RML Playground: Online Editing, Validation, and Execution for the RDF Mapping Language

Els de Vleeschauwer[1], Arthur Vercruysse[1], Ben De Meester[1] and Pieter Colpaert[1]

[1]IDLab, Department of Electronics and Information Systems,
Ghent University – imec, Technologiepark-Zwijnaarde 122, 9052 Ghent, Belgium

Abstract

The RDF Mapping Language (RML) has seen various extensions over the years and was overhauled and modularized by the W3C KGC Community Group. However, this variety of RML versions and varying feature support by different mapping engines makes it hard for users to know how to create valid mapping rules. We demonstrate RML Playground: a Web application that guides users to edit, validate, and execute mapping rules for configurable RML versions and mapping engines. The RML Playground offers (i) a text editor with real-time RDF syntax and SHACL shape validation and autocompletion, powered by the Semantic Web Language Server (SWLS); (ii) a knowledge graph construction pipeline using a configurable mapping engine; and (iii) curated example mapping rules illustrating the targeted RML version. RML Playground currently has two deployed instances, available at https://w3id.org/imec/rml/playground : the latest KGC Community RML version with Burp as mapping engine, and the RML.io-published RML version with RMLMapper as mapping engine. Completely automated using the specifications' ontology and shape descriptions, the RML Playground is easily extensible and maintainable to new RML versions and extensions. Its use of the specifications' documentation allows editors to validate their correctness. The provision of meaningful hints during the editing process lowers the barrier for using the (new) RML specifications and supports training and dissemination activities.

Keywords

Knowledge Graph Construction, Playground, RDF Mapping Language, Semantic Web Language Server

1. Introduction

Since its inception in 2013 [1][1], the RDF Mapping Language (RML)—a mature language to declare mapping rules for constructing Knowledge Graphs (KGs) from heterogeneous data sources [2]—has seen various extensions [3, 4, 5, 6, 7, 8, 9, 10] and was overhauled and modularized by the W3C KGC Community Group [11].

However, this variety of RML versions and varying feature support by different mapping engines makes it hard for users to know how to create valid mapping rules: the validity of the rules depends on the RML versions or extensions used and which mapping engine is employed.

In this paper, we demonstrate the RML Playground: a Web application that guides users to edit, validate, and execute mapping rules for configurable RML versions and engines. The primary target audience consists of novice RML users with prior RDF knowledge who are learning how to write RML mapping rules. The RML Playground currently encompasses two deployed instances, available at https://w3id.org/imec/rml/playground: the latest KGC Community RML version with Burp as mapping engine, and the RML.io-published RML version with RMLMapper as mapping engine.

2. Related Work

Various tools and languages are proposed to help the creation of mapping rules, described using RML or related declarative mapping languages. The work of Duchateau and Debruyne highlights the trade-

KGCW'26: 7th International Workshop on Knowledge Graph Construction, May 10th, 2026, Dubrovnik, Croatia
els.devleeschauwer@ugent.be (E. de Vleeschauwer); arthur.vercruysse@ugent.be (A. Vercruysse); ben.demeester@ugent.be (B. De Meester); pieter.colpaert@ugent.be (P. Colpaert)
0000-0002-8630-3947 (E. de Vleeschauwer); 0000-0003-1586-5122 (A. Vercruysse); 0000-0003-0248-0987 (B. De Meester); 0000-0001-6917-2167 (P. Colpaert)

[1]https://rml.io/specs/rml/

off between guidance and flexibility offered by visual tools and textual abstractions in an Integrated Development Environment (IDE) [12]. **Visual tools** such as RMLEditor [13] and JUMA [14] offer strong guidance for the construction of valid mapping rules but constrain the liberty needed to create complex mapping rules. **Textual abstractions**, such as YARRRML [15] (a YAML-based serialization) and XRM[2] (an abstract syntax aimed to resemble programming languages) offer more flexibility. YARRRML is integrated in Matey [15], an online IDE with an integrated workflow to upload data sources and execute mapping rules, XRM is offered as plugin for existing IDEs, e.g., Eclipse and Visual Studio Code. Their inclusion in an IDE allows providing guidance through syntax highlighting and autocompletion. Still, their expressivity is limited to the textual abstraction and prevents enriching the mapping rules with metadata that is not necessarily related to RML, e.g., PROV-O statements. GRAPE [12] introduces an IDE which allows users to interlace a textual abstraction of RML—through a Domain-Specific Language (DSL) similar to XML—with additional RDF metadata in one document. GRAPE includes DSLs for Turtle and three modules of the new RML specification for autocompletion. Its compositional approach allows adding DSLs for other RML modules or RDF ontologies, however, GRAPE also depends on the alignment between the DSL abstraction layer and the evolving RML specifications.

