Containment for XPath Fragments under DTD Constraints

Peter Wood

School of Computer Science and Information Systems
Birkbeck College
University of London
United Kingdom
email: ptw@dcs.bbk.ac.uk

Outline

- introduction, motivation and background
- related work
- containment under DTDs is decidable for $\mathrm{XP}^{\{[\,],*,//,|\}}$
- PTIME containment under *duplicate-free* DTDs for $\mathrm{XP}^{\{[\,]\}}$
- limitations of constraints implied by DTDs
- future work

Introduction

XPath is a simple language for selecting nodes from XML documents, used in

- other W3C recommendations, e.g.,
  - XQuery
  - XPointer
  - XSLT
- XML publish/subscribe systems
- active rule systems for XML

Motivation

- efficient evaluation of XPath queries crucial when
  - large number of queries (e.g., publish/subscribe) or
  - large repository and
  - high throughput required
- can be achieved using
  - physical cost-based optimisation (indexes, etc.)
  - logical equivalence of queries

Publish/Subscribe Example—XML Document

Air traffic control data from US Department of Transport
(example from Snoeren *et al.*, MIT)

```
<flight>
  <id airline="AA">1021</id>
  <flightleg>
    <speed>512</speed>
    <altitude>290</altitude>
    <coordinate>
      <lat>4928N</lat>
      <lon>12003W</lon>
    </coordinate>
  </flightleg>
</flight>
```

Publish/Subscribe Example—XPath Queries

- `/flight[flightleg/altitude < 100]`

  flights below 10 000 feet

- `/flight[id[@airline = 'AA']]`

  American Airlines flights

- `/flight[substring-before(string(flightleg/coordinate/lat), 'N') > 2327]`

  flights currently north of the Tropic of Cancer

XML Trees

- let Σ be a finite alphabet of XML element names

- a *document tree* (or *tree*) over Σ is an ordered, unranked finite structure with nodes labelled by element names from Σ

- the set of all trees over Σ is denoted by T_Σ

- for tree $t \in T_\Sigma$,
  - the root of t is denoted by $root(t)$,
  - the nodes of t by $nodes(t)$, and
  - the label of node $x \in nodes(t)$ by $\lambda(x) \in \Sigma$

Syntax of XPath Fragments

Syntax of an XPath query P in fragment XP$^{\{[\,],*,//,|\}}$ given by grammar:

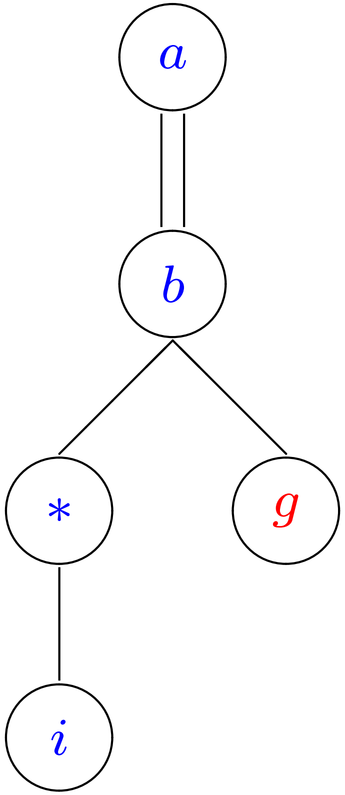
$$P \quad ::= \quad P \,/\, P \;\mid\; P \,//\, P \;\mid\; P\,[\,P\,] \;\mid\; P \mid P \;\mid\; * \;\mid\; n$$

- other fragments include XP$^{\{[\,]\}}$, XP$^{\{[\,],*\}}$, XP$^{\{[\,],//\}}$ and XP$^{\{[\,],*,//\}}$
- n denotes the name of an element
- $//$ captures descendant relationships
- $*$ matches any element

Given query Q in XP$^{\{[\,],*,//,|\}}$ and tree $t \in T_\Sigma$, $Q(t)$ denotes the set of nodes that is the result of evaluating Q on t

