A Revised Result on Chasing Tree Patterns Under Schema Graphs

Junhu Wang[a,*], Jeffrey Xu Yu[b], Jixue Liu[c], Chaoyi Pang[d]

[a] *Griffith University, Australia*
[b] *The Chinese University of Hong Kong, China*
[c] *The University of South Australia, Australia*
[d] *Zhejiang University (NIT), China*

Abstract

We first provide an example to show that two previously published results in [2] and [4] on tree pattern containment under schema graphs are incorrect. We then show that the original result in [2] holds under some special conditions on the schema graph.

1. Introduction

XPath plays a central role in all XML query languages. A major fragment of XPath can be represented as tree patterns [3]. Testing tree pattern containment, i.e., whether the answers to a tree pattern are all answers to another tree pattern, is fundamental for XML query optimization. It is shown in [3] that, when P and Q involve only /, // and [], P is contained in Q if and only if there is a homomorphism from Q to P. Unfortunately, when a DTD is present, the existence of a homomorphism from Q to P is no longer a necessary condition for P to be contained in Q. However, [5] shows that if the DTD is *duplicate-free* and the tree patterns involve / and [] only, then testing whether a tree pattern P is contained in another pattern Q under the DTD can be reduced to testing whether P is contained in Q under two types of constraints implied by the DTD. Subsequently [2] claimed this result can be extended to tree patterns involving /,// and [] under an acyclic schema graph (which is a type of simplified DTD), and [4] attempted to extend the result of [2] to cyclic schema graphs. Unfortunately, the claims in [2] and [4] are both incorrect, as we will show in this paper. We then present a special case and show that in this special case, the containment problem of two tree patterns can be reduced to the containment problem under the constraints identified in [2].

*Corresponding author
Email address: J.Wang@griffith.edu.au (Junhu Wang)

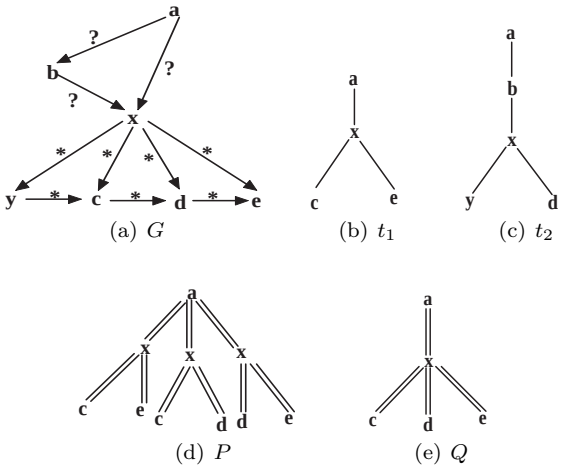
2. Preliminaries

For any rooted graph $\mathcal{G}$, we use $\texttt{root}(\mathcal{G})$ and $\texttt{node}(\mathcal{G})$ to denote the root and the node set, respectively, of $\mathcal{G}$.

Schema Graph and XML tree. Similar to [2], we model an XML schema as a connected directed graph G (called a **schema graph**) satisfying the following conditions: (1) Each node is labeled with a distinct label; (2) Each edge is labeled with one of 1, ?, +, and $*$; (3) There is a unique node, called the root, such that every other node is reachable from this node. The set of labels occurring in G is denoted Σ_G. Because a node in a schema graph G has a unique label, we also refer to a node by its label. A schema graph may be cyclic or acyclic. An example acyclic schema graph is shown in Fig. 1 (a).

As in previous work [1, 3, 2] we model an XML document as a finite unordered tree (referred to as an **XML tree**) where every node has a label. Let v be a node in an XML tree t. The label of v is denoted $label(v)$. An XML tree t is said to **conform** to schema graph G if (1) for every node $v \in \texttt{node}(t)$, $label(v) \in \Sigma_G$, (2) $label(\texttt{root}(t)) = label(\texttt{root}(G))$, (3) for every edge (u, v) in t, there is a corresponding edge $(label(u), label(v))$ in G, and (4) for every node $v \in \texttt{node}(t)$, the number of children of v labeled with x is constrained by the label of the edge $(label(v), x)$ in G: there must be exactly one, zero or one, one or many, or zero or many children of v labeled with x if the label of the edge $(label(v), x)$ is 1, ?, +, or $*$ respectively. Fig.1 (b) and (c) show two XML trees conforming to the schema graph G

in Fig.1 (a). The set of all XML trees conforming to G is denoted T_G.



