Ernesto Damiani; Letizia Tanca; Francesca Arcelli Fontana
Fuzzy XML queries via context-based choice of aggregations

Kybernetika, Vol. 36 (2000), No. 6, [635]--655

Persistent URL: http://dml.cz/dmlcz/135378

KYBERNETIKA — VOLUME 36 (2000), NUMBER 6, PAGES 635 - 655

FUZZY XML QUERIES VIA CONTEXT-BASED CHOICE OF AGGREGATIONS

ERNESTO DAMIANI, LETIZIA TANCA AND FRANCESCA ARCELLI FONTANA

A flexible query model is presented for semi-structured information stored in *well-formed* XML documents, modeled as *XML fuzzy graphs* by computing estimates of the *importance* of the information associated to XML elements and attributes. The notion of fuzzy graph closure with threshold is then used to obtain a fuzzy extension of the XML fuzzy graphs' topological structure. Weights associated to closure arcs are computed as a conjunction of the importance values of the underlying arcs in the original graph, via a context-based choice of conjunctions. Query results are subgraphs of the resulting fuzzy closure graph, presented as a *ranked list* according to their degree of matching to the user query.

1. INTRODUCTION AND MOTIVATIONS

XML *(eXtensible Markup Language)* is a markup metalanguage designed to enable semantics-aware tagging of World Wide Web information [32]. Generally speaking, an XML document is composed of a sequence of nested elements, each delimited by a pair of start and end tags (e. g., `<tag>` and `</tag>`). XML documents can be broadly classified into two categories: *well-formed* and *valid*. An XML document is well-formed if it obeys the basic syntax of XML (e. g., non-empty tags must be properly nested, each non-empty start tag must have the corresponding end tag). Well-formed documents are also valid if they conform to a proper *Document Type Definition* (DTD). A DTD is a file (external, included directly in the XML document, or both) which contains a formal definition of a particular type of XML documents. Using a database-borrowed notion, XML DTDs are often described as *schemata*, while XML documents referencing DTDs are *instances* of those schemata. Indeed, DTDs include declarations for *elements* (i. e. tags), *attributes*, *entities*, and *notations* that will appear in XML documents. DTDs state what names can be used for element types, where they may occur, how each element relates to the others, and what attributes and sub-elements each element may have. Attribute declarations in DTDs specify the attributes of each element, indicating their name, type, and, possibly, default value. Due to the semi-structured nature of XML data, it is possible (and, indeed, frequent) two instances of the same DTD to have a different structure. In fact, some elements in the DTD can be optional and other elements can be included in an XML

document zero, one, or multiple times. The *validation* or syntax-checking procedure involves a well-formed XML document and a DTD: if the XML document is valid with respect to the DTD, validation usually produces a memory representation of the document according to a data model, such as the *Document Object Model* (DOM) standard. However, not all XML documents comply to a DTD. Currently, a large amount of XML information is being made available on the WWW in unvalidated form; therefore, there is an increasing need for Web-enabled applications to access, process and query well-formed XML documents.

```
<?xml version=''1.0" encoding=''UTF-8"?>
<car>
  <maker> Mercury </maker>
  <model serialcode = "12303B">
    <modelname> Topaz </modelname>
    <year> 1998 </year>
    <d escription>
      A comfortable family car
    </description>
  </model>
  <plant>
    <address> 13 Cherry Blossom Ave, 22030 Fairfax,VA </address>
  </plant>
</car>
```

Fig. 1. A well-formed XML document.

A sample well-formed XML document is shown in Figure 1. In this paper we elaborate on the graph-based technique for posing *blind* queries to unvalidated XML information introduced in [13]. In this setting, a query expresses the user's interests more than the expected structure of the target document. Therefore, we shall *dynamically adapt* the target document, tailoring the query answer to the user interests and intended query semantics by means of a context-based choice of aggregation operators. The remainder of the paper is organized as follows: in Section 2 we introduce some basic notation and give an outline of our approach. Section 4 deals with weighting techniques for well-formed XML documents. Section 5 describes how XML documents' structure can be *augmented* by computing the (fuzzy) closure of the containment relation between elements and attributes. Then, Section 7 provides a logical formulation for our weighting and loosening procedure. Finally, Section 8 formalizes the classification of similarity matchings and Section 9 deals with context-based choice of conjunctions.

2. NOTATION

Following [13], we shall represent well-formed XML documents and queries as *labeled graphs* $G = (V, E, L, f, g)$, whose node set V comprises both nodes representing tags

and nodes representing text/multimedia content and attributes. Arcs belonging to $E \subseteq V \times V$ may represent, according to their labeling (given as usual by a function $f : E \rightarrow L$, where $L = \{e - contains, a - contains, link, id - idref\}$ is a set of *relation labels*) XML tag and attribute inclusion, hypertext links, and `ID-IDREF` relationships. Another function $g : V \rightarrow I^*$ (where I^* is the set of strings built over a suitable alphabet I) represents the information content associated to a terminal element or attribute. The sub-graph representing (element and attribute) containment alone is in most cases a tree, where leaf nodes represent content and values, while non-leaf nodes correspond to tags. As we shall see, in this setting a query can be, without loss of generality, represented as a *graph pattern*: query execution involves finding a match of the pattern inside *document graphs* representing XML documents. Graph-based representations have been widely used in the framework of DTD-based XML query languages [5, 12] as well as for a variety of XML-related environments and tools. In order to enhance flexibility, however, we shall provide a different query execution model for blind queries, which does *not* rely on the straightforward computation of pattern matching between the document and the query graphs; rather, we introduce two preliminary steps, with the aim of narrowing the gap between query and document structures. The rationale for this approach is that even without a DTD, target XML documents can be used to estimate the intended importance of XML elements as perceived by the document designer. Moreover, query structure identifies the importance of XML elements as perceived by the user. Our approach relies on three basic steps:

1. Assignment of weights to the target document content on the basis of the document's topological structure, and tag repertoire. This step is carried out at document design time and highlights information considered important by the document designer at the granularity of XML tags and attributes. We distinguish between *structure-related* and *tag-related* weighting techniques: the former attach importance to the containment relationships that specify spatial structure of the documents (not unlike weighting in image databases [15]), while the latter express the importance of XML elements and attributes content *per se*. Combining these techniques, we shall obtain a *fuzzy labeled graph* [8], where each edge $(x, y) \in E$ has *weight* or *strength* $w(x, y) \in [0, 1]$.

