Knowledge Representation for the Distributed, Social Semantic Web

Named Graphs, Graph Roles and Views in NRL

Michael Sintek[1], Ludger van Elst[1], Gunnar Grimnes[1],
Simon Scerri[2], Siegfried Handschuh[2]

[1] Knowledge Management Department
German Research Center for Artificial Intelligence (DFKI) GmbH,
Kaiserslautern, Germany
{firstname.surname}@dfki.de
[2] DERI, National University of Ireland, Galway
{firstname.surname}@deri.org

Abstract. The vision of the Semantic Web mainly aims at scenarios with multiple, distributed and autonomous information providers and consumers. However, the current standards for knowledge representation mainly see the Semantic Web as one big knowledge base without explicitly acknowledging the distributed, social nature of knowledge in the technological basics.
Originally coming from the specific requirements of the idea of a Social Semantic Desktop, we identified two questions as being fundamental also for the general Semantic Web endeavor: i) How can we cope with the heterogeneity of knowledge models and ontologies, esp. multiple knowledge modules with potentially different interpretations? ii) How can we support the tailoring of ontologies towards different needs in various exploiting applications?
In this paper, we present the NEPOMUK Representational Language (NRL), an approach to tackle these two question that is based on named graphs for the modularization aspect and a view concept for the tailoring of ontologies. This view concept turned out to be of additional value, as it also provides a mechanism to impose different semantics on the same syntactical structure.
In addition to the general construction of NRL we describe our first implementations on top of a standard Semantic Web representation framework, Sesame2. A first benchmark of this implementation shows that the level performance can be acceptable to many Semantic Web developers.

1 Motivation: Distributed Information on the Web

Semantic Web technology is intended to provide a middleware for a wide variety of applications in domains such as e-business, e-government, personal and organizational knowledge management, *etc.* Despite this diversity, the envisioned applications have in common that typically the (distributed) *data sources are*

generated and maintained by autonomous entities and that the problems to be solved ask for a comprehensive, *integrated use of data*. The current solution approach is to provide *protocol standards* as well as *data and information representation standards* for federated information access and services. The main technology for dealing with the problem of heterogeneity are *ontologies*, *i. e.*, representation schemes which are shared between several information providers and consumers.

While first applications show the prospects of the Semantic Web approach, widespread real-world use or "killer applications" are still missing. In early motivational work (*e. g.*, [4]), flexible, agent-supported problem solving has been envisioned, but today's applications have still a relatively tight, explicitly engineered coupling between the nodes of the current Semantic Web. This holds for the procedural aspects (services and protocols) just as for the representation aspects (ontologies and data). From theoretical analysis on the distributed nature of knowledge [13] as well as from practical experiences with implementing applications based on Semantic Web technologies, we identified two questions which we conjecture to be essential on the way to a more comprehensive Semantic Web success and which we try to tackle in the knowledge representation approach presented in this paper:

1. How can we cope with the heterogeneity of knowledge models and ontologies, esp. multiple knowledge modules with potentially different interpretation schemes?
2. How can we support the tailoring of ontologies towards different needs in various exploiting applications?

The first question is rooted in the fact that with heterogeneous generation and exploitation of knowledge there is no "master instance" which defines and ensures the "interpretation sovereignty." The second question turned out to be an important prerequisite for a clean ontology design for (distributed, social) Semantic Web scenarios, as many, diverse applications with different goals and intended user groups will have to access the ontologies and knowledge bases and (re)present them adequately.

From these general questions, we identified the following four main requirements for knowledge representation for the (distributed, social) Semantic Web:

Handling of multiple models: In order to adequately represent the social dimension of distributed knowledge generation and usage [13], a module concept is desirable which supports encapsulation of statements and the possibility to refer to such modules. The social aspect requires a support for provenance and trust information, when it comes to importing and exporting data. With the present RDF model, importing external RDF data from a remote application presents some difficulties, mainly revolving around the fact that there are no standard means of retaining provenance information of imported data. This means that data is propagated over multiple applications, with no information regarding the original provider and other crucial information like the context under which that data is valid. This can result in various situations like ending

up with outdated RDF data with no means to update it, as well as redundant RDF data which cannot be entirely and safely removed.

