Distributed Knowledge Representation on the Social Semantic Desktop: Named Graphs, Views and Roles in NRL

Michael Sintek[1], Ludger van Elst[1], Simon Scerri[2], and 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 *Social Semantic Desktop* defines a user's personal information environment as a source and end-point of the Semantic Web: Knowledge workers comprehensively express their information and data with respect to their own conceptualizations. Semantic Web languages and protocols are used to formalize these conceptualizations and for coordinating local and global information access. From the way this vision is being pursued in the NEPOMUK project, we identified several requirements and research questions with respect to knowledge representation. In addition to the general question of the expressivity needed in such a scenario, two main challenges come into focus: 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 NRL, an approach to 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.

We think that the elements of our approach are not only adequate for the semantic desktop scenario, but are also of importance as building blocks for the general Semantic Web.

1 Motivation: The Social Semantic Desktop

The very core idea of the Social Semantic Desktop is to enable data interoperability on the personal desktop based on Semantic Web standards and technologies, *e. g.*, ontologies and semantic metadata. The vision [6] aims at integrated personal information management as well as at information distribution and collaboration, envisioning two expansion states: i) the *Personal Semantic Desktop* for personal information management and later ii) the *Social Semantic Desktop* for distributed information management and social community aspects.

E. Franconi, M. Kifer, and W. May (Eds.): ESWC 2007, LNCS 4519, pp. 594–608, 2007.

In traditional desktop architectures, applications are isolated islands of data—each application has its own data, unaware of related and relevant data in other applications. Individual vendors may decide to allow their applications to interoperate, so that, *e. g.*, the email client knows about the address book. However, today there is no consistent approach for allowing interoperation and a system-wide exchange of data between applications. In a similar way, the desktops of different users are also isolated islands—there is no standardized architecture for interoperation and data exchange between desktops. Users may exchange data by sending emails or upload it to a server, but so far there is no way of seamless communication from an application used by one person on their desktop to an application used by another person on another desktop. The problem on the desktop is similar to that on the Web.

The Social Semantic Desktop paradigm adopts the ideas of the Semantic Web (SW) paradigm [3], which offers a solution for the web. Formal ontologies capture both a shared conceptualization of desktop data and personal mental models. RDF (Resource Description Format) serves as a common data representation format. Together, these technologies provide a means to build the semantic bridges necessary for data exchange and application integration. The Social Semantic Desktop will transform the conventional desktop into a seamless, networked working environment, by loosening the borders between individual applications and the physical workspace of different users. By aligning the Social Semantic Desktop paradigm with the Semantic Web paradigm, a Semantic Desktop can be seen as source and end-point of the Semantic Web.

This viewpoint of the user comprehensively generating, manipulating and exploiting private as well as shared and public data has to be adequately reflected in the representational basis of such a system. While we think in general the assumptions of knowledge representation in the Semantic Web are a good starting point the Semantic Desktop scenario generates special requirements. We identified two core questions 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 on the semantic desktop, as many applications shall use a knowledge worker's "personal ontology."

From these general questions, we specialized the following five main requirements for knowledge representation on the Social Semantic Desktop:

Epistemological adequacy of modeling primitives: In the Social Semantic Desktop scenario, 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.

Integration of open-world and closed-world assumptions: The main principle of the SW 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 semantic, with no unique-name assumption—the official OWL and RDF/S semantics. On the other hand, the main principle on the personal Semantic Desktop is that it is a closed-world as it mainly focuses on personal data. While most people find it difficult to understand the logical meaning and potential inferences statements of the open-world assumption, the closed-world assumption is easier to understand for the user. Hence, the Personal Semantic Desktop requires the closed-world semantics with a unique-name assumption or good smushing techniques to achieve the same effects. The next stage of expansion of the personal semantic desktop is the Social Semantic Desktop, which connects the individual desktops. This will require open-world semantics (in between desktops) with local closed-world semantics (on the personal desktop). Thus the desktop needs to be able to handle external data with open-world semantics. 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, the semantic desktop, a closed-world in theory, will also be able to handle data with open-world semantics.

Handling of multiple models: In order to adequately represent the social dimension of distributed knowledge generation and usage [12], 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 another desktop 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 desktops, 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: As stated before, the aspect of distributed (and independently created) information requires the support of the open-world assumption (as we have it in OWL and RDF/S), whereas local information created on a

single desktop will have closed-world semantics. Therefore, applications will be forced to deal with different kinds of semantics.

