Igel: Comparing document grammars using XQuery

C. M. Sperberg-McQueen
Black Mesa Technologies LLC
<cmsmcq@blackmesatech.com>

Oliver Schonefeld
Institut für Deutsche Sprache (IDS)
<schonefeld@ids-mannheim.de>

Marc Kupietz
Institut für Deutsche Sprache (IDS)
<kupietz@ids-mannheim.de>

Harald Lüngen
Institut für Deutsche Sprache (IDS)
<luengen@ids-mannheim.de>

Andreas Witt
Institut für Deutsche Sprache (IDS)
<witt@ids-mannheim.de>

How to cite this paper

Sperberg-McQueen, C. M., Oliver Schonefeld, Marc Kupietz, Harald Lüngen and Andreas Witt. “Igel: Comparing document grammars using XQuery.” Presented at Balisage: The Markup Conference 2013, Montréal, Canada, August 6 - 9, 2013. In *Proceedings of Balisage: The Markup Conference 2013*. Balisage Series on Markup Technologies, vol. 10 (2013). doi:10.4242/BalisageVol10.Schonefeld01.

Abstract

Igel is a small XQuery-based web application for examining a collection of document grammars; in particular, for comparing related document grammars to get a better overview of their differences and similarities. In its initial form, Igel reads only DTDs and provides only simple lists of constructs in them (elements, attributes, notations, parameter entities). Our continuing work is aimed at making Igel provide more sophisticated and useful information about document grammars and building the application into a useful tool for the analysis (and the maintenance!) of families of related document grammars.

Table of Contents

Introduction

Igel is a small XQuery-based web application for examining a collection of document grammars; in particular, for comparing related document grammars to get a better overview of their differences and similarities. Igel is a product of collaboration between the Institut für Deutsche Sprache (IDS) and Black Mesa Technologies. (The I and the G of the name can be thought of as standing for "IDS" and "grammar", and the E and L were added to make the German word Igel 'hedgehog'.)

Requirements

IDS creates, maintains, and exploits a number of corpora of modern German, including DeReKo (Deutsches Referenzkorpus, Kupietz et al., 2010), the German Reference Corpus, about 6.1 billion running words of text[1] from many different publishers and including documents from 1964 to the present. Over the years, IDS has developed and used a number of formats for representing corpora, including a local customization of XCES the XML Corpus Encoding Standard, itself a customization for corpus-linguistic work of the Text Encoding Initiative's TEI P3 and TEI P4 tag sets) and most recently I5, a reformulation of their existing vocabulary on the basis of TEI P5. The migration process to I5 is described in Lüngen & Sperberg-McQueen, 2012.

In the context of collaborations with other institutions which are also using variants of TEI P5, IDS has a requirement to develop a sound understanding of how the different variants of TEI P5 relate to each other. Igel is designed to provide accessible tools for answering this question, from the simple (what elements in vocabulary *A* are missing from vocabulary *B*? what elements do they have in common?) to the more complex (which parts of content model *C* are actually needed to cover the data in collection *D*?).

Design and current status of Igel tools

The main work of Igel is done by a collection of XQuery functions which operate on an XML representation of the document grammars involved. Given a suitable XML representation, it is very easy to write XQuery expressions to list all the elements and attributes declared in a given document grammar or to display the definition of a common element in several grammars.

Functionality

The current version of Igel has only basic functionality:

- From a control panel, an administrator can supply the URI of a document grammar and have it added to the collection.
- From a list of document grammars in the collection, the user can select one or more for closer examination and comparison.
- For any selection of document grammars, the user can get a list of declared element types, attributes, notations, entities, parameter entities, etc., showing which constructs are shared and which are unique to individual grammars. This makes it easy to see the union, intersection, and set difference between the lists of constructs in different grammars.
- For any given element, attribute, etc., the user can display the declarations for that construct from all selected grammars. This allows convenient examination of the declarations by a knowledgeable human. Parameter entities are expanded in place with appropriate labels at the beginning and end of each entity reference; this allows both the form of the declaration and the effective declaration to be examined.

