Towards a Semantics for XML Markup

Allen Renear and David Dubin
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
{renear, ddubin}@uiuc.edu

C. M. Sperberg-McQueen
World Wide Web Consortium
MIT Laboratory for Computer Science
cmsmcq@acm.org

Claus Huitfeldt
Department for Culture, Language, and Information Technology
Bergen University Research Foundation
Claus.Huitfeldt@hit.uib.no

ABSTRACT

Although XML Document Type Definitions provide a mechanism for specifying, in machine-readable form, the syntax of an XML markup language, there is no comparable mechanism for specifying the *semantics* of an XML vocabulary. That is, there is no way to characterize the meaning of XML markup so that the facts and relationships represented by the occurrence of XML constructs can be explicitly, comprehensively, and mechanically identified. This has serious practical and theoretical consequences. On the positive side, XML constructs can be assigned arbitrary semantics and used in application areas not foreseen by the original designers. On the less positive side, both content developers and application engineers must rely upon prose documentation, or, worse, conjectures about the intention of the markup language designer — a process that is time-consuming, error-prone, incomplete, and unverifiable, even when the language designer properly documents the language. In addition, the lack of a substantial body of research in markup semantics means that digital document processing is undertheorized as an engineering application area. Although there are some related projects underway (XML Schema, RDF, the Semantic Web) which provide relevant results, none of these projects directly and comprehensively address the core problems of XML markup semantics. This paper (i) summarizes the history of the concept of markup meaning, (ii) characterizes the specific problems that motivate the need for a formal semantics for XML and (iii) describes an ongoing research project — the BECHAMEL Markup Semantics Project — that is attempting to develop such a semantics.

Categories and Subject Descriptors

I.7.2 [**Document Preparation**]: Document and Text Processing—*Markup languages*

General Terms

Theory, Standardization

Keywords

SGML, XML, Markup, Semantics, Knowledge Representation

DocEng'02, November 8–9, 2002, McLean, Virginia USA.

1. INTRODUCTION

As a consequence of recent rapid advances in digital publishing, the explosion of WWW based applications, and the growing interest in electronic commerce, many aspects of our daily social, commercial, and cultural lives now involve systems based on SGML/XML document markup. SGML/XML provides a rigorous machine-readable technique for defining descriptive markup languages — languages which explicitly identify the underlying meaningful structure of document, apart from the intended processing. The expectation motivating the SGML/XML approach was that the widespread use of such a specification would support high-function interoperable document processing and publishing systems.

This expectation has been partly fulfilled, and the superiority of the SGML/XML approach over earlier strategies confirmed, but there remain continued opportunities to improve the functionality, interoperability, heterogeneity, and accessibility of SGML/XML-based document processing. The practical consequences of a failure to seize these opportunities are serious: on an industry-wide scale they have considerable financial costs and lost opportunities, in safety-critical applications they can result in disaster, and in their impact on the perceptually disabled they can prevent equitable access to the cultural and commercial benefits of contemporary society. Moreover, the continuing existence of these problems reminds us that our best models of digital document processing remain flawed, or at least underdeveloped.

The source of these problems is that even though SGML/XML is thought of as providing access to a document's meaningful structure, current SGML/XML methods cannot represent the fundamental semantic relationships amongst document components and features in a systematic machine-processable way. SGML/XML supports the specification of a machine readable "grammar", but because it has no mechanism for providing a semantics for that grammar what an SGML/XML vocabulary *means* still cannot be formally specified. Even very simple fundamental semantic facts about a document markup system — facts that are routinely intended by markup language designers, and relied on by markup language users and software — are inexpressible in current SGML/XML document formalisms.

The result is that SGML/XML markup language users must guess at the semantic relationships the markup language designer had in mind, but had no way to formally express. Content developers must conjecture what they believe to be in the mind of the markup language designer, and rely on these inferred relationships in encoding their content — and then these content developers in turn have no way to formally represent their inferences and intentions to each other, or to the software applications that process the encoded content. Software designers must also make their own guesses about

the likely intentions of markup language designers and then reflect these in the design of software tools and applications. Sometimes in fact a second-order conjecture is necessary: when software designers must also guess at what content encoders inferred about what markup language designers meant!

Obviously these conjectures will be incomplete, error-prone and unverifiable, as well as time-consuming both to make and to implement, resulting in high costs and low levels of functionality and interoperability. The natural-language prose documentation that accompanies an SGML/XML specification is not a wholly satisfactory solution to this problem. Prose documentation does of course provide some assistance to both content providers and software engineers, but there are no established common principles for developing SGML/XML documentation, and, in any case, prose documentation is not a *machine-readable formalism* — and that is what is required to address current problems with SGML/XML markup systems.

Although the absence in SGML and XML of any provision for a machine-processable semantic description of the constructs being defined has been noted by a number of researchers and identified as the source of current engineering problems as well an obstacle to future development, [25] [23] [43] [25] [36] the nature of the needed semantics remains little studied. The W3C Schema effort is relevant, but takes up only a small subset of these problems (e.g. data types). The W3C's "Semantic Web" activity is also relevant, but its agenda is to develop XML-based techniques for knowledge representation in general, while our project focuses on document markup semantics, as latent in actually existing document processing systems — one might say that the Semantic Web effort is designing a *semantic markup* while the solution to the problems identified here requires a *markup semantics*.

