

# Devising Mapping Interoperability With Mapping Translation

Ana Iglesias-Molina<sup>1</sup>, Andrea Cimmino<sup>1</sup> and Oscar Corcho<sup>1</sup>

<sup>1</sup>*Ontology Engineering Group, Universidad Politécnica de Madrid, Spain*

## Abstract

Nowadays, Knowledge Graphs are extensively created using very different techniques, mapping languages among them. The wide variety of use cases, data peculiarities, and potential uses has had a substantial impact in how mapping languages have been created, extended, and applied, resulting in a high amount of diverse languages. These languages, along with their compliant tools and usually the lack of information of both, lead users to use other techniques to construct Knowledge Graphs, such as ad hoc programming scripts that suit their needs. This choice is normally less reproducible and maintainable, what ultimately affects the quality of the generated RDF data. We devise with mapping translation an enhancement to the interoperability of existing mapping languages. Having the possibility of translating mappings to other languages can make more accessible mapping technologies, lowering its learning curve and widening the use of mapping tools. This position paper analyses the possible language translation approaches, presents the scenarios in which it is being applied and discusses how it can be implemented.

## Keywords

Mapping languages, Ontology Description, Mapping Translation

## 1. Introduction

Knowledge Graphs (KG) are increasingly used in academia and industry to represent and manage the increasing amount of data on the Web [1]. A large number of techniques to create KGs have been proposed: tools such as OpenRefine, programming ad hoc scripts, or mapping languages.


Mapping languages represent the relationships between heterogeneous data and an RDF version following the schema provided by an ontology, i.e., the rules on how to translate from non-RDF data into RDF. This data can be expressed in a wide variety of formats, such as tabular, JSON, or XML among many others. Due to the heterogeneous nature of data, the extensive range of techniques and the specific requirements that some scenarios may impose, an increasing number of mapping languages have been proposed [2, 3]. The differences among them are based on mainly three different aspects: (a) they focus on a particular feature, whereas to describe a specific data format (e.g., RML [2] extension of R2RML [4] for more data sources than RDBs) or to implement a new capability (e.g., R2RML-F [5] for incorporating functions to R2RML); (b) if they are designed for a particular technique or scenario that has special requirements, (e.g., the


---

*Third International Workshop On Knowledge Graph Construction, Co-located with the ESWC 2022, Crete - 30th May 2022*

✉ ana.iglesiasm@upm.es (A. Iglesias-Molina); andreajesus.cimmino@upm.es (A. Cimmino); oscar.corcho@upm.es (O. Corcho)

ORCID 0000-0001-5375-8024 (A. Iglesias-Molina); 0000-0002-1823-4484 (A. Cimmino); 0000-0002-9260-0753 (O. Corcho)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

WoT-mappings [6], which were designed as an extension of the Web of Things standard [7]); or (c) they are based on a schema/language that has some inherent capabilities (e.g., SPARQL-based languages, such as SPARQL-Generate [8], that extends SPARQL 1.1).

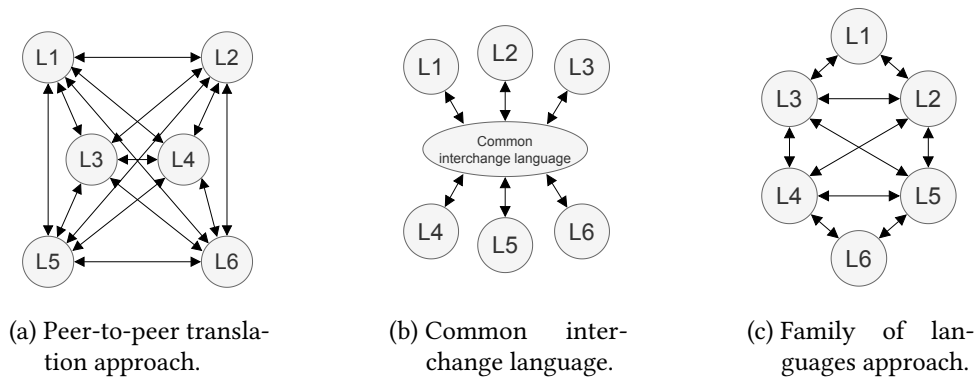
As a result, the diversity of mapping languages allows for the construction of KG from heterogeneous data sources in many different scenarios. Current mapping languages may be categorized by their schema: RDF-based (e.g. R2RML [4] and extensions, CSVW [9]), SPARQL-based (e.g., SPARQL-Generate [8], SPARQL-Anything [10]) or based on other schemas (e.g. ShExML [11], Helio mappings[12]). Nevertheless, the existing techniques usually implement just one mapping language, and sometimes not even the whole language specification [13, 14]. Deciding which language and technique should be used in each scenario becomes a costly task, since the choice of one language may not cover all needed requirements [15]. Some scenarios require a combination of mapping languages because of their differential features, which entails using different techniques. In many cases, this diversity leads to ad hoc solutions that reduce reproducibility, maintainability, and reusability [16].

