YAMAML: An Application Profile Based Lightweight RDF Mapping Language

Nishad Thalhath[1(✉)], Mitsuharu Nagamori[2], and Tetsuo Sakaguchi[2]

[1] Graduate School of Library, Information and Media Studies, Tsukuba, Japan
nishad@slis.tsukuba.ac.jp
[2] Faculty of Library, Information and Media Studies, University of Tsukuba, Tsukuba, Japan
{nagamori,saka}@slis.tsukuba.ac.jp
https://www.slis.tsukuba.ac.jp

Abstract. YAMA Mapping Language (YAMAML) is a lightweight mapping language for generating RDF. YAMAML is based on Yet Another Metadata Application Profiles (YAMA). YAMA is an extensible intermediary application profile authoring format for generating application profile expressions. Application profiles are a combination of vocabularies, which are mixed and matched from different namespaces and optimized for a particular local application. YAMA is based on Description Set Profiles (DSP), a Dublin Core Application Profiles constraint language. YAMA is implemented on YAML, one of the most human-readable data serialization formats. As a superset of JSON, YAML is highly interoperable and has parsers and emitters in all major programming languages. It adapts the basic application profile elements from YAMA and is designed as a simplified markup language to map non-RDF data structures to RDF and generate corresponding RDF based on the application profile. It is proposed as an intermediary format for generating RDF, but not as an RDF representation syntax. The authors demonstrate the capability of YAMAML by developing a basic specification and proof of concept implementations.

Keywords: Metadata · Application profiles · YAMA · RDF · Mapping language · Semantic web · Linked data

1 Introduction

The Resource Description Framework (RDF) is a standard directed labeled graph data format for representing information on the Semantic Web, an extension of the web. RDF is expressed in triples, consisting of a subject, a predicate, and an object [9]. RDF concept and related specifications were introduced by the World Wide Web Consortium (W3C) and are maintained by the W3C. The important formats of RDF include RDF/XML, RDFa, JSON-LD, and Turtle. The main difference between RDF and other data formats is that RDF uses a directed graph model, allowing for more flexible and powerful data representations. The advantages of RDF include its flexibility, extensibility, and interoperability. RDF has importance in the modern web because it provides a standard

Y.-H. Tseng et al. (Eds.): ICADL 2022, LNCS 13636, pp. 412–420, 2022.
https://doi.org/10.1007/978-3-031-21756-2_32

way to represent data that can be shared across different applications and platforms.

The difference between RDF and other data formats like csv and json is that RDF is a graph based data format. This means that data is represented as a set of interconnected nodes in a graph, as opposed to being represented as a table or set of key-value pairs. RDF is also a standard format, which means that there are well-defined rules for how data should be represented in RDF. This makes it easier to exchange data between different systems, and to query data using standard tools.

YAMA Mapping Language (YAMAML) is proposed to improve 5-star level[1] data publication with the notion that a profile-driven RDF generation can streamline the process by mapping multiple non-RDF sources to an RDF application profile of varying complexities.

1.1 Application Profiles, DCAP and DSP

Application profiles, often called Metadata Application Profiles, are a combination of vocabularies, which are mixed and matched from different namespaces and optimized for a particular local application [3]. Application profiles express the terms taken from other namespaces and the structural use of those terms in the local instance data. Application profiles also express constraints on those terms so that the data can be validated as well.

A Dublin Core Application Profile (DCAP) specifies how some metadata description sets are constructed. It includes information on the terms used in the description sets, how they are deployed, and constraints on the values and datatypes of the properties used. A DCAP is a declaration specifying which metadata terms an organization, information provider, or user community uses in its metadata. DCAPs can be used to document the semantics and constraints used for a set of metadata records or instance data. A DCAP can promote interoperability between different metadata models and harmonize metadata practices among different communities. A DCAP can also help communities of implementers harmonize metadata practice among themselves.

The Singapore Framework for Dublin Core Application Profiles [7] is a set of standards for designing metadata applications that are interoperable and reusable. The standards form a basis for reviewing Application Profiles for documentary completeness and conformance with Web-architectural principles. The standards define a set of descriptive components that are necessary or useful for documenting an Application Profile. The standards also describe how these documentary standards relate to standard domain models and Semantic Web foundation standards.

