Web Queries

From a **Web of Data** to a **Semantic Web**

François Bry, Tim Furche, Klara Weiand

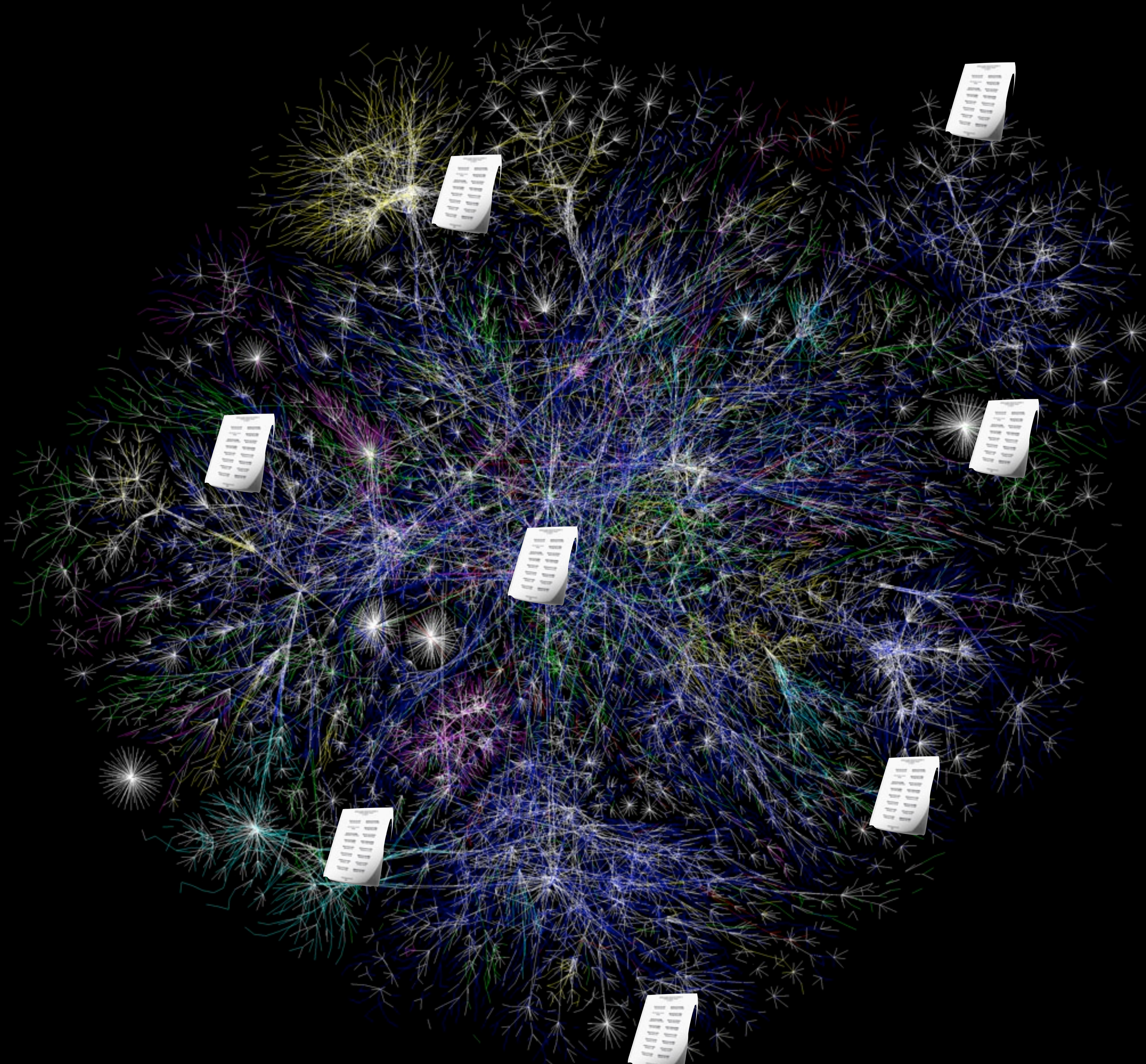
based on work in the European project **KiWi** “Knowledge in a Wiki”

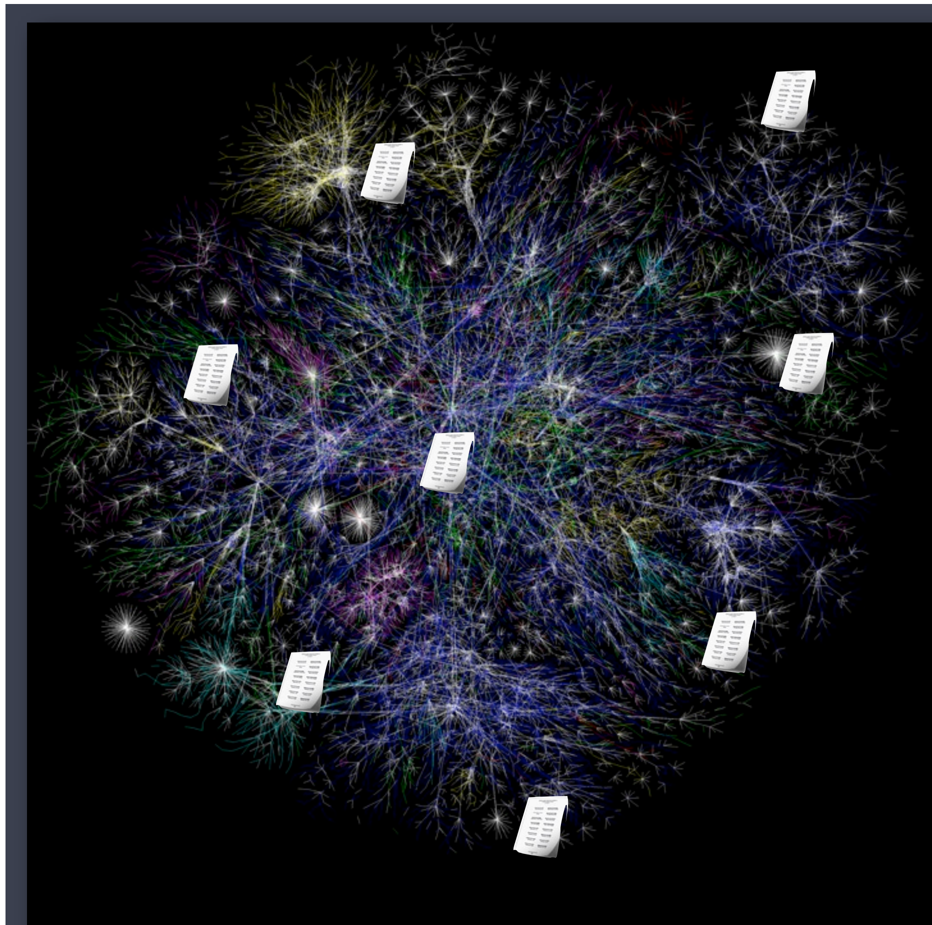
Google

Y!

YAHOO!

Ask.com





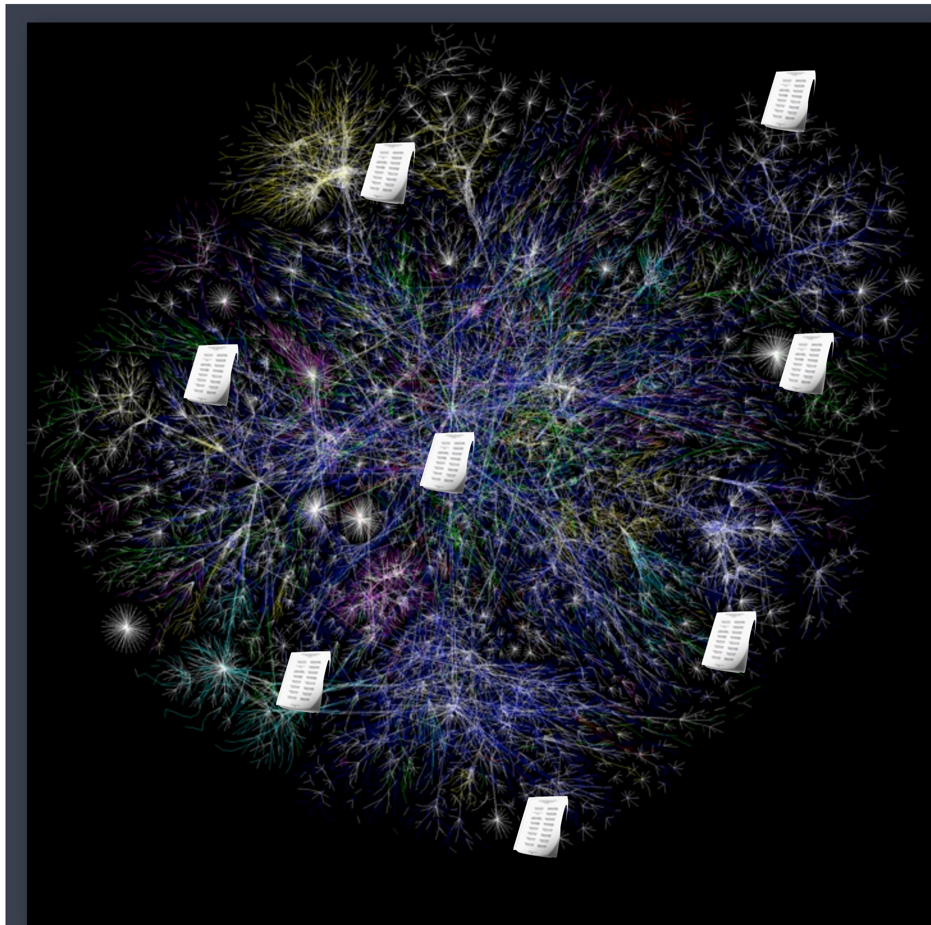
keyword search

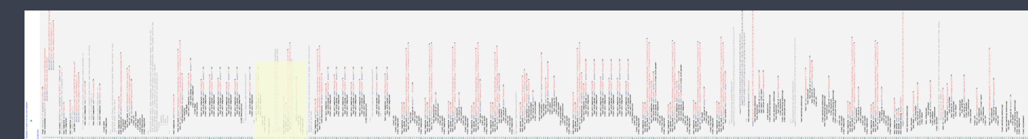
ranked collection of documents

?

?

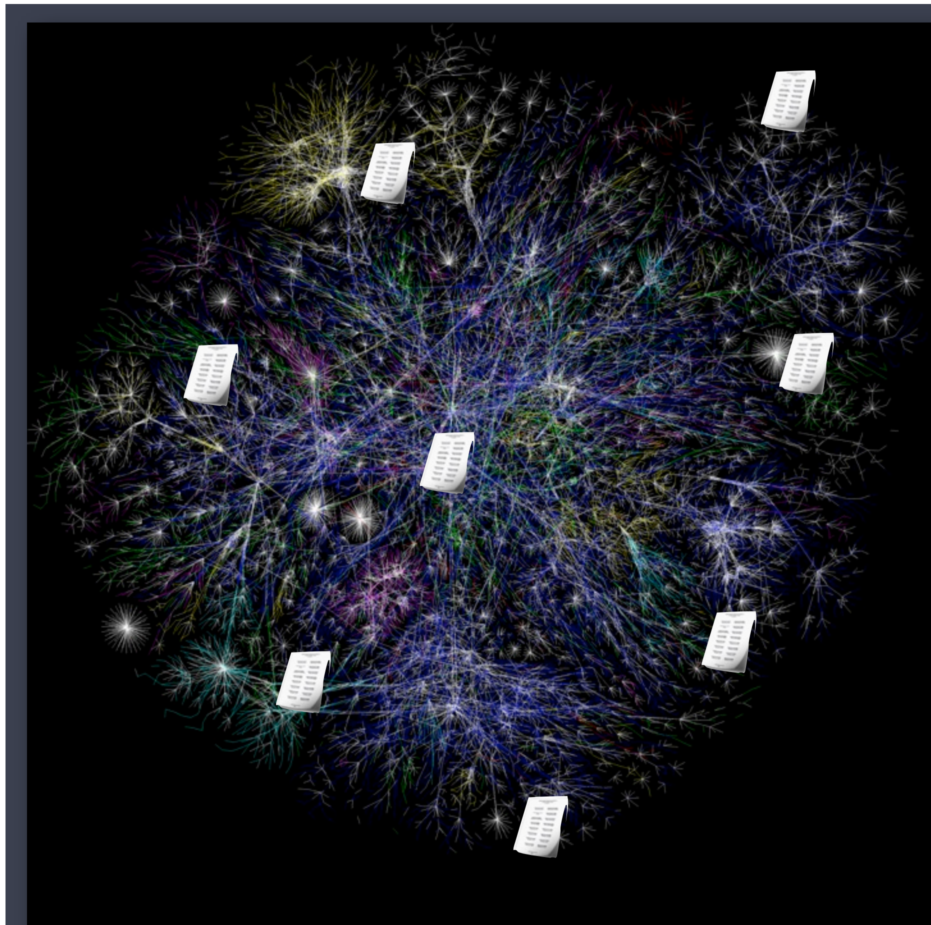
but what does Web **querying** do?

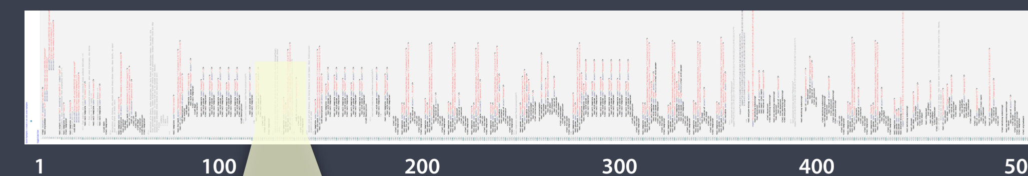





```
<xsl:template match="*[ matches(@class,'(^|\s)vevent(\s|$)')] ">
        <rdf:Description rdf:about="{ generate-id() }">
            <xsl:apply-templates select=".//*[ matches(@class,'(^|\s)summary(\s|$)')]
                                    [ ancestor::*[ matches(@class,'(^|\s)vevent(\s|$)')][ 1]
                                     = current() ]" />
            <xsl:apply-templates select=".//*[ matches(@class,'(^|\s)description(\s|$)')]
                                    [ ancestor::*[ matches(@class,'(^|\s)vevent(\s|$)')][ 1]
                                     = current() ]" />
            ...
        </rdf:Description>
</xsl:template>
```

Share value: **27.40$**






```
<xsl:template match="*[ matches(@class,'(^|\s)vevent(\s|$)')]">
        <rdf:Description rdf:about="{ generate-id()}">
            <xsl:apply-templates select=".//*[ matches(@class,'(^|\s)summary(\s|$)')]
                                  [ ancestor::*[ matches(@class,'(^|\s)vevent(\s|$)')][ 1]
                                   = current() ]" />
            <xsl:apply-templates select=".//*[ matches(@class,'(^|\s)description(\s|$)')]
                                  [ ancestor::*[ matches(@class,'(^|\s)vevent(\s|$)')][ 1]
                                   = current() ]" />
            ...
        </rdf:Description>
</xsl:template>
```

Share value: **27.40$**

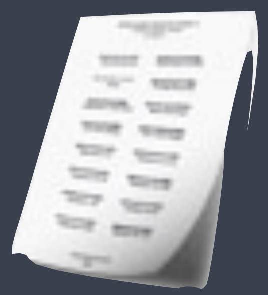
Automatically **sell** at < 28.00$

Action!

?

?

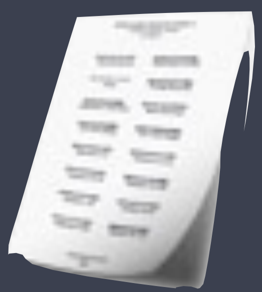
if you believe that I have some bank stocks for you ...



or

query: value of specific element

Share value: **27.40$**



or

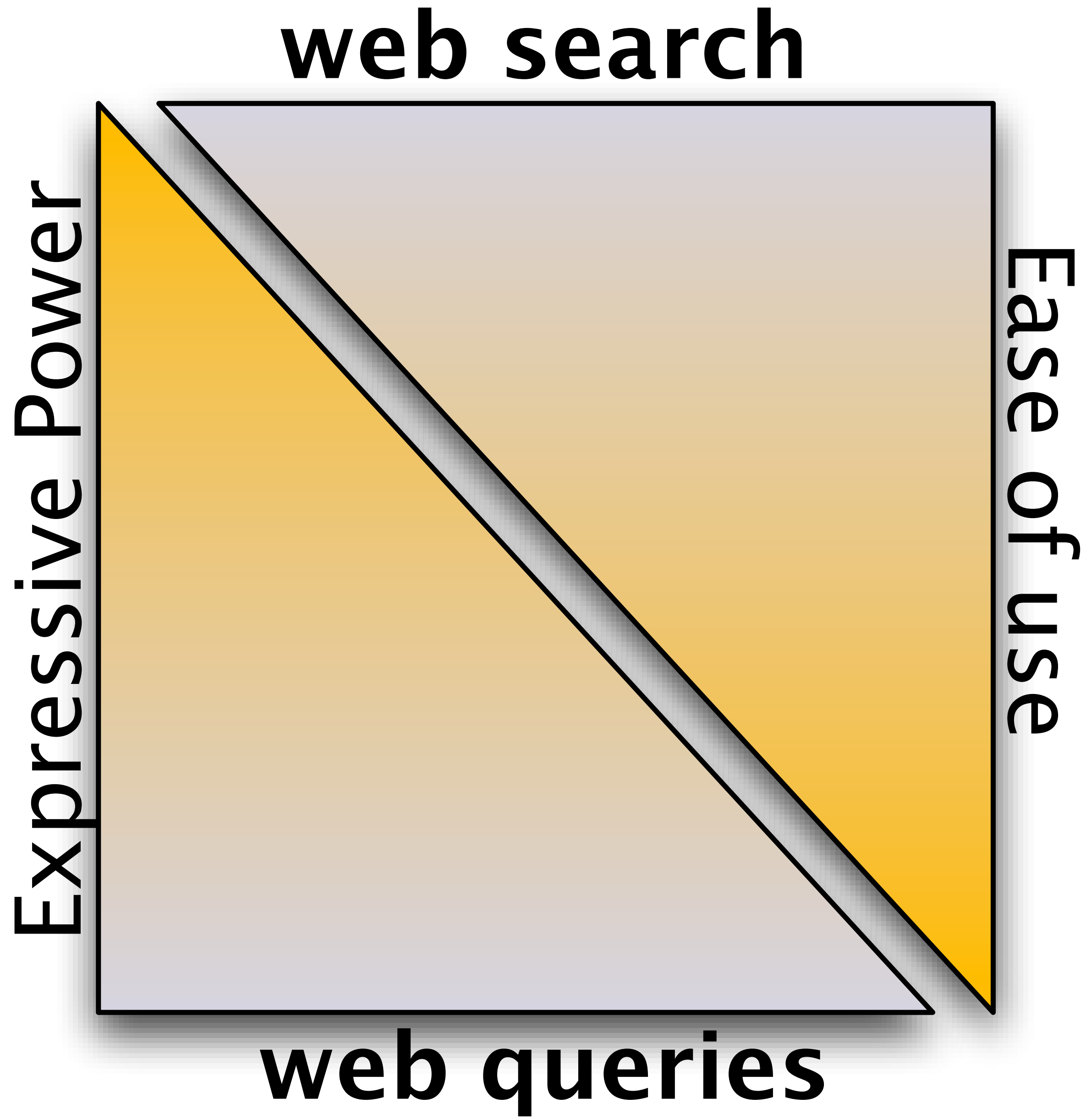
query: value of specific element

Share value: **27.40$**

rule: automatically **sell** at < 28.00$

Action!

	Web Search	Web Queries
Scale	**“Web”**, TB/PBs	a few “documents”, GBs
Parallelizability	very **high** (NC)	for basic selection lang. **high**, otherwise very **low**
Data	**independent** documents, heterogenous	trees (XML) or graphs (RDF), homogenous
Used by	(almost) everyone, many casual users	few experts
Expertise Level/Knowledge required	**low**	very **high**
Expressiveness	very **low**	very **high**
Result presentation	Ranking, clustering...	programmed
Matching	often fuzzy or vague	only **precise** answers
Return value	Documents (or summaries thereof)	parts of trees or graphs
Actionability	very **low**	very **high**



In a nutshell...

Web Queries

Search + Query = Easy + Automation

- **Goals:**
  - make web querying **accessible** to casual users
    - e.g., to enable data aggregation/mashups by users
  - allow **precise** queries over vastly **heterogeneous** data
  - **precise** queries and rules critical to the (Social) Semantic Web
    - unless they are **accessible** no widespread adoption
- **Classification** of **approaches** to combining Web search & query
  - enhance **search**: add data extraction / object search
  - enhance **querying**: add keyword search / information retrieval
  - **keyword-based QLs** for structured data: grounds-up **redesign**

Search + Query

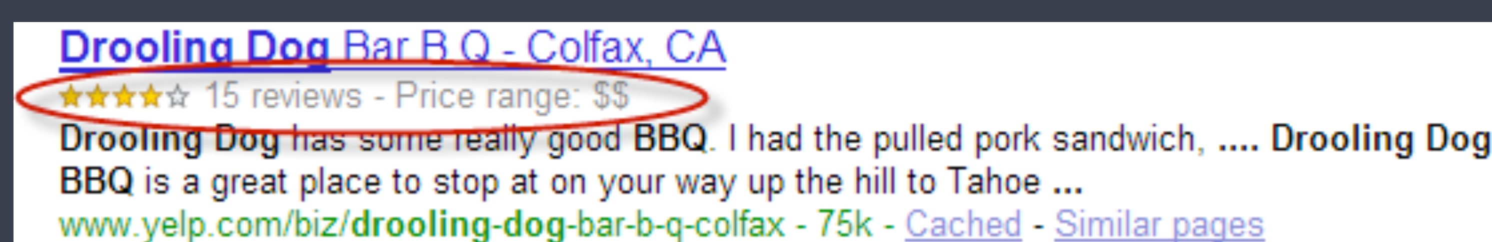
Approach 1: Enhance **Search**

- **"Peek"** into web documents
- **Extract** data items / Web objects from web documents
  - to provide more fine-grained answers
- **Examples**
  - Google (Squared)
  - Google Rich Snippets
  - Yahoo! Search Monkey

Search + Query

Approach 1: Enhance **Search**

- **“Peek”** into web documents
- **Extract** data items / Web objects from web documents
  - to provide more fine-grained answers
- **Examples**
  - Google (Squared)
  - Google Rich Snippets
  - Yahoo! Search Monkey



Approach 2: Enhance **Querying**

- Enhance existing **(XML) web query languages**
  - by adding information retrieval functionality
    - ranking, scoring, fuzzy matching
- **Examples**
  - XQuery and XPath Full Text 1.0, W3C Cand. Recommendation

Search + Query

Approach 2: Enhance **Querying**

- Enhance existing **(XML) web query languages**
  - by adding information retrieval functionality
    - ranking, scoring, fuzzy matching
- **Examples**
  - XQuery and XPath Full Text 1.0, W3C Cand. Recommendation

1

```
doc('bib.xml')/bib/book[title ftcontains 'programming']
```

2

```
for      $b score $s in doc('bib.xml')/bib/book
where    $b ftcontains 'web' ftand 'query'
order by $s descending
return   <title> {$b//title} </title>
```

Approach 3: **Keyword Queries**

- **Keyword query** for structured Web (XML and RDF) data
  - apply web **search paradigm** to querying tree and graph data
  - operates in the same **setting** as e.g. XQuery and SPARQL
    - few or single document, no Web scale
  - but: easier handling of **very heterogeneous** data
  - querying of semi-structured data through **easy-to-use interface**
- **Ultimate goal:**
  - **Easy** usage combined with enough **power**
    → to **automate** data processing tasks

Search + Query

Approach 3 … ries

- **Keyword query** for str … Web (XM
  - apply web **search pa** … to q … ying tr
  - operates in the sam … **ng** as … g. X … uery a … QL
    - few or single docu … o Web … cal
  - but: easier handlin … **y he … rog … ne … us** data
  - querying of semi-st … d da … thr … **y-to-us … nterface**
- **Ultimate goal:**
  - **Easy** usage combined …

→ to **au** … ng tasks

Focus of this tutorial

K. Weiand, T. Furche, and F. Bry.
Quo vadis, web queries?
In Web4Web, 2008.

Overview

1. Summary of **web query research** in the 00s
    1.1. XML
    1.2. RDF
2. **Keyword** query languages
    2.1. Motivation
    2.2. Classification
    2.3. Issues
3. **KWQL**
4. Discussion and Outlook

Part 1
Web Queries

Summary of XML & RDF
Query Language Research

1.1 XML

Tree Data & Tree Queries

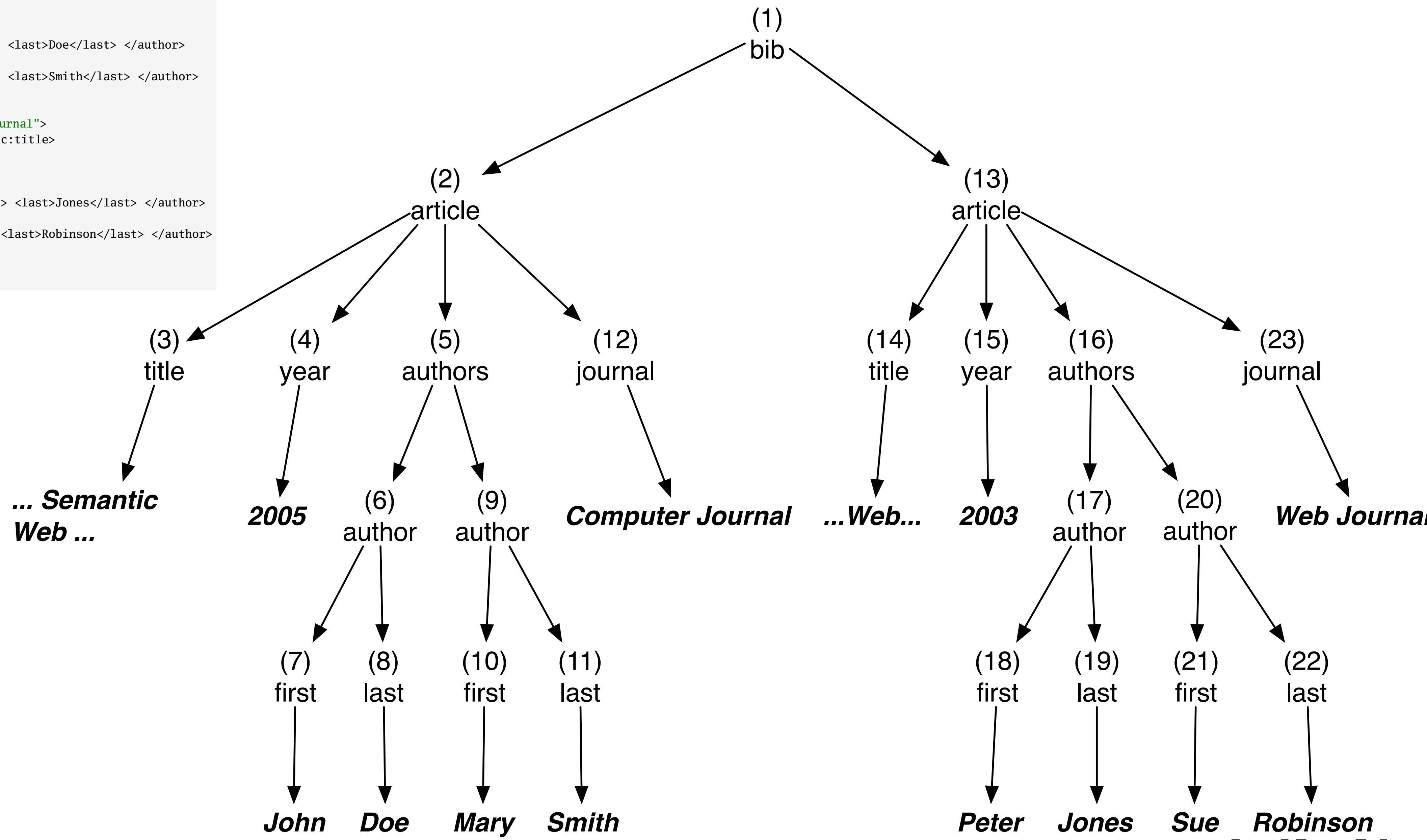
Data–XPath–XQuery

```xml
<bib xmlns:dc="http://purl.org/dc/elements/1.1/">
 <article journal="Computer Journal" id="12">
   <dc:title>...Semantic Web...</dc:title>
   <year>2005</year>
   <authors>
     <author>
       <first>John</first> <last>Doe</last> </author>
     <author>
       <first>Mary</first> <last>Smith</last> </author>
  </authors>
 </article>
 <article journal="Web Journal">
   <dc:title>...Web...</dc:title>
   <year>2003</year>
   <authors>
     <author>
       <first>Peter</first> <last>Jones</last> </author>
     <author>
       <first>Sue</first> <last>Robinson</last> </author>
  </authors>
 </article>
</bib>
```