In what follows we (i) present the issue of the meaning of markup in historical context (where it plays a interesting role in the evolution of approaches to document processing), (ii) characterize the specific problems that motivate the perceived need for a formal semantics for markup — and which determine the requirements for a satisfactory semantics); and (iii) briefly describe an ongoing multi-institutional research project — the BECHAMEL Markup Semantics Project — that is attempting to develop such a semantics.

2. HISTORICAL CONTEXT

Document "markup" might arguably be considered part of any communication system, including early writing, scribal publishing, and printing, but with the emergence of digital text processing and typesetting the use of markup became self-conscious and explicit and an important area of innovation in systems development [4] [40]. The 1960s, 70s, and 80s in particular saw extensive systematic development of document markup systems, typically focused on improving the efficiency and functionality of digital typesetting and text processing [12] [22] [19] [10] [26] [17] [18]. By the early 1980s there were also efforts to develop a theoretical framework for markup and to use that framework to support the development of high-function systems. Some of this work was published in the scientific literature [11] [27] [4] [40], but much of it was recorded only in the working documents and products of various standards bodies.

One of the key ideas that emerged during this period was the notion that documents, at least qua intellectual objects, were best modeled as ordered hierarchical structures of objects such as chapters, paragraphs, equations and the like, and not as, for instance, one-dimensional streams of text characters with interspersed formatting codes; structures of design objects (pages, columns, typographical lines); matrices of pixel values; or any of the other representational strategies implicit in many existing document processing and storage systems [5]. This model generalized an already existing natural distinction between markup that identified editorial text objects (titles, chapters, etc.) and markup that specified formatting instructions. Use of the former allowed a level of indirection that provided many advantages. [11] [27] [4] Document elements such as titles, chapters, sections, paragraphs, equations, quotations, and the like would be explicitly identified by delimited markup, and then processing would be indirectly applied via rules that mapped procedures to types of elements. This separation of processing and content, implementing a basic level of indirection and abstraction with the usual combinatoric economies, was not only of immense and varied practical value in all aspects of document processing [4] but moreover seemed to reflect a correct engineering model of what a document "really is" [5]. The descriptive markup used to implement this approach did not just mark element boundaries, but carried the meaning (e.g. *this text is a chapter*) needed to realize the specific underlying document model.

The influential SGML document markup meta-grammar emerged from the ANSI/ISO effort in the early 1980s and reflected the preceding theoretical and analytical work on markup and document structure. SGML provides a machine-readable formalism for defining descriptive markup languages. As a metagrammar SGML does not define a markup language, but rather specifies the techniques for developing a machine readable definition of a markup language. The central mechanism for that definition is a BNF-like formalism with additional rules for specifying typed attribute/value pairs and some other devices for further abstraction and indirection. (See [30] for warnings about the degree of similarity between DTDs and BNF, however.) Structurally an SGML document instance is a tree with ordered branches and labeled and annotated nodes and it is a formal production of its corresponding DTD.

The fundamental ideas behind SGML, based on years of both analysis and practical experience, were compelling. And in addition the specific mechanisms of SGML (the BNF-like metagrammar, typed attribute/value pairs, entity references, etc.) promised a high-function implementation in applications and tools, with the benefits of industry-wide standardization at the meta-grammar level, and local innovation at the level of vocabulary. The SGML markup language development process itself also seemed to both support and refine natural and ideal workflows for document systems design, implementation, and use. In the mid 1980s to early 1990s a number of SGML-based markup systems were developed[1] [42] [39].

Despite the care taken in the development of SGML, the sound and field-proven nature of its ideas, and the success of at least several implementations, there was disappointingly little adoption and success in the first decade. Although a number of circumstances contributed to this, the principal reason was the complexity of the standard itself, including the fact that SGML incorporated many complicated optional features that conforming software might, but need not, implement. SGML software development consequently proceeded very slowly. One further result of these accommodations and confusions was that even a non-validating parse of a document was not possible without processing the DTD — abbreviation options meant that element boundaries could not be reliably determined without reference to the document grammar. In addition, SGML includes several other features that resulted in a formal grammar unsuited for use with existing parser development tools and for which it is difficult to write efficient parsing routines.

The eventual widespread use of SGML systems in networked publishing and communication was a result of HTML (the Hypertext Markup Language). The initial versions were loosely defined

and lacked any formal specification of syntax; and when interest in an SGML DTD for HTML later developed, it proved difficult to design a DTD after the fact that would accommodate already established "correct" practice. More importantly the key distinction between descriptive and procedural markup was ignored by developers and users alike, as HTML vendors casually added obviously procedural markup (e.g.) to the predominantly descriptive markup (e.g. <title>) of the original HTML specification. Even the descriptive parts of HTML did little to reflect the document hierarchy, and the specification provided no stylesheet language to support indirection. Finally there was also no use of SGML mechanisms for extending an element set or using alternative element sets, and, indeed, the assumption seemed to be that HTML documents would be processed not by generic SGML processors (which would allow extensions or alternative DTDs) but by HTML-specific formatters that processed only HTML elements, and only with the formatting rules hard-coded in the processor.

