Toward textual encoding based on RDF

G.Tummarello[1] , C.Morbidoni[1], E. Pierazzo[2]

[1]DEIT, Università Politecnica delle Marche
Via Brecce Bianche, 60100 Ancona, Italy
giovanni@wup.it, christian@deit.univpm.it

[2]Dipartimento di Studi italianistici, Università di Pisa
Via del Collegio Ricci, 10, 56126 Pisa, Italy
pierazzo@ital.unipi.it

Abstract

In this paper we investigate the use of Semantic Web languages such as RDF and OWL in textual encoding and we discuss several advantages these tools could provide. Among these, we show how the overlapping markup encoding problem can be naturally solved as the graph structure of the RDF language is naturally suited for this purpose. Further advantages in using Semantic Web based technologies lie in the powerful query and reasoning facilities that are already available and successfully used in the Semantic Web. The distributed and resource centric nature of RDF also enables a novel cooperative annotation scenario, where different encoding "facets" of the same text can be naturally merged. We then suggest the basic structure for an encoding ontology and evaluate the simplicity by which a complex query can be solved either programmatically or using existing Semantic Web query languages.

1 Introduction

As practical and scientific interest into machine aided literature analysis and processing raises, pure XML textual encoding formats prove to be limiting under several point of view.

A number of successive enhancement proposals has shown a clear trend: given the strictness of the XML model, for advanced encoding, XML compliance must often be abandoned and in any case standard XML processing tools (e.g. Query languages) prove inadequate.

In this paper we investigate the feasibility of using the new methodology based on Semantic Web (SW) markup languages RDF and OWL, with special interest on how to solve the “overlapping markup” problem of classic XML markup.

We first observe that, at least in theory, RDF is suitable to fulfil all the tasks that have been traditionally done in XML (see the 2 way mappings that have proposed in [1] or in similar works).

The evaluation is then not in terms of what aspect can or cannot be encoded (as they all basically can) but rather how easy it is to cover the useful use cases, given the standard tools already available for the specific markup languages.

Also, to facilitate large and multidisciplinary use of machine encoded material, it is important to evaluate to which extent the tools that we're using are in fact meant to serve that purpose, e.g. backed up by a solid and well understood semantic, rather than serve as a pure placeholder which means nothing without a complex, idiosyncratic set of explanations and rules. With respect to this, the use of SW tools (RDF/S, OWL) is simply the next logical step.

After debating the possible advantages of using Semantic Web technologies, we propose an encoding methodology which is currently being implemented in the RDF Textual Encoding Framework (RDFTef), a demonstrative open source API allowing markup based on the above principles.

2 The tools of the Semantic Web: an overview

The term “Semantic Web” (SW) refers to a vision of machine readable metadata annotations that can be retrieved over the Internet much like HTML is retrieved today. Currently, by SW some indicate the tools that the semantic web community has created or is using to enable such vision.

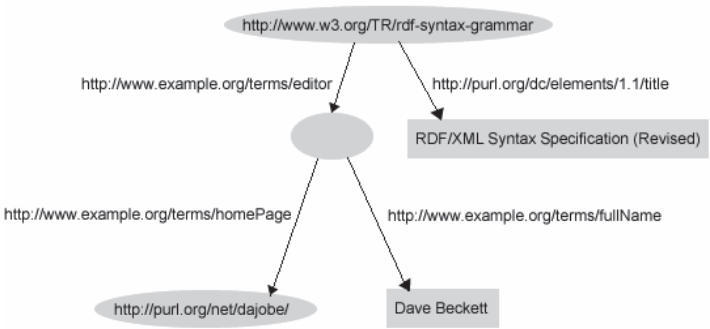
2.1 URIs – Uniform Resource Identifiers

The term “resource” refers to anything that can be somehow “identified”, that is, given a proper name. In particular, the identifiers used in both the SW and the MPEG-7 initiatives are known as Uniform Resource Identifiers (URI) and classically divided in URL (those that had a network locating mechanism, e.g. HTTP) and

URN for those that didn't. URIs are divided in “schemes” (e.g. Http:, ftp:) and form “namespaces” (e.g. the set of URI of URN in the form "urn:isbn:n-nn-nnnnnn-n").

