Data Exchange and Schema Mappings in Open and Closed Worlds

Leonid Libkin
University of Edinburgh
libkin@inf.ed.ac.uk

Cristina Sirangelo
University of Edinburgh
csirange@inf.ed.ac.uk

ABSTRACT

In the study of data exchange one usually assumes an open-world semantics, making it possible to extend instances of target schemas. An alternative closed-world semantics only moves 'as much data as needed' from the source to the target to satisfy constraints of a schema mapping. It avoids some of the problems exhibited by the open-world semantics, but limits the expressivity of schema mappings. Here we propose a mixed approach: one can designate different attributes of target schemas as open or closed, to combine the additional expressivity of the open-world semantics with the better behavior of query answering in closed worlds.

We define such schema mappings, and show that they cover a large space of data exchange solutions with two extremes being the known open and closed-world semantics. We investigate the problems of query answering and schema mapping composition, and prove two trichotomy theorems, classifying their complexity based on the number of open attributes. We find conditions under which schema mappings compose, extending known results to a wide range of closed-world mappings. We also provide results for restricted classes of queries and mappings guaranteeing lower complexity.

Categories and Subject Descriptors. H.2.5 [**Database Management**]: Heterogeneous Databases—*Data translation*; H.2.4 [**Database Management**]: Systems—*Relational databases, Query processing*

General Terms. Theory, Languages

Keywords. Data exchange, schema mappings, closed world assumption, open world assumption, incomplete information

1. Introduction

Data exchange is the problem of finding an instance of a target schema, given an instance of a source schema and a specification of a mapping between the source and the target schemas, and answering queries over target instances. The study of both data exchange and schema mappings has been actively pursued recently (see, e.g., recent SIGMOD and PODS keynotes [17, 6]). Existing implementations [25, 27] have been incorporated into major database products.

Theoretical foundations of data exchange were first developed in [10, 11]. For a source instance S and a schema mapping M, a target instance T is a *solution* for S if S and T together satisfy the conditions of M. Target instances often contain incomplete information as mappings are rarely fully specified: for example, it is common for target schemas to have attributes that are not present in the source. To account for missing information, target instances are populated with *nulls*.

Papers [10, 11] also developed query answering techniques for data exchange that work very well for positive relational algebra queries, but have been shown to exhibit strange behavior for queries involving negation. This happens even with very simple mappings, for example, mapping specifying that each tuple from the source be copied into the target [10, 3]. There are several reasons for such unnatural behavior, stemming from handling of incomplete information. We shall outline them below.

A source instance S may have many different solutions under a mapping M. Thus, the standard approach for answering a query Q over the target schema is to find *certain answers* $\mathsf{certain}_M(Q, S)$. These were defined in [10, 17] as the intersection of $Q(T)$'s for all solutions T. Normally, only one target instance T_0 is materialized (typically a canonical solution [10] or its core [11]). Hence, the goal of query answering in data exchange is to compute certain answers, by posing a query against that materialized instance. That is, one needs to evaluate some query Q' so that $\mathsf{certain}_M(Q, S) = Q'(T_0)$.

However, solutions T's (including the materialized solution T_0) are instances with nulls, and there is no well-defined concept of $Q(T)$ for databases with nulls [16, 21, 14]. Most commonly, one tries to find the set $\Box Q(T)$ of certain answers to Q over T, i.e., answers independent of the interpretation of nulls. There are several known evaluation mechanisms for computing them. The one used in [10, 11] is the *naive* *evaluation* $Q_{naive}(T)$: it treats nulls as atomic values (i.e., two nulls are equal iff they are syntactically the same) and only keeps null-free tuples in the output.

For conjunctive queries, and their unions, [10, 11] proved that $\mathsf{certain}_M(Q, S)$, defined as the intersection of $Q_{naive}(T)$ over all solutions T, can be computed as $Q_{naive}(T_0)$, where T_0 is the canonical solution. This follows from

$$\begin{aligned} \mathsf{certain}_M(Q, S) &= \square Q(T_0) & (1) \\ \square Q(T_0) &= Q_{naive}(T_0) & (2) \end{aligned}$$

for such queries. However, for full relational algebra, even if (1) were to remain true, relying on (2) for finding the result of a query is impossible, as the naive evaluation no longer produces the set of certain answers [16]. Moreover, [3] showed that there are relational calculus queries Q for which $\mathsf{certain}_M(Q, S)$ cannot be expressed as $Q'_{naive}(T_0)$, where Q' is a relational calculus (or even an aggregate) query.

Furthermore, the notion of solutions is not unique (see, e.g., [10, 11, 20]) and neither is the notion of $\square Q$ in general, as both depend on assumptions about tuples in solutions and interpretation of nulls. Papers [11, 10, 12] make the *Open World Assumption*, or *OWA* [28]. Under this assumption, tuples can be freely added to solutions. For example, if M is a mapping stating that tuples from the source S must be copied to the target T, then, under the OWA, every T that extends S is a solution. In particular, computing certain answers is as hard as finite validity (which is undecidable for relational calculus) even in such simple settings.

There is an alternative notion of solutions, proposed in [20, 15]. It is based on the *Closed World Assumption*, or *CWA* [28]. Such solutions T have "just as much as needed" to satisfy the conditions imposed by M. For example, if M states that every tuple in S must be in T, the only CWA-solution for S would be a copy of S, since instances are no longer open to adding new tuples. This approach guarantees $\mathsf{certain}_M(Q, S) = \square Q(T_0)$ for the canonical solution T_0, and eliminates some of the anomalies that have been shown to arise under the OWA approach [3]. On the other hand, under the CWA queries may produce counterintuitive answers too, this time because of the "uniqueness of value" constraints imposed by the CWA.

Fully open or fully closed mappings, being two extreme cases, are bound to have their shortcomings. Thus, *our goal* is to study mappings that are not rigidly controlled by the OWA, as in [10, 12], or by the CWA, as in [20, 15]. We adapt an old idea of [13], and permit nulls – or, more generally, attributes in targets – to be *open* or *closed*. Open attributes can be instantiated by many values, but for closed, only one value is permitted. We now illustrate this idea by an example.

Example Consider a source schema σ with binary relations *Papers(paper#, title)* and *Assignments(paper#, reviewer)*. Each instance of σ represents the list of papers submitted to a given conference and the assignments of papers to reviewers. The target schema τ consists of two binary relations *Reviews(paper#, review)* and *Submissions(paper#, author)*. The mapping between the source and the target is provided by a set of rules below:

$$\begin{array}{lll} Submissions(x^{cl}, z^{op}) & :- & Papers(x, y) \\ Reviews(x^{cl}, z^{cl}) & :- & Assignments(x, y) \\ Reviews(x^{cl}, z^{op}) & :- & Papers(x, y) \wedge \\ & & \neg\exists r\, Assignments(x, r) \end{array}$$

We use the syntax that will be introduced later; essentially, we formulate mappings as in [10, 12] (using rule-based notation as in [20]), with extra annotations *op* or *cl* (for open and closed) of variables in the target atoms. Intuitively, the first rule says that the target instance contains exactly the submitted papers from the source (enforced by the closed annotation of the attribute *paper#*). The *author* attribute is populated with nulls, and its open annotation models the one-to-many relationship between papers and their authors.

The second rule says that for each assigned paper and each of its reviewers, exactly one review is associated to the paper in the target. Completely closed annotation here prevents the target from having reviews of assigned papers without a corresponding reviewer in the source. The third rule deals with papers that have not been assigned, according to the source. In this case, the attribute *review* of *Reviews* is annotated as open, to allow several reviews to be generated for the same paper.

We remark that atoms of the same relation can be annotated differently in different rules. Indeed, the annotation of an atom of a given target relation R in a rule describes the way *the particular rule* allows data to be moved from the source to relation R in the target, and this may vary from a rule to a rule. $\square$

Open/closed annotations could be an easy addition to systems that handle schema mappings [25, 27, 7] as they essentially state whether we have a one-to-one or a one-to-many relationship for a correspondence between attributes in the source and the target, and only require one-bit annotations for target attributes.