(a) G (b) t_1 (c) t_2

(d) P (e) Q

Figure 1: Schema graph G, conforming XML trees t_1 and t_2, and tree patterns P and Q satisfiable under G

Tree Patterns. We consider the class of Boolean tree patterns (or simply *tree patterns* or *patterns*) $\mathcal{P}^{\{/,//,[]\}}$ as defined in [3]. Each pattern in $\mathcal{P}^{\{/,//,[]\}}$ is a node-labeled tree with two types of edges: /-edges and //-edges. It corresponds to an XPath expression involving the child-axis (/), descendant-axis (//), and branch condition ([]). Fig. 1 (d) and (e) show two tree patterns. Here, single and double lines represent /-edges and //-edges respectively. A branch in the tree represents a condition ([]) in the XPath expression.

Given a pattern P and an XML tree t, $P(t)$ returns `true` if and only if there is an embedding of P in t. An **embedding** of a tree pattern P in an XML tree t is a mapping δ from $\texttt{node}(P)$ to $\texttt{node}(t)$ with the following conditions: (1) label-preserving, i.e., $\forall v \in \texttt{node}(P)$, $label(v) = label(\delta(v))$, (2) root-preserving, i.e., $\delta(\texttt{root}(P)) = \texttt{root}(t)$, and (3) structure-preserving, i.e., for every edge (x, y) in P, if it is a /-edge, then $\delta(y)$ is a child of $\delta(x)$; if it is a //-edge, then $\delta(y)$ is a descendant of $\delta(x)$, i.e, there is a path from $\delta(x)$ to $\delta(y)$ of length greater than 0.

A tree pattern P is said to be **satisfiable** under schema graph G if there exists $t \in T_G$ such that $P(t)$ is true. The tree patterns shown in Fig. 1 (d) and (e) are satisfiable under the schema graph G in Fig. 1 (a). In this paper we implicitly assume all tree patterns are satisfiable under the schema graphs being discussed.

Tree pattern containment Given two patterns P and Q, P is said to be contained in Q (under G), denoted $P \subseteq Q$ ($P \subseteq_G Q$) if for every XML tree t ($\in T_G$), $P(t) \Rightarrow Q(t)$. P and Q are said to be *equivalent* (under G) if $P \subseteq Q$ and $Q \subseteq P$ ($P \subseteq_G Q$ and $Q \subseteq_G P$). It is well known that for any $P, Q \in \mathcal{P}^{\{/,//,[]\}}$, $P \subseteq Q$ if and only if there is a homomorphism from Q to P [1]. A **homomorphism** [3] from Q to P is a mapping δ from $\texttt{node}(Q)$ to $\texttt{node}(P)$ that is label-preserving, root-preserving as defined in the definition of embedding, and structure-preserving which now means that, for every edge (x, y) in Q, if it is a /-edge, then $(\delta(x), \delta(y))$ is a /-edge in P; and if it is a //-edge, then there is a path from $\delta(x)$ to $\delta(y)$ of length > 0.

3. The incorrect results in previous work

It is claimed in Theorem 4 of [2] that, if P and Q are both in $\mathcal{P}^{\{/,//,[]\}}$ and the schema graph G is acyclic, then whether P is contained in Q under G can be reduced to tree pattern containment under the following five types of constraints[1] (referred to as the LWZ constraints in this paper) implied by G (A schema graph G is said to **imply** a constraint C, denoted $G \vDash C$, if every XML tree conforming to G satisfies C).

(1) **Parent-Child Constraints (PC)**, denoted $a \Downarrow^1 x$, which means that whenever an x-node is the descendant of an a-node, it must be the child of the a-node.

(2) **Sibling Constraints (SC)**, denoted $a :\downarrow y$. The constraint means that every a-node must have a y-child[2].

