Approximating the DTD of a set of XML documents

Alberto Abelló
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Xavier de Palol
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Mohand-Saïd Hacid
LIRIS- UFR d'Informatique
Universite Claude Bernard Lyon 1

March 8, 2005

Abstract

The WWW contains a huge amount of documents. Some of them share the subject, but are generated by different people or even organizations. To guarantee the interchange of such documents, we can use XML. This allows to share documents that do not have the same structure. However, it makes difficult to understand the core of such heterogeneous documents (in general, schema is not available). In this paper, we offer a characterization and algorithm to obtain the midpoint (in terms of a resemblance function) of a set of semi-structured, heterogeneous documents without optional elements. The trivial case of midpoint would be the common elements to all documents. Nevertheless, in cases with several heterogeneous documents this may result in an empty set. Thus, we consider that those elements present in a given amount of documents belong to the midpoint. Once we have such midpoint, the algorithm is generalized for the obtaining of repetitions and optional elements. Thus, a exact schema can always be found generating optional elements. However, the exact schema of the whole set may result in overspecialization (lots of optional elements), which would make it useless.

1 Introduction

The web is a powerful medium for human communication and dissemination of information. Consequently, the web has become a popular knowledge base, where people add documents (private, educational and organizational) and navigate through its content. The rapid growth of information makes it sheer impossible to find, organize, access and maintain the information as the users require. There is a clear need for the web to become semantically richer. The aim of this semantic enrichment of the web is to enhance web searches and to introduce logical reasoning on the web contents [EL02]. The general trend towards Semantic Web (SW) is the use of ontologies to incorporate such semantics to existing and new documents.

For scalability reasons, one important aspect of SW consists in distilling the existing documents and extract valuable knowledge from them. There exist multiple formats for information sources, ranging from unstructured data to highly structured. The term semi-structured data has emerged to describe data that has some structure but neither regular, nor known a-priori to the system. It is precisely for this reason that semi-structured documents are self-describing.

The importance of knowing the structure (or schema) of a set of documents have been largely described in the literature. For example, [BGM04] outlines its importance on integrating and analyzing structure of the WWW. On the other hand, [ABS00] points out that a known structure would also facilitate the storage (by compressing and exploiting the commonalities) and it would also encourage the queries on these commonalities. It is key to improve the access methods to the data, thus availing query optimization. Moreover, one could use the structure to derive a new ontology or identify (map) the concepts in the documents into an existing ontology; an ontology establishes a common vocabulary for a given domain, which can be used to data interchange among companies. Ontologies are also used for documentation purposes, since they provide an structured view of a domain.

Here we consider a certain kind of semi-structured data, in particular, XML documents. XML has been adopted as standard for data interchange, availing the integration of heterogeneous information sources. A *well-formed* XML document is a document that conforms to the XML syntax rules in [W3C04] (roughly, markups nest properly and attributes are unique). Moreover, a *valid* XML document is a document that is *well-formed* and also conforms to the rules of its DTD. A DTD contains the declarations that provide a grammar for a class of documents. It determines the *element*s and *attribute*s that appear in a document, i.e., the name, type and constraints on every *element* and *attribute*.

As defined in [W3C04], an XML document primarily consists of a nested hierarchy of *elements* with a single root. *Elements* can contain character data (concepts) and *child element*s, where both can have *attributes*. *Child elements* consist either of a *sequence* list of *element*s or a *choice* list of *element*s. The standard states that *element*s in a *sequence* must be ordered. From a practical point of view, an application deals better with a well-known structure and a fixed appearing order, but in general the order of the different subelements among them is not relevant, that is, we can change the order of these subelements without changing the meaning of the overall document. In fact, many interfaces have been developed to access and manipulate an XML document (probably the most popular ones are DOM[1] and SAX[2]), and most of them do not consider the order among subelements.

The *choice* construct in a DTD indicates that one, and only one, *element* in the *choice* list of contents should appear in the document. The *choice* construct is the key to find a perfect typing. In the rare case that all the documents belong to the same class and use the same terms, the *choice* construct is not needed to find a perfect typing. Otherwise, in a grammar that lacks the *choice* construct we cannot find a common schema, so we have to approximate it. Such approximated schemas are called inexact schemas, whose usefulness was already pointed out in [Wid99].

If we use the *choice* construct, finding the schema is reduced to find the best grammar expression for each *element* (for example following a normal form like [AGW01]), so that all *element*s in the document belong to the corresponding grammar. Nevertheless, a perfect schema, one DTD that is followed by all the documents, may arise an overspecialization problem. Some works have overcome overspecialization by using clustering techniques to approximate typing [NAM98, SPBA03].

We aim at finding a common schema for a set of correct semi-structured documents. We take an inexact approach based on the resemblance of documents, thus using the structure similarity among the documents under study. We call this common schema the midpoint. We use the resemblance family of functions in [BGM04], which take into account extra *element*s both in the document and in the DTD. We could then redefine *valid* XML document as a document whose resemblance to its DTD is above a given threshold. The main contribution of this paper is the characterization of the midpoint in terms of a resemblance function and offer an efficient algorithm to obtain it. Although our approach

[1] http://www.w3.org/DOM
[2] http://www.saxproject.org/

deals with DTDs, it also applies to XML schemas.

The structure of the paper is as follows. In the next section we review the work related with our method. Section 3 presents the formalization of XML into DL that we propose. Section 4 characterizes the midpoint. Section 5 shows an efficient algorithm to obtain the midpoint. Section 6 shows how to deal with optional and repeated *element*s. Finally, section 7 gives the general conclusions and points out our future work.

2 Related work

As pointed out in [Wid99], everything needs to scale to web proportion. However, human mind cannot. Nobody would catch at once the essence of thousands of documents. Some kind of schema (i.e. either structured or semi-structured) should be available, summarizing the contents of every set of documents. Thus, several authors worked on the generation of DTDs from XML data.

A relevant result is [NAM98], which explains how we can get a well structured schema (i.e. not a DTD) approximating the documents. [JOKA02] describes an implementation of an algorithm to generate a DTD followed by an XML document. [SPBA03] classifies the documents in different classes and gets one DTD per class of documents. This is a good solution if there are a few classes with not many documents or *element*s each. However, it may result in lots of different classes or optional *element*s for every class, if we are dealing with a huge amount of heterogeneous documents.

On the other hand, [NAM98] pays attention to inexact schemas, outlining that the size of a perfect typing may be the order of the data set, prohibiting its use for query optimization and interfaces. Therefore, we are not searching a perfect typing but a human-friendly, computationally-tractable, and graphically-representable approximation. To this end, we should use some kind of resemblance or distance. The first option would be tree edit distance (like in [BdR04]), but it results in high complexity (see [ZS89]). Therefore, the most promising option is that of structure similarity. [NAM98] uses Manhattan distance (i.e. the number of different descendants/ancestors of two *element*s), and explains that there are several domain dependent ways to weight it. [BB95] shows different resemblance measures. Among those, [SPBA03] uses $\frac{|elem(d_1) \cap elem(d_2)|}{max(|elem(d_1)|,|elem(d_2)|)}$ (being d_1 and d_2 two documents), while [BGM04] uses $\frac{|elem(d_1) \cap elem(d_2)|}{|elem(d_1) \cap elem(d_2)| + \alpha \cdot |elem(d_1) \setminus elem(d_2)|, \beta \cdot |elem(d_2) \setminus elem(d_1)|)}$. We took this last measure, because it is more general, and allows to distinguish lack of *element*s in one side or another.

Regarding costs, that of [NAM98] is quadratic for its first step and NP-hard (approximated by a logarithmic greedy algorithm) for its second one. On the other hand, obtaining the schema of every class in [SPBA03] (which corresponds to a exact DTD for all documents in the class) is linear in the number of *element*s in the representatives, while finding the class of every document is worse case quadratic in the number of documents (we may need to check every document against every other document).

3 Formalizing XML documents by means of Description Logics

As we can see in [ABS00], an XML document uses to be thought as a rooted tree. A rooted tree is an acyclic graph (N,E), that has no more than one root. N is a set of nodes and E a set of edges. An edge e is an ordered pair of nodes

(n_{source}, n_{target}). A node is a leaf, if it is not the source of any edge in E. Along this paper we will use Description Logics (DL) notation to formalize those trees.

Since we only take into account *element* tags, we are not actually interested in XML documents, but in a restricted class of DTDs that can be automatically generated from one XML document. The problem tackled in this paper is that of finding a DTD from a set of XML documents. Nevertheless, for the sake of simplicity, from here on, we will use the DTDs corresponding to the documents instead of the documents themselves. We assume that we have a DTD exactly matching each document. The DTD of a document can be obtaining just parsing it and eliminating data (leaving *element* tags). Thus, these *DTD*s cannot contain *choice*, nor *unnumbered repetitions*, nor *optional elements*, nor *any*, because from one document we are not able to infer such structures. How could we know based only on one document that a present *element* may not be present? How could we decide that there is a possibly infinite repetition?

Regarding XML attributes, they could be used to match different *element* tags. For example, "<a ID='Id1'>" should be identified with "" in spite of the different tag name. Nevertheless, that is not the aim of this paper. Representing the information either as an attribute or a child is just a design choice. Thus, from here on, without loss of generality, we will consider XML *attribute*s as XML *child element*s without further nesting structure.

