The Absolute Consistency Problem for Relational Schema Mappings with Functional Dependencies**

Yasunori ISHIHARA†, Member, Takashi HAYATA††, Nonmember, and Toru FUJIWARA††, Fellow

SUMMARY This paper discusses a static analysis problem, called absolute consistency problem, for relational schema mappings. A given schema mapping is said to be absolutely consistent if every source instance has a corresponding target instance. Absolute consistency is an important property because it guarantees that data exchange never fails for any source instance. Originally, for XML schema mappings, the absolute consistency problem was defined and its complexity was investigated by Amano et al. However, as far as the authors know, there are no known results for relational schema mappings. In this paper, we focus on relational schema mappings such that both the source and the target schemas have functional dependencies, under the assumption that mapping rules are defined by constant-free tuple-generating dependencies. In this setting, we show that the absolute consistency problem is in coNP. We also show that it is solvable in polynomial time if the tuple-generating dependencies are full and the size of the left-hand side of each functional dependency is bounded by some constant. Finally, we show that the absolute consistency problem is coNP-hard even if the source schema has no functional dependency and the target schema has only one; or each of the source and the target schemas has only one functional dependency such that the size of the left-hand side of the functional dependency is at most two.

key words: data exchange, relational schema mapping, absolute consistency, functional dependency

1. Introduction

Exchanging and sharing differently represented data is old but still challenging problems. Schema mappings are widely accepted as a formal framework for not only discussing the theoretical aspects of data exchange but also specifying the behavior of a data-sharing system (e.g., [4]). A schema mapping \( M \) is a triple of a source schema (possibly with source constraints), a target schema (possibly with target constraints), and a set of dependencies between the source and the target schemas. \( M \) represents a set of pairs of source and target instances that satisfy all the dependencies.

So far, many static analysis problems of schema mappings have been investigated. For example, Arenas and Libkin [3] introduced the consistency problem for XML schema mappings, where a schema mapping is consistent if some source instance has a corresponding target instance. Amano et al. [4] introduced the absolute consistency problem for XML schema mappings, where a schema mapping is absolutely consistent if every source instance has a corresponding target instance. Absolute consistency is a much more desirable property than consistency because it guarantees that data exchange never fails for any source instance. Amano et al. [4] also presented two subclasses of XML schema mappings for which absolute consistency is decidable. Bojańczyk et al. [5] showed that the absolute consistency problem is \( \Pi_2 \text{EXP} \)-complete in general and \( \Pi_4 \text{NP} \)-complete if the schemas only admit XML documents of bounded height. Some tractability results on consistency and absolute consistency problems between restricted schemas were reported in our work [6], [7]. Boneva et al. [8] tackled the absolute consistency problem (called just “consistency” in the paper) for relation-to-RDF schema mappings. However, as far as the authors know, there are no known results on the absolute consistency problem for relational schema mappings.

This paper addresses the absolute consistency problem for relational schema mappings such that both of the source and the target schemas have functional dependencies, under the assumption that mapping rules are defined by constant-free tuple-generating dependencies. Constant-freeness brings us the following two important properties for deciding the absolute consistency problem: The first one is that we only have to consider schema mappings with just one tuple-generating dependency. The second one is that the relationship between variables appearing in the tuple-generating dependencies can be simply captured as a fixed point of a function \( \mathcal{E} \), which is essentially the same as the chase algorithm [9]. Using these properties, we show that the absolute consistency is in coNP in general, and it is solvable in polynomial time if the tuple-generating dependencies are full and the size of the left-hand side of each functional dependency is bounded by some constant. These results are obtained by extending our previous work [1] on graph databases with the correction of an error. In [1], we overlooked the case where a target instance is obtained by applying a tuple-generating dependency more than once. In this paper, we resolve this error by introducing a copy of a tuple-generating dependency, and then combining the computation of fixpoints of the function \( \mathcal{E} \). Finally, we show that the absolute consistency problem is coNP-hard even if the source schema has no functional dependency and the target schema has only one; or each of the source and the target

*Copyright © 2020 The Institute of Electronics, Information, and Communication Engineers*
schemas has only one functional dependency such that the size of the left-hand side of the functional dependency is at most two. The coNP-hardness is shown by reducing the unsatisfiability problem for CNF formulas to the absolute consistency problem.

The rest of this paper is organized as follows. Section 2 gives preliminary definitions. The two important properties for deciding the absolute consistency problem are provided in Sect. 3. In Sect. 4, we provide two results on the upper bound of the complexity of the problem. Then, in Sect. 5 we prove the coNP-hardness of the problem. In Sect. 6 we mention other static analysis problems for relational schema mappings. Finally, Sect. 7 summarizes the paper.

2. Definitions

2.1 Relational Databases

A relation schema \( R[U] \) is a pair of a relation name \( R \) and a finite set \( U \) of attributes. Fix a countable set \( \text{Dom} \) of values.

A tuple \( t \) over \( R[U] \) is a total function from \( U \) to \( \text{Dom} \). Let \( t.A \) denote the value of \( A \in U \) in \( t \). A relation instance \( I \) over \( R[U] \) is a finite set of tuples over \( R[U] \). A database schema \( R \) is a finite sequence of relation schemas. A database instance \( I \) over a database schema \( R = (R_1[U_1], \ldots, R_n[U_n]) \) is a sequence \( (I_1, \ldots, I_n) \), where each \( I_i \) is a relation instance of \( R_i[U_i] \).

For a tuple \( t \) over \( R[U] \) and \( X \subseteq U \), \( t[X] \) denotes the function \( t \) whose domain is restricted to \( X \).

**Definition 1** (Functional dependencies): A functional dependency (fd for short) over a relation schema \( R[U] \) is an expression of the form \( X \rightarrow Y \), where \( X, Y \subseteq U \). A relation instance \( I \) over \( R[U] \) satisfies \( X \rightarrow Y \), denoted \( I \models X \rightarrow Y \), if \( t[X] = t'[X] \) implies \( t[Y] = t'[Y] \) for all \( t, t' \in I \).

As usual, we write \( A_1 \cdots A_k \) to mean the set \( \{A_1, \ldots, A_k\} \) of attributes, and we often assume that every fd is in the form of \( A_1 \cdots A_k \rightarrow B \), i.e., the right-hand side is a singleton.

2.2 Schema Mappings

Schema mappings represent the correspondence between source instances and target instances. First, we define a constant-free fragment of conjunctive queries for specifying schema mappings.

Let \( R = (R_1[U_1], \ldots, R_n[U_n]) \) be a database schema. Fix a countable set \( \text{Var} \) of variables. A formula \( \phi \) over \( R \) is defined as follows:

- \( \phi = R(x_1, \ldots, x_{|U|}) \) is a formula, where each \( x_j \) is in \( \text{Var} \).
- \( \phi = \varphi_1 \land \varphi_2 \) is a formula if \( \varphi_1 \) and \( \varphi_2 \) are formulas.

Hereafter, we often annotate a variable \( x \) in a formula by an attribute \( A \), denoted \( x@A \), to explicitly represent that \( x \) is a variable for attribute \( A \).

We define the semantics of formulas. Let \( \mathcal{V}(\phi) \) denote the set of variables appearing in \( \phi \). Let \( g : \mathcal{V}(\phi) \rightarrow \text{Dom} \) be a variable assignment. Let \( I = \langle I_1, \ldots, I_n \rangle \) be a database instance.

- \( (I, g) \models R(x_1@A_1, \ldots, x_{|U|}@A_{|U|}) \) if \( I_i \) is a relation instance of \( R_i[U_i] \) containing a tuple \( t \) such that \( t.A_j = g(x_j) \) for all \( j \) (1 \( \leq \) \( j \leq |U| \)).
- \( (I, g) \models \varphi_1 \land \varphi_2 \) if \( (I, g) \models \varphi_1 \) and \( (I, g) \models \varphi_2 \).

If \( (I, g) \models \varphi \), we say \( (I, g) \) satisfies \( \varphi \) (or simply, \( I \) satisfies \( \varphi \)).

**Definition 2** (Schema mappings): A schema mapping \( M \) is a tuple \( (R_5, \Sigma_5), (R_T, \Sigma_T), \Gamma \), where

- \( R_5 \) and \( R_T \) are the source and the target schemas, respectively.
- \( \Sigma_5 \) and \( \Sigma_T \) are sets of functional dependencies over \( R_5 \) and \( R_T \), respectively, and
- \( \Gamma \) is a finite set of tuple-generating dependencies (tgds) for short of the form \( \varphi_S \rightarrow \varphi_T \), where \( \varphi_S \) is a formula over \( R_5 \) and \( \varphi_T \) is a formula over \( R_T \).

As usual, variables in \( \mathcal{V}(\varphi_S) \) are interpreted as universally quantified, and variables only in \( \mathcal{V}(\varphi_T) \) as existentially quantified. Definition 3 below formalizes this intuition. We say that a tgd \( \varphi_S \rightarrow \varphi_T \) is full if \( \mathcal{V}(\varphi_T) \subseteq \mathcal{V}(\varphi_S) \) (i.e., there is no “existentially-quantified” variable).

**Definition 3** (Solutions): Let \( M = ((R_5, \Sigma_5), (R_T, \Sigma_T), \Gamma) \) be a schema mapping. Let \( I_5 \) be an instance over \( R_5 \) satisfying \( \Sigma_5 \), and \( I_T \) an instance over \( R_T \) satisfying \( \Sigma_T \). A pair \( (I_5, I_T) \) satisfies \( M \) if the following condition holds: For each \( (\varphi_S \rightarrow \varphi_T) \in \Gamma \) and for any \( g_S : \mathcal{V}(\varphi_S) \rightarrow \text{Dom} \), there exists \( g_T : \mathcal{V}(\varphi_T) \rightarrow \text{Dom} \) such that

1. \( g_T \) is compatible with \( g_S \), i.e., \( g_T(x) = g_S(x) \) for each variable \( x \in \mathcal{V}(\varphi_S) \cap \mathcal{V}(\varphi_T) \); and
2. if \( (I_5, g_S) \models \varphi_S \), then \( (I_T, g_T) \models \varphi_T \).