XPath Query as a Tree Pattern

The tree pattern for the XPath query $a//b[*/i]/g$



g is the *result* node

selects g-nodes that are children of b-nodes, such that the b-nodes are both descendants of the root a-node and have an i-node as a grandchild

Containment and Equivalence of XPath Queries

- *containment* of XPath queries can be used
  - to show equivalence of queries for optimization
  - to determine triggering of active rules and those in XSLT
  - for inference of keys based on XPath

- for XPath queries P and Q,
  - P *contains* Q, written $P \supseteq Q$, if for all trees $t \in T_{\Sigma}$, $P(t) \supseteq Q(t)$
  - P is *equivalent* to Q, written $P \equiv Q$, if $P \supseteq Q$ and $Q \supseteq P$

Document Type Definitions (DTDs)

- a *document type definition* (DTD) D over Σ consists of
  - a root type in Σ, denoted $root(D)$, and
  - a mapping (or *production*) $a \rightarrow R^a$ that associates with each $a \in \Sigma$ a regular expression R^a over Σ (the *content model* of a)

$$\begin{array}{lcl} a & \rightarrow & ((b*, c) \mid d) \\ b & \rightarrow & (g?, h?) \end{array}$$

- tree $t \in T_\Sigma$ *satisfies* DTD D over Σ if
  - $\lambda(root(t)) = root(D)$ and
  - for each node x in t with sequence of children $y_1, \ldots, y_n$, the string $\lambda(y_1) \cdots \lambda(y_n)$ is in $L(R^{\lambda(x)})$

Containment and Equivalence under DTDs

- let $SAT(D)$ denote the set of trees satisfying DTD D

- for XPath queries P and Q,
  - P *D-contains* Q, written $P \supseteq_{SAT(D)} Q$, if for all trees $t \in SAT(D)$, $P(t) \supseteq Q(t)$
  - P is *D-equivalent* to Q, written $P \equiv_{SAT(D)} Q$, if $P \supseteq_{SAT(D)} Q$ and $Q \supseteq_{SAT(D)} P$

Constraints Implied by a DTD

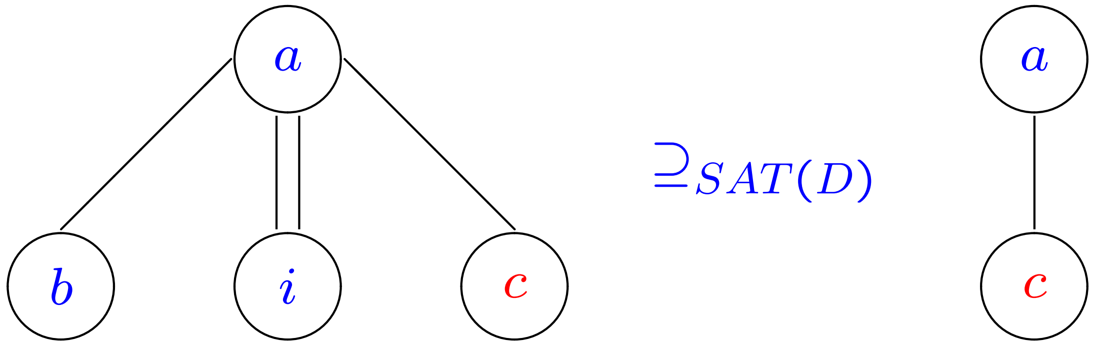
Consider DTD D:

$$
\begin{aligned}
a &\rightarrow (b, ((b, c)|d)) \\
b &\rightarrow ((e|f), (g|h)) \\
e &\rightarrow (i) \\
f &\rightarrow (i)
\end{aligned}
$$

- every a-node must have a b-node as a *child*
- every b-node must have an i-node as a *descendant*
- every path from an a-node to an i-node passes through a b-node
- if an a-node has a d-child, it has at most one b-child

