R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings

Christophe Debruyne
Trinity College Dublin
Dublin 2, Ireland
debruync@scss.tcd.ie

Declan O'Sullivan
Trinity College Dublin
Dublin 2, Ireland
declan.osullivan@scss.tcd.ie

ABSTRACT

Many initiatives have emerged to aid one in publishing structured resources as Linked Data on the Web with one of the major achievements being the R2RML W3C Recommendation. R2RML and its dialects assume a certain underlying technology (e.g., Core SQL 2008). This means that domain-specific data transformations – such as transforming geospatial coordinates – rely either on that underlying technology or data-preprocessing steps. We argue that one can incorporate and subsequently share that procedural domain knowledge in such mappings. Such an extension would make certain pre-processing steps redundant. One can furthermore attach metadata to these functions, which can be published as well. In this paper, we present R2RML-F, an extension to R2RML, that adopts ECMAScript for capturing domain knowledge and for which we have developed a prototype. We demonstrate the viability of the approach with a demonstration and compare its performance with different mappings in some initial experiments. Our preliminary results suggest that there is little or no overhead with respect to relying on underlying technology.

CCS Concepts

H.4 [Information Systems Applications]: Miscellaneous; H.5 [World Wide Web]: Web data description languages.

Keywords

R2RML; Linked Data; Mapping.

1. INTRODUCTION

Fifteen years ago, Tim Berners-Lee wrote a Design Issue [2] suggesting how to publish the information contained in relational databases on the Semantic Web. Many tools have emerged (see [15] for a survey) to generate RDF from these databases either via *direct mappings* – based on the ideas outlined in [2] with the RDF reflecting the database's structure and labels – or via *annotations* where tables, views and queries are related to concepts and relations in ontologies to create a knowledge base. Both approaches ultimately led to two W3C Recommendations; a direct mapping of relational data to RDF [1] and R2RML [6]. The R2RML specification assumes that the relational database is annotated to conform to the Core SQL 2008 specification.

According to [3], a model management system is "a component that supports the creation, compilation, reuse, evolution, and execution of mappings between schemas represented in a wide range of metamodels." They also argued that mapping languages should be more expressive if one wants to support various use case scenarios and that – provided that tractability is not a problem – one could include construct to support the declaration of user-defined functions (amongst others).

WWW2016 Workshop: Linked Data on the Web (LDOW2016)

There are cases where the underlying database does not support certain data manipulations because they are not expressive enough or because procedural domain knowledge is part of the application layer rather than the database layer. Converting units might be a straightforward conversion that one can formulate in a SQL query, but others are more complex and people have to resort to more complex data processing "pipelines"; preprocessing the data and transforming the result into RDF or transform the data into an RDF graph and manipulate the RDF graph to create a new RDF graph. Kovalenko et al. investigated the latter in [10] and they observed that some techniques (e.g., SPIN [11]) support user-defined functions. Both approaches are, however, still conducted in two steps and that renders the whole process more complex then it should be.

We, however, feel that such procedural knowledge can and should be included in the mapping. Moreover, by incorporating that knowledge in the mapping, one can also share the procedural knowledge and events described, as well as annotate with metadata (e.g., with predicates from PROV-O [12]). The requirements we state for our approach are:

1. Ensure a minimal extension of R2RML that reuses as much as possible existing predicates;
2. Adopt a standardized programming language to represent procedural domain knowledge for which multiple implementations exist.

The result of taking this approach is R2RML-F, where the 'F' stands for "function". R2RML-F adopts ECMAScript[1] as the programming language and we have extended the R2RML vocabulary to include notions for function calls and parameter bindings. Next to these contributions, this paper presents details of the prototype and examples demonstrating the viability of the approach.

The remainder of this paper is organized as follows: Section 2 presents R2RML-F; Section 3 provides some details on the implementation of the prototype; Section 4 is used to demonstrate the ideas as well as to present experiments to evaluate the performance of R2RML-F; Section 5 presents related work; and Section 6 concludes the paper.

2. EXTENDING R2RML

In this section we describe the predicates introduced and our extension of R2RML. We do assume the reader is familiar with R2RML and will introduce the notions of the approach through a simple example. In Listing 1 we declare a function that multiplies

[1] JavaScript is a popular implementation of ECMAScript. Implementations of ECMAScript often extend the language or provide APIs. One example is JavaScript providing APIs to interact with a Web browser.

two arguments[2]:

```
<#Multiply>
 rrf:functionName "multiply" ;
 rrf:functionBody """
  function multiply(var1, var2) {
   return var1 * var2 ;
  }
 """ ;
.
```

Listing 1: Declaring a function in R2RML-F

Each function is a resource that must have two relationships: a *function name* (a literal) and a *function body* (also a literal); the function name and name of the function in the function body must be identical. Each R2RML-F mapping cannot have two functions with the same name and each function must have exactly one function name and exactly one function body. Any errors will be reported to the user.

