XML Data Exchange: Consistency and Query Answering

Marcelo Arenas
U. of Toronto

Leonid Libkin
U. of Toronto

The Problem of Data Exchange

- Given: A source schema S, a target schema T and a specification Σ of the relationship between these schemas.

- Data exchange: Problem of finding an instance of T, given an instance of S.

  - Target instance should reflect the source data as accurately as possible, given the constraints imposed by Σ and T.

  - It should be efficiently computable.

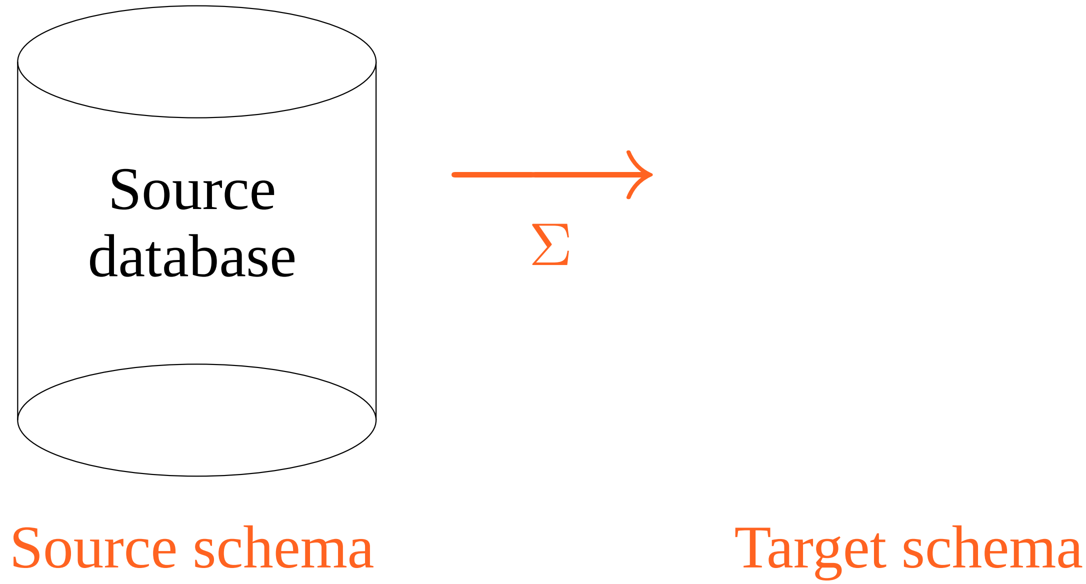
  - It should allow one to evaluate queries on the target in a way that is semantically consistent with the source data.

Data Exchange

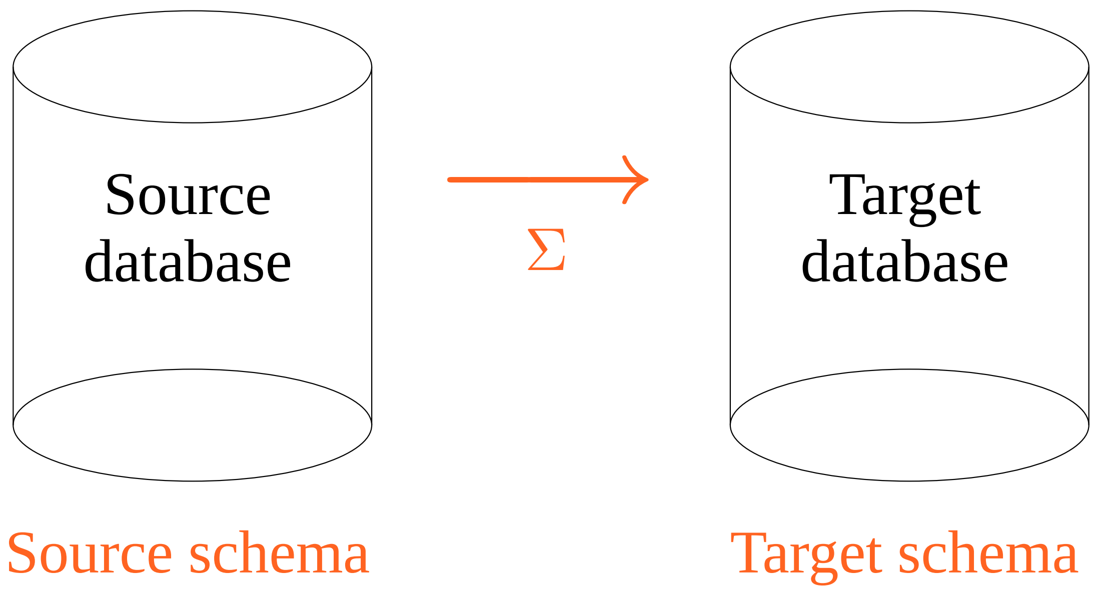
$$\xrightarrow[\Sigma]{}$$

Source schema Target schema

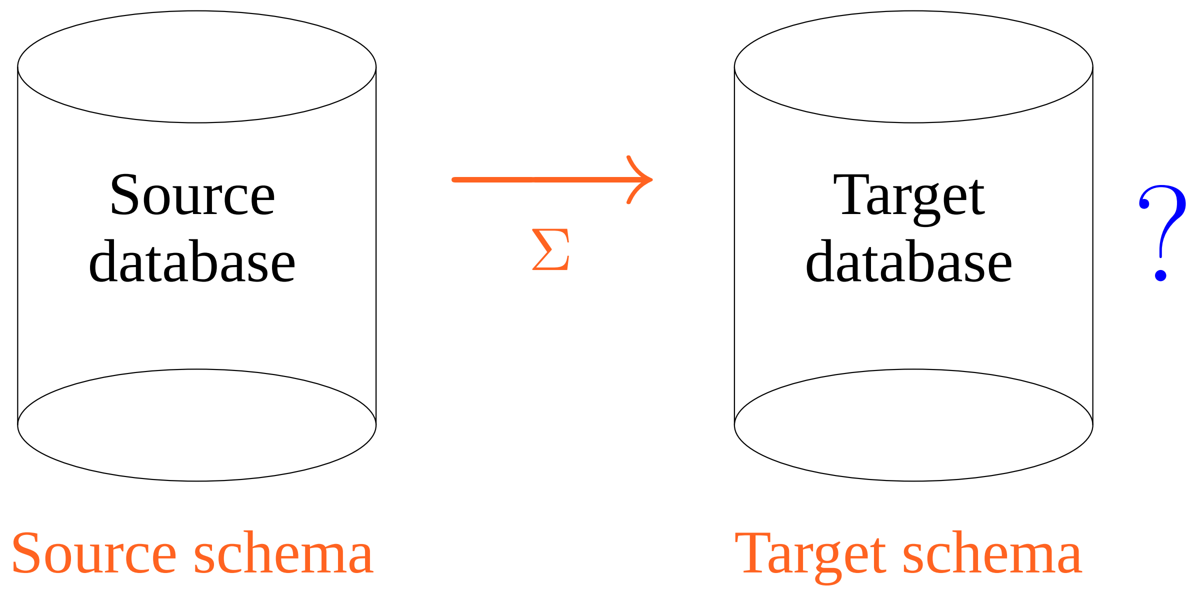
Data Exchange



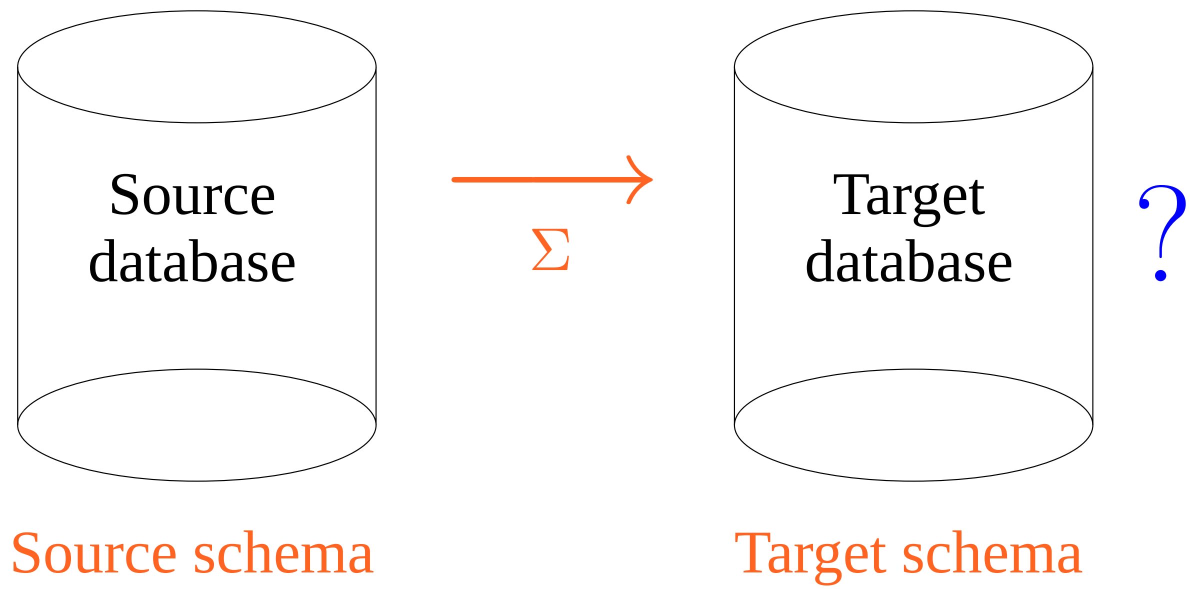
Data Exchange



Data Exchange



Data Exchange



Query over the target: Q

Answer to Q in the target instance should represent the answer to Q in the space of possible translations of the source instance.

