Analyzing Fuzzy Logic Computations with Fuzzy XPath

Jesús M. Almendros-Jiménez
Alejandro Luna Tedesqui
Ginés Moreno Valverde
Carlos Vázquez Pérez-Íñigo

Dep. of Languages and Computation
University of Almería, Spain

Dep. of Computing Systems,
University of Castilla-La Mancha, Spain

Prole 2013 – Madrid, Spain
Sep 18-20, 2013
Outline of the talk

✓ Introduction
✓ FuzzyXPath
✓ MALP and FLOPER
✓ Exploring Trees
✓ Conclusions
Aim of the work

• We focus on the ability of Fuzzy XPath for exploring derivation trees generated by FLOPER.

• This serves as a debugging/analyzing tool for
  – Discovering fuzzy computed answers
  – Performing depth/breadth-first traversals
  – Finding non fully evaluated branches
  – …

• Reinforcing the bi-lateral synergies between Fuzzy Xpath and FLOPER.
XML Path Language

- The *XPath* language has been proposed as a standard for XML querying.
- Based on the description of the path in the XML tree to be retrieved.
- XPath allows to specify the name of nodes (tags) and attributes together with boolean conditions.
- XPath querying mechanism is based on a boolean logic.

Examples:

- `/bib/book/title`
- `/bib/book/title[text = "Don Quijote de la Mancha"]`  
  Absolute Path
- `/bib/book[@price < 30 and @year < 2006]`

- `//title`
- `//book[@price < 25]`  
  Relative Path
FuzzyXPath

FuzzyXPath is an extension of the XPath query language for the handling of flexible queries:

- Two structural constraints called *deep* and *down* for which a certain degree of relevance can be associated.
- Fuzzy Operators *and*, *or* and *avg* for XPath conditions.

FuzzyXPath is defined by means of the following rules:

```
xpath := [deepdown]path
path := literal | text() | node | @att | node/path | node//@path
node := QName | QName[cond]
cond := path op path
deepdown := DEEP=degree; DOWN=degree
op := > | = | < | and | or | avg
```
Structural constraints **DEEP/DOWN**

A Fuzzy XPath expression can be adorned with

\[
\llbracket \text{DEEP} = r_1, \text{DOWN} = r_2 \rrbracket
\]

which means that the *deepness* of elements is penalized by \( r_1 \) and that the *order* of elements is penalized by \( r_2 \), and such penalization is proportional to the distance.
Fuzzy Logic Operators

The classical *and* and *or* connectives admit here a fuzzy behavior based on fuzzy logic, as also occur with the *avg* operator.

- \( \&_p(x, y) = x \cdot y \)
- \( \lor_p(x, y) = x + y - x \cdot y \)
- \( \@_{avg}(x, y) = (x + y)/2 \)

<table>
<thead>
<tr>
<th>p ( \in [0, 1] )</th>
<th>q ( \in [0, 1] )</th>
<th>( p \land q )</th>
<th>( p \lor q )</th>
<th>( @_{avg} q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.7</td>
<td>0.14</td>
<td>0.76</td>
<td>0.45</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4</td>
<td>0.32</td>
<td>0.88</td>
<td>0.60</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Input XML document in our examples

```xml
<bib>
  <book year="2001" price="45.95">
    <title>Don Quijote de la Mancha</title>
    <author>Miguel de Cervantes Saavedra</author>
    <publications>
      <book year="1997" price="35.99">
        <title>La Galatea</title>
        <author>Miguel de Cervantes Saavedra</author>
        <publications>
          <book year="1994" price="25.09">
            <title>Los trabajos de Persiles y Sigismunda</title>
            <author>Miguel de Cervantes Saavedra</author>
          </book>
          <book year="1988" price="12.5">
            <title>La Dragoneraz</title>
            <author>Felix Lope de Vega y Carpio</author>
          </book>
        </publications>
      </book>
    </publications>
  </book>
  <book year="1999" price="23.65">
    <title>La Celestina</title>
    <author>Fernando de Rojas</author>
  </book>
  <book year="2005" price="29.95">
    <title>William Shakespeare</title>
    <publications>
      <book year="2003" price="22.5">
        <title>Romeo y Julieta</title>
        <author>William Shakespeare</author>
      </book>
    </publications>
  </book>
  <book year="2007" price="22.95">
    <title>Las ferias de Madrid</title>
    <author>Felix Lope de Vega y Carpio</author>
    <publications>
      <book year="1996" price="27.5">
        <title>El remedio en la desdicha</title>
        <author>Felix Lope de Vega y Carpio</author>
      </book>
      <book year="1998" price="12.5">
        <title>La Dragontea</title>
        <author>Felix Lope de Vega y Carpio</author>
      </book>
    </publications>
  </book>
</bib>
```
Example Nº 1

