Integration of XML Data in XPathLog

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

CAiSE Workshop “Data Integration over the Web”
Interlaken, 4.6.2001

XML

A *data model* for *semi-structured data*

XML Data Model

- tree
- DOM, XML Query Data Model
- tree/container structure, nested elements
  - type (“tag”),
  - contents (ordered),
  - attributes (unordered)
- most popular *representation*: ASCII

EXAMPLE: MONDIAL

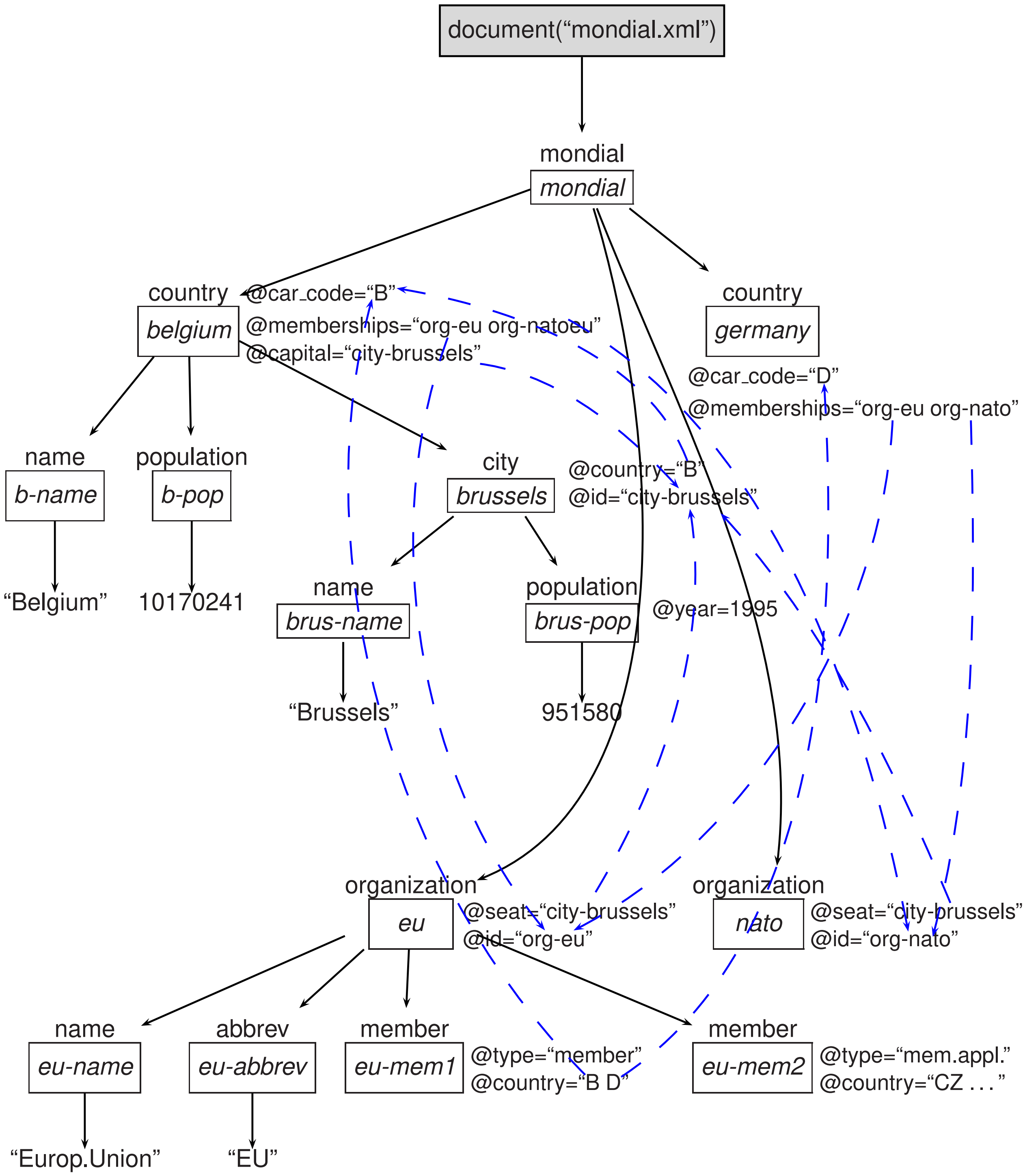
```
<mondial>
 <country car_code=“B” capital=“cty-Brussels”
              memberships=“org-eu org-nato . . . ”>
    <name>Belgium</name>
    <population>10170241</population>
    <city id=“cty-Brussels” country=“B”>
        <name>Belgium</name>
        <population year=“95”>951580</population>
    </city>
    :
 </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



ADDRESSING MECHANISM: XPATH

- based on navigation by path expressions

  //step/step/.../step

- every single *location step* is of the form

  *axis::nodetest[filter]**

- result set consisting of the selected nodes.