Multiple semantics: The main principle of the Semantic Web is that it is an open world in which documents can add new information about existing resources. Since the Web is a huge place in which everything can link to anything else, it is impossible to rule out that a statement could be true, or could become true in the future. Hence, the global Semantic Web relies on a open-world semantics, with no unique-name assumption—the official OWL and RDF/S semantics. On the other hand, within a single application using Semantic Web technology people find it very difficult to understand the logical meaning and consequences of the open-world assumption; the closed-world and unique-name assumptions are much easier to understand for most users. Therefore we require a scenario where we can always distinguish between data per se and the semantics or assumptions on that data. If these are handled analogously, it is possible to build applications that handle local data as a closed world, while processing external data with open world (or other) semantics.

Multiple views: Also required by the social aspect is the support for multiple views, since different actors (both persons and technical agents) might be interested in different aspects of the data. A view is dynamic, virtual data computed or collated from the original data. The best view for a particular purpose depends on the information the user needs.

Epistemological adequacy of modeling primitives: In many knowledge-intensive scenarios (*e. g.*, the Social Semantic Desktop idea [7]), knowledge modeling is not only performed offline (*e. g.*, by a distinguished knowledge engineer), but also by the end user, much like in the tagging systems of the Web 2.0 where a user can continuously invent new vocabulary for describing his information items. Even if much of the complexity of the underlying representation formalism can be hidden by adequate user interfaces, it is desirable that there is no big *epistemological gap* between the way an end-user would like to express his knowledge and the way it is represented in the system.

In this paper, we sketch the concept and current implementation of the NEPOMUK Representation Language (NRL) which was developed to fulfill the previously stated requirements. In the next section, we will briefly discuss the state of the art which served as input for the NEPOMUK Representation Language (NRL). Sec. 3 gives an overview of our approach, especially the *Named Graphs* for handling multiple models and the *Graph Views* for imposing different semantics on and application-oriented tailoring of models. In Sec. 4, we present our current implementation on top of a standard RDF store. An example (Sec. 5) shows how the concepts presented in this paper can be applied. Sec. 6 summarizes the NRL approach and discusses next steps.

2 State of the Art

The Resource Description Framework [9] and the associated schema language RDFS [5] set a standard for the Semantic Web, providing a representational

language whereby resources on the web can be mapped to designated classes of objects in some shared knowledge domain, and subsequently described and related through applicable object properties. With the gradual acceptance of the Semantic Web as an achievable rather than just an ideal World Wide Web scenario, and adoption of RDF/S as the standard for describing and manipulating semantic web data, there have been many attempts to improve some RDF/S shortcomings to handling such data. Most were in the form of representational languages that extend RDF/S, the most notable of which is OWL [2]. Other work attempted to provide further functionalities on top of semantic data to that provided by RDF/S by revising the RDF model itself.

The most successful idea perhaps is the named graph paradigm, where identifying multiple RDF graphs and naming them with distinct URIs is believed to provide useful additional functionality on top of the RDF model. Given that named graphs are manageable sets of data in an otherwise structureless RDF triple space composed of all existent RDF data, most of the practical problems arising from dealing with RDF data, like dealing with invalid or outdated data as well as issues of provenance and trust, could be addressed more easily if the RDF model supports named graphs. The RDF recommendation itself does not provide suitable mechanisms for talking about graphs or define relations between graphs [3, 9, 5, 8]. Although the extension of the RDF model with named graph support has been proposed [6, 11, 10], and the motivation and ideas are clearly stated, a concrete extension to the RDF model supporting named graph has not yet materialized. So far, a basic syntax and semantics that models minimal manipulation of named graphs has been presented by participants of the Semantic Web Interest Group.[3] Their intent is to introduce the technology to the W3C process once initial versions are finalized.

The SPARQL query language [10], currently undergoing standardization by the W3C, is the most successful attempt to provide a standard query language for RDF data. SPARQL's full support for named graphs has encouraged further research in the area. The concept of modularized RDF knowledge bases (in the spirit of named graphs) plus views that can be used to realize the semantics of a module (with the help of rules), amongst other things, has been introduced in the Semantic Web rule language TRIPLE [11].

Since the existing approaches are incomplete wrt. the needs of most (distributed) Semantic Web scenarios, we propose a combination of named graphs and TRIPLE's view concept as the basis for NRL, the representational language we are presenting. In contrast to TRIPLE, we will add the ability to define views as an extension of RDF and named graphs at the ontological level, thus we are not dependent on a specific rule formalism as in the case of TRIPLE.

In the rest of this paper, we will give a detailed description of the named graphs and views features of NRL. Other features of NRL (which consist of some RDFS extensions mainly inspired by Protégé and OWL) will not be discussed.

