Using the Right Tools: Enhancing retrieval from marked-up documents

Christopher Welty Nancy Ide

Vassar College
Computer Science Dept.
Poughkeepsie, NY 12604-0462
{weltyc,ide}@cs.vassar.edu

Abstract

We are experimenting with the representation of a DTD and associated documents (i.e., documents conformant to the DTD) in a knowledge representation (KR) system, in order to provide more sophisticated query and retrieval from TEI documents than current systems provide. We are using CLASSIC, a frame-based representation system developed at AT&T Bell Laboratories. Like many KR systems, CLASSIC enables the definition of structured concepts/frames, their organization into taxonomies, the creation and manipulation of individual instances of such concepts, and inference such as inheritance, relation transitivity, inverses, etc. In addition, CLASSIC provides for the key inferences of subsumption and classification. By representing a document as an individual instance of a hierarchy of concepts derived from the DTD, and by allowing the creation of additional user-defined concepts and relations, sophisticated query and retrieval operations can be performed. This paper describes CLASSIC and the formalism of description logic that underlies it, and demonstrates how it can be used for enhanced retrieval from richly encoded documents.

1. Introduction

The development of the Text Encoding Initiative (TEI) Guidelines enables the encoding of a wide variety of textual phenomena to any desired level of fine-grainedness and complexity, relevant to a broad range of applications and scholarly interests. The ability to encode complex phenomena has, in turn, created a demand for adequate means to manipulate the text once it has been marked up according to the user's interests and needs. One obvious and immediate need for users of the TEI scheme is a flexible means to query and retrieve from an encoded text, which does not require deep knowledge of the structure of the text by the user. There has been some work in this area (see, for example, Blake, et al. [1997], and Harié et al. [1996]), although so far most systems require that the user know the specific structure of the document as defined by the Document Type Definition (DTD).

Beyond the need to query and retrieve based on tags that exist in a TEI document, a means to manipulate and query classes of objects is also desirable. The TEI DTD uses SGML entity definitions to create "classes" of elements and attributes, in particular, for groups of elements with common structural properties (e.g., all elements that can appear between paragraphs), g attributes which apply to certain classes of elements (e.g., attributes for pointer elements), etc. In addition to grouping together elements and attributes with common structural properties, the definition of such classes recognizes common semantic properties among elements and attributes. However, the SGML entity definition mechanism is designed primarily for string substitution within the DTD itself, thereby enabling easy reference to these classes in later element definitions; the common semantic properties that are implicit in the classification scheme are lost for the purposes of retrieval and document manipulation. Obviously, a means to refer to and manipulate classes of elements and attributes in a query and retrieval system would provide substantial additional power for the user.

We are experimenting with the representation of a DTD and associated documents (i.e., documents conformant to the DTD) in a knowledge representation (KR) system, in order to provide more sophisticated query and retrieval from TEI documents than current systems provide. We are using CLASSIC, a frame-based representation system developed at AT&T Bell Laboratories (Brachman et al. [1989]). Like many KR systems, CLASSIC enables the definition of structured concepts/frames, their organization into taxonomies, the creation and manipulation of individual instances of such concepts, and inference such as inheritance, relation transitivity, inverses, etc. In addition, CLASSIC provides for the key inferences of subsumption and classification (Brachman [1983]). By representing a document as an individual instance of a hierarchy of concepts derived from the DTD, and by allowing the creation of additional user-defined concepts and relations, sophisticated query and retrieval operations can be performed.

In particular, we are exploring the use of KR techniques to enable the following:

- *Classification of elements:* in many cases, users want to manipulate groups of objects that are seen as belonging to a single general class. For example, the tags <author> and <name> may both be used to mark names of people, and these two types of elements can be seen as members of a general class of PEOPLE. Or, for linguistic analysis, all words, names, dates, etc. may need to be regarded, for certain purposes, as falling into the class of linguistic TOKENS. SGML provides no way to classify groups of elements and therefore no way to query and manipulate such groups as a whole.
- *Recognition of context-sensitive relationships:* SGML provides no scoping mechanisms, and therefore the definition of an element applies across the entire document. There is no means, for instance, to define an element NAME which, if it appears in the heading of the text, must include tags for FIRST NAME, MIDDLE INITIAL, and LAST NAME, and another element NAME which appears in the body of the text and may a different set of elements. SGML allows for specifying that each of these content models is an alternative, but cannot enforce or prevent the use of one of them in a given context.
- *Support for multiple views:* as digital libraries become more accessible, they must be capable of handling diverse requests from the variety of potential users. Any given text may be "viewed" from multiple perspectives, depending on the intended use: a text can be seen as a physical object, a logical object, a rhetorical object, a linguistic object; it may be seen as a historical database of information or a work of literature. In each of these views, the same elements may be considered as very different objects with correspondingly different relationships to other elements. For example, for the linguist, names may be linguistic tokens or proper nouns, which stand in relation to other syntactic elements and structures; he or she might make a request such as, "for every sentence in the text where a proper noun is the syntactic subject of the sentence, show me the corresponding sentence in the Romanian translation". On the other hand, for the historian names may be important insofar they relate to dates, places, etc. that also appear in the text, and he or she might ask for all the names mentioned in letters (but not the author or recipient) sent from Philadelphia on July 4, 1776. To answer each of these requests, an entirely different view, comprising an entirely different set of relationships among elements, attributes of elements, etc., must exist. SGML does not conveniently support multiple views, and in some cases cannot represent them at all.

