Chinese Journal of Electronics
Vol.18, No.2, Apr. 2009

Logical Implication of Structural Integrity Constraints for XML*

ZHANG Jianmei[1,2], TAO Shiqun[1] and LIANG Jiye[1]

(1.*Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China*)

(2.*Changzhi University, Changzhi 046011, China*)

Abstract — For tree XML, constraints that specify structural relationships among nodes or paths are very natural. In this paper, we introduce the concept of structural integrity constraints for XML (XSICs), which specify path implication, path cooccurrence, path mutual-exclusion, element obligatory inclusion and exclusive inclusion, and define the syntax and semantics of XSICs. For reasoning about XSICs, we rewrite all the other constraints into path implication constraints, and develop a sound and complete set of inference rules for path implication constraints. Meanwhile, we propose the concept of path implication closure. By using the path implication closure, we prove the completeness of inference rules, and determine the implication decision about XSICs.

Key words — XML, Structural integrity constraints for XML, Inference rules, Path implication closures.

I. Introduction

In databases, the integrity constraints have been proved useful in making a good design, update anomaly prevention, and query optimization[1,2], *etc.* For these problems, many scholars extend schema formalisms for XML with the traditional integrity constraints, such as keys, foreign keys, functional dependencies[3−6]. These constraints prescribe the value relationships among related nodes, but their structural relationships. Document type definitions (DTDs)[7] and XML Schema[8] specify admissible elements, attributes and element nesting, but they are inadequate to deal with complex structural relationships between different paths or nodes. However the structural relationships provide more opportunities for optimizing path expressions. As an example, let us consider path expression query $/a/b[c][d/e]$. If both c and e are not required child of their respective parent, the query can not be minimized further. However, if we know that the existence of $/a/b/c$ requires the existence of $/a/b/d/e$, predicate d/e is redundant and can be removed from the query; in contrast, if we know that the existence of $/a/b/c$ precludes $/a/b/d/e$, we can find the evaluation result of the query empty without accessing the actual document. Hence, it is indispensable to research on the constraints specifying the structural relationships between different paths or nodes, which are named Structural integrity constraints for XML (XSICs).

Related work. To date, there are only a few works about XSICs. Refs.[9,10] studied edge constraints in semistructured data, which can be see as the earliest XSICs. Ref.[11] proposed the concept of Structural constraints for XML (XSCs), including path implication, absence and cooccurrence, but they are defined based on simple path expressions including /. Ref.[12] studied trees pattern constraints, which extend XSCs with path expressions containing /, //, [ ], *. However, they are very intricate in reasoning and practical applications. We extend XSCs with linear path expressions including /, //, which are simple enough in efficient reasoning and expressive enough for practical applications. Besides path implication, cooccurrence and mutual-exclusion, XSICs specify element obligatory inclusion and exclusive inclusion. Ref.[11] studied logical implication of XSCs by giving a set of inference rules for XSCs. In contrast, we first give a set of inference rules for path implication constraints to solve its implication, and then determine the implication decision about XSICs by using path implication closures. In Ref.[13], element containment relationships are used to reason about query terms formulated in PAT algebra. Since PAT algebra is different from path expressions in the terms of expressiveness, the concept of exclusivity is different from ours. This work makes the following contributions.

• We define the syntax and semantics of XSICs based on linear path expressions, present the constraints rewriting technique to integrate path constraints with element inclusion constraints.

• We introduce the concepts of sub-path and path implication closure. Based on sub-path, we develop a set of inference rules for path implication.

• We prove the completeness of inference rules, and determine the implication decision about XSICs.

Organization. In Section II, we introduce basic notions about XML documents and path expressions, and present the concept of sub-path. Section III defines the syntax and semantics of XSICs, and describes the constraints rewriting technique. Section IV presents a set of inference rules for path implication and the concept of path implication closure. With path implication closures, we prove the completeness of inference rules and the implication decision about XSICs. Section V concludes the paper and future works.

II. Background

1. XML document trees

XML documents are often modeled as node-labeled trees. As used in Ref.[4], we assume the existence of three pairwise disjoint sets of labels: E of element types, A of attribute names, and a singleton set $\{S\}$ denoting text type. An XML tree T has a set of

*Manuscript Received May 2008; Accepted Oct. 2008. This work is supported by the National Natural Science Foundation of China (No.70471003) and the Research Foundation for the Doctoral Program of Ministry of Education of China (No.20050108004).

nodes, denoted V, where each node v in V has a label, denoted $lab(v)$. A node v in V is called an element if $lab(v) \in E$, an attribute if $lab(v) \in A$, and a text node if $lab(v) = S$. If a node v' in V is a sub-element or attribute of node v, v' is called a child of v and there is a directed edge from v to v'. Based on standard XML semantics[14], the root node of an XML tree corresponds to the document node. Fig.1 shows the tree of a sample XML document with text nodes ommittied to keep the figure simple.