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.

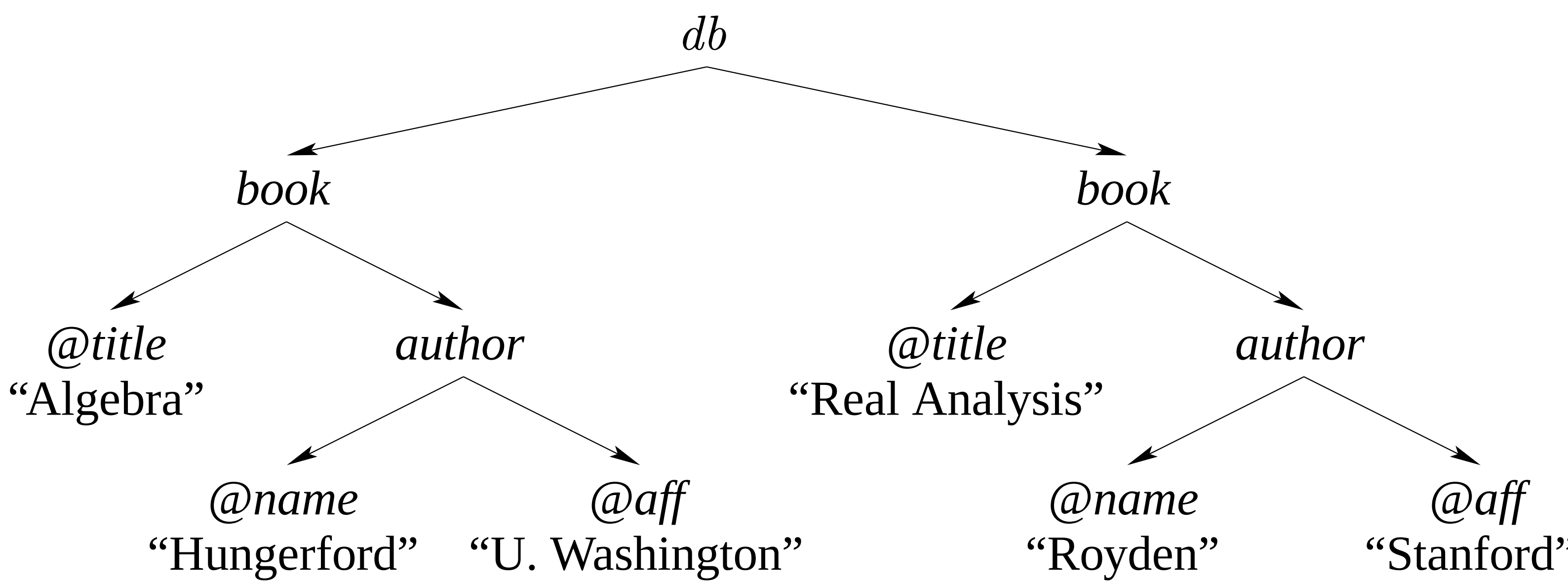
Outline

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

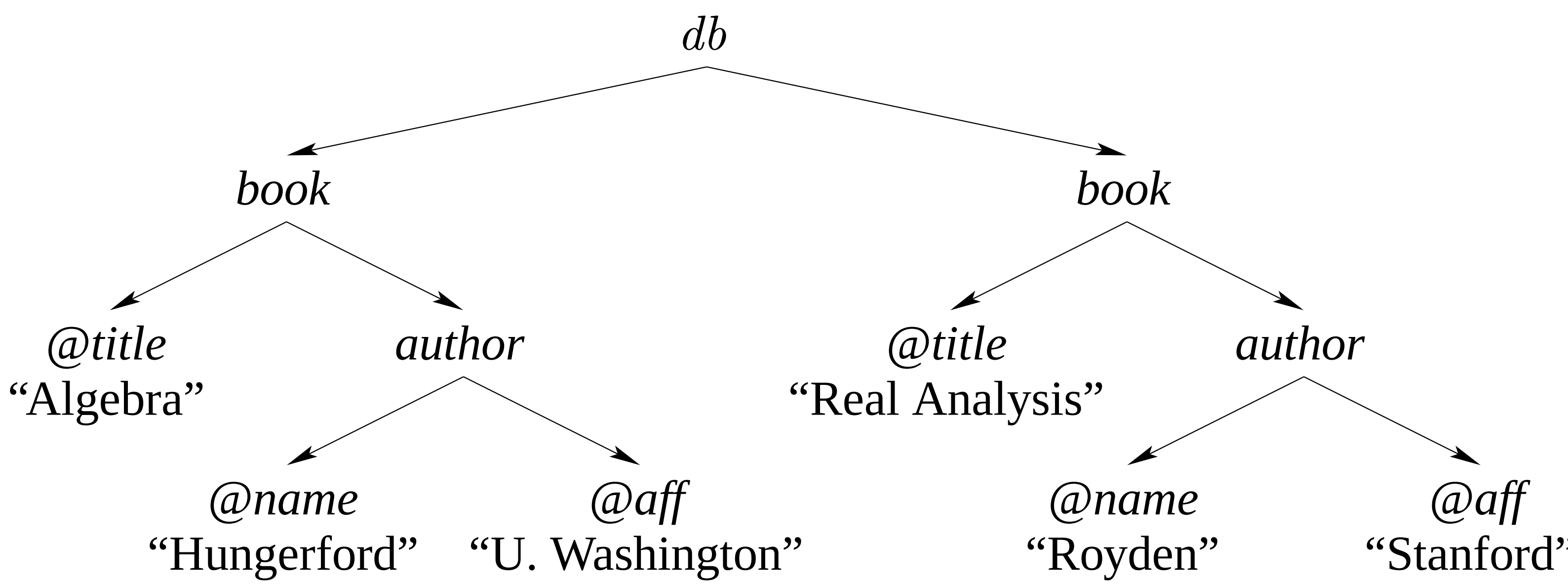
Outline

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

XML Documents



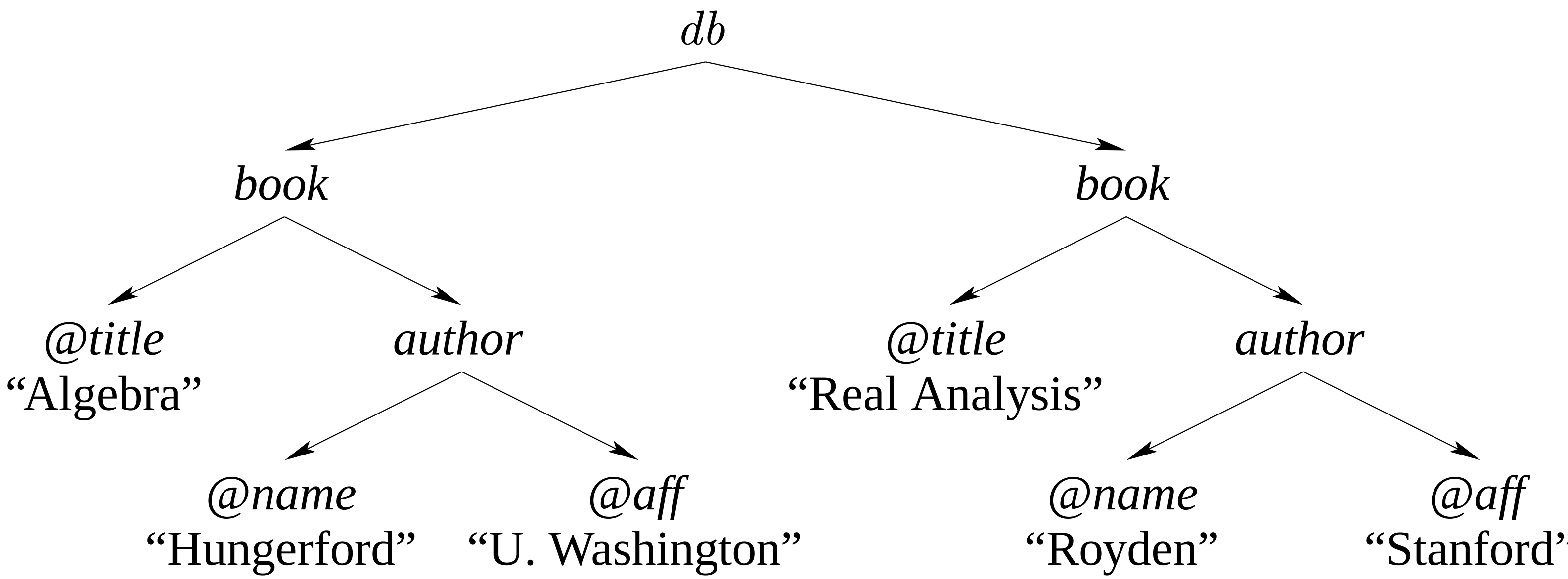
XML Documents



DTD :

$$\begin{aligned} db &\rightarrow book^{+} \\ book &\rightarrow author^{+} \\ author &\rightarrow \varepsilon \end{aligned}$$