For additional functionality not currently present but which we plan to add, see below.

XML representation of document grammars

The basic requirement for studying document grammars using XQuery is that we must have an XML representation for the document grammars.

DTDs, as is well known, do not use XML but use a non-XML notation inherited from SGML and ISO 8879. There are a number of XML representations for DTDs, like NekoDTD (Clark, n.d.), SWI-Prolog (Wielemaker, n.d.) or DTDinst (Clark, 2002) , but most interesting production DTDs make extensive use of parameter entities to simplify understanding and maintenance of the document grammar and none of the commonly used XML representations of DTDs provide the necessary access both to the parameter-entity structure of the grammar and to the expanded declarations. Wayne Wohler's DTD4DTD (Wohler, 1994) does provide access to both physical and logical structures, but goes rather far in the opposite direction: its element structure is so directly modeled on the form of the grammar in ISO 8879 (down to elements to contain the whitespace in the original) that it is not particularly convenient to process. It is also rather difficult to produce for a parser which has restructured the grammar in order to eliminate the many reduce/reduce conflicts and ambiguities in the grammar.

Some schema languages commonly used today (e.g. XSD and Relax NG) do have standard XML representations, but these representations are also not well suited to the kind of grammatical comparisons Igel is designed to support. The declaration of an element, for example, may refer to a complex type or a pattern declared elsewhere in the schema and may thus not be interpretable in isolation. The chains of dependencies can be arbitrarily long. The user of Igel, however, has the same requirement for these document grammars as for DTDs: both the expanded structure of the declaration and the maintenance-oriented short-hands need to be accessible.

So for the moment, at least, Igel uses an idiosyncratic XML representation derived from the first author's unpublished work on SGML representations of DTDs in the early 1990s. A yacc/lex parser named dpp (for “DTD pre-processor”) parses DTDs and produces XML output in which parameter-entity boundaries are made visible as milestone elements. XSLT stylesheets can be used to transform this representation of the DTD into another in which parameter entities are represented as nesting elements, but in practice those stylesheets have proven to be unnecessary most of the time in practice; many operations are slightly simpler when parameter entities are visible as milestones and not as normal nesting elements.

The dpp parser is not an integral part of Igel; instead, dpp is made available as a service on an HTTP server, and Igel loads DTDs by calling that service with appropriate arguments. The dpp service then returns the XML representation of the DTD, which Igel loads into the XML database.

A current limitation of dpp is that it does not allow parameter-entity boundaries in every location where they are legal according to ISO 8879 and the XML specification, because the DTD grammar as written in ISO 8879 has a large number of reduce/reduce conflicts. So support for explicit entity boundaries in dpp has been developed ad hoc, by tweaking the grammar until dpp succeeded in parsing a number of well known DTDs with extensive use of parameter entities (specifically the SGML DTD of ISO 8879 Annex E, HTML 4.01, XHTML 1.0, the NLM Journal Archiving Tag Set, TEI P3, and TEI P4).

A current limitation of the XML representation of document grammars used is that it is not yet documented. Since the XML representation is only used internally and is not exposed to the user, the lack of documentation is a burden primarily for those responsible for maintaining and extending Igel and

related tools.

Implementation

The initial implementation of Igel used Sausalito, an open-source XQuery engine adapted from Zorba by the commercial firm 28msec, and deployed on 28msec's free public demonstration server my28msec.com, at the address http://igel.my28msec.com/.

Sausalito, however, is now being retired as a commercial product by 28msec, so Igel has been reformulated as a BaseX application using the BaseX http interface. A conventional Apache web server handles security and communication with the user. PHP modules accept user input, translate it into calls on BaseX functions, and direct the BaseX output to the user.

At the time this document was written, the current version of Igel was on the web at http://clarin.ids-mannheim.de/igel/.

Future work

Several important parts of Igel remain to be implemented; some of these will, we hope, be completed and available for demonstration in Montreal.

