The Absolute Consistency Problem of Graph Schema Mappings with Uniqueness Constraints

Takashi Hayata*, Yasunori Ishihara* and Toru Fujiwara*
*Graduate School of Information Science and Technology
Osaka University, Suita, Japan
Email: {t-hayata, ishihara, fujiwara}@ist.osaka-u.ac.jp

***Abstract*—A schema mapping is a formal representation of the correspondence between source and target data in a data exchange setting. Schema mappings have been extensively studied so far in relational and XML databases. However, in graph databases, they have not received much attention yet. A given schema mapping is said to be absolutely consistent if every source data instance has a corresponding target data instance. Absolute consistency is an important property because it guarantees that data exchange never fails for any source data instance. In this paper, we define schema mappings for graph databases with uniqueness constraints. Our graph databases consist of nodes, edges, and properties, where a property is a key-value pair and gives detailed information to nodes. A uniqueness constraint guarantees the uniqueness of specified properties in the whole graph database, and therefore, is useful for realizing the functionality of primary keys in graph databases. Then, in this paper, we propose five classes of graph schema mappings for which absolute consistency is decidable in polynomial time.**

Keywords–graph database; property; uniqueness constraint; schema mapping; absolute consistency.

I. INTRODUCTION

In recent years, graph-structured data has become pervasive. For example, route information of transportation and communication network, connection of people on social networking services and so on are often cited. These data originally have a graph structure, and it is natural to store and manipulate them on a database while keeping the graph structure. Because of these backgrounds, graph databases have attracted attention in recent years. Graph-structured data has a feature of being flexible. This means that we can flexibly change the graph structure. Figure 1 illustrates flight route map between airports, where airports are represented by nodes. Each node has a unique *node id* (1–6) and a *property* (a key-value pair such as $Airport : ORY$). Figure 2 is a graph which represents direct flight information between countries. We can obtain this graph by integrating nodes whose countries are the same. In this way, you may want to obtain abstract data rather than concrete data. Also, considering big data analysis, it may be possible to discover new features by extracting only data having certain characteristics. Schema mappings are the foundation of such data exchange.

A schema mapping represents the correspondence between source databases and target databases. It is useful for formalizing data exchange between systems with different schemas and schema evolution caused by system change. Schema mappings have been extensively studied in relational databases [1]–[3] and XML databases [3]–[5]. On the other hand, schema mappings for graph databases have not been actively studied yet. Schema mappings of graph databases without properties is discussed in [6]. A study of schema mappings from a relational schema to a graph schema is reported in [7].

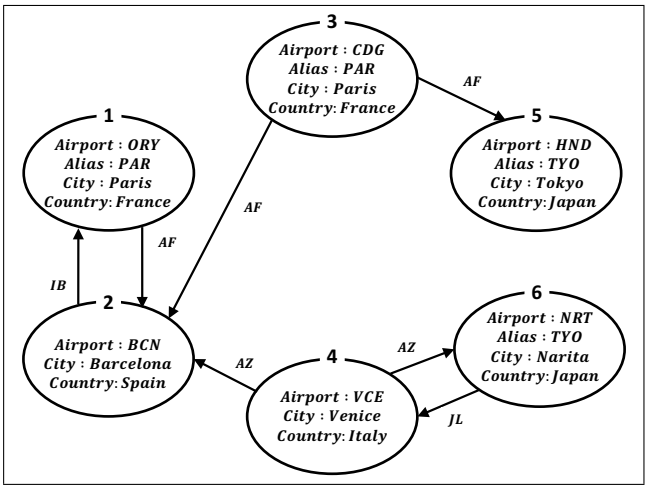

Figure 1. Flight route map between airports.

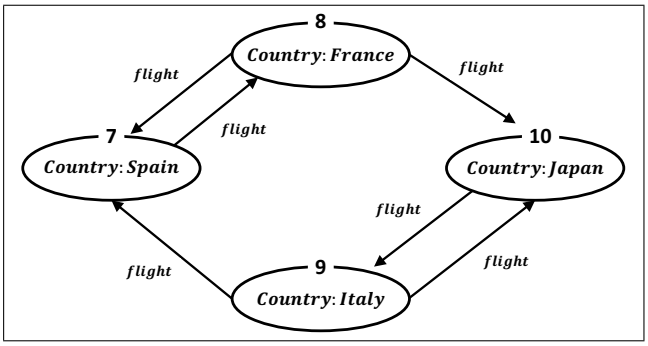

Figure 2. Flight information between countries.

In considering schema mappings, it is important to check the absolute consistency [4] of schema mappings. In XML databases, complexity of the absolute consistency problem has been studied [4], [8]. A schema mapping is absolutely consistent if any source database has a corresponding target database. Absolute consistency guarantees that data exchange based on the schema mapping never fails. Figure 3 illustrates flight information between IATA codes. We can compute this figure by applying a specific schema mapping to Figure 1 (see Example 5 for details). However, if there is a constraint that IATA codes are unique, this mapping does not satisfy the absolute consistency because a source graph in Figure 1 has

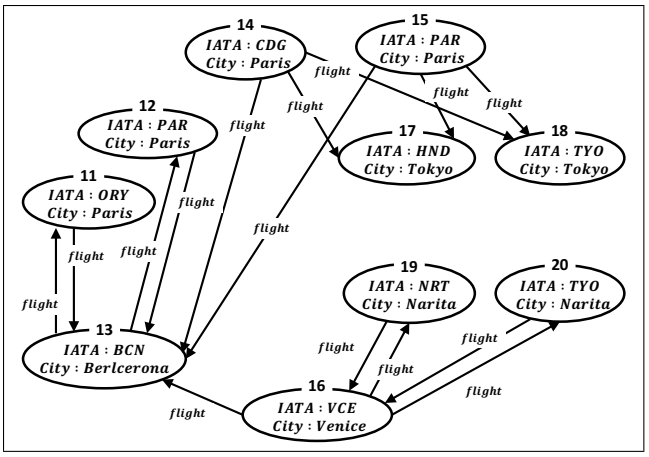

Figure 3. Flight information between IATA codes.

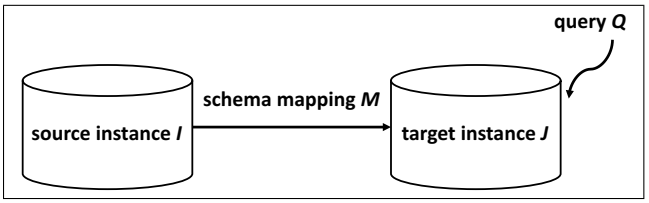

Figure 4. The the general setting of data exchange [3].

not a corresponding target graph.

