Using Semantic Web Technology to Support NLG
Case study: OWL finds RAGS

Chris Mellish
Computing Science
University of Aberdeen, Aberdeen AB24 3UE, UK
c.mellish@abdn.ac.uk

Abstract

The semantic web is a general vision for supporting knowledge-based processing across the WWW and its successors. As such, semantic web technology has potential to support the exchange and processing of complex NLG data. This paper discusses one particular approach to data sharing and exchange that was developed for NLG – the RAGS framework. This was developed independently of the semantic web. RAGS was relatively complex and involved a number of idiosyncratic features. However, we present a rational reconstruction of RAGS in terms of semantic web concepts, which yields a relatively simple approach that can exploit semantic web technology directly. Given that RAGS was motivated by the concerns of the NLG community, it is perhaps remarkable that its aspirations seem to fit so well with semantic web technology.

1 Introduction

The semantic web is a vision of a future world wide web where content, rather than being primarily in the form of unanalysed natural language, is *machine accessible* (Antoniou and van Harmelen, 2004). This could bring a number of advantages compared to the present web, in terms, for instance of the precision of web search mechanisms and the extent to which web resources can be brought together automatically for solving complex processing problems.

From the point of view of NLG, the semantic web offers a vision of a situation where resources can be formally described and composed, and where it is possible to live with the variety of different approaches and views of the world which characterise the users of the web. Given the heterogeneous nature of NLG, it seems worth considering whether there might be some useful ideas here for NLG.

The foundation of the semantic web is the idea of replacing formatting-oriented languages such as HTML by varieties of XML which can capture the structure of content explicitly. Markup of linguistic resources (text corpora, transcribed dialogues, etc.) via XML is now standard in NLP, but very often each use of XML is unique and hard to reconcile with any other use. The semantic web goes beyond this in proposing a more abstract basic language and allowing explicit representation of what things in it *mean*. For the semantic web, RDF (Klyne and Carroll, 2003), which is built on top of XML, represents a common language for expressing content as a "semantic network" of triples, and ontology languages, such as OWL (McGuinness and van Harmelen, 2004), allow the expression of constraints and principles which partially constrain possible interpretations of the symbols used in the RDF. These ontologies are statements that themselves can be inspected and modified. They can provide the basis for different people to express their assumptions, agreements and disagreements, and to synthesise complex data from multiple sources.

2 RAGS

RAGS ("Reference Architecture for Generation Systems") was an attempt to exploit previous ideas about common features between NLG systems in order to propose a reference architecture that would help researchers to share, modularise and evaluate NLG systems and their components. In practice, the project found that there was less agreement than expected among NLG researchers on the modules of an NLG system or the order of their running. On the other hand, there *was* reasonable agreement (at an abstract level) about the kinds of *data* that an NLG system needs to repre-

