XML Data Integration

Łucja Kot

Cornell University

11 November 2010

Data Integration and Query Answering

A **data integration system** is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ where

- $\mathcal{G}$ is the **global schema**
- $\mathcal{S}$ is the **source schema**
- $\mathcal{M}$ is a set of assertions relating elements of the source schema and elements of the global schema

Key issue in data integration: **query answering**

- given query on global schema, want to answer using source data

Data Integration and Query Answering (2)

Challenge: there may be more than one way to map source data to target schema

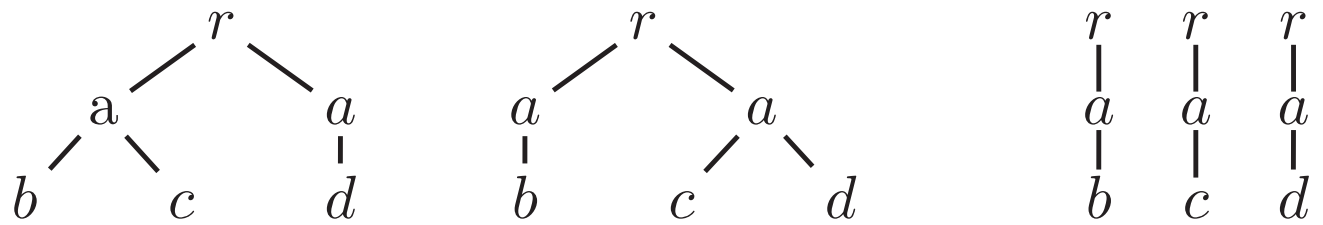
Solution: certain answers semantics for queries

- include only those tuples that always appear as answers
- first developed for databases with incomplete information
- now widely used in data integration and data exchange
  - source instance + schemas + mappings = incomplete description of target instance...

Moving to XML

How do we do data integration in XML?

- what does the setting look like, formally?
- given that some queries can return trees, what do "certain answers" look like?



This talk's focus: query answering problem as we move to XML

Talk outline

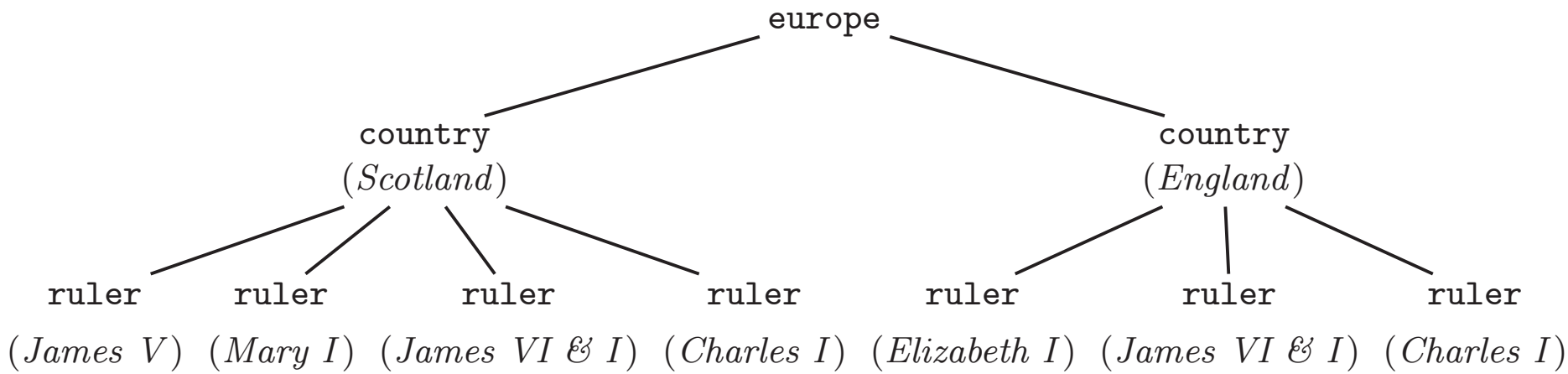
1. "Warm-up": representing incomplete information in XML
   - gets us thinking in XML
   - introduces interesting issues in XML query answering
2. A study of query answering complexity in XML in the presence of schema mappings
   - tradeoff between complexity of mapping and query languages
3. Certain answers for queries that return trees

(Re)introducing XML

While the details of formalisms differ, XML data has the following key features:

- tree structure
- nodes have labels
- nodes have attributes
- attributes have values
- nodes may have ids
- document order

An example XML document



Schema information

Can have schema for XML documents

- specifies tree structure and other related things
- XML Schema, DTD

Example DTD:

$$\begin{aligned} \texttt{europe} &\rightarrow \texttt{country}^* \\ \texttt{country} &\rightarrow \texttt{ruler}^* \\ \texttt{ruler} &\rightarrow \varepsilon \\ \texttt{country} &: \texttt{@name} \\ \texttt{ruler} &: \texttt{@name} \end{aligned}$$

Incomplete information

How do we represent incomplete information in XML?

Relational case: tables with null values

- **Codd tables:** all nulls distinct
- **naïve or v-tables:** repeated nulls (variables) permitted
- **c-tables:** constraints on variables permitted

A representation t corresponds to a **set** of complete (ground) instances $Rep(t)$

Interesting questions about incomplete data representations

Interesting problems:

- **Consistency:** given a representation t, does $Rep(t) \neq \emptyset$?
- **Membership:** given an instance T and a representation t, is $T \in Rep(t)$?
- **Query answering:** given a representation t and a query q, what are the **certain answers** to q over t?
  - that is, what is $\bigcap_{T \in Rep(t)} q(T)$?
- **Strong representation systems:** is it the case that for each q and t,there exists a computable representation u such that $Rep(u) = \{q(T) | T \in Rep(t)\}$?