The purpose of this paper is twofold. The first purpose is to define schema mappings for graph databases with properties. As stated above, the existing study [6] focuses on graph schema mappings without properties. However, properties enrich the expressive power of graph databases. In our study, we target graph databases such that nodes have properties. In addition, we introduce constraints on properties called *uniqueness constraints*, which are adopted by Neo4j [9]. A uniqueness constraint guarantees the uniqueness of a node with the specified property in the whole graph database. Then we define schema mappings for graph databases with properties and uniqueness constraints. The other purpose is to investigate classes of schema mappings for which absolute consistency is tractable. In this paper, we propose three classes whose member is always absolutely consistent. Then we propose two more classes for which absolute consistency is decidable in polynomial time.

II. Related Work

Data exchange is a problem of finding an instance of a target schema, given an instance of a source schema and a specification of the relationship between the source and the target. Such a target instance should correctly represent information from the source instance under the constraints imposed by the target schema, and it should allow one to evaluate queries on the target instance in a way that is semantically consistent with the source data. Figure 4 shows the image of the general setting of data exchange. In this figure, we have fixed source and target schemas, an instance I of the source schema, and a schema mapping M that specifies the relationship between the source and the target schemas. The goal is to construct an instance J of the target schema, based on I and M, and answer queries against the target data in a way consistent with the source data. Such a target instance is called a solution for the given source instance. As can be seen from Figure 4, a schema mapping is an important concept underlying data exchange.

In the relational scenario, schema mappings and data exchange have been studied in much literature (e.g., [1]–[3]). They define schema mappings and address the problem of materializing target instances and the problem of query answering.

In the XML scenario, schema mappings and data exchange have been studied in much literature (e.g., [3]–[5], [8]). Compared to the relational model, the tree model gives more opportunities for expressing structural properties of data even with simple queries based on patterns. On the other hand, schemas impose strong conditions on the structure of source and target instances, entering into complex interactions with source-to-target dependencies. These strong conditions cause consistency problem which is one of the central issues in XML data exchange. In [4], [8], they address the consistency and absolute consistency problems of XML schema mappings. A schema mapping is consistent if some source instance has a corresponding target instance. Also, a schema mapping is absolutely consistent if any source instance has a corresponding target instance. In XML schema mappings, it has been proved that the consistency problem is undecidable if we consider the comparison of data values [3]. Without considering the comparison of data values, it has been proved that the consistency problem is solvable in exponential time [3]. It has been proved that the absolute consistency problem for schema mappings based on downward navigation is decidable in EXPSPACE and NEXPTIME-hard [3], [4]. Some tractability results on consistency and absolute consistency problems between restricted schemas are reported in [8], [10].

In the graph scenario, schema mappings and data exchange have not been actively studied yet. In [6], they define schema mappings for graph data and address some problems of data exchange. Like relational and XML databases, they address the problem of materializing solutions and the problem of query answering. But they focus on graph schema mappings without properties and constraints such as uniqueness constraints, and hence schema mappings are always absolutely consistent in their setting.

Bidirectional transformations refer to a mechanism that maintains consistency through conversion between two sources of information. As one approach to bidirectional transformations, Triple Graph Grammars (TGGs) are well known, which define the correspondence between two different types of models in a declarative way [11]–[13]. A rule in a TGG is a triple of a source graph, a correspondence graph, and a target graph, and by applying rules to an axiom, both models are generated simultaneously. In TGGs, it is easy to define the correspondence between two different types of models, but it is not trivial to find the corresponding target instance when a source instance is given, as considered in data exchange. That is because TGGs grow a source graph, a correspondence graph, and a target graph in parallel, so it is first necessary to find how to apply rules to an axiom to obtain the source instance.

III. DEFINITIONS

A. Graph Databases

A *graph schema* S is a tuple (Σ, K), where

- Σ is a finite set of *edge labels*, and
- K is a finite set of *keys*.

Let fix a countable set $\mathcal{O}$ of *values*. A *property* (of a node) is a pair of a key in K and a value in $\mathcal{O}$. A *graph database* G over a graph schema $S = (\Sigma, K)$ is a tuple (N, E, f), where

- N is a finite set of *node ids*,
- $E \subseteq N \times \Sigma \times N$ is a finite set of *labeled edges*, and
- $f : N \times K \to \mathcal{O}$ is a partial function which binds node ids and properties.

When $f(n, k)$ is undefined, it is interpreted that node n does not have a property with key k, and we write $f(n, k) = \bot$. In this paper, it is assumed that edges have no properties.

Example 1: Let fix $\mathcal{O}$ as $\{ORY, BCN, CDG, VCE, HND, NRT, PAR, TYO, Paris, Barcelona, Venice, Tokyo, Narita, France, Spain, Italy, Japan\}$. Consider a graph schema $S = (\Sigma, K)$, where $\Sigma = \{IB, AF, AZ, JL\}$ and $K = \{Airport, Alias, City, Country\}$. Figure 1 illustrates a graph database $G = (N, E, f)$ over S, where

- $N = \{1, 2, 3, 4, 5, 6\}$,
- $E = \{(1, AF, 2), (2, IB, 1), (3, AF, 2), (3, AF, 5), (4, AZ, 2), (4, AZ, 6), (6, JL, 4)\}$, and
- $f(1, Airport) = ORY$, $f(1, Alias) = PAR$, $f(1, City) = Paris$, $f(1, Country) = France$, $f(2, Airport) = BCN$, $f(2, City) = Barcelona$, $f(2, Country) = Spain$, $f(3, Airport) = CDG$, $f(3, Alias) = PAR$, $f(3, City) = Paris$, $f(3, Country) = France$, and so on.

B. Uniqueness Constraitnts

A uniqueness constraint is specified as a set of keys and ensures that properties of the specified keys are unique in a graph database. In other words, there should not be different nodes with the same value for the specified keys.

Definition 1 (Uniqueness Constraints): A *uniqueness constraint* U over a graph schema $S = (\Sigma, K)$ is a subset of K. A graph database $G = (N, E, f)$ *satisfies* U if the following condition holds:

$$\forall n, n' \in N, \forall k \in U, \\ (f(n, k) \neq \bot) \wedge (f(n', k) \neq \bot) \wedge (f(n, k) = f(n', k)) \\ \Rightarrow n = n'.$$

Example 2: The graph database shown in Figure 3 does not satisfy $U = \{IATA\}$ because $f(12, IATA) = f(15, IATA)$ and $f(18, IATA) = f(20, IATA)$. On the other hand, the graph database shown in Figure 2 satisfies $U = \{Country\}$ because for each value $o \in \mathcal{O}$, there is at most one node n such that $f(n, Country) = o$.