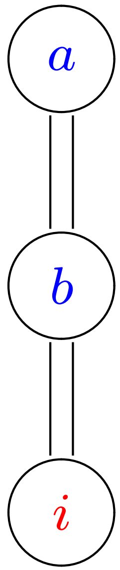
Example of D-Containment

every a-node must have a b-child and an i-descendant

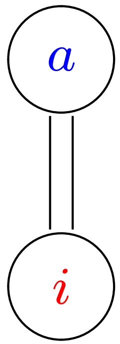


Example of D-Containment

every path from an a-node to an i-node must pass through a b-node

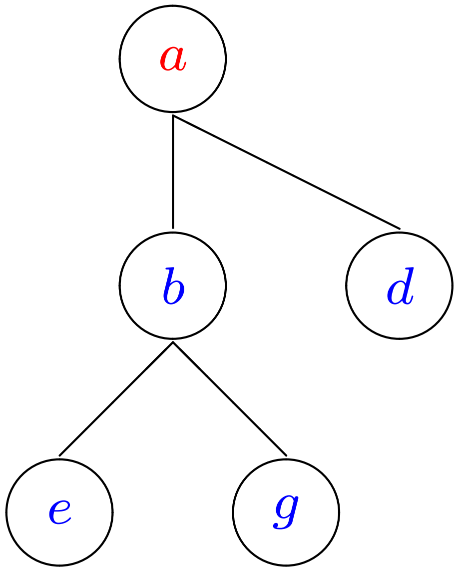


$\supseteq_{SAT(D)}$

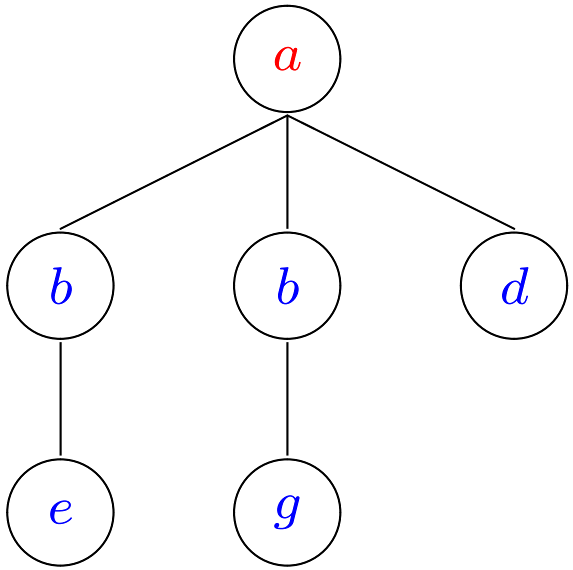


Example of D-Containment

if an a-node has a d-child, it has at most one b-child



$$\supseteq_{SAT(D)}$$



Related Work

- in the absence of constraints, containment was shown to be
  - in PTIME for queries in $XP^{\{[\,],//\}}$ [ACLS01]
  - coNP-complete for queries in $XP^{\{[\,],*,//\}}$ in [MS02]
- with constraints, containment was shown to be
  - in PTIME for queries in $XP^{\{[\,],//\}}$ with *child*, *descendant* and *type co-occurrence* constraints [ACLS01]
  - coNP-complete for $XP^{\{[\,]\}}$ with DTDs in [Wood01]
  - undecidable for $XP^{\{[\,],*,//,|\}}$ plus variables and equality, and various constraints, some implied by DTDs [DT01]
  - comprehensive classification for DTDs in [NS03]

Contributions

- that containment under DTDs is decidable (and EXPTIME-complete) for $\mathrm{XP}^{\{[\,],*,//,|\}}$

- that if DTD D is *duplicate-free*, then D-containment for $\mathrm{XP}^{\{[\,]\}}$ is captured by two types of simple constraint implied by D, and can be decided in PTIME

- that no set of constraints less expressive than those that express exactly the *unordered* language generated by each regular expression in DTD D is necessary and sufficient for D-containment for $\mathrm{XP}^{\{[\,]\}}$

