

Addressing Obstacles to the Retrieval of Structured Documents

David S. Dubin and Allen Renear
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign

C. M. Sperberg-McQueen
World Wide Web Consortium
MIT Laboratory for Computer Science

July, 2003

The potential for document markup, such as SGML and XML, to support information retrieval is receiving considerable attention. However the generally underdetermined and implicit nature of even the most basic semantic relationships expressed by SGML/XML markup is an obstacle to its effective exploitation, as is the assumption of “semantic transparency.” A project to develop a adequate machine-readable formalism for expressing markup semantics is described, along with specific applications to retrieval problems. A Prolog environment supporting inferences and queries is also used to generate document abstractions based a formalized semantics for markup vocabularies.

ISRN UIUCLIS- -2003/1+EPRG
Copyright ©2003 by David Dubin and the Trustees of the University of Illinois.

Contents

0.1	Introduction	1
0.2	Project Goals	4
0.3	Addressing Representative Problems	6
0.4	Conclusion	10

List of Figures

1	Block quote tagged as a list	7
2	List object must be inferred	9

0.1 Introduction

The role of structured document standards in advancing the state of the art in information retrieval has received considerable attention over the past ten years. To name only a few examples, researchers have proposed and tested methods for retrieving only the most relevant parts of documents [8], discovering representative structures in databases of structured documents [17], and have developed new query languages for retrieval of XML-encoded documents [6, 9, 11, 16]. Egnor and Lord have summarized a variety of ways in which the structure of documents represented using XML or SGML-based languages can be leveraged as part of an IR system:

- Limitation of search results to only those documents that conform to a particular schema.
- Disambiguation of word sense by the context of the element in which a word appears.
- The use of rich data types in queries (e.g., dates, locations, dollar amounts).
- Word proximity definitions based on hierarchical structure, rather than just the number of tokens separating two words.
- Retrieval of specific document elements, rather than entire documents. [5]

However the potential of every one of these opportunities to improve IR effectiveness is limited by the degree to which the meaning of the markup is readily available to processing. Most XML-related IR projects work directly with the markup tokens, and either make no explicit inferences about the meaning of the markup or make *ad hoc* inferences as needed based on intuitively obvious relationships or the documentation for the markup vocabulary. We believe that XML-related IR projects would be both simpler to carry out, and more successful in their results, if the fundamental semantic features of XML markup are made accessible to processing, enabling full access to the basic meaning of SGML markup at an appropriate level of abstraction.

The word “semantics” is used with many meanings. It is important at the outset to be clear that by “markup semantics” we do not mean either

1. the semantics of the document *contents*,

2. intended processing such as formatting rules or “business rules”, or
3. the functions, axioms, or abstract machines that define operations on the markup in the manner of programming language semantics [7].

By markup semantics we refer to those basic facts and relationships the expression of which the markup system is intended to enable (on a class-by-class, not occurrence-by-occurrence basis). Specific assertions of these facts and relationships are intended by the markup language user and easily inferred by a software designers. These facts and relationships are often described, at least in part, in the prose documentation for the markup vocabulary. But because there is no established systematic and machine-readable formalism for presenting them they cannot be easily and automatically exploited.

What sort of facts and relationships does “markup semantics” in our sense consist? Document authors and readers identify meaningful abstract objects within documents (e.g., paragraphs, headings, footnotes, tables). These objects have properties, such as having a particular title, or being in a particular language. Furthermore, the object instances stand in specific relations to one another (e.g., whole/part, citation, explanation, illustration, digression). Markup systems are designed to allow us to express the existence of these objects and relationships; to say for instance, that something *is a paragraph*, and is *in German*. So in our work the “semantics” of an SGML-encoded IRS publication would not be (for example) rules for declaring dependent care expenses, nor would they be rules for formatting the document, as expressed in a stylesheet. Instead, they would include the fact that something is an *example* and something is a *definition*, and that the example *illustrates* the definition.