STATE OF THE ART

- Most XML concepts are based on XPath
- Not yet any XML update language implemented
  - Problems!

DESIGN DECISIONS

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

XPathLog: Expression Syntax

Extends the XPath syntax

Extend *LocationSteps*

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

- reference expressions may start with variables
- navigation by dereferencing `IDREF` attributes.

Answer set semantics: Annotated result list

(i) a result list, and

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

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”

:

Navigation Variables

?- //Type→X[name/text()→“Monaco”].

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

"THREE-LEVEL" MODEL

Tailored to data integration

1. "basic" layer: source(s) provide tree structures
   W3C XML
2. "internal" layer: internal graph structure, created by integration operations.
   extends XML data model
3. "export" layer: result trees defined as *tree views*.
   W3C XML

DATA MODEL: THE INTERNAL MODEL

- Graph “database”
  - *XML element nodes*: vertices
  - text children are leaf literals
  - *XML attribute nodes*: literals
  - multivalued attributes (`NMTOKENS`, `IDREFS`) are split
  - reference attributes (`IDREF`) are resolved
- names are elements of the universe
- basic axes are stored extensionally (DOM-style)
- $\Rightarrow$ derived axes are defined intensionally
- Canonical mapping from XML instances

XPathLog: Horn Formulas over reference expressions

head(V_1,...,V_n) :- body(V_1,...,V_n)

- Evaluation of Rule bodies = Queries
- Constructive semantics for XPathLog atoms in rule heads

Semantics of Rule Heads

Constructive semantics of XPath expressions

- Definite XPathLog atoms:
  - use only the child and sibling axes
  - no negation, function applications, aggregation, and *proximity position predicates*

  $\Rightarrow$ unambiguous insertions

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

- $host[property{\rightarrow}value]$ modifies $host$
- $host/property\ remainder$
  inserts new element $host/property$ which satisfies $remainder$

Semantics of Rule Heads

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>

RULE HEADS: UPDATES

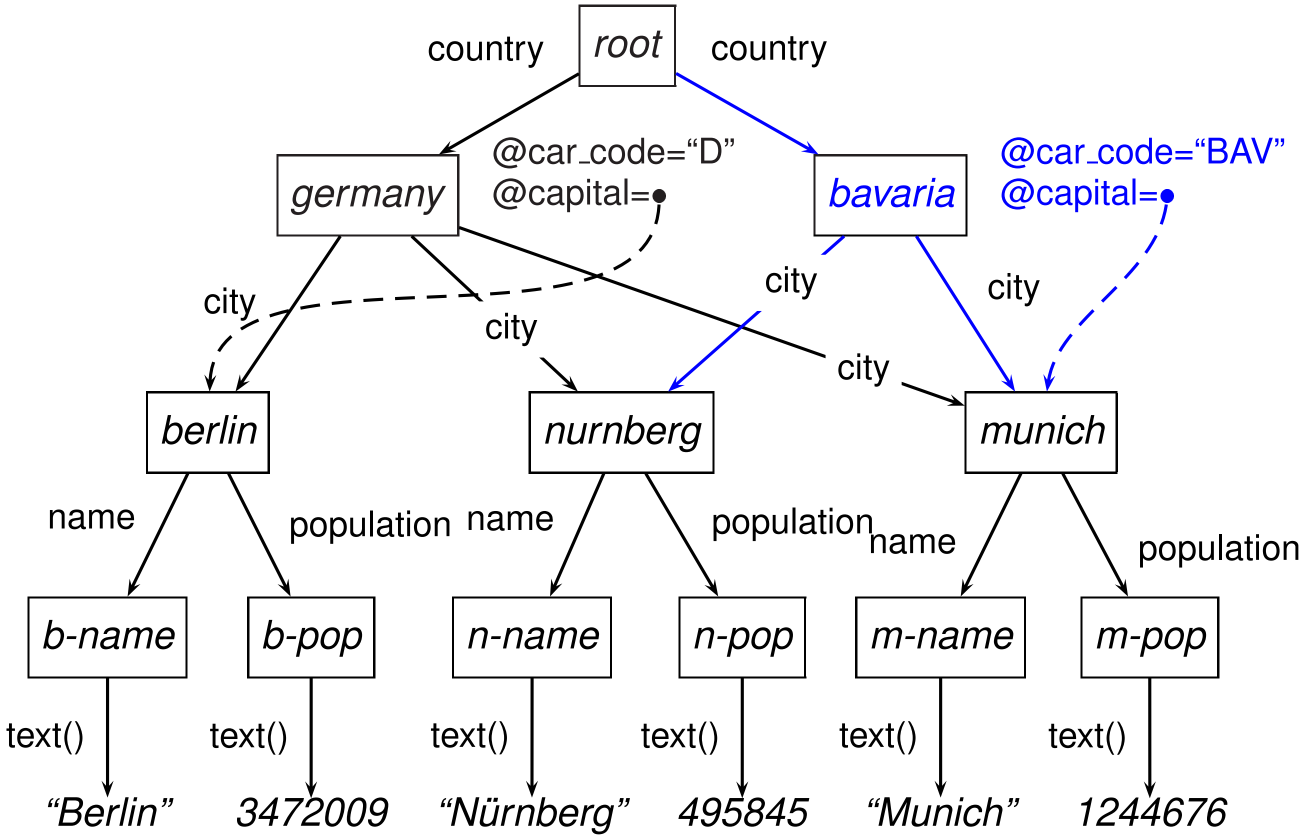
Create new elements: free elements

/country[@car_code→"BAV"].

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"].

Example: Linking



- city elements are *linked* as subelements.

⇒ multiple parents

- crucial for efficient in-place restructuring and integration

EXTENSIONS

- Classes, Class hierarchy
- Signatures

CASE STUDY IN INTEGRATION: THE MONDIAL GEOGRAPHIC DATABASE

XML representations of:

- CIA World Factbook: Countries
- CIA World Factbook: Organizations
- GlobalStatistics: Cities and Administrative Divisions
