemma 5.3, we show the intractability of EXISTENCE-COST and OPTIMALITY, in all considered languages:

PROPOSITION 5.8. EXISTENCE-COST *(respectively,* OPTIMALITY*) is **NP**-hard (respectively, **DP**-hard) in the languages $\mathcal{L}_{\text{tgd}}$, $\mathcal{L}_{\text{full}}$, $\mathcal{L}_{\text{acyc}}$, $\mathcal{L}_{\text{facyc}}$, and in the language of repairs of each of these languages.*

PROOF. To prove this result, we use, as in the proof of Proposition 5.4, a reduction from the vertex cover problem in tripartite 3-uniform hypergraphs. The core of the reduction is the same for EXISTENCE-COST and OPTIMALITY. Observe that a straightforward **DP**-complete problem obtained from this problem is that of knowing if, given two tripartite 3-uniform hypergraphs and two integers, the vertex cover of the first hypergraph is less than the first integer, and the vertex cover of the second hypergraph is greater than the second integer.

Let $\mathcal{H} = (V, E)$ be a tripartite 3-uniform hypergraph with N vertices and K an integer. We denote by $\tau(\mathcal{H})$ the minimum size of a vertex cover in $\mathcal{H}$. Let $\alpha \geqslant 1$ an integer, to be defined further. Each vertex of V is seen as a constant of $\mathcal{C}$. We also use $3(\alpha + N)$ additional constants c_{ik} and c'_{jk} with $1 \leqslant i \leqslant \alpha$, $1 \leqslant j \leqslant N$, $1 \leqslant k \leqslant 3$. We now consider the following schemata and instances:

$\mathbf{S} = \{(R, 3)\}$ and

$$\begin{aligned} I = &\{R(c_{i1}, c_{i2}, c_{i3}) \mid 1 \leqslant i \leqslant \alpha\} \\ &\cup \{R(c'_{i1}, c'_{i2}, c'_{i3}) \mid 1 \leqslant i \leqslant N\} \\ &\cup \{R(v, v', v'') \mid e = (v, v', v'') \in E\}; \end{aligned}$$

$\mathbf{T} = \{(R', 1), (S', 3)\}$ and

$$\begin{aligned} J = &\{R'(c_{i1}) \mid 1 \leqslant i \leqslant \alpha\} \\ &\cup \{S'(c_{i1}, c_{i2}, c_{i3}) \mid 1 \leqslant i \leqslant \alpha\} \\ &\cup \{S'(c'_{i1}, c'_{i3}, c'_{i3}) \mid 1 \leqslant i \leqslant N\}. \end{aligned}$$

Let $\Sigma_0 = \{\forall \mathbf{x}\ R(\mathbf{x}) \rightarrow S'(\mathbf{x})\}$. Σ_0 is valid, but fails to explain α tuples of J. Thus, $\mathsf{cost}_{(I,J)}(\Sigma_0) = 6 + 3\alpha$ (since the cost of a ground fact of arity 1 is 3, see Definition 3.6).

Let $\Sigma = \{\forall \mathbf{x}\ R(\mathbf{x}) \rightarrow R'(x_1) \wedge S'(\mathbf{x})\}$ ($\Sigma \subset \mathcal{L}_{\text{facyc}}$). An argument very similar to the one of the proof of Proposition 5.4 shows that

$$\mathsf{cost}_{(I,J)}(\Sigma) = \mathsf{size}(\Sigma) + 2\tau(\mathcal{H}) + 2N = 7 + 2\tau(\mathcal{H}) + 2N.$$

since S'^J guarantees that the size of any repair of Σ with an $x = c$ term on the left-hand side is at least:

$$\mathsf{size}(\Sigma) + 2 + 9(\alpha + N - 1) > \mathsf{size}(\Sigma) + 9N > \mathsf{cost}_{(I,J)}(\Sigma).$$

The idea now is to choose α such that $\tau(H) \leqslant K$ if and only if $\mathsf{cost}_{(I,J)}(\Sigma) \leqslant \mathsf{cost}_{(I,J)}(\Sigma_0)$. This is the case if:

$$\alpha = \frac{2(K + N) + 1}{3}.$$

With this value of α, we have $\mathsf{cost}_{(I,J)}(\Sigma_0) = 2(K + N) + 7$ and $\mathsf{cost}_{(I,J)}(\Sigma) = 2\tau(\mathcal{H}) + N + 7$, which yields the property we were looking for. However, $2(K + N) + 1$ may not be divisible by 3; in this case, we just transform the initial problem by observing that $\tau(\mathcal{H}) = K$ if and only if $\tau(\mathcal{H}') = K$ where $\mathcal{H}'$ is the tripartite 3-uniform hypergraph obtained from $\mathcal{H}$ by adding n new edges, each of which span 3 new vertices (this does not change the value of N mod 3). Up to such a transformation, we may then assume than $2(K + N) + 1$ is divisible by 3.