Much of the subsequent evolution of HTML can be seen as an effort to evolve the casually developed original HTML language to the sort of SGML language that would have been designed if there had been time and resources to apply established principles of document system design. At the same time, there was pressure on the newly formed W3C to allow for new element sets, and to provide for the use of SGML on the Web. It was soon recognized that the flaws in SGML were an obstacle to this project — and, more generally, to realizing the advantages of SGML and descriptive markup on the Web. These problems included the number of optional features in SGML, the complexity of its formal grammar, and the need to refer to a DTD to determine element boundaries.

In order to ensure that HTML and other related technologies could more easily benefit from advantages of a metagrammar, that users could easily develop and share new domain-specific elements, that documents could be parsed into trees of elements without reference to a DTD, and that the logjam of SGML tool and application development was broken, the W3C created a subset of SGML, intended to provide a simpler standard (no options), a simpler grammar, and support for the (non-validating) processing of documents without reference to DTDs. This is the metagrammar XML, released as a W3C Recommendation in 1998, after a year and a half of development.

Since 1998, there has been an explosion of new XML markup languages, and that rapidly expanding growth continues today. This explosion is the result of several factors:

1. the demand for new markup systems for special domains, caused by the increasing use of network electronic publishing in science, medicine, business, law, engineering, and in specialized areas with these large domains.

2. the lowering of the cost and complexity of developing new tools and applications, a result of the simplicity of parsing XML compared to SGML

3. the use of XML markup to support the integration of other sorts of processing and communication with publishing applications — or even to support the integration of application unrelated to publishing.

There is thus reason to be optimistic. We finally have a powerful easily implementable technique for creating high-performance markup languages, digital documents, and document processing and publishing systems that can be integrated with other information management applications. In particular, the prospect of finally having a deep processable purchase on the underlying *meaning* of document structure promises extraordinary new functionality, releasing for automated processing information that had up until now been inaccessible, at least without impractical amounts of human intervention.

3. THE PROBLEM

Unfortunately recent experience and reflection has made it clear that our understanding of how descriptive markup conveys meaning, and our techniques for supporting the expression and processing of that meaning, are inadequate to realize fully its promise.

The systematizing and theorizing of document markup that took place throughout the 1980s focused on three things:

1. the conceptualization of a general document model [5]

2. the development of techniques for the formal specification, vocabulary and syntax, of document markup languages that could define specific classes of documents and represent actual documents as instances of that model [14]

3. the development of those markup languages (e.g. CALS, AAP, TEI, HTML, etc.)

It seemed that identifying and annotating the logical parts of a document using a descriptive markup language explicitly delivered the "meaning" that had been only implicit, at best, in the procedural markup, making that meaning unambiguous, explicit, and available for mechanical processing.

Many people refer to XML documents as "self-describing data". Although there were a few early voices of dissent (see Mamrak [20] and most importantly Raymond and Tompa [24]), as the initial period of descriptive markup enthusiasm drew to a close it appeared that most document researchers felt that there was no need for a more expressive representation. A well-defined SGML markup language delivered the meaning latent in document structure, making it fully, and efficiently, available to processing. In a breathless sentence co-authored by one of the authors of this article (and later selected for leading quotation in an issue of TUGBoat, the TeX Users Group journal), "In the end, it should be clear that descriptive markup is not just the best approach of the competing markup systems; it is the best imaginable approach." [4].

The experience of the 1990s has shown that this confidence was not entirely warranted. Although our situation is today certainly much improved from a practical point of view, a pattern of repeated failures of interoperation and functionality have made it evident that SGML/XML still does not really succeed in providing, in an explicit machine-processable way, the meaning latent in document structure — this meaning still must be, as before, indirectly inferred. The rigor of the element and attributes in an SGML/XML DTD is not matched by a similar rigor in the rest of the document type definition, the part not formalized. The grounds of the needed inference are different to be sure, and no doubt improved, but our situation is qualitatively not as dissimilar to pre-SGML document processing as one might think — the key determinations of meaningful structure are still made by human beings reflecting on relatively implicit and ambiguous cues.

Reflection on the nature of the DTD shows why this is so: a DTD presents only a vocabulary and a *syntax* for that vocabulary — it does not provide a *semantics* for the vocabulary. Which of the ordinary meanings of "title" is meant by "<title>", or even whether "<title>" means anything like what we ordinary mean by "title", is undetermined by the DTD — the DTD only specifies that there is a certain element that has as its label the character string "title", and that it may be used in certain combinations with other

elements — those other elements being similarly defined. So what <title> *means* is often simply inferred — both by content developers using the markup language to represent a document, and by software developers designing software, from the natural-language associations of its name ("title") in a natural language and its use in context. The meaning of "<title>", presumably originally in the head of the language designer, is nowhere expressed in a systematic rigorous way.