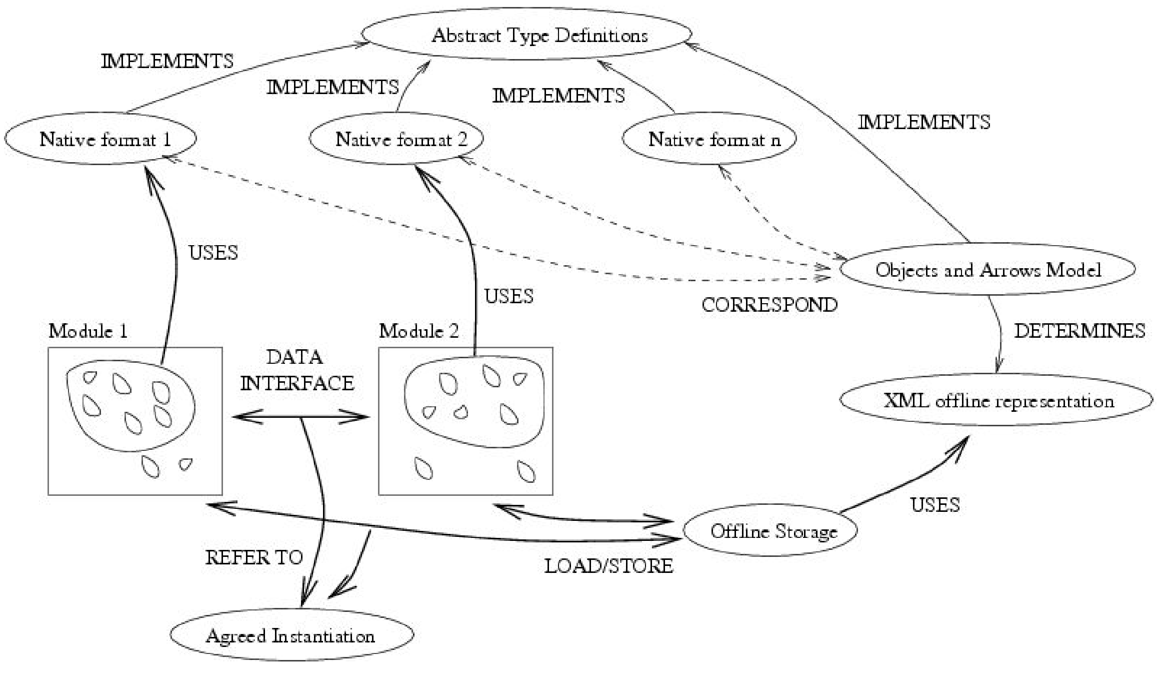

Figure 1: RAGS

sent, in passing from some original non-linguistic input to a fully-formed linguistic description as its output. Figure 1 summarises RAGS and how it was intended to be used. The following description simplifies/ rationalises in a number of ways; more information about RAGS can be found in (Mellish et al., 2006) and (Cahill et al., 2001).

RAGS provides *abstract type definitions* for 6 different types of data representations: conceptual, rhetorical, document, semantic, syntactic and "quote". As an example, here are the definitions associated with document representations (which describe the parts of a document and aspects of their logical formatting).

$$\begin{aligned} \mathit{DocRep} &= \mathit{DocAttr} \times \mathit{DocRepSeq} \\ \mathit{DocRepSeq} &= \mathit{DocRep}^* \\ \mathit{DocAttr} &= (\mathit{DocFeat} \rightarrow \mathit{DocAtom}) \\ \mathit{DocFeat}, \mathit{DocAtom} &\in \mathit{Primitives} \end{aligned}$$

These type definitions express the types in terms of set theory (using constructions such as union, Cartesian product, subset and function), where the "primitive" types correspond to basic sets that have to be defined in a theory-specific way. Thus a document representation (*DocRep*) has two components, a *DocAttr* and a *DocRepSeq*. A *DocRepSeq* is a sequence of zero or more *DocRep*s, which represent the document structure of the parts of the document. A *DocAttr* is a function from $DocFeat$s to $DocAtom$s. The former can be thought of as a set of names of "features" for parts of documents (e.g. text level, indentation) and the latter as a set of values for these (e.g. "Clause", 3). However the sets *DocFeat* and *DocAtom* are left unspecified in the RAGS formalisation. The idea is that researchers will not necessarily agree how to instantiate these primitives. Clusters of researchers may agree on standard possibilities for these sets and this will help them to share data (but even researchers not able to agree on the primitive sets will be able to understand one anothers' data to some extent). When two NLG modules need to exchange data, they need to refer to an agreed instantiation of the primitive types in order to share fully.

Although it gives some examples, RAGS does not specify any particular formats in which data should be represented in different programming languages and used by NLG modules – potentially, arbitrary "native formats" could be used, as long as they can be viewed as "implementations" of the abstract type definitions. Further conditions are, however, imposed by requiring a correspondance between native formats and representations in a "reference implementation" called the *Objects and Arrows* (OA) model. This provides answers to further questions, such as what partially-specified data representations are possible, where re-entrancy can occur and how data representations of different types can be mixed. The OA model represents data as directed graphs, whose nodes represent typed pieces of data and whose edges represent relations. The possible legal states of an OA representation are formally defined, in a way that resembles the way that information states in a unification grammar can be characterised (Shieber, 1986). Each node in the graph is labelled with a type, e.g. *DocRep*, *DocAtom*. Each node is assumed to have a unique identifier and for primitive types a node can also have a *subtype*, a theory-dependent elaboration that applies to this particular data object (e.g. a *DocAtom* could have the subtype 3). Some edges in the graph indicate "local arrows", which describe the parts of complex datastructures. For instance, edges labelled *el* indicate elements of unordered structures, and arrows labelled *el*-1, *el*-2 etc. indicate components of ordered structures. Edges can also represent "non-local arrows" which describe relationships between representations at different levels. Non-local arrows allow data representations at different levels to be mixed into a single graph.

