

# Learning XML Document Type Descriptors

Hennig Fernau

Universität Tübingen

The University of Newcastle

`fernau@ira.uka.de`

## Overview

- XML in a nutshell
- Grammatical inference
- Application scenarios
- XML grammars
- Inferring DTDs
- RE inference
- MDL principle and complexity

## **XML in a Nutshell**

- a subset of SGML (Standard Generalized Markup Language, ISO Standard 8879)
- enhances HTML
- Well-formed XML documents
- Valid XML documents

Extensible Markup Language (XML) 1.0,

Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen, 10 February 1998.

Available at <http://www.w3.org/TR/REC-xml>

## A Bookstore Example

```
<book>
  <author><last-name>Abiteboul</last-name></author>
  <author><last-name>Vercoustre</last-name></author>
  <title>Research and Advanced Technology for Digital Libraries. ... </ti-
tle>
  <price>56.24 Euros</price>
</book> <book>
  <author><last-name>Thalheim</last-name></author>
  <title>Entity-Relationship Modeling. ... </title>
  <price>50.10 Euros</price>
</book>
```

**well-formed:** correct tag-bracketization

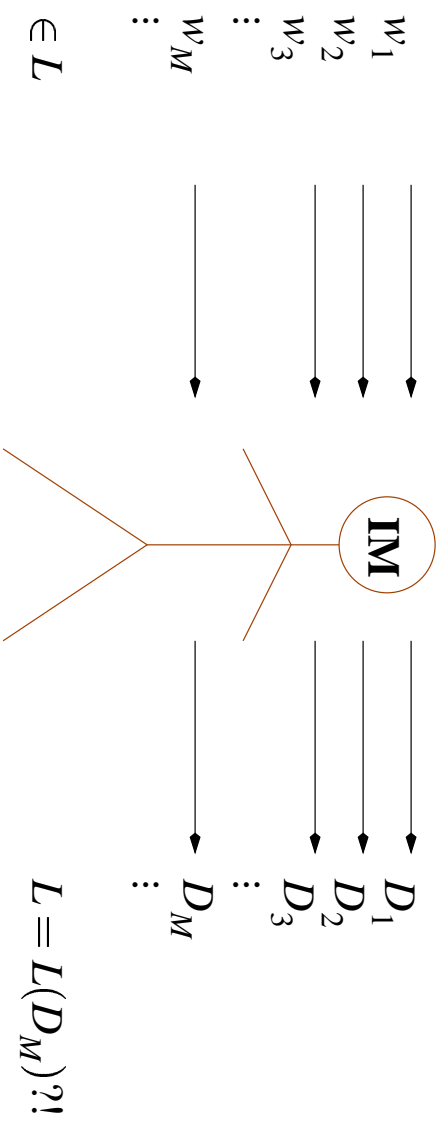
**valid:** according to document type definition (DTD), which is a special form of a context-free grammar.

## **Grammatical Inference / Grammar Induction (GI)**

Aim: Derive DTDs (or grammars, automata, ...) in some standard form from given examples (possibly with additional information)

Main Problem: When should the inference algorithm start to “generalize”?

# Gold's Model of Language Identification / Text Learning



## Application Scenarios for Learning DTDs

- XML document might **lack a “good” DTD**.
- XML document designers might **lack skill** of writing grammars (DTDs)
  - ↳ infer DTDs from well-formed sample document(s)
- Existing tool (example): [www.hit-sw.com/xml\\_utilities/](http://www.hit-sw.com/xml_utilities/)  
part of SAXON by ICL (Fujitsu)  
[Here](#): use of GI theory
- Automatic (**sub-**)**view creation**: automatic modification of large/general DTDs
- **Query optimization**

## Why GI? An “Official” Excuse

see [www.xml.com/axml/notes/Any1.html](http://www.xml.com/axml/notes/Any1.html)

1998, [Tim Bray](#) wrote:

Suppose you're given an existing well-formed XML document and you want to build a DTD for it. One way to do this is as follows:

1. Make a list of all the element types that actually appear in the document, and build a simple DTD which declares each and every one of them as ANY. Now you've got a DTD (not a very useful one) and a valid document.
2. Pick one of the elements, and work out how it's actually used in the document. Design a rule, and replace the ANY declaration with a ... content declaration. This, of course, is the tricky part, particularly in a large document.
3. Repeat step 2, working through the elements one by one, until you have a useful DTD.