Declarative, structural markup represents a significant advance over formatting markup for representing these objects, properties, and relations in explicit, machine-processable formats [3]. It might even appear that the automatic generation of markup semantics for a markup language would be trivial. However, a number of problems lurk behind the assumption of semantic transparency implicit in IR research on structured documents. These problems range from the subtle to the very crude:

1. As with the semantics of natural language, authors and readers are often unaware of inferences or distinctions which they make without effort, but for which the markup language provides no means to explicate. Examples include situations in which the object to which an expressed property applies

is not clear. Markup in a series of attribute assignments on a single element might appear to indicate that something — apparently the same thing — is-a-noun, is-a-French-citizen, is-illegible, and has-been-copyedited. But ontological coherence requires that either these predicates really refer to different things, or must be given non-standard interpretations.

2. There is no technique for specifying inheritance rules for properties expressed by XML markup. Obviously the property of being a paragraph is not inherited by its children, while the property of being in German is; but there is no way to express this. Nor is it possible to define the rules for inheritance: some properties exhibit defeasible inheritance behavior, (language properties), others display an additive algebra (rendition assignments, when not explicitly exclusive); some properties are inherited only by elements, others by elements and element content both. Again, this behavior is easily inferred and relied on by markup users, but there is no systematic machine readable way to describe specify it.
3. As noted by Cover [4], SGML and XML-based languages simply aren't well suited to complex data modeling. Research on the retrieval of partial documents may be carried out on the assumption that the parent-child relationship between SGML or XML elements represents a semantic *includes* or *whole/part* relationship. In practice, the generic relationship between parent and child elements is highly overloaded. This can be seen in the contrasts among (for example) a section element's relationship to a paragraph child, the same section's relationship with its heading child, a paragraph's relationship to an embedded emphasized phrase, and the same paragraph's relationship to an embedded footnote element. The W3C Resource Description Framework [14] is a demonstration that a data modeling language can be expressed in XML, but if most XML-based languages encoded their own semantics, there would be no need for RDF in the first place.
4. Not only can the parent-child relationship represent any number of specific named relations, in some documents it might *not* represent the relationship of parts to wholes. Most HTML documents on the web, for example, have a flat structure, and cue logical hierarchical divisions using header elements as milestones.
5. Perhaps most problematic for information retrieval applications is that even to the extent SGML and XML permit explicit meaningful labels to be as-

signed, this capability is under-utilized in actual practice. Notwithstanding the subtle distinction between labeling something as a list and encoding what it means to *be* a list, one would hope, at least, that all lists would be marked as lists, and no non-list elements would be so labeled. Sadly, this is no safe assumption apart from collections designed as research testbeds. In the real world, documents are often written using tools that aren't SGML-aware. Their conversion to an SGML or XML-based language is part of a larger editorial process geared to meet local needs. Their integration into a collection or repository involves reconciliation among incompatible or semi-compatible document classes and schema families [1]. In the following sections we illustrate how difficult markup interpretation can be under these circumstances.

The practical problems of interpreting markup in realistic documents can be worked around on a case by case basis as they arise. But they also invite attempts at more deliberate, systematic solutions. We next describe BECHAMEL, a research project aimed at developing such solutions.

0.2 Project Goals

The BECHAMEL Markup Semantics Project grew out of research initiated by Sperberg-McQueen (W3C/MIT) in the late 1990s [13] and is a partnership with the research staff and faculty at the Department for Culture, Language and Information Technology, Bergen University Research Foundation, and the Graduate School of Library and Information Science Electronic Publishing Research Group at the University of Illinois, Urbana-Champaign. The project has the following research goals:

1. Explicating every important representation and inference issue germane to document markup semantics, and developing a taxonomy and description of the problems any semantics-aware document processing system must solve or address.
2. Analyzing in detail the properties and semantic relationships common to popular markup languages, and evaluating the applicability of standard knowledge representation technologies — such as semantic networks, frames, logics, formal grammars, and production rules — with respect to their expressive adequacy, elegance, parsimony, and computational efficiency for modeling these relationships, properties, and constraints.

3. Developing and testing a formal, machine-readable representation scheme in which every semantic distinction of a markup language can be expressed.
4. Exploring applications of the semantic representation, such as transcoding support, information retrieval, accessibility improvement, etc. Current focus is on the support of semantic inferences from databases of document instances, as this is believed to best direct the general decisions about choice of representation techniques.
5. Conducting, in partnership with digital library content encoding projects from the humanities computing community, and associated software tool developers, large scale tests of the resulting semantics representation scheme.