XML

ordered tree (sometimes **graph**), mostly uniform edges

```
<bib xmlns:dc="http://purl.org/dc/elements/1.1/">
 <article journal="Computer Journal" id="12">
   <dc:title>...Semantic Web...</dc:title>
   <year>2005</year>
   <authors>
     <author>
       <first>John</first> <last>Doe</last> </author>
     <author>
       <first>Mary</first> <last>Smith</last> </author>
   </authors>
 </article>
 <article journal="Web Journal">
   <dc:title>...Web...</dc:title>
   <year>2003</year>
   <authors>
     <author>
       <first>Peter</first> <last>Jones</last> </author>
     <author>
       <first>Sue</first> <last>Robinson</last> </author>
   </authors>
 </article>
</bib>
```

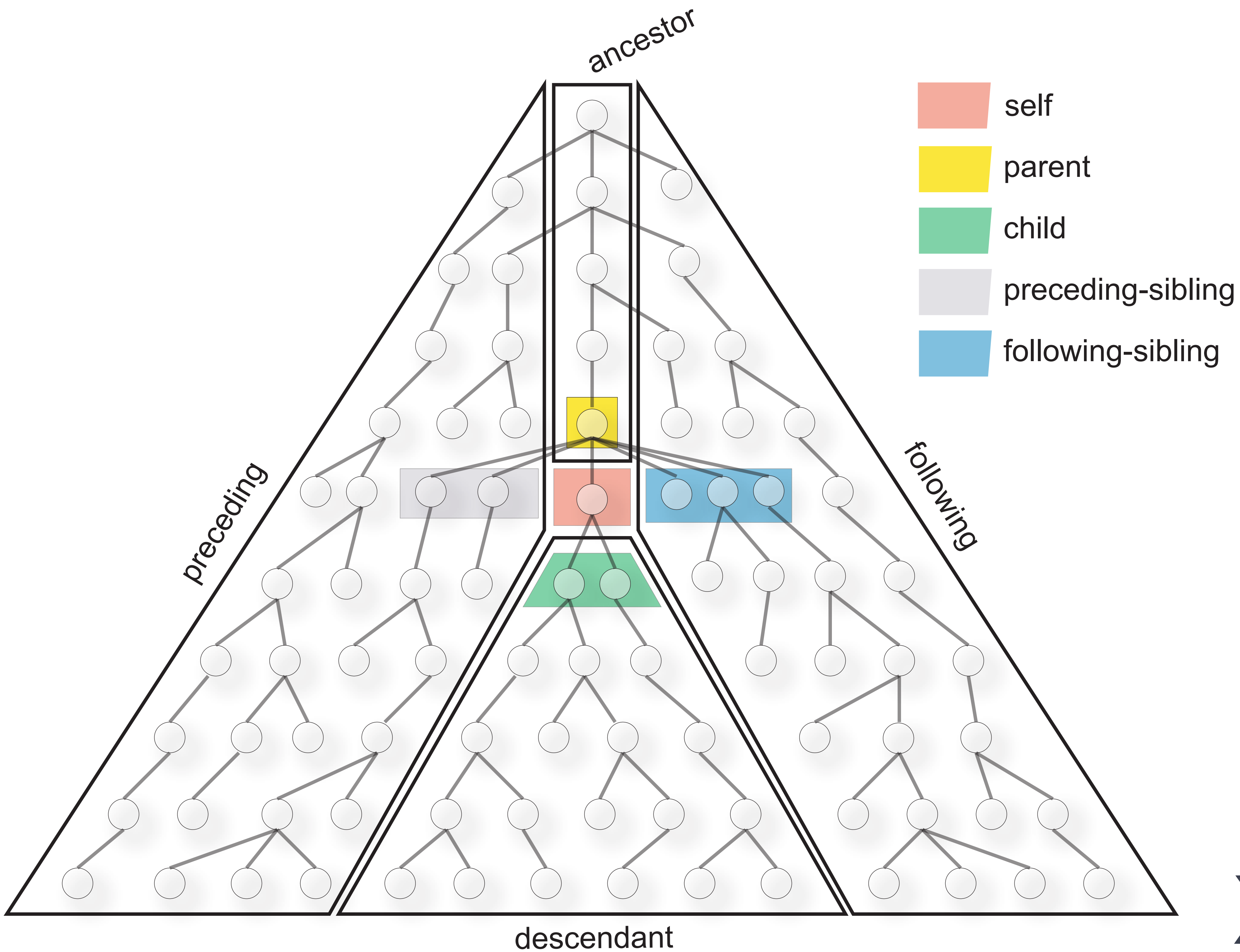


XML

ordered tree (sometimes **graph**), mostly uniform edges

?

what’s new about trees? didn’t we try that before?



XPath

axis for navigation/selection in tree, **context**, horizontal axes for **order**

XPath: Navigation in Trees

Intro in 5 Points

- **Data:** rooted, ordered, unranked, finite trees (no ID/IDREF resolution)
- **Paths** are sequences of **steps** (axis & test on properties of node)
  - adorned with existential **predicates** (in `[ ]`) to obtain **tree** queries
  - some more advanced features: value joins, aggregation, …
- **Answers** are sets of **nodes** from the input document
- `/child::html/descendant::h1[not(preceding::h1 = “Introduction”)]/child::p[attribute::class=”abstract”]`
- no **variables**, no **construction**, no **grouping/ordering**

$$
\begin{aligned}
[\![axis]\!]_{\mathsf{Nodes}}(n) &= \{(n' : R_{axis}(n, n')\} \\
[\![\lambda]\!]_{\mathsf{Nodes}}(n) &= \{(n' : \mathsf{Lab}^{\lambda}(n')\} \\
[\![\texttt{node()}]\!]_{\mathsf{Nodes}}(n) &= \mathsf{Nodes}(T) \\
[\![axis\texttt{::}nt\texttt{[}qual\texttt{]}]\!]_{\mathsf{Nodes}}(n) &= \{n' : n' \in [\![axis]\!]_{\mathsf{Nodes}} \wedge n' \in [\![nt]\!]_{\mathsf{Nodes}} \wedge \\
&\quad\; [\![qual]\!]_{\mathsf{Bool}}(n')\} \\
[\![step/path]\!]_{\mathsf{Nodes}}(n) &= \{n'' : n' \in [\![step]\!]_{\mathsf{Nodes}}(n) \wedge \\
&\quad\; n'' \in [\![path]\!]_{\mathsf{Nodes}}(n')\} \\
[\![path_1 \cup path_2]\!]_{\mathsf{Nodes}}(n) &= [\![path_1]\!]_{\mathsf{Nodes}}(n) \cup [\![path_2]\!]_{\mathsf{Nodes}}(n)
\end{aligned}
$$

$$
\begin{aligned}
[\![path]\!]_{\mathsf{Bool}}(n) &= [\![path]\!]_{\mathsf{Nodes}}(n) \neq \emptyset \\
[\![path_1 \wedge path_2]\!]_{\mathsf{Bool}}(n) &= [\![path_1]\!]_{\mathsf{Bool}}(n) \wedge [\![path_2]\!]_{\mathsf{Bool}}(n) \\
[\![path_1 \vee path_2]\!]_{\mathsf{Bool}}(n) &= [\![path_1]\!]_{\mathsf{Bool}}(n) \vee [\![path_2]\!]_{\mathsf{Bool}}(n) \\
[\![\neg path]\!]_{\mathsf{Bool}}(n) &= \neg [\![path]\!]_{\mathsf{Bool}}(n) \\
[\![\texttt{lab()} = \lambda]\!]_{\mathsf{Bool}}(n) &= \mathsf{Lab}^{\lambda}(n) \\
[\![path_1 = path_2]\!]_{\mathsf{Bool}}(n) &= \exists n', n'' : n' \in [\![path_1]\!]_{\mathsf{Nodes}}(n) \wedge \\
&\quad\; n'' \in [\![path_2]\!]_{\mathsf{Nodes}}(n) \wedge \cong (n', n'')
\end{aligned}
$$

XPath: Navigation in Trees

A Success Story

- **Commercial success:** One of the most successful QLs
  - widely implemented and used, several W3C standards
- **Research success:**
  - designed with little concern for formal "beauty"
  - but with few restrictions: turns out it hit a formal sweet spot
- **Expressiveness:** monadic datalog (datalog with only *unary* intensional predicates, i.e., answer predicates)
  - also: two-variable first-order logic (FO^2)
- **Polynomial** complexity, **linear** for navigational XPath

XPath: Navigation in Trees

A Success Story (cont.)

- **Completeness:** falls short of being first-order complete
  - for first-order completeness a “**conditional** axis” (UNTIL operator) needed
  - e.g., all *a* nodes reachable by a path of **only** *b* nodes
  - but: partially justified by results on **elimination of reverse axes**
    - **reverse** axis: parent, ancestor, preceeding, …
    - **eliminating** these axes possible in navigational XPath
      - though in few cases at exponential cost or resulting in introduction of expensive node-identity joins
      - we can **safely ignore** them in most cases
    - in **conditional** XPath this elimination is **not possible**

XPath: Navigation in Trees

A Success Story (cont.)

- **Completeness:** falls short of being first-order complete
  - for first-order completeness a "**conditional** axis" (UNTIL c
  - e.g., all *a* nodes reachable by a path of **only** *b* nodes
  - but: partially justified by results on **elimination of reverse axes**
    - **reverse** axis: parent, ancestor, preceeding, …
    - **eliminating** these axes possible in navigational XPath
      - though in few cases at exponential cost or resulting in introduction of expensive node-identity joins
      - we can **safely ignore** them in most cases
    - in **conditional** XPath this elimination is **not possible**

M. Marx. **Conditional XPath, the First Order Complete XPath Dialect.** PODS 2004.

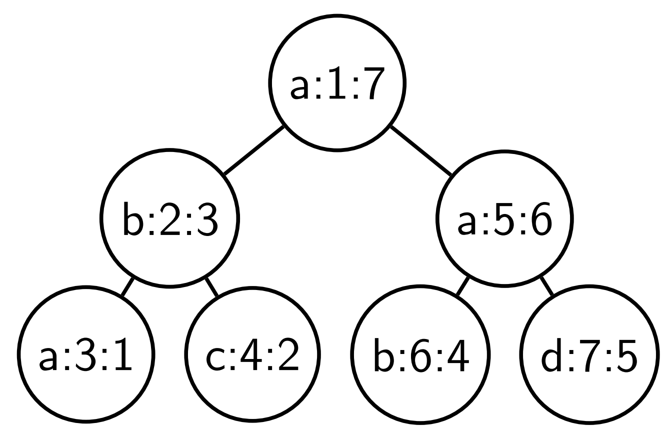
D. Olteanu, H. Meuss, T. Furche, and F. Bry. **XPath: Looking Forward**. XMLDM @ EDBT, LNCS 2490, 2002.

C. Ley and M. Benedikt. **How big must complete XML query languages be?** ICDT 2009.

XPath: Navigation in Trees

A Success Story (cont.)

- **Evaluation:**
  - naïve decomposition:
    - sequence of joins, descendant with closure over child relation
  - better: structural joins for descendant (single lookup using tree labeling)
    - e.g., pre/post encoding
    - fits nicely with relational storage



R	pre	post	lab
	1	7	a
	2	3	b
	3	1	a
	4	2	c
	5	6	a
	6	4	b
	7	5	d

  - twig joins: stack-based, holistic, not easily adapted to relational DBS

XPath: Navigation in Trees

A Success Story (cont.)

1. simple, fairly easy to learn
2. formal “sweet” spot
3. efficient evaluation

XPath: Navigation in Trees

A Success Story (cont.)

1. simple, fairly easy to learn
2. formal “sweet” spot
3. efficient evaluation

M. Benedikt and C. Koch. **XPath Leashed**. In *ACM Computing Surveys*, 2007

?

that’s it? everyone happy?

```
let $auction := doc('auction.xml')
for $o in $auction/open_auctions/open_auction
return
<corrupt id='{ $o/@id }'>
{ if (sum($o/(initial | bidder/increase)) = $o/current)
then text { 'no' }
else $auction//people/person[@id = $o//@person]/name }
</corrupt>
```

XQuery: Graph Queries & Construction

From XPath to Hell

- **Adds** an enormous set of features to XPath
  - price: highly complex language, Turing-complete, very hard to implement
  - here: just some highlights
- **Variables:** XPath plus variables → no longer FO^2
  - "an article that has the conference's chair as author"
  - not any more polynomial, but **NP**-/PSPACE-complete
- **Construction:** harmless if only at end of query
  - but: **composition** (i.e., querying of data constructed in the same query)
    - "find all papers written by the author with the most papers"
    - **NEXPTIME**-complete or in EXPSPACE (can not be captured by datalog)

XQuery: Graph Queries & Construction

From XPath to Hell

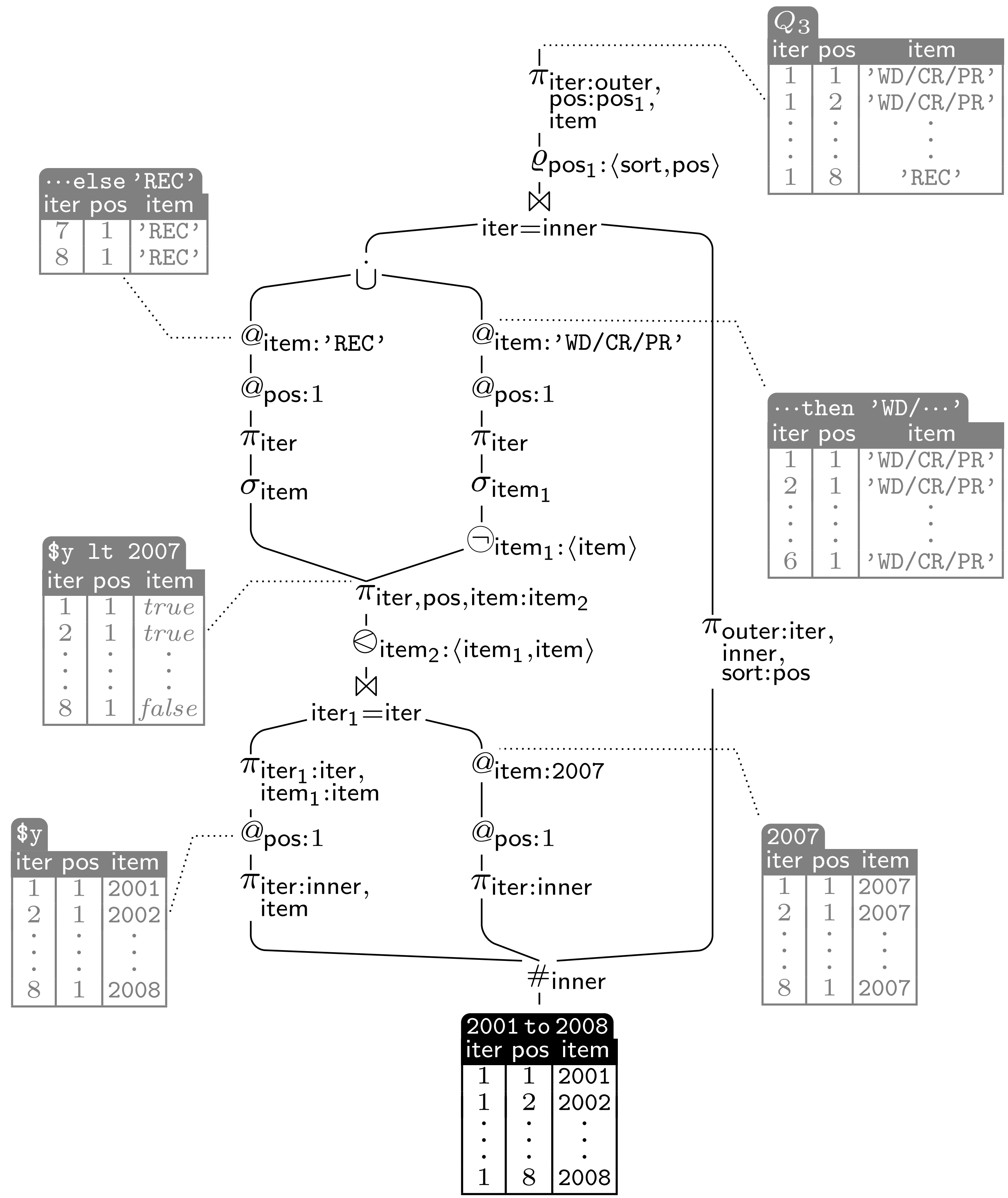
- **Recursion:** programmable
  - with composition → Turing complete
  - i.e., like Prolog or datalog with value invention
- **Sequences:** rather than sets
  - without composition little effect
  - with composition: drastically harder optimization as iteration semantics of a query must be precisely observed

s_0

```
for $y in 2001 to 2008
return
```

s_1

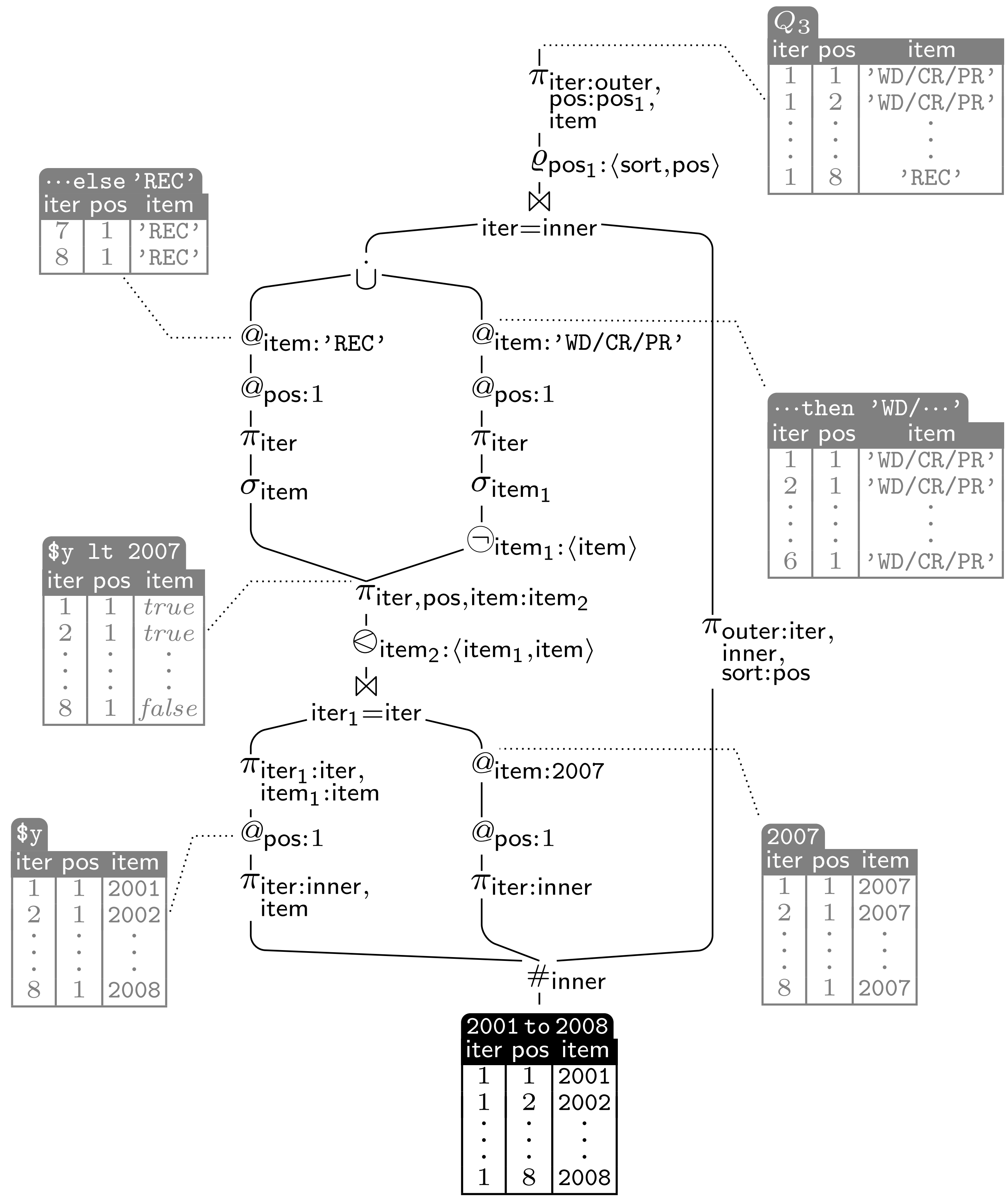
```
if ($y lt 2007)
then 'WD/CR/PR' else 'REC'
```



s_0

```
for $y in 2001 to 2008
return                          
  if ($y lt 2007)
  then 'WD/CR/PR' else 'REC'
```

s_1



P. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, J. Teubner. **MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine**, SIGMOD 2006.

XQuery: Graph Queries & Construction

From XPath to Hell

1. significantly **harder** than XPath
2. **composition** & **recursion**
   expensive operations
3. yet significant steps toward
   fast implementations

XQuery: Graph Queries & Construction

From XPath to Hell

1. significantly **harder** than XPath
2. **composition** & **recursion** expensive operations
3. yet significant steps toward **fast implementations**

M. Benedikt and C. Koch. **Interpreting Tree-to-Tree Queries**. ICALP 2006.

XQuery: Graph Queries & Construction

From XPath to Hell

1. significantly **harder** than XPath

2. **composition** & **recursion**
   expensive operations

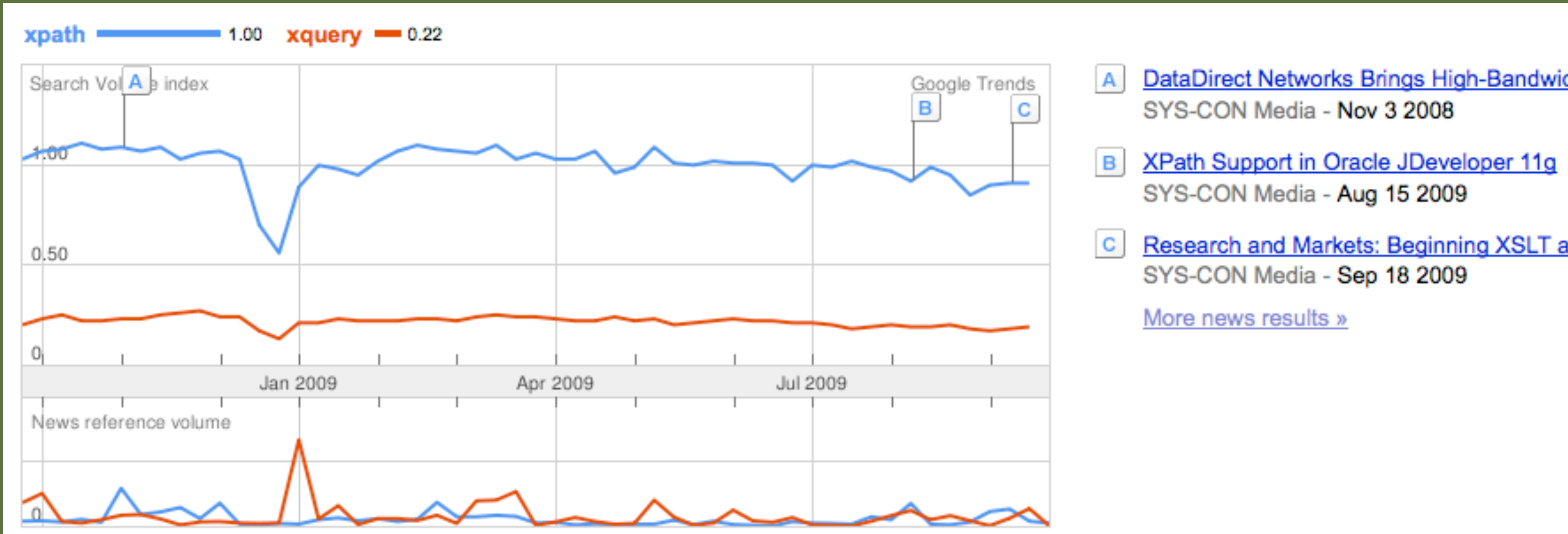
3. yet significant steps toward
   fast implementations

M. Benedikt and C. Koch. **Interpreting Tree-to-Tree Queries**. ICALP 2006.

P. Boncz, T. Grust, et al. **MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine**, SIGMOD 2006.

XQuery: Graph Queries & Construction

From XPath to Hell

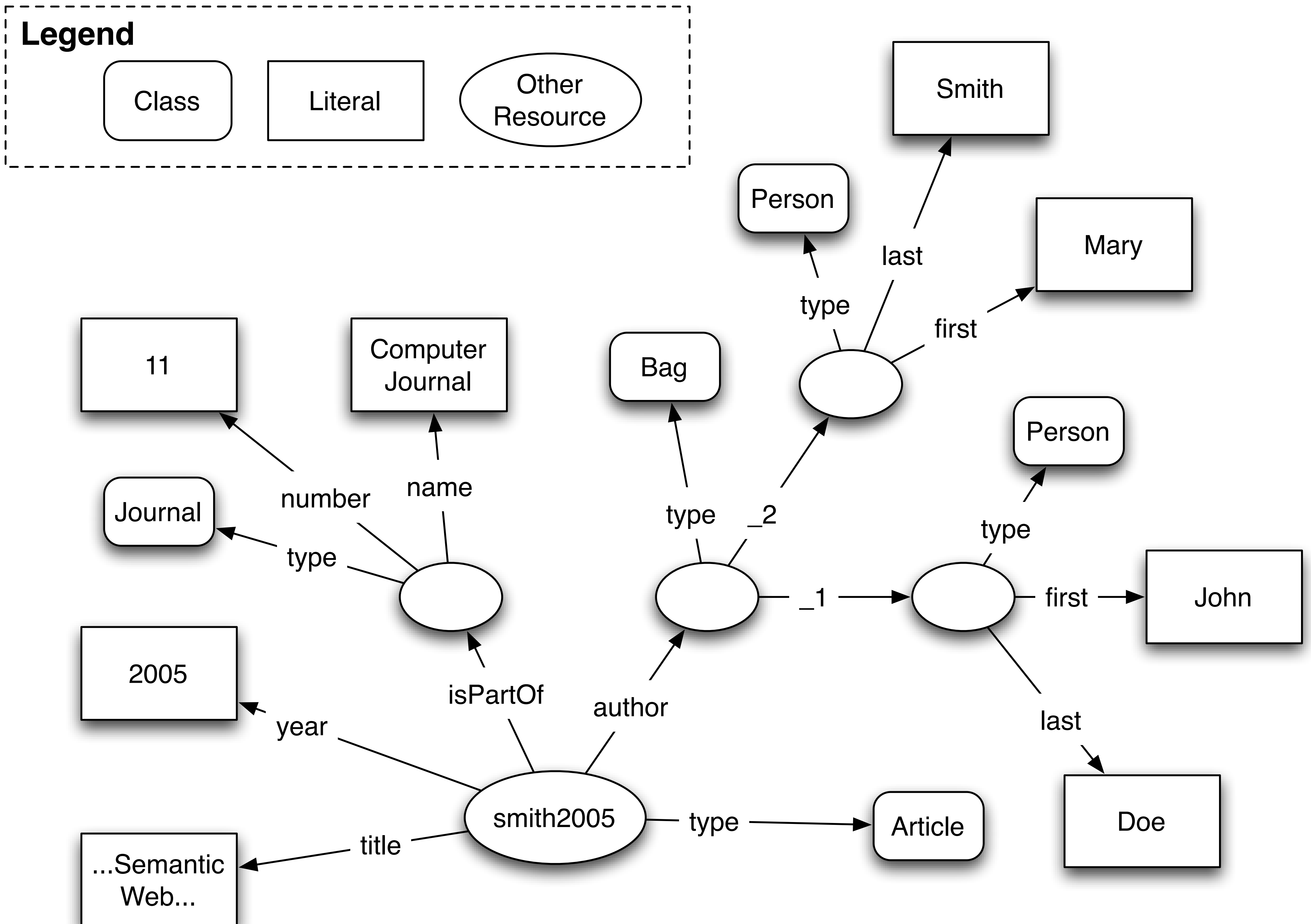


3 yet significant steps toward **fast implementations**

P. Boncz, T. Grust, et al. **MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine**, SIGMOD 2006.

1.2
RDF

Graph Data for the Web



RDF

unordered, unranked, unrooted finite graph, **edge** and node labeled, different node **kinds**

?

what's new about that? isn't that just a fancy repr. of a ternary relation?