Incomplete Information in XML

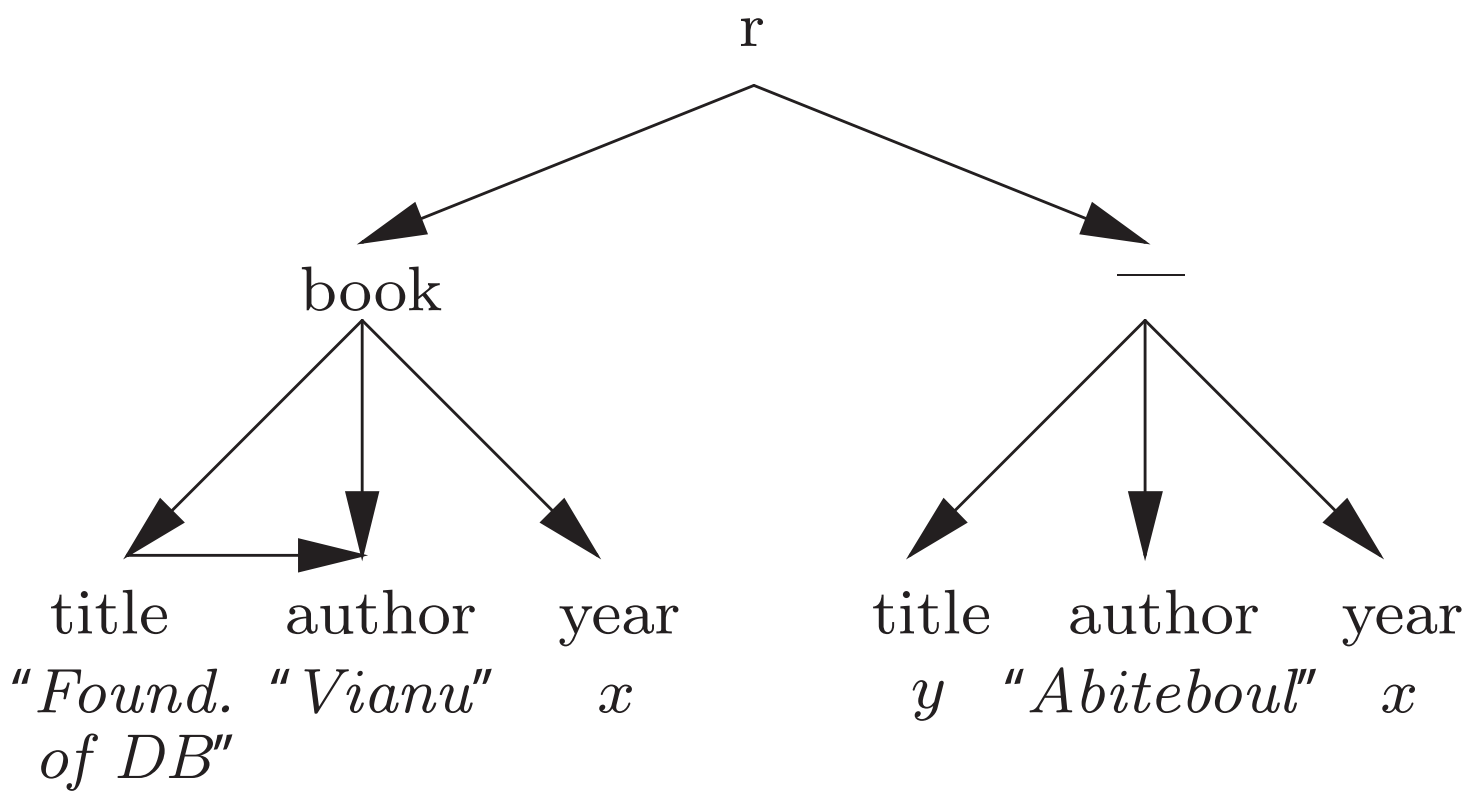
P. Barcelo, L. Libkin, A. Poggi, and C. Sirangelo. **XML with incomplete information: models, properties, and query answering**. PODS 2009.

- an in-depth study of various incomplete information models for XML

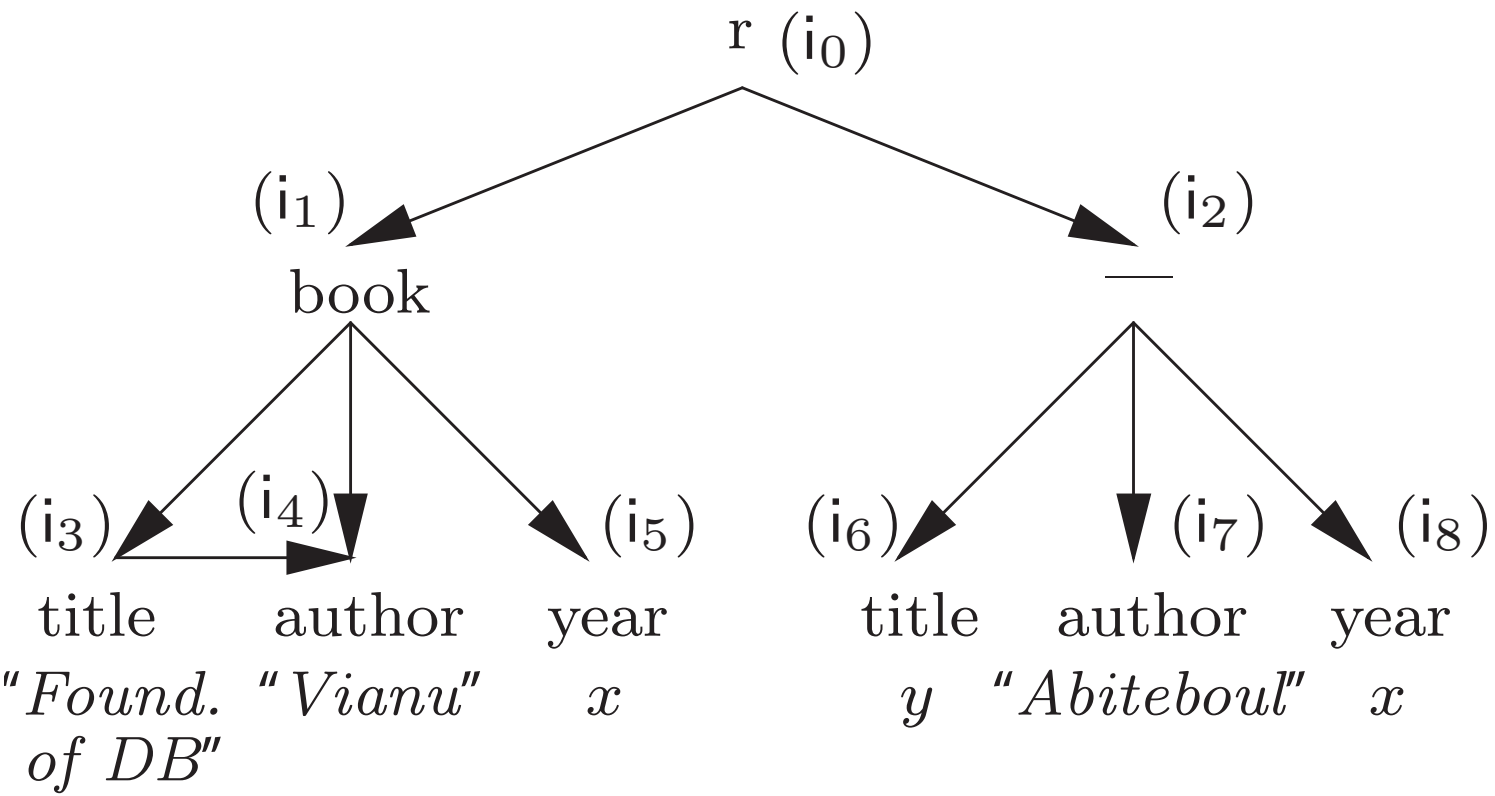
In XML, incompleteness can be structural as well as value-related