Multiple views: Also required by the social aspect is the support for multiple views, since different individuals on different desktops 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.

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. The following sections elaborate on two important aspects of NRL, the *Named Graphs* for handling multiple models (Sec. 4) and the *Graph Views* for imposing different semantics on and application-oriented tailoring of models (Sec. 5). In Sec. 6, we present an example which shows how the concepts presented in this paper can be applied. Sec. 7 summarizes the NRL approach and discusses next steps.

2 State of the Art

The Resource Description Framework [8] and the associated schema language RDFS [4] 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 where in the form of representational languages that extend RDF/S, the most notable of which is OWL [1]. 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 [2,8,4,7]. Although the extension of the RDF model with named graph support has been proposed [5,11,9], 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.[1] Their

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

intent is to introduce the technology to the W3C process once initial versions are finalized. The SPARQL query language [9], 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 NEPOMUK and most Semantic Web scenarios in general, 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 Knowledge Representation on the Social Semantic Desktop: The NRL Approach

NRL was inspired by the need for a robust representational language for the Social Semantic Desktop, that targets the shortcomings of RDF/S. NRL was designed to fulfill requirements for the NEPOMUK Social Semantic Desktop project,[2] hence the particular naming, but it is otherwise domain-independent.

As discussed in the previous section, 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. This has a great influence in the social aspect of the Social Semantic Desktop project, since the success of this particular aspect depends largely on how to successfully deal with these issues. All data handling on the semantic desktop including storage, retrieval and exchange, will therefore be carried out through RDF graphs. 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, *e. g.*, Ontology or Instance Base.

Desktop users may be interested in different aspects of data in a named graph at different times. Looking at the contents of an image folder for instance, 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

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

file. This would require the viewing application to realize the RDF/S semantics over the data to yield more results. The desktop system 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 application on the semantic desktop 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 predefined open-world semantics on the semantic desktop is therefore possible. If required (and meaningful), closed-world applications can then work with a closed-world semantics view over the imported graph.

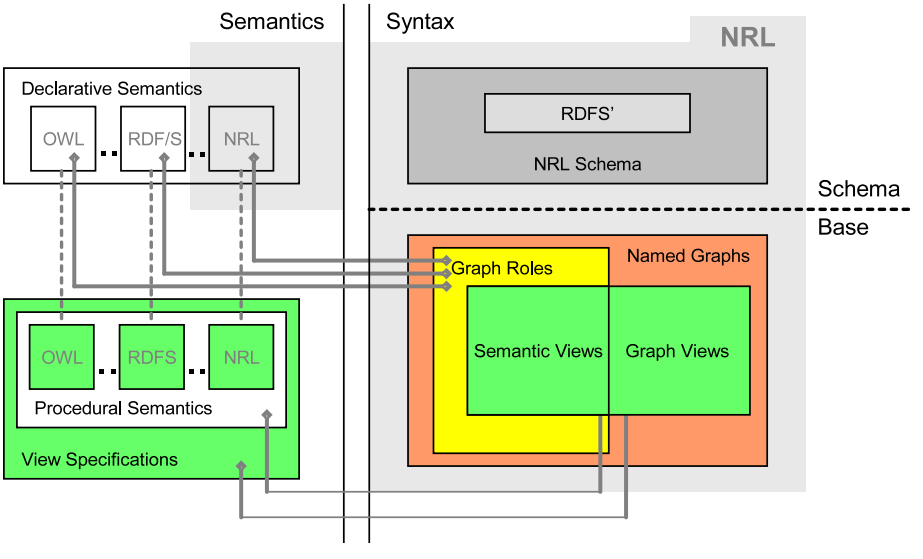

Fig. 1. Overview of NRL—Abstract Syntax, Concepts and Semantics

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

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, which will be explained in detail in the rest of this paper.

The left half of the figure sheds some light on the semantics of NRL, 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 simply 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.

4 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 [5] 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. Therefore, despite being designed as a requirement for the Semantic Desktop, which operates under a closed-world scenario, NRL itself does not impose closed-world semantics on data. This and other 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[3] 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 [9]. 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).