Listing 2 demonstrates how functions are called from within a Predicate Object Map. We have introduced the new term map *function call*. A term map must now be exactly one of the following: a constant-valued term map, a column-valued term map, a template-valued term map or a function-call-valued term map. A function call:

- must refer to exactly one function; and
- must have at most one list of parameter bindings.

When no list of parameter bindings or an empty list is provided, they are considered to be function calls where no parameters are being passed. The order of the parameters that are passed is important. Adopting the RDF Sequence container was not an option, as RDF does not provide a mechanism to state a container has no more members [5]. The RDF Collection vocabulary of classes and properties can describe a *closed* collection as a *list* [5]. RDF Collections are therefore suitable for passing arguments. The elements in this list – if any – must be term maps. Functions calls can thus be passed the results of columns, constants, templates or another function call.

```
<#TriplesMap1>
 rr:logicalTable [ rr:tableName "Employee"; ];
 rr:subjectMap [
  rr:template "http://org.com/employee/{ID}";
 ] ;
 rr:predicateObjectMap [
  rr:predicate ex:salary ;
  rr:objectMap [
   rr:datatype xsd:double ;
   rrf:functionCall [
    rrf:function <#Multiply> ;
    rrf:parameterBindings (
     [ rr:constant "12"^^xsd:integer ]
     [ rr:column "monthly_salary" ]
    ) ;
   ] ;
  ] ;
 ] ;
.
```

Listing 2: Using function in a Predicate Object Map

3. IMPLEMENTATION

An existing R2RML processor, called db2triples, was extended. The extension is available as a branch on GitHub.[3] Prior to processing the mappings, the R2RML-F processor first looks for and evaluates the functions using Java's Nashorn[4] JavaScript engine. Any problems with the function results in an error that will be sent back to the user.

Function calls are implemented as term maps and each function call has a list of term-maps that constitute its arguments. Each argument is evaluated before the results are passed on to the function being called. This can be compared to the greedy evaluation strategy found in some programming languages. Any runtime errors halt the process and the user is made aware of where the problem resides.

We currently support no monitoring of the functions and rely on Nashorn and the Java Runtime Environment to handle issues concerning memory management and the correctness of code (e.g., infinite loops).

4. DEMONSTRATION AND EXPERIMENT

We demonstrate and evaluate our approach with a use case provided by the Ordnance Survey Ireland (OSi)[5]. The OSi is Ireland's national mapping agency and the geometries in their system are represented using the Irish Transverse Mercator (ITM)[6] coordinate system. At an international level, however, World Geodetic System 84 (or WSG 84)[7] is the standard used in cartography and navigation (amongst others). Transforming coordinates from one coordinate system to another is a common task for which (integrated) tools exist, but not all relational databases support those.

For the demonstration and experiment, we created a table representing the 26 counties of The Republic of Ireland (see Table 1), with the following fields: "id" – an identifier stored as an integer; "name" – the name of the county as a varchar; "geom" – a point of that county as a geometry object (internal representation omitted for this paper); and "geoms" – capturing that same point as a varchar in WKT format.[8] The table is stored in a PostgreSQL database supporting geometries with PostGIS. The purpose of this table is to demonstrate the transformation of the textual representation of points with our approach and compare it with a mapping using PostGIS' functionality.

Table 1: The table "county" in the "boundaries" database

id	name	geom	geoms
10000	CARLOW	▨	POINT(671989.126 676051.233)
20000	CAVAN	▨	POINT(649270.487 796627.36)
30000	CLARE	▨	POINT(532617.573 676288.529)
40000	CORK	▨	POINT(159956.935 79499.879)
50000	DONEGAL	▨	POINT(599999.588 907696.186)
60000	GALWAY	▨	POINT(533393.941 731920.418)
70000	KERRY	▨	POINT(480286.589 603093.557)
80000	KILDARE	▨	POINT(683580.721 713638.655)
90000	KILKENNY	▨	POINT(650826.971 648269.702)
100000	LAOIS	▨	POINT(640275.125 640275.125)
110000	LEITRIM	▨	POINT(600000.431 818660.17)
120000	LIMERICK	▨	POINT(549077.109 638997.81)
130000	LONGFORD	▨	POINT(616521.486 768575.569)
140000	LOUTH	▨	POINT(698735.035 788136.627)
150000	MAYO	▨	POINT(517850.357 795236.186)
160000	MEATH	▨	POINT(688113.609 769372.026)
170000	MONAGHAN	▨	POINT(662572.109 833219.834)
180000	OFFALY	▨	POINT(633368.373 722298.005)
190000	ROSCOMMON	▨	POINT(183556.633 277830.739)
200000	SLIGO	▨	POINT(556553.63 833667.473)
210000	TIPPERARY	▨	POINT(611273.837 657290.162)

