On the midpoint of a set of XML documents

Alberto Abelló[1], Xavier de Palol[1], and Mohand-Saïd Hacid[2]

[1] Dept. de Llenguatges i Sistemes Informàtics, U. Politècnica de Catalunya
[2] LIRIS- UFR d'Informatique, U. Claude Bernard Lyon 1

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, which 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. A exact schema could 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. For scalability reasons, one important aspect consists in distilling those 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 has 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 and encourage queries. It is key to improve the access methods to the data, thus availing query optimization and data interchange among companies.

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, in both cases the *element* 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.

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. 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]. Such approximated schemas are called inexact schemas in [Wid99].

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 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 Description Logics that we propose. Section 4 characterizes the midpoint. Section 5 shows an efficient algorithm to obtain the midpoint. Finally, section 6 gives the general conclusions and points out our future work.

2 Related work

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.

[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 structure similarity. [NAM98] uses Manhattan distance (i.e. the number of different descendants/ancestors of two *element*s). [BB95] shows different more elaborate resemblance measures. Among those, [SPBA03] uses $\frac{|elem(d_1) \cap elem(d_2)|}{max(|elem(d_1)|, |elem(d_2)|)}$, while in [BGM04] $\frac{|elem(d_1) \cap elem(d_2)|}{|elem(d_1) \cap elem(d_2)| + \alpha \cdot |elem(d_1) \setminus elem(d_2)| + \beta \cdot |elem(d_2) \setminus elem(d_1)|}$ is used. We took this last measure, because it is more general, and allows to distinguish lack of *element*s in one side or another.

3 Formalizing XML documents by means of DL

As we can see in [ABS00], an XML document uses to be thought as a rooted tree. A rooted tree is an acyclic graph $(\mathcal{N}, \mathcal{E})$, that has no more than one root. $\mathcal{N}$ is a set of nodes and $\mathcal{E}$ 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 $\mathcal{E}$. Along this paper we will use Description Logics (DL) notation to formalize those trees.

Since we only take into account *element* tags (not contents), we are not actually interested in XML documents, but in a restricted class of DTDs that can be automatically generated from one XML document. We assume that we have a pseudo-*DTD* exactly matching each document. These are obtained just parsing documents and eliminating data (leaving *element* tags). Thus, a pseudo-*DTD* does not contain *choice*, nor *unnumbered repetitions*, nor *optional elements*, nor *any*. The problem tackled in this paper is that of finding a true-*DTD* from a set of pseudo-*DTD*s. From here on, we will use the term *DTD* for the pseudo-*DTD*s, and "midpoint" for the true-*DTD*.

Regarding XML attributes, they could be used to match different *element* tags. For example, "<a ID='Id1'>" could be identified with "<b ID='Id1'>" 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 decision. 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 pro-

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

$$\begin{aligned} dtd_1 &= \exists a.(\exists b.\exists c.\top \sqcap \exists d.\exists e.\top) \\ dtd_2 &= \exists a.(\exists b.\bot \sqcap \exists d.\bot) \\ dtd_3 &= \exists a.\exists d.\exists e.\top \\ dtd_4 &= \exists a.\exists d.\exists e.\top \end{aligned}$$

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

Fig. 1. DL representation of an XML document

cessed 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 for us.

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 as follows (C and D are concepts, and r is a role):

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

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. Our formalization allows the usage of DL algorithms like "Subsumption" and "Least Common Subsumer":

Subsumption (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, weighting 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 the size of common, plus, and minus *element*s.

$$\begin{aligned} w_c(C,E) &= \textstyle\sum_{dtd \in E} size(lcs(C, dtd)) \\ w_p(C,E) &= \textstyle\sum_{dtd \in E} (size(dtd) - size(lcs(C, dtd))) \\ w_m(C,E) &= \textstyle\sum_{dtd \in E} (size(C) - size(lcs(C, dtd))) \end{aligned}$$

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].

$$\begin{aligned} r(\exists a.\exists d.\top, \{dtd_2, dtd_3\}) &= \tfrac{2+2}{(2+2)+\alpha\cdot(1+1)+\beta\cdot(0+0)} = \tfrac{4}{4+4} \\ r(\exists a.\exists d.\exists e.\top, \{dtd_2, dtd_3\}) &= \tfrac{2+3}{(2+3)+\alpha\cdot(1+0)+\beta\cdot(1+0)} = \tfrac{5}{5+2+3} \\ r(\exists a.(\exists b.\top \sqcap \exists d.\top), \{dtd_2, dtd_3\}) &= \tfrac{3+2}{(3+2)+\alpha\cdot(0+1)+\beta\cdot(0+1)} = \tfrac{5}{5+2+3} \\ r(\exists a.(\exists b.\top \sqcap \exists d.\exists e.\top), \{dtd_2, dtd_3\}) &= \tfrac{3+3}{(3+3)+\alpha\cdot(0+0)+\beta\cdot(1+1)} = \tfrac{6}{6+6} \end{aligned}$$

Fig. 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, as we can see in figure 2, four different DTDs result in the same resemblance to $\{dtd_2, dtd_3\}$. Since this is the maximum resemblance, we can choose the midpoint of $\{dtd_2, dtd_3\}$ among those four DTDs. Theorem 1 states that one of the possible midpoints of the set can be obtained by a conjunction of LCS of the documents. Due to lack of space, proofs have been omited.