(3) **Cousin Constraints (CC)**, denoted $a : x \Downarrow y$. The constraint means that for every a-node, if it has an x-descendant, then it also has a y-descendant.

(4) **Intermediate Node Constraints (IC)**, denoted $a \xrightarrow{x} y$, which means that every path from an a-node to a y-node must pass through an x-node.

[1] The SC and FC were originally defined in [5].

[2] The original SC constraint in [5] is of the form $a : S \downarrow y$ where S is a set of labels, which means that if an a-node has a z-child for every $z \in S$, then it must also have a y-child. The SC constraint in [2] is a special case where S may contain at most 1 label. We note the schema graph in [2] allows a node to represent a sequence of labels. If there are no sequence types in the schema graph, then any SC must be of the form $a : \emptyset \downarrow y$ (called *child constraints* in [5]), which is abbreviated as $a :\downarrow y$ in this paper.

(5) **Functional Constraints (FC)**, denoted $a \twoheadrightarrow x$, which means that every a-node has at most one x-child.

To check containment, [2] uses the following *chase* rules to transform (a.k.a. *chase*) P and Q repeatedly until no change can be made, resulting in two new tree patterns P' and Q'. It is claimed that $P \subseteq_G Q$ if and only if $P' \subseteq Q'$.

PC-rule: If $G \vDash a \Downarrow^1 x$, then change any $a//x$-edge to an a/x-edge.

SC-rule: If $G \vDash a :\downarrow y$, and an a-node does not have a y-child already, then add a $(/, y)$-child to the a-node.

CC-rule: If $G \vDash a : x \Downarrow y$, and an a-node has a $(//, x)$-child, but it does not have a y-descendant already, then add a $(//, y)$-child to the a-node.

IC-rule: If $G \vDash a \xrightarrow{x} y$, then replace any $a//y$-edge with the path $a//x//y$.

FC-rule: If $G \vDash a \twoheadrightarrow x$, and there are two or more $(/, x)$-children of an a-node, then merge these x-children.

Unfortunately, this claim is incorrect as shown by the example below.

Example 1: Consider the schema graph G and the tree patterns P and Q in Fig. 1. G implies the FCs $a \twoheadrightarrow x$, $a \twoheadrightarrow b$, and $b \twoheadrightarrow x$, but it does not imply any SCs, nor does it imply any ICs of the form $a \xrightarrow{z} x$ or $x \xrightarrow{z} w$ for any label z and w, or PCs of the form $a \Downarrow^1 x$ or $x \Downarrow^1 z$ for any $z \in \{c, d, e\}$. It does not imply any CCs of the form $a : x \Downarrow w$ or $x : z \Downarrow w$ for any label w and $z \in \{c, d, e\}$. Therefore, P and Q cannot be chased, in other words, $P' = P$ and $Q' = Q$. Clearly there is no homomorphism from Q to P. However, $P \subseteq_G Q$ as we explain below. Under G the path from an a-node to an x-node is either a/x or $a/b/x$, thus the $a//x$-edge in P is equivalent to the disjunction of a/x and $a/b/x$, and P is equivalent to the union of six tree patterns, each of which is obtained by replacing every $a//x$-edge with either an a/x-edge or $a/b/x$-edge. Using the FCs, we know in each pattern at least two of the x-nodes must be merged, and the merged node will have at least three descendants labeled c, d and e respectively. Therefore there is a homomorphism from Q to each of the six tree patterns obtained from P (after merging any two x-nodes). Therefore, all of the six patterns are contained in Q, thus $P \subseteq_G Q$. □

The above example also demonstrates that Theorem 4 of [2] does not hold even if Q and P involve only // and [].

When the schema graph is cyclic, [4] identifies three additional types of constraints that can be implied by the schema graph. It is claimed that if $P, Q \in \mathcal{P}^{\{/,//,[]\}}$, and P cannot be chased (that is, applying the chase rules to P cannot change P) using the new types of constraints together with the LWZ constraints, then $P \subseteq_G Q$ if and only if there is a homomorphism from Q to P. The result unfortunately is incorrect too, and Example 1 can serve as a counter example.

4. Revised results for acyclic schema graphs

We first consider the special case where the schema graph G has no edges labeled ? or 1. Such a schema graph cannot imply any FCs. However, the other four types of LWZ constraints are all possible. This is clear from the lemma below.