[3] http://www.w3.org/2004/03/trix/

3 The NRL Approach

NRL was designed to fulfill the requirements for the NEPOMUK Social Semantic Desktop project,[4] hence the particular naming, but it is otherwise domain independent.

As discussed in the previous sections, the most notable shortcoming of the RDF model is the lack of support for handling multiple models. In theory Named Graphs solve this problem since they are identifiable, modularized sets of data. Through this intermediate layer handling RDF data, *e. g.*, exchanging data and keeping track of data provenance information, is much more manageable, which is very important for many distributed applications. Alongside provenance data, more useful information can be attached to named graphs. In particular we feel that named graphs should be distinguished by their roles,[5] *e. g.*, Ontology or Instance Base.

Users may be interested in different aspects of data in a named graph at different times. Looking at, *e. g.*, the contents of an image folder, the user might wish to see related concepts for an image, or any other files related to it, but not necessarily both concurrently even if the information is stored in the same graph. Additionally, advanced users might require to see data that is not usually visible to regular users, like additional indirect concepts related to the file. This would require the viewing application to realize the RDF/S semantics over the data to yield more results. The application is therefore required to work with extended or restricted versions of named graphs in different situations. However, we believe that such manipulations over named graphs should not have a permanent impact on the data in question. Conversely, we believe that the original named graph should be independent of any kind of workable interpretation executed by an application, which can be discarded if and when they are no longer needed.

For this reason, we present the concept of Graph Views as one of the core concepts in NRL. By allowing for arbitrary tailored interpretations for any established named graph, graph views fulfill our idea that named graphs should not innately carry any realized semantics or assumptions, unless they are themselves views on other graphs for exactly that purpose, and that they should remain unchanged and independent of any view applied on them. This means that different semantics can be realized for different graphs if required. In practice, different applications will require to apply different semantics, or assumptions on semantics, to named graphs. In this way, although the semantic desktop operates in a closed world, it is also possible to work with open-world semantic views over a graph. Importing a named graph with open-world semantics is therefore possible. If required (and meaningful), closed-world applications can then work with a closed-world semantics view over the imported graph.

Fig. 1 gives an overview of the NRL components, depicting both the syntactical and the semantic blocks of NRL. The syntax box contains, in the upper

[4] http://nepomuk.semanticdesktop.org/

[5] Note that the term "role" has a completely different meaning in description logics where it simply denotes a "property."

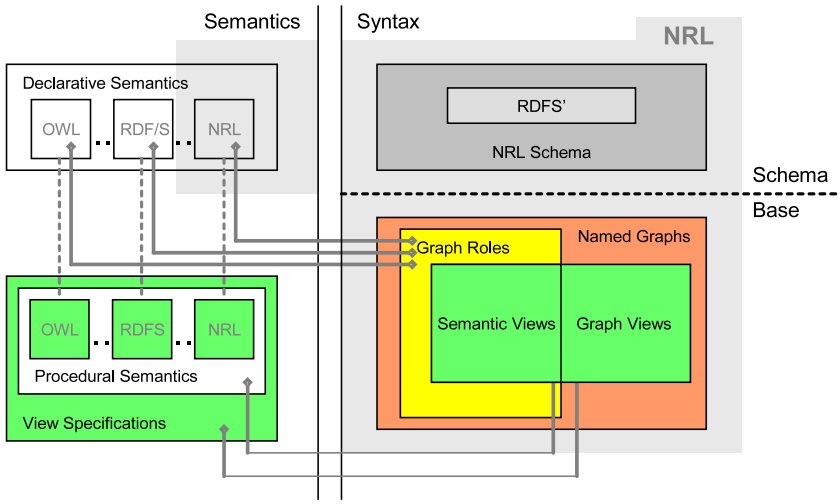

Fig. 1. Overview of NRL—Abstract Syntax, Concepts and Semantics.

part, the NRL Schema language, which is mainly an extension of a large subset of RDFS. The lower part shows how named graphs, graph roles, and views are related.

The left half of the figure explains the NRL semantics, which has a declarative and a procedural part. Declarative semantics is linked with graph roles, *i. e.*, roles are used to assign meaning to named graphs (note that not all named graphs or views must be assigned some declarative semantics, *e. g.*, in cases when the semantics is (not) yet known or not relevant). Views are also linked to view specifications, which function as a mechanism to express procedural semantics, *e. g.*, by using a rule system. The procedural semantics has, of course, to realize the declarative semantics that is assigned to a semantic view.