Representations in the Objects and Arrows model can be mapped to an XML interchange representation. The correspondance between native formats and the OA model can then be used to map between native data representations and XML (in both directions). Modules can communicate via agreed native formats or, if this is undesirable, via the XML representation.

3 Some Problems with RAGS

Some of the problems with RAGS, which have impeded its uptake, include:

- Complexity and lack of tools – RAGS was a proposal with a unique shape and takes some time to understand fully. It ploughs its own distinctive furrow. Because it was developed in a project with limited resources, there are limited tools provided for, for instance, displaying RAGS structures, supporting different programming languages and styles and automatic consistency checking. This means that engaging with RAGS involves initially a significant amount of low-level programming, with benefits only to be seen at some time in the future.

- Idiosyncratic use of XML – RAGS had to address the problem of expressing a graph in a serialised form, where there can be multiple, but different, serialisations of the same graph. It did this in its own way, which means that it is hard to exploit general tools which address this problem in other areas.

- Inclarity about how to "buy-in" to limited degrees - there is no defined mechanism for dividing generally agreed from non-agreed elements of a RAGS representation or for expressing or referring to an "agreed instantiation".

4 Recasting RAGS data in terms of RDF

The first step in recasting RAGS in semantic web terms is to exploit the fact that it is the OA model (rather than the abstract type definitions) that is the basis of data communication, since this model expresses more concrete requirements on the form of the data. Therefore initially we concentrate on the OA model and its XML serialisation.

RDF is a data model that fits OA graphs very well. It provides a way of creating "semantic networks" with sets of object-attribute-value triples. Objects and attributes are "resources", which are associated with Universal Resource Identifiers (URIs), and values are either resources or basic data items ("literals", e.g. strings or integers). Resources have types, indicated by the RDF `type` attribute. The idea of an RDF resource maps nicely to a RAGS object, and the idea of an RDF attribute maps nicely to a RAGS arrow.

URIs provide a natural way to allow reentrancy to be represented and at the same time permit unambiguous references to external objects in the way that RAGS intended should be possible. The XML namespace mechanism allows complex IDs to be abbreviated by names of the form `Prefix:N`, where `Prefix` is an abbreviation for the place where the name `N` is defined and `N` is the basic name (sometimes the prefix can be inferred from context and can be missed out). Thus, for instance, if the prefix `rags` is defined to stand for the start of a URI identifying RAGS then `rags:DocRep` identifies the type `DocRep` defined by RAGS, as distinct from any other definition anyone might have.

It follows from the preceding discussion that instances of the RAGS abstract types can be mapped naturally to RDF resources with the abstract type as the value for the RDF attribute `type`. Arrows can be mapped into RDF attributes, and so it really only remains to have a convention for the representation of "subtype" information in RAGS. In this paper, we will assume that instances of primitive types can have a value for the attribute `sub`.

RDF can be serialised in XML in a number of ways (which in fact are closely related to the possible XML serialisations of RAGS).

To summarise, using RDF rather than RAGS XML introduces negligable extra complexity but has a number of advantages:

- Because it is a standard use of XML, it means that generic software tools can be used with it. Existing tools, for instance, support reading and writing RDF from different programming languages, visualising RDF structures (see Figure 4) and consistency checking.

- Because it comes with a universal way of naming concepts, it means that it is possible for different RAGS resources to be unambiguous and reference one another.

5 Formalising the RAGS types using ontologies

RDF gives us a more standard way to interpret the OA model and to serialise OA instance information in XML. However, on its own it does not enforce data representations to be consistent with the intent of the abstract type definitions. For instance, it does not prevent an element of a $DocRepSeq$ being something other than a $DocRep$.

For RAGS, an XML DTD provided constraints on what could appear in the XML serialisation, but DTDs are not very expressive and the RAGS DTD had to be quite loose in order to allow partial representations. The modern way to define the terms that appear in a use of RDF, and what constraints there are on their use, is to define an *ontology* using a language like RDFS (Brickley and Guha, 2003) or OWL (McGuinness and van Harmelen, 2004). An ontology can be thought of as a set of logical axioms that limits possible interpretations of the terms. This could be used to show, for instance, that a given set of instance data is inconsistent with an ontology, or that further logical consequences follow from it. There are various versions of the web ontology language OWL. In this paper, we use OWL DL, which is based on a description logic, and we will use standard description logic notation in this paper.

