Learning XML Grammars

Henning Fernau

WSI-2001-1

Henning Fernau
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13
D-72076 Tübingen
Germany

email:fernau@informatik.uni-tuebingen.de
Telefon: (07071) 29-77565
Telefax: (07071) 29-5061

ISSN 0946-3852

Learning XML Grammars

Henning Fernau
Wilhelm-Schickard-Institut für Informatik,
Universität Tübingen,
Sand 13, D-72076 Tübingen,
Germany
email: fernau@informatik.uni-tuebingen.de

January 18, 2001

Abstract

We sketch possible applications of grammatical inference techniques to problems arising in the context of XML. The idea is to infer document type definitions (DTDs) of XML documents in situations when either the original DTD is missing or should be (re)designed or should be restricted to a more user-oriented view on a subset of the (given) DTD. The usefulness of such an approach is underlined by the importance of knowing appropriate DTDs; this knowledge can be exploited, e.g., for optimizing database queries based on XML.

1 Introduction

XML. The expectations surrounding XML (eXtendible Markup Language) as a universal syntax for data representation and exchange on the world wide web continues to grow. This is underlined by the amount of effort being committed to XML by the World Wide Web Consortium (W3C) (see `www.w3.org/TR/REC-XML`), by the huge number of academics involved in the research of the backgrounds of XML, as well as by numerous private companies. Moreover, an ever-growing number applications arise which make use of XML, although they are not directly related to the world wide web. For example, XML plays nowadays an important role in the integration of manufacturing and management in highly automated fabrication processes as in car companies [12]. Further information on XML could be found under `www.oasis-open.org/cover/xmlIntro.html`.

XML grammars. The syntactic part of the XML language describes the relative position of pairs of corresponding *tags*. This description is done by means of a *document type definition* (DTD). Ignoring attributes of tags, a DTD is a special form of a context-free grammar. This sort of grammars has been formalized and studied by Berstel and Boasson [7] as *XML grammars*.[1]

[1] Also Behrens and Buntrock [6] investigated formal language properties of DTDs.

Grammatical inference. Our paper can also be seen as a contribution to further promote the use of machine learning techniques within database technologies, in particular, when these are based on the XML framework. More specifically, we discuss learnability issues for XML grammars. This question is interesting for several reasons:

Three applications of grammatical inference. As already worked out by Ahonen, grammatical inference (GI) techniques can be very useful for automatic document processing, see [2, 3]. More specifically, Ahonen detailed on the following two applications of the inference of DTDs (of HTML documents) [1, 2]:

Firstly, GI techniques can be used to assist designing grammars for (semi-) structured documents. This is often desirable, since either the system users are not experts in grammar design or the grammars are rather huge and difficult to handle. The user feds several examples of syntactically correct tagged documents into the GI system, which then suggests a grammar describing these documents. In this application, an interaction between the human grammar designer and the GI system is desirable, e.g., for coping with erroneous examples, or when previous grammar design decisions are modified. If the given examples are not originally tagged (e.g., if they do not stem from an XML document), document recognition techniques can be applied in a first step, see [23, 31]. Fankhauser and Xu integrate both steps in their system [14].

Secondly, GI may be of help to create views and subdocuments. For several applications, standard DTDs have been proposed. However, these DTDs are usually large and designed to cover many different needs. GI may be used to find reasonable smaller subsets of the corresponding document descriptions.

Note that Ahonen used a rather direct approach to the inference of DTDs, by simply inferring right-hand sides of rules (as regular sets). Unfortunately, in this way grammars might be derived which are not satisfying the requirements of an XML grammar. Therefore, our approach is necessary and more adequate for XML documents.

We mention a third application of the inference of DTDs for XML documents in connection with databases: The importance of making use of DTDs —whenever known— to optimize the performance of database queries based on XML has been stressed by various authors, see [8, 11, 27, 33, 34]. Unfortunately, DTDs are not always transferred when XML documents are transmitted. Therefore, an automatic generation of DTDs can be also useful in this case, as well.

A contribution to the GI community. Finally, one can consider this paper also as a contribution to the GI community: Many GI results are known for regular languages, but it seems to be hard to get beyond. This has been formulated as a challenge by de la Higuera in a recent survey article [22].[2] Many authors try to transfer learnability results from the regular language case to the nonregular case by preprocessing. Some of these techniques are surveyed in [18]. Here, we develop a similar preprocessing technique for XML grammars, focussing on a learning model known as *identification in the limit from positive samples* or *exact learning from text.*

[2] A survey of results concerning learning of (subclasses of) context-free languages can be found in [26].