Let us now show that, for any schema mapping $\Sigma' \subset \mathcal{L}_{\text{tgd}}{}^*$,

$$\mathsf{cost}_{(I,J)}(\Sigma') \geqslant \min(\mathsf{cost}_{(I,J)}(\Sigma_0), \mathsf{cost}_{(I,J)}(\Sigma)). \tag{1}$$

This will conclude the proof, since we then have the following reductions, obviously polynomial and valid for any of the considered languages:

***NP**-Hardness of* EXISTENCE-COST. We have $\tau(\mathcal{H}) \leqslant K$ if and only if there exists a schema mapping whose cost with respect to (I, J) is lesser than or equal to $\mathsf{cost}_{(I,J)}(\Sigma_0) - 1$.

***DP**-Hardness of* OPTIMALITY. Let $\tilde{\mathcal{H}}$ be another tripartite 3-uniform hypergraph and $\tilde{K}$ another integer. We use similar notations: $\tilde{\Sigma}$, $\tilde{\Sigma}_0$, $\tilde{I}$, etc. We assume without loss of generality that $\mathbf{S}$, $\mathbf{T}$, $\tilde{\mathbf{S}}$, and $\tilde{\mathbf{T}}$ use different relation names, and that the constants in I and J on the one hand, and in $\tilde{I}$ and $\tilde{J}$ on the other, are disjoint. Then $\tau(\mathcal{H}) \leqslant K$ if and only if Σ is optimal with respect to (I, J), while $\tau(\tilde{\mathcal{H}}) \leqslant \tilde{K} - 1$ if and only if $\tilde{\Sigma}_0$ is not optimal with respect to $(\tilde{I}, \tilde{J})$. This means that $\tau(\mathcal{H}) \leqslant K$ and $\tau(\tilde{\mathcal{H}}) \geqslant \tilde{K}$ if and only if $\Sigma \cup \tilde{\Sigma}_0$ is optimal with respect to $(I \cup \tilde{I}, J \cup \tilde{J})$: indeed, since the relation names in $\mathbf{S}$, $\mathbf{T}$, $\tilde{\mathbf{S}}$, and $\tilde{\mathbf{T}}$ are disjoint,

$$\mathsf{cost}_{(I\cup\tilde{I},J\cup\tilde{J})}(\Sigma \cup \tilde{\Sigma}_0) = \mathsf{cost}_{(I,J)}(\Sigma) + \mathsf{cost}_{(\tilde{I},\tilde{J})}(\tilde{\Sigma}_0).$$

Let us now show (1). Given a schema mapping $\Sigma' \subset \mathcal{L}_{\text{tgd}}{}^*$, the schema mapping $\Sigma'' \subset \mathcal{L}_{\text{tgd}}$ obtained by removing all repairs from Σ' is such that $\mathsf{cost}_{(I,J)}(\Sigma') \geqslant \mathsf{cost}_{(I,J)}(\Sigma'')$. This means we only need to consider schema mappings of $\mathcal{L}_{\text{tgd}}$.

Let Σ' be a nonempty schema mapping of $\mathcal{L}_{\text{tgd}}$. Observe that, as the constants c_{i1} are completely indistinguishable from each other, Σ' must either explain all or none of the facts of R'^J. We shall consider each case in turn.

—If Σ' does not explain any of the facts of R'^J, each of these must be accounted for in the repairs, by one of the following methods:

- —adding ground facts (additional cost: 3 each);
- —adding an unconditioned "$x = c$" term to a $R'(x)$ atom where x is existentially quantified (additional cost: 2 each, but this can only be done once per $R'(x)$ atom, whose size is 1, or this yields an inconsistent formula);
- —adding a conditioned "$\tau \rightarrow x = c$" term to a $R'(x)$ atom where x is existentially quantified (minimum additional cost: 4 each, since the size of τ is at least 2).

Moreover, a repair of Σ' should also account for the facts of S'^J, either as explanations of Σ' (this cannot be done in a formula with size lesser than 6) or by enumerating all ground facts of S' (with a cost of $9(\alpha + N)$, which is greater than 6). This means that $\mathsf{cost}_{(I,J)}(\Sigma') \geqslant 3\alpha + 6 = \mathsf{cost}_{(I,J)}(\Sigma_0)$.