C. Graph Patterns

Graph patterns represent a part of a graph database and can be used as queries for a given graph database. We also use graph patterns for describing mapping rules in schema mappings for graph databases.

Let fix a countable set $\mathcal{X}$ of variables representing values. A *graph pattern* π over a graph schema $S = (\Sigma, K)$ is a tuple (θ, μ, λ), where

- θ is a finite set of *node variables*,
- $\mu \subseteq \theta \times \mathrm{REG}(\Sigma) \times \theta$ is a finite set of *path patterns*, where $\mathrm{REG}(\Sigma)$ is the set of regular expressions over Σ, and
- $\lambda \subseteq \{v.k == x \mid v \in \theta, k \in K, x \in \mathcal{X}\}$ is a finite set of *equalities* such that if two equalities $(v.k == x)$ and $(v.k == y)$ are in λ, then x and y are the same variable.

We assume that no node variables are shared by different graph patterns, although variables in $\mathcal{X}$ are shared in general, as in mapping rules defined later.

We define the semantics of a graph pattern π in terms of homomorphisms. Let $g : \mathcal{X} \to \mathcal{O}$ be a mapping that assigns a value to each variable. Let $G = (N, E, f)$ be a graph database. A graph pattern $\pi = (\theta, \mu, \lambda)$ is *modeled* by (G, h, g) if homomorphism $h : \pi \to G$ satisfies the following conditions:

1) for each node variable $v \in \theta$, $h(v) \in N$,
2) for each path pattern $(v, L, u) \in \mu$, there is a path from $h(v)$ to $h(u)$ in G such that the edge label sequence along the path matches the regular expression L, and
3) for each equality $(v.k == x) \in \lambda$, $f(h(v), k) = g(x)$.

We write $(G, h, g) \models \pi$ if π is modeled by (G, h, g). The semantics of π under S and g is defined as follows:

$$[[\pi]]_{S,g} = \{G \text{ over } S \mid (G, h, g) \models \pi \text{ for some } h\}.$$

Example 3: Let fix $\mathcal{X}$ as $\{x\}$ and $\mathcal{O}$ as $\{A, B\}$. Let $V = \{s, t, u, v\}$ be a finite set of node variables. Consider a graph schema $S = (\Sigma, K)$, where $\Sigma = \{a, b\}$ and $K = \{k\}$. Figure 5 illustrates a graph pattern $\pi = (\theta, \mu, \lambda)$ over S, where

- $\theta = \{s, t, u, v\}$,
- $\mu = \{(s, a, t), (u, b, v)\}$, and
- $\lambda = \{(s.k == x), (t.k == x)\}$.

Figure 6 illustrates a graph database $G = (N, E, f)$ over S, where

- $N = \{1, 2, 3, 4\}$,
- $E = \{(1, a, 2), (3, a, 4), (3, b, 1), (2, c, 4)\}$, and
- $f(1, k) = A$, $f(2, k) = A$, $f(3, k) = A$, and $f(4, k) = B$.

As shown in Figure 7, a homomorphism h such that $(G, h, g) \models \pi$ maps s into 1, t into 2, u into 3, and v into 1. We can confirm that the graph pattern π represents a part of the graph database G.

 ISBN: 978-1-61208-558-6

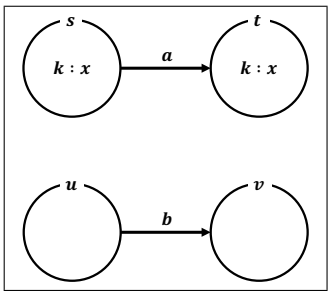

Figure 5. A graph pattern π.

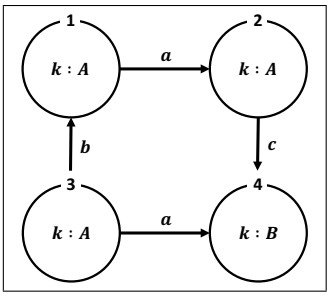

Figure 6. A graph database G.

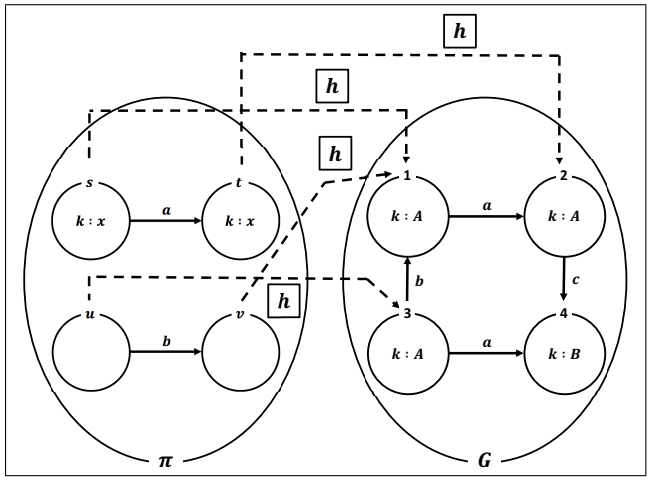

Figure 7. A homomorphism $h : \pi \to G$.

D. Schema Mappings

Schema mappings represent the correspondence between source databases and target databases. We define schema mappings for graph databases with properties from now.

Let S_S be a source graph schema and S_T a target graph schema. Let U_S and U_T be uniqueness constraints over S_S and S_T, respectively. A schema mapping M from S_S to S_T is a tuple $((S_S, U_S), (S_T, U_T), \Delta)$, where Δ is a finite set of *mapping rules* of the form $\pi_S \to \pi_T$. Here, $\pi_S = (\theta_S, \mu_S, \lambda_S)$ and $\pi_T = (\theta_T, \mu_T, \lambda_T)$ are graph patterns over S_S and S_T, respectively. Each λ_S must be *linear* with respect to variables in $\mathcal{X}$, that is, each variable in $\mathcal{X}$ appears at most once in λ_S.