Meanwhile, the Semantic Web Language Server (SWLS) leverages the Language Server Protocol (LSP)[3] to bring functionalities such as real-time syntax validation, ontology-aware autocompletion, and SHACL-based diagnostics to any LSP-compatible editor. This allows integrating SWLS in a broad ecosystem of development environments [16]. SWLS supports multiple serializations (e.g., Turtle and JSON-LD) and allows support for any RDF data model by importing its ontologies and SHACL shapes.

Because RML mapping rules are described in RDF, and are documented with ontologies and SHACL shapes, we can leverage the general-purpose SWLS to support mapping rule editing.

3. Architecture

The **RML Playground** builds on and complements the efforts of the actively maintained Matey [15] and SWLS [16], integrating SWLS into a Web-based editing environment for editing mapping rules.

First, we implemented generic improvements to SWLS to improve support for our use case: *guide* users when writing mapping rules for a *specific RML version.* For reliable guidance, the required ontologies and SHACL shapes must be accessible. By default, SWLS's autocompletion includes terms from ontologies available on Linked Open Vocabularies[4]. Additionally, SWLS allows importing ontologies and SHACL shapes through prefix declarations and `owl:import` statements inside the document that is being edited. However, this requires such RML-specific duplicate statements for each mapping document. For this, SWLS was extended with a configurable list of ontologies and SHACL shapes URLs, managed on application level. To secure valid mapping rules, autocompletion for RML terms must always fit the semantic context. By default, SWLS orders properties for autocompletion in such a way that properties with the correct domain are shown first, and visually distinguishes them from other properties. SWLS has been extended with an option to restrict the autocompletion for certain namespaces to *only* properties with the correct domain. This keeps the focus on the targeted RML version, as the user no longer gets suggestions related to other RML versions.

Next, we enhanced the JavaScript codebase of Matey by integrating the Monaco editor[5]. As Monaco supports the Languages Server Protocol, this allows to easily integrate SWLS. To facilitate the SWLS integration into the JavaScript codebase, this integration code was released as npm package: `swls-monaco`[6]. Finally, we enabled configurability of a specific deployment, customizing the playground to the targeted RML version: select a specific mapping engine, import specific ontologies and SHACL shapes, and include curated example mapping rules.

[2] https://zazuko.com/products/expressive-rdf-mapper/
[3] https://microsoft.github.io/language-server-protocol/
[4] https://lov.linkeddata.es/dataset/lov/
[5] https://microsoft.github.io/monaco-editor/
[6] https://www.npmjs.com/package/swls-monaco

The code of RML Playground is available online at https://github.com/RMLio/playground/v1.0.2 [7] under the permissive MIT license. RML Playground instances are deployed as static sites and run entirely in the browser. Only the mapping engine must be provided as a separate API endpoint; all editing and rule handling are client-side. RML Playground can integrate mapping engines if they expose an HTTP interface that adheres to the OpenAPI specification[8]

RML Playground provides user guidance in three complementary ways for any configured RML version: (i) the text editor provides real-time RDF and RML syntax checking: RDF syntax through native SWLS functionality, and RML syntax through preconfigured SHACL shape-based validation and autocompletion; (ii) a preconfigured KG construction pipeline that is compliant with the configured SHACL shape(s); and (iii) curated example mapping rules illustrating the targeted RML version.

To demonstrate its general applicability, we have currently deployed two instances of RML Playground. RML.kgc Playground[9] includes examples illustrating each module of the latest KGC Community RML version and uses its reference implementation Burp[10] [17] as mapping engine in its mapping pipeline. RML.io Playground[11] targets the RML version[12] published on RML.io and its extensions RML-FNO, RML-Target, incRML, RML HTTP Request Access and RML Dynamic Target[13], includes examples illustrating those specifications, and uses RMLMapper[14] as mapping engine in its mapping pipeline. The configurable set-up allows us to version these deployments so that we can later support different RML mapping engine versions in parallel.