—In the case where Σ' explains all facts of R'^J, there is a tgd $\theta \in \Sigma'$ such that θ explains all facts of $R'J$, and θ is necessarily of the form:

$$\forall x_1 \forall x_2 \forall x_3 \forall \mathbf{u}\ R(x_1, x_2, x_3) \wedge \varphi(x_1, x_2, x_3, \mathbf{u}) \rightarrow$$
$$\exists \mathbf{v}\ R'(x_1) \wedge \psi(x_1, x_2, x_3, \mathbf{u}, \mathbf{v})$$

with $\exists \mathbf{u}\ \varphi(c_{11}, c_{12}, c_{13}, \mathbf{u})$ valid in I. Then, for all $e = (v, v', v'') \in E$,

$$\exists \mathbf{u}\ \varphi(v, v', v'', \mathbf{u})$$

is valid in I since φ does not contain anything else than relation atoms $R(w_1, w_2, w_3)$ with all x_i's necessarily in the ith position, and other variables existentially quantified. That means that $R'(v)$ is an incorrect fact implied by the tgd. As we saw earlier, adding an $x = c$ term on the left-hand side of θ has prohibitive cost. The only way to cancel these facts is then as in the proof of Proposition 5.4. Finally, a repair of θ must also explain all facts of S'^J, either as facts explained by θ itself (then, $\mathsf{size}(\varphi) \geqslant 3$), or by enumerating ground facts of S'^J, with a cost of $9(\alpha + N) > 3$.

TABLE IV. DATA COMPLEXITY RESULTS.

	$\mathcal{L}_{\text{tgd}}$, $\mathcal{L}_{\text{full}}$, $\mathcal{L}_{\text{acyc}}$, $\mathcal{L}_{\text{facyc}}$ $\mathcal{L}_{\text{tgd}}{}^*$, $\mathcal{L}_{\text{full}}{}^*$, $\mathcal{L}_{\text{acyc}}{}^*$, $\mathcal{L}_{\text{facyc}}{}^*$
VALIDITY	**PTIME**
EXPLANATION	**PTIME**
ZERO-REPAIR	**PTIME**
COST (K fixed)	**PTIME**
COST (Σ fixed)	**NP**, **NP**-hard for some Σ
EXISTENCE-COST	**PTIME**
OPTIMALITY	$\mathbf{\Pi}_2^P$, **DP**-hard for some Σ

We have therefore:

$$\begin{aligned}\mathsf{cost}_{(I,J)}(\Sigma') \geqslant \mathsf{cost}_{(I,J)}(\theta) &\geqslant 7 + 2(\tau(\mathcal{H}) + N) \\ &\geqslant \mathsf{cost}_{(I,J)}(\Sigma). \quad \square\end{aligned}$$

Note that the same proof does not work in the case of $\mathcal{L}_{\text{rc}}$:

$$\begin{aligned}&\{\forall x_1 \forall x_2 \forall x_3 \; R(x_1, x_2, x_3) \wedge \\ &\quad \neg \left(\exists x_2' \exists x_3' \; R(x_1, x_2', x_3') \wedge (x_2 \neq x_2' \vee x_3 \neq x_3')\right) \rightarrow R'(x)\}\end{aligned}$$

may have a lower cost for some instances than Σ (for instance if $\mathcal{H}$ is a hypergraph where all nodes have a degree greater than 1).

5.7. DATA COMPLEXITY. As far as data complexity is concerned, the situation is simpler, since we do not have any difference in complexity for all four subsets of $\mathcal{L}_{\text{tgd}}$. The results are presented summarized in Table IV.

PROPOSITION 5.9

(1) *If Σ is fixed,* $\text{VALIDITY}_{\text{rc}^*}$ *is in* ***PTIME****.*
(2) *If Σ is fixed,* $\text{EXPLANATION}_{\text{tgd}^*}$ *is in* ***PTIME****.*
(3) *If K is fixed,* $\text{COST}_{\text{tgd}^*}$ *and* $\text{EXISTENCE-COST}_{\text{tgd}^*}$ *are in* ***PTIME****.*
(4) *For some fixed value of Σ,* $\text{COST}_{\text{facyc}}$ *is* ***NP****-hard.*
(5) *For some fixed value of Σ, the problem* OPTIMALITY *in $\mathcal{L}_{\text{tgd}}$, $\mathcal{L}_{\text{full}}$, $\mathcal{L}_{\text{acyc}}$, $\mathcal{L}_{\text{facyc}}$ and the language of repairs of each of these is* ***DP****-hard.*

PROOF

(1) If k is the number of quantified variables in a first-order formula φ in prenex normal form, it is easy to see that checking whether φ is valid in a database of size n is $O(n^k)$.

