XML data exchange

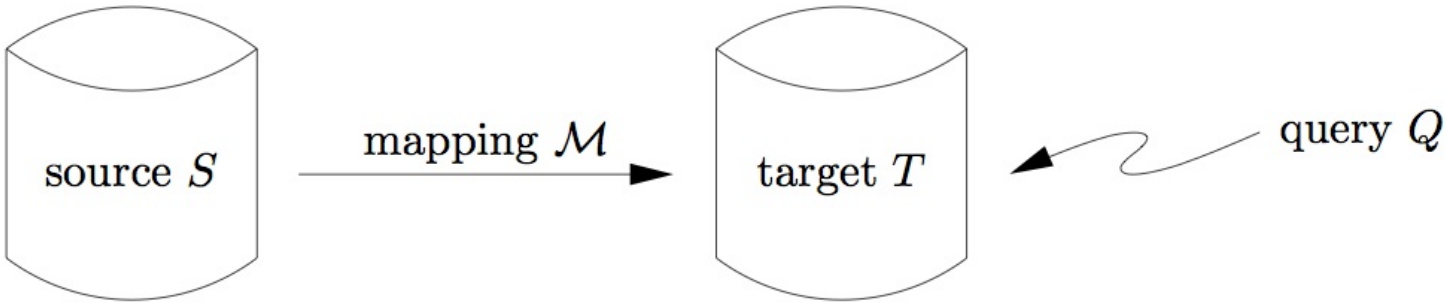
Amélie Gheerbrant

LFCS
University of Edinburgh

11/11/2010 - Dagstuhl DEIS'10

Outline

Data exchange



Goal:

- construct an instance T of the target schema (based on the source and the mapping)
- answer queries against the target data in a way consistent with the source data

Key notions: schema mappings, solutions, source-to-target tuple dependencies, certain answers

Main tasks in data exchange

Static analysis

- consistency of schema mappings (becomes an issue with XML)
- operations on mappings

Relatively small inputs, higher complexity bounds.