## XML Grammars

The (rather abstract) DTD

```
<!DOCTYPE a [  
    <!ELEMENT a ((a|b),(a|b))>  
    <!ELEMENT b (b)*>  
]>
```

would be written as an XML grammar with rules

$$\begin{aligned} X_a &\rightarrow a(X_a|X_b)(X_a|X_b)\bar{a} \\ X_b &\rightarrow b(X_b)^*\bar{b} \end{aligned}$$

and with axiom  $X_a$ .

Use “barring” to map from open to corresponding closed brackets.

## XML Grammars—More Formally (Berstel/Boasson 2000/02)

Thm.: Each XML grammar generates a well-bracketized language (Dyck).

Left brackets:  $[ = \{ [ ]_1, \dots, [ ]_n \}$

Right brackets:  $] = \{ ] ]_1, \dots, ] ]_n \}$

Brackets:  $B = [ \cup ]$ .

$D_i$  is generated by  $X_i \rightarrow [ ]_i (X_1 \mid \dots \mid X_n) ]_i$ .

$w \in D_i$  is **uniquely decomposable** as  $w = [ ]_i^u j_1 \dots j_m ]_i^i$

$[ ]_1 \dots [ ]_m$  is the **trace** of  $w$ .

Let  $L \subseteq [ ]^*$ .

$F_i(L) = \{ [ ]_i^y ]_i \in D_i \mid \exists x, z \in B^*, x [ ]_i^y ]_i^z \in L \}$  are the **factors** of  $L$ .

$S_i(L) = \{ x \in [ ]^* \mid \exists w \in F_i(L), x \text{ is trace of } w \}$  is the **surface set** of  $L$ .

Let  $\mathcal{S} = \{S_i \mid [i \in \Sigma]\}$  be a set of potential surfaces.

$G_{\mathcal{S}}^j$  with axiom  $X_j$  and rules  $X_i \rightarrow [iR_i]$ ,  $R_i = \{X_{j_1} \dots X_{j_m} \mid [j_1 \dots j_m] \in [\Sigma]^*\}$ , is called **standard XML grammar**.

Thm.: If  $L$  is an XML language, then there exists a standard XML grammar for  $L$ .

Observation: Surface sets of XML languages are always regular.

## Infering DTDs

Idea: Infer regular (!) surface sets  $\rightsquigarrow$  algorithm:

1. Transfer sample XML document into sets of positive samples  $I_+^a$  for each surface which must be learned.
2. Infer  $S^a$  from  $I_+^a$  by means of a “Known” regular language learner.
3. Output standard XML grammar.

## Infering Regular Languages

Gold: Not all regular languages can be learned from text.

Angluin: 0-reversible languages (characterized by DFAs which are backward deterministic) can be learned from text.

Angluin, Fernau, Radakrishnan/Nagaraja presented further generalizations.

↪ MLDM '01 paper

## 0-Reversible Language Inference—A Characterization

Thm.: A regular language  $L$  is 0-reversible iff there is a finite automaton  $A$  accepting  $L$  which is both forward- and backward-deterministic ( $\rightsquigarrow$  reversible).

**Inference algorithm:** (Merging state paradigm)

1. Start with “trivial description” of input sample  $I_+$ , e.g., with the **skeletal automaton**  $A := A(I_+)$ .
2. As long as automaton  $A$  is not reversible:  
Modify  $A$  by merging conflicting states.

Thm.: The resulting automaton is a minimal deterministic automaton generating the smallest 0-reversible language containing  $I_+$ .