(2) Each formula of Σ is either a ground fact, or of the form $\forall \mathbf{x}\, (\varphi(\mathbf{x}) \wedge \tau(\mathbf{x})) \rightarrow \exists \mathbf{y}\, (\psi(\mathbf{x}, \mathbf{y}) \wedge \tau'(\mathbf{x}, \mathbf{y}))$ with τ and τ' propositional combinations of terms expressing equalities between a variable and a constant. For each fact f of J, first check if it appears as a ground fact Σ; otherwise, for each valuation of $\mathbf{x} \cup \mathbf{y}$ (there is a constant number of such valuations, since Σ is fixed), check whether the left-hand side holds, and f is a consequence of the right-hand side.

(3) We can just enumerate all schema mappings of $\mathcal{L}_{\text{tgd}}{}^*$ whose size is lower than K, and check in polynomial time if they are valid and fully explain the target instance.

(4) This results from the proof of Proposition 5.4.
(5) This results from the proof of Proposition 5.8. □

Unsurprisingly, in terms of data complexity, the different problems appear to be somewhat more tractable than in terms of combined complexity, even though $\text{COST}_{\text{facyc}}$ is still **NP**-hard for a fixed Σ. Note, however, that since the objective is the discovery of schema mappings from database instances, the schema mapping itself is the object of interest. This is different from the traditional setting of query answering over databases, for instance, where it makes sense of reasoning about a fixed query and a varying dataset. In that sense, it seems to us that combined complexity results are more meaningful here.

6. *Extension and Variants*

We study in this section some extensions of our optimality notion to (i) formulas of the full relational calculus; (ii) other apparently simpler functions of cost.

6.1. EXTENSION TO THE RELATIONAL CALCULUS. We can extend the definitions of Section 3 to the language $\mathcal{L}_{\text{rc}}$ of relational calculus, by the following adaptation of the notion of repair.

A repair of a schema mapping $\Sigma \subset \mathcal{L}_{\text{rc}}$ is a set of formulas obtained from Σ by a finite sequence of the following operations:

—Replacing in a formula of Σ a subformula $\forall \mathbf{x}\ \varphi(\mathbf{x}, \mathbf{y})$ (we also assume that φ does not start with a $\forall$ symbol, and that the sub-formula is not preceded by a $\forall$ symbol) by $\forall \mathbf{x}\ \tau(\mathbf{x}, \mathbf{z}) \rightarrow \varphi(\mathbf{x}, \mathbf{y})$ where $\mathbf{z}$ is the set of variables free in φ and τ is a Boolean formula over terms $w = c$ of the following form:

$$\bigwedge_i \left(\left(\bigwedge_j z_{ij} = c'_{ij} \right) \rightarrow x_i \alpha_i c_i \right)$$

with z_{ij} variables from $\mathbf{z}$, x_i variables from $\mathbf{x}$, α_i either $=$ or $\neq$, and c'_{ij} and c_i constants.

—Replacing in a formula of Σ a subformula $\exists \mathbf{x}\ \psi(\mathbf{x}, \mathbf{y})$ (we also assume that ψ does not start with a $\exists$ symbol, and that the subformula is not preceded by a $\exists$ symbol) by $\exists \mathbf{x}\ \psi(\mathbf{x}, \mathbf{y}) \wedge \tau'(\mathbf{x}, \mathbf{z})$ where $\mathbf{z}$ is the set of variables free in ψ and τ' is a boolean formula over terms $w = c$ of the following form:

$$\bigwedge_i \left(\left(\bigwedge_j z_{ij} = c'_{ij} \right) \rightarrow x_i = c_i \right)$$

with z_{ij} variables from $\mathbf{z}$, x_i variables from $\mathbf{x}$, and c'_{ij} and c_i constants.

—Adding to Σ a ground fact $R(c_1 \cdots c_n)$ with R a relation of the target schema of arity n, and $c_1 \cdots c_n$ constants.