Project investigations are conducted with the aid of tools for exploring alternative approaches to specifying the relationships expressed by XML markup and for reasoning about those relationships. Specifically, a reasoning system (written in Prolog) includes the following functional components:

- A syntactic layer that encodes relationships among document elements, their attributes, parents, and children.
- A layer that encodes relationships among domain entities, objects, and their properties.
- Mechanisms that support various mappings between the syntax layer and the object layer.
- An inference engine that enables new facts to be drawn on the basis of the facts and rules in the knowledge base.

At the syntactic layer, the system employs an integrated parser so that SGML and XML instances can be input and output. A collection of predicates emulate a subset of the W3C's Document Object Model [15] methods for navigating the hierarchical structure of nodes, and retrieving attribute values and information from the document type definition. Other syntax-level predicates support deictic expression resolution. These allow rules of inference to include location-relative pointing from one part of the document to another. For example, predicates can resolve an element's closest ancestor having a particular generic identifier, attribute, or attribute value pair. Another set of predicates resolves the identity of an element of a particular type occurring most closely in terms of the linear structure of the document (rather than the closest in the hierarchy).

The object layer consists of predicates for reasoning about objects, their classes, properties, and the relationships holding among them. For a markup semantics application, the object classes may represent such things as paragraphs, sections, footnotes, and so on. The semantics of constraints and relationships concerning such objects would typically be based on documentation of a tag set or document class written in natural language.

The system supports experimentation with a variety of approaches to bridging the syntactic and object layers of representation. One approach uses a blackboard model [18], in which mapping rules have access to both the results of the parse and the emerging network of objects, properties and relations. A rule's execution may add a new object, set a property value, or establish a named relation between objects. Any such modification may then trigger the firing of additional rules.

0.3 Addressing Representative Problems

We next illustrate our approach to coping with the lack of congruence between syntactic and semantic representation in structured documents. The following examples are taken from a journal article [10] retrieved from the University of Illinois DLI Testbed [1, 2, 12], which conforms to a DTD based on ISO 12083. We present rules for overcoming the following incongruencies:

- Elements selected by the author or editor to exploit accidents of their expected formatting (i.e., common tag abuse).
- A semantic object to which no single syntactic element corresponds.

In the first example, elements for marking lists and list items have been to tag what is clearly a block quote:

```
In this paper, we adopt the loose definitions of an
agent as <L><LI><uichar class="default"
entity="&lsquo;" unicode="&#x2018;"></uichar>
a self-contained program capable of controlling its
own decision-making and acting, based on its
perception of its environment, in pursuit of one or
more objectives<uichar class="default"
entity="&rsquo;" unicode="&#x2019;">
</uichar><RB RID="BR11" OCC="1">
</RB></LI></L>
```

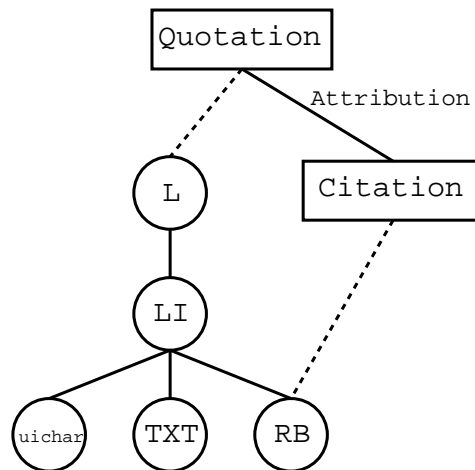


Figure 1: Block quote tagged as a list

The article has several block quotes marked this way, and the evidence available for recognizing them is that they are all single element lists, where the list item begins with a left single quote character (the **uichar** element) rather than a bullet or alphanumeric item label. Figure 1 illustrates the desired mapping from element nodes to object instances: the element tagged as a list is semantically a block quotation, and the **RB** element maps to a citation object. In this case, the generic parent/child relationship between the **LI** node and the **RB** node represents an attribution relationship: the citation attributes the quotation.

