Pattern Based XML Queries

Amélie Gheerbrant

LFCS
University of Edinburgh

07/12/2012 - BD - OAK

XML is the current standard for exchanging data on the web

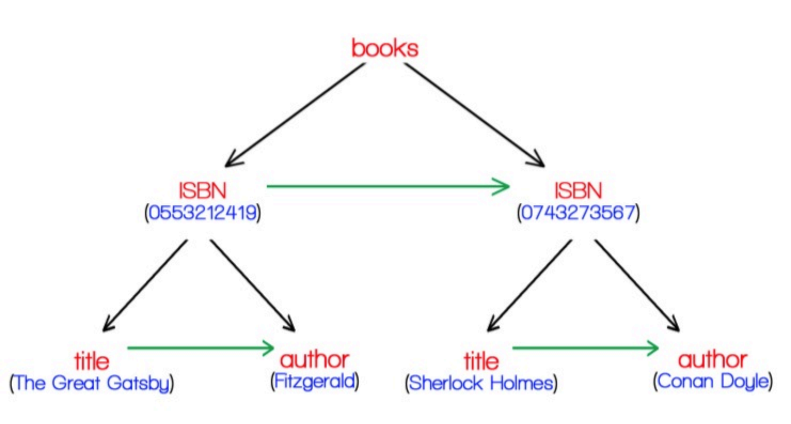
```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels</title>
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
</Books>
```

More flexible than old relational tables:

ISBN	Title	Author
0553212419	Sherlock Holmes: Complete Novels	Sir Arthur Conan Doyle
0743273567	The Great Gatsby	F. Scott Fitzgerald

Data trees

Abstractions of XML document



In black:	In green:	In red:	In blue:
child axes	sibling axes	node labels	data values

Database Query Languages

Relational Query Languages

- First-order logic (FO), the basis of *SQL*

$$\exists y\ Book(x, y, Sir\ Arthur\ Conan\ Doyle) \vee \exists y\ Book(x, Ulysses, y)$$

ISBN	Title	Author
05532	Sherlock Holmes: Complete Novels	Sir Arthur Conan Doyle
07432	The Great Gatsby	F. Scott Fitzgerald

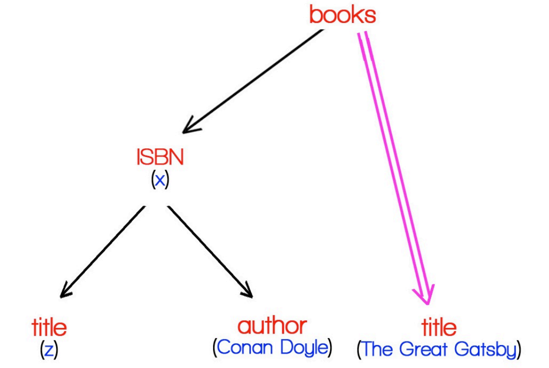
- On database below it returns 05532.
- Same query in relational algebra:

$$\pi_{ISBN}(\sigma_{author='' ConanDoyle''}(Book)) \cup \pi_{ISBN}(\sigma_{title='' Ulysses''}(Book))$$

XML Query Languages built out of tree patterns

- Tree patterns are basic objects for XML data exchange.
- Building blocks for XML analogue of FO.

Tree pattern: an example



In this example:

- Variables x,z stand for unknown attributes.
- The pink arrow is a descendant axis replacing a child path.
- Sibling axes are not specified.