We can check that this definition amounts to the same as Definition 3.4 if we restrict ourselves to $\mathcal{L}_{\text{tgd}}$. We can then use the same definitions of the size and cost of a formula, and consider the same decision problems. It is easy to see that the results of Proposition 5.1 still hold. We have the following complexity results:

PROPOSITION 6.1

(1) VALIDITY$_{rc^*}$ *and* VALIDITY$_{rc^*}$ *are* ***PSPACE****-complete;*

(2) EXPLANATION$_{rc^*}$ *is co-recursively enumerable;*

(3) EXPLANATION$_{rc}$ *and* ZERO-REPAIR$_{rc}$ *are not recursive.*

PROOF

(1) — Let us first show that VALIDITY$_{rc^*}$ is in **PSPACE**. Let $\varphi \in \Sigma$. If φ is a ground fact, this is trivial. Otherwise, we first rewrite φ in prenex normal form

$$\alpha_1 x_1 \alpha_2 x_2 \ldots \alpha_n x_n \ \psi(x_1, \ldots, x_n)$$

where each α_i is either $\exists$ or $\forall$.

Let C be the set of constants appearing in I and J, along with n distinct constant $\perp_1 \ldots \perp_n$. It can be shown that we do not need to consider valuations of the $x_1 \cdots x_n$ into other constants. For each valuation ν of $x_1 \cdots x_n$ into C, it is decidable in polynomial time whether $(I, J) \models \psi(\nu(x_1), ..., \nu(n_n))$. We then enumerate all valuations, enumerating recursively all valuations of x_{i+1} while keeping x_i fixed, and remembering for each $1 \leqslant i \leqslant n+1$ a single value ok_i which is equal to:

If $i = n + 1$: 1 if $(I, J) \models \psi(\nu(x_1), \ldots, \nu(n_n))$ in the current valuation ν, 0 otherwise;

If $\alpha_i = \exists$: the maximum of ok_{i+1} and the preceding value of ok_i (and the preceding value is reset to 0 whenever the valuation of x_{i-1} is changed).

If $\alpha_i = \forall$: the preceding value of ok_i, multiplied par the current value of ok_{i+1} (and the preceding value is reset to 1 whenever the valuation of x_{i-1} is changed).

The algorithm returns `true` if, after enumerating all valuations, all ok_i's are equal to 1. Otherwise, the algorithm returns `false`. This is obviously a **PSPACE** algorithm, and it returns `true` if and only if $(I, J) \models \varphi$. We can run the same algorithm in sequence on all $\varphi \in \Sigma$.

—The **PSPACE**-hardness of VALIDITY$_{rc}$ comes from a polynomial-time reduction of QSAT (also known as QBF), which is **PSPACE**-complete [Papadimitriou 1994; Stockmeyer and Meyer 1973]. Let $\varphi = \alpha_1 x_1 \cdots \alpha_n x_n \ \psi(x_1 \cdots x_n)$ be an instance of QSAT (the α_i's are either existential or universal quantifiers, and $\psi(x_1 \cdots x_n)$ is a propositional formula in CNF with variables $x_1 \cdots x_n$).

Let $\mathbf{S} = \{(B, 1), (T, 1), (F, 1)\}$, $\mathbf{T} = \varnothing$, $I = \{B(0), B(1), T(1), F(0)\}$ and $J = \varnothing$. We rewrite inductively the quantified propositional formula φ into a first-order formula ξ in the following way:

From the CNF propositional formula $\psi(x_1 \cdots x_n)$, we obtain a first-order formula with all free variables by replacing every positive literal x_i with $T(x_i)$ and every negative literal $\neg x_i$ with $F(x_i)$.

A $\forall x_i \theta_i$ subformula with universal quantification is rewritten as $\forall x_i \ (B(x_i) \rightarrow \theta_i)$.

A $\exists x_i \theta_i$ subformula with existential quantification is rewritten as $\exists x_i \ (B(x_i) \wedge \theta_i)$.

Then $(I, J) \models \{\xi\}$ if and only if φ is true.

(2) To see that EXPLANATION$_{rc^*}$ is co-recursively enumerable, just enumerate all instances K of the target schema, and whenever they are such that $(I, J) \models \Sigma$

(which is decidable, see just above), see if they contain K. If this is not the case, we can conclude that Σ does not fully explain J with respect to I.

(3) The uncomputability comes from a reduction of the satisfiability of the relational calculus, which is not recursive [Trakhtenbrot 1963; Di Paola 1969]. The reduction is the same for both problems EXPLANATION$_{\text{rc}}$ and ZERO-REPAIR$_{\text{rc}}$. Let φ be a formula of the relational calculus over a schema $\mathbf{U}$. Consider the following instance of EXPLANATION$_{\text{rc}}$ (this is also an instance of ZERO-REPAIR$_{\text{rc}}$):

$$\mathbf{S} = \{(R, 1)\} \text{ and } I = \{R(a)\};$$
$$\mathbf{T} = \mathbf{U} \cup \{(R', 1)\} \text{ and } J = \{R'(a)\};$$
$$\Sigma = \{\forall x \; R(x) \rightarrow \varphi \vee R'(x)\}$$

Observe that $(I, J) \models \Sigma$, therefore (I, J, Σ) is a solution of EXPLANATION$_{\text{rc}}$ if and only if it is a solution of ZERO-REPAIR$_{\text{rc}}$.