XML Documents



DTD :

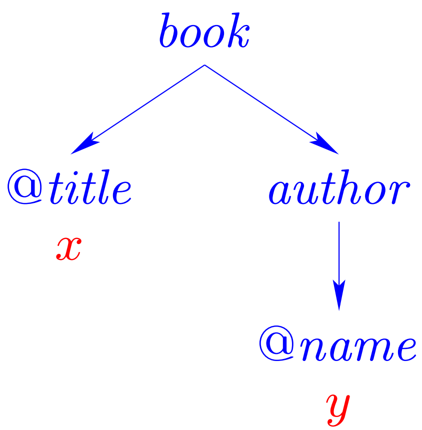
$$\begin{array}{rcl} db & \to & book^+ \\ book & \to & author^+ \\ author & \to & \varepsilon \end{array} \qquad \begin{array}{rcl} book & \to & @title \\ author & \to & @name, @aff \end{array}$$

XML Data Exchange Settings

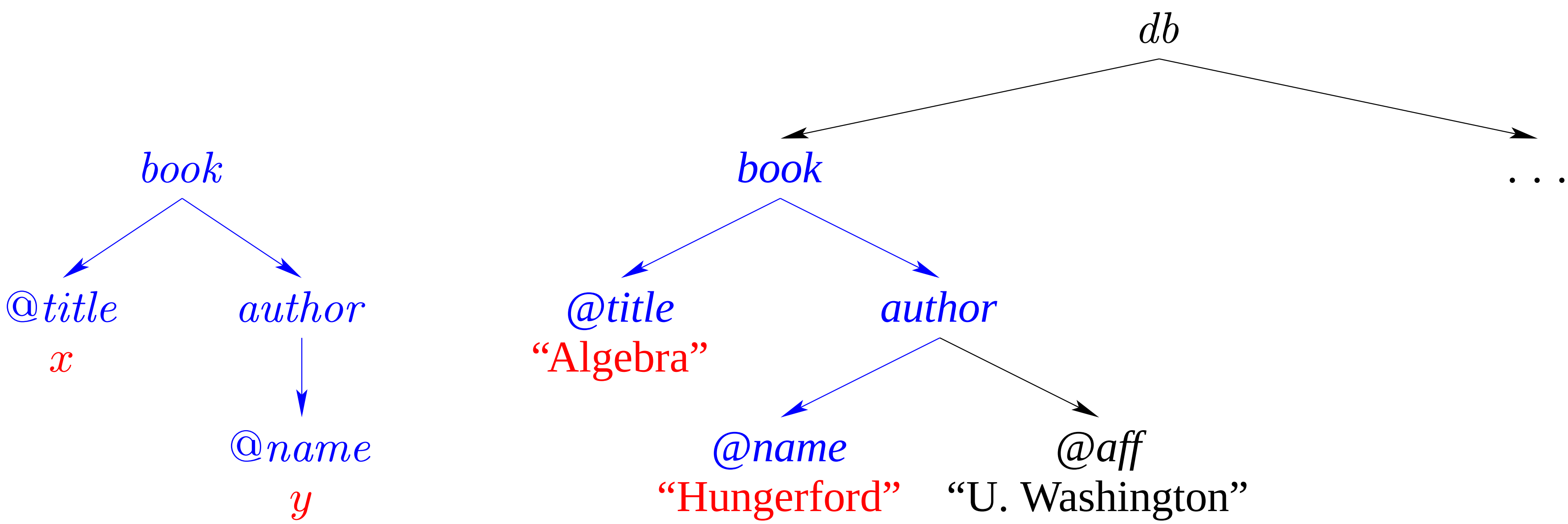
- Source and target schemas are given by DTDs.
- To specify the relationship between the source and the target schemas we use source-to-target dependencies.

To define these dependencies, we use tree patterns ...

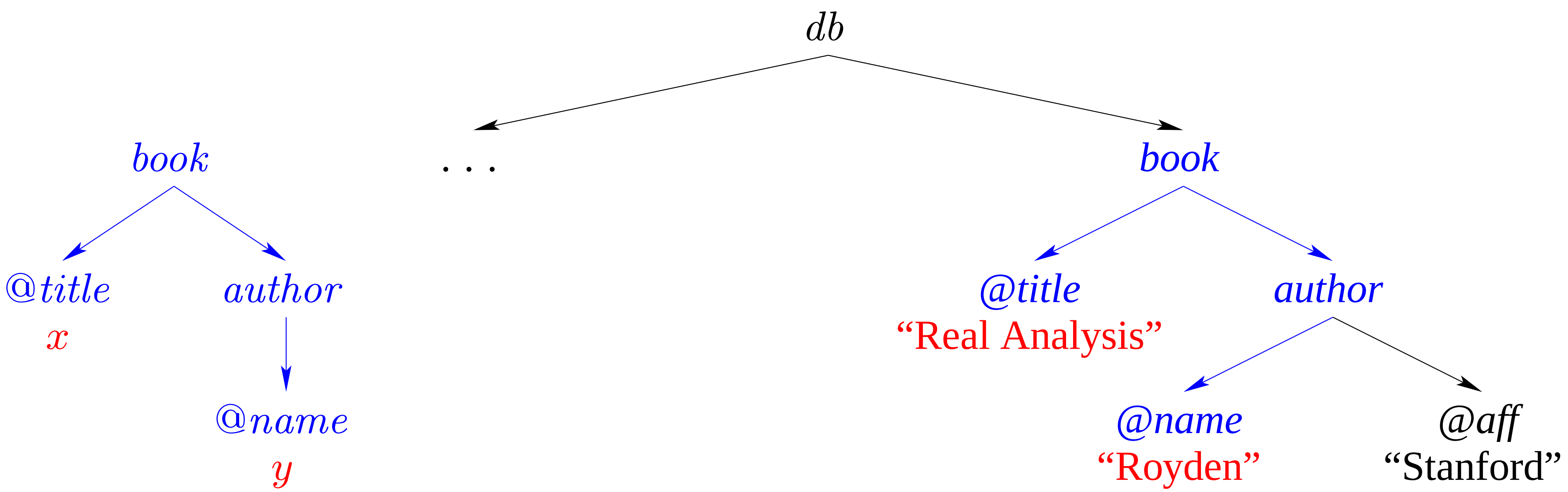
Tree Patterns: Example



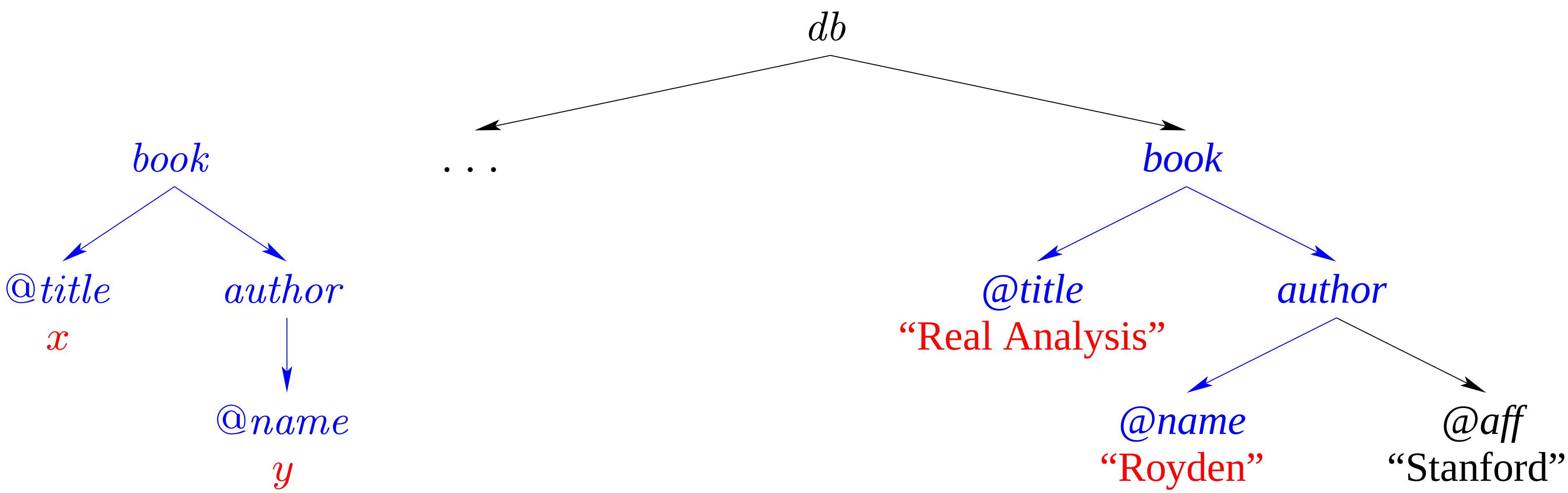
Tree Patterns: Example



Tree Patterns: Example



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 $//$.

