XPathLog: A Declarative, Native XML Data Manipulation Language

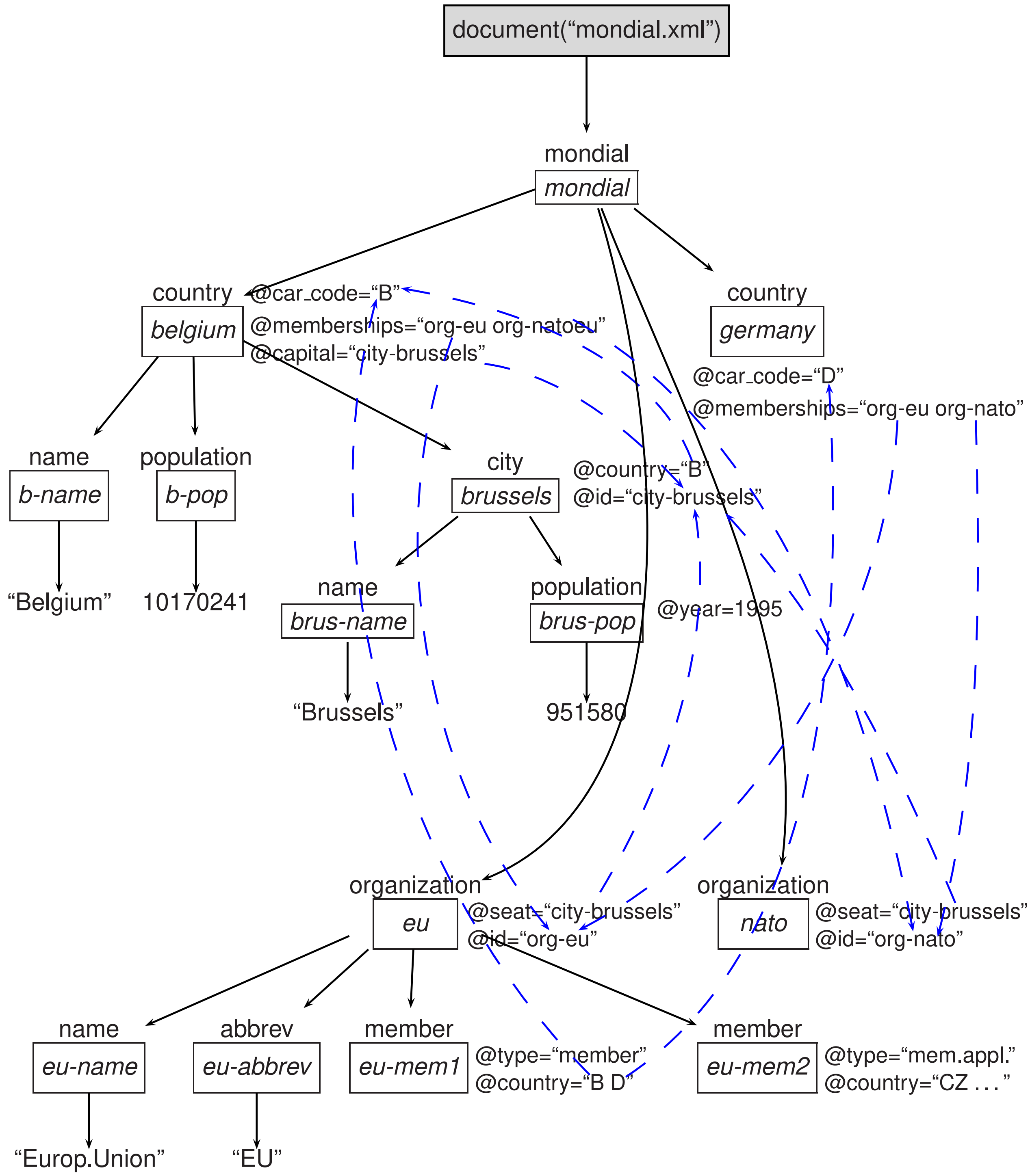
Wolfgang May
Institut für Informatik
Universität Freiburg
Germany
may@informatik.uni-freiburg.de