Of course this overstates the situation. The meanings of markup constructs may of course be expressed, in a sense, in the natural-language prose documentation that is sometimes provided by the developers of the markup language. But this clearly does not fully solve the problem, even for DTDs documented according to the best industrial and academic practices.

For software to reflect a semantic relationship present in a markup language, that language's designer must indicate the relationship in the documentation; the software engineer must then (seek, find, open, and) study the markup language documentation, and design the application to take advantage of those features. Neither of these two steps is machine verifiable, and neither happens with as much reliability as could be wished. The need for manual interventions is an obstacle to the development of high-performance networked document processing and publishing systems. What is needed is a mechanism that would allow the markup language designer to rigorously and formally specify semantic relationships; these specifications could then be read by processing applications which would configure themselves accordingly, without case by case human intervention.

Let's take a look at some specific examples. These are relationships that are all of at least potential practical importance, but which cannot be easily and systematically exploited as there are no standard machine-processable formalisms for representing them. Many in fact are sufficiently important that software designers are proceeding, in an ad hoc way, to infer their existence from prose documentation and build idiosyncratic systems that make use of them.

Class Relationships SGML/XML contains no general constructs for expressing class hierarchies or class membership among elements, attributes, or attribute values — one of the most fundamental and practical relationships in the most important constructs in contemporary software engineering. There is no way to say, e.g., that a paragraph is a prose-structural element (the **isa** relationship), or that all prose-structural elements are editorial elements (the **ako** relationship). Two native SGML/XML constructs are sometimes used to achieve rudimentary classing along these lines — attribute/value pairs (particularly as used with "type" and "class" attributes) and parameter entity references. These techniques are expressively weak and, perhaps because they re-purpose mechanisms designed with other uses in mind, SGML and XML provide no good mechanisms for controlling them or for constraining their use, although their use in practice does indeed reveal that many document type designers do work with class hierarchies. XML Schema does provide for the explicit declaration of class relations but does not itself specify what the derivation of complex types from other complex types means on the semantic level.

Propagation In many markup languages (e.g. TEI and HTML as of version 4.0), certain properties are to be propagated to all contained elements, or in some cases to textual content. For instance, if an element has the attribute/value notation "`lang="de"`", indicating that the text is in German, then it is implied (absent defeat, see below) that all of its child elements have the property of being in German. But DTDs provide no formal notation for specifying which attributes are thus propagated. Moreover, it is also assumed that such propagation is not monotonic but can sometimes be defeated by an explicit reassignment of the attribute value on a contained element. And there seems to be a variety of different sorts of propagation, some involving properties associated with elements as well as properties associated with attributes, and some having textual content as well as element content as a target. For instance, if the markup indicates that a sentence is in German, that implies that all of the word in that sentence are (unless the attribution is defeated) in German, and similarly all the words in a phrase marked as deleted are deleted, and all the words of a phrase marked as emphasized are emphasized — but it is not the case that marking a component as a paragraph implies that all of its contained words (or elements) are themselves paragraphs. One cannot specify in a DTD which properties propagate, and what the logic of that propagation (including rules for defeat) is, and yet such relationships are regularly inferred (correctly or incorrectly) by software designers developing software for a particular markup language, and then implemented in their tools and applications. [36]).

Context and reference In many markup languages an element, e.g. "<title>", may be used with different meanings, as inferred by context, though with a sufficiently invariant core meaning to justify the use of the same element type. For instance, the use of "<title>" to mark text as a title, but relying on the structural location of "<title>" to indicate its referent. So "<title>" within "<head>" means title of the object "<document>", and "<title>" within "<chapter>" means title of the enclosing "<chapter>". There is no mechanism for saying what the title is a title *of*, that in some cases it is the title of its immediate parent, and in others the title of the root element. A further complication is when a bibliographic entry contains a "<title>" element; here the title in question is the title of a entity external to the text. Again, such relationships cannot be expressed in the DTD, but are inferred by software designers, essential for high-function automated processing of the text. (Only a part of this problem would be solved if different generic identifiers were used for each sense, for it would still be necessary to articulate the dyadic nature of the property expressed, and supply a resolvable deictic expression in order to locate the objects the property applies to.)[36])

Ontological variation in reference A similar but more radical sort of ambiguity occurs when properties are assigned using forms that suggest identical referents, but that require careful interpretation to ensure this identity. For instance, attributes may indicate that a particular element instance is-a-theorem, is-in-German, and is-illegible. But can a straightforward interpretation of these predicates, which would have them all referring to the same thing (the element instance), really be robust enough for knowledge representation purposes? Rather it would seem that the abstract sentences are in German, the propositions they express are theorems, and their rendered concrete expressions are illegible — and that strictly speaking there is no single object has all of these properties.

