Towards Evolving Constraints in Data Transformation for XML Data Warehousing*

Md. Sumon Shahriar and Jixue Liu

Data and Web Engineering Lab
School of Computer and Information Science
University of South Australia, SA-5095, Australia
shamy022@students.unisa.edu.au, jixue.liu@unisa.edu.au

Abstract. Transformation of data is considered as one of the important tasks in data warehousing and data integration. With the massive use of XML as data representation and exchange format over the web in recent years, transformation of data in XML for integration purposes becomes necessary. In XML data transformation, a source schema and its conforming data is transformed to a target schema. Often, source schema is designed with constraints and the target schema also has constraints for data semantics and consistency. Thus, there is a need to see whether the target constraints are implied from the source constraints in data transformation. Towards this problem, we define two important XML constraints namely XML key and XML functional dependency(XFD). We then use important transformation operations to see if the source constraints are satisfied by the source document, then the target constraints are also satisfied by the target document. Our study is towards the utilization of constraints data integration and data warehousing in XML.

1 Introduction

Transformation of data is an important activity in some data intensive activities such as data integration and data warehousing[1,2]. Specifically in data integration, there is a need to transform a source schema with its conforming data to a target schema. In recent days, with the massive applications of XML[14] over the web, XML data transformation for integration purposes[6,7] becomes important. In XML data transformation[5,3,4], a source XML schema is often designed with XML constraints[11,12,13] to convey semantics and data integrity. Similarly, the XML target schema is also often designed with constraints. Thus after transformation, there is a need to see whether the target constraints are implied from the source constraints as a result of transformation operations. We illustrate the research question in Fig.1. In Fig.1, consider an XML source Document Type Definition(DTD) D_S, its conforming document T_S and valid constraints C_S on D_S. The transformation operation τ has two sub-operations: the schema transformation τ_D and the document transformation τ_T. The operations τ_D produce

* This research supported with Australian Research Council(ARC) Discovery Project(DP) Fund.

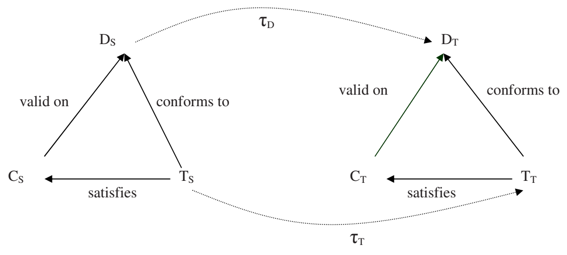

Fig. 1. The Problems

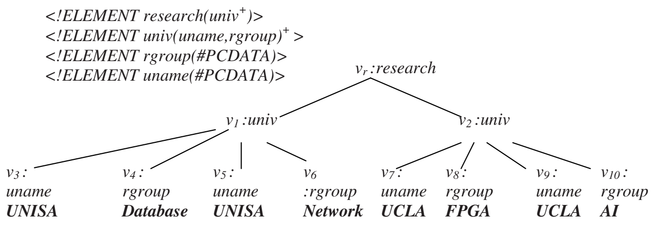

Fig. 2. An XML DTD D_a and the document T_a

the target schema D_T and the operations τ_T produce the target document T_T. Consider the constraints C_T on the target schema D_T. Now our question is: *If T_S satisfies C_S then whether T_T satisfies C_T.*

We now illustrate the research question using a motivating example. Consider D_a as the source DTD and its conforming document T_a in Fig.2. The figure illustrates the research groups of universities. We see that in each *univ* node, there are research group names(*rgroup*) with their associated university names(*uname*). Now consider the XML functional dependency(XFD) $\Phi_a(research/univ, \{uname\} \rightarrow univ)$ on the DTD D_a meaning that *uname* determines *univ*. We say XFD Φ_a is satisfied by the document T_a because in each *univ* node, there is at least one *uname* element(a technical definition of XFD and its satisfaction[15] will be given later). Note that there is more that one *uname* under each *univ* node as functional dependency allows redundant data. If we observe the document T_a, we see that under first *univ* node v_1, there are two *uname* nodes v_3, v_5 with the same "$UNISA$" value and under the second *univ* node v_2, there are two *uname* nodes v_7, v_9 with the same "$UCLA$" value. If we use *nest*(*rgroup*) meaning that research group names with same university are nested, then we get the document T_b. Surely we transform the source DTD D_a to D_b as the target DTD accordingly. Now consider the XML key[16] $\Bbbk_b(research/univ, \{uname\})$ on the DTD in Fig.3. Then we see that the document T_b satisfies the key $\Bbbk_b$ meaning that for all *univ* nodes, *uname* with values "$UNISA$" and "$UCLA$" are distinct.