- ... some additional smaller sources

⇒ Integration

DATA INTEGRATION: OPERATIONS

access multiple sources

- trees, optionally with namespaces

merge data from different sources

- synonyms for properties
- fuse elements/merge subtrees
- add subelement links
- generate elements

Internal database: graph

- overlapping trees
- multiple parents
- references

define views by projection as result trees

- root node
- signature

Data Integration

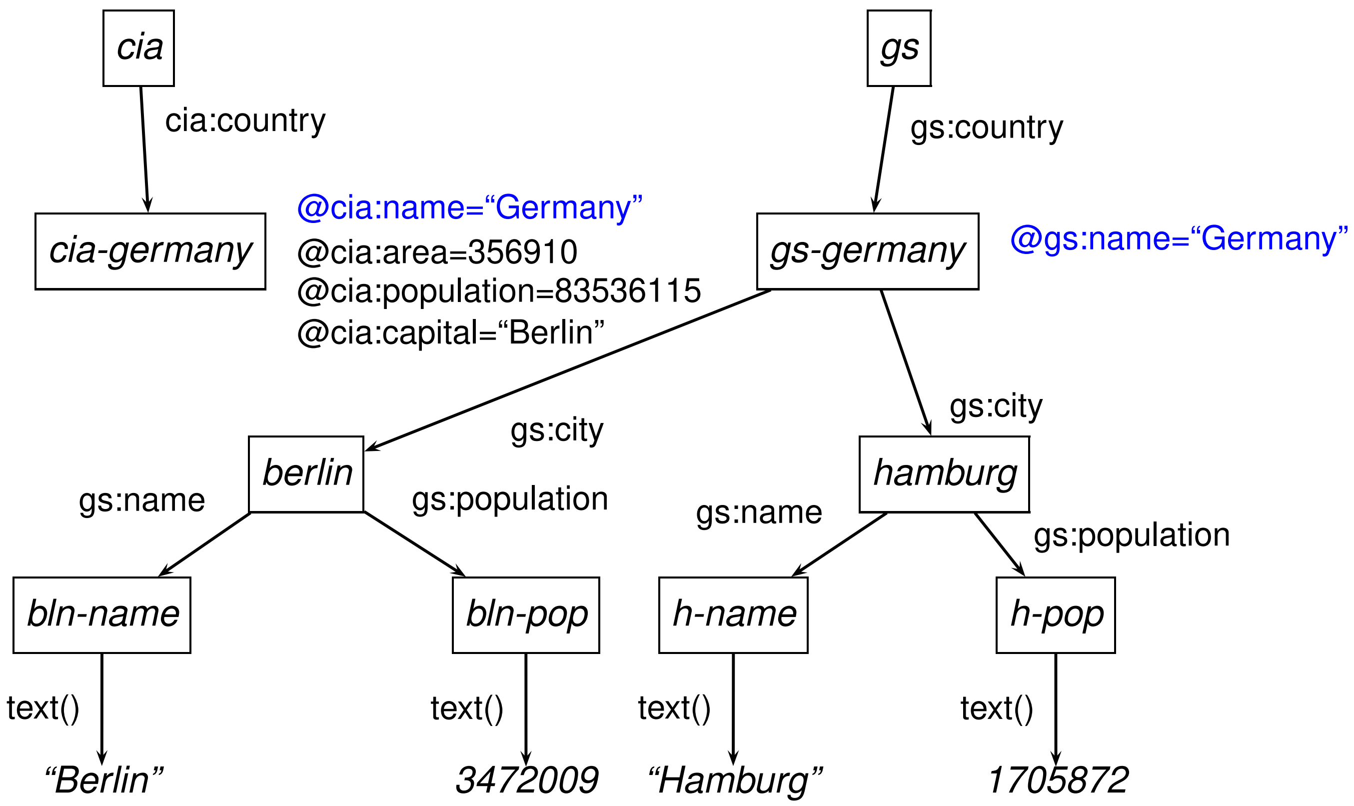
access multiple sources

- trees, optionally with namespaces

Example

Data Sources describing countries:

- cia: name, area, population and capital (by name)
- gs: cities with name, population



SYNONYMS

$$name_1 = name_2$$

$name_2$ is defined to have the same extension as the original property $name_1$.

```
cia:name = name.
cia:area = area.
cia:population = population.
cia:text() = text().
```

- does *not* introduce new children or attribute nodes,
- “only” defines an alternative navigation path,

INTEGRATION: FUSING ELEMENTS AND SUBTREES

Situation

- elements represent the same real-world entity in different sources
- fuse elements into a unified element: $e_1 = e_2$

Resulting element

1. globally replace e_2 in $\mathcal{X}$ by e_1.
2. e_1 is then an element of *both* source trees
3. e_1 collects the attributes and subelements of both original elements.