Definition 2 (Solutions): Let $M = ((S_S, U_S), (S_T, U_T), \Delta)$ be a schema mapping. Let G_S be a graph database over S_S satisfying U_S, and G_T a graph database over S_T satisfying U_T. A pair (G_S, G_T) *satisfies* M if the following condition holds: For each $(\pi_S \to \pi_T) \in \Delta$ and for any $g_S : \mathcal{X} \to \mathcal{O}$, there exists $g_T : \mathcal{X} \to \mathcal{O}$ such that

1) $g_S(x) = g_T(x)$ for each variable $x \in \mathcal{X}$ appearing in both π_S and π_T; and
2) if $G_S \in [[\pi_S]]_{S_S, g_S}$, then $G_T \in [[\pi_T]]_{S_T, g_T}$.

If a pair (G_S, G_T) satisfies M, we write $(G_S, G_T) \models M$, and G_T is called a *solution* for G_S under M. Let $\mathrm{Sol}_M(G_S)$ denote the set of solutions for G_S under M.

Without loss of generality, we assume that no variables in $\mathcal{X}$ are shared by different mapping rules.

Example 4: Let $S_S = (\Sigma_S, K_S)$ and $S_T = (\Sigma_T, K_T)$ be graph schemas such that

- $\Sigma_S = \{IB, AF, AZ, JL\}$,
- $K_S = \{Airport, Alias, City, Country\}$,
- $\Sigma_T = \{flight\}$, and
- $K_T = \{Country\}$.

Also, let

- $U_S = \{Airport\}$,
- $U_T = \{Country\}$, and
- $\Delta = \{\pi_S \to \pi_T\}$,

where $\pi_S = (\theta_S, \mu_S, \lambda_S)$ and $\pi_T = (\theta_T, \mu_T, \lambda_T)$ are graph patterns such that

- $\theta_S = \{v_1, v_2\}$,
- $\mu_S = \{(v_1, (IB|AF|AZ|JL), v_2)\}$,
- $\lambda_S = \{(v_1.Country == x), (v_2.Country == y)\}$,
- $\theta_T = \{u_1, u_2\}$,
- $\mu_T = \{(u_1, flight, u_2)\}$, and
- $\lambda_T = \{(u_1.Country == x), (u_2.Country == y)\}$.

Then, the graph database in Figure 2 is a solution for that in Figure 1 under $M = ((S_S, U_S), (S_T, U_T), \Delta)$.

Definition 3 (Absolute Consistency): A schema mapping $M = ((S_S, U_S), (S_T, U_T), \Delta)$ is *absolutely consistent* if $\mathrm{Sol}_M(G_S) \neq \emptyset$ for every graph database G_S over S_S satisfying U_S.

Example 5: Let $M = ((S_S, U_S), (S_T, U_T), \Delta)$ be a schema mapping, where $S_S = (\Sigma_S, K_S)$ and $S_T = (\Sigma_T, K_T)$ be graph schemas such that

- $\Sigma_S = \{IB, AF, AL, JL\}$,
- $K_S = \{Airport, Alias, City, Country\}$,
- $\Sigma_T = \{flight\}$, and
- $K_T = \{IATA, City\}$.

Also, let

- $U_S = \{Airport\}$,
- $U_T = \{IATA\}$, and
- $\Delta = \{\pi_{S_1} \to \pi_{T_1}, \pi_{S_2} \to \pi_{T_2}, \pi_{S_3} \to \pi_{T_3}, \pi_{S_4} \to \pi_{T_4}\}$,

where $\pi_{S_1} = (\theta_{S_1}, \mu_{S_1}, \lambda_{S_1})$, $\pi_{T_1} = (\theta_{T_1}, \mu_{T_1}, \lambda_{T_1})$, $\pi_{S_2} = (\theta_{S_2}, \mu_{S_2}, \lambda_{S_2})$, $\pi_{T_2} = (\theta_{T_2}, \mu_{T_2}, \lambda_{T_2})$, $\pi_{S_3} = (\theta_{S_3}, \mu_{S_3}, \lambda_{S_3})$, $\pi_{T_3} = (\theta_{T_3}, \mu_{T_3}, \lambda_{T_3})$, $\pi_{S_4} = (\theta_{S_4}, \mu_{S_4}, \lambda_{S_4})$, and $\pi_{T_4} = (\theta_{T_4}, \mu_{T_4}, \lambda_{T_4})$ are graph patterns such that

- $\theta_{S_1} = \{u_1, u_2\}$,
- $\mu_{S_1} = \{(u_1, (IB|AF|AZ|JL), u_2)\}$,
- $\lambda_{S_1} = \{(u_1.City == x_1), (u_1.Airport == y_1), (u_2.City == z_1), (u_2.Airport == w_1)\}$,
- $\theta_{T_1} = \{v_1, v_2\}$,
- $\mu_{T_1} = \{(v_1, flight, v_2)\}$,
- $\lambda_{T_1} = \{(v_1.City == x_1), (v_1.IATA == y_1), (v_2.City == z_1), (v_2.IATA == w_1)\}$,
- $\theta_{S_2} = \{u_3, u_4\}$,
- $\mu_{S_2} = \{(u_3, (IB|AF|AZ|JL), u_4)\}$,

- $\lambda_{S_2} = \{(u_3.City == x_2), (u_3.Airport == y_2), (u_4.City == z_2), (u_4.Alias == w_2)\}$,
- $\theta_{T_2} = \{v_3, v_4\}$,
- $\mu_{T_2} = \{(v_3, flight, v_4)\}$,
- $\lambda_{T_2} = \{(v_3.City == x_2), (v_3.IATA == y_2), (v_4.City == z_2), (v_4.IATA == w_2)\}$,
- $\theta_{S_3} = \{u_5, u_6\}$,
- $\mu_{S_3} = \{(u_5, (IB|AF|AZ|JL), u_6)\}$,
- $\lambda_{S_3} = \{(u_5.City == x_3), (u_5.Alias == y_3), (u_6.City == z_3), (u_6.Airport == w_3)\}$,
- $\theta_{T_3} = \{v_5, v_6\}$,
- $\mu_{T_3} = \{(v_5, flight, v_6)\}$,
- $\lambda_{T_3} = \{(v_5.City == x_3), (v_5.IATA == y_3), (v_6.City == z_3), (v_6.IATA == w_3)\}$,
- $\theta_{S_4} = \{u_7, u_8\}$,
- $\mu_{S_4} = \{(u_7, (IB|AF|AZ|JL), u_8)\}$,
- $\lambda_{S_4} = \{(u_7.City == x_4), (u_7.Alias == y_4), (u_8.City == z_4), (u_8.Alias == w_4)\}$,
- $\theta_{T_4} = \{v_7, v_8\}$,
- $\mu_{T_4} = \{(v_7, flight, v_8)\}$, and
- $\lambda_{T_4} = \{(v_7.City == x_4), (v_7.IATA == y_4), (v_8.City == z_4), (v_8.IATA == w_4)\}$.

