Performance Analysis of Algorithms to Reason about XML Keys

Emir Muñoz Jiménez

University of Santiago de Chile
Yahoo! Labs LatAm

Joint work with F. Ferrarotti, S. Hartmann, S. Link, M. Marin

DEXA 2012 @ Vienna, Austria, 3rd September 2012

Contribution

- An efficient implementation of an algorithm that decides the implication problem for a tractable and expressive class of XML keys.
- Performance Analysis:
  - Implication problem over large sets of XML Keys.
  - Non-redundant covers of XML keys and validation of XML documents.
- Our experiments show that reasoning about expressive notions of XML keys can be done efficiently in practice and scales well.

Outline

Introduction

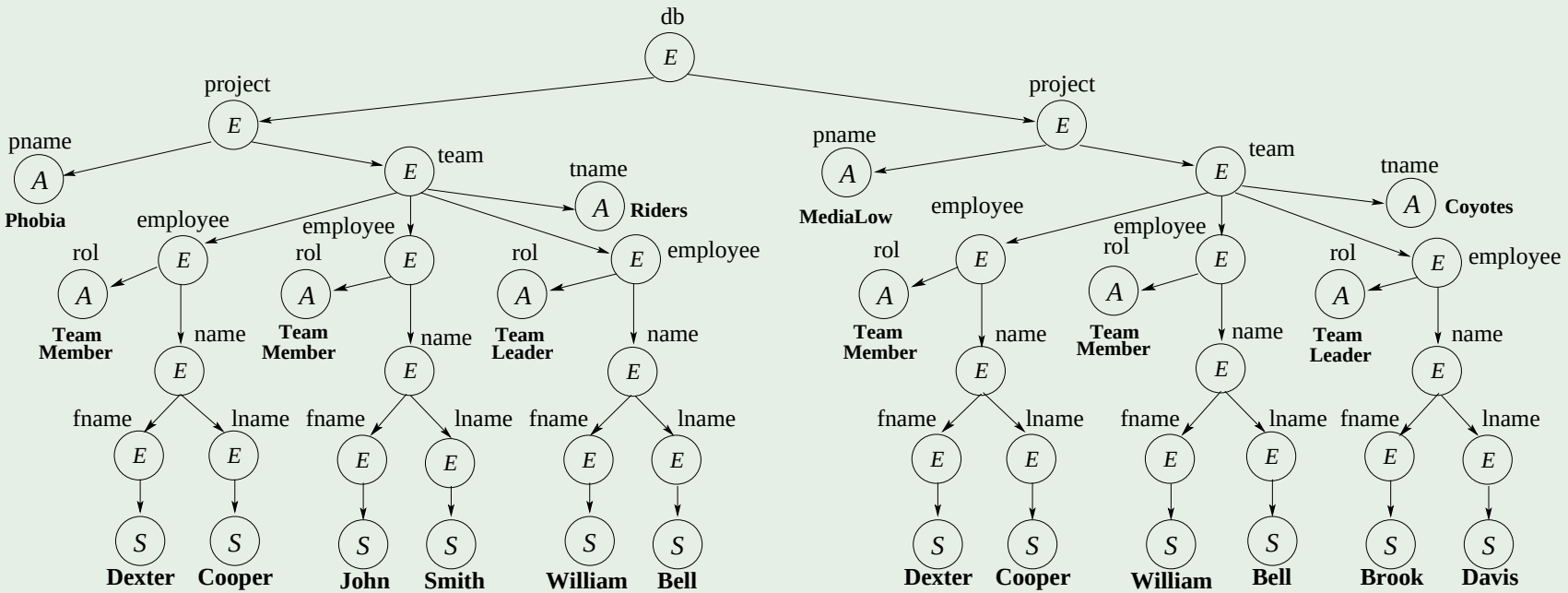
Keys

- XML: de-facto standard for Web data exchange and integration.
- XML $\leftrightarrow$ data storage $\leftrightarrow$ data processing.
- Both industry and academia have long since recognized the importance of keys in XML data management.
- Several notions of XML keys have been proposed. Most influential is due to Buneman et al. who defined keys on the basis of an XML tree model similar to the one suggested by DOM and XPath.