If a pair \( (I_5, I_T) \) satisfies \( M \), \( I_T \) is called a solution for \( I_5 \) under \( M \). Let \( \text{Sol}_M(I_5) \) denote the set of solutions for \( I_5 \) under \( M \).

Note that in Definition 3, we consider \( g_S \) and \( g_T \) for independently each tgd in \( \Gamma \), not over simultaneously all tgds in \( \Gamma \), and hence, variables in different tgds are independent even if they are the same variable. Without loss of generality, we can assume that no variables are shared by different tgds.

**Definition 4** (Absolute consistency): A schema mapping \( M = ((R_5, \Sigma_5), (R_T, \Sigma_T), \Gamma) \) is absolutely consistent if \( \text{Sol}_M(I_5) \neq \emptyset \) for every instance \( I_5 \) over \( R_5 \) satisfying \( \Sigma_5 \).

**Example 1**: Consider the following schema mapping \( M = ((R_5, \Sigma_5), (R_T, \Sigma_T), \Gamma) \):

- \( R_5 = \langle \text{EmpMac}[\text{EM}], \text{EmpRoom}[\text{ER}] \rangle \),
- \( \Sigma_5 = \{ E \rightarrow M \text{ over EmpMac}, E \rightarrow R \text{ over EmpRoom} \} \),
- \( R_T = \langle \text{MacRoom}[\text{MR}], \text{Office}[\text{ER}] \rangle \),
- \( \Sigma_T = \emptyset \),
- \( \Gamma \) consists of the following three tgds:

\[ \text{EmpRoom}(x@E, z@R) \rightarrow \text{Office}(x@E, z@R), \]
EmpMac(x@E, y@M) ∧ EmpRoom(x@E, z@R) 
→ MacRoom(y@M, z@R),
EmpMac(x@E, y@M) ∧ EmpRoom(x@E, z@R) ∧ EmpMac(x’@E, y@M) 
→ Office(x’@E, z@R).

Obviously, M is absolutely consistent because Σ_R = ∅.

Next, consider a schema mapping M’ that is equal to M except that Σ_R = \{E → R over Office\}. Then, M’ is not absolutely consistent. To see this, consider the following source instance I_S, where Alice and Bob are in different rooms but use the same machine (e.g., at least one of them uses the machine remotely):

| EmpMac | EmpRoom |
|--------|---------|
| E(mployee) M(machine) | E(mployee) R(room) |
| Alice | PC-007 |
| Bob | PC-007 |
| Carol | PC-025 |
| David | PC-025 |

Any solution for I_S under M contains the following subinstance I_T:

| MacRoom | Office |
|---------|--------|
| M(machine) R(room) | E(mployee) R(room) |
| PC-007 | R101 |
| PC-007 | R202 |
| PC-025 | R303 |
| Alice | R101 |
| Bob | R202 |
| Carol | R303 |
| David | R303 |

Hence, Σ_R is never satisfied, so I_S has no solution under M’. Intuitively, the last tgd may generate tuples that violate E → R over Office unless the instance of MacRoom satisfies M → R, although this tgd is useful for completion of missing tuples in EmpRoom (such as David’s entry).

The size |M| of a schema mapping M = ((R_S, Σ_S), (R_T, Σ_T), Γ) is the sum of the sizes of R_S, Σ_S, R_T, Σ_T, and Γ. The size of a database schema R is the sum of the numbers of attributes in R. The size of an fd X → Y is the sum of the numbers of the attributes in X and Y, and the size of the set Σ of fds is the sum of the sizes of fds in Σ. The size of a tgd is the number of variables used in the tgd, and the size of the set Γ of tgds is the sum of the sizes of the tgds in Γ.

3. Basic Properties

In this section, we introduce two important properties for deciding the absolute consistency problem. The first property states that we only have to consider schema mappings with just one tgd. The second is a property of a function E_φ, which is a nice tool for deciding whether, given φ, Σ, and g : \mathcal{V}(φ) → Dom, there exists an instance I such that (I, g) \models φ and I \models Σ.

3.1 Reduction to a Single Tgd

The next lemma shows that any schema mapping M can be transformed into M’ with a single tgd, preserving the property whether it is absolutely consistent or not. Note that M’ is not equivalent to M in general, in the sense that Sol_M’ is not necessarily equal to Sol_M.

Lemma 1: Let M = ((R_S, Σ_S), (R_T, Σ_T), Γ) be a schema mapping, where Γ = {φ_{S_i} → ψ_{T_i} | 1 ≤ i ≤ ℓ}. Define

Γ’ = \left\{ \bigwedge_{i=1}^{ℓ} φ_{S_i} \right\} \cap \left\{ \bigwedge_{i=1}^{ℓ} ψ_{T_i} \right\}.

M is absolutely consistent if and only if M’ = ((R_S, Σ_S), (R_T, Σ_T), Γ’) is absolutely consistent.

Proof. (Only if part.) Suppose that M’ is not absolutely consistent. Let I_S be a source instance that satisfies Σ_S and has no solution under M’. Then, for any target instance I_T satisfying Σ_T, there is some g_S : \mathcal{V}(\bigwedge_{i=1}^{ℓ} φ_{S_i}) → Dom such that

• (I_S, g_S) \models \bigwedge_{i=1}^{ℓ} φ_{S_i} and
• (I_T, g_T) \not\models \bigwedge_{i=1}^{ℓ} ψ_{T_i} for any variable assignment g_T : \mathcal{V}(\bigwedge_{i=1}^{ℓ} ψ_{T_i}) → Dom compatible with g_S.

Since no variables are shared by different ψ_{T_i}, for any target instance I_T satisfying Σ_T there are some tgd (φ_{S_k} → ψ_{T_k}) ∈ Γ (1 ≤ k ≤ ℓ) and variable assignment g_{S_k} : \mathcal{V}(φ_{S_k}) → Dom such that

• (I_S, g_{S_k}) \models φ_{S_k} and
• (I_T, g_{T_k}) \not\models ψ_{T_k} for any variable assignment g_{T_k} : \mathcal{V}(ψ_{T_k}) → Dom compatible with g_{S_k}.

This means that I_S has no solution under M.

(If part.) Suppose that M is not absolutely consistent.

Let I_S be a source instance that satisfies Σ_S and has no solution under M. Then, for any target instance I_T satisfying Σ_T, there are some tgd (φ_{S_k} → ψ_{T_k}) ∈ Γ (1 ≤ k ≤ ℓ) and variable assignment g_{S_k} : \mathcal{V}(φ_{S_k}) → Dom such that

• (I_S, g_{S_k}) \models φ_{S_k} and
• (I_T, g_{T_k}) \not\models ψ_{T_k} for any variable assignment g_{T_k} : \mathcal{V}(ψ_{T_k}) → Dom compatible with g_{S_k}.

In general, I_S does not satisfy all φ_{S_i} (1 ≤ i ≤ ℓ). From I_S, we construct I’_S that satisfies all φ_{S_i}, by merging I_S and a special instance satisfying all φ_{S_i} explained below. Choose a value c ∈ Dom that does not appear in I_S. For each relation schema R[U] in R_S, the special instance contains exactly one tuple t such that t.A = c for every attribute A ∈ U. The special instance satisfies Σ_S whatever Σ_S is since each relation has exactly one tuple, and satisfies all φ_{S_i} since they are constant-free. Clearly, the merged instance I’_S also satisfies Σ_S since c does not appear in I_S. Let g_S : \mathcal{V}(\bigwedge_{i=1}^{ℓ} φ_{S_i}) → Dom be a variable assignment defined as follows:
\[ g_S(x) = \begin{cases} 
  g_{S_1}(x) & \text{if } x \in \mathcal{V}(\varphi_{S_2}), \\
  c & \text{otherwise.} 
\end{cases} \]

It is easy to see that \( g_S \) is well defined because no variables are shared by different \( \varphi_{S_i} \), and that \( I'_S \) satisfies all \( g_{S_j} \) with \( g_S \). Now, we have that for any target instance \( I_T \) satisfying \( \Sigma_T \), there is some \( g_S \) such that

- \( (I'_T, g_{S_1}) \models \bigwedge_{i \in I} \varphi_{S_i} \), and
- \( (I_T, g_T) \not\models \bigwedge_{i \in I} \varphi_{S_i} \) for any variable assignment \( g_T : \mathcal{V}(\bigwedge_{i \in I} \varphi_{S_i}) \to \text{Dom} \) compatible with \( g_S \).