- For a given document grammar, list elements declared but not used elsewhere and elements used (referred to) but not declared.
- For a given element in a given set of document grammars, list ancestors, parents, children, and descendants.
- For a given parameter entity in a given set of document grammars, list other parameter entities this one refers to directly or indirectly; list other parameters which refer directly or indirectly to this one.
- For a given element or parameter entity in a given document grammar, list other elements and/or parameter entities which "compete" against this one in the sense of XSD 1.1. (For any *X* and *Y*, *X* and *Y* compete against each other in content-model expressions like `(X | Y)`, `(X*, Y)`, `(Y+, X)`, and `(X?, Z*, Y)`. If *X* and *Y* are parameter entities denoting classes of elements and compete against each other, adding the same element to both *X* and *Y* will result in a determinism error in a DTD, or a unique-particle attribution error in XSD.)
- From a content model, generate a drawing of a finite-state automaton that accepts the language defined by the model; display in the user's web browser.

  This task (and the others involving drawing diagrams) is easy with an XQuery engine that has an interface to GraphViz [2]; Sausalito has such an interface, which allows an XQuery application to generate diagrams by creating an XML representation of the GraphViz input format, calling an extension function, and receiving SVG back. BaseX, however, does not have a native GraphViz interface, so we must find another way to generate our FSA diagrams. Recent developments within the BaseX community look promising, but we have not yet investigated them for use within Igel.[3]
- For a given set of elements in a given set of document grammars, read a body of XML documents and report on the frequency with which different parts of the content models are used. That is, provide information on the degree to which the document grammar overgenerates by allowing structures not actually present in the sample data.

  Draw a weighted FSA for the content models in question, color-coding frequently used, rarely used, and unused arcs and nodes.
- For any two content models X and Y, test whether X is a subset of superset of Y; if not, generate expressions describing the intersection of X and Y, the difference of X and Y, and the difference of Y and X.

  Draw FSAs for all expressions involved.

- Support XSD and Relax NG schemas. The document grammars of most pressing interest to IDS currently are DTD-based, so that is where the effort on Igel has been concentrated. In the long run, however, it would be good to make Igel handle XSD and Relax NG schemas.
- For a given notation, list the elements or attributes in the selected document grammars which use that notation.
- For any given attribute name, list the elements it's declared for and the type it has for each element.
- For any given value, list all attributes or elements which have that value as their default.
- For a given document grammar, draw a parent/child graph for all elements.
- For a given document grammar, draw a reference graph for all parameter entities.
- Provide alternate deployment options for Igel.

  Some projects are already committed to another XQuery processor; provide a deployable Igel package for eXist (and optionally other XQuery processors).

  It's convenient to have a web interface for Igel, but not all projects will find it convenient to configure a Web server to handle the user interface. Provide an Oxygen-based interface to Igel.

Bibliography

[Kupietz et al., 2010] Kupietz, Marc / Belica, Cyril / Keibel, Holger / Witt, Andreas (2010): *The German Reference Corpus DeReKo: A primordial sample for linguistic research*. In: Calzolari, Nicoletta et al. (eds.): Proceedings of the 7th conference on International Language Resources and Evaluation (LREC 2010). Valletta, Malta: European Language Resources Association (ELRA), pp. 1848–1854.

[Lüngen & Sperberg-McQueen, 2012] Harald Lüngen, and C. M. Sperberg-McQueen (2012): *A TEI P5 Document Grammar for the IDS Text Model*. In: Journal of the Text Encoding Initiative (2012), H. 3. http://jtei.revues.org/508

[Wielemaker, n.d.] Jan Wielemaker (n.d.): *SWI-Prolog SGML/XML parser*. SWI-Prolog. http://www.swi-prolog.org/pldoc/package/sgml.html.

