XPath query containment and rewriting using views

Alin Deutsch, Ioana Manolescu, Vasilis Vassalos

View-based rewriting of XML queries

Alin Deutsch[1] Ioana Manolescu[2] Vasilis Vassalos[3]

[1]University of California in San Diego, abdeutsch@cs.ucsd.edu

[2]INRIA Saclay–Île-de-France and LRI, U. Paris Sud-XI, ioana.manolescu@inria.fr

[3]Athens University of Economics and Business, vassalos@aueb.gr

October 11, 2010

Thanks to: Pantelis Aravogliadis, Athens University of Economics and Business
Partially supported by *Agence Nationale de la Recherche*, ANR-08-DEFIS-004, and *Fondation Digiteo*

Part I

Introduction

XML and XPath

XML

- XML trees: rooted, ordered, unranked node-labeled trees
- Node labels come from an (in)finite alphabet S.
- We denote with T_S all XML trees over alphabet S.

XPath syntax

- XPath: A language for navigating and selecting nodes from XML documents.
- XPath expressions are a sequence of XPath steps: *axis* :: *nodetest*[*predicate*].
- axis: is a forward or backward axis (navigation).
- nodetest: is a label or a wildcard.
- predicate: is another step relative to the context where defined.

Tree pattern queries (twigs)

- We focus on XPath expressions with only the child and descendant axis.
- We model them as pattern trees or twigs.

Pattern tree (twig)

- Each XPath step is a node of the tree pattern.
- Steps along child axis are indicated with single lines edges (child edges).
- Steps along descendant axis are indicated with double lines edges (descendant edges).
- Predicates are denoted as subtrees of the corresponding step.
- The last step of the expression is called selection node and is denoted by underlining the nodetest.

Embedding & evaluation of XPath

Embedding

- An embedding from an XPath query p to an XML tree t, is a function $e : V_p \rightarrow V_t$, with the following properties:
- root preserving: $e(r_p) = r_t$,
- label preserving: $\forall n \in V_p$, if $n.label \neq *$, then $n.label = e(n).label$,
- structure preserving: $\forall e' = (n_1, n_2) \in E_p$, if e' is a child edge then $e(n_2)$ is a child of $e(n_1)$ in t ; otherwise $e(n_2)$ is a descendant of $e(n_1)$ in t.

XPath query evaluation

Given an XPath query p and an XML tree t, $p(t) = \cup_{e \in EB}\{(t)_{sub}^{e(o_p)}\}$, where EB is the set of all embeddings from p to t.

Part II

XPath query containment

Introduction

Definition

XPath query p is contained in q, denoted as $p \subseteq q$, if and only if $p(t) \subseteq q(t)$, for every XML tree t. (The definition can be easily extended for the case of a DTD existence).

Theorem

For XPath queries p, q there is a translation to boolean XPath queries p_0, q_0, such that $p \subseteq q$ if and only if $p_0 \subseteq q_0$ [17]

Techniques for checking containment

- Canonical models
- Homomorphism technique
- Automata technique
- Chase technique

Canonical model technique

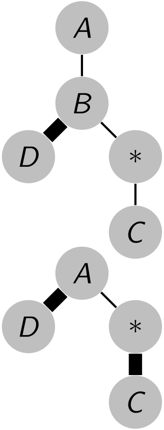
- We consider boolean tree pattern queries p, q.
- Intuition: $p \not\subseteq q$ if and only if there is a counter example XML tree t, such that $p(t) \not\subseteq q(t)$.
- If there is such a counter example it will be in the set of canonical models for p.

Canonical models for XPath query p

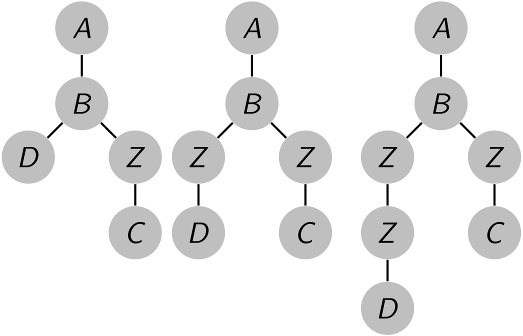
- Each canonical model is an XML tree t obtained from p.
- Wildcards: replace wildcard nodes in p with a new symbol say z not in p or q.
- Descendant Edges: replace a descendant with a chain of at most $m(q) + 1$ z-labeled nodes, where $m(q)$ is the longest chain of child edges with wildcard-labeled nodes in q.

