XML Data Exchange: Consistency and Query Answering

Marcelo Arenas
U. of Toronto

Leonid Libkin
U. of Toronto

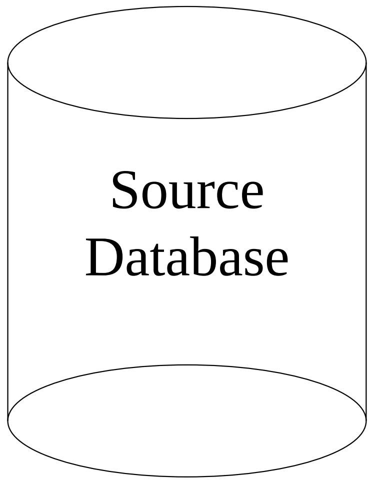
The Problem of Data Exchange

- Data exchange is the problem of finding an instance of a target schema, given an instance of a source schema and a specification of the relationship between the source and the target.

- Such a target instance should correctly represent information from the source instance under the constraints imposed by the target schema.

- It should also allow one to evaluate queries on the target instance in a way that is semantically consistent with the source data.

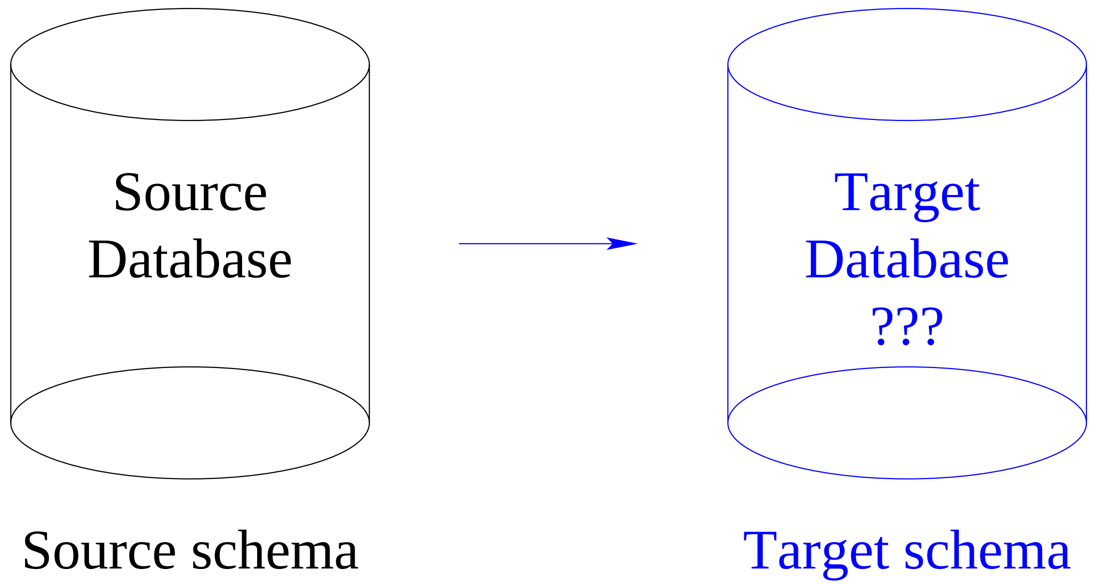
Data Exchange



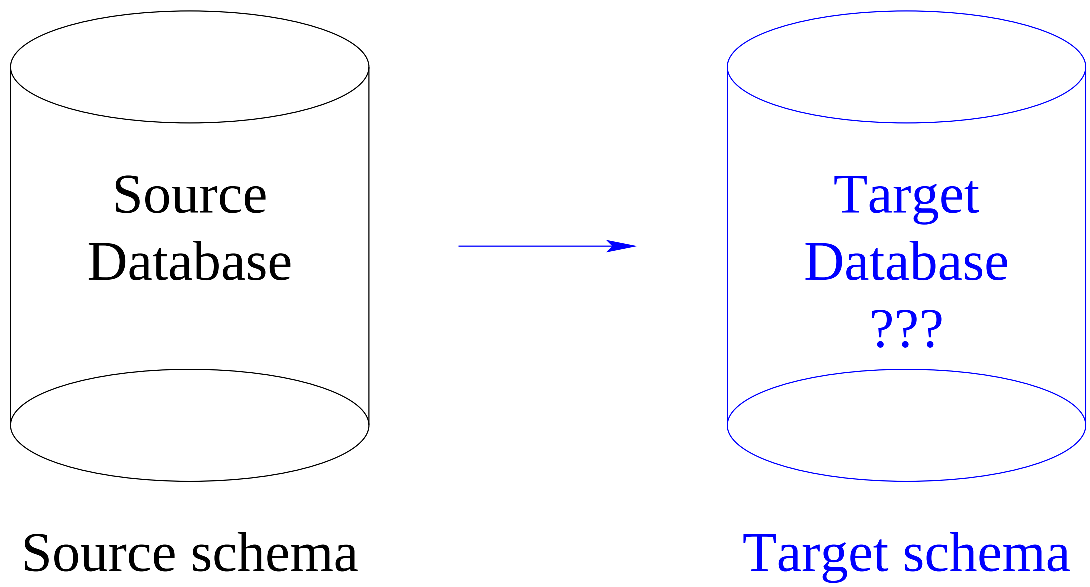
Source schema

Target schema

Data Exchange



Data Exchange



Query over the target schema: Q

How to answer Q so that the answer is consistent with the data in the source database?

Relational Data Exchange Settings

Data Exchange Setting: $(\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$

$\mathbf{S}$: Source schema.

$\mathbf{T}$: Target schema.

$\Sigma_{\mathbf{ST}}$: Set of source-to-target dependencies.

- Source-to-target dependency:

$$\psi_{\mathbf{T}}(\bar{x}, \bar{z}) :- \varphi_{\mathbf{S}}(\bar{x}, \bar{y}).$$

- $\varphi_{\mathbf{S}}(\bar{x}, \bar{y})$: conjunction of atomic formulas over $\mathbf{S}$.
- $\psi_{\mathbf{T}}(\bar{x}, \bar{z})$: conjunction of atomic formulas over $\mathbf{T}$.

Example: Relational Data Exchange Setting

- $\mathbf{S} \quad = \quad \mathit{Book}(\mathit{Title}, \mathit{AName}, \mathit{Aff})$

- $\mathbf{T} \quad = \quad \mathit{Writer}(\mathit{Name}, \mathit{BTitle}, \mathit{Year})$

- $\Sigma_{\mathbf{ST}} \quad = \quad \mathit{Writer}(x_2, x_1, z_1) \text{ :- } \mathit{Book}(x_1, x_2, y_1).$

Relational Data Exchange Problem

- Given a source instance I, find a target instance J such that (I, J) satisfies $\Sigma_{\mathbf{ST}}$.
  - (I, J) satisfies $\psi_{\mathbf{T}}(\bar{x}, \bar{z})$:– $\varphi_{\mathbf{S}}(\bar{x}, \bar{y})$ if whenever I satisfies $\varphi_{\mathbf{S}}(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that J satisfies $\psi_{\mathbf{T}}(\bar{a}, \bar{c})$.
  - J is called a solution for I.

- Previous example:

I:

Book	*Title*	*AName*	*Aff*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

Relational Data Exchange Problem

Possible solutions:

J_1:

Writer	*Name*	*BTitle*	*Year*
	Hungerford	Algebra	1974
	Royden	Real Analysis	1988

J_2:

Writer	*Name*	*BTitle*	*Year*
	Hungerford	Algebra	$\perp_1$
	Royden	Real Analysis	$\perp_2$

Query Answering

- Q is a query over target schema.

  What does it mean to answer Q?

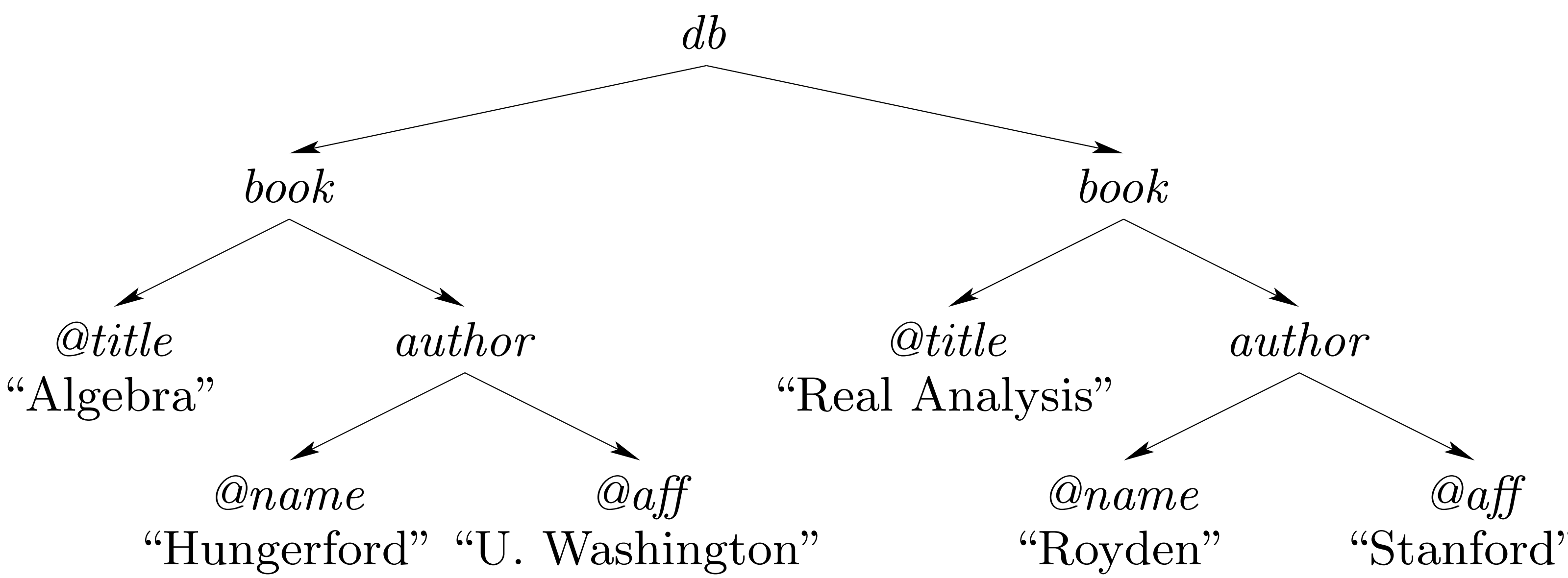
$$\underline{certain}(Q, I) \;=\; \bigcap_{J \text{ is a solution for } I} Q(J)$$