One of the rules for realizing the desired mapping is illustrated below. Its interpretation is as follows: if there exists an **L** node, and its first child has no next sibling (i.e., if something is tagged as a one-element list) and if the first child of the list item is a **uichar** that represents a left single quote, and if no quotation object has been mapped to this **L** node, then construct such an object instance.

```

mrule :- node_exists(Lnode, 'L'),
         first_child(Lnode, Inode),
         not(nsib(Inode, _)),
         first_child(Inode, Cnode),
         gi(Cnode, uichar),
         atval(Cnode, entity, Char),
         char(lsquo, Char),
         not(exists(_, quotation, [Lnode])),
         construct(quotation, [Lnode], _).

```

Not only does this document exhibit quotations tagged as lists, there are also lists that aren't tagged. A fragment of one such example is shown below:

```
which society will have to deal with through various
pieces of legislation. They include</P>
<P><uichar class="Symbol" entity="&bull;"
isocode="&#183;" unicode="&#x2022;">
</uichar>
  privacy: how do you ensure your agents maintain
your privacy when acting on your behalf?</P>
```

This sequence of paragraphs, each beginning with a bullet character, is clearly an itemized list. However, no syntactic element encloses the list items. The needed mapping from the syntactic to the object layer is illustrated in figure 2: the **P** nodes that begin with a bullet character must be mapped to list item objects, and pairs of such objects mapped from sibling paragraph nodes need a “follows” relation between them. These list item objects stand in part/whole relation to a list object that corresponds to no syntactic element. That list takes a value of “Itemized” for its **style** property.

A number of independent mapping rules are needed to realize the mapping shown in figure 2. Two such rules are shown below. The first discovers paragraphs that begin with a bullet character, and maps them individually to list items that are part of lists. Its interpretation is as follows: if a **P** node exists whose first child is a **uichar** representing a bullet character, and no list item has been mapped to that paragraph node, construct such an object and let it stand in part/whole relation to a new list object. A separate rule (not shown) discovers and merges list objects that contain list items that follow on one another.

```
mrule :- node_exists(Pnode, 'P'),
         first_child(Pnode, Bnode),
         gi(Bnode, uichar),
         atval(Bnode,entity,Char),
         char(bull,Char),
         not(exists(_,listitem,[Pnode])),
         construct(listitem,[Pnode],Listitem),
         construct(list,[Pnode],List),
         apply_relation(part_of,[Listitem,List]).
```

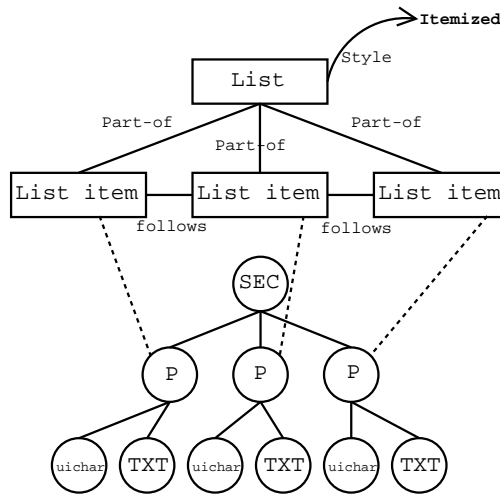


Figure 2: List object must be inferred

The second rule discovers list item objects that have been mapped from sibling nodes, and establishes a “follows” relation between them, if no such relation has yet been created:

```

mrule :- exists(Item1, listitem, [Node1]),
         exists(Item2, listitem, [Node2]),
         nsib(Node1, Node2),
         not(relation_applies([follows, Item2, Item1])),
         apply_relation(follows, [Item2, Item1]).

```

These are a few examples of how our approach permits the logical relationships among document constituents to be represented, even when markup syntax doesn’t encode those connections explicitly. We are also using the system to infer hierarchical structure in flat documents, where the logical divisions are cued by milestone elements (such as headers), rather than with properly nested sections and subsections. In addition, we have examples where the instance of an element (such as an index term) represents neither an object instance nor a property/value pair, but rather a relationship (e.g., an index term describes a logical division). Information retrieval applications of this framework may invite the kind of speculative, *post hoc* semantic analysis that we describe above. But the formalisms can also be used *in advance* of authorship to specify the semantics of markup tags, just as schemas and DTDs communicate their valid syntax.