Example

XPath queries P , Q



Canonical models for P, $M(Q) = 1$



All of them match Q, therefore $P \subseteq Q$.

Homomorphism technique

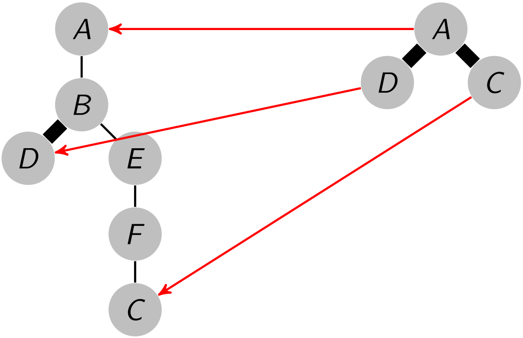
- Given two tree pattern queries p, q then: $p \subseteq q$ if and only if there is a homomorphism from q to p.
- This is not the case for XPath queries in $XP(/, //, [], *)$ where homomorphism is not necessary for containment.

Homomorphism from q to p

- An homomorphism from an XPath query q to an XPath query p, is a function $h : V_q \rightarrow V_p$, with the following properties:
- root preserving: $h(r_q) = r_p$,
- label preserving: $\forall n \in V_q$, if $n.label \neq *$, then $n.label = h(n).label$,
- structure preserving: $\forall e' = (n_1, n_2) \in E_q$, if e' is a child edge then $h(n_2)$ is a child of $h(n_1)$ in p ; otherwise $h(n_2)$ is a descendant of $h(n_1)$ in p.

Example

Homomorphism from Q to P



There is a Homomorphism from Q to P, therefore $P \subseteq Q$.

Automata technique (1/2)

- Intuition 1: Reduce the containment problem between XPath queries to the containment problem between tree automata or the languages that are accepted by these automata.
- Intuition 2: Compactly represent all counter examples that will imply no containment between two XPath queries and check if it is empty.

Theorem

Containment for XPath expression in $XP(/, //)$ in the presence of a DTD is in PTIME [25, 28]

Proof.

- Construct tree automaton P that accepts trees matching p.
- Construct tree automaton NQ that accepts trees not matching q.

□

Automata technique (2/2)

Proof.

- Construct tree automaton D that accepts trees conforming to DTD.
- Construct tree automaton $C = PxNQxD$, i.e. the cartesian product of P, NQ,D.
- C accepts all XML trees that conform to DTD and are matched by p but not q.
- Therefore, if C is empty then $p \subseteq q$.
- Every tree automaton construction and checking emptiness of C is in PTIME.

$\square$

Chase technique (1/2)

- Has been applied on tree pattern queries to obtain PTIME algorithms for containment in the presence of DTDs.
- Considered DTDs duplicate-free, choice-free.
- Intuition: To decide $p \subseteq q$ extract a set of constraints from DTD, chase query p with the constraints and try to establish a homomorphism from q to p.

Constraints Extracted from a DTD

- Functional (FC): $a \downarrow b$, node a has only one child with label b.
- Parent-Child (PC): $a \rightarrow b$, b when is desc. of a is always child.
- Sibling (SC): $a\colon B \Downarrow c$, node a with children in B has a c child.
- Cousin (CC): $a\colon b \Downarrow_c c$, node a with desc. b has also desc. c.
- Intermediate (IC): $a \rightarrow_C b$, c node is always in any path between a, b.