As stated in [W3C04], *child elements* are ordered. Order is an important characteristic for documents. However, in databases unordered data can be processed more efficiently, so it uses to be considered in that way (for example in DOM and SAX). Therefore, we will assume that order is not relevant in our case.

[CLN98] already showed the usefulness of DL on conceptual modeling. Thus, we will consider a set of documents as a knowledge base, which comprises two components, i.e. TBox (the terminology, we could recognize it as the schema) and ABox (the assertions about individuals, or instances). As explained in [BCM+03], the TBox contains concepts, and to define a formal semantics of the logic we use an interpretation $\mathcal{I}$. An interpretation is a pair $[\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}]$, where $\Delta^{\mathcal{I}}$ is the domain (a non-empty set), and $\cdot^{\mathcal{I}}$ is an interpretation function that assigns to every atomic concept A a set ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$) and to every atomic role r a binary relation ($r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). Inductively, this is extended to non-atomic concepts by the following definitions (where C and D are concepts, and r is a role):

$$\begin{aligned} \bot^{\mathcal{I}} &= \emptyset \\ \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\exists r.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \end{aligned}$$

document 1: <a><c>Hello</c><d><e>Bye</e></d>
document 2: <a><d></d>
document 3: <a><d><e>Bye</e></d>
document 4: <a><d><e>Bye bye</e></d>

$dtd_1 = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top)$
$dtd_2 = \exists a.(\exists b.\bot \sqcap \exists d.\bot)$
$dtd_3 = \exists a.\exists d.\exists e.\top$
$dtd_4 = \exists a.\exists d.\exists e.\top$

element:	C (concept)
child element:	$\exists r.C$ (existential quantification)
sequence:	$\sqcap$ (conjunction)
PCDATA or *String*:	$\top$ (top)
EMPTY:	$\bot$ (bottom)

Figure 1: DL representation of an XML document

As exemplified in figure 1, we will represent a *document* or piece of document by a concept "C". An unordered *sequence*

of pieces of documents will be represented by a conjunction "$C \sqcap D$". Data types (i.e. *PCDATA* and *string*) will be represented by the top concept "$\top$", while an empty *element* (i.e. *EMPTY*) will be represented by bottom concept "$\bot$". Finally, *children* will be represented by means of existential quantification "$\exists element.C$". Actually, existential quantification allows the presence of more than one *element* of the same kind. Nevertheless, as stated before, we do not consider such repetitions by now (see section 6.2 for the treatment of repetitions).

We did not use a formalization of XML documents like that in [CDL99] because it does not allow to reason. There, different kinds of elements are represented in the ABox, at the instance level, while reasoning algorithms needed work at the conceptual level. Our formalization allows the usage of DL algorithms like "Subsumption" and "Least Common Subsumer":

Subsumtion (also known as "Query Containment" in other areas and noted "$C \sqsubseteq$D", if C is subsumed by D) shows whether one concept is more general than another (i.e. one set contains the other for all interpretations). For example, $dtd_1 \sqsubseteq dtd_3$.

$$C \sqsubseteq D \Leftrightarrow \forall \mathcal{I} : C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$

Least Common Subsumer ("LCS" from here on) results in the subsumer of a set of concepts that is subsumed by any other subsumer of the set of documents. LCS uses to be applied to learning from examples, and bottom-up construction of knowledge bases. For example, $lcs(dtd_2, dtd_4) = \exists a.\exists d.\top$.

$$L = lcs(C_1, .., C_n) \Leftrightarrow \forall i : C_i \sqsubseteq L \wedge \nexists D : (\forall i : C_i \sqsubseteq D \wedge D \sqsubseteq L)$$

4 Characterization of the midpoint

Given a set of DTDs, we would like to find the DTD that has the maximum number of common *element*s wrt the set, at the same time that minimizes the *element*s being in the DTD not in the documents and those in the documents not in the DTD. We will call such DTD the midpoint of the set. In order to characterize the midpoint, we will use the resemblance family of functions used in [BGM04].

$$r : (DTD, setOfDTDs) \mapsto [0, 1]$$

$$r(C, E) = \frac{w_c(C, E)}{w_c(C, E) + \alpha \cdot w_p(C, E) + \beta \cdot w_m(C, E)} \; for \; \alpha, \beta \in \mathbb{R}^+$$

By instantiating α and β we get the concrete function we would like to use (notice that only if $\alpha = \beta$ the resemblance will be symmetric). Positive real values can be assigned to these parameters. They weight the importance of finding plus (*element*s in some DTD that do not appear in the midpoint) and minus (*element*s in the midpoint that do not appear in some DTD) *element*s respectively. The function relies now on three simpler ones that obtain respectively the size of common, plus, and minus *element*s. It is interesting to notice that the sum of common and plus *element*s corresponds to the size of all DTDs independently of the concept we are obtaining the distance to.

$$w_c(C, E) = \sum_{dtd \in E} size(lcs(C, dtd))$$

$$w_p(C, E) = \sum_{dtd \in E} (size(dtd) - size(lcs(C, dtd)))$$

$$w_m(C,E) = \sum_{dtd \in E} (size(C) - size(lcs(C,dtd)))$$

$$\forall C : w_c(C,E) + w_p(C,E) = \sum_{dtd \in E} size(dtd)$$

Any result in this paper does not depend on how we compute the size of a DTD. We only impose that the size of a DTD is smaller than the size of adding an *element* to that DTD. Therefore, from here on, in the examples we will assume that every *element* contributes to the size of a DTD with one unit independently of its position in the document. For example, $size(dtd_1) = 5$ and $size(dtd_2) = size(dtd_3) = 3$. A general, more complex and accurate algorithm for obtaining the size of a DTD is given in [BGM04].

$$r(\exists a.\exists d.\top, \{dtd_2, dtd_3\}) = \frac{4}{4+2\alpha+0\beta} = \frac{4}{4+4}$$
$$r(\exists a.\exists d.\exists e.\top, \{dtd_2, dtd_3\}) = \frac{5}{5+\alpha+\beta} = \frac{5}{5+2+3}$$
$$r(\exists a.(\exists b.\top \sqcap \exists d.\top), \{dtd_2, dtd_3\}) = \frac{5}{5+\alpha+\beta} = \frac{5}{5+2+3}$$
$$r(\exists a.(\exists b.\top \sqcap \exists d.\exists e.\top), \{dtd_2, dtd_3\}) = \frac{6}{6+0\alpha+2\beta} = \frac{6}{6+6}$$

Figure 2: Example of multiple midpoints

At this point, it is also important to notice that there may exist more than one DTD maximizing the resemblance (i.e. more than one midpoint). For example, let be $\alpha = 2$ and $\beta = 3$. In this case, $r(\exists a.\exists d.\top, \{dtd_2, dtd_3\}) = r(\exists a.\exists d.\exists e.\top, \{dtd_2, dtd_3\}) = r(\exists a.(\exists b.\top \sqcap \exists d.\top), \{dtd_2, dtd_3\}) = r(\exists a.(\exists b.\top \sqcap \exists d.\exists e.\top), \{dtd_2, dtd_3\})$, as we can see in figure 2. Since this is the maximum resemblance, we can choose the midpoint of $\{dtd_2, dtd_3\}$ among those four DTDs.

As stated by theorem 1, one of the possible midpoints of the set can be obtained by a conjunction of LCS of the documents.

Theorem 1. *Given a set of DTDs $E = \{dtd_1,...,dtd_n\}$, and being B_i branches of the form $\exists r^1_{B_i}.\exists r^2_{B_i}...\exists r^{l_i}_{B_i}.\top$ with $l_i \geq 1$*

$$\exists S_1, ..., S_p \in \mathscr{P}(E) : \forall B_1, ..., B_q : r(\bigsqcap_{i=1..q} B_i, E) \leq r(\bigsqcap_{j=1..p} lcs(S_j), E)$$

Proof. By hypothesis, let's suppose that there is a concept $M = \bigsqcap_{i=1..m} B_i$ that maximizes the resemblance and it is not a conjunction of LCSs. We will divide the proof in three steps:

$$E_C = \{dtd \in E \mid dtd \sqsubseteq C\}$$

Step 1: $\forall i = 1..q : E_{B_i} \neq \emptyset$

Let's suppose not, i.e. $\exists i = 1..q : E_{B_i} = \emptyset$. We can remove the last k existentials from it until there exists some DTD d with a branch matching B'_i (being $B'_i = \exists r^1_{B_i}.\exists r^2_{B_i}...\exists r^{l_i-k}_{B_i}.\top$). Now, $d \sqsubseteq B'_i$. Let be $M' = B_1 \sqcap ... \sqcap B_{i-1} \sqcap B'_i \sqcap B_{i+1} \sqcap ... \sqcap B_q$. It is easy to see that $w_c(M,E) = w_c(M',E)$, $w_p(M,E) = w_p(M',E)$, and $w_m(M,E) \geq w_m(M',E)$. Thus,

$$r(M,E) \leq r(M',E)$$

which means they are equal (if $\beta = 0$) or contradicts the hypothesis. Therefore, we can assume that $\forall i = 1..q : E_{B_i} \neq \emptyset$.