Summary of the paper. The paper is structured as follows. In Section 2, we present XML grammars as introduced by Berstel and Boasson. Section 3 reviews the necessary concepts from the algorithmics of identifying regular languages. In Section 4, we show how to apply the results of Section 3 to the automatic generation of DTDs for XML documents. Finally, we summarize our findings and outline further aspects and prospects of GI issues in relation with XML.

2 XML grammars

Definition and Examples. Berstel and Boasson gave the following formalization of an XML grammar:

Definition 1 An *XML grammar* is composed of a terminal alphabet $T = A \cup \bar{A}$ with $\bar{A} = \{\bar{a} \mid a \in A\}$, of a set of variables $V = \{X_a \mid a \in A\}$, of a distinguished variable called the *axiom* and, for each letter $a \in A$, of a regular set $R_a \subseteq V^*$ which defines the (possibly infinite) set of productions $X_a \to am\bar{a}$ with $m \in R_a$ and $a \in A$. We also write $X_a \to aR_a\bar{a}$ for short.

An *XML language* is generated by some XML grammar.

Note that the syntax of document type definitions (DTDs) as used in XML differs at first glance from the formalization of Berstel and Boasson, but the transfer is done easily.

Example 2 For example, the (rather abstract) DTD

$$
\begin{array}{l}
<!\texttt{DOCTYPE}\ a\ [\\
\qquad\qquad <!\texttt{ELEMENT}\ a \quad ((a|b),(a|b))\ > \\
\qquad\qquad <!\texttt{ELEMENT}\ b \quad (b)*\ > \\
\qquad]>
\end{array}
$$

would be written as:

$$
\begin{array}{rcl}
X_a & \to & a(X_a|X_b)(X_a|X_b)\bar{a} \\
X_b & \to & b(X_b)^*\bar{b}
\end{array}
$$

with axiom X_a.

In other words, an XML grammar corresponds to a DTD in a natural fashion and vice versa. As to the syntax of DTDs, the axiom of the grammar is introduced by `DOCTYPE`, and the set of rules associated to a tag by `ELEMENT`. Indeed, an element is composed of a *type* and a *content model*. Here, the type is the tag name and the content model is a regular expression for the right-hand sides of the rules for this tag. We finally remark that *entities* as well as `#PCDATA` (i.e., textual) information are ignored in the definition of XML grammars. Below, we will show that it is easy to cope with the textual information.

Example 3 Let $A = \{a_1, \ldots, a_n\}$. The language D_A of *Dyck primes* over $T = A \cup \bar{A}$, generated by

$$
\begin{array}{rcl}
X & \to & X_{a_1}|\ldots|X_{a_n}, \text{ where, for } a \in A, \\
X_a & \to & a(X_{a_1}|\ldots|X_{a_n})^*\bar{a}
\end{array}
$$

with axiom X is not an XML language. However, each variable X_a of this grammar generates the XML language

$$D_a := D_A \cap a(A \cup \bar{A})^* \bar{a}.$$

Especially, $D_{\{a\}}$ is an XML language.

Simple properties. By definition of an XML grammar, the following is quite clear:

Lemma 4 If $L \subseteq (A \cup \bar{A})^*$ is an XML language, then $L \subseteq D_A$.

Therefore, Berstel and Boasson derived necessary and sufficient conditions for a subset L of D_A to be an XML language.

We now give some notions we need for stating some of these conditions. We denote by $F(L)$ the set of *factors* of $L \subseteq \Sigma^*$, i.e., $F(L) = \{x, y, z \in \Sigma^* \mid xyz \in L\}$. For $L \subseteq (A \cup \bar{A})^*$, let $F_a(L) = D_a \cap F(L)$ be the set of those factors in L that are also Dyck primes starting with letter $a \in A$. Using these notions, we may sharpen the previous lemma as follows:

Lemma 5 If $L \subseteq (A \cup \bar{A})^*$ is an XML language, then $L = F_a(L)$ for some $a \in A$.

Characterizing XML languages via regular languages. Consider $w \in D_a$. w is uniquely decomposable as $w = au_{a_1}u_{a_2} \ldots u_{a_n}\bar{a}$, with $u_{a_i} \in D_{a_i}$ for $i = 1, \ldots, n$. The *trace* of w is defined as $a_1 \ldots a_n \in A^*$. The set $S_a(L)$ of all traces of words in $F_a(L)$ is called *surface of $a \in A$ in $L \subseteq D_A$*.

Surfaces are useful for defining XML grammars. Consider a family $\mathcal{S} = \{S_a \mid a \in A\}$ of regular languages over A. The *standard XML grammar* $G_{\mathcal{S}}$ associated to $\mathcal{S}$ is defined as follows. The set of variables is $V = \{X_a \mid a \in A\}$. For each $a \in A$, set $R_a = \{X_{a_1} \ldots X_{a_n} \mid a_1 \ldots a_n \in S_a\}$ and consider the rules $X_a \to aR_a\bar{a}$. By definition, $G_{\mathcal{S}}$ is indeed an XML grammar for any choice of the axiom. Moreover, for each language L_a generated from axiom X_a by using the rules of $G_{\mathcal{S}}$, it can be shown that $S_a(L_a) = S_a$.