2.2 RDF

RDF forms the basis of the SW. This language defines a method to connect resources (generally identified by URI’s) and literal (data values) creating semantic networks. RDF's main strength is simplicity: it is a network of nodes connected by directed and labelled arcs. Arcs are used to express properties of resources. The following illustration depicts an example annotation (or model or graph; these terms are used here equivalently) where the address of a staff member is expressed in RDF:



In this graph (taken from [2]) we see how resources (elliptic nodes stating a URI) are connected to Literal nodes (rectangular ones), and possibly via “blank” nodes. Blank nodes do not have URI and therefore cannot be addressed from outside the current model. They are used to glue together statements into a logical group (in this case “the editor”). At basic level, RDF models are uniquely defined by the set of their triples (no duplicated triples allowed). Triples connect *subjects* (URI or a Blank Node) via *predicates* (URI) to *objects* (URI, Blank Nodes or Literal). Rigorous semantics is furthermore provided to express a variety of useful structures such as hierarchies of classes and utility structures like “bags” or “sequences”.

2.3 The SW ontology tools: RDFS/OW

RDFS and OWL are tools that allow a formal definition of the concepts used in the specific domain of interest, something referred to as Ontology. RDFS provides the basic description tools, allowing the creation of “light weight ontologies”. These define the fundamental classes of objects in the domain, their hierarchical relationship and a listing of the properties they might (along with some restrictions on these). Building on and extending RDFS, OWL gives much richer descriptions of the domain ontologies by providing tools such as cardinality constraints on properties, (e.g., that a Person has exactly one biological father), property transitivity, unique identifier (or key) for instances of a particular class etc. OWL has just recently reached a “golden” status by becoming a W3C recommendation [3].

2.4 Higher level initiatives: RuleML and SPARQL

“Rules” to be applied on models are needed for applications to display automated behaviours and for useful graph transformations. RuleML is an undergoing effort for standardizing a language to describe rules that operate on RDF/S/OWL datasets. Started in 2000, it has seen an interest shift from XML to RDF, and it's working its way to a 1.0 specification including concepts from a number of different logic paradigms.
SPARQL[13], on the other hand, is a standardization effort by the W3C RDF Data Access Working Group (DAWG) for a RDF query language. A number of RDF store repositories are in fact available and all mostly implement different RDF query languages; thus the need of standardization. Furthermore, SPARQL is set to support the most interesting aspects from the existing languages

3 Semantic Web Textual Encoding: resource centric (RDF) vs. text metadata (XML)

With the above overview in mind it is clear that RDF (although often seen in its XML serialization) is something fundamentally different. Rather than just providing a (forcefully) hierarchical structure of metadata to a text (XML), RDF deals with generic description of the relationships between resources. Then it is clear how, once a text is somehow available as a resource (e.g. single words or chars), adding “metadata” to it is then simply a sub-case of the overall possibilities.

3.1 Existing frameworks supporting advanced (overlapping) markup

Due to the XML specifications that require a strict nesting of the elements, classic textual encoding has been suffering from the "Markup overlap" problem.
Almost all texts in the Humanities potentially imply overlapping hierarchies. To consider only very base aspects, it is possible to detect at least two main hierarchies that overlaps quite always: the book structure (a sequence of pages or leaves) and the text structure (a sequence of parts, chapters, paragraphs). The TEI Guidelines [4] provide a number of examples of possible multiple hierarchies:

- in narrative, a speech by a character may begin in the middle of a paragraph and continue for several more paragraphs
- in a verse text, the encoder may need to tag both the formal structure of the verse (its stanzas and lines) and its syntactic structures (which sometimes nest within the metrical structure and sometimes cross metrical boundaries)
- in any kind of text, the encoder may wish to record the physical structure of volume, page, column, and line, as well as the formal or logical structure of chapters and paragraphs or acts and scenes, etc.
- in verse drama, the structure of acts, scenes, and speeches often conflicts with the metrical structure
- in any kind of text, an embedded text (e.g. a play within a play, or a song) may be interrupted by other matter; the encoder may wish to establish explicitly the logical unity of the embedded material (e.g. to identify the song as a single song, and to mark its internal formal structure)
- in a dictionary, different types of information (e.g. orthography, syllabification, and hyphenation) may be combined within a single notation; the encoder may wish both to preserve the presentation of the material in the source text and to disentangle the logically distinct pieces of information in the interests of more convenient processing of the lexical information.