Lemma 1: *[2, 5] Let G be an acyclic schema graph.*

(1) *$G \vDash a \Downarrow^1 x$ if and only if the only path from a to x is the edge (a, x).*
(2) *$G \vDash a:\downarrow y$ if and only if the edge (a, y) is labeled $+$ or 1.*
(3) *$G \vDash a : x \Downarrow y$ if and only if every path from a to x has a node from which there is a* **mandatory path** *(i.e., a path where every edge is labeled $+$ or 1) to y.*
(4) *$G \vDash a \xrightarrow{x} y$ if and only if every path from a to y passes through the node x.*
(5) *$G \vDash a \twoheadrightarrow x$ if and only if the edge (a, x) is labeled 1 or ?.* □

We will prove the revised result below:

Theorem 1: Suppose G is acyclic and has no edges labeled ? or 1. If P cannot be chased using the LWZ constraints implied by G, then $P \subseteq_G Q$ if and only if $P \subseteq Q$. □

To prove the above theorem, we need to define some concepts and prove some lemmas. Throughout this section, we assume G is an acyclic schema graph, and P and Q are tree patterns in $\mathcal{P}^{\{/,//,[]\}}$ satisfiable under G. We will *explicitly state* G has no edges labeled ? or 1 when we need this condition. This is deliberate so that we can use the same lemmas to prove other special cases at the end of

this section. In the sequel, when we say "P cannot be chased", we mean P cannot be chased using the LWZ constraints implied by G. We will use *G-path* to refer to a path in G; */-child* (resp. //-child) to refer to a child connected to the parent via a /-edge (resp. //-edge); *(/,x)-child* to refer to a /-child labeled x, and x//y-edge to refer to a //-edge from an x-node to a y-node.

Let Q be a tree pattern in $\mathcal{P}^{\{/,//,[]\}}$. Given a node v in Q, the subtree of Q rooted at v, denoted Q_v, is called a *node subtree* of Q. Given an edge (u, v) in Q, the subtree of Q consisting of the edge (u, v) and Q_v, denoted $Q_{(u,v)}$, is called an *edge subtree* of Q. The following lemma is straightforward.

Lemma 2: *Let $u_1, \ldots, u_m$ be all of the children of $\texttt{root}(Q)$. For any tree pattern P, Q has a homomorphism to P if and only if every edge subtree $Q_{(\texttt{root}(Q),u_i)}$ has a homomorphism to P.* □

Let x be a node in schema graph G. Denote G_x the subgraph rooted at x, i.e, the subgraph consisting of all nodes and edges reachable from x. The next lemma is also straightforward.

Lemma 3: *Let v be an x-node in $P \in \mathcal{P}^{\{/,//,[]\}}$. If P is satisfiable under G, then P_v is satisfiable under G_x. Furthermore, if P cannot be chased using the LWZ rules under G, then P_v cannot be chased using the LWZ rules under G_x.* □

If u is a node in tree $t \in T_G$, we call any child of u that must exist due to sibling constraints a **mandatory child** of u.

Let (c_1, d_1) be a $c//d$-edge in a tree pattern. An **instance** of (c_1, d_1) is a path $c_1, v_1, \ldots, v_m, d_1$, such that $c, label(v_1), \cdots, label(v_m), d$ is a path in G. A **complete instance** of (c_1, d_1) is a tree obtained from some instance of (c_1, d_1) by adding mandatory children to each node repeatedly (as follows: if there is a node v_i in the instance, and there is an edge $(label(v_i), x)$ in G which is labeled + or 1, and v_i does not have an x-child already, then add an x-child for v_i), until no more nodes can be added.

