Odruba: Ontology-driven Rule-based Visualisation

Andreas Nareike
Leipzig University
Augustusplatz 10
04109 Leipzig
nareike@informatik.uni-leipzig.de

Johannes Schmidt
Leipzig University
Augustusplatz 10
04109 Leipzig
jschmidt@informatik.uni-leipzig.de

ABSTRACT

Data visualisation techniques have gained growing importance within the last years, not only for data scientists. To visualise RDF data, a transformation into graph description languages is needed. The result of this process is often not satisfactory, because most of the semantics get lost and customisation options are limited. This paper presents an ontology-driven, rule-based visualisation approach to flexibly transform and visualise RDF data as a graph while taking advantage of current developments in semantic data modelling and reasoning.

1. INTRODUCTION

For statistical analysis or data science, visualisation is an essential tool. It can help to see important connections or discover trends at one glance. A large number of visualisation frameworks and applications exist that can be utilised in different application areas.

Apart from statistical analysis, visualisation of connections between resources or individuals is another point of interest, especially in the field of social networks. In a scientific context, graph visualisations can help to identify previously unnoticed relations. Several graph algorithms provide automatic clustering of strongly interlinked nodes to increase the usability to a human user.

The semantic web heavily relies on graphs and networks, e.g. graph databases and RDF. There exist large RDF knowledge bases like the DBpedia, the Linked Movie Database or UniProt. RDF data can easily be transformed to more general graph formats like DOT, GraphML or GEXF that are widely supported by graph visualisation tools.

This conversion results in a loss of the semantics when dealing with graphs or knowledge bases that use RDFS or OWL concepts. While nodes in graphs can have types or belong to groups, hierarchies of RDFS classes or properties are not easily carried over to graph description languages. Additionally, class membership of resources in an RDF graph does not have to be stated explicitly but can also be entailed by other triples (e.g. via a domain or range statement). A suitable visualisation approach should make use of the semantics of the data.

We present an ontology-driven, rule-based visualisation (Odruba) approach to control the depiction of RDF graphs by exploiting RDFS, OWL and rule mechanisms making use of basic relational properties like transitivity or symmetry. The specific style of the visualisation depends on rule sets that allow adding reasoned implicit connections or resources, suppressing information or highlighting missing information. We further discuss the application of Odruba for decision-making processes.

SEMANTICS 2016: Posters and Demos Track
September 13-14, 2016, Leipzig, Germany

2. RELATED WORK

Graph analysis and visualisation is well examined in literature. Many tools exist that help to process, analyse and render data in various formats as a graph.

The multi-platform tool Gephi [1] provides many algorithms and filters to render an insightful graph. It allows clustering nodes automatically and provides different colouring functions based on the output of the algorithms. There are plugins for Gephi that can import RDF data from files or a SPARQL endpoint. Some RDF properties can be translated to Gephi specific attributes (e.g. `rdfs:label` to node labels) but there is no further processing with respect to the RDFS and OWL semantics.

Cytoscape is a cross-platform software to render graphs. It is very much oriented towards bioinformatics. Nevertheless, the algorithms provided are applicable to a wide variety of data. Feature-wise, Cytoscape is comparable to Gephi. As well as Gephi, it does not provide native support for RDF but can be extended with so-called 'apps' to load RDF data. One such app is SemScape [7] that focuses on flexible graph exploration. A user can extend a graph by including the results of SPARQL queries at run time. The visualisation options however are completely controlled by Cytoscape and do not take into account the RDFS/OWL semantics.

IsaViz [4] is a visualization and authoring tool for RDF data. To control the visualisation result, they introduce Graph Style Sheets (GSS) [5]. GSS defines selectors similar to Cascading Style Sheets (CSS) to determine the styling of resources based on the triples in which they appear. Logically, GSS uses techniques similar to reification combined with Horn clauses to apply visualisation options to classes of triples. Although GSS is an RDF application, its semantics are not defined in RDFS, OWL, SPARQL or a rule extension, but within the application itself. This is by no means necessary. Rule languages like SWRL [3] provide the expressive power to define the formal semantics of such constructs. Alternatively, SPARQL CONSTRUCT queries can make the semantics explicit. Either approach (i.e. rules or SPARQL) would result in the ability to process GSS with established tools. IsaViz/GSS does not take into account implicit information. The GSS selectors address RDF data

only on a syntactical level. To our knowledge, the Graph Style Sheets proposed by IsaViz are not used by other tools. Furthermore, the software is no longer under development since 2007.

There are a number of tools that can render a graphical representation of OWL ontologies like OWLGReD, VOWL, Lodlive or OWLViz. These tools are well aware of the RDFS and OWL semantics and can translate them into visual constructs. They are often used to explore ontologies. So, the visualisation is mostly concerned with classes and their relationships. Their goal is to provide a somewhat standardised visualisation that makes different ontologies easier to compare. A specific style for selected resources is not supported.