This limitation has been subject of intensive study in the community, and a number of proposals have been made [5][6][7] (for a comprehensive overview see [8]). With time, the use of pure XML has been set apart in favour of richer "internal" models, which then are filtered with pre or post processors to provide different XML compliant encoded files [6] or new markup languages all together [9]. Almost in all the proposed schema, self overlapping, schema validation and XML processing by standard tools is somehow limited or not possible [8].
One of the most interesting frameworks for advanced textual encoding is certainly [10]. The proposed solution is composed by a set of user editors, an ad hoc data structure (the GODDAG) and an ad hoc query language, EXPath, which extends the XML query language "XPath" with support for querying multiple hierarchical structures. In order to achieve output and compatibility with other formats, the GODDAG internal representation can be serialized into a series of parallel XML files.
These specific solutions seems to suitable to solve the immediate overlapping markup problem but are weak under an engineering point of view: they all propose a own set of idiosyncratic rules and semantics for the encoding. This in turns requires that developers not only need to understand new specifications but also cannot use existing tools (such as XML processors or RDF toolkit and reasoners). Implementing the proposed extension to the XPath language also seems as a rather involved task, so while a JAVA version exists, it might be difficult to operate on this format with other languages.

3.2 Experiments with RDF based model for text encoding

Given the resource centric graph structure of the RDF model, once text and annotations themselves have been made "resources" and have been identified with URIs or Blank Nodes, the overlapping markup is handled naturally, with no need of added semantics and syntactic specifications.

3.2.1 Encoding text into resources

One possible way of encoding a text into RDF resources is to create a "row" RDF graph where nodes represent every "printable element" in the analyzed text document. These elements are usually words, punctuation or (if needed) lower level concepts such as characters, typographical signs, or, in case of encoding manuscripts, scratches and more. At this point, the natural ordering of the words can be encoded by connecting these row elements using their "next" property thus supporting a simple rendering procedure.
Figure 1 illustrates this.

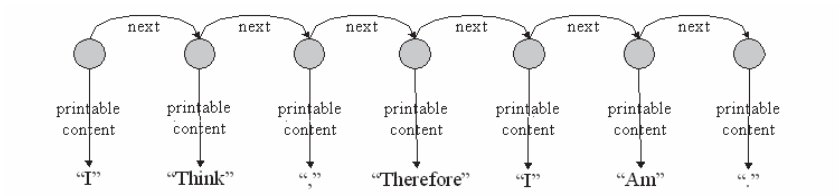


Figure 1 The row symbol chain extracted from the textual content. Every element (word or punctuation) correspond to a RDF node that has a property named "printable content" used for rendering purpose. Printable symbols are connected by a "next" property that respects the original ordering of elements in text.

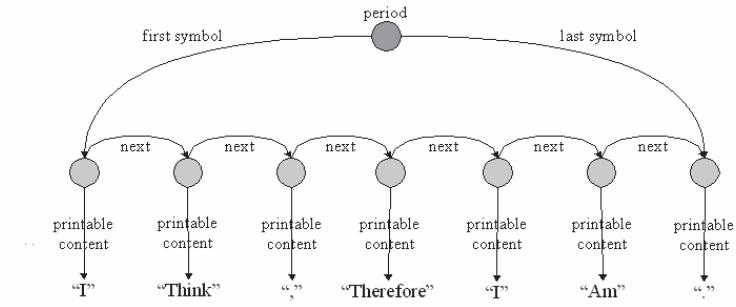


Figure 2 A period is encoded in this graph structure. The properties "first symbol" and "last symbol" point to the first and the last words of the period.

The same concept of "sequence of symbols" can then be applied also to annotation kind which have a natural ordering. This is the case, for example, of annotations such as the concept of "row", "page", "period" and so on (Figure 3).

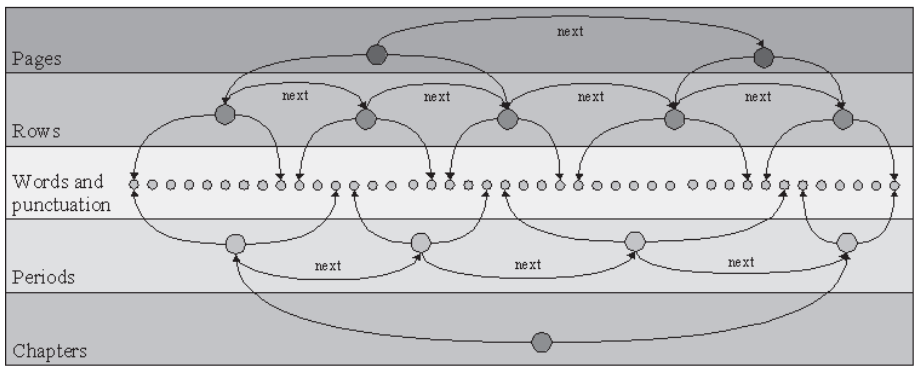


Figure 3 Different overlapping hierarchies are represented in the same graph. The blue levels represent the "rows and pages" hierarchy where rows are directly mapped on top of the row printable symbols chain and pages are mapped on top of rows in a similar way. Relaying on the same low level chain other hierarchies can be represented within the same RDF graph, as happens for periods and chapters (orange levels in the figure).

