Valid Query Answers for XML

Slawomir Staworko[1] Jan Chomicki[2]

[1] University at Buffalo and INRIA Futurs

[2] University at Buffalo and Warsaw University

June 25-29, 2007

Invalid XML documents

Querying Invalid XML

- Integration of XML documents
- Slight differences between schemas (e.g. different cardinality constraints)
- Legacy XML databases
- Database updates

Invalid XML documents

Querying Invalid XML

- Integration of XML documents
- Slight differences between schemas (e.g. different cardinality constraints)
- Legacy XML databases
- Database updates

Document Type Definition (D_0)

proj → (name, emp, proj*, emp*)
emp → (name, salary)
name → #PCDATA
salary → #PCDATA

Query: get salaries of all employees that are not managers

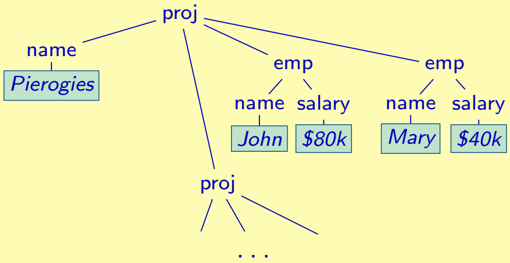
//proj/name/emp/following_sibling::emp/salary

Invalid XML documents

Querying Invalid XML

- Integration of XML documents
- Slight differences between schemas (e.g. different cardinality constraints)
- Legacy XML databases
- Database updates

Document with errors



Document Type Definition (D_0)

proj → (name, emp, proj*, emp*)
emp → (name, salary)
name → #PCDATA
salary → #PCDATA

Query: get salaries of all employees that are not managers

//proj/name/emp/following_sibling::emp/salary

Invalid XML documents

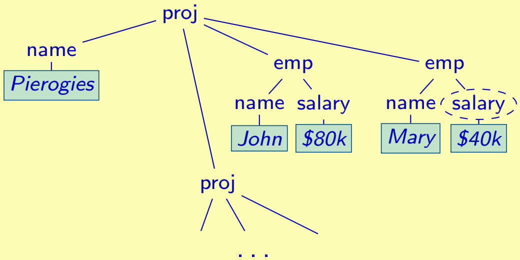
Querying Invalid XML

- Integration of XML documents
- Slight differences between schemas (e.g. different cardinality constraints)
- Legacy XML databases
- Database updates

Document Type Definition (D_0)

proj → (name, emp, proj*, emp*)
emp → (name, salary)
name → #PCDATA
salary → #PCDATA

Document with errors



Query: get salaries of all employees that are not managers

//proj/name/emp/following_sibling::emp/salary

Invalid XML documents

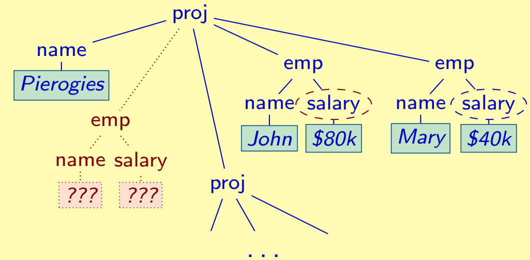
Querying Invalid XML

- Integration of XML documents
- Slight differences between schemas (e.g. different cardinality constraints)
- Legacy XML databases
- Database updates

Document Type Definition (D_0)

proj $\longrightarrow$ (name, emp, proj*, emp*)
emp $\longrightarrow$ (name, salary)
name $\longrightarrow$ #PCDATA
salary $\longrightarrow$ #PCDATA

Document with errors



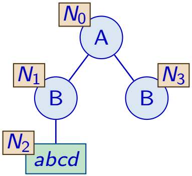
Query: get salaries of all employees that are not managers

//proj/name/emp/following_sibling::emp/salary

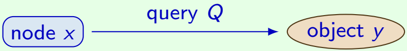
Core XPath

Core XPath Queries

- text values but no attributes
- all standard axes
- subexpressions, value tests, and joins:
  `//*[A/B]`, `//*[B/text()=C/text()]`
- negation and disjunction:
  `//*[not A/B]`, `//A | //B`
- no functions
  `count(/A/*)`