Step 2: B_i is exactly a branch of $lcs(E_{B_i})$

Let's suppose not, because the corresponding chain of existentials in $lcs(E_{B_i})$ is longer than B_i (notice that it can never be shorter, by definition of the LCS). Let's call B_L to $\exists r^1_{B_i}.\exists r^2_{B_i}...\exists r^{l_i}_{B_i}...\exists r^{l_i+k}_{B_L}.\top$ so that it is a branch of $lcs(E_{B_i})$. Let be $M' = B_1 \sqcap ... \sqcap B_{i-1} \sqcap B_L \sqcap B_{i+1} \sqcap ... \sqcap B_q$. Notice that $\forall dtd \in E \setminus E_{B_i} : size(lcs(dtd, B_L)) = size(lcs(dtd, B_i))$, because if exists a DTD with a branch B' so that $B_L \sqsubseteq B' \sqsubset B_i$, by definition it belongs to E_{B_i}. Therefore, it is easy to see that:

$$w_c(M', E) = w_c(M, E) + \mid E_{B_i} \mid \cdot(size(B_L) - size(B_i))$$

$$w_p(M', E) = w_p(M, E) + \mid E_{B_i} \mid \cdot(size(B_i) - size(B_L))$$

$$w_m(M', E) = w_m(M, E) + \mid E \setminus E_{B_i} \mid \cdot(size(B_L) - size(B_i))$$

By hypothesis, $r(M, E) \geq r(M', E)$. Thus,

$$\frac{w_c(M,E)}{w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)} \geq \frac{w_c(M,E) + \mid E_{B_i} \mid \cdot(size(B_L) - size(B_i))}{(w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)) + (\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid) \cdot (size(B_L) - size(B_i))}$$

Let be $B'_i = \exists r^1_{B_i}.\exists r^2_{B_i}...\exists r^{l_i-1}_{B_i}.\top$. Since, by hypothesis, B_i is not exactly a branch of any DTD, $\forall dtd \in E \setminus E_{B_i} : size(lcs(dtd, B'_i)) = size(dtd, B_i))$. Therefore, defining $M'' = B_1 \sqcap ... \sqcap B_{i-1} \sqcap B'_i \sqcap B_{i+1} \sqcap ... \sqcap B_q$:

$$w_c(M'', E) = w_c(M, E) + \mid E_{B_i} \mid \cdot(size(B'_i) - size(B_i))$$

$$w_p(M'', E) = w_p(M, E) + \mid E_{B_i} \mid \cdot(size(B_i) - size(B'_i))$$

$$w_m(M'', E) = w_m(M, E) + \mid E \setminus E_{B_i} \mid \cdot(size(B'_i) - size(B_i))$$

and given that by hypothesis r(M,E) is the maximum:

$$\frac{w_c(M,E)}{w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)} \geq \frac{w_c(M,E) - \mid E_{B_i} \mid \cdot(size(B_i) - size(B'_i))}{(w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)) - (\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid) \cdot (size(B_i) - size(B'_i))}$$

However, both inequalities cannot be possible at the same time, because $size(B_L) - size(B_i)$ and $size(B_i) - size(B'_i)$ are both positive numbers. Therefore, the hypothesis is not true and B_i must be exactly a branch of $lcs(E_{B_i})$.

Step 3: $M \sqsubseteq lcs(E_{B_L})$

As before, let be $M' = B_1 \sqcap ... \sqcap B_{i-1} \sqcap B_L \sqcap B_{i+1} \sqcap ... \sqcap B_q$. We already showed that

$$r(M, E) \leq r(M', E)$$

$$\frac{w_c(M,E)}{w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)} \leq \frac{w_c(M,E) + \mid E_{B_i} \mid \cdot(size(B_L) - size(B_i))}{(w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)) + (\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid) \cdot (size(B_L) - size(B_i))}$$

Being $a, b, c, d \in \mathbb{R}^+ \setminus \{0\} : \frac{a}{b} \leq \frac{a+c}{b+d} \Leftrightarrow \frac{a}{b} \leq \frac{c}{d}$. Therefore,

$$\frac{w_c(M,E)}{w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)} \leq \frac{\mid E_{B_i} \mid \cdot(size(B_L) - size(B_i))}{(\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid) \cdot (size(B_L) - size(B_i))} = \frac{\mid E_{B_i} \mid}{\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid}$$

In our case, functions (i.e. w_c, w_p, and w_m) are always positive (which makes a and b also positive). If a (i.e. w_c) is zero, inequality is still true, because any other DTD would never worsen the resemblance. On the other hand, b (i.e. $w_c(M, E) + \alpha \cdot w_p(M, E) + \beta \cdot w_m(M, E)$) being zero means that $E = \emptyset$ (we do not have any DTD). In some cases, depending on α and β, d (i.e. $\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid$) may be negative or zero. If so,

it is easy to see that $r(M,E) \leq r(M',E)$, which would contradict the hypothesis. Therefore, from here on, we can assume that $\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid$ is strictly positive.

Let's suppose that B'_L is branch of $lcs(E_{B_L})$ so that $\neg(M \sqsubseteq B'_L)$. Let be $M'' = B_1 \sqcap ... \sqcap B_q \sqcap B'_L$, and $B_j = lcs(M, B'_L)$.

$$w_c(M'',E) = w_c(M,E) + \sum_{dtd \in E} (size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd)))$$

$$w_p(M'',E) = w_p(M,E) + \sum_{dtd \in E} (size(lcs(B_j, dtd)) - size(lcs(B'_L, dtd)))$$

$$w_m(M'',E) = w_m(M,E) + \sum_{dtd \in E} (size(B'_L) - size(B_j) - size(lcs(B'_L, dtd)) + size(lcs(B_j, dtd)))$$

Therefore,

$$r(M,E) \leq r(M'',E) \Leftrightarrow r(M,E) \leq \frac{w_c(M'',E) - w_c(M,E)}{w_c(M'',E) - w_c(M,E) + \alpha \cdot (w_p(M'',E) - w_p(M,E)) + \beta \cdot (w_m(M'',E) - w_m(M,E))}$$

$$\begin{aligned}
&\frac{w_c(M'',E) - w_c(M,E)}{w_c(M'',E) - w_c(M,E) + \alpha \cdot (w_p(M'',E) - w_p(M,E)) + \beta \cdot (w_m(M'',E) - w_m(M,E))} \\
&= \frac{\sum_{dtd \in E}(size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd)))}{(1-\alpha) \cdot (\sum_{dtd \in E}(size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd)))) + \beta \cdot (\sum_{dtd \in E}(size(B'_L) - size(B_j) - size(lcs(B'_L, dtd)) + size(lcs(B_j, dtd))))} \\
&= \frac{1}{(1-\alpha) + \beta \cdot (\frac{\sum_{dtd \in E}(size(B'_L) - size(B_j))}{\sum_{dtd \in E}(size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd)))} - 1)} \\
&= \frac{1}{(1-\alpha) + \beta \cdot (\frac{|E| \cdot (size(B'_L) - size(B_j))}{\sum_{dtd \in E}(size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd)))} - 1)} \\
&= \frac{1}{(1-\alpha) + \beta \cdot (\frac{|E|}{\sum_{dtd \in E} \frac{size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd))}{size(B'_L) - size(B_j)}} - 1)}
\end{aligned}$$

Since $B'_L \sqsubset B_j$, then $\forall dtd \in E \setminus E_{B_j} : lcs(B'_L, dtd)) = lcs(B_j, dtd)$. Therefore,

$$\frac{1}{(1-\alpha) + \beta \cdot (\frac{|E|}{\sum_{dtd \in E} \frac{size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd))}{size(B'_L) - size(B_j)}} - 1)} = \frac{1}{(1-\alpha) + \beta \cdot (\frac{|E|}{\sum_{dtd \in E_{B_j}} \frac{size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd))}{size(B'_L) - size(B_j)}} - 1)}$$

Moreover, $\forall dtd \in E_{B_j} : lcs(B_j, dtd) = B_j$, $\forall dtd \in E_{B'_L} : lcs(B'_L, dtd) = B'_L$, and $\forall dtd \in E_{B_j} \setminus E_{B'_L} : size(B'_L) > size(lcs(B'_L, dtd)) > size(B_j)$. Therefore,

$$\frac{1}{(1-\alpha) + \beta \cdot (\frac{|E|}{\sum_{dtd \in E_{B_j}} \frac{size(lcs(B'_L, dtd)) - size(lcs(B_j, dtd))}{size(B'_L) - size(B_j)}} - 1)} \geq \frac{1}{(1-\alpha) + \beta \cdot (\frac{|E|}{|E_{B_j}|} - 1)} = \frac{1}{(1-\alpha) + \beta \frac{|E \setminus E_{B_j}|}{|E_{B_j}|}}$$

Finally, since $\mid E_{B_j} \mid \geq \mid E_{B'_L} \mid \geq \mid E_{B_L} \mid = \mid E_{B_i} \mid$, then

$$\frac{1}{(1-\alpha) + \beta \frac{|E \setminus E_{B_j}|}{|E_{B_j}|}} \geq \frac{1}{(1-\alpha) + \beta \frac{|E \setminus E_{B_i}|}{|E_{B_i}|}}$$

$$\frac{\mid E_{B_j} \mid}{\mid E_{B_j} \mid -\alpha \mid E_{B_j} \mid +\beta \mid E \setminus E_{B_j} \mid} \geq \frac{\mid E_{B_i} \mid}{\mid E_{B_i} \mid -\alpha \mid E_{B_i} \mid +\beta \mid E \setminus E_{B_i} \mid} \geq \frac{w_c(M,E)}{w_c(M,E) + \alpha \cdot w_p(M,E) + \beta \cdot w_m(M,E)}$$

Therefore, if adding B_i improves the resemblance, adding B_j improves the resemblance even more. So, every branch of $lcs(E_{B_L})$ subsumes M, which means that $M \sqsubseteq lcs(E_{B_L})$.

