Mapping languages analysis of comparative characteristics

Ben De Meester[0000−0003−0248−0987], Pieter Heyvaert[0000−0002−1583−5719], Ruben Verborgh[0000−0002−8596−222X], and Anastasia Dimou[0000−0003−2138−7972]

Ghent University – imec – IDLab,
Department of Electronics and Information Systems,
Technologiepark-Zwijnaarde 122, 9052 Ghent, Belgium
{firstname.lastname}@ugent.be

Abstract. RDF generation processes is becoming more interoperable, reusable, and maintainable due to the increased usage of mapping languages: languages used to describe how to generate an RDF graph from (semi-)structured data. This leads to a rise of new mapping languages, each with different characteristics. However, it is not clear which mapping language can be used for a given task. Thus, a comparative framework is needed. In this paper, we investigate a set of mapping languages that inhibit complementary characteristics, and present an initial set of comparative characteristics based on requirements put forward by those mapping languages. Initial investigation found 9 broad characteristics, classified in 3 categories. To further formalize and complete the set of characteristics, further investigation is needed, requiring a joint effort of the community.

Keywords: Mapping Language · RDF graph generation

1 Introduction

RDF graph generation started as an ad-hoc process: hard-coded applications generated specific RDF graphs, typically from specific data sources in specific data formats. The advent of mapping languages – the first time standardized by the W3C recommended RDB to RDF Mapping Language (R2RML) to generate RDF graphs from relational databases (RDBs) [1] – improved interoperability of this generation process. Using a mapping language, the rules that specify the generation are detached from the processor that executes them [7]. This improves interoperability, reusability, and maintainability of both rules and processor [12].

New mapping languages – or extensions to existing mapping languages – are proposed to increase the functionality or user-experience and adhere to different use cases. For example, mapping languages were proposed extending W3C recommendations, such as R2RML and SPARQL. The RDF Mapping Language (RML) [7] was proposed as an extension of R2RML to support heterogeneous

data sources, and SPARQL-Generate is proposed as an alternative mapping language, serialized using a modified SPARQL syntax [11].

This multitude of mapping languages allows to support more use cases. However, this multitude also defeats the mapping language's purpose, as interoperability and reusability of existing mapping rules and processors decrease. The rules of a mapping language cannot be processed by a processor of another mapping language.

The differences between mapping languages are currently not presented in a unified framework. Finding a suitable mapping language thus involves a laborious process where end-users need to investigate the characteristics of different mapping languages and interpret the differences. There is no clear interoperable description of (non-)functional characteristics of different mapping languages.

In this position paper, we investigate a set of mapping languages of which the referred works claim complementary characteristics. We align and present an initial set of comparative characteristics as put forward by those mapping languages. These characteristics can be used to compare different mapping languages against each other, however, this which is not meant to be complete, but to trigger discussion. Further and thorough investigation is required. After presenting our reference works in Section 2, we discuss the characteristics in Section 3 and conclude in Section 4.

2 Reference works

R2RML is the W3C recommended mapping language for describing RDF graph generation from a relational database (RDB) [1]. It presents a language specifically designed as mapping language, serialized in RDF. Other recommendations that allow to interpret a datasource as an RDF graph exist, such as JSON-LD [15] and CSVW [16]. These recommendations target a single datasource type. However, the need for supporting different datasource types for different use cases influenced the proposal of (i) other not-standardized mapping languages, (ii) extensions of existing mapping languages, and (iii) different notations.

We discuss RML(+FnO), an R2RML extension that supports heterogeneous datasources and data transformations [5, 7]; xR2RML, an [R2]RML extension that supports collections and nested mappings [13, 14]; FunUL, an R2RML extension that supports data transformations [10]; SPARQL-Generate, an alternative mapping language based on SPARQL [11]; and YARRRML, an alternative notation to, a.o., RML [9]. Specifically, we discuss the references of Table 1, and the requirements they put forward.

This list is not exhaustive, however, we choose to discuss these works as their reference work(s) list a set of requirements, and these sets of requirements are (partially) complementary with those of the other mapping languages.

RML The RDF Mapping Language (RML) extends the R2RML recommendation to take into account heterogeneous data sources [7]. Apart from allowing to specify how to generate the subject, predicate, object, and optionally graph

Table 1. The reference works discussed in this paper.