Contributions Our first goal is to study data exchange based on mappings that allow annotating target attributes as open or closed. We define their semantics via different interpretations of null values, and show the following:

- The solutions of [10, 20] are the two extreme cases: when all attributes are open (solutions of [10]), and when all are closed (solutions of [20]).
- For conjunctive (and positive) queries, certain answers can be computed by the tractable naive evaluation, regardless of annotations.
- Under the appropriate notion of certain answers with mixed open and closed nulls, we always have (1) – that is, $\mathsf{certain}_M(Q, S) = \square Q(T_0)$, where T_0 is the canonical solution. Thus, query answering in data exchange is reduced to query answering over a particular polynomial-time computable instance with nulls.
- For full relational algebra, we prove a *trichotomy* result, classifying the complexity of certain answers in terms of the maximum number k of open attributes per atom in a rule of the mapping M: it is coNP-complete if $k = 0$ (under the CWA), it is coNEXPTIME-complete if $k = 1$, and undecidable for $k > 1$. Lower complexity can be achieved by putting additional restrictions on queries.

We then study schema mappings themselves. This subject too has witnessed a lot of activity recently (see [6]). A central topic is the study of operations on mappings, with perhaps the most common one being *composition*: for mappings $M_{\sigma\tau}$ and $M_{\tau\omega}$ between schemas σ and τ, and τ and ω, resp., how do we obtain a mapping $M_{\sigma\omega}$ that transforms σ-databases into ω-databases by applying $M_{\sigma\tau}$ first, followed by $M_{\tau\omega}$?

Composition is crucial for understanding schema evolution, and it has been extensively studied [5, 12, 26, 22]. The idea of the standard approach of [12] is to define composition semantically, and then capture the same notion syntactically. Semantically, a schema mapping M is a binary relation with pairs (S, T) such that T is a solution for S. Then the composition of mappings is the composition of binary relations. The definition of [12] does not permit instances with nulls, and interprets both mappings and solutions under the OWA. Then, under the OWA, [12] showed how to capture the semantic notion of composition syntactically with Skolemized constraints. But it is then natural to ask what happens if a different interpretation, e.g. closed-world, is used?

As our second contribution, we study composition of schema mappings that mix open and closed attributes. The notion of [12] is obtained when all attributes are interpreted under the OWA. Our main results are:

- We classify the complexity of composition (i.e., recognizing pairs of instances that belong to the composition of two mappings) by the maximum number k of open attributes in rules of $M_{\sigma\tau}$, proving another trichotomy: NP-completeness for $k = 0$; NEXPTIME-completeness for $k = 1$; and undecidability for $k > 1$.
- If only conjunctive queries are used in mappings (as in [10, 11, 12]), then under both CWA and OWA the composition problem is NP-complete.
- We show that the Skolemized constraints of [12] are closed under composition not only under the OWA but also under the CWA, and look at other conditions that make composition work for mixed open/closed mappings.

Organization In Section 2 we review schema mappings, data exchange solutions, and the basics of incomplete information. Section 3 introduces mappings that combine open and closed-world semantics. Complexity of query answering under such mappings is studied in Section 4. In Section 5 we study the complexity and syntactic characterizations of mapping composition. Concluding remarks are in Section 6.

2. Preliminaries

Schema mappings and data exchange

Let σ and τ be two relational database schemas; σ is thought of as a *source* schema, and τ as a *target* schema. A mapping M between schemas σ and τ is a condition that states how instances of σ and τ are related [6, 17, 18]. In data exchange, mappings are typically specified by sets of *source-to-target dependencies (STDs)* of the form

$$\psi_\tau(\bar{x}, \bar{z}) \ :\!-\ \varphi_\sigma(\bar{x}, \bar{y}),$$

where φ_σ is a first-order (FO) formula over vocabulary σ, and ψ_τ is a conjunction of atomic τ-formulae [10, 17]. A *mapping* for us is thus a triple (σ, τ, Σ), where Σ is a set of STDs. If S is a source instance, then a target τ-instance T is called a *solution* for S under Σ if $(S, T) \models \Sigma$. More precisely, for every $\psi_\tau(\bar{x}, \bar{z}) :\!- \varphi_\sigma(\bar{x}, \bar{y})$ in Σ, we have $(S, T) \models \forall\bar{x}\forall\bar{y}\big(\varphi_\sigma(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\psi_\tau(\bar{x}, \bar{z})\big)$.

Target instances can be populated by two different kinds of elements: *constants* and *nulls*. Constants are elements that come from the source, and nulls are new elements created in targets. We assume two countably infinite disjoint domains Const and Null; elements of Const are denoted by lowercase letters, and elements of Null by $\perp$ with sub/superscripts. Source instances are interpreted as instances over Const, and targets as instances over Const$\cup$Null. We assume that we can distinguish nulls from constants (e.g., by a unary predicate testing for nulls, like `IS NULL` in SQL).

One particular solution plays a special role in data exchange: the *canonical* (universal) *solution* $\mathrm{CSOL}^{\Sigma}(S)$, for a mapping (σ, τ, Σ) and a source S [10]. As in [3, 20], it is computed as follows: for each STD $\psi(\bar{x}, \bar{z}) :\!- \varphi(\bar{x}, \bar{y})$ in Σ and for each pair of tuples $\bar{a}, \bar{b}$ such that $\varphi(\bar{a}, \bar{b})$ holds in S, create a fresh tuple of distinct nulls $\bar{\perp} = \bar{\perp}_{(\varphi,\psi,\bar{a},\bar{b})}$ (so that $|\bar{\perp}| = |\bar{z}|$) and put tuples in the target so that $\psi(\bar{a}, \bar{\perp})$, which is a conjunction of atoms, holds. If the mapping is understood from the context, we write just $\mathrm{CSOL}(S)$. The schemas σ and τ will always be clear from the context.

For example, if $\sigma = \{E\}, \tau = \{R\}$, where E and R are binary, and Σ contains $R(x, z) :\!- E(x, y)$, then for $E = \{(a, c_1), (a, c_2), (b, c_3)\}$, the canonical solution has tuples $\{(a, \perp_1), (a, \perp_2), (b, \perp_3)\}$ in R.

Databases with incomplete information

We briefly review some standard definitions [14, 16]. A database instance with incomplete information is an instance whose domain is a subset of Const $\cup$ Null. Nulls are treated as existing but unknown values. A *valuation* is a partial map v : Null $\rightarrow$ Const. Given an instance T with incomplete information, and a valuation v defined on all of its nulls, $v(T)$ stands for the instance over Const in which every null $\perp$ in T is replaced by $v(\perp)$. The semantics of T, denoted by $Rep(T)$ [16], consists of all such instances:

$$Rep(T) = \{v(T) \mid v \text{ is a valuation}\}.$$

Evaluation of queries Q on databases with nulls normally means finding *certain answers* $\Box Q(T) = \bigcap\{Q(R) \mid R \in Rep(T)\}$, i.e. tuples that belong to $Q(R)$ for all possible R in $Rep(T)$.

If Q is a positive relational algebra query, then $\Box Q(T)$ is obtained by the naive evaluation of Q on T (i.e. treating nulls as values) and then discarding tuples containing nulls [16]. For full relational algebra queries one needs a rather complicated mechanism of conditional tables [16] to represent certain answers.

Data exchange under CWA

The definitions of solutions and query answering under the CWA were given in [20]. The main idea is not to open the target to arbitrary new tuples, and instead put there just what is needed to satisfy the STDs. Solutions under the CWA must satisfy three criteria: (a) the presence of each null must be justified by the source instance and the STDs; (b) a justification for a null should not generate multiple nulls; and (c) facts true in the target instance must be justified by the source instance and the STDs.

These were formalized in [20], which showed that a target instance T is a CWA-solution iff it is a homomorphic image of $\mathrm{CSol}(S)$ and has a homomorphism back into $\mathrm{CSol}(S)$.