3.1 Handling Multiple Models: NRL Named Graphs

Named graphs (NGs) are an extension on top of RDF, where every distinct RDF graph is identified by a unique name. NGs provide additional functionality on top of RDF particularly with respect to metametadata (metadata about metadata), provenance, and data (in)equivalence issues, besides making data handling more manageable. Our approach is based on the work described in [6] excluding, however, the open-world assumption stated there. As stated earlier (*cf.* Sec. 3) we believe that named graphs should not innately carry any realized semantics or assumptions on the semantics. Closed-world, open-world, and other forms of semantics can instead be realized through designated views on graphs.

A named graph is a pair (n, g), where n is a unique URI reference denoting the assigned name for the graph g. Such a mapping fixes the graph g corresponding to n in a rigid, non-extensible way. The URI representing n can then be used

from any location to refer to the corresponding set of triples belonging to the graph g. A graph g' consistent[6] with a distinct graph g named n cannot be assigned the same name n.

An RDF triple can exist in a named graph or outside any named graph. However, for consistency reasons, all triples must be assigned to some named graph. For this reason NRL provides a special named graph, `nrl:DefaultGraph`. Triples existing outside any named graph are considered part of this default graph. This ensures backward compatibility with triples that are not based on named graphs. This approach gives rise to the term RDF Dataset as defined in [10]. An RDF dataset is composed of a default graph and a finite number of distinct named graph, formally defined as the set $\{g, (n_1, g_1), (n_2, g_2), ..., (n_n, g_n)\}$ comprising of the default graph g and zero or more named graphs (n_i, g_i).

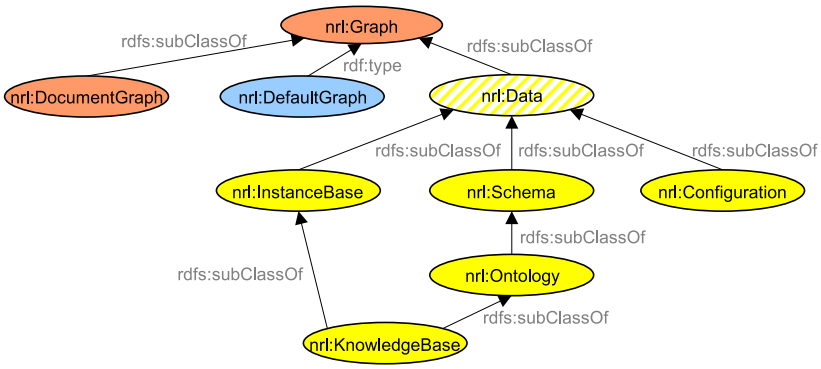

Fig. 2. NRL Named Graph Class Hierarchy.

NRL distinguishes between graphs and graph roles, in order to have orthogonal modeling primitives for defining graphs and for specifying their role. A graph role refers to the characteristics and content of a named graph (*e. g.*, simple data, an ontology, a knowledge base, *etc.*) and how the data is intended to be handled. The NEPOMUK Graph Metadata vocabulary (NGM)[7] provides a vocabulary for annotating graph roles. Graph metadata will be attached to roles rather than to the graphs themselves, because its more intuitive to annotate an ontology, for example, rather than the underlying graph. Roles are more stable than the graphs they represent.

Fig. 2 depicts the class hierarchy supporting NGs in NRL. Graph roles are defined as specialization of the general graph representation `nrl:Data`. A special graph, `nrl:DocumentGraph`, is used as a marker class for graphs that are represented within and identified by a document UR, thus allowing existing RDF files to be re-used as named graphs which avoids the need of a syntax

[6] Two different datasets asserting two unique graphs but having the same URI for a name contradict one another.

[7] NGM will not be described in this paper.

like TriG[8] to define named graphs. `nrl:subGraphOf`, `nrl:superGraphOf`, and `nrl:equivalentGraph` are relations between named graphs having the obvious semantics: they are defined as $\subseteq$, $\supseteq$, and $=$ on the bare triple sets in these graphs. `nrl:imports` is a subproperty of `nrl:superGraphOf` and models graph imports. Apart from implying the $\supseteq$ relation between the triple sets, it also requires that the semantics of the two graphs is compatible if used on, *e. g.*, graphs that are ontologies. `nrl:DefaultGraph`, an instance of `nrl:Graph`, represents the graph containing all triples existing outside any user-defined named graph. Since we do not apply any semantics to triples automatically, this allows views to be defined on top of triples defined outside of all named graphs analogously to the named-graph case.

