

Extracting minimum length
Document Type Definitions is NP-hard

Henning Fernau

Universität Tübingen

The University of Newcastle

`fernau@ira.uka.de`

Overview

- XML in a nutshell
- Grammatical inference
- Application scenarios
- XML grammars
- MDL principle
- NP-hardness of RE inference based on MDL

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. ...</title>
```

```
  <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”?

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.hitsw.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

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 (Berstel/Boasson 2002) 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.
The right-hand sides correspond to regular expressions.

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 for regular language learner: $I_+^b = \{aatp, atp\}$.

Possible guess: $S^b = a^*tp$.

```
<!ELEMENT book (author*,title,price) >
```

is proposed as part of DTD.

Applying MDL Principle to DTD Inference

Basic idea dates back to [Solomonoff 1964](#)

[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.

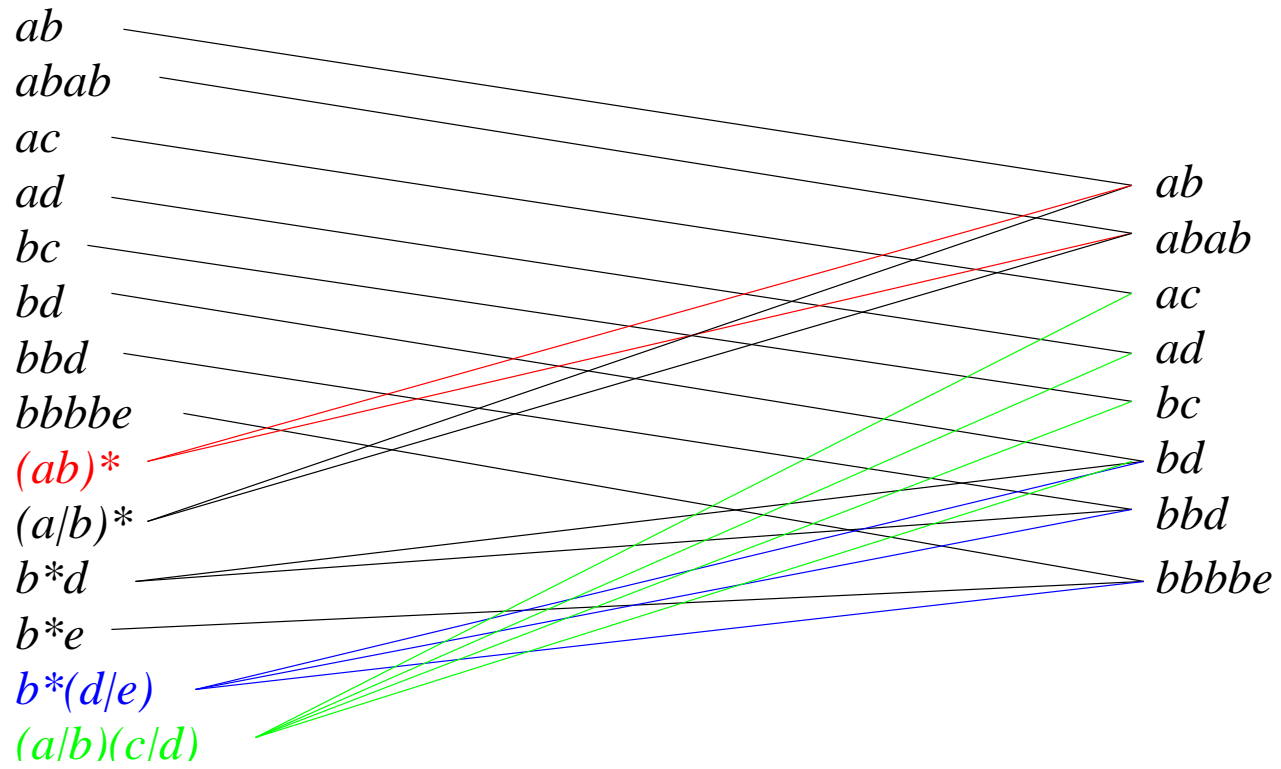
MDL trade-off according to Solomonoff (1997):

“I was trying to find an algorithm for the discovery of the ‘best’ grammar for a given set of acceptable sentences. One of the things I sought was: Given a set of positive cases of acceptable sentences and several grammars, any of which is able to generate all the sentences, what goodness of fit criterion should be used?”

It is clear that the **ad hoc grammar**, that lists all of the sentences in the corpus, fits perfectly.

The **promiscuous grammar**, that accepts any conceivable sentence, also fits perfectly.

The first grammar has a long description; the second has a short description. It seemed that **some grammar half-way between these, was ‘correct’**—but what criterion should be used?”



MDL as a Combinatorial Problem MDL-OPTIMAL-CODING

Given:

- a set $R = \{r_1, \dots, r_n\}$ of **regular expressions** (over the basic alphabet Σ),
- 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.