Now, (I, J, Σ) is a solution of EXPLANATION$_{\text{rc}}$ if and only if, for all K such that $(I, K) \models \Sigma$, $R'(a)$ is a fact of K. This is the case if and only if φ is not satisfiable. □

Interestingly, the computability of OPTIMALITY$_{\text{rc}}$ remains open. It seems to be a "harder" problem than ZERO-REPAIR$_{\text{rc}}$, but as there is no simple reduction between them, we cannot even be sure that OPTIMALITY$_{\text{rc}}$ is not recursive. We do not even know whether it is recursively enumerable or co-recursively enumerable (but COST$_{\text{rc}^*}$ and EXISTENCE-COST$_{\text{rc}^*}$ are both co-recursively enumerable because of the co-recursive enumerability of ZERO-REPAIR$_{\text{rc}^*}$).

Note that a related problem is studied in Fletcher [2007], where it is shown that determining whether there exists a schema mapping in $\mathcal{L}_{\text{rc}}$ that is valid and explain all facts of the target instance is **coNP** and co-graph-isomorphism-hard. In the case of $\mathcal{L}_{\text{rc}}{}^*$, such a mapping obviously always exists since one can enumerate all ground facts of the target instance.

6.2. VARIANTS OF THE COST FUNCTION. The definition of repairs and cost that we presented in Section 3 may appear, at first, unnecessarily complicated. We argued in Section 4 for a justification of this notion by showing that it has nice properties with respect to instances that are derived from each other with elementary operations of the relational algebra. We consider in this section two alternative definitions of cost and optimality of a schema mapping with respect to database instances, and show that neither, although simpler and perhaps more intuitive, present the same properties and are thus adapted to our context.

We keep our notions of validity of a schema mapping, of full explanation of a database instance by a schema mapping, and of size of a schema mapping, and we want to consider alternative ways to characterize the *cost* of a given schema mapping. The first idea is to assign as the cost of a schema mapping the minimal number of tuples that have to be added or removed to the target instance J for the schema mapping to become valid and to fully explain J. (Each tuple may also be weighted by its arity, to get something closer to our original cost definition.) Thus, the cost of the empty schema mapping corresponds to the size of the target instance, as before, while the cost of a schema mapping that fully explains the target instance but also incorrectly explains some tuples is the (possibly weighted) number of such tuples. This sounds like a reasonable definition, but it presents an important problem: We lose the linear bound on the cost of a schema mapping in

the size of the data and the schema mapping itself. Indeed, consider the following schema mapping, for a given n, where R is a source relation of arity 1 and R' a target relation of arity n:

$$\{\forall x_1 \cdots \forall x_n \; R(x_1) \wedge \cdots \wedge R(x_n) \rightarrow R'(x_1, \ldots, x_n)\}.$$

If J is empty, the cost of this schema mapping according to the definition given in this paragraph is $|I|^n$ (or $n|I|^n$ if we weight with the arity of the relations), which is exponential in the size of the schema mapping. This combinatorial explosion discourages all hopes of getting an optimal schema mapping by local search techniques. Besides, all the problems that we describe for the variant that we consider next also arise here.

An alternate definition of cost, close to the previous one but for which we still have a linear bound on the cost of a mapping is the following: The cost of a schema mapping Σ is the minimal number of tuples to add or remove from the source and target instances I and J so that Σ becomes valid and fully explains J. As before, we assume that we weight tuples by their arity; we could also choose to add an arbitrary constant weight to operations on J with respect to operations on I, or to deletion with respect to addition of tuples, without much difference. The linear bound is clear since we can just remove all tuples of I and of J for Σ to be valid and to fully explain J. However, there is still a fundamental problem with this definition, which can be seen by looking back at Section 4. We showed there that, for elementary operations of the relational algebra, the definition of optimality of Section 3 yielded the same as the intuitive tgds expressing these operations. This is not true any more here, however, in particular in the presence of selections and projections. For projections, this is due to the fact that a schema mapping that predicts existentially quantified tuples has a higher cost than the same schema mapping where these existentially quantified relation atoms are removed. We exhibit next a concrete example of database instances that illustrate the problem with selections.