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 $l_i \geq 1$*

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

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

Corollary 1. *There exists a DTD of the form $\prod_{k=1..p} lcs(S_k)$ maximizing the resemblance, so that $p \leq \binom{|E|}{\frac{|E|}{2}}$*

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 *elements* in the midpoint wrt every individual DTD. Therefore, $\prod_{dtd \in E} dtd$ 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 $\alpha \neq 0$ and $\beta \neq 0$.

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 2, 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 2 *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).*

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

Table 1. Trivial cases on finding a midpoint

Proof. Let be $r(C,E) \geq r(C',E)$ ("s" stands for "size" if necessary).

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

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

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

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

$$\frac{\sum_{d \in E} size(lcs(C,d))}{\alpha \cdot \sum_{d \in E} size(d) + \beta \cdot |E| \cdot size(C)} \geq \frac{\sum_{d \in E} size(lcs(C',d))}{\alpha \cdot \sum_{d \in E} size(d) + \beta \cdot |E| \cdot size(C')} \qquad \square$$

Once we know that it is only necessary to compare the common *element*s, the next question is how we could improve the resemblance. By lemma 2, 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 check. Nevertheless, the branches with the same number of appearances do not generate alternative solutions, but all together belong to the same solution.

Lemma 2. *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 be $C \sqsubset C'$ and $r(C,E) \geq r(C',E)$.

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

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

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

Which is true if and only if

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

Since $C \sqsubset C'$ and it does not matter in which DTD the *element*s appear, but whether they appear or not, then $\frac{\sum_{d \in E}((size(lcs(C,d)) - size(lcs(C',d))))}{\beta \cdot |E| \cdot ((size(C) - size(C')))}$ 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. $\square$

Finally, in corollary 2, 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 2. *Independently of its size, a branch b_1 appearing k_1 times in E improves the resemblance more than another 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 $\frac{k_2}{\beta\cdot|E|} \geq \frac{\sum_{d\in E} size(lcs(C,d))}{\alpha\cdot\sum_{d\in E} size(d)+\beta\cdot|E|\cdot size(C)}$), then b_1 improves it even more: $\frac{k_1}{\beta\cdot|E|} \geq \frac{\sum_{d\in E} size(lcs(C,d))+(k_2\cdot size(b_2))}{\alpha\cdot\sum_{d\in E} size(d)+\beta\cdot|E|\cdot size(C)+(\beta\cdot|E|\cdot size(b_2))}$ □

```
$WDTD := \emptyset$;
**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**;
$M := \top$; $m :=| E |$;
**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**;
```

Fig. 3. Algorithm based on appearance

From these theorems, 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 3, 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. $| E |$) 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).

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

Fig. 4. Candidate DTDs generated during the execution

If we run this algorithm on the DTDs in figure 1, it would result in the "WDTD" in figure 4 (each 2-upla 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.

$$\begin{aligned} WDTD = \{&[\exists a.\top, \{dtd_1, dtd_2, dtd_3, dtd_4\}], [\exists a.\exists b.\top, \{dtd_1, dtd_2\}], [\exists a.\exists b.\exists c.\top, \{dtd_1\}], \\ &[\exists a.\exists d.\top, \{dtd_1, dtd_2, dtd_3, dtd_4\}], [\exists a.\exists d.\exists e.\top, \{dtd_1, dtd_3, dtd_4\}]\} \\ M = {}& lcs(dtd_1, dtd_2) \sqcap lcs(dtd_1, dtd_3, dtd_4) = \exists a.(\exists b.\top \sqcap \exists d.\exists e.\top) \end{aligned}$$

Fig. 5. 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 set 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 5 shows how it results in our example.

6 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 one midpoint of the set. This midpoint has been characterized in terms of conjunction of Least Common Subsumers of the documents. Moreover an efficient algorithm has also been presented to obtain it. The obtained resemblance may be improved by considering optional *element*s (eventually reaching the perfect typing).

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] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web - From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

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

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

[BCM$^+$03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

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

[BGM04] E. Bertino, G. Guerrini, and M. 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.

[JOKA02] J-S. Jung, D-I. Oh, Y-H. Kong, and J-K. Ahn. Extracting Information from XML Documents by Reverse Generating a DTD. In *Proc. of the EurAsia-ICT 2002*, volume 2510 of *LNCS*, pages 314–321. Springer, 2002.

[NAM98] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1998)*, pages 295–306. ACM, 1998.

[SPBA03] I. Sanz, J. M. Pérez, R. Berlanga, and M. J. Aramburu. XML Schemata Inference and Evolution. In *Proc. of 14th Int. Conf. on Databases and Expert Systems Applications (DEXA'03)*, volume 2736 of *LNCS*, pages 109–118. Springer, 2003.

[W3C04] W3C. *Extensible Markup Language (XML) 1.0*, 3rd edition, February 2004.

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

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