In this paper, we are interested in a generic approach to control the visualisation by providing styling information. However, the expressiveness of a visualisation depends on the design as well as the number and density of graphical elements. To obtain suitable graph representation, both challenges must be addressed. Two recent approaches (RDF4U [2], RDF2graph [9]) focus on the second challenge and explore techniques to condense RDF graphs by merging nodes and/or omitting edges.

In conclusion, there are roughly two classes of tools: On the one hand, there are tools that provide flexible visualisation options. They are usually not aware of the RDFS and OWL semantics and do not use reasoning techniques. On the other hand, there are tools that are well aware of RDFS and OWL semantics. These usually only provide one type of visualisation for terminological data to make ontologies easier to understand.

3. MOTIVATIONAL EXAMPLE

We illustrate the core ideas of Odruba using a simple example. Listing 1 shows a Turtle serialization of a small RDF graph with five triples. The FOAF ontology is referenced as well as the FaBiO ontology, which is a bibliographic ontology designed for the use in libraries.

```
ex:Alice foaf:knows ex:Bob ;
    a foaf:Person ;
    foaf:made ex:Insp_16-06 ;
    rdfs:label "Alice" .

ex:Insp_16-06 a fabio:TechnicalReport .
```

Listing 1: A small RDF Turtle graph

We want to stress, that we do not want to prescribe any kind of visualisation. Instead, we present a methodological approach to define and customize a visualisation that is best suited for a given use case. This often comprises presenting information stored in RDF to an audience that is not familiar with details of RDF.

A simple visualisation can be achieved by mapping each triple to a pair of nodes connected by an edge that is labelled with the predicate URI. The solid black elements in Figure 1 show this visualisation.

We are not very interested in representing the raw triples, but the encoded information. Our central premise for this work is: Not every triple needs to be rendered as an edge connecting two nodes. In some cases, aggregated representation of resources or additional reasoned information should be rendered. Nevertheless, we still use a resource-centric approach in the sense that each node in the visualised graph represents an RDF resource.

As a first extension, literals are directly attached to the node of the resource they describe. In an interactive graph, this could be realized as a tooltip. Especially RDFS labels (`rdfs:label`) can be used to label nodes and edges. Additional labels can be imported from the referenced ontologies instead of reiterating them.

Classes and their instances are both represented as nodes in Figure 1, with a directed `rdf:type` edge connecting them. The visual complexity can be reduced by using colours or icons to represent the type of a resource instead of a connected class or type node. For example, a human icon could be used for persons (i.e. instances of `foaf:Person`) and a book icon for works (`fabio:Work`). Because a resource can belong to multiple classes of an ontology, a suitable representation must be determined for every use case. Alternatively, they can have multiple visualisation properties like an icon combined with a specific colour.

Figure 1 shows in dashed, red lines, how we annotate resources with additional information to describe the desired visualisation. The fact that Alice (`ex:Alice`) is a person (`foaf:Person`) is explicitly included in our example. The triples in the FOAF ontology entail that not only Alice is a person, but Bob is one too. The FaBiO ontology further states, that `fabio:TechnicalReport` is a subclass of `fabio:Report` which in turn is a subclass of `fabio:Work`. From this follows that `ex:Insp_16-06` should be represented by a book icon.

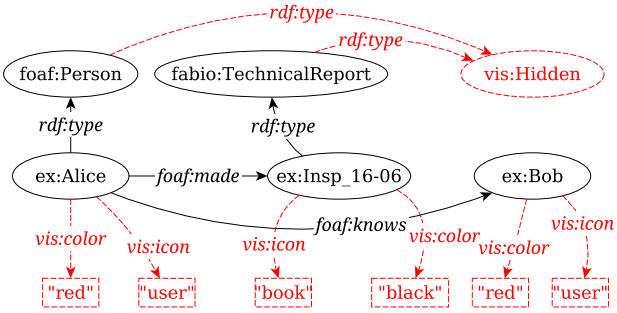

Figure 1: An annotated graph (labels are omitted)

In order to automatically get the desired visualisation information, an RDFS reasoner is needed. It is not sufficient to only include the referenced ontologies. On the other hand, not all implicit statements are required. Which additional triples must be taken into account depends on the desired visualisation and the given triples.

For instance, the statement that Alice writes an inspection report could be included in a background knowledge base (not in the RDF graph to be rendered). Nevertheless, it might be required that persons who write reports should have different icons (e.g. a human holding a pen). In this case, the additional information that Alice authors a report needs to be taken into account.