Pattern based XML languages

XML query languages

- $\mathrm{CQ}_{\mathsf{XML}}$: closure of tree patterns under $\exists, \wedge$

  $\Rightarrow$ XML analogue of the

  $\sigma, \pi, \bowtie$-fragment of relational algebra

- $\mathrm{UCQ}_{\mathsf{XML}}$: closure of $\mathrm{CQ}_{\mathsf{XML}}$ under $\vee$

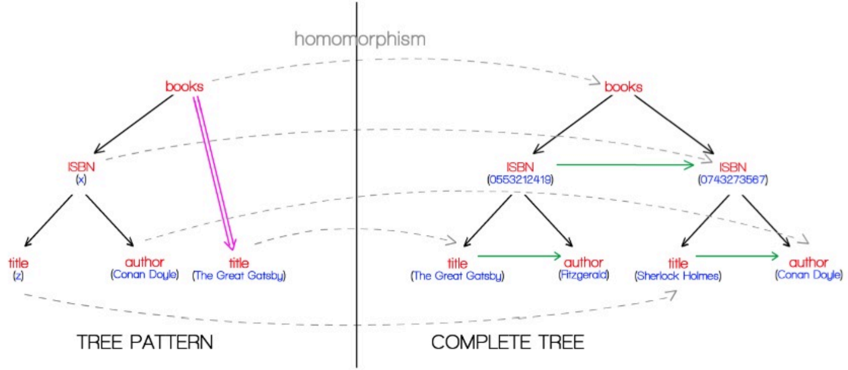
  $\Rightarrow$ XML analogue of the

  $\sigma, \pi, \bowtie, \cup$-fragment of relational algebra

- $\mathrm{FO}_{\mathsf{XML}}$

  $\Rightarrow$ XML analogue of the relational algebra

Semantics of tree patterns via homomorphism



XML homomorphisms

They are required to preserve both node and data structure.

Main uses of XML tree patterns

Provide:

an abstraction for XML with incomplete data.

Provide:

building blocks for XML query languages and XML schema mappings.

Computational Problems

Query Evaluation

$\Rightarrow$ evaluating queries over incomplete XML data

(C.f. Imielinski-Lipski 1984; Abiteboul-Kanellakis-Grahne 1991.)

Query Optimization

How to rewrite queries to improve their efficiency?

$\Rightarrow$ the basic problem is query containment.

The Certain Answers Problem

Incomplete vs complete databases

R:

ISBN	Title	Author
04592	x_1	Sophocles
x_2	x_1	Anouilh

R':

ISBN	Title	Author
04592	Antigone	Sophocles
07432	Antigone	Anouilh
08945	Oedipus	Sophocles

Incomplete databases contain null values $x_1, x_2, \ldots$.

Semantics

$R' \in [\![R]\!] \Leftrightarrow$ there exists a homomorphism $h : R \to R'$

Certain Answers: the intuition

- Collect tuples satisfying a query in every $R' \in [\![R]\!]$.

$\Rightarrow$ 04592 is the only "certain" ISBN of a book written by Sophocles.

Certain Answers

Problem

What does it mean to evaluate a query on an incomplete database?

Consensus: restrict to certain answers

$$certain(Q, D) \;=\; \bigcap\{Q(D') \mid D' \in [\![D]\!]\}$$

An answer is certain if it does not depend on the interpretation of null values.

Complexity of Computing Certain Answers

Problem: Combined Complexity of a query language L

Input:	a database D
	a query $Q \in L$ and a tuple $\bar{a}$ of the same arity
Question:	Does $\bar{a} \in certain(Q, D)$?

Data Complexity of a query language L

The query Q is fixed (not part of the input).

In practise

Databases are large and queries are small.

A well-behaved class: unions of conjunctive queries

Unions of conjunctive queries UCQ

Existential positive FO formulas (built from $\exists, \wedge, \vee$)

- $\Rightarrow$ Example: $\exists x R(x, a) \vee \exists x R(x, x)$
- $\Rightarrow$ Equivalent to $\sigma, \pi, \bowtie, \cup$-fragment of relational algebra.

Data complexity

Certain Answers for UCQ is in PTIME in $|D|$

$\Rightarrow$ Via naïve evaluation:

1. evaluate the formula treating nulls as constants, *as if* the database was complete,
2. throw away tuples with nulls.

Combined complexity

Certain Answers for UCQ is in NP in $|D|$ and $|Q|$.
(exact same as query evaluation)

- $\Rightarrow$ Homomorphism-based techniques (more on that later).