[2] http://kdeg.scss.tcd.ie/ns/rrf# is the namespace URI for the R2RML-F predicates and is prefixed with "rrf" in this paper.
[3] Available at https://github.com/CNGL-repo/db2triples
[4] https://blogs.oracle.com/nashorn/

[5] http://www.osi.ie/
[6] https://en.wikipedia.org/wiki/Irish_Transverse_Mercator
[7] https://en.wikipedia.org/wiki/World_Geodetic_System
[8] https://en.wikipedia.org/wiki/Well-known_text

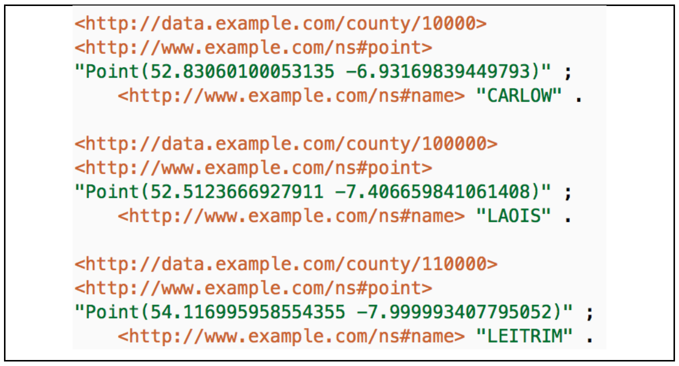
220000	WATERFORD		POINT(634140.854 611037.497)
230000	WESTMEATH		POINT(633172.984 750116.57)
240000	WEXFORD		POINT(684871.008 639468.666)
250000	WICKLOW		POINT(706277.46 695537.898)
260000	DUBLIN		POINT(716330.154 742154.083)

Simulating a relational database that does not support geometries, we have created a mapping M1 that maps "id", "name" and "geoms" to triples. This mapping is shown in Listing 3, but the function – which takes a fair amount of space and therefore omitted – is based on a PHP script[9] that has been ported to ECMAScript. One can see that the mapping should result in two RDF triples for each record.

```
<#TriplesMap1>
 rr:logicalTable [
  rr:sqlQuery "SELECT id, name, geoms FROM county" ;
 ] ;
 rr:subjectMap [
  rr:template "http://data.example.com/county/{id}" ;
 ] ;
 rr:predicateObjectMap [
  rr:predicate ex:name ;
  rr:termType rr:Literal;
  rr:objectMap [ rr:column "name" ];
 ] ;
 rr:predicateObjectMap [
  rr:predicate ex:point ;
  rr:objectMap [
   rr:termType rr:Literal;
   rrf:functionCall [
    rrf:function <#Transform> ;
    rrf:parameterBindings (
     [ rr:column "geoms" ]
    ) ;
   ] ;
  ] ;
 ] ;
.
<#Transform>
 rrf:functionName "transform" ;
 rrf:functionBody """
  // Omitted
 """ ;
.
```

Listing 3: An R2RML-F mapping transforming points in a text field with a function in the mapping (mapping M1).

The execution of our processor generated 52 triples of which some are shown in Figure 1. We will now proceed with evaluating our approach with other mappings in order to analyze the impact of incorporating domain knowledge in the mapping.



```
<http://data.example.com/county/10000>
<http://www.example.com/ns#point>
"Point(52.83060100053135 -6.93169839449793)" ;
    <http://www.example.com/ns#name> "CARLOW" .

<http://data.example.com/county/100000>
<http://www.example.com/ns#point>
"Point(52.5123666927911 -7.406659841061408)" ;
    <http://www.example.com/ns#name> "LAOIS" .

<http://data.example.com/county/110000>
<http://www.example.com/ns#point>
"Point(54.116995958554355 -7.999993407795052)" ;
    <http://www.example.com/ns#name> "LEITRIM" .
```

Figure 1: Partial result of the transformation.