## The Bookstore Revisited

Create nonterminals for tags:

$X_b$  corresponds to `<book>` and `</book>`,

$X_a$  corresponds to `<author>` and `</author>`,

$X_n$  corresponds to `<last-name>` and `</last-name>`,

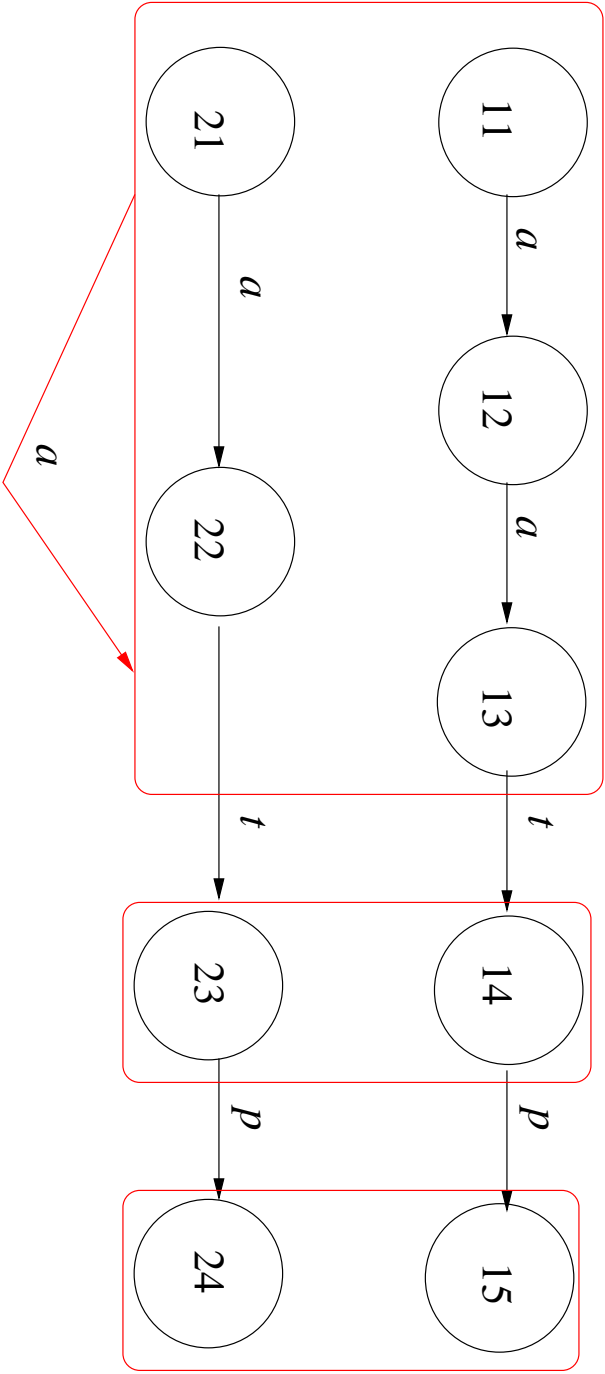
$X_t$  corresponds to `<title>` and `</title>`,

$X_p$  corresponds to `<price>` and `</price>`.

Read off sample surfaces:  $I_+^b = \{atp, atp\}$ .

Next slide:  $S^b = a^*tp$  (!) is inferred.

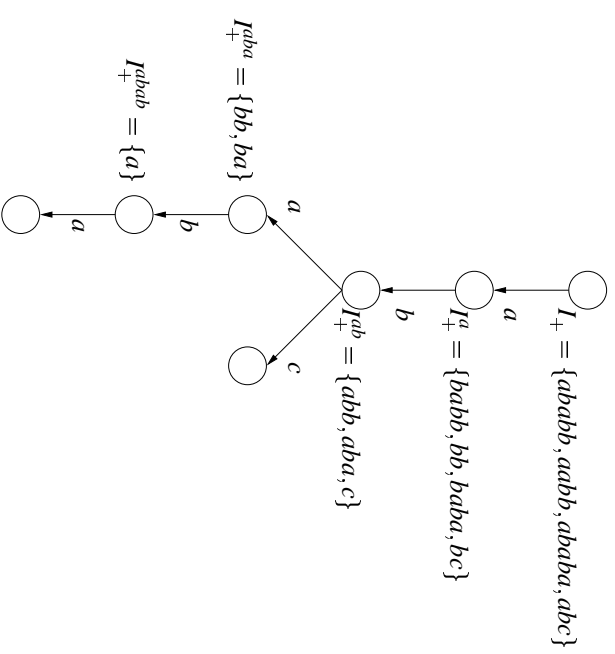
```
<!ELEMENT book (author*, title, price) >  
is proposed as part of DTD.
```



