LoPiX: A System for XML Data Integration and Manipulation

Wolfgang May
may@informatik.uni-freiburg.de

Institut für Informatik, Universität Freiburg, Germany

1 Introduction

LoPiX is an implementation of XPathLog [May01b], an XML/XPath-native, rule-based programming language for manipulation and integration of XML documents. The main syntactical constructs are XPath expressions, extended with variables. Due to the close relationship with XPath, the semantics of rules is easy to grasp. In contrast to other approaches, the XPath syntax and semantics is also used for a declarative specification how the database should be *updated*: when used in rule heads, XPath filters are interpreted as specifications of elements and properties which should be added to the database. The LoPiX system provides an implementation of XPathLog tailored to data integration, using a suitable graph-based data model. It extends the pure XPathLog language with schema information obtained from DTDs, a class concept, data-driven Web access and export functionality and data integration functionality [May01c] such as element fusion, synonyms, and tree view projections of the internal database. Binaries of LoPiX together with a detailed paper on XPathLog can be found at [LoP].

2 XPathLog

XPath is the common language for addressing node sets in XML documents. XPathLog extends the XPath syntax with the Datalog-style variable concept and implicit dereferencing. The variables are bound to the names/nodes/literals which result from the respective match: In XPath-Logic steps, $axis :: nodetest$ may be replaced by $axis :: nodetest \to V$, $axis :: V$, or $axis :: V \to W$ where V and W are variables.

Example 1 *Consider the following excerpt of the* Mondial *database [Mon] for illustrations.*

```
<country car_code="CH" capital="cty-Bern"
         memberships="org-efta org-un ...">
    <name>Switzerland</name>
    <city id="cty-Bern" name="Bern"> ...</city>
       :
</country>
```

```
<organization id="org-un" seat="cty-USA-NewYork">
  <name>United Nations</name> <abbrev>UN</abbrev>
  <members type="member" country="F E A ..."/>
</organization>
```

Output Result Set: *The query "?- xpath→V" for any xpath binds V to all nodes belonging to the result set of xpath. For a result set consisting of elements, logical ids are returned:*

```
?- //country[name/text()="Germany"]//city→C.
C/berlin
:
```

Dereferencing: *For all organizations, give the name of the seat city and all names and types of members:*

```
?- //organization
     [abbrev/text()→A and @seat/name/text()→S]
     /members[@type→T]/@country/name/text()→N.
```

One element of the answer set is e.g.,

```
A/"UN"  S/"New York"  T/"member"  N/"Germany"
```

Additionally, variables at nodetest position allow for schema querying and generic handling of properties.

XPathLog **programs** are evaluated by LoPiX regarding XPathLog as an *update language for XML databases*: The evaluation of the rule body yields variable bindings which are propagated to the rule head where facts are added to the database. XPathLog expressions in the head have an *update semantics*: The host of the expression gives the element to be updated or extended; and the / operator and the [...] construct specify which properties should be added or updated (thus, [...] does not act as a filter, but as a *constructor*).

Example 2 *Switzerland gets a data code and is made a member of the European Union:*

```
C[@datacode→"ch"], C[@memberships→O] :-
     //country→C[@car_code="CH"],
     //organization→O[abbrev/text()→"EU"].
```

**Proceedings of the 27th VLDB Conference,
Roma, Italy, 2001**

results in

```
<country datacode="ch" car_code="CH"
         memberships="org-efta org-un org-eu . . . "> . . .
</country>
```

Elements. Elements can either be created as *free* elements by atoms of the form /name[...] (meaning "some element of type name"), or as subelements.

Example 3 *We create a new (free) country element with some properties:*

```
/country[@car_code→"BAV" and @capital→X
         and city→X and city→Y] :-
    //city→X[name/text()="Munich"],
    //city→Y[name/text()="Nurnberg"].
```

Elements are assigned as subelements to existing elements using filter syntax in the rule head.

Additionally, subelements can be created by *path expressions* in the rule head which create nested elements which satisfy the given path expression (according to its atomic steps). XPathLog also allows to have variables at name position. Thus, new structures can be generated dependent on the metadata of the original ones.