- may only know that one node is a descendant of another, not that it is a grandchild
- can be missing node ids and/or node labels
- may or may not have a DTD present

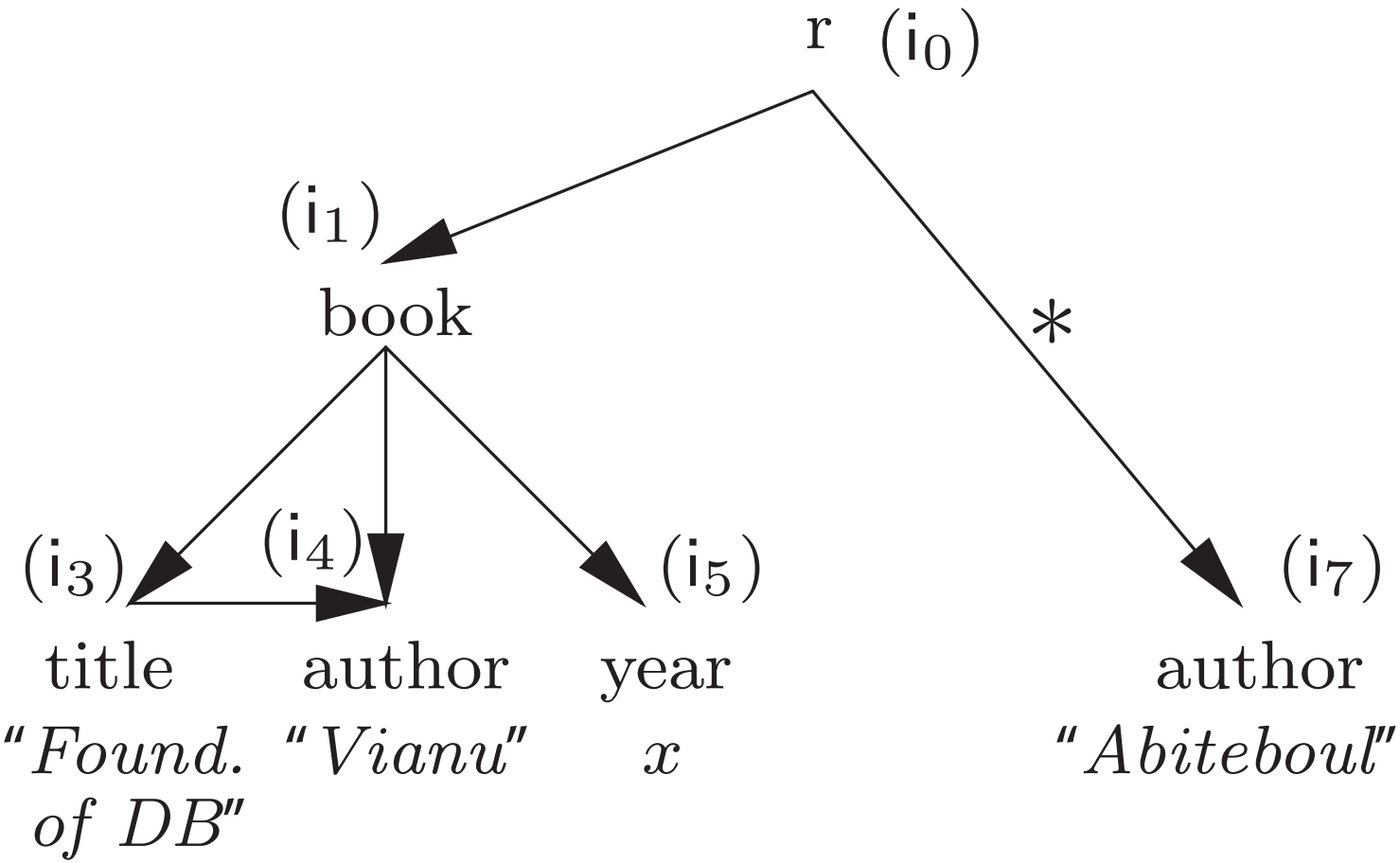
Incomplete Information (1)



Incomplete Information (2)



Incomplete Information (3)



Contributions

Give a taxonomy of incomplete information models for XML

Study the complexity of key computational problems as a function of the types of incompleteness allowed

- consistency
- membership
- query answering (for queries that return tuples)

Kinds of incomplete information considered

- **labels:** may be replaced by wildcards
- **node ids:** either all absent or all present
- **structural information**
  - may use any subset of the axes $\downarrow, \downarrow^*, \rightarrow, \rightarrow^*$
  - may specify siblings without sibling order
  - may use **markings**: `root`, `leaf`, `first child`, `last child`
- **data values:** either constants and variables (cf. naïve tables) or totally absent
- **DTD:** may be present or not

Goal: understand which of these features impact complexity

Consistency

This is always in NP

Results overview:

- without node ids and without a DTD
  - only markings can lead to inconsistency
  - with markings, NP-complete for three specific fragments and in PTIME otherwise
- adding a (fixed) DTD leads to intractability even for very simple descriptions
- node ids help a lot
  - always in PTIME without a DTD
  - even with a fixed DTD, PTIME as long as descendant relation not used
  - but remains NP-complete if DTD not fixed