NRL distinguishes between graphs and graph roles, in order to have orthogonal modeling primitives for defining graphs and for specifying their role. A

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

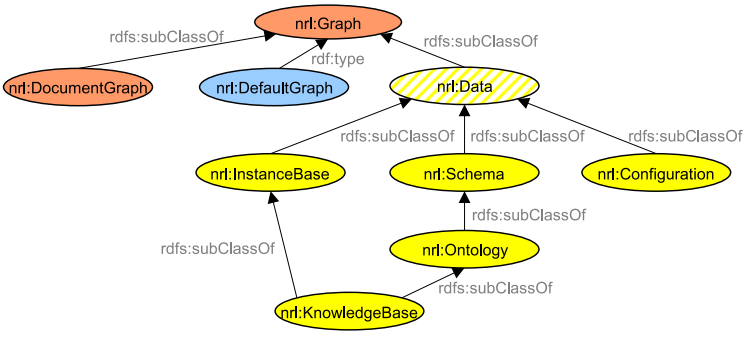

Fig. 2. NRL Named Graph Class Hierarchy

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)[4] 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, and while the graph for a particular role might change constantly, evolution of the role itself is less frequent. An instantiation of a role represents specific type of graph and the corresponding triple set data.

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 URL. We now present the NRL vocabulary supporting named graphs. General graph vocabulary is defined in Sec. 4.1 while Sec. 4.2 is dedicated entirely to graph roles.

4.1 Graph Core Vocabulary

nrl:Graph and nrl:DocumentGraph Instances of these classes represent named graphs. The name of the instance coincides with the name of the graph. The graph content for a `nrl:DocumentGraph` is located at the URL that is the URIref for the `nrl:DocumentGraph` instance. This allows existing RDF files to be re-used as named graphs, avoiding the need of a syntax like TriG[5] to define named graphs.

nrl:subGraphOf, nrl:superGraphOf, and nrl:equivalentGraph. These relations between named graphs have 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

[4] NGM will not be described in this paper.
[5] http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriG/

that the semantics of the two graphs is compatible if used on, *e. g.*, graphs that are ontologies.

nrl:DefaultGraph This 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.

4.2 Graph Roles Vocabulary

nrl:Data This subclass of `nrl:Graph` is an abstract class to make graph roles easy-to-use marker classes. It represents the most generic role that a graph can have, namely that it contains data.

nrl:Schema and nrl:Ontology are roles for graphs that represent data in some kind of conceptualization model. `nrl:Ontology` is a subclass of `nrl:Schema`.

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 Declarative semantics for a graph role can be specified 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 [7]).

5 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 of an input graph into a corresponding output graph. Informally, they can be seen as arbitrary wrappings for a named graph.

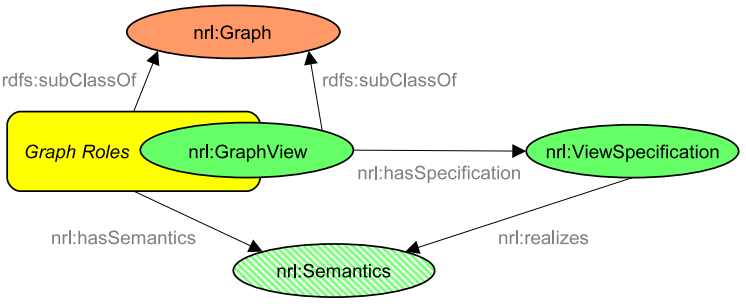

Fig. 3. Graph Views in NRL

Fig. 3 depicts graph view support in NRL. Views are themselves named graphs. Therefore one can have a named graph that is a different interpretation, or view, of another named graph. This modeling can be applied recurrently, yielding a view of a view and so on.

View specifications can execute 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 language or schema (*e. g.*, RDFS, OWL, NRL *etc.*). 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.

5.1 Views Vocabulary

In this section we briefly present the NRL vocabulary supporting graph view specifications.

nrl:GraphView represents a view, modeled as a subclass of named graph. A view is realized through a view specification, defined by an instance of `nrl:ViewSpecification` via `nrl:hasSpecification`. The named graph on which the view is being generated is linked by `nrl:viewOn`. The separation between different interpretations of a named graph and the original named graph itself is thus retained.

nrl:ViewSpecification This class represents a general view specification, which can currently take one of two forms, modeled as the two subclasses `nrl:RuleViewSpecification` and `nrl:ExternalViewSpecification`. As discussed earlier, semantic views realize procedural semantics and are linked to some semantics via `nrl:realizes`. This is however to be differentiated

from `nrl:hasSemantics`, which states that a named graph carries (through a role) declarative semantics which is not necessarily (explicitly) realized via a view specification.

nrl:RuleViewSpecification Views can be specified by referring to a rule language (via `nrl:ruleLanguage`) and a corresponding set of given rules (via `nrl:rule`). These views are realized by executing the rules, generating the required output named graph.

nrl:ExternalViewSpecification Instances of this class map to the location of (via `nrl:externalRealizer`) an external application, service, or program that is executed to create the view.