Introduction

XML Tree Example (1/2)

Example (An XML tree representing an XML document)



Introduction

XML Tree Example (2/2)

Some Constraints: (*gathered as business rules*)

(a) A **project** node is identified by **pname**, no matter where the **project** node appears in the document.

(b) A **team** node can be identified by **tname** relatively to a **project** node.

(c) Within any given subtree rooted at **team**, an **employee** node is identified by **name**.

Introduction

XML Tree Example (2/2)

Some Constraints: (*gathered as business rules*)

(a) A **project** node is identified by **pname**, *no matter where* the **project** node appears in the document. *Absolute key*

(b) A **team** node can be identified by **tname** *relatively to* a **project** node. *Relative key*

(c) *Within any* given subtree rooted at **team**, an **employee** node is identified by **name**. *Relative key*

Related Work & Motivation

- Keys are important in many perennial tasks in database management.
- The hope is that keys will turn out to be equally beneficial for XML.
- In practice, expressive yet tractable notions of XML keys have been ignored so far.
- We initiate an empirical study of the fragment of XML keys with nonempty sets of simple key paths.
  - One of the most fundamental questions on keys is: Can we decide if a new key holds given a set of known keys? (*Logical Implication Problem.*)

Related Work & Motivation

Key implication example

Some Constraints: (*gathered as business rules*)

(a) A **project** node is identified by **pname**, no matter where the **project** node appears in the document.

(b) A **team** node can be identified by **tname** relatively to a **project** node.

(c) Within any given subtree rooted at **team**, an **employee** node is identified by **name**.

Suppose, the consideration of a further key:

(d) A **project** node can be identified in the document by its child nodes **pname** and **team**.

Related Work & Motivation

Key implication example

Some Constraints: (*gathered as business rules*)

(a) A **project** node is identified by **pname**, no matter where the **project** node appears in the document.

(b) A **team** node can be identified by **tname** relatively to a **project** node.

(c) Within any given subtree rooted at **team**, an **employee** node is identified by **name**.

Suppose, the consideration of a further key:

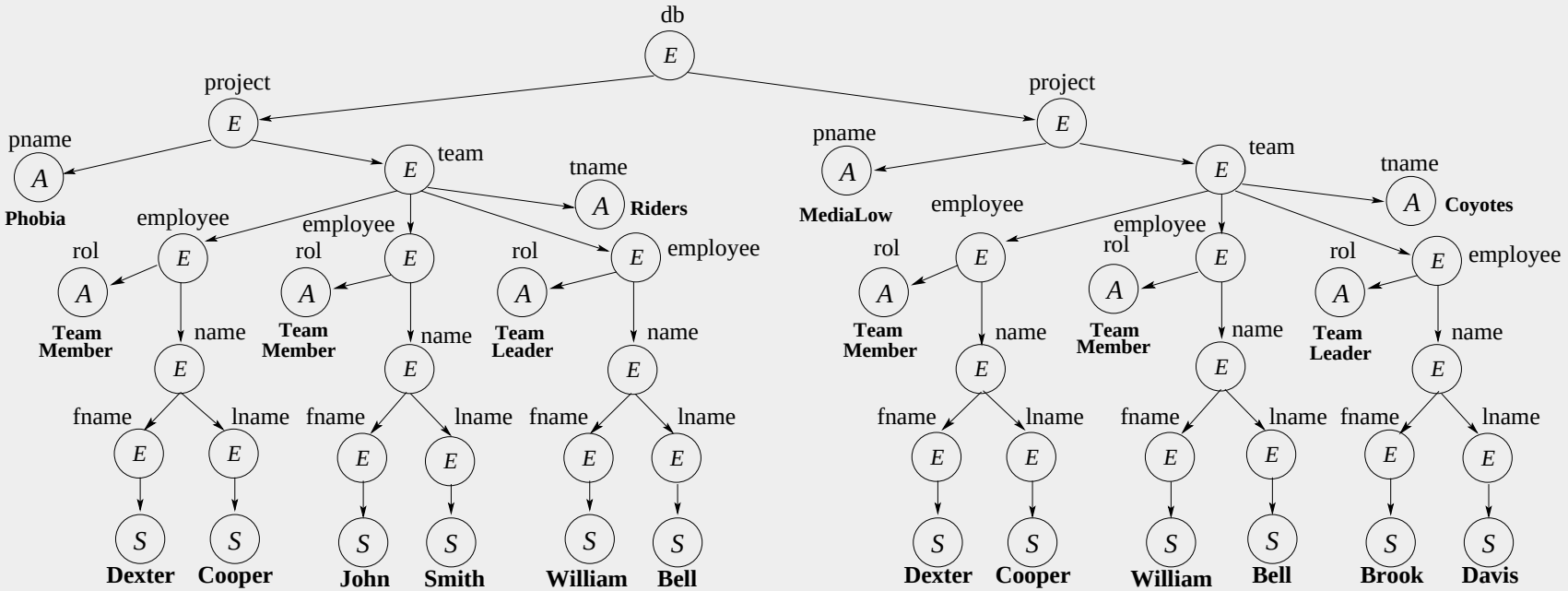
(d) A **project** node can be identified in the document by its child nodes **pname** and **team**. **key (a) actually implies (d)!! because (d) is a superkey of (a).**