Full and partial synonymy Full and partial synonymy within and across markup languages is an extremely important semantic relationship, and the lack of mechanisms to characterize

it create serious problems with heterogeneity. And while full synonymy may be eliminable within a single markup language, both full and partial synonymy are difficult and important relations *across* markup languages, particularly now as the number of markup languages rapidly increases. But we currently have no suitable formal machine-processable way to document synonymy of elements, attributes, and attribute values in different languages; architectural forms (see below) can capture many cases of full synonymy, but partial synonymy is more difficult to document, as well as much more common in practice. Partial synonymies of the sort represented by class inclusion relationships could also go a long way to resolving problems of heterogeneity.

4. THE BECHAMEL PROJECT

The BECHAMEL Markup Semantics Project grew out of research initiated by Sperberg-McQueen (W3C/MIT) in the late 1990s [36] 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. [28] The name of the project is an acronym formed from the names of the cities where the cooperating investigators are located (Bergen, Norway; Champaign, Illinois; Española, New Mexico).

The Project now has the following research goals:

1. Characterizing the representation and inference issues 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. Surveying 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 and properties [41] [31].

3. Developing and testing a formal, machine-readable representation scheme in which the semantics of a markup language can be expressed.

4. Exploring applications of the semantics representation, such as transcoding support, information retrieval, accessibility improvement, etc. Our current focus is on the support of semantic inferences from databases of document instances as we believe that this will best stress 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.

Our early Prolog testbed[36] has been extensively developed into a prototype knowledge representation workbench for representing facts and rules of inference about structured documents[6] [38]. The system permits an analyst to specify facts about the markup syntax (e.g., generic identifiers and attribute values) separately from facts and rules of inference about semantic entities and properties. The system provides a level of abstraction at which the meaning of the markup can be explicitly represented in machine-readable and executable form. Inferences can then be drawn regarding document components, including problematic structures, such as those participating in overlapping hierarchies. We have developed a collection of predicates that emulate a subset of the W3C's Document Object Model methods for navigating the hierarchical structure of nodes, and retrieving attribute values and information from the document type definition. These afford a clear separation of the syntactic information captured by the parser and the document semantics expressed by the analyst.

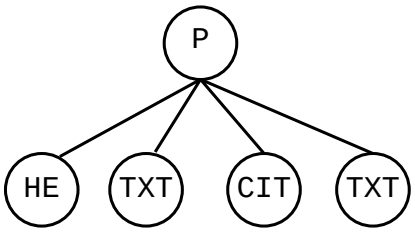

Figure 1: The Syntactic View of Markup

Preliminary findings, reported earlier, revealed the complexity of identifying all licensed inferences [36] [29] and the additional complicating presence of non-assertive illocutionary force [28]. The rudimentary inferencing system demonstrated that automated reasoning about markup was possible, and that complexities such as non-monotonicity and contextual ambiguity could be handled with Prolog rules [37]. Further work is reported in [38] [35].

5. MODELING MARKUP SEMANTICS

The semantics of document markup are the abstract structures, properties and relationships understood by a user of a markup language, as cued by markup and its syntax. Markup semantics are modeled computationally by applying knowledge representation technologies to the problem of making those structures, relationships, and properties explicit.

Consider the following fragment of an XML-tagged document:

```
<P><HE>The Translation Problem</HE>
Translation between different
SGML/XML applications, or
reconciliation of incompatible
document classes is a well-known
challenge <CIT>Fausey and Shafer
(1997)</CIT>. A variety of
techniques are used... </P>
```

It's natural for a reader familiar with structured markup to infer that the document element marked with the **P** tag is a paragraph, that the paragraph has a heading, and that the contents of the paragraph consist of the text that starts after the heading element, and ends at the close paragraph tag. Inasmuch as the meaning or use of a tag may not be immediately obvious, the author/reader can consult prose tag set documentation.

However obvious inferences may be to a human reader, they cannot be drawn from the data structure emerging from a syntactic parse of the fragment. As shown in figure 1, the parse tree (available to, e.g., a stylesheet programmer) represents the heading, citation, and the text before and after the citation each as a separate child node of the paragraph. Nothing in the parse tree represents that fact that that the heading is a *feature* of the paragraph as a whole, or that the text nodes are two parts of the same *content* structure, or that the citation is considered *embedded* in the text.

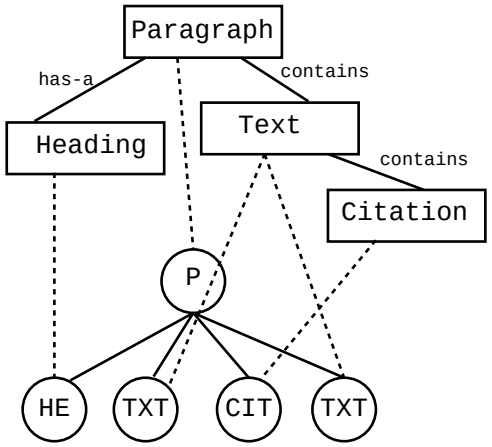

Figure 2: A syntax tree enriched with semantic representation

In fact, the data structure *per se* doesn't have, or say anything about, paragraphs and citations at all. The data structure is only a graph of nodes with associated information, such as a generic identifier having the value "paragraph". The programmer is, of course, expected to draw inferences consistent with the documented meaning and use of the tags, and to use that knowledge, for instance, in constructing the transformation of the tree from one form to another. But that transformation (expressed, for example, in XSLT, DSSSL or a procedural language like C++), *relies* on the semantic inferences without explicitly encoding them.