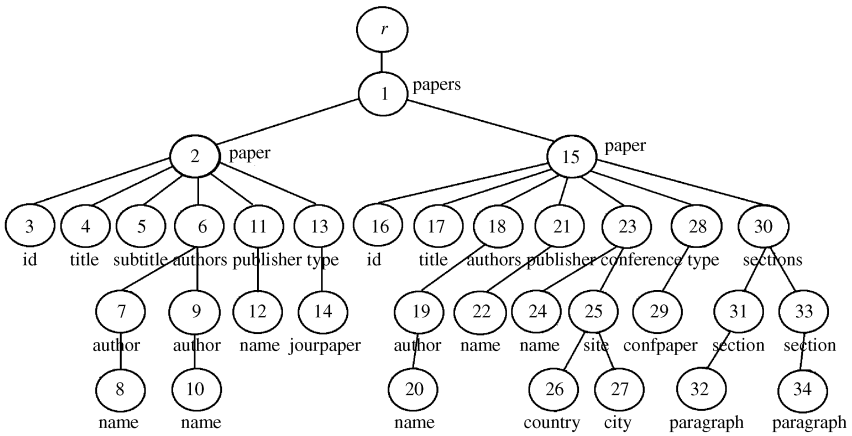

Fig. 1. A sample of XML document tree

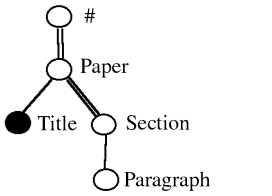

Fig. 2. A sample of tree pattern

2. Path expressions and tree pattern

A path expression defines a way of navigating XML trees. This paper focus on a set of XPath including /, //, [ ]. Expression Q can be defined recursively by the following grammar:

$$Q ::= \varepsilon|/P|//P,$$
$$P ::= e|.|P/P|P//P|P[P]|P[.//P], \quad e \in E \cup A$$

where ε is the empty path, . the current node, / parent-child navigation, // ancestor-descendant navigation, expressions enclosed in [ ] are called predicates.

Path expressions can be represented by tree patterns. For example, Fig.2 shows the tree pattern of path expression "//paper[.//section/paragraph]/title", where single edges (child edges) and double edges (descendant edges) correspond to / and // in the original expression, respectively, and the output node is darkened. A node v is called a child of node u if (u, v) is a child edge, and a descendant if there exists a child or descendant edge (u, v), or a sequence of edges from u to v. The root node # is introduced to show that the original expression is an absolute path or a relative one. The root node in tree patterns can match the root node of any XML tree.

The path expressions starting with / are named absolute paths, and others named relative paths. The path expressions without predicates are called linear path expressions. We consider the linear path expressions without recursive elements which are used frequently in practice. Thus every node type of a linear path expression corresponds a node in its tree pattern. Therefore, when it comes to linear path expression, we directly use e_i to denote node v whose type is e_i, unless otherwise stated. $P \in LP$ denotes that P is a linear path expression, $nodes(P)$ and $last(P)$ represent the node type set and the last node type in P, respectively.

Definition 1 $\forall P, Q \in LP$, $P = c_1e_1c_2e_2 \cdots e_n$, $Q = c'_1e'_1c'_2e'_2 \cdots e'_m$, if $m < n$, $c_i = c'_i$ and $e_i = e'_i$ $(i = 1, \cdots, m)$, then Q is a path prefix of P, denoted $Q \prec P$. If $m = n$, then Q equals to P, denoted $Q = P$.

Note that ε is a path prefix of any path expression.

Definition 2 $\forall P \in LP$, $P = c_1e_1c_2e_2 \cdots c_ne_n$, if $c_2 = c_3 \cdots = c_n = /$, then P is called a simple path expression.

3. Sub-path and containment of path expressions

Definition 3 $\forall P, Q \in LP$, let N_p and N_q be the set of nodes in the tree patterns of P and Q, respectively. Q is a sub-path of P, denoted $Q \preceq P$, if and only if there exists a mapping $f : N_q \to N_p$, such that:

(1) f preserves nodes types: $\forall v \in N_q$, if $lab(v) = e$, $lab(f(v)) = e$;

(2) f preserves structural relationships: $\forall u, v \in N_q$, if (u, v) is a child edge (resp., a descendant edge) in Q, $f(v)$ is a child (resp., descendant) of $f(u)$ in P.

If $Q \preceq P$ and P is a simple path expression, P is named the access path of Q, denoted $acc(Q)$.

Definition 4 Let $Q(T)$ be the query results of path expression Q in XML tree T, if $Q_1(T) \subseteq Q_2(T)$ holds for any document tree T, then Q_2 contains Q_1, denoted $Q_1 \subseteq Q_2$.

Note that the concept of sub-path is distinct from query containment. Given linear path expressions P and Q, $P \subseteq Q$ if and only if $Q \preceq P$ and $last(Q) = last(P)$. For example, "/papers/paper//section" is a sub-path of "/papers/paper//section//paragraph", but their query results are disjoint.

III. Structural Integrity Constraints for XML

The integrity constraints, which specify the structural relationships between different paths or nodes, are called structural integrity constraints for XML (XSICs). we classify them into element-based XSICs and path-based XSICs.

1. Path-based XSICs

Path-based XSICs specify the structural relationships between two different paths, including path implication, path mutual-exclusion and path cooccurrence, which are named path constraints.