Keys for XML

Definitions (1/2)

Definition (Value equality)

u is value equal to v ($u =_v v$) if the trees rooted at u and v are isomorphic by an isomorphism that is the identity on string values. (e.g., element nodes **employee** for "Dexter Cooper")



Keys for XML

Definitions (2/2)

Some Constraints: (in class $\mathcal{K}$ of XML keys)

(a) A **project** node is identified by **pname**, no matter where the **project** node appears in the document.

$$(\varepsilon, (project, \{pname\}))$$

(b) A **team** node can be identified by **tname** relatively to a **project** node.

$$(project, (team, \{tname\}))$$

(c) Within any give subtree rooted at **team**, an **employee** node is identified by **name**.

$$(\_^*.team, (employee, \{name\}))$$

(d) A **project** node can be identified by its child nodes **pname** and **team**.

$$(\varepsilon, (project, \{pname, team\}))$$

Deciding XML Key Implication

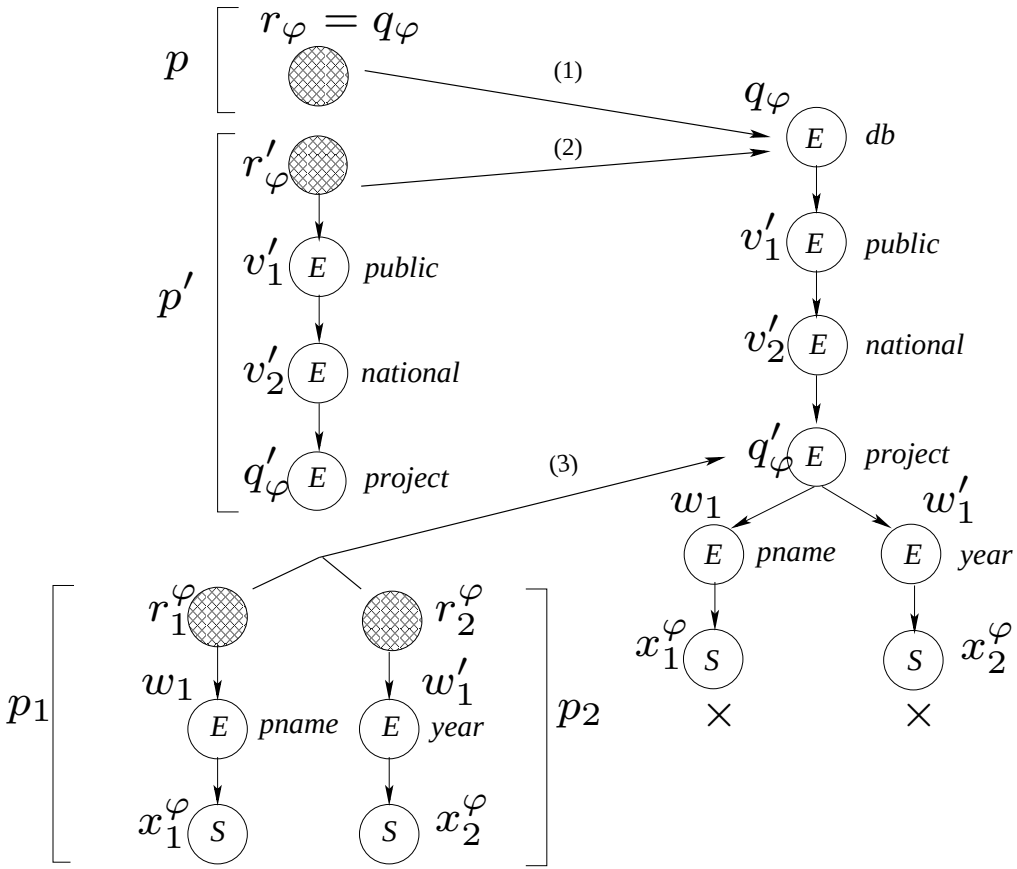
Notions

- We say that Σ *implies* φ, denoted by $\Sigma \models \varphi$, iff every finite XML tree T that satisfies all $\sigma \in \Sigma$ also satisfies φ.
- Implication Problem for a class $\mathcal{C}$ of XML keys: given any $\Sigma \cup \{\varphi\}$ in $\mathcal{C}$, decide whether $\Sigma \models \varphi$.
- e.g., let be $\Sigma = \{(a), (b), (c)\}$ and $\varphi = (d)$, then $\Sigma \models \varphi$ is true.
- e.g., let be $\Sigma = \{(a), (b)\}$ and $\varphi = (c)$, then $\Sigma \models \varphi$ is false.
- Hartmann & Link characterize the implication problem for the class $\mathcal{K}$ of XML keys in terms of the of reachability problem for fixed nodes in a suitable digraph.

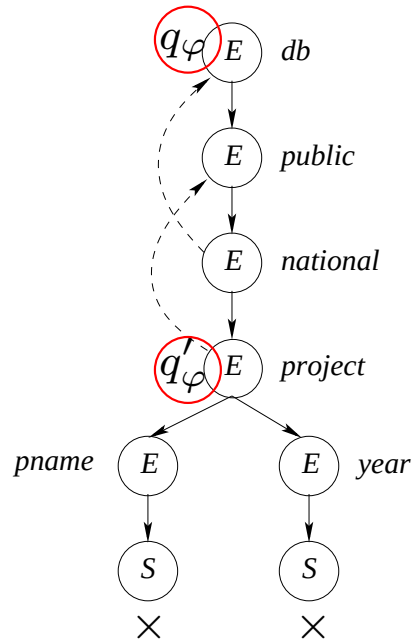
Deciding XML Key Implication

Mini-trees and Witness Graphs (Hartmann & Link, 2009)