Description Set Profiles (DSP) is a constraint language for Dublin Core Application Profiles [6]. DSP is based on the DCMI Abstract Model (DCAM), which defines Description Set, Description, and Statement [8]. A DSP defines constraints on Description Sets, Descriptions, and Statements. Description Set

[1] https://www.w3.org/DesignIssues/LinkedData.

Templates hold one or more Description Templates composed of Statement Constraints. DSP supports the RDF-oriented data design with properties and datatypes.

1.2 Yet Another Metadata Application Profile (YAMA)

Yet Another Metadata Application Profile (YAMA) is a user-friendly interoperable preprocessor for creating, maintaining, and publishing Metadata Application Profiles [12], developed to be a direct adaption of DublinCore DSP. It is heavily inspired by the Simple-DSP (SDSP) format [5] for the MetaBridge project.[2]. Even though it helps to produce various formats and standards to express the application profiles, YAMA is not defined as a new standard for application profiles but as an easy-to-use preprocessor to create standard application profile formats; extensible [11] with custom elements and structure with a syntax based on YAML 1.2 specification[3]. However, it is parsable with any YAML 1.2 parser; the processing capabilities of the profile depend on implementations.

YAMAML is built with a minimal YAMA application profile concept, that a standard RDF can be expressed in an application profile with `descriptions` and `statements`.

1.3 Related Works

There are different attempts for Application Profiles and RDF mapping languages. A brief overview of the state-of-the-art is provided below.

DC Tabular Application Profiles (DC TAP) is a way to create application profiles in the form of tables. These tables can be read by humans and saved in a CSV format, which can be read by a computer program.[4]

Tarql: SPARQL for Tables is a command-line tool that uses SPARQL 1.1 syntax to convert CSV files to RDF.[5]

LinkML is a flexible modeling language that allows authors to create schemas in YAML that describe the structure of data. LinkML is also a framework for working with and validating data in a variety of formats (JSON, RDF, TSV) and can be used to compile LinkML schemas to other frameworks.[6]

R2RML is a language expressing customized mappings from relational databases to RDF datasets. [1] This language allows different mapping implementations, such as creating a virtual SPARQL endpoint over the mapped relational data, generating RDF dumps, or offering a Linked Data interface.

RDF Mapping Language (RML) is a mapping language that can express customized mapping rules from heterogeneous data structures and serializations to the RDF data model. RML is defined as a superset of the W3C-standardized mapping language R2RML [2].

[2] https://metabridge.jp/.
[3] https://yaml.org/spec/1.2.2/.
[4] https://github.com/dcmi/dctap.
[5] http://tarql.github.io.
[6] https://linkml.io.

YARRRML is a human-readable text-based representation for declarative Linked Data generation rules. It is a subset of YAML that can be used to represent R2RML and RML rules.[7]

CSV2RDF defines the procedures and rules for converting tabular data into RDF, including how metadata annotations can describe the structure, meaning, and interrelation of tabular data. [10]

RDF Transform is an extension for OpenRefine[8] that allows users to transform data into RDF formats. The RDF Transform extension provides a graphical user interface (GUI) for transforming OpenRefine project data to RDF-based formats. The transform maps the data with a template graph designed using the GUI.[9]

Among these attempts, DC TAP is to devise a method of creating application profiles in tabular format. Other mapping language attempts are for generating RDF data from different types of input sources, but they are not oriented to application profiles. Considering these facts, YAMAML is a novel approach in devising an RDF mapping language based on application profiles.

2 Methods

The major goals of this attempt are :

1. Derive a subset of YAMA to express minimal RDF as an application profile.
2. Use the derived subset as a general purpose RDF data mapping language, suitable for both RDF generation and minimal profiling.
3. Use data mapping in a descriptive and opinionated format to make RDF mapping easier.
4. Develop a set of ready-to-use and simplified tooling for basic RDF generation.