We now recall how (a), (b), (c) are formalized. Let (σ, τ, Σ) be a mapping, where Σ is a set of STDs $\{\psi_i(\bar{x}_i, \bar{z}_i) \;:\!-\; \varphi_i(\bar{x}_i, \bar{y}_i) \mid 1 \leq i \leq m\}$, and let S be a source instance. A *justification* for a null consists of an STD $\psi_i :\!- \varphi_i$, a tuple $(\bar{a}, \bar{b})$ so that $\varphi_i(\bar{a}, \bar{b})$ holds, and a variable among the $\bar{z}$'s. Note that justifications generate nulls in the canonical solution $\mathrm{CSol}(S)$.

Each null in a target T must have a justification for it, but the same justification should not justify different nulls. This means that there is a mapping h from justifications onto the set of nulls of T, i.e. a *homomorphism* $h : \mathrm{CSol}(S) \rightarrow T$ that maps nulls of $\mathrm{CSol}(S)$ onto the nulls of T. Such homomorphic images of $\mathrm{CSol}(S)$ were called *CWA-presolutions*. In our previous example of an STD $R(x, z) \;:\!-\; E(x, y)$ and a source $E = \{(a, c_1), (a, c_2), (b, c_3)\}$, the canonical solution has nulls $\perp_1, \perp_2, \perp_3$ given by justifications: $((a, c_1), z)$, $((a, c_2), z)$, and $((b, c_3), z)$. If we have a homomorphism $h(\perp_1) = h(\perp_2) = \perp$ and $h(\perp_3) = \perp'$, we obtain a CWA-presolution $\{(a, \perp), (b, \perp')\}$.

Requirement (c) closes instances to unjustified facts, i.e., it prohibits inventing facts based on equating nulls unless they are implied by the source and the STDs. In our example, a homomorphism h' such that $h'(\perp_1) = h'(\perp_3) = \perp$ gives us tuples $(a, \perp), (b, \perp)$ in the presolution. This says that a and b are connected to *the same* element, which is not implied by S and the STDs. Formally, a *fact* is a formula $f(\bar{a}) = \exists \bar{z}\, \gamma(\bar{a}, \bar{z})$, where $\bar{a}$ is over Const, and γ is a conjunction of τ-atoms; it is satisfied in a target instance T if there is a tuple of nulls $\bar{\perp}$ such that $\gamma(\bar{a}, \bar{\perp})$ is true. Then *CWA-solutions* are defined as CWA-presolutions T so that every fact true in T is also true in $\mathrm{CSol}(S)$. The presolution $\{(a, \perp), (b, \perp')\}$ is a CWA-solution.

The characterization of CWA-solutions leads to algorithms for finding certain answers, i.e. sets of tuples that belong to $Q(R)$ for every CWA-solution T for S and every $R \in \mathit{Rep}(T)$. Namely, they can be computed as $\Box Q(\mathrm{CSol}(S))$ [20]. If Q is a union of conjunctive queries and $\Box Q$ can be computed by the naive evaluation, this coincides with the semantics used in [10]. As we move beyond positive queries, the CWA semantics behaves nicer than the OWA semantics. For example, even in *copying* mappings, with all STDs of the form $R'(\bar{x}) \;:\!-\; R(\bar{x})$, under the semantics of [10] there are FO-queries that cannot be answered over the canonical, or other, solutions [3]. Under the CWA, certain answers coincide with $Q(\mathrm{CSol}(S))$ in such mappings.

3. Mixing OWA and CWA: mappings and solutions

We define mappings that need not follow the all-OWA or the all-CWA policy: in them, attributes of target atoms of STDs can be *annotated* as open or closed. This results in target instances in which different elements have different semantics, so we define an appropriate semantics Rep_A for them.

Annotated mappings

We shall allow each variable in the left-hand side ψ of an STD to be annotated with an element of the set $\{\mathit{op}, \mathit{cl}\}$, referring to them as *open* or *closed* variables, respectively. So formally an annotated STD is a usual STD

$$\psi(x_1, \ldots, x_n,\; z_1, \ldots, z_k) \;:\!-\; \varphi(x_1, \ldots, x_n,\; y_1, \ldots, y_m),$$

together with an annotation mapping α that assigns each occurrence of a variable in ψ either *op* or *cl*. An *annotated mapping* consists of source and target schemas σ and τ, and a set of annotated STDs. We put annotation as a superscript, writing x^{op} or x^{cl} when $\alpha(x) = \mathit{op}$ or $\alpha(x) = \mathit{cl}$, resp.

Closed annotations specify one-to-one relationships, so closed nulls behave just as nulls in CWA-solutions. Open annotations specify one-to-many relationships and exhibit the behavior of solutions of [10]. In the earlier example, according to the STD $\mathit{Submissions}(x^{cl}, z^{op}) :\!- \mathit{Papers}(x, y)$, only papers from the source are moved to the target in the exchange of data, but the *paper-author* relationship is not one-to-one, and hence multiple values are allowed in the second attribute.

Annotation in instances

Solutions under annotated mappings will be annotated instances, which we now define. A finite relation over attributes $A_1, \ldots, A_n$ with domain D is a finite set of tuples, and each tuple is a mapping $t : \{A_1, \ldots, A_n\} \rightarrow D$. An *annotated tuple* is a pair (t, α), where t is a tuple and α is a mapping $\{A_1, \ldots, A_n\} \rightarrow \{\mathit{op}, \mathit{cl}\}$. An annotated relation is a finite set of annotated tuples, and an annotated instance is a set of annotated relations. Again we use superscripts for annotations, denoting, for example, a tuple (a, b) with annotations *cl* and *op* as (a^{cl}, b^{op}).

For purely technical reasons (to deal with empty tables) we also have empty annotated tuples, denoted by $(\_, \alpha)$, where α is an annotation on the set of attributes.

If T is an annotated relation over $\mathsf{Const} \cup \mathsf{Null}$, in the semantics $\mathit{Rep}_A(T)$, after applying a valuation v to T, any tuple $(\ldots, a^{op}, \ldots)$ in $v(T)$ can be replicated arbitrarily many times with $(\ldots, b, \ldots)$, for $b \in \mathsf{Const}$. For example, $\mathit{Rep}_A(\{(a^{cl}, \perp^{op})\})$ contains all relations R whose projection on the first attribute is $\{a\}$, and $\mathit{Rep}_A(\{(a^{cl}, \perp^{cl})\})$ contains all one-tuple relations $\{(a, b)\}$ with $b \in \mathsf{Const}$.

Formally, if $T = \{(t_i, \alpha_i) \mid 1 \leq i \leq n\}$, then a relation R over Const is in $\mathit{Rep}_A(T)$ if, for some valuation v, the relation R contains the nonempty tuples among $v(t_1), \ldots, v(t_n)$, and every tuple $t \in R$ coincides with some $v(t_i)$ in all positions annotated by *closed* by α_i. Thus if α is an all-open annotation, then the tuple $(\_, \alpha)$ allows any tuple to be added to relations in $Rep_A(T)$; otherwise such tuples do not change the semantics. The difference between a tuple of *op*-annotated nulls and such $(\_, \alpha)$ is that the semantics of the latter also includes the empty table. Finally, $Rep_A(\cdot)$ extends naturally from relations to database instances.

Annotated canonical solution

Let $(\sigma, \tau, \Sigma_\alpha)$ be an annotated mapping (i.e., Σ_α is a set of annotated STDs). Let S be a source instance. The annotated canonical solution is defined by the same procedure as before, except that now it is populated with annotated tuples. That is, for each STD $\psi(\bar{x}, \bar{z})$:- $\varphi(\bar{x}, \bar{y})$, we evaluate φ over S, and for each tuple $(\bar{a}, \bar{b})$ in the result, we create a fresh tuple of nulls $\bar{\perp}$, and put annotated tuples in the solution to satisfy $\psi(\bar{a}, \bar{\perp})$, annotated as prescribed by α. If φ evaluates to the empty set over S, we add empty tuples for each atom in ψ, annotated according to α. The result is the *annotated canonical solution* denoted by $\mathrm{CSol}_{\mathrm{A}}^{\Sigma_\alpha}(S)$, or just $\mathrm{CSol}_{\mathrm{A}}(S)$, if the mapping is understood (the subscript 'A' distinguishes it from an unannotated solution).

