A XPath Debugger Based on Fuzzy Chance Degrees

Jesús M. Almendros-Jiménez[1], Alejandro Luna[2], and Ginés Moreno[2]

[1] Dept. of Languages and Computation, University of Almería, Spain
jalmen@ual.es
[2] Dept. of Computing Systems, University of Castilla-La Mancha, Spain
Alejandro.Luna@alu.uclm.es, Gines.Moreno@uclm.es

Abstract. We describe how we can manipulate an XPath expression in order to obtain a set of alternative XPath expressions that match to a given XML document. For each alternative XPath expression we will give a chance degree that represents the degree in which the expression deviates from the initial expression. Thus, our work is focused on providing the programmer a repertoire of paths that (s)he can use to retrieve answers. The approach has been implemented and tested.

The eXtensible Markup Language (XML) provides a very simple language to represent the structure of data, using tags to label pieces of textual content, and a tree structure to describe the hierarchical content. The XPath language [2] was designed as a query language for XML in which the path of the tree is used to describe the query. This work is motivated by the evidence that users frequently omit some tags on paths, add more than necessary, or employ wrong tags, in order to help them when formulating XPath queries.

XPath debugging has to take into account the previous considerations. Particularly, there is an underlying notion of *chance degree*. When the programmer makes mistakes, the number of bugs can be higher or lower, and the chance degree is proportional to them. Moreover, there are several ways on which each bug can be solved, and therefore the chance degree is also dependent from the number of solutions for each bug.

Our debugging method acts on a given initial XPath query Q preceded by the $[\texttt{DEBUG} = r]$ command, where r is a real number in the unit interval used at debugging time for labeling *swapping* [1], *deletion* and *jumping* actions. So, assume that we plan to process "[DEBUG=0.5]/bib/book/title" with respect to the following XML document:

```
<bib>
    <name>Classic Literature</name>
    <book year="2001" price="45.95">
        <title>Don Quijote de la Mancha</title>
        <author>Miguel de Cervantes Saavedra</author>
        <references>
```

[1] We are nowadays equipping our tool with techniques for automatically extracting *similarity degrees* between tags from standard internet resources such as WordNet.

P. Herrero et al. (Eds.): OTM 2012 Workshops, LNCS 7567, pp. 669–672, 2012.
© Springer-Verlag Berlin Heidelberg 2012

```
            <novel year="1997" price="35.99">
                <name>La Galatea</name>
                <author>Miguel de Cervantes Saavedra</author>
                <references>
                    <book year="1994" price="25.99">
                        <title>Los trabajos de Persiles y Sigismunda</title
                            >
                        <author>Miguel de Cervantes Saavedra</author>
                    </book>
                </references>
            </novel>
        </references>
    </book>
    <novel year="1999" price="25.65">
        <title>La Celestina</title>
        <author>Fernando de Rojas</author>
    </novel>
</bib>
```

Our technique produces a set of alternative queries $Q_1, ..., Q_n$, packed into an output XML document, each one adorned with attributes about the changes that deviates Q_i from Q. The set of proposals is sorted with respect to a *CD* key meaning that, as much changes are performed on Q_i and as more *traumatic* they are w.r.t to Q, then its "*Chance Degree*" becomes lower according a policy based on the *product fuzzy logic*.

```
<result>
  <query cd="1.0">/bib/book/title</query>
  <query cd="0.8" book="novel">/bib/[SWAP=0.8]novel/title</query>
  <query cd="0.5" bib="//">/[JUMP=0.5]//book/title</query>
  <query cd="0.45" book="" title="name">
        /bib/[DELETE=0.5][SWAP=0.9]name
  </query>
  <query cd="0.225" bib="" book="//" title="name">
        /[DELETE=0.5][JUMP=0.5]//[SWAP=0.9]name
  </query>
  ...
</result>
```

So, executing the first alternative -which coincides with the original query-, we can retrieve "Don Quijote de La Mancha", the second query returns "La Celestina", the third one adds "Los trabajos de Persiles y Sigismunda" to "Don Quijote", whereas the case with commands `DELETE`, `JUMP` and `SWAP` of lowest CD value (i.e., 0.225) is even able to produce "Classic Literature", as shown in Figure 2. The *RSV* key -*Retrieval Status Value*- means the satisfaction degree of each solution with respect to the considered query.

In order to explain how our technique works, let us consider a path P of the form "$/tag_1/.../tag_i/tag_{i+1}/...$", where $P[i]$ references tag_i in P: this notation is used in the following sketched algorithm where symbols `Q` and `B` refers to the original Xpath query and any *branch* of the input XML document, respectively.

```
For each branch B in the input document
  For each tag Q[i] in the input query
    If Q[i] and B[i] are ‘‘syntactically’’ different tags then
      Add a new query with SWAP if Q[i] and B[i] are ‘‘similar’’ tags
      Add a new query with JUMP if Q[i] coincides with B[j], being j>i
      Add a new query with DELETE if Q[i] coincides with B[i+1]
```