Figure 2 suggests how the syntactic tree could be enriched or enhanced using semantic knowledge. The whole-part and containment relationships are encoded at a higher level using a knowledge representation technology. The diagram suggests a traditional semantic net representation, but any of a number of alternatives might be employed, including a frame, rule, formal grammar or logic-based representation [31] [41]. Advances in Semantic Web projects (section 8) may even provide suitable expressive power in markup languages themselves. The point is to have a level at which to represent abstractions, relationships, and constraints that cannot be modeled or enforced by a conventional XML/SGML parser. The knowledge is encoded in a machine-readable file which (like a DTD or syntactic schema) can be used to validate a document against semantic constraints and provide a richer document model to an application programmer. These more expressive representations can support the design and implementation of better document processing systems.

6. TARGET APPLICATIONS

Many new technologies have been developed in recent years to augment the usefulness of conventional structured markup. Among the information management problems they aim to address are the following:

Translation and Federation One of the most common projects for an SGML/XML developer is to design a transformation from one application syntax into another [21]. Typically, this is with the aim of creating a new presentation of the document, or preparing it for storage in a database. Sometimes, however, a developer faces the task of federating or reconciling a large collection of digital documents, each tagged according to one of a number of non-interoperable markup languages [3] [32]. Whatever the scope of the translation problem, the conventional solution usually involves applying a transformational programming language that acts directly on the syntactic parse tree [8]. The tree emerging from the parse of the source document is transformed into a tree representing a valid instance of the target language. The transformed tree is then serialized into a new document instance, graphical or audio presentation.

Information Islands This challenge is similar to the translation problem. However, rather than transforming one or more documents from one into another, the goal is to permit documents stored in different parts of a distributed information system to present a common transparent interface to the system's users [9] [13]. It's not necessary to literally transform the documents from one markup language to another, but the system must provide apparently seamless integration of the document contents, notwithstanding any differences in the way the documents are encoded.

Accessibility The growing acceptance of structured markup in authoring tools has been greeted as a boon for promoting the accessibility of digital documents for visually disabled users. Declarative markup allows a person reading a document with the aid of a screen reader or braille display to draw inferences on the basis of mnemonic tag names, rather than graphical formatting cues. But currently the success of such applications depends on the ability of the user and/or interface software to deduce every important structural inference on the basis of the tag names and syntax alone. It also relies on document authors' fidelity in adhering, not only to markup syntax constraints, but also strictly to the meaning and use of the tags, as described in tag set documentation. Regrettably, authors often misuse tags for the purpose of exploiting known presentation side effects; the most notorious examples of this problem are the use of heading markup on web pages for particular kinds of typographic emphasis.

Secure Transactions Part of the impetus driving the development of more expressive markup schema languages (such as the W3C XML Schema language) is the recognition that consequences of tagging errors, misinterpretation, or fraud can be far more drastic than poorly formatted output. Declarative markup is used not only in electronic commerce, but in safety-critical information domains such as medical records [33] and the aircraft industry [7]. Developers in these domains have to ensure not only that digital documents are syntactically well-formed, but that their applications support protocols for ensuring accurate transcription, storage, transmission, and presentation of document contents.

7. BENEFITS OF MARKUP SEMANTICS

Encoded markup semantics, as investigated in the current project, promises contributions to addressing the aforementioned challenges in the following ways:

Declarative, Machine-Readable Semantic Descriptions As it currently stands, the designers of structured markup languages express the meaning and appropriate use of their tags in natural-language text. Formal markup semantics will accord forms in which these relationships can be unambiguously expressed and automatically processed by a computer program.

Validation of Hypotheses In situations where no formal written tag set documentation exists, a system capable of processing

markup semantics declarations can act as an interactive environment for testing conjectures and validating hypotheses. In such situations, the user of an undocumented markup language conjectures a property or rule that he or she believes to be consistently applied in a database of document instances. The document processing software can then retrieve all document elements that are consistent or inconsistent with the hypothesized rule.

Enforcement of Semantic Constraints Just as a semantics-aware parser can test conjectures in a process of discovering or hypothesizing a language's semantics, such a parser can enforce semantic constraints, in addition to the syntactic validation accomplished by a conventional parser. The operation is identical to that of hypothesis validation, but in this situation, the semantic constraints are known and normative.

Richer, More Expressive APIs SGML and XML application programmers employ markup semantics on a daily basis in constructing transformations and presentations of digital documents. But the higher level properties and relationships are only manifest in the execution of the software that they write. Formalized, machine-readable semantics will enrich application programming interfaces, and foster the design of software that is easier and safer to maintain as the markup languages they process evolve and change.

8. RELATED WORK

A number of other document processing technologies, standards, and research projects have responded to the challenges and problems described above. In the following paragraphs we situate the current proposal with respect to these efforts.