Now, consider for a family $\mathcal{S} = \{S_a \mid a \in A\}$ of regular languages over A and some fixed letter $a_0 \in A$ the family $\mathcal{L}(\mathcal{S}, a_0)$ of those languages languages $L \subseteq D_{a_0}$ such that $S_a(L) = S_a$ for all $a \in A$. Since $\mathcal{L}(\mathcal{S}, a_0)$ is closed under (arbitrary) union, there is a maximal element in this family. Berstel and Boasson derived the following nice characterization [7, Theorem 4.1]:

Theorem 6 Consider a family $\mathcal{S} = \{S_a \mid a \in A\}$ of regular languages over A and some fixed letter $a_0 \in A$. The language generated by the standard XML grammar $G_{\mathcal{S}}$ with axiom X_{a_0} is the maximal element of the family $\mathcal{L}(\mathcal{S}, a_0)$. Moreover, this is the only XML language in $\mathcal{L}(\mathcal{S}, a_0)$.

Finally, [7, Proposition 3.8] yields:

Lemma 7 If L is an XML language, then there exists a standard XML grammar generating L.

Therefore, there is a *one-to-one correspondence between surfaces and XML languages*. This is the key observation for transferring learnability results known for regular languages to XML languages.

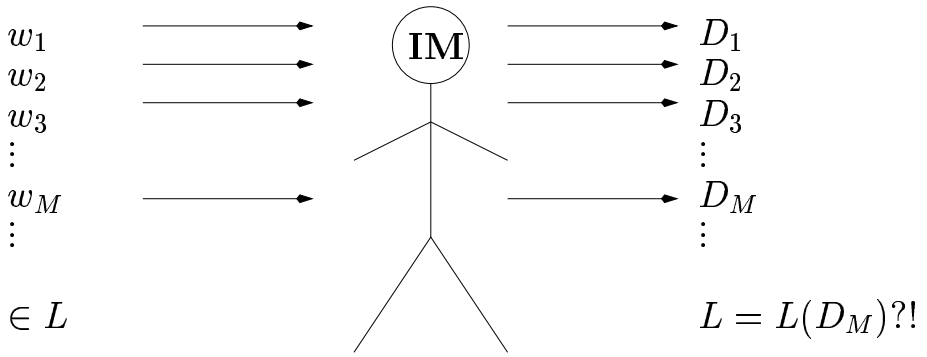

Figure 1: Gold's learning scenario

3 A learning scenario

Gold-style learning. When keeping in mind the possible applications of inferring XML grammars, the typical situation is that an algorithm is needed that, given a set of examples that should fit the sought DTD, proposes a valid DTD. This corresponds to the learning model *identification in the limit from positive samples*, also known as *exact learning from text*, which was introduced by Gold [20] and has been studied thoroughly by various authors within the computational learning theory and the grammatical inference communities.

Definition 8 Consider a language class $\mathcal{L}$ defined via a class of language describing devices $\mathcal{D}$ as, e.g., grammars or automata. $\mathcal{L}$ is said to be *identifiable* if there is a so-called *inference machine* IM to which as input an arbitrary language $L \in \mathcal{L}$ may be enumerated (possibly with repetitions) in an arbitrary order, i.e., IM receives an infinite input stream of words $E(1), E(2), \ldots$, where $E : \mathbb{N} \to L$ is an enumeration of L, i.e., a surjection, and IM reacts with an output stream $D_i \in \mathcal{D}$ of devices such that there is an $N(E)$ so that, for all $n \geq N(E)$, we have $D_n = D_{N(E)}$ and, moreover, the language defined by $D_{N(E)}$ equals L.

Figure 1 tries to illustrate this learning scenario for a fixed language class $\mathcal{L}$ described by the device class $\mathcal{D}$. Often, it is convenient viewing IM mapping a finite sample set $I_+ = \{w_1, \ldots, w_M\}$ to a hypothesis D_M. The aim is then to find algorithms which, given I_+, produce a hypothesis D_M describing a language $L_M \supseteq I_+$ such that, for any language $L \in \mathcal{L}$ which contains I_+, $L_M \subseteq L$. In other words, L_M is the smallest language in $\mathcal{L}$ extending I_+.

Already Gold [20] established:

Lemma 9 The class of regular languages is not identifiable.