2. Extension of the fuzzy labeled graph. In this step, which is also carried out at document design time, the closure of weighed documents is computed. At query execution time, the result is then tailored performing an α-cut operation on the basis of a threshold parameter provided by the user, or computed by the system on the basis of the user's profile. The output of this step is a new, tailored target graph.

3. Computation of a similarity matching between the subgraphs of the tailored document and the query graph, according to the type of matching expressing the query semantics selected by the user. Again, this step is carried out at query execution time.

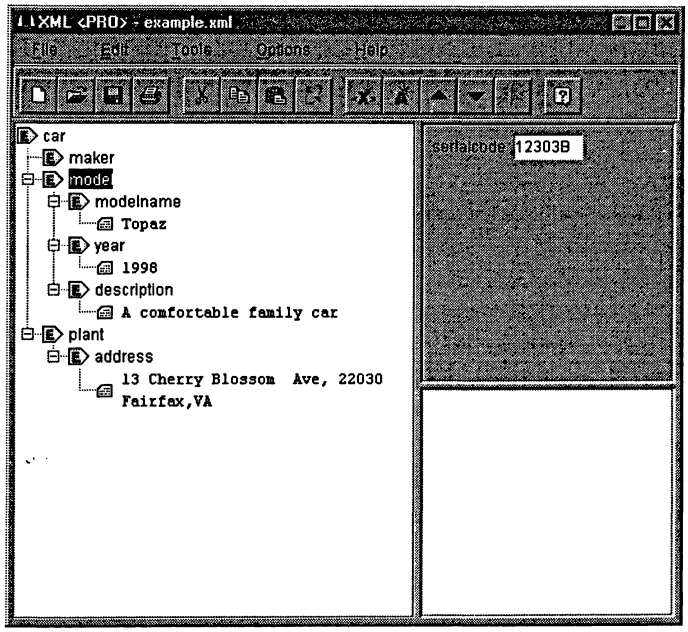

Fig. 2. A Sample XML graph.

It should be noted that our technique provides two useful tools to increase flexibility: first of all, it offers the choice between different notions of graded similarity matchings between the query graph and document subgraphs. Such matchings can be classified on a scale, going from looser to stricter correspondence in the topological structure [21]. Secondly, matchings can be carried out between fuzzy rather than crisp graphs, providing a degree of matching that will be used to rank results. We shall describe our matching procedure in some detail in Section 8. From the system designer point of view, combining information filtering with the matching techniques introduced offers a rich palette of techniques allowing for high flexibility in query execution while ensuring full control on the query semantics. Figure 2 shows the graph corresponding to the document of Figure 1. For the sake of generality, here we do not adopt the notation of a specific language or environment; rather, we reproduce the graphical interface of a popular XML editor. Since document nodes are unique, we shall assume a numerical *Object ID*, (`OID`) to be specified for each node. This causes no loss of generality, since such a `OID` can be easily computed based on the unique path reaching the element on the containment tree. Note also that, since our sample document only features containment links, the resulting graph is a tree, and labels on arcs have been omitted in Figure 2.

3. QUERIES TO XML DATA

We are now ready to define a general concept of *query* to XML information sources. To begin with, we remark that standard text retrieval techniques could be used to search for tags (such as, in our example, `<model>` and `<address>`) as well as for their desired content. Standard *Boolean* techniques for text retrieval search rely on a lexicon, i. e. a set of terms $r_1, r_2, \ldots, r_k$ and model each document as a Boolean vector of length k, whose ith entry is `true` if r_i belongs to the document. In this setting, a query is simply a Boolean expression (e. g., a conjunction) whose operands are terms or stems (possibly including *wildcards*), and its result is the set of documents where the Boolean expression evaluates to `true`. In other words, document ranking is not supported in a pure Boolean setting. A variety of fuzzy techniques have been proposed to overcome this problem [4, 27]. On the other hand, *probabilistic* text retrieval techniques model documents as *multisets* of terms, and queries as standard sets of terms, aiming at computing $P(R/Q, d)$, i. e. the probability that a document d is relevant with respect to query Q, based on the frequency distribution of terms inside d. The result is usually a ranked list of documents according to values of $P(R/Q, d)$. Variations of these techniques are currently in use for search engines dealing with HTML documents, and could of course be employed for XML data as well, though at the price of loosing all the information conveyed by the document's structure. This loss is indeed very important when the XML elements' content is made of typed values rather than of text blobs, as it is nearly always the case when XML documents are dynamically extracted from relational databases. More sophisticated approaches (e. g. algebraic ones, [6]) have been proposed for searching and structuring documents, leading to XML processing languages such as XQL [28], which also provides search capabilities. In the last few years, the database community has proposed several fully-fledged query languages for XML, some of them as a development of previous languages for querying semi-structured data; two detailed comparisons (both involving four languages) can be found in [7] and [12], while many preliminary contributions and position papers about XML querying are collected in [26]. Here, we shall not attempt to describe such languages in detail; rather, we only refer to the common features of XML-QL [14], YaTL [11], XML-GL [5] and the recent Quilt proposal [29]. Specifically, two features shared by these languages [12, 29] are relevant to our discussion:

- *User-provided patterns*, based on the assumption that the user is aware enough of the target document structure to be able to formulate a pattern that can be matched against the target XML documents for locating the desired information. Syntactically, patterns are often given in the standard form of *XPaths*. XPaths have been adopted as a W3C Recommendation and are used in several XML-related applications such as XPointer and XSLT. We regard XPaths as a special case of general *graph patterns* whose application to a target document returns a *forest* of nodes, preserving hierarchy and sequence. Flexibility support is obtained by means of wildcards [12]. This feature is also shared by the XQL document processing language [28].

- *Set-oriented query result*: all query languages retrieve portions of XML documents, namely the ones matching the user-provided pattern. Although different binding techniques are used [29], all retrieved portions equally belong to the query result set, even when the query exploits the facilities provided by the language for partial or flexible pattern matching.

With respect to the first feature, we observe that when querying well-formed XML information, the assumption that the user is aware of the target document structure is indeed debatable, because users cannot exploit a DTD or a Schema as a basis for the query graph's structure. Often, all users can rely on is a sample document, or at most a tag repertoire, i. e. the XML *vocabulary* used throughout the XML document base. In this situation, trying to find a match of the query pattern to a part of the target document is likely to result in *silence*, as the query topology, however similar, will probably not match the document's structure. Figure 3 shows a simple *blind query* composed on the basis of the vocabulary of the document in Figure 1. The user is interested in finding a car whose manufacturing plant is located in Virginia, but has no clue on the target document structure, and searching for a match of the query pattern inside the document would result in a failure. It should also be observed that no wildcard-based path expression involving the tags `car`, `maker` and `address` could result in a match, as `maker` and `address` belong to different sub-trees in the document of Figure 2. Regarding the second feature, we remark that in our opinion queries like the one in Figure 3 do not intend to dictate the exact structure of the query result; rather, they provide a loose example of the information the user is interested in. Therefore several degrees of matching should be possible. This situation has been dealt with in the field of multimedia databases [20] where query results are typically ranked lists according to some similarity measure.