IDEAS
Grenoble, 16.7.2001

Example: Mondial

```
<mondial>
  <country car_code="B" capital="cty-Brussels"
              memberships="org-eu org-nato ...">
     <name>Belgium</name>
     <population>10170241</population>
     <city id="cty-Brussels" country="B">
        <name>Belgium</name>
        <population year="95">951580</population>
     </city>
     :
  </country>
  <country car_code="D" capital="cty-Berlin"
              memberships="org-eu org-nato ...">
     :
  </country>
  <organization id="org-eu" seat="cty-Brussels">
    <name>European Union</name>  <abbrev>EU</abbrev>
    <members type="member" country="GR F E A D I B L ..."/>
    <members type="membership applicant" country="AL CZ ..."/>
  </organization>
  <organization id="org-nato" seat="cty-Brussels" ...>
    :
  </organization>
    :
</mondial>
```

Example: Mondial XML Tree



LANGUAGES FOR XML

- two proposals of querying languages:
  XML-QL, W3C
- W3C XML Query Requirements/Data Model/Algebra
- no XML update language
  “Updating XML” @ SIGMOD 2001

Selection

- Navigation: XPath, XQuery
- XML Patterns: XML-QL

Generation

- instantiating XML patterns: XML-QL and XQuery

Languages

- Why not use XPath for *updating* and *generating*?

DESIGN DECISIONS

- experiences with F-Logic for semi-structured data and data integration
- declarative rule-based language with bottom-up semantics
- extend XPath with variable bindings
- graph data model, overlapping trees, multiple parents

XPathLog: Syntax

Extends the XPath syntax

XPathLog *reference expressions* are XPath *location paths*

```
[0]   ReferenceExpr ::= AbsLocPath
                      | ConstLocPath
[2b]  ConstLocPath ::=  constant "/" RelLocPath
                      | variable "/" RelLocPath
```

Extend *LocationSteps*

```
[4] Step ::= AxisSpec NodeTest Pred*
           | AxisSpec NodeTest Pred* "->" Var Pred*
           | AxisSpec Var Pred*
           | AxisSpec Var Pred* "->" Var Pred*
```

- navigation by dereferencing `IDREF` attributes.
- Predicates over reference expressions

XPathLog by Examples

Pure XPath expressions

?- //country[name/text() = "Belgium"]//city/name/text().

true

Output Result Set

?- //country[name/text() = "Belgium"]//city/name/text()→N.

N/"Brussels"

:

Additional Variables

?- //country[name/text()→N1 and

@car_code→C]//city/name/text()→N2.

N2/"Brussels"   C/"B"   N1/"Belgium"

:

Local Variables

?- //country[name/text()→N1]//city[population/text()→_P]

/name/text()→N2,

_P > 100000.

Dereferencing

?- //organization[@seat = members/@country/@capital]

/@seat/name/text()→N.

XPathLog by Examples

Metadata: Tag Variables and Schema querying

Navigation Variables

?- //Type→X[name/text()→"Monaco"].

Type/country	X/*country-monaco*
Type/city	X/*city-monaco*

Schema Querying

?- //city/N.

N/name

N/population

:

DATA MODEL: XTREEGRAPH

- extends DOM/XML Query Data Model
- Tailored to data integration
- Graph "database"
  - *XML element nodes*: vertices
  - subelement relationship,
  - text children are leaf literals
  - *XML attribute nodes*: literal values
  - multivalued attributes (`NMTOKENS`, `IDREFS`) are split
  - reference attributes (`IDREF`) are resolved
- element may be subelement of several vertices

⇒ multiple trees, overlapping,
no *global* "document" order

- (names are elements of the universe)

Data Model

For every vertex x:

- $\mathcal{A}_{\mathcal{X}}(\text{child}, x)$
- $\mathcal{A}_{\mathcal{X}}(\mathtt{attribute}, x)$

⇒ derived axes as views

Mapping: XML instance $\rightarrow$ XTreeGraph

Semantics of XPathLog

- Extends semantics for XPath by P.Wadler (1999)

XPathLog expressions: annotated result list

(i) a result list, and

(ii) with every element of the result list, a list of variable bindings is associated.