In our previous example with $\sigma = \{E\}, \tau = \{R\}$, let the STD be $R(x^{cl}, z^{op})$:- $E(x, y)$. Then, if $E = \{(a, c_1), (a, c_2), (b, c_3)\}$, the canonical solution has annotated tuples $\{(a^{cl}, \perp_1^{op}), (a^{cl}, \perp_2^{op}), (b^{cl}, \perp_3^{op})\}$ in R.

Note that the same variable can be annotated differently in different atoms. For example, if we have an STD $R(x^{op}, z_1^{cl}) \wedge R(x^{cl}, z_2^{op})$:- $E(x, y)$ and a single tuple (a, c) in the source, then $\mathrm{CSol}_{\mathrm{A}}(S) = \{(a^{op}, \perp_1^{cl}), (a^{cl}, \perp_2^{op})\}$.

Open (resp., closed) versions of the canonical solution capture the semantics of solutions in [10] and [20]. For reasons to become clear soon, we call the solutions of [10] *OWA-solutions*: i.e., an OWA-solution for a source S under Σ is any target instance T over $\mathsf{Const} \cup \mathsf{Null}$ such that $(S, T) \models \Sigma$. We then define

$$\begin{aligned} [\![S]\!]^{\Sigma}_{\mathrm{OWA}} &= \{R \in Rep(T) \mid T \text{ an OWA-solution for } S\} \\ [\![S]\!]^{\Sigma}_{\mathrm{CWA}} &= \{R \in Rep(T) \mid T \text{ a CWA-solution for } S\} \end{aligned}$$

These semantics produce sets of relations without nulls represented by OWA and CWA-solutions, respectively.

If Σ is a set of unannotated STDs, let Σ_{op} (resp., Σ_{cl}) be the set of all Σ-STDs where each variable is annotated with *op* (resp., *cl*). The canonical solutions under these two extremes capture the semantics of the unannotated OWA- and CWA-solutions:

Lemma 1. $$\begin{aligned} [\![S]\!]^{\Sigma}_{\mathrm{OWA}} &= Rep_A(\mathrm{CSol}_{\mathrm{A}}^{\Sigma_{op}}(S)); \\ [\![S]\!]^{\Sigma}_{\mathrm{CWA}} &= Rep_A(\mathrm{CSol}_{\mathrm{A}}^{\Sigma_{cl}}(S)). \end{aligned}$$

Annotated solutions

We now define a general notion of solutions under annotated mappings using an approach similar to the CWA-solutions in Section 2, except that now we distinguish open and closed nulls. A *homomorphism* of annotated instances $h : T \to T'$ is a mapping from Null to Null so that for each annotated tuple (t, α) in a relation R in T, the tuple $(h(t), \alpha)$ is in R' – that is, homomorphisms preserve annotations (by $h(t)$ we denote the tuple obtained from t by replacing each null $\perp$ with $h(\perp)$).

Given an annotated mapping $(\sigma, \tau, \Sigma_\alpha)$ and a source S, each null in a target solution still needs to be justified by an STD ψ :- φ and a witness for φ. It is the annotation that will account for differences in the semantics: while closed nulls behave as nulls in CWA-solutions, open nulls can be instantiated by many values. Hence, we still define *presolutions* as homomorphic images of $\mathrm{CSol}_{\mathrm{A}}(S)$, since homomorphisms preserve annotations.

Our last requirement for CWA-solutions was that facts true in them must be implied by the source and the STDs, and thus true in $\mathrm{CSol}(S)$. We still want to apply this restriction, but only to closed nulls. For that, we use *annotated facts*, i.e. pairs $(f(\bar{a}), \alpha)$ where $f(\bar{a}) = \exists \bar{z}\ \gamma(\bar{a}, \bar{z})$ is a fact over the target schema, and α is an annotation over all atoms in γ. The notion of satisfaction is restricted to closed positions of T. That is, $T \models_{cl} \big(f(\bar{a}), \alpha\big)$ if there exists a tuple $\bar{\perp}$ of nulls such that for each atom $R(t)$ in $\gamma(\bar{a}, \bar{\perp})$, there is a tuple (t_0, α_0) in relation R of instance T which coincides with (t, α) in all positions annotated as *closed* in α_0.

Then a presolution T is a Σ_α*-solution* for S if each annotated fact that is true in T under $\models_{cl}$ is also true, under $\models_{cl}$, in the canonical solution $\mathrm{CSol}_{\mathrm{A}}(S)$.

If all annotations in Σ_α are *cl*, then $\models_{cl}$ is the usual notion of satisfaction, and thus Σ_α-solutions are precisely the CWA-solutions. If all annotations in Σ_α are *op*, then every fact is true under $\models_{cl}$ which means that under the OWA arbitrary facts could be true in solutions. We shall see soon that the semantics of all-open solutions is equivalent to the semantics of [10].

Example Consider an STD $R(x^{op}, z_1^{cl}) \wedge R(y^{cl}, z_2^{cl})$:- $S(x, y)$ and a source $S = \{(a, b)\}$ generating $\mathrm{CSol}_{\mathrm{A}}(S) = \{(a^{op}, \perp_1^{cl}), (b^{cl}, \perp_2^{cl})\}$. Let $R = \{(a^{op}, \perp_1^{cl}), (b^{cl}, \perp_1^{cl})\}$. The fact $\exists z\ R(a^{op}, z^{cl}) \wedge R(b^{cl}, z^{cl})$ is true in $\mathrm{CSol}_{\mathrm{A}}(S)$ with $z = \perp_1$ (under $\models_{cl}$, both atoms $R(a^{op}, \perp_1^{cl})$ and $R(b^{cl}, \perp_1^{cl})$ are satisfied by $(a^{op}, \perp_1^{cl})$), and R is a solution. $\square$

Annotated mappings: basic properties

We know that CWA-solutions have a homomorphism back into the canonical solution. A similar result is true for Σ_α-solutions, except that we need to expand the canonical solution, allowing for the open nulls to be replicated. We say that $T' \supseteq T$ is an *expansion* of T if every annotated tuple $t' \in T' - T$ coincides with some tuple $t \in T$ in all closed positions of t.

Proposition 1. *An annotated instance T is a Σ_α-solution iff it is a homomorphic image of $\mathrm{CSol}_{\mathrm{A}}(S)$, and there is a homomorphism from T to an expansion of $\mathrm{CSol}_{\mathrm{A}}(S)$.*

Similarly to the semantics $[\![\cdot]\!]_{\text{CWA}}$ and $[\![\cdot]\!]_{\text{OWA}}$, we define the semantics for arbitrary annotated mappings:

$$[\![S]\!]^{\Sigma_\alpha} \;=\; \{R \in Rep_A(T) \mid T \text{ is a } \Sigma_\alpha\text{-solution}\}.$$

If α and α' are annotations of a set Σ of STDs, we write $\alpha \preceq \alpha'$ if for each occurrence of a variable in a Σ-STD, either both α and α' annotations are *cl*, or α' annotation is *op* (i.e., closed annotations can be extended to open). The following states that changing closed annotations to open makes the semantics larger, that the extreme points are the OWA and the CWA semantics of [10] and [20], and that for every annotated mapping, $[\![S]\!]^{\Sigma_\alpha}$ is determined by the annotated canonical solution.