2.1 Modeling Application Profile of RDF with Data Mapping

Application profiles are constrainers as well as explainers of the data. A typical YAMA application profile also includes constraining options to help generate data validation formats and ensure data quality. These constraining elements, such as cardinality and value constraints, are not part of the modeling or generating RDF, but they help generate RDF validation formats such as Shape Expressions (ShEx)[10] and Shapes Constraint Language (SHACL) [4]. For modeling a minimal application profile for RDF, YAMA constraining elements were avoided for the YAMAML subset. Also, an application profile is intended to generate human-readable documentation of the profile. Explainer elements such as labels and notes were removed from the YAMA profile to create the subset.

[7] https://rml.io/yarrrml/.
[8] https://github.com/AtesComp/rdf-transform.
[9] https://openrefine.org.
[10] https://shex.io.

So the subset required to explain minimal RDF application profile structure is limited to 'descriptions' and 'statements' with essential parameters.

Minimal YAMA application profile, which is based on DSP is adapted for YAMAL as well. A basic overview of YAMAML mapping of the application profile to data is explained in Figure 1.

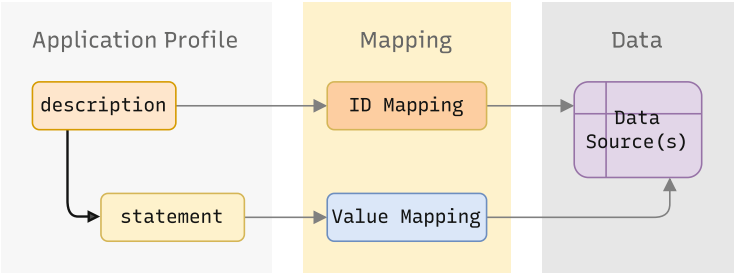

Fig. 1. YAMAML mapping overview - basic application profile and data mapping.

2.2 Data Mapping

YAMAML's data mapping is designed as multi-source capable, so users can use many data sources to generate a single RDF. The sources can be heterogeneous and require only a proper mapping to ID, so various data sources can be mixed and matched to create RDF of any level of complexity. A detailed description of all mapping elements is provided in Table 1.

YAMAML adapts all basic YAML collections from YAMA. Other than the YAMA collections base, namespaces, descriptions, and defaults, a special data collection is defined as a data holder. This optional collection container can store structured data as YAML or JSON. Since YAML is a superset of JSON, valid JSON is treated as valid YAML. All basic collection containers are explained in Table 2.

3 Results

YAMAML basic tooling and documentation are published at https://yamaml.org. The command line (CLI) toolkit can be used to generate relatively big and complex RDF. The playground web app is an in-browser environment, so it may not be sufficient for generating massive RDFs, but it will help to understand the basic implementation. A simple example of converting a basic CSV dataset is illustrated in Fig. 2. The command line tools for YAMAML are written in Javascript for Deno runtime.

Table 1. Elements in YAMAML data mapping

Element	Description	Type	Required	Examples
`source`	Path of the data source	string	yes	`/path/to/example.csv` http://ex.tld/ex.csv
`type`	Type of the data source	string	yes	Csv,xlsx,yaml,json
`path`	Path to the data element	string	yes	CSV column name, JSON/YAML key path
`strip`	Strip one or a group of characters, given as a list of strings	list	no	`[_," "]`
`replace`	Replace one or a group of characters with the given value. This should be a list of replace pair as a list	list	no	`[[0,1],[" ", -]]`
`separator`	A separator character to split the given values into a group of values	string	no	`,` or `|`
`prepend`	Prepend a text string to the value	string	no	
`append`	Append a text string to the value	string	no	

YAMA is an extensible application profile authoring environment. It was extended to cover many use-cases like versioning, application profile change log management, and provenance [13]. YAMA can be used to generate application profile expressions, documentation, and validation schemas, and now with the data mapping, it can also be an RDF generator. This attempt to subset YAMA as an RDF mapping language is aiding YAMA to be an application profile ecosystem for Semantic Web and Linked Data.