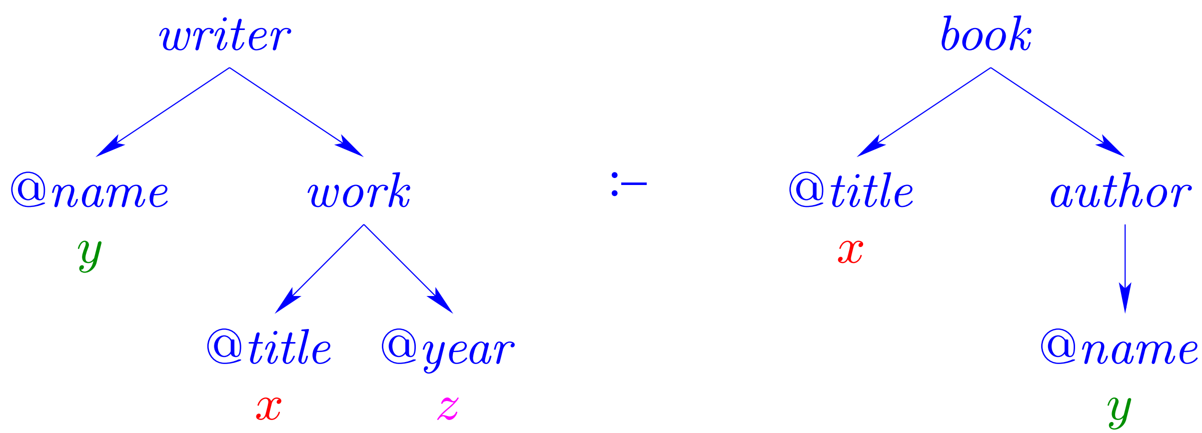
XML Source-to-target Dependencies

- Source-to-target dependency (STD):

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

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 $D_{\mathbf{T}}$.

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.

Example: Finding Solutions

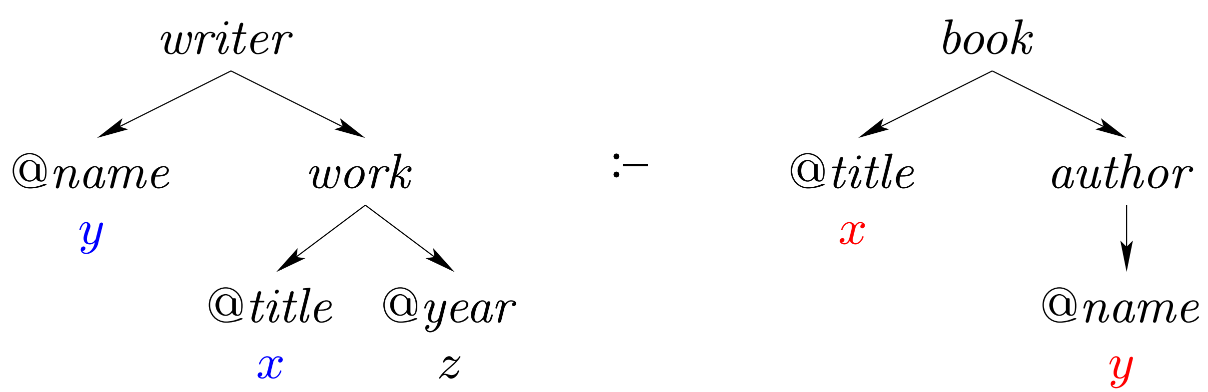
Source DTD:

db	$\rightarrow$	$book^+$			
$book$	$\rightarrow$	$author^+$	$book$	$\rightarrow$	$@title$
$author$	$\rightarrow$	ε	$author$	$\rightarrow$	$@name, @aff$

Target DTD:

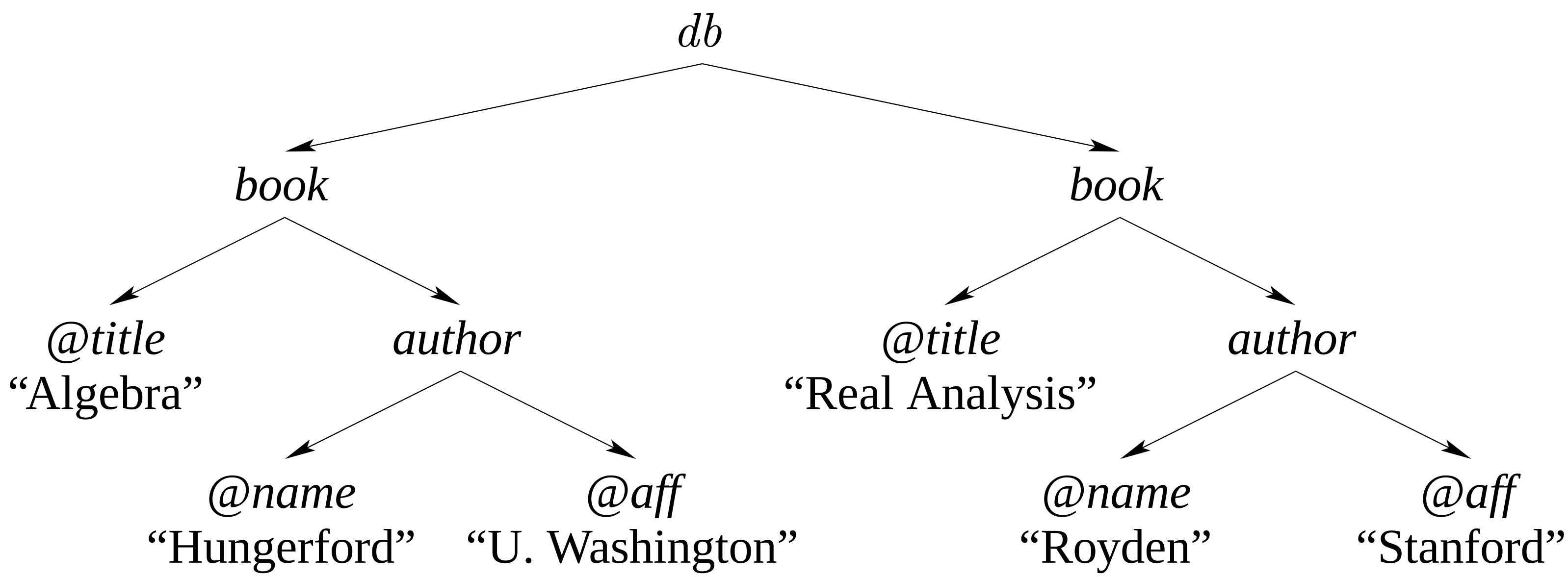
bib	$\rightarrow$	$writer^+$			
$writer$	$\rightarrow$	$work^+$	$writer$	$\rightarrow$	$@name$
$work$	$\rightarrow$	ε	$work$	$\rightarrow$	$@title, @year$

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



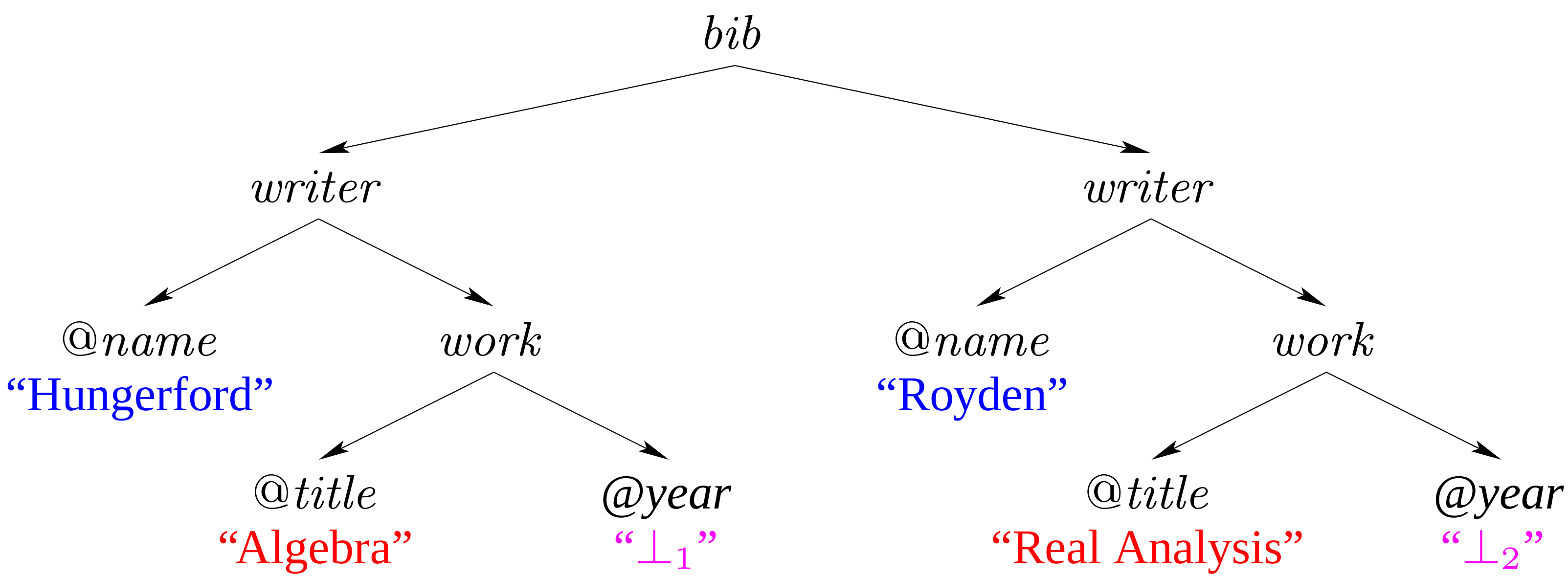
Example: Finding Solutions

Let T be our original tree:



Example: Finding Solutions

A solution for T:



Outline

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

Consistency of XML Data Exchange Settings

- An XML data exchange setting $(D_{\mathbf{S}}, D_{\mathbf{T}}, \Sigma_{\mathbf{ST}})$ can be inconsistent:

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

- 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.