Decidability of D-Containment for XP$^{\{[\,],*,//,|\}}$

- given query Q in XP$^{\{[\,],*,//,|\}}$ and alphabet Σ for DTD D, we can construct a regular tree grammar (RTG) G such that the set of trees generated by G is precisely the set of trees in T_Σ that satisfy Q

- result then follows from the facts that
  - DTDs are a special case of RTGs
  - RTGs are closed under intersection
  - containment is decidable (and EXPTIME-complete) for RTGs

- same result is proved independently by Neven and Schwentick

Regular Tree Grammars (RTGs)

A *regular tree grammar* (RTG) G is a 4-tuple $\langle \mathbf{\Sigma}, N, P, n_0 \rangle$, where

1. $\mathbf{\Sigma}$ is a finite set of element names

2. N is a finite set of nonterminals

3. P is a finite set of productions of the form

   $$n \longrightarrow a(R)$$

   where $n \in N$, $a \in \mathbf{\Sigma}$, and R is a regular expression over N

4. $n_0 \in N$ is the start symbol

RTG Corresponding to a Query

Given alphabet $\Sigma = \{a_1, \ldots a_k\}$ and query Q in XP$^{\{[\,],*,//,|\}}$ with m nodes, construct RTG G from Q as follows:

- number each node in Q uniquely, with the root node numbered 1
- contruct RTG $G = \langle \Sigma, N, P, n_1 \rangle$ *corresponding to* Q inductively, where $N = \{n_1, \ldots, n_m, n_\Sigma\}$
- use $n \to \Sigma\ (r)$ as shorthand notation for the set of productions

$$\begin{array}{rcl} n & \to & a_1\ (r) \\ & \vdots & \\ n & \to & a_k\ (r) \end{array}$$

- nonterminal n_Σ generates arbitrary tree over Σ: $n_\Sigma \to \Sigma\ (n_\Sigma^*)$

RTG Productions for a Query

Ignore $*$ and $|$ for simplicity:

1. If node i in Q is a leaf node with label $a_j \in \Sigma$, then P includes

$$n_i \;\rightarrow\; a_j\ (n_\Sigma^*)$$

2. If node i in Q has label $a_l \in \Sigma$ and has child nodes $j_1, \ldots, j_m$, then P includes

$$n_i \;\rightarrow\; a_l\ (n_\Sigma^* \;\&\; n_{j_1} \;\&\; n_\Sigma^* \;\&\cdots\;\&\; n_\Sigma^* \;\&\; n_{j_m} \;\&\; n_\Sigma^*)$$

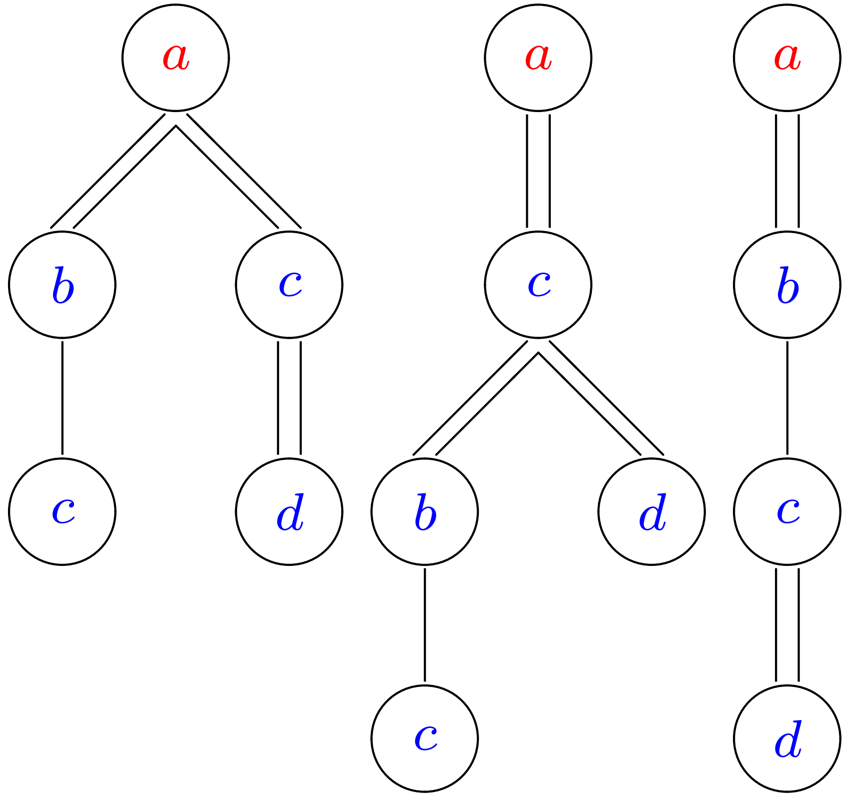
3. If node i in Q is connected to its parent by a descendant edge, then P includes