Theorem 1. *If Σ is a set of STDs and S is a source instance, then*

1. $[\![S]\!]^{\Sigma_{cl}} = [\![S]\!]^{\Sigma}_{\text{CWA}}$.
2. $[\![S]\!]^{\Sigma_{op}} = [\![S]\!]^{\Sigma}_{\text{OWA}}$.
3. *If $\alpha \preceq \alpha'$ then $[\![S]\!]^{\Sigma_\alpha} \subseteq [\![S]\!]^{\Sigma_{\alpha'}}$.*
4. $[\![S]\!]^{\Sigma_\alpha} = Rep_A(\text{CSOL}_{\text{A}}^{\Sigma_\alpha}(S))$.

There is a natural decision problem of recognizing instances in $[\![S]\!]^{\Sigma_\alpha}$.

Proposition 2. *The problem of checking, for source and target instances S and T, whether $T \in [\![S]\!]^{\Sigma_\alpha}$ is always in* NP*, and furthermore:*

- *it is in PTIME if all annotations in Σ_α are open;*
- *there is a mapping Σ_α in which at most one closed-annotated variable is used per atom so that the problem of checking $T \in [\![S]\!]^{\Sigma_\alpha}$ is* NP*-complete.*

Note that the complexity of recognizing instances representing tables with incomplete information normally increases with additional constraints on nulls: for example, checking if an instance R is in $Rep(S)$ is in PTIME if S is a Codd table (which cannot equate nulls), but the same problem is NP-complete for naive tables, which can equate nulls [2]. Thus, it is natural that the complexity of this particular recognition problem increases as one allows closed variables, which introduce extra constraints on nulls. But as we shall see soon, most of the time it suffices to work with the canonical solution, which can be constructed in PTIME regardless of annotation, and thus the higher complexity of $[\![\cdot]\!]^{\Sigma_\alpha}$ with closed annotations will not affect problems such as query answering.

4. Query answering

Query answering in data exchange normally means finding certain answers. Since the notion of $Q(T)$, where T is a solution, is not well-defined due to T containing nulls, we must find certain answers to Q over each solution T, and then find tuples that belong to such certain answers over all solutions T. That is, given an annotated mapping with STDs Σ_α, a source instance S and a query Q, we define

$$\mathsf{certain}_{\Sigma_\alpha}(Q,S) = \bigcap_{T \text{ is a } \Sigma_\alpha-\text{solution}} \;\; \bigcap_{R \in Rep_A(T)} Q(R)$$

We compare this with two existing notions of certain answers in data exchange. The original open-world notion $\mathsf{certain}_{\Sigma}^{\text{OWA}}(Q,S)$ of [10] and many others was defined as the set of tuples that belong to $Q(T)$ for every OWA-solution T, where Q is evaluated under the naive semantics. In [20, 15], $\mathsf{certain}_{\Sigma}^{\text{CWA}}(Q,S)$ was defined as the set of tuples in all $Q(R)$'s where R ranges over $Rep(T)$ for CWA-solutions T. Using a simple observation that in the definition of [10] it suffices to look only at instances over Const, we show:

Proposition 3. *If Σ is an arbitrary set of STDs, and Σ_{op} and Σ_{cl} are its annotations that assign op (resp., cl) to each variable, then*

$$\begin{array}{rcl}\mathsf{certain}_{\Sigma}^{\text{OWA}}(Q,S) & = & \mathsf{certain}_{\Sigma_{op}}(Q,S)\\ \mathsf{certain}_{\Sigma}^{\text{CWA}}(Q,S) & = & \mathsf{certain}_{\Sigma_{cl}}(Q,S)\end{array}$$

Furthermore, for an arbitrary annotation α,

$$\mathsf{certain}_{\Sigma_{op}}(Q,S) \subseteq \mathsf{certain}_{\Sigma_\alpha}(Q,S) \subseteq \mathsf{certain}_{\Sigma_{cl}}(Q,S).$$

Hence the semantics of [10] and [20] are indeed the two extreme semantics. For one class of queries, which was the focus of several papers on data exchange [10, 11, 3], the semantics coincide, regardless of annotations (we recall that positive relational algebra refers to the fragment of relational algebra allowing only projection, union, product and selection with positive Boolean combinations of equalities).

Proposition 4. *Let (σ, τ, Σ) be a mapping, Σ_α an arbitrary annotation of Σ, and Q a positive relational algebra query. Then*

$$\mathsf{certain}_{\Sigma_\alpha}(Q,S) \;=\; \Box Q(\text{CSOL}^{\Sigma}(S)).$$

Thus, to compute certain answers for positive queries, one can simply construct the canonical solution and apply the standard naive evaluation [16] to compute $\Box Q$ over it, as was done in [10].

We now study the general case. Our goal is to find $\mathsf{certain}_{\Sigma_\alpha}(Q,S)$ using one materialized target. We can use the annotated canonical solution as this target. Indeed, under the natural notion of certain answers in annotated instances defined as $\Box Q(T) = \bigcap\{Q(R) \mid R \in Rep_A(T)\}$, we conclude, from Theorem 1:

Corollary 1. $\mathsf{certain}_{\Sigma_\alpha}(Q,S) = \Box Q(\text{CSOL}_{\text{A}}^{\Sigma_\alpha}(S))$.

We know that $\text{CSOL}_{\text{A}}^{\Sigma_\alpha}(S)$ can be constructed in polynomial time. Thus, to describe the complexity of query answering in data exchange, we need to determine the complexity of finding $\Box Q$. We do this for relational algebra (i.e., FO) queries. Consider the problem $\text{DEQA}(\Sigma_\alpha, Q)$ of data exchange query answering for an annotated mapping $(\sigma, \tau, \Sigma_\alpha)$ and a query Q:

PROBLEM	$\text{DEQA}(\Sigma_\alpha, Q)$
INPUT:	a source database S, a tuple t
QUESTION:	is $t \in \mathsf{certain}_{\Sigma_\alpha}(Q,S)$.

Some partial answers under CWA or OWA are known [1, 10, 20]. We now classify the complexity of $\text{DEQA}(\Sigma_\alpha, Q)$ for FO queries Q using, as the main parameter, the *maximum* *number of open positions* per atom in an STD in a set of annotated STDs Σ_α. It is denoted by $\#_{op}(\Sigma_\alpha)$.

In a CWA mapping, $\#_{op}(\Sigma_\alpha) = 0$ (since there are no open positions), and in an all-open mapping, it is the maximum arity of a relation. Note that we measure the number of open annotations per atom and not per rule. For example, for the rule $T(x^{cl}, y^{op}) \wedge T(x^{cl}, z^{op})$:–φ, the value of $\#_{op}(\Sigma_\alpha)$ is 1, even though two variables occur with an open annotation.

We prove the following *trichotomy* result:

Theorem 2. *The complexity of* $\mathrm{DEQA}(\Sigma_\alpha, Q)$ *for FO queries is:*

- *coNP-complete if* $\#_{op}(\Sigma_\alpha) = 0$*;*
- *coNEXPTIME-complete if* $\#_{op}(\Sigma_\alpha) = 1$*;*
- *undecidable if* $\#_{op}(\Sigma_\alpha) > 1$.

More precisely, we prove:

1. if $\#_{op}(\Sigma_\alpha) = 0$, then $\mathrm{DEQA}(\Sigma_\alpha, Q) \in$ coNP, and there exists a mapping with $\#_{op}(\Sigma_\alpha) = 0$ and an FO query Q so that $\mathrm{DEQA}(\Sigma_\alpha, Q)$ is coNP-hard;
2. if $\#_{op}(\Sigma_\alpha) = 1$, then $\mathrm{DEQA}(\Sigma_\alpha, Q)$ is in coNEXPTIME, and there exists a mapping with $\#_{op}(\Sigma_\alpha) = 1$ and an FO query Q so that $\mathrm{DEQA}(\Sigma_\alpha, Q)$ is coNEXPTIME-hard;
3. if $k > 1$, then there is a mapping with $\#_{op}(\Sigma_\alpha) = k$ and an FO query Q so that $\mathrm{DEQA}(\Sigma_\alpha, Q)$ is undecidable.

