

Inference of Concise DTDs from XML Data¹

Geert Jan Bex ^a Frank Neven ^a Thomas Schwentick ^b Karl Tuyls ^c

^a *Hasselt University and Transnational University of Limburg*

^b *Dortmund University*

^c *Maastricht University and Transnational University of Limburg*

1 Motivation

XML is the lingua franca for data exchange on the Internet. Within applications or communities, XML data is usually not arbitrary but adheres to some structure possibly imposed by a schema. The advantages offered by the presence of such a schema are numerous. The most direct application is of course automatic validation of the document structure. Input validation, for instance, not only facilitates automatic processing but also ensures soundness of the input data. Indeed, unvalidated input from web requests is considered as the number one vulnerability for web applications. The presence of a schema allows for automation and optimization of search, integration, and processing of XML data. Further, the existence of schemas is imperative when integrating (meta) data through schema matching and in the area of generic model management. A final advantage of a schema is that it assigns meaning to the data. That is, it provides a user with a concrete semantics of the document and aids in the specification of meaningful queries over XML data. Although the examples mentioned here just scrape the surface of current applications, they already underscore the importance of schemas accompanying XML data. References are provided in the original paper.

2 Problem setting

Given a collection of XML documents, a schema should be inferred without user intervention and can therefore solely be based on the XML data at hand. Like for any effective inference algorithm, the generated schema should strike a good balance between (1) specialization, i.e., covering all XML documents in the sample in a minimal way; and, (2) generalization, i.e., covering all documents satisfying the target schema but which are not necessarily present in the sample.

In this respect, schema inference problems come in two flavors. First, there is the setting when only little XML data is present, for instance, when XML is returned as answers to queries or Web service requests. In such a case, a schema inference algorithm should balance more towards generalization than to specificity as it is unlikely that a rich class of schemas can be learned from few data instances. In the other scenario, a huge amount of XML data is available, for instance, when the data resides in a native XML databases or is generated in bulk from existing (say relational) data. In this case, there usually is enough information to derive a highly specific schema in a rich class and a learning algorithm should therefore favor specialization over generalization.

We consider the inference of concise Document Type Definitions (DTDs) in both of the above settings rather than XML Schema. The most important reason is that DTD inference has not adequately been addressed yet: existing systems do not perform well when tested on real world or sparse XML data. Secondly, DTD inference is a subcase of XSD inference [3].

As DTDs can be abstracted by context-free grammars with regular expressions (REs) at their right-hand sides, DTD inference reduces to learning of REs from positive example strings. Unfortunately, Gold showed that the class of *all* REs cannot be learned from positive data only [2]. As the

¹To appear in Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, South Korea, September 12–15, 2006

framework for schema inference from XML data is exactly such that only positive example strings are provided, it is unrealistic to develop inference algorithms for the class of all DTDs. One of the main challenges is therefore to identify subclasses of REs which (1) include the large majority of REs occurring in practical DTDs, (2) which are concise, and (3) which can be learned efficiently from positive data only. We present two such classes:

1. The class of **single occurrence REs (SOREs)**, these are REs in which every element name can occur at most once, e.g., $((b?(a+c)^+d)^+e$ is SORE but $a(a+b)^*$ is not: a occurs twice.
2. The class of **chain regular expressions (CHAREs)** which are those SOREs consisting of a sequence of factors $f_1 \cdots f_n$ where every factor is an expression of the form $(a_1 + \cdots + a_k)$, $(a_1 + \cdots + a_k)?$, $(a_1 + \cdots + a_k)^+$, or, $(a_1 + \cdots + a_k)^*$, where $k \geq 1$ and every a_i is an alphabet symbol. For instance, $a(b+c)^*d^+(e+f)?$ is a CHARE, while $(ab+c)^*$ and $(a^*+b^*)^*$ are not.

These classes certainly satisfy the relevance criteria mentioned above: an examination of the 819 DTDs gathered from the Cover Pages as well as from the web at large, reveals that more than 99% of the REs occurring in practical schema's are CHAREs (and therefore also SOREs) [3]. Furthermore, they are succinct by definition: every element name can occur only once.

3 Approach

We introduce the algorithm iDTD that first infers an automaton from a set of strings and then rewrites the latter into an equivalent SORE when one exists, and into a SORE that is a super-approximation otherwise. The size of the produced regular expression is always linear in the number of different alphabet symbols. As every alphabet symbol needs to occur at least once, a SORE can be seen as the most concise representation.

Furthermore, we introduce the algorithm CRX that derives CHAREs (a subclass of SOREs) directly without going through an automaton representation. Whereas iDTD derives more specific REs, the strength of CRX is its strong generalization ability. As a consequence only very small data sets are necessary to infer an optimal CHARE.

Both iDTD and CRX are validated on real world data, incorporating small and large data sets, and on real world DTDs containing REs in and outside the target classes. Our experiments show that iDTD and CRX outperform existing systems such as XTRACT [1] on this type of data. Some results are shown in Table 1. Further, we asses the strong generalization ability of CRX by establishing on average the minimal number of strings needed to derive optimal REs.

Finally extensions of iDTD and CRX are presented to incrementally compute the inferred RE when new data arrives, how to address noise, and how to deal with numerical predicates. We briefly explain how our inference algorithms can be used to infer simple XSDs.

$(a_1 a_2 ? a_3 ?) ? a_4 ? (a_5 + \cdots + a_{18})^*$ $a_1 ? a_2 ? a_3 ? a_4 ? (a_5 + \cdots + a_{18})^*$ $(a_1 a_2 ? a_3 ?) ? a_4 ? (a_5 + \cdots + a_{18})^*$ an expression of 252 tokens	$a_1 ? a_2 a_3 ? a_4 ? (a_5^+ + ((a_6 + \cdots + a_{61})^+ a_5^*))$ $a_1 ? a_2 a_3 ? a_4 ? (a_6 + \cdots + a_{61})^* a_5^*$ $a_1 ? a_2 a_3 ? a_4 ? (a_6 + \cdots + a_{61})^* a_5^*$ an expression of 185 tokens
$a_1 ? (a_2 a_3 ?) ? (a_4 + \cdots + a_{44})^* a_{45}^+$ $a_1 ? a_2 ? a_3 ? (a_4 + \cdots + a_{44})^* a_{45}^+$ $a_1 ? (a_2 a_3 ?) ? (a_4 + \cdots + a_{44})^* a_{45}^+$ an expression of 142 tokens	$a_1 (a_2 + a_3)^* (a_4 (a_2 + a_3 + a_5)^*)^*$ $a_1 (a_2 + a_3 + a_4 + a_5)^*$ $a_1 ((a_2 + a_3 + a_4)^+ a_5^*)^*$ an expression of 85 tokens

Table 1: Results on real world DTDs: original DTD, inferred DTDs by CRX, iDTD and XTRACT.

References

- [1] Garofalakis, Gionis, Rastogi, Seshadri, Shim. XTRACT: learning document type descriptors from XML document collections. *Data mining and knowledge discovery*, 7:23–56, 2003.
- [2] Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
- [3] Martens, Neven, Schwentick, Bex. Expressiveness and Complexity of XML Schema. *ACM TODS*, 31(3), 2006.