Beyond UCQ

FO queries beyond UCQ

- If $\neq$ is allowed, the problem is coNP-hard,
- For unrestricted FO queries the problem is undecidable.

(Abiteboul-Kanellakis-Grahne 1991)

Another way of adding negation: BCCQ

Closure of CQ's under $\cup$ and $\neg$:

very restricted form of negation

Example:

"If there is a book titled Antigone with ISBN 04592, then there is a book titled Antigone with author Sophocles."

$\Rightarrow$ complexity? was unknown even in the relational case!

ICDT'12 (G.-Libkin-Tan)

Boolean Combinations of Conjunctive Queries

- Computing Certain Answers for BCCQ is in PTIME for:
  - incomplete relational databases
  - "rigid" incomplete XML trees

How we discovered this

- We started by looking at XML.
- We then discovered a gap in the relational theory.

Back to the PTIME proof for UCQ's

Naïve evaluation and counter-models

- What does $certain(Q, D) = false$ mean for $Q \in UCQ$?
  $\Rightarrow$ D can be extended to a counter-model of Q.
- Example: $Q = \exists x\ R(x, x)$

  R:

a	*x*

  can be extended to R:

a	*b*

  $\not\models Q$

  $\Rightarrow$ Here $certain(Q, D) = false$.

Do the same for BCCQ?

- Rossman LICS'05 implies that naïve evaluation will not work... (Libkin PODS'11)
- But we can adapt the idea.
- Idea of the proof:

  search for a counter model in PTIME.

The BCCQ algorithm: the basic reduction

The basic case

Boolean query $Q = q \rightarrow q'$, where q is a CQ, and q' is a UCQ.

$Q = q \rightarrow q'$ only has one falsifying valuation

q	q'	$q \rightarrow q'$
false	false	true
true	true	true
true	false	false
false	true	true

A simple example

ISBN	title	author
x_1	Antigone	x_2
04592	x_3	Sophocles

- $q =$ There is a book titled Antigone with ISBN 04592.
- $q' =$ There is a book titled Antigone with author Sophocles.
- $Q =$ Does q implies q'?

The OWA algorithm: the basic reduction

1. Convert q into its tableau and glue it into D:

ISBN	title	author
x_1	Antigone	x_2
04592	x_3	Sophocles
04592	Antigone	x_4

2. Evaluate q' naïvely on the new database.
3. Verdict: it's not true, there is no book titled Antigone with author Sophocles.
4. The counter model search was successful.
5. Hence $certain(D, Q) = false$.

Generalization to arbitrary BCCQ's

Generalization to arbitrary $Q \in \mathrm{BCCQ}$

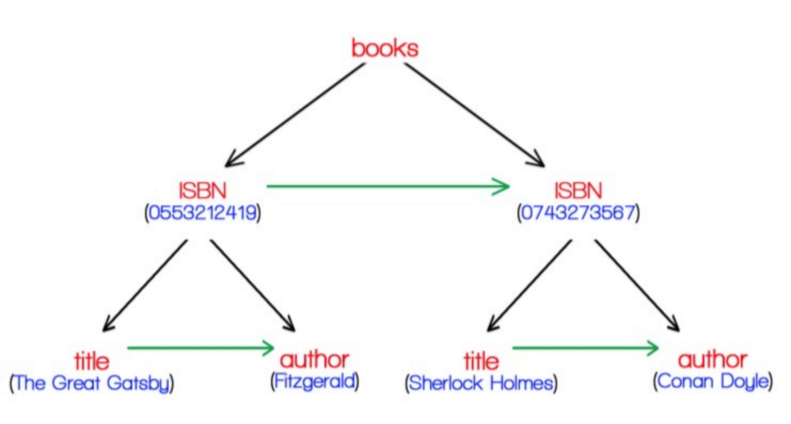
1. list every falsifying valuation of Q,
2. apply the previous procedure for each one of them.