- Previous example:
  - $\underline{certain}(\exists y \exists z\, Writer(x, y, z),\ I) = \{\text{Hungerford},\ \text{Royden}\}$

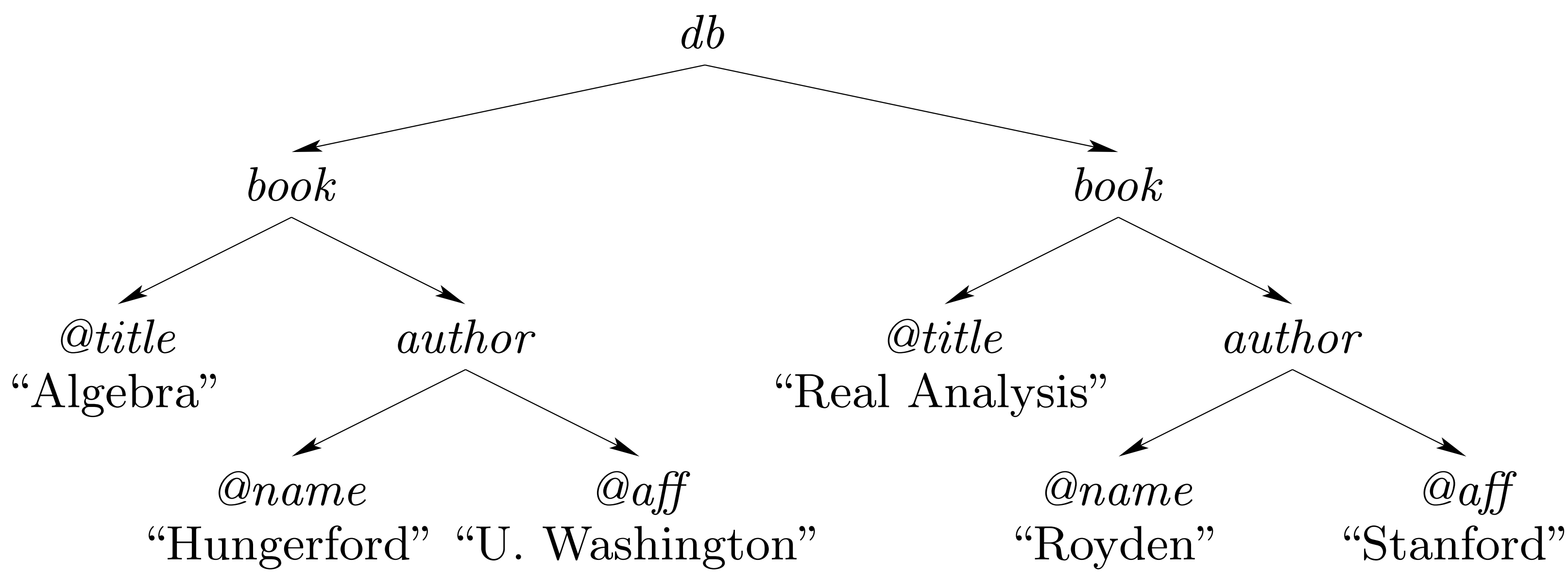
Outline

- XML data exchange settings.
    - XML source-to-target dependencies.
- Consistency of XML data exchange settings.
- Query answering in XML data exchange.
- Future work.

XML Documents

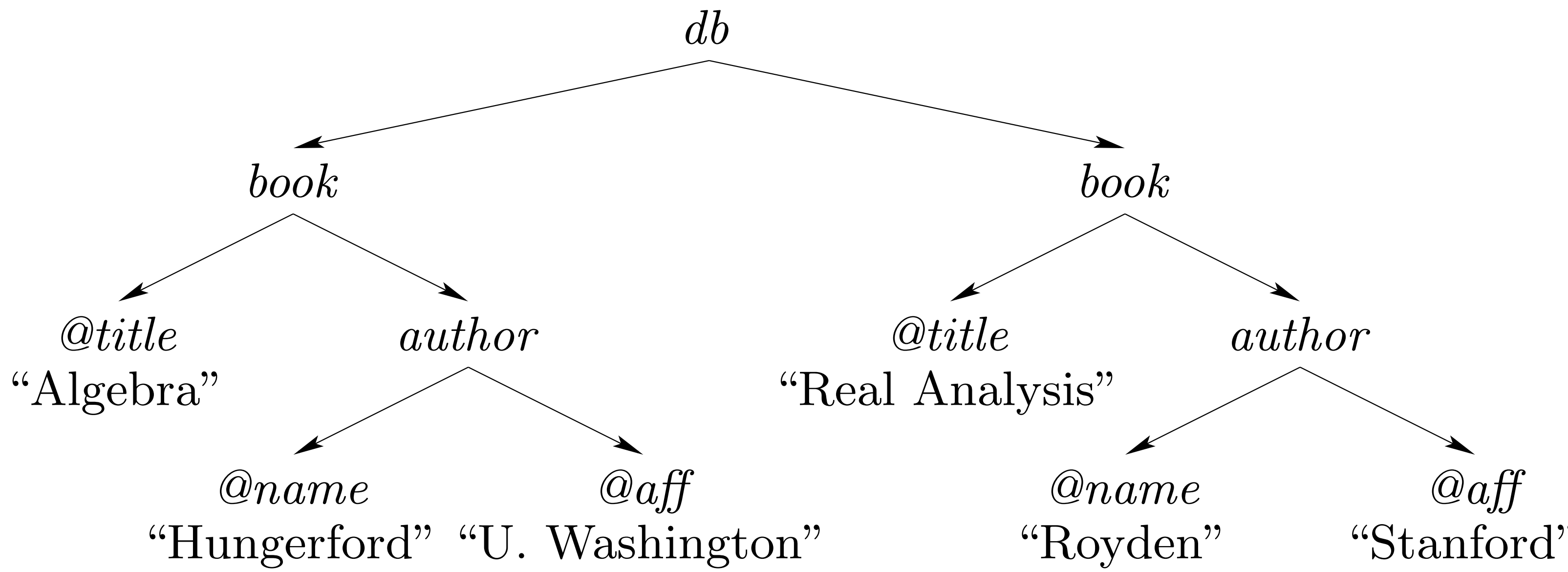


XML Documents



$$\text{DTD}: \quad \begin{array}{rcl} db & \to & book^{+} \\ book & \to & author^{+} \\ author & \to & \varepsilon \end{array}$$

XML Documents



$$\text{DTD}: \quad \begin{array}{rcl} db & \rightarrow & book^{+} \\ book & \rightarrow & author^{+} \\ author & \rightarrow & \varepsilon \end{array} \qquad \begin{array}{rcl} book & \rightarrow & @title \\ author & \rightarrow & @name,\ @aff \end{array}$$

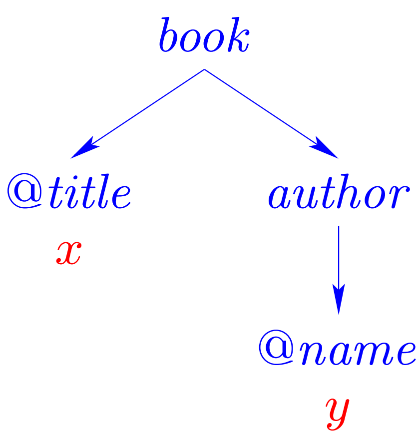
XML Data Exchange Settings

- Instead of source and target relational schemas, we have source and target DTDs.

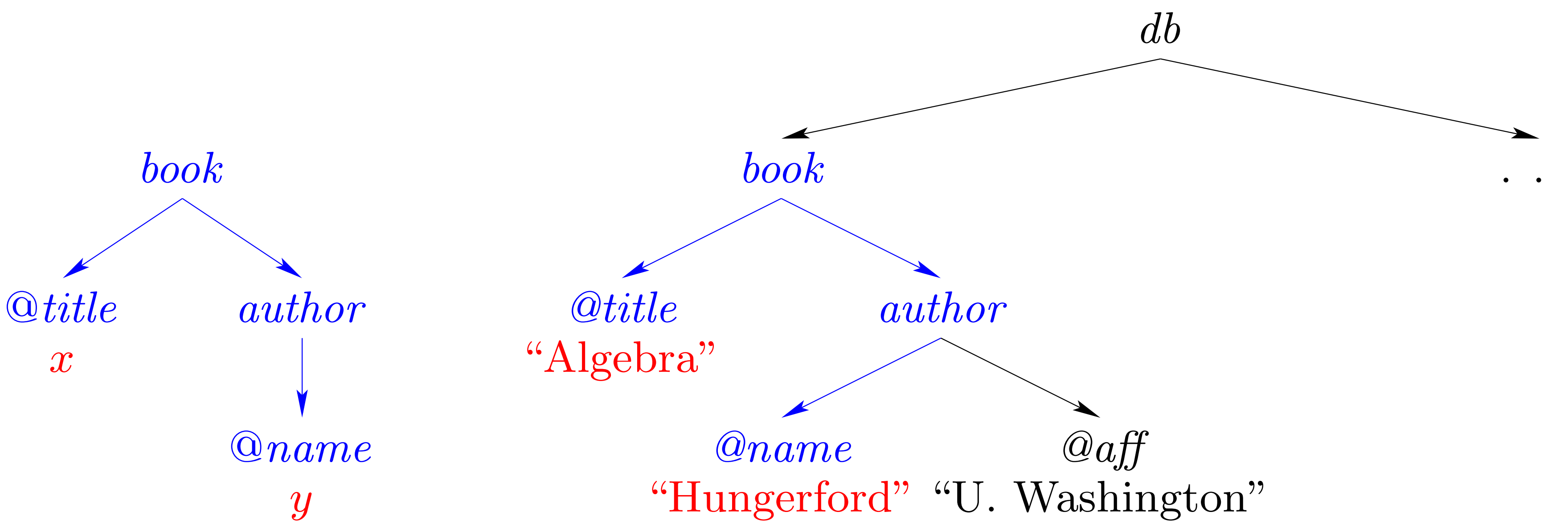
- But what are the source-to-target dependencies?