With markup of non contiguous elements, the RDF Bags and Sequences collection tools come handy [1]. RDF Bags and Sequences are in fact classes with specified syntax and semantic to group in an ordered or unordered fashion respectively. Figure 4 shows how different, non continuous words are grouped into a single annotation.

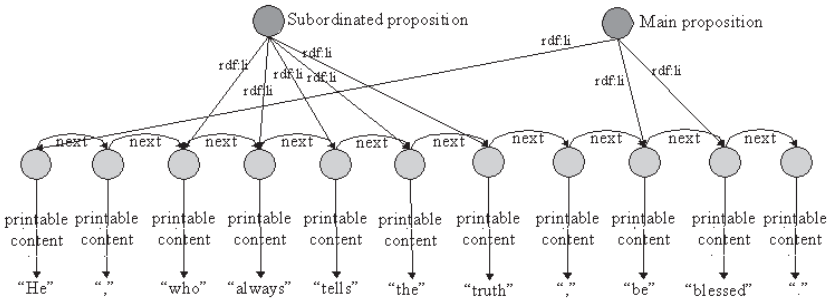


Figure 4 Proposition are non contiguous group of symbols, so they are represented with a "bag" structure, where every symbol belonging to the proposition is directly linked to it.

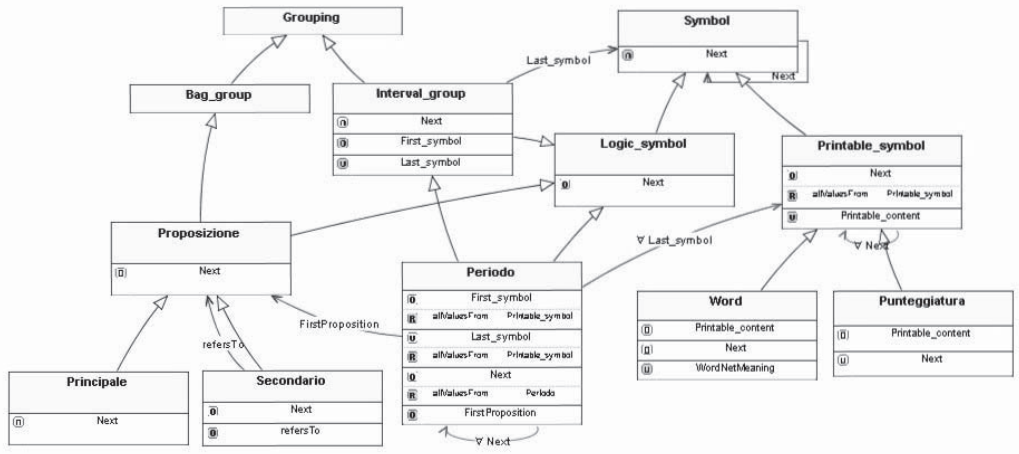


Figure 5 A UML diagram representing classes and relations between classes in our prototype ontology for supporting generic Italian language encoding.

3.2.2 Implementing a markup ontology

On top of these basic structures specific markup ontologies can be implemented as needed.

Figure 5 shows a schematic ontology for which supports the encoding of metric and grammar suitable for the Commedia of Dante Alighieri. The 2 main classes reflect what has been described so far. Classes deriving from the abstract "Symbol" class can be form "chains". Examples of Symbols are the printable symbols (such as words or punctuation) and logical symbols such as periods and propositions. The abstract class Grouping represents a generic annotation. These can be intervals (therefore relying on the underlying chain of symbols) or bags (forming arbitrary sets). Multiple inheritance is a very useful feature of the RDF/S semantic; in this example both the class period and proposition are at the same time Symbols and Grouping, albeit supporting different properties.