4. Demo

A screencast demonstrating all features of the RML Playground is available at https://w3id.org/imec/rml/playground/screencast/2026-kgcw. Below, we briefly describe the demo. The RML Playground's Graphical User Interface (GUI) consists of three panels (Figure 1). The top left panel (Figure 1 A) is the editing area for data sources from which RDF data is generated. Multiple data sources can be added through the use of a drop-down menu with data in different formats, such as CSV, JSON, and XML.

The top right panel (Figure 1 B) is the editing area for mapping rules. When writing mapping rules, the user is guided by real-time Turtle syntax validation (Figure 2a), autocompletion (Figure 2b), and SHACL shape validation (Figure 2c). The Turtle syntax validation marks syntax problems such as unexpected characters or undefined prefixes, adding textual explanations on hover. Additionally, consistent formatting and syntax highlighting improve readability of the mapping rules. The autocompletion of RML terms is restricted to properties with a compatible domain. The autocompletion is not restricted to RML terms: terms are suggested for any prefix included in Linked Open Vocabularies[15], and all prefixes declared at the top of the mapping document (on the condition that the referred resources are accessible online), as help for the user when mapping source data to RDF terms or when extending mapping rules with metadata. However, RML terms from other RML versions, not targeted by the deployed playground, will not be proposed to the user. Visual markings indicate a violation of the SHACL shapes of the targeted RML version. On hover, the user can read which SHACL rule is unsatisfied and the related SHACL message (if this is specified in the underlying SHACL shape).

The bottom panel (Figure 1 C) displays constructed RDF data after executing the mapping pipeline by clicking the "Generate RDF" button (Figure 1 D). When multiple targets are defined in the mapping rules, multiple datasets will be generated. The number of generated RDF datasets is displayed in the title

[7] https://doi.org/10.5281/zenodo.19627151

[8] Available at https://rml.io/api/playground/rmlmapper/. Further documentation is provided in the repository at https://github.com/RMLio/rmlmapper-webapi-js branch feature-extend-for-rmlkgc.

[9] https://w3id.org/imec/rml/playground/kgc/burp/v0

[10] https://github.com/kg-construct/BURP/commit/5b6845034557a3ac1923a96140f987c0b6ce285a

[11] https://w3id.org/imec/rml/playground/rmlio/rmlmapper/v8

[12] https://rml.io/specs/rml/

[13] https://fno.io/rml/, https://rml.io/specs/rml-target/, https://knowledgeonwebscale.github.io/incrml-spec/, https://w3id.org/imec/rml/specs/access/httprequest, and https://w3id.org/imec/rml/specs/target/dynamictarget, respectively