Example

$expr$ =

//organization→O[member/@country[@car_code→C and name/text()→N]]/abbrev/text()→A.

results in

list((“UN”, {(O/*un*, A/“UN”, C/“AL”, N/“Albania”),
(O/*un*, A/“UN”, C/“GR”, N/“Greece”),
: }),
(“EU”, {(O/*eu*, A/“EU”, C/“D”, N/“Germany”),
(O/*eu*, A/“EU”, C/“F”, N/“France”),
: }),
:)

XPathLog Rules

$$\text{head}(V_1,\ldots,V_n) \text{ :- } \text{body}(V_1,\ldots,V_n)$$

- Evaluation of rule bodies = queries
- Constructive semantics for XPathLog atoms in rule heads
- Definite XPathLog atoms:
  - uses only the child and sibling axes
  - no negation, function applications, aggregation, and *proximity position predicates*

"/" and "[...]" act as constructors:

- $host[property{\rightarrow}value]$ modifies $host$
- $host/property\ remainder$
  inserts new element $host/property$ which satisfies $remainder$
- $property$ of the form
  - child::$name$
  - child(i)::$name$
  - preceding/following-sibling::$name$
  - preceding/following-sibling(i)::$name$
  - attribute::$name$

$\Rightarrow$ unambiguous insertions

RULE HEADS: UPDATES

Attributes

C[@datacode→"be"], C[@memberships→O] :-
  //country→C[@car_code="B"],
  //organization→O[abbrev/text()→"EFTA"].

$\Downarrow$

<country $\boxed{\text{datacode="be"}}$ car_code="B"
  memberships="org-eu org-un $\boxed{\text{org-efta}}$ ...”>
  ⋮
</country>

- C: host element
- O: target of reference
- extends $\mathcal{A}_{\mathcal{X}}(\mathtt{attribute}, \textit{belgium})$

SEMANTICS OF RULE HEADS

Create “free” elements

/country[@car_code→“BAV”].