This result readily transfers to XML languages:

Lemma 10 The class of all XML languages (over a fixed alphabet) is not identifiable.

Identifiable regular subclasses. Since we think that the inference of XML grammars has important practical applications (as detailed in the Introduction), we show how to define identifiable subclasses of the XML languages. To this end, we reconsider the identification of subclasses of the regular languages, because XML grammars and regular languages are closely linked due to the one-to-one correspondence of XML standard grammars and regular surfaces as stated in the preceding section.

Since the regular languages are a very basic class of languages, many attempts have been made to find nice identifiable subclasses of the regular languages. According to Gregor [21], among the most popular identifiable regular language classes are the k-reversible languages [4] and the terminal-distinguishable languages [29, 30]. Other identifiable subclasses are surveyed in [28]. A nice overview on the involved automata and algorithmic techniques can be found in [13]. Recently, we developed a framework which generalizes the explicitly mentioned language classes in a uniform manner [15]. We will briefly introduce this framework now.

Definition 11 Let F be some finite set. A mapping $f : T^* \to F$ is called a *distinguishing function* if $f(w) = f(z)$ implies $f(wu) = f(zu)$ for all $u, w, z \in T^*$.

$L \subseteq T^*$ is called *f-distinguishable* if, for all $u, v, w, z \in T^*$ with $f(w) = f(z)$, we have $zu \in L \iff zv \in L$ whenever $\{wu, wv\} \subseteq L$.

The family of f-distinguishable languages (over the alphabet T) is denoted by (f, T)-DL.

For $k \geq 0$, the example $f(x) = \sigma_k(x)$ (where $\sigma_k(x)$ is the suffix of length k of x if $|x| \geq k$, and $\sigma_k(x) = x$ if $|x| < k$) leads to the k-reversible languages, and $f(x) = \mathrm{Ter}(x) = \{ a \in T \mid \exists u, v \in T^* : uav = x \}$ yields (reversals of) the terminal-distinguishable languages.

We derived another characterization of (f, T)-DL based on automata [15].

Definition 12 Let $\mathcal{A} = (Q, T, \delta, q_0, Q_F)$ be a finite automaton. Let $f : T^* \to F$ be a distinguishing function. A is called *f-distinguishable* if:

1. $\mathcal{A}$ is deterministic.
2. For all states $q \in Q$ and all $x, y \in T^*$ with $\delta^*(q_0, x) = \delta^*(q_0, y) = q$, we have $f(x) = f(y)$.

   (In other words, for $q \in Q$, $f(q) := f(x)$ for some x with $\delta^*(q_0, x) = q$ is well-defined.)
3. For all $q_1, q_2 \in Q$, $q_1 \neq q_2$, with either (i) $q_1, q_2 \in Q_F$ or (ii) there exist $q_3 \in Q$ and $a \in T$ with $\delta(q_1, a) = \delta(q_2, a) = q_3$, we have $f(q_1) \neq f(q_2)$.

Theorem 13 A language is f-distinguishable iff it is accepted by an f-distinguishable automaton.

We return now to the issue of learning. In [15], we have shown the following theorem:

Theorem 14 For each alphabet T and each distinguishing function $f : T \to F$, the class (f, T)-DL is identifiable.

Moreover, there is an identification algorithm which, given the finite sample set $I_+ \subset T^*$ as input, yields a finite automaton hypothesis $\mathcal{A}$ in time $O(\alpha(|F|n)|F|n)$, where α is the inverse Ackermann function[3] and n is the total length of all words in I_+.

The language recognized by $\mathcal{A}$ is the smallest f-distinguishable language containing I_+.

Note 15 Since, in principle, the language classes (f,T)-DL grow when the size of the range F of f grows, the algorithm mentioned in the preceding theorem offers a natural trade-off between precision (i.e., getting more and more of the regular languages) and efficiency. From another viewpoint, f can be seen as the *explicit bias* or *commitment* one has to make when learning regular languages from text exactly. Since, due to Lemma 9, restricting the class of regular languages towards identifiable subclasses cannot be circumvented, having an explicit and well-formalized bias which characterizes the identifiable language class is of natural interest.

A merging state inference algorithm. For reasons of space, we will only sketch the inference algorithm. Note that the algorithm is a merging state algorithm similar to the algorithm for inferring 0-reversible languages as developed by Angluin [4].