3.2.3 Querying the model

In this section we present two ways to query a model as the one described above, namely programmatically and by using the existing Semantic Web query languages. The RDF/S languages have in fact query languages such as SeRQL[11], RQL[12] and the forthcoming SPARQL[13] which provide powerful construct for operating over graph structures often also taking into consideration the class hierarchies (SeRQL, SPARQL).

However, in order to apply these tools to our model, we have to make a distinction and a few simple adaptations. By design, all query languages operate on fixed path length matching. This means that constructs like the linked lists (implemented by the "firstSymbol" in IntervalGroup and "next" property in the Symbol class) are not directly supported. There are different, relatively straightforward, solutions to this problem.

We could, instead of using the first_Symbol and last_Symbol property, use the proper RDF constructs for grouping such as rdf:bag. This connects directly all the contained elements with the RDF element "rdf:li". In this case we could use any existing query language out of the box. We'll take as an example query "Which periods are entirely or partially in page 2?" to be executed over a model built with the above overlapping

hierarchy (Page->Rows->Words<-Sentences). This could be written as (pseudo SeRQL):

```
SELECT distinct PERIOD where {PERIOD} <rdf:type> {<a:period>},
                belongsTo(WORD,PERIOD},belongsTo(WORD,ROW}, belongsTo(ROW,PAGE)
```

Another solution, which also enables much more powerful analysis, is to explore the model programmatically, using the existing manipulation tools such as Jena [14] and Sesame [15]. While this in theory could be compared with using the DOM model exploration API available for XML, the higher expressivity of the API and the perfect adherence of the RDF graph model with the annotation model make query by programming a much more realistic task. Given just a very basic wrapping API built on top of any RDF toolkit, such query could be performed by the following script in pseudo Java.

```
page=Model.getNode(Pagina2);
rowIterator=page.getSymbolIterator();
while (row=rowItarator.getNext) {
        wordIterator=row.getSymbolIterator();
        while (word=wordIterator.getNext()) {
                period=word.getConnected("first_or_last",PeriodType);
                if period!=null then results.add(period); }}
```

The example above makes use of an imperative language and a graph exploration API. This is certainly not the ideal combination, albeit a very popular one.
However, once in the Semantic Web domain, a number of alternatives are available. One of the most powerful ones is using a Semantic Web aware Prolog interpreter such as SWI-Prolog [16]. Using one such language, it is possible to craft very powerful constructs for later reuse.
For example, once the following general purpose constructs have been defined for handling the Interval Class:

```
// Checks if or lists the symbol follows another in the chain
follows(X,X).
follows(X,Y):-next(X,Y).
follows(X,Y):-follows(X,Z),next(Z,Y).
// Checks if or lists the elements between 2 symbols
range_belongs(X,First,Last):-follows(First,X), not follows(Last,X).
// Checks if or lists the symbols in a interval
belongs(Leaf,Root):-first(Root,First),last(Root,Last),range_belongs(Leaf,First,Last).
// Checks if or lists (recursively) the leaves given a root interval
sub_belongs(Leaf,Root):-belongs(Leaf,Root).
sub_belongs(Leaf,Root):-belongs(NewRoot,Root),sub_belong(Leaf,NewRoot).
```

Formulating our test query simply means adding an explicit rule and firing it:

```
sentence_in_page(Sentence,Page):-sub_belongs(Word,Page),belongs(Word,Sentence).
sentence_in_page(X,21);
```

to obtain the matching sentences. Given the nature of Prolog, the same rule, as well as those above, can immediately be used for the opposite purpose, e.g., given a sentence find which page (or pages) it belongs to.
Without resorting to a full a powered Prolog, it is also possible to use other reasoners such as those implemented in inside toolkits like Jena to encode similar query rules.

3.3 Cooperative, incremental markup

Given that any conceptual entity (e.g. Aspect of the encoding) is a resource in RDF, it might be very useful to give this entity a proper, globally addressable, name that is, a "stable" URI. Once URIs are agreed upon, the SW semantics clearly specifies the rules for document merging, immediately allowing an interesting scenario: annotations could be made cooperatively and incrementally in a distributed way. Independent documents, independently edited, could provide encodings of different aspects based on the same base URIs. At any time these could be merged into a unique document or simply taken into consideration, e.g. by a SW reasoner, as if they were one.