The Semantic Web [2] The *Semantic Web* refers to a number of interrelated research and standardization efforts which, like the current proposal, lie at the intersection of markup technologies and knowledge representation. The most central of these enterprises is the World Wide Web Consortium's Resource Description Framework, but other projects are often included, such as the ISO Topic Maps standard [16]. The Semantic Web is a very broad, ambitious effort aiming to equip markup languages with the functionality of general-purpose knowledge representation technologies, and thereby "assist the evolution of human knowledge as a whole" [2]. Semantic Web research and standardization efforts are unlike the current proposal in that they aim not for a semantic description of any *particular* domain, but for the capability of markup to encode semantic knowledge about *every* domain. The current proposal specifically targets the domain of "document markup semantics," not "general semantic markup." But advances in Semantic Web technologies may make it possible for us to encode our markup semantics in a Semantic Web markup language.

W3C Document Object Model The Document Object Model is an application programming interface to the hierarchical data structure that results from parsing an XML document. We aim to design systems that will provide interfaces to markup semantics that are analogous to what the DOM provides with respect to markup syntax, in effect a "semantic DOM" to complement the W3C syntactic DOM.

W3C Schema XML Schema is an XML-based language that provides an alternative to traditional DTDs for expressing constraints on XML documents. The limitations of DTDs that motivated the design of this language overlap with the problems and challenges that our project responds to. Schema allows a document class designer to define elements that respect complex data types, such as are defined in a high-level programming language. However, encoding the full range of relationships and constraints found in prose tag set documentation will require more expressive power than XML Schema currently provides.

HyTime Architectural Forms The *enabling architecture* technology emerged from a recognition that different markup language applications often encode structures that are semantically equivalent, though expressed in very different syntax [15]. Architectural forms allow document class designers to create mappings from their own specific element instances to more general architectural instances that are easier to map between diverse applications [34]. These mappings do represent a limited form of semantic knowledge, and can contribute to solutions to some of the translation and federation challenges described earlier. Our project proposes to model a somewhat broader range of semantic relationships than architectural forms can express.

9. REFERENCES

[1] AAP. *Author's Guide to Electronic Manuscript Preparation and Markup*. Electronic Manuscript Series. Association of American Publishers, Washington, DC, 1986. Current version: ANSI/NISO/ISO 12083 - 1995 Electronic Manuscript Preparation and Markup, National Information Standards Organization, 1995.

[2] BERNERS-LEE, T., HENDLER, J., AND LASSILA, O. The semantic web. *Scientific American 284*, 5 (May 2001), 35–43.

[3] COLE, T., AND KAZMER, M. SGML as a component of the digital library. *Library High Tech 13*, 4 (1995), 75–90.

[4] 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.

[5] DEROSE, S. J., DURAND, D., MYLONAS, E., AND RENEAR, A. H. What is text, really? *Journal of Computing in Higher Education 1*, 2 (1990), 3–26.

[6] DUBIN, D., RENEAR, A., SPERBERG-MCQUEEN, C. M., AND HUITFELDT, C. A logic programming environment for document semantics and inference. Presented at ALLC/ACH, Tübingen, Germany, July 2002.

[7] ENSIGN, C. *SGML: The Billion Dollar Secret*. Prentice Hall, Upper Saddle River, NJ, 1997, ch. 5: United Technologies Sikorsky Aircraft Corporation.

[8] FAUSEY, J., AND SHAFER, K. All my data is in SGML. Now what? *Journal of the American Society for Information Science 48*, 7 (1997), 638–643.

[9] FAY, C. The document management alliance. *Bulletin of the American Society for Information Science 25*, 1 (October/November 1998), 20–24.

[10] GOLDFARB, C. F. *Document Composition Facility: Generalized Markup Language (GML) Users Guide*. IBM General Products Division, 1978. SH20-9160-0.

[11] GOLDFARB, C. F. A generalized approach to document markup. In *Proceedings of the ACM SIGPLAN-SIGOA Symposium on Text Manipulation* (New York, 1981), ACM, pp. 68–73.

[12] IBM CORP. *Application Description, IBM System/360 Document Processing: System*. White Plains, NY, 1967. Form No. H20-0315.

[13] IDE, N. M., AND SPERBERG-MCQUEEN, C. M. Toward a unified docuverse: Standardizing document markup and access without procrustean bargains. In *Proceedings of the 60th Annual Meeting of the American Society for Information Science* (Medford, NJ, 1997), C. Schwartz and M. Rorvig, Eds., Information Today, Inc., pp. 347–360.

[14] ISO. *ISO 8879-1986 (E). Information processing — Text and Office Systems — Standard Generalized Markup Language (SGML)*. International Organization for Standardization, Geneva, 1986.

[15] ISO. *ISO/IEC 10744:1997: Information processing – Hypermedia/Time-based Structuring Language (HyTime)*, second ed. International Organization for Standardization, Geneva, May 1997, appendix A.3 Architectural Form Definition Requirements.

[16] ISO. *ISO/IEC 13250: 2000 Information technology – SGML Applications – Topic Maps*. International Organization for Standardization, Geneva, 2000.