□

Lemma 2. *There exists a DTD of the form* $\bigsqcap_{k=1..p} lcs(S_k)$ *maximizing the resemblance, so that* $\forall 1 \leq i, j \leq p : (S_i \nsubseteq S_j)$.

Proof. If $\exists i, j : S_i \subset S_j$, then $\forall dtd \in E : lcs(\bigsqcap_{k=1..p} lcs(S_k), dtd) = lcs(\bigsqcap_{k=1..j-1} lcs(S_k) \sqcap \bigsqcap_{k=j+1..p} lcs(S_k), dtd)$ and $size(\bigsqcap_{k=1..p} lcs(S_k)) = size(\bigsqcap_{k=1..j-1} lcs(S_k) \sqcap \bigsqcap_{k=j+1..p} lcs(S_k))$. Therefore,

$$r(\bigsqcap_{k=1..p} lcs(S_k), E) = r(\bigsqcap_{k=1..j-1} lcs(S_k) \sqcap \bigsqcap_{k=j+1..p} lcs(S_k), E)$$

□

Corollary 3. *There exists a DTD of the form* $\bigsqcap_{k=1..p} lcs(S_k)$ *maximizing the resemblance, so that* $p \leq \begin{pmatrix} \mid E \mid \\ \frac{\mid E \mid}{2} \end{pmatrix}$

5 Obtaining the midpoint of a set of DTD

First of all, it is important to notice that depending on the values of α and β there are some trivial cases (as shown in table 1). If $\alpha = 0$, we do not mind having extra *element*s in the DTDs wrt the midpoint. Therefore, among the multiple solutions to the problem, we find $\exists element.\top$ (where "element" is the most frequent root *element* in the documents). If $\beta = 0$, we do not mind having extra *element*s in the midpoint wrt every individual DTD. Therefore, $\bigsqcap_{dtd \in E} dtd$ is among the solutions. Both equaling zero means that just by matching some *element*s in some DTD we get maximum resemblance (i.e. $\forall w_c \neq 0 : \frac{w_c}{w_c + 0w_p + 0w_m} = 1$). Thus, from here on, we will only consider the non-trivial case where $\alpha \neq 0$ and $\beta \neq 0$.

Midpoint	$\beta = 0$	$\beta \neq 0$
$\alpha = 0$	any	$\exists element.\top$
$\alpha \neq 0$	$\bigsqcap_{dtd \in E} dtd$	?

Table 1: Trivial cases on finding a midpoint

In this section we analyze three different possibilities to obtain the midpoint of a set of DTDs. Section 5.1 explores the behaviour of a blind search in the space of candidate DTDs. Section 5.2 tries to get benefit from the knowledge of the structure of the midpoint by restricting the search to those DTDs being LCS of some DTDs. Finally, section 5.3 shows how we could find the midpoint just looking at the number of appearances of each *element*.

5.1 Blind search

The first possibility to find the midpoint of a set of documents is generating the whole search space and compare every point to the documents until we find the maximum resemblance.

Firstly, as we can see in figure 3, we need to get all *element*s in the set of documents, which needs linear time on the number of *element*s (assuming there is a small number of children or they are ordered). After that, we need to generate all possible points in the search space (i.e. the parts of the set of *element*s), which is exponential in the

```
BSET := ∅;
foreach dtd ∈ E do
    foreach branch ⊒ dtd do
        if not branch ∈ BSET then
            BSET := BSET ∪ {branch};
            endif;
        endforeach;
    endforeach;
M := ⊤;
max := 0;
foreach bset ∈ 𝒫(BSET)
    C := ⨅_{B∈bset} B;
    if r(C, E) > max then
        M := C;
        max := r(C, E);
        endif;
    endforeach;
```

Figure 3: Blind search algorithm

number of *element*s (without repetitions). Moreover, for each one of this points we need to calculate its resemblance to the set of documents. It is clear that this approach is prohibitive in terms of time cost.

Based on the four DTDs in figure 1, figure 4 shows the nine candidates in the search space (generated by blind search) and their distances to the set of documents. If we take $\alpha = \beta = 1$, then the best solution would be C_8, whose resemblance is $\frac{13}{17}$.

5.2 Finding the combination of LCSs

From section 4, we know that the midpoint is composed by a set of LCS. The idea of this section is to restrict the search space, by limiting it to those DTDs that are LCS of some subset of DTDs.

Figure 5 sketches the algorithm. We can see that it consists of two phases. The first one generates all LCSs of the documents. The number of LCSs is exponential in the number of documents, and generating one of them can be considered linear in the number of *element*s in one document (for small number of children or ordered children). Afterwards, we need to generate all possible combinations of LCSs (exponential in the number of LCS) and get the resemblance from each one of them to the set of documents. The maximum resemblance would result in the midpoint.

As before, we consider the four DTDs in figure 1. Figure 6 shows the five different candidates (the other ten candidates coincide with one of those) and their distances to the set of documents. If we take $\alpha = \beta = 1$, then the best solution would be C_5, whose resemblance is $\frac{13}{17}$. One may think that the midpoint should be composed by the LCSs with highest resemblance. Nevertheless, in this example, we can see that $lcs(dtd_2)$ participates in the midpoint, while $lcs(dtd_1)$ does not, in spite of $r(lcs(dtd_1), E) > r(lcs(dtd_2), E)$. The problem is that the contribution of one LCS to the resemblance of the midpoint depends on the other LCS belonging to the midpoint.

$$BSET = \{\exists a.\top, \exists a.\exists b.\top, \exists a.\exists b.\exists c.\top, \exists a.\exists d.\top, \exists a.\exists d.\exists e.\top\}$$

$C_1 = \exists a.\top$	$r(C_1, E) = \frac{4}{4+10\alpha+0\beta}$
$C_2 = \exists a.\exists b.\top$	$r(C_2, E) = \frac{6}{6+8\alpha+2\beta}$
$C_3 = \exists a.\exists d.\top$	$r(C_3, E) = \frac{8}{8+6\alpha+0\beta}$
$C_4 = \exists a.\exists b.\exists c.\top$	$r(C_4, E) = \frac{7}{7+7\alpha+5\beta}$
$C_5 = \exists a.(\exists b.\top \sqcap \exists d.\top)$	$r(C_5, E) = \frac{10}{10+4\alpha+2\beta}$
$C_6 = \exists a.\exists d.\exists e.\top$	$r(C_6, E) = \frac{11}{11+3\alpha+1\beta}$
$C_7 = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\top)$	$r(C_7, E) = \frac{11}{11+3\alpha+5\beta}$
$C_8 = \exists a.(\exists b.\top \sqcap \exists d.\exists e.\top)$	$r(C_8, E) = \frac{13}{13+1\alpha+3\beta}$
$C_9 = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top)$	$r(C_9, E) = \frac{14}{14+0\alpha+6\beta}$

Figure 4: Candidate DTDs generated in a blind search

$LCS := \emptyset$;
foreach $S \in \mathscr{P}(E)$
  $LCS := LCS \cup lcs(S)$;
  endforeach;
$M := \top$;
$max := 0$;
foreach $L \in \mathscr{P}(LCS)$
  $C := \bigsqcap_{l \in L} l$
  if $r(C, E) > max$ **then**
    $M := C$;
    $max := r(C, E)$;
    endif;
  endforeach;

Figure 5: Algorithm by using LCS

5.3 Picking up based on appearance

This section shows the possibility of finding a midpoint just based on the appearances of each *element* in the set of documents. The first question to answer is how we could know whether the point in the search space we are treating is better than another candidate or not. Surprisingly, it is not necessary to get all plus and minus *element*s. By theorem 4, we know that all we need is the number of common *element*s between each of both DTDs and the set of DTDs E.