$$\varphi = (\varepsilon, (public._^*.project, \{pname.S, year.S\}))$$



(a) Mini tree $T_{\Sigma,\varphi}$

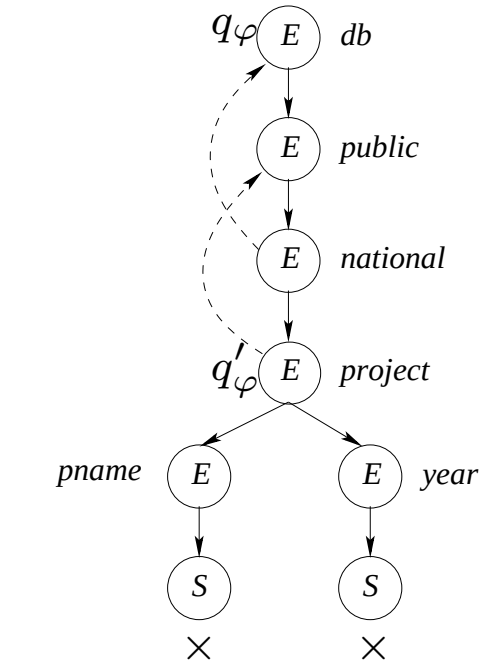


(b) Witness graph $G_{\Sigma,\varphi}$

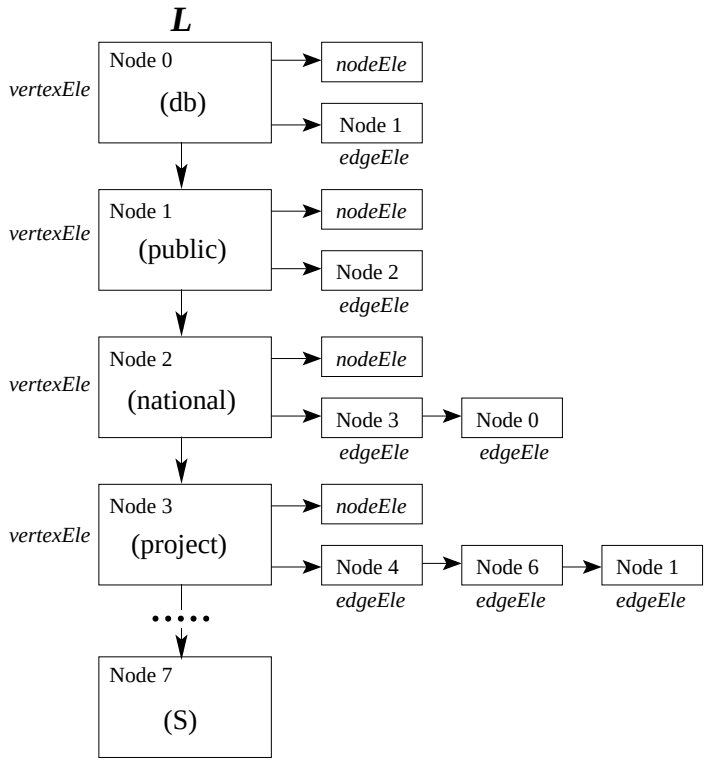
An Efficient Implementation

Data structures and implementation

- Suitable data structures for Mini-tree & Witness-graph.



(c) Witness graph $G_{\Sigma,\varphi}$



(d) Adj. list for $G_{\Sigma,\varphi}$

XML Key Reasoning for Document Validation

Applications

- Fast algorithms for the validation of XML documents against keys are crucial to ensure the consistency and semantic correctness of data stored or exchanged between applications.
- We use our implementation of the implication problem to compute non-redundant covers for sets of XML keys.
- Which in turn can be used to significantly speed up the process of XML document validation against sets of XML keys.

XML Key Reasoning for Document Validation

Cover Sets for XML Keys

Fact

Σ is non-redundant if there is no key $\psi \in \Sigma$ such that $\Sigma - \{\psi\} \models \psi$.

Algorithm 1: Non-redundant Cover for XML keys

Input : Finite set Σ of XML keys
Output: A non-redundant cover for Σ

1 $\Theta = \Sigma$;
2 **foreach** *key* $\psi \in \Sigma$ **do**
3 **if** $\Theta - \{\psi\} \models \psi$ **then**
4 $\Theta = \Theta - \{\psi\}$;
5 **return** Θ;
6

This set can be computed in $\mathcal{O}(|\Sigma| \times (max\{|\psi| : \psi \in \Sigma\})^2)$ time.

Experimental Results

Performance Analysis

- We analyze:
  - (i) The scalability of the implication problem.
  - (ii) The viability of computing non-redundant cover sets to speed up the validation of XML documents against XML keys.
- For (i) we generated large sets of XML keys in the following two systematic ways:
  1. using a manually defined set of 5 to 10 XML keys as seeds, we computed new implied keys by successively applying the inference rules from the axiomatization presented by Hartmann & Link (2009).
  2. we defined some non-implied XML keys, adding witness edges keeping q_φ not reachable from q'_φ.
- For (ii) we generated sets of keys following the same strategy.

Experimental Results – Datasets (1/2)

Table: XML Documents.