RDF: Semantics for the Web?

What's new about RDF?

- **data model** of RDF very similar to relational
  - but: "unique" identifiers in form of URIs that can be shared on the Web
    - uniqueness only thanks to careful assignment practice (unlike GUIDs)
    - human-readable (unlike GUIDs)
  - but: existential information in form of **blank nodes**
    - like named null values in SQL, also known as **Codd tables**
  - but: implied information due to RDF/S **semantics** (and, thus, **entailment**)
    - e.g., class hierarchy and typing, domain and range of properties, axiomatic triples
- notion of **redundancy-free** or **lean** graph/answer

```
CONSTRUCT { ?j bib:hasPart ?a }
WHERE { ?a rdf:type bib:Article AND ?a bib:isPartOf ?j
          AND ?j bib:name ‘Computer Journal’ }
```

$$
\begin{aligned}
[\![(s,p,o)]\!]^D_{\mathsf{Subst}} &= \{\theta : \mathrm{dom}(\theta) = \mathsf{Vars}((s,p,o)) \wedge t\theta \in D\} \\
[\![\mathit{pattern}_1 \ \texttt{AND}\ \mathit{pattern}_2]\!]^D_{\mathsf{Subst}} &= [\![\mathit{pattern}_1]\!]^D_{\mathsf{Subst}} \bowtie [\![\mathit{pattern}_2]\!]^D_{\mathsf{Subst}} \\
[\![\mathit{pattern}_1 \ \texttt{UNION}\ \mathit{pattern}_2]\!]^D_{\mathsf{Subst}} &= [\![\mathit{pattern}_1]\!]^D_{\mathsf{Subst}} \cup [\![\mathit{pattern}_2]\!]^D_{\mathsf{Subst}} \\
[\![\mathit{pattern}_1 \ \texttt{MINUS}\ \mathit{pattern}_2]\!]^D_{\mathsf{Subst}} &= [\![\mathit{pattern}_1]\!]^D_{\mathsf{Subst}} \setminus [\![\mathit{pattern}_2]\!]^D_{\mathsf{Subst}} \\
[\![\mathit{pattern}_1 \ \texttt{OPT}\ \mathit{pattern}_2]\!]^D_{\mathsf{Subst}} &= [\![\mathit{pattern}_1]\!]^D_{\mathsf{Subst}} \ltimes [\![\mathit{pattern}_2]\!]^D_{\mathsf{Subst}} \\
[\![\mathit{pattern}\ \texttt{FILTER}\ \mathit{condition}]\!]^D_{\mathsf{Subst}} &= \{\theta \in [\![\mathit{pattern}]\!]^D_{\mathsf{Subst}} : \mathsf{Vars}(\mathit{condition}) \subset \mathrm{dom}(\theta) \\
&\qquad\qquad \wedge [\![\mathit{condition}]\!]^D_{\mathsf{Bool}}(\theta)\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathit{condition}_1 \wedge \mathit{condition}_2]\!]^D_{\mathsf{Bool}}(\theta) &= [\![\mathit{condition}_1]\!]^D_{\mathsf{Bool}}(\theta) \wedge [\![\mathit{condition}_2]\!]^D_{\mathsf{Bool}}(\theta) \\
[\![\mathit{condition}_1 \vee \mathit{condition}_2]\!]^D_{\mathsf{Bool}}(\theta) &= [\![\mathit{condition}_1]\!]^D_{\mathsf{Bool}}(\theta) \vee [\![\mathit{condition}_2]\!]^D_{\mathsf{Bool}}(\theta) \\
[\![\neg \mathit{condition}]\!]^D_{\mathsf{Bool}}(\theta) &= \neg [\![\mathit{condition}]\!]^D_{\mathsf{Bool}}(\theta) \\
[\![\texttt{BOUND}(?v)]\!]^D_{\mathsf{Bool}}(\theta) &= v\theta \neq \mathbf{nil} \\
[\![\texttt{isLITERAL}(?v)]\!]^D_{\mathsf{Bool}}(\theta) &= v\theta \in L \\
[\![\texttt{isIRI}(?v)]\!]^D_{\mathsf{Bool}}(\theta) &= v\theta \in I \\
[\![\texttt{isBLANK}(?v)]\!]^D_{\mathsf{Bool}}(\theta) &= v\theta \in B \\
[\![?v = \mathit{literal}]\!]^D_{\mathsf{Bool}}(\theta) &= v\theta = \mathit{literal} \\
[\![?u = ?v]\!]^D_{\mathsf{Bool}}(\theta) &= u\theta = v\theta \wedge u\theta \neq \mathbf{nil}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathit{triple}]\!]^D_{\mathsf{Graph}}(\theta) &= \mathit{triple}\theta \ \text{ if } \forall v \in \mathsf{Vars}(\mathit{triple}) : v\theta \neq \mathbf{nil},\ \top \ \text{ otherwise} \\
[\![\mathit{template}_1 \ \texttt{AND}\ \mathit{template}_2]\!]^D_{\mathsf{Graph}}(\theta) &= [\![\mathit{template}_1]\!]^D_{\mathsf{Graph}}(\theta) \cup [\![\mathit{template}_2]\!]^D_{\mathsf{Graph}}(\theta) \\
[\![\texttt{CONSTRUCT}\ t\ \texttt{WHERE}\ p]\!]^D &= \bigcup_{\theta \in [\![P]\!]^D_{\mathsf{Subst}}} [\![\mathsf{std}(t)]\!]^D_{\mathsf{Graph}}(\theta) \\
[\![\texttt{SELECT}\ V\ \texttt{WHERE}\ p]\!]^D &= \pi_V([\![P]\!]^D_{\mathsf{Subst}})
\end{aligned}
$$

RDF: Semantics for the Web?

Simple RDF QL?

- **SPARQL: S**imple **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
- really **simple**?
  - same expressiveness as **full** relational algebra, **PSPACE**-complete
  - but: **no composition**, **no order** → simpler than full SQL or XQuery
- really an **RDF** query language?
  - blank node **construction** only limited (no *quantifier alternation*)
  - talks *vaguely* about extension to entailment regimes
    - but no support in SPARQL as defined
  - no support for **lean** answers (justified partially by high computational cost)

RDF: Semantics for the Web?

SPARQL: Simple RDF QL?

- **Complexity:**
  - SPARQL with only `AND`, `FILTER`, and `UNION`: NP-complete
    - i.e., no computationally interesting subset of **full** relational SPJU queries
  - full SPARQL: PSPACE-complete
    - again no computationally interesting subset of **full** first-order queries
    - why? due to negation “hidden” in `OPTIONAL`
      - reduction from 3SAT using `isBound` to encode negation
- lacks completeness w.r.t. **RDF transformations**:
  - relational algebra: can express **all** PSPACE-transformations on relations
  - SPARQL: fails at the same for RDF

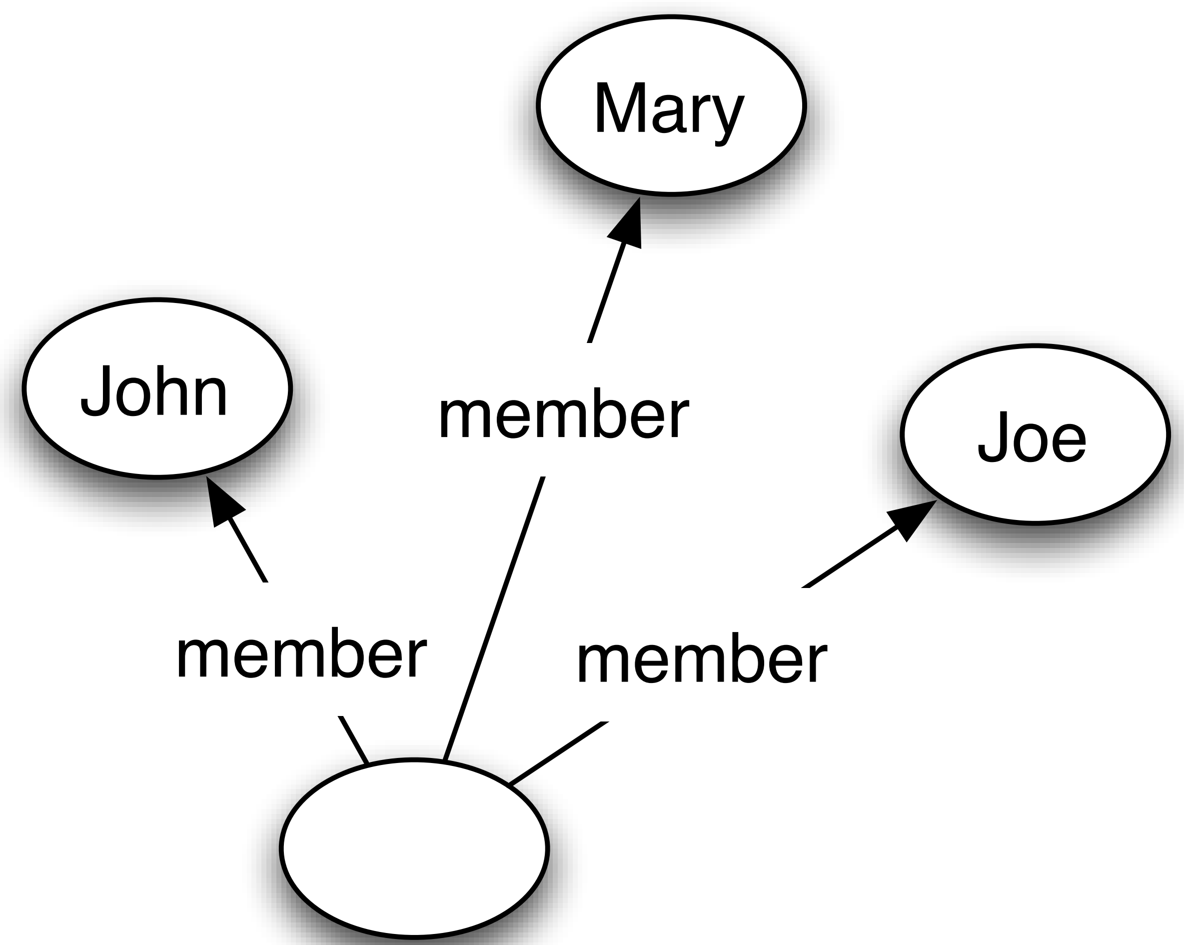
RDF: Semantics for the Web?

SPARQL: Simple RDF QL?

J. Perez, M. Arenas, and C. Gutierrez. **Semantics and Complexity of SPARQL**. ISWC 2006

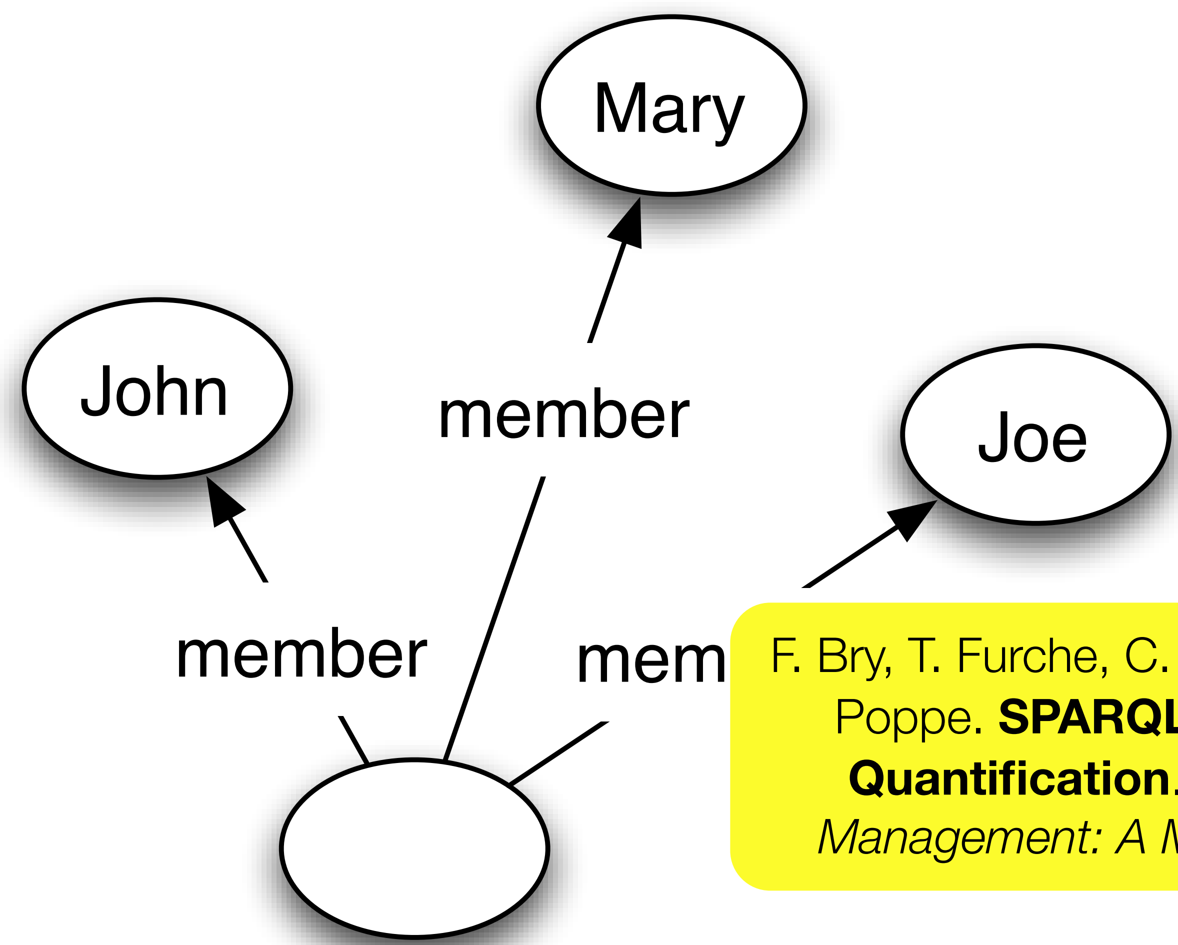
- **Complexity:**
  - SPARQL with only `AND`, `FILTER`, and `UNION`: NP-complete
    - i.e., no computationally interesting subset of **full** relational SPJU queries
  - full SPARQL: PSPACE-complete
    - again no computationally interesting subset of **full** first-order queries
    - why? due to negation "hidden" in `OPTIONAL`
      - reduction from 3SAT using `isBound` to encode negation
- lacks completeness w.r.t. **RDF transformations**:
  - relational algebra: can express **all** PSPACE-transformations on relations
  - SPARQL: fails at the same for RDF

Select all persons and return **one** blank node connected to all these persons using “member”



Not expressible in SPARQL
Why not? no **quantifier alternation**

Select all persons and return **one** blank node connected to all these persons using “member”



F. Bry, T. Furche, C. Ley, B. Linse, B. Marnette, and O. Poppe. **SPARQLog: SPARQL with Rules and Quantification**. In: *Semantic Web Information Management: A Model-based Perspective*, 2009.

Not expressible in SPARQL
Why not? no **quantifier alternation**

SPARQL

limitations of blank node construction

SPARQL: Future

- doesn't hit a **sweet spot** (like XPath)
  - at least from a formal, database perspective
    - contributions from database community **very** rare
- yet a **significant** improvement over most previous RDF QLs
- already forms **basis** for future QL research on RDF
  - rule extensions for RDF
    - "From SPARQL to Rules" (Polleres, WWW 2007), Networked Graphs (Schenk et al, WWW 2008)
    - no or limited blank node support
    - but: with rules, restricted quantification as in SPARQL ($\exists\forall$) as expressive as unrestricted quantifier alternation

RDF: Semantics for the Web?

SPARQL: Future

- doesn't hit a **sweet spot** (like XPath)
  - at least from a formal, database perspective
    - contributions from database community **very** rare
- yet a **significant** improvement over most previous RDF QLs
- already forms **basis** for future QL research on RDF
  - rule extensions for RDF
    - "From SPARQL to Rules" (Polleres, WWW 2007), Networked Graphs (Schenk et al, WWW 2008)
    - no or limited blank node support
    - but: with rules, restricted quantification as in SPARQL (∃∀) as expressive as unrestricted quantifier alternation

F. Bry, T. Furche, C. Ley, B. Linse, B. Marnette, and O. Poppe. **SPARQLog: SPARQL with Rules and Quantification**. In: *Semantic Web Information Management: A Model-based Perspective*, 2009.

RDF: Semantics for the Web?

SPARQL: Summary

1. **syntax** for first-order queries on RDF
2. **foundation** for research but little innovation in the language itself
3. lackluster support for RDF specifics

RDF: Semantics for the Web?

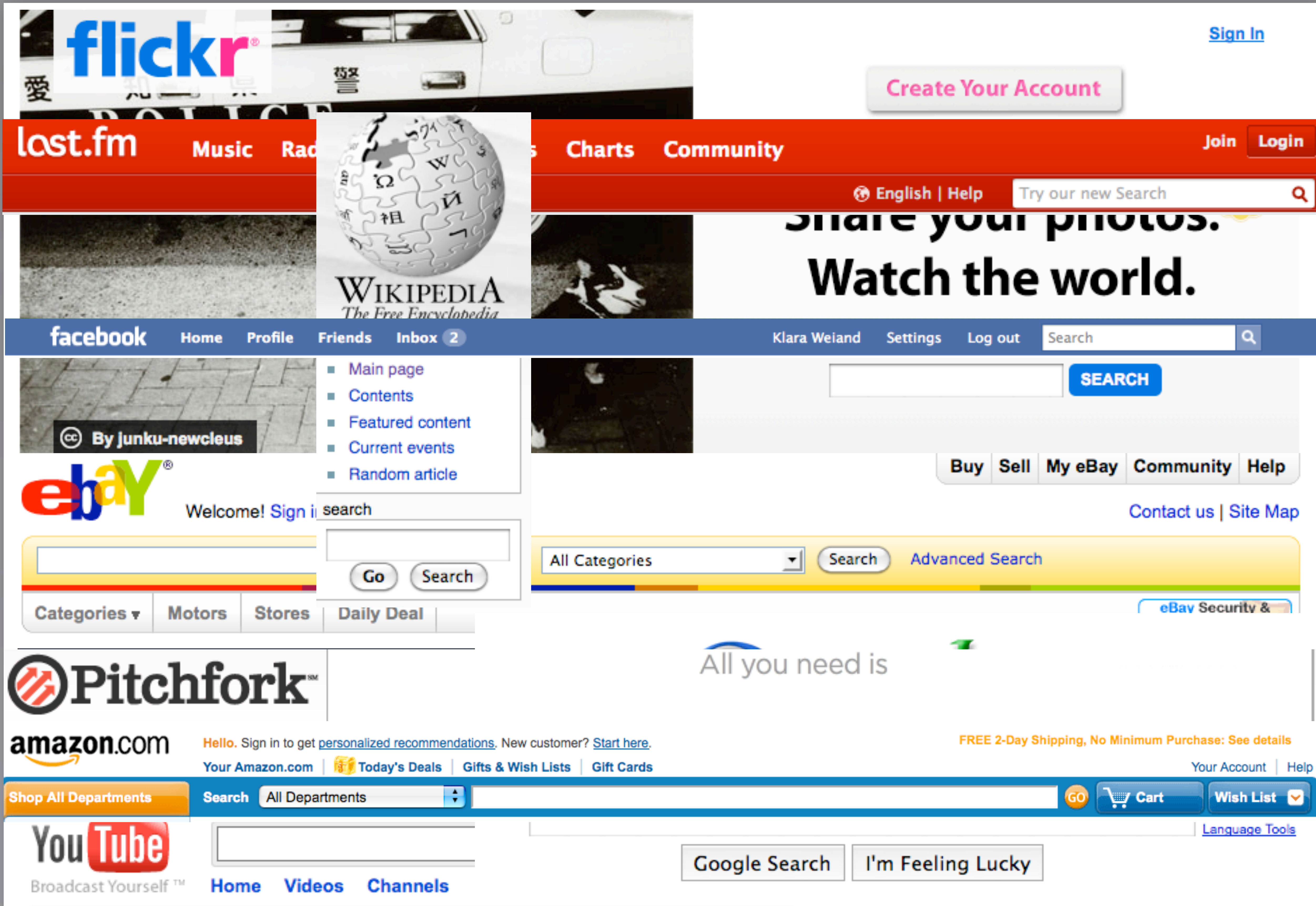
SPARQL: Summary

1. **syntax** for first-order queries on RDF
2. **foundation** for research but little innovation in the language itself
3. lackluster support for RDF specifics

J. Perez, M. Arenas, and C. Gutierrez. **Semantics and Complexity of SPARQL**. ISWC 2006

Part 2
Keyword Queries

Classification & main issues


flickr
Sign In
Create Your Account
Share your photos.
Watch the world.
SEARCH
By junku-newcleus
last.fm
Music
Charts
Community
Join
Login
English | Help
Try our new Search
WIKIPEDIA
The Free Encyclopedia
facebook
Home
Profile
Friends
Inbox 2
Klara Weiand
Settings
Log out
Search
Main page
Contents
Featured content
Current events
Random article
search
Go
Search
ebY
Welcome! Sign i
Buy
Sell
My eBay
Community
Help
Contact us | Site Map
All Categories
Search
Advanced Search
Categories
Motors
Stores
Daily Deal
eBay Security &
Pitchfork
All you need is
amazon.com
Hello. Sign in to get personalized recommendations. New customer? Start here.
FREE 2-Day Shipping, No Minimum Purchase: See details
Your Amazon.com
Today's Deals
Gifts & Wish Lists
Gift Cards
Your Account
Help
Shop All Departments
Search
All Departments
GO
Cart
Wish List
YouTube
Broadcast Yourself ™
Home
Videos
Channels
Language Tools
Google Search
I'm Feeling Lucky

Queries as Keywords

Main Characteristics

- Queries are (mostly) unstructured **bags of words**
- Used in general purpose web search engines
  - but also elsewhere (Amazon, Facebook,...)
- Implicit **conjunctive semantics** (with limitations)
- Often combined with **IR techniques**
  - ranking, fuzzy matching
- Research focuses on
  - application to **semi-structured** data
  - general structured data (Web **objects** & tables, **relations**)

Why **Keyword** Qs for Structured Data

- **Success** in other areas
- Users are already **familiarized** with the paradigm
- Allow **casual users** to query structured data without having deep knowledge of
  - the query language
  - the structure & schema of the data
- Enable querying of **heterogeneous** data

Classification of Keyword QLs

- **Data type**
  - XML
  - RDF
- **Implementation**
  - stand-alone systems
  - translation to conventional query language
  - keyword-enhanced query languages

Queries as Keywords

Classification of Keyword QLs

- Complexity of **atomic queries**
  - keyword-only
  - label-keyword
  - keyword-enhanced
- Querying of elements
  - Values
  - Node labels
  - Edge labels

Queries as Keywords

XML Keyword QLs

	Stand-alone	Enhancement
Keyword-only	9	0
Label-keyword	2	0
Keyword-enhanced	--	3

Queries as Keywords

K. Weiand, T. Furche, and F. Bry.
Quo vadis, web queries?
In Web4Web, 2008.

XML Keyword QLs

	Stand-alone	Enhancement
Keyword-only	9	0
Label-keyword	2	0
Keyword-enhanced	--	3

Queries as Keywords

RDF Keyword QLs

	Stand-alone	Translation
Keyword-only	2	2
Label-keyword	0	1
Keyword-enhanced	--	--

Queries as Keywords

K. Weiand, T. Furche, and F. Bry.
Quo vadis, web queries?
In Web4Web, 2008.

RDF Keyword QLs

	Stand-alone	Translation
Keyword-only	2	2
Label-keyword	0	1
Keyword-enhanced	--	--

Queries as Keywords

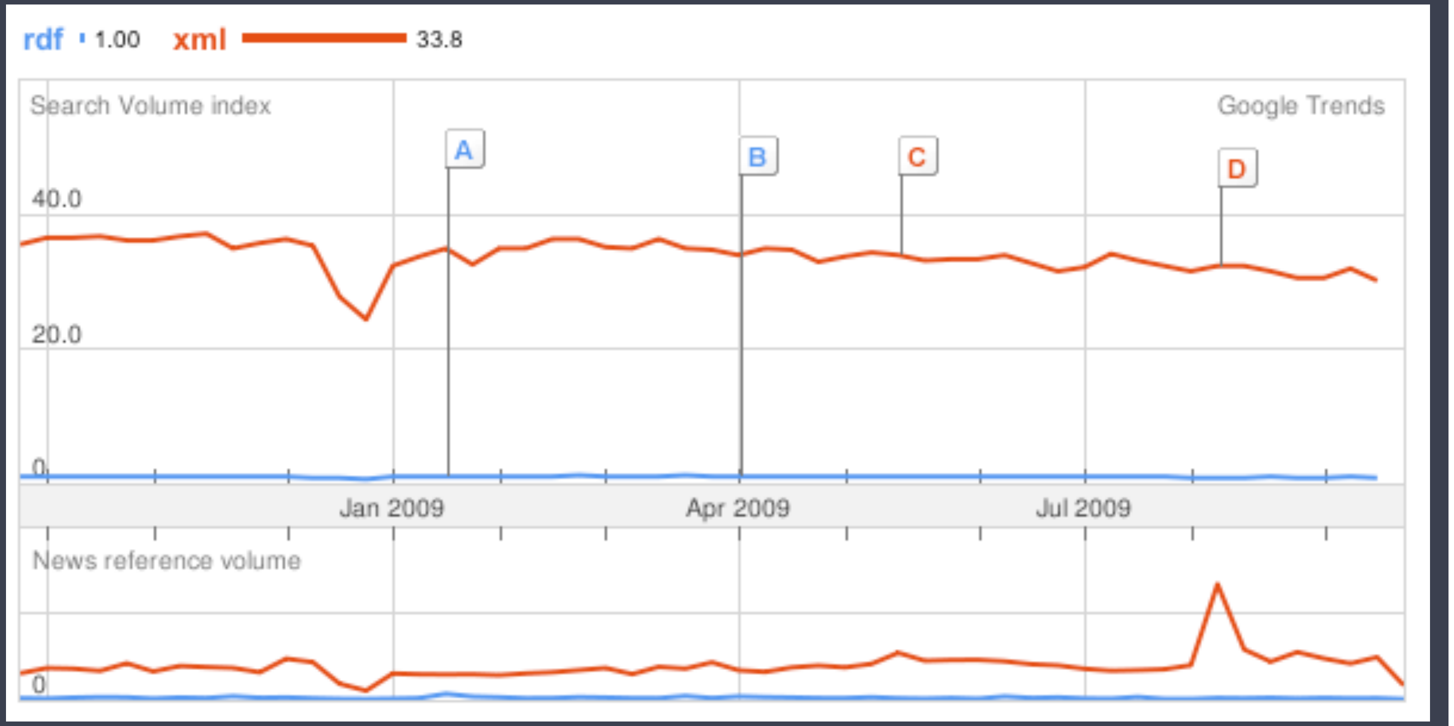
XML Keyword QLs More Popular

- **Time**
  - first XML keyword query languages are as old as RDF
- **Familiarity with XML**
- **Complexity of RDF**
  - Graph-shaped
  - Labeled Edges
  - Blank nodes

Queries as Keywords

XML Keyword QLs More Popular

- **Time**
  - first XML keyword query languages are as old as RDF
- **Familiarity with XML**
- **Complexity of RDF**
  - Graph-shaped
  - Labeled Edges
  - Blank nodes



Queries as Keywords

Focus **Issues**

1. Grouping keyword matches
2. Determining answer representations
3. Expressive power
4. Ranking
5. Limitations

2.1 Computing Query Answers

Turning keyword matches into **answers**

Computing Query Answers

Problem Setting

- Query K yields **match lists** $L_1 \ldots L_{|K|}$
- **Answer sets** $S_1 \ldots S_m$ are constructed from L
  - may contain either
    - only one match per match list ($m = |K|$) or
    - several ($m \geq |K|$)
- Data-centric vs. document-centric XML