Observation 1: *XML key is implied from XML functional dependency using nest operation.*

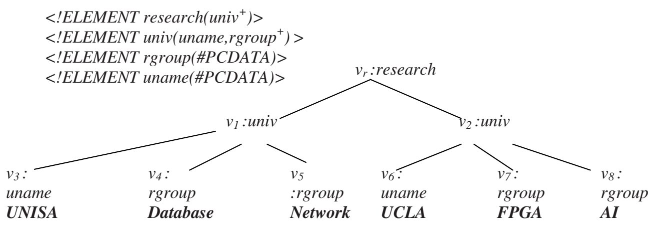

Fig. 3. An XML DTD D_b and the document T_b

While observing the research problems, we aim to achieve the following contributions.

- We define XML key[16] and XFD[15] on DTD and their satisfactions. The definitions for XML key and XFD consider the ordered model of XML data.
- We study how target constraints are implied from source constraints given some transformation operations.
- We finally show the experimental results on satisfactions for XML key and XFD in constraint implications.

2 Basic Definitions and Notation

In this section, we illustrate the definitions for XML key and XFD using examples. For detailed definitions, we refer to [15] for XML key and [16] for XFD. First, we give basic definitions and notation for DTD and the document those are necessary in defining XML key and XFD. A DTD is defined as $D = (EN, \beta, \rho)$ where EN contains element names and ρ is the root of the DTD and β is the function defining the types of elements. For example, the DTD D_a in Fig.2 is defined as $\beta(research) = [univ^+]$, $\beta(univ) = [uname{\times}rgroup^+]$, $\beta(uname) = Str$, $\beta(rgroup) = Str$, $EN = \{research, univ, uname, rgroup, Str\}$, $\rho = research$ and $Str = \#PCDATA$. An element name and a pair of squared brackets '[ ]' each, with its multiplicity, is called a *component*. For example, $[univ^+]$, $[uname{\times}rgroup^+]$ are two components. A sequence of components, often denoted by g, is called a *structure* s.t. $g = [uname{\times}rgroup^+]$. A structure can be further decomposed into *substructures* such as g can be decomposed into $g = g_1{\times}g_2$ and $g_1 = uname$ and $g_2 = rgroup^+$. We note that special cases of structures are components and that multiplicities can only be applied to components as g^c where $c \in [?, 1, +, *]$.

We say $research/univ$ is a complete path and $univ/uname$ is a simple path. The function $beg(research/univ) = research$, $last(univ/uname) = uname$ and $par(uname) = univ$.

Now we define XML key $\Bbbk(Q, \{P_1, \cdots, P_l\})$. We say Q as *selector* that is a complete path, $\{P_1, \cdots, P_l\}$ is called $fields$ those are simple paths. All paths P_i are ended with $\#PCDATA$ meaning that $\beta(last(P_i)) = Str$. For example, Consider the key $\Bbbk_b(research/univ, \{uname\})$ on D_b in Fig.3. We see $research/univ$ is a complete path, $uname$ is a simple path and $\beta(uname) = Str$.