Membership

This is also always in NP

Results overview:

- with node ids, is in PTIME
- without node ids, is NP-complete even for simple descriptions
- but drops to PTIME if we restrict each (data value) variable to occur only once in the tree
- cf. relational case – membership complexity for Codd tables vs. naïve tables
  - although proof technique used is different

Query answering

Query language:

- a query is an incomplete tree with no node ids and existential quantification over the attribute value variables it contains
  - a **tree pattern**
  - answers are valuations
  - analogous to relational conjunctive queries
  - full language: unions of such queries
- classes of queries can be defined based on the structural information they use
- since queries return tuples, can define certain answers in the usual way

$$certain(q, t) = \bigcap \{q(T) \mid T \in Rep(t)\}$$

Query answering

Results overview: generally, the news is not good

- problem is always in co-NP
- DTDs or markings in trees and queries induce co-NP completeness
- but can get co-NP completeness even without either of these
- $\downarrow^*$ and $\rightarrow^*$ cause problems too
- a tractable case: the trees are severely restricted to **rigid incomplete trees**
  - essentially a complete tree that may use variables for attribute values and wildcards for node labels
  - can perform relational-style naïve evaluation over relational representations of such trees for tractable query answering
    - as long as the query does not use markings

Query answering under mappings in XML

S. Amano, C. David, L. Libkin, and F. Murlak. **On the tradeoff between mapping and querying power in XML data exchange**. ICDT 2010.

- a study of the complexity of query answering in data exchange setting

Setting: have an XML schema mapping $\langle D_s, D_t, \Sigma \rangle$ where

- D_s and D_t are source and target DTDs
- Σ is a set of source-to-target dependencies in a suitable language

Also have a query language and want to pose queries over D_t

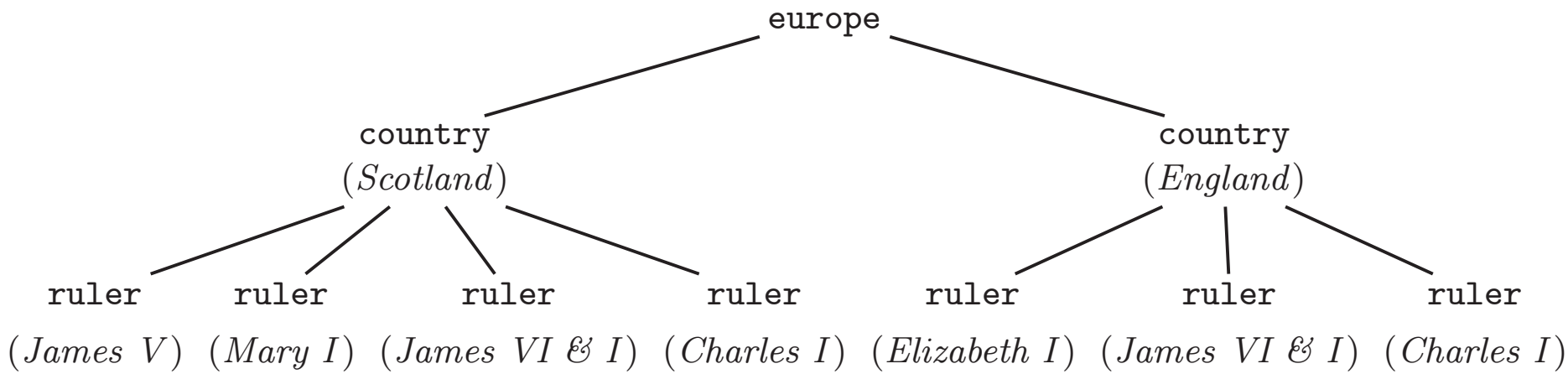
- queries still return tuples

Contributions

The paper is a study of the (data) complexity of computing certain answers as we vary the expressiveness of:

- the query language
- the mapping language used for source-to-target dependencies
- (the DTDs)

An example source document



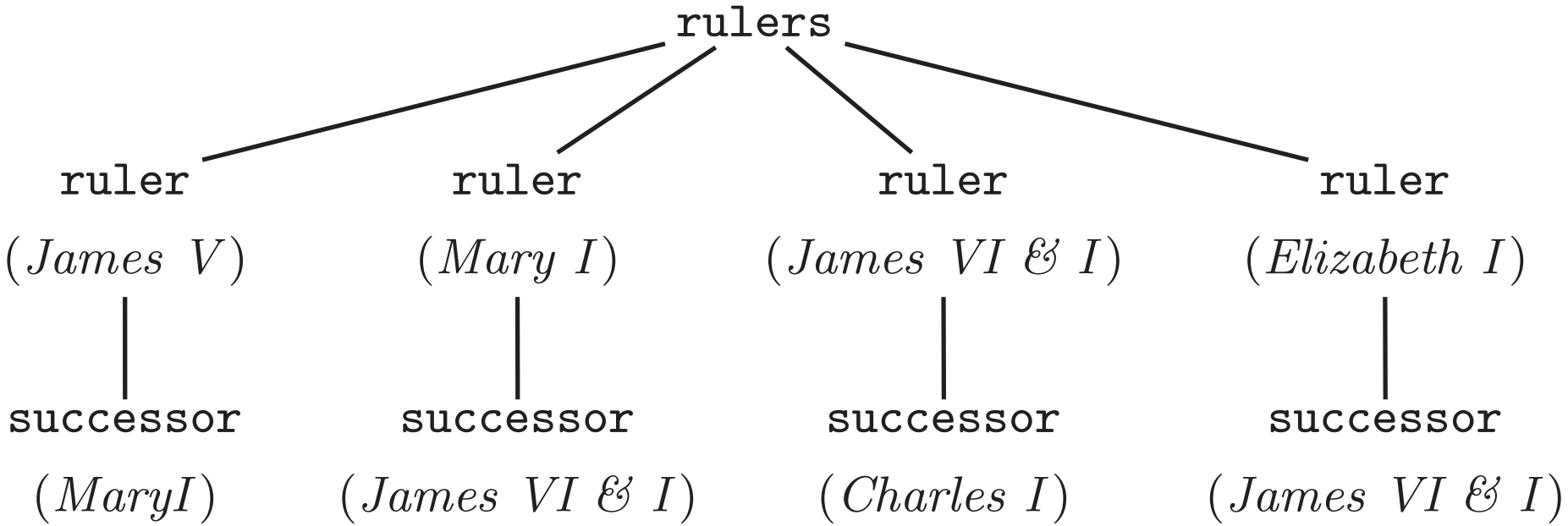
Source DTD

$$\begin{aligned}
\texttt{europe} &\rightarrow \texttt{country}^* \\
\texttt{country} &\rightarrow \texttt{ruler}^* \\
\texttt{ruler} &\rightarrow \varepsilon \\
\texttt{country} &: \texttt{@name} \\
\texttt{ruler} &: \texttt{@name}
\end{aligned}$$