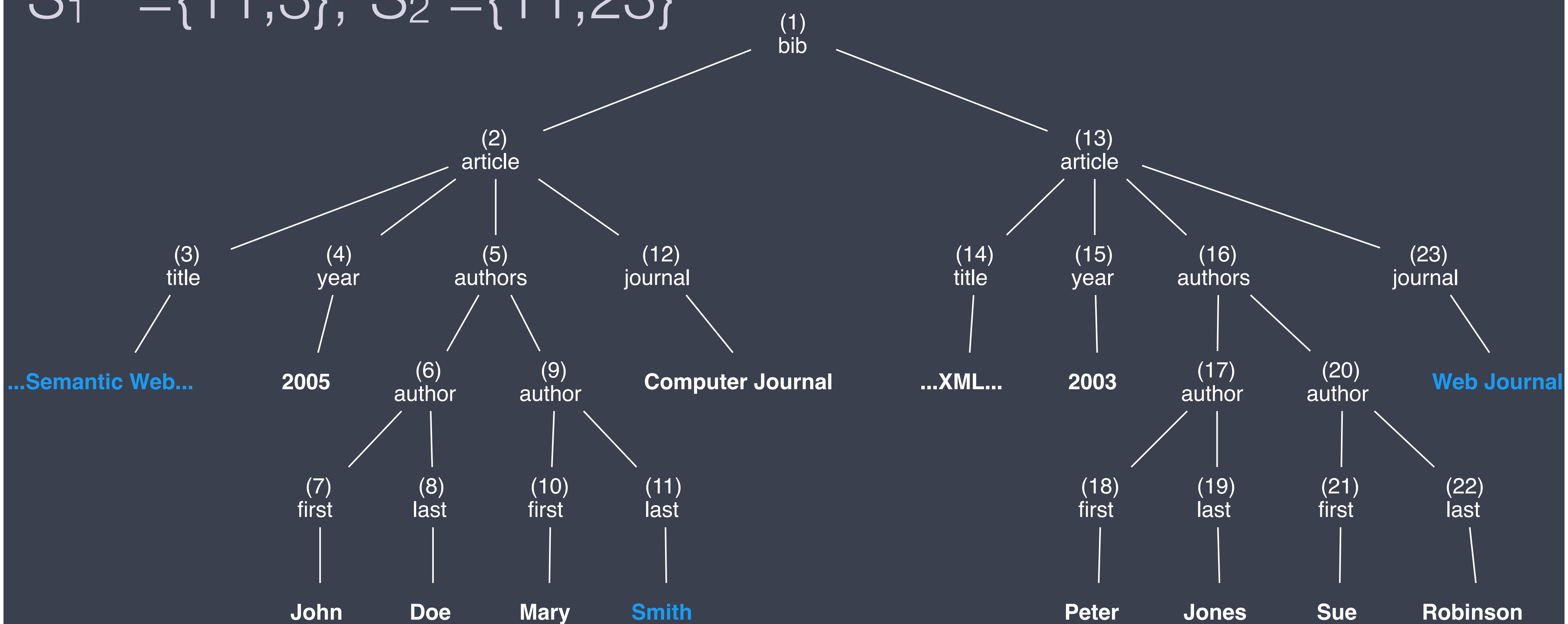
Computing Query Answers

Problem Setting

$K = \{Smith, web\}$,

$L_1 = \{11\}$, $L_2 = \{3,23\}$,

$S_1 = \{11,3\}$, $S_2 = \{11,23\}$



Computing Query Answers

Problem Setting

- **Entire XML document**
  - usually **too big** to serve as a good query answer
- **Matched nodes alone**
  - usually not **informative**
- **Meaningful results**
  - a **smaller unit** has to be **found**

Computing Query Answers

Approaches

1. Determine entities **based on the schema**
   - only keyword matches within **one entity or**
   - apply **LCA at entity-level**
   - done manually **or** using **schema partitioning** with cardinality as criterion
     - Lowest Entity Node, Minimal Information Unit, XSeek...
2. Determine entities **by connecting keyword matches**
   - Answer entity:
     - contains (at least) **one match for each** keyword

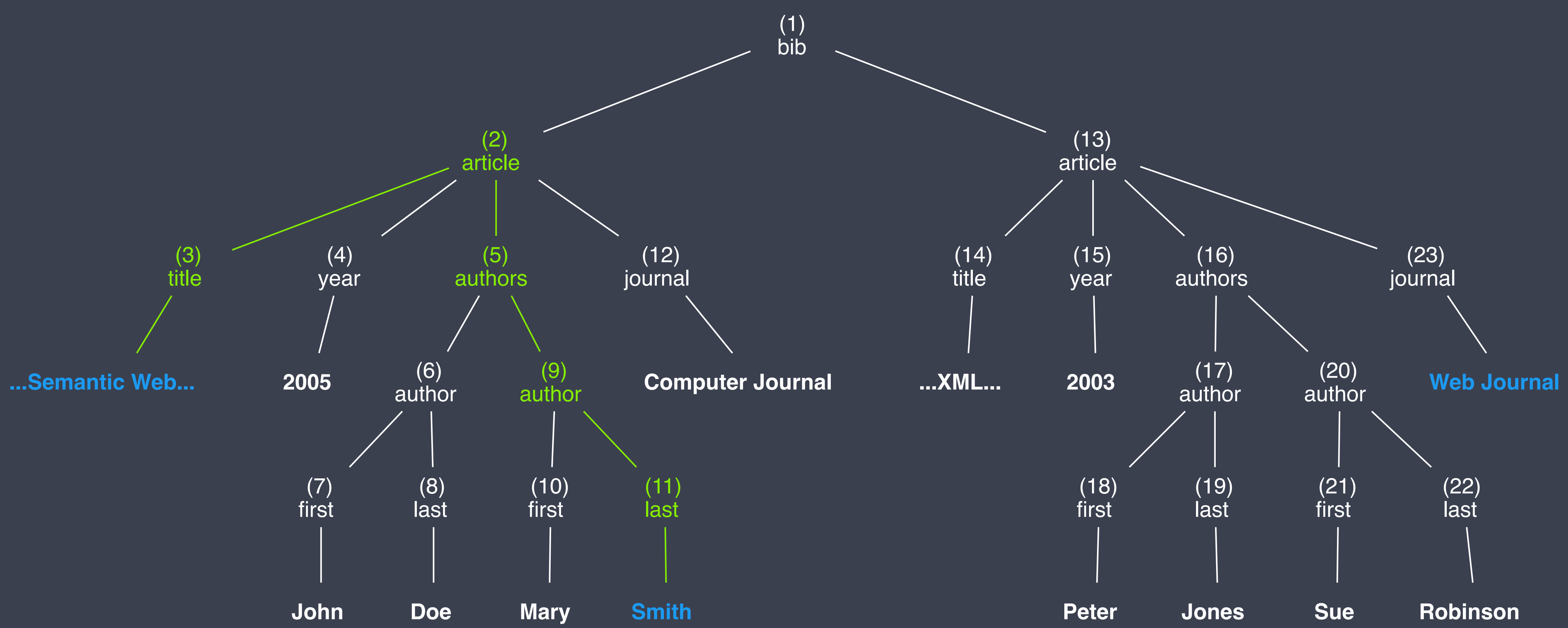
Connecting Keyword Matches

- Classic concept: **Lowest Common Ancestor** (LCA)
  - Use LCA to find the **maximally specific concept** that is **common** to the keyword matches
- LCA: Lowest node that is ancestor to all elements in an answer set

Computing Query Answers

Lowest Common Ancestor (LCA)

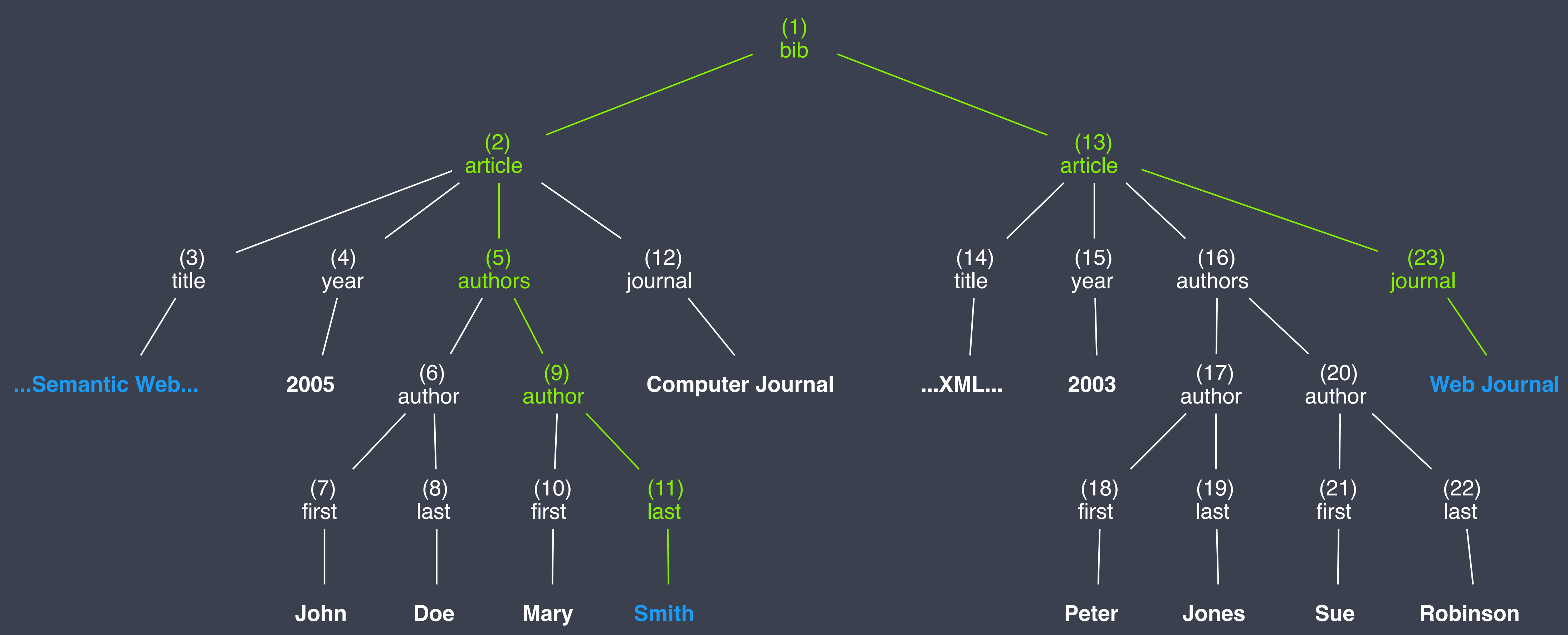
$$LCA(S_1=\{3,11\}) = 2$$



Computing Query Answers

Lowest Common Ancestor **(LCA)**

LCA(S_2={11,23}) = 1 — False positive



2.1.1

Improving LCA

Approaches to **improve LCA** for computing answer entities

Computing Query Answers

Lowest Common Ancestor **(LCA)**

- Many approaches to **improve over LCA**
  - SLCA, MLCA, Interconnection Semantics, VLCA, Amoeba Join...
- Filter LCAs to **avoid false positives**
- Result is **subset of LCA result**

Improving LCA

Overview of **Approaches**

1. Interconnection relationship
2. XKSearch: Smallest LCA
3. Meaningful LCA (MLCA)
4. Amoeba Join
5. (Valuable LCA (VLCA), Compact LCA, CVLCA)
6. (Relaxed Tightest Fragment (RTF))
7. XRank: Exclusive LCA

Improving LCA

Interconnection Relationship

- Idea: Two **different** nodes with the **same label** correspond to **different** entities of the same type.
- Two nodes are **interconnected** if,
  - on the **shortest** path between them,
  - every node **label** occurs only **once**
- Interconnection in **answer sets:**
  - *star-related*: a node in S_i interconnected with all nodes in S_i
  - *all-pairs related*: all nodes in S_i pair-wise interconnected

Improving LCA

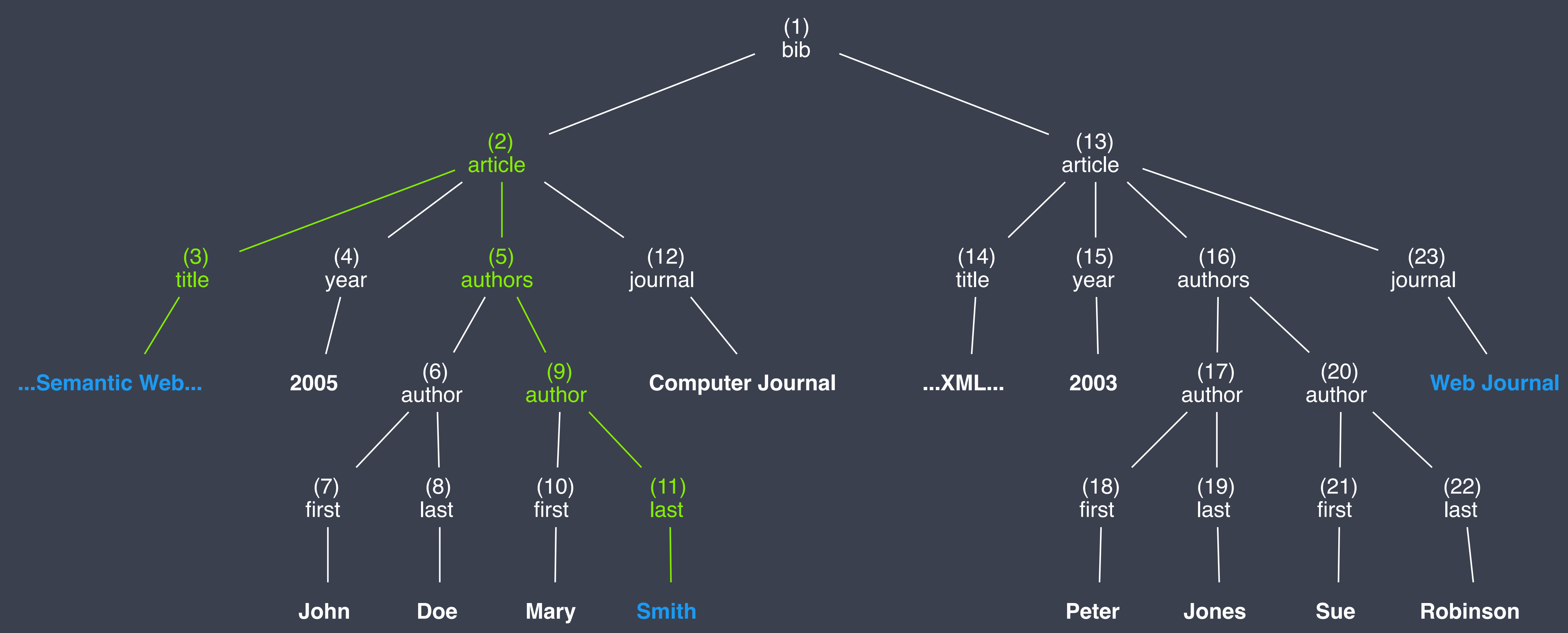
Interconnection Relationship

- Idea: Two **different** nodes with the **same label** correspond to **different** entities of the same type.
- Two nodes are **interconnected** if,
  - on the **shortest** path between them,
  - every node **label** occurs only **once**
- Interconnection in **answer sets:**
  - *star-related*: a node in S_i interconnected with all nodes in S_i
  - *all-pairs related*: all nodes in S_i pair-wise interconnected

S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. **XSearch: A Semantic Search Engine for XML**. VLDB 2003

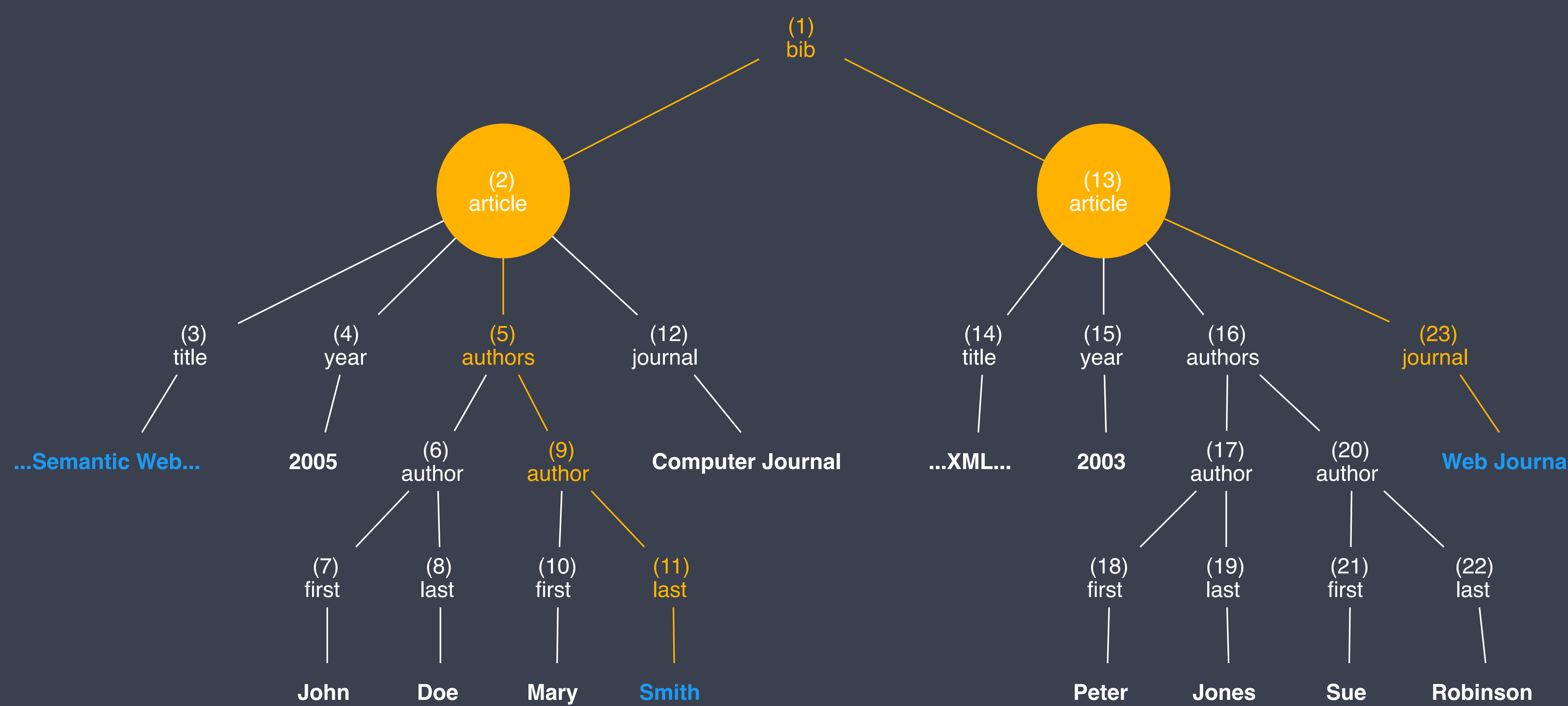
Improving LCA

Interconnection Relationship



Improving LCA

Interconnection Relationship



Improving LCA

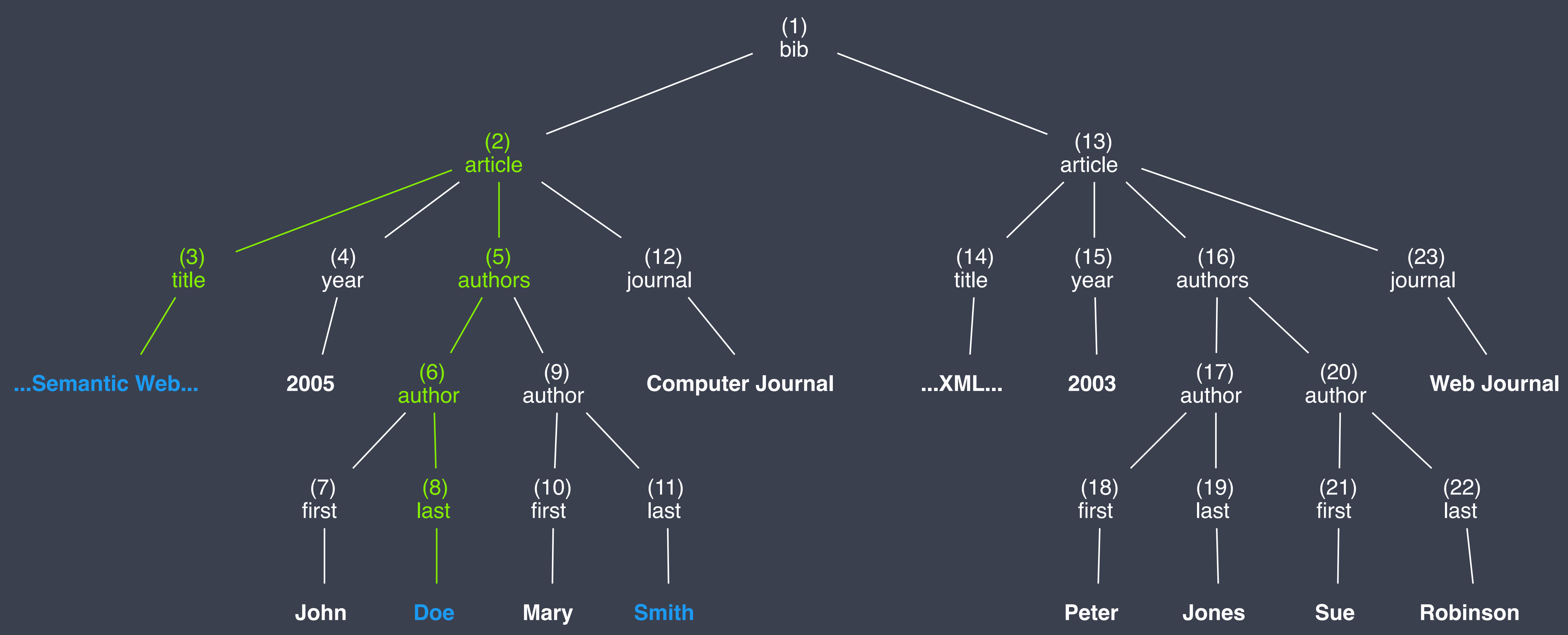
Interconnection Relationship

- For $|K| = 2$, star-related $\equiv$ all-pairs related
- Consider $S_1=\{3, 8, 11\}$

Improving LCA

Interconnection Relationship

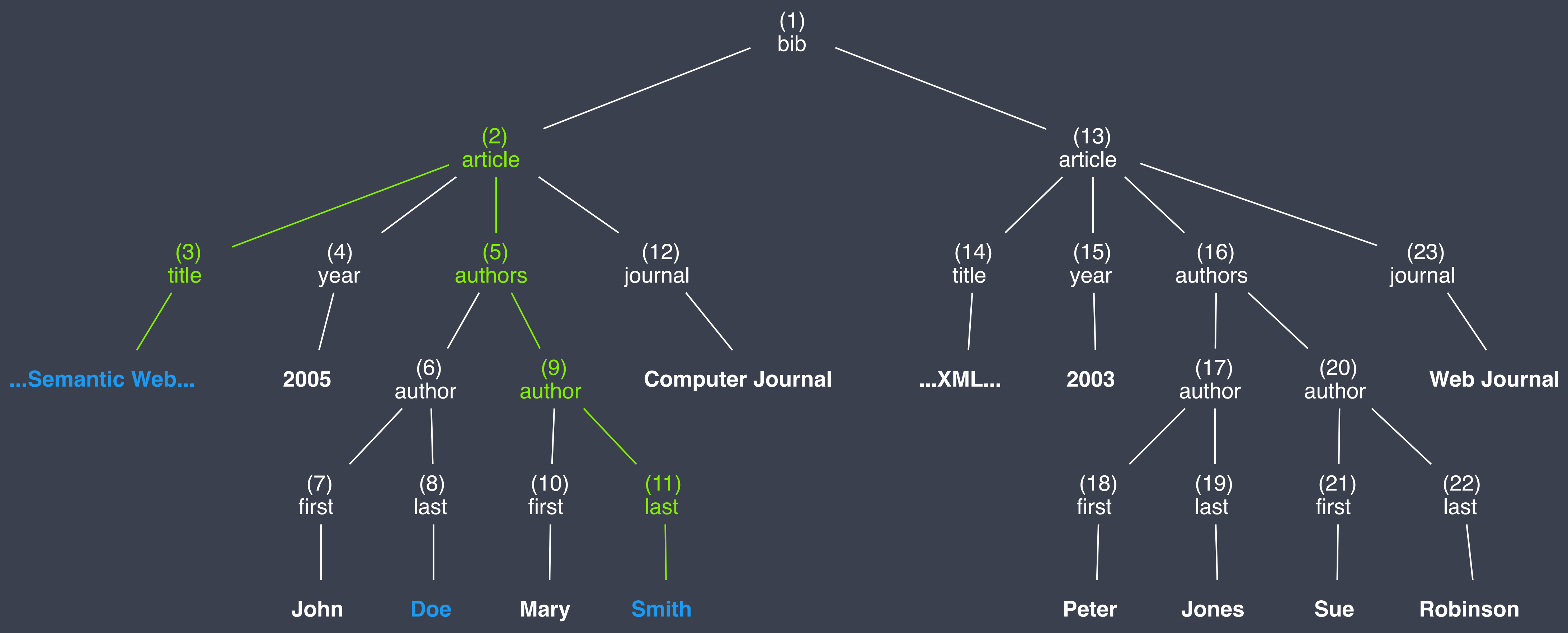
3 and 8 are interconnected



Improving LCA

Interconnection Relationship

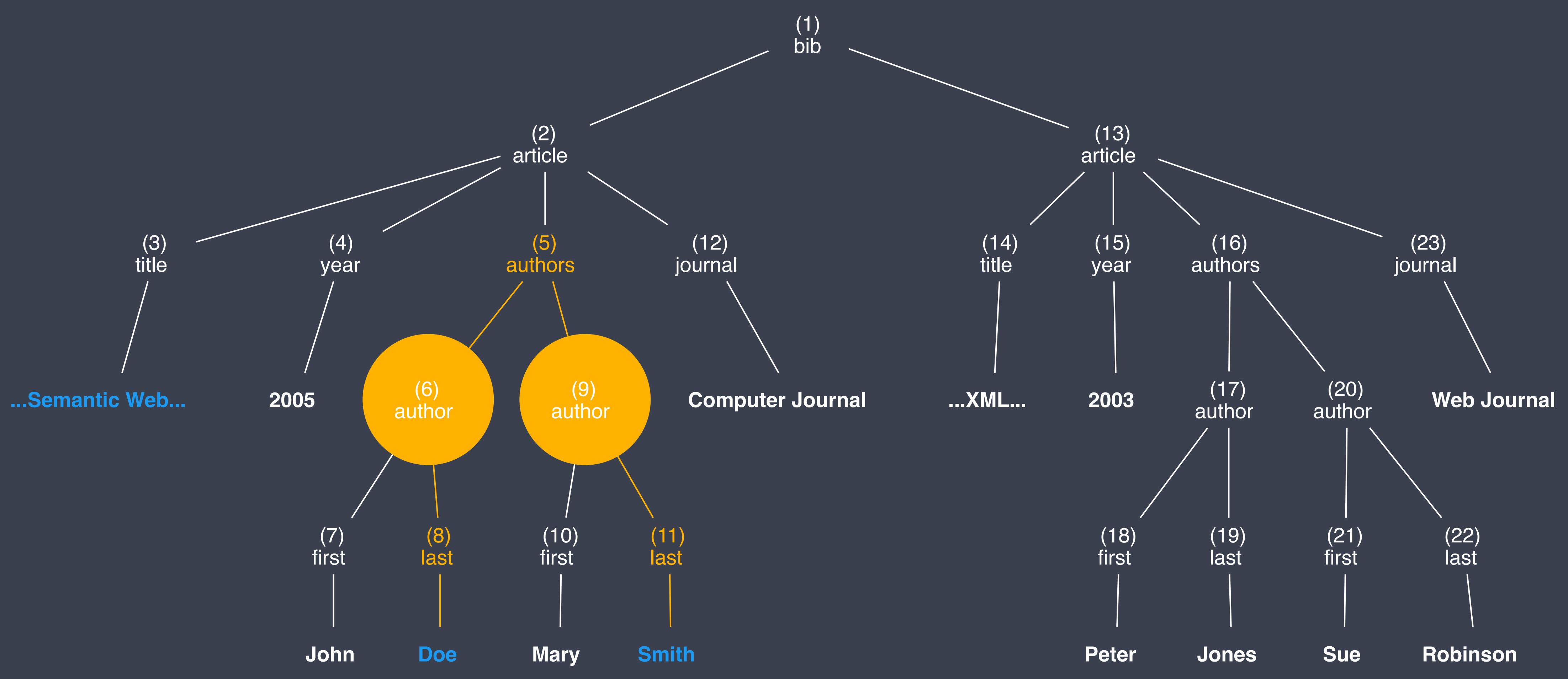
...so are 3 and 11



Improving LCA

Interconnection Relationship

...but 8 and 11 are not



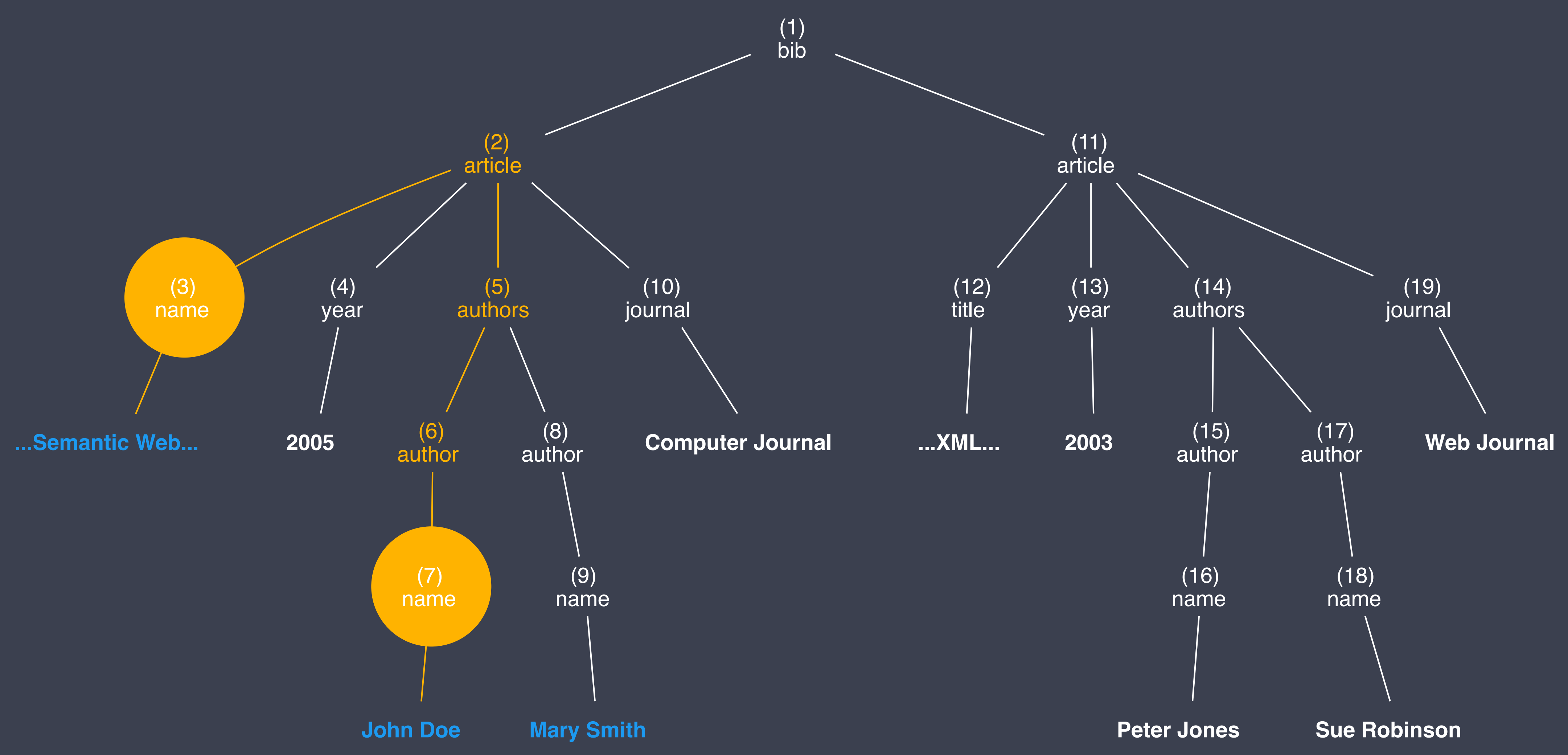
Improving LCA

Interconnection Relationship

- S1={3, 8, 11} is
  - **star-related** but
  - not **all-pairs related**
- False negative when using all-pairs relatedness
- Consider S1={3,7,9}