In defining XML key satisfaction, we need some definitions and notation on XML document. For example, the document T_b in Fig.3 is defined as $T_{v_r} = (v_r : research : T_{v_1}T_{v_2})$, $T_{v_1} = (v_1 : univ : T_{v_3}T_{v_4}T_{v_5})$,$T_{v_2} = (v_2 : univ : T_{v_6}T_{v_7}T_{v_8})$. We then define $T_{v_3} = (v_3 : uname : UNISA)$ and the trees $T_{v_4}, T_{v_5}, T_{v_6}, T_{v_7}, T_{v_8}$ are defined in the same way. We say two trees or a sequence of trees are value equivalent($=_v$) if the node names and their values are the same. For example, $T_x = (x : uname : UNISA)$ and $T_y = (y : uname : UNISA)$ are value equivalent. Now consider $g = [uname_{\times} rgroup^+]$. We then say hedge $H^g = T_{v_3}T_{v_4}T_{v_5}$ for node v_1 and $H^g = T_{v_6}T_{v_7}T_{v_8}$ for node v_2. The necessity of hedge is to make the production of values for paths of fields. For example, if the paths of fields are $\{uname, rgroup\}$ in a key, then we need to produce close pair values as $(T_{v_3}T_{v_4})$, $(T_{v_3}T_{v_5})$ for node v_1 and $(T_{v_6}T_{v_7})$, $(T_{v_6}T_{v_8})$ for node v_2. We term these pair-wise values as *tuple*.

In case of key $\Bbbk_b(research/univ, \{uname\})$, we find the tuples (T_{v_3}) for node v_1 and (T_{v_6}) for node v_2 and these tuples are value distinct in the whole document T_b. Thus we say that key $\Bbbk_b$ is satisfied by the document T_b. However the key $\Bbbk_b$ is not satisfied by the document T_a in Fig.2 because there are duplicate tuples, for example T_{v_3} and T_{v_5} for those are the value same.

Now we define XFD $\Phi(S, P \to Q)$. We say S is the *scope* that is a complete path, P is *determinant*(LHS) that is simple path and Q is *dependent*(RHS) that is also simple path. The path Q can be ϵ(empty) meaning that $P \to last(S)$. In defining XFD satisfaction, we say that in each scope, if two tuples for paths P are the same, then their corresponding tuples for Q are also the same. For example, in Fig.3, the XFD $\Phi_b(research/univ, \{rgroup\} \to uname)$ is satisfied by the document T_b, but the XFD $\Phi'_b(research/univ, \{uname\} \to rgroup)$ is not satisfied by T_b. Consider another XFD $\Phi_a(research/univ, \{uname\} \to \epsilon)$ on the DTD D_a in Fig.2. The XFD Φ_a is satisfied by the document T_a because there is at least one tuple for path P in each scope *univ*.

3 Implication of XML Keys for Nest Operation

In XML data transformation, different transformation operators are used[5,3,4]. The important transformation operations those are found in most literatures are *Nest* and *UnNest*. In this section, we study how XML key is implied to the target schema from XFD on the source schema using *Nest* operation.

Before studying implication, we explain the *Nest* operation.

Definition 1 (Nest). *The nest operation on g_2 in $[g_1 \times g_2^{c_2}]^c$ is defined as, if $g = [g_1 \times g_2^{c_2}]^c \wedge c \supseteq +$, then nest$(g_2) \to [g_1 \times g_2^{c_2 \oplus +}]^c$. We say g_1 as comparator and g_2 as collector. The multiplicity operation $c_2 \oplus +$ means the multiplicity whose interval encloses those of c_1 and $+$.*

The nest operator restructures the document and it transforms the the flat structure of the document to the nested structure. The *nest* operator merges the hedges of type construct g_2 (the *collector*) based on the value equivalence of the hedges of type construct g_1 (the *comparator*). For example, given

$\beta(e) = [A_\times B_\times C_\times D]^*$ and $T = (e(A:1)(B:1)(C:2)(D:3)(A:1)(B:2)(C:2)(D:4)(A:1)(B:1)(C:2)(D:4))$, the operator $nest(C_\times D)$ combines the hedges of the collector $C_\times D$ based on the value equivalence of the hedges of the comparator $A_\times B$ and produces $\beta_1(e) = [A_\times B_\times [C_\times D]^+]^*$ and $T_1 = (e(A:1)(B:1)(C:2)(D:3)(C:2)(D:4)(A:1)(B:2)(C:2)(D:4))$. Thus we see that after *Nest* operation, the values for the collector g_1 in the document becomes distinct.

We get the following theorem for the nest operation.

Theorem 1. *Given the transformation* $Nest(g_2)$*, an XML key* $\Bbbk_t(Q, \{P\})$ *on the target schema is implied from an XFD* $\Phi_s(S, P \to \epsilon)$ *on the source schema if the path* P *is involved in* g_1*.*

The proof of the theorem follows the transformation definition of the *Nest* operation. In XFD Φ_s, the tuples for path P needs to be complete and can have two tuples with same value. If the path P in XFD is involved in the structure g_1, then after transformation using *Nest*, the values for path P become distinct which satisfies the key satisfaction property.