This means that \( I'_S \) has no solution under \( M' \). \( \square \)

3.2 Decision Tool for Instance Existence

Consider a database schema \( R \) and a set \( \Sigma \) of fds over \( R \). Let \( \varphi \) be a formula over \( R \). Due to \( \Sigma \), the values of the variables in \( \mathcal{V}(\varphi) \) are not independent in general. To capture this dependency among variables, below we introduce a function \( \mathcal{E}_{\varphi,\Sigma}(EQ) \) which returns, for a given set \( EQ \) of equalities on \( \mathcal{V}(\varphi) \) (and other variables in general), all the resulting equalities caused by \( \varphi \) and \( \Sigma \). Formally, for \( EQ \subseteq X \times X \), where \( X \subset \mathcal{V}(\varphi) \) is a finite set of variables, define \( \mathcal{E}_{\varphi,\Sigma}(EQ) \subseteq (\mathcal{V}(\varphi) \cup X) \times (\mathcal{V}(\varphi) \cup X) \) as the least equivalence relation with the following properties:

1. \( EQ \subseteq \mathcal{E}_{\varphi,\Sigma}(EQ) \); and
2. Suppose that an fd \( A_1 \cdots A_k \rightarrow B \) over \( R[U] \) is in \( \Sigma \), and \( \varphi \) contains subformulas \( R(x_1 \oplus A_1, \ldots, x_k \oplus A_k, z \oplus B, \ldots) \) and \( R(y_1 \oplus A_1, \ldots, y_k \oplus A_k, w \oplus B, \ldots) \) (the positions of the arguments are irrelevant). If \( (x_1, y_1, \ldots, x_k, y_k) \in \mathcal{E}_{\varphi,\Sigma}(EQ) \), then \((z, w) \in \mathcal{E}_{\varphi,\Sigma}(EQ)\).

For a variable assignment \( g : X \to \text{Dom} \), let \( EQ_g \subseteq X \times X \) denote the equivalence relation on variables induced by \( g \), i.e., \((x, y) \in EQ_g\) if and only if \( g(x) = g(y) \). The following two lemmas say that \( EQ_g \) is a fixpoint of \( \mathcal{E}_{\varphi,\Sigma} \) if and only if there is \( I \) such that \( (I, g) \models \varphi \) and \( I \models \Sigma \).

**Lemma 2:** Let \( \varphi \) be a formula over a database schema \( R \), \( g : \mathcal{V}(\varphi) \to \text{Dom} \) be a variable assignment, and \( \Sigma \) be a set of fds over \( R \). Suppose that \( (I, g) \models \varphi \) and \( I \models \Sigma \) for some database instance \( I \) over \( R \). Then, \( EQ_g = \mathcal{E}_{\varphi,\Sigma}(EQ_g) \).

**Proof.** Since \( EQ_g \subseteq \mathcal{E}_{\varphi,\Sigma}(EQ_g) \) by definition, it suffices to show that \( g(z) = g(w) \) for any pair \((z, w) \in EQ_g \). Consider a proof \( P_{(z,w)} \) of the membership of an arbitrary \((z, w) \in EQ_g \), i.e., an application sequence of reflexivity, symmetry, transitivity, and the two rules of the definition of \( \mathcal{E}_{\varphi,\Sigma} \). We show \( g(z) = g(w) \) by the induction on the length of \( P_{(z,w)} \).

For the basis, there are two cases. If \((z, w) \in \mathcal{E}_{\varphi,\Sigma}(EQ_g) \) by reflexivity, then \( z \) and \( w \) are the same variable. If \((z, w) \) is in \( \mathcal{E}_{\varphi,\Sigma}(EQ_g) \) by the first rule of the definition of \( \mathcal{E}_{\varphi,\Sigma} \), then \((z, w) \) must be in \( EQ_g \). In both cases, we have \( g(z) = g(w) \).

For the induction step, there are three cases. If \((z, w) \) is in \( \mathcal{E}_{\varphi,\Sigma}(EQ_g) \) by symmetry, we must already have \((w, z) \in \mathcal{E}_{\varphi,\Sigma}(EQ_g) \). Then, we have \((w, z) = (w, w) \) by the inductive hypothesis. If \((z, w) \) is in \( \mathcal{E}_{\varphi,\Sigma}(EQ_g) \) by transitivity, we can show that \((z, w) = (w, w) \) in a similar way to the symmetry case. Now, suppose that \((z, w) \) is in \( \mathcal{E}_{\varphi,\Sigma}(EQ_g) \) by the second rule of the definition of \( \mathcal{E}_{\varphi,\Sigma} \). Then, there are some fds \( A_1 \cdots A_k \rightarrow B \) over \( R[U] \) in \( \Sigma \) and subformulas \( R(x_1 \oplus A_1, \ldots, x_k \oplus A_k, z \oplus B, \ldots) \) and \( R(y_1 \oplus A_1, \ldots, y_k \oplus A_k, w \oplus B, \ldots) \) in \( \varphi \) such that \((x_1, y_1, \ldots, x_k, y_k) \in \mathcal{E}_{\varphi,\Sigma}(EQ_g) \). Since \((I, g) \models \varphi \), an instance \( I \) of \( R[U] \) in \( I \) contains tuples \( t \) and \( s \) such that
- \( t.A_1 = g(x_1), \ldots, t.A_k = g(x_k), t.B = g(z) \), and
- \( s.A_1 = g(y_1), \ldots, s.A_k = g(y_k), s.B = g(w) \).

By the inductive hypothesis, we have \((x_1, y_1, \ldots, x_k, y_k) \in \mathcal{E}_{\varphi,\Sigma}(EQ_g) \). Therefore, \( g(z) = g(w) \).

**Lemma 3:** Let \( \varphi \) be a formula over a database schema \( R \), \( g : \mathcal{V}(\varphi) \to \text{Dom} \) be a variable assignment, and \( \Sigma \) be a set of fds over \( R \). Suppose that \( EQ_g = \mathcal{E}_{\varphi,\Sigma}(EQ_g) \). Then, there is a database instance \( I \) over \( R \) such that \( (I, g) \models \varphi \) and \( I \models \Sigma \).

**Proof.** Suppose that \( EQ_g = \mathcal{E}_{\varphi,\Sigma}(EQ_g) \). Consider the following database instance \( I \): Each relation instance \( I \) of \( R[U] \) in \( I \) consists of all tuples \( t \) such that \( t.A_1 = g(x_1), \ldots, t.A_k = g(x_k) \) for some subformula \( R(x_1 \oplus A_1, \ldots, x_k \oplus A_k, z \oplus B, \ldots) \) in \( \varphi \).

Now, it is easy to see that \((I, g) \models \varphi \). To see that \( I \models \Sigma \), assume contrarily that \( I \) does not satisfy some fd \( A_1 \cdots A_k \rightarrow B \in \Sigma \) over some \( R[U] \). There would be two tuples \( t \) and \( s \) in a relation instance \( I \) of \( R[U] \) such that \( t.A_1 = g(x_1), \ldots, t.A_k = g(x_k) \) but \( t.B \neq s.B \). By the definition of \( I \), \( \varphi \) would contain subformulas \( R(x_1 \oplus A_1, \ldots, x_k \oplus A_k, z \oplus B, \ldots) \) and \( R(y_1 \oplus A_1, \ldots, y_k \oplus A_k, w \oplus B, \ldots) \) such that
- \( t.A_1 = g(x_1), \ldots, t.A_k = g(x_k), t.B = g(z) \), and
- \( s.A_1 = g(y_1), \ldots, s.A_k = g(y_k), s.B = g(w) \).

Hence, we would have that \( g(x_1) = g(y_1), \ldots, g(x_k) = g(y_k) \) but \( g(z) \neq g(w) \). However, since \( EQ_g \) is a fixpoint of \( \mathcal{E}_{\varphi,\Sigma} \), we must have \((z, w) \in EQ_g \) due to the second rule of the definition of \( \mathcal{E}_{\varphi,\Sigma} \). That is, we must have \( g(z) = g(w) \). This is a contradiction. \( \square \)

4. Results on the Upper Bound

In this section, we show two results on the upper bound for deciding absolute consistency. First, it is in coNP for general schema mappings. Secondly, it is solvable in polynomial time if all the tgdvs are full and the size of the left-hand side of any fd is bounded by some constant.

4.1 coNP Upper Bound for General Schema Mappings

Let \( M = ((R_S, \Sigma_S), (R_T, \Sigma_T), (\varphi_S \rightarrow \varphi_T)) \) be a schema mapping. In order to decide whether \( M \) is absolutely consistent, we have to check whether for any source instance \( I_S \) satisfying \( \Sigma_S \), tuples generated by the tgd \( \varphi_S \rightarrow \varphi_T \) from \( I_S \) satisfy \( \Sigma_T \). First, we define the notion of “generated tuples”
formally.

**Definition 5 (Generated tuples):** Let $R_S$ and $R_T$ be source and target schemas, respectively, and $\Sigma_S$ be a set of fids over $R_S$. Let $I_S$ be an instance over $R_S$ satisfying $\Sigma_S$. A tuple $t$ over $R[U] \in R_T$ is generated from $I_S$ by a tgd $\phi_S \rightarrow \psi_T$ with variable assignments $g_S : \mathcal{V}(\phi_S) \rightarrow \text{Dom}$ and $g_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom}$ if

- $g_T$ is compatible with $g_S$,
- $(I_S, g_S) \models \phi_S$,
- $\psi_T$ contains a subformula $R(x_1 \in A_1, \ldots, x_{|U|} \in A_{|U|})$, and
- $t \cdot A_j = g_T(x_j)$ for all $j$ (1 ≤ $j$ ≤ $|U|$).

Using the notion of generated tuples, we provide a necessary and sufficient condition for $I_S$ to have no solution:

**Lemma 4:** Let $M = ((R_S, \Sigma_S), (R_T, \Sigma_T), (\phi_S \rightarrow \psi_T))$ be a schema mapping. A source instance $I_S$ satisfying $\Sigma_S$ has no solution under $M$ if and only if there exist $g_S : \mathcal{V}(\phi_S) \rightarrow \text{Dom}$ and $g'_S : \mathcal{V}(\phi_S) \rightarrow \text{Dom}$ such that

- $(I_S, g_S) \models \phi_S$,
- $(I_S, g'_S) \models \phi_S$, and
- for any $g_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom}$ and $g'_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom}$ compatible with $g_S$ and $g'_S$ respectively, there exist tuples $t$ and $t'$ such that
  - $t$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g_T$ and $g_T$,
  - $t'$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g'_T$ and $g'_T$, and
  - $(t, t')$ does not satisfy $\Sigma_T$.