Doc ID	Document	No. of Elements	No. of Attributes	Size	Max. Depth	Average Depth
Doc1	321gone.xml	311	0	23 KB	5	3.76527
Doc2	yahoo.xml	342	0	24 KB	5	3.76608
Doc3	dblp.xml	29,494	3,247	1.6 MB	6	2.90228
Doc4	nasa.xml	476,646	56,317	23 MB	8	5.58314
Doc5	SigmodRecord.xml	11,526	3,737	476 KB	6	5.14107
Doc6	mondial-3.0.xml	22,423	47,423	1 MB	5	3.59274

Experimental Results – Datasets (2/2)

SigmodRecord (seed keys)

$(\varepsilon, (issue, \{volume.S, number.S\}))$

$(\varepsilon, (\_^*.issue, \{volume.S, number.S\}))$

$(issue, (articles, \{article.title.S\}))$ → $(Q, (Q', \{P.P_1, \ldots, P.P_k\}))$,

$(issue.articles, (article, \{title.S\}))$ → $(Q.Q', (P, \{P_1, \ldots, P_k\}))$

$(issue, (articles.article, \{initPage.S, endPage.S\}))$ $(Q, (Q'.P, \{P_1, \ldots, P_k\}))$

$(\varepsilon, (issue.articles.article.authors.author, \{position\}))$ $(issue, (articles.article, \{title.S\}))$

Interaction rule

Sigmod.xml

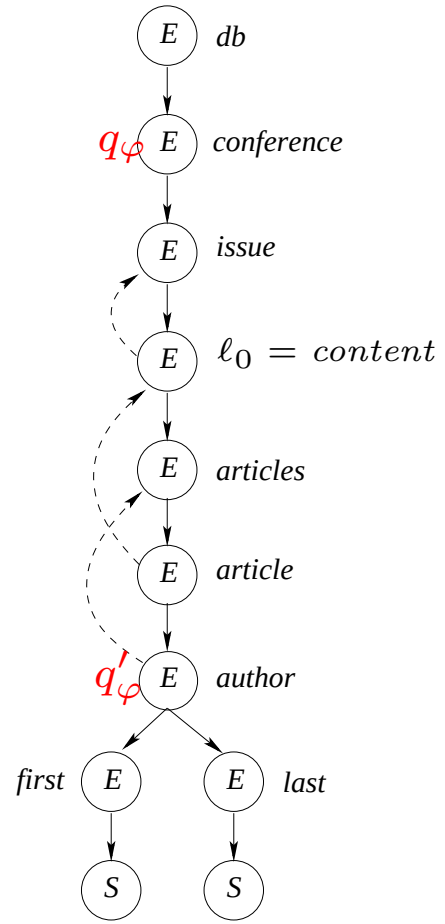
```
<SigmodRecord>
   <issue>
      <volume>13</volume>
      <number>2</number>
      <articles>
         <article>
            <title>Deadlock Detection is Cheap.</title>
            <initPage>19</initPage>
            <endPage>34</endPage>
            <authors>
               <author position="00">Rakesh Agrawal</author>
               <author position="01">Michael J. Carey</author>
               <author position="02">David J. DeWitt</author>
            </authors>
         </article> ...
      </articles>
   </issue> ...
</SigmodRecord>
```