We illustrate the theorem using an example.

Example 1. Consider the XFD $\Phi_a(research/univ, \{uname\} \to \epsilon)$ on the source DTD D_a in Fig.2. This XFD is satisfied by the document T_a because in the selector node v_1, there are two tuples $(v_3 : uname : UNISA)$ and $(v_5 : uname : UNISA)$ and in the selector node v_2, there are two tuples $(v_7 : uname : UCLA)$ and $(v_9 : uname : UCLA)$. After $Nest(rgroup)$, we see that there is one tuple $(v_3 : uname : UNISA)$ for the node v_1 and there is one tuple $(v_6 : uname : UCLA)$ for node v_2 in Fig.3. Considering $[uname_\times rgroup]^+$ where $g_1 = uname$ and $g_2 = rgroup$, the path $uname$ in Φ_a is involved in g_1 and it follows the theorem1. Thus the key $\Bbbk_b(research/univ, \{uname\})$ is satisfied by the document T_b in Fig.3.

4 Implication of XFD and XML Key for UnNest Operation

As we mentioned in the previous section that *UnNest* is one of the important transformation operations, thus we study how XFD and XML key are implied to the target schema from XML keys on the source schema using *UnNest* operation.

Definition 2 (UnNest). *The unnest operation on* g_2 *in* $[g_1 \times g_2^{c_2}]^c$ *is defined as, if* $g = [g_1 \times g_2^{c_2}]^c \wedge c_2 = +|*$*, then* $unnest(g_2) \to [g_1 \times g_2^{c_2 \ominus +}]^{c \oplus +}$*. The multiplicity operation* $c_2 \ominus +$ *means the multiplicity whose interval equals to the interval of* c_2 *taking that of* $+$ *and adding* $'1'$*.*

The *unnest* operator spreads the hedge of the *comparator* type construct g_1 to the hedges of the *collector* type construct g_2. For example, given $\beta(e) = [A_\times B_\times [C_\times D]^+]^*$ and $T = (e(A:1)(B:1)(C:2)(D:3)(C:2)(D:4)(A:$

$1)(B : 2)(C : 2)(D : 4))$, the operator $unnest(C_{\times}D)$ spreads the hedge of the comparator $A_{\times}B$ to the hedges of the collector $C_{\times}D$ and produces $\beta_1(e) = [A_{\times}B_{\times}[C_{\times}D]]^*$ and $T_1 = (e(A : 1)(B : 1)(C : 2)(D : 3)(A : 1)(B : 1)(C : 2)(D : 4)(A : 1)(B : 2)(C : 2)(D : 4))$. We see that the comparator g_1 is distributed to all g_2. Thus the number of g_2 remains unchanged but the number of g_1 is increased with the same value.

We get the following theorems for $UnNest$ operation.

Theorem 2. *Given the operation $UnNest(g_2)$, an XML key $\Bbbk_t(Q, \{P_1, P_2\})$ on target schema is implied from XML keys $\Bbbk'_s(Q, \{P_1\})$ and $\Bbbk''_s(Q, \{P_2\})$ on the source schema if P_1 of $\Bbbk'_s$ is involved in g_1 and P_2 of $\Bbbk''_s$ is involved in g_2.*

The proof of the theorem follows the definition of the $UnNest$ operation. We illustrate the theorem using an example.