Definition 5 $\forall C, P, Q \in LP$, $u, v, v' \in V$, let r denote the root node of XML tree T, and $u[P]$ the set of nodes reached by following P from node u in T. T satisfies

(1) a path implication constraint $C(P \to Q)$, denoted $T \models C(P \to Q)$, if and only if $\forall u \forall v(u \in r[C] \wedge v \in u[P] \to \exists v'(v' \in u[Q]))$;

(2) a path mutual-exclusion constraint $C(P \nrightarrow Q)$, denoted $T \models C(P \nrightarrow Q)$, if and only if $\forall u \forall v(u \in r[C] \wedge v \in u[P] \to \neg\exists v'(v' \in u[Q]))$ and $\forall u \forall v(u \in r[C] \wedge v \in u[Q] \to \neg\exists v'(v' \in u[P]))$;

(3) a path cooccurrence constraint $C(P \leftrightarrow Q)$, denoted $T \models C(P \leftrightarrow Q)$, if and only if $C(P \to Q) \wedge C(Q \to P)$; where:

• C is the context path which specifies the document sub-trees, where path constraints must hold. Whenerver $C = \varepsilon$, the path constraint holds in the whole document and ε can be omited;

• P and Q, with C as their path prefix, are Left hand path (LHP) and Right hand path (RHP) repectively. When $|P| = |Q| = |C| + 1$, the path constraints become the relationships between two sibling nodes.

In other words, path implication $C(P \rightarrow Q)$ states that in all document sub-trees determined by C, the existence of P requires the existence of Q, and a path cooccurrence is a two-way path implication. In contrast, path mutual-exclusion $C(P \nrightarrow Q)$ precludes the existence of Q if P occurs, and vice versa[11]. Fig.3 shows the relationships between paths specified by $C(P \leftrightarrow Q)$, $C(P \nrightarrow Q)$, $C(P \rightarrow Q)$, respectively.

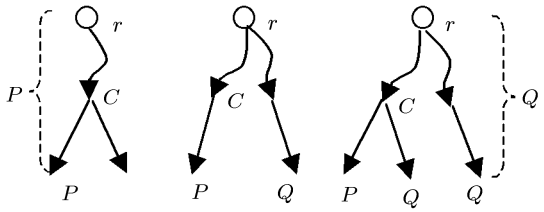

Fig. 3. Path constraints illustration

Example 1 The following constraints are satisfied by the XML tree in Fig.1.

$$\varphi_1 : //paper \rightarrow //paper/authors/author/name$$

$$\varphi_2 : /papers/paper(title \leftrightarrow .//author/name)$$

$$\varphi_3 : /papers/paper(type/jourpaper \nrightarrow conference)$$

In real constraint examples, LHP and RHP are written relative paths with respect to C to simplify writing. For example, φ_2 is short for $/papers/paper(/papers/\ paper/title \leftrightarrow /papers/paper//author/name)$.

2. Element-based XSICs

Element-based XSICs specify implication, cooccurrence, mutual-exclusion and structural containment relationships among nodes. Since implication, cooccurrence and mutual-exclusion between nodes are special cases of path constraints, the section only gives the definition of structural containment relationships between nodes: element obligatory inclusion and exclusive inclusion, which are named element inclusion constraints.

Definition 6 $\forall P, Q \in LP$, $e_i \in E$, $e_j \in E \cup A$, u, v, v_1, $v_2 \in V$, let r denote the root node of XML tree T, and $u[P]$ the set of nodes reached by following P from node u in T. T satisfies

(1) an element obligatory inclusion $e_i \Rightarrow e_j$, denoted $T \models e_i \Rightarrow e_j$, if and only if $\forall u(lab(u) = e_i \rightarrow \exists v(lab(v) = e_j \wedge v \in u[P])$;

(2) an element exclusive inclusion $e_i \mapsto e_j$, denoted $T \models e_i \mapsto e_j$, if and only if $\forall u(lab(u) = e_i \wedge \exists v_1 \exists v_2(v_1 \in u[P] \wedge v_2 \in u[Q] \wedge lab(v_1) = lab(v_2) = e_j) \rightarrow acc(P) = acc(Q))$.

Example 2 The following constraints are satisfied by the XML tree in Fig.1.

$$\varphi_4 : paper \mapsto author$$

$$\varphi_5 : author \Rightarrow name$$

Theorem 1 $\forall e_i, e_j \in E$, $\forall e_k \in E \cup A$, if $e_i \Rightarrow e_j$ and $e_j \Rightarrow e_k$, then $e_i \Rightarrow e_k$.

Theorem 1 states that element obligatory inclusions are transitive.

3. Constraint rewriting

Path cooccurrence constraints are essentially two-way path implication constraints, so $C(P \leftrightarrow Q)$ is rewritten into $C(P \rightarrow Q)$ and $C(Q \rightarrow P)$. Based on the semantics of path mutual-exclusion, $C(P \nrightarrow Q)$ is rewritten into $C(P \rightarrow \neg Q)$ and $C(Q \rightarrow \neg P)$, where $\neg P$ means not to exist P.

Given a set of linear path expressions B, for any linear path expression P, $P \propto B$ if and only if there exists Q in B such that $P \preceq Q$; $\neg P \propto B$ if and only if there exists $\neg Q$ in B such that $Q \preceq P$.

Due to the interaction between path constraints and element inclusion constraints, it is indispensable for reasoning about XSICs to integrate path constraints with element inclusion constraints. In database, the chase technique is used to rewrite a query to incorporate the effects of integrity constraints[2]. We develop a chase technique to rewrite a given set of path constraints to integrate the effects of element inclusion constraints.