[14] https://github.com/RMLio/rmlmapper-java/, v8.1.0

[15] https://lov.linkeddata.es/dataset/lov/vocabs

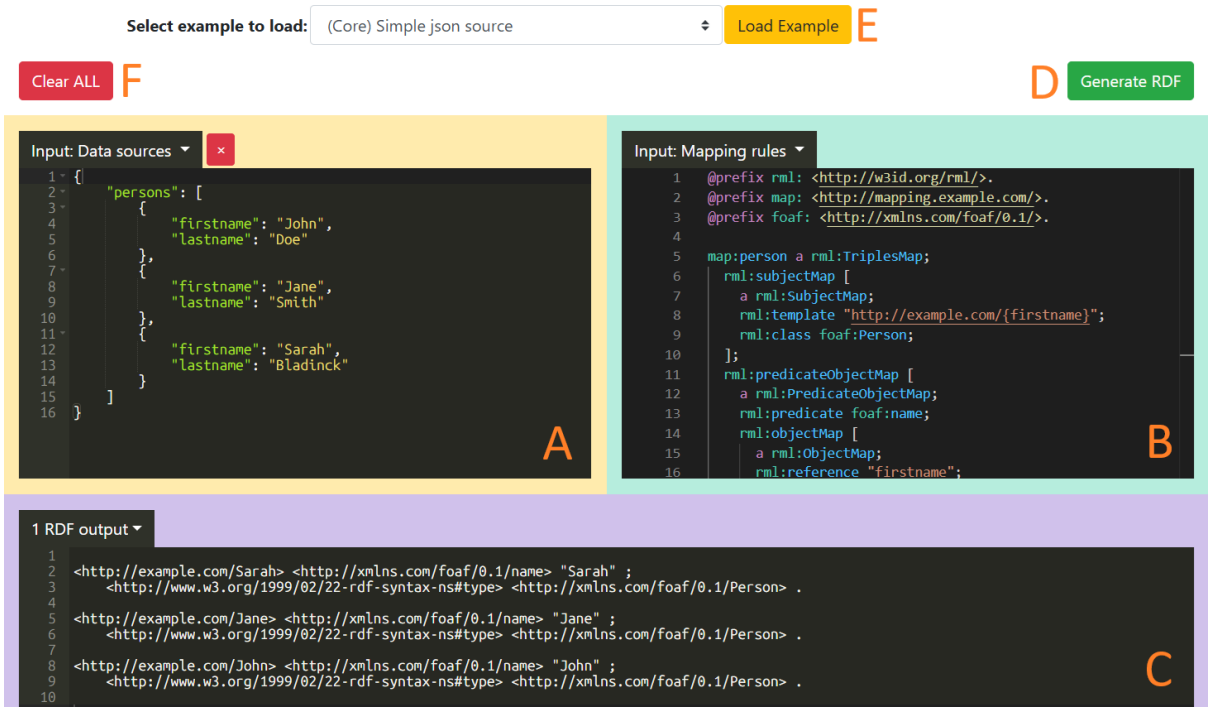
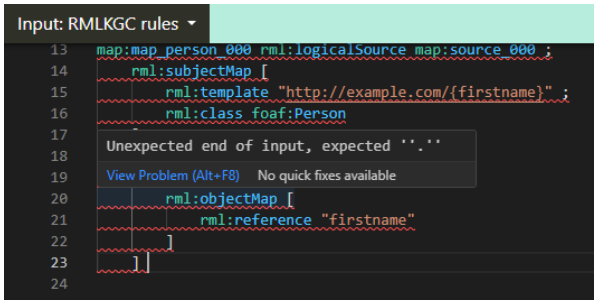
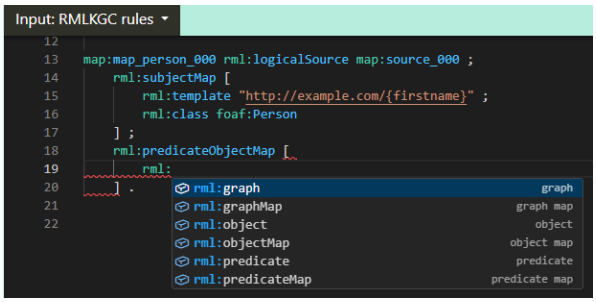


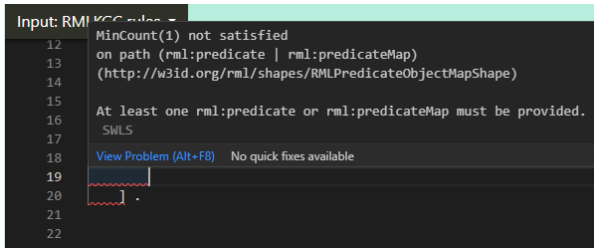
Figure 1: The GUI of RML Playground consists of three panels: editing source data (top-left, A), editing RML mapping rules (top-right, B), and displaying constructed RDF data after executing the mapping pipeline (bottom, C).