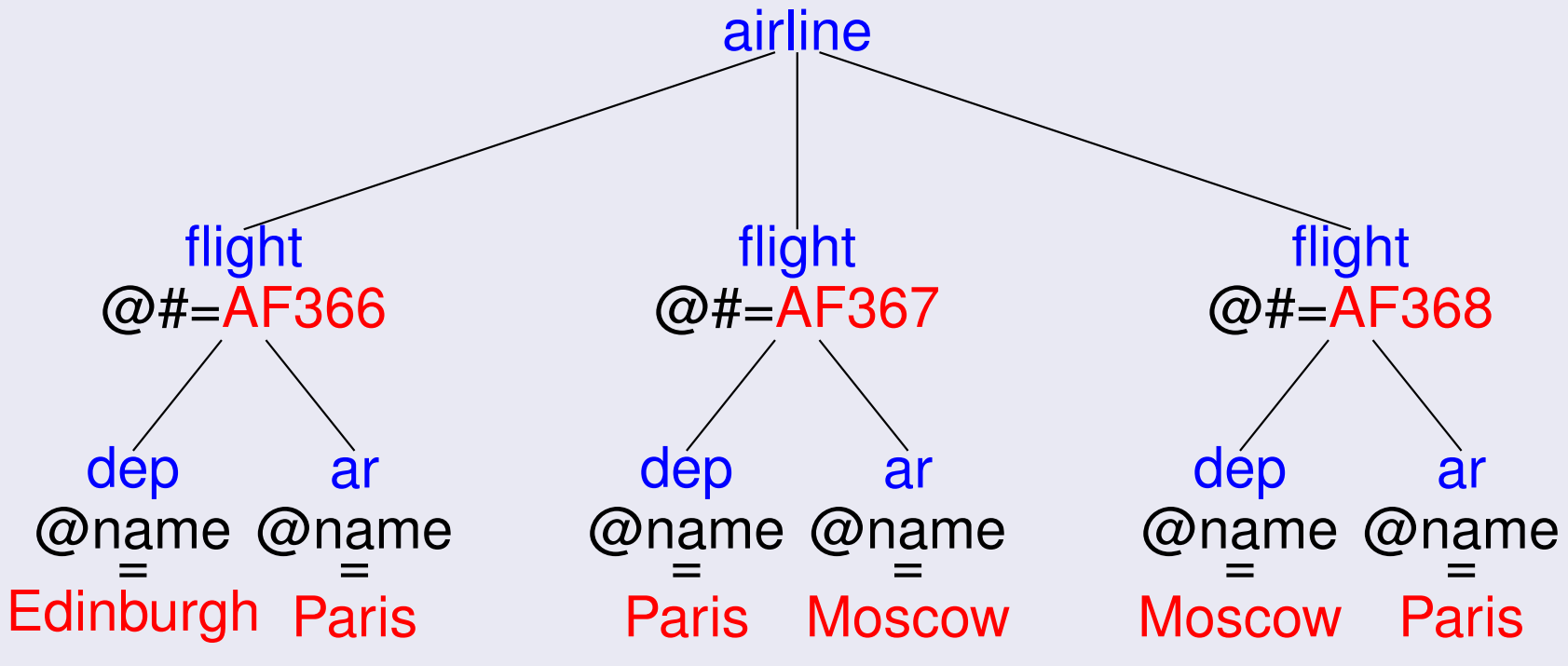
Dealing with data

- materializing target instances
- query answering

Typically large databases, only low complexity algorithms.

XML databases

An XML document



Theoretical abstraction of XML documents

Tree structures $T = \langle U, \downarrow, \rightarrow, lab, (\rho_a)_{a \in Att} \rangle$ over countable:

- labeling alphabet Γ (elements types, e.g., flight)
- set *Att* of attributes names (e.g., @name)
- set *Str* of possible attribute values (e.g., *Paris*)

where:

- U is an unranked finite tree domain
- $\downarrow$ and $\rightarrow$ are the child and the next sibling relations
- $lab : U \rightarrow \Gamma$ is the labeling function
- each ρ_a is a partial function from U to Str

DTD (Document Type Definition)

XML data exchange settings

Source and target DTD's
(instead of source and target relational schemas)

A DTD D over Γ and Att consists of two mappings

- $P : \Gamma \to$ regular expressions over $\Gamma - \{root\}$
- $A : \Gamma \to 2^{Att}$

A tree T conforms to a DTD D, i.e., $T \models D$ if

- its root is labeled $root$
- the set of attributes for a node labeled ℓ is $A(\ell)$ and the labels of its children, read left-to-right, form a string in the language of $P(\ell)$

Example

The previous tree conforms to any DTD D where:

$$flight : @\# \ ; \ dep : @name \ ; \ ar : @name$$

$$airline \rightarrow flight^* \text{ or } airline \rightarrow flight, flight, flight$$

and either

- $flight \rightarrow dep, ar$
- $flight \rightarrow dep, ar \mid flight$
- $flight \rightarrow dep, ar, time?$
- $flight \rightarrow dep, ar \mid depcity, arcity$
- etc

Nested-relational DTD's

A lot of things are easier for nested relational DTD's (important part of real world DTD's).

Nested relational DTD's

All productions are of the form $\ell \to \hat{\ell}_1, \ldots, \hat{\ell}_m$ where

- all ℓ_i's are distinct labels from Γ
- $\hat{\ell}_i$ is either ℓ_i, ℓ_i^*, $\ell_i^+ = \ell_i \ell_i^*$, or $\ell_i? = \ell_i | \epsilon$

and the graph in which we put an edge between ℓ and all the ℓ_i's for each production has no cycle (the DTD is not recursive)

Examples of non nested relational DTD's

DTD's D where:

$$airline \rightarrow flight^*$$

$$flight : @\# \ ; \ dep : @name \ ; \ ar : @name$$

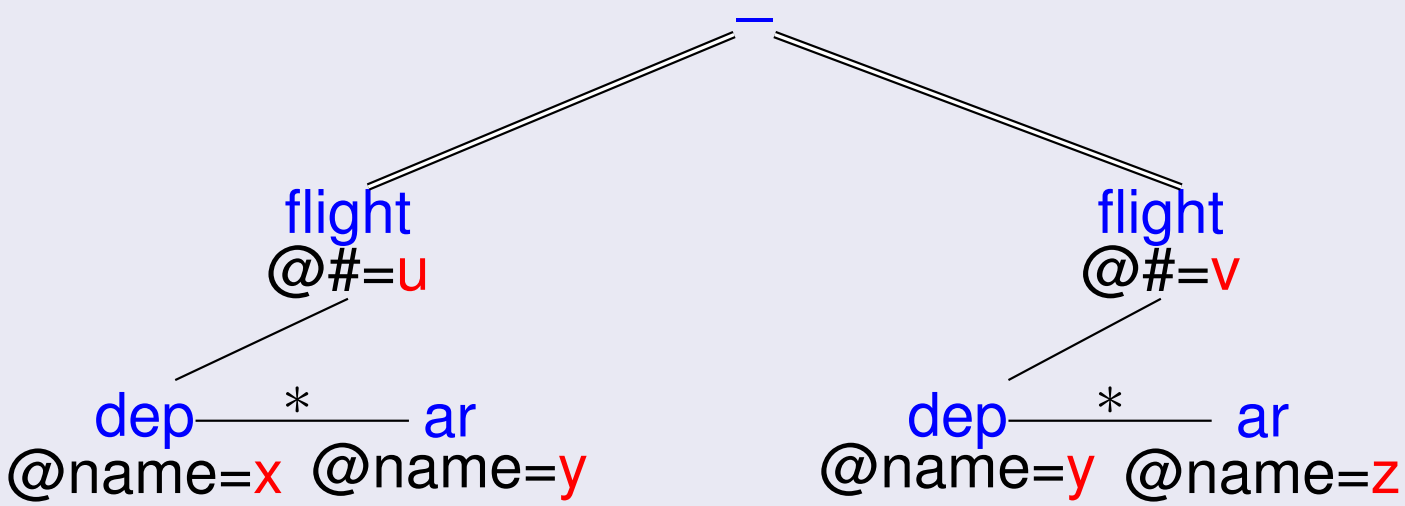
and either

- $flight \rightarrow dep, ar \mid flight$
- $flight \rightarrow dep, ar \mid depcity, arcity$

Schema mappings via tree patterns

st-tgds are defined using tree patterns.

$$\_//[flight(u)[dep(x) \rightarrow^* ar(y)], flight(v)[dep(y) \rightarrow^* ar(z)]]$$



- the wildcard _ can be used instead of label names
- variables correspond to attributes names
- special edges are used for $\rightarrow^*$ and $\downarrow^*$

Tree patterns: syntax

Tree patterns are given by:

π	:=	$\ell(\bar{x})[\lambda]$, where $\ell \in \Gamma \cup \{\_\}$	patterns
λ	:=	$\epsilon \mid \mu \mid //\pi \mid \lambda, \lambda$	sets
μ	:=	$\pi \mid \pi \rightarrow \mu \mid \pi \rightarrow^* \mu$	sequences

Nodes are described by subformulas $\ell(\bar{x})$ where $\bar{x}$ is a tuple of variables corresponding to the attributes of the node.

Generalized tree patterns

Equalities

Using variables allows to express things like:

$$airline[flight(x)[dep(y)], flight(z)[dep(y)]]$$

Equivalently:

$$airline[flight(x)[dep(y)], flight(z)[dep(w)]] \wedge y = w$$

In generalized tree patterns inequalities are also allowed

$$airline[flight(x)[dep(y)], flight(z)[dep(w)]] \wedge y = w \wedge x \neq z$$

Tarskian notion of satisfaction: $(T, s) \models \pi(\bar{a})$

The following tree patterns are satisfied at the root s of our tree

- $airline[flight(x)[dep(y) \rightarrow ar(z)]]$
- $airline[//\_(y) \rightarrow^* ar(z)] \wedge y \neq z$
- $airline[//dep(y)]$

For the following assignments:

- $x = AF366, y = Edinburgh, z = Paris$
- $x = AF367, y = Paris, z = Moscow$
- ...

Semantics of tree patterns via homomorphism

A tree pattern π can be seen as a *tree like* structure $S_\pi = \langle U, \downarrow, \downarrow^*, \rightarrow, \rightarrow^*, lab, \rho \rangle$ with root π.

Hence $T \models \pi$ iff there exists a homomorphism from π to T

A homomorphism between a pattern π and a tree T maps:

- the domain of π into the domain of T
- attribute values of the π_i's to attributes values of the image of the π_i's in T

and preserves:

- relations $\downarrow, \downarrow^*, \rightarrow, \rightarrow^*$
- labels (except the wildcard _)
- (in)equalities between attribute values

Schema mappings based on tree patterns

An XML schema mapping is a triple $\mathcal{M} = (D_s, D_t, \Sigma_{st})$ where

- D_s is the source DTD,
- D_t is the target DTD,
- Σ_{st} is a set of st-tgds of the form

$$\pi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \pi'(\bar{x}, \bar{z})$$

where π and π' are tree patterns

Solutions for S under $\mathcal{M}$

$T \in Sol_{\mathcal{M}}(S)$ with $S \models D_s$ if:

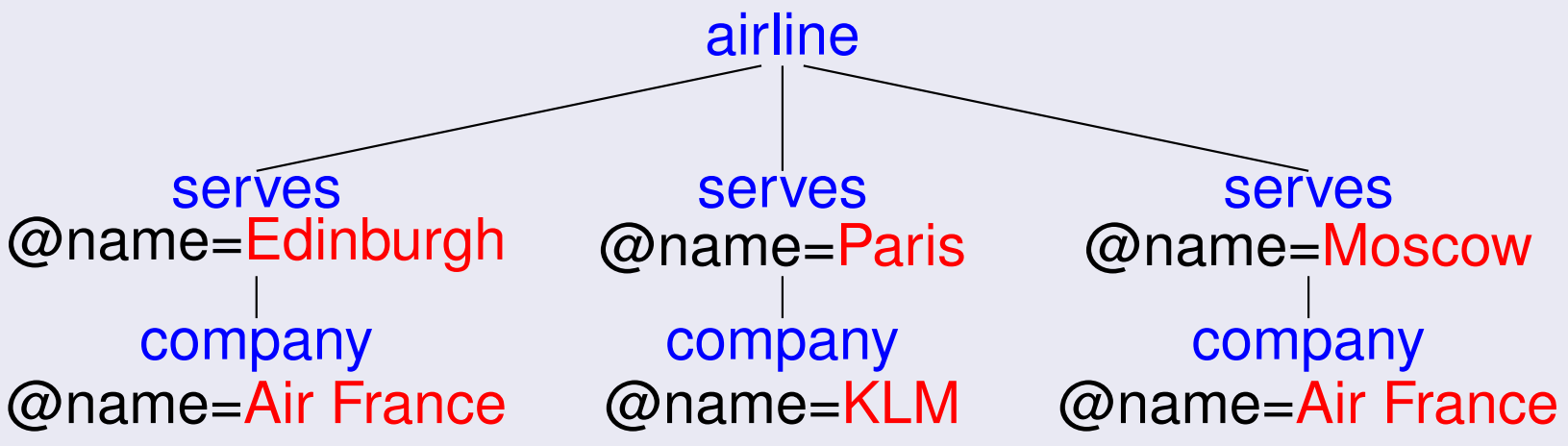
- $T \models D_t$
- (S, T) satisfy all st-tgds from Σ_{st}
  (i.e. whenever $S \models \pi(\bar{a}, \bar{b})$, there is $\bar{c}$ s.t. $T \models \pi'(\bar{a}, \bar{c})$)

Some schema mapping $\mathcal{M}$

target DTD: $airline \rightarrow serves^*; serves \rightarrow company*$
$serves : @name; company : @name$

st-tgd: $airline[//dep(x), //ar(y)] \rightarrow \exists z \exists z'$
$airline[//serves(x)[company(z)],$
$//serves(y)[company(z')]]$

A solution for $\mathcal{M}$



Classification of patterns and schema mappings

Restricted set of available axes and comparisons

Classes of patterns $\Pi(\sigma)$ with $\sigma \subseteq \{\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, =, \neq, \_\}$

Restricted set of features available in st-tgds

- $SM(\sigma)$=mappings where source and target side patterns come from $\Pi(\sigma)$
- $SM^{nr}(\sigma)$= nested relational schema mappings (whose target DTD's are nested relational)

All relational schema mappings fall in $SM^{nr}(\downarrow, =)$.

Complexity of evaluating tree patterns

Data complexity

Fix a pattern π and check for a given tree T and a tuple $\bar{a}$ whether $T \models \pi(\bar{a})$.

Combined complexity

Check for a given tree T, pattern π and tuple $\bar{a}$ whether $T \models \pi(\bar{a})$.

Complexity of evaluating tree patterns

- The data complexity is NLogSpace-complete.
- The combined complexity is in PTIME.

Complexity of the tree pattern satisfiability problem

The satisfiability problem

For a DTD D and a pattern $\pi(\bar{x})$; check whether there is a tree T that conforms to D and has a match for π.

Complexity

The satisfiability problem for tree patterns is NP-complete.

Complexity of schema mappings

Data complexity

- Fix a mapping $\mathcal{M}$ and check for two trees S, T, whether (S, T) satisfy $\mathcal{M}$ (membership problem).
- The data complexity is Logspace-complete.

Combined complexity

- Check, for two trees S, T and a mapping $\mathcal{M}$, whether (S, T) satisfy $\mathcal{M}$.
- The combined complexity is Π_2^p-complete.
- The combined complexity is in PTime if the maximum number of variables per pattern is fixed.