  To define them, we use tree patterns ...

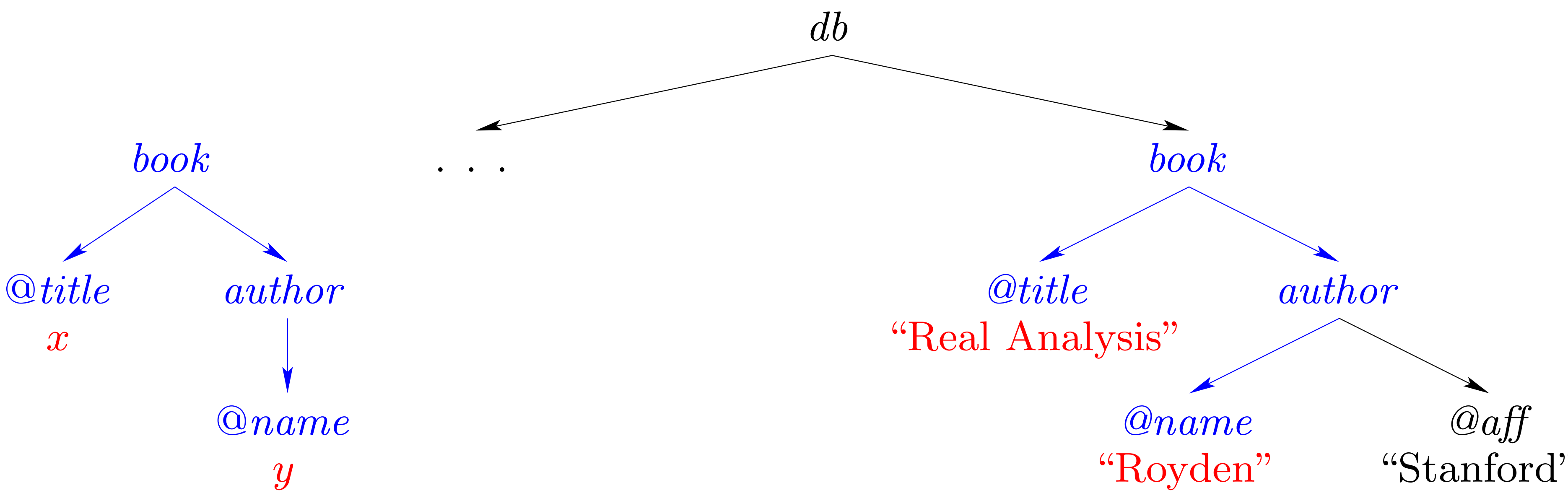
Tree Patterns: Example



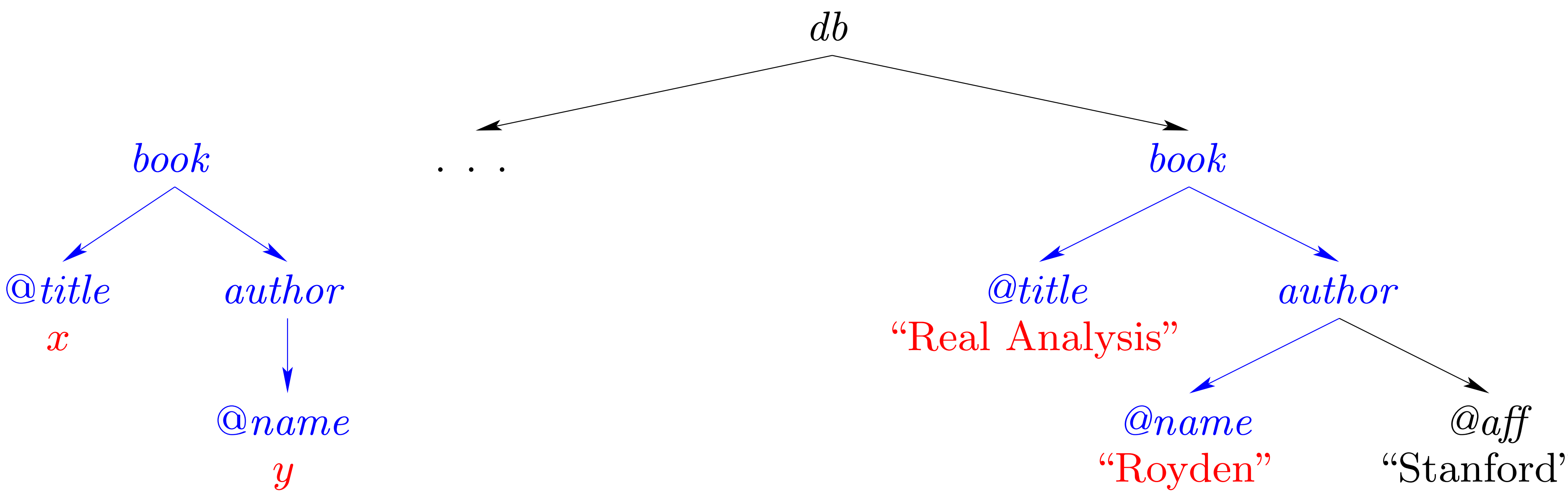
Tree Patterns: Example



Tree Patterns: Example



Tree Patterns: Example



Collect tuples (x, y): $(\text{Algebra}, \text{Hungerford})$, $(\text{Real Analysis}, \text{Royden})$

Tree Patterns

- Example: $book(@title = x)[author(@name = y)]$.

- Language also includes wildcard $\_$ (matching more than one symbol) and descendant operator $//$.

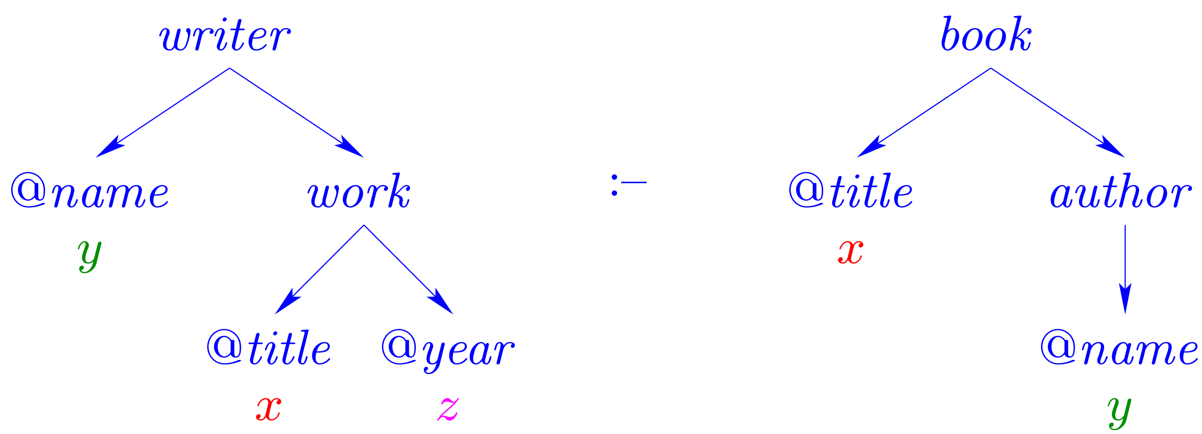
XML Source-to-target Dependencies

- Source-to-target dependency (STD):

$$\psi_{\mathbf{T}}(\bar{x}, \bar{z}) \;\text{:--}\; \varphi_{\mathbf{S}}(\bar{x}, \bar{y}),$$

  where $\varphi_{\mathbf{S}}(\bar{x}, \bar{y})$ and $\psi_{\mathbf{T}}(\bar{x}, \bar{z})$ are tree-pattern formulas over the source and target DTDs, resp.

- Example:



XML Data Exchange Settings

XML Data Exchange Setting: $(D_{\mathbf{S}}, D_{\mathbf{T}}, \Sigma_{\mathbf{ST}})$

$D_{\mathbf{S}}$: Source DTD.

$D_{\mathbf{T}}$: Target DTD.

$\Sigma_{\mathbf{ST}}$: Set of XML source-to-target dependencies.

Each constraint in $\Sigma_{\mathbf{ST}}$ is of the form $\psi_{\mathbf{T}}(\bar{x}, \bar{z}) :- \varphi_{\mathbf{S}}(\bar{x}, \bar{y})$.

- $\varphi_{\mathbf{S}}(\bar{x}, \bar{y})$: tree-pattern formula over $D_{\mathbf{S}}$.
- $\psi_{\mathbf{T}}(\bar{x}, \bar{z})$: tree-pattern formula over $D_{\mathbf{T}}$.