The main result is the decidable case 2 (others are easy adaptations of known techniques [1, 2, 10, 20]). The proof is quite involved: it first uses a games argument to establish an exponential bound on the number of replicated open nulls in a possible witness for $t \notin \mathsf{certain}_{\Sigma_\alpha}(Q, S)$, and then codes a version of the tiling problem. Below, we give instead a sketch of an easier result, showing that for $\#_{op}(\Sigma_\alpha) = 1$, the query answering problem could be hard for an arbitrary level of the polynomial hierarchy (PH).

Example Suppose the source database is a graph with vertices $V(\cdot)$ and edges $E(\cdot, \cdot)$, the target schema has two binary relations, and the STDs are

$$\begin{array}{lcl} E'(x^{cl}, y^{cl}) & :- & E(x, y) \\ P(x^{cl}, z^{op}) & :- & V(x) \end{array}$$

That is, E' is a copy of the graph, and P assigns open nulls to vertices: the semantics of P is any relation whose first projection is V.

We next consider a sentence Φ_p saying that P encodes the powerset of the set of vertices (i.e., for each value a of the first attribute of P, there is a c so that $P(a, c)$ holds, and no other $P(\cdot, c)$ holds; and, for any c_1, c_2, there is a c so that $\{a \mid P(a, c)\} = \{a \mid P(a, c_1)\} \cup \{a \mid P(a, c_2)\}$ – all these are easily stated in FO). Let Ψ be an arbitrary monadic second-order sentence over E. If P encodes the powerset on V, we can easily restate Ψ as a sentence ψ over the schema $\{E', P\}$. Thus, the certain answer of $\Phi_p \rightarrow \psi$ is true iff the original graph satisfies Ψ. But it is well-known that in monadic second-order logic one can encode problems complete for all levels of PH [24] – hence query answering is hard for every level of PH. □

We now look at some special cases when we can guarantee better complexity of query answering. The hardness results for $\#_{op}(\Sigma_\alpha) = 1$ are achieved in simple mappings with all STDs either copying, i.e. $R'(\bar{x}^{cl})$:– $R(\bar{x})$, or the simplest open null introductions $U'(x^{cl}, z^{op})$:– $U(x)$. Combining several relations into one, we can also see that hardness is witnessed by a two-rule mapping of the form $R'_1(\bar{x}^{cl})$:– $R_1(\bar{x})$, $R'_2(\bar{x}^{cl}, z^{op})$:– $R_2(\bar{x})$. Thus, to achieve better complexity we should look at subclasses of queries rather than mappings.

We start with positive relational algebra queries. From Proposition 4, we obtain

Corollary 2. *If Q is a positive relational algebra query, then* $\mathrm{DEQA}(\Sigma_\alpha, Q)$ *is in PTIME.*

But adding inequalities even to conjunctive queries takes us to a larger class. Combining results of [23, 20] with properties of annotated solutions, we derive:

Proposition 5. *Let Σ be a set of STDs, α an arbitrary annotation, and Q a monotone polynomial-time query. Then* $\mathrm{DEQA}(\Sigma_\alpha, Q)$ *is in coNP. Moreover, there exists a set Σ of STDs and a conjunctive query with two inequalities Q so that* $\mathrm{DEQA}(\Sigma_\alpha, Q)$ *is coNP-complete for every annotation α.*

Finally, we describe the complexity of universal or $\forall^*\exists^*$ queries. It can also be viewed as the complexity of validating constraints in data exchange, since most commonly used integrity constraints, equality- or tuple-generating, are expressed as $\forall^*$ or $\forall^*\exists^*$ sentences.

Proposition 6. *If Q is a $\forall^*\exists^*$ query, and Σ_α is an arbitrary annotated set of STDs, then* $\mathrm{DEQA}(\Sigma_\alpha, Q)$ *is in coNP.*

5. Composing mappings

Composition and incomplete information

We now move to handling schema mappings themselves, and see how they behave under open, closed, or mixed open/closed annotations. We shall look in particular at composition of schema mappings, which is a key operation in schema evolution and model management in general [5, 6, 12, 26, 22].

We shall be dealing with schema mappings used in data exchange, i.e. triples (σ, τ, Σ). Since semantically a mapping is a binary relation consisting of pairs (S, T), where S and T are source and target instances satisfying Σ, [12] made a very natural proposal to use the composition of such relations to define the composition of mappings. One more condition, however, is required. Note that a pair (S, W) is in the composition of the binary relations given by the mappings (σ, τ, Σ) and (τ, ω, Δ) iff there is an instance T such that T is a solution for S (under Σ) and W is a solution for

T (under Δ). But while solutions as defined in [10, 20] and here are instances over $\mathsf{Const} \cup \mathsf{Null}$, we do not have a definition of a solution for a source instance with nulls. Indeed, doing so would require evaluating universal constraints over instances with nulls, something that has long been known to be problematic [4, 16, 19].

So the definition of composition of [12] and others is restricted to instances over Const: a pair (S, W) of instances over Const belongs to the composition iff there is a solution T for S over Const, and W is a solution for T. But recall that, under the definition of a solution of [10], T that only uses elements of Const is a solution for S iff $T \in Rep(T')$ for some OWA-solution T' – equivalently, iff $T \in [\![S]\!]^{\Sigma}_{\text{OWA}}$. Thus, the semantics of a schema mapping used in [12] is

$$(\!|\Sigma|\!)^{\text{OWA}}_{\sigma,\tau} = \{(S,T) \mid T \in [\![S]\!]^{\Sigma}_{\text{OWA}}\},$$

where S and T range over σ- and τ-instances. Hence, their notion of composition is $(\!|\Sigma|\!)^{\text{OWA}}_{\sigma,\tau} \circ (\!|\Delta|\!)^{\text{OWA}}_{\tau,\omega}$. (We use slightly different brackets to denote the semantics of a schema mapping as opposed to the semantics of solutions.) In general, for an *annotated* mapping $(\sigma, \tau, \Sigma_\alpha)$, we define its semantics as

$$(\!|\Sigma_\alpha|\!)_{\sigma,\tau} = \{(S,T) \mid T \in [\![S]\!]^{\Sigma_\alpha}\},$$

and the composition of two annotated mappings $(\sigma, \tau, \Sigma_\alpha)$ and $(\tau, \omega, \Delta_{\alpha'})$ as

$$\Sigma_\alpha \circ \Delta_{\alpha'} \overset{\text{def}}{=} (\!|\Sigma_\alpha|\!)_{\sigma,\tau} \circ (\!|\Delta_{\alpha'}|\!)_{\tau,\omega}.$$

We omit the schemas from our notation $\Sigma_\alpha \circ \Delta_{\alpha'}$ as they will always be clear from the context.

Notice that if both α and α' are all-open annotations, then this is the definition of composition of [12].

We now deal with two basic problems related to schema mappings and their composition: their complexity, and syntactic representations (i.e., what is a class of constraints that captures $\Sigma_\alpha \circ \Delta_{\alpha'}$?).

For several results, the class of queries used as source formulae φ_σ in STDs will be important. In [11, 10, 12] only conjunctive queries are allowed in STDs; so far, as in [3, 20], we allowed arbitrary FO queries. We refer to STDs $\psi_\tau(\bar{x}, \bar{z}) \text{ :- } \varphi_\sigma(\bar{x}, \bar{y})$ as *CQ-STDs* or *monotone STDs* if φ_σ is a conjunctive (resp., monotone) query. Otherwise it is assumed to be an FO query, as before.

Complexity of composition

Let $(\sigma, \tau, \Sigma_\alpha)$ and $(\tau, \omega, \Delta_{\alpha'})$ be two annotated mappings. We consider the following *composition problem*:

PROBLEM	$\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$
INPUT:	a σ-database S, a ω-database W
QUESTION:	is (S, W) in $\Sigma_\alpha \circ \Delta_{\alpha'}$?

The all-open version $\text{COMP}(\Sigma_{op}, \Delta_{op})$ with CQ-STDs was shown to be NP-complete in [12]. We first extend this to more general annotations.