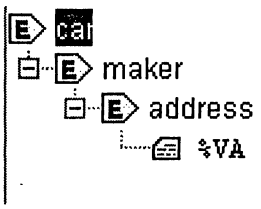

Fig. 3. A blind query.

4. WEIGHTING XML INFORMATION

Intuitively, we need to match the query graph against the document after extending the document's graph in order to by-pass links and intermediate elements which are not relevant from the user's point of view. In order to perform the extension in a sensible way, we shall first evaluate the *importance* of well-formed XML information at the granularity of XML elements. To achieve this result, we rely on *fuzzy*

weights to express the *relative importance* [2] of information at the granularity of XML elements and attributes. Low values will correspond to a negligible amount of information, while a value of 1 means that the information provided by the element (including its position in the document graph) is extremely important according to the document author. Other than that, the semantics of weights is only defined in relation to other weights in the same document/query. Again, we would like the computation of such weights to be carried out automatically, or at least to require limited manual effort.

4.1. Weighting strategies

Two main approaches can be used to compute automatically an estimate of elements' importance: *tag-related* and *structure-related* document weighting.

- **Tag-related weights** This technique labels *the nodes* of an XML graph with their relative importance, by means of a function $w_{\text{node}} : V \to [0, 1]$. This function associates in a natural way a fuzzy value to the node's label, though the resulting graph is not a fuzzy graph in the classical sense [24]. Tag-related weights can be obtained by polling the document designer; alternatively, they can be computed using a linear combination of two notions of frequency, each having a different *scope*. The first notion associates tag importance with the frequency of XSL-like *XPath expressions* [31] ending with that tag, throughout the document base. A path expression uniquely identifies a tag in a given position inside an XML document; for instance, `car:plant:address[Cherry Ave]` uniquely identifies the `<address>` tag in Figure 1. A path's frequency w.r.t. a sample set of XML documents extracted from a document base measures how many times a given tag is found in a certain position, as an estimate of the probability of finding it there in a randomly-selected document. Intuition suggests that this frequency relates inversely to importance. The second notion of frequency is the usual *term frequency* used in Information Retrieval Systems; it is document-centered, as it takes into account the frequency profile of the set of terms composing the elements' content in the target document. In this case, one can assume that elements containing high frequency terms convey less information than those containing low frequency ones, and their importance is comparatively low.

- **Structure-related weights** The structure-related technique weighs *the arcs* of an XML graph using topological parameters related to the position of XML elements and attributes. This is obtained by computing a function $w_{\text{arc}} : E \to [0, 1]$ estimating the importance of the arc. This function associates importance degrees (fuzzy values) to the arcs, and its application gives a fuzzy graph in a natural way. Topological parameters to be considered include *nesting*, i. e. the length of the path to the terminal element of the arc from the document root node, and *fan-out*, i. e. the number of elements/attributes directly contained in the terminal element of the arc under consideration. Structure-related weighting can readily be applied to both documents and queries; however, a basic distinction should be drawn. When weighting a query, topological

Table 1. Sample tag-related weights for the document in Figure 1.

tag	weight
maker	0.5
model	0.8
serialcode	0.8
plant	0.8

parameters estimate the importance of a generic XML element as perceived by the user; for instance, an element being closer than another element to the query tree root suggests that the user considers that type of information to be more important. When weighting a document, weights are estimates of the importance of individual tags, as perceived by the document designer. Though in this paper we shall deal with document weighting only, our techniques are readily extendable to take into account weighted queries as well.

Note that in both the above models, weights are constant and do not depend upon the content/value of the XML element or attribute involved; this is indeed a drawback which limits the semantics of weights. For instance, an XML element such as <PRICE> could be considered important only when its content lies inside a given range of values (for a detailed discussion on this subject see [19]). In principle [18], this problem could be solved introducing an additional dependency between weights and the content/value of the corresponding element/attribute. Checking this additional dependency is however bound to be computationally very expensive; therefore in the remainder of the paper we shall only deal with content-independent weights.

4.2. Weight computation

We are now ready to outline the actual computation of the fuzzy weights. We rely on a semi-automatic approach, using tag-related weights provided by the document designer for a small number of key tags, and then propagating them to the document graph.

In the tag-related model, the weight w_{node} of each node in the XML document tree expresses either the *relative importance* of a tag as estimated by the document designer, or the *probability* (estimated via the corresponding sample frequency) associated to the (unique) path expression ending in that node.

Using the structure-related technique, weighting is performed as follows:

- Use a function $w_{\text{arc}} : E \rightarrow [0,1]$ to weigh each arc $(n_i, n_j) \in V$ of the target document graph G.

- A simple function is, for instance, the *normalized distance* from root, defined

as follows:

$$w_{\text{arc}}(n_i, n_j) = \frac{d_{\max} - l}{d_{\max}} \tag{1}$$

where $d_{\max}$ is the length of the longest path starting from a root node and l is the distance from n_j to the root. This function establishes a simple inverse relation between arcs importance and their nesting level.

- Another suitable function associates to each arc (n_i, n_j) in G the *normalized cardinality* of the sub-tree G' (obtained taking inclusion arcs only into account) whose root is n_j, namely

$$w_{\text{arc}}(n_i, n_j) = \frac{\mid G'(n_j) \mid}{\mid G \mid} \tag{2}$$

Note that in this case $w(n_i, n_j)$ does not depend on n_i. Also, arcs from element to attribute nodes enjoy no special status and are weighted as the others. This weighting function is non-monotonic w.r.t distance from root and estimates each arc's importance via the size of the subtree rooted in its final node.

Applying the above weighting procedure to the well-formed document in Figure 1, and using the function of Eq. (1), we obtain Table 2.

Table 2. Structure-related weights for the sample document in Figure 1.

n_i	n_j	w
car	maker	2/3
car	model	2/3
model	serialcode	1/3
model	modelname	1/3
model	year	1/3
model	description	1/3
car	plant	2/3
plant	address	1/3

The two weighting techniques can be then combined as follows:

$$w'_{\text{arc}}(n_i, n_r) = T(w_{\text{node}}(n_r), w_{\text{arc}}(n_i, n_r)) \tag{3}$$

where T is a standard triangular norm or t-norm [25], i. e. a binary function $[0,1]^2 \to [0,1]$ satisfying the following properties:

$$\textit{Monotonicity}: x_1 \leq x'_1 \wedge x_2 \leq x'_2 \to T(x_1, x_2) \leq T(x'_1, x'_2) \tag{4}$$

$$\textit{Commutativity}: T(x_1, x_2) = T(x_2, x_1) \tag{5}$$

Table 3. t-norms.