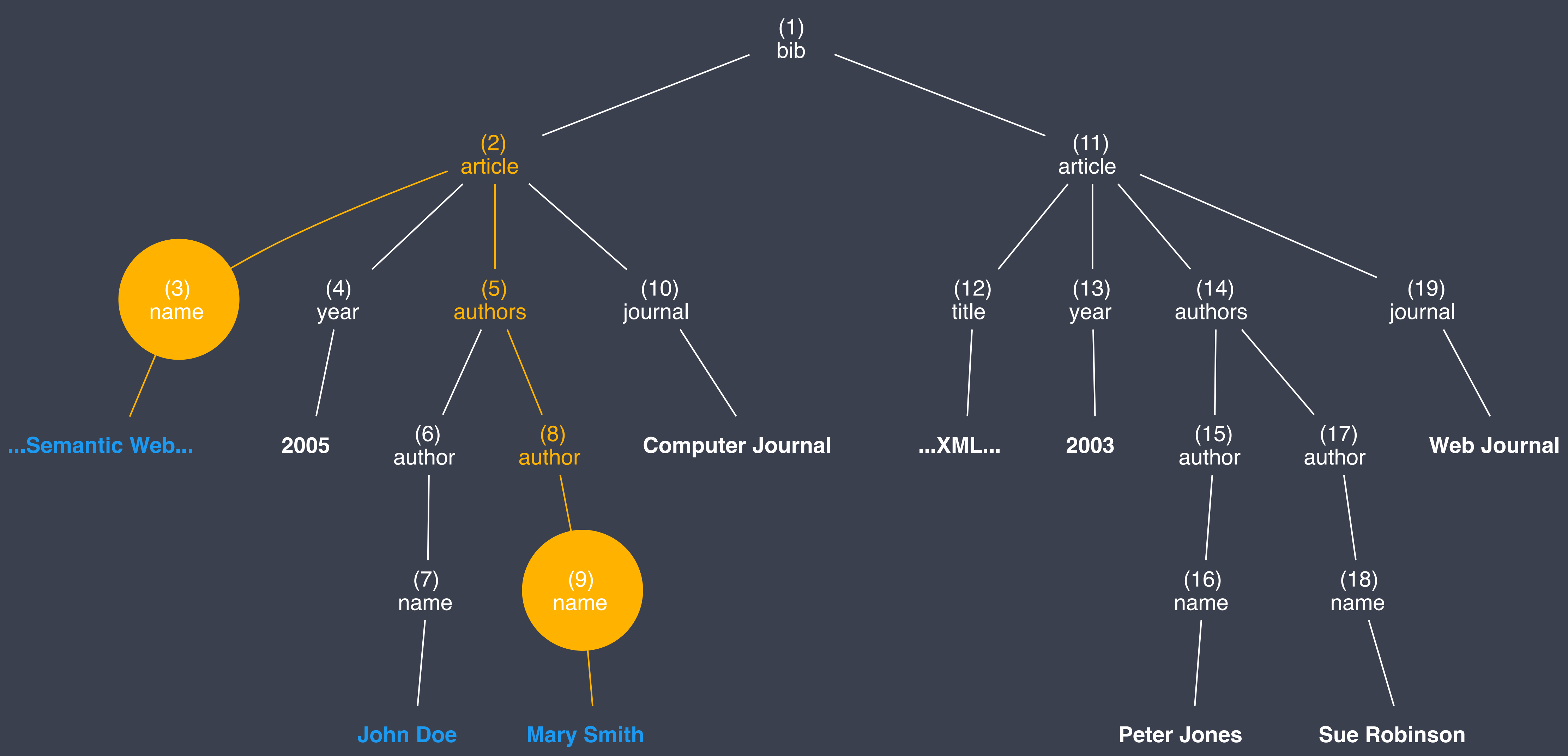
Improving LCA

Interconnection Relationship



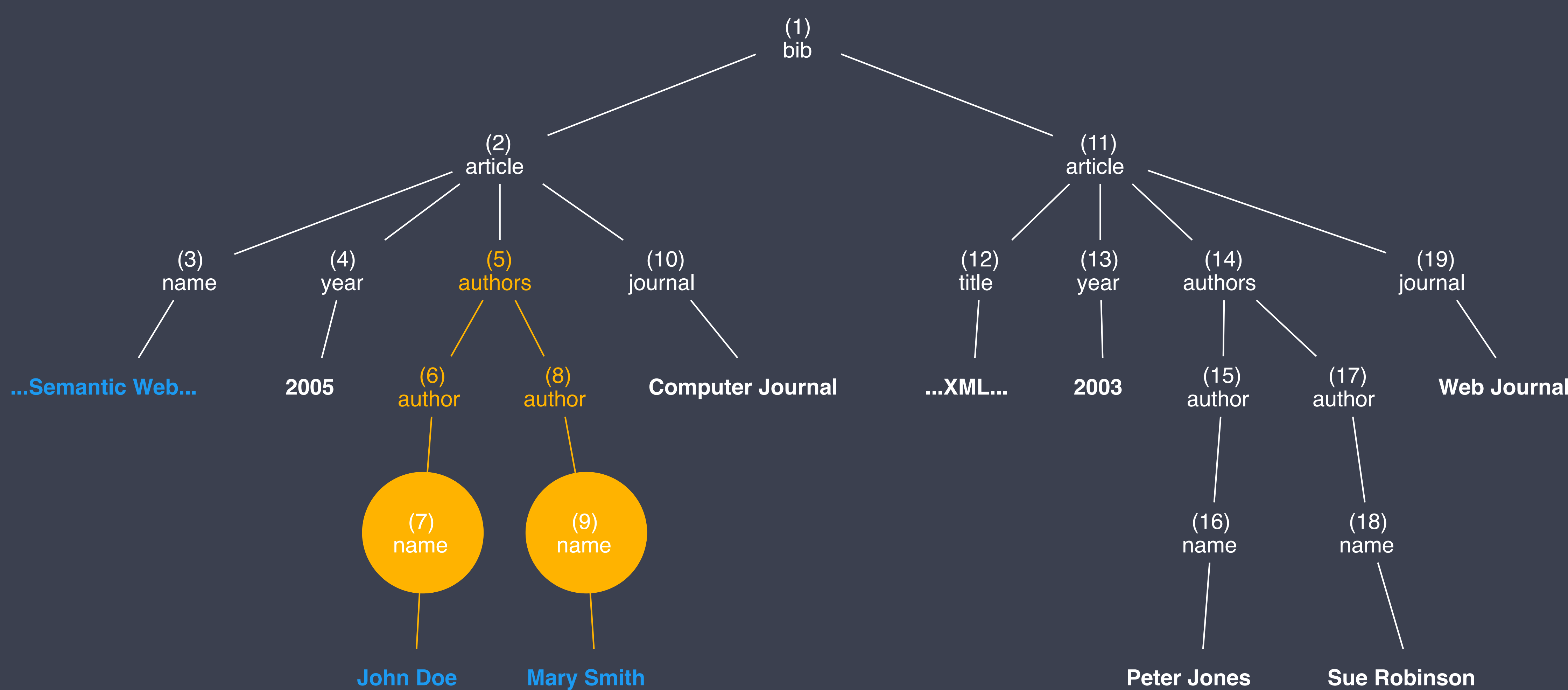
Improving LCA

Interconnection Relationship



Improving LCA

Interconnection Relationship



Improving LCA

Interconnection Relationship

- S_1={3,7,9} is a false negative in both measures
- False positives for both when node **labels** are **different** but refer to **similar concepts**
  - e.g. “article” vs. “book” vs. “text”
- False negatives when node **labels** are **identical** but refer to **different concepts**
  - e.g. the name of a person vs. the name of a journal

Improving LCA

XKSearch: Smallest LCA (SLCA)

- Idea: Enhance LCA with a **minimality** constraint
- **SLCA** nodes are LCAs which
  - **do not** have **LCA** nodes among their **descendants**
- Similar to XRank/ELCA but stricter

Improving LCA

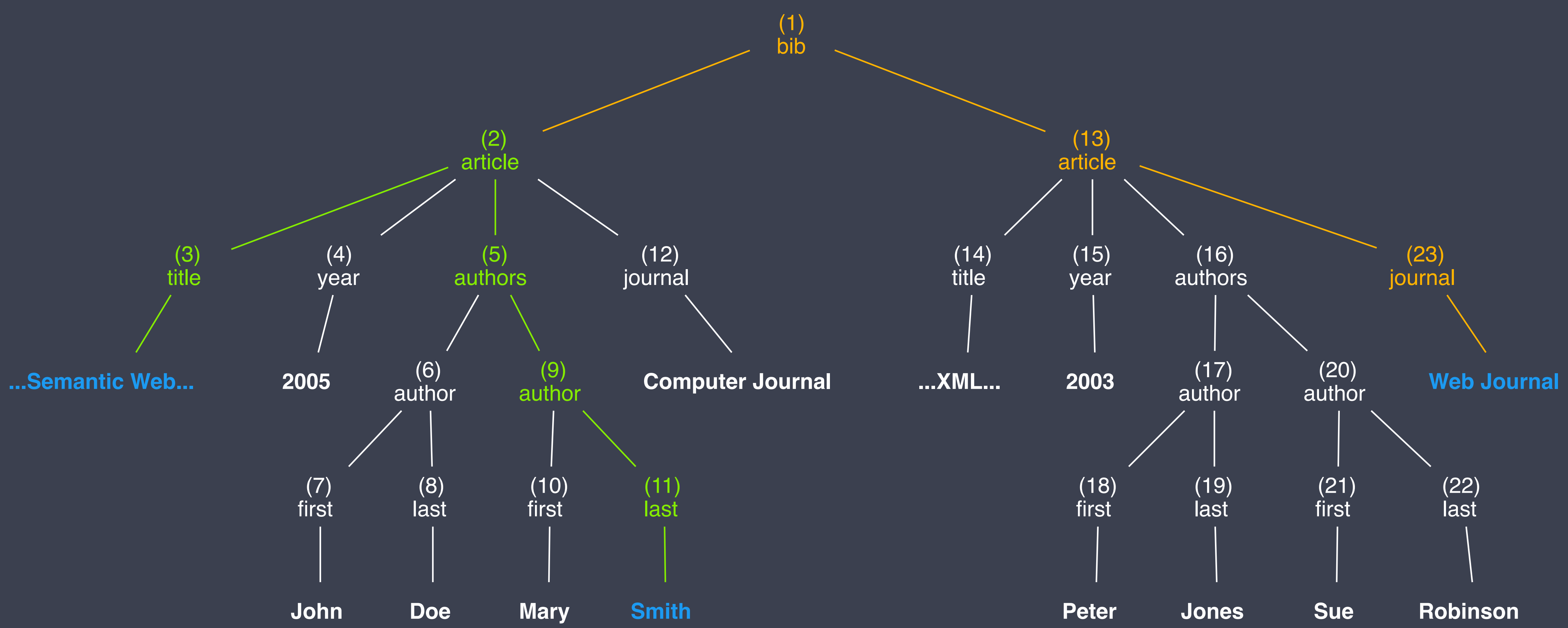
XKSearch: Smallest LCA (SLCA)

- Idea: Enhance LCA with a **minimality** constraint
- **SLCA** nodes are LCAs which
  - **do not** have **LCA** nodes among their **descendants**
- Similar to XRank/ELCA but stricter

Y. Xu and Y. Papakonstantinou. **Efficient Keyword Search for Smallest LCAs in XML Databases**. SIGMOD 2005

Improving LCA

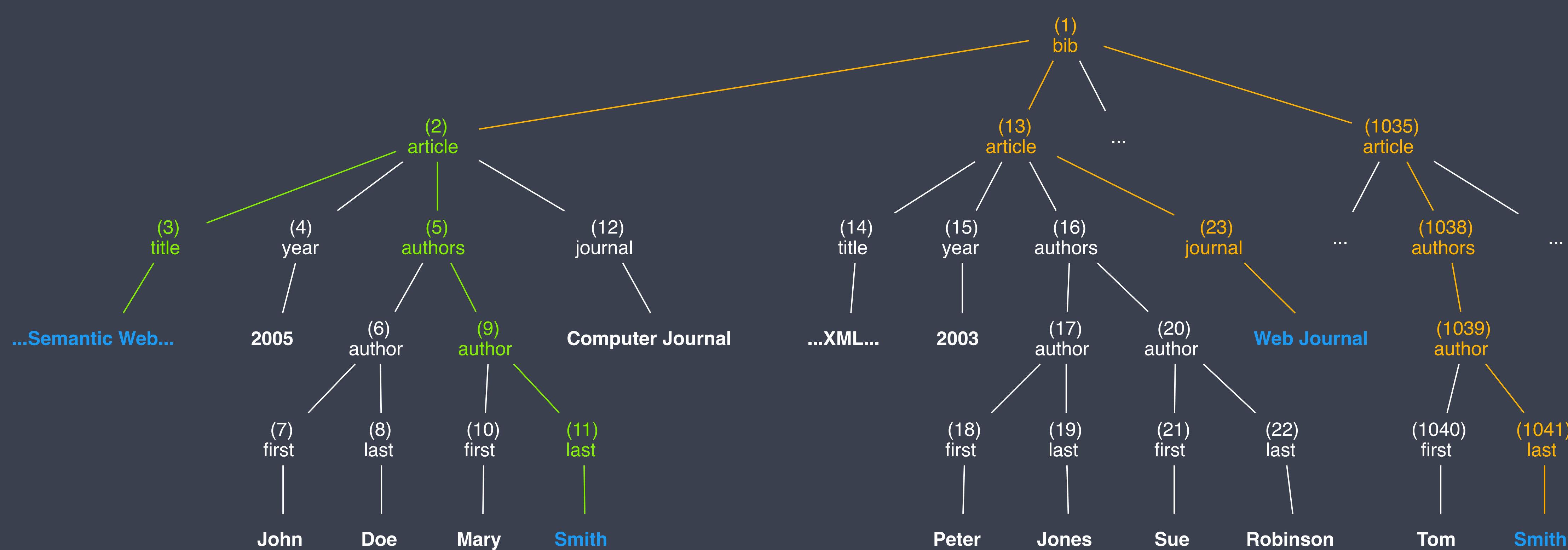
XKSearch: Smallest LCA (SLCA)



Improving LCA

XKSearch: Smallest LCA (SLCA)

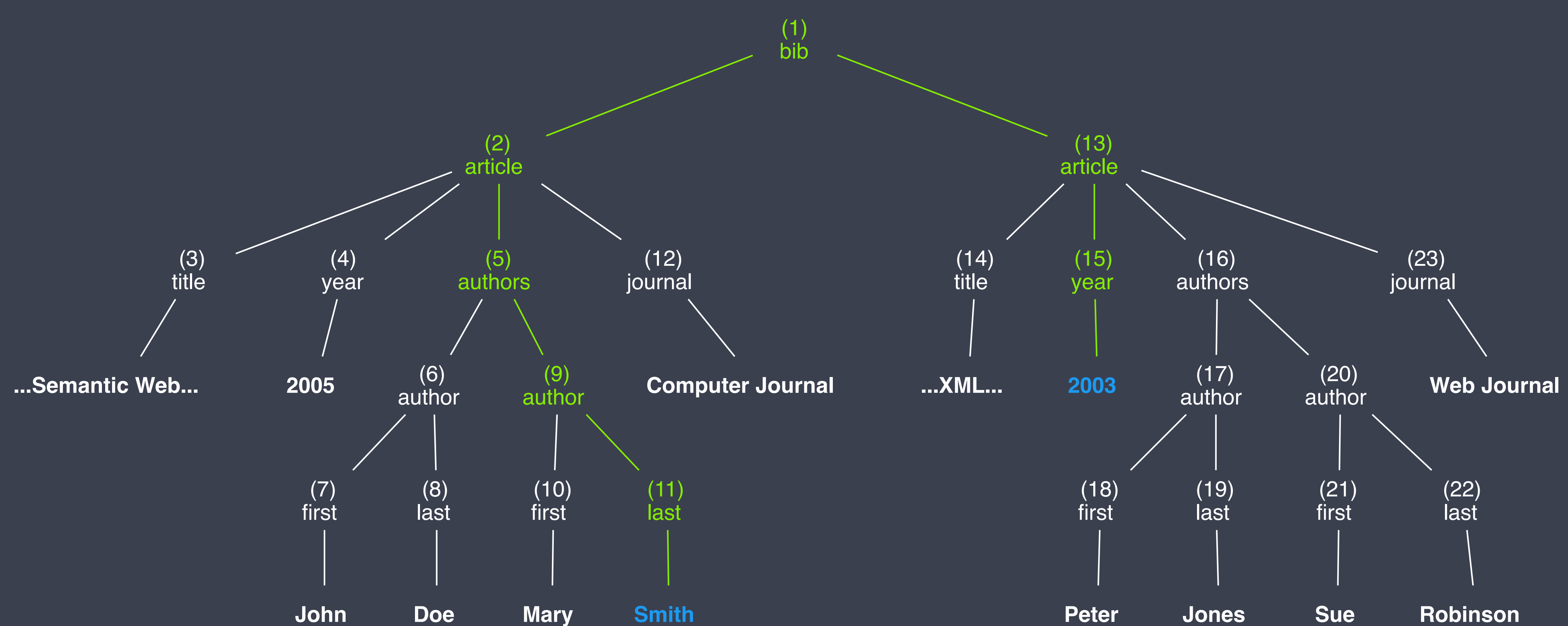
No false positive



Improving LCA

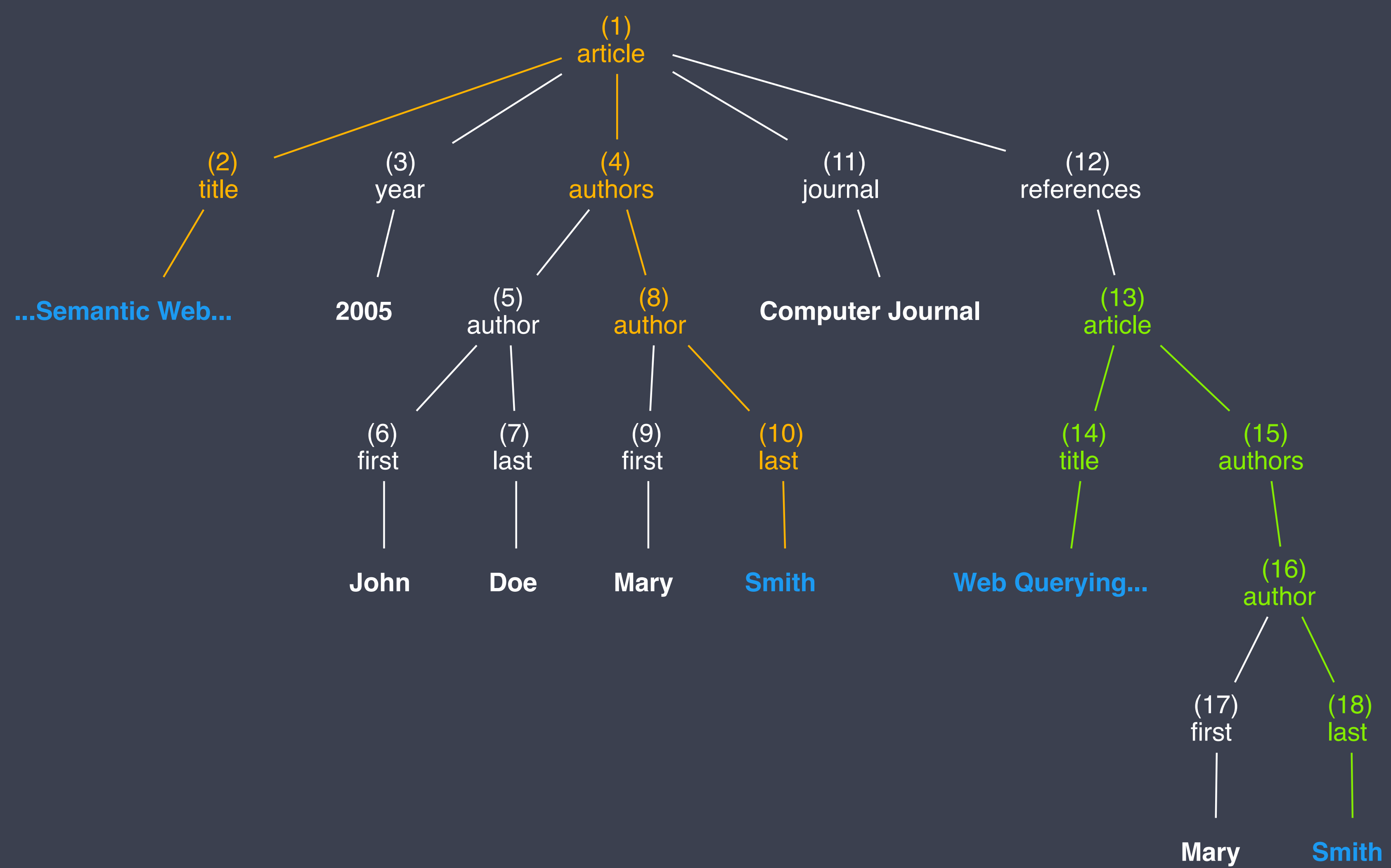
XKSearch: Smallest LCA (SLCA)

But of course...