Given a graph in Figure 1 as a source instance and this schema mapping, we can compute a graph in Figure 3, if we do not consider U_T. However, taking into account U_T, the graph in Figure 3 does not satisfy U_T because $f(18, IATA) = f(20, IATA)$. That is, this schema mapping M is not absolutely consistent.

The size $|M|$ of a schema mapping $M = ((S_S, U_S), (S_T, U_T), \Delta)$ is the sum of the sizes of S_S, U_S, S_T, U_T, and Δ. The size of a schema $S = (\Sigma, K)$ is the sum of the numbers of elements in Σ and K. The size of a uniqueness constraint U is the number of elements in U. The size of Δ is the sum of the sizes of graph patterns appearing in Δ. The size of a graph pattern $\pi = (\theta, \mu, \lambda)$ is the sum of the numbers of elements in θ, μ, and λ.

IV. Tractable Classes of Schema Mappings for Absolute Consistency

Let $M = ((S_S, U_S), (S_T, U_T), \Delta)$ be a schema mapping, where $S_S = (\Sigma_S, K_S)$ and $S_T = (\Sigma_T, K_T)$. In this section, we show that the absolute consistency of M is decidable in polynomial time if M belongs to one of the following five classes:

1) K_S is empty;
2) U_T is empty;
3) K_T is a singleton;
4) U_S is empty; and
5) Δ is a singleton and the mapping rule is projecting, where $\pi_S \to \pi_T$ is *projecting* if the set of variables for values appearing in π_T is a subset of those appearing in π_S.

For example, the schema mapping shown in Example 4 belongs to not only the third class but also the fifth class because Δ is a singleton and the set $\{x, y\}$ of variables appearing in π_T is the same as that in π_S.

A. Three Easy Classes

In this section, we show that M is always absolutely consistent if M belongs to one of the first three classes. First, we show the following lemma, which states the existence of a graph database G_0 that matches any graph pattern.

Lemma 1: Let $S = (\Sigma, K)$ be a graph schema and U be a uniqueness constraint over S. There is a graph database G_0 over S satisfying U such that for any graph pattern π over S, there is $g : \mathcal{X} \to \mathcal{O}$ such that $G_0 \in [[\pi]]_{S,g}$.

Proof: Define $G_0 = (N_0, E_0, f_0)$ as follows:

- $N_0 = \{n\}$,
- $E_0 = \{(n, a, n) \mid a \in \Sigma\}$, and
- $f_0(n, k)$ is the same value for all $k \in K$.

It is easy to see that G_0 satisfies the lemma. ■

Theorem 1: M is absolutely consistent if K_S is empty.

Proof: We show that G_0 introduced in the proof of Lemma 1 is a solution for any G_S over S. Suppose that $G_S \in [[\pi_S]]_{S_S,g_S}$ for some $(\pi_S \to \pi_T) \in \Delta$ and $g_S : \mathcal{X} \to \mathcal{O}$. Since K_S is empty, λ_S is also empty, and hence π_S has no variable in $\mathcal{X}$. Therefore, the first condition of Definition 2 holds for any $g_T : \mathcal{X} \to \mathcal{O}$. By Lemma 1, we have $G_0 \in [[\pi_T]]_{S_T,g}$ for some g. ■

The next lemma says that each graph pattern is matched by a graph database with the "same shape."

Lemma 2: Let $S = (\Sigma, K)$ be a graph schema and $\pi = (\theta, \mu, \lambda)$ be a graph pattern over S. For any $g : \mathcal{X} \to \mathcal{O}$, $[[\pi]]_{S,g}$ is not empty.

Proof: Consider the following graph database $G_\pi = (N_\pi, E_\pi, f_\pi)$:

- N_π contains θ, where node variables are regarded as node ids,
- for each $(v_0, L, v_n) \in \mu$, E_π contains $(v_0, a_1, v_1), \ldots, (v_{n-1}, a_n, v_n)$ such that $a_1 \cdots a_n$ matches L and $v_1, \ldots, v_{n-1}$ are not in θ, and
- $f_\pi(v, k) = x$ for each $(v.k == x) \in \lambda$, where variable x is regarded as a value.

It is easy to see that such G_π is well defined and $G_\pi \in [[\pi]]_{S,id}$, where $id(x) = x$ for every variable x.

Now, consider a graph database $G_{\pi,g}$ obtained by replacing each value x in G_π with $g(x)$. Then, $G_{\pi,g} \in [[\pi]]_{S,g}$. ■

Theorem 2: M is absolutely consistent if U_T is empty.

Proof: Let G_S be a source graph database over S_S satisfying U_S. Consider a target graph database G_T over S_T that contains G_{π_T,g_S} as its subgraph for all $g_S : \mathcal{X} \to \mathcal{O}$ such that $G_S \in [[\pi_S]]_{S_S,g_S}$ for some $(\pi_S \to \pi_T) \in \Delta$, where G_{π_T,g_S} is the same graph database as in the proof of Lemma 2. Such G_T always exists since U_T is empty. Also, G_T is a solution for G_S because $G_{\pi_T,g_S} \in [[\pi_T]]_{S_T,g_S}$. ■

Theorem 3: M is absolutely consistent if K_T is a singleton.

Proof: Consider again the target graph database G_T in the proof of Theorem 2. Since $U_T \subseteq K_T$, we have $U_T = \emptyset$ or $U_T = K_T$. In either case, G_T satisfies U_T. Actually, if U_T and K_T are the same singleton sets, each node of G_T has at most one property, and the nodes with the same property can be an identical node. ■

B. No Uniqueness Constraints for Source Databases

Now, we consider the case where $U_S = \emptyset$. In this case, the variables appearing in the graph patterns for source databases can have independent, arbitrary values. Hence, we can statically analyze M based on abstract interpretation, where the variable names represent abstract values.

Let X_S and X_T be the sets of variables for values appearing in π_S and π_T, respectively, for some mapping rule $(\pi_S \to \pi_T) \in \Delta$. Let $X = X_S \cup X_T$. Define

$$\Lambda_T \quad = \bigcup_{(\pi_S \to (\theta_T, \mu_T, \lambda_T)) \in \Delta} \lambda_T.$$

Define $\sim$ as the least equivalence relation on X satisfying the following condition: $x \sim y$ if $v.k == x$ and $v.k' == y$ are in Λ_T for some $k \in U_T$, $k' \in K_T$, and $z \in X_S$ such that $x \sim z$.