[Clark, n.d.] Andy Clark (n.d.): *CyberNeko DTD Converter*. http://people.apache.org/~andyc/neko/doc/dtd/index.html

[Clark, 2002] James Clark (2002): *DTDinst*. http://www.thaiopensource.com/dtdinst/

[Wohler, 1994] Wayne L. Wohler (1994): *DTDecl Document Type Declaration* . http://xml.coverpages.org/wohlerDTDDTD.txt

[Murata et al., 2005] Makoto Murata, Dongwon Lee, Murali Mani, and Kohsuke Kawaguchi (2005). *Taxonomy of XML Schema Languages Using Formal Language Theory*. ACM Transactions on Internet Technology, 5(4):660–704. doi:10.1145/1111627.1111631.

---

[1] As of 2013-03-19; see http://www.ids-mannheim.de/kl/projekte/korpora/ for an overview.

[2] http://www.graphviz.org/

[3] Andy Bunce gave a presentation at the BaseX user group meeting at XMLprage 2012 and reported on “Adventures with BaseX and web applications” (http://files.basex.org/xmlprague2013/slides/Andy-Bunce-BaseX-User-Group-Talk.pdf), which, among others, featured a GraphVis integration (https://github.com/apb2006/graphxq).

C. M. Sperberg-McQueen
<cmsmcq@blackmesatech.com>
Black Mesa Technologies LLC

C. M. Sperberg-McQueen is the founder of Black Mesa Technologies LLC, a consultancy specializing in the use of descriptive markup to help memory institutions preserve cultural heritage information for the

long haul. He has served as co-editor of the XML 1.0 specification, the Guidelines of the Text Encoding Initiative, and the XML Schema Definition Language (XSD) 1.1 specification. He holds a doctorate in comparative literature.

Oliver Schonefeld

<schonefeld@ids-mannheim.de>
Institut für Deutsche Sprache (IDS)

Oliver Schonefeld works at the Institut für Deutsche Sprache (Institute for the German Language) in Mannheim and is involved in the projects CLARIN and TextGrid. He studied computer science with specialization in text technology at Bielefeld University until 2005. After graduating he worked as a researcher at Bielefeld University and later at Tübingen University's collaborative research center Linguistic Data Structures. His major research interests are the limitations of markup languages (especially overlapping markup) and the use of markup languages in linguistic description of language data.

Harald Lüngen

<luengen@ids-mannheim.de>
Institut für Deutsche Sprache (IDS)

Harald Lüngen has been a researcher in the area of corpus linguistics at the Institut für Deutsche Sprache (Institute for the German Language) in Mannheim, Germany, since 2011, specialising in the construction and maintenance of the German Reference Corpus DeReKo and in methods of corpus analysis. He was a researcher at the Centre for Media and Interactivity at Justus-Liebig-Universität Gießen (2008–2011), in the Research group Text-technological modelling of information (2002–2008) and in the Verbmobil project (1995–2000). He also worked as a project manager at Lingsoft Oy, Helsinki (2000–2002). He got his Ph.D. in linguistics from Bielefeld University in 2002.

Andreas Witt

<witt@ids-mannheim.de>
Institut für Deutsche Sprache (IDS)

Andreas Witt received his Ph.D. in Computational Linguistics and Text Technology from the Bielefeld University in 2002 (dissertation title: “Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie”). After graduating in 1996, he started as a researcher and instructor in Computational Linguistics and Text Technology. He was heavily involved in the establishment of the minor subject Text Technology in Bielefeld University's Magister and B.A. program in 1999 and 2002 respectively. After his Ph.D. in 2002 he became an assistant lecturer, still at the Text Technology group in Bielefeld. In 2006 he moved to Tübingen University, where he was involved in a project on “Sustainability of Linguistic Resources” and in projects on the interoperability of language data. Since 2009 he is senior researcher at Institut für Deutsche Sprache (Institute for the German Language) in Mannheim. Witt is and was a member of several research organizations, amongst them the TEI Special Interest Group on overlapping markup, for which he was involved in the writing of the latest version of the chapter “Multiple Hierarchies”, which is included in TEI-Guidelines P5.