Tree Reachability Fact: (x, Q, y)



node x query Q object y

Basic facts use only /* and `following_sibling::`

$$(N_0,/*,N_1),\ (N_1,/*,N_2),\ldots$$

Other facts are inferred with implications:
$(X, Q/P, Y) \Leftarrow (X, Q, Z) \wedge (Z, P, Y)$.

$$(N_0,/*/*,N_2).$$

Query Answers

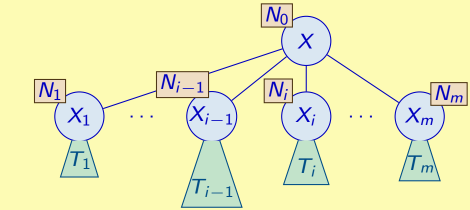
- given query Q and document T with the root node r
- find all tree facts that hold in T
- x in an answer to Q in T iff the tree fact (r, Q, x) holds in T

Query Evaluation for Positive Core XPath (no negation)

Bottom-up approach

- computing tree facts for query Q
- tree facts for $T_1, \ldots, T_m$ computed before
- including inferred facts (involving subqueries of Q)

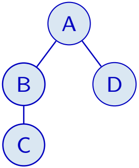
Tree



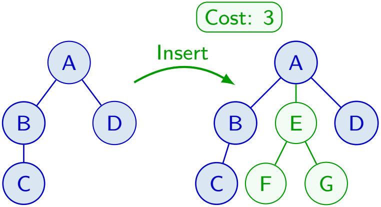
Algorithm

```
I  start with $\varnothing$
II for subtree $T_i$ ($i = 1, \ldots, m$)
     1 add all facts of the subtree (obtained by recursion)
     2 add ($N_0$,/*,$N_i$)
     3 if $i > 1$ add ($N_{i-1}$,following_sibling::*,$N_i$)
```

Editing operations



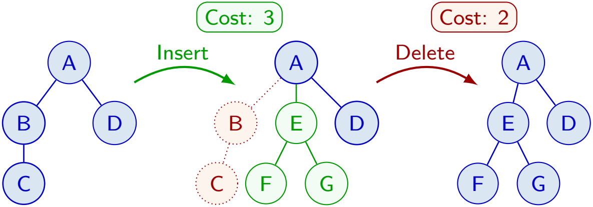
Editing operations



Editing operations

- Inserting a subtree

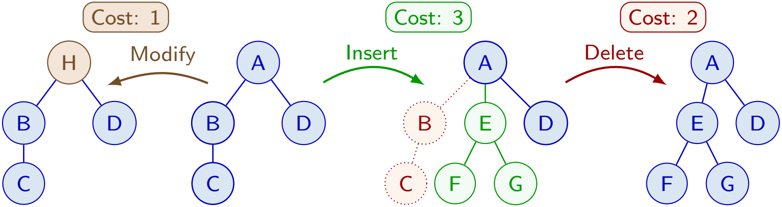
Editing operations



Editing operations

- Inserting a subtree
- Deleting a subtree

Editing operations



Editing operations

- Inserting a subtree
- Deleting a subtree
- Modifying a node's label

Edit Distance and Repairs [3]

Distance between documents

$dist(T, S)$ is the minimal cost of transforming T into S

Distance to a DTD

$dist(T, D)$ is the minimal cost of repairing T w.r.t D i.e.,

$$\min\{dist(T, S) | S \text{ valid w.r.t } D\}$$

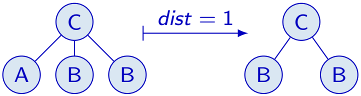
DTD

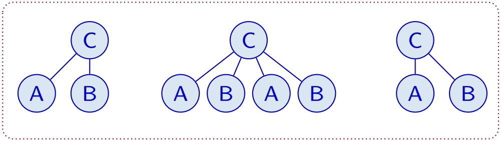
C → (A,B)*
A → EMPTY
B → EMPTY

Repair

T' is a repair of T w.r.t D iff

$$dist(T', T) = dist(T, D)$$





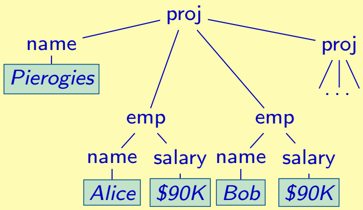
There can be an exponential number of repairs

Valid Query Answers

Valid Query Answers

x is a valid answer to query Q in T w.r.t. D iff
x is an answer to Q in every repair of T w.r.t. D.