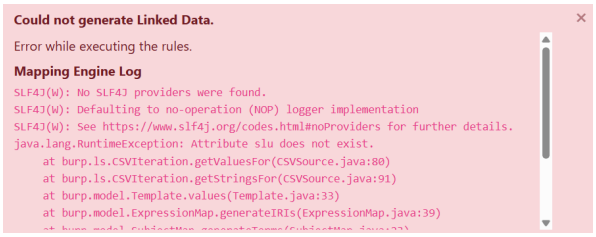
(a) Turtle syntax validation



(b) Autocompletion based on ontology descriptions



(c) Shape validation based on SHACL shapes



(d) Error thrown by the mapping engine

Figure 2: User guidance: Turtle syntax validation, autocompletion, shape validation, and error descriptions.

of the bottom panel. Clicking this title, a drop-down list appears, allowing the user to switch between the generated datasets or to download them. Any error messages reported by the underlying mapping engine are shown as pop-up notifications to assist in correcting the mapping document (Figure 2d).

The user can load curated examples, selecting them from the example drop-down list and clicking the "Load example" button (Figure 1 E). Each example includes one or more datasets and a mapping document. Finally, the "Clear all" (Figure 1 F) button allows a fresh restart with blank editing panels.

5. Conclusion

The demonstrated RML Playground guides users to edit, validate, and execute RML mapping rules. The direct integration of ontologies and SHACL shapes enables consistent alignment with the targeted RML specification, avoiding the dependency on an RML-specific abstraction layer. Completely automated using the specifications' ontology and shape descriptions, the RML Playground is easily maintainable and extensible to new and updated RML versions, extensions, and mapping engines. Its use of the specifications' documentation allows editors to validate their correctness. The provision of meaningful hints during the editing process lowers the barrier for using the (new) RML specifications and supports training and dissemination activities.

The two deployed RML Playground instances include a link inviting users to participate in a user study. The collected feedback will serve as the basis for a future human-centered evaluation of the RML Playground.

Acknowledgments

The described research activities were supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10), the European Union's Horizon Europe research and innovation program under grant agreement no. 101058682 (Onto-DESIDE), and the imec ICON project Solid4Media (Agentschap Innoveren en Ondernemen project nr. HBC.2022.0770).

Declaration on Generative AI

The authors used Copilot for grammar and spelling check, reviewed and edited the content as needed, and take full responsibility for the publication's content.

References

[1] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, in: C. Bizer, T. Heath, S. Auer, T. Berners-Lee (Eds.), Proceedings of the 7th Workshop on Linked Data on the Web, volume 1184 of *CEUR Workshop Proceedings*, CEUR, 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[2] D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester, A. Dimou, Declarative RDF graph generation from heterogeneous (semi-)structured data: A systematic literature review, Journal of Web Semantics (2022). doi:10.1016/j.websem.2022.100753.

[3] F. Michel, L. Djimenou, C. Faron-Zucker, J. Montagnat, Translation of relational and non-relational databases into rdf with xr2rml, in: WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, SciTePress, Lisbon, Portugal, 2015, pp. 443–454. doi:10.5220/0005448304430454.

[4] J. Slepicka, C. Yin, P. Szekely, C. A. Knoblock, Kr2rml: An alternative interpretation of r2rml for heterogeneous sources, in: Proceedings of the 6th International Workshop on Consuming Linked Data (COLD 2015), volume 1426 of *CEUR Workshop Proceedings*, 2015. URL: http://ceur-ws.org/Vol-1426/paper-08.pdf.

[5] C. Debruyne, D. O'Sullivan, R2rml-f: Towards sharing and executing domain logic in r2rml mappings, in: Proceedings of the 9th Workshop on Linked Data on the Web, volume 1593 of *CEUR Workshop Proceedings*, 2016. URL: http://ceur-ws.org/Vol-1593/article-13.pdf.

[6] B. De Meester, T. Seymoens, A. Dimou, R. Verborgh, Implementation-independent Function Reuse, Future Generation Computer Systems 110 (2020) 946–959. URL: https://doi.org/10.1016/j.future.2019.10.006. doi:10.1016/j.future.2019.10.006.

[7] T. Delva, D. Van Assche, P. Heyvaert, B. De Meester, A. Dimou, Integrating nested data into knowledge graphs with RML fields, in: D. Chaves-Fraga, A. Dimou, P. Heyvaert, F. Priyatna, J. Sequeda (Eds.), Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), volume 2873, CEUR, 2021. URL: http://ceur-ws.org/Vol-2873/paper9.pdf.