Lemma 2. *If Δ contains only monotone STDs, Σ is arbitrary, and α is any annotation of Σ, then*

$$\Sigma_\alpha \circ \Delta_{op} = \Sigma_{op} \circ \Delta_{op}.$$

Corollary 3. *If Δ contains only monotone STDs, then $\text{COMP}(\Sigma_\alpha, \Delta_{op})$ is in NP for every Σ_α. Moreover, there exist mappings with CQ-STDs Σ and Δ so that for an arbitrary annotation α of Σ, the problem $\text{COMP}(\Sigma_\alpha, \Delta_{op})$ is NP-complete.*

We now look at arbitrary FO-STDs. The complexity of the composition problem, as the complexity of query answering, is classified by the parameter $\#_{op}(\Sigma_\alpha)$, the maximum number of open positions per atom in an STD in Σ_α. The classification is another trichotomy:

Theorem 3. *The complexity of $\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$ is*

- *NP-complete if $\#_{op}(\Sigma_\alpha) = 0$;*
- *NEXPTIME-complete if $\#_{op}(\Sigma_\alpha) = 1$;*
- *undecidable if $\#_{op}(\Sigma_\alpha) > 1$.*

More precisely, we prove the following. If (σ, τ, Σ) and (τ, ω, Δ) are two schema mappings with arbitrary FO-STDs, and α and α' are annotations of Σ and Δ, then:

- If $\#_{op}(\Sigma_\alpha) = 0$ (i.e., α is the all-closed annotation), then $\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$ is in NP. Moreover, there exist Σ and Δ, using only CQ-STDs, so that $\text{COMP}(\Sigma_{cl}, \Delta_{\alpha'})$ is NP-hard for every annotation α'.
- If $\#_{op}(\Sigma_\alpha) = 1$, then $\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$ is in NEXPTIME. Moreover, there exist Σ_α with $\#_{op}(\Sigma_\alpha) = 1$ and Δ so that $\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$ is NEXPTIME-hard for every annotation α'.
- For each $k > 1$, there exist Σ_α with $\#_{op}(\Sigma_\alpha) = k$ and Δ so that $\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$ is undecidable for every annotation α'.

Thus, if both Σ and Δ contain only CQ-STDs, the composition problem is in NP if Σ has an all-closed annotation, or Δ has an all-open annotation. Moreover, composing Σ_{cl} with any $\Delta_{\alpha'}$ (even if both have FO-STDs) matches the complexity of OWA-composition achieved only for CQ-STDs. For more open nulls, our results suggest that one needs to restrict STDs to monotone to keep the complexity reasonable.

The results on the complexity of the composition problem $\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$ presented in this section are summarized in the table below:

Σ_α \ $\Delta_{\alpha'}$	arbitrary	$\alpha' = op$ and monotone STDs
$\#_{op} = 0$	NP-complete	NP-complete
$\#_{op} = 1$	NEXPTIME-complete	
$\#_{op} > 1$	undecidable	

Table 1: Complexity of $\text{COMP}(\Sigma_\alpha, \Delta_{\alpha'})$

Syntactic descriptions of composition

As was noticed in [12], under the OWA, schema mapping composition cannot be captured syntactically without increasing the class of STDs: there exist Σ and Δ (with CQ-STDs only) such that one cannot find Γ with FO-STDs satisfying $(\!|\Gamma|\!)^{\mathrm{OWA}} = (\!|\Sigma|\!)^{\mathrm{OWA}} \circ (\!|\Delta|\!)^{\mathrm{OWA}}$. One can see this by a complexity gap argument: OWA-schema mappings have low (AC^0) complexity, but their composition could be NP-hard [12]. Arbitrarily annotated mappings could be of higher complexity, but we can still show the following strong failure of closure under composition without any additional assumptions.

Proposition 7. *There exist schema mappings with CQ-STDs Σ and Δ such that, given their arbitrary annotations α and α', there is no annotated mapping $\Gamma_{\alpha''}$ with FO-STDs that satisfies $(\!|\Gamma_{\alpha''}|\!) = \Sigma_\alpha \circ \Delta_{\alpha'}$.*

So we need to extend the class of mappings to make it closed under composition. We say that a class of mappings $\mathcal{C}$ with a semantics $(\!| \cdot |\!)^{\mathcal{C}}$ is *closed under composition* if for every two mappings $(\sigma, \tau, M_{\sigma\tau})$ and $(\tau, \omega, M_{\tau\omega})$ from $\mathcal{C}$, there exists another mapping $(\sigma, \omega, M_{\sigma\omega})$ from $\mathcal{C}$ so that

$$(\!|M_{\sigma\omega}|\!)^{\mathcal{C}} = (\!|M_{\sigma\tau}|\!)^{\mathcal{C}} \circ (\!|M_{\tau\omega}|\!)^{\mathcal{C}}.$$

In [12], such a class was found under the OWA: it was based on Skolemized CQ-STDs[1]. We now define such Skolemized STDs in an annotated setting, and prove a composition lemma for them that gives us two classes of annotated mappings closed under composition: the class of [12] and its closed-world analog.

Assume that we have a countable collection $\mathcal{F}$ of function symbols. Given two schemas σ and τ, an *annotated Skolemized STD*, or an annotated *SkSTD*, over them is an expression of the form:

$$\psi_\tau(u_1, \ldots, u_k) \;{:}\!-\; \varphi_\sigma(x_1, \ldots, x_n),$$

together with an annotation α of ψ_τ where

- φ_σ is an FO formula over $\sigma \cup \mathcal{F}$ whose atomic subformulae are either $R(\bar{z})$, where $\bar{z}$ are variables, or $y = f(\bar{z})$, where y is a variable;
- ψ_τ is a conjunction of atomic τ formulae; and
- each u_i is either one of the x_j's, or $f(\bar{z})$, for some $f \in \mathcal{F}$ and $|arity(f)|$ variables $\bar{z}$ among $\bar{x}$.

Annotations are defined as before, i.e. by assigning *op* or *cl* to each position in each atom in ψ_τ.

For example, if our source has tuples (*em,proj*) of employee names and projects and we want to create a target with tuples (*empl_id,em,phone*) that invents ids and phones of employees, we can capture this by an annotated SkSTD

$$T(f(em)^{cl},\ em^{cl},\ g(em,proj)^{op}) \;{:}\!-\; S(em,proj) \tag{3}$$

[1]Such STDs were called second-order in [12] because their semantics was defined by existentially quantifying over the Skolem functions. We prefer to call them Skolemized STDs, and use CQ, FO, etc in STDs to restrict the class of formulae φ in $\psi \;{:}\!-\; \varphi$.

indicating that one id is created for each name, with f being the function from names to ids. Using a null instead of $f(em)$ would have generated a new null for each (*em,proj*) pair, rather than just the name. The *phone* attribute is open, allowing employees to have multiple phones.

Next, we extend the definition of the semantics to annotated mappings $(\sigma, \tau, \Sigma_\alpha)$ with SkSTDs. Let S be a source instance. Let $F = \{f_1, \ldots, f_r\}$ be the set of function symbols used in Σ_α, and for each m-ary f_i, let f'_i be a function from Const^m to Const. For this set $F' = \{f'_1, \ldots, f'_r\}$, we construct a solution $\mathrm{SOL}_{F'}^{\Sigma_\alpha}(S)$ as follows: compute the result of φ_σ in S, with functions interpreted as F', and for each tuple $\bar{a}$ in it, put annotated tuples in the target to satisfy $\psi_\tau(\bar{u}')$, where, if $u_i = x_j$, then $u'_i = a_j$, and if $u_i = f(\bar{x})$ then $u'_i = f'(\bar{a})$. The annotation is the same as in Σ_α. If φ_σ evaluates to the empty set, then, as before, empty annotated tuples are added.

In our example (3), if $S = \{(John, P1)\}$ and $f'(John) = 001$ and $g'(John, P1) = 1234$, then $\mathrm{SOL}_{\{f',g'\}}(S)$ has one tuple $(001^{cl}, John^{cl}, 1234^{op})$.