Theorem 4. *To decide whether the resemblance of a DTD C against a set of DTDs is better than that of another DTD C', it is only necessary to consider the common* element*s (neither plus, nor minus).*

Proof.

$$r(C, E) \geq r(C', E)$$

$$\frac{w_c(C,\,E)}{w_c(C,\,E) + \alpha \cdot w_p(C,\,E) + \beta \cdot w_m(C,\,E)} \geq \frac{w_c(C',\,E)}{w_c(C',\,E) + \alpha \cdot w_p(C',\,E) + \beta \cdot w_m(C',\,E)}$$

$lcs(dtd_1) = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top)$
$lcs(dtd_2) = lcs(dtd_2, dtd_1) = \exists a.(\exists b.\top \sqcap \exists d.\top)$
$lcs(dtd_3) = lcs(dtd_4) = lcs(dtd_1, dtd_3) = lcs(dtd_1, dtd_4) = lcs(dtd_1, dtd_3, dtd_4) = \exists a.\exists d.\exists e.\top$
$lcs(dtd_2, dtd_3) = lcs(dtd_1, dtd_2, dtd_3) = lcs(dtd_2, dtd_4) = lcs(dtd_1, dtd_2, dtd_4) = lcs(dtd_1, dtd_2, dtd_3, dtd_4) = \exists a.\exists d.\top$

$$
\begin{array}{ll}
C_1 = lcs(dtd_1) = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top) & r(C_1, E) = \frac{14}{14+0\alpha+6\beta} \\
C_2 = lcs(dtd_2) = \exists a.(\exists b.\top \sqcap \exists d.\top) & r(C_2, E) = \frac{10}{10+4\alpha+2\beta} \\
C_3 = lcs(dtd_3) = \exists a.\exists d.\exists e.\top & r(C_3, E) = \frac{11}{11+3\alpha+1\beta} \\
C_4 = lcs(dtd_2, dtd_3) = \exists a.\exists d.\top & r(C_4, E) = \frac{8}{8+6\alpha+0\beta} \\
C_5 = lcs(dtd_2) \sqcap lcs(dtd_3) = \exists a.(\exists b.\top \sqcap \exists d.\exists e.\top) & r(C_5, E) = \frac{13}{13+1\alpha+3\beta}
\end{array}
$$

Figure 6: Candidate DTDs generated in a search using LCSs

$$\frac{\sum_{dtd\in E} size(lcs(C, dtd))}{\sum_{dtd\in E} size(lcs(C, dtd)) + \alpha \cdot \sum_{dtd\in E}(size(dtd) - size(lcs(C, dtd))) + \beta \cdot \sum_{dtd\in E}(size(C) - size(lcs(C, dtd)))}$$
$$\geq$$
$$\frac{\sum_{dtd\in E} size(lcs(C', dtd))}{\sum_{dtd\in E} size(lcs(C', dtd)) + \alpha \cdot \sum_{dtd\in E}(size(dtd) - size(lcs(C', dtd))) + \beta \cdot \sum_{dtd\in E}(size(C') - size(lcs(C', dtd)))}$$

$$(\sum_{dtd\in E} size(lcs(C, dtd)))(\sum_{dtd\in E} size(lcs(C', dtd)) + \alpha \cdot \sum_{dtd\in E}(size(dtd) - size(lcs(C', dtd))) + \beta \cdot \sum_{dtd\in E}(size(C') - size(lcs(C', dtd))))$$
$$\geq$$
$$(\sum_{dtd\in E} size(lcs(C', dtd)))(\sum_{dtd\in E} size(lcs(C, dtd)) + \alpha \cdot \sum_{dtd\in E}(size(dtd) - size(lcs(C, dtd))) + \beta \cdot \sum_{dtd\in E}(size(C) - size(lcs(C, dtd))))$$

$$(\sum_{dtd\in E} size(lcs(C, dtd)))(\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \sum_{dtd\in E} size(C')) \geq (\sum_{dtd\in E} size(lcs(C', dtd)))(\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \sum_{dtd\in E} size(C))$$

$$\frac{\sum_{dtd\in E} size(lcs(C, dtd))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot size(C)} \geq \frac{\sum_{dtd\in E} size(lcs(C', dtd))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot size(C')}$$

□

Once we know that it is only necessary to compare the common *element*s, the next question is how we could improve the resemblance of a point in the search space. By theorem 5, we know that if adding a branch to the midpoint improves resemblance, all branches appearing the same number of times also improve it independently of their sizes. We may have thought that we have a set of possible improvements to investigate. Nevertheless, the branches with the same number of appearances do not generate alternative solutions, but all together belong to the same solution.

Lemma 5. *If adding a branch b to a concept increases its resemblance to the set, adding all branches appearing in the same number of DTDs than b will also improve its resemblance.*

Proof. Let's suppose that $C \sqsubset C'$ and $r(C, E) \geq r(C', E)$.

$$\frac{\sum_{dtd\in E} size(lcs(C, dtd))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot size(C)} \geq \frac{\sum_{dtd\in E} size(lcs(C', dtd))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot size(C')}$$

$$\frac{\sum_{dtd\in E}(size(lcs(C', dtd)) + (size(lcs(C, dtd)) - size(lcs(C', dtd))))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot (size(C') + (size(C) - size(C')))} \geq \frac{\sum_{dtd\in E} size(lcs(C', dtd))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot size(C')}$$

$$\frac{\sum_{dtd\in E} size(lcs(C', dtd)) + \sum_{dtd\in E}((size(lcs(C, dtd)) - size(lcs(C', dtd))))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot size(C') + \beta \cdot \mid E \mid \cdot ((size(C) - size(C')))} \geq \frac{\sum_{dtd\in E} size(lcs(C', dtd))}{\alpha \cdot \sum_{dtd\in E} size(dtd) + \beta \cdot \mid E \mid \cdot size(C')}$$

Which is true if and only if

$$\frac{\sum_{dtd\in E}((size(lcs(C,dtd))-size(lcs(C',dtd))))}{\beta\cdot\mid E\mid\cdot((size(C)-size(C')))}\geq\frac{\sum_{dtd\in E}size(lcs(C',dtd))}{\alpha\cdot\sum_{dtd\in E}size(dtd)+\beta\cdot\mid E\mid\cdot size(C')}$$

Since $C \sqsubset C'$ and it does not matter in which DTD the *element*s appear, but whether they appear or not, then $\frac{\sum_{dtd\in E}((size(lcs(C,dtd))-size(lcs(C',dtd))))}{\beta\cdot|E|\cdot((size(C)-size(C')))}$ can be seen as $\frac{\#appearance\cdot size(newElement)}{\beta\cdot|E|\cdot size(newElement)}$. Therefore, either adding an *element* or not does not depend on the size of the *element*, but on the number of times it appears in the DTDs. Thus, if adding an *element* is worthwhile, so it is adding any other *element* appearing the same number of times. □

Finally, in theorem 6, we show that *element*s appearing more times result in higher improvement of resemblance. As a special case of this, if an *element* improves resemblance, its parents improve resemblance even more. Thus, before adding an *element* to the result, all its parents should have been added (which otherwise could not have been avoided).

Corollary 6. *Independently of the size of the* element*s, a branch b_1 appearing k_1 times in E improves the resemblance more than another branch b_2 appearing k_2 times if $k_1 > k_2$.*

Proof. Since, $k_1 > k_2$, then $\frac{k_1}{\beta\cdot|E|} > \frac{k_2\cdot size(b_2)}{\beta\cdot|E|\cdot size(b_2)}$. Therefore, if b_2 improved the resemblance (i.e. we know that $\frac{k_2}{\beta\cdot|E|}\geq\frac{\sum_{dtd\in E}size(lcs(C,dtd))}{\alpha\cdot\sum_{dtd\in E}size(dtd)+\beta\cdot|E|\cdot size(C)}$), then b_1 improves the resemblance even more.

$$\frac{k_1}{\beta\cdot\mid E\mid}\geq\frac{\sum_{dtd\in E}size(lcs(C,dtd))+(k_2\cdot size(b_2))}{\alpha\cdot\sum_{dtd\in E}size(dtd)+\beta\cdot\mid E\mid\cdot size(C)+(\beta\cdot\mid E\mid\cdot size(b_2))}$$

□

$WDTD := \emptyset$;
foreach $dtd \in E$ **do**
  foreach $branch \sqsupseteq dtd$ **do**
    if $[branch, k] \in WDTD$
      then $WDTD := WDTD \setminus \{[branch, k]\} \cup \{[branch, k+1]\}$;
      else $WDTD := WDTD \cup \{[branch, 1]\}$;
      endif;
    endforeach;
  endforeach;
$M := \top$;
$m :=\mid E \mid$;
while $(\frac{m}{\beta\cdot|E|}\geq\frac{\sum_{dtd\in E}size(lcs(M,dtd))}{\alpha\cdot\sum_{dtd\in E}size(dtd)+\beta\cdot|E|\cdot size(M)})$
  foreach $branch \in getSubsetByWeight(WDTD, m)$ **do**
    $M := M \sqcap branch$;
    endforeach;
  $m := m - 1$;
  endwhile;

Figure 7: Algorithm based on appearance

From theorems 4, 5, and 6, we infer that we can build the midpoint of a set of DTDs from $\top$, by iteratively adding the most frequent *element* in the set of DTDs. Firstly, as we can see in figure 7, we build a weighted DTD (i.e. WDTD),

whose contents are $\bigsqcap_{dtd \in E} dtd$, where each piece of branch is weighted depending on its number of appearances in the set of DTDs. Once we have the weight of each branch, we take the maximum possible weight (i.e. $\mid E \mid$) and check if it would improve resemblance from $\top$ (i.e. *PCDATA*) to the set of DTDs. If this maximum weight improves the resemblance, we add all branches having such weight to the result and get the next weight smaller than that. We loop adding another subset of branches while their weight improves resemblance.