[8] D. Van Assche, G. Haesendonck, G. De Mulder, T. Delva, P. Heyvaert, B. De Meester, A. Dimou, Leveraging Web of Things W3C Recommendations for Knowledge Graphs Generation, in: Web Engineering, 21st International Conference, ICWE 2021, Proceedings, 2021, pp. 337–352. URL: https://dylanvanassche.be/assets/pdf/icwe2021-wot-logical-target.pdf. doi:10.1007/978-3-030-74296-6_26.

[9] J. Arenas-Guerrero, A. Alobaid, M. Navas-Loro, M. S. Pérez, O. Corcho, Boosting knowledge graph generation from tabular data with RML views, in: The Semantic Web, Springer Nature Switzerland, 2023, pp. 484–501. doi:10.1007/978-3-031-33455-9_29.

[10] D. Van Assche, A. R. Julián, B. De Meester, P. Colpaert, Incremental knowledge graph construction from heterogeneous data sources, Semantic Web Journal (2024).

[11] A. Iglesias-Molina, D. Van Assche, J. Arenas-Guerrero, B. De Meester, C. Debruyne, S. Jozashoori, P. Maria, F. Michel, D. Chaves-Fraga, A. Dimou, The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF, in: Proceedings of the International Semantic Web Conference (ISWC), Lecture Notes in Computer Science, Springer, Cham, 2023, pp. 152–175. doi:10.1007/978-3-031-47243-5_9.

[12] J. Duchateau, C. Debruyne, Grape: Guiding RML authoring with a projectional editor, in: D. Chaves-Fraga, I. Dasoulas, C. Debruyne, A. Dimou, U. Serles, D. V. Assche (Eds.), Proceedings of the 6th International Workshop on Knowledge Graph Construction (KGCW 2025), volume 3999 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2025. URL: https://ceur-ws.org/Vol-3999/paper3.pdf.

[13] P. Heyvaert, A. Dimou, A.-L. Herregodts, R. Verborgh, D. Schuurman, E. Mannens, R. Van de Walle, RMLEditor: a graph-based mapping editor for Linked Data mappings, in: H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, P. S. Ponzetto, C. Lange (Eds.), The Semantic Web – Latest Advances and New Domains (ESWC 2016), volume 9678 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 709–723. URL: http://dx.doi.org/10.1007/978-3-319-34129-3_43. doi:10.1007/978-3-319-34129-3_43.

[14] A. C. Junior, C. Debruyne, D. O'Sullivan, Juma: An Editor that Uses a Block Metaphor to Facilitate the Creation and Editing of R2RML Mappings, in: E. Blomqvist, K. Hose, H. Paulheim, A. Ławrynowicz, F. Ciravegna, O. Hartig (Eds.), The Semantic Web: ESWC 2017 Satellite Events, volume 10577 of *Lecture Notes in Computer Science*, Springer, Cham, 2017, pp. 87–92. doi:10.1007/978-3-319-70407-4_17.

[15] P. Heyvaert, B. De Meester, A. Dimou, R. Verborgh, Declarative Rules for Linked Data Generation at your Fingertips!, in: The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15, Springer, 2018, pp. 213–217. URL: https://2018.eswc-conferences.org/files/posters-demos/paper_297.pdf.

[16] Vercruysse, Arthur and Rojas Melendez, Julian Andres and Colpaert, Pieter, The semantic web language server : enhancing the developer experience for semantic web practitioners, in: Curry, Edward and Acosta, Maribel and Poveda-Villalón, Maria and van Erp, Marieke and Ojo, Adegboyega and Hose, Katja and Shimizu, Cogan and Lisena, Pasquale (Ed.), The Semantic Web : 22nd European Semantic Web Conference, ESWC 2025, Proceedings, Part II, volume 15719, Springer, 2025, pp. 210–225. URL: http://doi.org/10.1007/978-3-031-94578-6_12.

[17] Debruyne, C. and Van Assche, Dylan, The conformance of an RML processor built from scratch to validate RML specifications and test cases, in: KGCW 2024 : Knowledge Graph Construction 2024 : Proceedings of the 5th International Workshop on Knowledge Graph Construction co-located with 21th Extended Semantic Web Conference (ESWC 2024), volume 3718, 2024, p. 4. URL: https://ceur-ws.org/Vol-3718/.