Theorem 2 Let $P = c_1e_1c_2e_2 \cdots c_ke_k$, the following conclusions hold:

(1) if $e_i \mapsto e_j$ and $e_i, e_j \in nodes(P)$ $(i < j,\ i = 1, 2 \cdots k-1,\ j = 2 \cdots k)$, then $c_1e_1 \cdots c_ie_i//e_j \cdots c_ke_k \rightarrow P$;

(2) if $e_{k-1} \Rightarrow e_k$ and $e_{k-1}, e_k \in nodes(P)$, then $c_1e_1c_2e_2 \cdots c_{k-1}e_{k-1} \rightarrow P$;

(3) if $e_i \Rightarrow e_n$ and $e_i \in nodes(P) \wedge e_n \notin nodes(P)$ $(i = 1, 2 \cdots k)$, then $P \rightarrow c_1e_1c_2e_2 \cdots c_ie_i//e_n$.

It is obvious that the theorem holds from Definition 5 and 6. The theorem states the chase technique to rewrite a set of path constraints with element inclusion constraints. We leave out the details of the chase technique due to space limitations.

IV. Logical Implication of XSICs

In this section, we study the implication problem of XSICs. Since all constraints are rewritten into path implication constraints, the implication problem for XSICs focuses on the implication for path implication constraints. Without loss of generality, we present a set of inference rules for path implication constraints.

1. Inference rules for path implication

Let $C, C', P, Q, S \in LP$, the inference rules for path implication constraints are listed in Table 1.

Theorem 3 The set **R** of inference rules is sound for logical implication of path implication constraints.

Given a set Σ of path implication constraints, the soundness of the set **R** of inference rules means that any constraint deduced from Σ using **R** must be implied by Σ, namely the inference rules are correct. Let T_c be the set of sub-trees, t_c a sub-tree specified by path C, namely $t_c \in T_c$, the proof of soundness is as follows.

Table 1. R: Rules for path implication

Rule 1:	$C(P \rightarrow Q),\ C' \prec C \Rightarrow C'(P \rightarrow Q)$
Rule 2:	$C(P \rightarrow Q),\ C' \subseteq C \Rightarrow C'(P \rightarrow Q)$
Rule 3:	$\forall P, Q \in LP,\ Q \preceq P \Rightarrow P \rightarrow Q$
Rule 4:	$C(P \rightarrow Q),\ C(Q \rightarrow S) \Rightarrow C(P \rightarrow S)$
Rule 5:	$C(P \rightarrow Q),\ C(Q \rightarrow \neg S) \Rightarrow C(P \rightarrow \neg S)$
Rule 6:	$C(P \rightarrow \neg Q),\ C(S \rightarrow Q) \Rightarrow C(P \rightarrow \neg S)$
Rule 7:	$C(P \rightarrow \neg Q),\ C \preceq C' \Rightarrow C'(P \rightarrow \neg Q)$

Proof (1) Rule 1. If $C' \prec C$, then for any $t_{c'}$, there is a t_c such that t_c is a sub-tree of $t_{c'}$. Since the existence of P requires the existence of Q in all t_c, so is it in $t_{c'}$ (see Fig.4). Hence, the rule is sound.

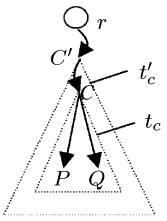

Fig. 4. Rule 1 illustration

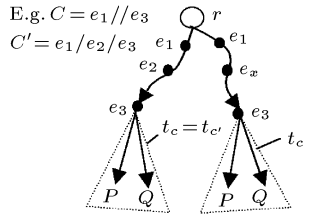

Fig. 5. Rule 2 illustration

(2) Rule 2. If $C' \subseteq C$, then $T_{C'} \subseteq T_C$, namely, for any $t_{c'}$, there is a t_c such that $t_c = t_{c'}$. Since the existence of P requires the

existence of Q in all t_c, so is it in $t_{c'}$ (see Fig.5). Hence, the rule is sound.

(3) Rule 3. This rule states that if there is a path in any XML sub-tree, there also exists its sub-path in the same sub-tree. It follows from the concept of sub-path that the rule is sound.

(4) Rule 4 and 5. Rule 4 states the transitivity of path implication, and Rule 5 states the relationship between path implication and path mutual-exclusion. It is clear from the semantics of path implication.

(5) Rule 6. Suppose $C(P \rightarrow \neg S)$ is false, then there is at least one t_c, where there are P and S. Since $C(S \rightarrow Q)$, there are P and Q in the t_c, which contradict $C(P \rightarrow \neg Q)$. Hence, the supposition does not hold.

(6) Rule 7. If $C \preceq C'$, there may be three relationships between C and C' in XML document trees.

① $C \prec C'$: For any $t_{c'}$, there is a t_c such that $t_{c'}$ is a sub-tree of t_c. Hence, if the existence of P precludes the existence of Q in t_c, then so is it in $t_{c'}$ (see Fig.6(a));

② $Last(C') = Last(C) : T_{C'} \subseteq T_C$, namely, for any $t_{c'}$, there is a t_c such that $t_c = t_{c'}$. Hence, if the existence of P precludes the existence of Q in t_c, then so is it in $t_{c'}$ (see Fig.6(b)). Hence, $C'(P \rightarrow \neg Q)$ holds;

③ $C \not\prec C'$ and $Last(C') \neq Last(C)$: For any $t_{c'}$, there is a t_c such that $t_{c'}$ is a sub-tree of t_c. Since the existence of P precludes the existence of Q in t_c, then so is it in $t_{c'}$ (see Fig.6(c)). Hence, $C'(P \rightarrow \neg Q)$ holds.