The first phase of the algorithm is really cheap in terms of complexity. Taking into account that the number of possible children of an *element* should be small, building the weighted tree is linear in the number of *element*s in the set of documents, because we can find a piece of branch in "WDTD" just searching the children of the previous piece of branch we modified/added to "WDTD" (assuming a deep first search of the document we are treating). Regarding the second phase of the algorithm, all calls to "getSubsetByWeight" can be done in linear time in the number of different *element*s, if we kept the *element*s with the same weight in a list. Therefore, the space we need is linear in the number of different *element*s (not counting repetitions), while the time is also linear in the number of *element*s in the set of documents (counting repetitions).

$$WDTD = \{[\exists a.\top, 4], [\exists a.\exists b.\top, 2], [\exists a.\exists b.\exists c.\top, 1], [\exists a.\exists d.\top, 4], [\exists a.\exists d.\exists e.\top, 3]\}$$

$$\begin{array}{ll} M_0 = \top & \frac{4}{4\beta} \geq 0 \\ M_1 = \exists a.\exists d.\top & \frac{3}{4\beta} \geq \frac{8}{14\alpha+2\cdot 4\beta} \\ M_2 = \exists a.\exists d.\exists e.\top & \frac{2}{4\beta} \geq \frac{11}{14\alpha+3\cdot 4\beta} \\ M_3 = \exists a.(\exists b.\top \sqcap \exists d.\exists e.\top) & \frac{1}{4\beta} < \frac{13}{14\alpha+4\cdot 4\beta} \end{array}$$

Figure 8: Candidate DTDs generated in a construction based on appearance

If we run this algorithm on the DTDs in figure 1, it would result in the "WDTD" in figure 8 (each dupla consists of a branch and the number of documents that contain it). Thus, in the first loop, condition evaluates true (for $\alpha = \beta = 1$, and every *element* contributing by one to the size), and we add the branches appearing four times. Since it still evaluates true, we add those appearing three times, and eventually twice. Since the condition evaluates false for weight equal one, the corresponding branch does not belong to the solution.

$$WDTD = \{[\exists a.\top, \{dtd_1, dtd_2, dtd_3, dtd_4\}], [\exists a.\exists b.\top, \{dtd_1, dtd_2\}], [\exists a.\exists b.\exists c.\top, \{dtd_1\}], [\exists a.\exists d.\top, \{dtd_1, dtd_2, dtd_3, dtd_4\}], [\exists a.\exists d.\exists e.\top, \{dtd_1, dtd_3, dtd_4\}]\}$$

$$M = lcs(dtd_1, dtd_2) \sqcap lcs(dtd_1, dtd_3, dtd_4)$$

Figure 9: Obtaining the sets of documents that generate the midpoint

Obtaining the sets of documents that generate the midpoint (see theorem 1) a posteriori (once we know the midpoint) is easy with a small modification of the algorithm. All we need is that "WDTD" keep the identifiers of the documents that contain every branch instead of just a counter of them. Thus, it is trivial to see that the conjunction of the LCS of the documents containing the leafs of the midpoint result in the midpoint. Figure 9 shows how this would result in our example.

6 Generalizations

Along this paper, we have simplified the class of DTDs we treated. This section is devoted to study the two ignored cases, i.e. that of optional *element*s (section 6.1) and repeated *element*s (section 6.2).

6.1 Optional *element*s

Until now, we assumed that we do not had the *choice* XML construct (i.e. disjunction "$\sqcup$" in terms of DL) . In this section, we will study the possibility of using it. Thus, resemblance needs to be redefined as follows:

$$M = \bigsqcup_{i=1}^{k} M^i$$

$$E^i = \{dtd \in E \mid \forall j \neq i, r(dtd, M^i) > r(dtd, M^j)\}$$

$$r(M, E) = \frac{\sum_{i=1}^{k} w_c(M^i, E^i)}{\sum_{i=1}^{k} w_c(M^i, E^i) + \alpha \sum_{i=1}^{k} w_p(M^i, E^i) + \beta \sum_{i=1}^{k} w_m(M^i, E^i)}$$

As explained in [NAM98], the size of a perfect typing (which is always possible if using *choice*) may be quite large (roughly the order of the data set), which would prohibit its use for query optimization and render it impractical for graphical query interfaces. Therefore, we want to improve the resemblance by allowing only a limited number of optional *element*s, limiting the size of the schema and avoiding its overspecialization. Sections 6.1.1 and 6.1.2 show, respectively, how w_p and w_m can be reduced by adding optional *element*s.

6.1.1 Reducing plus *element*s (those in the documents, not in the midpoint)

We want to improve the resemblance to the whole set of documents by adding *element*s to the midpoint. Nevertheless, since we already reached the maximum resemblance, we could only worsen it. To solve this, we may consider those *element*s as optional. Thus, they will improve the resemblance to those documents containing them, while never worsening the resemblance to those that do not contain them.

It is easy to see that by just extending (strictly adding *element*s) the midpoint with optional *element*s, we will increase w_c, reduce w_p, and preserve w_m. Since we already showed that $w_c + w_p$ is constant, it is only necessary to consider how much w_c increases, i.e. how many documents match the optional *element*s and how big these are. The more documents matching that *element*s, the better; and the bigger the *element*s, also the better. Therefore, optional *element*s should be big and present in a set of documents of high cardinality.

```
...
branch = nextByWeightTimesSize(WDTD, m));
while (r(M, E) < target and branch ≠ ⊥)
    M := M ⊔ (M ⊓ branch);
    branch = nextByWeightTimesSize(WDTD, m));
    endwhile;
```

Figure 10: Algorithm for the selection of optional *element*s

Figure 10 shows the third phase of the algorithm in figure 7. Once we got the DTD of maximum resemblance, without any optional *element*, we may add optional *element*s also based on the number of appearance until we get the target resemblance ($target = 1$ would result in overspecialization, while $target = 0$ would result in the absence of optional *element*s). Priorizing small or big *element*s would depend on the sorting criterion used to get the branches. There are several possibilities to choose the best *element* to be optional among those of high appearance. We may want the smallest one first (to get several small optional *element*s) or the biggest one first (to get few big optional parts

$$WDTD = \{[\exists a.\top, 4], [\exists a.\exists b.\top, 2], [\exists a.\exists b.\exists c.\top, 1], [\exists a.\exists d.\top, 4], [\exists a.\exists d.\exists e.\top, 3]\}$$
$$M_4 = \exists a.(\exists b.\top \sqcap \exists d.\exists e.\top) \sqcup (\exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top))$$
$$E^1 = \{dtd_2, dtd_3, dtd_4\}; E^2 = \{dtd_1\}$$
$$r(M_4, E) = \frac{w_c(M_4^1,E^1)+w_c(M_4^2,E^2)}{w_c(M_4^1,E^1)+w_c(M_4^1,E^2)+\alpha\cdot(w_p(M_4^1,E^1)+w_p(M_4^1,E^2))+\beta\cdot(w_m(M_4^1,E^1)+w_m(M_4^1,E^2))} = \frac{9+5}{9+5+\alpha\cdot(0+0)+\beta\cdot(3+0)} = \frac{14}{17}$$

Figure 11: Improvement reducing plus *element*s

of the document). In this case, we decided to pick *element*s up sorted by "$\#appearance \cdot size(element)$", because it increases resemblance faster (minimizing the number of *element*s in the midpoint).

What really matters is that only branches with less than m appearance are returned (those of higher appearance already belong to the midpoint).

Figure 11 shows how we can improve the resemblance. In figure 8, we stopped the iteration before adding those branches appearing only one time. Therefore, $m = 1$ and the next branch to be added is "$\exists a.\exists b.\exists c.\top$". Thus, we get a new midpoint being the disjunction of two parts, and we calculate the resemblance taking into account the best part for each document. In this way, denominator does not change, while numerator increases, improving resemblance.

6.1.2 Reducing minus *element*s (those in the midpoint, not in the documents)

Notice that, by using the algorithm in figure 10 we do not modify w_m (which may not be zero after the second phase). Thus, in general, we will not reach the maximum resemblance, because the midpoint contains non-optional *element*s that are not present in all documents. The more heterogeneous the documents are, the worse will be the result, if we try to obtain only one midpoint. In order to solve this, we should divide the documents into several classes, and obtain separately the midpoint of each of these classes. In this way, at the extreme, each of these midpoints would only contain *element*s that are present in all its corresponding documents. The midpoint of the whole set of documents will be the disjunction of these partial midpoints M^i.

Therefore, we should detect the presence of different clusters of documents, and treat them separately. We may trigger this phase of the algorithm, if after the third phase we did not reach the target resemblance; we may also trigger it, if we cannot reach the target resemblance after a given number of iterations of phase three (i.e. a given number of optional *element*s); or if $\frac{w_p}{w_m}$ after phase two is below a threshold; or we may even trigger it based on the number of appearances of *element*s at first level (i.e. if $\frac{|E|}{getMaxWeight(WDTD)}$ is above a threshold).