Consider an input sample set $I_+ = \{w_1, \ldots, w_M\} \subseteq T^+$ of the inference algorithm. Let $w_i = a_{i1} \ldots a_{in_i}$, where $a_{ij} \in T$, $1 \leq i \leq M$, $1 \leq j \leq n_i$. We are going to describe a simple nondeterministic automaton accepting exactly I_+. Namely, the *skeletal automaton* for the sample set is defined as

$$\begin{aligned}
\mathcal{A}_S(I_+) &= (Q_S, T, \delta_S, Q_0, Q_f), \quad \text{where} \\
Q_S &= \{\, q_{ij} \mid 1 \leq i \leq M, 1 \leq j \leq n_i \,\}, \\
\delta_S &\cup \{\, (q_{ij}, a_{i,j+1}, q_{i,j+1}) \mid 1 \leq i \leq M, 1 \leq j < n_i \,\}, \\
Q_0 &= \{\, q_{i1} \mid 1 \leq i \leq M \,\} \quad \text{and} \\
Q_f &= \{\, q_{in_i} \mid 1 \leq i \leq M \,\}.
\end{aligned}$$

Observe that we allow a *set* of initial states. The *frontier string* of q_{ij} is defined by $\mathrm{FS}(q_{ij}) = a_{ij} \ldots a_{in_i}$. The *head string* of q_{ij} is defined by $\mathrm{HS}(q_{ij}) = a_{i1} \ldots a_{i,j-1}$. In other words, $\mathrm{HS}(q_{ij})$ is the unique string leading from an initial state into q_{ij}, and $\mathrm{FS}(q_{ij})$ is the unique string leading from q_{ij} into a final state. Therefore, the skeletal automaton of a sample set simply spells all words of the sample set in a trivial fashion. Since there is only one word leading to any q, namely $\mathrm{HS}(q)$, $f(q) = f(\mathrm{HS}(q))$ is well-defined.

Now, for $q_{ij}, q_{k\ell} \in Q_S$, define $q_{ij} \rightleftharpoons_f q_{k\ell}$ iff (1) $\mathrm{HS}(q_{ij}) = \mathrm{HS}(q_{k\ell})$ or (2) $\mathrm{FS}(q_{ij}) = \mathrm{FS}(q_{k\ell})$ as well as $f(q_{ij}) = f(q_{k\ell})$. In general, $\rightleftharpoons_f$ is not an equivalence relation. Hence, define $\equiv_f := (\rightleftharpoons_f)^+$, denoting in this way the transitive closure of the original relation. Then, we can prove:

Lemma 16 For each distinguishing function f and each sample set I_+, $\equiv_f$ is an equivalence relation on the state set of $\mathcal{A}_S(I_+)$.

[3] as defined by Tarjan [32]; α is an extremely slowly growing function

The gist of the inference algorithm is to merge $\equiv_f$-equivalent states of $\mathcal{A}_S(I_+)$. Formally speaking, the notion of quotient automaton construction is needed. We briefly recall this notion:

A *partition* of a set S is a collection of pairwise disjoint nonempty subsets of S whose union is S. If π is a partition of S, then, for any element $s \in S$, there is a unique element of π containing s, which we denote $B(s, \pi)$ and call the *block* of π containing s. A partition π is said to *refine* another partition π' iff every block of π' is a union of blocks of π. If π is any partition of the state set Q of the automaton $\mathcal{A} = (Q, T, \delta, q_0, Q_F)$, then the *quotient automaton* $\pi^{-1}\mathcal{A} = (\pi^{-1}Q, T, \delta', B(q_0, \pi), \pi^{-1}Q_F)$ is given by $\pi^{-1}\hat{Q} = \{ B(q, \pi) \mid q \in \hat{Q} \}$ (for $\hat{Q} \subseteq Q$) and $(B_1, a, B_2) \in \delta'$ iff $\exists q_1 \in B_1 \exists q_2 \in B_2 : (q_1, a, q_2) \in \delta$.

We consider now the automaton $\pi_f^{-1}\mathcal{A}_S(I_+)$, where π_f is the partition induced by the equivalence relation $\equiv_f$. We have shown [17]:

Theorem 17 For each distinguishing function f and each sample set I_+, the automaton $\pi_f^{-1}\mathcal{A}_S(I_+)$ is an f-distinguishable automaton.

Moreover, the language accepted by $\pi_f^{-1}\mathcal{A}_S(I_+)$ is the smallest f-distinguishable language containing I_+.

Therefore, it suffices to compute $\mathcal{A}_S(I_+)$, $\equiv_f$ and finally $\pi_f^{-1}\mathcal{A}_S(I_+)$ in order to obtain a correct hypothesis in the sense of Gold's model. Observe that the notion of quotient automaton formalizes the intuitive idea of "merging equivalent states."

4 Learning document type definitions

An XML grammar identification algorithm. We propose the following strategy for inferring XML grammars.