Resulting XML Document:

```xml
<result>
  <title rsv="0.81">Don Quijote de la Mancha</title>
  <title rsv="0.52488">La Galatea</title>
  <title rsv="0.340122">Los trabajos de Persiles ...</title>
  <title rsv="0.648">La Celestina</title>
  ...<title rsv="0.214991">La Dragontea</title>
</result>
```

```
<book year="2001" price="45.95">
  <title>Don Quijote de la Mancha</title>
  <author>Miguel de Cervantes Saavedra</author>
  <publications>
    ...<book year="1999" price="25.65">
      <title>La Galatea</title>
      ...<author>Miguel de Cervantes Saavedra</author>
      <publications>
        ...</book>
  </publications>
</book>
```

```
0.81=0.9^2
0.52488=0.99*0.8
0.340122=0.9^2*0.8
0.648=0.9^2*0.8
0.5184=0.9^2*0.8
0.335923=0.9^3*0.8
0.41472=0.9^2*0.8^3
0.268739=0.9^2*0.8^4
0.214991=0.9^4*0.8^3
```
Example Nº 2

\[ /bib/book[@price<30 \text{ avg } @year<2006] \]

Resulting XML Document:

```xml
<result>
  <book rsv="0.5" year="2001" price="45.95">
    <title>Don Quijote de la Mancha</title>…
  </book>
  <book rsv="1.0" year="1999" price="25.65">
    <title>La Celestina</title>…
  </book>
  <book rsv="1.0" year="2005" price="29.95">
    <title>Hamlet</title>…
  </book>
  <book rsv="0.5" year="2007" price="22.95">
    <title>Las ferias de Madrid</title>…
  </book>
</result>
```
Example Nº 3

$$\text{[DEEP=0.9;DOWN=0.8]} //\text{book[@price>25 and @price<30] avg (@year<2000 or @year>2006)} /\text{title}$$

Resulting XML Document:

```
<result>
  <title rsv="0.3645">La Galatea</title>
  <title rsv="0.59049">Los trabajos de Persiles y Sigismunda</title>
  <title rsv="0.72">La Celestina</title>
  <title rsv="0.288">Hamlet</title>
  <title rsv="0.2304">Las ferias de Madrid</title>
  <title rsv="0.373248">El remedio en la desdicha</title>
  <title rsv="0.149299">La Dragontea</title>
</result>
```
MALP: *Multi-Adjoint Logic Programming*

PROLOG-like language with predicates, constants, functions and variables but a wide class of connectives!!!

\[ \&1, \&2, \ldots, \&k \quad \text{(conjunctions)} \]
\[ \vee 1, \vee 2, \ldots, \vee l \quad \text{(disjunctions)} \]
\[ \leftarrow 1, \leftarrow 2, \ldots, \leftarrow m \quad \text{(implications)} \]
\[ @1, @2, \ldots, @n \quad \text{(aggregations)} \]

Moreover, instead of naive \{true, false\}, we use multi–adjoint lattices to model truth degrees \{ L, \leq, \leftarrow 1, \& 1, \ldots, \leftarrow n, \& n \}. For instance:

\[ \{ [0, 1], \leq, \leftarrow \text{Luka}, \& \text{Luka}, \leftarrow \text{Prod}, \& \text{Prod}, \leftarrow \text{Godel}, \& \text{Godel} \} \]
Syntax of MALP programs

Assume some connective definitions like:

\[ \&\text{godel}(x_1, x_2) = \min(x_1, x_2) \]
\[ \&\text{prod}(x_1, x_2) = x_1 \times x_2 \]
\[ \&\text{luka}(x_1, x_2) = \min(1, x_1 + x_2) \]
\[ \&\text{aver}(x_1, x_2) = (x_1 + x_2)/2 \]