Consistency

Some XML schema mappings do not make sense.

An inconsistent XML schema mapping

- Source DTD:

$$airline \rightarrow flight^{+} \; ; \; flight : @\#$$

- Target DTD:

$$airline \rightarrow (nb, comp)^{+} \; ; \; nb : @\# \; ; \; comp : @name$$

- st-tgd:

$$airline[flight(x)] \rightarrow \exists y \; airline[flight[nb(x), comp(y)]]$$

The consistency problem

A mapping is

- consistent if $\mathcal{M}$ makes sense for some $S \models D_s$
- absolutely consistent if $\mathcal{M}(S)$ makes sense for all $S \models D_s$ (preserved for composition of mappings).

The consistency problem $CONS(\sigma)$

Input: A mapping $\mathcal{M} = (D_s, D_t, \Sigma_{st}) \in SM(\sigma)$
Question: Is $\mathcal{M}$ consistent?

The absolute consistency problem $ABCONS(\sigma)$

Input: A mapping $\mathcal{M} = (D_s, D_t, \Sigma_{st}) \in SM(\sigma)$
Question: Is $\mathcal{M}$ absolutely consistent?

The consistency problem: tools

- DTD's can be represented by tree automata.
- As long as they don't talk about data, tree patterns can also be represented using tree automata.
- For mappings without = and $\neq$, the consistency problem can be reduced to testing emptiness of tree automata.
- For absolute consistency, or when mappings allow comparison of data values, we cannot abstract from data, so we cannot use automata (we need to reason about counts of occurrences for different data values).

Complexity of the consistency problem

	arbitrary DTD's	nested relational DTD's
$CONS(\Downarrow)$	EXPTIME-complete	PTIME
$CONS(\Downarrow, \Rightarrow)$	EXPTIME-complete	PSPACE-hard
$CONS(\Downarrow, =)$	undecidable	NEXPTIME-complete
$CONS(\Downarrow, \Rightarrow, =)$	undecidable	undecidable
$ABCONS(\Downarrow)$	in EXPSPACE; NEXPTIME-hard	PTIME for $ABCONS(\downarrow)$

$\Downarrow$ stands here for $\{\downarrow, \downarrow^*, \}$

$\Rightarrow$ stands here for $\{\rightarrow, \rightarrow^*, \}$

XML data exchange

Goal of data exchange

Answer queries over target data in a way consistent with the source data.

XML data exchange

Tree patterns with $\neq$ (analogue of conjunctive queries with $\neq$).

Conjunctive tree queries (CTQ)

CTQ

A conjunctive tree query is an expression of the form

$$Q(\bar{x}) := \exists \bar{y} \pi_1(\bar{x}, \bar{y}) \wedge \ldots \wedge \pi_n(\bar{x}, \bar{y})$$

where the π_i's are tree patterns

UCTQ

Unions of conjunctive tree queries are of the form

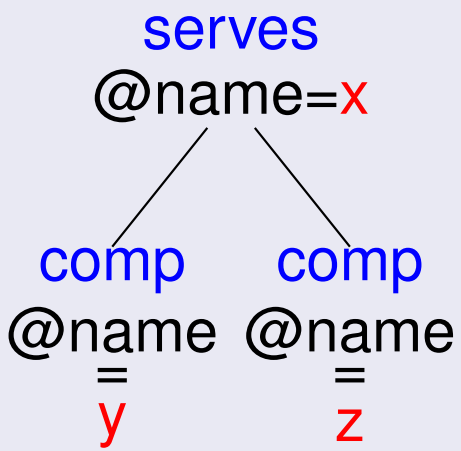
$$Q_1(\bar{x}) \cup \ldots \cup Q_m(\bar{x})$$

Subclasses of queries

$CTQ(\sigma)$ and $UCTQ(\sigma)$ for $\sigma \subseteq \{\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, =, \neq, \_\}$

Example

This query should return the set of cities which are served by more than one company

$\exists y \exists z \ ($  $) \wedge y \neq z$

Certain answers semantics

As queries return tuples, the certain answer approach from the relational case can also be used here.