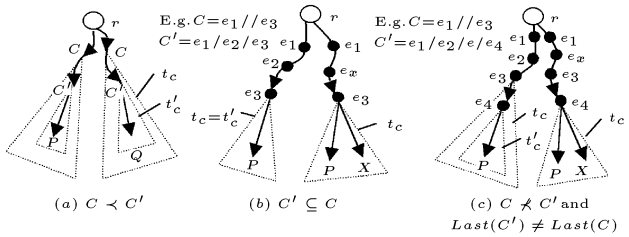

Fig. 6. Rule 7 illustration

From the above stated, Rule 7 is sound.

2. Path implication closures

Definition 7 Given a set Σ of path implication constraints and two linear path expressions C, P, the path implication closure of P under Σ and C is defined as the minimal set of all path expressions implied by P, denoted $P^+_{(\Sigma,C)}$ or simply $P^+_{(\Sigma)}$ if the context path is ε, namely $P^+_{(\Sigma,C)} = \{X | C(P \rightarrow X)$ can be deduced from Σ with $\mathbf{R}$, where $X \in \{Q, \neg Q\}\}$.

Let Σ denote the set of rewritten path constraints, and Ω the set of element inclusion constraints, the algorithm to evaluate $P^+_{(\Sigma,C)}$ is as follows.

Algorithm 1 $ClosureComp(\Sigma, \Omega, C, P)$
Input non-empty Σ, Ω, and $C, P \in LP$
Output $P^+_{(\Sigma,C)}$
/* initialization */
$B = \{P\}$;
if $(e_i \Rightarrow e_x \in closure(\Omega)$ && $e_i \in nodes(P))$
  if $e_i = last(P)$ **then** $B = \{P//e_x\}$;
  else $B = B \cup \{c_1e_1c_2e_2 \cdots c_ie_i//e_x\}$;
/**computing B */
check each $X \in B$ **until** all expressions are checked
  for each $\varphi : C'(P' \rightarrow Q') \in \Sigma$
    if $(X == S)$ **then**
      if $(C \prec C' || C = C')$ **then**
        if $(P' \preceq S$ && $Q' \not\preceq S)$ **then** $B = B \cup \{Q'\}$;/*Rule 1,3,4*/
      else if $(C' \preceq S$ && $last(C') \notin nodes(C) - last(C)$ && $P' \preceq S)$ **then**
        replace prefix C' of Q' with C_s; /*C_s is a sub-path of S from its first node type to $last(C')$*/
        if $(Q' \not\preceq S)$ **then** $B = B \cup \{Q'\}$; /*Rule 2,3,4*/
    if $(X == \neg S)$ **then**/* $X == \neg S$ means that X is the linear path of the form $\neg S$ */
      if $(C \prec C' || C = C')$ **then**
        if $(S \preceq Q')$ **then** $B = B \cup \{\neg P'\}$; tag $\neg P'$ with C_s; /*Rule 1,3,6*/
      else if $(C' \preceq S$ && $last(C') \notin nodes(C) - last(C))$ **then** replace the prefix C' of P' and Q' with C_s;
        if $(S \preceq Q')$ **then** $B = B \cup \{\neg P'\}$; tag $\neg P'$ with C_s; /*Rule 2,3,6*/
  for each $\varphi : C'(P' \rightarrow \neg Q') \in \Sigma$
    if $(X == S)$ **then** /*Rule 3,5,7*/
      if $(C \preceq C'$ && $P' \preceq S)$ **then** $B = B \cup \{\neg Q'\}$;
      else if $(C' \preceq S$ && $P' \preceq S)$ **then** $B = B \cup \{\neg Q'\}$; tag $\neg Q'$ with C_s;
  /*delete redundant path expressions from B */
  for each $P \in B$
    if $(Q \in B$ && $Q \preceq P)$ **then** $B = B - \{Q\}$
  for each $\neg P \in B$
    if $(\neg Q \in B$ && $P \preceq Q)$ **then** $B = B - \{Q\}$
  return B

Example 3 Given a path constraint set $\Sigma = \{\varphi_1, \varphi_2, \varphi_3\}$ from Example 1, let $\Omega = \Phi$, $C = /papers/paper$, and $P = /papers/paper/type/jourpaper$, we show how to derive $P^+_{(\Sigma,C)}$ based on the algorithm.