$$n_i \;\rightarrow\; \Sigma\ (n_\Sigma^*\ n_i\ n_\Sigma^*)$$

Contractions of a Query

A *contraction* of query Q_1 is a query Q_2 comprising a subset of Q_1's nodes such that there is a containment mapping from Q_1 to Q_2

Some contractions for the XPath query $a[.//b/c][.//c//d]$:



Decidability Result

- let D be a DTD over Σ
- Q_1 and Q_2 be queries over Σ in $\mathrm{XP}^{\{[\,],*,//\}}$
- G_1 and G_2 be the RTGs corresponding to the sets of contractions of Q_1 and Q_2, respectively
- then $Q_1 \supseteq_{SAT(D)} Q_2$ if and only if $D \cap G_1 \supseteq D \cap G_2$
- so containment for queries in $\mathrm{XP}^{\{[\,],*,//\}}$ under DTDs is decidable and, in fact, EXPTIME-complete

PTIME Classes

- deciding containment under DTDs is coNP-complete for $\mathrm{XP}^{\{[\,]\}}$ [Wood, WebDB01]

- so consider subclasses of $\mathrm{XP}^{\{[\,]\}}$ and subclasses of DTDs

- some constraints imposed by DTDs not relevant to $\mathrm{XP}^{\{[\,]\}}$

- look for classes of simple constraints implied by a DTD D which are necessary and sufficient to show D-containment
  - *sibling constraints* (SCs)
  - *functional constraints* (FCs)

Sibling Constraints

- let $t \in T_{\Sigma}$ be a (document) tree, $a, c \in \mathbf{\Sigma}$ be element names, and $B \subseteq \mathbf{\Sigma}$ be a set of element names

- t *satisfies* the *sibling constraint* (SC)

$$a : B \Downarrow c$$

  if whenever a node labelled a in t has children labelled with each $b \in B$, it has a child node labelled with c

- when $B = \emptyset$, the SC is called a *child constraint*

Duplicate-Free XPath Queries

- XPath query P in $\mathrm{XP}^{\{[\,]\}}$ is *duplicate-free* if, for each element n in P, each element name labels at most one child of n
- e.g., $a[b[e][g]][d]$ is duplicate-free, while $a[b/e][b/g][d]$ is not
- let P and Q be duplicate-free queries in $\mathrm{XP}^{\{[\,]\}}$
- let S be the set of SCs implied by DTD D over Σ
- $SAT(S)$ denotes set of trees in T_Σ which satisfy each SC in S
- if Q is D-satisfiable, then $P \supseteq_{SAT(D)} Q$ if and only if $P \supseteq_{SAT(S)} Q$
- $P \supseteq_{SAT(D)} Q$ can be decided in PTIME (if SCs are given)

Functional Constraints (FCs)

- let $t \in T_{\Sigma}$ and $a, b \in \Sigma$ be element names

- t *satisfies* the *functional constraint* (FC) $a \downarrow b$ if no node labelled a in t has two distinct children labelled with b

- if C is a set of SCs and FCs over Σ, then $SAT(C)$ denotes the set of trees in T_{Σ} which satisfy each SC and FC in C