**Proof.** Let $M' = ((R_S, \Sigma_S), (R_T, \emptyset), (\phi_S \rightarrow \psi_T))$. Note that $\text{Sol}_M(I_S) \subseteq \text{Sol}_M(I_S)$ for any source instance $I_S$ satisfying $\Sigma_S$.

(If part.) Suppose that the latter half of the statement of the lemma holds. By the definition of solutions, we can easily see that each $I_T \in \text{Sol}_M(I_S)$ contains tuples $t$ and $t'$ such that $(t, t')$ does not satisfy $\Sigma_T$. Hence $I_S$ has no solution under $M$.

(Only if part.) Suppose that $I_S$ has no solution under $M$. Since $\text{Sol}_M(I_S)$ cannot be empty, we have that no instance in $\text{Sol}_M(I_S)$ satisfies $\Sigma_T$.

Let us fix an arbitrary instance $I_T \in \text{Sol}_M(I_S)$. Let $s$ and $s'$ be tuples in $I_T$ such that $(s, s')$ does not satisfy an $\text{fid}$ in $\Sigma_T$. We can assume that both $s$ and $s'$ are generated from $I_S$ by $\phi_S \rightarrow \psi_T$. To see this, suppose conversely that for every pair of tuples $s$ and $s'$ in $I_T$ such that $(s, s')$ does not satisfy an $\text{fid}$ in $\Sigma_T$, either $s$ or $s'$ is generated from $I_S$. Then, the instance obtained by removing the non-generated tuple from $I_T$ must be in $\text{Sol}_M(I_S)$. By repeating the removal for every pair of such tuples, we could obtain an instance in $\text{Sol}_M(I_S)$ that satisfies $\Sigma_T$.

Now, fix variable assignments $g_S : \mathcal{V}(\phi_S) \rightarrow \text{Dom}$ and $g'_S : \mathcal{V}(\phi_S) \rightarrow \text{Dom}$ so that

- $s$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g_S$ and some $g_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom}$ compatible with $g_S$, and
- $s'$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g'_S$ and some $g'_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom}$ compatible with $g'_S$.

Since no instance in $\text{Sol}_M(I_S)$ satisfies $\Sigma_T$, we have that for any $g_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom}$ and $g'_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom}$ compatible with $g_S$ and $g'_S$ respectively, there exist tuples $t$ and $t'$ such that

- $t$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g_S$ and $g_T$,
- $t'$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g'_S$ and $g'_T$, and
- $(t, t')$ does not satisfy $\Sigma_T$.

To conclude the proof of this part, we have to show that the fixed $g_S$ and $g'_S$ are common in all the instances in $\text{Sol}_M(I_S)$. Consider any instance $I_T \in \text{Sol}_M(I_S)$. $I_T$ must include tuples $u$ and $u'$, where

- $u$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g_S$ and some $g_T$, and
- $u'$ is generated from $I_S$ by $\phi_S \rightarrow \psi_T$ with $g'_S$ and some $g'_T$,

because $(I_S, g_S) \models \phi_S$ and $(I_S, g'_S) \models \phi_S$. However, as the discussion above, for any variable assignments $g_T$ and $g'_T$ compatible with $g_S$ and $g'_S$ respectively, $(u, u')$ does not satisfy $\Sigma_T$. □

The above lemma states that considering two variable assignments is sufficient for deciding whether $I_S$ has a solution or not. To integrate this observation with the tool $E$, we introduce a notion of a $\text{copy}$ of a variable. A formula $\varphi'$ is a $\text{copy}$ of $\varphi$ if $\varphi'$ is obtained by renaming all the variables in $\varphi$ by new variable names. We say that a tgd $\phi_S \rightarrow \psi_T$ is a $\text{copy}$ of $\phi_S \rightarrow \psi_T$ if $\phi'_S$ and $\psi'_T$ are copies of $\phi_S$ and $\psi_T$, respectively, and each common variable appearing in $\phi_S$ and $\psi_T$ is renamed by the same new name in $\phi'_S$ and $\psi'_T$. Then, we focus on the fixpoints of $E_{\phi_S \cup \psi_T, \{\phi'_S \cup \psi'_T\}}$ and $E_{\phi'_S \cup \psi'_T, \{\phi_S \cup \psi_T\}}$, rather than $E_{\phi_S \cup \psi_T, \{\phi'_S \cup \psi'_T\}}$ and $E_{\phi'_S \cup \psi'_T, \{\phi_S \cup \psi_T\}}$. Using a copy, we can simulate two different variable assignments by one variable assignment.

Hereafter, for an equivalence relation $EQ \subseteq X \times X$ and $Y \subseteq X$, we write $EQ \cap Y \times Y$. $EQ$$EQ$.

**Theorem 1:** Let $M = ((R_S, \Sigma_S), (R_T, \Sigma_T), (\phi_S \rightarrow \psi_T))$ be a schema mapping. Let $\varphi_S \rightarrow \psi_T$ be a copy of $\varphi_S \rightarrow \psi_T$. Define $X_S = \mathcal{V}(\phi_S) \cup \mathcal{V}(\phi'_S)$, $X_T = \mathcal{V}(\psi_T) \cup \mathcal{V}(\psi'_T)$, and $X = X_S \cup X_T$. $M$ is absolutely consistent if and only if for any equivalence relation $EQ_{S} \subseteq X_S \times X_S$ such that

$$EQ_s = E_{\phi_S \cup \psi_T, \{\phi'_S \cup \psi'_T\}}(EQ_S),$$

there exists an equivalence relation $EQ \subseteq X \times X$ such that

$$EQ|_{X_S} = EQ_S = E_{\phi'_S \cup \psi'_T, X_T}(EQ|_{X_S}).$$

**Proof.** (Only if part.) Suppose that there exists a fixpoint $EQ_S \subseteq X_S \times X_S$ of $E_{\phi_S \cup \psi_T, \{\phi'_S \cup \psi'_T\}}$ such that for any $EQ \subseteq X \times X$, if $EQ|_{X_S} = EQ_S$ then $EQ|_{X_S} \neq E_{\phi'_S \cup \psi'_T, \{\phi_S \cup \psi_T\}}(EQ|_{X_S})$. Let $\bar{g}_S = EQ_S$. By Lemma 3, there is $I_S$ satisfying $\Sigma_S$ such that $(I_S, \bar{g}_S) \models \varphi_S \land \varphi'_S$. □
Consider an arbitrary equivalence relation \( EQ \subseteq X \times X \) such that \( E\mathcal{Q}|_{X_S} = EQ_{X_S} \). We have \( E\mathcal{Q}|_{X_S} \subseteq \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ)|_{X_S} \) by assumption, and hence, \( EQ \) cannot be a fixpoint of \( \mathcal{E}_{\psi_T/\psi'_T,S_T} \). From this fact and Lemma 2, for any \( \tilde{g}_T : X_T \rightarrow \text{Dom compatible with } \tilde{g}_S \), if \( (I_T, \tilde{g}_T) \models \psi_T \land \psi'_T \), then \( I_T \models \Sigma_T \).

Now, we decompose \( \tilde{g}_S \) and \( \tilde{g}_T \) and continue the discussion. First, define \( g_S : \mathcal{V}(\varphi_S) \rightarrow \text{Dom} \) and \( g'_S : \mathcal{V}(\varphi_S) \rightarrow \text{Dom} \) so that

\[
\begin{align*}
g_S(x) &= \tilde{g}_S(x), \\
g'_S(x) &= \tilde{g}_S(x'),
\end{align*}
\]

where \( x' \) denotes a copied variable of \( x \). We have \( (I_S, g_S) \models \varphi_S \) and \( (I_S, g'_S) \models \varphi_S \). Next, arbitrarily fix \( \tilde{g}_T : X_T \rightarrow \text{Dom compatible with } \tilde{g}_S \). Note that this is equivalent to arbitrarily fix two variable assignments \( g_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom} \) and \( g'_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom} \), where

\[
\begin{align*}
g_T(x) &= \tilde{g}_T(x), \\
g'_T(x) &= \tilde{g}_T(x'),
\end{align*}
\]

compatible with \( g_S \) and \( g'_S \), respectively. Let \( I_T \) be an instance such that \( (I_T, \tilde{g}_T) \models \psi_T \land \psi'_T \). Since \( I_T \not\models \Sigma_T \), there exist tuples \( t \) and \( t' \) in \( I_T \) such that \( (t, t') \) does not satisfy \( \Sigma_T \). By the same discussion in the only if part of the proof of Lemma 4, we can assume that both \( t \) and \( t' \) are generated from \( I_S \) by \( \text{tg}_D (\varphi_S \land \varphi'_S) \rightarrow (\psi_T \land \psi'_T) \) with \( g_S \) and \( \tilde{g}_T \). Moreover, \( \varphi_S \) and \( \psi_T \) contain only original variables and \( \varphi'_S \) and \( \psi'_T \) contain only copied variables, we have either

- \( t \) is generated from \( I_S \) by \( \varphi_S \rightarrow \psi_T \) with \( g_S \) and \( g_T \), or
- \( t' \) is generated from \( I_S \) by \( \varphi'_S \rightarrow \psi_T \) with \( g'_S \) and \( g'_T \).

The same holds for \( t' \).