Example 6.2. Let us consider instances I and J of the following schemata: $\mathbf{S} = \{(P, 2)\}$ and $\mathbf{T} = \{(P', 1)\}$, where: I contains a list of titles of publications (as first attribute) along with their kind: `article`, `book`, `report`, etc.; J contains a list of book titles. Let us assume that J and I contain the same book titles. In other words, $J = \pi_1(\sigma_{2=\texttt{book}}(I))$. It is quite natural to expect $\Sigma = \{\forall x \forall y \; P(x, y) \rightarrow P'(x)\}$ as the "optimal" schema mapping in the language of tgds for these database instances, and indeed, $\mathsf{cost}_{(I,J)}(\Sigma) = 5$ is minimal as soon as J is large enough and there is no hidden relation between the second attribute of I and J. Now, observe that with the variant proposed in the preceding paragraph, the cost will be: $3 + \min(2 \cdot (|I| - |J|), |J|)$, which is, in all cases when there are more publications of another kind than `book` (a common situation), greater than the cost of the empty schema mapping, which is then the optimal schema mapping for these instances.

Then, although our definition of optimality is a bit complex, it is much more adapted to the addressed problem than these simpler definitions, since it can capture such things as the worth of an existentially quantified relation atom, or the possibility of limiting the scope of a tgd with a simple selection.

7. *Conclusion*

We discussed a theoretical framework that addresses the problem of finding a schema mapping *optimal* with respect to a pair of database instances, based solely on the structure and occurrences of constants in the instances. We showed that this problem is **DP**-hard (in particular, both **NP**-hard and **coNP**-hard) even for a very restricted language, namely full acyclic tuple-generating dependencies. This is not unexpected, since it is well known that such learning problems have high complexity even in very simple cases (see, e.g., Gottlob et al. [1997] for ILP). Such a study is still useful since (i) it provides a formal framework for the discovery of schema mappings; (ii) complexity lower bounds are useful to detect the source of the complexity; (iii) complexity upper bounds often give practical algorithms.

There are a number of open theoretical issues, especially on the computability and precise complexity of OPTIMALITY, but the most obvious direction for future work would be to connect such a theoretical framework with practical heuristics and approximation algorithm; in particular, the relation to inductive logic programming has to be explored. We believe that this is an especially important problem, and that discovering and understanding hidden relations in data is one of the most fundamental tasks of artificial intelligence. Other problems of interest would be to improve our complexity upper bounds by generalizing the notion of acyclicity to that of bounded hypertree width [Gottlob et al. 1999], and to look at the same problems when some fixed set of preconditions on instances is given. One can imagine for instance that some part of the schema mapping is known in advance, or that an approximate schema mapping has been discovered by a schema matching algorithm, and that we extend it using the framework developed in this article. Finally, a potentially fruitful direction of research would be to explore the connection between the work presented here and other works on schema mapping operators [Bernstein and Chiu, 1981; Fagin et al. 2004; 2007]: Is the composition (in the sense of Fagin et al. [2004]) of two schema mappings from I to J and J to K optimal with respect to I and K? Are there cases when our framework can be used to minimize the representation of such a schema mapping? Is it possible (and perhaps easier) to find the optimal schema mapping from J and I and then construct its quasi-inverse [Fagin et al. 2007] as an optimal schema schema mapping from I to J? Note that works on data exchange usually consider additional dependencies in addition to source-to-target tgds (egds, source-only tgds, etc.), and that more expressive languages for schema mappings may be required (e.g., second-order tgds in the case of Fagin et al. [2004], tgds with constants and inequalities for Fagin et al. [2007]).

ACKNOWLEDGMENT. We would like to thank Serge Abiteboul and Yann Ollivier for their input and feedback about this work. We also want to acknowledge the anonymous referees for their valuable comments.

REFERENCES

ARENAS, M., BERTOSSI, L., AND CHOMICKI, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the ACM SIGACT-SIGMOD-SIGAAT Symposium on Principles of Database Systems (PODS)*. ACM, New York.

BEERI, C., FAGIN, R., MAIER, D., MENDELZON, A., ULLMAN, J., AND YANNAKAKIS, M. 1981. Properties of acyclic database schemes. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York.

BERNSTEIN, P. 2003. Applying model management to classical meta data. In *Proceedings of Conference on Innovative Data Systems*.

BERNSTEIN, P. A., AND CHIU, D.-M. W. 1981. Using semi-joins to solve relational queries. *J ACM 28,* 1, 25–40.

CHANDRA, A. K., AND MERLIN, P. M. 1977. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York.

CRESCENZI, V., MECCA, G., AND MERIALDO, P. 2001. Roadrunner: Towards automatic data extraction from large Web sites. In *Proceedings of Symposium on Very Large Databases (VLDB)*. VLDB Endowment.

