LIXR: Quick, succinct conversion of XML to RDF

John P. McCrae[1] and Philipp Cimiano[2]

[1]Insight Centre for Data Analytics, National University of Ireland, Galway
john@mccr.ae
[2]Cognitive Interaction Technology, Cluster of Excellence, Bielefeld University
cimiano@cit-ec.uni-bielefeld.de

Abstract. This paper presents LIXR, a system for converting between RDF and XML. LIXR is based on domain-specific language embedded into the Scala programming language. It supports the definition of transformations of datasets from RDF to XML in a declarative fashion, while still maintaining the flexibility of a full programming language environment. We directly compare this system to other systems programmed in Java and XSLT and show that the LIXR implementations are significantly shorter in terms of lines of code, in addition to being conceptually simple to understand.

Keywords: RDF, XML, Scala, format conversion

1 Introduction

An important aspect towards realizing a web of data is the conversion of legacy resources into the Resource Description Framework [2, RDF]. There are tools and W3C recommendations[1] supporting the transformation of relational databases into RDF (e.g. D2RQ [1]), and even declaratively languages such as R2RML[2]. Besides the relational model, legacy data represented in XML is quite frequent. While there exist generic mechanisms for transforming XML data into some other form, such as Extensible Stylesheet Language Transformations (XSLT), these mechanisms are not ideal for the conversion into RDF for the following reasons:

1. The generated RDF typically contains more triples than necessary due to the fact that generic converters create both a property and a node for each individual element in the XML.
2. It is uncommon for XML documents to reuse URLs from other resources. For example it is typical for a resource to reuse the data categories of Dublin Core [7], but to recast them under a new namespace, that is not compatible with RDF.

[1] http://www.w3.org/TR/rdb-direct-mapping/
[2] http://www.w3.org/TR/r2rml/

3. XML provides no generic mechanisms for the representation of external links by URIs, using a proprietary linking schema instead such as XLink [3].

Further, transformation from XML to some other format are typically specified by means of XSLT, an extension of XSLT such as Krextor [5], a specific mapping language such as RML [4] or by writing a short script in some programming language. Thus, these transformations are generally very verbose, as they must repeat many standard RDF modelling structures, and unidirectional, as they are not well-formulated to cope with the polymorphic nature of RDF.

In order to meet these shortcomings, we developed a new system for specifying the translation of XML documents into RDF and *vica versa*, which we call the Lightweight Invertible XML and RDF language (LIXR, pronounced 'elixir'). LIXR is significantly more compact than existing systems and allows for transformation in both the direction of RDF to XML and from XML to RDF.

2 The LIXR Language

The LIXR language was created as a domain-specific language based on the Scala Language. This choice was made as Scala has an exceptional amount of freedom in expression, allowing us to compactly and clearly state transformations, although there would be some learning curve for those not familiar with the Scala language.

The basic structure of LIXR is inspired by XSLT and is based around *handlers*, which describe the action that should be taken when a specific XML element is encountered. These handlers are stated by linking an XML element name to a list of *generators* with the `-->` operator. For example the following LIXR expression can be used:

```
xml.language --> (
  dc.language > content
)
```

This associates the XML element `<xml:language>` to generating the triple *s* `dc:language "`*c*`"`, where *s* is the current subject node, and *c* is the text content of the node[3] More typically the converter works by means of two features: firstly, `nodes` instruct the RDF generation to create a new node in the RDF graph and use it as subject for all triples from this point in the generation. Secondly, the `handle` tells the XML parser to look for all children matching a given element and call the appropriate handler for each matching case. For example:

```
xml.metadata --> (
  node("http://.../metadata")(
```

[3] Note that for technical reasons the `.` is used to conjoin the namespace to the local name instead of the customary `:`. This, in fact, is a dynamic call, a relatively recent feature of the Scala language.

```
    handle(xml.language),
    handle(xml.source)
  )
)
```

This code asks the RDF generator to create a new root node with the given URI. The parser then looks for matching children and calls the appropriate handlers (such as in the first example). More detail of the language can be found on Github[4].