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 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 $\ell?$

Subsume non-relational data exchange handled by Clio.

Theorem For non-recursive DTDs that only have these rules, consistency can be checked in time $O\big((\|D_{\mathbf{S}}\| + \|D_{\mathbf{T}}\|) \cdot \|\Sigma_{\mathbf{ST}}\|^2\big)$.

Outline

- XML data exchange settings.
  - XML source-to-target dependencies.
- Consistency of XML data exchange settings.
- 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 $\mathcal{CTQ}^{//}$ is defined by

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

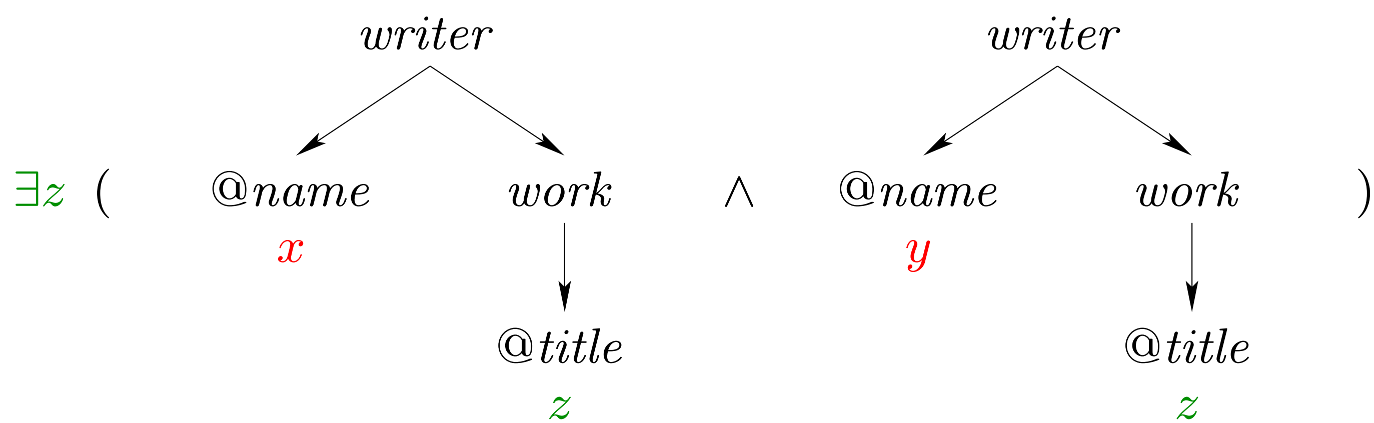
where φ ranges over tree-pattern formulas.

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

Example: Conjunctive Tree Query

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

$Q(x, y) \;:=$



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:

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

Computing Certain Answers

We study the following problem.

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

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

Computing Certain Answers: General Picture

Theorem For every XML data exchange setting and $\mathcal{CTQ}^{//}$-query Q, CERTAIN-ANSWERS(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 CERTAIN-ANSWERS(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 CERTAIN-ANSWERS(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, CERTAIN-ANSWERS(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 CERTAIN-ANSWERS(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 target 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 $\mathcal{CTQ}^{//}$.

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

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

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

A Tractable Class: Univocal Regular Expressions

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

- Univocal regular expressions: Given a source tree T, one can compute in PTIME a solution $T^\star$ for T such that

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

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

Non-tractable Classes

Is there any other tractable class of regular expressions?

Theorem $\mathcal{C}_U$ is maximal: 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.

Outline

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

Tractable Case: Univocal Regular Expressions

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

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

- 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:

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

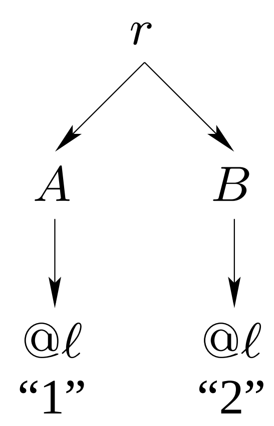
- Target DTD:

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

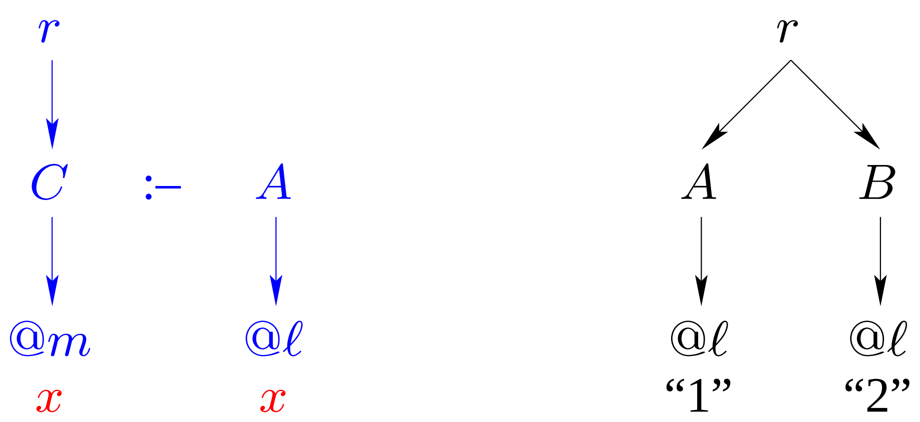
- $\Sigma_{\mathbf{ST}}$:

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

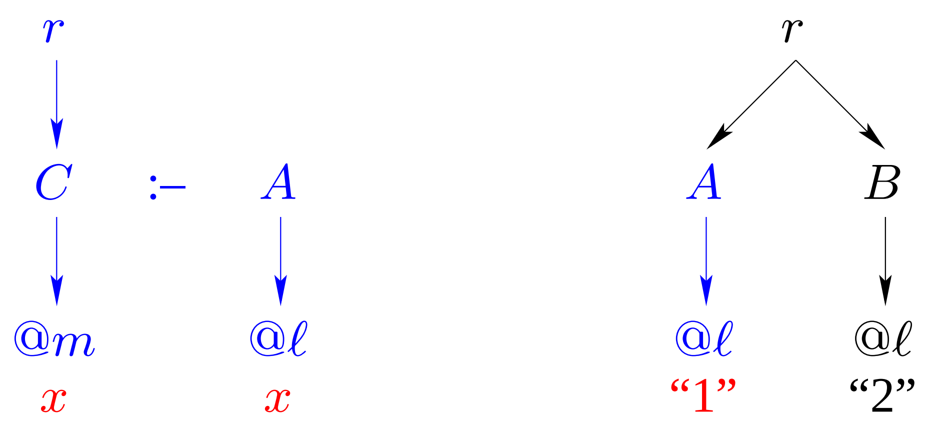
Example: Computing Canonical Pre-solution



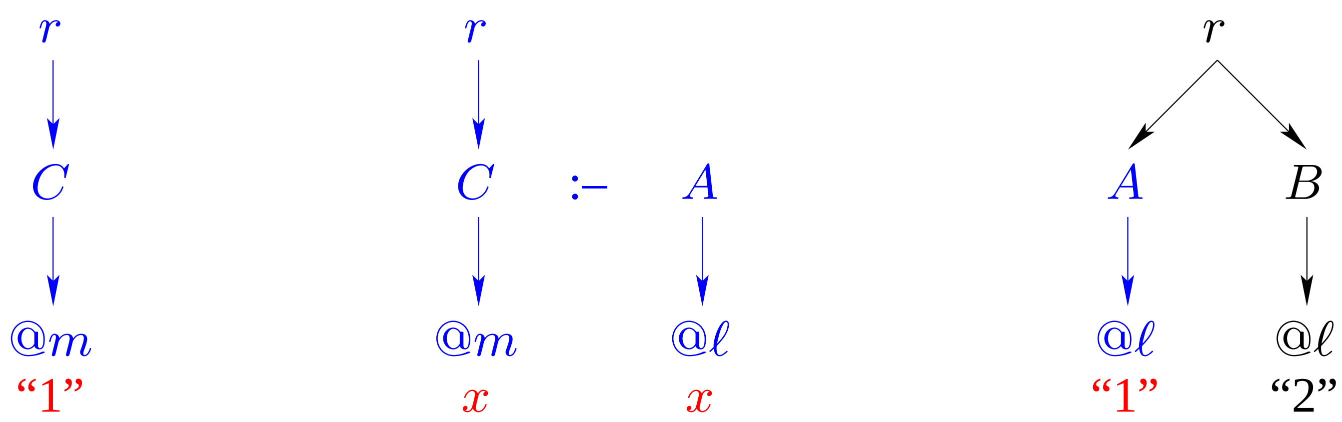
Example: Computing Canonical Pre-solution



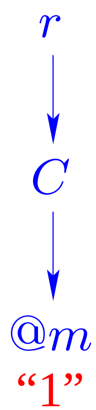
Example: Computing Canonical Pre-solution