The increasing and heterogeneous emergence of new use cases still motivates the community to keep developing solutions that are, more commonly than desired, not compatible with existing ones. This position paper develops the concept of mapping translation, proposed by Corcho et al. [17], that is defined as “a function that transforms a set of mappings described in one language into a set of mappings described in another language”. Enabling mapping translation can enhance the interoperability among existing mapping languages in several aspects, such as reducing (a) the need of learning many different languages and (b) the effort to manually translate mappings to use different tools. Then, as a result, mapping technologies can be more accessible to users. Just by learning one language, users could use any tool that suits best their needs without necessarily learning its compliant mapping language, what usually poses a challenge and supposes a barrier for using them [18]. This paper presents some approaches for language translation, shows the current situations in which mapping translation is being applied and their benefits, and outlines different techniques in which this issue may be addressed.

The remaining of this article is structured as follows: Section 2 provides some insights about language translation and the situations in which it is being applied. Section 3 proposes three different techniques to address mapping translation at a larger scale. Finally, Section 4 draws some conclusions of the concepts presented in the paper.

## 2. Mapping translation: Context

Corcho et al. [17] define mapping translation as “a function that transforms a set of mappings described in one language (we call them original mappings) into a set of mappings described in another language (we call them target mappings)”. In addition, they attach to that function two desirable properties, the *information preservation property* (IPP) and the *query result preservation property* (QRPP). IPP states that the original and target mappings will be able to transform data to RDF that must contain the same information. QRPP states that the data transformed with the original and the target mappings must return the same information when queried. In other words, mapping translation must ensure that the same information must be preserved in the original and the target mappings so that, when evaluated, their transformed data and queried



**Figure 1:** Types of language translations (Adapted from [20]).

results return the same information.

Having these concepts present, following we introduce some approaches to language translation and present a set of scenarios in which mapping translation is being applied. Authors assume the reader is familiar with current mapping languages and their general characteristics (e.g., R2RML [19], RML [2], SPARQL-Generate [8], CSVW [9] and ShExML [11]).

## 2.1. Approaches to language translation

In the context of language translation, there are several approaches that carry out translations among a set of languages. Depending on the situation at hand, an approach can be advantageous with respect to the other ones. We highlight the following [21]:

**Peer-to-peer translation** (Fig. 1a) supports ad hoc translation solutions between pairs of languages. This one may seem as the most straightforward approach, requiring the development of only the translator services needed for the situation at hand and with the possibility of adjusting it ad hoc for each situation. However, it becomes decreasingly feasible as the number of required translations increases.

**Common interchange language** (Fig. 1b) uses a language that serves as an intermediary among several languages. This approach reduces the number of translator services needed to develop and it is the most feasible of the three to scale in amount. It involves creating (or luckily having) a language able to represent the expressiveness of all languages, to avoid information loss. Additionally, this implies that there are common patterns shared by the languages independently of their representation, and that an abstract manner of gathering them is possible, which may not be thus for highly heterogeneous languages.

**Family of languages** (Fig. 1c) considers sets of languages and translations between the representatives of each set. This approach stands out for situations where there are clear subgroups of languages similar among them but dissimilar among languages from other groups. In addition, this scenario could imply having "chains of mappings". That means, instead of having a straight-forward translation or a two-step translation (like in previous approaches, respectively), a mapping may need to be translated into intermediate languages until reaching

the target one if the differences between the original and the target are too many.