A program is a set of “weighted” rules \( A \leftarrow_i B \) with \( v \):

\[ R_1 : \text{oc}(X) \leftarrow_{\text{prod}} \text{s}(X) \&_{\text{prod}} (f(X) \@_{\text{aver}} w(X)) \quad \text{with 1.} \]
\[ R_2 : \text{s}(\text{madrid}) \quad \text{with 0.8.} \]
\[ R_3 : \text{f}(\text{madrid}) \quad \text{with 0.8.} \]
\[ R_4 : \text{w}(\text{madrid}) \quad \text{with 0.9.} \]
\[ R_5 : \text{s}(\text{tokyo}) \quad \text{with 0.9.} \]
\[ R_6 : \text{f}(\text{tokyo}) \quad \text{with 0.7.} \]
\[ R_7 : \text{w}(\text{tokyo}) \quad \text{with 0.6.} \]
\[ R_8 : \text{s}(\text{istambul}) \quad \text{with 0.3.} \]
\[ R_9 : \text{f}(\text{istambul}) \quad \text{with 0.4.} \]
\[ R_{10} : \text{w}(\text{istambul}) \quad \text{with 0.8.} \]
\[ R_{11} : \text{s}(\text{baku}) \quad \text{with 0.3.} \]
\[ R_{12} : \text{f}(\text{baku}) \quad \text{with 0.2.} \]
\[ R_{13} : \text{w}(\text{baku}) \quad \text{with 0.5.} \]
FLOPER: *Fuzzy LOgic Programming Environment for Research*

http://dectau.uclm.es/floper/

PROLOG has been largely used for four purposes.....

1. **Implementing the tool** (about 1000 clauses, DCG’s)

2. **Compiling fuzzy programs** (compiled code in Prolog):
   - **HIGH LEVEL**: transparent execution of goals
   - **LOW LEVEL**: drawing derivations and trees

3. **Modeling multi-adjoint lattices** (to represent truth degrees)
FLOPER: *Fuzzy LOgic Programming Environment for Research*

**Basic options for:**
- **Loading** a prolog file with extension “.pl”
- **Parsing** a “.fpl” fuzzy program. The resulting Prolog code is also asserted into the system
- **Saving** the generated Prolog code into a “.pl” file
- **Listing** both the fuzzy and Prolog code
- **Clean, Stop and Quit**

**Advanced options for:**
- **Running** a fuzzy program after introducing a goal
- **Drawing** derivations and unfolding trees (varying its depth and the level of detail of interpretive steps)
- **Loading/showing** multi-adjoint lattices (“.pl” files)
Compiling to PROLOG and running MALP programs

Rule: \( oc(X) \leftarrow \text{prod} \ s(X) \ \& \text{prod} \ (f(X) \ @\text{aver} \ w(X)) \) \quad \text{with} \ 1. \\

\[
\begin{align*}
oc(X, TV0) : & \quad \text{s}(X, TV1), \ f(X, TV2), \ w(X, TV3), \\
& \quad \text{agr}_\text{aver}(TV2, TV3, TV4), \\
& \quad \text{and}_\text{prod}(TV1, TV4, TV0), \\
\end{align*}
\]

GOAL: \( oc(X) \rightleftharpoons ? \rightleftharpoons oc(X, \text{Truth} _\text{degree}) \)

OUTPUT:

\[
\begin{align*}
[X = \text{madrid}, \qquad \text{Truth} _\text{degree} = 0.68] \\
[X = \text{istambul}, \qquad \text{Truth} _\text{degree} = 0.18] \\
[X = \text{tokyo}, \quad \quad \quad \text{Truth} _\text{degree} = 0.585] \\
[X = \text{baku}, \quad \quad \quad \text{Truth} _\text{degree} = 0.105] \\
\end{align*}
\]
Multi-Adjoint lattices modeled in PROLOG

Prolog file modeling an infinite lattice of real numbers $[0, 1]$:

```prolog
member(X) :- number(X), 0=<X,X=<1.  leq(X, Y) :- X=<Y.  bot(0).  top(1).
agr_aver(X, Y, Z) :- pri_add(X, Y, U), pri_div(U, 2, Z).
or_prod(X, Y, Z) :- pri_prod(X, Y, U1), pri_add(X, Y, U2), pri_sub(U2, U1, Z).
and_prod(X, Y, Z) :- pri_prod(X, Y, Z). and_godel(X, Y, Z) :- pri_min(X, Y, Z).
and_luka(X, Y, Z) :- pri_add(X, Y, U1), pri_sub(U1, 1, U2), pri_max(0, U2, Z).
pri_add(X, Y, Z) :- Z is X+Y.  pri_min(X, Y, Z) :- (X=<Y, Z=X; X>Y, Z=Y).
pri_sub(X, Y, Z) :- Z is X-Y.  pri_max(X, Y, Z) :- (X=<Y, Z=Y; X>Y, Z=X).
pri_prod(X, Y, Z) :- Z is X * Y.  pri_div(X, Y, Z) :- Z is X/Y.
```