Thus, we have that the latter half of the statement of Lemma 4 holds. Therefore, \( I_S \) has no solution, and \( M \) is not absolutely consistent.

(If part.) Suppose that \( M \) is not absolutely consistent, i.e., there is an instance \( I_S \) that has no solution. Then, by Lemma 4, there exist \( g_S : \mathcal{V}(\varphi_S) \rightarrow \text{Dom} \) and \( g'_S : \mathcal{V}(\varphi_S) \rightarrow \text{Dom} \) such that

- \( (I_S, g_S) \models \varphi_S \),
- \( (I_S, g'_S) \models \varphi_S \), and
- for any \( g_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom} \) and \( g'_T : \mathcal{V}(\psi_T) \rightarrow \text{Dom} \) compatible to \( g_S \) and \( g'_S \) respectively, there exist tuples \( t \) and \( t' \) such that
  - \( t \) is generated from \( I_S \) by \( \varphi_S \rightarrow \psi_T \) with \( g_S \) and \( g_T \),
  - \( t' \) is generated from \( I_S \) by \( \varphi'_S \rightarrow \psi_T \) with \( g'_S \) and \( g'_T \), and
  - \( (t, t') \) does not satisfy \( \Sigma_T \).

Arbitrarily fix \( g_T \) and \( g'_T \) compatible with \( g_S \) and \( g'_S \), respectively. Define \( \tilde{g}_S \) and \( \tilde{g}_T \) as follows:

\[
\tilde{g}_S(x) = \begin{cases} g_S(x) & \text{if } x \text{ is an original variable}, \\ g'_S(y) & \text{if } y \text{ is a copied variable of } y, \end{cases}
\]

Then, it is easy to see that both \( t \) and \( t' \) can be generated from \( I_S \) by \( \text{tg}_D \varphi_S \land \varphi'_S \rightarrow \psi_T \land \psi'_T \) with \( \tilde{g}_S \) and \( \tilde{g}_T \). Define \( \tilde{g} \) as follows:

\[
\tilde{g}(x) = \begin{cases} \tilde{g}_S(x) & \text{if } x \in X_S, \\ \tilde{g}_T(x) & \text{if } x \in X_T. \end{cases}
\]

\( \tilde{g} \) is well-defined because \( \tilde{g}_T \) is compatible with \( \tilde{g}_S \).

Since \( (I_S, \tilde{g}_S) \models \varphi_S \land \varphi'_S \) and \( I_T \models \Sigma_T, \) \( E\mathcal{Q}_S \) is a fixpoint of \( \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \) by Lemma 2. On the other hand, since there is no \( I_T \) such that \( (I_T, \tilde{g}_S) \models \psi_T \land \psi'_T \), \( E\mathcal{Q}_S \) cannot be a fixpoint of \( \mathcal{E}_{\psi_T/\psi'_T,S_T} \) by Lemma 3. Since \( g_T \) and \( g'_T \) are arbitrarily fixed, we can conclude that for any \( \tilde{g} \) compatible with \( \tilde{g}_S \), \( E\mathcal{Q}_S \) is not a fixpoint of \( \mathcal{E}_{\psi_T/\psi'_T,S_T} \). This implies that \( E\mathcal{Q}_S \) is not in \( \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \), which can be induced by some variable assignment compatible with \( \tilde{g}_S \), would be a fixpoint of \( \mathcal{E}_{\psi_T/\psi'_T,S_T} \).

\begin{algorithm}
\begin{algorithmic}
\State \textbf{Input:} \( \varphi, \Sigma, \) and \( EQ \)
\State \textbf{Ans := } \( EQ \)
\Repeat
\ForAll \text{tuple } \( A_1, \ldots, A_t \to B \) \text{ in } \Sigma \do
\ForAll \text{pair of subformulas } \begin{align*}
&\begin{array}{c}
\varphi_1 \land \varphi_2, \\varphi_3 \land \varphi_4 \end{array}
\end{align*}
\text{ appearing in } \varphi \do
\If \((x_1, y_1), \ldots, (x_n, y_n) \in \text{Ans} \) \Text{ then} \text{Ans := } \text{the least equivalence relation containing } \text{Ans } \cup \{(z, w)\};
\EndIf
\EndFor
\EndFor
\Until \text{Ans does not change;}
\Return \text{Ans}
\end{algorithmic}
\end{algorithm}

\begin{theorem}
Absoluteness of consistency of \( M \) is in \( \text{coNP} \).
\end{theorem}

\begin{proof}
An NP algorithm for the complement of the absolute consistency problem is as follows: Nondeterministically guess an equivalence relation \( EQ_S \subseteq X_S \times X_S \) and check whether \( EQ_S \neq \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \) by \( X_S \). Computation of \( \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \) can be done in polynomial time by Algorithm 1.

To show the correctness of the above algorithm, it suffices to prove that if \( EQ_S \neq \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \), then \( \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \neq EQ \) for all \( EQ \) such that \( E\mathcal{Q}_S = EQ \). Suppose that \( EQ_S \neq \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \), i.e., there is \((z, w) \in X_S \times X_S \) such that \((z, w) \notin EQ \) and \((z, w) \in \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \). Consider a proof \( P_{(z, w)} \) of the membership of \((z, w) \) in \( \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \), i.e., an application sequence of reflexivity, symmetry, transitivity, and the two rules of the definition of \( \tilde{E} \). For any \( EQ \) such that \( E\mathcal{Q}_S = EQ \), \( P_{(z, w)} \) can also be a proof of the membership of \((z, w) \) in \( \mathcal{E}_{\psi_T/\psi'_T,S_T}(EQ) \) because \( EQ_S \subseteq EQ \).
\end{proof}
4.2 PTIME Upper Bound for Schema Mappings with Full tgds

As a special case of Theorem 1, we have the following corollary:

**Corollary 1:** Let $M = ((R_S, \Sigma_S), (R_T, \Sigma_T), \{\varphi_S \rightarrow \psi_T\})$ be a schema mapping and suppose that $\varphi_S \rightarrow \psi_T$ is full. Let $\varphi'_S \rightarrow \psi'_T$ be a copy of $\varphi_S \rightarrow \psi_T$. $M$ is absolutely consistent if and only if every fixpoint of $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}$ is also a fixpoint of $E_{\varphi_S \rightarrow \psi_T, \Sigma_T}$.

The condition in Corollary 1 requires the computation of $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}$ and $E_{\varphi_S \rightarrow \psi_T, \Sigma_T}$ for exponentially many $EQ_S$s in general. In what follows, we show that the condition can be checked in polynomial time, provided that the size of the left-hand side of any fd is bounded by some constant $K$.

**Lemma 5:** Let $M = ((R_S, \Sigma_S), (R_T, \Sigma_T), \{\varphi_S \rightarrow \psi_T\})$ be a schema mapping and suppose that $\varphi_S \rightarrow \psi_T$ is full. Let $\varphi'_S \rightarrow \psi'_T$ be a copy of $\varphi_S \rightarrow \psi_T$. Define $X_S = \mathcal{V}(\varphi_S) \cup \mathcal{V}(\varphi'_S)$. The following statements are equivalent:

1. Every fixpoint of $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}$ is also a fixpoint of $E_{\varphi_S \rightarrow \psi_T, \Sigma_T}$.
2. $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ) \subseteq E_{\varphi_S \rightarrow \psi_T, \Sigma_T}(EQ)$ for each $EQ \subseteq X_S \times X_S$.

**Proof.** (1 $\Rightarrow$ 2) Let $EQ \subseteq X_S \times X_S$. Let $EQ_S = E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ)$ and $EQ_T = E_{\varphi_S \rightarrow \psi_T, \Sigma_T}(EQ)$. Since $EQ_S$ is a fixpoint of $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}$, it is also a fixpoint of $E_{\varphi_S \rightarrow \psi_T, \Sigma_T}$, i.e., $EQ_S = E_{\varphi_S \rightarrow \psi_T, \Sigma_T}(EQ_S)$. Since $EQ \subseteq EQ_S$, we have $EQ_T = E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ) \subseteq E_{\varphi_S \rightarrow \psi_T, \Sigma_T}(EQ_S) = EQ_S$.

(2 $\Rightarrow$ 1) Let $EQ$ be a fixpoint of $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}$. Then, $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ) \subseteq E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ) = EQ$. On the other hand, by the definition of $E$, we have $EQ \subseteq E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ)$. Hence $EQ = E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ)$. \qed

We say that an fd $X \rightarrow Y$ is $K$-bounded if $X$ contains $K$ or less attributes.

**Lemma 6:** Let $M = ((R_S, \Sigma_S), (R_T, \Sigma_T), \{\varphi_S \rightarrow \psi_T\})$ be a schema mapping and suppose that $\varphi_S \rightarrow \psi_T$ is full. Let $\varphi'_S \rightarrow \psi'_T$ be a copy of $\varphi_S \rightarrow \psi_T$. Define $X_S = \mathcal{V}(\varphi_S) \cup \mathcal{V}(\varphi'_S)$. Let $K$ be a constant such that every fd in $\Sigma_S \cup \Sigma_T$ is $K$-bounded. Then, the following statements are equivalent:

1. $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ) \subseteq E_{\varphi_S \rightarrow \psi_T, \Sigma_T}(EQ)$ for each $EQ \subseteq X_S \times X_S$.
2. $E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ) \subseteq E_{\varphi_S \rightarrow \psi_T, \Sigma_T}(EQ)$ for each $EQ \subseteq X_S \times X_S$ with $|EQ| \leq K$.