0.4 Conclusion

SGML/XML document markup appears to hold considerable promise for enhancing information retrieval. However there are obstacles to realizing this promise. Overcoming these obstacles requires improved techniques both for expressing the meaning of SGML/XML markup and for automating the development of document abstractions from document instances and markup language semantics specifications. Obviously it is possible for retrieval projects to proceed without these improvements—by continuing the current practice of partial and *ad hoc* exploitation of markup. But it is obvious that this approach, although effective with fixed document sets of modest size and simple structures, will not scale to large or heterogeneous collections; let alone provide the improvements in effectiveness document markup seems to offer. Fortunately the development of both markup semantics formalisms, and the automated techniques for document preparation are, as we have shown, manageable tasks.

Bibliography

- [1] COLE, T., AND KAZMER, M. SGML as a component of the digital library. *Library High Tech* 13, 4 (1995), 75–90.
- [2] COLE, T. W., MISCHO, W. H., FERRER, R., AND HABING, T. G. Using XML, XSLT, and CSS in a digital library. In *ASIS 2000: Proceedings of the 63rd ASIS Annual Meeting* (Medford, NJ, 2000), D. H. Kraft, Ed., American Society for Information Science, Information Today, pp. 430–439.
- [3] COOMBS, J. H., RENEAR, A. H., AND DEROSE, S. J. Markup systems and the future of scholarly text processing. *Communications of the Association for Computing Machinery* 30, 11 (1987), 933–947.
- [4] COVER, R. XML and semantic transparency. Technology report, Cover Pages, 1998. Published on the Worldwide Web at <http://www.oasis-oprn.org/cover/xmlAndSemantics.html>.
- [5] EGNOR, D., AND LORD, R. Structured information retrieval using XML. Presented at the ACM SIGIR 2000 Workshop on XML and Information Retrieval, Athens, Greece, July 2000. Published on the World Wide Web at <http://www.haifa.il.ibm.com/sigir00-xml/final-papers/Egnor/>.
- [6] FUHR, N., AND GROSSJOHANN, K. XIRQL: A query language for information retrieval in XML documents. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, 2001), ACM SIGIR, Association for Computing Machinery, pp. 172–180.
- [7] HOROWITZ, E. *Fundamentals of Programming Languages*. Computer Science Press, Rockville, MD, 1983, ch. 2.

- [8] LALMAS, M. Dempster-shafer's theory of evidence applied to structured documents: modelling uncertainty. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, 1997), ACM SIGIR, Association for Computing Machinery, pp. 110–118.
- [9] NAVARRO, G., AND BAEZA-YATES, R. A language for queries on structure and contents of textual databases. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, 1995), ACM SIGIR, Association for Computing Machinery, pp. 93–101.
- [10] NDUMU, D. T., AND NWANA, H. S. Research and development challenges for agent-based systems. *IEEE Proceedings on Software Engineering 144*, 1 (1997), 2–10.
- [11] ROBIE, J., LAPP, J., AND SCHACH, D. XML query language (XQL). Published by the World Wide Web Consortium at <http://www.w3.org/TandS/QL/QL98/pp/xql.html>, September 1998. Proposal to the XSL Working Group.
- [12] SCHATZ, B., MISCHO, W. H., COLE, T. W., HARDIN, J. B., BISHOP, A. P., AND CHEN, H. Federating diverse collections of scientific literature. *Computer 29* (May 1996), 28–36.
- [13] SPERBERG-MCQUEEN, C. M., HUITFELDT, C., AND RENEAR, A. Meaning and interpretation of markup. *Markup Languages: Theory and Practice 2*, 3 (2000), 215–234.
- [14] W3C. Resource Description Framework (RDF) model and syntax specification. Published by the World Wide Web Consortium at <http://www.w3.org/TR/REC-rdf-syntax/>, February 1999.
- [15] W3C. Document Object Model (DOM) level 1 specification. Published by the World Wide Web Consortium at <http://www.w3.org/TR/REC-DOM-Level-1>, September 2000. W3C Recommendation.
- [16] W3C. XQuery 1.0: An XML query language. Published by the World Wide Web Consortium at <http://www.w3.org/TR/xquery>, November 2002. W3C Recommendation.

- [17] WANG, K., AND LIU, H. Discovering typical structures of documents: A road map approach. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, 1998), ACM SIGIR, Association for Computing Machinery, pp. 146–154.
- [18] WINSTON, P. H. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 1984, ch. 5.