Prolog file modeling the lattice used in our application:

```prolog
member(tv(N, L)) :- number(N), 0=<N,N=<1, (L=[]; L=[_|_]).
leq(tv(X1, _), tv(X2, _)) :- X1 =< X2.
bot(tv(0, _)).         top(tv(1, _)).
and_prod(tv(X1, X2), tv(Y1, Y2), tv(Z1, Z2)) :- pri_prod(X1, Y1, Z1), pri_app(X2, Y2, Z2).
or_prod(tv(X1, X2), tv(Y1, Y2), tv(Z1, Z2)) :- pri_prod(X1, Y1, U1), pri_add(X1, Y1, U2),
pri_sub(U2, U1, Z1), pri_app(X2, Y2, Z2).
agr_aver(tv(X1, X2), tv(Y1, Y2), tv(Z1, Z2)) :- pri_add(X1, Y1, Aux), pri_div(Aux, 2, Z1),
pri_app(X2, Y2, Z2).
pri_add(X, Y, Z) :- Z is X+Y.  pri_sub(X, Y, Z) :- Z is X-Y.
pri_prod(X, Y, Z) :- Z is X * Y.  pri_div(X, Y, Z) :- Z is X/Y.
pri_app([], X, X).
```

[Analysis of Fuzzy Logic Computations with Fuzzy XPath](#)  Prolog 2013  Sep 18-20, 2013  17/28
Derivation trees in FLOPER

Fuzzy Computed Answers

Analyzing Fuzzy Logic Computations with Fuzzy XPath  Prole 2013  Sep 18-20, 2013  18/28
Exploring Trees with Fuzzy XPath

<node>
  <rule>R0</rule>
  <goal>oc(X)</goal>
  <substitution>{}\</substitution>
  <children>
    <node>
      <rule>R1</rule>
      <goal>and\_prod(s(X),agr\_aver(f(X),w(X)))</goal>
      <substitution>{X1/X}</substitution>
      <children>
        ...
      </children>
    </node>
  </children>
</node>

XML file

Derivation tree
Obtaining *fuzzy computed answers*

- To obtain the set of fuzzy computed answers, we have two alternative fuzzy XPath queries:
  - `//node[/rule/text()=result]`
  - `//node[children[not(text())]]`

that, in this case, finds the leaves of the tree.
Obtaining *fuzzy computed answers*

- Note that these queries don’t necessarily coincide

**PROGRAM**

\[ p(a) \text{ with 0.8.} \]

\[ p(X) <\text{prod} p(s(s(s(X)))) \text{ with 0.9.} \]

\[ p(b) \text{ with 0.6.} \]

**QUERY**

\[ q(X) \@\text{aver} \ p(X). \]
Obtaining *fuzzy computed answers*

//node[/rule/text()=result]

TWO ANSWERS

< 0.4 ; \{X/a\} >

< 0.3 ; \{X/b\} >
Obtaining *fuzzy computed answers*

```
//node[children[not(text())]]
```
Partial breadth-first traversal tree

- Here we make use of DEEP and FILTER

```
[FILTER=0.5][DEEP=0.9]//node/goal
```
Partial depth-first traversals tree

- We make use of DOWN and DEEP

\[ \text{FILTER}=0.5 \text{][DOWN}=0.7\] //node/goal

\( \text{Rsv} = 1 \quad \text{Rsv} = 0.7 \)
More complex queries

- Focus of interest:
  - Wheather (→ goal contains “w(“)
  - Istambul (→ substitution contains “istambul”)
- Combine with prioritized average

```
node[
  /goal[contains(text(),"w(")]
  aver{1,2}
  substitution[contains(text(),"istambul")]
]//goal»
```
More complex queries

Rsv = 0.33

Rsv = 0.67

Rsv = 1
Conclusions and future work

1. We show the mutual benefits between fuzzy XPath and FLOPER.

2. FLOPER was conceived as a tool for implementing fuzzy applications –such as fuzzy XPath-, and allows the debugging of those programs by showing their derivation trees in, among other formats, XML files.

3. In this work we have seen that the fuzzy XPath interpreter is able to analyze derivation trees generated by FLOPER.

4. For the future we plan to study the role that the fuzzy XPath debugger (that offers alternative queries) should play for developing applications in FLOPER.