Description logics allow one to make statements about the terms (names of concepts and roles) used in some knowledge representation. In our case, a concept corresponds to a RAGS type (implemented by an RDF resource linked to from individuals by a `type` attribute) and a role corresponds to a RAGS arrow (implemented by an RDF attribute). Complex names of concepts can be built from simple names using particular constructors. For instance, if α and β are two concept names (simple concept names or more complex expressions made from them) and ρ is a role name, then, the following are also concept names:

$\alpha \sqcup \beta$ - names the concept of everything which is α or β

$\alpha \sqcap \beta$ - names the concept of everything which is α and β

$\exists \rho.\alpha$ - names the concept of everything which has a value for ρ which is an instance of concept α

$\forall \rho.\alpha$ - names the concept of everything which only has values for ρ which are instances of concept α

$=_n \rho$ - names the concept of everything with exactly n different values of ρ

Constructors can be nested, so that, for instance, $C_1 \sqcap \exists r.C_2$ is a possible concept name, assuming that C_1 and C_2 are simple concept names and r is a role name.

For an ontology, one then writes logical axioms stating relationships between simple or complex concept names, e.g.

$\alpha_1 \sqsubseteq \alpha_2$ - states that α_1 names a more specific concept than α_2

$\alpha_1 \equiv \alpha_2$ - states that α_1 names the same concept as α_2

$disjoint(\{\alpha_1, \ldots \alpha_n\})$ - states that $\alpha_1, \ldots \alpha_n$ are disjoint concepts (no pair can have a common instance).

$\rho_1 \sqsubseteq_r \rho_2$ - states that ρ_1 names a subproperty of ρ_2

$domain(\rho, \alpha)$ - states that ρ can only apply to things satisfying concept α

$range(\rho, \alpha)$ - states that values of ρ must satisfy concept α

$functional(\rho)$ - states that ρ is a functional role

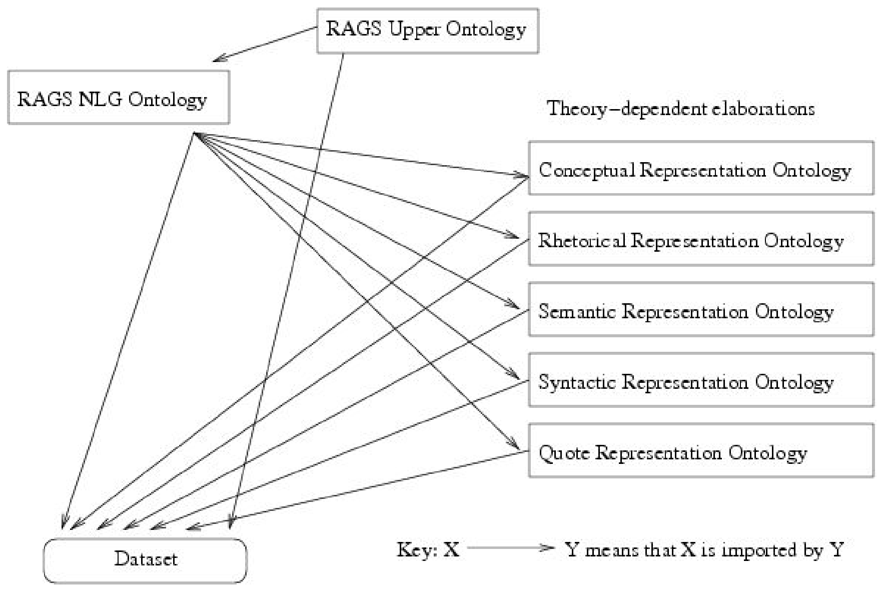
RAGS Upper Ontology
RAGS NLG Ontology
Theory–dependent elaborations
Conceptual Representation Ontology
Rhetorical Representation Ontology
Semantic Representation Ontology
Syntactic Representation Ontology
Quote Representation Ontology
Dataset
Key: X ——> Y means that X is imported by Y

Figure 2: Using Multiple Ontologies in RAGS

For more information on the formal basis of description logics and their relevance for ontologies, see (Horrocks, 2005).