Target DTD

$$\begin{aligned}
\texttt{rulers} &\rightarrow \texttt{ruler}^* \\
\texttt{ruler} &\rightarrow \texttt{successor} \\
\texttt{successor} &\rightarrow \varepsilon \\
\texttt{ruler} &: \texttt{@name} \\
\texttt{successor} &: \texttt{@name}
\end{aligned}$$

An example solution (target) document



Mapping language

Language for mappings between source and target documents based on tree patterns

- very expressive, allows vertical and horizontal navigation as well as equality/inequality constraints on variables

Example tree pattern:

$$\texttt{europe/country}(z)[\texttt{ruler}(x) \rightarrow \texttt{ruler(y)}]$$

Example source-to-target dependency:

$$\texttt{europe/country}(z)[\texttt{ruler}(x) \rightarrow \texttt{ruler(y)}] \Rightarrow$$

$$\texttt{rulers/ruler}(x)\texttt{/successor(y)}$$

Query answering

Query language: same tree patterns as used for mappings

- can restrict queries (or mappings!) to disallow some features e.g. horizontal navigation
- answers are valuations as before

Assume we are given a query q, a mapping $M = \langle D_s, D_t, \Sigma \rangle$ and a source document T conforming to D_s.

$$certain_M(q, T) = \bigcap \{q(T') \mid T' \text{ is a solution for } T \text{ under } M\}$$

Query answering – known results

Some results known from previous work which used a subset of the mapping language without horizontal navigation and inequality comparisons

- for tractability, need to restrict DTDs, specifically wrt disjunction
  - nested relational DTDs
- also need to restrict mappings to **fully specified** ones
  - use neither $\downarrow^*$ nor _ in target patterns
- otherwise the problem is co-NP complete

Main question in this paper: how do the new language features affect complexity?

New results – the good news

Even with the most expressive mappings and queries, the complexity of query answering remains in co-NP

If the query language and DTD is kept simple, full horizontal navigation can be added to mappings without loss of tractability

Query answering remains in PTIME when:

- DTDs are nested relational
- queries may use vertical navigation ($\downarrow, \downarrow^*$) and equality comparisons
- mappings may use everything except $\downarrow^*$ and _ (still!)

New results – the bad news

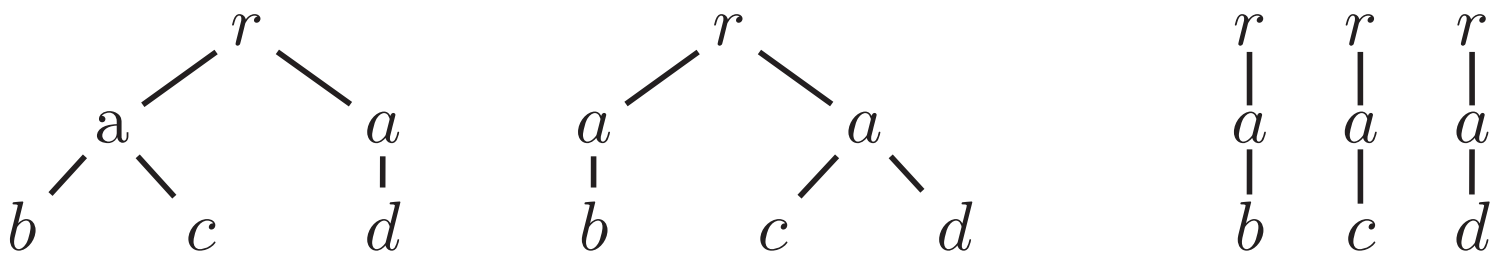
Extending the expressiveness of queries leads to intractability quickly

- any form of horizontal navigation leads to co-NP completeness
- even if the mappings can only use the child relation
- and even if the DTDs are nested relational
- and even under some additional restrictions

Takeaway on horizontal order: ok to use in mappings, but not in queries.

Certain answers for queries that return trees

C. David, L. Libkin, and F. Murlak. **Certain answers for XML queries**. PODS 2010.



First step: revisit foundations of relational certain answers

The theory of certain answers

Observation 1: Given a representation of a set of databases $\mathcal{D}$, need a way to represent all the information that is true for all $D \in \mathcal{D}$

- $\mathcal{D}$ can be a set of query results, i.e. $\{q(D') \mid D' \in \mathcal{D}'\}$, but does not need to

The notion of "all the information that is true" depends on what language we have available to represent it

- if we can only represent ground tuples, the our "certain information" is limited to the ground tuples that are found in all $D \in \mathcal{D}$
- but if we can use naïve tables, can represent more information (weak representation systems)

The theory of certain answers

Observation 2: A representation of a set of databases $\mathcal{D}$ in a language $\mathcal{L}$ can be viewed as a logical theory $\mathcal{L}_\mathcal{D}$.