1	t-norms
2	$\min(x, y)$
3	xy
4	$\max(x + y - 1, 0)$
5	$\frac{xy}{x+y-xy}$
6	$x(y = 1), y(x = 1)$, 0 o/w

Table 4. The final weights.

n_i	n_j	w
car	maker	0.58
car	model	0.73
model	serialcode	0.52
model	modelname	0.33
model	year	0.33
model	description	0.33
car	plant	0.73
plant	address	0.33

$$\textit{Associativity}: T(T(x_1, x_2), x_3) = T(x_1, T(x_2, x_3)). \tag{6}$$

T-operators also enjoy $\wedge$-Conservation, namely $T(0,0) = 0; T(x,1) = T(1,x) = x$. Several well-known triangular norms are shown in Table 3; for more t-norms, see [22].

Note that, when the norm $T = \min$, $w_{\text{node}}(n_r) = 0$ means that the arc (n_i, n_r) must be ignored in the computation whatever the value of $w_{\text{arc}}(n_i, n_r)$ for all nodes n_i connected to it. Of course, other aggregation operators that are not t-norms can be used as well; for instance, the final weighting of the document in Figure 1 when T is the arithmetic average is reported in Table 4.

5. FUZZY CLOSURE COMPUTATION

Once the weighting is completed, the *fuzzy closure* C of the fuzzy labeled graph is computed. Intuitively, computing graph-theoretical closure entails inserting a new arc between two nodes if they are connected via a path of any length in the original graph. Computing the closure is well-known to be polynomial w.r.t. the number of nodes of the graph. In our model, the weight of each closure arc in $C - G$ is computed aggregating via a t-norm T the weights of the arcs belonging to the path it corresponds to in the original graph. Namely, for each arc (n_i, n_j) in the closure graph C we write:

$$w_{\text{arc}}(n_i, n_j) = T(w_{\text{arc}}(n_i, n_r), w_{\text{arc}}(n_r, n_s), \ldots, w_{\text{arc}}(n_t, n_j)) \tag{7}$$

where $\{(n_i, n_r)(n_r, n_s), \ldots, (n_t, n_j)\}$ is the set of arcs comprising the shortest paths from n_i to n_j in G and, again, T is a standard t-norm [25]. Intuitively, the closure computation step gives an extended structure to the document, providing a looser view of the containment and reachability relations. Selecting the type of t-norm to be used for combining weights means deciding if and how a low weight on an intermediate element should affect the importance of a nested high-weight element. This can be a very difficult problem, as the right choice may depend on the dataset or even on the single data instance at hand. There are some cases in which the t-norm of the minimum best fits the context, other cases in which it is more reasonable to use the product or the Lukasiewicz t-norm. Often, it is convenient to use a family of t-norms indexed by a tunable parameter. In general, however, it is guessing the right context, or better the knowledge associated to it from some background of preliminary knowledge, that leads to the right t-norm for a given application. For instance, suppose a node n_j is connected to the root via a single path of length 2, namely $(n_{\text{root}}, n_i)(n_i, n_j)$. If $w_{\text{arc}}(n_{\text{root}}, n_i) << w_{\text{arc}}(n_i, n_j)$ the weight of the closure arc (n_{root}, n_j) will depend on how the t-norm T combines the two weights. In other words, how much should the high weight of (n_i, n_j) be depreciated by the fact that the arc is preceded by (comparatively) low-weight one (n_{root}, n_i)? It is easy to see that we have a conservative choice, namely $T = \min$. However, this conservative choice does not always agree with humans' intuition, because the min operator gives a value that depends only on one of the operands without considering the other [17] (for instance, we have the *absorption property*: $T(x, 0) = 0$). Moreover, it does not provide the *strict-monotonicity* property $(\forall y, x' > x \rightarrow T(x', y) > T(x, y))$. In other words, an increase in one of the operands does not ensure the result to increase if the other operand does not increase as well. To understand the effect of the min's *single operand dependency* in our case, consider the two arc pairs shown below:

1. (<car><model>0.2)(<model><serialcode>0.9)
2. (<car><model>0.3)(<model><serialcode>0.4)

when the min operation is used for conjunction, arc pair (2) is ranked above arc (1), while most people would probably decide that arc pair (1), whose second element has much higher importance, should be ranked first. The other t-operators have the following common properties [25]:

$$x = 1 \vee x = 0 \vee y = 1 \vee y = 0 \rightarrow T(x, y) = x \vee T(x, y) = y \tag{8}$$

$$T(x, y) \leq \min(x, y). \tag{9}$$

Property (9) warns us that, while the other t-norms somewhat alleviate the single operand dependency problem of the min for arc pairs (using the product, for instance, the outcome of the previous example would be reversed), they may introduce other problems for longer paths. Let's consider the following example, where we add a `modelnamecode` attribute to the `<modelname>` element:

1. (<car><model>0.1)(<model><modelname>0.9)(<modelname><modelnamecode>0.1)

2. (<car><model>0.2)(<model><modelname>0.5)(<modelname><modelnamecode>0.2)

In this case using the product we get $T(x, y, z) = T(x, T(y, z)) = 0.009$ for the first path, while the second gets 0.02; again this estimate of importance that ranks path (2) above path (1) may not fully agree with users' intuition. The graph corresponding to our sample document, computed using the arithmetic mean as an aggregation operator is depicted in Figure 4. For the sake of clarity, only internal element nodes are shown.

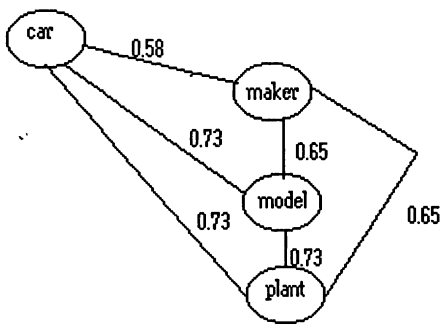

Fig. 4. The closure of the XML graph.