## 2.2. Mapping translation scenarios

Mapping translation is a concept that is already being applied to specific use cases and in different ways. For instance, there are currently some implementations that unidirectionally translate pairs of mapping languages. ShExML and YARRRML in their respective online editors<sup>1,2</sup> enable translation to RML. Another case is when tools implement RML/R2RML mapping translation into the language they are designed to parse; such as Helio<sup>3</sup> and SPARQL-Generate<sup>4</sup>, that translate from RML to their respective language; and Ontop [22], that translates R2RML into its compliant language, OBDA mappings [23]. These translations makes it possible to extend the outreach of the tool, since they enable the possibility of using them without the need for learning their specific language, but using one that is widely used, in this case the W3C Recommendation R2RML and its extension RML. Another similar case is Morph-KGC<sup>5</sup>, a materializer that translates from R2RML to RML, and more recently, to RML-star [24], to be able to process the three languages.

Another case we want to present is Mapeathor [18], a tool that takes the mapping rules specified in spreadsheets and transforms them into a mapping in either R2RML, RML or YARRRML. It aims to lower the learning curve of those languages for new users and ease the mapping writing process. Similarly, XRM<sup>6</sup> (Expressive RDF Mapper) provides a user-friendly intermediate syntax to create mappings in CSVW, R2RML and RML.

Finally, we remark tools that provide a set of optimizations on the construction of RDF graphs exploiting the translation of mapping rules. This is the case of Morph-CSV [25] and FunMap [26]. Morph-CSV performs a transformation over the tabular data with RML+FnO mappings and CSVW annotations, and outputs a database and R2RML mappings. FunMap takes an RML+FnO mapping, performs the transformation functions indicated, outputs the parsed data and generates a function-free RML mapping.

The approaches presented are, mainly, examples of peer-to-peer translation for specific uses. The exception are Mapeathor and XRM, that from an intermediate user-friendly representation translate to some languages, and thus align with the approach of a common interchange language. Even though most of these translation examples involve R2RML or RML, there is no holistic approach of a general translation framework.

## 3. Mapping translation: Techniques

This section presents three proposals to implement a mapping translator service general enough to enable translation among several languages taking into account the *information preservation*

---

<sup>1</sup><http://shexml.herminiojarcia.com/editor/>

<sup>2</sup><https://rml.io/yarrml/matey/#>

<sup>3</sup><https://github.com/oeg-upm/helio/wiki/Streamlined-use-cases#materialising-rdf-from-csv--xml-and-json-files-using-rml>

<sup>4</sup><https://github.com/sparql-generate/rml-to-sparql-generate>

<sup>5</sup><https://morph-kgc.readthedocs.io/en/latest/>

<sup>6</sup><https://zazuko.com/products/expressive-rdf-mapper/>

and *query result preservation* properties described in Section 2. These proposals are, namely, (1) Software-based, (2) Construct Query-based, and (3) Executable Mapping-based. These implementations can be applied to any of the language translation approaches of Section 2.1.

**Software-based translation.** It consists on ad-hoc software implementation for each pair of languages to perform bidirectional translations between them. As any ad hoc solution, it benefits from adjusting specifically to any situation with the (almost) unlimited possibilities that programming languages provide. This is the approach that all situations presented in Section 2.2 have applied, although with unidirectional translations.

**Construct query-based translation.** This approach takes advantage of SPARQL query language with construct queries, which return an RDF graph. These particular queries extract the data by matching graph patterns of the query (*WHERE* clause) and builds the output graph based on a template (*CONSTRUCT* clause). Since many languages are defined by the schema of an ontology and are usually written in Turtle syntax (e.g., R2RML and extensions), this approach can be applicable to them. This approach benefits from relying on a well-established standard, as SPARQL is nowadays, and its compliant engines. However, it would leave out languages with other schemes (e.g., SPARQL-based), without relying on software-based solutions.

**Executable mapping-based translation.** This last approach makes use of executable mappings automatically generated from ontology alignment to perform data translation between the two ontologies [27]. Similarly to the previous approach, this one also makes use of construct queries from SPARQL in the executable mappings. While the previous one relied on manual effort to build queries, this one takes advantage of the ontologies that define RDF-based mapping languages. In addition to the benefits and setbacks that the previous approach has, this approach may be hindered by the language constructs to build mappings. That is to say, single one-to-one correspondences of ontology entities may not be enough to gather and be able to translate their expressiveness and capabilities, especially for considerably different languages.