- $\mathcal{D} = Mod(\mathcal{L}_\mathcal{D})$

Example: if $\mathcal{D}$ is represented by a naïve table R, then R defines a conjunctive query q_R (R is the tableau of q_R)

- view q_R as a logical formula
- for a database D, $D \in Rep(R)$ if and only if D is a model of q_R

Given a query q on $\mathcal{D}$, the certain answers are those implied by $\mathcal{L}_\mathcal{D}$

- e.g. if we are interested in ground facts, we want tuples $\overline{a}$ such that $\mathcal{L}_\mathcal{D} \vdash q(\overline{a})$

Max-descriptions

Suppose $\mathcal{L}$ is a logical formalism and $\mathcal{D}$ a class of databases

The certain $\mathcal{L}$-knowledge of the class $\mathcal{D}$ is the $\mathcal{L}$-theory of $\mathcal{D}$, denoted $Th_{\mathcal{L}}(\mathcal{D})$

- this is the set of all $\mathcal{L}$-formulae satisfied in all structures from $\mathcal{D}$

Want a finite set of $\mathcal{L}$-formulae Φ such that $Mod(\Phi) = Mod(Th_{\mathcal{L}}(\mathcal{D}))$

- if such a set exists, we call it a **max-$\mathcal{L}$-description** of $\mathcal{D}$ (or max-description if $\mathcal{L}$ is clear)

Max-descriptions and certain answers

Back to certain answers:

- given a set $\mathcal{D}$ and a query q, the certain answers to q over $\mathcal{D}$ are represented by a max-description of $\{q(D) \mid D \in \mathcal{D}\}$

A max-description of a set $\mathcal{D}$, if it exists, need not be unique

- but there may be a **core** – a smallest max-description with the property that all others can be minimized to it

Applying this to XML

Language $\mathcal{L}$ – simple tree patterns π:

- fully specified trees with attribute variables

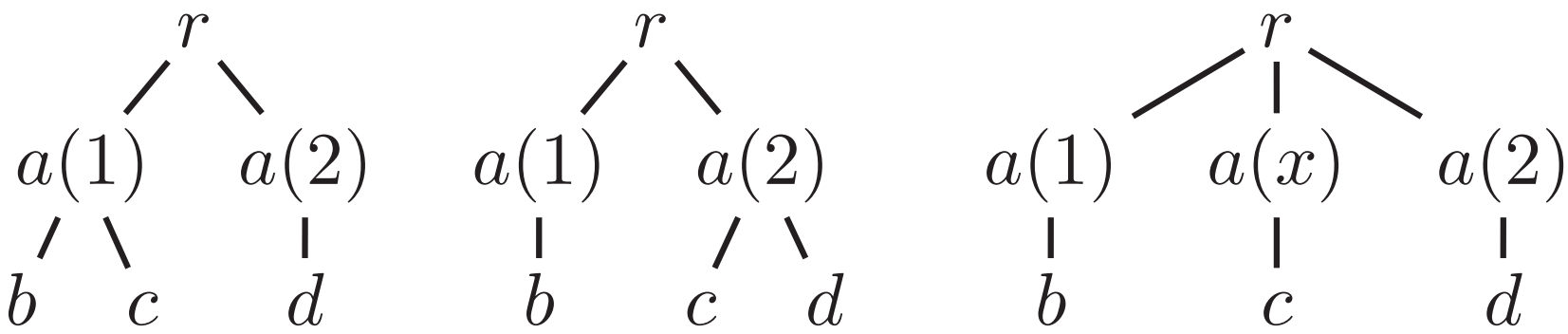
If $\mathcal{T}$ is a set of trees, then

$$Th(\mathcal{T}) = \{\pi \mid \forall T \in \mathcal{T} \ : \ T \models \pi\}$$

A pattern π is a max-description for a set of trees $\mathcal{T}$ if

$$Mod(\pi) = Mod(Th(\mathcal{T}))$$

Max-description for our example trees



Max-descriptions and cores

The paper gives results about the complexity of computing max-descriptions for sets of XML trees

Also give a definition of **core** of a max-description

- defined using homomorphisms
- theorem with bounds on the core size (upper and lower)

Back to query answering

Give a query language that returns trees

- uses patterns
- has the flavor of XQuery FLWR expressions

Certain answers to query q over $\mathcal{T}$ to given by a max-description of $q(\mathcal{T})$

Introduce the notion of a **basis** for a set $\mathcal{T}$ – intuitively, a more concise representation

- a basis $\mathcal{B}$ for $\mathcal{T}$ can help in computing certain answers
- if q is a query in their language, then certain answers to q over $\mathcal{B}$ and $\mathcal{T}$ coincide
- sufficient to compute a max-description of $q(\mathcal{B})$

Putting it all into practice

Paper gives two case studies for certain answers

- XML with incomplete information
- data exchange

Show how to compute small bases for the appropriate sets $\mathcal{T}$

Answer several open complexity questions

Define a new tractable class of data exchange problem by placing a suitable restrictions on source-to-target dependencies

- restriction guarantees the existence of small bases

Summary

- incomplete information in XML
  - many more kinds of incompleteness than in relational case
  - complexity of query answering very sensitive to the kind of incompleteness allowed in the representation
- query answering in XML under mappings
  - again, complexity very sensitive to parameters chosen for expressiveness
  - sometimes surprising/nonintuitive
- certain answers for queries that return trees
  - nontrivial, but definitely doable
- lots of interesting work remains to be done!

Additional slides

This section contains additional slides for the following paper:

S. Abiteboul, L. Segoufin, and Victor Vianu: Representing and querying XML with incomplete information. ACM Trans. Database Syst. 31(1), pp. 208-254, 2006

Representing and Querying XML with Incomplete Information

One of the first papers on representing incomplete information in XML with a query answering focus

Setting: maintain an incomplete, but growing XML document that represents Web data

- document is a data repository
- can be grown by asking more queries of external data sources to retrieve more information
- assumptions:
  - data is static
  - DTDs of sources are available

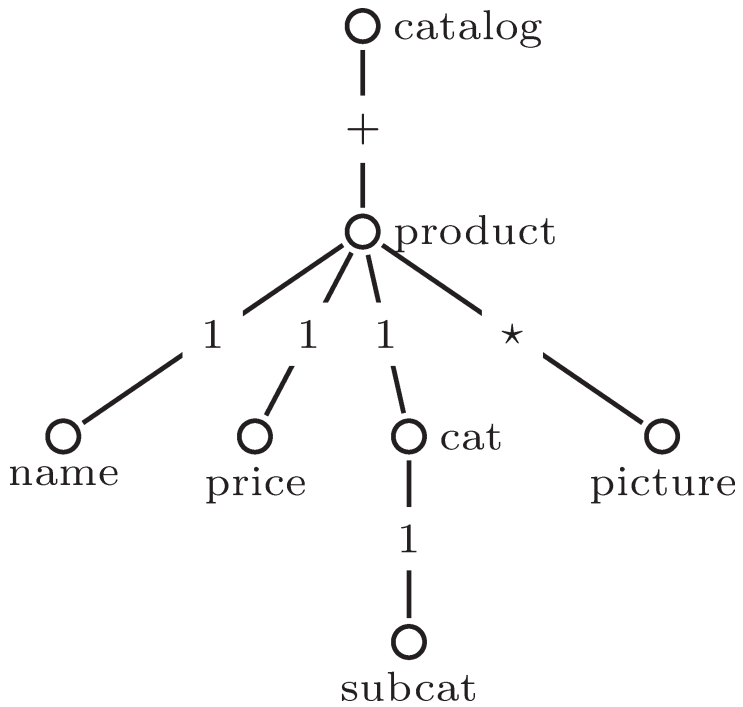
Document and schema model

Document model:

- assume node ids, no order, no attribute names

Schema model:

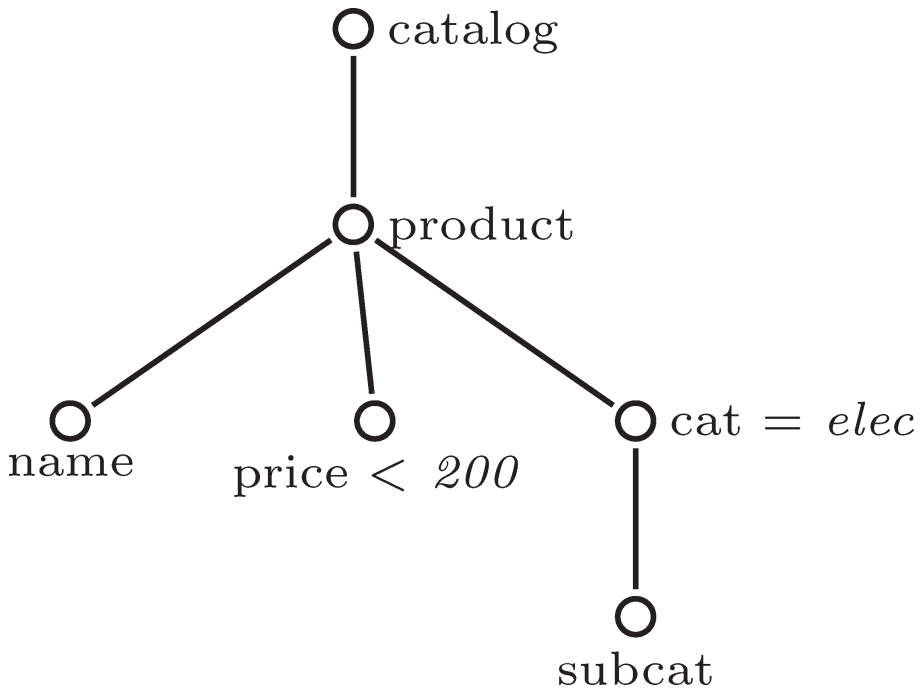
- simplified DTD with all child multiplicities restricted to $\{*, +, ?, 1\}$
- no ordering constraints as with standard regular expressions



Query language

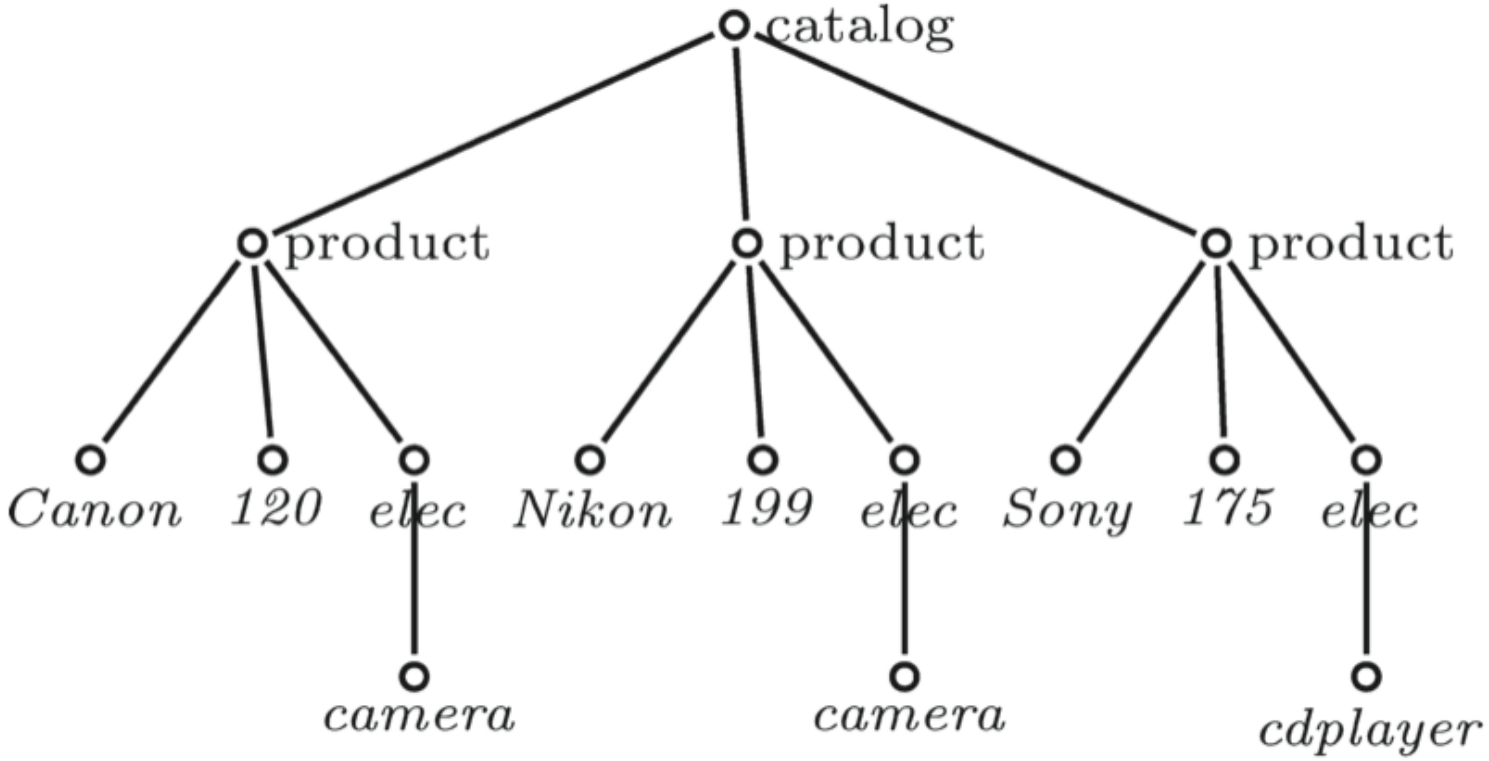
Queries are tree patterns with optional selection conditions

- two sibling nodes cannot have the same label
- no descendant navigation, data joins, etc.