Mapping Language	Reference Work(s)
(1) RML(+FnO)	[5, 7]
(2) xR2RML	[13, 14]
(3) FunUL	[10]
(4) SPARQL-Generate	[11]
(5) YARRRML	[9]

resources, RML allows specifying the logical source that describes the iteration over the data records (e.g., iterating over tabular data or JSON objects), and specifying the data source that specifies the actual data connection (e.g., reading a local XML file or accessing a remote NoSQL endpoint). Later, RML was combined with the Function Ontology (FnO) [2, 4] to include arbitrary data transformations in the generation descriptions (RML(+FnO)) [5].

xR2RML An [R2]RML extension to include mapping functionalities targeted at hierarchical (NoSQL) data structures [13, 14]. These advanced functionalities include (i) nested triplemaps (for addressing hierarchical relationships), (ii) collections and lists, and (iii) combining multiple query languages (for addressing, e.g., a JSON record saved in an RDB).

FunUL An R2RML extension to include data transformations in the form of JavaScript snippets, and support for CSV data sources [10]. The presented work poses a set of eleven requirements for mapping languages, applied to generating RDF graphs from CSV files.

SPARQL-Generate An alternative mapping language using a SPARQL-like syntax to describe the mapping [11]. It has extensible support for different data sources and data transformations. The presented work poses seven functional and non-functional requirements.

YARRRML A notation using YAML[1] to provide a user-friendly way to describe mappings [9]. The mappings described in YARRRML can be translated into RML mappings.

3 Characteristics

Based on the aforementioned related works, and their posed requirements, we discuss following characteristics, which we further classify as non-functional, data source support, or functional characteristics (summarized in Table 2). For every characteristics, we state which reference work posed the original requirement.

[1] https://yaml.org/

Table 2. summarizing the characteristics of discussed mapping languages. If a paper claims adherence (or non-adherence) of another mapping language to a certain characteristic, the citation is provided. Self-claimed statements are not cited, statements claimed by the author of this paper are starred. YARRRML's characteristics except for NF1 and NF2 are taken from RML(+FnO), as YARRRML mappings can be tranlated into RML(+FnO) mappings.

Language	NF1	NF2	NF3	DS1	DS2	F1	F2	F3	F4
(1) RML(+FnO)	✗(4)	✓(4)	✓(4)	✓(2/3/4)	RDF CSV (3/4) XML (4) JSON (4) HTML (4) RDF (4)	✓	✓(4)	✗(2)	✗*
(2) xR2RML	✗(4)	✓*	✗*	✓	RDB NoSQL	✓	✗*	✓	✓
(3) FunUL	✗(4)	✓*	✓	✗*	RDB CSV	✓	✓	✗*	✗*
(4) SPARQL-Generate	✓	✓	✓*	✓	CSV XML JSON HTML Binary	✓	✓	✓*	✓*
(5) YARRRML	✓	✓*	✓(1*)	✓(1*)	✓(1*)	✓(1*)	✓(1*)	✗(1*)	✗(1*)

3.1 Non-functional characteristics

Non-functional characteristics are in relation to the user of the mapping language. This user is either the end-user (creating the mapping), or the developer (integrating the mapping language processor into his/her application).

NF1: Is easy to use by Semantic Web experts (from SPARQL-Generate/YARRRML) If the mapping language is "easy to use by Semantic Web experts" [11] to describe the generation process. This characteristic is addressed by SPARQL-Generate and YARRRML, which both provide a syntax that is either "familiar" (SPARQ-like) [11] or "human-readable" (YAML) [9] to write rules in. The other mapping languages are described in RDF, without a mapping language specific notation.

NF2: Integrates in a typical semantic web engineering workflow (from SPARQL-Generate) If the mapping language "[integrates] in a typical semantic web engineering workflow" [11], i.e., it is related to existing standards. This characteristic is fulfilled by all mapping languages, namely, they are related to R2RML, SPARQL, or YAML.

NF3: Reusable (from FunUL) If the mapping language "allows the serialization of the [generation] process for further reuse" [10]: whether the generation

description is fully covered by the mapping language, or certain parts are hard-coded. xR2RML requires retrieval of the data records as part of the hard-coded process. The other mapping languages allow to specify the connection to the physical data source.