Roughly speaking, the condition in the following theorem says that if a target node has a key k in U_T and the value of k is dependent on a value in the source database, then all the other keys of the node must be dependent on the same value.

Theorem 4: $M = ((S_S, \emptyset), (S_T, U_T), \Delta)$ is absolutely consistent if and only if there are no distinct variables x, $y \in X_S$ such that $x \sim y$.

Proof: (Only if part.) Suppose that $x \sim y$ for some distinct variables x, $y \in X_S$. It can be shown that $(v.k == z)$ and $(v.k' == y)$ in Λ_T such that $x \sim z$ and $k \in U_T$.

For the simplest case, let us consider the case where $(v.k == x)$ and $(v.k' == y)$ in Λ_T for some $k \in U_T$. Consider two mappings g_1 and g_2 such that $g_1(x) = g_2(x)$ but $g_1(y) \neq g_2(y)$. There is a source database G_S such that $G_S \in [[\pi_S]]_{S_S, g_1}$ and $G_S \in [[\pi_S]]_{S_S, g_2}$ because the uniqueness constraints for source databases is empty. Now, a solution $G_T = (N_T, E_T, f_T)$ for G_S must satisfy the following conditions:

- G_T has a node n_1 such that $f_T(n_1, k) = g_1(x)$ and $f_T(n_1, k') = g_1(y)$; and
- G_T has a node n_2 such that $f_T(n_2, k) = g_2(x)$ and $f_T(n_2, k') = g_2(y)$.

Since $k \in U_T$, n_1 and n_2 must be the same node, but that contradicts the assumption that $g_1(y) \neq g_2(y)$.

The general cases can be shown in a similar way.

(If part.) Let G_S be a source graph database over S_S. Let $G_T = (N_T, E_T, f_T)$ be a target graph database containing all $G_{\pi_T, g_{\pi_S}, g_S}$ defined below, such that $G_S \in [[\pi_S]]_{S_S, g_S}$ for some $(\pi_S \to \pi_T) \in \Delta$ and $g_S : X_S \to \mathcal{O}$. We will show that G_T is a solution for G_S and U_T can be satisfied by G_T.

First, we define $G_{\pi_T, g_{\pi_S}, g_S}$. Suppose that $G_S \in [[\pi_S]]_{S_S, g_S}$ for some $(\pi_S \to \pi_T) \in \Delta$ and $g_S : X_S \to \mathcal{O}$. For each of such pairs of π_S and g_S, we choose $g_{\pi_S, g_S} : X_T \to \mathcal{O}$ so that

$$g_{\pi_S, g_S}(x) = \begin{cases} g_S(y) & \text{if } x \sim y \text{ for some } y \in X_S, \\ o_{\pi_S, g_S, x} & \text{otherwise,} \end{cases}$$

where $o_{\pi_S, g_S, x}$ is a unique, distinct value in $\mathcal{O}$ determined by π_S, g_S, and x. g_{π_S, g_S} is well defined because each equivalence class derived by $\sim$ has at most one variable in X_S. Now, $G_{\pi_T, g_{\pi_S}, g_S} \in [[\pi_T]]_{S_T, g_{\pi_S}, g_S}$ is the graph database introduced in the proof of Lemma 2.

It is obvious that G_T is a solution for G_S because G_T contains all $G_{\pi_T, g_{\pi_S}, g_S} \in [[\pi_T]]_{S_T, g_{\pi_S}, g_S}$. Let $n \in N_T$ be a node id of some $G_{\pi_T, g_{\pi_S}, g_S}$. Suppose that $f_T(n, k) = o$ for some key $k \in U_T$ and value $o \in \mathcal{O}$. There must be an equality $v.k == x$ in π_T. We consider the following two cases:

1) If $x \sim y$ for some $y \in X_S$, then by the definitions of $\sim$ and g_{π_S, g_S}, we have $f_T(n, k') = o$ for any $k' \in K_T$ such that $f_T(n, k')$ is defined. Therefore, all such nodes $n \in N_T$ with $f_T(n, k) = o$ can be an identical node.
2) If there is no $y \in X_S$ such that $x \sim y$, then o must be a unique value in G_T, hence the existence of n does not violate U_T.

In summary, U_T can be satisfied by G_T. ■

The equivalence relation $\sim$ can be computed in $O(|M|^3)$ time. Hence we have the following theorem:

Theorem 5: The absolute consistency of M is decidable in polynomial time if U_S is empty.

C. A Single Projecting Rule

Now, we consider the case where $\Delta = \{\pi_S \to \pi_T\}$ and $\pi_S \to \pi_T$ is projecting. Let X be the set of variables for values appearing in π_S. Since U_S is not empty, the values of the variables in X are not independent. For example, suppose that π_S has the following equalities: $v.k == x$, $u.k == y$, $v.k' == z$, and $u.k' == w$. If $k \in U_S$, equality between x and y causes equality between z and w. To capture this phenomenon, below we introduce a function $\mathcal{E}_{\pi_S, U_S}(EQ)$ which returns, for a given set of equalities on X, all the resultant equalities caused by π and U.

Formally, let $\pi = (\theta, \mu, \lambda)$ be a graph pattern with variables in X, and $U \subseteq K$ be a uniqueness constraint. For $EQ \subseteq X \times X$, define $\mathcal{E}_{\pi, U}(EQ) \subseteq X \times X$ as the least equivalence relation such that

1) $EQ \subseteq \mathcal{E}_{\pi, U}(EQ)$, and
2) for any pair $(x, y) \in \mathcal{E}_{\pi, U}(EQ)$ and any node variables v, $u \in \theta$, if $(v.k == x), (u.k == y) \in \lambda$ for some key $k \in U$, then $(z, w) \in \mathcal{E}_{\pi, U}(EQ)$ for every $z, w \in X$ such that $(v.k' == z), (u.k' == w) \in \lambda$ for some $k' \in K$.

Let EQ_g denote the equivalence relation on variables induced by $g : X \to \mathcal{O}$, 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 $\mathcal{E}_{\pi, U}$ if and only if there are G, h, and g such that $(G, h, g) \models \pi$ and G satisfies U.

Lemma 3: Suppose that $(G, h, g) \models \pi$ and G satisfies a uniqueness constraint U. Then, $EQ_g = \mathcal{E}_{\pi, U}(EQ_g)$.