With Odruba, we present a tool that can calculate these annotations based on a set of rules. Odruba uses reasoning techniques and also takes into account RDFS/OWL semantics when processing these rules. In a second step, Odruba translates the fully annotated RDF graph into a graph description language that can be rendered to a graph similar

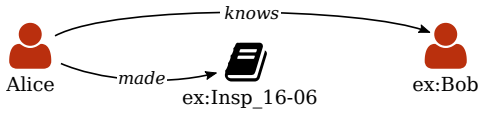

Figure 2: Visualisation realized

to Figure 2 by an external tool.

4. TECHNICAL SETUP

For Odruba, we use the Jena framework to work with RDF data as well as with RDFS and OWL semantics. Additionally, the internal Jena reasoner is used to work with Jena rules. Other reasoners like Pellet [6] or TrOWL [8] will be evaluated in the future.

We decided to use a JavaScript library for the visualisation. There are several libraries available that address graph rendering like D3, sigma.js or vis.js. As a starting point, we decided to use vis.js, because it provides a good documentation and supports a flexible configuration of the visualisation. In general, the JSON formats of different JavaScript libraries are very similar to each other. Consequently, a migration to another library can be achieved easily.

In order to connect the Jena back end and the JavaScript front end, a HTTP RESTful back end service is used, to provide JSON objects, which subsequently can be rendered by vis.js. Additionally, there are services to get an RDF representation of the rendered graph or to get JSON data that expands the existing graph. This setup is flexible enough to provide additional services for other kind of visualisations.

4.1 A Visualisation Ontology

Since the back end has to handle knowledge about the details of the visualisation, it is a natural approach to encode this information in triples. We use a custom ontology that is largely influenced by the configuration data formats of graph visualisation frameworks. To our experience, the structure and semantics of the visualisation configuration settings are very similar.

Vis.js provides different methods to attach styling information to nodes and edges. One can define a default style that is used when a graphical object (i.e. a node or an edge) has no further styling details. Secondly, nodes and edges can belong to groups. Groups themselves can provide styling information for nodes and edges. The default styling can be seen as a group that contains all graphical objects that have no group assignment. Lastly, nodes and edges can be customised directly.

Groups provide a concise method to style a number of graphical elements at once. The group membership has to be explicitly set on each graphical object. Consequently, we need an ontology that can attach styling properties or group memberships directly to RDF resources.

This might be deemed controversial for resources from external namespaces over which one has no authoritative power. For us, there is no greater problem here. Firstly, visualisations are often used in an internal or limited use case. It is usually not the goal to provide global styling information. Secondly, our technique can be adapted or expanded to work with techniques like reification similar to Graph Style Sheets. The ontology we use can be seen as a working draft and will be re-evaluated and refined in the future.

We briefly list the key concepts of our visualisation ontology (cf. Figure 1). We use properties to control the size (`vis:size`) of rendered nodes as well as the colour (`vis:color`), the shape (`vis:shape`) or alternatively the icon (`vis:icon`). For labelling, we use `vis:label` or optionally another property (e.g. `rdfs:label`). Apart from properties, we also use classes to determine which properties are hidden (`vis:Hidden`) and which are potential expansion points (`vis:OutgoingExpanding` and conversely `vis:IncomingExpanding`).

The Odruba approach also has some limitations. As styling attributes are attached to resources, literals cannot be styled directly. We usually do not include literals as nodes, since they are often leaves with only one edge connecting to them. As a workaround for simple use cases, we assign a special group to nodes that represent literals. A second limitation refers to RDF properties in general, because they are usually not instantiated in the RDF data model. In order to have different colours for edges that represent the same RDF property, it is necessary to use a basic reification with `rdf:subject`, `rdf:property` and `rdf:object`. While there are other reification techniques, this variant is best supported by the Jena framework.

4.2 Rules and Reasoner

It is possible to provide styling information for each resource by hand. This is only a practical approach for small graphs. Visualisation tools usually provide methods to define groups of nodes with respect to a given criteria. For instance, in our motivational example above, there could be a group for persons and a group for books. However, in the resulting RDF graph, the group membership must be set explicitly.

Conceptually, this can be seen as horn clauses: Whenever a resource satisfies one or more conditions, a specific style attribute is set for this resource. The OWL semantics provide only limited support for this kind of modelling. There have been some proposals to expand OWL with rules, most prominent the Semantic Web Rule Language (SWRL) and the SPARQL Inference Notation (SPIN). Since we are using the Jena framework, we use Jena Rules, that are somewhat similar to SWRL rules.