Now let us switch to XML

- We will see that the new tractability results extend to XML.
- But only for incomplete “rigid” trees.

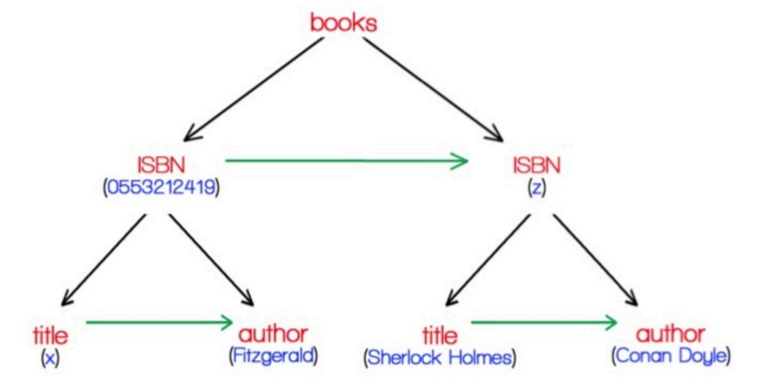
Features of Incompleteness in XML

A complete XML document



Features of Incompleteness in XML

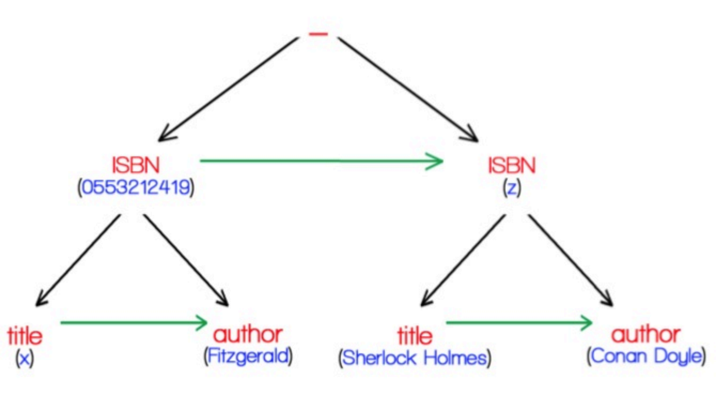
Missing information: ATTRIBUTE VALUES



Variables $x, y, z, \ldots$ replace unknown attribute names.

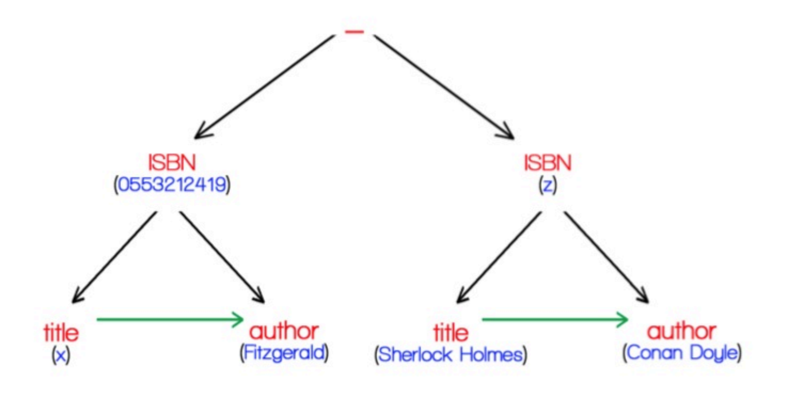
Features of Incompleteness in XML

Missing information: NODE LABELS



Special “wildcard” label _ placeholder for unknown labels.