3.4 Ontologically speaking

While the graph structure provides the fundamentals of the encoding, it is the ontological aspects that probably could bring the most interesting possibilities. In general, instruments available for XML such as XPath and the relative enhancement for the overlapping annotations EXPath do not take ontological aspects into consideration. Even if XQuery supports concepts such as type extension, this is a way to provide more a validation syntax, rather than to create ontological concepts such as classes, properties and restrictions on these [17].

Using the proposed encoding scheme, the SW tool OWL becomes available. OWL, a W3C recommendation since 2004, is semantically coherent with the RDF model and enjoys ever increasing supported by APIs and understanding by the community.
OWL provides a number of extended functionalities which we will just hint at here. First of all it provides a solid base for model validation. For example, in our example annotation ontology, cardinality constraints are specified so that there has to be 1 and only 1 next for each basic symbol (except the last which points to a special "end of the document" symbol). Each interval must then have 1 and only 1 "first" and "last" and so on.
More advanced aspects of the ontology, which cannot be directly validated by the OWL description, can nevertheless be checked with simplicity if rule systems such as those previously defined are available.
Reasoners, however, go well beyond checking a model validity. When certain conditions are met or found in the model, new statements can be added or existing ones operated upon. Some of the advantages with respect to the simple XML schema are mentioned in [18]. As far as querying is concerned, the use of ontologies gives important capabilities. As a basic example, given an ontology for "manuscripts" including a taxonomy of "cancellations" (e.g. from light pencil map, to the word or a paragraph being ripped off the paper), one could encode the content using the most appropriate term for each case and be able to perform queries that contemplate all the sub cases such as "what is the most cancelled word in this manuscript?".

4 Conclusions and future work

In this paper we presented an overview for a proposed textual encoding methodology based on the languages of the SW, namely RDF, RDFS and OWL. Although in this proposal we described possible advantages over other schemes which simply take care of the overlapping hierarchy problem, we believe this overview shows how the soundness of the semantic theory and the power of the available tools fully justifies further studies and analysis in this direction, investigating on a new encoding standard and how existing ones can be ported, made compatible or enhanced to the SW model.
We report being currently in the process of developing RDFTef, an open source java framework supporting textual encoding in RDF+OWL and capable of exporting these models either as XML/RDF and, in form of aspect slices, as XML TEI format. As the implementation is in an early phase, detailed evaluation of the proposed method will be a future task.

References

[1] Steve Battle, Round-tripping between XML and RDF, ISWC 2004
[2] RDF Primer, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-primer/
[3] Web Ontology Language Overview, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-features/
[4] Sperberg-McQueen C. M., Burnard L. (2002), TEI P4: Guidelines for Electronic Text Encoding and Interchange.
[5] Steven DeRose, Markup Overlap: A Review and a Horse, 2004
[6] Patrick Durusau, Just-In-Time-Trees (JITTs): Next Step in the Evolution of Markup?, 2002
[7] Patrick Durusau , Implementing Concurrent Markup in XML, 2001
[8] Steven DeRose, Markup Overlap: A Review and a Horse, 2004
[9] Jeni Tennison, Wendell Piez, Layered Markup and Annotation Language (LMNL), 2002
[10] Alex Dekhtyar, Ionut Emil Iacob, A Framework For Management of Concurrent XML Markup, 2004
[11] Jeen Broekstra and Arjohn Kampman. SeRQL: An RDF Query and Transformation, Language. ISWC 2004.
[12] Karvounarakis, Gregory and Alexaki, Sofia and Christophides, Vassilis and Plexousakis, Dimitris and Scholl, Michel (2002) RQL: A Declarative Query Language for RDF. International WWW Conference (11), Honolulu, Hawaii
[13] SPARQL Query Language for RDF, W3C Working Draft 12 October 2004, http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/
[14] Carroll, Jeremy J.; Dickinson, Ian; Dollin, Chris; Reynolds, Dave; Seaborne, Andy; Wilkinson, Kevin, Jena: Implementing the Semantic Web Recommendations, HP Technical Report, 2004
[15] J. Broekstra, A. Kampman and F. van Harmelen: Sesame: A Generic Architecture for Storing and Querying RDF, proceedings of the first International Semantic Web Conference (ISWC2002), Lecture Notes in Computer Science
[16] Jan Wielemaker, An Overview of the SWI-Prolog Programming Environment, WLPE'03, 2003, Mumbai, India
[17] Yolanda Gil and Varun Ratnakar, A Comparison of (Semantic) Markup Languages
[18] Matthias Ferdinand, Christian Zirpins, and David Trastour, Lifting XML Schema to OWL, 2004