Chase technique (2/2)

Theorem

Let D be a duplicate-free DTD and C be the set of SC and FC constraints extracted from D. Let p, q be XPath queries in $XP(/, [\,])$ then the D-containment $p \subseteq_D q$ can be computed in PTIME using the equivalence: $p \subseteq_D q$ iff $chase(p) \subseteq q$ [35]

Theorem

Let D be a acyclic and choice-free DTD and C be the set of SC, PC, IC, CC, FC constraints extracted from D. Let p, q be XPath queries in $XP(/, //, [\,])$ then the D-containment $p \subseteq_D q$ can be computed in PTIME using the equivalence: $p \subseteq_D q$ iff $chase(p) \subseteq q$ [19]

Example: chasing an XPath query

- Let $q \equiv /a[b//i][b/d]/j$ be an XPath query in $XP(/, //, [\,])$.
- Consider the following (duplicate-free, acyclic and choice-free) DTD:

$$a \longrightarrow b, f^*, g^+, h$$

$$b \longrightarrow (c|d), l$$

$$c \longrightarrow i;\, d \longrightarrow i;\, l \longrightarrow j$$

- Apply SC $a\colon b \Downarrow h$ resulting in $q \equiv /a[b//i][b/d][h]/j$
- Apply FC $a \downarrow b$ resulting in $q \equiv /a[b[//i][d]][h]/j$
- Apply IC $a \rightarrow_l j$ resulting in $q \equiv /a[b[//i][d]][h]//l//j$
- Apply PC $l \rightarrow j$ resulting in $q \equiv /a[b[//i][d]][h]//l/j$

Part III

XPath query rewriting

XPath query rewriting

- XPath query rewriting existence problem: Given an XPath query q and a materialized v, is there any rewriting e of q using v such that for every XML tree t: $e(v(t)) = q(t)$?
- Finding (minimal) rewriting: if there is a positive answer to the rewriting existence problem, we seek to find the (minimal) rewriting query e.

Problem Statement

- Equivalent rewriting: as defined above.
- Maximally contained rewriting: a rewriting e such that for every XML tree t: $e(v(t)) \subseteq q(t)$ and $e(v(t))$ is maximal.

Rewriting existence: compensation pattern (1/2)

Concatenation Operator

- Concatenation Operator: $p \oplus q$: an XPath pattern constructed by merging the root of p and the selection node of q into one node.
- Rewriting existence problem: Decide whether an XPath compensation pattern q' exists, such that $q' \oplus v \equiv q$.

Theorem

Let q, v be two tree pattern queries with selection nodes as roots, then $q \subseteq v$ iff there exists a rewriting of q using v [37]

Theorem

The rewriting existence problem is CoNP-Hard for $XP(/, //, [], *)$ [37]

Rewriting existence: compensation pattern (2/2)

Theorem

For the subclasses of $XP(/, //, [], *)$ the rewriting existence problem is in PTIME. [37]

Lemma

Let v, q be in the subclasses of $XP(/, //, [], *)$, and let n_q be the node in the selection path of q with the same position as the selection node of v at v's selection path. If a compensation pattern q' of q using v exist, the subpattern of q rooted at n_q (i.e. $(q)_{sub}^{n_q}$) is a compensation pattern of q using v [37]

Rewriting existence: natural rewriting candidates (1/3)

Natural rewriting candidate

- Let q be a tree pattern query, with $q_{r//}$ we denote the tree pattern obtained from q by turning all edges that emanate from root to be descendant edges.
- Given a query q and a view v then q' is a natural rewriting candidate if is either $(q)_{sub}^{n_q}$ or $((q)_{sub}^{n_q})_{r//}$, where n_q is the node in the selection path of q with the same position as the selection node of v at $v's$ selection path [2]

- What if none natural rewriting candidate is a rewriting of q using v?
- Under certain condition a natural rewriting candidate is a potential rewriting, that is if a rewriting of q using v exists then one such rewriting is among the two natural rewriting candidates.

Rewriting existence: natural rewriting candidates (2/3)

Conditions for completeness of natural rewriting candidates

- query q:
- if the selection path of q by node n_q has only child edges, then $(q)_{sub}^{n_q}$ is a potential rewriting.
- if $(q)_{sub}^{n_q}$ is stable (i.e. the label of root is not wildcard or if the length of the selection path of q is at least 1 and a node emanating from root has unique label in the subtree of q starting under the root node), then it is a potential rewriting.
- View v:
- if a descendant edge enters the selection node of v, then $(q)_{sub}^{n_q}$ is a potential rewriting.
- if the selection path of v has only child edges, a potential rewriting exists among the two natural rewriting candidates.