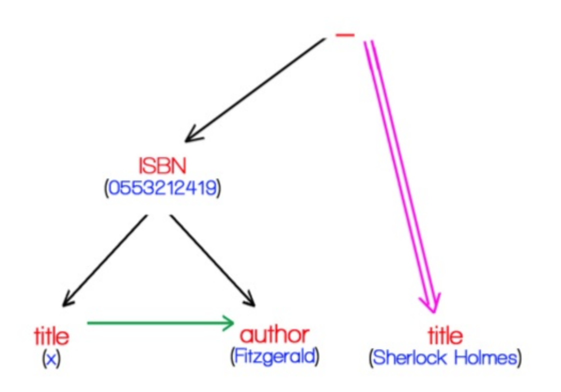
Missing information: STRUCTURAL INFORMATION



Lost information about sibling ordering.

Features of Incompleteness in XML

Missing information: STRUCTURAL INFORMATION



Lost information about child ordering:
a descendant axis can replace a child path.

A special Class

The general case

Whenever structural incompleteness is allowed, Certain Answers very quickly becomes coNP-hard, even for UCQ's.

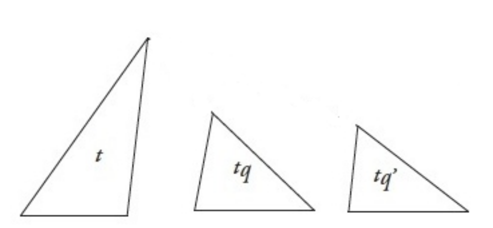
Rigid Incomplete Trees

- no structural incompleteness.
  (Incompleteness only at the level of attributes and labels.)
- Same good computational properties as naïve relational databases:
  $\Rightarrow$ UCQ's can be evaluated naïvely (in Ptime)
  $\Rightarrow$ BCCQ's can be evaluated in Ptime using a more refined algorithm

The XML algorithm for BCCQ's

First step

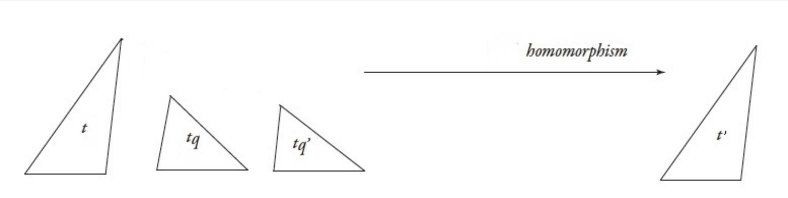
- Pick a falsifying valuation v.
- Convert the q_i's set to true into incomplete trees t_{q_i}'s.
- Group them into a forest F (union of trees) together with t:



The XML algorithm for BCCQ's

Second step

Generate a small number of new rigid trees t' s.t. there is a homomorphism $h : F \rightarrow t'$:



$\rightarrow$ if there is a counter-model, some t' will represent it.

Main Theorem: the number of new rigid trees that we need to construct is polynomial in $|t|$.
The intuition: they serve as "minimal" representatives for counter models.

From Certain Answers to Query Containment

Optimization problem

Rewrite queries so as to improve their efficiency,
i.e.,
find better, equivalent queries.
(such that $Q \subseteq Q'$ and $Q' \subseteq Q$)

Query Containment

Input: Queries Q, Q'
Question: is it true that for all D, $Q(D) \subseteq Q'(D)$?

Relational case

Testing containment is:

- Π_2^p-complete for BCCQ's,
- NP-complete for UCQ's (homomorphism-based techniques).

Homomorphism based techniques

An obvious case of containment $Q \subseteq Q'$

Q: Some author named *Joyce* wrote a book called *Ulysses*.
Q': Some author wrote a book called *Ulysses*.

But why is it so easy to see?