6. QUERY EXECUTION

We are now ready to outline our query execution technique for well-formed XML documents, which relies on the following procedure:

1. Weight the target document graph G and the query graph Q according to structure-related or tag-related techniques described in Section 4. Weights on target documents can be computed once for all (in most cases, at the cost of a visit to the document tree). Though weighting the queries must be done on-line, their limited cardinality is likely to keep the computational load negligible in most cases.

2. Compute the closure graph C of G using a T-norm or a suitable fuzzy aggregation of the weights. This operation is dominated by matrix multiplication, and its complexity lies in between $O(n^2)$ and $O(n^3)$ where n is the cardinality of the node-set V of the target document graph. Again, graph closure can be pre-computed once for all and cached for future requests.

3. Perform a *cut* operation on C using a threshold (this operation gives a new, tailored target graph TG). The cut operation simply deletes the closure arcs whose weight is below a user-provided threshold α, and is linear in the cardinality of the edge-set of $C - G$.

4. Compute a fuzzy similarity matching between the subgraphs TG of the tailored document and the query graph Q, according to selected type of matching. This operation coincides with the usual query execution procedure of pattern-based query languages, and its complexity can be exponential or polynomial w.r.t the cardinality of the node-set V of the target document graph [10], depending on the allowed topology for queries and documents [9].

The first steps of the above procedure are reasonably fast (as document weights and closure can be pre-computed, required on-line operation consists in a sequence of one-step lookups) and does not depend on the formal definition of weights. The last step coincides with standard pattern matching in the query execution of XML query languages [5], and its complexity clearly dominates the other steps.

7. A LOGICAL FORMULATION

All graph-theoretical notions given in previous subsections can be readily translated in a simple logical formulation to obtain an extensional fuzzy database. First of all, we express the document graph as a conjunction of *ground facts*, e. g. instances of 1-ary and binary predicates *contains*, *value* and *content* with constant values. Typed predicates like *e-contains* and *a-contains* will be used to distinguish between element and attribute containment. For example, for the document in Figure 1 we have the following conjunction of facts:

e-contains(OID1-car,OID2-maker)∧ content(0ID2-maker,"Mercury")
∧ e-contains(OID1-car,OID3-model) ∧
a-contains(OID3-model,OID4-serialcode)∧ value(OID4-serialcode, "1230B")
...
e-contains(OID8-plant,OID9-address)∧content(OID9-address, "Cherry Blossom Ave")

Then, we use the weighting procedure of Section 4 to establish importance, to be used as *truth-value* for the facts in the extensional database. For instance, using the fuzzy weighting model of Section 4, we have *e-contains(car, maker)* = 0.58. Now we are ready to perform a *closure procedure* to augment the facts, according to the following *transitivity* rule:

$$e\text{-}contains(x,y) \Rightarrow le\text{-}contains(x,y) \tag{10}$$

$$le\text{-}contains(x,y) \wedge e\text{-}contains(y,z) \Rightarrow le\text{-}contains(x,z) \tag{11}$$

Formula (11) gives the truth-value of the new predicate *le-contains(x,z)* in terms of the truth-values of predicates *le-contains(x,y)* and *e-contains(y,z)*. Indeed, graph-based queries are inherently compound, raising the issue of finding the appropriate aggregation operator for combining the elementary truth-values; this is exactly the same problem of the choice of the t-norm to aggregate weights discussed in Section 5. Selecting a conjunction means deciding if and how a lightweight intermediate element should affect the importance of a heavier element nested inside it. As we have

seen, the straightforward approach to this problem is to use triangular norms, but the aggregation provided by t-norms may not coincide with users' intuition. However, our logical formulation allows us to see more clearly the association between conjunction and query execution semantics. Indeed, the conjunction to be employed can be a *logical*, *compensatory* or *product-based* one, depending on the user-selected semantics that was used to compute truth values of the initial predicates. In the following, we shall briefly discuss how the choice of a conjunction may affect query execution in our setting.

- **Logical conjunctions** are modeled by t-norms and express a conservative view in which the total degree of importance of a XML fragment is linked to the importance of its least important element. The most natural choice for conjunction; pure min, is the largest associative aggregation operator which extends ordinary conjunction. It is also the only idempotent one and, thanks to these properties, it well preserves query optimization properties [20]. Once again, we note that using the min conjunction, we adopt the most conservative attitude: for instance, in our example we get *e-contains(department, group)* with a truth value of $3/5$. Unfortunately, as shown in Section 5, its behavior does not always coincide with users' intuition. An intermediate behavior is obtained by using *Lukasiewicz* norm $T = \max(a + b - 1, 0)$. Product-based conjunctions introduce a *probabilistic* view which also may create problems with user intuition (Section 5). They also pose other problems, as they are unfit for query optimization.

- **Weighted averages (WA)** promote a more *utilitaristic* view where the higher value of importance of an element can often *compensate* for a lower value of another one. In other words, it may happen that $WA(x, y) \geq \min(x, y)$. Table 5 shows some classical average-based choices for the aggregation operation. The degree of compensation for these operators depends on a tunable parameter $\gamma \in [0, 1]$. We shall require this positive compensation to occur for all values of γ; therefore we rule out operator A_1, which coincides with the min when $\gamma = 0$, and operator A_3, which coincides with the product. On the other hand, operator A_2 from Table 5 presents the single operand dependency problem, as it exhibits the absorption property (it always gives 0 when one of the operands is 0). Operator A_4 aggregates a conservative view with a utilitaristic one. When $\gamma = 0$, it coincides with simple arithmetic mean (operator A_5), which has been shown in previous examples and will be used in the sequel.

The logical counterpart of the *α-cut* operation we performed on the weighed closure graph is thresholding truth values. Thresholding involves all predicates in the extensional database; intuitively, it will eliminate predicates having a low truth-value, providing a set of facts tailored to the user interests. Then, we can express the query as a logical formula, e. g.

$$Q = \textit{e-contains(x-plant, x-address)} \wedge \textit{value(x-address, "Cherry Blossom Ave")}$$

Note that *x-plant* and *x-address* are typed logical variables, where types are element or attribute names such as `car`, `maker`, `model`, `plant`. We shall write $x - t$

to denote variable x belonging to type t. Matching the query formula to the transformed facts means to compute its truth-value, which is obtained taking the conjunction of the truth-values of the atomic predicates. Consistently with our previous choice, the conjunction to be used is the same one that was used to compute the closure. Namely, we compute $\mu(Q) = \mu(e\text{-}contains(x\text{-}plant, x\text{-}address)) \wedge \mu(content(x\text{-}address, \text{"Cherry Blossom Ave"}))$

Table 5. Average-based fuzzy conjunctions.

norm	$T(x,y)$
A_1	$\gamma \max(x,y) + (1-\gamma)) \min(x,y)$
A_2	$(x+y-xy)^{\gamma}(xy)^{1-\gamma}$
A_3	$\gamma(x+y-xy) + (1-\gamma)(xy)$
A_4	$\frac{\gamma \min(x,y)+(1-\gamma)(x+y)}{2}$
A_5	$\frac{(x+y)}{2}$

Once again we remark that the choice of the aggregation will affect query result; for instance, in the compensatory vision, there is no absorption property and $\mu(Q)$ may well be above zero even if either $\mu(e\text{-}contains(x\text{-}plant, x\text{-}address))$ or $\mu(content(x\text{-}address, \text{"Cherry Blossom Ave"})$ are zero (but not both). More importantly, whatever the conjunction we use, the query result is a *ranked list* of couples *(x-plant, x-address)*, ordered according to their truth values.