(1) Before running the algorithm, path cooccurrence and mutual-exclusion constraints are rewritten into path implication constraints.

$$\varphi_2 \Rightarrow \begin{cases} \varphi_2 : /papers/paper(title \rightarrow .//author/name) \\ \varphi_2' : /papers/paper(.//author/name \rightarrow title) \end{cases}$$

$$\varphi_3 \Rightarrow \begin{cases} \varphi_3 : /papers/paper(type/jourpaper \rightarrow \neg conference) \\ \varphi_3' : /papers/paper(conference \rightarrow \neg type/jourpaper) \end{cases}$$

(2) Since $\Omega = \Phi$, the algorithm initializes B to $\{P\}$;

(3) Inside the double loop, the algorithm checks whether path expressions in B imply RHP of path implication constraints in Σ. When the algorithm runs the outer loop for the first time, P in B is compared with every constraint in Σ. Since the context path of φ_1 is a path prefix of C and LHP of φ_1 is a sub-path of P, its RHP $//paper/authors/author/name$ (denoted P_1) is implied by P and added to B in the first inner loop. With the same reason, RHP $\neg/papers/paper/conference$ (denoted P_2) of φ_3 is added to B in the second inner loop. For the second time, P_1 in B is compared with every constraint in Σ, and RHP $/papers/paper/title$ (denoted P_3) of φ_2' is added to B. For the third time, P_2 in B is compared with every constraint in Σ, and no path is added to B. For the fourth time, P_3 in B is compared with every constraint in Σ, and RHP $/papers/paper//author/name$ (denoted P_4) of φ_2 is added to B. Since P_4 is a sub-path of P_1, it is removed from B in the third inner loop. Here, all path expressions in B have been checked, so the double loop is ended. Therefore $P^+_{(\Sigma,C)} = B = \{P, P_1, P_2, P_3\}$.

3. Completeness of inference rules

Lemma 1 $\varphi : C(P \rightarrow X)$ $(X \in \{Q, \neg Q\})$ can be deduced from Σ with $\mathbf{R}$ if and only if $X \propto P^+_{(\Sigma,C)}$.

The correctness of the lemma can be directly derived from the definition of $P^+_{(\Sigma,C)}$.

Lemma 2 Let $X \in \{Q, \neg Q\}$, T be a XML tree, if P does not occur in T, then for any path C and Q in T, $T \models C(P \rightarrow X)$ holds; if C does not occur in T, then for any path P and Q in T, $T \models C(P \rightarrow X)$ holds.

Proof $C(P \rightarrow Q)$ (resp., $C(P \rightarrow \neg Q)$) states that in all sub-trees rooted at the nodes whose path matches C, the existence of path P requires (resp., precludes) the existence of path Q. Thus when P does not occur in these sub-trees, it does not matter whether path Q occurs or not and what path Q occurs, hence

$C(P \rightarrow Q)$ holds. With the same reason, when C does not occur in T, $C(P \rightarrow Q)$ (resp., $C(P \rightarrow \neg Q)$) also holds.

Theorem 4 The set $\mathbf{R}$ of inference rules is complete for logical implication of path implication constraints.

Given a set Σ of path implication, the set $\mathbf{R}$ of reference rules is complete if and only if for any constraint φ, if $\Sigma \models \varphi$, then φ can be deduced from Σ with $\mathbf{R}$. It follows from the inverse proposition equivalence that if constraint φ can't be deduced from Σ, then $\Sigma \not\models \varphi$. That is to say, there is at least one XML tree T such that $T \models \Sigma$ and $T \not\models \varphi$. In the following section, we prove the completeness of $\mathbf{R}$ by using the path implication closure as a main tool.

Proof Suppose that path constraint φ can't be deduced from Σ with $\mathbf{R}$. Let $\varphi = C(P \rightarrow Q)$ or $\varphi = C(P \rightarrow \neg Q)$, we now prove that there exists an XML tree T such that $T \models \Sigma$ and $T \not\models \varphi$.

(1) We first construct an XML tree T, in which all path expressions in $P^+_{(\Sigma)}$ occur except for Q (resp., $\neg Q$).

Step 1 Divide $P^+_{(\Sigma)}$ into two subsets: $P^+_{(\Sigma)1}$ and $P^+_{(\Sigma)2}$, where $P^+_{(\Sigma)1}$ is the set of path expressions of the form Q, denoting path expressions implied by P; $P^+_{(\Sigma)2}$ is the set of path expressions of the form $\neg Q$, denoting path expressions precluded by P.

Step 2 Delete Q (resp., $\neg Q$) from $P^+_{(\Sigma)1}$ (resp., $P^+_{(\Sigma)2}$) in the following way:

• When $\varphi = C(P \rightarrow Q)$, search Q in $P^+_{(\Sigma)1}$. If Q is found, then delete it, otherwise search Q' such that $Q \preceq Q'$ in $P^+_{(\Sigma)1}$. If Q' is found, delete it;

• When $\varphi = C(P \rightarrow \neg Q)$, first search $\neg Q$ in $P^+_{(\Sigma)2}$. If $\neg Q$ is found, then delete it, otherwise search $\neg Q'$ such that $Q' \preceq Q$ in $P^+_{(\Sigma)2}$. If $\neg Q'$ is found, then delete it. Then add Q to $P^+_{(\Sigma)1}$.

Step 3 Create the root node r of T, and connect path P to r in the following way:

If P is an absolute path, as the root element node of XML tree T, the first node of tree pattern P is connected to r; otherwise create a root element node re, connect re to r, and connect the first node of tree pattern P to re.

Step 4 For the other path expressions in $P^+_{(\Sigma)1}$, connect their sub-paths, which are not in T, to the correct position of P in turn;

Step 5 If $P^+_{(\Sigma)2}$ is not empty, then do the following for every $\neg X$ in $P^+_{(\Sigma)2}$. First search the tag path of $\neg X$ in T, and create a copy T'_s of sub-tree T_s rooted at the node determined by the tag path, then delete P (or P' such that $P \preceq P'$) from T'_s, finally connect tree pattern X to T'_s.