**Proof.** The part (1 $\Rightarrow$ 2) is obvious. Consider an arbitrary $EQ \subseteq X_S \times X_S$, and let $EQ_S = E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}(EQ)$ and $EQ_T = E_{\varphi_S \rightarrow \psi_T, \Sigma_T}(EQ)$. To show the part (2 $\Rightarrow$ 1), consider a proof $P_{(w)}$ of the membership of an arbitrary $(z, w)$ in $EQ_T$, i.e., an application sequence of reflexivity, symmetry, transitivity, and the two rules of the definition of $E$. We show $(z, w) \in EQ_S$ by the induction on the length of $P_{(w)}$.

For the basis, there are two cases. If $(z, w)$ is in $EQ_T$ by reflexivity, then it is also in $EQ_S$ by reflexivity. If $(z, w)$ is in $EQ_T$ by the first rule of the definition of $E$, then $(z, w)$ must be in $EQ$ and hence it is also in $EQ_S$ by the same rule.

For the induction step, there are three cases. If $(z, w)$ is in $EQ_T$ by symmetry, we must already have $(w, z) \in EQ_T$. Then, we have $(w, z) \in EQ_S$ by the inductive hypothesis, and hence, $(z, w) \in EQ_S$. If $(z, w)$ is in $EQ_T$ by transitivity, we can show that $(z, w) \in EQ_S$ in a similar way to the symmetry case. Now, suppose that $(z, w)$ is in $EQ_T$ by the second rule of the definition of $E$. Then, there are some fd $A_1 \cdots A_k \rightarrow B$ over $R(U)$ in $\Sigma_T$ and subformulas $R(x_1@A_1, \ldots, x_k@A_k, z@B, \ldots)$ and $R(y_1@A_1, \ldots, y_k@A_k, w@B, \ldots)$ in $\psi_T$ and $\psi'_T$ such that $(z, w) \in E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}((x_1, y_1), \ldots, (x_k, y_k))$. Since $k \leq K$, we have $(z, w) \in E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}((x_1, y_1), \ldots, (x_k, y_k)) \subseteq E_{\varphi'_S \rightarrow \psi'_T, \Sigma_T}((x_1, y_1), \ldots, (x_k, y_k))$. Hence, $(z, w) \in EQ_S$ since $(x_1, y_1), \ldots, (x_k, y_k) \in EQ_S$ by the inductive hypothesis. \qed

**Theorem 3:** Absolute consistency of $M$ is decidable in polynomial time if all the tgds are full and there is a constant $K$ such that every fd is $K$-bounded.

5. Results on the Lower Bound

In this section, we show that the absolute consistency problem for a schema mapping with non-full tgds is coNP-hard even if

1. the source schema has no fd and the target schema has only one fd; or
2. the source schema has only one, 1-bounded fd and the target schema has only one, 2-bounded fd.

5.1 Case 1: No Source fd, One Target fd

We show that the absolute consistency problem is coNP-hard even if the source schema has no fd and the target schema has only one fd. The proof is done by reducing unsatisfiability of CNF formulas to the absolute consistency problem.

Let $F = C_1 \land \cdots \land C_n$ be a CNF formula, where each clause $C_j$ is a disjunction of literals. Each literal is either positive $x$ or negative $\neg x$, where $x$ is a logical variable. We suppose that there are $m$ logical variables $x_1, \ldots, x_m$ in $F$. By the reduction, we construct the following schema mapping $M = ((R_S, \Sigma_S), (R_T, \Sigma_T), \{\varphi_S \rightarrow \psi_T\})$:

- $R_S = \langle R_S \{0\} \rangle$, where $0$ is a single attribute,
- $\Sigma_S = \emptyset$,
- $R_T = \langle R_T[XAY] \rangle$, where $X = A_1 \cdots A_m A_f$, $A$ is a single attribute, and $Y = B_0 \cdots B_{n+1}$,
- $\Sigma_T = \{X \rightarrow Y\}$,
- $\varphi_S = R_S(x_1@O) \land \cdots \land R_S(x_m@O) \land R_S(x_1@O) \land R_S(x_f@O)$, and
- $\psi_T$ is the conjunction of the following subformulas:

1. $R_T(x@X, x_1@A_\_@Y[y_{j-1}@B_j]) \land R_T(x@X, x_1@A_\_@Y[y_j@B_j])$ for each $C_j$ containing a positive literal $x_i$,
2. \( R_T(\mathbf{X}, x_1 \to A, \ldots, x_j \to A, \ldots, x_m \to A, x_j \to A_f) \) and \( R_T(\mathbf{X}, x_j \to A_f, \ldots, x_m \to A) \) for each \( C_j \) containing a negative literal \( \neg x_i \).

3. \( R_T(x \to A, \ldots, x \to A_n, \ldots, x \to A) \) and \( R_T(x \to A, \ldots, x \to A_n, \ldots, x \to A) \), and

4. \( R_T(\mathbf{X}, x_1 \to A, \ldots, x_n \to A, x_1 \to A) \) and \( R_T(\mathbf{X}, x_1 \to A, \ldots, x_n \to A, x_1 \to A) \),

where \( \mathbf{x} \to \mathbf{X} \) represents

\[
x_1 \to A_1, \ldots, x_m \to A_m, x_j \to A_f
\]

and \( \ldots, Y(y \to B_j) \) (0 \( j \leq n + 1 \)) represents

\[
v_0 \to B_0, \ldots, v_{n+1} \to B_{n+1}
\]

such that \( v_j = y \) and other variables \( v_j (j \neq j) \) are anonymous, i.e., they appear only once in \( \psi_T \).

Hereafter, let \( \mathbf{x}' \) denote the copied version of a variable \( x \). Let \( X_S = \mathbf{V}(\varphi_S) \cup \mathbf{V}(\varphi_S') \), \( X_T = \mathbf{V}(\psi_T) \cup \mathbf{V}(\psi_T') \), and \( X = X_S \cup X_T \).

Intuitively, in our reduction, variables \( x_1, \ldots, x_m \) in \( \varphi_S \) correspond to the logical variables \( x_1, \ldots, x_m \) in \( F \). Let \( g_S : X_S \to \text{Dom} \), and let \( \mu \) be a truth assignment for \( F \). In our reduction, \( (x_i, x_i) \in EQ_{g_S} \) means \( \mu(x_i) = \text{True} \) and \( (x_i, x_j) \in EQ_{g_T} \), \( \mu(x_i) = \text{False} \). For example, consider \( g_S \) such that \( EQ_{g_S} \) is the least equivalence relation containing \( (x_i, x_i) \) and \( (x_j, x_j) \) (see Fig. 1). Then, \( g_S \) corresponds to \( \mu \) such that \( \mu(x_1) = \text{True}, \mu(x_2) = \text{False} \), and \( \mu(x_3) \) is “don’t care.” Hence, the main role of \( x_i \) and \( x_j \) is to represent the truth values, and therefore \( g_S \) is supposed to satisfy \( (x_i, x_j) \notin EQ_{g_T} \). Moreover, \( x_1 \) and \( x_2 \) have another role: the “flag” for satisfaction of \( F \) under \( \mu \). Precisely, \( \mu \) does not satisfy \( F \) if and only if \( (x_i, x_j) \notin EQ_{g_S} \), for any \( g_T \) compatible with \( g_S \) corresponding to \( \mu \). To be more specific, we have \( (y_{j-1}, y_j) \in EQ_{g_T} \), if and only if clause \( C_j \) is \( \text{True} \) under \( \mu \), due to the target \( \text{fd} \) in \( \Sigma \) and the first and the second rules of the definition of \( \psi_T \). Since we always have \( (x_i, y_0), (y_0, x_j) \in EQ_{g_T} \), by the target \( \text{fd} \) and the third and the fourth rules of the definition of \( \psi_T \), it is concluded that \( (x_i, x_j) \notin EQ_{g_T} \) if and only if \( F \) is satisfied by \( \mu \) (see Fig. 2).

As stated above, the intuition is rather simple. However, we also have to take account of the effect by a copy \( \psi_T' \) of \( \psi_T \). Especially, we want to avoid the case where \( \psi_T \) and \( \psi_T' \) are corresponding to different truth assignments, i.e., the values assigned to the variables \( \mathbf{x} \to \mathbf{X} \) in \( \psi_T \) and \( \psi_T' \) are different. That is why the target \( \text{fd} \) is \( X_A \to Y \), not \( A \to Y \), in our reduction.

Lemma 7: Let \( EQ_S \subseteq X_S \times X_S \) and \( EQ \subseteq X \times X \) be equivalence relations such that \( EQ_{|X_S} = EQ_S \). Let \( (w_1, w_2) \in X_S \times X_S \). Suppose that \( (w_1, w_2) \notin EQ_S \) and \( (w_1, w_2) \notin EQ_{\psi_T, \psi_T'}(EQ) \). Then, at least one of the following two conditions hold:

1. \( (w_1, x_i), (w_2, x_i) \in EQ_S \) or \( (w_1, x_j), (w_2, x_i) \in EQ_S \) and \( (x_i, x_j) \notin EQ_S \).
2. \( (w_1, x_i'), (w_2, x_j') \in EQ_S \) or \( (w_1, x_j'), (w_2, x_i') \in EQ_S \) and \( (x_i', x_j') \notin EQ_S \).