Example: XML Data Exchange Setting

- Source DTD:

$$\begin{array}{rcl} db & \to & book^+ \\ book & \to & author^+ \\ author & \to & \varepsilon \end{array} \qquad \begin{array}{rcl} & & \\ book & \to & @title \\ author & \to & @name, @aff \end{array}$$

- Target DTD:

$$\begin{array}{rcl} bib & \to & writer^+ \\ writer & \to & work^+ \\ work & \to & \varepsilon \end{array} \qquad \begin{array}{rcl} & & \\ writer & \to & @name \\ work & \to & @title, @year \end{array}$$

- $\Sigma_{\mathbf{ST}}$:

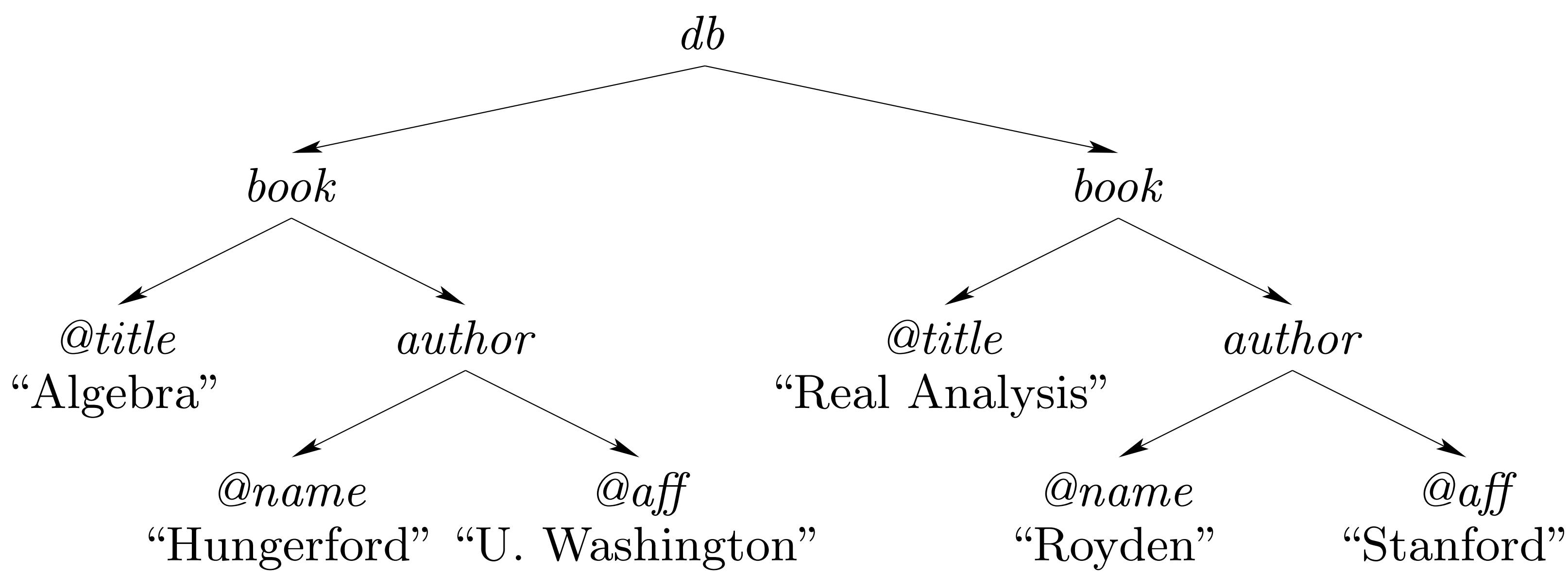
$$writer(@name = y)[work(@title = x, @year = z)] \;\; :- $$
$$book(@title = x)[author(@name = y)].$$

XML Data Exchange Problem

- Given a source tree T, find a target tree T' such that (T, T') satisfies $\Sigma_{\mathbf{ST}}$.
  - (T, T') satisfies $\psi_{\mathbf{T}}(\bar{x}, \bar{z})$:– $\varphi_{\mathbf{S}}(\bar{x}, \bar{y})$ if whenever T satisfies $\varphi_{\mathbf{S}}(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that T' satisfies $\psi_{\mathbf{T}}(\bar{a}, \bar{c})$.
  - T' is called a solution for T.

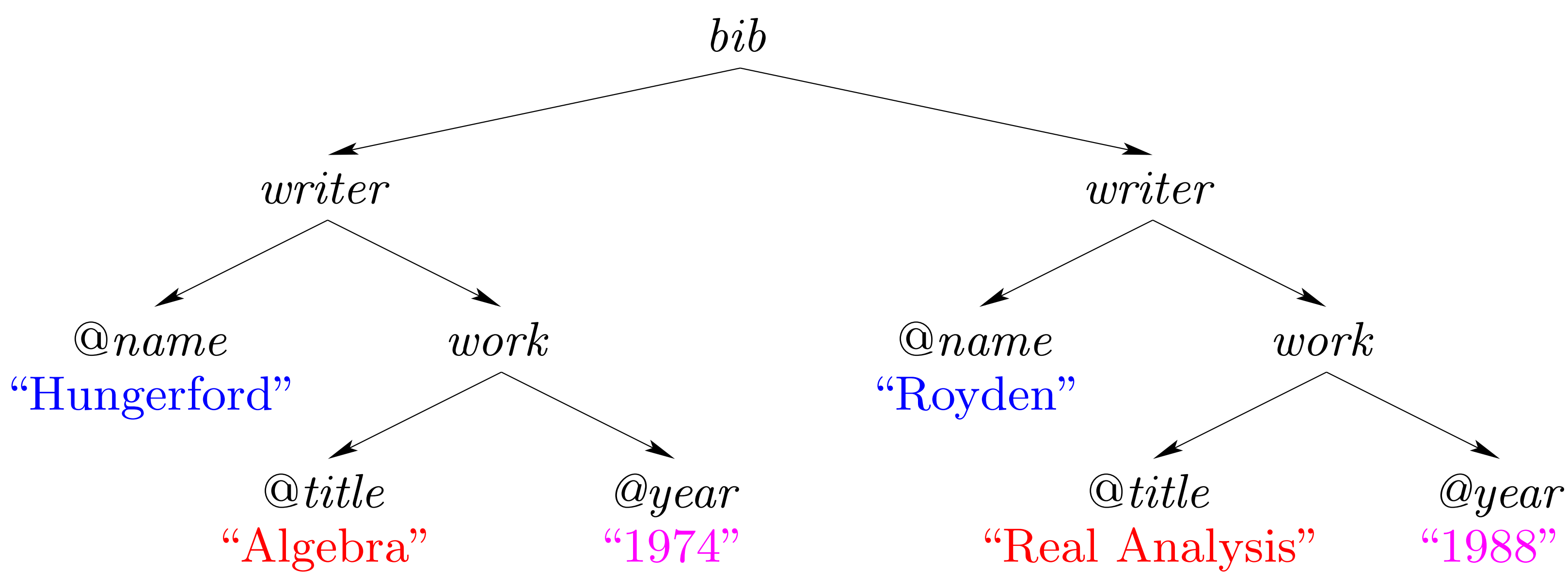
XML Data Exchange Problem

Let T be our original tree:



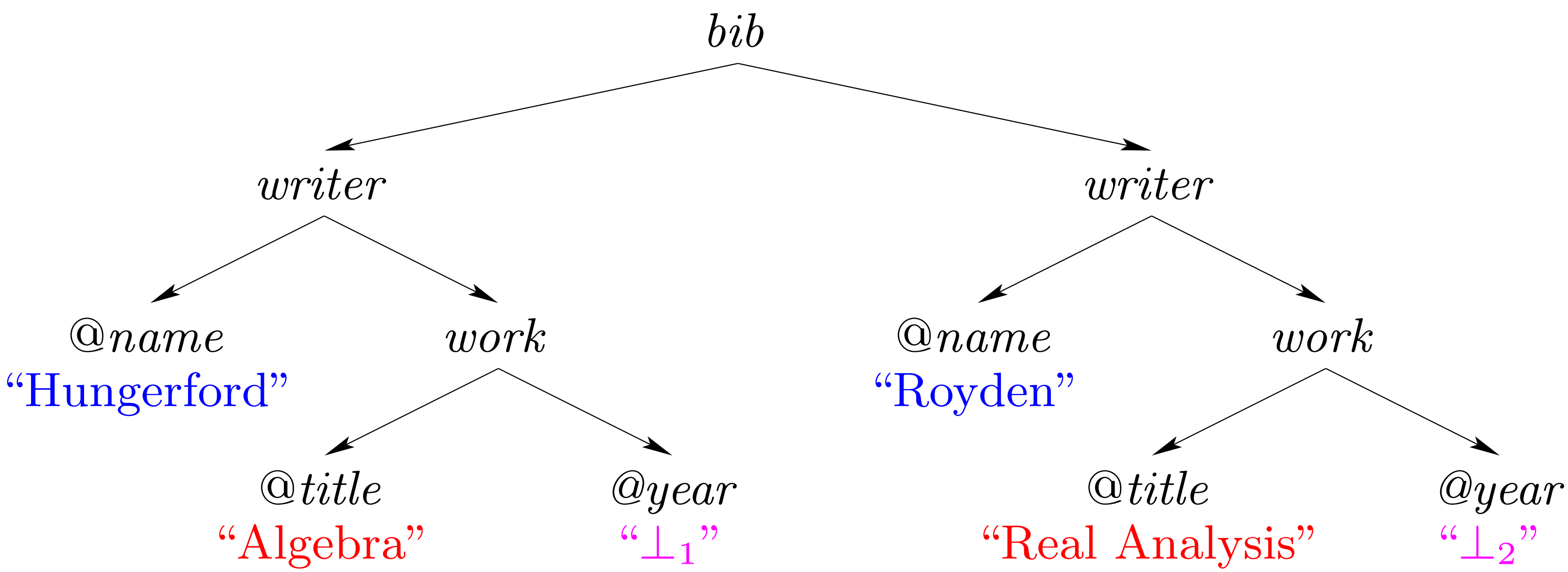
XML Data Exchange Problem

A solution for T:



XML Data Exchange Problem

Another solution for T:



Consistency of XML Data Exchange Settings

- What if we have target DTD

$$
\begin{array}{rcl}
bib & \rightarrow & writer^{+} \\
writer & \rightarrow & novelist^{*}, poet^{*} \\
novelist & \rightarrow & work^{+} \\
poet & \rightarrow & work^{+} \\
work & \rightarrow & \varepsilon
\end{array}
\qquad
\begin{array}{rcl}
 & & \\
writer & \rightarrow & @name \\
 & & \\
 & & \\
work & \rightarrow & @title,\ @year
\end{array}
$$

  in our previous example?