XML Document



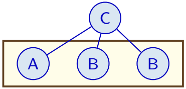
Document Type Definition (D_0)

$$
\begin{array}{lll}
\text{proj} & \longrightarrow & (\text{name, emp, proj*, emp*}) \\
\text{emp} & \longrightarrow & (\text{name, salary}) \\
\text{name} & \longrightarrow & \#\text{PCDATA} \\
\text{salary} & \longrightarrow & \#\text{PCDATA}
\end{array}
$$

Queries and Valid Answers

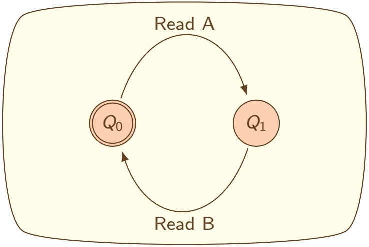
`//proj[emp[1]/salary='$90K']/name/text()`	$\longrightarrow \{Pierogies\}$
`//proj[name='Pierogies']/emp[1]/salary/text()`	$\longrightarrow \{\$90K\}$
`//proj[name='Pierogies']/emp[1]/name/text()`	$\longrightarrow \emptyset$
`//proj[name='Pierogies']/emp[1]/salary`	$\longrightarrow \emptyset$

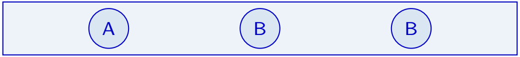
Trace graph



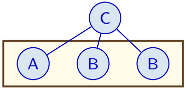
DTD

C → (A,B)*
A → EMPTY
B → EMPTY



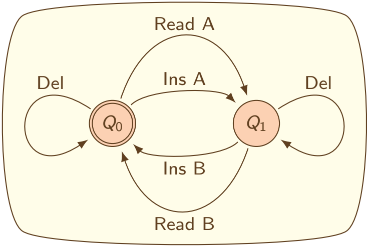


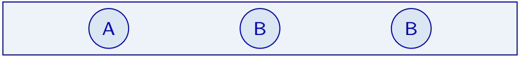
Trace graph



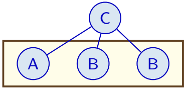
DTD

C → (A,B)*
A → EMPTY
B → EMPTY



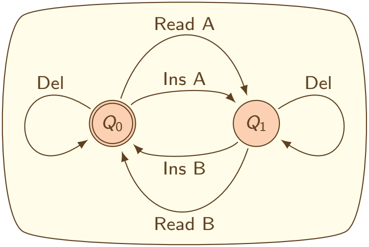