3.1 Comparison with State of the Art

DC TAP is focused more on authoring application profiles in a tabular way and is not extensible as YAMA. So it may not cover the use-case of RDF generation. Though DCTAP and YAMA are primarily based on Dublincore application profiles, YAMA follows the DC-DSP approach in modeling profiles. Thus YAMAML is modeled with a basic DC-DSP structure. Tarql requires a practical knowledge of SPARQL to map CSV to RDF. This is the potential limitation, where YAMAML is relatively simple to author a mapping. Though LinkML uses YAML as the serialization format, it has a steep learning curve. The same challenge is with YARRRML, which demands proper knowledge of R2ML and RML concepts. CSV2RDF requires CSVW to map the data, and it would be challenging to model complex RDF structures with CSVW. OpenRefine is the easiest option for RDF conversion with powerful data transformation capabilities. It has a user-friendly GUI and a sound reconciliation system with Wikidata support. OpenRefine RDF addon requires modeling the data in a certain way, which is not correctly equivalent to the application profile. In short, with all limitations, YAMAML tries to be a minimal, easy-to-use application profile-based RDF mapping language.

Table 2. YAMAML Containers

Component	Description
`base`	Same as RDF base, preferred a URI as value and will be treated as the base of the generated RDF
`namespaces`	A discriminatory of key-value pairs indicating prefix and URI of the namespace
`descriptions`	Same as YAMA descriptions, which holds the basic application profile in YAMA format
`defaults`	An optional container for declaring default values for the descriptions and data mapping
`data`	An optional YAMAML-specific structured data holder. Data can be serialized in JSON or YAML format. If the data can be included within the YAMAML file instead of external sources, this data holder can be used for that

4 Discussion

YAMA is proposed as an extensible format [11] so that it can be extended to cover more use-cases and specific needs. With YAML's flexibility, it can be a handy format for various RDF and linked data-related projects. Most of these requirements need custom tooling, which can help grow YAMA as an application profile ecosystem. So the authors believe that an RDF generation mapping language will be an added advantage in adapting YAMA for many real-world scenarios.

4.1 Limitations of This Approach

YAMAML mapping depends on declaring IDs for the descriptions; at least the main or initial description requires an ID mapping. ID can be any common data element similar to the primary key and foreign key concept of traditional relational database systems. This approach is a significant limitation and forces the users to pre-process their data with proper relationships. Another issue is that modeling complex RDF will require essential skills and time. Since YAMAML tries to be simple enough for basic uses, many advanced use-cases and edge cases were not considered in the design decisions. Though YAMAML can be used to map linked data using IRI stems, it is not intended to do a reconciliation of entities to any linked dataset. Tools like OpenRefine can do reconciliation and RDF generation from a GUI environment.