Proof: Since $EQ_g \subseteq \mathcal{E}_{\pi, U}(EQ_g)$ by definition, it suffices to show that $g(z) = g(w)$ for any pair $(z, w) \in \mathcal{E}_{\pi, U}(EQ_g)$. Consider the shortest proof $P_{(z,w)}$ of the membership of an arbitrary $(z, w) \in \mathcal{E}_{\pi, U}(EQ_g)$, i.e., an application sequence of reflexivity, symmetry, transitivity, and the two rules of the definition of $\mathcal{E}_{\pi, U}$. 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) is in $\mathcal{E}_{\pi, U}(EQ_g)$ by reflexivity, then z and w are the same variable. If (z, w) is in $\mathcal{E}_{\pi, U}(EQ_g)$ by the first rule of the definition of $\mathcal{E}_{\pi, U}$, 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 $\mathcal{E}_{\pi, U}(EQ_g)$ by symmetry, we must already have $(w, z) \in$

$\mathcal{E}_{\pi,U}(EQ_g)$. Then, we have $g(z) = g(w)$ by the inductive hypothesis. If (z, w) is in $\mathcal{E}_{\pi,U}(EQ_g)$ by transitivity, we can show that $g(z) = g(w)$ in a similar way to the symmetry case. Now, suppose that (z, w) is in $\mathcal{E}_{\pi,U}(EQ_g)$ by the second rule of the definition of $\mathcal{E}_{\pi,U}$. Then, there are some pair $(x, y) \in \mathcal{E}_{\pi,U}(EQ_g)$, node variables $v, u \in \theta$, and keys $k \in U$ and $k' \in K$ such that $(v.k == x)$, $(u.k == y)$, $(v.k' == z)$, $(u.k' == w) \in \lambda$. Since $(G, h, g) \models \pi$, we have $f(h(v), k) = g(x)$, $f(h(u), k) = g(y)$, $f(h(v), k') = g(z)$, and $f(h(u), k') = g(w)$, where $G = (N, E, f)$. It holds that $g(x) = g(y)$ by the inductive hypothesis. Since G satisfies uniqueness constraint U and U contains k, we have $h(v) = h(u)$. Hence, $g(z) = f(h(v), k') = f(h(u), k') = g(w)$. ■

Lemma 4: Suppose that $EQ = \mathcal{E}_{\pi,U}(EQ)$. Then, there are G, h, and g such that G satisfies the uniqueness constraint U, $(G, h, g) \models \pi$, and $EQ = EQ_g$.

Proof: Suppose that $EQ = \mathcal{E}_{\pi,U}(EQ)$, where $\pi = (\theta, \mu, \lambda)$. Choose an arbitrary $g : X \to \mathcal{O}$ such that $EQ_g = EQ$.

Let θ/EQ be the finest equivalence classes of node variables such that if there are $(x, y) \in EQ$ and $k \in U$ such that $(v.k == x), (u.k == y) \in \lambda$, then v and u are in the same equivalence class in θ/EQ. Let $[v]$ denote the equivalence class which v belongs to.

Similarly to G_π in the proof of Lemma 2, consider a graph database $G = (N, E, f)$ such that

- N contains θ/EQ, where the equivalence classes are regarded as node ids,
- for each $(v_0, L, v_k) \in \mu$, E contains $(n_0, a_1, n_1), \ldots, (n_{k-1}, a_k, n_k)$ such that $a_1 \cdots a_k$ matches L, $n_0 = [v_0]$, $n_k = [v_k]$, and $v_1, \ldots, v_{k-1}$ are not in θ/EQ, and
- $f([v], k') = g(x)$ for each $(v.k' == x) \in \lambda$.

To see that G is well defined, consider the shortest proof $P_{(v,u)}$ of that $[v] = [u]$, i.e., an application sequence of reflexivity, symmetry, transitivity, and the definition of θ/EQ. We show $f([v], k') = f([u], k')$ for all $k' \in K$ by the induction on the length of $P_{(v,u)}$.

For the basis, there are two cases. If $[v] = [u]$ by reflexivity, then $v = u$, and hence $f([v], k') = f([u], k')$. If $[v] = [u]$ by the definition of θ/EQ, there are $(x, y) \in EQ$ and $k \in U$ such that $(v.k == x), (u.k == y) \in \lambda$. Since EQ is a fixpoint of $\mathcal{E}_{\pi,U}$, we have $(z, w) \in EQ$ for every $z, w \in X$ such that $(v.k' == z), (u.k' == w) \in \lambda$, by the second rule of the definition of $\mathcal{E}_{\pi,U}$. Hence, $f([v], k') = g(z) = g(w) = f([u], k')$.

The induction step is trivial. If $[v] = [u]$ by symmetry, then we have $[u] = [v]$, and by the inductive hypothesis, $f([u], k') = f([v], k')$ for all $k' \in K$. The case of transitivity can be shown in a similar way.

Now, we have to show that G satisfies U. Assume contrarily that G does not satisfy U. There would be $v, u \in \theta$ and $k \in U$ such that $[v] \neq [u]$ and $f([v], k) = f([u], k)$. Hence, there would be $v' \in [v]$, $u' \in [u]$, $x, y \in X$ such that $(v'.k == x), (u'.k == y) \in \lambda$ and $(x, y) \in EQ$. However, by the definition of θ/EQ, we must have $[v'] = [u']$. This is a contradiction. ■

Theorem 6: Let $M = ((S_S, U_S), (S_T, U_T), \{\pi_S \to \pi_T\})$ be a schema mapping such that $\pi_S \to \pi_T$ is projecting. M is absolutely consistent if and only if every fixpoint of $\mathcal{E}_{\pi_S,U_S}$ is also a fixpoint of $\mathcal{E}_{\pi_T,U_T}$.

Proof: Immediate from Lemmas 3 and 4 since $\pi_S \to \pi_T$ is projecting. ■

In what follows, we show that the condition in Theorem 6 can be checked in polynomial time. The condition seems to require the computation of $\mathcal{E}_{\pi_S,U_S}$ and $\mathcal{E}_{\pi_T,U_T}$ for exponentially many EQs. Surprisingly, by the following two lemmas, such computation can be reduced to the computation for only EQs that are singletons.

Lemma 5: The followings are equivalent:

1) Every fixpoint of $\mathcal{E}_{\pi_S,U_S}$ is also a fixpoint of $\mathcal{E}_{\pi_T,U_T}$.
2) $\mathcal{E}_{\pi_T,U_T}(EQ) \subseteq \mathcal{E}_{\pi_S,U_S}(EQ)$ for each subset $EQ \subseteq X \times X$.

