XML Data Exchange

Marcelo Arenas
P. Universidad Católica de Chile

Joint work with Leonid Libkin (U. of Toronto)
Data Exchange in Relational Databases

- Data exchange has been extensively studied in the relational world.
  - It has also been implemented: Clio.

- Relational data exchange settings:
  - Source and target schemas: Relational schemas.
  - Relationship between source and target schemas: Source-to-target dependencies.

- Semantics of data exchange has been precisely defined.
  - Algorithms for materializing target instances and for answering queries over the target have been developed.
Outline

• XML data exchange settings.
  - XML source-to-target dependencies.

• Query answering in XML data exchange.

• Final remarks.
Outline

- XML data exchange settings.
  - XML source-to-target dependencies.
- Query answering in XML data exchange.
- Final remarks.
XML Documents

```
<db>
    <book>
        @title="Algebra"
        <author>
            @name="Hungerford"
            @aff="U. Washington"
        </author>
    </book>
    <book>
        @title="Real Analysis"
        <author>
            @name="Royden"
            @aff="Stanford"
        </author>
    </book>
</db>
```
XML Documents

DTD:

\[
\begin{align*}
  db & \rightarrow \text{book}^+ \\
  \text{book} & \rightarrow \text{author}^+ \\
  \text{author} & \rightarrow \varepsilon
\end{align*}
\]
XML Documents

DTD:

\[
\begin{align*}
  db & \rightarrow \ \text{book}^+ \\
  \text{book} & \rightarrow \ \text{author}^+ \\
  \text{author} & \rightarrow \ \varepsilon \\
  \text{author} & \rightarrow \ \text{@name, @aff}
\end{align*}
\]
• Source and target schemas are given by DTDs.

• To specify the relationship between the source and the target schemas we use source-to-target dependencies.

To define these dependencies, we use tree patterns ...
Tree Patterns: Example
Tree Patterns: Example

```
book
  @title x
  author
    @name y

book
  @title “Algebra”
  author
    @name “Hungerford”
    @aff “U. Washington”

... db
```
Tree Patterns: Example

```
db
  └── book
      └── book
          ├── @title: “Real Analysis”
          └── author
              └── @aff: “Stanford”
        └── book
            └── @title: x
                └── author
                    └── @name: y
```
Tree Patterns: Example

Collect tuples \((x, y)\): (Algebra, Hungerford), (Real Analysis, Royden)
Tree Patterns

- Tree patterns: XPath-like language.
  - Example: $book(@title = x)[author(@name = y)]$

- Language also includes wildcard \_ (matching more than one symbol) and descendant operator //.
XML Source-to-target Dependencies

Source-to-target dependency (STD):

$$\psi_T(\bar{x}, \bar{z}) : \neg \varphi_S(\bar{x}, \bar{y}),$$

where $\varphi_S(\bar{x}, \bar{y})$ and $\psi_T(\bar{x}, \bar{z})$ are tree-pattern formulas over the source and target DTDs, resp.
XML Data Exchange Settings

Source  
\[ db \rightarrow {\text{book}}^+ \]  
DTD: \[ \begin{align*}  
{\text{book}} & \rightarrow {\text{author}}^+ \\
{\text{author}} & \rightarrow \varepsilon 
\end{align*} \]

Target  
\[ {\text{bib}} \rightarrow {\text{writer}}^+ \]  
DTD: \[ \begin{align*}  
{\text{writer}} & \rightarrow {\text{work}}^+ \\
{\text{work}} & \rightarrow \varepsilon 
\end{align*} \]

\[ \Sigma_{ST}: \begin{align*}  
&{\text{writer}} \\
&\quad \rightarrow {\text{name}} \\
&\quad \rightarrow {\text{title}} \\
&\quad \rightarrow {\text{year}} \\
&{\text{work}} \\
&\quad \rightarrow {\text{name}} \\
&\quad \rightarrow {\text{title}}, {\text{year}} \\
&{\text{book}} \\
&\quad \rightarrow {\text{title}} \\
&\quad \rightarrow {\text{author}} \\
&\quad \rightarrow {\text{name}} \\
\end{align*} \]
Example: Finding Solutions

Let $T$ be our original tree:

$\begin{tikzpicture}
\node (db) {db}
  child {node (book) {book}
    child {node {@title "Algebra"}
      child {node {@name "Hungerford"}
        child {node {@aff "U. Washington"}}}
    } child {node {author}
      child {node {@name "Royden"}
        child {node {@aff "Stanford"}}}
    }
  } child {node (book) {book}
    child {node {@title "Real Analysis"}
      child {node {@name "Royden"}
        child {node {@aff "Stanford"}}}
    }
  }
\end{tikzpicture}$
Example: Finding Solutions

A solution for $T$:

```
bib
  writer
    @name "Hungerford"
    work
      @title "Algebra"
      @year "1"
  writer
    @name "Royden"
    work
      @title "Real Analysis"
      @year "2"
```
Outline

- XML data exchange settings.
  - XML source-to-target dependencies.