Experimental Results

Strategies to generate implied and non-implied XML keys

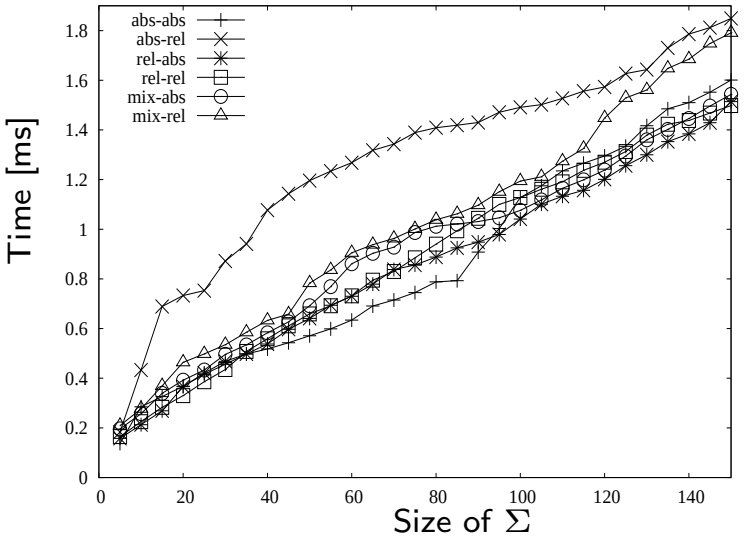
- If we apply the interaction rule to
  - $(issue, (articles, \{article.title.S\}))$
  - $(issue.articles, (article, \{title.S\}))$
- We derived the implied key
  - $(issue, (articles.article, \{title.S\}))$
- We applied the interaction, context-target, subnodes, context-path containment, target-path containment, subnodes-epsilon and prefix-epsilon rules whenever possible.

Non-implied keys

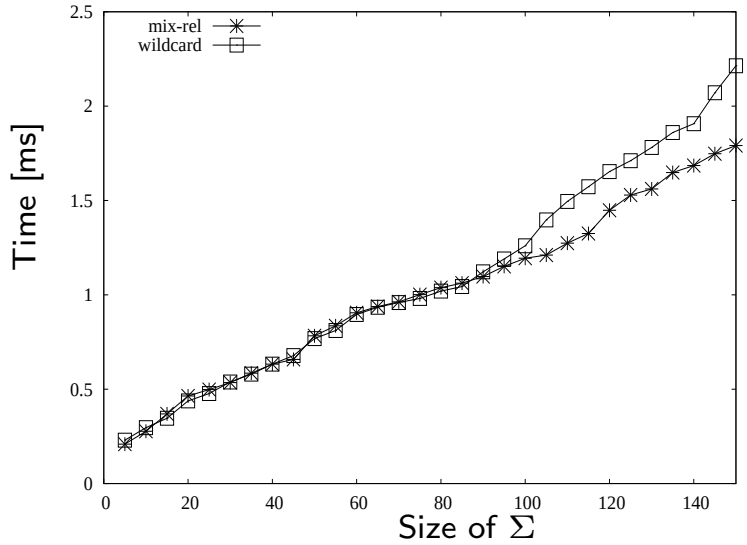


Results for Implication Problem

Deciding implication of XML keys



(e) XML key implication, all cases.



(f) Effect of wildcards presence.

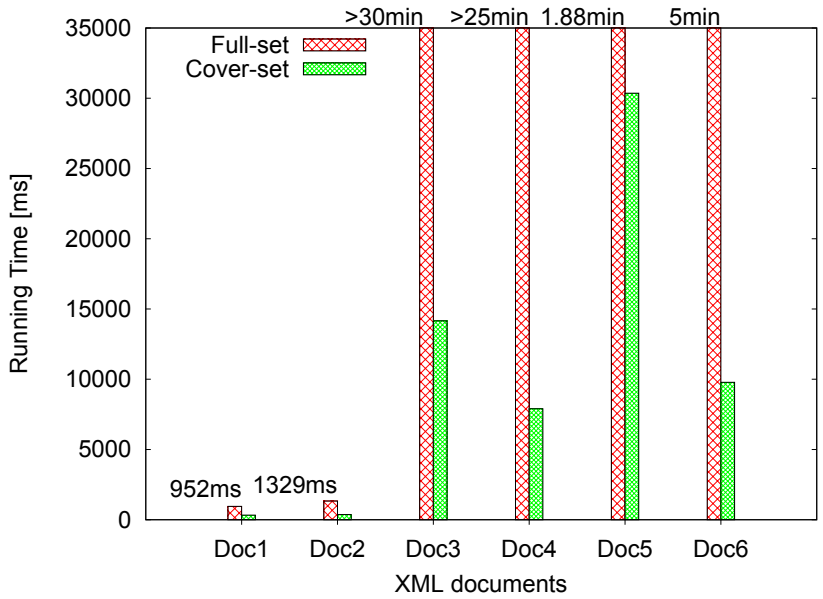
Figure: Performance of the Algorithm for the Implication of XML Keys ($\Sigma \models \varphi$).