Lemma 4: *Let P be a pattern that cannot be chased. Suppose $\texttt{root}(P)$ has no x-descendants. Then for every //-edge (u, v) in P, (1) if $u \neq \texttt{root}(P)$, then there is a complete instance of (u, v) which has no x-nodes. (2) if $u = \texttt{root}(P)$, then there is a complete instance of (u, v) which has no x-nodes under u.* □

Proof Since $\texttt{root}(P)$ has no x-descendant, for every //-edge (u, v) in P, we know $label(v) \neq x$, and if $u \neq \texttt{root}(P)$, then $label(u) \neq x$. We note that G can not imply the constraint $label(u) : label(v) \Downarrow x$, nor the constraint $label(u) \xrightarrow{x} label(v)$, otherwise an x-node would be added under u, contradicting the assumption that P cannot be chased. By Lemma 1, there is a G-path from $label(u)$ to $label(v)$, denoted p, which does not pass through x (except in the case $x = label(u)$), and from any node on p, there is no mandatory path to x. The complete instance of (u, v) corresponding to p does not have any x-node (except u in the case $x = label(u)$ and $u = \texttt{root}(P)$). Hence the lemma holds. □

Given tree pattern P, and XML tree $t \in T_G$, if $P(t) = true$, we say t is a **model** of P under G. For any P satisfiable under G, we can construct a model of P from P as follows: (i) replace every //-edge with a complete instance of the edge; (ii) merge nodes using the FCs implied by G (this step can be omitted if G has no edges labeled with ? or 1). Next we show that any tree constructed this way is indeed a model of P under G.

Lemma 5: *Suppose t is a tree constructed from P using the steps (i) and (ii) above. Then t is a model of P.* □

Proof P clearly has an embedding in t. So we only need to show $t \in T_G$. Since P is satisfiable, we know $\texttt{root}(t) = \texttt{root}(P)$ has the same label as $\texttt{root}(G)$, and every path in t corresponds to a path in G. Furthermore, for every node u in t, the number of children of u labeled with a particular label x conforms to the label of the edge $(label(u), x)$: if the label is 1 or ?, that is, G implies the FC $label(u) \twoheadrightarrow x$, then there is at most one child of u labeled with x due to step (ii). By definition, $t \in T_G$. □

In the following, we will call a mapping from $\texttt{node}(P)$ to $\texttt{node}(t)$ an **unrooted embedding** if the mapping satisfies all conditions of an embedding except the condition of 'root-preserving'.

Lemma 6: *Let G be a schema graph that has no edges labeled with 1 or ?. Suppose $\texttt{root}(Q)$ has no /-child and it has only one //-child v and $label(v)$=x. If P cannot be chased, then $P \subseteq_G Q$ if and only if $\texttt{root}(P)$ has a descendant u, such that $label(u) = x$, and $P_u \subseteq_{G_x} Q_v$.* □

Proof The 'if' direction is obvious. We show 'only if' next. Suppose $P \subseteq_G Q$. We show $\texttt{root}(P)$ has an x-descendant u such that $P_u \subseteq_{G_x} Q_v$.

We first show that $\texttt{root}(P)$ must have an x-descendant. If not, we can obtain an XML tree t by replacing each //-edge (u, v) in P with a complete instance of (u, v) that does not have x-nodes if $u \neq \texttt{root}(P)$, and replacing each //-edge $(\texttt{root}(P), v)$ with a complete instance that does not have x-nodes under $\texttt{root}(P)$ (By Lemma 4, such instances exist). By Lemma 5 t is a model of P. Obviously, $\texttt{root}(t)$ does not have x-descendants, hence Q has no embeddings in t. This contradicts the assumption $P \subseteq_G Q$.

Now let $x_1, \ldots, x_k$ be all of the x-descendants of $\texttt{root}(P)$. We will show there exists $i \in [1, k]$ such that $P_{x_i} \subseteq_{G_x} Q_v$. If not, that is, for every i, $P_{x_i} \not\subseteq_{G_x} Q_v$, then we will construct a model t of P such that $Q(t) = \texttt{false}$, contradicting the assumption $P \subseteq_G Q$. The tree t can be constructed from P as follows.

(A) For every $i \in [1, k]$, replace P_{x_i} with a model t_i of P_{x_i}, such that Q_v does not have embeddings in t_i (t_i exists because $P_{x_i} \not\subseteq_{G_x} Q_v$).

(B) For each $c//d$-edge (c_1, d_1) in P that is not in any P_{x_i}, there is a complete instance of (c_1, d_1) such that Q_v has no unrooted embedding in the instance (see Claim 1 below). We replace (c_1, d_1) with such a complete instance of (c_1, d_1).