Example 2. Consider D_b as the source DTD, the document T_b as the source document and two keys $\Bbbk'_b(research/univ, \{uname\})$ and $\Bbbk''_b(research/univ, \{rgroup\})$ in Fig.3. Both keys are satisfied by the document T_b. We use $UnNest(rgroup)$ to transform D_b and T_b to D_a as the target DTD and T_a as the target document. Considering $uname_{\times}rgroup^+$ where $g_1 = uname$ and $g_2 = rgroup^+$, we see that path $uname$ of key $\Bbbk'_b$ is involved in g_1 and the path $rgroup$ in key $\Bbbk''_b$ is involved in g_2. This follows the condition of the theorem 2. After $UnNest$, we see that the tuples $(v_3 : uname : UNISA, v_4 : rgroup : database)$ and $(v_5 : uname : UNISA, v_6 : rgroup : Network)$ of node v_1 and the tuples $(v_7 : uname : UCLA, v_8 : rgroup : FPGA)$ and $(v_9 : uname : UCLA, v_{10} : rgroup : AI)$ of node v_2 for paths $uname$ and $rgroup$ are distinct in the document T_a that conforms to D_a. Thus the key $\Bbbk_a(research/univ, \{uname, rgroup\})$ is satisfied by the document T_a in Fig.2.

Theorem 3. *Given the operation $UnNest(g_2)$, an XFD $\Phi_t(S, P \rightarrow \epsilon)$ on the target schema is implied from an XML key $\Bbbk_s(Q, \{P\})$ on the source schema if P is involved in g_1.*

We illustrate the theorem using an example.

Example 3. Consider the D_b as the source DTD, the document T_b as the source document and the XML key $\Bbbk_b(research/univ, \{uname\})$ on D_b. The key $\Bbbk_b$ is satisfied by the document T_b as the tuples for path $uname$ are value distinct in the document. We use $UnNest(rgroup)$ to transform D_b and T_b to D_a as the target DTD and T_a as the target document. Considering $uname_{\times}rgroup^+$ where $g_1 = uname$ and $g_2 = rgroup^+$, we see that path $uname$ of key $\Bbbk_b$ is involved in g_1. This follows the condition of the theorem 3. After $UnNest$, we see that the tuples $(v_3 : uname : UNISA)$ and $(v_5 : uname : UNISA)$ of node v_1 and the tuples $(v_7 : uname : UCLA)$ and $(v_9 : uname : UCLA)$ of node v_2 for paths $uname$ in the document T_a that conforms to D_a. Thus the XFD $\Phi_a(research/univ, \{uname\} \rightarrow \epsilon)$ is satisfied by the document T_a in Fig.2.

5 Performances on Checking Implied XML Keys and XFDs at Target Schema

We have already shown how XML key and XFD can be implied from sources to the target schema when $Nest$ and $UnNest$ transformation operations are used respectively. In this section, we study the performances of checking XML key and XFD satisfactions by the transformed, loaded and integrated data at the target schema. All experiments are implemented in Java using a PC with Intel(R) Centrino Duo CPU T2050 at 1.60GHz, 1.49GB RAM and Microsoft Windows XP.

In Fig.4, we show the key satisfaction time where we fix the number of fields to 4 but varying the number of tuples. We see the significant time is spent for tuple generation while the hashing time is nearly constant. We use Java $Hastable(Key, Value)$ to put the values of tuple to check distinctness incrementally. As the tuple generation time and the hashing time are linear, thus the satisfaction time which is the sum of the tuple generation time and the hashing time is also linear.

In similar way of reasoning, the satisfaction time of checking key in Fig.5 is also linear where we fix the number of tuples to 600K but we vary the number of paths in the key.

We show XFD satisfaction time that is linear in Fig.6 where we fix the number of paths to 3 in the LHS but we vary the number of tuples. In Fig.7, the XFD satisfaction time is also linear where we fix the number of tuples to 500K but we vary the number of paths in LHS for an XFD.