Proof. During the computation of \( EQ_{\psi_T, \psi_T'}(EQ) \), possibly added pairs involving variables in \( X_S \) are only \( (x_1, y_0), (y_0, x_j), (x_i', y_0'), \) and \( (y_0', x_j') \). Hence, in order for \( w_1 \in X_S \) to become a pair with \( w_2 \in X_S \) in \( EQ_{\psi_T, \psi_T'}(EQ) \), at least one of \( (w_1, x_i), (w_1, x_j'), (w_1, x_j') \), and \( (w_1, x_j') \) must be in \( EQ_S \) (and the same applies to \( w_2 \)). In what follows we examine all the possible combinations:

- \( (w_1, x_i), (w_2, x_i') \notin EQ_S \) or \( (w_1, x_j'), (w_2, x_i') \notin EQ_S \).
- \( (w_1, x_j), (w_2, x_j') \notin EQ_S \) or \( (w_1, x_j), (w_2, x_j') \notin EQ_S \).
- \( (w_1, x_j'), (w_2, x_j') \notin EQ_S \) or \( (w_1, x_j'), (w_2, x_j') \notin EQ_S \).

Since \( (w_1, w_2) \notin EQ_S \), we have \( (x_i, x_j') \notin EQ_S \).

On the other hand, since \( (w_1, w_2) \notin EQ_{\psi_T, \psi_T'}(EQ) \), we have \( (x_i, x_j') \in EQ_{\psi_T, \psi_T'}(EQ) \).

To obtain this, we need at least one application of an \( \text{fd} \) to original atomic formula \( R_T(\mathbf{X} \to \mathbf{X}, \ldots) \) and copied atomic formula \( R_T(\mathbf{X} \to \mathbf{X}, \ldots) \).

However, this application is possible only when the values of \( x \) and \( x' \) are equal. This is a contradiction to the assumption that \( (x_i, x_j') \notin EQ_S \).

Hence this combination is impossible.

- \( (w_1, x_j), (w_2, x_j') \in EQ_S \) or \( (w_1, x_j'), (w_2, x_j) \in EQ_S \).

This combination is impossible. It can be shown in the same way as the first combination.

- \( (w_1, x_j), (w_2, x_j') \in EQ_S \) or \( (w_1, x_j'), (w_2, x_j) \in EQ_S \).

As in the first case, we need at least one application of an \( \text{fd} \) to original atomic formula \( R_T(\mathbf{X} \to \mathbf{X}, \ldots) \) and copied atomic formula \( R_T(\mathbf{X} \to \mathbf{X}, \ldots) \).

Again, this application is possible only when the values of \( x \) and \( x' \) are equal. Hence, we have \( (w_1, x_j'), (w_2, x_j), (w_2, x_j') \in EQ_S \) or \( (w_1, x_j), (w_1, x_j'), (w_2, x_j) \in EQ_S \) or \( (w_1, x_j), (w_1, x_j'), (w_2, x_j') \in EQ_S \).

In either case, we have \( (x_i, x_j'), (x_i', x_j') \notin EQ_S \) since \( (w_1, w_2) \notin EQ_S \).

- \( (w_1, x_j), (w_2, x_j') \in EQ_S \) or \( (w_1, x_j'), (w_2, x_j) \in EQ_S \).

In the same way as the previous combination, \( (w_1, x_j), (w_1, x_j'), (w_2, x_j), (w_2, x_j') \in EQ_S \) or \( (w_1, x_j), (w_1, x_j'), (w_2, x_j') \in EQ_S \) or \( (w_1, w_2) \notin EQ_S \).

In either case, we have \( (x_i, x_j), (x_i', x_j') \notin EQ_S \) since \( (w_1, w_2) \notin EQ_S \).

- \( (w_1, x_j), (w_2, x_j) \in EQ_S \) or \( (w_1, x_j'), (w_2, x_j) \in EQ_S \).

Since \( (w_1, w_2) \notin EQ_S \), we have \( (x_i, x_j) \notin EQ_S \).

- \( (w_1, x_j'), (w_2, x_j') \in EQ_S \) or \( (w_1, x_j'), (w_2, x_j') \in EQ_S \).
Since \((w_1, w_2) \notin EQ_S\), we have \((x'_i, x'_j) \notin EQ_S\).

In summary, the statement of the lemma is concluded.

\(\square\)

**Lemma 8:** Let \(EQ_S \subseteq X_S \times X_S\) and \(EQ \subseteq X \times X\) be equivalence relations such that \(EQ|_{X_S} = EQ_S\). The following two conditions are equivalent:

1. \(EQ_S \neq \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\). That is, there exists some \((w_1, w_2) \in X_S \times X_S\) such that \((w_1, w_2) \notin EQ_S\) and \((w_1, w_2) \in \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\).

2. \((x_i, x_j) \notin EQ_S\) and \((x'_i, x'_j) \notin \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\) or \((x'_i, x'_j) \notin EQ_S\) and \((x'_i, x'_j) \notin \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\).

**Proof.** The part \((2 \Rightarrow 1)\) is obvious because \((x_i, x_j), (x'_i, x'_j) \in X_S \times X_S\). We show \((1 \Rightarrow 2)\). Condition 1 of this lemma implies that one of the conditions of Lemma 7 holds. Suppose that condition 1 of Lemma 7 holds. Immediately we have \((x_i, x_j) \notin EQ_S\). Moreover, by the assumption that \((w_1, w_2) \in \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\) and Lemma 7 again, we have \((x_i, x_j) \notin \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\). Hence, we obtain the former half of condition 2 of this lemma. The latter half is obtained in the same way.

\(\square\)

**Lemma 9:** \(M\) is absolutely consistent if and only if \(F\) is unsatisfiable.

**Proof.** (Only if part.) Suppose that \(F\) is satisfiable. Let \(\mu\) be a truth assignment satisfying \(F\). Define \(EQ_\mu \subseteq X_S \times X_S\) as the least equivalence relation satisfying the following conditions:

- \((x_i, x_j), (x'_i, x'_j) \in EQ_\mu\) if \(\mu(x_i) = \text{TRUE}\), and
- \((x_i, x_j), (x'_i, x'_j) \notin EQ_\mu\) if \(\mu(x_i) = \text{FALSE}\).

Note that \(EQ_\mu = \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ_\mu)|_{X_S}\) since \(\Sigma_S = \emptyset\). Also note that \((x_i, x_j) \notin EQ_\mu\) since \(\mu\) does not assign both \text{TRUE} and \text{FALSE} to the same variable. On the other hand, consider any equivalence relation \(EQ \subseteq X \times X\) such that \(EQ|_{X_S} = EQ_\mu\). Since \(\mu\) satisfies \(F\), each clause \(C_j\) has a literal that makes \(C_j\) \text{TRUE}. If a positive literal \(x_i\) makes \(C_j\) \text{TRUE}, we have \((y_{j-1}, y_j) \in \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\) from the first item of the definition of \(\psi_T\). Similarly, if a negative literal \(-x_i\) makes \(C_j\) \text{TRUE}, we have \((y_{j-1}, y_j) \in \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\) from the second item of the definition of \(\psi_T\). Hence, for each \(j (1 \leq j \leq n)\), we have \((y_{j-1}, y_j) \in \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\). Moreover, from the third and the fourth items of the definition of \(\psi_T\), we always have \((x_i, y_i), (y_{i+1}, x_i) \in \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\). Consequently, we have \((x_i, x_j) \notin \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\). By Theorem 1, \(M\) is not absolutely consistent.

(If part.) Suppose that \(M\) is not absolutely consistent. Then, by Theorem 1, there exists some fixpoint \(EQ_S \subseteq X_S \times X_S\) of \(\mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}|_{X_S}\) such that for any equivalence relation \(EQ \subseteq X \times X\) such that \(EQ|_{X_S} = EQ_S\), it holds that \(EQ_S \neq \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\). Thus \(EQ = EQ_S \cup \{ (x, x) \mid x \in X \setminus X_S\}\). By Lemma 8, either \((x_i, x_j)\) or \((x'_i, x'_j)\) is an evidence of the inequality \(EQ_S \neq \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\), and we can focus on only \((x_i, x_j)\) due to the symmetry of \(\psi_T\) and \(\varphi_j\). In summary, we have \((x_i, x_j) \notin EQ_S\) and \((x_i, x_j) \notin \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\) without loss of generality.

In order for \((x_i, x_j)\) to be in \(\mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\), we must have \((y_{j-1}, y_j) \in \mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}(EQ)|_{X_S}\) for each \(j (1 \leq j \leq n)\). Since \(EQ\) does not contain any pair \((y_{j-1}, y_j)\), the pairs must be added during the computation of \(\mathcal{E}_{\psi_T \land \varphi_j \land \varphi_k}\). Based on the first and the second items of the definition of \(\psi_T\), for each \(j (1 \leq j \leq n)\), at least one of the following two conditions holds:

- \(C_j\) contains \(x_i\) and \((x_i, x_j) \in EQ_S\), and
- \(C_j\) contains \(-x_i\) and \((x_i, x_j) \in EQ_S\).

Moreover, since \((x_i, x_j) \notin EQ_S\), we have no \(x_i\) such that \((x_i, x_j) \in EQ_S\) and \((x_i, x_j) \in EQ_S\). Hence, the following truth assignment \(\mu\) is well defined and satisfies \(F\):

- \(\mu(x_i) = \begin{cases} \text{TRUE} & \text{if } (x_i, x_j) \in EQ_S, \\ \text{FALSE} & \text{if } (x_i, x_j) \in EQ_S, \\ \text{don't care} & \text{otherwise.} \end{cases}\)

\(\square\)

The following theorem is immediate from the previous lemma:

**Theorem 4:** The absolute consistency problem is coNP-hard even if the source schema has no fd and the target schema has only one fd.