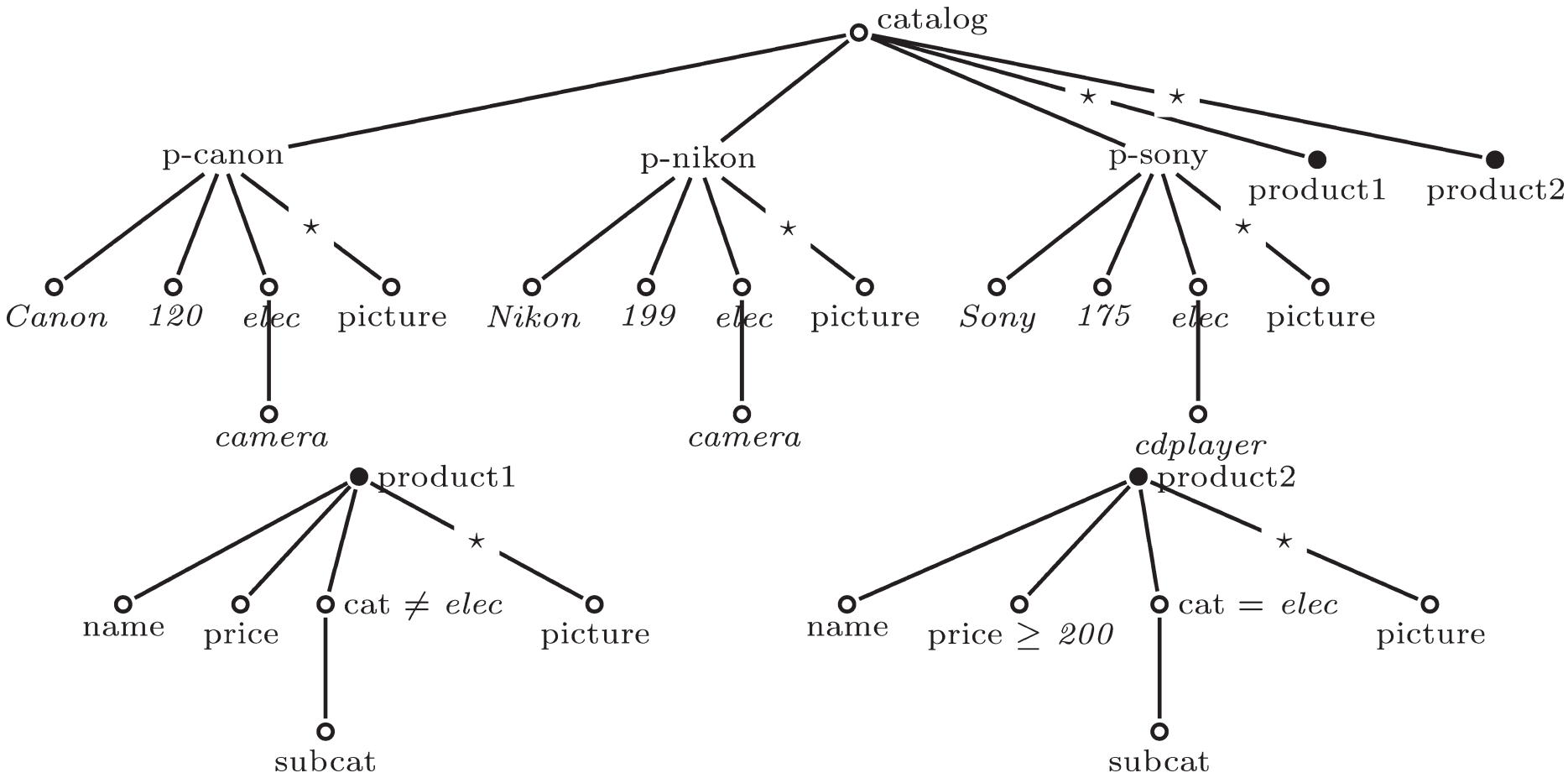
Query semantics

Queries return subtrees of the document based on matches of the query pattern



Motivation for incomplete documents

Based on a query result, can build incomplete representation of the underlying data



Model for representing incomplete information in XML

Main features of the incomplete representation:

- conditions on data values e.g. > 200
- specialization of node labels, e.g. `product1` and `product2` are specializations of `product`
- some nodes may be "fully instantiated" (node ids are known)
- the "DTD" may now contain disjunctions of multiplicity atoms, e.g. $na^* + a^+$

Theorem: Consistency can be decided in PTIME

Query answering

Theorem: this representation is a strong representation system for the query language in the paper

Let Σ be a fixed set of node labels. Given an incomplete tree $\mathbf{T}$ and a query q, one can construct an incomplete tree $q(\mathbf{T})$ such that

$$Rep(q(\mathbf{T})) \;=\; \{q(T) \mid T \in Rep(\mathbf{T})\} \;=\; q(Rep(\mathbf{T}))$$

Furthermore, $q(\mathbf{T})$ can be constructed in PTIME with respect to q and $\mathbf{T}$ (exponential in Σ)

Certain answers

Can define certain answers using $q(\mathbf{T})$

Idea: given any incomplete representation, can define **certain prefixes** (and **possible prefixes**)

- tree prefix defined formally in paper (need to account for node ids)
- a certain prefix of $q(\mathbf{T})$ is a certain answer to q with respect to $\mathbf{T}$
- it can be determined in PTIME whether a given tree is a certain prefix of $q(\mathbf{T})$

Other results in paper

Focus on the specific application setting

- algorithm for refining representation based on successive query answers
- methods for shrinking the size of the representation
  - representation that allows conjunction
  - restrictions on queries
  - algorithm for generating queries that supply crucial information
- "deep search"
  - given a query q, if the answer to q on the local document is unsatisfactory, generate additional queries for a more precise answer
- extensions: more expressive queries, document order, no node ids
  - all associated with an increase in complexity