The tree constructed above is clearly a model of P under G. Note that no nodes in the tree need to be merged, because of the assumption that G has no edges labeled 1 or ?.

With t constructed as above, Q cannot have an embedding in t, because any embedding of Q in t would map v to one of $x_1, \ldots, x_k$, and map Q_v into one of $t_1, \ldots, t_k$, contradicting the assumption that Q_v has no embedding in any t_i. Next we prove Claim 1.

Claim 1. For each $c//d$-edge (c_1, d_1) in P that is not in any P_{x_i}, there is a complete instance of (c_1, d_1) in which Q_v has no unrooted embedding.

Proof Since (c_1, d_1) is not in any P_{x_i}, we know $c \neq x$ (because, if $c_1 = \texttt{root}(P)$, then c_1 cannot be labeled x as the schema graph is acyclic; if $c_1 \neq \texttt{root}(P)$, then c_1 cannot be labeled x as otherwise c_1 will be one of $x_1, \ldots, x_k$, contradicting the assumption the edge (c_1, d_1) is not in any P_{x_i}). Since P cannot be chased, we know G does not imply $c \xrightarrow{x} d$. By Lemma 1 (4), there is a G-path from c to d which does not pass through x (except the case $x = d$). Therefore there is an instance of (c_1, d_1) which does not pass through any x-node (except d_1 in the case where d_1 is one of $x_1, \ldots, x_k$). Denote this instance of (c_1, d_1) by $I_{(c_1,d_1)}$. We show Q_v has no unrooted embeddings in the complete instance of (c_1, d_1) obtained from $I_{(c_1,d_1)}$. If Q_v has an unrooted embedding in the complete instance, suppose the embedding maps node v to node u, and maps Q_v into a subtree t' rooted at u, then u must be labeled x, and by construction of complete instance, t' is obtained by adding mandatory nodes under the x-node u. In other words, we can add mandatory nodes under x-node u to obtain a tree t' such that Q_v has an embedding in t'. Therefore, similar nodes would have to be added in $t_1, \ldots, t_k$ (since $\texttt{root}(t_i)$ is an x-node) such that Q_v would have embeddings in $t_1, \ldots, t_k$ as well. This contradicts the assumption that Q_v has no embeddings in $t_1, \ldots, t_k$. □

Lemma 7: *Let P be a pattern that cannot be chased, and $(\texttt{root}(P), v)$ be a //-edge in P. Then for any label $x \in \Sigma_G$, there is an instance of $(\texttt{root}(P), v)$ where the node immediately following $\texttt{root}(P)$ is not an x-node.* □

Proof Since we assumed the schema graph is acyclic, if $label(\texttt{root}(P)) = x$, then $\texttt{root}(P)$ cannot have x-child in any instance of $(\texttt{root}(P), v)$. Now let us assume $label(\texttt{root}(P)) = a$ where $a \neq x$. There must be a G-path from a to $label(v)$ where the node immediately after a is not x, otherwise G would imply $a \xrightarrow{x} label(v)$ (in the case $label(v) \neq x$) or $a \Downarrow^1 x$ (in the case $label(v) = x$), contradicting the assumption P cannot be chased. Therefore, there is an instance of $(\texttt{root}(P), v)$ where the node immediately following $\texttt{root}(P)$ is not an x-node. □

Lemma 8: *Let P be a pattern that cannot be chased. Then, if $\texttt{root}(P)$ has no /-child labeled x, then there is a model t of P such that $\texttt{root}(t)$ does not have any x-child.* □

Proof Since G is acyclic, if $\texttt{root}(P)$ is labeled x, then any model t of P cannot have other node than $\texttt{root}(t)$ that is also labeled x. Now let us assume $label(\texttt{root}(P)) = a$ ($a \neq x$). Then G cannot imply the SC $a :\downarrow x$, otherwise an $(/, x)$-child of $\texttt{root}(P)$ would be added in P by the SC-rule, contradicting the assumption that P cannot be chased. In addition, by Lemma 7, for each //-edge $(\texttt{root}(P), u)$ in P, there is an instance $I_{(\texttt{root}(P),u)}$ of $(\texttt{root}(P), u)$ where the node immediately following $\texttt{root}(P)$ is not an x-node. We can replace the edge $(\texttt{root}(P), u)$ with a complete

instance obtained from $I_{(\texttt{root}(P),u)}$, and replace P_u with a model of P_u. Finally we merge nodes in t using FCs implied by G. The tree t obtained this way is a model of P and $\texttt{root}(t)$ does not have any /-child labeled x. □

