jInfer XML Schema Inference Framework

Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, Matej Vitásek
Advisors: RNDr. Irena Mlýnková, Ph.D., Martin Nečaský, Ph.D.

Praha, 2011

Abstract

At the present day, there are many algorithms solving the XML schema inference problem. However, none of them is widely used by the public to practically solve the problem. This is partly because of the lack of a simple user interface to these algorithms. We present a NetBeans ([**?**]) based pluggable framework to fill this gap. This way we enable people to deal with XML in practice, to obtain schema for their documents by following simple steps, without need to read and understand theoretical aspects. We hope our solution will boost usage of XML schemas in practice, since creating schema for existing set of documents will be more affordable (one doesn't have to write one by hand). We suggest framework will be used as a experimental sandbox for scientific developers, willing to test their own new algorithms without need to develop complicated GUI.

Introduction

Although many algorithms (for example [Aho96, BNST06, BNV07, VMP08, Vyh]) try to solve the problem of schema inference for an existing set of XML documents, have you ever tried to solve the problem in practice?

When the problem arises, one wants to find a solution as quickly as possible, without the need to investigate many scientific papers. We focus on this group of potential users - programmers, system administrators, XML coders, XML maintaners in private and R & D.

Framework is designed to be extensible and is implemented as a set of modules for NetBeans platform. This means a second group of users (although much smaller in counts) of our framework - researchers willing to experiment with their new algorithms may easily gain benefits of user interface provided, thus speeding up their development, easing them comparison with other algorithms already developed for framework.

But it is not all about user interface. We provide XML/XSD/DTD import/export modules ready to use. We implement a sample inference algorithm (see [Aho96]). We provide visualization tools to display input XML documents as a set of grammar rules, and tools to visualize non-deterministic finite automatons used in many of the inference algorithms. This way, extending existing NFA base algorithms with user interaction is easier for researchers.

Related work

Most of the algorithms named, doesn't have avalaible implementation, nor are made ready to use. Probably best solution nowadays is [BNV08]. It deals with schema infering and schema evolution as well. It employs user interaction, schema visualization and user-aided schema refinement to obtain better results. Authors use their own algorithms for automatic schema inference ([BNST06, BNV07]) and then present user generated schema with option of GUI refining it. Schema refinement and visualization are strong sides of this solution and also possibility to do schema evolution (given old schema and not-all-valid XML documents, update schema) is unique in the field. Since we both, are trying to solve nearly same problem, we predict that we will approach some good ideas from other solution in further work. There is no mention of extensibility and pluggable design, however, which we solve as our top priority.

Short architecture overview

We divide the inference process into three steps (as illustrated on fig. 1):

1. Import of various formats (XML/XSD/DTD/XPath) into internal representation - *Initial Grammar*.
2. Simlification (generalization) of *Initial Grammar* with optional help of user interaction.
3. Export into various schema formats (XSD/DTD).

Our internal representation consist of regular grammar, that is list of rules. We represent right side of grammar rule as regular expression with our own set of classes to represent regular expressions.

On input, all various formats are first converted into internal representation - from XML rules are extracted as positive examples, concatenations, from XSD/DTD rules

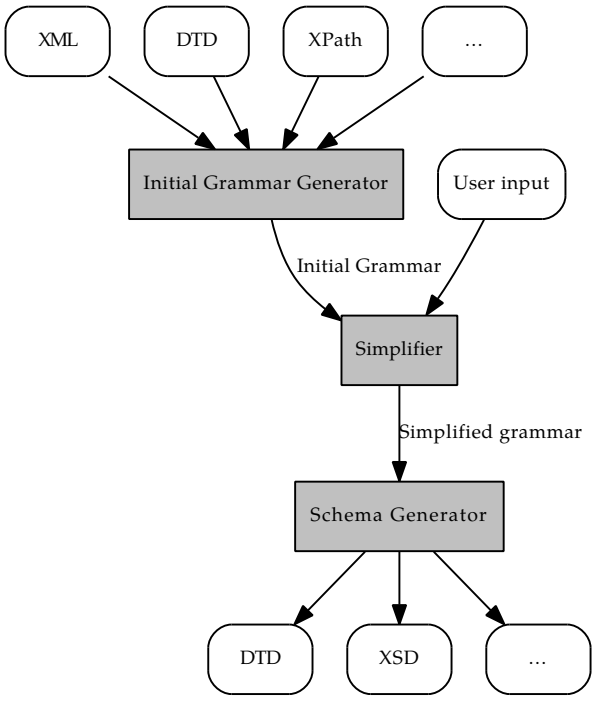

Figure 1: High-level view of the inference process

are extracted as ther are written in schemata. From XPath we extract only explicit concatenations for now.

Rules are send to *Simplifier* module, which has to do the simplification algorithm and return *Simplified Grammar*, which is then easily exporter by exporter modules.

Simplification can be either automatic, or user-aided. We provide visualization of non-deterministic finite automatons. This helps algorithms that use merging state algorithm with user interaction. Using simple API, algorithm developer can call our library to present automaton to user to make human decisions and continue in simplification process. See figure 2 for automaton visualization tool.