6 Example: NRL in Use

In this section, we demonstrate the utilization of the various NRL concepts in a more complex scenario: Ella is a biologist and works as a senior researcher at Institute Pasteur in central Paris. She would like to compile an online knowledge base describing animal species for her students to access. She knows that a rather generic ontology describing the animal species domain, O_1, is already available (which, technically speaking, means it exists as a named graph). Someone else had also supplied data consisting of a vast amount of instances for the animals ontology as a named graph with the role of instance base, I_1. However this combined data does not provide extensive coverage of the animal kingdom as required by Ella. Therefore Ella hires a SW knowledge engineer to model another ontology that defines further species not captured in O_1, and this is stored as another named graph, O_2. Since Ella requires concepts from both ontologies, 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 real instances of the new animal species defined in O_2 is compiled in an instance base, I_2.

Ella now requires to use all the acquired and generated data to power a useful service for the students to use. Schematic data from the graph O, and the instances from I_1 and I_2 are all imported to a new graph, KB, acting as a knowledge base. Ella would like the students to be able to query the knowledge base with questions like 'Are flatworms Deuterostomes or Platyzoa?'. Although by traversing the animals hierarchy it is clear that they are Platyzoa, the statement is not innately part of the graph KB. This can be discovered by realizing the semantics of `rdfs:subClassOf` as defined in the RDFS semantics. However KB might be required as is, with no assumed semantics, for other purposes. Directly enriching KB with entailment triples permanently would make this impossible.

Therefore the knowledge engineer creates a view over KB for Ella, consisting of the required extended graph, without modifying the original KB in any way. This is done by defining a view specification that computes the procedural semantics for KB. The specification uses a rule language of choice that provides a number of rules, one of which computes the transitive closure of `rdfs:subClassOf` for a set of RDF triples. Executing that rule over the triples in KB results in the

semantic view $V_1(KB)$, which consists of 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 and through simple queries over $V_1(KB)$, students can instantly get answers to their questions.

Ella later on decides to provide another service for younger students by using 'Graph Taxonomy Extractor', a graph visualization API that generates an interactive graph depicting the animal hierarchy within $V_1(KB)$. However this graph contains other information in addition to that required (*e. g.*, properties attributed to classes). Of course, Ella does not want to discard all this useful information from $V_1(KB)$ permanently just to generate the visualization. The knowledge engineer is aware of a Semantic Web application that does exactly what Ella requires. The application acts as an external view specification and generates a view, consisting of only triples defining the class hierarchy, over an input named graph. The view generated by this application, $V_2(V_1(KB))$, is fed to the API to effectively generate the interactive graph for the students to explore.

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. However, the content of KB needs to be validated, or generated, each time it is used since one of its subgraphs (O_1, O_2, I_1 and I_2) can change. Although from a practical point of view this might sound laborious, from a conceptual point of view it solves problems regarding data consistency and avoids other problems like working with outdated data that can't be updated because links to underlying models have been lost.

Fig. 4 presents the "dataflow" in our example scenario, demonstrating how the theoretical basis of NRL can be applied in practice to effectively model data for use in different scenarios in a clear and consistent way.

We now model the dataflow in Fig. 4 in TriG syntax.[6] TriG is a straightforward extension of Turtle.[7] Turtle itself is an extension of N-Triples[8] which carefully takes the most useful and appropriate things added from Notation3[9] while keeping it in the RDF model. TriG is a plain text format created for serializing NGs and RDF Datasets. Fig. 5 demonstrates how one can make use of the named graph paradigm and the syntax for named graphs:

- `[1]` namespace declarations
- `[2-5]` ontology graphs (`ex:o1` and `ex:o2` are defined and then imported into `ex:o`)
- `[6-8]` instance/knowledge base definitions
- `[9]` contents of ontology `ex:o2`, defining extended animal domain
- `[10]` contents of instance base `ex:i2`, defining instances of animals in (`ex:o2`