More on the coding function

For a string s and a regular expression r such that $s \in L(r)$, define $c(s|r)$:

- $c(s|s) = e$ (the empty word),
- $c(s|r_1 \cup \dots \cup r_m) = b(i)c(s|r_i)$ if $s \in L(r_i)$,
- $c(e|r^*) = c(e|r?) = 0$,
- $c(s|r?) = 1c(r)$ if $s \neq e$,
- $c(s_1 \dots s_k|r^*) = c(s_1 \dots s_k|r^+) = 1^{\log_2 k} 0b(k)c(s_1|r) \dots c(s_k|r)$ if $k > 0$ and $s_i \in L(r)$,
- $c(s_1 \dots s_k|r_1 \dots r_k) = c(s_1|r_1) \dots c(s_k|r_k)$ for $k > 1$ if $s_i \in L(r_i)$.

Sample codings

$$c(ab|(ab)^*) = 101$$

$$c(abab|(ab)^*) = 11010$$

$$c(ac|(a \cup b)(c \cup d)) = 00 \quad \text{similar } ad, bc, bd$$

$$c(bd|b^*(d \cup e)) = 1010$$

$$c(bbd|b^*(d \cup e)) = 110100$$

$$c(bbbbe|b^*(d \cup e)) = 11101001$$

Solving 3-SAT with MDL-OPTIMAL-CODING

E a boolean expression which contains n variables x_1, \dots, x_n in m clauses C_1, \dots, C_m with $n = 2^j$.

The strings and the regular expressions we construct have as basic alphabet (*literal alphabet*) the $2n$ letters x_1, \dots, x_n and $\bar{x}_1, \dots, \bar{x}_n$.

A clause $C_i = \ell(i, 1) \vee \ell(i, 2) \vee \ell(i, 3)$ is interpreted as a string of length three:

$$s(C_i) = \ell(i, 1)\ell(i, 2)\ell(i, 3)$$

(for $i = 1, \dots, m$). Moreover, we introduce the strings $s_i = x_i\bar{x}_i$ for $i = 1, \dots, n$.

All these together are the strings $s(E)$ of the MDL instance.

Now the theory proposals:

For each variable x_i , let $NON(x_i)$ ($NON(\bar{x}_i)$, resp.) denote the following regular expression:

$$NON(x_i) = \left(\bigcup_{j \neq i} x_j \cup \bigcup_j \bar{x}_j \right)^*$$

$$NON(\bar{x}_i) = \left(\bigcup_j x_j \cup \bigcup_{j \neq i} \bar{x}_j \right)^* .$$

Hence, for each literal ℓ , the language $L(NON(\ell))$ contains all sequences of literals in which ℓ does not occur.

Then, we take for each x_i the expressions

$$RE(x_i) = NON(x_i)x_iNON(x_i)$$

and

$$RE(\bar{x}_i) = NON(\bar{x}_i)\bar{x}_iNON(\bar{x}_i)$$

as given regular expressions $RE(E)$ from which the “theory” has to be selected. In words, for a literal ℓ , $RE(\ell)$ contains all words over the literal alphabet in which ℓ occurs exactly once.

This reduces the 3-SAT instance E to the MDL instance $(RE(E), S(E), k(E))$, where $k(E)$ is some constant.

Useful claims

Claim 1: Each feasible theory selection has to select at least one of the expressions $RE(x_i)$ or $RE(\bar{x}_i)$ for each variable x_i .

Proof: Otherwise, the string s_i would not be covered.

Claim 2: If E is satisfiable via a variable assignment α , then selecting as theory T_α defined by: for $i = 1, \dots, n$, $RE(x_i)$ if $\alpha(x_i) = 1$ and $RE(\bar{x}_i)$ if $\alpha(x_i) = 0$ covers all strings in $S(E)$.

Proof: Since α satisfies E , the selected theory surely covers all s_i and all $s(C_j)$ for each clause C_j in E .

Claim 3: If s_i is covered by $E_i = RE(x_i)$ or by $E_i = RE(\bar{x}_i)$, then

$$|c(s_i|E_i)| = 4 + \log_2(2n - 1).$$

Proof: $s_i = x_i\bar{x}_i$ can be encoded as 0101... using $RE(x_i)$ and as 101...0 using $RE(\bar{x}_i)$, where the ... are used to actually specify the selected literal within the Kleene loop given by $NON(x_i)$ (or $NON(\bar{x}_i)$, resp.).

Claim 3': If T is the selected theory, then $|c(s_i|T)| = 4 + \log_2(2n - 1) + \log_2(|T|)$.