Rewriting existence: natural rewriting candidates (3/3)

Conditions for completeness of natural rewriting candidates

- query q and View v:
- Let q be a tree pattern query and v be a view, such that the last descendant edge on the selection path of q is the k_{th} edge and the k_{th} edge at v selection path is a descendant edge, then $(q)^{n_q}_{sub}$ is a potential rewriting.

Theorem

The rewriting existence problem for $XP(/, //, [\,], *)$ under the before mentioned conditions is CoNP-Complete [2]

Finding minimal rewritings: compensation pattern (1/2)

Theorem

For the subclasses of $XP(/, //, [], *)$ the problem of finding minimal rewriting is in PTIME [37]

Proof.

- finding the minimal rewriting of q using v is equivalent of finding the minimal rewriting of the subpattern of q at node n_q (i.e. $(q)_{sub}^{n_q}$) using the subpattern of v at its selection node, say sn_v (i.e. $(v)_{sub}^{sn_v}$).
- Given a query q and a node n, we denote with $q - n$ the subpattern obtained by q by pruning the subpattern of q at node n (i.e. $(q)_{sub}^{n}$).
- the minimal compensation pattern of q using v is the $(q)_{sub}^{n_q} - R_{(v)_{sub}^{sn_v}}^{(q)_{sub}^{n_q}}$.

□

Finding minimal rewritings: compensation pattern (2/2)

rewriting Redundant node set R_v^q

- Let q,v be tree pattern queries, such that $v's$ selection node is its root.
- let n be a node in q then $(q)^n$ is a subpattern of q constructed by adding to the $(q)_n^{sub}$ the root of q and the path, as in q, connecting that root to the root of $(q)_n^{sub}$.
- C_q, C_v are sets including all children nodes of q, v roots respectively.
- $n_q \in C_q$ is redundant if there exists $n_v \in C_v$ s.t. $(q \oplus v)^{n_q} \equiv (q \oplus v)^{n_v}$.
- R_v^q: set of all redundant nodes.

Proof.

- Computing R_v^q and checking pattern equivalence is in PTIME.

□

Finding natural rewriting candidates: using stability

Generalized Normal Form (GNF)

Given a tree pattern query q with d nodes on its selection path. Then q is in $GNF_{/*}$ if for all $1 \leq i \leq d$ one of the following holds:

- A child edge enters the i node of q.
- The subpattern of q at the i_{th} node is stable.
- The subpattern of q at the i_{th} node forms a path (not a tree).

Theorem

if q is in $GNF_{/*}$, then at least one of the natural rewriting candidate is a potential rewriting [2]

Finding natural rewriting candidates: looking only for the last descendant

Theorem

if the deepest descendant on the selection path of view v is at least as deep as the deepest descendant on the selection path of the query q, then $(q)_{sub}^{n_q}$ is a potential rewriting, where n_q is the node in the selection path of q with the same position as the selection node of v at v's selection path [2]

Pattern extension

Let q be a tree pattern query and l be a label, then the l-extension of q denoted with q^{+l} is obtained:

- Add a wildcard labeled child to every leaf of query q, but the selection node (in case it is leaf).
- Add a l labeled child to the selection node of q.

Finding natural rewriting candidates: pattern extension & output lifting

Output Lifting

Let q be a tree pattern query with d nodes on its selection path. Let $1 \leq j \leq d$, then the output lifting of q denoted with $q^{j\rightarrow}$ is obtained: from q by considering as selection node the j node on the selection path of q.

Theorem

Let q, v be tree pattern queries and for some $k \leq j \leq d$, where k is the number of nodes in v selection path and d the corresponding for q, the j node of q selection path is not wildcard labeled, then:

- *there is a rewriting of q using v iff there is a rewriting of $(q^{+m})^{j\rightarrow}$ using v^{+*}.*
- *$(q^{+m})^{j\rightarrow}$ using v^{+*} has a rewriting among natural candidates iff so do for q, v [2]*

Form of the rewriting

- Given a query q and a view v with one selection node we saw that the rewriting of q using v is one pattern that is formulated by combining a rewriting pattern with v.
- The question is whether the rewriting can be expressed as the union of tree pattern queries.
- The answer is no!