- The setting becomes inconsistent!
  - There are no T conforming to $D_{\mathbf{S}}$ and T' conforming to $D_{\mathbf{T}}$ such that (T, T') satisfies $\Sigma_{\mathbf{ST}}$.

Consistency of XML Data Exchange Settings

- An XML data exchange setting is inconsistent if it does not admit solutions for any given source tree. Otherwise it is consistent.

- A relational data exchange setting is always consistent.

- An XML data exchange setting is not always consistent.
  - What is the complexity of checking whether a setting is consistent?

Bad News: General Case

Theorem Checking if an XML data exchange setting is consistent is EXPTIME-complete.

Known results on containment of XPath expressions as well as universality of tree automata imply that EXPTIME-hardness is unavoidable.

Good News: Consistency for Commonly used DTDs

A large number of DTDs that occur in practice have rules of the following form:

$$\ell \;\rightarrow\; \hat{\ell}_1, \ldots, \hat{\ell}_m,$$

where all the ℓ_i's are distinct, and $\hat{\ell}$ is one of the following: ℓ, or ℓ^*, or ℓ^+, or $\ell?$

Theorem For non-recursive DTDs that only have these rules, checking if an XML data exchange setting is consistent is solvable in time $O\big((\|D_{\mathbf{S}}\| + \|D_{\mathbf{T}}\|) \cdot \|\Sigma_{\mathbf{ST}}\|^2\big)$.

Query Answering in XML Data Exchange

- Decision to make: what is our query language?

- XML query languages such as XQuery take XML trees and produce XML trees.
  - This makes it hard to talk about certain answers.

- We use a query language that produces tuples of values.

Conjunctive Tree Queries

- Query language $\mathcal{CTQ}^{//}$ is defined by

$$Q \quad := \quad \varphi \;\mid\; Q \wedge Q \;\mid\; \exists x\, Q,$$

where φ ranges over tree-pattern formulas.

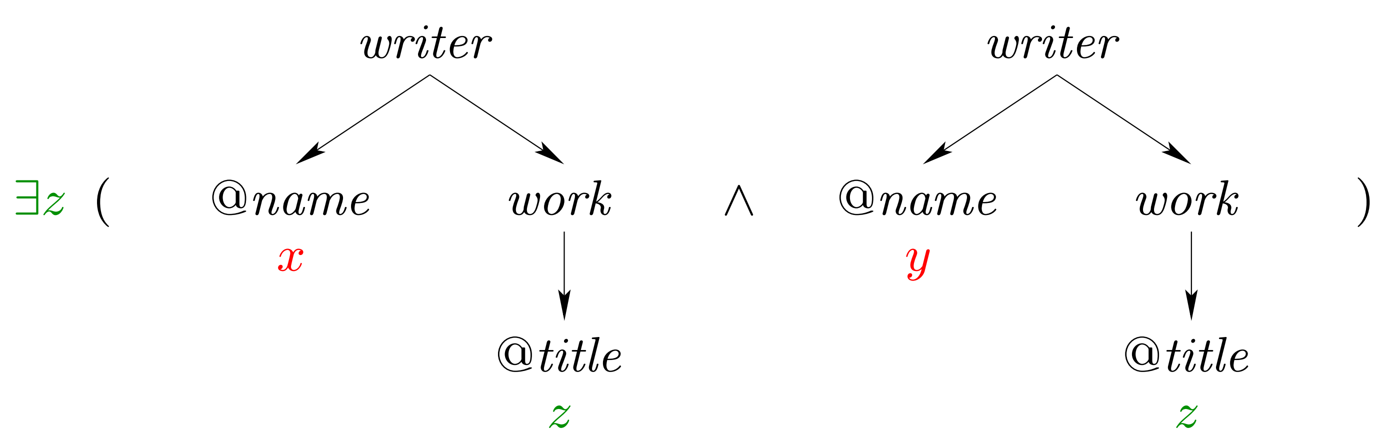
- By disallowing descendant // we obtain restriction $\mathcal{CTQ}$.

- Results extend to unions of conjunctive queries.

Example: Conjunctive Tree Query

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

$Q(x, y) \ :=$



Computing Certain Answers

- Semantics: as in the relational case.

$$\underline{certain}(Q,T) \quad = \bigcap_{T' \text{ is a solution for } T} Q(T').$$

- Given data exchange setting $(D_{\mathbf{S}}, D_{\mathbf{T}}, \Sigma_{\mathbf{ST}})$ and query Q:

PROBLEM:	$\textsc{CertAnsw}(Q)$.
INPUT:	Tree T conforming to $D_{\mathbf{S}}$ and tuple $\bar{a}$.
QUESTION:	Is $\bar{a} \in \underline{certain}(Q,T)$?

Computing Certain Answers: General Picture

Theorem For every XML data exchange setting and $\mathcal{CTQ}^{//}$-query Q, $\textsc{CertAnsw}(Q)$ is in coNP.

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

Theorem There exist an XML data exchange setting and a $\mathcal{CTQ}^{//}$-query Q such that $\textsc{CertAnsw}(Q)$ is coNP-hard.

We want to find tractable cases ...

Computing Certain Answers: Finding Tractable Cases

Theorem Suppose one of the following is allowed in tree patterns over the target in STDs:

- descendant operator $//$, or
- wildcard $\_$, or
- patterns that do not start at the root.

Then one can find source and target DTDs and a $\mathcal{CTQ}$-query Q such that $\textsc{CertAnsw}(Q)$ is coNP-complete.

Remark: Even if all the rules in the DTDs are of the form:

$$\ell \;\rightarrow\; (\ell_1 \mid \cdots \mid \ell_n)^*$$

where all the ℓ_i's are distinct.

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.

Computing Certain Answers: Towards a Classification

Given a class $\mathcal{C}$ of regular expressions and a class $\mathcal{Q}$ of queries:

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

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

Remark (Ladner): if PTIME $\neq$ NP, there are problems in coNP which are neither tractable nor coNP-complete.

Computing Certain Answers: Towards a Classification

- Our classification is based on classes of regular expressions used in DTDs.

- We only impose one restriction to these classes:

  - They must contain the simplest type of regular expressions.

- Such classes will be called admissible.

Computing Certain Answers: Dichotomy

Theorem

1) Every admissible class $\mathcal{C}$ of regular expressions is either tractable or coNP-complete for $\mathcal{CTQ}^{//}$.

   Remark: also holds for unions of conjunctive queries.

2) Moreover, given an XML data exchange setting, it is decidable whether the regular expressions used in the source and the target DTD belong to a tractable class.

A Tractable Case

- Idea: given a source tree T, compute a solution $T^\star$ for T such that

$$\underline{certain}(Q, T) \;=\; remove\_null\_tuples(Q(T^\star)).$$

- $T^\star$ is a canonical solution for T.

- We compute $T^\star$ in two steps:

  - We use STDs to compute a canonical pre-solution $cps(T)$ from T.
  - Then we use target DTD to compute $T^\star$ from $cps(T)$.

Example: XML Data Exchange Setting

- Source DTD:

$$\begin{array}{lllllll} r & \rightarrow & A^*, B^* & & & & \\ A & \rightarrow & \varepsilon & \qquad & A & \rightarrow & @\ell \\ B & \rightarrow & \varepsilon & \qquad & B & \rightarrow & @\ell \end{array}$$

- Target DTD:

$$\begin{array}{lllllll} r & \rightarrow & (C, D)^* & & & & \\ C & \rightarrow & \varepsilon & \qquad & C & \rightarrow & @m \\ D & \rightarrow & E & & & & \\ E & \rightarrow & \varepsilon & \qquad & E & \rightarrow & @n \end{array}$$

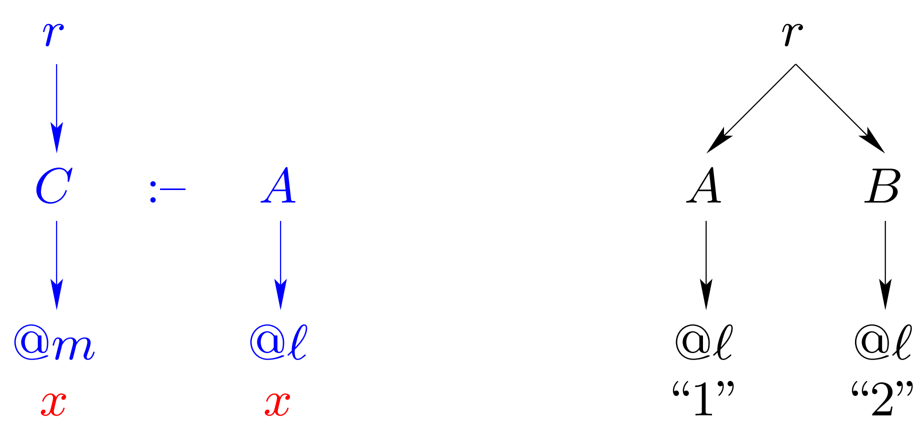
- $\Sigma_{\mathbf{ST}}$:

$$\begin{array}{lcl} r[C(@m = x)] & :- & A(@\ell = x), \\ r[C(@m = x)] & :- & B(@\ell = x). \end{array}$$