The techniques proposed are presented in decreasing order of manual effort required. The first one is completely ad hoc, and even though it could use some modules of the developed solutions presented in Section 2.2, many more would be needed to provide a complete set of bidirectional translations covering a good number of languages. The second one requires considerable effort to build queries for RDF-based languages, assuming no extra help from software implementation is needed. The third one could ideally be automatically done from ontology alignments creation to mapping execution generation. However, the rate of success of this approach without manual intervention is not expected to be high, especially for the ontology alignment part when the input ontologies considerably differ from one another or present different constructs (with different number of elements or differently structured).

## 4. Conclusions

This paper develops the concept of mapping translation, proposed by Corcho et al. [17]. It analyses the possible language translation approaches, updates the scenarios in which it is being applied, and proposes some implementation techniques to perform it.

There are several possibilities in order to fully develop a complete solution to achieve mapping translation that ensures information preservation, as described in previous sections. It not only

requires choosing the technical implementation according to the available efforts and resources, but more importantly, it involves deciding wisely the language translation approach that suits best this particular case of mapping languages.

All mapping languages, independently of their design, have been built for the same purpose: describing non-RDF data in terms of an ontology. It can be that the rules that the different mappings create can be represented in an abstract, language-independent manner. However, the sometimes large differences among these languages may question this assumption. Some languages, inside their categories, are similar to each other, R2RML and its extensions, for instance. Languages from different groups can be related, such as ShExML and RML, despite some inevitable differences in their features. There are others that are more unique, such as CSVW. Lastly, the SPARQL-based group is more isolated from the others due to the great possibilities that SPARQL already provides.

This scenario poses challenges for every language translation approach. Peer-to-peer translation would require a substantial amount of effort for divergent languages. Using families of languages could work better with respect to the previous one, but it still would have to face several challenges in language representation and the amount of translator services required. Meanwhile, using a common interchange language would be the one that reduces most efforts, but there is no absolute certainty that a common interchange language could be able to represent them all. Still, authors advocate for this approach, and some steps have already been taken to draft this language<sup>7</sup>, with the base idea that the mapping rules can be abstracted and represented in an ontology-based language.

Even though it does not present itself as an easy task, mapping translation is a concept that can only benefit the current landscape of heterogeneous mapping languages. After years of Knowledge Graph construction, the increasing and heterogeneous emergence of new use cases still motivates the community to keep developing solutions, whether ad hoc or with extensions of widely used languages. Mapping translation has the potential of building bridges among the past (but still used) and new solutions to improve interoperability.

## Acknowledgments

The work presented in this paper is partially funded by Knowledge Spaces project (Grant PID2020-118274RB-I00 funded by MCIN/AEI/ 10.13039/501100011033); and partially funded by the European Union's Horizon 2020 Research and Innovation Programme through the AURORAL project, Grant Agreement No. 101016854.

## References

- [1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, *ACM Computing Surveys (CSUR)* 54 (2021) 1–37.
- [2] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van De Walle, RML: A generic language for integrated RDF mappings of heterogeneous data, in: LDOW, 2014.

---

<sup>7</sup>[w3id.org/conceptual-mapping/portal](http://w3id.org/conceptual-mapping/portal)