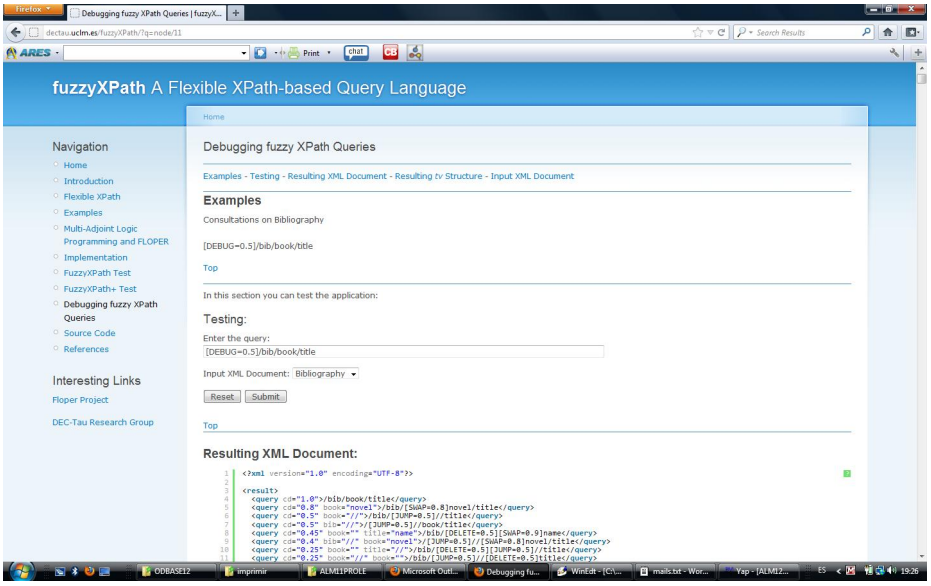

Fig. 1. Screen-shot of an on-line work session with the XPath debugger

```
<result>
  <name rsv="0.225">Classic Literature</name>
  <name rsv="0.028125">La Galatea</name>
</result>
```

Fig. 2. Execution of query "/[DELETE=0.5][JUMP=0.5]//[SWAP=0.9]name"

```
<result>
    <query cd="0.54" classic="" cost="price">
         /bib/[DELETE=0.6]//book
                [(@year < 2000) avg{3,1} ([SWAP=0.9]@price < 50)]/title
    </query>
    <query cd="0.54" classic="//" cost="price">
         /bib/[JUMP=0.6]//book
                [(@year < 2000) avg{3,1} ([SWAP=0.9]@price <50)]/title
    </query>
    <query cd="0.432" classic="" book="novel" cost="price">
          /bib/[DELETE=0.6]//[SWAP=0.8] novel
                [(@year < 2000) avg{3,1} ([SWAP=0.9]@price < 50)]/title
    </query>
   .........
   ......
   ..
</result>
```

Fig. 3. Debugging effects on XPath conditions
[DEBUG=0.6]/bib/classic//book[@year < 2000 avg{3,1} @cost < 50]/title

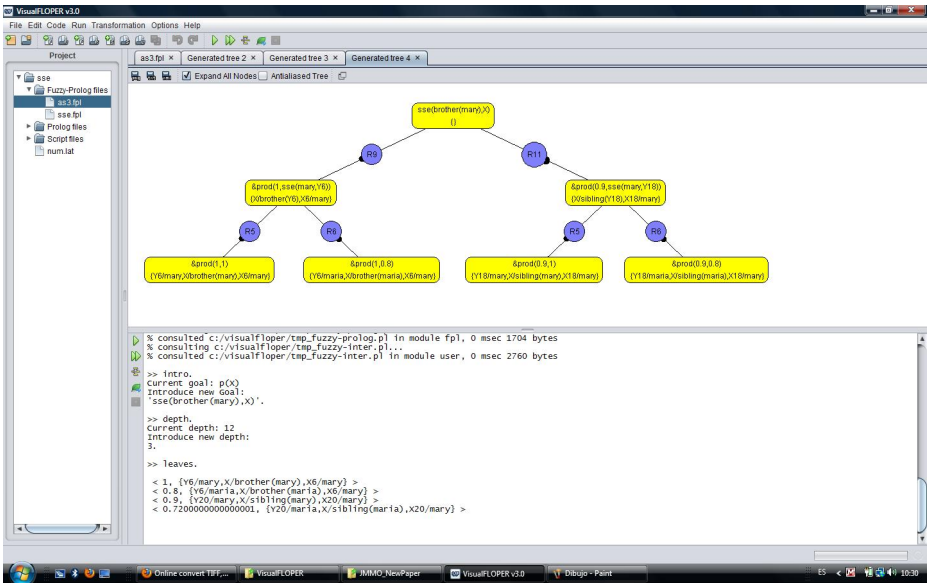

Fig. 4. Screen-shot of a work session with FLOPER

We would like to remark that even when we have worked with a very simple query with three tags in our examples, our technique works with more complex queries with larger paths and connectives in conditions. For instance, in Figure 3 we debug a query which needs to `SWAP` the wrong occurrence of *"cost"* by the similar word *"price"*. The first alternative deletes the tag *"classic"*, but our debugger achieves also more changes based on `JUMP` and `SWAP` commands.
To finish, we would like to mention that both the interpreter and the debugger of the fuzzy XPath dialect considered in this paper have been implemented with a fuzzy logic language (see [1] for more details) called FLOPER (see Figure 4 and visit `http://dectau.uclm.es/floper/`) and an on-line session is available from `http://dectau.uclm.es/fuzzyXPath/`.

Acknowledgements. Work partially supported by the EU, under FEDER, and the Spanish Science and Innovation Ministry (MICINN) under grant TIN2008-06622-C03-03, as well as by Ingenieros Alborada IDI under grant TRA2009-0309, and the JUNTA ANDALUCIA administration under grant TIC-6114.

References

1. Almendros-Jiménez, J.M., Luna Tedesqui, A., Moreno, G.: A Flexible XPath-based Query Language Implemented with Fuzzy Logic Programming. In: Pasche, A. (ed.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 186–193. Springer, Heidelberg (2011)
2. Berglund, A., Boag, S., Chamberlin, D., Fernandez, M.F., Kay, M., Robie, J., Siméon, J.: XML path language (XPath) 2.0. W3C (2007)