Topics in DB: Foundations of XML (CSC2538)
XPath Query Containment

Prof. Leonid Libkin
Presentation by Zheng Zhang

Dec. 2nd, 2005

1 Introduction

In this presentation, we will learn the algorithmic problem of XPath Query Containment: given two XPath queries p and q, if all documents that match p also match q, then p is contained in q, where a match between a document and a XPath query is a function from the nodes in the query to those (elements) of the document, which is root-preserving, respects node labels, and respects edge relationships.

As a comparison, we recall the definition classical relational conjunctive query containment: given two conjunctive queries p and q, if for all databases d, the set of answers to p over d is contained in that to q over d, then p is contained in q.

We expect there are quite some similarities between the solutions to the above two problems. The problem of XPath query containment is on the one hand easier, as the structures are trees rather than arbitrary relational structures, but on the other hand much harder, as the queries might involve recursion (e.g. by navigation along the descendant-axis).

2 XPath

We model XML documents as rooted trees with labels from an infinite (unranked) alphabet Σ. The symbols in Σ corresponds to XML tags. Every node corresponds to an XML element. The root of the tree corresponds to the root element of the document, denoted by root. Refer to such trees as XML trees.

We only consider a fragment of XPath: a *step* expression s: axis :: expr pred* where axis could be self, child, descendant, descendant-or-self, parent, ancestor, ancestor-or-self, following-sibling, preceding-sibling; expr is a *node test*, either a tag name or *; pred* is a possible empty sequence of items of the form $[p]$ as follows. The syntax of expression p is $p ::= s \mid s/p \mid p|p \mid /p$ where s is a step expression. An expression of the form $/p$ is called *absolute*. Others are called *relative*.

We could write $p/e/q$ instead of $p/child :: e/q$, $p//e/q$ instead of $p/descendant :: e/q$, $./p$ instead of $self :: */p$. Expression $p_0 = child :: a[descendant : d]/child :: */descendant :: c$ is equivalent to $a[.//d]/*//c$.

Given axis x and XML tree t, set $A(x,t)$ is the set of pairs (u,v) of nodes from t s.t. node u and v are in x-relation. Given expression p and XML tree t, a binary relation $R(p,t)$ is as follows. For each step $s = x :: ep_1, \ldots, p_k$, relation $R(s,t)$ is the set of pairs (u,v) of nodes s.t. (u,v) is in $A(x,t)$; v matches e ($e = *$, or the label of v is e); and each set $R(p_i,t)$ contains at least one pair (v,w).

Expression $p = s/q$, where s is a step expression and q is an expression, the relation $R(p,t)$ is $\{(u,v)|u,v,w \in t, (u,w) \in R(s,t), (w,v) \in R(q,t)\}$. $R(p|q,t) = R(p,t) \cup R(q,t)$. $R(/p,t) = \{(root,v)|(root,v) \in R(p,t)\}$. The semantics of $p_0 = child :: a[descendant : d]/child :: */descendant :: c$ is $\{(u,v)\}$ where v is an element labeled c, which is a descendant of an element with arbitrary label, which is a child of a child of u with label a. This child has to possess a descendant with label d.

For expressions p with only child and descendant axes, we could use pattern tree $T(p)$ representation.

- Each step of an expression corresponds to a node, which is a child of the node of the previous step.
- Steps along the child axis are indicated by single lines. Denote *child edges*.
- Steps along the descendant axis are indicated by double lines. Denote *descendant edges*.
- A predicate expression of a step s gives a subtree of the node corresponding to s.
- The node which corresponds to the last step of an expression (the selection node) is underlined.

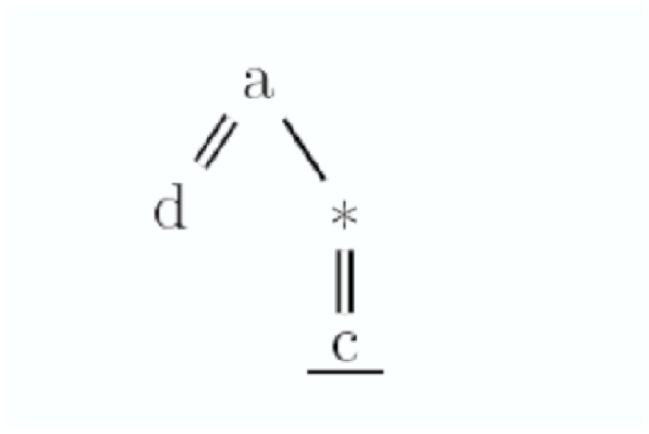


Figure 1: Pattern Tree for p_0.