INTEGRATION: FUSING ELEMENTS AND SUBTREES

Example

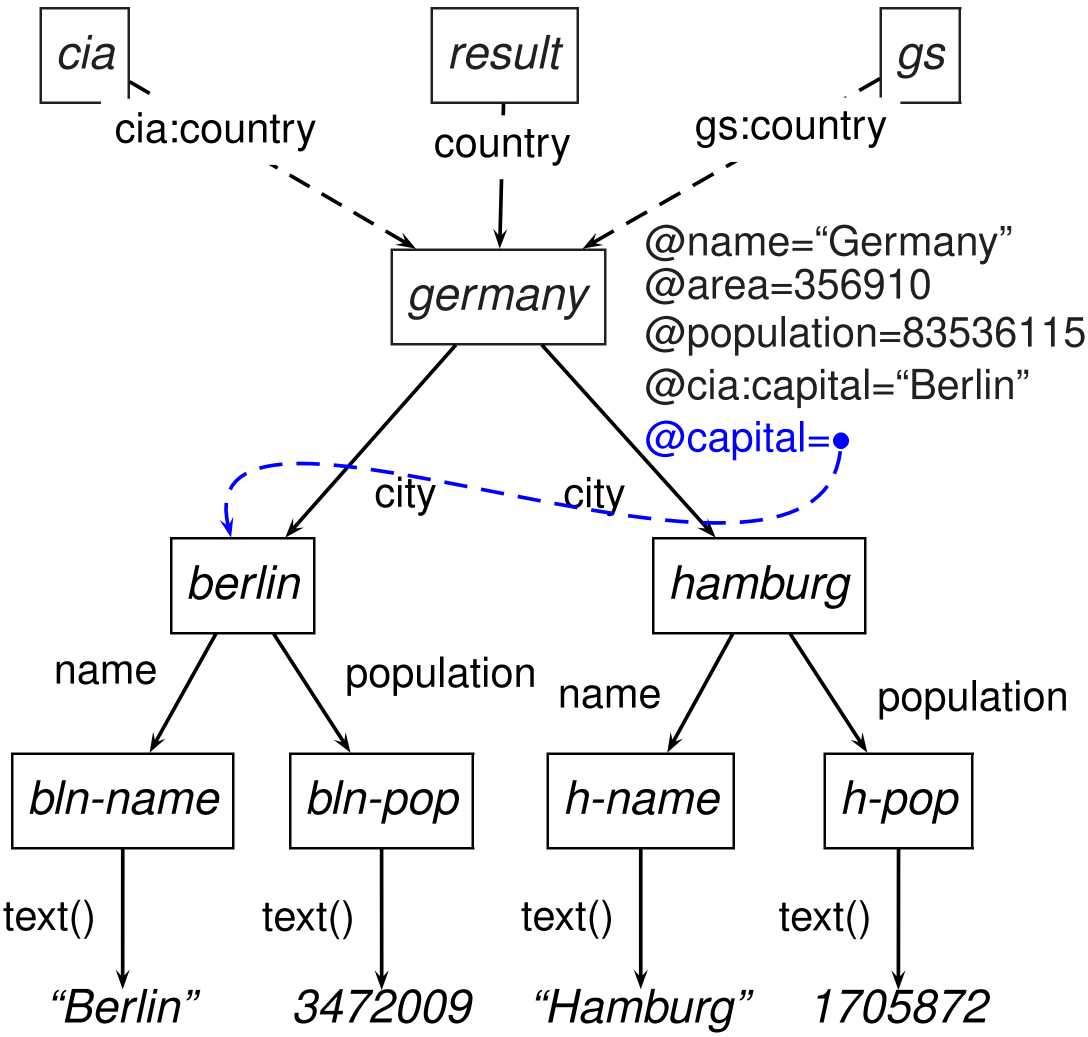
result[country→C1], C1 = C2 :-
    cia/cia:country→C1[@cia:name→N],
    gs/gs:country→C2[@gs:name→N].

Synonyms:

cia:name = name.	gs:city = city.
cia:area = area.	gs:name = name.
cia:population = population.	gs:population = population.
	gs:text() = text().

Integration: Fusing Elements and Subtrees

Example



C[@capital→City] :-

result/country→C[@cia:capital→Name and

city→City[name/text()=Name]].

RESULT VIEWS

Projection by Signatures

Given: a XML graph database, a set of signature atoms, and a root node.

mondial[country⇒country].

country[@name⇒string].

country[@area⇒numeric].

country[@population⇒numeric].

country[@capital⇒city].