Lemma 9: *Suppose* $\texttt{root}(Q)$ *has only one /-child* v*,* $label(v) = x$*, and* $\texttt{root}(Q)$ *has no //-children. Let* P *be a pattern that cannot be chased. Then* $P \subseteq_G Q$ *if and only if* $\texttt{root}(P)$ *has a /-child* u*, such that* $label(u) = x$*, and* $P_u \subseteq_{G_x} Q_v$*.* □

Proof The 'if' direction is obvious. We show 'only if' next. Suppose $P \subseteq_G Q$. By Lemma 8, there must be a /-child u of $\texttt{root}(P)$ such that $label(u) = x$, otherwise there will be a model t of P such that $\texttt{root}(t)$ does not have any x-children, so Q would have no embedding in t, contradicting $P \subseteq_G Q$. Let $x_1, \ldots, x_k$ be all of the $(/, x)$-children of $\texttt{root}(P)$. We show there exists $i \in [1, k]$ such that $P_{x_i} \subseteq_{G_x} Q_v$. If not, that is, for every i, $P_{x_i} \not\subseteq_{G_x} Q_v$, then there would exist a model t of P such that $Q(t) = \texttt{false}$. This t can be constructed from P as follows. (i) For every i, find a model t_i of P_{x_i} under G_x, such that Q_v has no embedding in t_i, and replace P_{x_i} with t_i; (ii) For each other /-child z of $\texttt{root}(P)$, replace P_z with any model of P_z; (iii) For each //-child z of $\texttt{root}(P)$, there exists an instance of $(\texttt{root}(P), z)$ where the node immediately following $\texttt{root}(P)$ is not an x-node (by Lemma 7), replace the edge $(\texttt{root}(P), z)$ with a complete instance obtained from this instance, and replace P_z with any model of P_z. Finally we merge nodes in t using FCs implied by G. Note that no two nodes in $x_1, \ldots, x_k$ of t can be merged (otherwise they would have been merged in P), and there are no other $(x, /)$-children of $\texttt{root}(t)$. For such t, Q has no embedding in t because any such embedding would map v to one of $x_1, \ldots, x_k$, and if it maps v to x_i, then it must also map Q_v into t_i. This contradicts the assumption that Q_v has no embedding in t_i. □

Using Lemmas 2, 9 and 6, we can prove Theorem 1 by induction on the height of Q. Here the height of Q, denoted $height(Q)$, is defined as the number of nodes on the longest path of Q.

Proof [of Theorem 1] We only need to show the 'only if' direction. We do this by induction.

If $height(Q) = 1$, the theorem is trivially true.

Now suppose the theorem is true when $height(Q) = n$, we show it is also true when $height(Q) = n + 1$.

Since $P \subseteq_G Q$, we know for every child u of $\texttt{root}(Q)$, $P \subseteq_G Q_{(\texttt{root}(Q),u)}$.

Suppose u is a /-child of $\texttt{root}(Q)$. By Lemma 9, there is a /-child u' of $\texttt{root}(P)$ such that $label(u) = label(u')$, and $P_{u'} \subseteq_{G_{label(u)}} Q_u$. By the induction hypothesis, there is a homomorphism from Q_u to $P_{u'}$, thus there is a homomorphism from $Q_{(\texttt{root}(Q),u)}$ to $P_{(\texttt{root}(P),u')}$.

Suppose v is a //-child of $\texttt{root}(Q)$. By Lemma 6, there is a descendant v' of $\texttt{root}(P)$ such that $label(v) = label(v')$, and $P_{v'} \subseteq_{G_{label(v)}} Q_v$. By the induction hypothesis, there is a homomorphism from Q_v to $P_{v'}$, thus there is a homomorphism from $Q_{(\texttt{root}(Q),v)}$ to $P_{(\texttt{root}(P),v')}$.