Algorithm:

1. Convert Q and Q' into "their tableaux" D_Q and $D_{Q'}$,

$D_{Q'}$:

ISBN	Title	Author
x_1	Ulysses	x_2

D_Q:

ISBN	Title	Author
x_3	Ulysses	Joyce

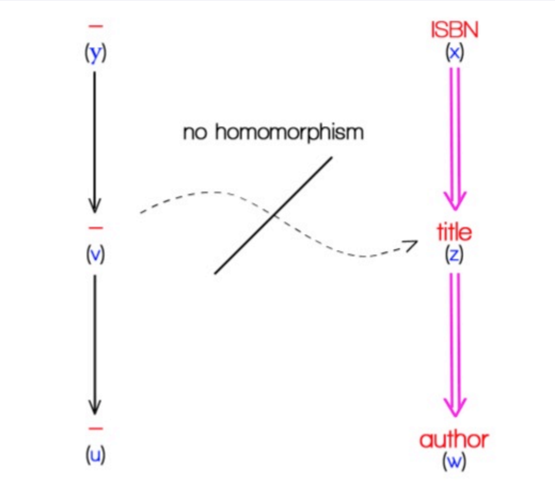
2. Guess a homomorphism $h : D_{Q'} \to D_Q$.
   - $h(x_1) = x_3$
   - $h(x_2) = \text{Joyce}$

$\Rightarrow$ works for Q, Q' conjunctive queries (built from $\exists, \wedge$)

$\Rightarrow$ extends to unions of conjunctive queries (built from $\exists, \wedge, \vee$)

Homomorphism based techniques for XML?

Even for very simple CQ's, the technique fails.



A Π_2^p-upper bound

Relational databases

- Chandra-Merlin 1977: containment reduces to query evaluation.
- Same for patterns, except that it reduces to finding certain answer.

Certain answers for BCCQ as a query containment problem

$$Q \subseteq Q'$$

iff

$$\mathit{certain}(Q \rightarrow Q', D) \text{ with } D \text{ empty}$$

ICDT'12 (G., Libkin, Tan)

Combined complexity of Certain Answers:

in Π_2^p for BCCQ.

$\Rightarrow$ containment also in Π_2^p

Classification

ICDT'13 (David, G., Libkin, Martens)
Containment for pattern based queries over data trees

We separated the cases in NP from the Π_2^p-hard cases.

Π_2^p-hard cases

Π_2^p-hard cases

- BCCQ even with only the child axis,
- CQ if all axes are available,
- UCQ almost always when wildcard is allowed.

NP cases

How do we get reasonable NP upper bounds?

The straightforward NP cases

Two models of XML trees with no structural incompleteness

- CQ's rigid (sibling ordering always specified)
- CQ's with child only (no sibling ordering)

  $\Rightarrow$ both in NP.

  It is enough to guess a homomorphism.
  (as in the relational case)

From CQ's to UCQ's

Asymmetry:

- UCQ's rigid: Π_2^p-hard,
- UCQ's with child only: still in NP.

The less straightforward NP cases

The standard example

CQ's with child and descendant: in NP.

Deciding $Q \subseteq Q'$: the trick (without wildcard)

- Construct the incomplete trees $t_{Q'}$ and t_Q corresponding to Q' and Q,
- Form $(t_Q)^*$ by adding to t_Q a (polynomially small in $|Q|$) set of descendant axis,
- Guess a homomorphism $h : t_{Q'} \rightarrow (t_Q)^*$.

From CQ's to UCQ's

The problem becomes immediately Π_2^p-hard for UCQ's.

The tough NP cases (work in progress)

Classes of queries for which the method works

Almost all CQ's, even with wildcard.

Similar idea as with the BCCQ XML algorithm

Construction of a (polynomially small in $|t|$) family of potential counter-models.

Deciding $Q \subseteq Q'$: the idea

Construct a (polynomially small in $|Q|$ and $|Q'|$) family of models of Q.

- Construct the incomplete tree $t_{Q'}$ corresponding to Q',
- Construct a family $\mathcal{F}$ of "minimal models" of Q,
- For each $t \in \mathcal{F}$, guess a homomorphism $h : t_{Q'} \to t$.

Future work

Semantics

Naïve evaluation behaves differently under different semantics.

Applications

Certain Anwers used in data exchange and data integration.

Constraints

How to deal with constraints? (complexity)

Optimization

Can we develop optimization techniques as a follow up to these theoretical results?