DI PAOLA, R. A. 1969. The recursive unsolvability of the decision problem for the class of definite formulas. *J ACM 16,* 2, 324–327.

DIESTEL, R. 2005. *Graph Theory*. Springer-Verlag, New York.

FAGIN, R., KOLAITIS, P. G., MILLER, R. J., AND POPA, L. 2003. Data exchange: Semantics and query answering. In *Proceedings of the International Conference on Database Theory (ICDT)*. Springer-Verlag, Berlin, Germany.

FAGIN, R., KOLAITIS, P. G., POPA, L., AND TAN, W. C. 2007. Quasi-inverses of schema mapping. In *Proceedings of the ACM SIGACT-SIGMOD-SIGAAT Symposium on Principles of Database Systems (PODS)*. ACM, New York.

FAGIN, R., KOLAITIS, P. G., TAN, W.-C., AND POPA, L. 2004. Composing schema mappings: Second-order dependencies to the rescue. In *Proceedings of the ACM SIGACT-SIGMOD-SIGAAT Symposium on Principles of Database Systems (PODS)*. ACM, New York.

FLETCHER, G. H. L. 2007. On the data mapping problem. Ph.D. dissertation, Indiana University.

GAREY, M. R., AND JOHNSON, D. S. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York.

GOTTLOB, G., LEONE, N., AND SCARCELLO, F. 1997. On the complexity of some inductive logic programming problems. In *Proceedings of the International Conference of Inductive Logic Programming (ILP)*.

GOTTLOB, G., LEONE, N., AND SCARCELLO, F. 1999. Hypertree decompositions and tractable queries. In *Proceedings of the ACM SIGACT-SIGMOD-SIGAAT Symposium on Principles of Database Systems (PODS)*. ACM, New York.

HAAS, L. M., HERNÁNDEZ, M. A., HO, H., POPA, L., AND ROTH, M. 2005. Clio grows up: from research prototype to industrial tool. In *Proceedings of SIGMOD*. ACM, New York, 805–810.

ILIE, L., SOLIS-OBA, R., AND YU, S. 2002. Reducing the size of NFAs by using equivalences and preorders. In *Combinatorial Pattern Matching*.

KOLAITIS, P. G. 2005. Schema mappings, data exchange, and metadata management. In *Proceedings of the ACM SIGACT-SIGMOD-SIGAAT Symposium on Principles of Database Systems (PODS)*. ACM, New York.

KŐNIG, D. 1936. *Theorie der endlichen und unendlichen Graphen*. Akademische Verlagsgesellschaft, Leipzig, Germany.

LAVRAČ, N., AND DŽEROSKI, S. 1994. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York.

LENZERINI, M. 2002. Data integration: a theoretical perspective. In *Proceedings of the ACM SIGACT-SIGMOD-SIGAAT Symposium on Principles of Database Systems (PODS)*. ACM, New York.

LI, M., AND VITÁNYI, P. 1997. *An Introduction to Kolmogorov Complexity and Its Applications*, second ed. Springer-Verlag, New York.

PAPADIMITRIOU, C. H. 1994. *Computational Complexity*. Addison Wesley, Reading, MA.

RAHM, E., AND BERNSTEIN, P. A. 2001. A survey of approaches to automatic schema matching. *VLDB J 10,* 4, 334–350.

SCHAEFER, M., AND UMANS, C. 2002. Completeness in the polynomial hierarchy, a compendium. *SIGACT News 33,* 3 (Sept.), 32–49.

SENELLART, P., AND GOTTLOB, G. 2008. On the complexity of deriving schema mappings from database instances. In *Proceedings of the ACM SIGACT-SIGMOD-SIGAAT Symposium on Principles of Database Systems (PODS)*. ACM, New York, 23–32.

STOCKMEYER, L. J., AND MEYER, A. R. 1973. Word problems requiring exponential time. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York.

TARJAN, R. E., AND YANNAKAKIS, M. 1984. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J Comput 13,* 3, 566–579.

TRAKHTENBROT, B. A. 1963. Impossibility of an algorithm for the decision problem in finite classes. *AMS Translations Series 2 23*, 1–5.
WRATHALL, C. 1976. Complete sets and the polynomial-time hierarchy. *Theoret Comput Sci 3,* 1, 23–33.
YANNAKAKIS, M. 1981. Algorithms for acyclic database schemes. In *Proceedings of the Symposium on Very Large Databases (VLDB)*. VLDB Endowment.

RECEIVED JANUARY 2009; REVISED SEPTEMBER 2009; ACCEPTED SEPTEMBER 2009