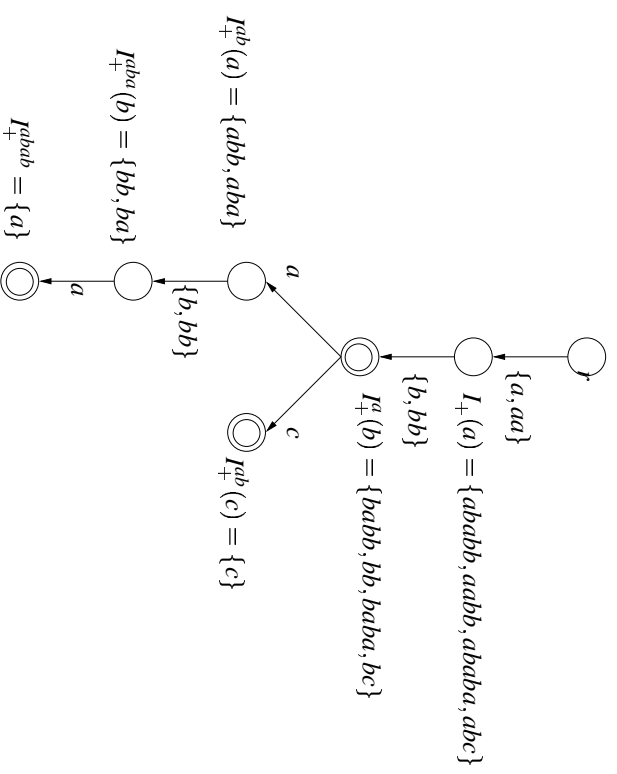
## Problems With this Approach

- DTD grammars rather require **regular expressions** than **deterministic finite automata**
- The theoretical “exponential blowup” turns into a “practical unreadability.”
- ↪ recent work on **inference of regular expressions** based on left-alignment principle (as Brad’s work)