`nrl:Data` represents the most generic role that a graph can have, namely that it contains data. `nrl:Schema` and `nrl:Ontology` (a subclass of `nrl:Schema`) are roles for graphs that represent data in some kind of conceptualization model. `nrl:InstanceBase` marks a named graph to contain instances from schemas or ontologies. The properties `nrl:hasSchema` and `nrl:hasOntology` relate an instance base to the corresponding schema or ontology. `nrl:KnowledgeBase` marks a named graph as containing a conceptual model plus instances from schemas or ontologies. `nrl:Configuration` is used to represent technical configuration data that is irrelevant to general semantic web data within a graph. Other additional roles serving different purposes might be added in the future. `nrl:Semantics` is used to specify the declarative semantics for a graph role by referring to instances of this class via `nrl:hasSemantics`. These will usually link (via `nrl:semanticsDefinedBy`) to a document specifying the semantics in a human readable or formal way (*e. g.*, the RDF Semantics document [8]).

3.2 Imposing Semantics on Graphs: NRL Graph Views

A named graph consists only of the enumerated triples in the triple set associated with the name, and does not inherently carry any form of semantics (apart from the basic RDF semantics). However in many situations it is desirable to work with an extended or restricted interpretation of simple syntax-only named graphs. These can be realized by applying some algorithm (*e. g.*, specified through rules) which enhances named graphs with entailment triples, returns a restricted form of the triple set, or an entirely new triple set. To preserve the integrity of a named graph, interpretations of one named graph should never replace the original. To model this functionality and retain the separation between original named graph and any number of their interpretations, we introduce the concept of *Graph Views*.

Views are different interpretations for a particular named graph. Formally, a view is an executable specification mapping an input graph to a corresponding output graph. Informally, they can be seen as arbitrary wrappings for a named graph. Fig. 3 depicts graph view support in NRL. Views are themselves named graphs, allowing us to have a named graph that is a different interpretation, or

[8] http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriG/

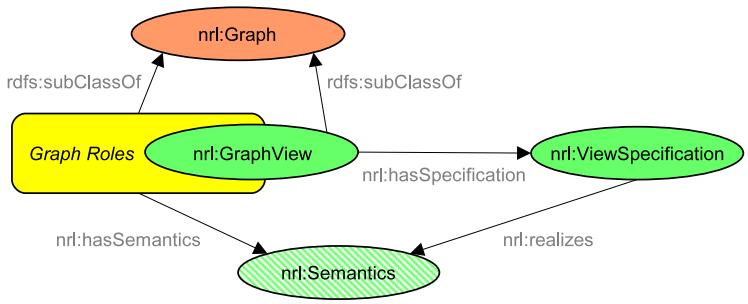

Fig. 3. Graph Views in NRL.

view, of another named graph. This modeling can be applied recurrently, yielding a view of a view and so on.

View specifications define the view realization for a view, via a set of queries/rules in a query/rule language (*e. g.*, a SPARQL query over a named graph), or via an external application (*e. g.*, an application that returns the transitive closure of `rdfs:subClassOf`). As in the latter example, view realizations can also realize the implicit semantics of a graph according to some schema language (*e. g.*, RDFS, OWL, NRL). We refer to these as *Semantic Views*, represented in Fig. 3 by the intersection of `nrl:GraphView` and graph roles. One can draw a parallel between this figure and Fig. 1. In contrast to graph roles, which have only declarative semantics defined through the `nrl:hasSemantics` property, semantic views also carry procedural semantics, since the semantics of these graphs are always realized, (through `nrl:realizes`) and not simply implied.

4 Implementation

We have implemented NRL on top of the open-source RDF store Sesame2.[9] Sesame2 has an architecture based on several Storage and Inference Layers (Sail) stacked on top of each other to allow flexible customization of the features needed from a store. We have implemented NRL as a combination of two Sails designed to be stacked on top of one of Sesame2's storage sails (*i. e.*, the *MemoryStore* or *NativeStore*). The first sail is called *UnionSail* and adds support for virtual union graphs, *i. e.*, where one virtual named graph is composed of several other graphs. Sesame2 has built in support for doing SPO-queries and size queries over several contexts, and our UnionSail adds rewriting of queries to query the appropriate named graphs. Our second Sail builds on the UnionSail, and handles the details of NRL views and inference. This Sail offers two additional methods in addition to the required Sail methods, one for importing a named graph into another, and one for creating a new view on a named graph. The View specifications are based on rules, allowing creation of custom views as well as the provided

[9] http://www.openrdf.org/

rule-base for NRL semantics (not detailed in this paper). The NRLSail uses a forward-chaining rule engine for performing the inference. The rule engine is an extended version of the engine from Jena,[10] and uses the same rule format.