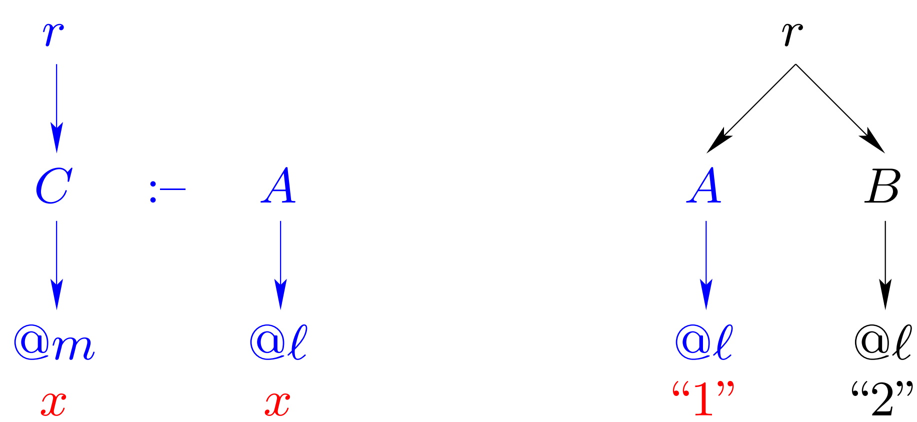
Example: Computing Canonical Pre-solution

$$
\begin{tikzcd}
 & r \arrow[dl] \arrow[dr] & \\
A \arrow[d] & & B \arrow[d] \\
\substack{@\ell \\ \text{``1''}} & & \substack{@\ell \\ \text{``2''}}
\end{tikzcd}
$$

Example: Computing Canonical Pre-solution



Example: Computing Canonical Pre-solution



Example: Computing Canonical Pre-solution

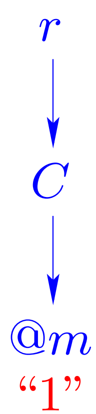
$$
\begin{tikzcd}[row sep=small]
r \arrow[d] \\
C \arrow[d] \\
@m \\
\text{“1”}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}[row sep=small]
r \arrow[d] & & \\
C \arrow[d] & {:\!-} & A \arrow[d] \\
@m & & @\ell \\
x & & x
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}[row sep=small]
& r \arrow[dl] \arrow[dr] & \\
A \arrow[d] & & B \arrow[d] \\
@\ell & & @\ell \\
\text{“1”} & & \text{“2”}
\end{tikzcd}
$$

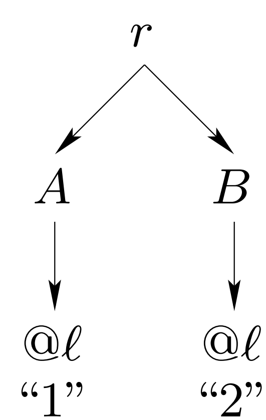
Example: Computing Canonical Pre-solution

$$\begin{tikzcd} r \arrow[d] \\ C \arrow[d] \\ @m \end{tikzcd}$$

"1"

Example: Computing Canonical Pre-solution





Example: Computing Canonical Pre-solution

$$
\begin{tikzcd}
r \arrow[d] \\
C \arrow[d] \\
@m \\[-1.5em]
\text{“1”}
\end{tikzcd}
$$

$$
\begin{tikzcd}
r \arrow[d] & & \\
C \arrow[d] & :- & B \arrow[d] \\
@m & & @\ell \\[-1.5em]
x & & x
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& r \arrow[dl] \arrow[dr] & \\
A \arrow[d] & & B \arrow[d] \\
@\ell & & @\ell \\[-1.5em]
\text{“1”} & & \text{“2”}
\end{tikzcd}
$$

Example: Computing Canonical Pre-solution

$$
\begin{tikzcd}
r \arrow[d] \\
C \arrow[d] \\
@m \\[-20pt]
\text{“1”}
\end{tikzcd}
$$

$$
\begin{tikzcd}
r \arrow[d] & & \\
C \arrow[d] & :- & B \arrow[d] \\
@m & & @\ell \\[-20pt]
x & & x
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & r \arrow[dl] \arrow[dr] & \\
A \arrow[d] & & B \arrow[d] \\
@\ell & & @\ell \\[-20pt]
\text{“1”} & & \text{“2”}
\end{tikzcd}
$$

Example: Computing Canonical Pre-solution

$$
\begin{tikzcd}
r \arrow[d] \\
C \arrow[d] \\
@m \\[-20pt]
\text{“1”}
\end{tikzcd}
$$

$$
\begin{tikzcd}
r \arrow[d] \\
C \arrow[d] \\
@m \\[-20pt]
\text{“2”}
\end{tikzcd}
\qquad
\begin{tikzcd}
r \arrow[d] & & \\
C \arrow[d] & :\!- & B \arrow[d] \\
@m & & @\ell \\[-20pt]
x & & x
\end{tikzcd}
\qquad
\begin{tikzcd}
& r \arrow[dl] \arrow[dr] & \\
A \arrow[d] & & B \arrow[d] \\
@\ell & & @\ell \\[-20pt]
\text{“1”} & & \text{“2”}
\end{tikzcd}
$$

Example: Computing Canonical Pre-solution

$$\begin{tikzcd} r \arrow[d] \\ C \arrow[d] \\ @m \end{tikzcd}$$

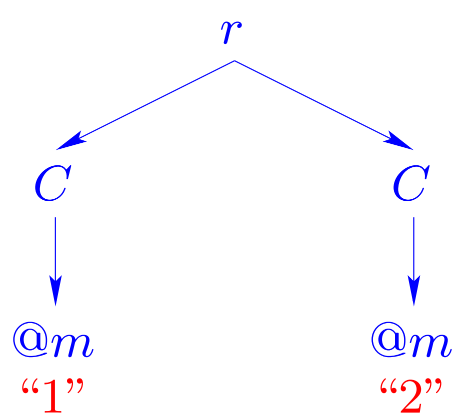
"1"

$$\begin{tikzcd} r \arrow[d] \\ C \arrow[d] \\ @m \end{tikzcd}$$

"2"

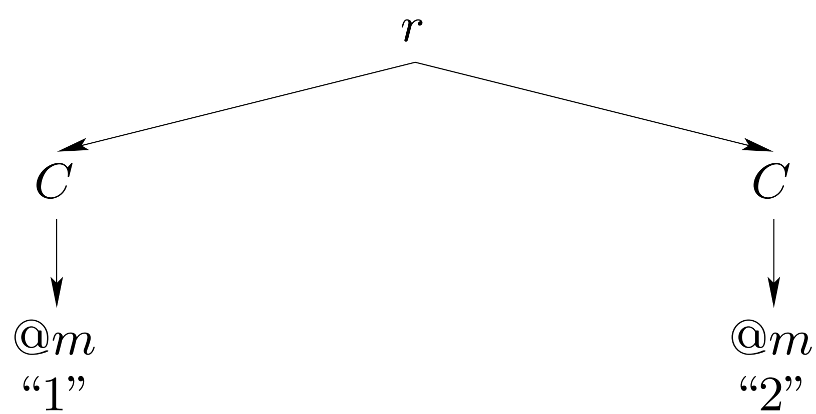
Example: Computing Canonical Pre-solution

Canonical pre-solution:



Not yet a solution: it does not conform to the target DTD.

Example: Computing Canonical Solution

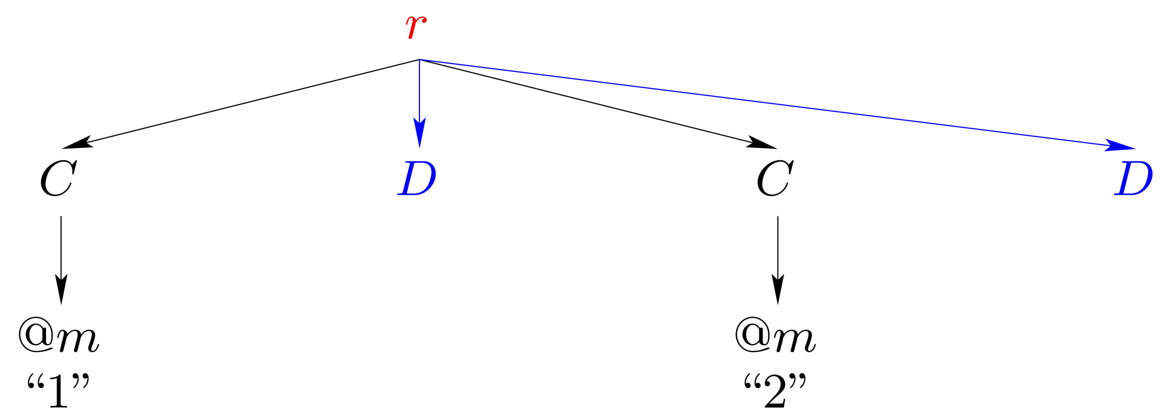


Example: Computing Canonical Solution

$$\begin{tikzcd} & r \arrow[dl] \arrow[dr] & \\ C \arrow[d] & & C \arrow[d] \\ \begin{array}{c} @m \\ \text{“1”} \end{array} & & \begin{array}{c} @m \\ \text{“2”} \end{array} \end{tikzcd}$$