By Lemma 2, there is a homomorphism from Q to P. □

Extension of Theorem 1. Note that the condition that "no edges in the schema graph is labeled ? or 1" is required in the proof of Lemma 6, but not in the proof of other lemmas. A closer examination of the proof of Lemma 6 shows that the above condition can be relaxed to "if any edge (x, y) is labeled ? or 1, then there is no outgoing edge from y". This is because that if the revised condition holds, then any nodes in P that can be merged by FCs must be leaf nodes. Therefore, the claim "any embedding of Q in t would map v to one of $x_1, \ldots, x_k$, and map Q_v into one of $t_1, \ldots, t_k$, contradicting the assumption that Q_v has no embedding in any t_i" in the proof of Lemma 6 is still correct (this becomes clear if we consider two cases: (1) when no two nodes in $x_1, \ldots, x_k$ can be merged by the FCs, (2) when two nodes in $x_1, \ldots, x_k$ can be merged by the FCs. In the former case the claim is true as before; in the latter case $x_1, \ldots, x_k$ must be leaf nodes in P, and v must be a leaf node in Q, thus Q_v can be mapped to x_i for all $i \in [1, k]$).

Another Special Case. Another special case is when G is a tree (but the edges can be labeled with any of *, ?, 1, and +). In this case, there is a unique path between any two nodes in G. For any path from x to y in G, suppose $x_1, \ldots, x_k$ is the sequence of nodes on the path between x and y, then G would imply $x \xrightarrow{x_1} y$, $x_1 \xrightarrow{x_2} y$, …, $x_{k-1} \xrightarrow{x_k} y$, as well as $x \Downarrow^1 x_1$, …, $x_{k-1} \Downarrow^1 x_k$, $x_k \Downarrow^1 y$. Therefore, an $x//y$-edge in a satisfiable tree pattern will be chased into a single path $x/x_1/\ldots/x_k/y$. Thus any pattern $P \in \mathcal{P}^{\{/,//,[]\}}$ satisfiable under G will be chased into a pattern with only /-edges by the LWZ constraints. Hence Theorem 1 is still correct

if G is such a schema graph, because Lemma 6 still holds in this case: P is a model of itself, and any embedding of Q in this model of P must map v to an x-descendant u of $\texttt{root}(P)$, hence if $P \subseteq_G Q$, then $P_u \subseteq_{G_x} Q_v$ (the "only if" direction of the lemma).

5. Conclusion

The paper [2] has many innovative ideas as well as many citations, and it has inspired further research such as [4]. Unfortunately, the result about chasing tree patterns using schema constraints in [2] is incorrect. However, there are conditions we can impose on the schema graph to make the result correct. The result in [4] is also incorrect, and we can correct it by imposing the same conditions as in Theorem 1 on G and requiring Q to have no /-edges. The proof is similar to that of Theorem 1. Due to page limit we omit it.

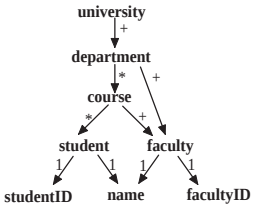


Figure 2: Example of a real-world schema graph

We believe that the revised results presented in Section 4 have practical applications, since there are real-world DTDs that can be represented as schema graphs satisfying the special conditions. For instance, a university DTD could be represented by the schema graph shown in Figure 2.

Acknowledgement This work is partially supported by ARC discovery grant DP1093404.

[1] S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava. Minimization of tree pattern queries. In *SIGMOD*, pages 497–508, 2001.

[2] L. V. S. Lakshmanan, H. Wang, and Z. J. Zhao. Answering tree pattern queries using views. In *VLDB*, pages 571–582, 2006.

[3] G. Miklau and D. Suciu. Containment and equivalence for an XPath fragment. In *PODS*, pages 65–76, 2002.

[4] J. Wang and J. X. Yu. Chasing tree patterns under recursive DTDs. In *DASFAA (1)*, pages 250–261, 2010.

[5] P. T. Wood. Containment for XPath fragments under DTD constraints. In *ICDT*, pages 300–314, 2003.