Algorithm 18 (Sketch)

1. Firstly, one has to commit oneself to a distinguishing function f formalizing the bias of the learning algorithm.
2. Then, the sample XML document has to be transformed into sets of positive samples, one such sample set I_+^a for each surface which has to be learned.
3. Thirdly, each I_+^a is input to an identification algorithm for f-distinguishable languages, yielding a family $\mathcal{S} = \{S_a \mid a \in A\}$ of regular f-distinguishable languages over A.
4. Finally, the corresponding XML standard grammar is output.

Note 19 Let us comment on the first step of the sketched algorithm. Due to Lemma 10, it is impossible to identify any XML language in the limit from positive samples. Note 15 explains the advantage of having an explicit bias in such situations. Choosing a bias can be done in an incremental manner, starting with the trivial distinguishing function which characterizes the 0-reversible languages and integrating more and more features into the chosen distinguishing function whenever appropriate. This is also important due to the exponential

dependence of the running time of the employed algorithm on the size of the range of the chosen distinguishing function, see Theorem 14. Conversely, a too simplistic commitment would entail the danger of "over-generalization" which is a frequently discussed topic in GI. Hence, when a user encounters a situation where the chosen algorithm generalizes too early or too much, she may choose a more sophisticated distinguishing function.

Note 20 Of course, it is also possible to use other than f-distinguishable identifiable language classes in order to define identifiable subclasses of XML languages. For example, Ahonen [2, 3] proposed taking a variant of what is known as k-testable languages [19] (which is basically a formalization of the empiric k-gram approach well-known in pattern recognition, see the discussion in [16]).

Note 21 Theorem 17 immediately implies for the class XML(f, A) of XML languages over the tag alphabet $T = A \cup \bar{A}$ whose surface is f-distinguishable is identifiable by means of Algorithm 18.

A bookstore example. Let us clarify the procedure sketched in Alg. 18 by an extended example:

Example 22 We discuss a bookstore which would like to prepare its internet appearance by transforming its offers into XML format. Consider the following entry for a book:

```
<book>
   <author><last-name>Abiteboul</last-name></author>
   <author><last-name>Vercoustre</last-name></author>
   <title>Research and Advanced Technology for Digital Libraries.
       Third European Conference</title>
   <price>56.24 Euros</price>
</book>
```

Further, assume that, for $f : \Sigma \to F$, $|F| = 1$, i.e., we are considering the distinguishing function f corresponding to the 0-reversible languages in the dictum of Angluin [4]. First, let us rewrite the given example in the formalism of Berstel and Boasson. To this end, let X_b correspond to the tag pair `<book>` and `</book>`, X_a correspond to `<author>` and `</author>`, X_n correspond to `<last-name>` and `</last-name>`, X_t correspond to `<title>` and `</title>` and X_p correspond to `<price>` and `</price>`. Let us further write each tag pair belonging to variable X_y as $y, \bar{y}$ as in the examples above. The given concrete book example then reads as $w = ban\bar{n}\bar{a}an\bar{n}\bar{a}t\bar{t}p\bar{p}\bar{b}$. Here, we ignore an arbitrary data text. Obviously, $w \in D_b$. We find the decomposition $w = bu_au_au_tu_p\bar{b}$ with $u_a = an\bar{n}\bar{a} \in D_a$ and $u_t = t\bar{t} \in D_t$ and $u_p = p\bar{p} \in D_p$. The trace belonging to w is therefore $aatp$. By definition, $aatp$ belongs to the surface S_b which has to be learned.

Consider as a second input example:

```
<book>
   <author><last-name>Thalheim</last-name></author>
   <title>Entity-Relationship Modeling.
       Foundations of Database Technology</title>
   <price>50.10 Euros</price>
</book>
```

From this example, we may infer that atp belongs to S_b, as well. The inference algorithm for 0-reversible languages would now yield the hypothesis $S_b = a^+tp$, which is in fact a reasonable generalization for our purpose, since probably a book in a bookstore will be always specified by a non-empty list of authors, its title and its price. Incorporating arbitrary data text (`#PCDATA`) by means of a place-holder τ in a natural fashion, the following XML grammar will be inferred:

$$\begin{aligned}
X_b &\to bR_b\bar{b} \quad \text{with} \\
R_b &= \{X_a^j X_t X_p \mid j > 0\}, \\
X_a &\to aR_a\bar{a} \quad \text{with} \\
R_a &= \{X_n\}, \\
X_n &\to n\tau\bar{n}, \\
X_t &\to t\tau\bar{t}, \\
X_p &\to p\tau\bar{p}.
\end{aligned}$$

We conclude this section with a remark concerning a special application described in the introduction.

Note 23 When creating restricted or specialized views on documents (which is one of the possible inference tasks proposed by Ahonen), one can assume that the large DTD is known to the inference algorithm. Then, it is of course useless to infer regular languages which are not subsets of the already given "maximal" surfaces S_a. Therefore, it is reasonable to take as "new" hypothesis surfaces $S'_a \cap S_a$, where S'_a is the surface output by the employed regular language inference algorithm.