Theorem

Let q, v be tree pattern queries, then any equivalent rewriting of q using v will contain only one tree pattern query [33]

A general framework for XPath/query rewriting (1/2)

- Let q and v be a tree pattern query and a view respectively with more than one node as selection node.
- A rewriting of q using v is a mapping from the selection nodes of q to the selection nodes of v, called selection node mapping (SNM).
- A rewriting is correct iff we can use it to establish an embedding from q to any XML tree t, utilizing also the embedding from v to t.
- Let L be the number of nodes in the longest star-path of q and m be the number of descendant edges in q.

A General Framework for XPath query rewriting (2/2)

$\vec{u}$ Extension of Q

Let $\vec{u} = \langle u_1, \ldots, u_m \rangle$, where for all $1 \leq i \leq m$, $u_i \geq 0$ and z be a label not in q. Then the $\vec{u}$ extension of q is an XML tree obtained from q as:

- replace all wildcard labeled nodes with the z label.
- replace the i_{th} descendant edge with a path containing u_i z labeled nodes.

Theorem

A rewriting of q using v is correct iff it is correct on the $\vec{u}$ extension of q for all $\vec{u} = \langle u_1, \ldots, u_m \rangle$, where for all $1 \leq i \leq m$, $u_i \leq L + 1$ [29]

Maximally contained rewriting: MCR existence

Useful Embedding

An embedding e from q to v is useful provided:

- e is empty and the root of q is qualified with the descendant axis.
- OR (a) Each node in the selection path of q (if mapped) is mapped to a node in the selection path of v
- and (b) For each path p in q either p is fully mapped by e or if x is the last node in p, where e is defined, and y is its successor node, then x is mapped to the selection node of v or to a descendant of that node or the edge between x, y is descendant.

Theorem

Let q, v be two tree pattern queries, then a MCR of q using v exists iff there is a useful embedding from q to v [19]

Maximally contained rewriting: MCR generation (1/2)

Clip-away tree (CAT)

Given a useful embedding e from q to v, the CAT is obtained as follows:

- Find the terminal nodes in q, i.e. nodes x s.t. $e(x)$ is defined and x has at least one child y, where $e(y)$ is not defined.
- For each child y of every terminal node x in q create a tree pattern connecting a dummy root to the subtree of q rooted at node y (i.e. $(q)^{y}_{sub}$) with an edge of the same type (child or desc.) as the edge between x and y in q.
- Merge the dummy roots of all the above tree patterns and label that root with the label of the selection node of v.

Maximally contained rewriting: MCR generation (2/2)

Algorithm

1. Compute all useful embeddings from q to v.
2. Prune the set of useful embeddings with embeddings that will lead to redundant rewritings [19]
3. Produce the set of CATs that correspond to each of useful embedding.
4. The MCR is the union of the tree patterns (contained rewriting, CR) that are in the set of CATs.

- The algorithm is exponential.
- Note that the MCR is expressed as the union of CRs that are tree pattern queries.

Part IV

XPath query rewriting under integrity constraints

Motivation

- Let $q \equiv /a//c$ be an XPath query and $v \equiv /a//b$ be a view in $XP(/, //)$.
- Consider the following (duplicate-free, acyclic and choice-free) DTD:

$$a \rightarrow d, e, f$$

$$d \rightarrow b; b \rightarrow c$$

$$c \rightarrow i; e \rightarrow i; f \rightarrow i$$

- The compensation pattern $e \equiv /b//c$ can be used to rewrite q using v.
- To compute e we need to chase query q with the IC $a \rightarrow_C b$.
- Therefore, $q \equiv /a//c \equiv /a//b//c$.
- Note that $e \equiv (q)^{b}_{sub}$.