Output of a query on a tree

$$Q(T) = \{\bar{a} \mid T \models \exists \bar{y} \pi(\bar{a}, \bar{y})\}$$

Adaptation of the relational case

For a mapping $\mathcal{M}$, a query Q and a tree $S \models D_s$:

$$certain_{\mathcal{M}}(Q, S) = \bigcap \{Q(T) \mid T \text{ is a solution for } S \text{ under } \mathcal{M}\}$$

The data exchange problem

We are interested in the following problem, for fixed $\mathcal{M}$ and Q:

Problem: $certain_{\mathcal{M}}(Q)$

Input: a source tree S, a tuple $\bar{s}$
Question: $\bar{s} \in certain_{\mathcal{M}}(Q, S)$

Relational case

The problem $certain_{\mathcal{M}}(Q)$ is

- *coNP*-complete for conjunctive queries with inequalities
- in *Ptime* for conjunctive queries without inequalities

Complexity: upper bounds

coNP results

For every:

- schema mapping $\mathcal{M}$ from $SM(\Downarrow, \Rightarrow, =, \neq)$
- query Q from $UCTQ(\Downarrow, \Rightarrow, =, \neq)$

the problem $certain_{\mathcal{M}}(Q)$ is in *coNP*.

$certain_{\mathcal{M}}(Q)$ easily becomes coNP-hard

This can come from:

- DTD's (disjunctions)
- st-tgds (descendant, wildcard)
- queries (horizontal navigation, inequalities)

Complexity: easy restrictions

A robust subclass: fully specified mappings, nested relational DTD's

For every:

- schema mapping $\mathcal{M}$ from $SM^{nr}(\downarrow, \rightarrow, \rightarrow^*, =, \neq)$
- query Q from $UCTQ(\downarrow, \downarrow^*, \_ =)$

the problem $certain_{\mathcal{M}}(Q)$ is computable in polynomial time.

More precisely : there is a full dichotomy between NP-complete and PTime classes.

- Depends on regular expressions in target DTD's
- The actual definition is quite involved, but $(A \mid B)^*$; $A, B^+, C^*, D?$; $(A^* \mid B^*)$, $(C, D)^*$ are "good", while $A, (B \mid C)$ is "bad"

How these easy restrictions are obtained: universal solutions

Restrictions are obtained by showing that certain answers can be computed via universal solutions in polynomial time.

Universal solution

U is a universal solution for S under $\mathcal{M}$ if

- U is a solution for S
- for each other solution T, there is a homomorphism from U to T preserving data values used in S

If $Q \in UCTQ(\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \_ =)$, then for every $\bar{a}$

$$\bar{a} \in certain_{\mathcal{M}}(Q, S) \Leftrightarrow \bar{a} \in Q(U)$$

A case with no universal solution

source DTD: $root$

target DTD: $root \rightarrow A|B$

source instance: $root$

st-tgd: $root \rightarrow r[\_]$

Implementing XML data exchange by a relational system

- Translate CTQ into CQ and let the relational system do the computation.
- This is possible only for robust subclasses.
- A lot of cases become coNP-complete.

"Real life" XML schema mapping tools for XML data exchange and integration

- "Good' fragment of XML data exchange has been implemented by the Clio system.
- Instead of native XML, the documents are transformed into nested-relational databases.

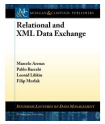
XML to XML queries

- Our query languages return tuples.
- But XML query languages such as XQuery take XML trees and produce XML trees.
- So what about XML to XML query languages?

Summary

- st-tgds state how patterns over the source translate into patterns over the target
- XML schema mappings can easily be inconsistent ($\neq$ relational case)
- Consistency undecidable in general (with $\neq$ of data value). Otherwise, exponential time (and tractable subclasses).
- Query answering is often intractable (coNP-complete), tractable restrictions:
  - nested relational mappings with $\downarrow$, $\rightarrow$, $\rightarrow^*$, $=$ and $\neq$ only
  - queries with $\downarrow$, $\downarrow^*$, _, $=$ only

Bibliographic References



- Relational and XML Data Exchange
  (Arenas, Barceló, Libkin, Murlak, 2010)
- On the tradeoff between mapping and querying power in XML data exchange
  (Amano, David, Libkin, Murlak - ICDT 2010)
- Certain answers for XML queries
  (David, Libkin, Murlak - PODS 2010)
- XML schema mappings
  (Amano, Libkin, Murlak - PODS 2009)
- *XML data exchange*
  (Arenas, Libkin - JACM 2008)
- Mapping-driven XML transformation
  (Jiang, Ho, Popa, Han - WWW 2007)
- Nested mappings: schema mapping reloaded
  (Fuxman et al. - VLDB 2006)

The book (but now: [scale=0.6])