For RAGS, a number of advantages follow from adopting DLs as the basis for formalising data representations:

Modularity. A given set of instance data may relate to more than one ontology which expresses constraints on it. One ontology is said to *import* another if it inherits the constraints that the other provides. The standard (monotonic) logic approach applies, in that one can choose to describe the world in terms of any consistent set of axioms. Ontologies package up sets of axioms into bundles that one might decide to include or not include in one's own model of the world. Ontologies for different purposes can be built by different people but used together in an eclectic way. This formalises the idea of "variable buy-in" in RAGS.

Openness. Also corresponding to the usual approach with logic, the semantics of OWL makes no *closed world assumption*. Thus a statement cannot be inconsistent purely by *failing* to specify something. This means that it is only necessary to describe the properties of *complete* datastructures in an ontology. Partial descriptions of data will be not be inconsistent by virtue of their partiality. Only having to describe complete datastructures makes the specification job much simpler. In a similar way, the semantics of OWL makes no *unique names assumption*. Thus individuals with different names are not necessarily distinct. This means that it is generally possible to make a given description more specific by specifying the identity of two individuals (unless inconsistency arises through, for instance, the individuals having incompatible types). This is another requirement if one wishes the power to add further information to partial representations.

Software tools. As with RDF, use of OWL DL opens up the possibility of exploiting generic tools developed elsewhere, for instance reasoners and facilities to translate RAGS concepts into programming language structures.

6 The RAGS Ontologies

It is convenient to modularise what RAGS requires as well-formedness constraints as a *set of* ontologies. This allows us to formalise what it means to "buy-in" to one or more parts of RAGS. It simply means importing one or more of the RAGS ontologies (in addition to one's own) and making use of some of the terms defined in them. We now outline one possible version of the core RAGS ontologies.

Figure 2 shows the way that the RAGS ontologies are intended to be used. A dataset in general makes use of concepts defined in the core RAGS ontologies (the "upper ontology" and the "NLG

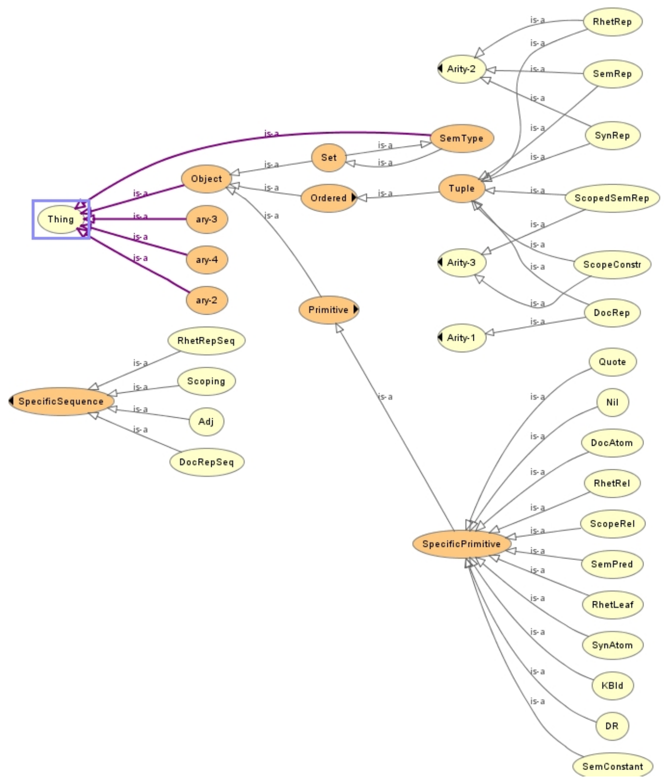

Figure 3: The RAGS "NLG ontology"

ontology")[1] and also theory-dependent elaborations defined in separate ontologies (which may correspond one-to-one to the different levels, as shown, but need not do so necessarily). These elaborations are not (initially) provided by RAGS but may arise from arbitrary research subcommunities. Logically, the dataset is simple described/constrained by the union of the axioms coming from the ontologies it makes use of. In general, different datasets will make consistent references to the concepts in the core RAGS ontologies, but they may make use of different theory-dependent elaborations.