Before describing the experiment, we note that our experiment was run on a MacBook Pro 12.1 with an Intel Core i5 processor (2.7 GHz) and a memory of 8 GB (1867 MHz DDR3). The IDE used is the Eclipse IDE Mars Release (4.5.0) and the Java version 1.8.0_45.

To compare the performance of our approach, we have created two additional mappings. The mappings we will compare are:

- M1 (Listing 3) mapping a text field containing points which are transformed via a function;
- M2 (Listing 4) casting the geometry into a string and providing that field as input to the function; and
- M3 (Listing 5) transforming the geometry into WGS84 in the SQL query, therefore not relying on the function provided in the mapping.

M2 was created to analyze the impact of casting geometries to strings.

For each mapping we have run a script that executes a mapping 110 times in the same virtual machine and measures the execution times. The script was thus run three times, once for each mapping. For each mapping, we ignore the first ten measurements to avoid a bias created by cold starting the Java Virtual Machine. Even though M3 does not use a function, we have not removed the function from the mapping. As functions are loaded only once (prior to executing the mappings), this creates a constant across the three mappings allowing us to more reliably compare the execution times measured. The results are shown in Table 2.

Table 2: Average, maximum and minimum execution time in milliseconds for 100 runs with each mapping.

	M1	M2	M3
Average	77.66	78.62	87.86
Standard Deviation	37.69	34.80	40.19
Min out of 100 runs	16.00	20.00	15.00
Max out of 100 runs	205.00	214.00	205.00

Looking at the average runtime, it is surprising to see that our approach in M1 seemingly performs better than a mapping with no function calls. In order to investigate whether the differences are significant, we performed a Welch Two Sample T-Test for each pair of mappings M1-M2, M1-M3 and M1-M2. The significance level we adopt is 0.05. The p-values for each test are:

1. M1-M2: 0.8518 > 0.05
2. M1-M3: 0.06564 > 0.05
3. M2-M3: 0.0838 > 0.05

All p-values are above 0.05, which suggests that the differences are not significant. This also suggests – at the moment – that our approach, viable from a technical point of view, seems not to cause any serious overhead for this particular use case. We notice that the last two p-values – comparing a mapping applying the function with a mapping that does not – are closer to the confidence level than the first. M1 and M2 rely on calling the function and their difference seems not significant.

4.1 Additional Experiments

We recognize the size of the dataset is limited and the function very complex, so we have ran additional experiments with a larger dataset and a simple function that are similar in setup. We created a table with two columns ("id", "x" and "y" – all integers) and generated 1,000 records with random values between 0 and 100 for x and y. The first experiment mapped the multiplication of x with x, the second the multiplication of x and y. The main difference is the number of parameters (one vs. two), with the first be-

[9] The function to transform ITM into WGS84 is based on http://www.nearby.org.uk/blog/2009/05/13/itm-wgs84-and-irish-grid-and-british-national-grid-php-code/

ing more similar to the conversion of coordinates (i.e., only one argument is passed).

Two mappings were created for the first experiment: one applying a function that multiplies x with x, and one where that multiplication is part of the SQL query.[10] Again executing each mapping 110 times in the same process and only taking the last 100 times into account, we now notice that the p-value is 0.00541 indicating that there is strong evidence against the two mappings being equal, with M1 performing better. Similarly, two mappings were created for the second experiment: one passing x and y to a function and one doing the multiplication in the query. Again, there is strong evidence that both mappings are different (p-value = 0.0000000204), and this time in favor of M2. From these additional experiments, we conclude that further experiments – both in terms of dataset size and varying complexity of functions – are necessary to evaluate performance

Table 3: Average, maximum and minimum execution time in milliseconds for 100 runs with each mapping for the two additional experiments.

	1 variable		2 variables	
	M1	**M2**	**M1**	**M2**
Average	1,646.10	1,716.76	1,980.22	1,729.95
Standard Deviation	154.53	197.88	318.57	285.58
Min out of 100 runs	1,305.00	1,290.00	1,321.00	1,058.00
Max out of 100 runs	2,115.00	2,102.00	2,622.00	2,304.00

5. RELATED WORK

We have to note that functions in transformations have been studied in different domains, such as model-to-model transformations, for which specifications have been developed. One noteworthy example of such a specification in that domain is Query/View/Transformation (QVT) defined by the Object Management Group allowing one access to operators, constructs and functions to transform UML into, for instance, ER [14]. QVT and similar approaches have been adopted in the Semantic Web community to transform meta-models in e.g., UML into OWL ontologies [16], but this is a different problem than the one we address. In this study, we incorporated the notion in a W3C Recommendation for publishing data as RDF. The adoption of RDF allows one to furthermore publish the functions that were developed.