8. FUZZY GRAPH MATCHINGS

In this section we shall outline the fuzzy matching algorithm used for locating the fuzzy subgraphs of the extended document graph and computing their degree of matching with respect to the user query. To allow for maximum flexibility, several notions of matching can be employed. Here, we only outline their classification:

- *Lexical Distance* Matching between document subgraphs and the query graph depends on the number of nodes they have in common, regardless of their position in the graphs. Different distance measures can be defined taking into account the fact that nodes may belong to different XML *lexical categories*, e. g. elements in the query graph may correspond to attributes in the document and vice versa.

- *Graph Simulation* Matching between document subgraphs and the query graph depends on the number of paths spanning the same nodes they have in common. Again, different distance measures can be defined taking into account the fact that nodes represent different types of XML lexical terms.

- *Graph Embedding* Matching between document subgraphs and the query graph is defined as a function φ associating query nodes to document nodes in such a way that edges and labels are preserved.
- *Graph Isomorphism* Matching between document subgraphs is a function φ as above, which in this case is required to be a one-to-one mapping.

The procedure consists of three steps:

1. Given the query $Q = (V, E, f)$, without taking membership values into account, locate a matching subgraph $G' = (V', E', f')$ in the extended document graph (using, for instance, crisp depth first search), such that that there is a mapping $\varphi : V \rightarrow V'$ preserving arcs and arc labels.[1] A procedure `FindMatch` is used, according to the desired type of matching; in the case of graph embedding, its complexity is polynomial in $\mid V \mid$ for simple queries [10].
2. Compute the *ranking function* $J(Q, G')$ as follows:

$$J = \wedge_{(n_i,n_j)\in E} w_{\text{arc}}(\varphi(n_i), \varphi(n_j)) = T(w_{\text{arc}}(\varphi(n_i), \varphi(n_j), \dots). \tag{12}$$

   In the second part of Eq. (12), we straightforwardly use t-norm associativity to compute the conjunction T over all edges $\varphi(n_i), \varphi(n_j)$ in the document graph corresponding to edges n_i, n_j in the query graph. This is the same procedure that was used for computing the truth-value of the sample query in Section 7. When T is the arithmetic average, we cannot rely on associativity and we get

$$\frac{1}{\mid E \mid} \sum_{(n_i,n_j)\in E} w_{\text{arc}}(\varphi(n_i), \varphi(n_j)). \tag{13}$$

   Function (12) plays the same role as the objective function in standard fuzzy graph matching algorithms [21], expressing the degree of membership of a candidate subgraph in the result set as a conjunction of the weights on corresponding arcs.
3. Output the matching subgraph and its rank J.

As a very simple example, let's now execute the blind query of Figure 3 on the graph of Figure 4 (after applying the α-cut with $\alpha = 0.45$). The tailored graph contains the requested path `<car><maker><address>`, and since the cardinality $\mid E \mid$ of the edge set of the match is 2, the rank function value for this match (when T is the average) is given by $J = \frac{1}{2}T(w_{\text{arc}}(car, maker) = 0.58, w_{\text{arc}}(maker, address) = 0.49) = 0.53$. Note that $w_{\text{arc}}(car, maker)$ was simply looked up from Table 4), while $w_{\text{arc}}(maker, address)$ was computed in the closure step as

$$T(w_{\text{arc}}(maker, plant), w_{\text{arc}}(plant, address)).$$

[1] Moreover, this matching ensures that if values are specified on terminal nodes in the query graph, they also must appear as content labels of the corresponding nodes in the input document graph

9. CONTEXT BASED CHOICE OF T-NORMS

We have presented what we consider to be a novel fuzzy technique to execute blind XML queries, suitable for integration in current XML query languages. However, as we have seen, the query engine's behavior depends on the choice of a t-norm or an aggregation operator, and the discussion of the previous Sections has shown how difficulty may be to decide a priori which aggregation will match the user's intuition. Indeed, the choice of t-norms is a moot point: in recent years a variety of fuzzy operators have been developed, potentially providing high flexibility but sometimes making it difficult to choose the one best suited to a particular application. In this section, we deal with *similarity* as a way to approximate knowledge in a given context [3], leading the choice of a suitable t-norm. Specifically, we describe the application of a similarity-based method to the choice of a conjunction in relation to its effects on query execution semantics.

Fuzzy conjunctions are deeply depending on the context. For example, the assertion `IF` $City_3$ `is close to` $City_2$ `AND` $City_3$ `is close to` $City_1$ `THEN` $City_1$ `is not so close to` $City_3$ contains an important information about the context. Implicitly, the statement pictures a context where the three cities lie on a straight line on a map. Intuition suggests that in such a context, the conjunction of the two predicates expressing closeness must be somewhat less true than each of the conjuncts. We intend to sketch a formal notion of contextual knowledge [1], derived from some sample features of a certain environment (i.e. some samples of distances in spatial structures of the XML document). Then, we shall discuss how such knowledge can be applied to the choice of a conjunction.

9.1. Contextual knowledge

The main aspects that should be fulfilled by a model aiming at representing a context are related to the possibility to predict some actions, or more generally to derive new information and to be *scalable*, i.e. to contain elements that can assume different scales of values. These specifications seem to be grasped by the mathematical notion of similarity, since similarity can be determined by a small set of initial values and then extended to whole values of the set in which it is defined and moreover similarity, depending on a t-norm, can be easily parametrized.