Remark When deleting P from T'_s, we don't delete its path prefix C_s to preserve the connectivity of T'_s (see Fig.7).

Step 6 If there are double edges in T, then translate the double edges into single ones.

Example 4 Let Σ be the path constraint set from Example 3, and given a path constraint $\varphi = /papers/paper(type/jourpaper \rightarrow \neg conference)$, construct an XML tree T such that $T \models \Sigma$ but $T \not\models \varphi$.

After step 1 and 2, $P^+_{(\Sigma,C)1} = \{/papers/paper/type/jourpaper,\ /papers/paper/authors/author/name,\ /papers/paper/conference,\ /papers/paper/title\}$ and $P^+_{(\Sigma,C)2} = \Phi$. XML tree T is shown in Fig.8.

(2) We then prove that XML tree T satisfies Σ. In other words, for any $\psi \in \Sigma$, $T \models \psi$ holds. Let $\psi = C'(P' \rightarrow Q')$ or $\psi = C'(P' \rightarrow \neg Q')$, the proof is as follows:

① If C' or P' do not occur in T, it follows from Lemma 2 that $T \models \psi$ holds.

② If both C' and P' occur in T, $P' \propto P^+_{(\Sigma)}$. It follows from Lemma 1 that $(P \rightarrow P')$ holds.

• For each $\psi = C'(P' \rightarrow Q') \in \Sigma$, $\varepsilon \prec C'$ holds, it follows from Rule 1 that $P' \rightarrow Q'$. Based on Rule 4, we know $P \rightarrow Q'$ holds, namely $Q' \propto P^+_{(\Sigma)}$ holds. Hence, Q' must occur in T, namely $T \models \psi$.

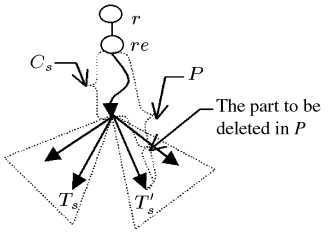

Fig. 7. Illustration of deleting P from T'_s

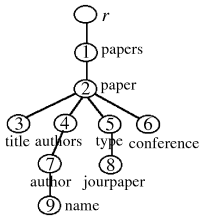

Fig. 8. XML tree for example 4

• For each $\psi = C'(P' \rightarrow \neg Q') \in \Sigma$, we prove XML tree T satisfies ψ by contradiction.

Suppose $T \not\models \psi$, then at least in one of sub-trees specified by C', there exist both P' and Q'. If there is only one sub-tree specified by C' in T, it follows from the semantics of path implication that $C'(P' \rightarrow Q')$ holds. If there are two sub-trees specified by C' in T, from the construction of T, we know that there exist P' and Q' in one sub-tree, and not exists P' in the other sub-tree. Based on the semantics of path implication, $C'(P' \rightarrow Q')$ holds. The conclusion contradicts $C'(P' \rightarrow \neg Q')$, hence the assumption does not hold, namely $T \models \psi$.

(3) Finally, we prove that XML tree T does not satisfy φ. Suppose $T \models \varphi$:

• When $\varphi = C(P \rightarrow Q)$, there exist both P and Q in the sub-tree specified by C. Hence $Q \propto P^+_{(\Sigma)}$;

• When $\varphi = C(P \rightarrow \neg Q)$, it follows from the construction of T that there exists at least two sub-trees specified by C in T, and P occurs in one sub-tree and Q in other sub-tree, namely, $\neg Q \propto P^+_{(\Sigma,C)}$. From Algorithm 1, we know $P^+_{(\Sigma,C)} \subseteq P^+_{(\Sigma)}$ holds, hence $\neg Q \propto P^+_{(\Sigma)}$;

It follows from Lemma 1 that φ can be deduced from Σ with $\mathbf{R}$. This contradicts assumption "φ can not be deduced from Σ with $\mathbf{R}$". Hence assumption $T \models \varphi$ does not hold, namely $T \not\models \varphi$.

From the above stated, it follows that if φ can not be deduced from Σ with $\mathbf{R}$, φ must not be implied by Σ. Hence, the set $\mathbf{R}$ of inference rules is complete.

4. Implication decision about XSICs

In the section, we show how the path implication closures are used for checking whether a constraint is implied by a given constraint set Σ.

Theorem 5 Given path constraint $C(X \rightarrow Y)$ $(Y \in \{Q, \neg Q\})$, if $X \propto P^+_{(\Sigma,C)}$, then $Y \propto P^+_{(\Sigma,C)}$.

Proof If $X \propto P^+_{(\Sigma,C)}$, then $C(P \rightarrow X)$. It follows from Rule 4 that $C(P \rightarrow Y)$ holds, hence $Y \propto P^+_{(\Sigma,C)}$.

For path constraints, one can directly check whether a new constraint is implied by a given constraint set based on the theorem. But when it comes to element inclusion constraints, the check is a little trickier.