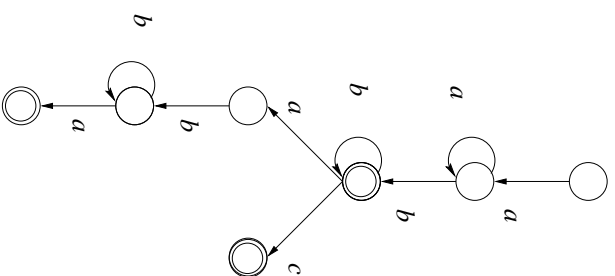
# A simple prefix block tree



# The resulting NFA



# The resulting DFA



## Alternative Approaches

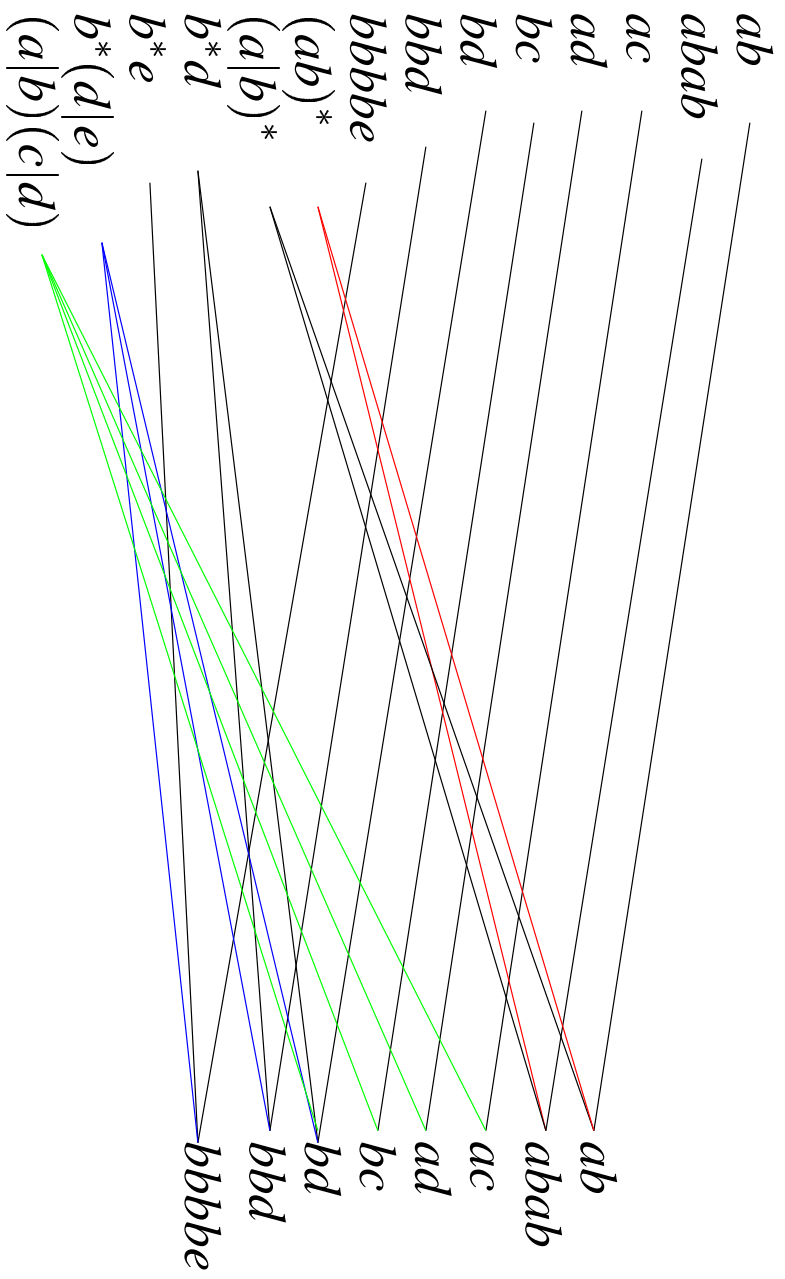
- “Mere heuristics” (not characterized; not well documented)
- Stochastic and error correcting approaches (see later)
- Comparative and competitive:
  - Minimum description length (MDL) principle.
  - likewise: Occam’s razor.
  - The shortest possible explanation is the best one.

## Applying MDL Principle to DTD Inference

Garofalakis et al. 2003 XTRACT; Witten et al. 1994.

The **length of a description** consists of two parts:

1. the **length of the theory** (in bits) and
2. the **length of the data** (in bits) when encoded with the help of the theory.



## MDL as a Combinatorial Problem

MDL-OPTIMAL-CODING

Given:

- a set  $R = \{r_1, \dots, r_n\}$  of **regular expressions** (over the basic alphabet  $\Sigma$ ),
- a set of **strings**  $S = \{s_1, \dots, s_m\} \subset \Sigma^+$ ,
- and a positive **integer**  $k$ .

Question: Is it possible to find a subset  $R'$  of  $R$  such that

$$\sum_{r \in R'} |c(r)| + \sum_{s \in S} |c(s|R')| \leq k \quad ?$$

Thm.: MDL-OPTIMAL-CODING is NP-complete.