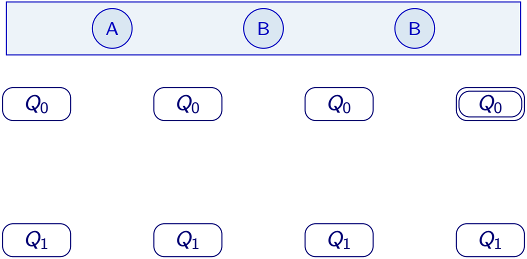
Trace graph



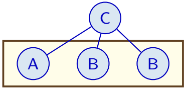
DTD

C → (A,B)*
A → EMPTY
B → EMPTY



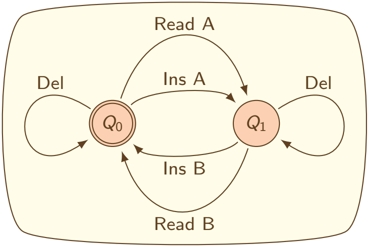


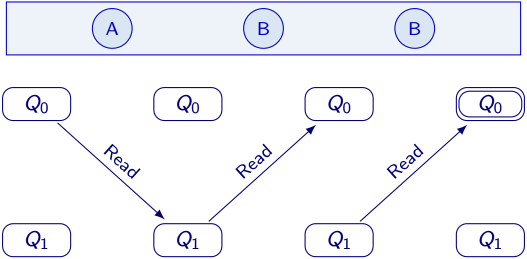
Trace graph



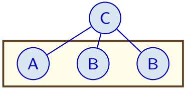
DTD

C → (A,B)*
A → EMPTY
B → EMPTY



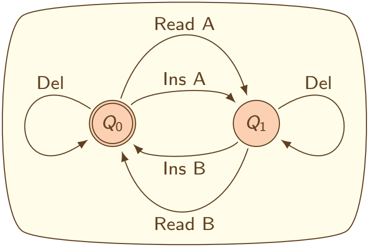


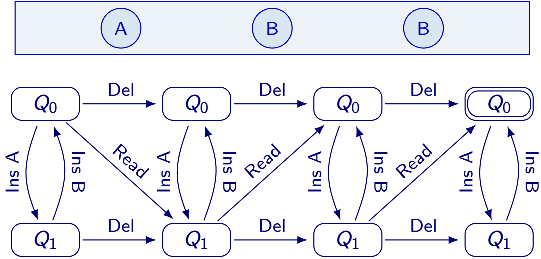
Trace graph



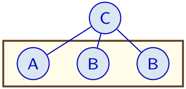
DTD

C → (A,B)*
A → EMPTY
B → EMPTY



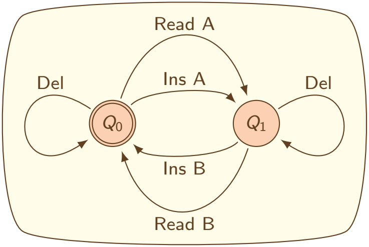


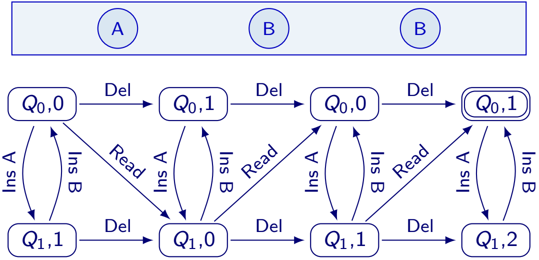
Trace graph



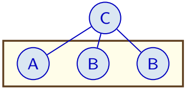
DTD

C → (A,B)*
A → EMPTY
B → EMPTY



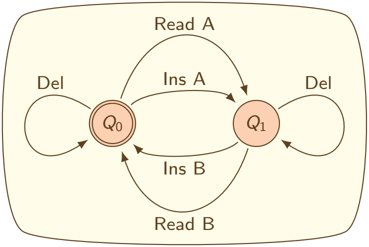


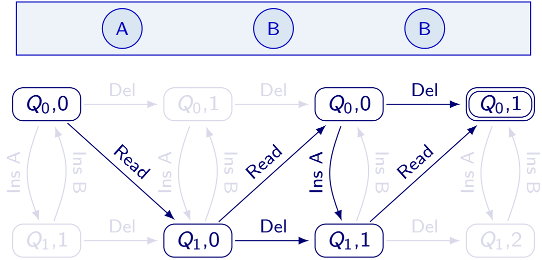
Trace graph



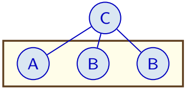
DTD

C → (A,B)*
A → EMPTY
B → EMPTY



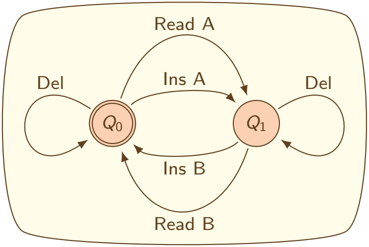


Trace graph

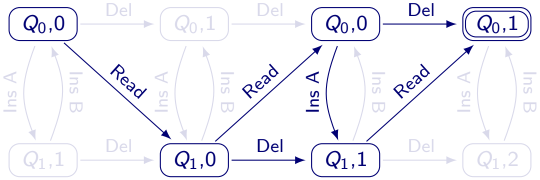


DTD

C → (A,B)*
A → EMPTY
B → EMPTY



Compact representation of all repairs



Repairing Paths:

- (Read, Read, Del)
- (Read, Read, Ins A, Read)
- (Read, Del, Read)

Computing Valid Query Answers

Certain Tree Facts

Tree facts present in every repair of a given tree

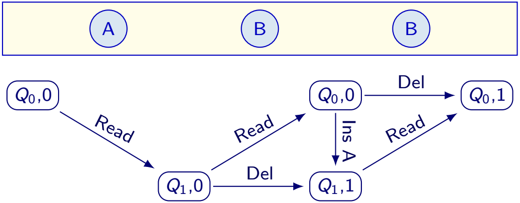
Bottom-up approach

Precomputed values:

- certain facts for all children
- certain facts common for every minimal tree satisfying DTD

Obtain certain facts by

Intersection of the sets of all repairing paths



For every repairing path, construct:

set of facts collected "so far":

- start with ∅
- **Read** adds certain facts of the corresponding child
- **Del** adds nothing
- **Ins A** adds certain facts common for minimal trees labeled with A

Eager intersection

Problem

Possibly an exponential number of paths

Solution: Eager Intersection

Intersect all sets of certain facts for paths sharing the same last operation adding facts (Read/Ins).

Computing VQA

Queries	Combined-complexity	Data-complexity
Descending axes	PTIME	PTIME
+ Ascending axes	co-NP-complete	???
+ Sliding axes	co-NP-complete	???
+ Negation/Disjunction	co-NP-complete	???
+ Joins	co-NP-complete	co-NP-complete

Experimental Results: Edit Distance Computation

Compared algorithms

PARSE	base line
VALIDATE	regular automata
DIST	distance computation

Data generation

1. random valid document
2. removing and adding random nodes
3. invalidity ratio

$$dist(T, D)/|T| \simeq 0.1\%$$

4. small height (8-10)

Experimental Results: Edit Distance Computation

Compared algorithms

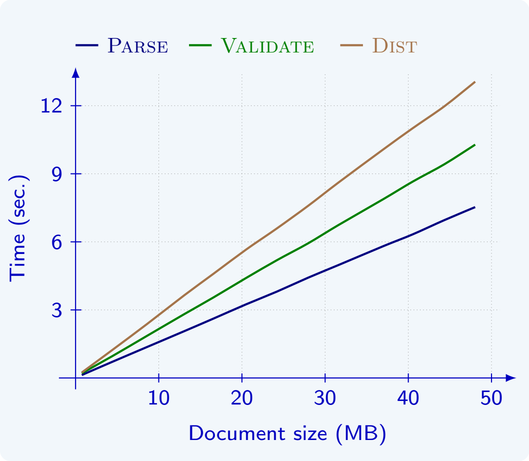
PARSE	base line
VALIDATE	regular automata
DIST	distance computation

Data generation

1. random valid document
2. removing and adding random nodes
3. invalidity ratio

   $$dist(T, D)/|T| \simeq 0.1\%$$

4. small height (8-10)



Experimental Results: Edit Distance Computation

Compared algorithms

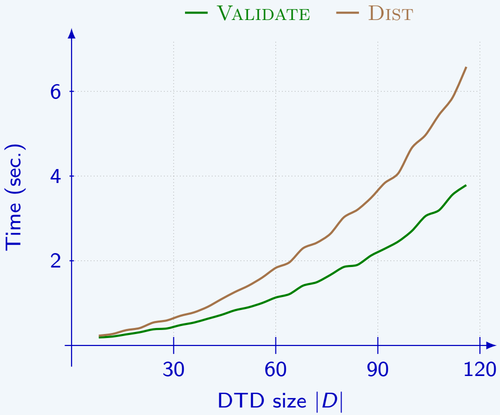
PARSE	base line
VALIDATE	regular automata
DIST	distance computation