[17] KNUTH, D. E. *TEX and Metafont: New Directions in Typesetting*. Digital Press, Bedford, MA, 1979.

[18] LAMPORT, L. *LATEX – A document preparation system*. Addison-Wesley, Reading, MA, 1985.

[19] LESK, M. E. *Typing Documents on UNIX and GCOS: The -ms Macros for Troff*, 1977.

[20] MAMRAK, S. A., BARNES, J., HONG, H., JOSEPH, C., KAELBLING, M., NICHOLAS, C., O'CONNELL, C., AND SHARE, M. Descriptive markup – the best approach? *Communications of the Association for Computing Machinery 31*, 7 (1988), 810–811.

[21] MAMRAK, S. A., KAELBLING, M. J., NICHOLAS, C. K., AND SHARE, M. A software architecture for supporting the exchange of electronic manuscripts. *Communications of the ACM 30*, 5 (1987), 408–414.

[22] OSSANNA, J. F. NROFF/TROFF user's manual. Tech. Rep. 54, Bell Laboratories, Murray Hill, NJ, October 1976.

[23] RAMALHO, J. C., AND HENRIQUES, P. R. Beyond DTDs: constraining data content. In *Proceedings of SGML/XML Europe 98* (Paris, May 1998), GCA.

[24] RAYMOND, D. R., AND TOMPA, F. W. Markup reconsidered. Technical Report 356, Department of Computer Science, The University of Western Ontario, 1993. Presented at the First International Workshop on the Principles of Document Processing, Washinton DC, October 21-23 1992; an earlier version was circulated privately as "Markup Considered Harmful" in the late 1980s.

[25] RAYMOND, D. R., TOMPA, F. W., AND WOOD, D. From data representation to data model: Meta-semantic issues in the evolution of sgml. *Computer Standards and Interfaces 18*, 1 (January 1996), 25–36.

[26] REID, B. K. *Scribe Introductory User's Manual*, first ed. Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA, August 1978.

[27] REID, B. K. *Scribe: A Document Specification Language and its Compiler*. PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, 1981. Also available as Technical Report CMU-CS-81-100.

[28] RENEAR, A. The descriptive/procedural distinction is flawed. *Markup Languages: Theory and Practice 2*, 4 (2000), 411–420.

[29] RENEAR, A. Raising the bar: Text encoding from a logical point of view. CLIP 2001: Computers, Literature, Philology, Gerhard-Mercator University, Duisburg, Germany, December 2001.

[30] RIZZI, R. Complexity of context-free grammars with exceptions and the inadequacy of grammars as models for XML and SGML. *Markup Languages: Theory and Practice 3*, 1 (2002), 107–116.

[31] ROWE, N. C. *Artificial Intelligence through Prolog*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[32] 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.

[33] SHOBOWALE, G. SGML, XML, and the document-centered approach to electronic medical records. *Bulletin of the American Society for Information Science 25*, 1 (October/November 1998), 7–10.

[34] SIMONS, G. F. Using architectural forms to map TEI data into an object-oriented database. *Computers and the Humanities 33*, 1-2 (1999), 85–101. Originally delivered in 1997 at the TEI 10 conference in Providence, RI.

[35] SPERBERG-MCQUEEN, C. M., DUBIN, D., HUITFELDT, C., AND RENEAR, A. Drawing inferences on the basis of markup. In *Proceedings of Extreme Markup Languages 2002* (Montreal, Canada, August 2002), B. T. Usdin and S. R. Newcomb, Eds.

[36] SPERBERG-MCQUEEN, C. M., HUITFELDT, C., AND RENEAR, A. Meaning and interpretation of markup. *Markup Languages: Theory and Practice 2*, 3 (2000), 215–234.

[37] SPERBERG-MCQUEEN, C. M., HUITFELDT, C., AND RENEAR, A. Practical extraction of meaning from markup. Paper delivered at ACH/ALLC 2001, New York, 2001.

[38] SPERBERG-MCQUEEN, C. M., RENEAR, A., HUITFELDT, C., AND DUBIN, D. Skeletons in the closet: Saying what markup means. Presented at ALLC/ACH, Tübingen, Germany, July 2002.

[39] SPERBERG-MCQUEEN, M., AND BURNARD, L., Eds. *Guidelines for Text Encoding and Interchange (TEI P3)*. ACH/ALLC/ACL Text Encoding Initiative, Chicago, Oxford, 1994.

[40] SPRING, M. B. The origin and use of copymarks in electronic publishing. *Journal of Documentation 45*, 2 (June 1989), 110–123.

[41] TANIMOTO, S. L. *The Elements of Artificial Intelligence*. Computer Science Press, Rockville, MD, 1987.

[42] UNITED STATES DEPARTMENT OF DEFENSE. *MIL-M-28001 Military Specification: Markup Requirements and Generic Style Specification for Electronic Printed Output and Exchange of Text*, 1988.

[43] WELTY, C., AND IDE, N. Using the right tools: Enhancing retrieval from marked-up documents. *Computers and the Humanities 33*, 1-2 (1999), 59–84. Originally delivered in 1997 at the TEI 10 conference in Providence, RI.