When a new view is created the entailments over the base graph are computed immediately. The rule engine will compute several strata of entailments, and each strata is stored in a separate named graph. At the moment, we use a plain semi-naive bottom-up evaluation which is a traditional evaluation strategy used in logic programming [1]. This technique can easily be enhanced by more advanced strategies like magic set transformation.

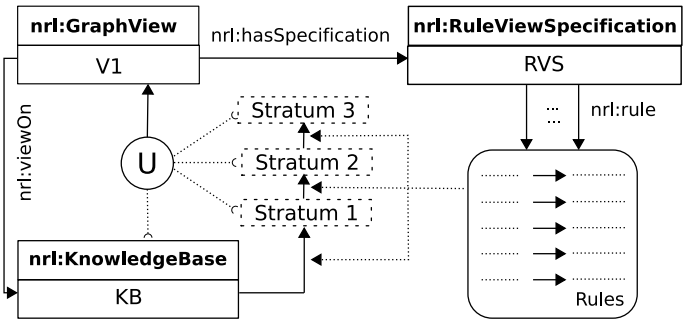

Fig. 4. A Graph View and the Associated Strata.

In the end, the named graph of the view will be a virtual graph consisting of the union of the base graph (if desired) and each of the strata, as shown in Fig. 4. The NRLSail will watch for changes to the base graph and will perform addition inference whenever further triples are added.

4.1 Evaluation

The NRLSail has not yet been thoroughly evaluated, however we have carried out initial experimentation with a data-set based on a Personal Information Model (PIMO) from a user of the Semantic Desktop System NEPOMUK. The data-set consists of 10850 triples, describing 1800 instances of a hierarchical taxonomy of personal information management concepts, such as projects, people, etc. We compared our NRLSail stacked on top of the Sesame2 NativeSail, which provides persistent storage, with Jena2.4 backed by a MySQL database. For each solution we loaded the triples, then created a view with RDFS semantics over these (*i. e.*, in Jena we created an *InfModel* using the *RDFSForwardRuleReasoner*). We then evaluated three complex queries typical for the NEPOMUK system over this model.[11] This simple benchmark showed that our NRLSail out-

[10] http://jena.sourceforge.net/

[11] The three queries were: 1. Get all resources that are instance of `pimo:Thing`, their label and class. 2. Get all wiki pages, their title, version number, content, last change

performs Jena with a factor of about six, both for the initial inference step and for the subsequent querying. Note that this does in any way not constitute a rigorous evaluation: we used the default settings for both RDF stores and large optimizations might be possible by tweaking the settings; note also that Jena includes much more performant inferencing engines than the simple forward-rule reasoner, but we felt comparing two forward chaining reasoners to be most appropriate. Also, creating a single view does not really reflect a real deployment of NRL, and performance might be much worse for a larger data-set with large numbers of imported named graphs. Finally, we believe that Sesame2 generally outperforms Jena, and we cannot claim that our implementations accounts for the factor of six speedup.

5 Example: NRL in Use

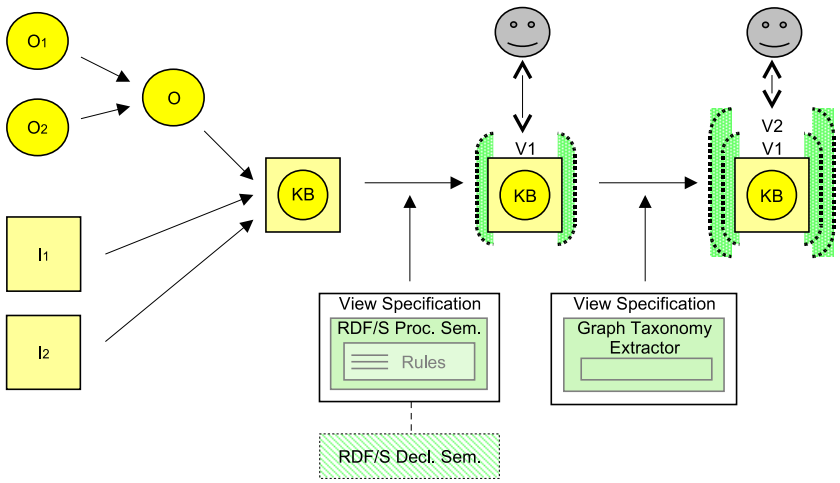

Fig. 5. NRL Example.