5.2 Case 2: One 1-Bounded Source fd, one 2-Bounded Target fd

We show that the absolute consistency problem is coNP-hard even if the source schema has only one 1-bounded fd and the target schema has only one 2-bounded fd. The proof is almost the same as the previous case. The difference is that we introduce an extra attribute \(K\) and use it as a key attribute for determining the values of \(x@X\), instead of using \(X\) themselves as a key. Precisely, given a CNF formula \(F = C_1 \land \cdots \land C_n\), we construct the following schema mapping \(M = \{(R_S, \Sigma_S), (R_T, \Sigma_T), (\varphi_S \rightarrow \psi_T)\}:

- \(R_S = (R_S[KX], K, A_1 \cdots A_m A_1 A_j)\),
- \(\Sigma_S = \{K \rightarrow X\}\),
- \(R_T = (R_T[KAY], Y, B_0 \cdots B_{n+1})\),
- \(\Sigma_T = \{K \rightarrow Y\}\),
- \(\varphi_S = R_S(w@K, x_1@A_1, \ldots, x_m@A_m, x_i@A_1, x_j@A_j)\), and
- \(\psi_T = \text{the conjunction of the following subformulas:}\)

1. \(R_T(w@K, x_i@A_i, y_i@B_i) \land R_T(w@K, x_i@A_i, y_i@B_i)\) for each \(C_j\) containing a positive literal \(x_i\),
2. \(R_T(w@K, x_i@A_i, y_i@B_i) \land R_T(w@K, x_i@A_i, y_i@B_i)\) for each \(C_j\) containing a negative literal \(-x_i\),
3. \(R_T(w@K, z@A_i, y_i@B_i) \land R_T(w@K, z@A_i, y_i@B_i)\), and
4. \(R_T(w@K, z@A_i, y_i@B_{n+1}) \land R_T(w@K, z@A_i, y_i@B_{n+1})\).
where _@ Y[y@ B_j] is the same notation as in the previous case.

In the same way as Lemma 9, we can prove that M is absolutely consistent if and only if F is unsatisfiable. Hence, we have the following theorem:

**Theorem 5:** The absolute consistency problem is coNP-hard even if the source schema has only one 1-bounded fd and the target schema has only one 2-bounded fd.

### 6. Related Work

At an earlier stage of the research on schema mappings, much of the effort has been spent on solving instance-dependent problems such as deciding the existence of solutions [10], [11], finding a universal solution and a core [10], [12], computing certain answers [10], [13], and so on. Nowadays, static analysis of schema mappings is extensively studied.

Fagin et al. [14] discussed three notions of equivalence between relational schema mappings. Gottlob et al. [15] proposed optimality criteria for sets of dependencies between the source and the target schemas, and presented rewrite rules to transform a given set of dependencies into an optimal one. Calvanese et al. [16] discussed simplification of schema mappings based on implication rather than equivalence. Arenas et al. [17] discussed query rewriting over canonical universal solutions and cores, and gave a tool for deciding non-rewritability.

As a topic closer to the absolute consistency problem, Marnette and Geerts [18] discussed the logical implication problem of relational schema mappings, i.e., the problem of deciding whether, for every pair of source and target instances specified by a given schema mapping, the target instance satisfies the target constraints whenever the source instance satisfies the source constraints. This problem resembles absolute consistency. However, for data exchange not to fail, the logical implication is too strong.

Another topic close to the absolute consistency problem is **terminating chase**. The chase algorithm can be used for computing a universal solution of a given source instance [10]. In general, however, it is undecidable whether for given source instance and dependencies, the chase algorithm on them terminates or not [19]. The topic of terminating chase is to find a large class of dependencies on which the chase algorithm terminates for any given source instance. Hence, the goal of the topic is to guarantee that the existence of a universal solution of a given source instance is decidable and a universal solution is computable if exists. On the other hand, absolute consistency guarantees that any given source instance has a solution. Note that in our setting, the existence of a solution implies the existence of a universal solution, because dependencies on the target schema are only equality-generating dependencies (Proposition 6.10 of [20]). By combining our contribution with terminating chase, we can guarantee both the existence and the computability of a (universal) solution of any given source instance. Onet [21] provided a good survey of terminating chase.

In our setting, the source schema has fds. However, in most of the research on relational schema mappings so far, source dependencies are assumed to be empty. This assumption would make the absolute consistency problem easier because we could find more easily a source instance that has no solution. So, we take account of source dependencies in this paper.

### 7. Conclusions

This paper has discussed the absolute consistency problem for relational schema mappings, where both of the source and the target schemas have fds. Under the assumption that mapping rules are defined by constant-free tgdts, we have shown the following results. First, the absolute consistency problem is in coNP. Secondly, it is solvable in polynomial time if the tgdts are full and there is a constant K such that every fd is K-bounded. Finally, it is coNP-hard even if the source schema has no fds and the target schema has only one fd; or the source schema has only one, 1-bounded fd and the target schema has only one, 2-bounded fd.

There still exists a slight gap between the tractability and the intractability results exhibited in this paper. To be specific, the upper and the lower bounds are not tight when the tgdts are not full, the source schema has no fds, and the target schema has only K-bounded fds for some constant K. Further elaboration is necessary for filling the gap.

Another direction of the future work is to extend the setting of this paper, e.g., admitting tgdts with constants, extending fds to general equality-generating dependencies, etc. Especially, constant-freeness is used for deriving at least Lemma 1 and Theorem 2, so we are conjecturing that such extension will make the absolute consistency problem much harder. We are also planning to incorporate the tractability result to our P2P data-sharing infrastructure [22] based on bidirectional transformations for relational databases [23].

### Acknowledgments

The authors are grateful to the anonymous reviewers for their valuable comments to improve the paper. This work is partially supported by the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (S) No. 17H06099 and by ROIS NII Open Collaborative Research 2017 (Grant Number 24).

### References

[1] T. Hayata, Y. Ishihara, and T. Fujiwara, “The absolute consistency problem of graph schema mappings with uniqueness constraints,” Proc. 9th Int. Conf. Advances in Databases, Knowledge, and Data Applications, pp.103–110, 2017.

[2] G. Karvounarakis, T.J. Green, Z.G. Ives, and V. Tannen, “Collaborative data sharing via update exchange and provenance,” ACM Trans.
Interoperability, pp.1–5, 2019.

M. Arenas and L. Libkin, “XML data exchange: Consistency and query answering,” J. ACM, vol.55, no.2, pp.1–72, 2008.

S. Amano, L. Libkin, and F. Murlak, “XML schema mappings,” Proc. Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.33–42, 2009.

M. Bojańczyk, Ł.A. Kołodziejezyk, and F. Murlak, “Solutions in XML data exchange,” Proc. 14th Int. Conf. Database Theory, pp.102–113, 2011.

H. Kuwada, K. Hashimoto, Y. Ishihara, and T. Fujiwara, “The consistency and absolute consistency problems of XML schema mappings between restricted DTDs,” World Wide Web, vol.18, no.5, pp.1443–1461, 2015.

Y. Ishihara, H. Kuwada, and T. Fujiwara, “The absolute consistency problem of XML schema mappings with data values between restricted DTDs,” Proc. 25th Int. Conf. Database and Expert Systems Applications, pp.317–327, 2014.

I. Boneva, J. Lozano, and S. Staworko, “Relational to RDF data exchange in presence of a shape expression schema,” Proc. 12th Alberto Mendelzon International Workshop on Foundations of Data Management, 2018.

D. Maier, A.O. Mendelzon, and Y. Sagiv, “Testing implications of data dependencies,” ACM Trans. Database Systems, vol.4, no.4, pp.455–469, 1979.

R. Fagin, P.G. Kolaitis, R.J. Miller, and L. Popa, “Data exchange: semantics and query answering,” Theoretical Computer Science, vol.336, no.1, pp.89–124, May 2005.

P.G. Kolaitis, J. Panttaja, and W.C. Tan, “The complexity of data exchange,” Proc. Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.30–39, June 2006.

R. Fagin, P.G. Kolaitis, and L. Popa, “Data exchange: getting to the core,” ACM Trans. Database Systems, vol.30, no.1, pp.174–210, March 2005.

L. Libkin, “Data exchange and incomplete information,” Proc. Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.60–69, June 2006.

R. Fagin, P.G. Kolaitis, A. Nash, and L. Popa, “Towards a theory of schema-mapping optimization,” Proc. Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.33–42, June 2008.

G. Gottlob, R. Pichler, and V. Savenkov, “Normalization and optimization of schema mappings,” Proc. VLDB Endowment, vol.2, no.1, pp.1102–1113, Aug. 2009.

D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Yardi, “On simplification of schema mappings,” J. Comput. Syst. Sci., vol.79, no.6, pp.816–834, Sept. 2013.

M. Arenas, P. Barceló, R. Fagin, and L. Libkin, “Solutions and query rewriting in data exchange,” Information and Computation, vol.228-229, pp.28–61, July 2013.

B. Marnette and F. Geerts, “Static analysis of schema-mappings ensuring oblivious termination,” Proc. 13th Int. Conf. Database Theory, pp.183–195, March 2010.

A. Deutsch, A. Nash, and J. Remmel, “The chase revisited,” Proc. Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.149–158, June 2008.

M. Arenas, P. Barceló, L. Libkin, and F. Murlak, Foundations of Data Exchange, Cambridge University Press, 2014.

A. Onet, “The chase procedure and its applications in data exchange,” Data Exchange, Integration, and Streams, Dagstuhl Follow-Ups, vol.5, pp.1–37, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

Y. Ishihara, H. Kato, K. Nakano, M. Onizuka, and Y. Sasaki, “Toward BX-based architecture for controlling and sharing distributed data,” Proc. Second Workshop on Software Foundations for Data Interoperability, pp.1–5, 2019.

V.D. Tran, H. Kato, and Z. Hu, “Programmable View Update Strategies on Relations,” Proc. VLDB Endowment, vol.13, no.5, pp.726–739, Jan. 2020.