Improving LCA

SLCA: False Negatives



Improving LCA

Meaningful LCA (MLCA)

- **MLCA:** LCA where
  - for each pair of nodes,
  - **no descendant LCA** of nodes with the **same** label exists
- Similar to SLCA but less strict since only applies when node label constraint is fulfilled
- Problems similar to SLCA and Interconnection Semantics

Improving LCA

Meaningful LCA (MLCA)

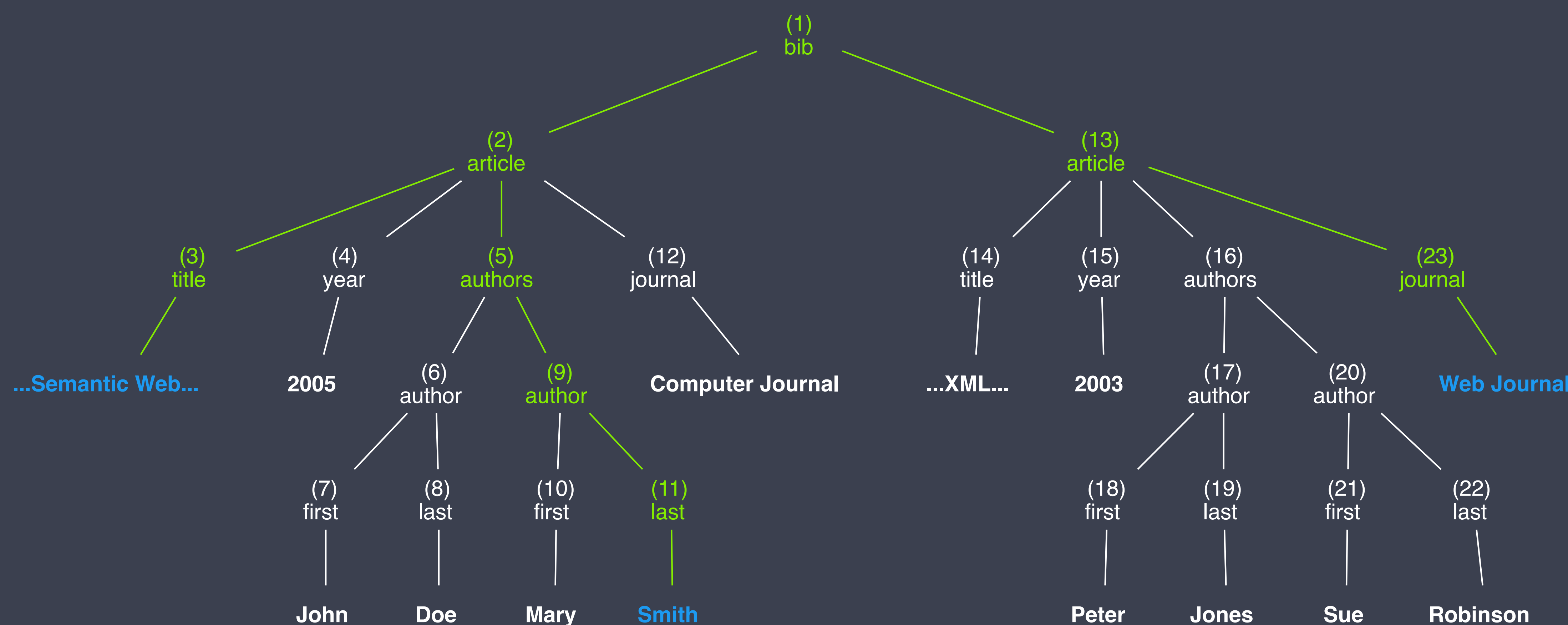
- **MLCA:** LCA where
  - for each pair of nodes,
  - **no descendant LCA** of nodes with the **same** label exists
- Similar to SLCA but less strict since only applies when node label constraint is fulfilled
- Problems similar to SLCA and Interconnection Semantics

Y. Li, C. Yu, and H. V. Jagadish. **Schema-Free XQuery**. VLDB 2004

Improving LCA

Meaningful LCA (MLCA)

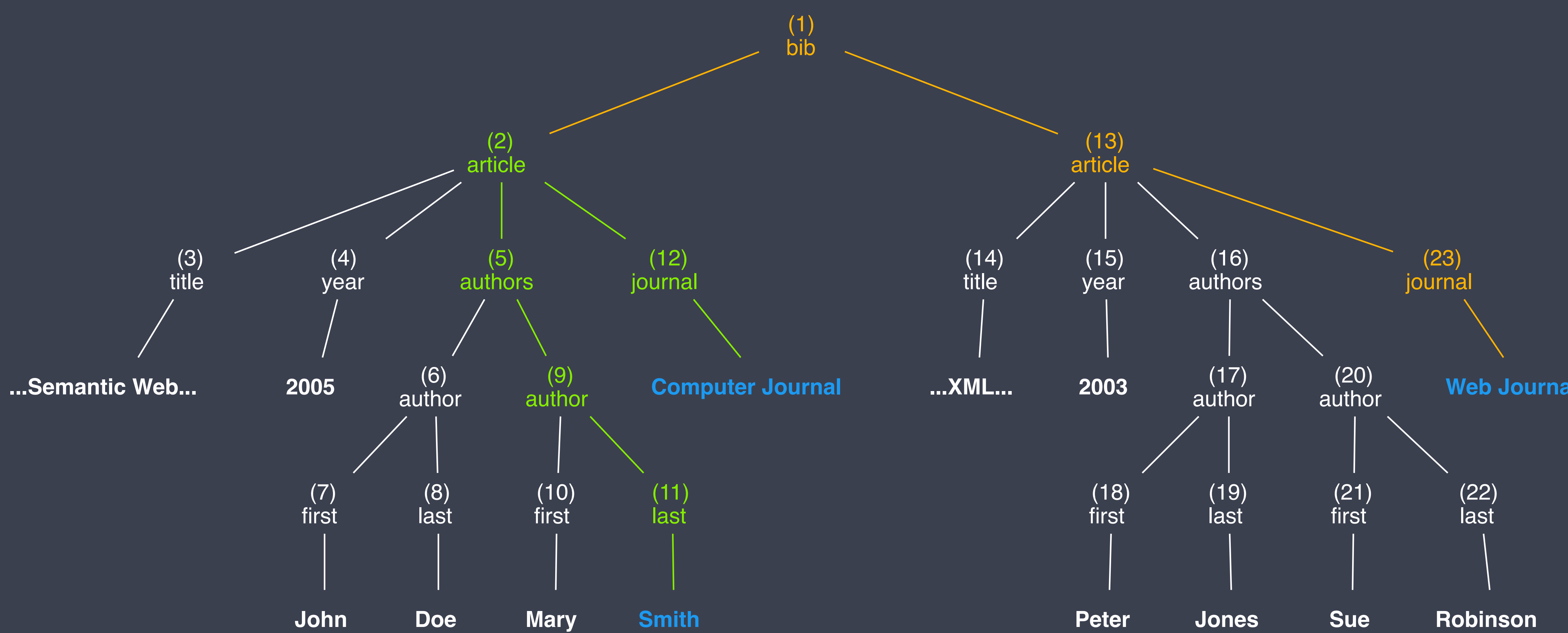
$S_1=\{3,11\}, S_2=\{23,11\}$



Improving LCA

Meaningful LCA (MLCA)

K={Smith, journal}



Improving LCA

Amoeba Join

- S_i is a **valid answer set** if $LCA(S_i) \in S_i$
  - only allows matches where
    - one **matched** node is
    - **an ancestor** of (or identical to) all matched nodes
- K={Smith, web}, L_1={11}, L_2={3,23}, S_1={11,3}, S_2={11,23}

Improving LCA

Amoeba Join

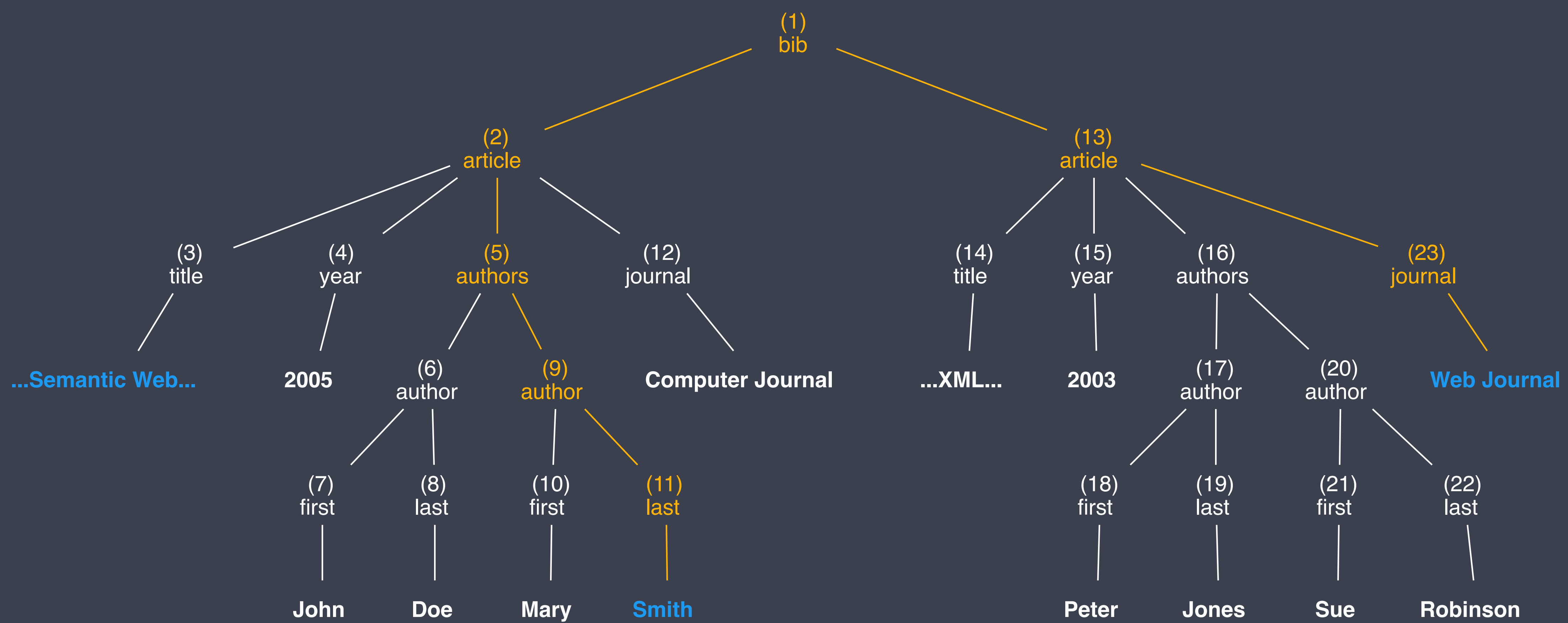
- S_i is a **valid answer set** if $LCA(S_i) \in S_i$
  - only allows matches where
    - one **matched** node is
    - **an ancestor** of (or identical to) all matched nodes
- K={Smith, web}, L_1={11}, L_2={3,23}, S_1={11,3}, S_2={11,23}

T. Saito and S. Morishita. **Amoeba Join: Overcoming Structural Fluctuations in XML Data**. WebDB

Improving LCA

Amoeba Join

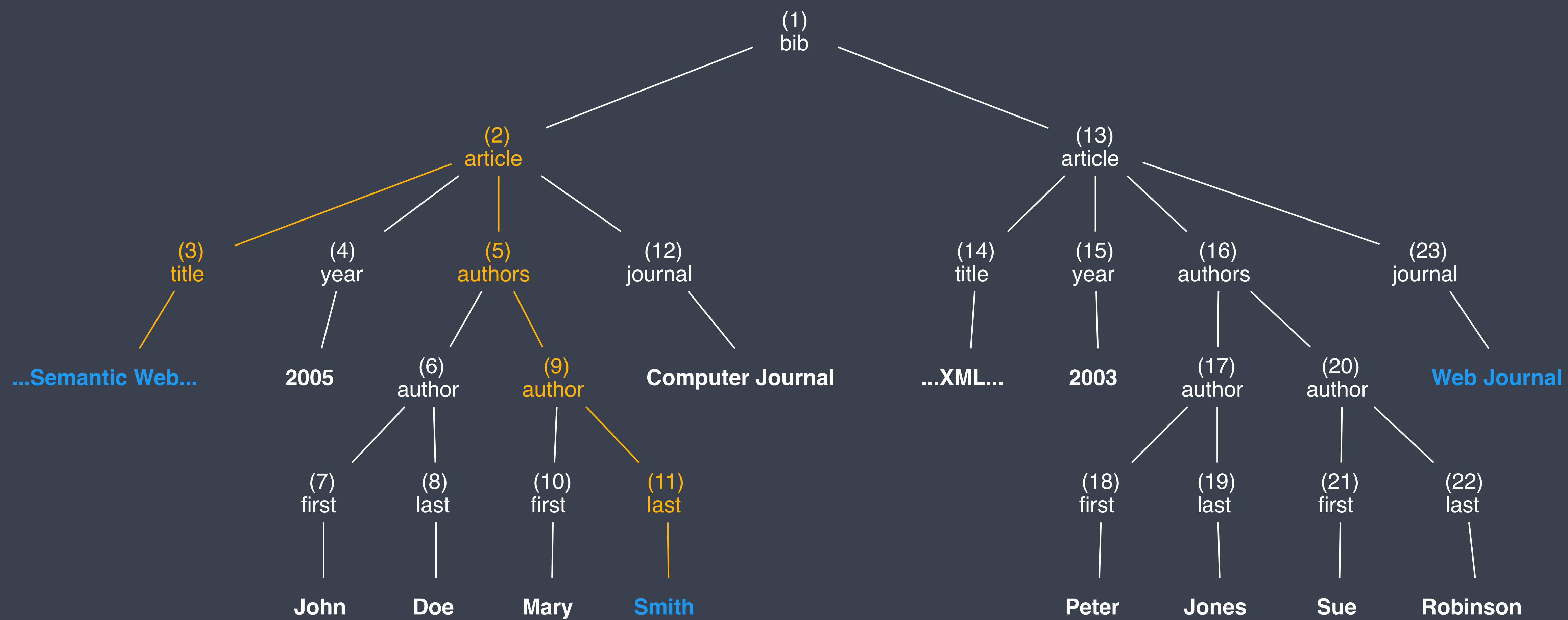
K={web, Smith} — No false positive



Improving LCA

Amoeba Join

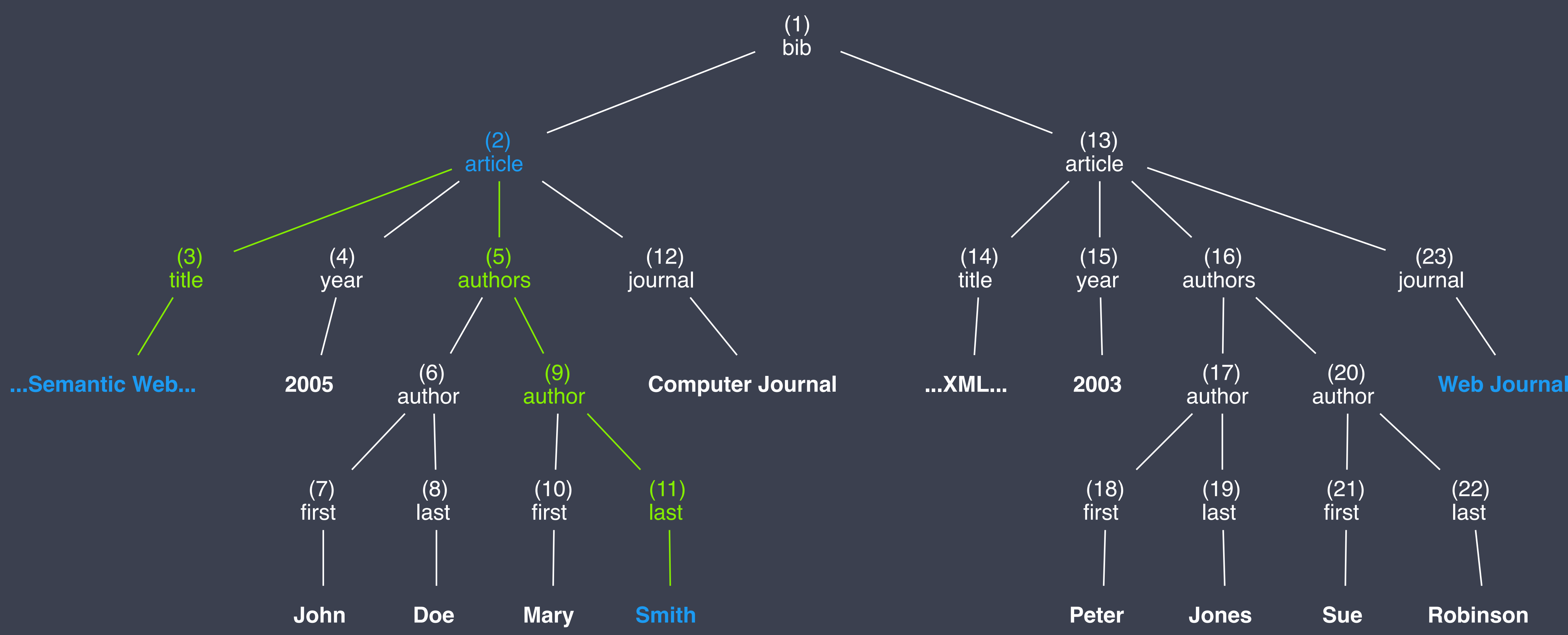
K={web, Smith} — False negative



Improving LCA

Amoeba Join

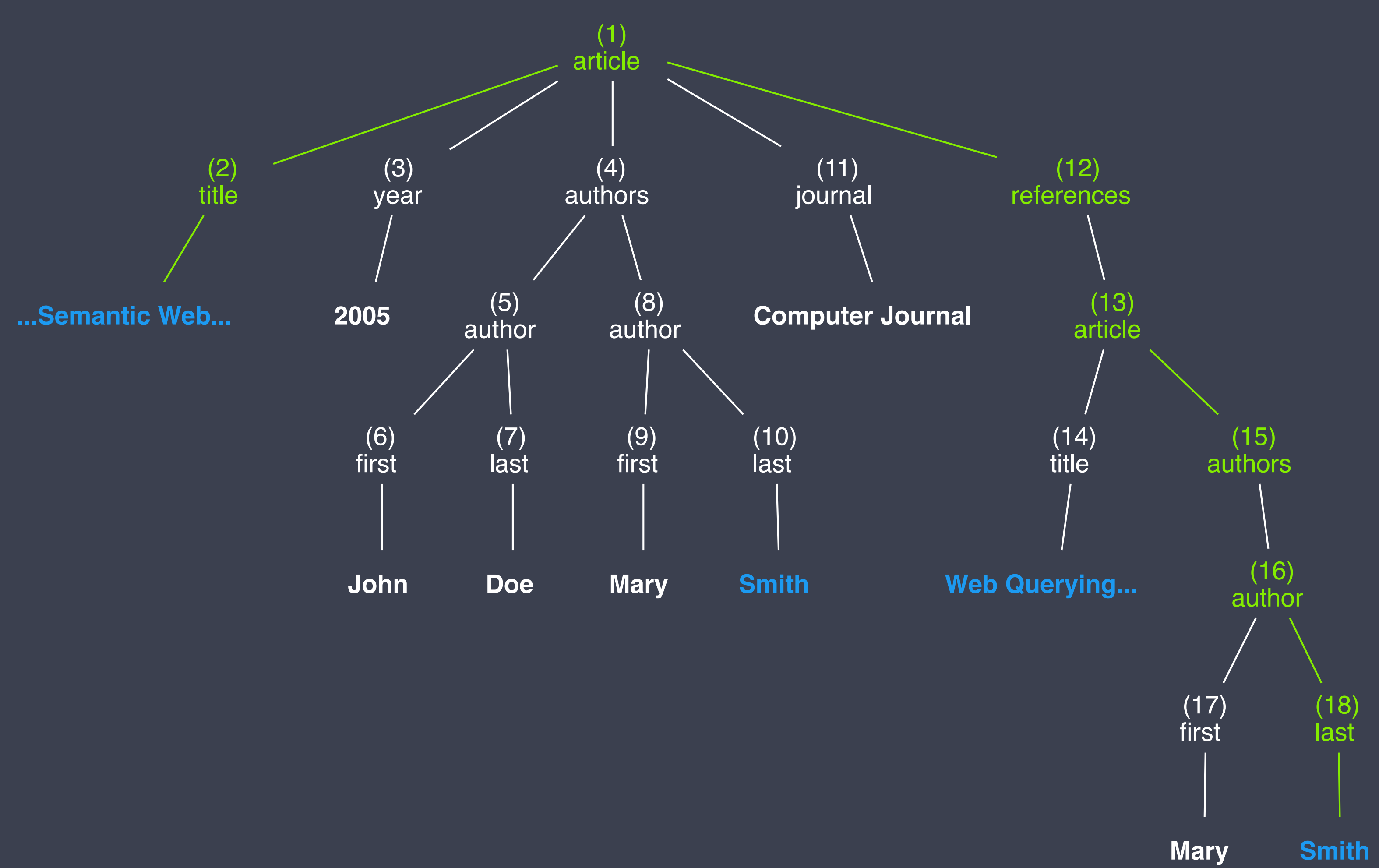
K={web, Smith, article}



Improving LCA

Amoeba Join

K={web, Smith, article} — False positive



Improving LCA

Compact LCA (CLCA)

- **CLCA:** LCA node of an answer set S_i
  - where **LCA(S_i) dominates all nodes** in S_i
  - Node v_i **dominates** node v_j if there is no $v_j \in S_i$ where LCA (S_i) is descendant of v_i
- Simpler: CLCA is an LCA
  - where **no node** in the associated answer set can be
    - **part** of a **more specific** answer set
- Similar to SLCA

Compact Valuable LCA (VLCA)

- **CVLCA**: CLCA node which is also a VLCA node
- Recall problems of SLCA and Interconnection Semantics

Improving LCA

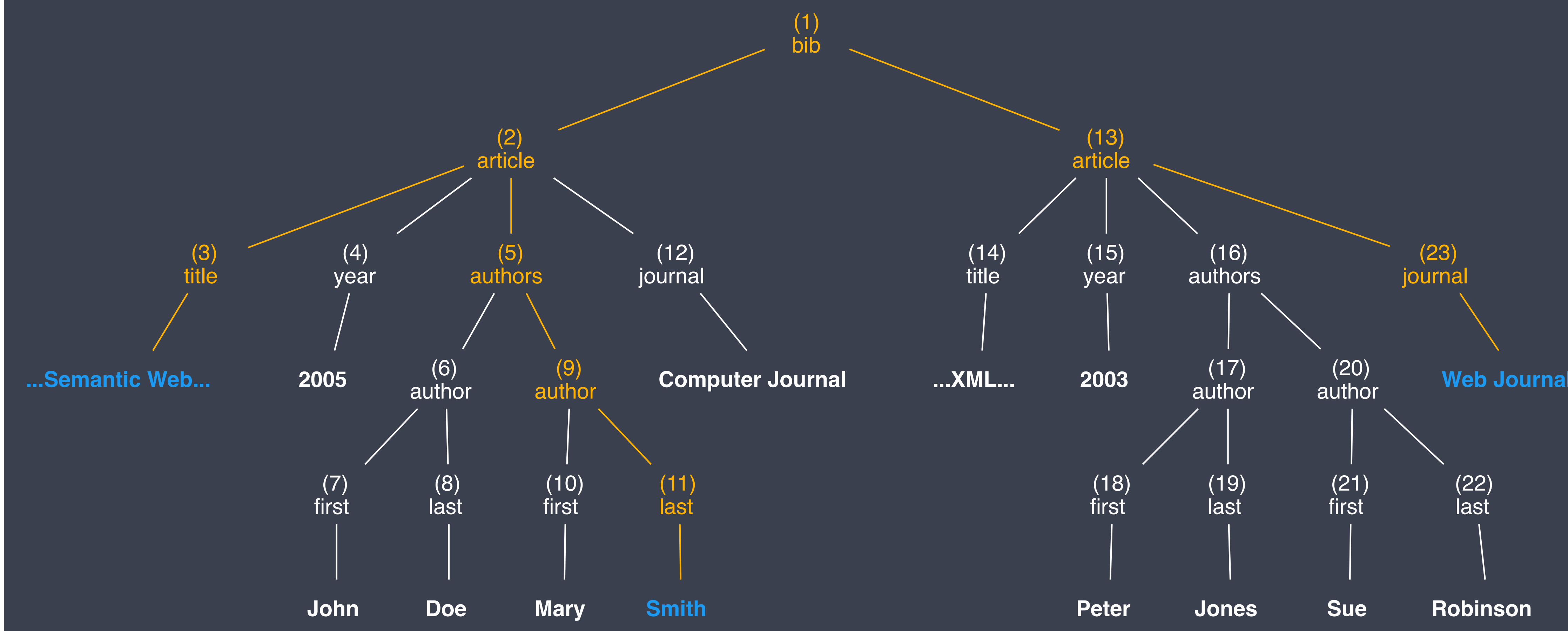
Relaxed Tightest Fragment (RTF)

- **Subtrees** are complete with respect to matches
  - while being as small as possible
- Similar to **XRank**:
  - maximum match without contained descendant LCAs
- K={Smith, web}, L_1={11}, L_2={3,23}, S_1={11,3,23}, S_2={11,3}, S_3={11,23} and LCA(11,3)=2, LCA(11,14)=1, LCA(11,3,23)=1

Improving LCA

Relaxed Tightest Fragment (RTF)

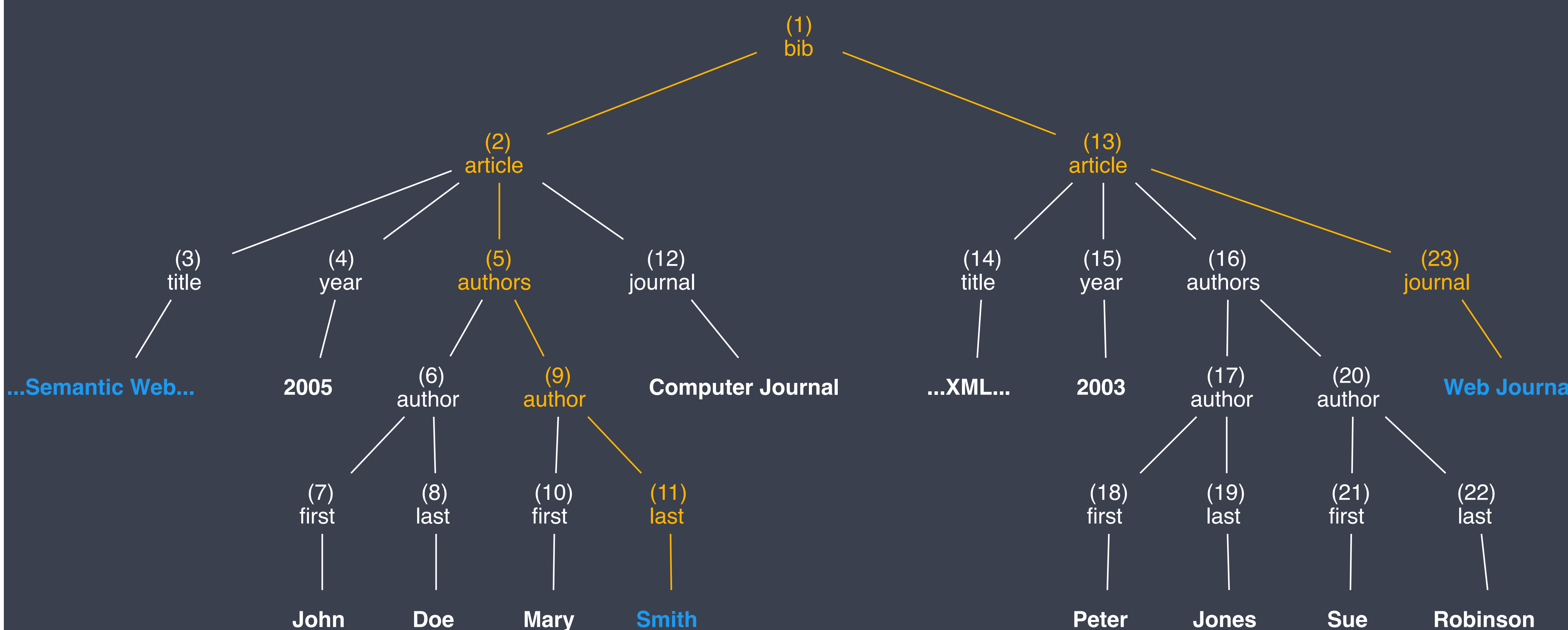
$S_1 = \{11, 3, 23\}$, c_1: ✖



Improving LCA

Relaxed Tightest Fragment (RTF)

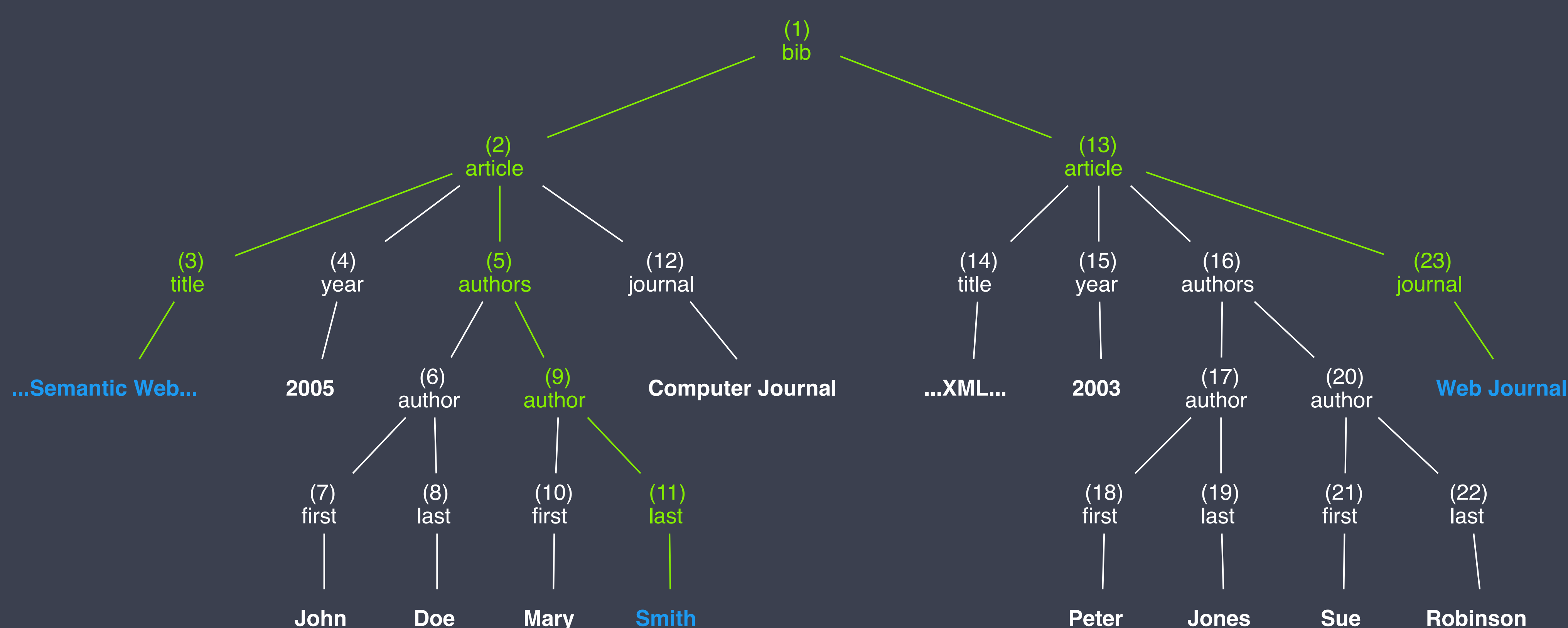
$S_3=\{11,23\}$, c_1: ✔, c_2: ✖



Improving LCA

Relaxed Tightest Fragment (RTF)