Given a set S and a t-norm T, a *T-similarity*, or simply a *similarity* on S, is a map $\mathcal{R} : S \times S \to [0,1]$ such that:

$$\mathcal{R}(x,x) = 1, \mathcal{R}(x,y) = \mathcal{R}(y,x) \tag{14}$$

$$\mathcal{R}(x,y) \geq T(\mathcal{R}(x,z), \mathcal{R}(z,y)), \forall x,y,z \in S. \tag{15}$$

An *extended pseudometric* on S is a mapping $d : S \times S \to [0,\infty]$ such that, for any $x,y,z \in S$, $d(x,x) = 0$, $d(x,y) = d(y,x)$, and $d(x,z) \leq d(x,y) + d(y,z)$. Given a t-norm T and $x \in [0,1]$, we denote by $T^n(x))$ the number $T(x, x \ldots, x)$ n-times. Finally, we recall that a t-norm T is called *Archimedean* provided that, for any $x \in [0,1]$ there exists an integer n such that $T^n(x) < x$ [25].

We now consider a noteworthy result on similarity, established by Valverde in [30], which outlines a kind of duality between similarities and metric notions. In particular, Valverde has pointed out that, given a t-norm T and a pseudometric, a class of similarities with respect to T is uniquely defined. Moreover, given a similarity with respect to a t-norm T, a class of pseudometrics is wholly determined.

Let $d : S \times S \to [0, +\infty]$ be an extended pseudometrics, and let T be a continuous and Archimedean t-norm. Then, there exists a continuous and strictly decreasing map $f : [0,1] \to [0, +\infty]$ such that $f(1) = 0$ and

$$\mathcal{R}_d(x,y) = f^{-1}(d(x,y)) \tag{16}$$

is a T-similarity on S.

Moreover, for any x, y in $[0,1]$, $x * y = f^{[-1]}(f(x) + f(y))$, where $f^{[-1]}$ is the pseudoinverse of f.

Let $\mathcal{R}$ be a similarity with respect to a continuous and Archimedean t-norm T. Then there exists a mapping $f : [0,1] \to [0, +\infty]$ continuous and strictly decreasing such that $f(1) = 0$ and

$$d_{\mathcal{R}}(x,y) = f(\mathcal{R}(x,y)) \tag{17}$$

is an extended pseudometrics.

This results provide a mathematical foundation to the discussion in Section 7: since in our application the notion of distance depends on the context (i. e., the concepts of "closeness" between elements and attributes may well change according to the dataset), distance is dual to the notion of similarity and similarity depends on t-norms, we need to employ different t-norms in order to model several context-based notions of distance.

To model a context by similarity, given a set S representing the whole context (in our case, an XML document base), and a finite set $N = \{P_1, P_2, \ldots, P_n\}$ of elements of S, we consider the case where a similarity $\mathcal{R}$ is described by "examples" i. e. the values $\mathcal{R}(x,y)$ are defined for any $x, y \in N$. Our goal is to determine the t-norm T that best fits the behavior of $\mathcal{R}$ as a T-similarity. More precisely, we seek a function $T : [0,1] \times [0,1] \to [0,1]$ such that T is a continuous t-norm and, for any $x, y, z \in S$, $\mathcal{R}(x,y) \geq T(\mathcal{R}(x,z), \mathcal{R}(z,y))$.

We observe that the set of t-norms is a partially ordered set with respect to the relation $\preceq$, defined by setting $T \preceq T'$ whenever $T(x,y) \leq T'(x,y)$ for any $x, y \in S$. If T and T' are t-norms, then the function $T \wedge T'$ defined by setting $T \wedge T'(x,y) = \min(T(x,y), T'(x,y))$ is a t-norm, too. Moreover, if $\mathcal{R}$ is T-transitive and T' is a t-norm such that $T' \preceq T$, then $\mathcal{R}$ is a T'-transitive, too. As a consequence, given a fuzzy relation $\mathcal{R}$, we can consider the t-norm $T_{\mathcal{R}}$ defined by setting, for any $x, y \in S$:

$$T_{\mathcal{R}}(x,y) = Inf\{T | \mathcal{R}(x,y) \geq T(\mathcal{R}(x,z), \mathcal{R}(z,y))\}. \tag{18}$$

Observe that, by denoting the t-norm of the minimum as T_M, if $\mathcal{R}$ is not transitive with respect to T_M then $T_{\mathcal{R}}$ cannot be defined.

For example, consider the case where $S = \{a, b, c\}$ and $\mathcal{R}$ is defined by setting $\mathcal{R}(a,b) = 0.2$, $\mathcal{R}(b,c) = 0.2$ and $\mathcal{R}(a,c) = 0.04$. In this case it is not difficult to show

that $T_{\mathcal{R}}$ coincides with the usual product in $[0,1]$. However, in general it is not easy to determine the minimum t-norm such that a certain fuzzy relation is transitive.

So, either we give a comprehensive set of t-norms and try to pick out the one that best fits a given fuzzy relation, or we give a parametric t-norm and try to fit the fuzzy relation with the choice of a suitable parameter. In this case, one could rely on a set of some basic t-norms such as the minimum, the usual product and the Lukasiewicz t-norm. Such t-norms can be parametrized to constitute a bundle of contexts. Parametrization of average-based norms was discussed in Section 7; the above discussion shows that this technique has a sound basis and can be applied to any parameter-based t-norm. For example, we can consider $T_\lambda = (x \cdot y)^\lambda$, where $\cdot$ is the usual product in $[0,1]$ and $\lambda \in [0,1]$. According to Eq. (16) and Eq. (17), the function f that generates T_λ is, up to a constant, $f(y) = -\lambda \log y$. This suggests a method to derive a fine tuned context where the t-norm is represented by the product. For example, if $S = \{a, b, c\}$ and $\mathcal{R}(a,b) = 0.1, \mathcal{R}(a,c) = 0.2, \mathcal{R}(b,c) = 0.3$, then we can set $\lambda = \min_{\alpha \in R}\{(0.2 \cdot 0.3)^\alpha \geq 0.1\} = \frac{-1}{-2+Log6}$.

10. CONCLUSION

While we are well aware that the approach to XML querying described in this paper needs further development and experimentation, we believe some important notions were established, while others have been highlighted for future research. As XML is the language of choice for Web-based knowledge representation applications, it is particularly interesting to observe that once we have reconstructed a t-norm suitable for a given document base, a dataset-dependent definition of similarity easily follows through. In fact, given $\mathcal{R}$ and a continuous t-norm T, the T-similarity $\overline{\mathcal{R}}$ generated by $\mathcal{R}\}$is given by $n \in N \bigvee \mathcal{R}^\backslash$, where $\mathcal{R}^\backslash$ is the usual power of a fuzzy relation with respect to T. Therefore, given a set S of documents and a subset N of elements of S, we can define a contextual knowledge on the set S by first defining a fuzzy relation $\mathcal{R}$ over N and then establishing which of the three main kind of t-norms best fits the values of $\mathcal{R}$ over N. Then, we can use a parametrized t-norm in order to fine tune the context to the chosen environment. Once we get a suitable parameter and hence a suitable t-norm T, we can generate the similarity $\overline{\mathcal{R}}$, representing the contextual knowledge we were looking for, by applying some classical methods. We intend to explore this subject in a future paper.