Proof: (1 $\Rightarrow$ 2) Let $EQ \subseteq X \times X$. Let $EQ_S = \mathcal{E}_{\pi_S,U_S}(EQ)$ and $EQ_T = \mathcal{E}_{\pi_T,U_T}(EQ)$. Since EQ_S is a fixpoint of $\mathcal{E}_{\pi_S,U_S}$, it is also a fixpoint of $\mathcal{E}_{\pi_T,U_T}$, i.e., $EQ_S = \mathcal{E}_{\pi_T,U_T}(EQ_S)$. Since EQ is a subset of EQ_S, we have $EQ_T = \mathcal{E}_{\pi_T,U_T}(EQ) \subseteq \mathcal{E}_{\pi_T,U_T}(EQ_S) = EQ_S$.

(2 $\Rightarrow$ 1) Let EQ be a fixpoint of $\mathcal{E}_{\pi_S,U_S}$. Then, $\mathcal{E}_{\pi_T,U_T}(EQ) \subseteq \mathcal{E}_{\pi_S,U_S}(EQ) = EQ$. On the other hand, by the definition of $\mathcal{E}_{\pi_T,U_T}(EQ)$, we have $EQ \subseteq \mathcal{E}_{\pi_T,U_T}(EQ)$. Hence $EQ = \mathcal{E}_{\pi_T,U_T}(EQ)$. ■

Lemma 6: The followings are equivalent:

1) $\mathcal{E}_{\pi_T,U_T}(EQ) \subseteq \mathcal{E}_{\pi_S,U_S}(EQ)$ for each subset $EQ \subseteq X \times X$.
2) $\mathcal{E}_{\pi_T,U_T}(\{(x, y)\}) \subseteq \mathcal{E}_{\pi_S,U_S}(\{(x, y)\})$ for each pair $(x, y) \in X \times X$.

Proof: Let $EQ_S = \mathcal{E}_{\pi_S,U_S}(EQ)$ and $EQ_T = \mathcal{E}_{\pi_T,U_T}(EQ)$. The part (1 $\Rightarrow$ 2) is obvious. To show the part (2 $\Rightarrow$ 1), consider the shortest proof $P_{(z,w)}$ 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 $\mathcal{E}_{\pi,U}$. We show $(z, w) \in EQ_S$ by the induction on the length of $P_{(z,w)}$.

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 $\mathcal{E}_{\pi,U}$, 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 $\mathcal{E}_{\pi,U}$. Then, we have $(z, w) \in \mathcal{E}_{\pi_T,U_T}(\{(x, y)\})$. By the assumption that $\mathcal{E}_{\pi_T,U_T}(\{(x, y)\}) \subseteq \mathcal{E}_{\pi_S,U_S}(\{(x, y)\})$, (z, w) is also in $\mathcal{E}_{\pi_S,U_S}(\{(x, y)\})$. Hence, $(z, w) \in EQ_S$ since $(x, y) \in EQ_S$ by the inductive hypothesis. ■

$\mathcal{E}_{\pi,U}(\{(x, y)\})$ can be computed in $O(|M|^6)$ time (by a very naive algorithm). So, the second condition of Lemma 6 can be checked in $O(|M|^8)$ time.

Theorem 7: The absolute consistency of M is decidable in polynomial time if $\Delta = \{\pi_S \to \pi_T\}$ is a singleton and $\pi_S \to \pi_T$ is projecting.

V. CONCLUSION

In this paper, we have defined schema mappings for graph databases with properties. In considering graph databases with properties, we have introduced uniqueness constraints, which are constraints on properties. Then, we have proposed five classes of schema mappings for which absolute consistency is decidable in polynomial time.

There still are remaining tasks. We are going to examine the complexity of the absolute consistency problem for schema mappings not belonging to the five classes because these classes seem restrictive from the practical point of view. Also, we plan to implement a program to check the absolute consistency problem for these five classes. It is also necessary to consider schema mappings for graph databases whose edges have properties as well as nodes.

ACKNOWLEDGMENT

We thank Professor Soichiro Hidaka at Hosei University for his kind introduction to bidirectional transformations and TGGs. We would like to thank all the reviewers for their valuable comments on our paper.

REFERENCES

[1] P. Barceló, "Logical foundations of relational data exchange," SIGMOD Record, vol. 38, no. 1, 2009, pp. 49–58.

[2] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data exchange: semantics and query answering," Theor. Comput. Sci., vol. 336, no. 1, 2005, pp. 89–124.

[3] M. Arenas, P. Barceló, L. Libkin, and F. Murlak, Relational and XML Data Exchange, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[4] S. Amano, L. Libkin, and F. Murlak, "XML schema mappings," in Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2009, pp. 33–42.

[5] M. Arenas and L. Libkin, "XML data exchange: Consistency and query answering," J. ACM, vol. 55, no. 2, 2008, pp. 7:1–7:72.

[6] P. Barceló, J. Pérez, and J. L. Reutter, "Schema mappings and data exchange for graph databases," in Joint 2013 EDBT/ICDT Conferences, ICDT '13 Proceedings, 2013, pp. 189–200.

[7] I. Boneva, A. Bonifati, and R. Ciucanu, "Graph data exchange with target constraints," in Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), 2015, pp. 171–176.

[8] 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, 2015, pp. 1443–1461.

[9] J. J. Miller, "Graph database applications and concepts with Neo4j," in Proceedings of the Southern Association for Information System Conference, Atlanta, GA, USA, March 23-24th, 2013, 2013.

[10] Y. Ishihara, H. Kuwada, and T. Fujiwara, "The absolute consistency problem of XML schema mappings with data values between restricted dtds," in Database and Expert Systems Applications - 25th International Conference, DEXA 2014, Munich, Germany, September 1-4, 2014. Proceedings, Part I, 2014, pp. 317–327.

[11] A. Schürr, "Specification of graph translators with triple graph grammars," in Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG '94, Herrsching, Germany, June 16-18, 1994, Proceedings, 1994, pp. 151–163.

[12] F. Hermann, H. Ehrig, F. Orejas, and U. Golas, "Formal analysis of functional behaviour for model transformations based on triple graph grammars," in Graph Transformations - 5th International Conference, ICGT 2010, Enschede, The Netherlands, September 27 - - October 2, 2010. Proceedings, 2010, pp. 155–170.

[13] E. Kindler and R. Wagner, "Triple graph grammars: Concepts, extensions, implementations, and application scenarios," Department of Computer Science, University of Paderborn, Technical Report tr-ri-07-284, 2007.

 ISBN: 978-1-61208-558-6