$\Downarrow$

<country car_code=“BAV”> </country>

Semantics of Rule Heads

Add subelement relationships

C[@capital→X and city→X and city→Y] :-

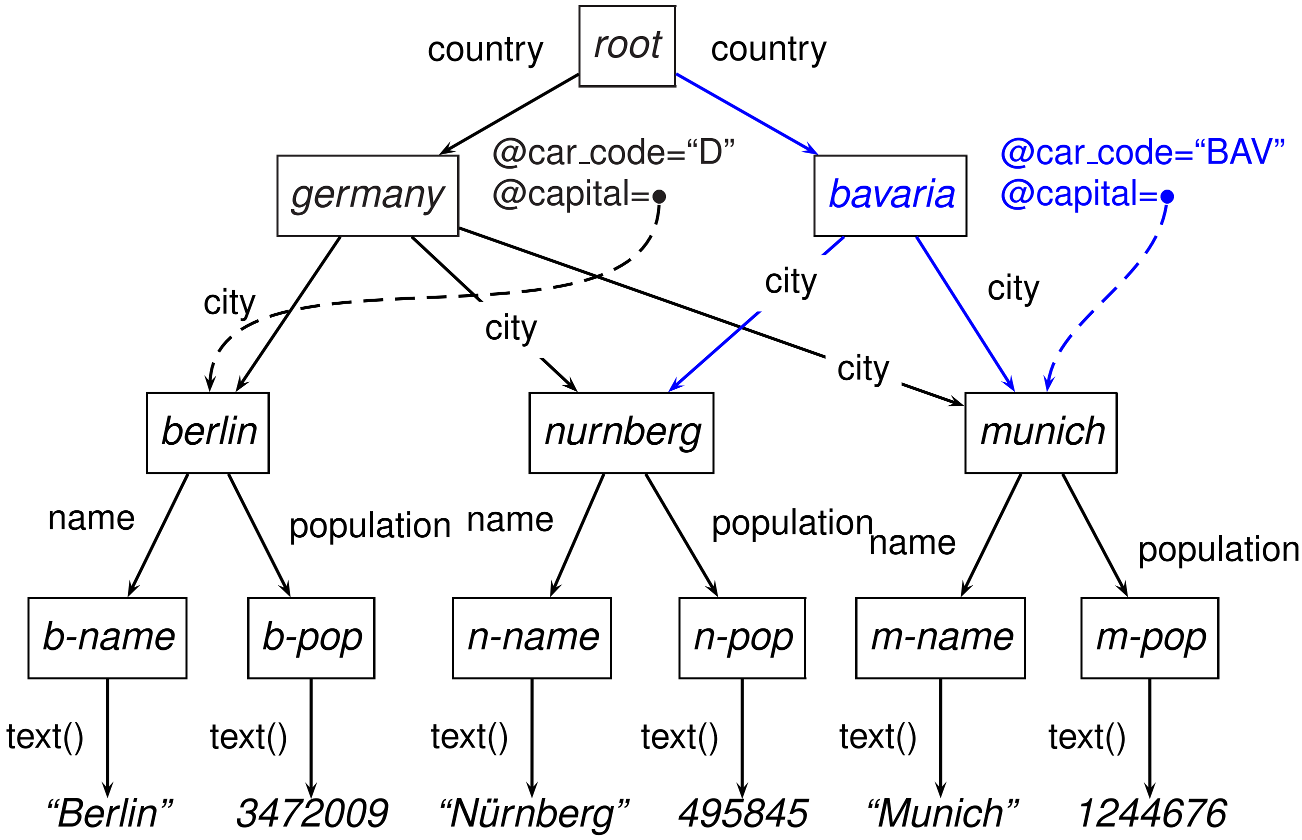
//country→C[@car_code→“BAV”],

//city→X[name/text()=“Munich”],

//city→Y[name/text()=“Nurnberg”].

- city elements are *linked* as subelements.
- extends $\mathcal{A}_{\mathcal{X}}(\mathrm{child}, \textit{bavaria})$ with $(\textit{munich}, \mathrm{city})$ and $(\textit{nurnberg}, \mathrm{city})$.
- crucial for efficient in-place restructuring and integration

Example: Linking



- elements may have multiple parents

Semantics of Rule Heads

Generation of Elements by Path Expressions

C/name[text()→"Bavaria"] :-
    //country→C[@car_code="BAV"].

$\Downarrow$

<country car_code="BAV" capital="city-munich">
  <city>... </city>
  <city>... </city>
  <name>Bavaria</name>
</country>

Atomized:

C[name→_N], _N[text()→"Bavaria"] :-
    root[descendant::country→C], C[@car_code="BAV"].

- variable binding C/*bavaria*.
- insert *bavaria*[child::name→x_1]:
  add an (empty) name subelement (x_1, name) to $\mathcal{A}_{\mathcal{X}}(\textit{bavaria}, \text{child})$.
- bind the local variable $\_N$ to x_1.
- generate the text contents to x_1:
  add ("Bavaria",text) to $\mathcal{A}_{\mathcal{X}}(x_1, \text{child})$.

Data Integration

- objects of different sources represent the same real-world object

$\Rightarrow$ Fusing objects, merging their properties

- synonyms
- not compatible with XML Data Model (DOM, XML Query Data Model)
- graph vs. tree