3.2 Data source support characteristics

DS1: Supports heterogeneous data sources (from RML/xR2RML/SPARQL-Generate) Whether the mapping language is focused on a single type of data source, or can support multiple. FunUL is targeted to tabular data sources. The other mapping languages have support for – and are extensible to – other data sources.

DS2: Supports data source [X] (from RML/xR2RML/SPARQL-Generate) Which data sources are currently supported by the mapping language processors.

3.3 Functional characteristics

F1: Supports general mapping functionalities (from FunUL) If general mapping functionalities are provided, namely, specifying "M:N relationships", "literal to IRI", "vocabulary reuse", "data types", "named graphs", and "blank nodes" [10]. These functionalities include (i) generating subject, predicate, object, and (optionally) graph resources, (ii) joining data, and (iii) specifying the used ontology and datatypes. This is typically supported by most mapping languages.

F2: Is extensible (from RML/FunUL/SPARQL-Generate) If additional functionalities can be added, to "allow data to be manipulated and transformed" [10]. RML supports this and proves it by combining data descriptions and FnO, FunUL allows including JavaScript snippets, and SPARQL-Generate relies on SPARQL 1.1's extension functions.

F3: Supports nested hierarchies (from xR2RML) If nested data records are supported, to "map data elements from rows as well as structured values (nested collections [...])" [14]. This is different from being able to query a hierarchical dataset. Instead, this characteristic relates to whether the mapping language can handle hierarchical data structures or not. All R2RML extensions use the tabular datamodel to join data, except for xR2RML. When a nested data record has a different serialization than the parent data record, a combination of query languages might be needed. xR2RML also supports this. SPARQL-Generate is assumed to support this, as it extends SPARQL, a graph-based query language.

F4: Supports collections and lists (from xR2RML) If the mapping language allows "to generate hierarchical values in the form of RDF collections or containers" [14]. xR2RML has built-in support, SPARQL-Generate allows to generate all triples required to generate compliant RDF lists.

3.4 Discussion

The difference in support of certain functional characteristics can help finding a mapping language that is suitable for a certain use case. However, other characteristics also influence the choice of a mapping language. For example, depending on the use case, a different mapping language notation might be preferred. On the one hand, a description in RDF can be less ambiguous, allowing more accurate analysis of the generation description. On the other hand, a mapping description serialized in RDF is arguably less user-friendly to edit and maintain by users.

Further investigation is needed into the difference between serialization and notation of mapping languages, and the underlying model and semantics they employ. For example, comparing SPARQL-Generate with YARRRML: the latter is designed to be human-friendly, and used to employ the semantics (and functionalities) as proposed by RML(+FnO). The former is based on the SPARQL language, but exhibits similar functionalities compared to RML(+FnO). RML, xR2RML and FunUL extend the model of R2RML, which is based on a tabular data model. SPARQL-Generate, following SPARQL, is based on a graph model.

Finally, how to divide which characteristics apply to the mapping language and which apply the processors of that mapping language need to be investigated and distinguished. For example, a mapping language can be extended to a specific datasource, however, support of that datasource might not be easily achieved in the processor. Another example is the automatic generation of metadata of the generation process. This is easier when the mapping language is described in a machine-understandable format, i.e., RDF [3, 6].

4 Conclusion

In this position paper, we provide an initial investigation towards a comparative framework for mapping languages. However, a more systematic review is required. It is apparent that this effort cannot continue without support from the larger community. This work needs to be extended to consider a more complete range of existing mapping languages. Choices made in this work – specifically, the classification of characteristics into *non-functional, data source support, functional*, and the level of detail of different characteristics – need to be verified.

Further formalization and comparison between mapping languages can be tested using a uniform set of test cases. Recent works are looking into test cases sets for specific mapping languages [8]. This work can be extended to provide a set of test cases across mapping languages. We expect this work to start a larger discussion, and provides a basis for a more complete and accurate comparative framework. The recently started Knowledge Graph Construction Community Group can be a crucial driver for further investigation.

Acknowledgements The described research activities were funded by Ghent University, imec, Flanders Innovation & Entrepreneurship (VLAIO), and the Euro-

pean Union. Ruben Verborgh is a postdoctoral fellow of the Research Foundation – Flanders (FWO).