$S_2 = \{11, 3\}$, c_1: ✔, c_2: ✔, c_3: ✔



Improving LCA

XRank: Exclusive LCA

- Idea: Prefer **more specific LCAs**
  - unless reason not to (more matches)
- Result node is LCA node which
  - does **not contain further** LCAs or
  - which is **still LCA** if LCA subtrees are ignored
- As before, K={Smith, web}, L_1={11}, L_2={3,23}, S_1={11,3}, S_2={11,23} and LCA(11,3)=2, LCA(11,23)=1

Improving LCA

XRank: Exclusive LCA

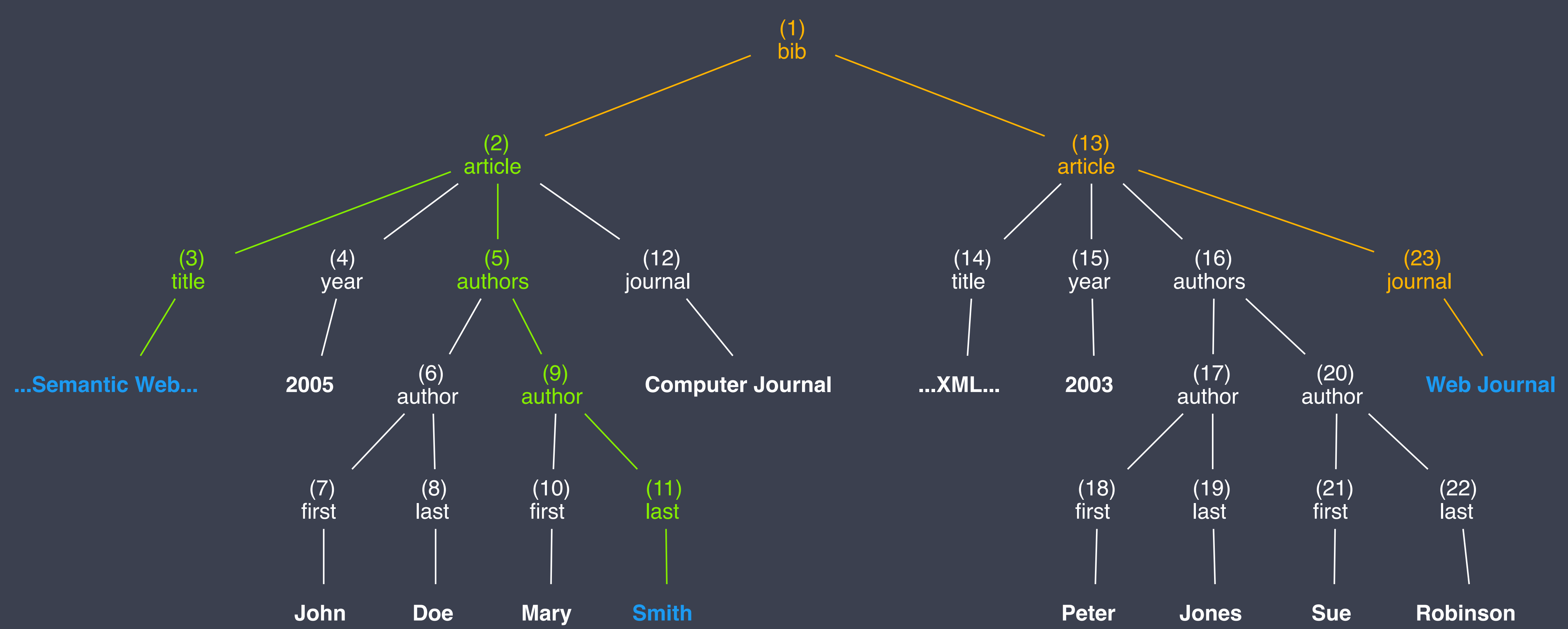
- Idea: Prefer **more specific LCAs**
  - unless reason not to (more matches)
- Result node is LCA node which
  - does **not contain further** LCAs or
  - which is **still LCA** if LCA subtrees are ignored
- As before, K={Smith, web}, L_1={11}, L_2={3,23}, S_1={11,3}, S_2={11,23} and LCA(11,3)=2, LCA(11,23)=1

Lin Guo et al. **XRANK: Ranked keyword search over XML documents**. SIGMOD 2003

Improving LCA

XRank: Exclusive LCA

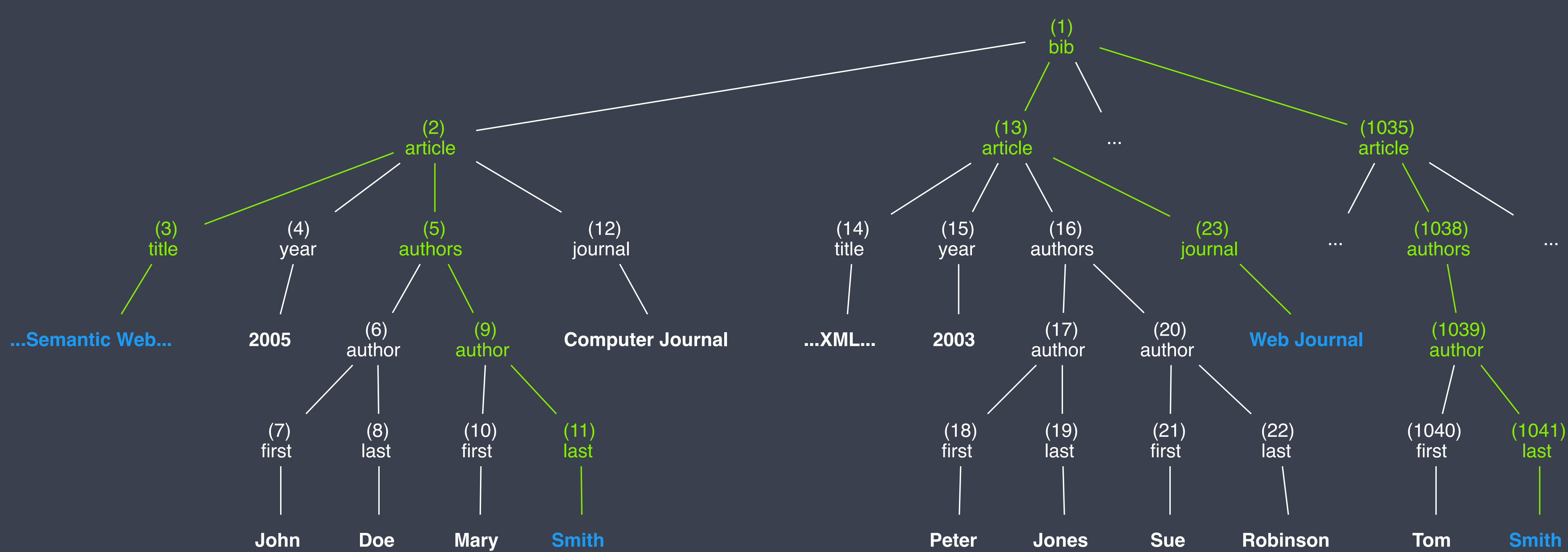
Only (2) is a valid result node



Improving LCA

XRank: Exclusive LCA

But...



Improving LCA

XRank: Exclusive LCA

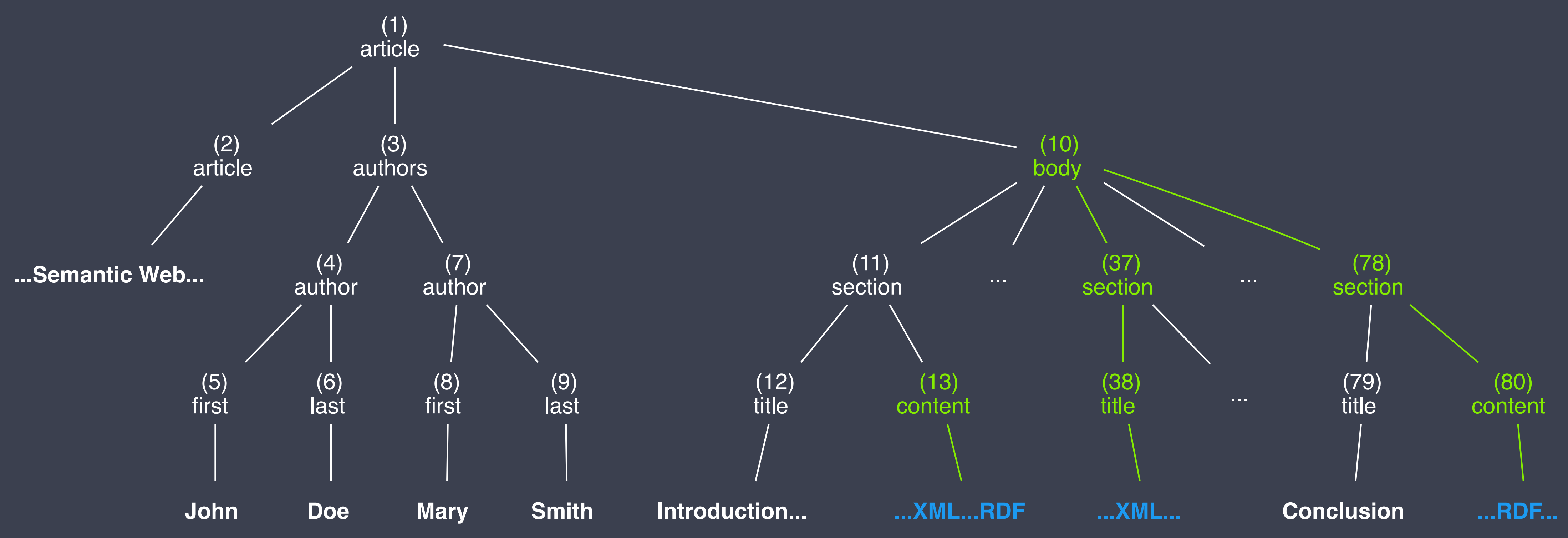
- Objection:
  - XRank targeted at **document-centric XML**
- Does this solve the problem?
- Consider K={XML,RDF}

Improving LCA

XRank: Exclusive LCA

S_1={13}, S_2={38,80}...

Is 10 a good return node?



Improving LCA

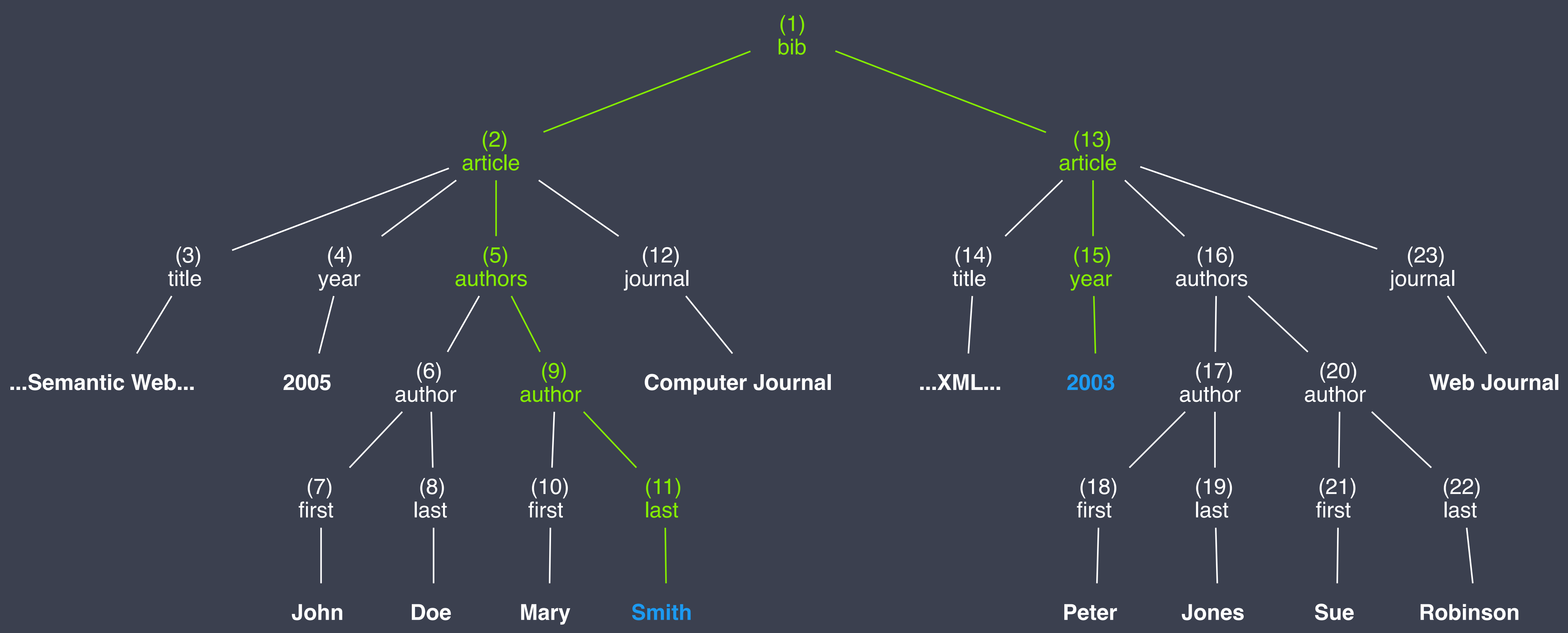
Summary

- No heuristic with perfect precision and recall
- Data-driven solutions, not universally applicable
- Monotonicity and consistency are desirable but often violated
- In some cases, no heuristic produces suitable results

Improving LCA

Summary

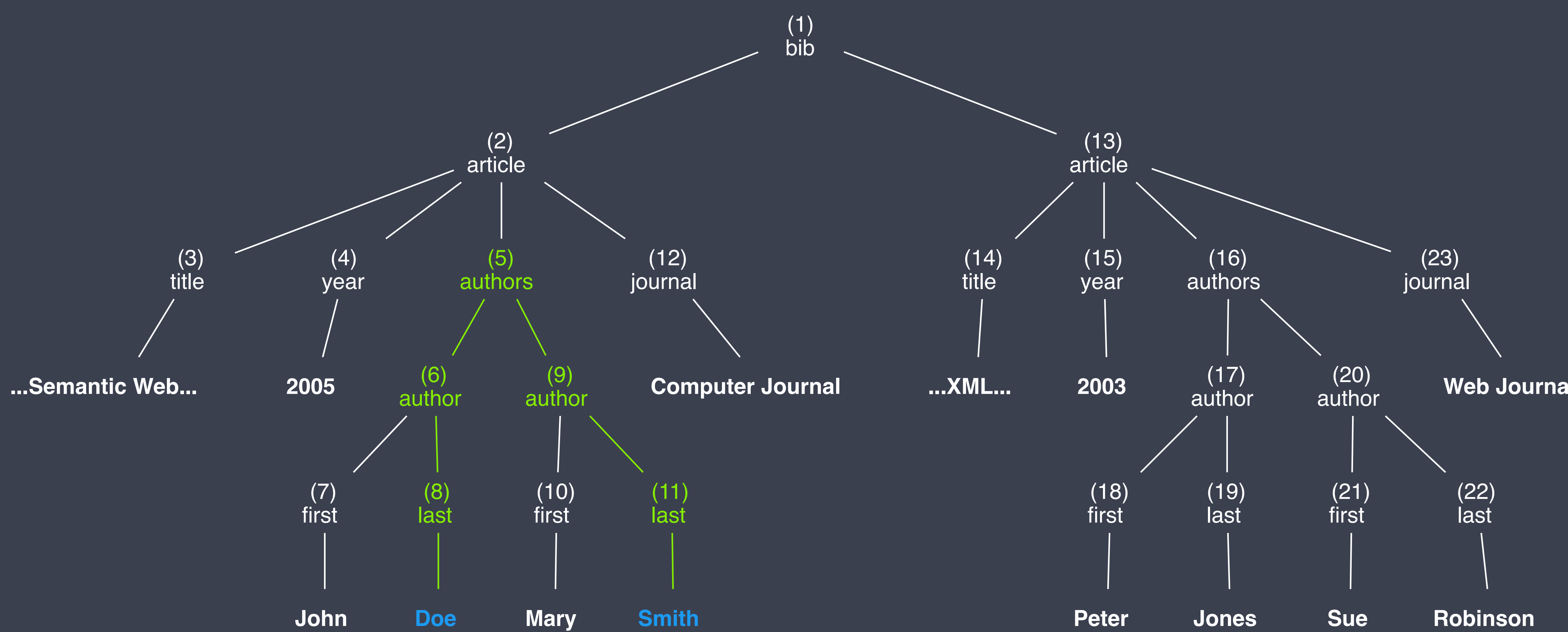
K={Smith,2003}



Improving LCA

Summary

K={Doe, Smith}



Queries as Keywords

RDF

- Summarize RDF graph (Q2RDF, Q2Semantic)
- Generate queries or query results by connecting matches
  - Dijkstra's Algorithm (Q2RDF)
  - Kruskal's Minimum Spanning Tree Algorithm (SPARK)
  - Templates based on types (SemSearch)
  - Cost-based heuristics (Q2Semantic)

2.2 Determining Return Values

From an answer node
to an **answer representation**

Determining Return Values

Approach 1: **LCA**

- LCA (or similar) node, e.g. ELCA, MLCA

(2)
article

Determining Return Values

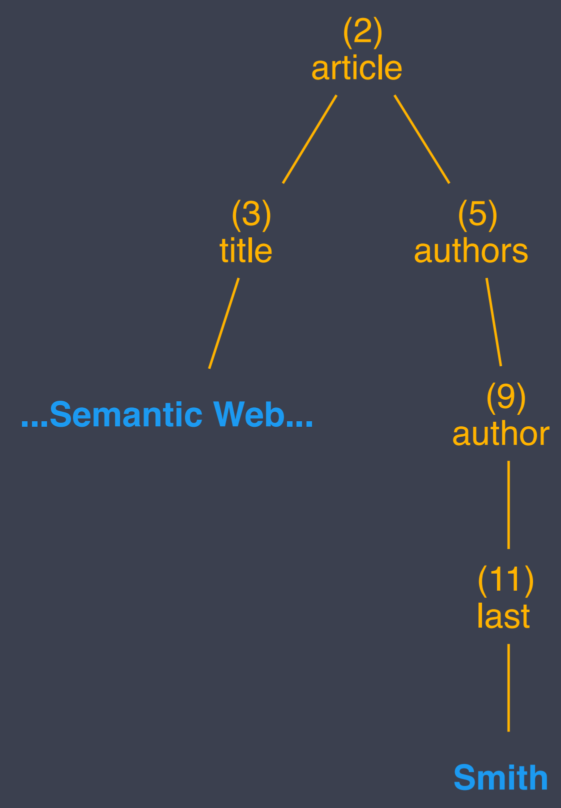
Approach 2: **Matched Nodes**

- Matched nodes (Interconnection Semantics)

Determining Return Values

Approach 3: **LCA & Path**

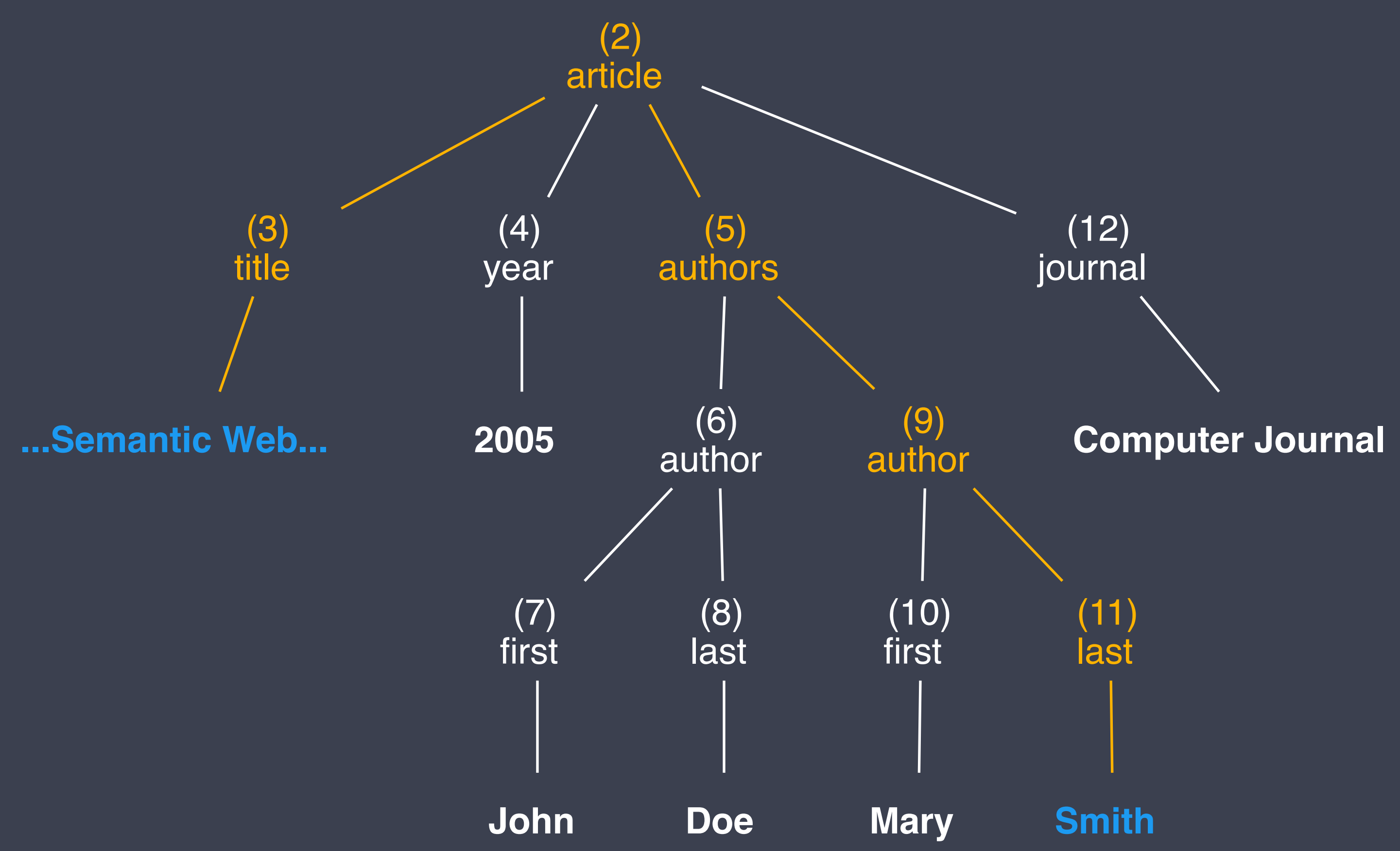
- Path from LCA to matched nodes
  - e.g. BANKS, RTF



Determining Return Values

Approach 4: **Entire Subtree**

- LCA or entity subtree e.g. SLCA, XRank, VLCA



Determining Return Values

Comparison & Summary

- Neither approach is always satisfying
  - **subtree** may be too **big**
  - **path or node** may **not** provide **enough information**
- More controlled return value is desirable
  - use query elements to determine a suitable return value automatically

Determining Return Values

Exemplar: **XSeek**

- Processing steps:
  1. Match query on **node labels and content**
  2. **Group** matches using **SLCA**
  3. Extract **return nodes** from query: If a term in K matches a node label and no descendant content is matched, consider the term a return node (explicit)
  4. When there are no return nodes, return **SLCA entity subtree** (implicit)

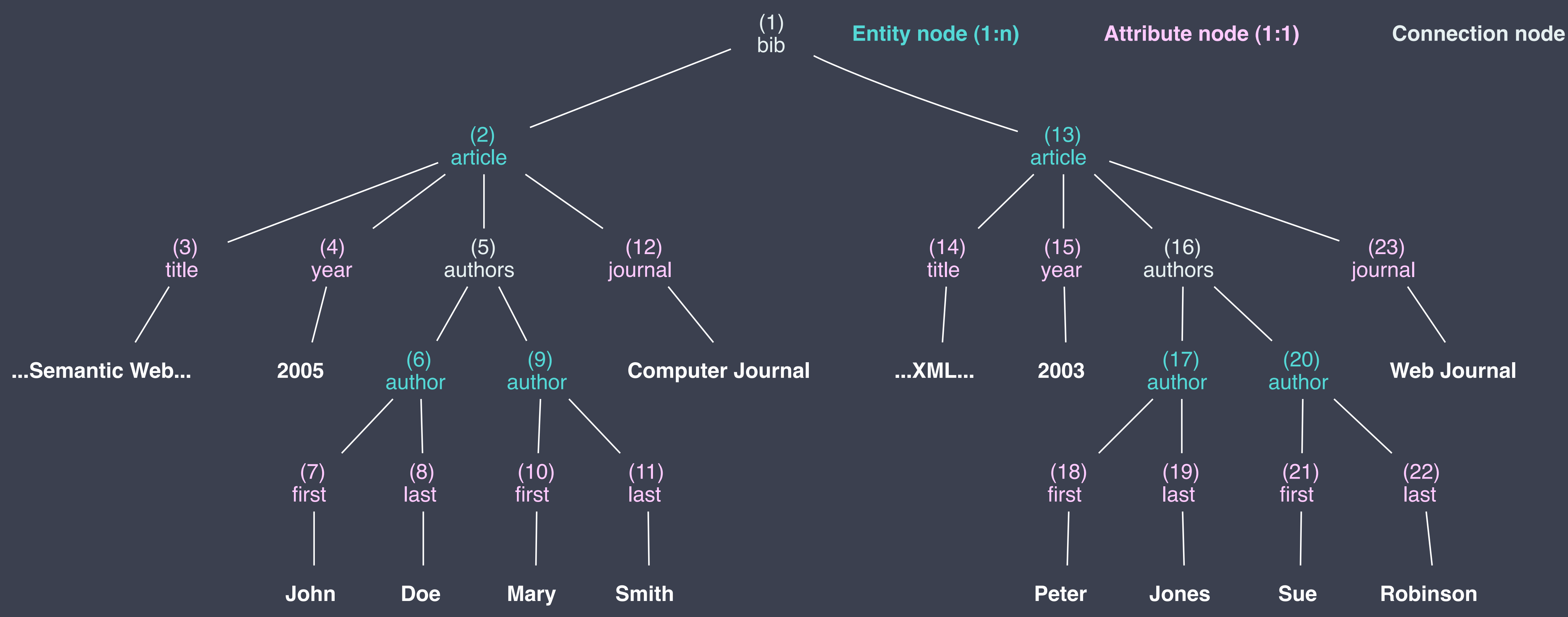
Determining Return Values

Exemplar: **XSeek**

- Terms in the query that are not return nodes are **search predicates** i.e. keywords to find
- **Entities** and their attributes inferred from the **schema** using cardinality information
- **Final return value:**
  - explicit or implicit return nodes and entities’ attributes

Determining Return Values

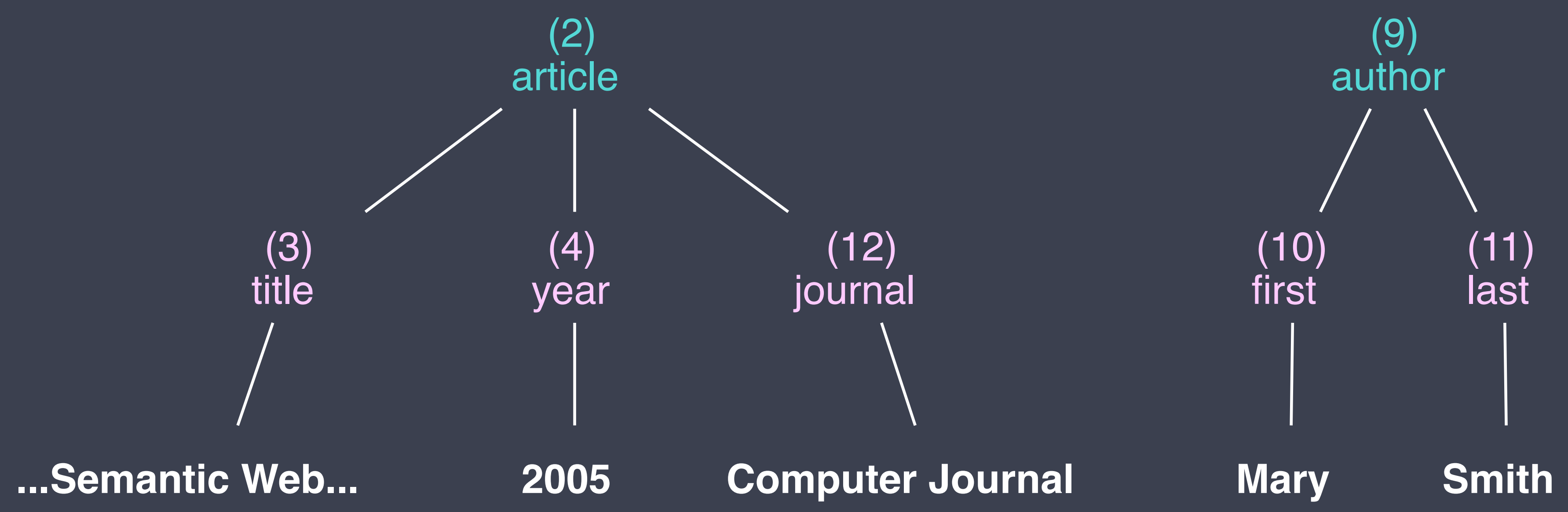
Exemplar: **XSeek**



Determining Return Values

Exemplar: **XSeek**

K={Smith, web}



Determining Return Values

Exemplar: **XSeek**

K={Smith, title}

(3)
title

...Semantic Web...