Motivation

- Constraints enable rewritings in cases for which there would be no rewriting without the constraints.
- In what follows there is a direct reduction of the rewriting techniques of the schemaless case under the DTD existence.
- Some DTDs allow PTIME extraction of constraints and PTIME chase:

PTIME cases For DTDs

- Duplicate-free (DF) DTD: At each right-hand side of any DTD rule each element name *n* appears at most once.
- Acyclic & choice-free (ACF) DTD: At each right-hand side of any DTD rule there is no alternation. Also, there is no recursion in the DTD.

Equivalent rewriting for $XP(/, [], *)$ under DF DTDs

- Given a query q and a view v in $XP(/, [], *)$ under a DF DTD.
- Applying the compensation pattern technique we can answer the rewriting existence problem and find a (minimal) rewriting.
- Though, this technique requires to be able to decide containment between tree patterns in this setting.

Theorem

Containment of queries in $XP(/, [], *)$ in the presence of a duplicate-free DTD is CoNP-hard [24]

- Two approaches:
  incomplete but efficient
  complete but exponential

Equivalent rewriting for $XP(/, [], *)$ under DF DTDs

PTIME containment

- Extract SC and FC constraints from DTD.
- Extend them to the wildcard case, resulting in WSC and WFC constraints.
- Define the SIC (single child constraints) and the WSIC as a stronger version of the FC, that capture some cases not captured by FCs.
- Decide containment using the chase technique based on the following theorem:

Theorem

Let D be a duplicate-free DTD and C be the set of (W)FCs, (W)SCs, (W)SICs implied by D, then $q \subseteq_{SAT(D)} p$ iff $chase_C(q) \subseteq p$, if $chase_C(q)$ is 1-1 homomorphic to a subtree in $SAT(D)$ (i.e. all trees conforming to D) [4]

Equivalent rewriting for $XP(/, [\,], *)$ under DF DTDs

- Chasing with (W)SC, (W)FC and (W)SIC constraints a tree pattern in $XP(/, [\,], *)$ will not always yield a tree pattern 1-1 homomorphic to a subtree in $SAT(D)$.
- There is no set of constraints that can guarantee that [4].

Complete containment

- Intuition: reduce query q into $XP(/, [\,])$ utilizing the DTD D, so that $chase_C(q)$ is always 1-1 homomorphic to a subtree in $SAT(D)$ [35].
- Therefore, we can decide the containment $q \subseteq_{SAT(D)} p$.
- However, reduction of q from $XP(/, [\,], *)$ to $XP(/, [\,])$ by eliminating wildcard is exponential [4]

Maximally contained rewriting for $XP(/, //, [])$ ACF DTD: MCR existence

- A direct application of the useful embedding as in the schemaless case will not work.
- The reason is that the CAT induced by the embedding composed with the view might result to an unsatisfiable rewriting w.r.t. the ACF DTD.

Useful embedding DTD case

An embedding e from q to $chase_C(v)$, where C is the set of PC, SC, FC, CC, IC constraints implied from a ACF DTD D, is useful provided:

- is a useful embedding as in the schemaless case.
- AND For each node x in q that is not mapped by e, then there is a way in D starting from a rule that carries the label of the selection node of v to produce a node labeled as x.

Maximally contained rewriting for $XP(/, //, [])$ ACF DTD: MCR existence

Theorem

Let q, v be two tree pattern queries and D be an ACF DTD, then a MCR of q using v exists iff:

1. *there is a useful embedding from q to $chase_C(v)$, where C is the set of constraints implied from D*
2. *the rewriting composed by the CAT induced by this embedding and the view v is satisfiable under D [19]*

Maximally contained rewriting for $XP(/, //, [])$ ACF DTD: MCR generation

Algorithm

1. Extract constraints C implied by ACF DTD D.
2. Chase v with only the appropriate constraints from C as to establish a useful embedding from from q to $chase_C(v)$.
3. Compute useful embeddings and prune this set to avoid redundant rewritings.
4. Produce the set of CATs for each of useful embedding.
5. The MCR is the union of the tree patterns (contained rewriting: CR) that are in the set of CATs [19]

- The algorithm will yield at most one useful embedding and therefore the MCR consists of only one tree pattern.
- Note that there is no need to chase v will every constraint in C (for details see [19]).