In this section, we demonstrate the utilization of the various NRL concepts in a more complex scenario, depicted in Fig. 5. An application requires an online knowledge base to be made available. A generic base ontology already exists, this ontology is stored in the named graph O_1. A different source supplies data consisting of instances in this ontology. This is stored as a named graph with the role of instance base, I_1. However, the base ontology is not sufficiently detailed and a SW knowledge engineer is hired to model another ontology that defines

date, and last author. 3. Get all relations that relate two things, the label and class of the things being related, and the label of the relation.

further classes not captured in O_1, and this is stored as another named graph, O_2. The final application requires concepts from both ontologies, so the engineer merges O_1 and O_2 in the required conceptualization by creating a named graph O as an ontology and defining it as supergraph of O_1 and O_2. Furthermore, a number of instances of O_2 is compiled in an instance base, I_2.

The application can now combine schematic data from the graph O, and the instances from I_1 and I_2 into new graph, KB, acting as a knowledge base and make this graph available for querying. However, in many scenarios it is also necessary to realize semantics over an ontology to make full use of class-hierarchies, etc. This can be achieved by creating a view over KB, using a view specification that realizes the procedural semantics for RDFS using a rule language of choice. Executing these rules over the triples in KB results in the semantic view $V_1(KB)$, which consists of all the RDF triples in KB plus the generated entailment triples. The separation between the underlying model and the model with the required semantics is thus retained. Note that the original KB is of course still available if RDFS semantics are not required.

Finally, the application also has to cater for less experienced users and it is desirable to hide some of the details contained in $V_1(KB)$. Discarding useful information from $V_1(KB)$ permanently is of course not ideal, so a knowledge engineer creates another view $V_2(V_1(KB))$, this time using an external view specification. It is worth to note that all seven named graphs on which this last view is generated upon are still intact and have not been affected by any of the operations along the way. If the knowledge engineer requires to apply some different semantics over KB, it may still be done since generating $V_1(KB)$ did not have an impact on KB.

Fig. 6 shows the example in TriG[12] which is a plain text format created for serializing Named Graphs and RDF Datasets. For additional detail on the NRL elements used in this example please refer to [12].

6 Summary and Outlook

In this paper, we described those parts of the NEPOMUK Representational Language (NRL) which are rooted in the requirements risen by the *distributed knowledge representation and heterogeneity aspects* of distributed, social Semantic Web applications and which we think cannot satisfactorily be dealt with by the current state of the art. In a nutshell, the basic arguments and design principles of NRL are as follows:

– Due to the heterogeneity of the data creating and consuming entities, a single interpretation schema cannot be assumed. Therefore, NRL aims at a *strict separation between data (sets of triples, graphs) and their interpretation/semantics.*

[12] http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriG/; TriG is a straight-forward extension of Turtle (http://www.dajobe.org/2004/01/turtle/)

```
[1] @prefix nrl: <http://semanticdesktop.org/ontology/nrl-yyyymmdd#> .
    @prefix ex: <http://www.example.org/vocabulary#> .
[2] ex:o2 rdf:type nrl:Ontology .
[3] <http://www.domain.com/o1.rdfs> rdf:type nrl:Ontology ,
                        nrl:DocumentGraph .
[4] ex:o1 rdf:type nrl:Ontology ;
         nrl:equivalentGraph <http://www.domain.com/o1.rdfs> .
[5] ex:o rdf:type nrl:Ontology ;
         nrl:imports ex:o1, ex:o2 .
[6] ex:i2 rdf:type nrl:InstanceBase ;
         nrl:hasOntology ex:o2 .
[7] http://www.anotherdomain.com/i1.rdf> rdf:type nrl:InstanceBase ,
                                                  nrl:DocumentGraph .
[8] ex:kb rdf:type nrl:KnowledgeBase ;
      nrl:imports ex:o, ex:i2, <http://www.anotherdomain.com/i1.rdf> .
[9] ex:o2 {
      ex:Animal rdf:type rdfs:Class .
             ## further Animal Ontology definitions here ## }
[10]ex:i2 {
     ex:CandyCaneWorm rdf:type ex:Flatworm ;
              ## further Animal Instance definitions here ## }
[11] ex:v1kb rdf:type nrl:KnowledgeBase, nrl:GraphView ;
            nrl:viewOn ex:kb ; nrl:superGraphOf ex:kb ;
            nrl:hasSpecification ex:rvs .
[12] ex:rvs rdf:type nrl:RuleViewSpecification ;
            nrl:realizes ex:RDFSSemantics ; nrl:ruleLanguage "SPARQL" ;
            nrl:rule "CONSTRUCT {?s rdfs:subClassOf ?v} WHERE ..." ;
            nrl:rule "CONSTRUCT {?s rdf:type ?v} WHERE ..." .
[13] ex:RDFSSemantics rdf:type nrl:Semantics ; rdfs:label "RDFS" ;
            nrl:semanticsDefinedBy "http://www.w3.org/TR/rdf-mt/" .
[14] ex:v2v1kb rdf:type nrl:GraphView, nrl:KnowledgeBase ;
            nrl:viewOn ex:v1kb ; nrl:hasSpecification ex:evs .
[15] ex:evs rdf:type nrl:ExternalViewSpecification ;
            nrl:externalRealizer "GraphTaxonomyExtractor" .
```