On capturing functions in mappings. Hert et al. presented a survey in which they investigated the capabilities of RDB-2-RDF tools, both direct and annotated [9]. One of the aspects they investigated was *"Transformation Functions"*, which are user-defined functions to transform the (syntactic) representation in RDF. In their survey, they consider this aspect covered when the underlying database technology supports these conversions. Our approach is different as we capture those functions *in* the mapping. ECMAScript provides the ability to load (external) scripts, and hence functionality not necessarily available by the underlying technology.

XSPARQL [4] – initially developed to mediate between XML and RDF via SPARQL and XQuery – has recently been used as an R2RML processor [7]. Since the XQuery specification prescribes support for defining user-defined functions, one should be able to create richer mappings, but limited to XML documents. The same holds for RML [8] when mapping XML documents to RDF.

On representing functions. SPIN [11] provides concepts and predicates for describing functions that we have decided not to reuse. One of the important differences is that SPIN proposes the use of predicates `sp:arg1`, …, `sp:arg`*`n`* for binding the parameters. We consider the use RDF Collections more elegant. As [10] noted, Apache Jena[11] allows one to integrate functions by extending special abstract Java classes which are then available via a special namespace. This, however, creates a dependency on both Java and Apache Jena. All of this also holds for SPIN as it is built on top of Jena.

On R2RML extensions. RML [8] and xR2RML [13] extended R2RML to provide support for generating RDF from different types of structured resources such as XML and CSV. One can see how the ideas presented in this paper can easily be merged with those initiatives. At this stage, however, we choose to extend R2RML validate our ideas and will consider RML at a later stage.

6. CONCLUSIONS AND FUTURE WORK

Capturing procedural knowledge in R2RML either relies on the underlying relational database technology or on data preprocessing. If one is willing to trade tractability in for richer mappings, sharing and declaring richer mappings is possible.

We presented R2RML-F, which extends R2RML with treating function calls as a term map. Using a use case from the geospatial domain, we demonstrated our approach and conducting an experiment to evaluate its performance with a mapping relying on functions provided by the underlying database technology. Our initial results show that our approach is viable and that there seems to be little impact on the performance.

Future work includes additional experiments to validate our findings and developing additional scenarios and use cases to motivate the need for incorporating functions in a mapping, e.g., those calling web services.

7. ACKNOWLEDGMENTS

This study is supported by the 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. We thank the Ordnance Survey Ireland (OSi) for permitting us to use their boundaries dataset for the purposes of this research project. Within OSi, we are especially grateful for the input and domain expertise provided by Lorraine McNerney and Éamonn Clinton.

8. REFERENCES

[1] M. Arenas, A. Bertails, E. Prud'hommeaux, J. Sequeda. A Direct Mapping of Relational Data to RDF. W3C Recommendation, W3C, Sep. 2012. https://www.w3.org/TR/rdb-direct-mapping/

[2] T. Berners-Lee. Relational Databases on the Semantic Web, 1998. Via https://www.w3.org/DesignIssues/RDB-RDF.html, last accessed January 2015.

[3] P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In C. Y. Chan, B. C. Ooi, and A. Zhou, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 1–12. ACM, 2007.

[10] Unlike the previous experiment, the outcome of the function using the underlying technology does not need to be cast to a different data type and therefore no third mapping to analyze that overhead is needed.

[11] https://jena.apache.org/

[4] S. Bischof, S. Decker, T. Krennwallner, N. Lopes, and A. Polleres. Mapping between RDF and XML with XSPARQL. *J. Data Semantics*, 1(3):147–185, 2012.

[5] D. Brickley and R. V. Guha. RDF Schema 1.1. W3C Recommendation, W3C, Feb. 2014. https://www.w3.org/TR/rdf-schema/

[6] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language. W3C Recommendation, W3C, Sep. 2012. https://www.w3.org/TR/r2rml/

[7] D. Dell'Aglio, A. Polleres, N. Lopes, and S. Bischof. Querying the web of data with XSPARQL 1.1. In R. Verborgh and E. Mannens, editors, *Proceedings of the ISWC Developers Workshop 2014, co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 19, 2014, volume 1268 of CEUR Workshop Proceedings,* pages 113–118. CEUR-WS.org, 2014.