(1) Element obligatory inclusion: Let $\varphi = e_i \Rightarrow e_j$, the check technique is as follows:

Step 1 Search P that $e_i \in nodes(P)$ holds in Σ. If P is not found, then $\Sigma \models \varphi$; otherwise go to Step 2;

Step 2 Evaluate $P^+_{(\Sigma,C)}$;

Step 3 If $//e_i \propto P^+_{(\Sigma,C)}$ holds, then $\Sigma \models \varphi$; otherwise $\Sigma \not\models \varphi$.

(2) Element exclusive inclusion: Let $\varphi = e_i \mapsto e_j$, the check technique is as follows:

Step 1 Search P that $e_j \in nodes(P)$ holds in Σ. If P is not found, then $\Sigma \models \varphi$; otherwise go to Step 2;

Step 2 Evaluate $P^+_{(\Sigma,C)}$;

Step 3 If $e_i//e_j \propto P^+_{(\Sigma,C)}$ holds and there is only one path expression Q such that $e_i//e_j \preceq Q$ in $P^+_{(\Sigma,C)}$, then $\Sigma \models \varphi$; otherwise $\Sigma \not\models \varphi$.

V. Conclusions and Future Works

Structural integrity constraints for XML, which specify the structural relationships between different elements or paths, play an important role in the path expression query optimization. In this paper, we study the implication for XSICs as stand-alone schema formalism. We show that the containment problem for linear path expressions can be effectively decided using sub-path. Based on this, we present a sound and complete set of inference rules. By using path implication closures, we prove the completeness of the reference rule set, and determine the implication decision problem of XSICs.

Since XSICs play important roles in XML query optimization. We are currently investigating how XSICs are used in minimizing path expressions to improve the evaluation performance of path expressions. Furthermore, XSICs can be regarded as an extension of DTDs. We will study how to extend DTDs with XSICs to obtain a compact representation of path implication closures and optimize path expression evaluation thoroughly.

References

[1] S. Abiteboul, R. Hull and V. Vianu, *Foundations of Databases*, Addison-Wesley, Boston, USA, 1995.

[2] L. Popa, "Object/relational query optimization with chase and backchase", *Ph.D. Thesis*, University of Pennsylvania, USA, 2000.

[3] P. Buneman, S. Davidson, W. Fan, C. Hara and W. Tan, "Keys for XML", *Computer Networks*, Vol.39, No.5, pp.473–487, 2002.

[4] W. Fan and J. Simeon, "Integrity constraints for XML", *Proc. of 19th ACM Symposium on Principles of Database Systems*, Dallas, Texas, USA, pp.23–34, 2000.

[5] M.L. Lee, T.W. Ling and W.L. Low, "Designing functional dependencies for XML", *Proc. of the 8th International Conference on Extending Database Technology: Advances in Database Technology*, Prague, Czech Republic, pp.124–141, 2002.

[6] Z.J. Tan, Y.M. Pang and B.L. Shi, "Reasoning about functional dependency for XML", *Journal of Software*, Vol.14, No.9, pp.1564–1570, 2003. (in Chinese)

[7] *World Wide Web Consortium* (*W3C*), *Extensible Markup Language* (*XML*) *1.0*, Third Edition, W3C Recommendation, http://www.w3.org/TR/xml, 2006.

[8] *World Wide Web Consortium* (*W3C*), *XML Schema Part 1: Structures*, Second Edition, W3C Recommendation, http://www.w3.org/TR, 2004.

[9] D. Calvanese, G.D. Giacomo, M. Lenzerini, "What can knowledge representation do for semi-structured data?", *Proc. of the 15th National Conference on Artificial Intelligence*, Madison, USA, pp.205–210, 1998.

[10] A. Cali, D. Calvanese, M. Lenzerini, "Semistructured data schemas with expressive constraints", *Proc. of the 7th International Workshop on Knowledge Representation Meets Databases*, Berlin, Germany, pp.3–16, 2000.

[11] A. Kwong and M. Gertz, "Structural constraints for XML", *Technical Report*, Department of Computer Science, University of California at Davis, USA, 2002.

[12] A. Kwong and M. Gertz, "On tree pattern constraints for XML documents", *Technical Report*, Department of Computer Science, University of California at Davis, USA, 2003.

[13] K. Böhm, K. Aberer, M.T. Özsu and K. Grayer, "Query optimization for structured documents based on knowledge on the document type definition", *Proc. of IEEE Forum on Research and Technology Advances in Digital Libraries*, IEEE Computer Society Press, Santa Barbara, USA, pp.196–205, 1998.

[14] *XML Path Language* (*XPath*) *2.0*. W3C Recommendation, www.w3c.org/TR/xpath20, Jan. 2007.

ZHANG Jianmei was born in 1970. She is an associate professor at Changzhi University, and Ph.D. candidate at Shanxi University, China. She received M.S. degree from Shanxi University in 2000. His research interests include XML data management, query processing and optimization. (Email: jmzhang@sxu.edu.cn)

TAO Shiqun is a professor at Shanxi University, China, and a supervisor of Ph.D. students. He received B.S. degree from Beijing Institute of Technology in 1969. His research interests include databases theory, database query processing and optimization. (Email: tsq@sxu.edu.cn)

LIANG Jiye is a professor at Shanxi University, China, and a supervisor of Ph.D. students. He received M.S. degree and Ph.D. degree from Xi'an Jiaotong University in 1990 and 1998, respectively. His research interests include data mining and knowledge discovery in database. (Email: ljy@sxu.edu.cn)