Notice the order of children of a node does not carry any meaning.

Work on XPath query containment has mainly focused on the child and descendant axis. Sometimes, disjunction, predicates $[q]$ and/or the wildcard * are allowed. We refer to such fragments by writing $XP(L)$, where L is a list of the allowed components. $XP(/, [], *)$ is a fragment of child, predicates and wildcard.

3 Containment

Given two XPath expressions p and q, the general definition of containment is: $p \subseteq_2 q$ if $R(p,t) \subseteq R(q,t)$ for every XML tree t. Alternatively, we could consider whether nodes match relative to the root of the tree. Let p be an absolute expression. $R(p, root, t)$ is a set of nodes v, where $(root, v) \in R(p,t)$. We say $p \subseteq_1 q$ if $R(p, root, t) \subseteq R(q, root, t)$ for every XML tree t. A boolean containment only asks whether p and q match at all, relative to the root. Write $t \models p$ if $R(p, root, t) \neq \emptyset$. We say $p \subseteq_0 q$ if $t \models p$ implies $t \models q$ for any XML tree t.

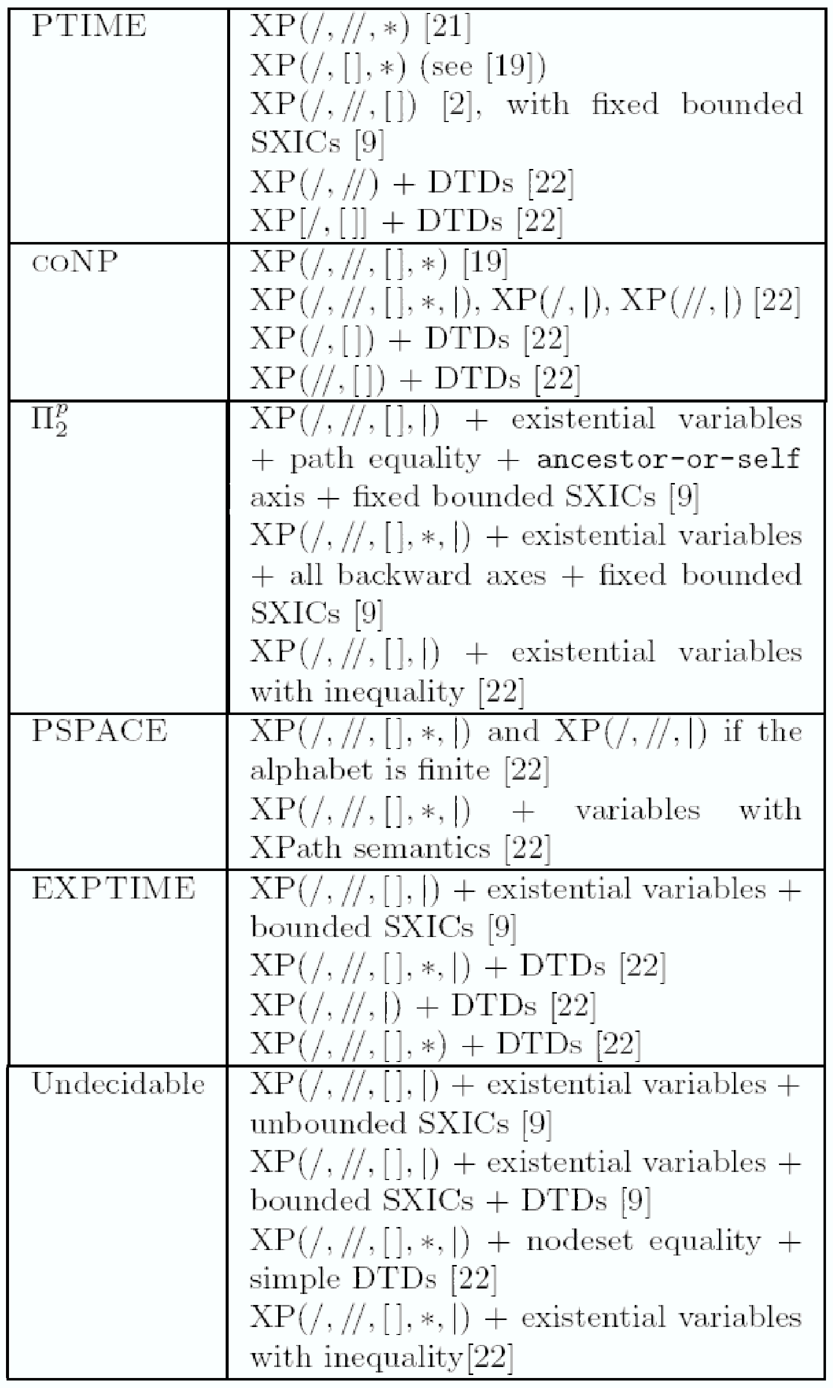
When only child and descendant axes are allowed. $\subseteq_2$ and $\subseteq_1$ are equivalent. When predicates are allowed, it is sufficient to consider Boolean queries. A tree pattern $T(p)$ can be modified into a new tree pattern $T(p')$ by adding a child to its selection node, *i.e.*, a tree for a boolean query. It has been shown that $p \subseteq_1 q$ iff $p' \subseteq_0 q'$.

