Ademar Crotti Junior, Christophe Debruyne, and Declan O'Sullivan. Incorporating functions in mappings to facilitate the uplift of CSV files into RDF. In H. Sack, G. Rizzo, N. Steinmetz, D. Mladenic, S. Auer, and C. Lange, editors, The Semantic Web - ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers, volume 9989 of Lecture Notes in Computer Science, pages 55–59, 2016

Incorporating Functions in Mappings to Facilitate the Uplift of CSV Files into RDF

Ademar Crotti Junior, Christophe Debruyne, Declan O'Sullivan

ADAPT Centre for Digital Content Platform Research, Knowledge & Data Engineering Group, School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, Ireland
{crottija,debruync,declan.osullivan}@scss.tcd.ie

Abstract. Many solutions have been developed to convert data to RDF. A common task during this conversion is applying data manipulation functions to obtain the desired output. Depending on the data format, one can rely on the underlying technology, such as RDBMS for relational databases or XQuery for XML, to manipulate – to a certain extent – the data while generating RDF. For CSV files, however, there is no such underlying technology. One has to resort to pre- or post-processing techniques when data manipulation is needed, which renders the process of generating RDF more complex (in terms of number of steps), and therefore also less traceable and transparent. Another solution is to declare functions in mappings. KR2RML provides data manipulation functions as part of the mapping, but due to its complex format, it is difficult to create or maintain mappings without their editor. In this paper, we propose a method to incorporate functions into mapping languages in a more amenable way.

Keywords. Linked Data; Mapping; Data Manipulation.

1 Introduction

The CSV file format is a convenient and popular way to exchange data, but provides no support for capturing semantics; i.e., the semantics of the information captured in such files are not explicit. RDF, the standard data model for the Semantic Web, provides one means to describe, annotate and exchange information in meaningful ways such that machines can process them [1]. The process of converting data in any format (XML, CSV, stored in databases) into RDF is called *uplift*. Many applications have been developed to support uplift. For CSV files, the state-of-the-art includes RML [2], SML [3] and KR2RML [8] – which we will discuss in Section 2.

In many cases, however, it is necessary to manipulate data during the uplift process. Depending on the data format, this can be very straightforward. With uplift languages for relational databases such as R2RML [6], for example, one can rely on the underlying RDBMS to support some data manipulation tasks. The same is true for XSPARQL [5], where one can use XQuery – on which it is built upon – to manipulate the data contained in XML. In some cases, however, relying on the underlying technology might not be sufficient [4]. For CSV datasets there is no such underlying technology. If data has to be manipulated, one needs to resort to applying data manipulation functions as a pre- or post-processing step. This increases complexity, and ren-

ders the whole data processing "pipeline" less transparent. A better solution is to capture these functions in mappings declaring how RDF is generated from CSV data.

We propose a method to incorporate functions into a mapping language in a more amenable way. Our method draws inspiration and generalizes ideas presented in [4]. To demonstrate our method, we extend RML's vocabulary and engine to include notions for function calls and parameter bindings. The main contributions of this paper can be summarized as follows: i) a method to incorporate functions in a mapping language; ii) an implementation of the method extending RML; and iii) a demonstration of functions incorporated into mappings applied to a real world dataset.

2 Related Work

Many applications have been developed to support the uplift process to RDF. Some applications use annotation languages, examples include R2RML and R2RML-F [4] for relational databases; SML for relational databases and CSV; RML and KR2RML [8] for an even wider array of data formats.