- [3] H. García-González, I. Boneva, S. Staworko, J. E. Labra-Gayo, J. M. Cueva-Lovelle, ShExML: improving the usability of heterogeneous data mapping languages for first-time users, *PeerJ Computer Science* 6 (2020) e318. URL: <https://peerj.com/articles/cs-318>.
- [4] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation 27 September 2012, [www.w3.org/TR/r2rml](http://www.w3.org/TR/r2rml) (2012).
- [5] C. Debruyne, D. O’Sullivan, R2rml-f: towards sharing and executing domain logic in r2rml mappings, in: *LDOW@ WWW*, 2016.
- [6] A. Cimmino, M. Poveda-Villalón, R. García-Castro, ewot: A semantic interoperability approach for heterogeneous iot ecosystems based on the web of things, *Sensors* 20 (2020) 822.
- [7] M. Kovatsch, R. Matsukura, M. Lagally, T. Kawaguchi, K. Kajimoto, Web of Things (WoT) Architecture, W3C Recommendation 9 April 2020, <https://www.w3.org/TR/wot-architecture/> (2020).
- [8] M. Lefrançois, A. Zimmermann, N. Bakerally, A SPARQL extension for generating RDF from heterogeneous formats, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10249 LNCS (2017) 35–50.
- [9] J. Tension, G. Kellogg, I. Herman, Model for tabular data and metadata on the web, W3C Recommendation (2015).
- [10] E. Daga, L. Asprino, P. Mulholland, A. Gangemi, Facade-x: an opinionated approach to sparql anything, *arXiv preprint arXiv:2106.02361* (2021).
- [11] H. García-González, A shexml perspective on mapping challenges: already solved ones, language modifications and future required actions, in: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction*, 2021.
- [12] A. Cimmino, R. García-Castro, Helio: a framework for implementing the life cycle of knowledge graphs, *Semantic Web* (2022).
- [13] D. Chaves-Fraga, F. Priyatna, A. Cimmino, J. Toledo, E. Ruckhaus, O. Corcho, Gtfs-madrid-bench: A benchmark for virtual knowledge graph access in the transport domain, *Journal of Web Semantics* 65 (2020) 100596.
- [14] J. Arenas-Guerrero, M. Scrocca, A. Iglesias-Molina, J. Toledo, L. P. Gilo, D. Dona, O. Corcho, D. Chaves-Fraga, Knowledge graph construction with r2rml and rml: An etl system-based overview, in: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction*, 2021.
- [15] B. De Meester, W. Maroy, A. Dimou, R. Verborgh, E. Mannens, Declarative data transformations for linked data generation: The case of DBpedia, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10250 LNCS (2017) 33–48.
- [16] A. Iglesias-Molina, D. Chaves-Fraga, F. Priyatna, O. Corcho, Enhancing the maintainability of the bio2rdf project using declarative mappings., in: *The 12th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences*, 2019.
- [17] O. Corcho, F. Priyatna, D. Chaves-Fraga, Towards a new generation of ontology based data access, *Semantic Web* 11 (2020) 153–160.
- [18] A. Iglesias-Molina, L. Pozo-Gilo, D. Doña, E. Ruckhaus, D. Chaves-Fraga, Ó. Corcho, Mapeathor: Simplifying the specification of declarative rules for knowledge graph con-

- struction, in: ISWC (Demos/Industry), 2020.
- [19] B. Villazón-Terrazas, M. Hausenblas, R2RML and Direct Mapping Test Cases, W3C Note, W3C, 2012. <http://www.w3.org/TR/rdb2rdf-test-cases/>.
  - [20] O. Corcho, A. Gómez-Pérez, A layered approach to ontology translation with knowledge representation, Ph.D. thesis, UPM, 2004.
  - [21] J. Euzenat, H. Stuckenschmidt, The ‘family of languages’ approach to semantic interoperability, *Knowledge transformation for the semantic web* 95 (2003) 49.
  - [22] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodríguez-Muro, G. Xiao, Ontop: Answering sparql queries over relational databases, *Semantic Web* 8 (2017) 471–487.
  - [23] M. Rodríguez-Muro, M. Rezk, Efficient sparql-to-sql with r2rml mappings, *Journal of Web Semantics* 33 (2015) 141–169.
  - [24] T. Delva, J. Arenas-Guerrero, A. Iglesias-Molina, O. Corcho, D. Chaves-Fraga, A. Dimou, Rml-star: A declarative mapping language for rdf-star generation, in: *International Semantic Web Conference - Demos&Industry*, 2021.
  - [25] D. Chaves-Fraga, E. Ruckhaus, F. Priyatna, M.-E. Vidal, O. Corcho, Enhancing virtual ontology based access over tabular data with morph-csv, *Semantic Web* (2021) 1–34.
  - [26] S. Jozashoori, D. Chaves-Fraga, E. Iglesias, M.-E. Vidal, O. Corcho, Funmap: Efficient execution of functional mappings for knowledge graph creation, in: *International Semantic Web Conference*, Springer, 2020, pp. 276–293.
  - [27] C. R. Rivero, I. Hernández, D. Ruiz, R. Corchuelo, Generating sparql executable mappings to integrate ontologies, in: *International Conference on Conceptual Modeling*, Springer, 2011, pp. 118–131.