For this clustering, we may use an algorithm like "k-means" which is considered to need linear time (see [ECY00]). If we take $k =\mid E \mid$, the problem becomes trivial taking $M = \bigsqcup_{dtd \in E} dtd$. Therefore, we are looking for a small k so that maximizes $r(M, E)$. For example, we could assume that there should be, at least, $\frac{|E|}{getMaxWeight(WDTD)}$ different kinds of documents, and generate such number of optional blocks of *element*s.

Figure 12 sketches the algorithm. In our case, we could benefit from the existence of "WDTD" to improve performance, if it keeps the sets of documents that contain every branch instead of just counting them. We can codify every set E^i as a list of bits $b_1 b_2..b_{|E|}$, where bit j shows whether the corresponding DTD contains the branch or not ($dtd_j \sqsubseteq branch$). In this way, we could take k random disjoint chains of bits as seeds (s^i) for "k-means". Then, we can find the midpoint corresponding to each seed by running the second phase of our algorithm on "WDTD" using the corresponding mask of bits. Finally, we could also get the resemblance from each DTD to one midpoint also going through "WDTD" one

Choose k initial seeds (may be randomly)
do
    Assign each document to its nearest center
    For each cluster get a new center
    while (centers changed)

Figure 12: K-means algorithm

more time (by keeping all $\mid E \mid$ resemblances in memory).

Figure 13 shows an example for clustering documents into two sets, and how this improves the resemblance. As initial seeds, we take odd and even documents (i.e. $\{dtd_1, dtd_3\}$ vs $\{dtd_2, dtd_4\}$). For each one of them, we obtain the midpoint (i.e. "$\exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top)$" and "$\exists a.(\exists b.\top \sqcap \exists d.\exists e.\top)$" respectively) by applying the algorithm in figure 7, anding the corresponding seed to the sequence of bits of each branch in "WDTD". Obtaining the resemblance of each document to both midpoints, we see that "dtd_3" is in the wrong class, because its resemblance to the midpoint of even documents is bigger, while it is an odd one. Therefore, we perform a second iteration with one cluster for "dtd_1" and another one for the rest of the documents. This time, we detect that "dtd_2" is in the wrong class, we move it and perform the third iteration. Now every document is in the right class, so we have finished. The midpoint of E is the disjunction of both midpoints, and overall resemblance improves by reducing the denominator. It is easy to see that with three clusters we had obtained the exact DTD.

Notice also that all resemblances can be obtained from WDTD, by crossing only once it in parallel with the corresponding midpoint. For example, let's see how to obtain in the first iteration of figure 13 the resemblances from "$\exists a.(\exists b.\top \sqcap \exists d.\exists e.\top)$" to each document (i.e. "$r(M^2, dtd_1)$", "$r(M^2, dtd_2)$", "$r(M^2, dtd_3)$", and "$r(M^2, dtd_4)$"). At the first step, we would take "$\exists a.\top$" that belongs to the midpoint. Since the sequence of bits indicates that it belongs to the four documents, it would increase all four common *element*s counters (i.e. $w_c(M^2, dtd_1)$, $w_c(M^2, dtd_2)$, $w_c(M^2, dtd_3)$, and $w_c(M^2, dtd_4)$). At the second step, we would take "$\exists a.\exists b.\top$" that also belong to the midpoint. Since the sequence of bits indicates that it belongs to the first two documents, it would increment common *element*s of these and minus of the others (i.e. $w_c(M^2, dtd_1)$, $w_c(M^2, dtd_2)$, $w_m(M^2, dtd_3)$, and $w_m(M^2, dtd_4)$). At the third step, we would take "$\exists a.\exists b.\exists c.\top$" that does not belong to the midpoint. Since the sequence of bits indicates that it belongs only to the first document, it would just increment plus *element*s of this (i.e. $w_p(M^2, dtd_1)$). We would follow this way for the other two *element*s in "WDTD".

6.2 Repeated *element*s

It is possible that the same *element* appears at different places (i.e. different levels, or just having a different parent) in the same document (or even in different documents). If we consider that in this case all *element*s share the same DTD internal structure independently of their position in the document, we should start a previous process to find the midpoint of such *element* (i.e. we should consider it a whole document, get its internal structure, and treat it as a black box). This section does not deal with this kind of repetitions, but with one *element* that contains several others of the same kind in a *sequence*.

First of all, on talking about repetitions, it is important to distinguish between unnumbered repetitions (i.e. $+$ in XML notation) or numbered repetitions (i.e. a fixed number of children of the same kind). The point is that we cannot decide that there exists an unnumbered repetition without human participation. How could we decide (based

$$WDTD = \{[\exists a.\top, \{1111\}], [\exists a.\exists b.\top, \{1100\}], [\exists a.\exists b.\exists c.\top, \{1000\}], [\exists a.\exists d.\top, \{1111\}], [\exists a.\exists d.\exists e.\top, \{1011\}]\}$$

	$s^1 = 1010, s^2 = 0101$
	$WDTD^1 = \{[\exists a.\top, \{1010\}], [\exists a.\exists b.\top, \{1000\}], [\exists a.\exists b.\exists c.\top, \{1000\}], [\exists a.\exists d.\top, \{1010\}], [\exists a.\exists d.\exists e.\top, \{1010\}]\}$
	$M_0^1 = \top \quad \frac{2}{2\beta} \geq 0$
	$M_1^1 = \exists a.\exists d.\exists e.\top \quad \frac{1}{2\beta} \geq \frac{6}{8\alpha+3\cdot 2\beta}$
	$M_2^1 = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top) \quad \frac{0}{2\beta} < \frac{8}{8\alpha+5\cdot 2\beta}$
	$WDTD^2 = \{[\exists a.\top, \{0101\}], [\exists a.\exists b.\top, \{0100\}], [\exists a.\exists b.\exists c.\top, \{0000\}], [\exists a.\exists d.\top, \{0101\}], [\exists a.\exists d.\exists e.\top, \{0001\}]\}$
Iteration 1:	$M_0^2 = \top \quad \frac{2}{2\beta} \geq 0$
	$M_1^2 = \exists a.\exists d.\top \quad \frac{1}{2\beta} \geq \frac{4}{6\alpha+2\cdot 2\beta}$
	$M_2^2 = \exists a.(\exists b.\top \sqcap \exists d.\exists e.\top) \quad \frac{0}{2\beta} < \frac{6}{6\alpha+4\cdot 2\beta}$
	$r(M^1, dtd_1) = \frac{5}{5+0\alpha+0\beta} \quad r(M^2, dtd_1) = \frac{4}{4+1\alpha+0\beta}$
	$r(M^1, dtd_2) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_2) = \frac{3}{3+0\alpha+1\beta}$
	$r(M^1, dtd_3) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_3) = \frac{3}{3+0\alpha+1\beta}$
	$r(M^1, dtd_4) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_4) = \frac{3}{3+0\alpha+1\beta}$
	$s^1 = 1000, s^2 = 0111$
	$WDTD^1 = \{[\exists a.\top, \{1000\}], [\exists a.\exists b.\top, \{1000\}], [\exists a.\exists b.\exists c.\top, \{1000\}], [\exists a.\exists d.\top, \{1000\}], [\exists a.\exists d.\exists e.\top, \{1000\}]\}$
	$M_0^1 = \top \quad \frac{1}{1\beta} \geq 0$
	$M_1^1 = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top) \quad \frac{0}{1\beta} < \frac{5}{5\alpha+5\cdot 1\beta}$
	$WDTD^2 = \{[\exists a.\top, \{0111\}], [\exists a.\exists b.\top, \{0100\}], [\exists a.\exists b.\exists c.\top, \{0000\}], [\exists a.\exists d.\top, \{0111\}], [\exists a.\exists d.\exists e.\top, \{0011\}]\}$
Iteration 2:	$M_0^2 = \top \quad \frac{3}{3\beta} \geq 0$
	$M_1^2 = \exists a.\exists d.\top \quad \frac{2}{3\beta} \geq \frac{6}{9\alpha+2\cdot 3\beta}$
	$M_2^2 = \exists a.\exists d.\exists e.\top \quad \frac{1}{3\beta} < \frac{8}{9\alpha+3\cdot 3\beta}$
	$r(M^1, dtd_1) = \frac{5}{5+0\alpha+0\beta} \quad r(M^2, dtd_1) = \frac{3}{3+2\alpha+0\beta}$
	$r(M^1, dtd_2) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_2) = \frac{2}{2+1\alpha+1\beta}$
	$r(M^1, dtd_3) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_3) = \frac{3}{3+0\alpha+0\beta}$
	$r(M^1, dtd_4) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_4) = \frac{3}{3+0\alpha+0\beta}$
	$s^1 = 1100, s^2 = 0011$
	$WDTD^1 = \{[\exists a.\top, \{1100\}], [\exists a.\exists b.\top, \{1100\}], [\exists a.\exists b.\exists c.\top, \{1000\}], [\exists a.\exists d.\top, \{1100\}], [\exists a.\exists d.\exists e.\top, \{1000\}]\}$
	$M_0^1 = \top \quad \frac{2}{2\beta} \geq 0$
	$M_1^1 = \exists a.(\exists b.\top \sqcap \exists d.\top) \quad \frac{1}{2\beta} \geq \frac{6}{8\alpha+3\cdot 2\beta}$
	$M_2^1 = \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top) \quad \frac{0}{2\beta} < \frac{8}{8\alpha+5\cdot 2\beta}$
	$WDTD^2 = \{[\exists a.\top, \{0011\}], [\exists a.\exists b.\top, \{0000\}], [\exists a.\exists b.\exists c.\top, \{0000\}], [\exists a.\exists d.\top, \{0011\}], [\exists a.\exists d.\exists e.\top, \{0011\}]\}$
Iteration 3:	$M_0^2 = \top \quad \frac{2}{2\beta} \geq 0$
	$M_1^2 = \exists a.\exists d.\exists e.\top \quad \frac{1}{2\beta} \geq \frac{6}{6\alpha+3\cdot 2\beta}$
	$M_2^2 = \exists a.\exists d.\exists e.\top \quad \frac{0}{2\beta} < \frac{6}{6\alpha+3\cdot 2\beta}$
	$r(M^1, dtd_1) = \frac{5}{5+0\alpha+0\beta} \quad r(M^2, dtd_1) = \frac{3}{3+2\alpha+0\beta}$
	$r(M^1, dtd_2) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_2) = \frac{2}{2+1\alpha+1\beta}$
	$r(M^1, dtd_3) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_3) = \frac{3}{3+0\alpha+0\beta}$
	$r(M^1, dtd_4) = \frac{3}{3+0\alpha+2\beta} \quad r(M^2, dtd_4) = \frac{3}{3+0\alpha+0\beta}$