Data generation

1. random valid document
2. removing and adding random nodes
3. invalidity ratio

$$dist(T, D)/|T| \simeq 0.1\%$$

4. small height (8-10)



Experimental Results: Valid Query Answer Computation

Compared algorithms

QA	base line
VQA	eager intersection

Data generation

- invalidity ratio

$$dist(T, D)/|T| \simeq 0.1\%$$

- small height (8-10)

Implemented Queries

- `//`, `/`, `following_sibling::`
- `name()=A,text()='`*str*`'`
- QA works in linear time

Experimental Results: Valid Query Answer Computation

Compared algorithms

QA	base line
VQA	eager intersection

Data generation

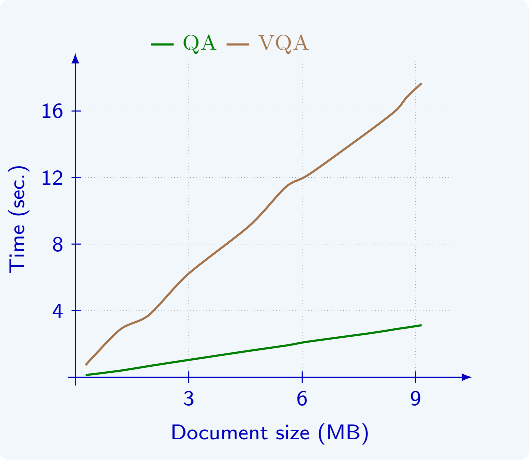
- invalidity ratio

$$dist(T, D)/|T| \simeq 0.1\%$$

- small height (8-10)

Implemented Queries

- `//, /, following_sibling::`
- `name()=A,text()=`'*str*'
- QA works in linear time



Experimental Results: Valid Query Answer Computation

Compared algorithms

QA	base line
VQA	eager intersection

Data generation

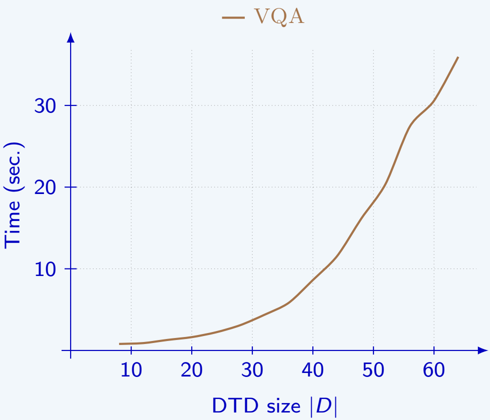
- invalidity ratio

$$dist(T, D)/|T| \simeq 0.1\%$$

- small height (8-10)

Implemented Queries

- `//, /, following_sibling::`
- `name()=A,text()='`*str*`'`
- QA works in linear time



Conclusions and Future Work

Conclusions

- Framework for querying documents with validity violations of local nature (missing or superfluous nodes)
- Efficient algorithm for computing valid answers to a class of XPath queries

Future Work

- Valid answers by query rewriting
- In-depth analysis of data complexity
- Other tree operations: inner node deletion/insertion, node move, . . . [1]
- Semantic inconsistencies: keys, functional dependencies,. . . [2]

Acknowledgments

Marcelo Arenas

Leo Bertossi

Wenfei Fan

Jerzy Marcinkowski

Slawomir Staworko

Jef Wijsen

P. Bille.
A Survey on Tree Edit Distance and Related Problems.
Theoretical Computer Science, 337(1-3):217–239, 2003.

S. Flesca, F. Furfaro, S. Greco, and E. Zumpano.
Querying and Repairing Inconsistent XML Data.
In *Web Information Systems Engineering*, pages 175–188. Springer, LNCS 3806, 2005.

S. Staworko and J. Chomicki.
Validity-Sensitive Querying of XML Databases.
In *EDBT Workshops (dataX)*, pages 164–177. Springer, LNCS 4254, 2006.