The basis of RAGS is a very neutral theory about datatypes (and how they can be encoded in XML). This is in fact independent of the fact that RAGS is intended for NLG - at this level, RAGS could be used to describe data in other domains, or NLG-oriented data that is not covered by RAGS. It therefore makes sense to think of this as a separable part of the theory, the "upper ontology". At the top level, datastructures (instances of `Object`) belong to one of the concepts `Ordered`, `Set` and `Primitive`. Ordered structures are divided up in terms of the number of components (concepts `Arity-1`, `Arity-2` etc) and whether they are `Tuples` or `Sequences`. For convenience, union types such as `Arity-atleast-2` are also defined.

The RAGS NLG ontology (see Figure 3 for an overview) contains the main substance of the RAGS type definitions. As the figure shows, it introduces a number of new concepts as subconcepts of the upper ontology concepts. For instance, `DocRepSeq`, `RhetRepSeq`, `Adj` and `Scoping` are introduced as subconcepts of `SpecificSequence` (these concepts correspond to types used in RAGS at the document, rhetorical, syntactic and semantic levels). Not shown in the diagram is the type of roles, `Functional` that includes all arguments of RAGS functional objects[2]. The set of type definitions describing a level of representation in RAGS translates quite directly into a set of axioms in this ontology. For instance, the following is the encoding of the type definitions for document rep-

[1]These are both available in full from `http://www.abdn.ac.uk/~csc248/ontologies/`

[2]Whereas in RAGS a functional type (e.g. *DocAttr*) is represented as an unordered set of (ordered) pairs of the form <function argument,function value>, here we can simply implement the function arguments as RDF attributes and omit the functional types.

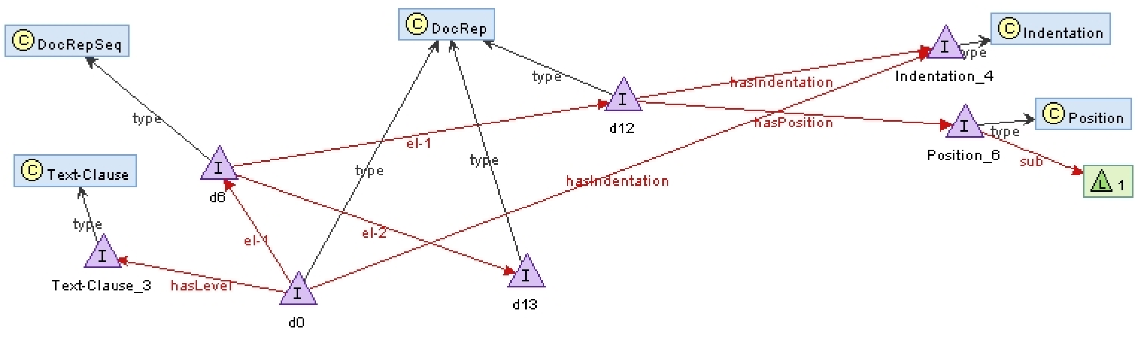

Figure 4: Visualisation of example Document Representation

resentations. First of all, it is necessary to specify that a $DocRep$ is a tuple with arity 1 (the $DocAttr$ is not needed), and that its component must have a specific type:

$$DocRep \sqsubseteq Tuple \sqcap Arity{-}1$$
$$DocRep \sqsubseteq (\forall el{-}1.DocRepSeq)$$

The next few axioms do a similar job for $DocRepSeq$, a kind of sequence:

$$DocRepSeq \sqsubseteq SpecificSequence$$
$$DocRepSeq \sqsubseteq (\forall n{-}el.DocRep)$$

Finally, a high level role $DocFeat$ is introduced, whose subroles will correspond to particular document features like $Indentation$. The domain and range of such roles are constrained via constraints on $DocFeat$:

$$DocFeat \sqsubseteq_r Functional$$
$$domain(DocFeat, DocRep)$$
$$range(DocFeat, DocAtom)$$

7 Other Ontologies and RAGS

As stated above, in general one produces specialisations of the RAGS framework by creating new ontologies that:

- Introduce specialisations of the RAGS primitive concepts (and perhaps new roles that instances of these can have).
- Introduce subroles of the RAGS functional roles.
- Add new axioms that specialise existing RAGS requirements, involving the core concepts and roles and/or the newly introduced ones.