$$M = (\exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top)) \sqcup (\exists a.\exists d.\exists e.\top)$$

$$E^1 = \{dtd_1, dtd_2\}; E^2 = \{dtd_3, dtd_4\}$$

$$r(M, E) = \frac{w_c(M^1, E^1) + w_c(M^2, E^2)}{w_c(M^1, E^1) + w_c(M^2, E^2) + \alpha \cdot (w_p(M^1, E^1) + w_p(M^2, E^2)) + \beta \cdot (w_m(M^1, E^1) + w_m(M^2, E^2))} = \frac{8+6}{8+6+\alpha\cdot(0+0)+\beta\cdot(2+0)} = \frac{14}{16}$$

Figure 13: Clustering the documents

on a finite set of finite documents) that there is a possibly infinite repetition of *element*s? We cannot. Therefore, we should decide a priori if we are interested on generating numbered or unnumbered repetitions.

If we want to generate numbered repetitions, we should just consider that each sibling *element* is a completely different one and we can use again the same algorithms. For example, in order to be able to treat the XML document "<a>brothersister", we should translate it into $\exists a.(\exists b1.\top \sqcap \exists b2.\top)$. This would work specially well for ordered *element*s, because their position indicates which sibling they are. In the example, the first one would always be identified as $b1$, and the second as $b2$. Doing it this way, a different position indicates different internal structure.

If we are treating unordered documents, repeated *element*s result in undistinguishable twins with the same intensional internal structure. Otherwise, if they have a different schema, there is a design problem in the document. Even when dealing with semi-structured data, two *element*s of the same (undistinguishable) class should share the same (semi-structured) schema. Thus, we have two different problems. The first one is deciding when there exists a unnumbered repetition, and the second is how to find the internal structure of the repeated *element*s.

To be able to handle repeated elements, we should modify the parser of XML documents that generates the DTDs. If the parser finds a repetition it should use a special mark showing the presence of sibling *element*s, and indicating the number t of them existing in the corresponding parent. Since XML documents are deep-first written, we would visit the children before we know the number of twins. Therefore, we should keep in memory the branches of each DTD, to generate t.

In order to decide whether there is a repetition or not, if the first phase of the algorithm in figure 7 finds the mark, "branch" and "branch+" should both be increased in "WDTD". During the second phase, on adding "branch+", we should remove "branch" from M. Notice that the appearance of "branch" will always be higher than that of "branch+", because we always increase the counter of the first, while only increase the latter if there is a repetition.

Regarding the problem of finding the internal structure of repeated *element*s, we treat the subelements in only one pass. When, during the first phase of the algorithm, we find a subelement in any of the repetitions, we should just increase the counter of the corresponding branch in $\frac{1}{t}$, where t is the number of twins. By doing this, we avoid overweighting the subelements of repetitions, and keep the basic idea of the algorithm still true (i.e. a child cannot appear more times in the documents that its parent).

document 5: <a>Single
document 6: <a>Twin2 $Twin1$
document 7: <a>PlainTwin<c>ComposedTwin</c>

$$WDTD = \{[\exists a.\top, 3], [\exists a.\exists .b.\top, 3], [\exists a.\exists b + .\top, 2], [\exists a.\exists b.\exists c.\top, 0.5]\}$$

Figure 14: Example of documents with repetitions

Figure 14 exemplifies how repetitions should be treated by the algorithm. In this case, "$\exists a.\top$" weights three, because appears in three documents. The same happens for "$\exists a.\exists .b.\top$", because three documents contain such branch. Moreover, there are two documents containing repetitions of "b", which is recorded by the appearance of "$\exists a.\exists b+$". Finally, "$\exists a.\exists b.\exists c.\top$" appears once in one document. Nevertheless, it is part of a repetition of two twins. Therefore, its weight is $\frac{1}{2} = 0.5$.

7 Conclusions and future work

Along this paper, we have studied the possibility of approximating the schema (DTD) of a set of XML documents. Based on a given measure of resemblance, we are able to find the midpoint of the set. This midpoint have been characterized in terms of conjunction of Least Common Subsumers of the documents. Moreover different algorithms (of different costs) have also been presented to obtain it. We begun by considering only a restricted class of DTDs (without repetition nor *choice*), and it has been generalized to any DTD. Thus, we are able to approximate the schema as much as we want to the set of documents (eventually until we get an exact matching) in linear time.

In order to get integrated access to several XML sources, first of all we should guarantee that a given element uses the same tag name everywhere, i.e. they share the same vocabulary. Therefore, as future work, we plan to deal with the problem of matching tag names, where ontologies can be used. The presence of optional elements in the schema may lead to the identification of equivalent tags from different sources.

Acknowledgements

Our work has been partially supported by the Spanish Research Program PRONTIC and FEDER under project TIC2002-00744.

References

[ABS00] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web - From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

[AGW01] Jürgen Albert, Dora Giammarresi, and Derick Wood. Normal Form algorithms for extended Context-Free Grammars. *Theoretical Computer Science*, 267(1-2):35–47, 2001.

[BB95] Vladimir Batagelj and Matevz Bren. Comparing resemblance measures. *Journal of Classification*, 12(1):73–90, 1995.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[BdR04] Utsav Boobna and Michel de Rougemont. Correctors for XML Data. In *Proceedings of 2nd International XML Database Symposium (XSYM'04)*, volume 3186 of *LNCS*, pages 97–111. Springer, 2004.

[BGM04] Elisa Bertino, Giovanna Guerrini, and Marco Mesiti. A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications. *Information Systems*, 29(1):23–46, March 2004.

[CDL99] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Representing and Reasoning on XML Documents: A decription Logic Approach. *Journal of Logic and Computation*, 9(3):295–318, 1999.

[CLN98] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. *Logics for Databases and Information Systems*, chapter Description Logics for Conceptual Data Modeling, pages 229–264. Kluwer, 1998.

[ECY00] Vladimir Estivill-Castro and Jianhua Yang. Fast and Robust General Purpose Clustering Algorithms. In *Proceedings of 6th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)*, volume 1886 of *LNCS*, pages 208–218. Springer, 2000.

[EL02] Robert Engels and Till Christopher Lech. Generating ontologies for the semantic web: OntoBuilder. In J. Davies, D. Fensel, and F. van Harmelen, editors, *Towards the semantic web: ontology-driven knowledge management*, chapter 6, pages 91–115. John Wiley and Sons, Ltd., 2002.

[JOKA02] Jong-Seok Jung, Dong-Ik Oh, Yong-Hae Kong, and Jong-Keun Ahn. Extracting Information from XML Documents by Reverse Generating a DTD. In *Proceedings of the (EurAsia-ICT 2002)*, volume 2510 of *LNCS*, pages 314–321. Springer, 2002.

[NAM98] Svetlozar Nestorov, Serge Abiteboul, and Rajeev Motwani. Extracting schema from semistructured data. In *Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD 1998)*, pages 295–306. ACM, 1998.

[SPBA03] Ismael Sanz, Juan Manuel Pérez, Rafael Berlanga, and María José Aramburu. XML Schemata Inference and Evolution. In *Proceedings of 14th International Conference on Database and Expert Systems Applications (DEXA'03)*, volume 2736 of *LNCS*, pages 109–118. Springer, 2003.

[W3C04] W3C. *Extensible Markup Language (XML) 1.0*, third edition, February 2004.

[Wid99] Jennifer Widom. Data Management for XML: Research Directions. *IEEE Data Engineering Bulletin*, 22(3):44–52, 1999.

[ZS89] Zaizhong Zhang and Dennis Shasha. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.