Summary

This thesis presents an extension of the popular XPath language which provides ranked answers to flexible queries taking profit of fuzzy variants of *and*, *or* and *avg* operators for XPath conditions, as well as two structural constraints, called *down* and *deep*, for which a certain degree of relevance is associated. In practice, this degree is very low for some answers weakly accomplishing with the original query, and hence, they should not be computed in order to alleviate the computational complexity of the information retrieval process. In order to improve the scalability of our interpreter for dealing with massive XML files, we make use of the ability of fuzzy logic programming for prematurely disregarding those computations leading to non significant solutions (i.e., with a poor degree of relevance according the preferences expressed by users when using the new command FILTER). Since our proposal has been implemented with a fuzzy logic language, we have exploited the high expressive resources of this declarative paradigm for performing "dynamic thresholding" in a very natural and efficient way. But apart from using our FLOPER environment for developing the interpreter, we also propose an implementation coded with the standard XQuery language. Basically, we have defined an XQuery library able to diffusely handle XPath expressions in such a way that our proposed FuzzyXPath can be encoded as XQuery expressions. The advantages of our approach is that any XQuery processor can handle a fuzzy version of XPath by using the library we have implemented.

On the other hand, we present a method for debugging XPath queries by describing how XPath expressions can be manipulated for obtaining a set of alternative queries matching a given XML document. For each new proposed query, we give a "chance degree" that represents an estimation on its deviation w.r.t. the initial expression. Our work is focused on providing to the programmers a repertoire of paths (containing new commands for "JUMP/DELETE/SWAP" tags) which can be used

to retrieve answers. Our debugger is able to manage big XML documents by making use of the new command FILTER which is intended to prematurely disregard those computations leading to non significant solutions (i.e., with a poor "chance degree" according to the user's preferences). The key point again is the natural capability for performing "dynamic thresholding" enjoyed by the fuzzy logic language used for implementing the tool, which somehow connects with the so-called «top-k answering problem» very well-known in the fuzzy logic and soft computing.

Regarding non standard applications, in the last block of this thesis we reinforce the bi-lateral synergies between FuzzyXPath and FLOPER. In particular, we deal with propositional fuzzy formulae containing several propositional symbols linked with connectives defined in a lattice of truth degrees more complex than *Bool*. We firstly recall a fuzzy SMT (*Satisfiability Modulo Theories*) based method for automatically proving theorems in relevant infinitely-valued (including *Łukasiewicz* and *Gödel*) logics. Next, instead of focusing on satisfiability (i.e., proving the existence of at least one model) as usually done in a SAT/SMT setting, our interest moves to the problem of finding the whole set of models (with a finite domain) for a given fuzzy formula. We re-use a previous method based on fuzzy logic programming where the formula is conceived as a goal whose derivation tree, provided by our FLOPER tool, contains on its leaves all the models of the original formula, together with other interpretations (by exhaustively interpreting each propositional symbol in all the possible forms according the whole set of values collected on the underlying lattice of truth-degrees). Next, we use the ability of the FuzzyXPath tool for exploring these derivation trees once exported in XML format, in order to automatically discover whether the formula is a tautology, satisfiable, or a contradiction.