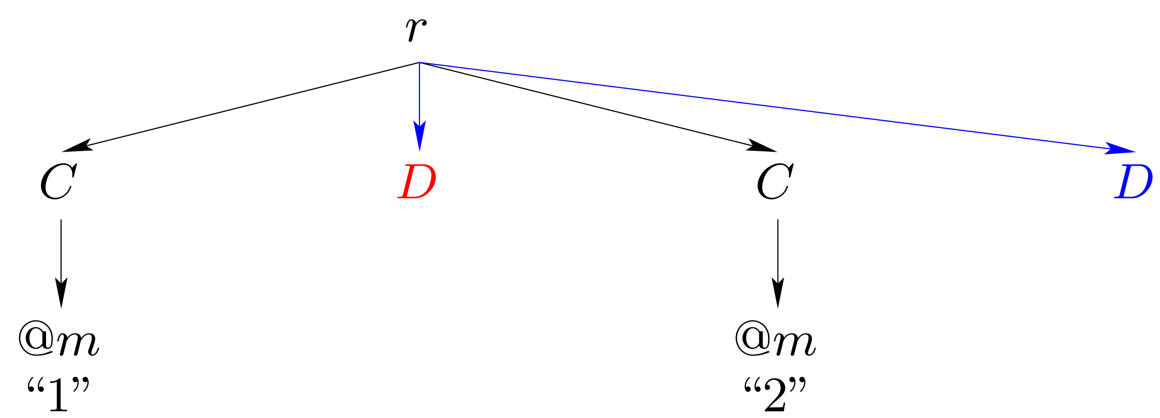
$$r \quad \rightarrow \quad (C, D)^*$$

Example: Computing Canonical Solution



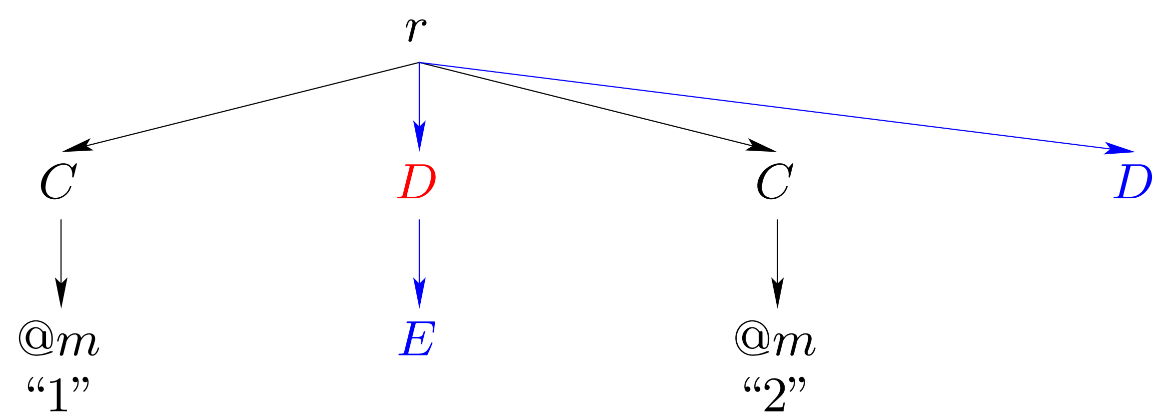
$$r \quad \rightarrow \quad (C, D)^*$$

Example: Computing Canonical Solution



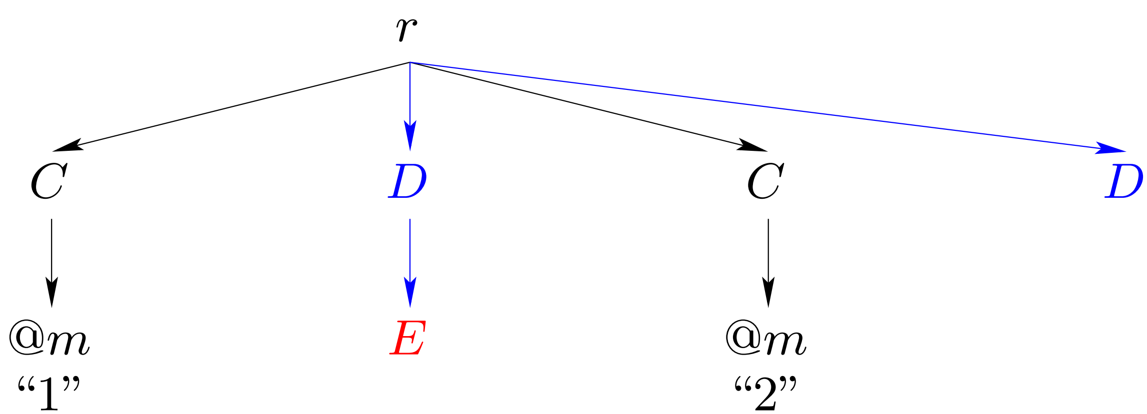
$$D \rightarrow E$$

Example: Computing Canonical Solution



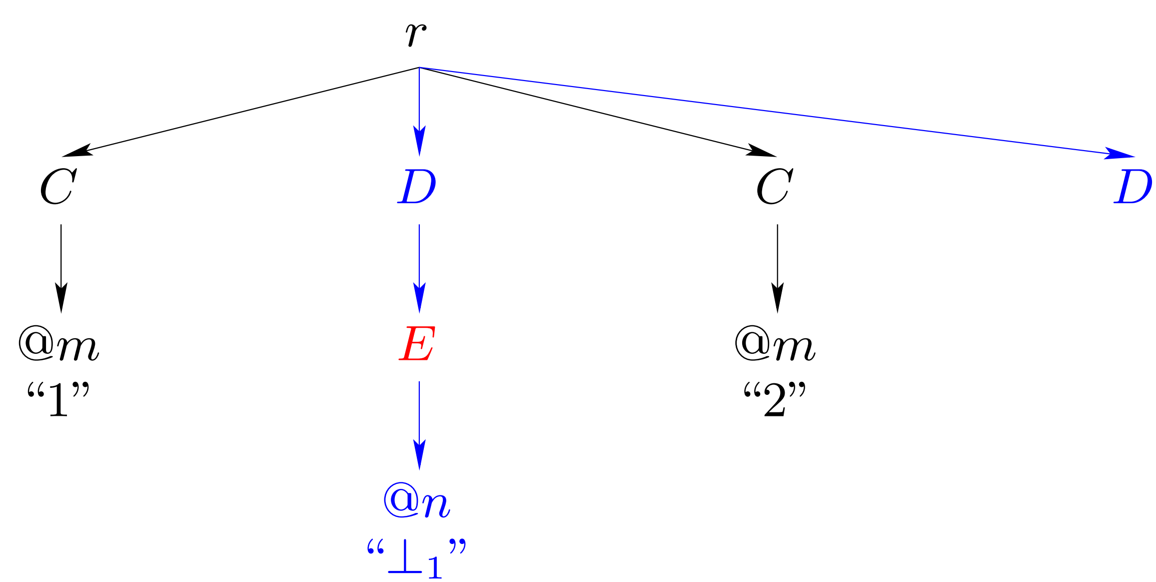
$D \quad \rightarrow \quad E$

Example: Computing Canonical Solution



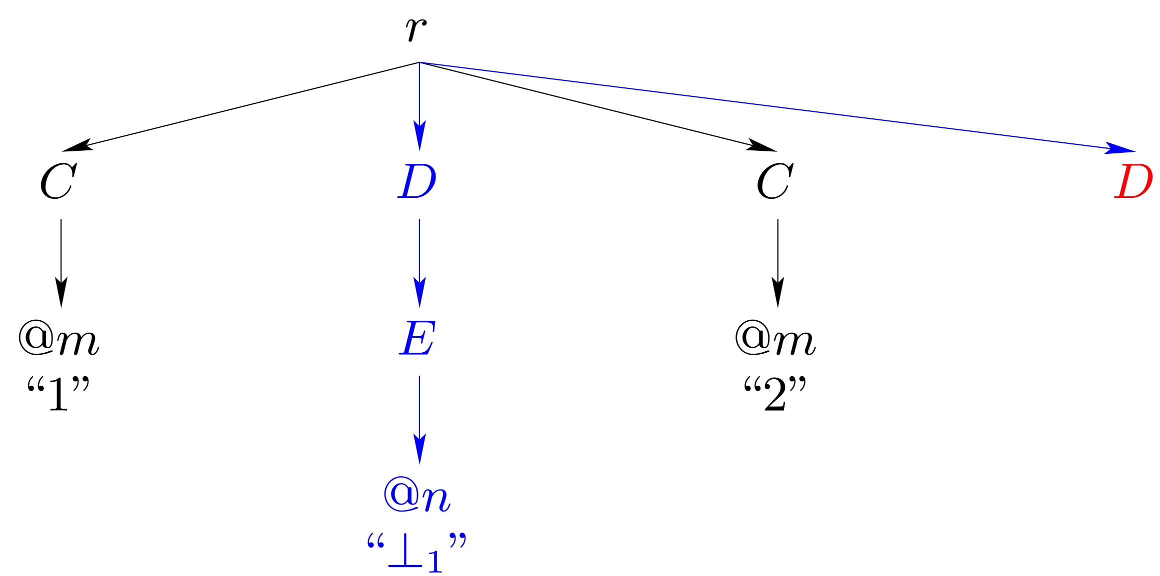
$$E \quad \rightarrow \quad @n$$

Example: Computing Canonical Solution



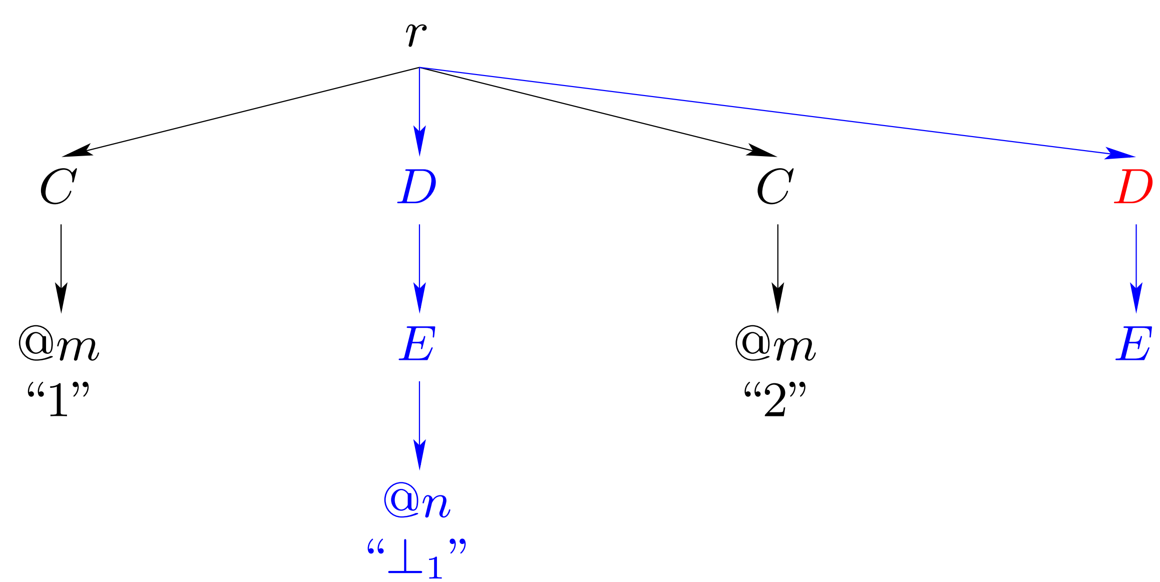
$$E \;\rightarrow\; @n$$

Example: Computing Canonical Solution



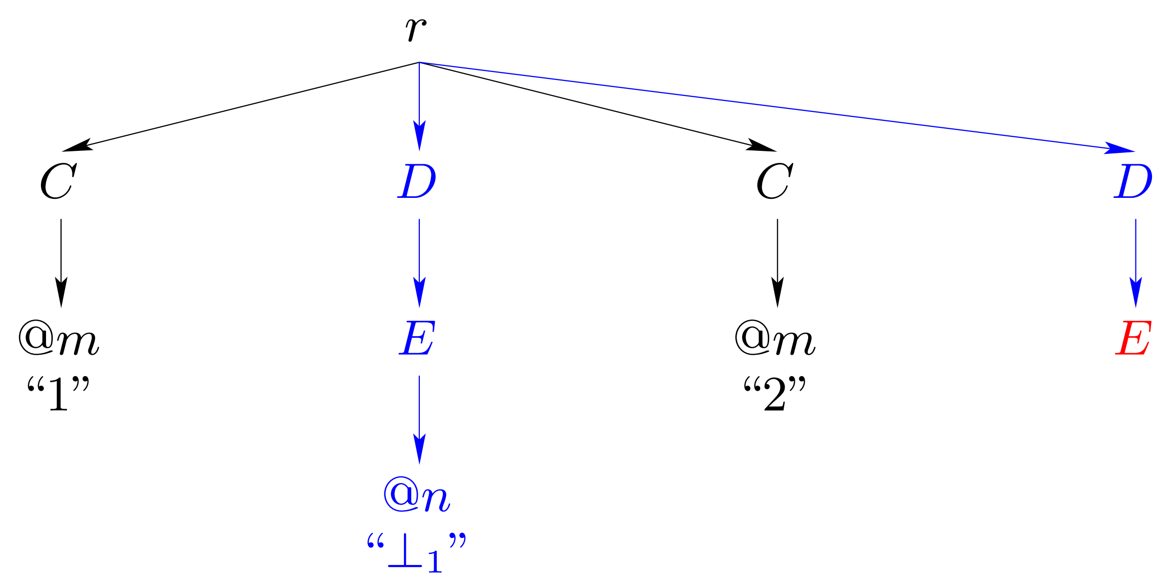
$D \rightarrow E$

Example: Computing Canonical Solution



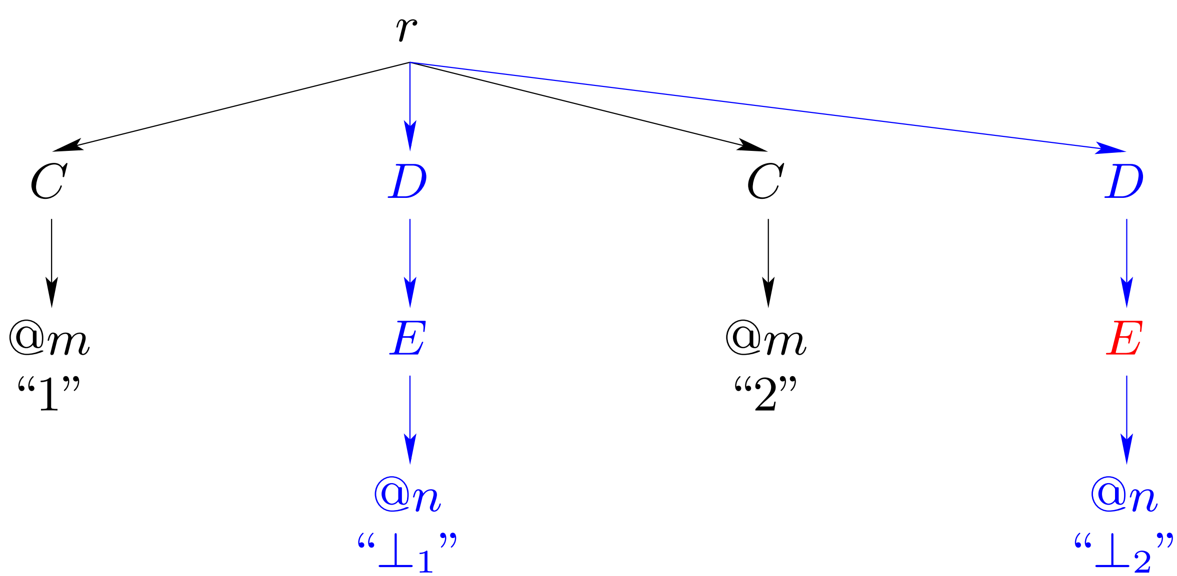
$$D \rightarrow E$$

Example: Computing Canonical Solution



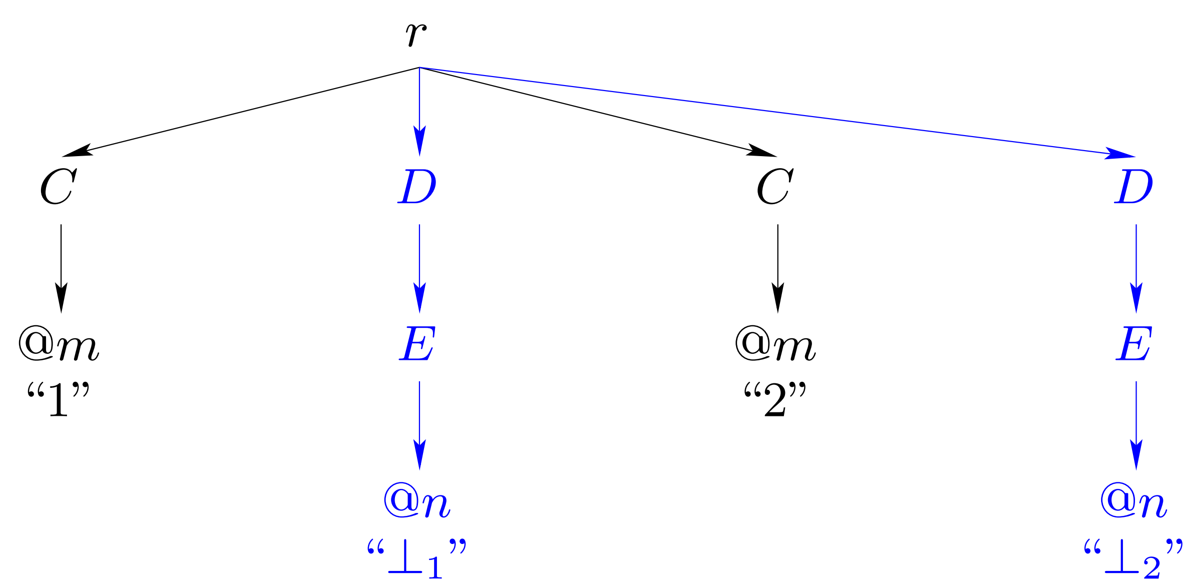
$$E \rightarrow @n$$

Example: Computing Canonical Solution



$$E \rightarrow @n$$

Example: Computing Canonical Solution



A Tractable Case: Univocal Regular Expressions

- $\mathcal{C}_U$: class of univocal regular expressions.
  - Examples: $(A|B)^*$, $A, B^+, C^*, D?$, $(A^*|B^*)$, $(C, D)^*$.
  - Non-univocal: $A, (B|C)$.

- For target DTDs only using univocal regular expressions:
  - There exists a solution for a tree T iff there exists a canonical solution $T^\star$ for T.
  - Previous algorithm computes canonical solution $T^\star$ for T in polynomial time.
  - $\underline{certain}(Q, T) = remove\_null\_tuples(Q(T^\star))$, for every $\mathcal{CTQ}^{//}$-query.

- **Theorem** $\mathcal{C}_U$ is tractable for $\mathcal{CTQ}^{//}$.

Computing Certain Answers: Non-tractable Cases

Is there any other tractable class of regular expressions?

Theorem $\mathcal{C}_U$ is the maximal tractable class: If $\mathcal{C}$ is an admissible class of regular expressions such that $\mathcal{C} \not\subseteq \mathcal{C}_U$, then $\mathcal{C}$ is coNP-complete for $\mathcal{CTQ}$-queries.

Dichotomy follows from this theorem and tractability of $\mathcal{C}_U$.

Theorem It is decidable whether a regular expression is univocal.

Future Work

- What about XML languages like XQuery that return XML documents? How do we define certain answers?

- The notion of reasonable solutions needs to be investigated further.

  We would like to consider different certain-answers semantics.