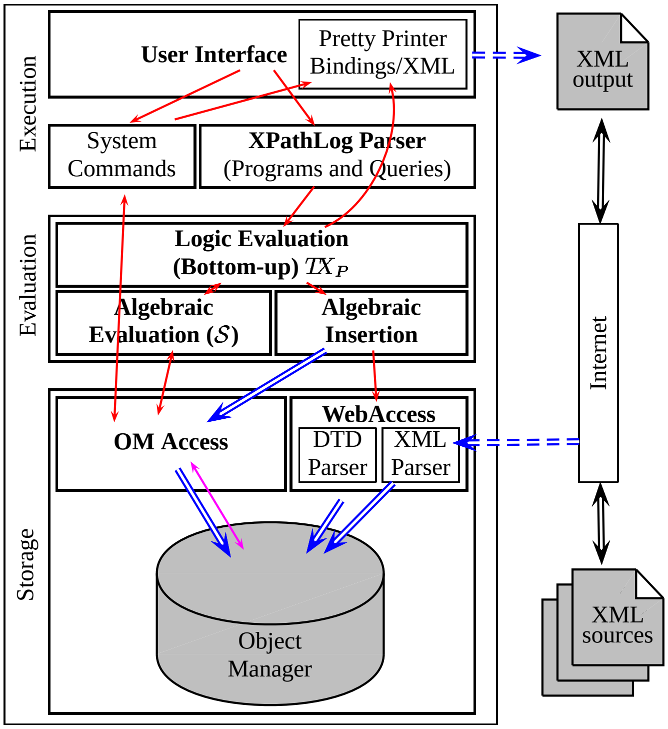

Figure 1: Architecture of the LoPiX System

3 The LoPiX System

LoPiX (Logic Programming in XML) is an implementation of XPathLog. The LoPiX architecture is shown in Figure 1. The central deductive engine is an adaptation of the *Evaluation* component of Florid [HKL+98]. The storage component consists of an *Object Manager* which stores the XML document(s) together with meta information such as indexes, signatures, and class information. The *OMAccess* module maps queries and insertions coming from the *Algebraic Evaluation* and *Algebraic Insertion* to the storage. Web access is controlled by rule heads, allowing for a completely data-driven strategy. New documents are fetched by their *url*s depending on information and links (i.e., urls or XLinks) found in already known documents. They are added to the database using the *WebAccess* component which consists of an *XML Parser* and a *DTD Parser* (handling signature information). The *Execution* component handles system commands, user queries and program evaluation (feeded into the *Evaluation* component).

4 Demonstration Outline

The demonstration shows the use of LoPiX as a tool for XML data integration by the case study [Mon] – from which the above examples are taken – which integrates XML representations of data sources on the Web to a geographic database. The integration process illustrates data-driven access to data sources, identification of relevant entities for the result document, generation of the resulting XML tree, handling of overlapping sources, handling data inconsistencies, different-name-for-an-object problems (german-language and english-language sources), heuristics, incremental insertion of facts into the result tree, generation of cross-references, and final output.

References

[HKL+98] R. Himmeröder, P.-T. Kandzia, B. Ludäscher, W. May, and G. Lausen. Search, Analysis, and Integration of Web Documents: A Case Study with FLORID. In *Proc. Intl. Workshop on Deductive Databases and Logic Programming (DDLP'98)*, 1998.

[LoP] The LoPiX System. http://www.informatik.uni-freiburg.de/ may/lopix/.

[May01a] W. May. A Logic-Based Approach for Declarative XML Data Manipulation. Technical report, Universität Freiburg, Institut für Informatik, 2001. Available from [LoP].

[May01b] W. May. XPathLog: A Declarative, Native XML Data Manipulation Language. In *International Database Engineering and Applications Workshop (IDEAS'01)*. IEEE Computer Science Press, 2001.

[May01c] W. May. Integration of XML Data in XPathLog. In *CAiSE Workshop "Data Integration over the Web" (DIWeb'01)*, 2001.

[Mon] The Mondial Database. http://www.informatik .uni-freiburg.de/ may/Mondial/.