Extensions to equivalent rewriting

Immediate Consequences of the previous techniques

- Equivalent query rewriting and containment for queries and views in $XP(/, [], *)$ are in PTIME under ACF DTD.
- Equivalent query rewriting for queries and views in $XP(/, //, [])$ is in PTIME under ACF DTD.

Part V

XPath/XQuery rewriting using multiple views

- Hidden hypothesis in the *single-view* rewriting context: the query cache (view) stores only copies of XML elements.
- Using multiple views in query rewriting allows exploiting node identifiers from the original XML tree to combine views.
- Given a query q and a set of views V, we seek to find a rewriting r obtained by intersecting materialized views in V.
- There is no interest in examining the union of these views due to the following theorem:

Theorem

If a tree pattern is equivalent to a union of tree patterns, then it is equivalent to a member of the union [12]

DAG pattern rewriting for $XP(/, //, [])$

Tree patterns extend to DAG patterns, representing the intersection of a set of views.

DAG pattern

- For each view v the $dag(v)$ is the tree pattern corresponding to v.
- $dag(v_1 \cap v_2)$ is obtained by coaleascing the roots and selection nodes of $dag(v_1)$ and $dag(v_2)$, provided there is no label conflict.

Theorem

Any DAG pattern is equivalent to the union of its interleavings [12]

DAG pattern rewriting for $XP(/, //, [\,])$

Definition

Interleaving of a DAG pattern d is a tree pattern p_i whose selection path corresponds to a path in d from its root to its selection node (MB path) and if a predicate s appears below node n in MB path, then it also appears below the corresponding node n' in p_i's selection path.

Algorithm for finding a rewriting

1. Test equivalence from dag pattern d to query q (the opposite is trivial).
2. Compute interleavings p_i from d (d is equivalent to their union).
3. Test union-freedom, i.e. only one interleaving p such that $p \equiv q$. (rule based procedure that transforms d)
4. The rewriting is $(q)^n_{sub}$, where n is the selection node of p [12]

Structural join rewriting for $XP(/, //, [], *)$

- A different approach based on view selection for rewriting a query using multiple views.
- How do we produce the set of views V to rewrite q? Use automata on tree patterns.
- Then we can find compensation patterns for each view separately and then join the results from applying these patterns to the materialized XML fragment of the corresponding views.
- Limitations: We need algorithms to perform these joins and most of these algorithms impose certain encodings in the XML trees.

- S. Abiteboul, R. Hull, and V. Vianu.
  Foundations of Databases.
  Addison-Wesley, 1995.

- F. Afrati, R. Chirkova, M. Gergatsoulis, V. Pavlaki, B. Kimelfeld, and Y. Sagiv.
  On rewriting XPath queries using views.
  EDBT, pages 168–179, 2009.

- P. Aravogliadis.
  XPath query rewriting using views.
  Master's thesis, Athens University of Economics and Business, 2007.

- P. Aravogliadis and V. Vassalos.
  Rewriting XPath queries using views under DTD constraints.
  Technical report, AUEB, 2009.
  Available at http://wim.aueb.gr/papers/xpathrewrite-ext.pdf.

- A. Arion, V. Benzaken, I. Manolescu, and Y. Papakonstantinou.
  Structured materialized views for XML queries.

In *Proc. VLDB Conf.*, pages 87–98, 2007.

A. Balmin, F. Ozcan, K. Beyer, R. Cochrane, and H. Pirahesh.
A framework for using materialized XPath views in XML query processing.
Proc. of the 30th VLDB conference, pages 60–71, 2004.

M. Benedikt, W. Fan, and F. Geerts.
XPath satisfiability in the presence of DTDs.
In *Proc. PODS 2005*, pages 25–36, 2005.

M. Benedikt and C. Koch.
XPath leashed.
ACM Comptuting Survey, 41(1), 2008.

A. Berglund, S. Boag, and D. C. et al.
XML Path Language (XPath) 2.0.
W3C.
Available at http://www.w3.org/TR/XPath20.

S. Boag, D. Chamberlin, and M. F. F. et al.

XQuery 1.0: An XML Query Language.
W3C.
Available at http://www.w3.org/TR/XQuery.