5 Conclusions

Our findings. We presented a method which allows to transfer results known from the learning of regular languages towards the learning of XML languages. We will provide a competitive implementation of our algorithms shortly via the WWW.

Two further applications. The derivation of DTDs is not the only possible application of GI techniques in XML design. Another important issue is the design of appropriate *contexts*. For example, Brüggemann-Klein and Wood [9, 10] introduced so-called caterpillar expressions (and automata) which can be used to model contexts in XML grammars. Since a caterpillar automaton is nothing else than a finite automaton whose accepted input words are interpreted as commands of the caterpillar (which then walks along the assumed syntax tree induced by the XML grammar), also for the purpose of designing caterpillar expressions describing contexts, GI techniques may assist the XML designer.

Ahonen [1, 2] mentioned another possible application of GI for DTD generation, namely, assembly of (parts of) tagged documents from different sources (with different original DTDs). Hence, the assembled document is a transformation of one or more existing documents. The problem is to infer a common DTD. This assembly problem has also been addressed for XML recently [5] without referring to GI. The integration of both approaches seems to be promising.

Approximation. One possible objection against our approach could be to note that not *every* possible XML language can be inferred, irrespectively of the chosen distinguishing function, due to Lemma 10. We have observed [17] that, for any distinguishing function f and for every finite subset I_+ of an arbitrary regular set $R \subseteq \Sigma^*$, the language $\pi_f^{-1}\mathcal{A}_S(I_+)$ proposed by our algorithm for identifying f-distinguishable languages is the smallest language in (f, Σ)-DL which contains R. This sort of approximation property was investigated before by Kobayashi and Yokomori [24, 25]. Due to the one-to-one correspondence between regular languages and XML languages induced by the notion of surface, this means that our proposed method for inferring XML languages can be used to approximate any given "spelled" XML language arbitrarily well.

Other learning models. Finally, we mention that the preprocessing technique developped in Algorithm 18 can be applied to other learning scenarios, as well. We showed in [18] how to apply preprocessing methods to query learning and to the morphic generator method. Also negative examples may be included. We do not elaborate these issues here, since we are not aware of nice application scenarios of these models in the XML framework.

References

[1] H. Ahonen. Automatic generation of SGML content models. In *Electronic Publishing '96 (Palo Alto, California, USA)*, September 1996.

[2] H. Ahonen. *Generating grammars for structured documents using grammatical inference methods.* Phd thesis. Also: Report A-1996-4, Department of Computer Science, University of Helsinki, Finland, 1996.

[3] H. Ahonen, H. Mannila, and E. Nikunen. Forming grammars for structured documents: an application of grammatical inference. In R. C. Carrasco and J. Oncina, editors, *Proceedings of the Second International Colloquium on Grammatical Inference (ICGI-94): Grammatical Inference and Applications*, volume 862 of *LNCS/LNAI*, pages 153–167. Springer, 1994.

[4] D. Angluin. Inference of reversible languages. *Journal of the Association for Computing Machinery*, 29(3):741–765, 1982.

[5] R. Behrens. A grammar based model for XML schema integration. In B. Lings and K. Jeffery, editors, *Advances in Databases, 17th British National Conference on Databases (BNCOD 17)*, volume 1832 of *LNCS*, pages 172–190. Springer, 2000.

[6] R. Behrens and G. Buntrock. XML, eine Verwandte der Dyck-Sprachen. In *9.Theorietag der GI-Fachgruppe 0.1.5 Automaten und Formale Sprachen*, volume Preprint 12/99 of *Mathematische Schriften Kassel*, September 1999.

[7] J. Berstel and L. Boasson. XML grammars. In N. Nielsen and B. Rovan, editors, *Mathematical Foundations of Computer Science (MFCS'2000)*, volume 1893 of *LNCS*, pages 182–191. Springer, 2000. Long Version as Technical Report IGM 2000-06, see `www-igm.univ-mlv.fr/~berstel/Recherche.html`.

[8] K. Böhm, K. Aberer, M. T. Özsu, and K. Gayer. Query optimization for structured documents based on knowledge on the document type definition. In *Proc. Advances in Digital Libraries*. IEEE Press, 1998.

[9] A. Brüggemann-Klein, S. Herrmann, and D. Wood. Context and caterpillars and structured documents. In E. V. Munson, C. Nicholas, and D. Wood, editors, *Principles of Digital Document Processing; 4th International Workshop (PODDP'98)*, volume 1481 of *LNCS*, pages 1–9. Springer, 1998.