[8] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. RML: A generic language for integrated RDF mappings of heterogeneous data. In C. Bizer, T. Heath, S. Auer, and T. Berners-Lee, editors, *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014*, volume 1184 of CEUR Workshop Proceedings. CEUR-WS.org, 2014.

[9] M. Hert, G. Reif, and H. C. Gall. A comparison of RDF-to-RDF mapping languages. In C. Ghidini, A. N. Ngomo, S. N. Lindstaedt, and T. Pellegrini, editors, *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011, ACM International Conference Proceeding Series*, pages 25–32. ACM, 2011.

[10] O. Kovalenko, C. Debruyne, E. Serral, and S. Biffl. Evaluation of technologies for mapping representation in ontologies. In R. Meersman, H. Panetto, T. S. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. D. Leenheer, and D. Dou, editors, *On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings, volume 8185 of LNCS*, pages 564–571. Springer, 2013.

[11] H. Knublauch, J. A. Hendler, K. Idehen. SPIN – Overview and Motivation. W3C Member Submission, W3C, Feb. 2011. https://www.w3.org/Submission/spin-overview/

[12] D. McGuinness, T. Lebo, and S. Sahoo. PROV-O: The PROV ontology. W3C Recommendation, W3C, Apr. 2013. http://www.w3.org/TR/2013/REC-prov-o-20130430/.

[13] F. Michel, L. Djimenou, C. Faron-Zucker, and J. Montagnat. Translation of relational and non-relational databases into RDF with xR2RML. In V. Monfort, K. Krempels, T. A. Majchrzak, and Z. Turk, editors, *WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015*, pages 443–454. SciTePress, 2015.

[14] The Object Management Group. Meta Object Facility (MOF) 2.0 Query/View/-Transformation Specification, 2007.

[15] J. Unbehauen, S. Hellmann, S. Auer, and C. Stadler. Knowledge extraction from structured sources. In S. Ceri and M. Brambilla, editors, Search Computing - Broadening Web Search, volume 7538 of LNCS, pages 34–52. Springer, 2012.

[16] J. Zedlitz, J. Jörke, and N. Luttenberger. *Knowledge Technology: Third Knowledge Technology Week, KTW 2011, Kajang, Malaysia, July 18-22, 2011. Revised Selected Papers*, chapter From UML to OWL 2, pages 154–163. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

APPENDIX

Here we provide listings referred to in the text.

```
<#TriplesMap1>
 rr:logicalTable [
  rr:sqlQuery """SELECT id, name, ST_AsText(geom)
                 AS geom FROM boundaries.county""" ;
  ] ;
 rr:subjectMap [
  rr:template "http://data.example.com/county/{id}" ;
 ] ;
 rr:predicateObjectMap [
  rr:predicate ex:name ;
  rr:termType rr:Literal;
  rr:objectMap [ rr:column "name" ];
 ] ;
 rr:predicateObjectMap [
  rr:predicate ex:point ;
  rr:objectMap [
   rr:termType rr:Literal;
   rrf:functionCall [
    rrf:function <#Transform> ;
    rrf:parameterBindings (
     [ rr:column "geom" ]
    ) ;
   ] ;
  ] ;
 ] ;
.
<#Transform>
 rrf:functionName "transform" ;
 rrf:functionBody """
  // Omitted
 """ ;
.
```

Listing 4: An R2RML-F mapping transforming points from a geometry cast to a string with a function captured in the mapping (mapping M2). Differences wrt Listing 3 are highlighted.

```
<#TriplesMap1>
 rr:logicalTable [
  rr:sqlQuery """SELECT id, name,
                 ST_AsText(ST_Transform(geom,4326))
                 AS geom FROM boundaries.county""" ;
  ] ;
 rr:subjectMap [
  rr:template "http://data.example.com/county/{id}" ;
 ] ;
 rr:predicateObjectMap [
  rr:predicate ex:name ;
  rr:termType rr:Literal;
  rr:objectMap [ rr:column "name" ];
 ] ;
 rr:predicateObjectMap [
  rr:predicate ex:point ;
  rr:objectMap [
   rr:objectMap [ rr:column "geom" ];
  ] ;
 ] ;
.
<#Transform>
 rrf:functionName "transform" ;
 rrf:functionBody """
  // Omitted
 """ ;
.
```

Listing 5: An R2RML-F mapping transforming points from a geometry using a database's built-in function, which is then cast to a string (mapping M3). Differences wrt Listing 3 are highlighted.