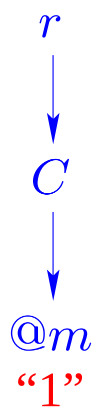
Example: Computing Canonical Pre-solution

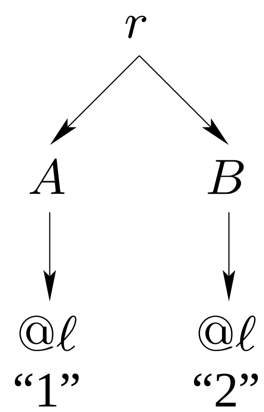


Example: Computing Canonical Pre-solution

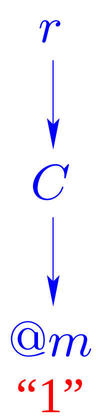


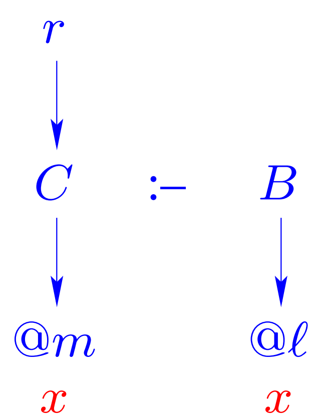
Example: Computing Canonical Pre-solution

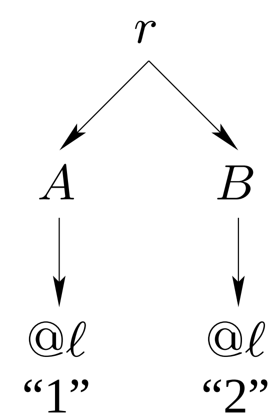




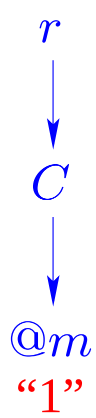
Example: Computing Canonical Pre-solution

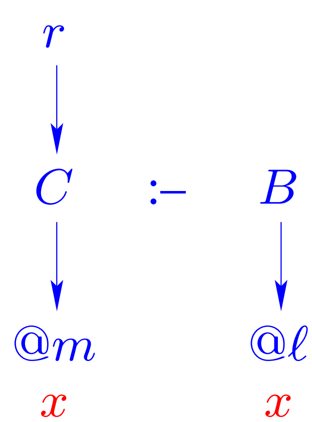


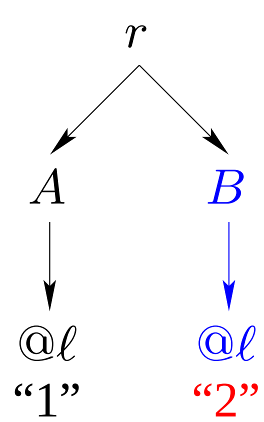




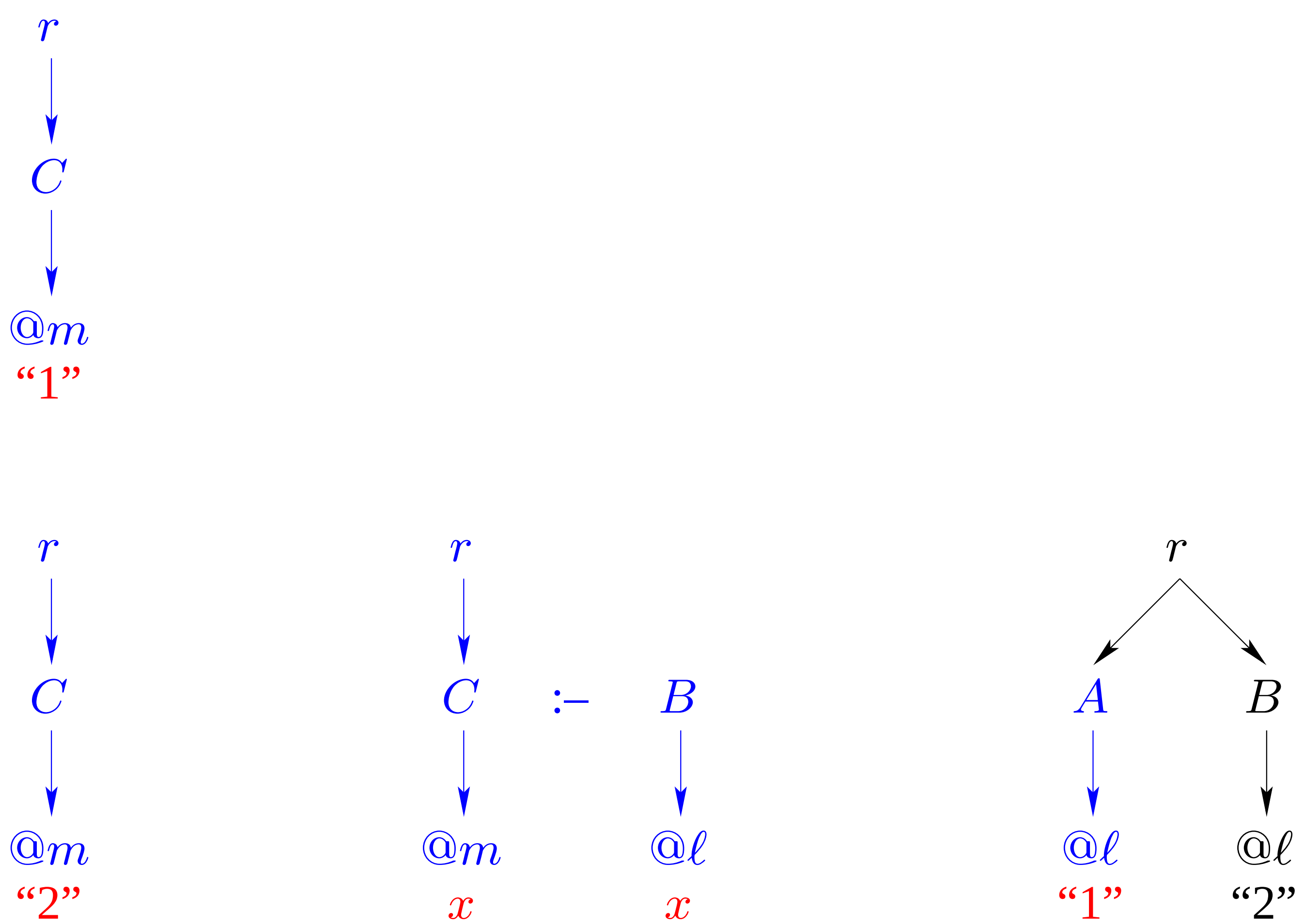
Example: Computing Canonical Pre-solution







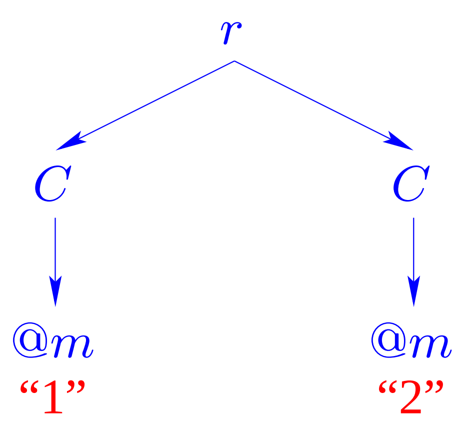
Example: Computing Canonical Pre-solution



Example: Computing Canonical Pre-solution

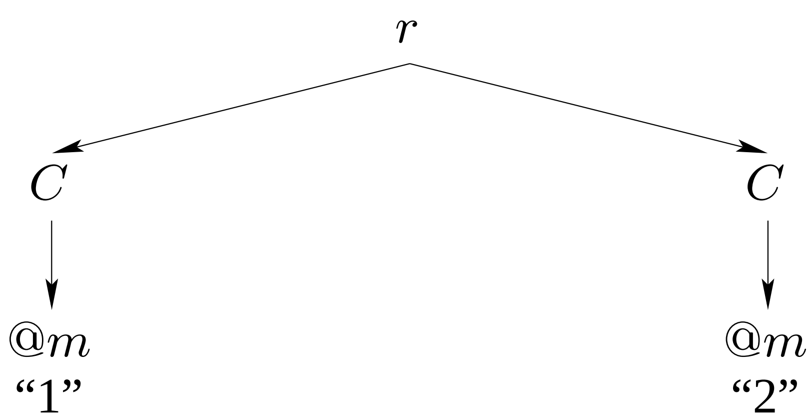
Example: Computing Canonical Pre-solution

Canonical pre-solution:

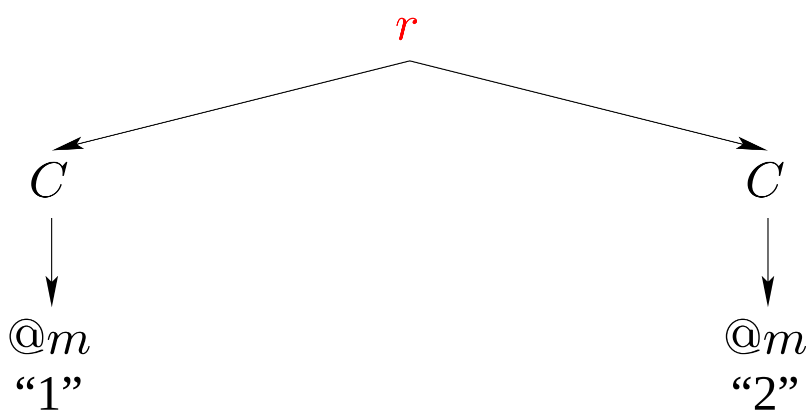


Not yet a solution: It does not conform to the target DTD.