## Any Ways Out?

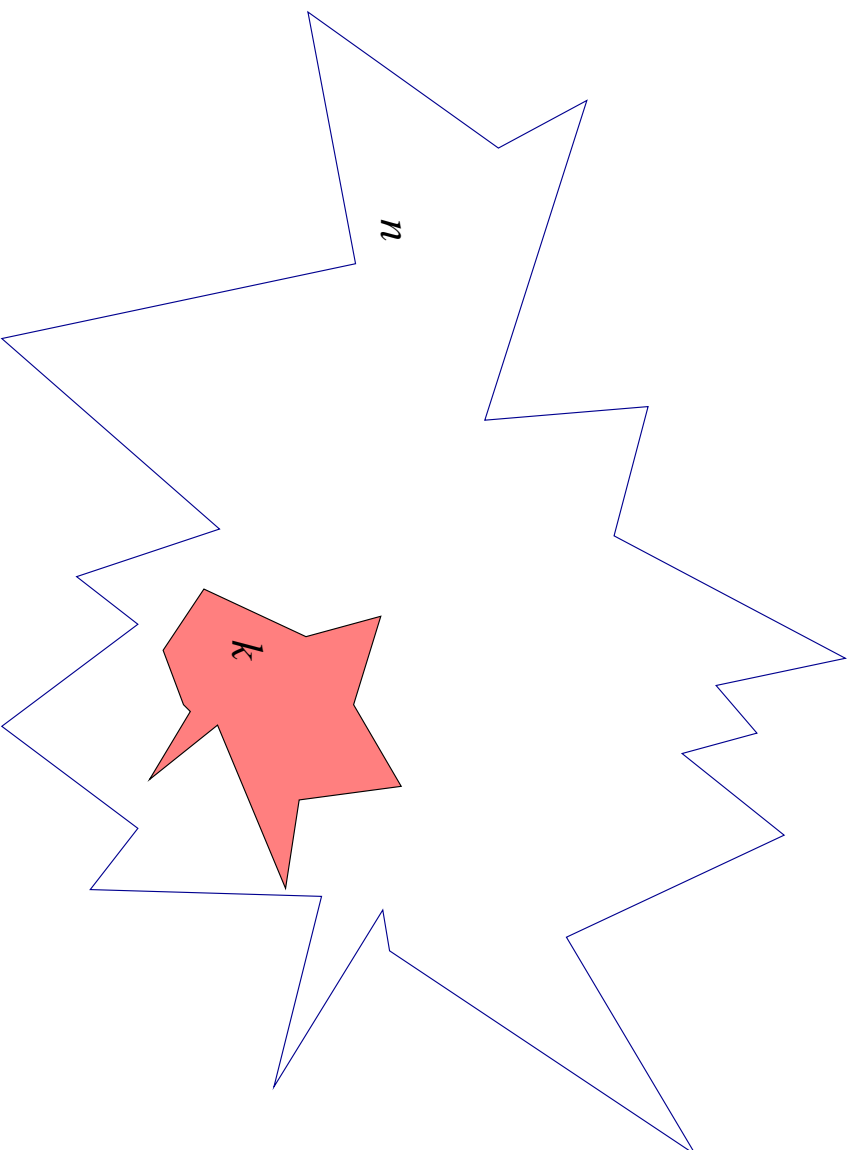
Garofalakis propose using the related **facility location problem**.

- place **facilities** (in our case: the **selected theory**)
- into some **locations** (here: the **regular expressions** to choose from)
- so that the **costs** incurred by the facilities (here: the **number of bits** needed to encode the theory) plus the costs incurred by serving a given set of customers (in this case: # of bits for encoding the given strings)

is minimal.

**Problem:** Facility location is also NP-complete and HARD to approximate.

# The Curse of Combinatorics



## A Parameterized Approach (together with M. Fellows)

Thm.: Given  $k$ , the question if a MDL-OPTIMAL-CODING instance  $I$  incurs a cost of at most  $k$  can be answered in time  $f(k)p(|I|)$  for some function  $f$  and a polynomial  $p$ .

**Ongoing work:** Get  $f$  “as small as possible.”

**A further natural parameter:** # of permitted errors

In our example,  $(a|b)^*$  and  $b^*d$  explain all instances with at most one edit error.