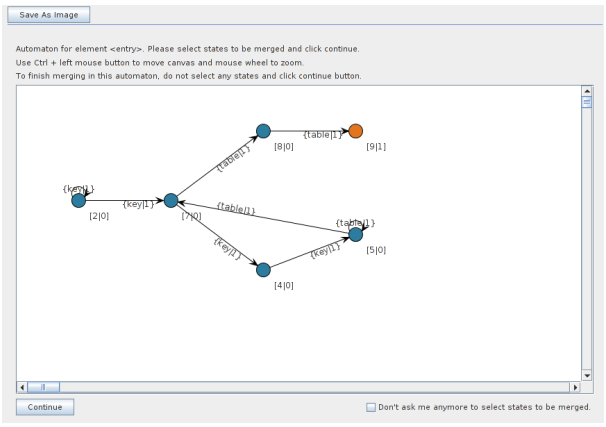

Figure 2: Visualization of automaton in inference process

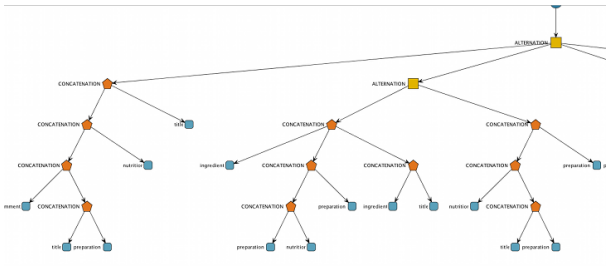

Figure 3: Visualization of grammar rule

We provide visualization of rules itself, which is useful for methods not using automatons at all. Algorithm can use API to display rules to user in progress (see fig. 3).

Extensibility

Developers willing to experiment with their own algorithms have to take care only of *Simplifier* module. We provide sample one, which is divided into submodules again, and again and again. One can find exact place he wants to replace functionality, and replace only that part of simplification chain. It then remains configurable whether to use his new experimental module or the old one, from GUI. He doesn't need to code into jInfer sources, keeping his software installed in same NB instance as jInfer is enough.

As we implement [Aho96], we provide good support for merging state algorithms - automaton visualization for example. We divide merging of automaton states in two tasks: merging strategy and merging criterion. Former one is responsible of deciding whether to merge two states or not. It has to decide where to stop merging and work is done, it uses merging criterion to test two states equivalency and can measure automaton as it wants to decide whether to continue or stop process. Merging criterion (we implement k, h-context) can be replaced and plugged into merging strategy. One can implement for example s, k-strings ([GGR+00]).

We provide replacable parts in automaton to regular expression conversion problem, so one can replace our state removal method (with simple weighted heuristic).

Possibilities are open.

Open issues

There are still many issues to be worked on. First is set of supported input/output formats. On input, one can imagine XML Queries, on output Relax NG or Schematron. There is room for implementing schema visualization tool into jInfer, to enable user-aided schema refinement (to approach [BNV08]).

For now, we ignore content models of attribute and text node values, but there is interface available to be imple-

mented. So simple data type detection is waiting for its imlementor.

Many algorithms published wait to be implemented as jInfer modules, maybe some of them will be tried in study of our master theses.

Conclusion

Framework presented aims to be the tool used to obtain schemas from XML/XSD/DTD files (and more inputs after extending). While maintaining distributable bundle for users, with best of module selections and algorithms implemented so far, we predict it will be used in future studies of new algorithms as a test environment. Best of them will probably migrate into jInfer source tree to aid common users to solve their document set quickly enough, to even bother with schema generation at all.

Weak side of the solution is the lack of good quality automatic or semi-automatic algorithms to be used by common user and lack of content model inference. Goal was to produce stable framework for future development and we hope to negate this issues by implementing master theses in this environment, providing it with bundle of different algorithms, solving different input and output formats and infering more concise schemas.

References

[Aho96] H. Ahonen. *Generating grammars for structured documents using grammatical inference methods*. PhD thesis, Department of Computer Science, University of Helsinki, Series of Publications A, Report A-1996-4, 1996.

[BNST06] Geert Jan Bex, Frank Neven, Thomas Schwentick, and Karl Tuyls. Inference of concise dtds from xml data. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 115–126. VLDB Endowment, 2006.

[BNV07] Geert Jan Bex, Frank Neven, and Stijn Vansummeren. Inferring xml schema definitions from xml data. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 998–1009. VLDB Endowment, 2007.

[BNV08] Geert Jan Bex, Frank Neven, and Stijn Vansummeren. Schemascope: a system for inferring and cleaning xml schemas. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1259–1262, New York, NY, USA, 2008. ACM.

[GGR+00] Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. Xtract: a system for extracting document type descriptors from xml documents. *SIGMOD Rec.*, 29:165–176, May 2000.

[VMP08] Ondřej Vošta, Irena Mlýnková, and Jaroslav Pokorný. Even an ant can create an xsd. In *DASFAA'08: Proceedings of the 13th international conference on Database systems for advanced applications*, pages 35–50, Berlin, Heidelberg, 2008. Springer-Verlag.

[Vyh] Julie Vyhnanovská. Automatic construction of an xml schema for a given set of xml documents.