Example: Computing Canonical Solution

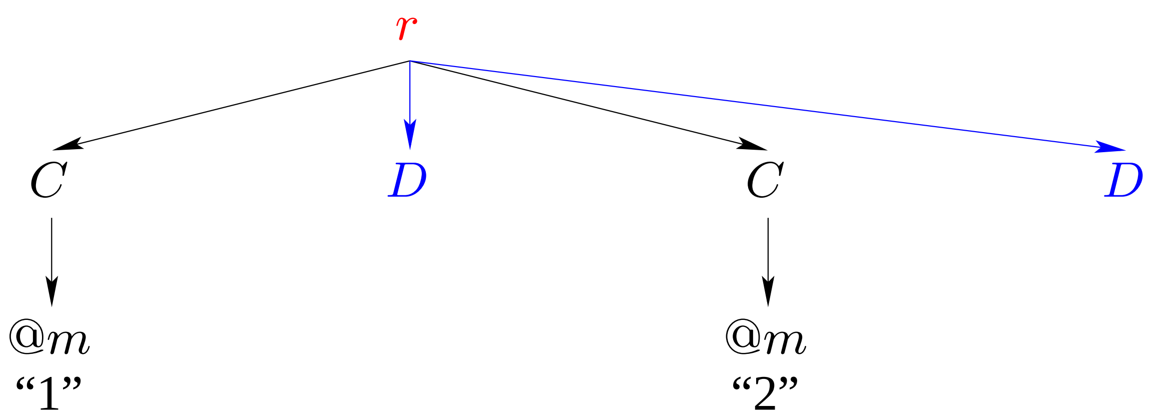


Example: Computing Canonical Solution



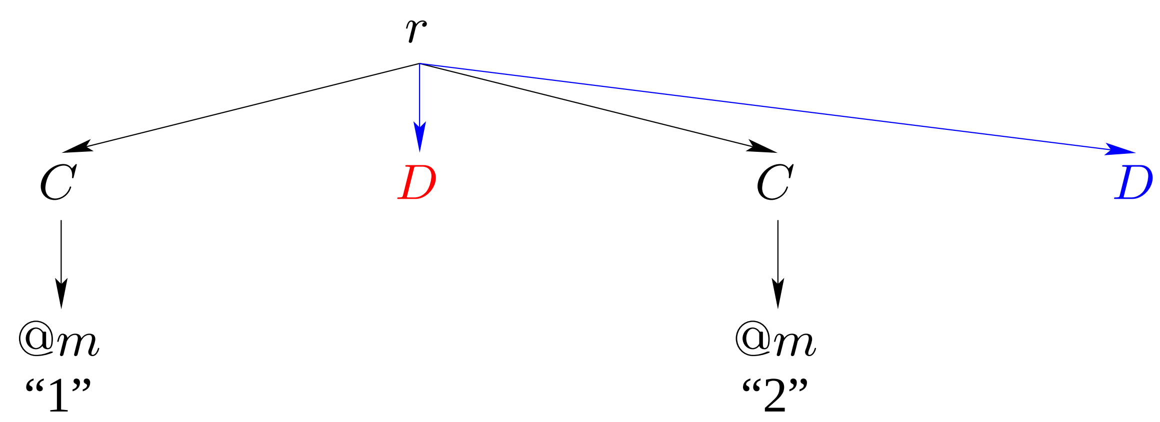
$$r \;\rightarrow\; (C, D)^*$$

Example: Computing Canonical Solution



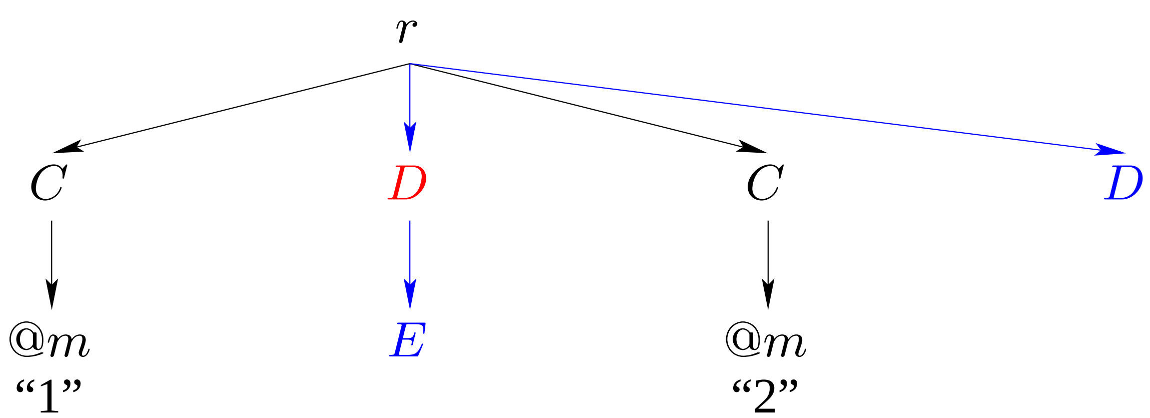
$$r \quad \longrightarrow \quad (C, D)^*$$