An example of this might be an example ontology that instantiates a simple theory of document structure, following (Power et al., 2003). Given the notion of document structure introduced in section 2 and formalised in section 6, it is really only necessary to specify the "features" of pieces of document structure ($DocFeat$) and their "values" ($DocAtom$). The former are modelled as roles and the latter in terms of concepts. First we introduce the basic types of values:

$$DocAtom \equiv (Position \sqcup Indentation \sqcup Level \sqcup Connective)$$
$$disjoint(\{(Position, Indentation, Level, Connective\})$$

Positions in the text could be modelled by objects whose sub values are positive integers (there is a standard (RDFS) datatype for these). The following axioms capture this and the characteristics of the role $hasPosition$:

$$Position \sqsubseteq (\forall sub.xsd : positiveInteger)$$
$$hasPosition \sqsubseteq_r DocFeat$$
$$range(hasPosition, Position)$$
$$functional(hasPosition)$$

For text levels, on the other hand, there is a fixed set of possible values. These are introduced as disjoint concepts. In addition, the role $hasLevel$ is introduced:

$$Level \equiv (Chapter \sqcup Paragraph \sqcup Section \sqcup Text{-}Clause \sqcup Text{-}Phrase \sqcup Text{-}Sentence)$$
$$disjoint(\{Chapter, Paragraph, Section, Text{-}Clause, Text{-}Phrase, Text{-}Sentence\})$$
$$hasLevel \sqsubseteq_r DocFeat$$
$$range(hasLevel, Level)$$
$$functional(hasLevel)$$

Figure 4 shows an example $DocRep$ (labelled "d12") described by this ontology, as visualised by the RDF-Gravity tool developed by Salzburg Research. It consists of a $DocRepSeq$ ("d6") with two $DocRep$ components ("d0" and "d13"). The indentations of "d12" and "d0" are not known, but they are constrained to be the same.

It is easy to think of examples of other (existing or potential) ontologies that could provide theories of the RAGS primitive types. For instance, WordNet (Miller, 1995) or the Generalised Upper Model (Bateman et al., 1995) could be used to bring in a theory of semantic predicates ($SemPred$). An ontology of rhetorical relations ($RhetRel$) could be built based on RST, and so on.

Ontologies can use the expressive power of OWL to make relatively complex statements. For instance, the following could be used in an RST ontology to capture the concept of nucleus-satellite relations and the constraint that a rhetorical representation with such a relation (as its first component) has exactly two subspans (recorded in the second component):

$$NS \sqsubseteq RhetRel$$
$$(RhetRep \sqcap \exists el\text{-}1.NS) \sqsubseteq (\exists el\text{-}2.Arity\text{-}2)$$

8 Relation to Other Work

Reworking RAGS to use semantic web technology relates to two main strands of previous work: work on XML-based markup of linguistic resources and work on linguistic ontologies.

The trouble with applying existing annotation methods (e.g. the Text Encoding Initiative) to NLG is that they presuppose the existence of a linear text to start with, whereas in NLG one is forced to represent more abstract structures before coming up with the actual text. A recent proposal from Linguistics for a linguistic ontology for the semantic web (Farrar and Langendoen, 2003) is again based around making annotations to existing text. Research is only just beginning to escape from a "time-based" mode of annotation, for instance by using "stand-off" annotations to indicate layout (Bateman et al., 2002). In addition, most annotation schemes are partial (only describe certain aspects of the text) and non-structured (assign simple labels to portions of text). For NLG, one needs a way of representing *all* the information that is needed for generating a text, and this usually has complex internal structure.

Linguistic ontologies are ontologies developed to describe linguistic concepts. Although ontologies are used in a number of NLP projects (e.g. (Estival et al., 2004)), the ontologies used are usually ontologies of the application domain rather than the linguistic structures of natural languages. The development of ontologies to describe aspects of natural languages is comparatively rare. The WordNet ontologies are a widely used resource describing the repertoire of word senses of natural languages, but these concentrate on individual words rather than larger linguistic structures. More relevant to NLG is work on various versions of the Generalised Upper Model (Bateman et al., 1995), which outlines aspects of meaning relevant to making NLG decisions. This has been used to help translate domain knowledge in a number of NLG systems (Aguado et al., 1998).