Data Integration: "Three-level" model

access multiple sources

- "basic" layer: source(s) provide tree structures,
- optionally with namespaces

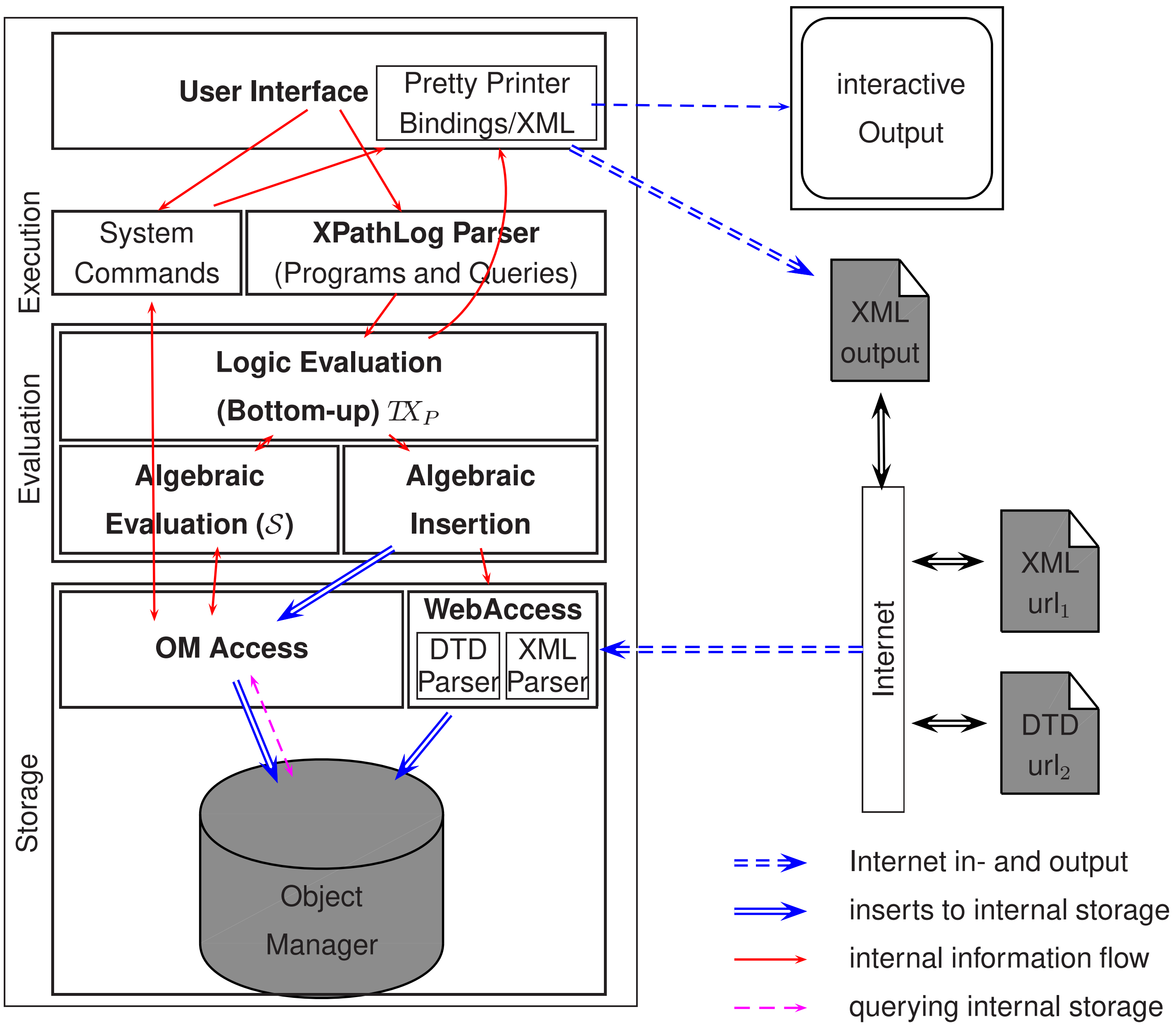
merge data from different sources

- "internal" layer: XTreeGraph
  - overlapping trees
  - multiple parents
  - references
- fuse elements/merge subtrees
- add subelement links
- generate elements
- synonyms for properties

"export" layer: result trees views defined as projections

- root node
- signature

Implementation: LoPiX



RESULTS

- declarative semantics for generating XML with XPath
- graph data model suitable & necessary for integration
- firstly implemented XML updates
- DOM/XML Query Data Model not suitable:
  unique-parent, no fusion, synonyms, linking
- Updates in XML: requires to copy trees
  Problems with IDREFs

CONCLUSION

- practicable approach
  (implementation + case study)
  - Faster than Kweelt, XML-QL, Excelon, and Tamino for queries
- data model
- powerful, flexible language
  - (metadata/schema querying)
  - element creation
  - element fusion
- extension concepts
  (classes, signatures)
- data-driven Web access,
  extensible for XLinks

Contents