For Σ_α with SkSTDs, the semantics of S is given by

$$[\![S]\!]^{\Sigma_\alpha} = \bigcup_{F'} Rep_A\big(\mathrm{SOL}_{F'}(S)\big),$$

as F' ranges over functions from Const to Const that match the arity of functions in F. Note that as $\mathrm{SOL}_{F'}(S)$ has no nulls, the only effect of applying Rep_A to it is adding tuples that coincide with some tuple t in $\mathrm{SOL}_{F'}(S)$ on all attributes annotated *cl*. For example, $[\![S]\!]^{\Sigma_\alpha}$ for the SkSTD (3) and the above source will contain a relation $\{(001, John, 1234), (001, John, 5678)\}$.

Then, finally, we define

$$(\!|\Sigma_\alpha|\!) = \{(S, T) \mid T \in [\![S]\!]^{\Sigma_\alpha}\}.$$

First observe that if Σ is a set of un-annotated SkSTDs, then $(\!|\Sigma_{op}|\!)$ is precisely the semantics of [12] — that is, [12] used the open-world semantics (a formal proof is given in the appendix).

Second, even though mappings with SkSTDs do not explicitly allow null values, semantically they extend the usual STD-based mappings:

Lemma 3. *For every annotated mapping $(\sigma, \tau, \Sigma_\alpha)$ based on STDs there exists an equivalent mapping $(\sigma, \tau, \Gamma_\alpha)$ with the same annotations based on SkSTDs, i.e. $(\!|\Sigma_\alpha|\!) = (\!|\Gamma_\alpha|\!)$. Furthermore, the right-hand sides of STDs in Σ and in Γ are the same.*

We now state the main technical lemma which shows when two annotated mappings can be composed.

Lemma 4. *Let Σ_α and $\Delta_{\alpha'}$ be two schema mappings with annotated SkSTDs such that either*

- *the annotation of $\Delta_{\alpha'}$ is all-open, and it only has monotone queries in its SkSTDs; or*
- *the annotation of Σ_α is all-closed.*

Then one can construct a composition mapping $\Gamma_{\alpha'}$ (i.e. $(\!|\Gamma_{\alpha'}|\!) = (\!|\Sigma_\alpha|\!) \circ (\!|\Delta_{\alpha'}|\!)$) with annotated SkSTDs such that

- *the left-hand sides and annotations of SkSTDs in $\Gamma_{\alpha'}$ and $\Delta_{\alpha'}$ are the same;*
- *the right-hand sides of SkSTDs in $\Gamma_{\alpha'}$ are CQs if the same is true for Σ_α and $\Delta_{\alpha'}$.*

As a corollary, we have our main composition result:

Theorem 4. *The following two classes of schema mappings given by annotated SkSTDs are closed under composition:*

1. *mappings with all-open annotations in which only conjunctive queries are used in SkSTDs; and*
2. *mappings with all-closed annotations in which arbitrary FO queries are used in SkSTDs.*

The first case of course is that of [12]. Theorem 4 says that we can also achieve compositionality under the CWA with more general queries used in mappings.

6. Conclusions

Two previous approaches to data exchange have been based either on the OWA, or on the CWA, and both had their limitations. We have shown that, using an old idea of allowing both open and closed null values, we obtain mappings that can mix OWA and CWA in an arbitrary manner. We looked at query evaluation and composition of mappings, proved two classification results for their complexity, established criteria for schema compositionality, and showed particularly nice behaviour of positive queries in mixed contexts.

Several extensions of our results can be obtained. We mention three. The first trichotomy theorem is true for any query language of PTIME data complexity that contains FO. Second, if we allow 1-to-m relationships in place of 1-to-many relationships and define such limited open nulls (i.e. each such null can be replicated at most m times), then all the complexity results about CWA mappings apply to this case. Third, if a mapping Δ has only existential queries, then every composition $\Sigma_\alpha \circ \Delta_{\alpha'}$ is in NP, regardless of annotations.

A few open problems remain. The next step is extending results to cover mappings with target constraints, as was done in [15]. It is likely that adding weakly acyclic constraints [10, 9] would lead to a terminating chase as in both open-world [10] and closed-world [15] cases. We also would like to see if the mixed open/closed mappings are applicable in more general frameworks that try to unify data exchange, integration, and peer-to-peer scenarios, such as in [8].

Acknowledgments The authors were supported by the EPSRC grant E005039; the first author also by the EC Marie Curie Excellence grant MEXC-CT-2005-024502.

7. References

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

[2] S. Abiteboul, P. Kanellakis, G. Grahne. On the representation and querying of sets of possible worlds. *TCS* 78 (1991), 158–187.

[3] M. Arenas, P. Barceló, R. Fagin, L. Libkin. Locally consistent transformations and query answering in data exchange. In *PODS 2004*, pages 229–240.

[4] P. Atzeni, N. Morfuni. Functional dependencies and constraints on null values in database relations. *Information and Control* 70(1): 1–31 (1986).

[5] P. Bernstein, T.Green, S. Melnik, A. Nash. Implementing mapping composition. *VLDB'06*, pages 55–66.

[6] P. Bernstein, S. Melnik. Model management 2.0: manipulating richer mappings. *SIGMOD'07*, pages 1–12.

[7] L. Chiticariu, W.-C. Tan. Debugging schema mappings with routes. In *VLDB'06*, pages 79–90.

[8] G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. On reconciling data exchange, data integration, and peer data management. In *PODS'07*, pages 133–142.

[9] A. Deutsch, V. Tannen. Reformulation of XML queries and constraints. In *ICDT'03*, pages 225–241.

[10] R. Fagin, Ph. Kolaitis, R. Miller, L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336(1): 89–124 (2005).

[11] R. Fagin, Ph. Kolaitis, L. Popa. Data exchange: getting to the core. *ACM TODS* 30(1): 174–210 (2005).

[12] R. Fagin, Ph. Kolaitis, L. Popa, W.C. Tan. Composing schema mappings: second-order dependencies to the rescue. *ACM TODS* 30(4) 994–1055 (2005).

[13] G. Gottlob, R. Zicari. Closed world databases opened through null values. In *VLDB'88*, pages 50–61.

[14] G. Grahne. *The Problem of Incomplete Information in Relational Databases*. Springer, 1991.

[15] A. Hernich, N. Schweikardt. CWA-solutions for data exchange settings with target dependencies. In *PODS'07*, pages 113–122.

[16] T. Imielinski, W. Lipski. Incomplete information in relational databases. *J. ACM* 31 (1984), 761–791.

[17] Ph. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS 2005*.

[18] M. Lenzerini. Data integration: a theoretical perspective. In *PODS'02*, pages 233–246.

[19] M. Levene, G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science* 206 (1998), 283–300.

[20] L. Libkin. Data exchange and incomplete information. In *PODS'06*, pages 60–69.

[21] W. Lipski. On semantic issues connected with incomplete information in databases. *ACM Trans. Database Systems* 4 (1979), 262–296.

[22] J. Madhavan, A. Halevy. Composing mappings among data sources. In *VLDB'03*, pages 572–583.

[23] A. Madry. Data exchange: on the complexity of answering queries with inequalities. *IPL* 94 (2005) 253–257.

[24] J. Makowsky and Y. Pnueli. Arity and alternation in second-order logic. *APAL*, 78 (1996), 189–202.

[25] R. Miller, M. Hernandez, L. Haas, L. Yan, C. Ho, R. Fagin, L. Popa. The Clio project: managing heterogeneity. *SIGMOD Record* 30 (2001), 78–83.

[26] A. Nash, P. Bernstein, S. Melnik. Composition of mappings given by embedded dependencies. *ACM TODS* 32(1): 4 (2007).

[27] L. Popa, Y. Velegrakis, R. Miller, M. Hernández, R. Fagin. Translating web data. In *VLDB 2002*, pages 598–609.

[28] R. Reiter. On closed world databases. In *Logic and Databases*, Plenum Press, 1978, pages 55–76.