N. Bruno, N. Koudas, and D. Srivastava.
Holistic twig joins: optimal XML pattern matching.
In *Proc. SIGMOD Conference*, pages 310–321, 2002.

B. Cautis, A. Deutsch, and N. Onose.
Xpath rewriting using multiple views: Achieving completeness and efficiency.
In *Proc. WebDB 2008*, 2008.

B. Cautis, A. Deutsch, N. Onose, and V. Vassalos.
Efficient rewriting of XPath queries using query set specifications.
Proceedings of the VLDB Endowment, 2:301–312, 2009.

A. Deutsch and V. Tannen.
Containment and integrity constraints for XPath.
In *KRDB Workshop*, 2001.

A. Halevy.
Answering queries using views: A survey.
VLDB J., 10(4):270–294, 2001.

J. E. Hopcroft and J. D. Ullman.
Introduction to Automata Theory, Languages and Computation.
Addison-Wesley, 1979.

B. Kimelfeld and Y. Sagiv.
Revisiting redundancy and minimization in an XPath fragment.
In *EDBT*, pages 61–72, 2008.

L. Lakshmanan, G. Ramesh, H. Wang, and Z. Zhao.
On testing satisfiability of tree pattern queries.
Proc. of the 30th VLDB conference, pages 120–131, 2004.

L. Lakshmanan, H. Wang, and Z. Zhao.
Answering tree pattern queries using views.
Proc. of the 32th VLDB conference, pages 571–582, 2006.

B. Mandhani and D. Suciu.

Query caching and view selection for XML databases.
Proc. of the 31th VLDB conference, page 2005, 469-480.

M. Marx.
Xpath with conditional axis relation.
In *Proc. EDBT 2004*, pages 477–494, 2004.

G. Miklau and D. Suciu.
Containment and equivalence for an XPath fragment.
Journal of the ACM, 51(1), 2004.

T. Milo and D. Suciu.
Index structures for path expressions.
Proc. Database Theory - ICDT '99, 7th International Conference, pages 277–295, 1999.

M. Montazerian, P. Wood, and S. Mousavi.
XPath query satisfiability is in ptime for real-world dtds.
XSym, pages 17–30, 2007.

F. Neven and T. Schwentick.

On the complexity of XPath containment in the presence of disjunction, DTDs, and variables.
Proc. Logical Methods in Computer Science, 2, 2006.

D. Olteanu, H. Meuss, T. Furche, and F. Bry.
XPath: Looking forward.
In *Workshop on XML-Based Data Management*, 2002.

Y. Papakonstantinou and V. Vassalos.
The Enosys Markets Data Integration Platform: Lessons from the trenches.
In *CIKM*, pages 538–540, 2001.

T. Schwentick.
XPath query containment.
SIGMOD Record, 33(1):101–109, 2004.

J. Tang and S. Zhou.
A theoretic framework for answering XPath queries using views.
XSym, 2005.

N. Tang, J. X. Yu, M. T. Ozsu, B. Choi, and K. Wong.
Multiple materialized view selection for XPath query rewriting.
In *Proc. ICDE 2008*, pages 873–882, 2008.

B. ten Cate and C. Lutz.
The complexity of query containment in expressive fragments of XPath 2.0.
In *Proc. PODS 2007*, pages 73–82, 2007.

J. Wang and J. X. Yu.
XPath rewriting using multiple views.
DEXA, 2008.

J. Wang, J. X. Yu, and C. Liu.
On tree pattern query rewriting using views.
Web Information Systems Engineering,WISE, pages 1–12, 2007.

P. Wood.
Minimizing simple XPath expressions.
In *Proc. WebDB 2001*, pages 13–18, 2001.

P. Wood.
Containment for XPath fragments under DTD constraints.
In *Proc. ICDT 2003*, pages 300–314, 2003.

W. Xu.
The framework of an XML semantic caching system.
In *Proc. WebDB 2005*, 2005.

W. Xu and Z. M. Ozsoyoglu.
Rewriting XPath queries using materialized views.
In *Proc. of the 31st VLDB conference*, pages 121–132, 2005.