References

1. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. Working group recommendation, World Wide Web Consortium (W3C) (Sep 2012), http://www.w3.org/TR/r2rml/
2. De Meester, B., Dimou, A.: The Function Ontology. Unofficial Draft, Ghent University – imec – IDLab (2016), https://w3id.org/function/spec
3. De Meester, B., Dimou, A., Verborgh, R., Mannens, E.: Detailed provenance capture of data processing. In: Garijo, D., van Hage, W.R., Kauppinen, T., Kuhn, T., Zhao, J. (eds.) Proceedings of the First Workshop on Enabling Open Semantic Science (SemSci). CEUR Workshop Proceedings, vol. 1931, pp. 31–38. CEUR (Oct 2017)
4. De Meester, B., Dimou, A., Verborgh, R., Mannens, E., Van de Walle, R.: An Ontology to Semantically Declare and Describe Functions. In: Sack, H., Rizzo, G., Steinmetx, N., Mladenić, D., Auer, S., Lange, C. (eds.) The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9989, pp. 46–49. Springer (Oct 2016)
5. De Meester, B., Maroy, W., Dimou, A., Verborgh, R., Mannens, E.: Declarative data transformations for Linked Data generation: the case of DBpedia. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) Proceedings of the 14th ESWC. Lecture Notes in Computer Science, vol. 10250, pp. 33–48. Springer, Cham (May 2017)
6. Dimou, A., De Nies, T., Verborgh, R., Mannens, E., Van de Walle, R.: Automated metadata generation for Linked Data generation and publishing workflows. In: Auer, S., Berners-Lee, T., Bizer, C., Heath, T. (eds.) Proceedings of the Workshop on Linked Data on the Web co-located with 25th International World Wide Web Conference (WWW2016). CEUR Workshop Proceedings, vol. 1593. CEUR, Montreal, Canada (Apr 2016)
7. 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: Proceedings of the 7th Workshop on Linked Data on the Web. CEUR Workshop Proceedings, vol. 1184. CEUR (2014)
8. Heyvaert, P., Chaves-Fraga, D., Priyatna, F., Corcho, O., Mannens, E., Verborgh, R., Dimou, A.: Conformance Test Cases for the RDF Mapping Language (RML). In: Proceedings of the 1st Iberoamerican Knowledge Graphs and Semantic Web Conference (2019), under submission
9. Heyvaert, P., De Meester, B., Dimou, A., Verborgh, R.: Declarative Rules for Linked Data Generation at your Fingertips! In: Proceedings of the 15th ESWC: Posters and Demos (2018)
10. Junior, A.C., Debruyne, C., Brennan, R., O'Sullivan, D.: An evaluation of uplift mapping languages. International Journal of Web Information Systems **13**(4), 405–424 (2017)
11. Lefrançois, M., Zimmermann, A., Bakerally, N.: A SPARQL extension for generating RDF from heterogeneous formats. In: The Semantic Web 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 – June 1, 2017, Proceedings. pp. 35–50. Springer International Publishing, Portoroz, Slovenia (May 2017)

12. Maroy, W., Dimou, A., Kontokostas, D., De Meester, B., Verborgh, R., Lehmann, J., Mannens, E., Hellmann, S.: Sustainable linked data generation: The case of DBpedia. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) The Semantic Web – ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10588, pp. 297–313. Springer, Cham, Vienna, Austria (Oct 2017)
13. Michel, F., Djimenou, L., Faron-Zucker, C., Montagnat, J.: Translation of heterogeneous databases into rdf, and application to the construction of a skos taxonomical reference. In: International Conference on Web Information Systems and Technologies. pp. 275–296. Springer (2015)
14. Michel, F., Djimenou, L., Faron-Zucker, C., Montagnat, J.: xR2RML: Relational and Non-Relational Databases toRDF Mapping Language. Rapport de recherche, Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S) (Oct 2017), https://hal.archives-ouvertes.fr/hal-01066663/document/
15. Sporny, M., Kellogg, G., Lanthaler, M.: JSON-LD 1.0 – A JSON-based Serialization for Linked Data. Recommendation, World Wide Web Consortium (W3C) (Jan 2014), http://www.w3.org/TR/json-ld/
16. Tennison, J., Kellogg, G., Herman, I.: Generating RDF from Tabular Data on the Web. Recommendation, World Wide Web Consortium (W3C) (Dec 2015), https://www.w3.org/TR/csv2rdf/