Two XPath queries p and q, $p \subseteq_0 q$ w.r.t. DTD d, if $t \models p$ implies $t \models q$, for all trees t that are valid w.r.t. d.

Example 3.1 *Two XPath queries $p = /a/b//d$, $q = /a//c$, p is not contained in q in general. But $p \subseteq_0 q$ w.r.t. the following DTD d_2.*

$$\begin{aligned} root &\rightarrow a*, \\ a &\rightarrow b*|c*, \\ b &\rightarrow d+c+, \\ c &\rightarrow b?c? \end{aligned}$$

A general class of constraints studied is simple XPath Integrity Constraints (SXICs). They are reminiscent of embedded dependencies in relational databases.



PTIME	XP(/, //, ∗) [21] XP(/, [ ], ∗) (see [19]) XP(/, //, [ ]) [2], with fixed bounded SXICs [9] XP(/, //) + DTDs [22] XP[/, [ ]] + DTDs [22]
coNP	XP(/, //, [ ], ∗) [19] XP(/, //, [ ], ∗, |), XP(/, |), XP(//, |) [22] XP(/, [ ]) + DTDs [22] XP(//, [ ]) + DTDs [22]
Π_2^p	XP(/, //, [ ], |) + existential variables + path equality + `ancestor-or-self` axis + fixed bounded SXICs [9] XP(/, //, [ ], ∗, |) + existential variables + all backward axes + fixed bounded SXICs [9] XP(/, //, [ ], |) + existential variables with inequality [22]
PSPACE	XP(/, //, [ ], ∗, |) and XP(/, //, |) if the alphabet is finite [22] XP(/, //, [ ], ∗, |) + variables with XPath semantics [22]
EXPTIME	XP(/, //, [ ], |) + existential variables + bounded SXICs [9] XP(/, //, [ ], ∗, |) + DTDs [22] XP(/, //, |) + DTDs [22] XP(/, //, [ ], ∗) + DTDs [22]
Undecidable	XP(/, //, [ ], |) + existential variables + unbounded SXICs [9] XP(/, //, [ ], |) + existential variables + bounded SXICs + DTDs [9] XP(/, //, [ ], ∗, |) + nodeset equality + simple DTDs [22] XP(/, //, [ ], ∗, |) + existential variables with inequality[22]

Table 1: Complexity results for XPath containment.

Figure 2: The complexity table from [1]

4 Complexity Results

All complexities for coNP and the higher classes are tight. See Figure 2 for a complete list of complexity results for XPath containment.

5 Some Algorithmic Techniques

We present some techniques that were used to obtain upper bounds for various XPath fragments: Canonical models, Homomorphisms, Tree automata and Chase procedure. We only consider boolean containment. The fundamental fact that is used is that p is not contained in q iff there is a tree t s.t. $t \models p$ is true but $t \models q$ is false. For space reasons, we do not touch techniques for lower bounds here.

5.1 Canonical Model

The containment test is in coNP: if for a fragment X it holds that p is not contained in q iff there is a counter-example t of polynomial size in p and q.

Theorem 5.1 *Test of containment of $XP(/, //, [], *)$ is in coNP.*

Query p is not contained in q if there is a counter-example t obtained from p as follows.

- Let z be a new symbol not in p and q (the proof relies on the existence of z).
- Replace by z every * in the pattern tree $T(p)$.
- Every descendant edge is replaced by a chain of child edges with at most $m(q)+1$ interior nodes labeled by z.
- Let $m(q)$ be the max length of a chain in $T(q)$ consisting only of child edges and *-nodes.

Clearly, all the above trees match p.