[6] http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriG/
[7] http://www.dajobe.org/2004/01/turtle/
[8] http://www.w3.org/TR/rdf-testcases/#ntriples
[9] http://www.w3.org/DesignIssues/Notation3

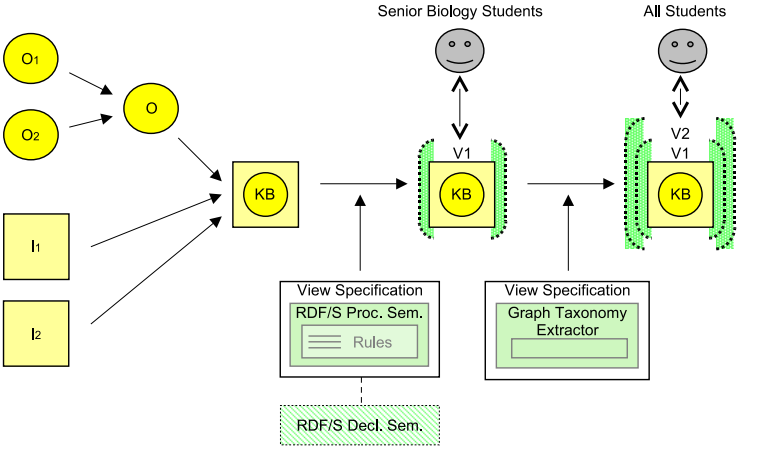

Fig. 4. NRL Dataflow Diagram

```
[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. 5. NRL Example—TriG Serialization

`[11-13]` `ex:v1kb` is defined as a view on `ex:kb` via the view specification `ex:rvs`; furthermore, `ex:v1kb` is a super graph of `ex:kb` as it realizes the RDFS semantics and thus contains the original graph plus the inferred triples; the view specification is realized (as an example) with some SPARQL-inspired `CONSTRUCT` queries (for this to work, a real rule language is required)

`[14-15]` similar to `[11-13]`, but here we define `ex:v2v1kb` with the help of an external tool, the "GraphTaxonomyExtractor"

7 Summary and Outlook

Aligning knowledge representation on a Social Semantic Desktop with the general Semantic Web approaches (RDF, RDFS, OWL, ...) promises a comprehensive use of data and schemas and an active, personalized access point to the Semantic Web [10]. In such a scenario, ontologies play an important role, from very general ontologies stating which entities can be modeled on a Semantic Desktop (*e. g.*, people, documents, ...) to rather personal vocabulary to structure information items. One of the most important design decisions is the question of the *representational ontology*, constraining the general expressivity of such a system. In this paper, we concentrated on those parts of the NEPOMUK Representational Language (NRL) which are rooted in the requirements risen by the *distributed knowledge representation and heterogeneity aspects* of the Semantic Desktop scenario 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 in the social semantic desktop scenario, a single interpretation schema cannot be assumed. Therefore, NRL aims at a *strict separation between data (sets of triples, graphs) and their interpretation/semantics*.
- 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.[10]
- 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

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

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.*

Within NEPOMUK, we are developing the approach technically, by complementing the NRL standard with tools that facilitate its use by the application programmer, as well as conceptually, by the development and integration of accompanying ontology standards, *e. g.*, an annotation vocabulary, an information element ontology, and an upper-ontology for *personal information models*.

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. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinnes, P. Patel-Schneider, and L. Stein. OWL web ontology language reference, 2004.
2. D. Beckett. RDF/XML syntax specification (revised). W3C recommendation, W3C, February 2004. http://www.w3.org/TR/rdf-syntax-grammar/.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 89, May 2001.
4. 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/.
5. 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.
6. 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.
7. P. Hayes. RDF semantics. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/rdf-mt/.
8. F. Manola and E. Miller. RDF primer. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/rdf-primer/.
9. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C working draft, W3C, 2005. http://www.w3.org/TR/rdf-sparql-query/.
10. L. Sauermann, A. Dengel, L. Elst, A. Lauer, H. Maus, and S. Schwarz. Personalization in the EPOS project. In M. Bouzid and N. Henze, editors, *Proceedings of the International Workshop on Semantic Web Personalization, Budva, Montenegro, June 12, 2006*, pages 42–52, 2006.
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. 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.