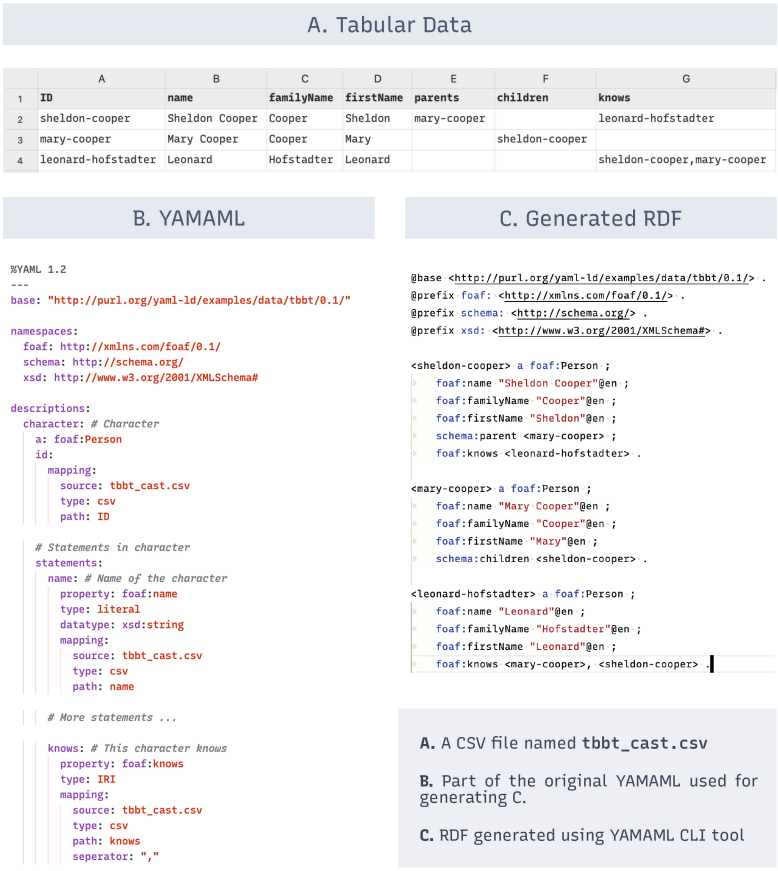

Fig. 2. YAMAML mapping example - a simplified example to demonstrate application profile mapping to a single CSV file with minimal data.

5 Conclusion

There are various tools and mapping languages for RDF, though some have high learning curves, and some are complex for basic use-cases. GUI-oriented tools like OpenRefine helps novice users to convert their data to RDF. The adaption of RDF will be less painstaking and popular if there are many tools from which the users can freely choose something that suits their needs. Many of these attempts have overlapping feature sets but still provide many unique features and options. The authors are optimistic that more accessible tools will eventually help to grow semantic web-oriented data sharing and expand the linked data cloud with more 5-star open data.

Acknowledgements. This work was supported by JSPS KAKENHI Grant Number 21K12579.

References

1. Das, S., Cyganiak, R., Sundara, S.: R2RML: RDB to RDF mapping language. W3C recommendation, W3C (2012). https://www.w3.org/TR/2012/REC-r2rml-20120927/
2. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: RML: A generic language for integrated RDF mappings of heterogeneous data. In: LDOW (2014)
3. Heery, R., Patel, M.: Application profiles: mixing and matching metadata schemas. Ariadne 25 (2000). http://www.ariadne.ac.uk/issue/25/app-profiles/
4. Kontokostas, D., Knublauch, H.: Shapes constraint language (SHACL). W3C recommendation, W3C (2017). https://www.w3.org/TR/2017/REC-shacl-20170720/
5. Nagamori, M., Kanzaki, M., Torigoshi, N., Sugimoto, S.: Meta-bridge: a development of metadata information infrastructure in Japan. In: Proceedings International Conference on Dublin Core and Metadata Applications 2011, p. 6 (2011)
6. Nilsson, M.: DCMI: description set profiles: a constraint language for dublin core application profiles (2008). http://www.dublincore.org/specifications/dublin-core/dc-dsp/
7. Nilsson, M., Baker, T., Johnston, P.: DCMI: the Singapore framework for Dublin core application profiles (2008). http://dublincore.org/specifications/dublin-core/singapore-framework/
8. Powell, A., Nilsson, M., Naeve, A., Johnston, P., Baker, T.: DCMI: DCMI abstract model (2007). http://www.dublincore.org/specifications/dublin-core/abstract-model/
9. Raimond, Y., Schreiber, G.: RDF 1.1 primer. W3C note, W3C (2014). https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
10. Tandy, J., Herman, I., Kellogg, G.: Generating RDF from tabular data on the web. W3C recommendation, W3C (2015). https://www.w3.org/TR/2015/REC-csv2rdf-20151217/
11. Thalhath, N., Nagamori, M., Sakaguchi, T., Sugimoto, S.: Authoring formats and their extensibility for application profiles. In: Jatowt, A., Maeda, A., Syn, S.Y. (eds.) ICADL 2019. LNCS, vol. 11853, pp. 116–122. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34058-2_12
12. Thalhath, N., Nagamori, M., Sakaguchi, T., Sugimoto, S.: Yet another metadata application profile (YAMA): authoring, versioning and publishing of application profiles. In: International Conference on Dublin Core and Metadata Applications, pp. 114–125 (2019). https://dcpapers.dublincore.org/pubs/article/view/4055
13. Thalhath, N., Nagamori, M., Sakaguchi, T., Sugimoto, S.: Metadata application profile provenance with extensible authoring format and PAV ontology. In: Wang, X., Lisi, F.A., Xiao, G., Botoeva, E. (eds.) JIST 2019. LNCS, vol. 12032, pp. 353–368. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41407-8_23