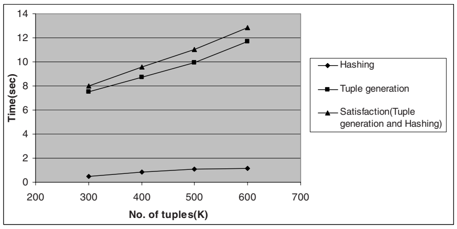

Fig. 4. Key Satisfaction time when the number of fields is fixed to 4

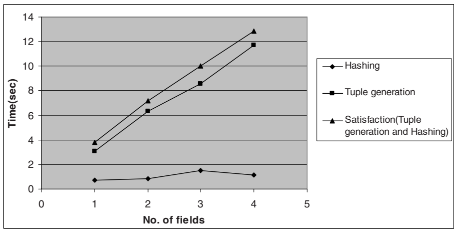

Fig. 5. Key Satisfaction time when the number of tuples is fixed to 600K

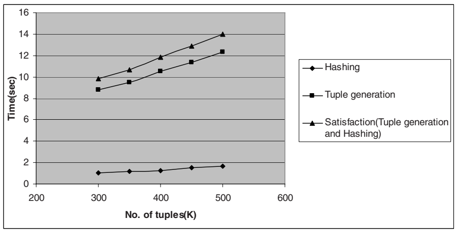

Fig. 6. XFD Satisfaction time when the number of paths in LHS is fixed to 3

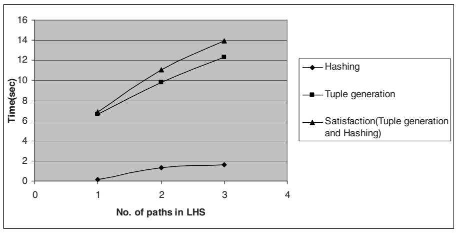

Fig. 7. XFD Satisfaction time when the number of tuples is fixed to 500K

6 Conclusions

We studied the implication of XML constraints in data transformations using important transformation operations namely *Nest* and *UnNest* for constraints implications. In constraints implication, we used our proposed definition for XML key and XML functional dependency and also showed the performances of checking satisfactions of constraints for implication purpose. We further plan to research on how the implications of other XML constraints such as XML inclusion dependency and XML foreign key in XML data integration purposes.

References

1. Fankhouser, P., Klement, T.: XML for Datawarehousing Chances and Challenges. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2003. LNCS, vol. 2737, pp. 1–3. Springer, Heidelberg (2003)
2. Golfarelli, M., Rizzi, S., Vrdoljak, B.: Datawarehouse Design form XML sources. In: DOLAP, pp. 40–47 (2001)
3. Su, H., Kuno, H., Rudensteiner, E.A.: Automating the Transformation of XML Documents. In: WIDM, pp. 68–75 (2001)
4. Erwig, M.: Toward the Automatic Derivation of XML Transformations. In: ER, pp. 342–354 (2003)
5. Liu, J., Park, H., Vincent, M., Liu, C.: A Formalism of XML Restructuring Operations. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 126–132. Springer, Heidelberg (2006)
6. Zamboulis, L., Poulovassilis, A.: Using Automed for XML Data Transformation and Integration. In: DIWeb, pp. 58–69 (2004)
7. Zamboulis, L.: XML Data Integration by Graph Restructuring. In: Williams, H., MacKinnon, L.M. (eds.) BNCOD 2004. LNCS, vol. 3112, pp. 57–71. Springer, Heidelberg (2004)
8. Poggi, A., Abiteboul, S.: XML Data Integration with Identification. In: Bierman, G., Koch, C. (eds.) DBPL 2005. LNCS, vol. 3774, pp. 106–121. Springer, Heidelberg (2005)
9. Li, C.: Describing and utilizing Constraints to Answer Queries in Data Integration Systems. In: IIWeb (2003)
10. Fuxman, A., Miller, R.e.J.: Towards Inconsistency Management in Data Integration Systems. In: IIWeb 2003 (2003)
11. Buneman, P., Fan, W., Simeon, J., Weinstein, S.: Constraints for Semistructured Data and XML. In: SIGMOD Record, pp. 47–54 (2001)
12. Fan, W.: XML Constraints: Specification, Analysis, and Applications. In: DEXA, pp. 805–809 (2005)
13. Fan, W., Simeon, J.: Integrity constraints for XML. In: PODS, pp. 23–34 (2000)
14. Bray, T., Paoli, J., Sperberg-McQueen, C.M.: Extensible Markup Language (XML) 1.0., World Wide Web Consortium (W3C) (February 1998), `http://www.w3.org/TR/REC-xml`
15. Shahriar, M. S., Liu, J.: Preserving Functional Dependency in XML Data Transformation. In: Atzeni, P., Caplinskas, A., Jaakkola, H. (eds.) ADBIS 2008. LNCS, vol. 5207, pp. 262–278. Springer, Heidelberg (2008)
16. Shahriar, M.S., Liu, J.: Towards the Preservation of Keys in XML Data Transformation for Integration. In: COMAD 2008, pp. 116–126 (2008)