5.2 Homomorphism Technique

Homomorphism is a classical characterization result for conjunctive queries against relational databases. A conjunctive query p is contained in a conjunctive query q iff there is a homomorphism from q to p. Given two XPath queries p and q, a homomorphism h from q to p maps each node of $T(q)$ to a node of $T(p)$ such that the following holds.

- The root of $T(q)$ must be mapped to the root of $T(p)$.
- If (u, v) is a child-edge of $T(q)$ then $(h(u), h(v))$ is a child-edge of $T(p)$.
- If (u, v) is a descendant-edge of $T(q)$ then $h(v)$ has to be below $h(u)$ in $T(p)$.
- If u is labeled with e other than *, then $h(u)$ also has to carry label e.

Homomorphism may not be injective.

Theorem 5.2 *Let p, q be expressions from $XP(/, //, [])$ (or from $XP(/, [], *)$). Then $p \subseteq_0 q$ iff there is a homomorphism from $T(q)$ to $T(p)$.*

However, the existence of a homomorphism is only sufficient and not necessary for containment in $XP(/, //, [], *)$.

For $XP(/, //, [], *)$ and its fragments, the homomorphism technique is a special case of the canonical model technique, *i.e.*, only one tree is tested. Let p and q be two expressions.

- Define a special chain: a chain with two edges and an intermediate node labeled with a new symbol y.

- In $T(p)$, replace every child-edge with a special chain and each descendant-edge with a normal edge. Call the new tree $t(p)$.
- In $T(q)$, replace every child-edge with a special chain. The corresponding expression is q'.
- There is a homomorphism from $T(q)$ to $T(p)$ iff $t(p) \models q'$.

5.3 The Automata Technique

The basic idea here is to compute the set C of all counter-examples and check whether C is empty. C is often represented by a tree automaton. Solving the containment problem can be accomplished as follows. Given two XPath queries p, q and d is a DTD, construct automata $\mathcal{A}(p)$, $\mathcal{A}(\bar{q})$, $\mathcal{A}(d)$ where $\mathcal{A}(p)$ accepts a tree t if $t \models p$, $\mathcal{A}(\bar{q})$ accepts a tree t if $t \not\models q$, $\mathcal{A}(d)$ accepts a tree t if t conforms to d. The product of the automata accepts all counter-example trees that conform to d. This test can be done in polynomial time.
Note: expressions p and q might be matched to different paths in a tree.

Theorem 5.3 *Containment test of $XP(/,//)$ w.r.t. DTDs is in PTIME.*

The automata techniques can be used for more expressive XPath fragments, *i.e.*, involving predicates and disjunction. Then the corresponding automata are larger.

Theorem 5.4 *Containment test of $XP(/,//,[],*,|)$ is*

- *EXPTIME-complete w.r.t. DTDs.*
- *coNP-complete w.r.t. infinite alphabet.*
- *PSPACE-complete w.r.t. finite alphabet.*

5.4 The Chase Technique

In Relational Databases, containment test with integrity constraints are solved by chase, *i.e.*, an extension of homomorphism technique. Hence, the XPath query containment can be solved as follows.

- Two XPath queries p and q could be translated into relational queries p' and q'.
- Translate the given XML constraints into relational constraints.
- Create additional general constraints for recovering some of the information lost by the translation.
- Apply relational chase to p' and check containment.

Moreover, chase could be applied to pattern trees directly. If there is a homomorphism from $T(q)$ to the chased tree of $T(p)$, we get $p \subseteq_0 q$ w.r.t. d.

6 Related Work

XPath equivalence: it could be reduced to XPath containment. When there are only forward axes and in the presence of predicates, two problems are equivalent: $p \subseteq_0 q$ iff p and $p[/q]$ are equivalent.
XPath minimization Given an expression p, find a minimal equivalent expression p'. Minimization is possible in polynomial time for an XP-fragment, if containment for this fragment can be decided via homomorphisms. Moreover, it is proved for XP(/, [], *) and XP(/, //, []) that p' is always

a subpattern of p. $XP(/, //, [], *)$ has the subpattern property, but no homomorphism property. Minimization problem is coNP-hard.
XPath evaluation Evaluation can be done in polynomial time (combined complexity) in much larger fragment of XPath considered here.
Charactering XPath Consider existential first-order logic, closure properties, conditional axes and axiomatizability of many fragments.
Containment for path queries on graphs Regular path queries in semi-structured data.
Tree pattern matching

Reference [1] Thomas Schwentick, *XPath Query Containment*, Sigmod Record, Vol. 33, No. 1, 101-109, March 2004.