- Query answering in XML data exchange.

- Final remarks.
Query Answering in XML Data Exchange

- Decision to make: What is our query language?

- We start by considering a query language that produces tuples of values.
Conjunctive Tree Queries

• Query language $CTQ//\text{ is defined by}$

$$Q := \varphi \mid Q \land Q \mid \exists x \ Q,$$

where $\varphi$ ranges over tree-pattern formulas.

• By disallowing descendant $//\text{ we obtain restriction } CTQ.$
Example: Conjunctive Tree Query

List all pairs of authors that have written articles with the same title.

\[ Q(x, y) := \exists z \left( \begin{array}{c}
\text{@name} \quad x \\
\text{work} \\
\text{@title} \\
\end{array} \quad \land \\
\begin{array}{c}
\text{@name} \quad y \\
\text{work} \\
\text{@title} \\
\end{array} \quad \right) \]
Certain Answers Semantics

• Given: A source tree $T$ and a conjunctive tree query $Q$ over the target.

• Answer to $Q$ should represent the answer to this query in the space of solutions for $T$.

• Certain answers semantics:

$$certain(Q, T) = \bigcap_{T' \text{ is a solution for } T} Q(T').$$
Computing Certain Answers

We study the following problem. Given data exchange setting \((D_S, D_T, \Sigma_{ST})\) and query \(Q\):

**PROBLEM:** \textsc{Certain-Answers}(Q).

**INPUT:** Tree \(T\) conforming to \(D_S\) and tuple \(\bar{a}\).

**QUESTION:** Is \(\bar{a} \in \text{certain}(Q, T)\)?
Computing Certain Answers: General Picture

**Theorem** For every XML data exchange setting and $CTQ//\text{-query } Q$, $\text{CERTAIN-ANSWERS}(Q)$ is in $\text{coNP}$.

Remark: In terms of the size of the document (data complexity).

**Theorem** There exist an XML data exchange setting and a $CTQ//\text{-query } Q$ such that $\text{CERTAIN-ANSWERS}(Q)$ is $\text{coNP-hard}$.

We want to find tractable cases ...
Computing Certain Answers: Finding Tractable Cases

• To find tractable cases, we have to concentrate on fully-specified STDs:

  We impose restrictions on tree patterns over target DTDs:
  - no descendant relation //; and
  - no wildcard _; and
  - all patterns start at the root.

  No restrictions imposed on tree patterns over source DTDs.

• Subsume non-relational data exchange handled by Clio.

From now on, all STDs are fully-specified.
Given a class $C$ of regular expressions and a class $Q$ of queries:

$C$ is tractable for $Q$ if for every data exchange setting in which target DTDs only use regular expressions from $C$ and every $Q$-query $Q$, $\text{CERTAIN-ANSWERS}(Q)$ is in $\text{PTIME}$.

$C$ is coNP-complete for $Q$ if there is a data exchange setting in which target DTDs only use regular expressions from $C$ and a $Q$-query $Q$ such that $\text{CERTAIN-ANSWERS}(Q)$ is coNP-complete.

Remark (Ladner): If $\text{PTIME} \neq \text{NP}$, there are problems in coNP which are neither tractable nor coNP-complete.
Our classification is based on classes of regular expressions used in target DTDs.

They must contain the simplest type of regular expressions: \((a + b + c)^*\)

Such classes will be called admissible.
Computing Certain Answers: Dichotomy

**Theorem**

1) Every admissible class \( C \) of regular expressions is either *tractable* or *coNP-complete* for \( CTQ// \).

2) For every tractable class: Given a source tree \( T \), one can compute in \( PTIME \) a solution \( T^* \) for \( T \) such that

\[
certain(Q, T) = remove_{null\_tuples}(Q(T^*)).
\]

3) It is decidable whether the regular expressions used in a target DTD belong to a tractable class.
Outline

• XML data exchange settings.
  - XML source-to-target dependencies.

• Query answering in XML data exchange.

• Final remarks.
Final Remarks

- Dichotomy also holds for unions of conjunctive queries.

- Future work:
  - We would like to consider XML query languages that produce XML trees.
    
    How do we define certain answers?
  
  - The notion of reasonable solutions needs to be investigated further.