Containment Under Duplicate-Free DTDs

- DTD D is *duplicate-free* if, in each content model in D, each element name appears at most once

- e.g., $a \rightarrow ((b*, c) \,|\, d)$ is duplicate-free, while $a \rightarrow (b, ((b, c)|d))$ is not

- let P and Q be queries in XP$^{\{[\,]\}}$

- let C be the set of *sibling constraints* (SCs) and *functional constraints* (FCs) implied by duplicate-free DTD D over Σ

- if Q is D-satisfiable, then $P \supseteq_{SAT(D)} Q$ if and only if $P \supseteq_{SAT(C)} Q$

Complexity

- D-satisfiability of queries in XP$^{\{[\,]\}}$ can be tested in PTIME if D is duplicate-free

- given SC s and *duplicate-free* DTD D, whether D implies s can be decided in PTIME

- given FC f and DTD D, whether D implies f can be decided in PTIME

- for P and Q in XP$^{\{[\,]\}}$ and Q being D-satisfiable, $P \supseteq_{SAT(C)} Q$ can be tested using a variant of the *chase*

The Chase

- *chase* of Q by C, denoted $chase_C(Q)$
  - apply each FC in C to Q (only polynomially many of them)
  - for "corresponding" nodes x in P and u in Q such that $\lambda(x) = \lambda(u)$, if x has child y and u has no child with label $\lambda(y)$, check if D implies $\lambda(u) : B \Downarrow \lambda(y)$, where B denotes the set of labels of children of u in Q
  - if so, add a node v as a child of u with $\lambda(v) = \lambda(y)$
- $P \supseteq_{SAT(C)} Q$ if and only if $P \supseteq chase_C(Q)$
- $P \supseteq_{SAT(D)} Q$ can be decided in PTIME

Limitations of Constraints

Can we extend the classes of constraints to capture D-containment of $\mathrm{XP}^{\{[\,]\}}$ queries when neither D nor the queries are duplicate-free?

- for string w, let $[w]$ denote the *bag* of symbols appearing in w

- if symbol a_i appears m_i times in w, $1 \leq i \leq k$, then

$$[w] = \{a_1^{m_1}, \ldots, a_k^{m_k}\}$$

- the *unordered regular language* denoted by regular expression R, written $UL(R)$, is defined as

$$UL(R) = \{[w] \mid w \in L(R)\}$$

Limitations of Constraints—Constructing the DTD

- let Σ be the set of symbols used in R, and w be an arbitrary string over Σ, where $[w] = \{c_1^{m_1}, \ldots, c_k^{m_k}\}$

- D contains productions

$$\begin{array}{rcl} b & \rightarrow & R \\ c_i & \rightarrow & (d_1 \mid \cdots \mid d_{m_i}), 1 \leq i \leq k \\ a & \rightarrow & ((b, b^+, e) \mid (f, b)) \end{array}$$

  - where each d_j, $1 \leq j \leq m_i$, is distinct
  - (an a-node in a tree t satisfying D has an e-child if and only if it has at least two b-children)

Limitations of Constraints—the Queries

- query Q_1 is

$$\begin{array}{l} a[b/c_1/d_1]\cdots[b/c_1/d_{m_1}] \\ [b/c_2/d_1]\cdots[b/c_2/d_{m_2}] \\ \qquad\qquad \vdots \\ [b/c_k/d_1]\cdots[b/c_k/d_{m_k}][e] \end{array}$$

- query Q_2 is the same as Q_1 without the predicate $[e]$

- $Q_1 \supseteq_{SAT(D)} Q_2$ if and only if $[w] \notin UL(R)$

- none of D, Q_1 or Q_2 is duplicate-free

- constraints less powerful than those which characterize unordered regular languages cannot capture query containment for XP$^{\{[\,]\}}$

Future work

- characterise and determine complexity of D-containment for
  - other classes of XPath queries
  - other practical restrictions on DTDs
- incorporate optimizations into XML servers and active rule systems
- determine the utility of optimizations through experimentation