(Received June 13, 2000.)

REFERENCES

[1] F. Arcelli Fontana and F. Formato: User adaptive models based on similarity. In: Proc. ACM Symp. on Applied Computing (SAC 2000), Como.

[2] P. Bosc: On the primitivity of the division of fuzzy relations. Soft Computing *2* (1998), 2.

[3] P. Brezillon, C. Gentile, I. Saker, and M. Secron: SART: A system for supporting operators with contextual knowledge. In: Proc. Internat. Conference on Modelling and Using Context (Context 97), Rio de Janeiro.

[4] D. A. Buell: A general model of query processing in information retrieval systems. Inform. Process. Management *17* (1981), 5. Soft Computing *2* (1998), 2.
[5] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca: XML-GL: A graphical language for querying and restructuring XML documents. Computer Networks *31* (1999), 2.
[6] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski: An algebra for structured text search and a framework for its implementation. The Computer Journal *38* (1995), 1.
[7] S. Ceri and A. Bonifati: Comparison of XML query languages. SIGMOD Record *29* (2000), 1.
[8] K. P. Chan, Y. S. Cheung: Fuzzy attribute graph with applications to character recognition. IEEE Trans. Systems Man Cybernet. *22* (1992), 1.
[9] R. Cohen, G. Di Battista, A. Kanevsky, and R. Tamassia: Reinventing the wheel: An optimal data structure for connectivity queries. In: Proc. ACM-TOC Symp. on the Theory of Computing, S. Diego 1993.
[10] S. Comai, E. Damiani, R. Posenato, and L. Tanca: A schema-based approach to modeling and querying WWW data. In: Proceedings of Flexible Query Answering Systems (FQAS'98) (H. Cristiansen, ed., Lecture Notes in Artificial Intelligence 1495), Springer, Roskilde 1998.
[11] S. Cluet, C. Delobel, J. Simeon, and K. Smaga: Your mediators need data conversion. In: Proc. ACM-SIGMOD Intl. Conf. on Management of Data, Seattle 1998.
[12] S. Cluet, A. Deutsch, D. Florescu, A. Levy, D. Maier, J. McHugh, J. Robie, D. Suciu, and J. Widom: XML Query Languages: Experiences and Exemplars. http://www-db.research.bell-labs.com/user/simeon/xquery.html
[13] E. Damiani and L. Tanca: Blind queries to XML data. In: Proc. 11th Database and Expert Systems Applications Conference (DEXA 2000) (M. Ibrahim, J. Kung, N. Revell, and eds., Lecture Notes in Computer Science 1873), Springer, London 2000.
[14] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu: A query language for XML. Computer Networks *31* (1999), 2.
[15] A. Del Bimbo and E. Vicario: Using weighted spatial relatioship in retrieval by visual content. In: Proc. IEEE Workshop on Content Based Access of Images, Santa Barbara 1998.
[16] D. Dubois, R. Martin Clouaire, and H. Prade: Practical computing in fuzzy logic. In: Fuzzy Computing (M. M. Gupta and T. Yamakawa, eds.), North Holland, Amsterdam 1988.
[17] D. Dubois, H. Fargier, and H. Prade: Refinements of the maximum approach to decision making in fuzzy environments. Fuzzy Sets and Systems *81* (1996), 3.
[18] D. Dubois, F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, and H. Prade: Fuzzy set modelling in case-based reasoning. Internat. J. Intelligent Systems *13* (1998), 1.
[19] D. Dubois, H. Prade, and F. Sedes: Fuzzy logic techniques in multimedia database querying: A preliminary investigations of the potentials. In: Database Semantics: Semantic Issues in Multimedia Systems (R. Meersman, Z. Tari, and S. Stevens, eds.), Kluwer, Dordrecht 1999.
[20] R. Fagin: Combining fuzzy information from multiple systems. In: Proc. Fifteenth ACM Symposium on Principles of Database Systems, Montreal 1996.
[21] S. Gold and A. Rangarajan: A graduated assignment algorithm for graph matching. IEEE Trans. on Pattern Analysis and Machine Intelligence *18* (1996), 2.
[22] M. M. Gupta and J. Oi: Theory of T-norms and fuzzy inference methods. Fuzzy Sets and Systems *40* (1991), 3.
[23] Microsoft Corporation White Paper: XML-Data Specification. `msdn.microsoft.com/xml/articles/xmldata.html`.

[24] J. Mordeson and P. Nair: Fuzzy Graphs and Hypergraphs. Studies in Fuzziness and Soft Computing. Physica Verlag, Heidelberg 2000.
[25] J. Klir and T. Folger: Fuzzy Sets, Uncertainty and Information. Prentice–Hall, Englewood Cliffs, N.J. 1988.
[26] The Query Language Workshop. `www.w3.org/xml/xql98`.
[27] T. Radecki: A fuzzy set theoretical approach to document retrieval. Information Processing and & Management *15* (1979), 5.
[28] J. Robie: The Design of XQL. `www.texcel.no/whitepapers/xql-design.html`.
[29] J. Robie, D. Chamberlin, and D. Florescu: Quilt: An XML Query Language. `http://www.almaden.ibm.com/cs/people/chamberlin/usecases.html`.
[30] L. Valverde: On the structure of F-indistinguishability operators. Fuzzy Sets and Systems *17* (1995), 3.
[31] World Wide Web Council. XSL Transformations, Version 1.0. W3C Recommendation, `www.w3.org/TR/1999/REC-xslt-19991116`.
[32] World Wide Web Council. Extensible Markup Language, Version 1.0. W3C Recommendation, `www.w3.org/TR/1998/REC-xml-19980210`.

Prof. Dr. Ernesto Damiani, Polo di Crema – Università di Milano, Via Bramante 65, Crema. Italy.

Prof. Dr. Letizia Tanca, Dipartimento di Elettronica e Informazione – Politecnico di Milano, Via Ponzio 121, Milano. Italy.

Prof. Dr. Francesca Arcelli Fontana, Dipartimento di Informatica – Università di Milano–Bicocca, Via Bicocca degli Arcimboldi, Milano. Italy.

e-mails: edamiani@crema.unimi.it, tanca@elet.polimi.it, arcelli@disco.unimib.it