Fig. 6. NRL Example—TriG Serialization.

- Imposing specific semantics to a graph is realized by generating *views* on that graph. Such a generation is directed by an (executable) *view specification* which may realize a declarative semantics (*e. g.*, the RDF/S or OWL semantics specified in a standardization document).
- Graph views cannot only be used for semantic interpretations of graphs, but also for application-driven tailoring of a graph.[13]
- Handling of multiple graphs (with different provenance, ownership, level of trust, ...) is essential. *Named graphs* are the basic means to this problem.
- Graphs can play different roles in different contexts. While for one application a graph may be an ontology, another one may see it as plain data. These *roles* can explicitly be specified.

While originally designed as a NEPOMUK internal standard for the Social Semantic Desktop, we believe that the arguments also hold for the general Semantic Web, especially when we review the current trends which more and more show a development from the view of "the Semantic Web as one big, global knowledge base" to "a Web of (machine and human) actors" with local perspectives and social needs like trust, ownership, *etc.*

[13] This corresponds to a database-like view concept.

A first, prototypical (and incomplete) implementation of NRL based on Sesame2 and Jena rules has been developed, and is released as open-source.[14] We will continue this implementation and evaluate it in the context of NEPOMUK and also other, distributed social Semantic Web applications.

Acknowledgements This work was supported by the European Union IST fund (Grant FP6-027705, Project NEPOMUK) and by the German Federal Ministry of Education, Science, Research and Technology (bmb+f), (Grant 01 IW F01, Project Mymory: Situated Documents in Personal Information Spaces). The authors would especially like to thank all contributors to NEPOMUK's ontology taskforce.

References

1. I. Balbin and K. Ramamohanarao. A generalization of the differential approach to recursive query evaluation. *J. Log. Program.*, 4(3):259–262, 1987.
2. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinnes, P. Patel-Schneider, and L. Stein. OWL web ontology language reference, 2004.
3. D. Beckett. RDF/XML syntax specification (revised). W3C recommendation, W3C, February 2004. http://www.w3.org/TR/rdf-syntax-grammar/.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 89, May 2001.
5. D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF Schema. Technical report, W3C, February 2004. http://www.w3.org/TR/rdf-schema/.
6. J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM Press.
7. S. Decker and M. Frank. The social semantic desktop. In *Proc. of the WWW2004 Workshop Application Design, Development and Implementation Issues in the Semantic Web*, 2004.
8. P. Hayes. RDF semantics. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/rdf-mt/.
9. F. Manola and E. Miller. RDF primer. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/rdf-primer/.
10. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C working draft, W3C, 2005. http://www.w3.org/TR/rdf-sparql-query/.
11. M. Sintek and S. Decker. TRIPLE—A query, inference, and transformation language for the Semantic Web. In *1st International Semantic Web Conference (ISWC2002)*, June 2002.
12. M. Sintek, L. van Elst, S. Scerri, and S. Handschuh. Distributed knowledge representation on the social semantic desktop:named graphs, views and roles in nrl. In *Proceedings of the 4th European Semantic Web Conference (ESWC)*, 2007.
13. L. van Elst, V. Dignum, and A. Abecker. Towards agent-mediated knowledge management. In L. van Elst, V. Dignum, and A. Abecker, editors, *Agent-Mediated Knowledge Management International Symposium AMKM 2003, Stanford, CA, USA, March 24-26, 2003, Revised and Invited Papers*, volume 2926 of *LNAI*, pages 1–31. Springer, Heidelberg, 2004.

[14] NRLSail, https://dev.nepomuk.semanticdesktop.org/wiki/NRLSail