[10] A. Brüggemann-Klein and D. Wood. Caterpillars, context, tree automata and tree pattern matching. In G. Rozenberg and W. Thomas, editors, *Developments in Language Theory; Foundations, Applications, and Perspectives (DLT'99)*, pages 270–285. World Scientific, 2000.

[11] P. Buneman, W. Fan, and S. Weinstein. Interaction between path and type constraints. In *Proc. 18th ACM Symposium on Principles of Database Systems (PODS'99)*, pages 56–67. ACM Press, 1999.

[12] CZ-Redaktion. Maschinenmenschen plauern per XML mit der Unternehmens-IT. *Computer Zeitung*, (50):30, December 2000.

[13] P. Dupont and L. Miclet. Inférence grammaticale régulière: fondements théoriques et principaux algorithmes. Technical Report RR-3449, INRIA, 1998.

[14] P. Fankhauser and Y. Xu. Markitup! An incremental approach to document structure recognition. *Electronic Publishing – Origination, Dissemination and Design*, 6(4):447–456, 1994.

[15] H. Fernau. Identification of function distinguishable languages. In H. Arimura, S. Jain, and A. Sharma, editors, *Proceedings of the 11th International Conference Algorithmic Learning Theory ALT 2000*, volume 1968 of *LNCS/LNAI*, pages 116–130. Springer, 2000.

[16] H. Fernau. k-gram extensions of terminal distinguishable languages. In *International Conference on Pattern Recognition (ICPR 2000)*, volume 2, pages 125–128. IEEE/IAPR, IEEE Press, 2000.

[17] H. Fernau. Approximative learning of regular languages. Technical Report WSI–2001–2, Universität Tübingen (Germany), Wilhelm-Schickard-Institut für Informatik, 2001.

[18] H. Fernau and J. M. Sempere. Permutations and control sets for learning non-regular language families. In A.L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications, 5th International Colloquium (ICGI 2000)*, volume 1891 of *LNCS/LNAI*, pages 75–88. Springer, 2000.

[19] P. García and E. Vidal. Inference of k-testable languages in the strict sense and applications to syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:920–925, 1990.

[20] E. M. Gold. Language identification in the limit. *Information and Control (now Information and Computation)*, 10:447–474, 1967.

[21] J. Gregor. Data-driven inductive inference of finite-state automata. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(1):305–322, 1994.

[22] C. de la Higuera. Current trends in grammatical inference. In F. J. Ferri et al., editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR'2000*, volume 1876 of *LNCS*, pages 28–31. Springer, 2000.

[23] T. Hu and R. Ingold. A mixed approach toward an efficient logical structure recognition. *Electronic Publishing – Origination, Dissemination and Design*, 6(4):457–468, 1994.

[24] S. Kobayashi. Approximate identification, finite elasticity and lattice structure of hypothesis space. Technical Report CSIM 96-04, Department of Computer Science and Information Mathematics; The University of Electro-Communications, Tokyo 182, Japan, 1996.

[25] S. Kobayashi and T. Yokomori. Learning approximately regular languages with reversible languages. *Theoretical Computer Science*, 174(1–2):251–257, 1997.

[26] L. Lee. Learning of context-free languages: a survey of the literature. Technical Report TR-12-96, Center for Research in Computing Technology, Harvard University, Cambridge, MA, 1996.

[27] D. Maier. Database desiderata for an XML query language. In *Proc. QL'98 — The Query Languages Workshop*, 1998.

[28] E. Mäkinen. Inferring regular languages by merging nonterminals. *International Journal of Computer Mathematics*, 70:601–616, 1999.

[29] V. Radhakrishnan. *Grammatical Inference from Positive Data: An Effective Integrated Approach.* PhD thesis, Department of Computer Science and Engineering, Indian Institute of Technology, Bombay (India), 1987.

[30] V. Radhakrishnan and G. Nagaraja. Inference of regular grammars via skeletons. *IEEE Transactions on Systems, Man and Cybernetics*, 17(6):982–992, 1987.

[31] G. Semeraro, F. Esposito, and D. Malerba. Learning contextual rules for document understanding. In *Proceedings of the 10th IEEE Conference on Artificial Intelligence for Applications*, pages 108–115, 1994.

[32] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the Association for Computing Machinery*, 22(2):215–225, 1975.

[33] P. T. Wood. Optimizing web queries using document type definitions. In *ACM CIKM'99 2nd International Workshop on Web Information and Data Management (WIDM'99)*, pages 28–32. ACM Press, 1999.

[34] P. T. Wood. Rewriting XQL queries on XML repositories. In B. Lings and K. Jeffery, editors, *Advances in Databases, 17th British National Conference on Databases (BNCOD 17)*, volume 1832 of *LNCS*, pages 209–226. Springer, 2000.