In summary, existing approaches to using ontologies or XML for natural language related purposes are not adequate to describe the datastructures needed for NLG. Semantic web technology applied to specifications with the complexity of those generated by RAGS might, however, be able to fill this gap.

9 The Semantic Web for NLG tasks

In the above, we have made a case for the use of semantic web technology to aid inter-operability and sharing of resources for NLG. This was justified largely by the fact that the most significant NLG "standardisation" effort so far, RAGS, can be straightfowardly recast in semantic web terms, bringing distinct advantages. Even if RAGS itself is not taken forward in its current form, this suggests that further developments of the idea could bear fruit in semantic web terms.

The semantic web is certainly not a panacea for all the problems of NLG, and indeed there are aspects of the technology that are still at an early stage of development. For instance, the problems of matching/ reconciling alternative ontologies are many and complex. Some researchers even dispute the viability of the general approach. On the other hand, the semantic web community is concerned with a number of problems that are also very relevant to NLG. Fundamentally, the semantic web is about sharing and exploiting distributed computational resources in an open community where many different goals, viewpoints and theories are represented. This is something that NLG also seeks to do in a number of ways. The semantic web movement has considerable momentum. There are more of them than us. Let's see what we can get from it.

Acknowledgments

This work was supported by EPSRC grant EP/E011764/1.

References

G. Aguado, A. Ba nón, John A. Bateman, S. Bernardos, M. Fernández, A. Gómez-Pérez, E. Nieto, A. Olalla, R. Plaza, and A. Sánchez. 1998. Ontogeneration: Reusing domain and linguistic ontologies for Spanish text generation. In *Proceedings of the ECAI'98 Workshop on Applications of Ontologies and Problem Solving Methods*, pages 1–10, Brighton, UK.

Grigoris Antoniou and Frank van Harmelen. 2004. *A Semantic Web Primer*. MIT Press.

John A. Bateman, Renate Henschel, and Fabio Rinaldi. 1995. Generalized Upper Model 2.0: documentation. Technical report, GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt, Germany.

John Bateman, Renate Henschel, and Judy Delin. 2002. A brief introduction to the GeM annotation scheme for complex document layout. In *Proceedings of NLP-XML 2002*, Taipei.

D. Brickley and R. V. Guha. 2003. Rdf vocabulary description language 1.0: Rdf schema. Technical Report http://www.w3.org/TR/rdf-schema, World Wide Web Consortium.

Lynne Cahill, Roger Evans, Chris Mellish, Daniel Paiva, Mike Reape, and Donia Scott. 2001. The RAGS Reference Manual . Available at `http://www.itri.brighton.ac.uk/projects/rags`.

Dominique Estival, Chris Nowak, and Andrew Zschorn. 2004. Towards ontology-based natural language processing. In *Proceedings of NLP-XML 2004*, Barcelona.

Scott Farrar and Terry Langendoen. 2003. A linguistic ontology for the semantic web. *Glot International*, 7(3):1–4.

Ian Horrocks. 2005. Description logics in ontology applications. In B. Beckert, editor, *Proc. of the 9th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2005)*, pages 2–13. Springer Verlag LNCS 3702.

Baden Hughes and Steven Bird. 2003. Grid-enabling natural language engineering by stealth. In *Proceedings of the HLT-NAACL 2003 Workshop on The Software Engineering and Architecture of Language Technology Systems*.

G. Klyne and J. Carroll. 2003. Resource description framework (rdf): Concepts and abstract syntax. Technical Report http://www.w3.org/TR/rdf-concepts, World Wide Web Consortium.

D. L. McGuinness and F. van Harmelen. 2004. Owl web ontology language overview. `http://www.w3.org/TR/owl-features/`.

Chris Mellish, Donia Scott, Lynne Cahill, Daniel Paiva, Roger Evans, and Mike Reape. 2006. A reference architecture for generation systems. *Natural language engineering*, 1:1–34.

G. Miller. 1995. Wordnet: A lexical database for english. *CACM*, 38(11):39–41.

Richard Power, Donia Scott, and Nadjet Bouayad-Agha. 2003. Document structure. *Computational Linguistics*, 29:211–260.

Stuart M. Shieber. 1986. *An introduction to unification-based approaches to grammar*. CSLI.