4.3 Example Continued

In the following, the Odruba approach is applied on the example we introduced in section 3. Firstly, we load the graph into a simple Jena model. Secondly, the required ontologies (i.e. FOAF and FaBiO) have to be included into a Jena ontology model that is stacked on top of the Jena model. Finally, we add a Jena inference model set up with the following rules:

```
[work: (?s rdf:type fabio:Work)
 -> (?s vis:icon "book"), (?s vis:color "black") ]

[person: (?s rdf:type foaf:Person)
 -> (?s vis:icon "user"), (?s vis:color "red") ]
```

Listing 2: A set of Jena rules

The Jena rule syntax is straightforward. For example, whenever a resource is of type `fabio:Work`, infer the triples that are listed after the arrow symbol (`->`).

5. TOWARDS DECISION SUPPORT

Our method is versatile enough to allow support of decision-making processes, e.g. by providing tailored visualisations of data excerpts to experts using domain-specific icons and colours. We examine mechanism that can help a decision-maker to collect relevant details. For example, one can start from a small graph or a single resource. Depending on a set of configuration options, properties can be hidden. To support explorative browsing, properties can be defined to be expansion points of the graph. In this case, the complete graph or a single node can be expanded along these properties.

When expanding a node, a closure of the new graph is calculated with respect to chosen properties by including edges between the newly added nodes and the nodes that existed in the graph before the expansion.

To continue our running example, we further assume that Bob is a technician who requires the inspection report created by Alice (`ex:Insp_16-06`). This information can be included as a triple into the graph:

```
ex:Bob ex:requires ex:Insp_16-06 .
```

Listing 3: A single triple

Based on this fact, one could guess, that there is a dependence between Bob and Alice. Generally speaking, they have a producer-consumer relationship. This can be formalised by the following rule:

```
[dependsOn:
    (?b ex:requires ?o), (?a foaf:made ?o)
 -> (?b ex:dependsOn ?a) ]
```

Listing 4: Jena rule specifying dependence

In the resulting visualisation (see Figure 3), the connection between Bob and Alice is highlighted, because it is inferred. Additionally, a trace of the rules applied can be displayed.

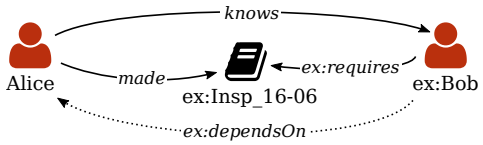

Figure 3: Graph with inferred edges

6. CONCLUSION

The visualisation of RDF graphs and RDFS/OWL semantics are tightly connected. While there are tools and frameworks that can work with these semantics, none of them provide a flexible way to render customisable visualisations. We presented Odruba as a method to leverage the semantics of RDFS/OWL in combination with rule-based reasoning to enrich an RDF graph with visualisation information. Lastly, we briefly discussed how Odruba can be used to support decision-making processes.

7. ACKNOWLEDGMENTS

This work is a result of the CVtec research project, supported by the German Federal Ministry of Education and Research (BMBF) as grant 01IS14016C.

8. REFERENCES

[1] M. Bastian, S. Heymann, M. Jacomy, and others. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.

[2] R. Chawuthai and H. Takeda. RDF Graph Visualization by Interpreting Linked Data as Knowledge. In G. Qi, K. Kozaki, J. Z. Pan, and S. Yu, editors, *Semantic Technology*, number 9544 in Lecture Notes in Computer Science, pages 23–39. Springer International Publishing, Nov. 2015.

[3] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, and others. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21:79, 2004.

[4] E. Pietriga. Isaviz: a visual environment for browsing and authoring rdf models. In *Eleventh International World Wide Web Conference Developers Day*, page 68, 2002.

[5] E. Pietriga. Semantic web data visualization with graph style sheets. In *Proceedings of the 2006 ACM symposium on Software visualization*, pages 177–178. ACM, 2006.

[6] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[7] A. Splendiani, A. Waagmeester, C. Haupt, and H. Deus. SemScape: Visualizating Semantic Web Data Landscapes with Cytoscape 3.0. 2012.

[8] E. Thomas, J. Z. Pan, and Y. Ren. TrOWL: Tractable OWL 2 reasoning infrastructure. In *Extended Semantic Web Conference*, pages 431–435. Springer, 2010.

[9] J. C. van Dam, J. J. Koehorst, P. J. Schaap, V. A. M. dos Santos, and M. Suarez-Diez. RDF2graph a tool to recover, understand and validate the ontology of an RDF resource. *Journal of biomedical semantics*, 6(1):1, 2015.

APPENDIX

A. PREFIX DEFINITION

We define the following prefixes for the document at hand:

```
@prefix ex: <http://example.org/> .
@prefix vis: <http://vis.example.org/> .
@prefix rdf:
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:
   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix fabio: <http://purl.org/spar/fabio/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```