The general approach when data manipulation is required is the use of pre- or post-processing techniques. This adds complexity and makes the process less traceable and transparent. KR2RML is the only tool to support data manipulation functions inside a mapping language (not relying on the underlying technology). Though they provide an editor in which you can load data and input mappings to create functions in Python to manipulate that data, once those functions are stored several problems can be observed. First, a lot of the structured information containing the function is captured as a string. This thus requires both parsing the file and that string. Secondly, the mapping becomes rather complex, which makes it more difficult for users to create similar mappings with other tools. Their editor, however, does facilitate the mapping creation process for their mapping language. In [4] an extension to R2RML called R2RML-F is proposed. R2RML-F adds supports for capturing domain knowledge inside the mapping language for relational databases. Unlike, KR2RML, functions in R2RML-F are resources related to a Literal containing the function. These functions are part of the same document and are thus treated as one RDF file. We will adopt the ideas presented in [4] to develop a more generic, usable and amenable approach to incorporate functions into mapping languages.

3 Incorporating Functions into Mapping Languages

In this section we describe how to incorporate functions into mapping languages based on the work presented in [4]. These functions can be used to capture both domain knowledge (e.g., transforming units) and other – more syntactic – data manipulation tasks (e.g., transforming values to create valid URIs). Function names are unique and each function must have one function name and one function body. A function body defines a function with a return statement; parameters are optional.

Our proof-of-concept extends RML's vocabulary and engine[1] by introducing construct for describing functions, function calls and parameter bindings. Listing 1 defines a function. This function has one string as a parameter and returns a URL concatenated with its camel case version. Although this function executes a simple string

[1] Available at `https://github.com/CNGL-repo/RMLProcessor`

transformation, functions in this method are generic and capable of complex data transformations. Furthermore, functions work with any data format and can be reused in the mapping. Listing 2 demonstrates how the function is called.

```
<#Camelize>
 rrf:functionName "camelize" ;
 rrf:functionBody """ function camelize (str) { var camelCaseString =
str.toLowerCase().replace( /[-_]+/g, ' ').replace( /[^\\w\\s]/g, ' ').replace( /
(.)/g, function($1) { return $1.toUpperCase(); }).replace( / /g, '' );
return "http://dacura.cs.tcd.ie/data/seshat/" + camelCaseString; } """ ; .
```

Listing 1: Declaring a function

```
<#Variable>
 rml:logicalSource [ rml:source "data.csv"; rml:referenceFormulation ql:CSV ];
 rr:subjectMap [ rr:termType rr:BlankNode; ];
 rr:predicateObjectMap [
   rr:predicateMap [
     rrf:functionCall [
       rrf:function <#Camelize> ;
       rrf:parameterBindings ( [ rml:reference "Variable" ] );]; ];
  rr:objectMap [ rr:parentTriplesMap <#Value> ] ].
```

Listing 2: Calling a function in a PredicateMap

One notices that the parameters are themselves TermMaps (see [6]). These TermMaps are evaluated before the results are passed as arguments to the function. Any errors loading or executing the function are reported back to the user.

4 Demonstration

The dataset used to demonstrate the incorporation of domain knowledge into a mapping language comes from a project called Seshat: Global History Databank [7]. This project is developing a knowledge base to describe human history. This knowledge base is created by hand via a wiki – where contributors are expected to adhere to certain conventions to structure the facts – and the Seshat dataset is currently made available for analysis as CSV/TSV files by scraping the wiki pages. A current development within the project is to gather the data into an OWL knowledge base. However predicates from the dataset differ from the predicates defined in the OWL ontology that are being used to create an OWL knowledge base. Thus, there is need to develop an approach to transform CSV/TSV values into the ontology predicates. For example, in the dataset one predicate is defined as “Capital”, but in the ontology the predicate is *seshat:capital*. Other examples include “Language” and “Linguistic family”. Although this seems trivial, current mapping languages have no support for such transformations as *part of* the mapping in a reusable and amenable way. Table 1 shows a fragment of the data where the values for the column Variable will be transformed into predicates using the mappings from Listing 1 and Listing 2. The RDF output can be seen in Fig. 1.

Table 1: Excerpt of the CSV file shown as a table.