(11)
last

Smith

Determining Return Values

Exemplar: **XSeek**

K={Smith, title}

(3)
title

...Semantic Web...

(11)
last

Smith

Z. Liu, J. Walker and Y.Chen. **XSeek: a semantic XML search engine using keywords**. VLDB 2007

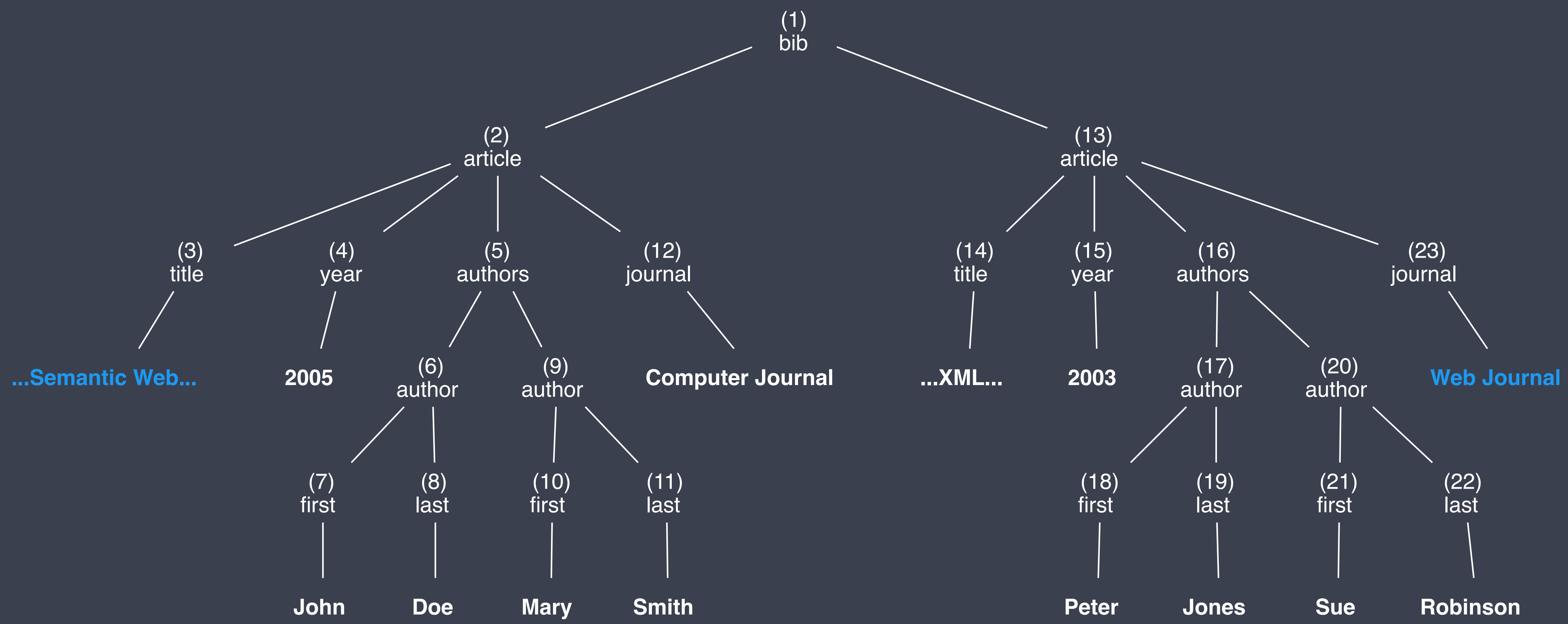
Queries as Keywords

Expressiveness

- Queries: **unordered lists** with **implicit** conjunction
  - \+ Conjunction, disjunction (Multiway, Abbaci et al.)
  - \+ Inclusion, sibling, negation, precedence (Abbaci et al.)
- User-selected return value (XSeek, MIU)
- Numeric comparison operators (MIU)
- Optional terms (Interconnection)
- label:keyword terms (Interconnection, XSearch)
- Keyword-enhanced languages

Queries as Keywords

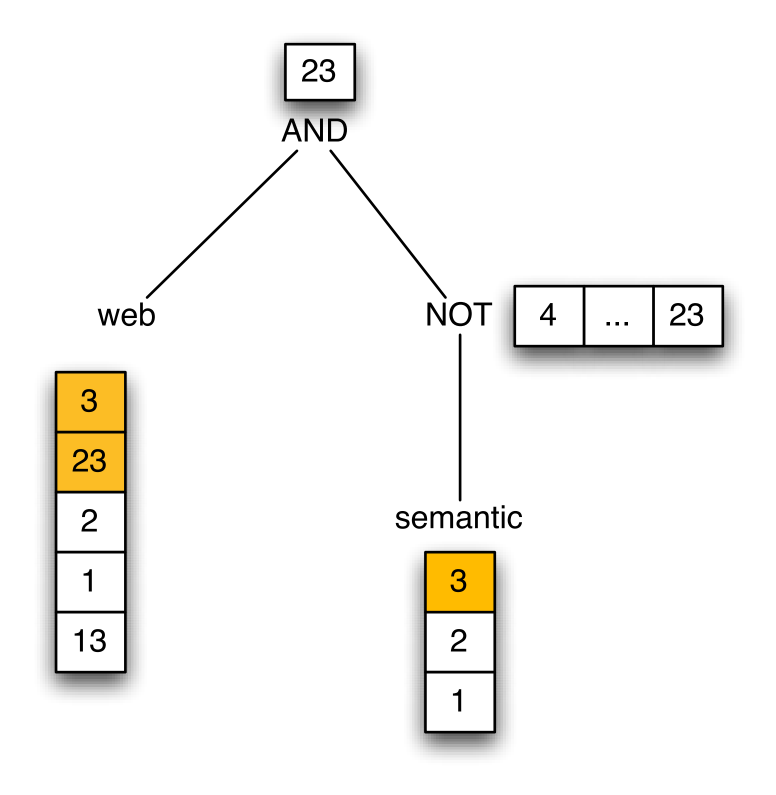
Expressiveness



Queries as Keywords

Expressiveness

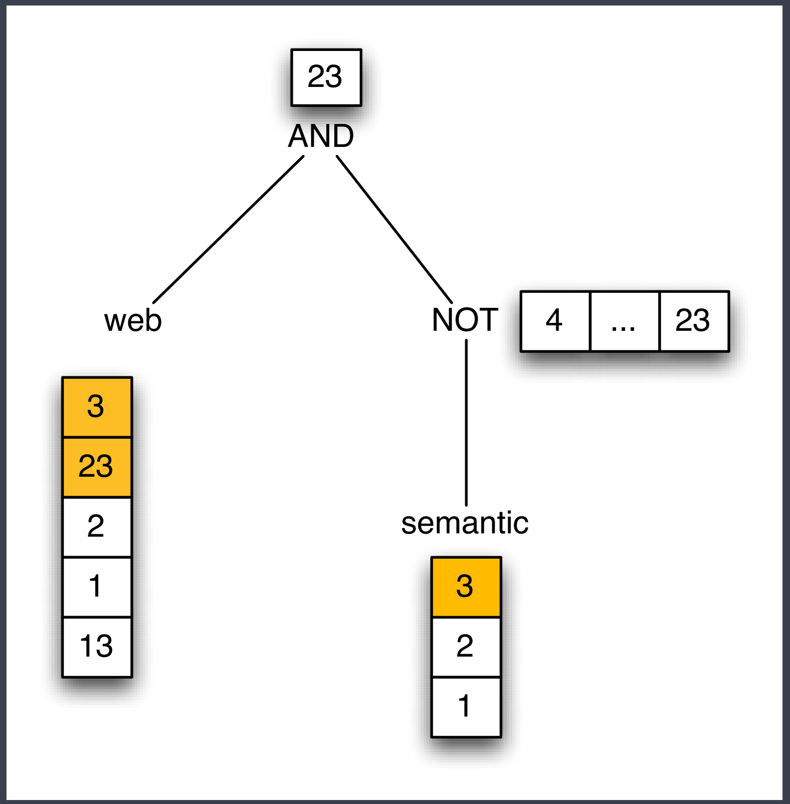
- Transform query into **binary tree**
- Construct set of **matching nodes** and their **ancestors** for each term
- **Bottom-up** processing:
  - Apply operator to sets of nodes (i.e. intersection for conjunction)
  - The nodes remaining at the root are **valid answers** (LCA-like)



Queries as Keywords

Expressiveness

- Transform query into **binary tree**
- Construct set of **matching nodes** and their **ancestors** for each term
- **Bottom-up** processing:
  - Apply operator to sets of nodes (i.e. intersection for conjunction)
  - The nodes remaining at the root are **valid answers** (LCA-like)



F. Abbaci et al. **Index and Search XML Documents by Combining Content**. International Conference on Internet Computing 2006

Queries as Keywords

Ranking

- **Size** of answer subtree
- **Distance** between matched nodes and answer tree root node
- **tf-idf**/vector space model
- XRank: PageRank-like ranking

XRank — Ranking factors

- **Result specificity**: vertical distance
- **Keyword proximity**: horizontal distance
- **Hyperlink awareness**: PageRank value, adapted for XML
  - distinction between XML edges and IDREF links
  - Bidirectional propagation between XML edges
  - distinction between following forward and backward links

Queries as Keywords

XRank — Ranking factors

- **Result specificity**: vertical distance
- **Keyword proximity**: horizontal distance
- **Hyperlink awareness**: PageRank value, adapted for XML
  - distinction between XML edges and IDREF links
  - Bidirectional propagation between XML edges
  - distinction between following forward and backward links

Lin Guo et al. **XRANK: Ranked keyword search over XML documents**. SIGMOD 2003

Queries as Keywords

Limitations

- Mostly **limited to tree** data
- Determining semantic entities in structured data
  - Assumption: No element outside of LCA subtree is relevant
  - **No universal solution**, data-driven
- Relatively **low expressiveness**

Queries as Keywords

Limitations

- **Query answers**
  - Little control over selection
  - Result may be too verbose or not informative enough
  - No construction or aggregation
  - Exception: Keyword-enhanced languages
- No querying of data in **mixed formats (RDF & XML)**

Part 3
KWQL

Keyword-based query language
for Semantic Wikis

Semantic **Wikis**:

The **(Semantic) Web in the Small**

- As on the Semantic Web we have
  - **pages** and **links** between them and annotations
  - content created by **many different** people
- But we also have
  - central control, organization and administration
  - a small (or at least manageable) number of pages
  - Strong social factors and collaboration

→**Wikis as a testbed for the “real” web**

KWQL: Keyword-Based QL for Wikis

Characteristics

- KWQL can access all elements the user interacts with
  - Combined querying of text, annotation and metadata
  - Querying of informal to formal annotations
  - Combination of selection criteria from several data sources in one query
- Aggregation and construction
  - Data construction
  - Embedded queries
  - Continuous queries

KWQL: Keyword-Based QL for Wikis

Characteristics

- KWQL can access all elements the user interacts with
  - Combined querying of text, annotation and metadata
  - Querying of informal to formal annotations
  - Combination of selection criteria from several data sources in one query
- Aggregation and construction
  - Data construction
  - Embedded queries
  - Continuous queries

F. Bry and K. Weiand. **Flavors of KWQL, a Keyword Query Language for a Semantic Wiki.** SOFSEM, 2010.

KWQL: Keyword-Based QL for Wikis

Characteristics

- Varying complexity of queries
  - Simple label-keyword queries
  - Conjunction/disjunction/optional
  - Structural queries
  - Link traversal

KWQL: Keyword-Based QL for Wikis

Examples

1
```
Java
```

2
```
author:"Mary"
```

3
```
ci(text:Java OR (tag(name:XML) AND author:Mary))
```

4
```
ci(tag(name:Java) link(target:ci(title:Lucene)
   tag(name:uses)))
```

5
```
ci(title:Contents text:($A "–" ALL($T,",")))
   @ ci(title:$T author:$A)
```

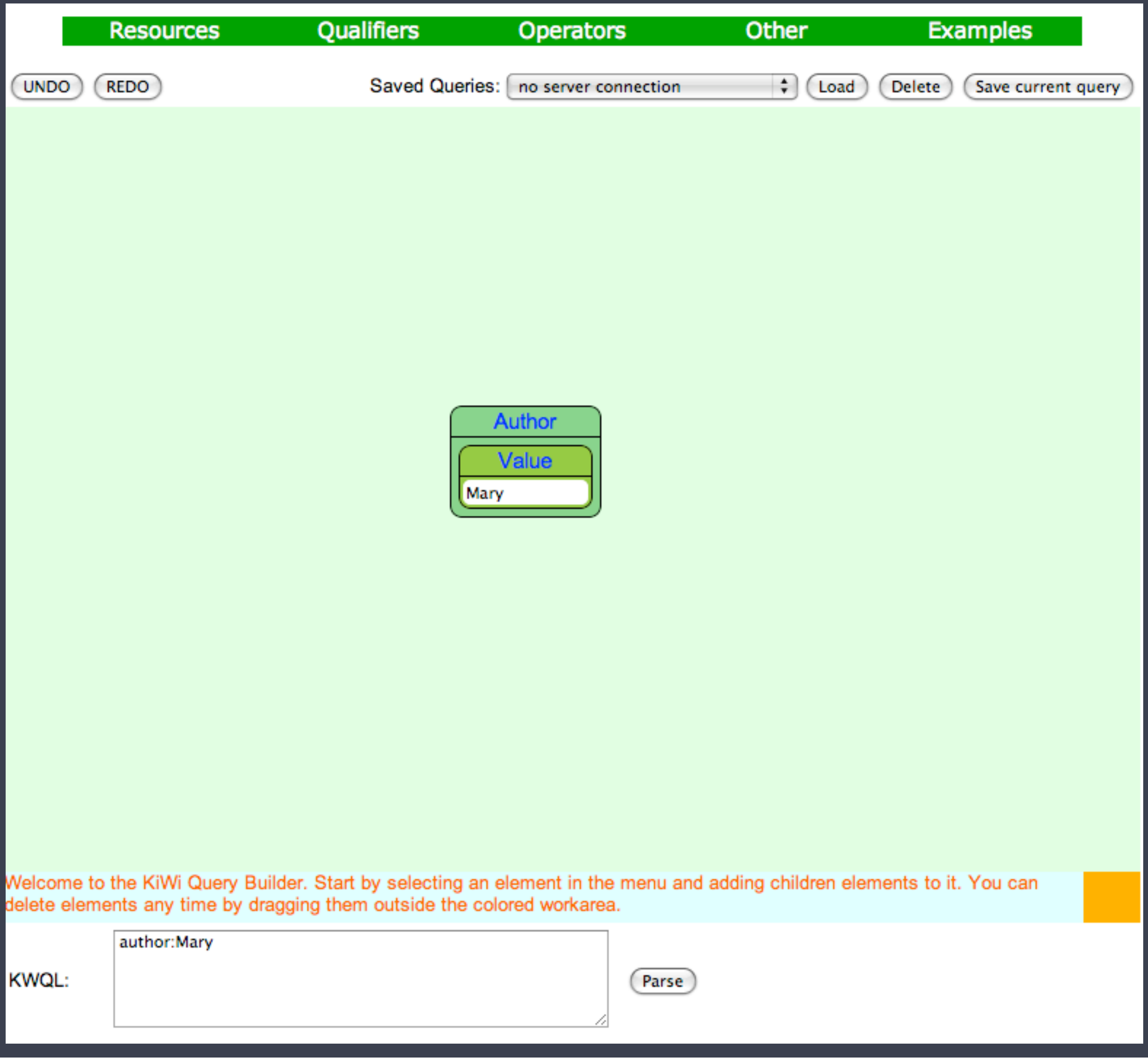
KWQL: Keyword-Based QL for Wikis

visKWQL

- KWQL's **visual** counterpart
- Query by example paradigm
- Round-tripping between KWQL and visKWQL
  - Visualization of textual queries
  - visKWQL as a tool to learn KWQL

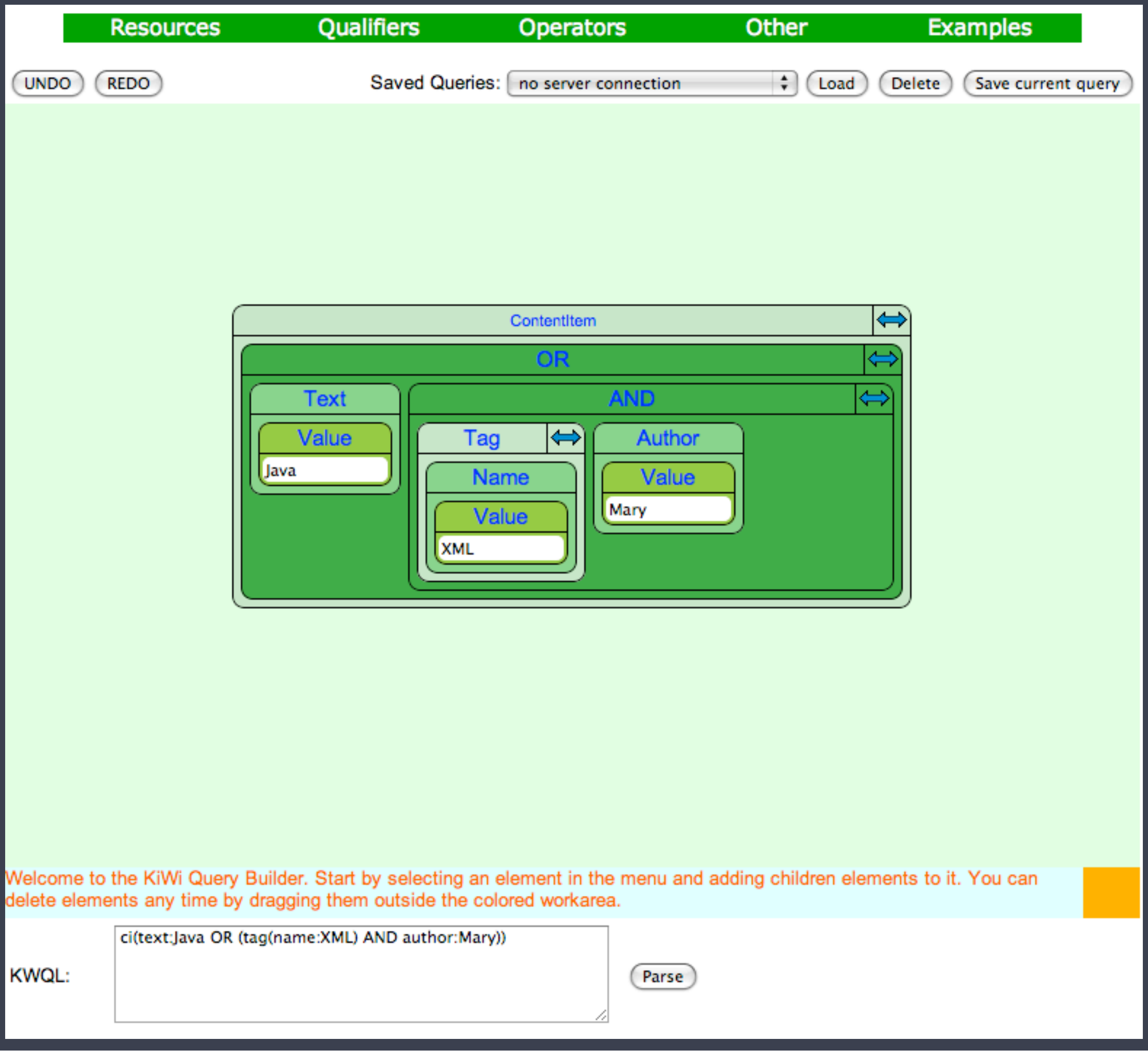
KWQL: Keyword-Based QL for Wikis

KWQL and visKWQL



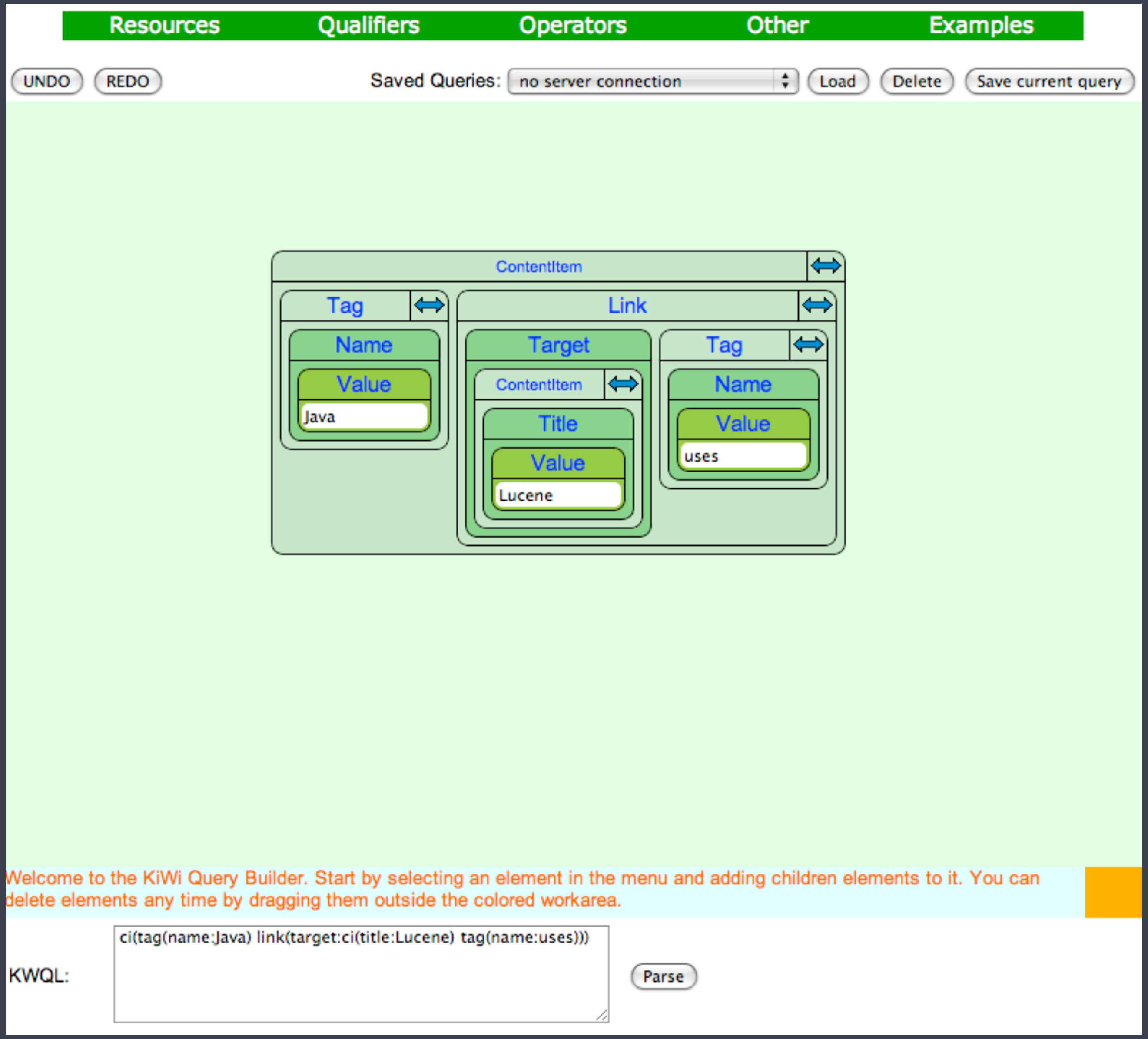
KWQL: Keyword-Based QL for Wikis

KWQL and visKWQL



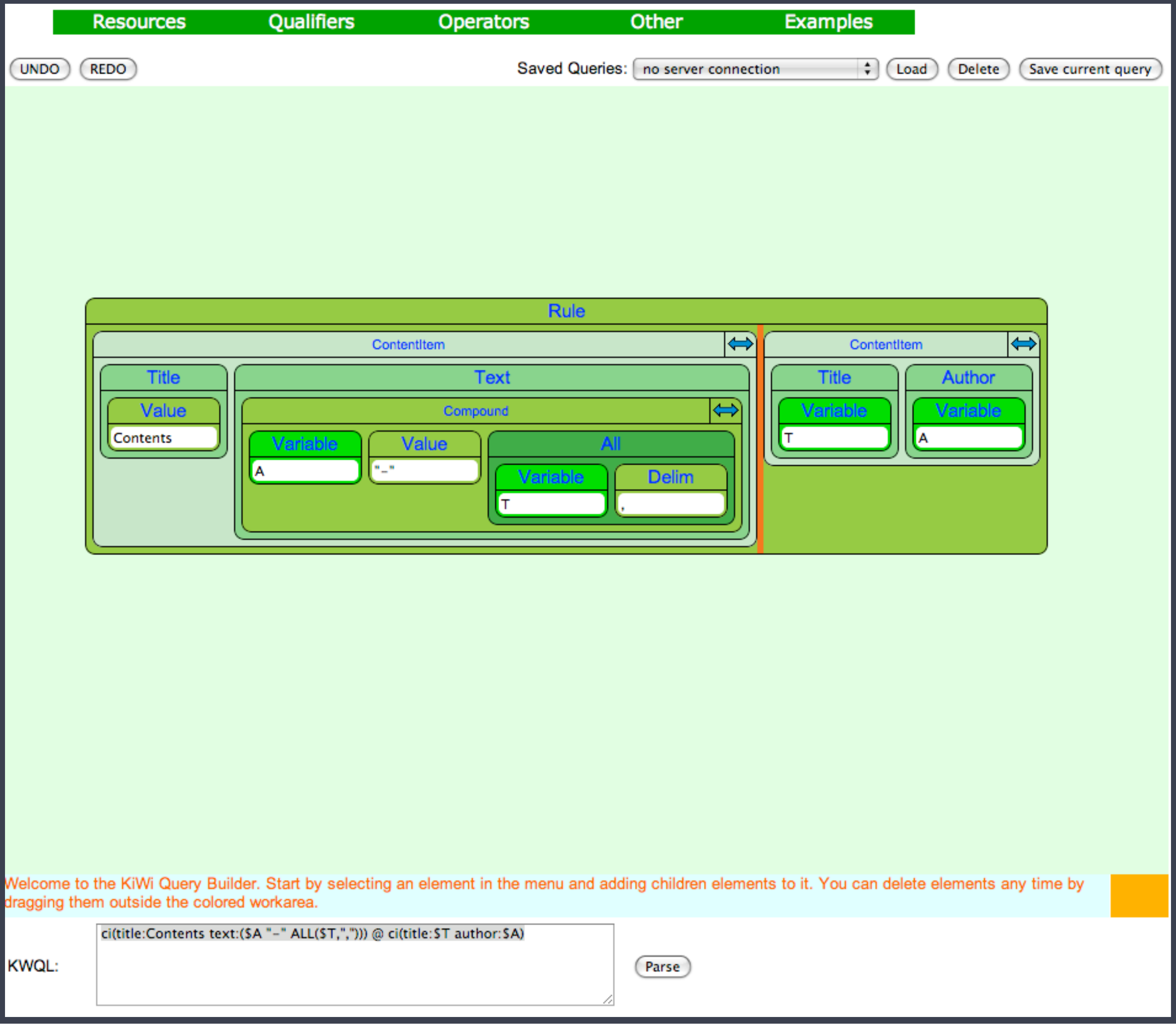
KWQL: Keyword-Based QL for Wikis

KWQL and visKWQL



KWQL: Keyword-Based QL for Wikis

KWQL and visKWQL



Part 4

Conclusion

Conclusion

- Is it even possible to find a universal grouping mechanism?
- How easy to use can a query language be while still being powerful enough?
- How complicated can a query language be without becoming too hard to use for casual users?

Conclusion

Slides and links at

http://pms.ifi.lmu.de/wise