Example: Computing Canonical Solution



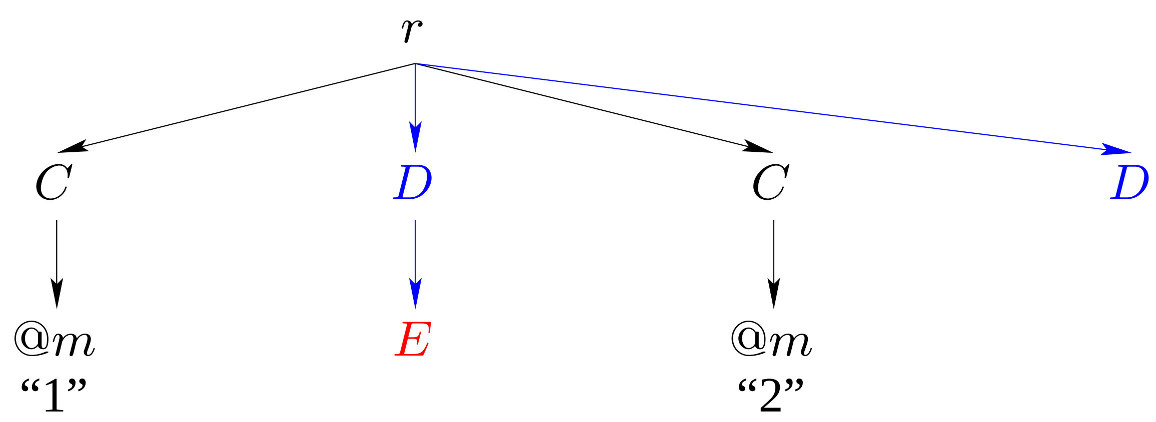
$$D \quad \longrightarrow \quad E$$

Example: Computing Canonical Solution



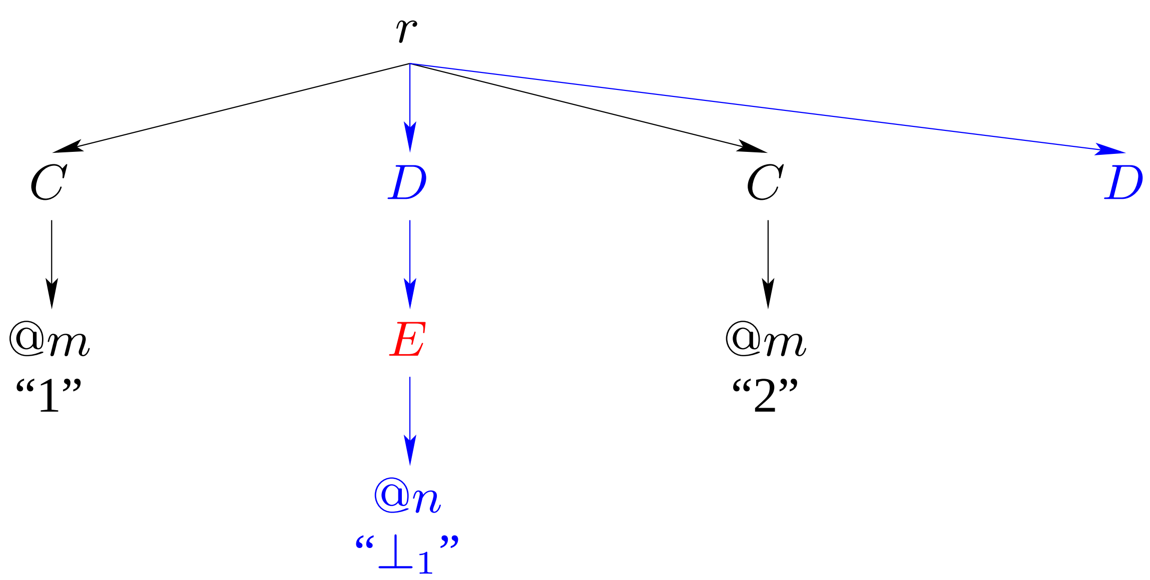
$$D \rightarrow E$$

Example: Computing Canonical Solution



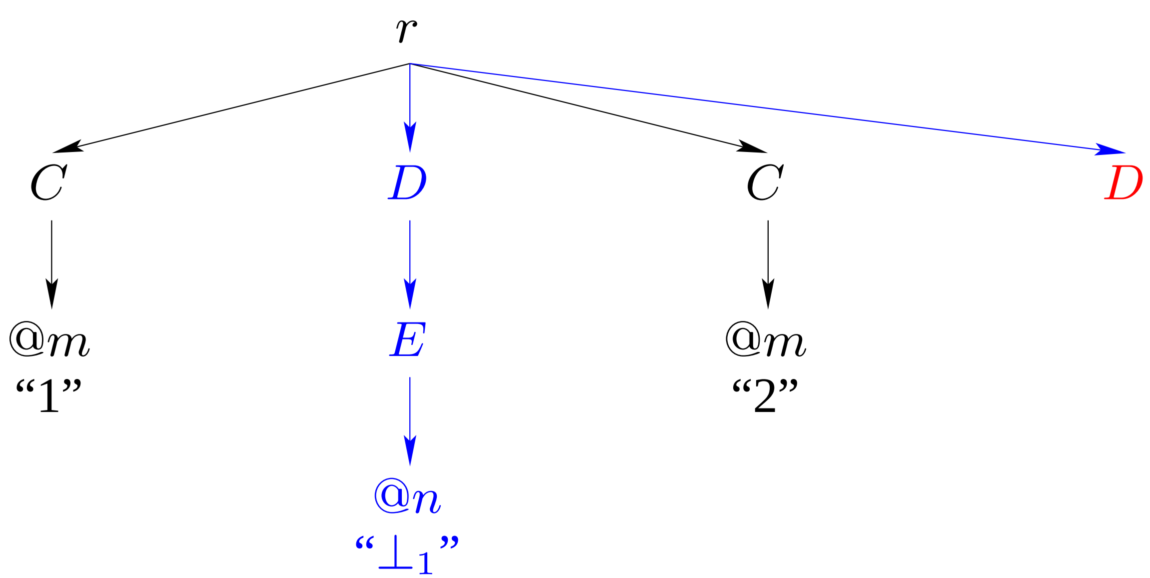
$$E \rightarrow @n$$

Example: Computing Canonical Solution



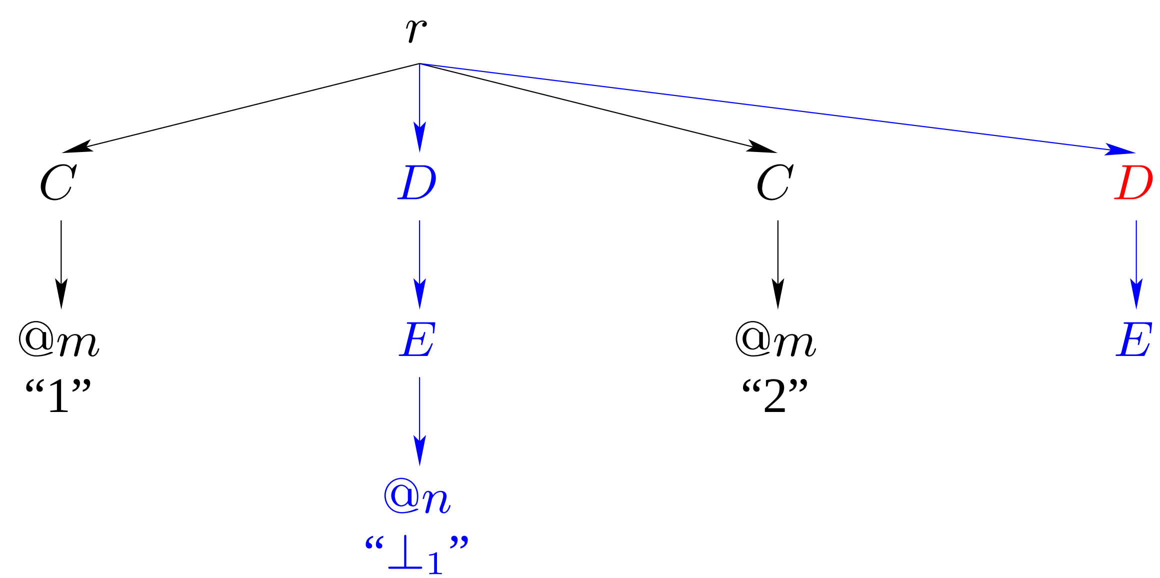
$$E \rightarrow @n$$

Example: Computing Canonical Solution



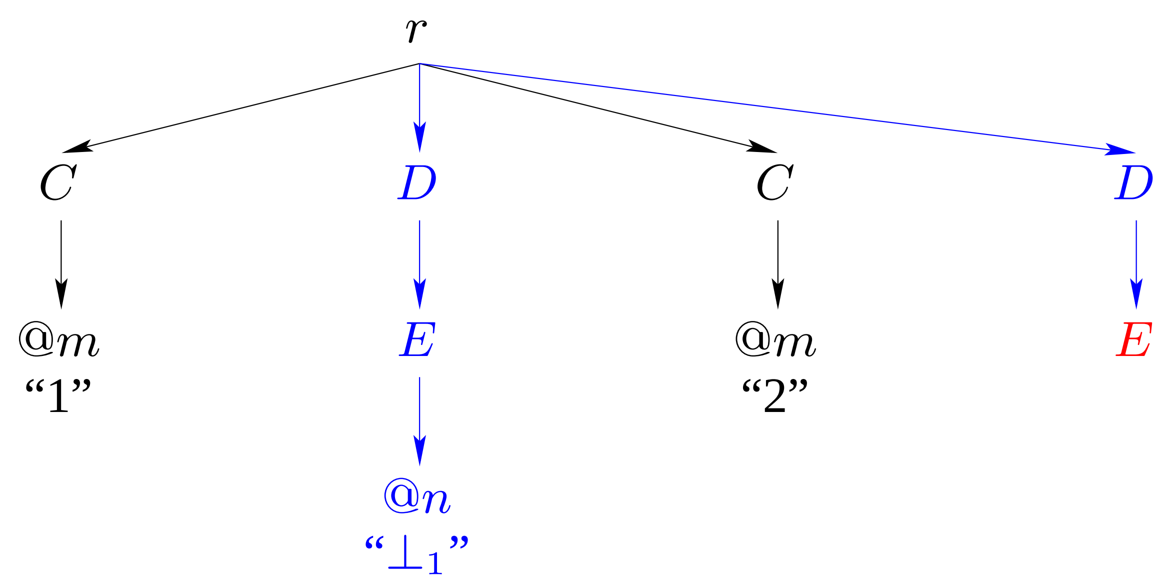
$$D \rightarrow E$$

Example: Computing Canonical Solution



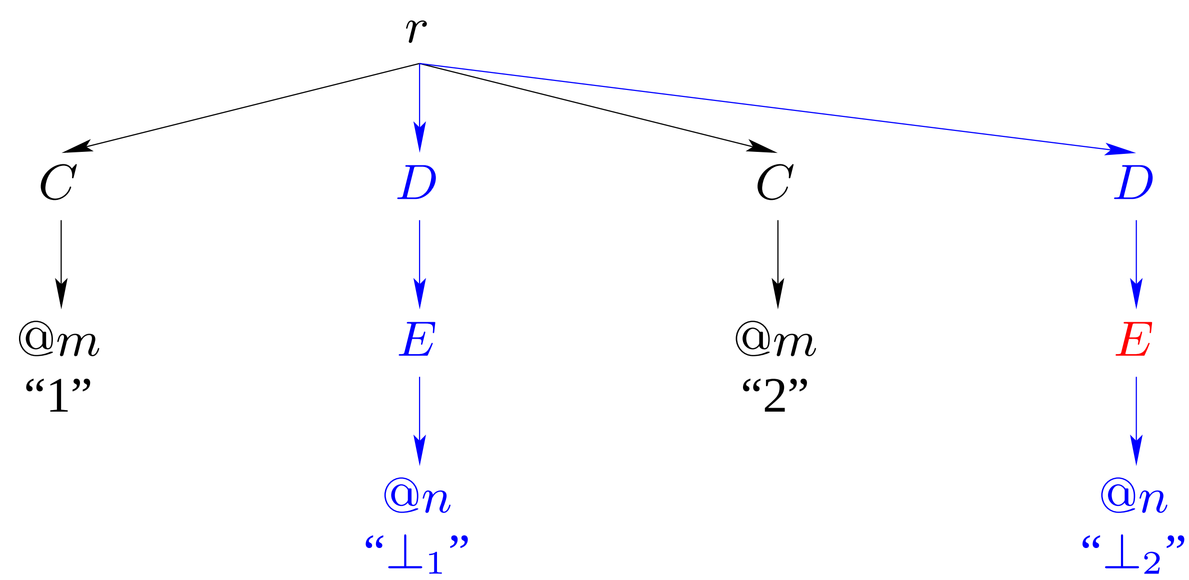
$$D \rightarrow E$$

Example: Computing Canonical Solution



$$E \longrightarrow @n$$

Example: Computing Canonical Solution



$$E \;\; \longrightarrow \;\; @n$$

Example: Computing Canonical Solution