NGA	Polity	Section	Variable	Value From	Fact Type	Value Note
Latium	ItRomPr	General variables	Capital	Rome	simple	simple
Latium	ItRomPr	General variables	Language	Latin	simple	simple
Latium	ItRomPr	General variables	Linguistic family	Indo-European	simple	simple

```
<http://dacura.cs.tcd.ie/data/seshat/ItRomPr> <http://dacura.cs.tcd.ie/data/seshat#hasVariable> _:d3SRTs6uGh .
_:d3SRTs6uGh <http://dacura.cs.tcd.ie/data/seshat/capital> _:d8fF2wuiam .
_:d8fF2wuiam <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dacura.cs.tcd.ie/data/seshat#NameVariable> .
_:d8fF2wuiam <http://dacura.cs.tcd.ie/data/seshat#definiteDataValue> "Rome" .
<http://dacura.cs.tcd.ie/data/seshat/ItRomPr> <http://dacura.cs.tcd.ie/data/seshat#hasVariable> _:genid344T94t1y4CS .
_:genid344T94t1y4CS <http://dacura.cs.tcd.ie/data/seshat/language> _:qjeWSDRDcd .
_:qjeWSDRDcd <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dacura.cs.tcd.ie/data/seshat#NameVariable> .
_:qjeWSDRDcd <http://dacura.cs.tcd.ie/data/seshat#definiteDataValue> "Latin" .
<http://dacura.cs.tcd.ie/data/seshat/ItRomPr> <http://dacura.cs.tcd.ie/data/seshat#hasVariable> _:genid388sbqxWXT4T .
_:genid388sbqxWXT4T <http://dacura.cs.tcd.ie/data/seshat/linguisticFamily> _:yeEQUyN1yW .
_:yeEQUyN1yW <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dacura.cs.tcd.ie/data/seshat#NameVariable> .
_:yeEQUyN1yW <http://dacura.cs.tcd.ie/data/seshat#definiteDataValue> "Indo-European" .
```

Fig. 1: RDF output

5 Conclusions and Future Work

Most tools to convert data to RDF rely on underlying technology for data manipulation, but there is no manipulation language for CSV datasets. Moreover, when data manipulation is needed for CSV datasets one depends on pre- or post-processing techniques, which renders the process less traceable. Another possible solution is to incorporate functions into mappings, but the state-of-the-art does not offer a manageable way to do so. We tackled this problem by presenting a more amenable method to incorporate functions into mapping languages. We demonstrated our approach by extending RML's vocabulary and engine and applied it on a real world dataset.

Future work includes more use cases and experiments to compare performance and expressiveness of our method and other mapping languages.

Acknowledgements. This study is supported by CNPQ, National Counsel of Technological and Scientific Development – Brazil and by Science Foundation Ireland (Grant 13/RC/2106) as part of the ADAPT Centre for Digital Content Platform Research (http://www.adaptcentre.ie/) at Trinity College Dublin.

6 References

1. Hitzler, P., Krotzsch, M., Rudolph, S.: Foundations of semantic web technologies. CRC Press, (2009).
2. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van deWalle, R.: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Workshop on Linked Data on the Web. (2014).
3. Stadler, C., Unbehauen, J., Westphal, P., Sherif, M.A., Lehmann, J.: Simplified RDB2RDF Mapping. In: Workshop on Linked Data on the Web. (2015).
4. Debruyne, C., O'Sullivan, D.: R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings. In: Workshop on Linked Data on the Web (2016).
5. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. *Journal on Data Semantics* 1(3) 147-185 (2012).
6. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. (2012) https://www.w3.org/TR/r2rml/.
7. Turchin, P., Brennan, R., Currie, T., Feeney, K., Francois, P., Hoyer, D., Manning, J., Marciniak, A., Mullins, D., Palmisano, A., et al.: Seshat: The global history data-bank. Cliodynamics: The Journal of Quantitative History and Cultural Evolution 6 (2015).
8. Slepicka, J., Yin, C., Szekely, P., Knoblock, C.: KR2RML: An alternative interpretation of R2RML for heterogeneous sources. In: Proceedings of the 6th International Workshop on Consuming Linked Data (2015).