Claim 4: If $s(C_j)$ is covered by $E_i = RE(x_i)$ or by $E_i = RE(\bar{x}_i)$, then

$$|c(s(C_j)|E_i)| = 6 + 2\log_2(2n - 1).$$

Proof: $s(C_j)$, with $C_j = \ell(j, 1) \vee \ell(j, 2) \vee \ell(j, 3)$, can be encoded as follows using $RE(x_i)$ if $x_i = \ell(j, r)$ for some r (the case that $RE(\bar{x}_i)$ covers $s(C_j)$ can be treated similarly):

- If $\ell(j, 1) = x_i$, then $011010\cdots = c(s(C_j)|RE(x_i))$: the first star loop is executed zero times and the second star loop is executed twice; additional $2\log_2(2n - 1)$ bits are required to specify the exact positions.
- If $\ell(j, 2) = x_i$, then $101 \dots 101 \cdots = c(s(C_j)|RE(x_i))$: the first star loop and the last star loop are executed once; additional $\log_2(2n - 1)$ bits are required to specify the exact position each time.
- If $\ell(j, 3) = x_i$, then $11010 \dots 0 = c(s(C_j)|RE(x_i))$, symmetrical to the first case.

Claim 4': If T is the selected theory and if $s(C_j)$ is covered by $E_i = RE(x_i)$ or by $E_i = RE(\bar{x}_i)$, then $|c(s(C_j)|T)| = 6 + 2\log_2(2n - 1) + \log_2(|T|)$.

Claims 2, 3' and 4' imply, together with the observation that $|T_\alpha| = n$ for the theory T_α as described in Claim 2:

Claim 5: If T_α is a theory selected in accordance with a satisfying truth assignment α of the given boolean expression E , then

$$\begin{aligned} |c(S(E)|T_\alpha)| &= m(6 + 2\log_2(2n - 1) + \log_2(n)) \\ &+ n(4 + \log_2(2n - 1) + \log_2(n)). \end{aligned}$$

Claim 6: For every clause $C_j = \ell(j, 1) \vee \ell(j, 2) \vee \ell(j, 3)$, if $s(C_j) \in RE(\ell)$, then $\ell \in \{\ell(j, 1), \ell(j, 2), \ell(j, 3)\}$.

Proof: Recall that $RE(\ell)$ collects all strings that contain the literal ℓ exactly once. Hence, for the string $s(C_j) = \ell(j, 1)\ell(j, 2)\ell(j, 3)$, there are exactly three sets of the form $RE(\ell)$ to which $s(C_j)$ belongs.

Claim 7: If E is not satisfiable, then any feasible theory T covering $S(E)$ contains more than n elements of the form $RE(\ell)$, ℓ being some literal.

Proof: According to Claim 1, at least n elements of the form $RE(\ell)$ must be selected. If exactly n elements were selected, then again by Claim 1 this selection must contain, for each variable x_i , either $RE(x_i)$ or $RE(\bar{x}_i)$. Since every $s(C_j) = \ell(j,1)\ell(j,2)\ell(j,3)$ is covered, according to Claim 6, one of the $RE(\ell(j,r))$ must have been put into T , which means that T corresponds to a satisfying assignment α_T which assigns 1 to x_i iff $RE(x_i)$ was put into T . This contradicts our assumption and shows the claim.

Claims 7, 3' and 4' imply, keeping in mind that $n = 2^j$ for some j :

Claim 8: If E is not satisfiable, then any feasible theory T covering $S(E)$ verifies:

$$\begin{aligned} |c(S(E)|T)| &> m(6 + 2\log_2(2n - 1) + \log_2(n)) \\ &+ n(4 + \log_2(2n - 1) + \log_2(n)). \end{aligned}$$

Claim 9: If E is satisfiable via the assignment α , then encoding the theory T_α costs

$$c(T_\alpha) = n \times c(RE(\ell))$$

many bits, where $c(RE(\ell))$ denotes the cost for encoding the regular expression $RE(\ell)$ corresponding to an arbitrary literal ℓ .

Namely, for all literals ℓ, ℓ' , $c(RE(\ell)) = c(RE(\ell'))$. Since for each variable, an assignment has to be described, the claim follows.

Claim 10: If E is not satisfiable, then any feasible theory T covering $S(E)$ verifies:

$$c(T) \geq n \times c(RE(\ell))$$

many bits, where $c(RE(\ell))$ denotes the cost for encoding the regular expression $RE(\ell)$ corresponding to an arbitrary literal ℓ .

Namely, since in particular every s_i needs to be covered, the claim follows with the reasoning in the previous Claim.

Claims 5 and 8 together with Claims 9 and 10 yield the theorem, selecting

$$\begin{aligned} k(E) &= m(6 + 2\log_2(2n - 1) + \log_2(n)) \\ &+ n(4 + \log_2(2n - 1) + \log_2(n)) + n \times c(RE(\ell)). \end{aligned}$$

Any Ways Out?

Garofalakis et al. 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.