Results for Validation Problem

Document Validation (1/2)

XML doc.	Key Set	Time[ms]
321gone & yahoo (Doc1 & 2)	Processed Keys: 23 Discarded keys: 15 Cover set: 8 keys	3.458
DBLP (Doc3)	Processed Keys: 36 Discarded keys: 24 Cover set: 12 keys	12.757
nasa (Doc4)	Processed Keys: 35 Discarded keys: 28 Cover set size: 7 keys	9.23
Sigmod Record (Doc5)	Processed Keys: 24 Discarded Keys: 19 Cover set: 5 keys	5.294
mondial (Doc6)	Processed Keys: 26 Discarded Keys: 16 Cover set: 10	4.342

(a) Non-redundant Cover Sets.



(b) Validation Against Cover Sets.

Figure: Non-redundant Cover Sets of XML keys and Validation of XML Documents

Results for Validation Problem

Document Validation (2/2)

- There are some very expensive XPath queries.
- **SigmodRecord**
  - $(issue, (articles.article, \{title.S\}))$ - there are **1503 nodes** in the XPath query "//issue/articles/article".
  - $(\varepsilon, (issue.articles.article.authors.author, \{S\}))$ - there are **3737 nodes** in the XPath query "//issue/articles/article/authors/author".

Main conclusion

This work was motivated by two objectives:

1. To demonstrate that there are expressive classes of XML keys that can be reasoned about efficiently.
2. To show that our observations on the problem of deciding implication has immediate consequences for other perennial task in XML database management.

[1.1] We studied a fragment of XML keys with nonempty sets of simple key paths.

[1.1.1] Presenting an efficient implementation for the implication problem.

[1.1.2] Showing through experiments that the proposed algorithm runs fast in practice and scales well.

[2.1] We studied the problem of validating an XML document against a set of XML keys.

[2.2] Presenting an optimization method for this validation via the computation of non-redundant cover sets of XML keys.

[2.3] Our experiments show that enormous time savings can be achieved in practice.

Questions?



THANKS!

Emir Muñoz Jiménez– emir@emunoz.org

Axiomatization

- Let be PL from the grammar $Q \to \ell \mid \varepsilon \mid Q.Q \mid \_^*$
- where $\ell \in \mathcal{L}$ is a label, ε is the empty word, "." is the concat operator, and "$\_^*$" is the wildcard.
- For nodes v and v' of an XML tree T, the *value intersection* of $v[\![Q]\!]$ and $v'[\![Q]\!]$ is given by $v[\![Q]\!] \cap_v v'[\![Q]\!] = \{(w, w') \mid w \in v[\![Q]\!], w' \in v'[\![Q]\!], w =_v w'\}$
- We define *semantic closure* by: $\Sigma^* = \{\varphi \in \mathcal{C} \mid \Sigma \models \varphi\}$
- and also *syntactic closure* by: $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$
- A set of rules $\mathfrak{R}$ is *sound (complete)* if $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^* (\Sigma^* \subseteq \Sigma^+_{\mathfrak{R}})$
- A sound and complete set of rules is called *axiomatization*

Deciding XML Key Implication

Mini-trees and Witness Graphs (2/2)

Theorem (Hartman & Link, 2009)

Let $\Sigma \cup \{\varphi\}$ be a finite set of keys in the class $\mathcal{K}$. We have $\Sigma \models \varphi$ if and only if q_φ is reachable from q'_φ in $G_{\Sigma,\varphi}$.

Algorithm 2: XML key implication in $\mathcal{K}$

Input : finite set of XML keys $\Sigma \cup \{\varphi\}$ in $\mathcal{K}$
Output: yes, if $\Sigma \models \varphi$; no, otherwise

1 Construct $G_{\Sigma,\varphi}$ for Σ and φ;
2 **if** q_φ is reachable from q'_φ in G **then return** yes; **else return** no; **end if**