Name	Tags	Implementation	LoC	LoC/Tag
TBX	48	Java	2,752	57.33
CMDI	79	XSLT	404	5.11
CMDI	79	XSLT (No closing tags)	255	3.22
CMDI	79	XSLT (Using Krextor [5])	454	5.75
CMDI	79	RML [4]	339	4.29
CMDI	79	LIXR	176	2.23
TBX	48	LIXR	197	4.10
MetaShare	730	LIXR	2,487	3.41

Table 1. Comparison of XML to RDF mapping implementations, by number of elements in XML schema, and non-trivial lines of code (LoC)

3 Evaluation

To evaluate the effectiveness of our approach we compared directly with four other XML to RDF transformations in terms of an objective measure, that is *lines of code*. The other transformation programs were written by the lead author of the project[5], and reimplemented using the LIXR language. In particular, *lines of code* is easily measured and it has been claimed [6] that the average number of errors made per lines of code is approximately constant for a given programmer, regardless of what language he or she is programming in. As such, lines of code can be a good proxy not only for ease of development but also for software quality. As such, we measure the code in terms of *non-trivial* lines of code, where a line of code is considered trivial if it only contains closing brackets or braces or is empty.

We consider three existing XML schemas as targets to be converted into RDF: the TermBase eXchange format (TBX, ISO 30042:2008), the META-SHARE schema[6] for representing very rich metadata about language resource, and the Component Metadata Initiative (CMDI) used by CLARIN[7].

[4] `https://github.com/liderproject/lixr`
[5] The lead author has over 5 years experience in all languages
[6] `http://metashare.ilsp.gr/META-XMLSchema/v3.0/`
[7] `http://catalog.clarin.eu/ds/ComponentRegistry/rest/registry/profiles/clarin.eu:cr1:p\_1288172614026/xsd`

The results of the comparison in terms of lines of code for the datasets and the different transformations is given in Table 1. The results show that LIXR leads to significantly shorter code in terms of lines-of-code than the other methods we attempted. In fact, we observe a 10-fold reduction of effort over directly writing a converter in a general purpose programming language (Java) and we see a halving of effort in comparison to using a specialist language (XSLT, Krextor or RML). In addition, we note that the reduction is such that only a few lines of code are needed for each element class. We note that all of the systems ran quickly over the data we tested and thus we do not believe that processing time or memory are radically different between these implementations.

4 Conclusion

We have presented a declarative yet flexible approach supporting the conversation of XML to RDF. The approach is based on a domain-specific language embedded in the Scala programming language. We have shown that this converter supports the implementation of concise and shorter conversion programs than with other transformation languages.

In addition to the reduction in effort using this approach, we also note several other advantages of the LIXR approach that could easily be added, due to its declarative nature, including stream processing of XML, reverse mapping from RDF to XML and extraction of an ontology from the mapping.

Acknowledgments

References

1. Bizer, C., Seaborne, A.: D2RQ-treating non-RDF databases as virtual RDF graphs. In: Proceedings of the 3rd international semantic web conference (ISWC2004). vol. 2004 (2004)
2. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 concepts and abstract syntax. W3C recommendation, World Wide Web Consortium (2014)
3. DeRose, S., Maler, E., Orchard, D.: XML linking language (XLink) version 1.0. W3C recommendation, World Wide Web Consortium (2001)
4. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: A generic language for integrated RDF mappings of heterogeneous data. In: Linked Data on the Web (2014)
5. Lange, C.: Krextor–an extensible XML $\to$ RDF extraction framework. Scripting and Development for the Semantic Web (449), 38 (2009)
6. McConnell, S.: Code Complete: A Practical Handbook of Software Construction, Second Edition. Microsoft Press (2014)
7. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin core metadata for resource discovery. Request for Comments 2413 (198)