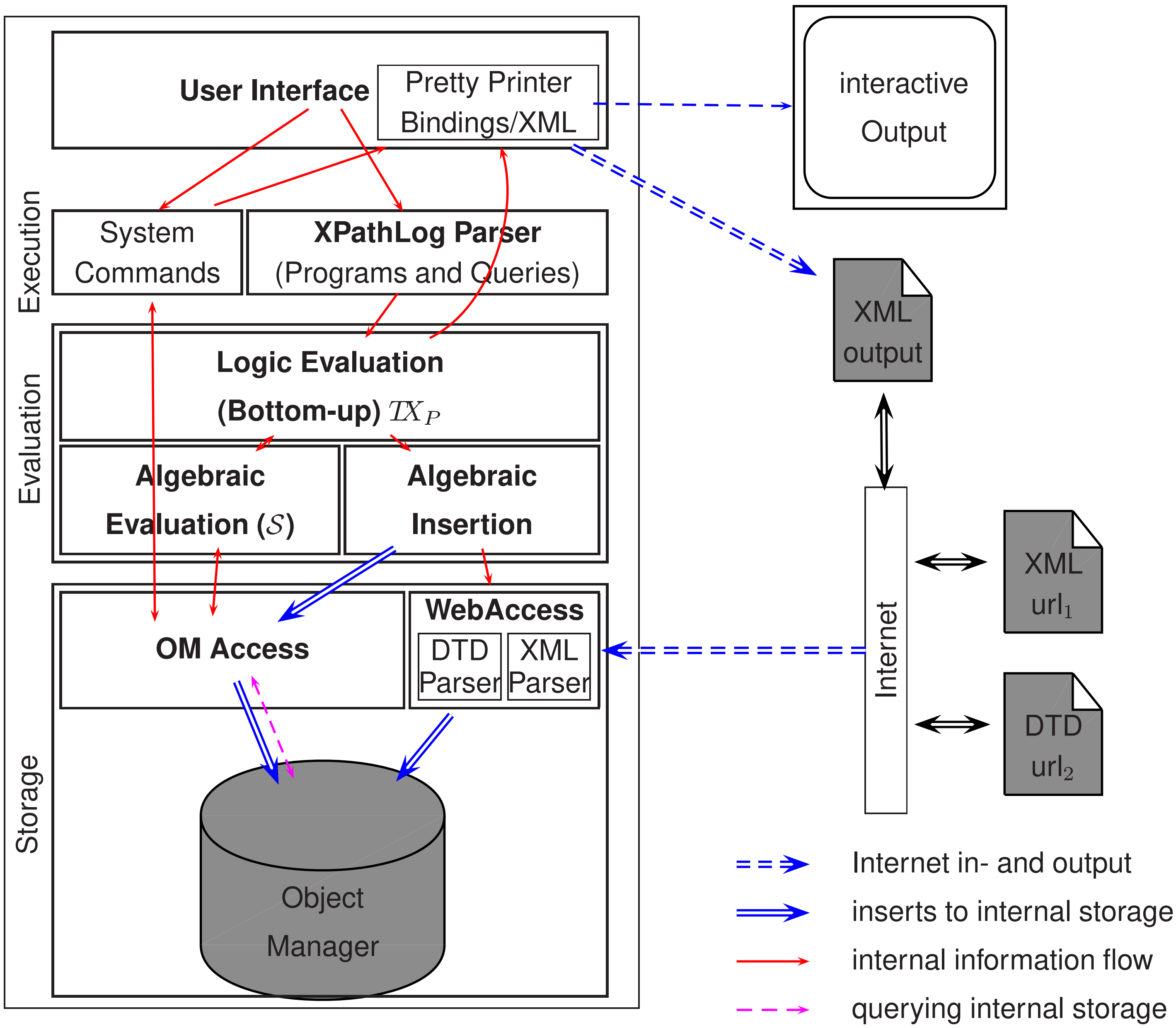
country[@city⇒city].

Further Topics in Integration

- Signature atoms
- Metadata trees (XML Schema)
- XML Ontology trees (XOL, OIL)
  - cross-links
  - metadata reasoning

IMPLEMENTATION: LOPIX

- developed using major components from FLORID



CASE STUDY IN INTEGRATION: EXPERIENCES

Integration of “consistent sources”

(CIA countries, CIA organizations, GlobalStatistics):

- synonyms
- fusing
- reference attributes

Integration of “add-on-1”

- stepwise merging
- stepwise linking

Integration of “add-on-2”

- a lot of “classical” integration problems
- stepwise linking and refinement
- individual, local solutions

Conclusion

- practicable approach
  (implementation + case study)
- declarative semantics for generating XML with XPath
- firstly implemented XML updates
- powerful, flexible language
- graph data model suitable & necessary for integration
- Updates in XML: requires to copy trees
  Problems with IDREFs
- DOM/XML Query Data Model not suitable:
  unique-parent, no fusion, synonyms, linking
- Non-W3C data model, uses W3C notions

Contents