The definition of multiple sets of complex inter-relationships and properties is easily handled using techniques employed in the field of knowledge representation. Therefore, we propose "using the right tools" for the different kinds of tasks involved in providing access to complex documents: a comprehensive SGML encoding scheme for specifying tag syntax, and powerful KR techniques to enhance and extend retrieval capabilities.

Our underlying premise is that we are developing technology for a digital library whose contents are fully marked up texts. The foundations of the technology we are developing are in two areas: text-encoding, in particular the TEI and Corpus Encoding Standard, and knowledge representation, in particular description logics and formal ontology. We begin with some

background information on these two fields and the relevant sub-fields, and then describe the research itself.

2. Text Encoding

In general, text-encoding is the practice of marking up text with tags that indicate a section of text should be interpreted or rendered in a particular way. The Standard Generalized Markup Language (SGML), which provides a meta-language for developing specific tag sets, has rapidly become the basis of most markup schemes intended for general use. The best known SGML-based encoding scheme is HTML, which provides a tag set suitable for document display. However, HTML is neither an adequate nor a pure *descriptive* markup scheme, in which text elements are consistently marked according to their role or function rather than the way in which they should be rendered. For intelligent retrieval and manipulation of complex text objects, descriptive markup is essential.

The Text Encoding Initiative (TEI) has developed an SGML-based descriptive markup scheme, which provides an extensive set of tags for marking a wide range of textual phenomena. We have developed a TEI-compliant encoding scheme, the Corpus Encoding Specification (CES) (Ide [1998a,b]) for encoding linguistic corpora, which both constrains and extends the TEI scheme to suit our particular needs. In particular, we constrain the TEI in the following ways:

- selection of only those tags relevant to our application;
- specification of a precise semantics for tag use, that is, specifications of exactly what the contents of each tag, in terms of both its form and linguistic function, must be;
- closed lists of attribute values;
- tighter constraints on allowed tag syntax (i.e., embedding, tag content, etc.)

These constraints were imposed after consideration of several factors, including the processing needs of our tools, the need for incremental addition of markup and to facilitate automatic tagging (especially in the early stages of text preparation), to limit to the extent possible the types of content that may appear within particular tags, and to enable tighter validation for correct markup syntax.

We extend the TEI as follows:

- addition of elements not provided by the TEI, or elaboration of tag content to suit our needs;
- definition of a data architecture which provides for a hyper-linked set of SGML documents representing the original text and different types of linguistic annotation (e.g., part of speech tags, alignment information, etc.). Thus each of the hyper-linked documents represents a different *view* of the same text.

The reader is referred to http://www.cs.vassar.edu/CES/ for full documentation of the CES scheme. The CES is well-suited to our purposes here because of its data architecture, and because it provides a tight DTD for a smaller set of elements and is therefore useful for testing purposes. However, our methodology should scale up to handle the full TEI DTD. Note that the CES is currently being adapted to the Extensible Markup Language (XML), which is likely to replace SGML as the meta-language of choice.

3. Knowledge Representation

Knowledge Representation (KR) is a wide and varied field, which focuses almost exclusively on semantics and the ways in which symbolic systems can convey it. It is important to realize, however, that we are proposing KR as a natural *complement* to the syntax-based systems currently in place for supporting markup. The strengths of SGML, such as support for structural specifications, naturally complement the weaknesses of KR techniques and systems,

and vice-versa.

3.1 Relational Representation Systems

Our approach is centered on an in-depth and disciplined study of the domain and a realization of that domain in a *relational* system. While we provide capabilities beyond this basic technology, it is important to realize that a large number of the benefits we describe here come simply from the relational paradigm.

A relational system has three principle elements: *classes*, *objects*, and *relations.* Any of the wide variety of systems such as Relational and Object-Oriented Databases, Semantic Networks, and Frame Systems, fall into this category. A class (a.k.a. *concept* or *schema*) is *terminological*, providing merely a description of what the actual data may look like: i.e., its attributes and type information. Objects are the data itself, which are typed according to class. Relations are the links that specify attributes and join the objects.

For example, we could define a single class, *PERSON*, and three relations, *NAME*, *GENDER*, and *SPOUSE*, and define the class *PERSON* as follows:

```
{PERSON::
  NAME: [1] String
  GENDRE: [1] Male, Female
  SPOUSE: [1] Person
}
```

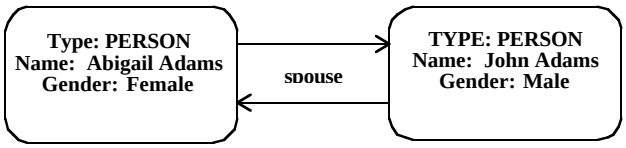
This should be interpreted as "A person has one name, which is a string, one gender, which is either male or female, and one spouse, which is another person." When the classes have been set up that describe all the possible types of data we wish to store, we populate the system with objects. Two objects representing a husband and wife would look something like:

```
{OBJECT-9876::
  Type: PERSON
  Name: "Abigail Adams"
  Gender: Female
  Spouse: OBJECT-7654
}

{OBJECT-7654::
  Type: PERSON
  Name: "John Adams"
  Gender: Male
  Spouse: OBJECT-9876
}
```

As simple as this example may be, it illustrates several important points. First of all, object names (OBJECT-9876 and OBJECT-7654) are simply unique symbols used as placeholders to allow cross-referencing the objects. One might expect, for example, the spouse relation of the object representing Abigail Adams to have a value "John Adams." Instead, we put in the object name, which can be effectively interpreted as "the object of type person whose name is John Adams."

Figure 1. Two simple objects.



The second point is that data like this is typically more conveniently illustrated in diagrams that convey the relational aspect of the representation, as shown in Figure 1. One advantage of these pictures is there is no need for the object-names as placeholders.

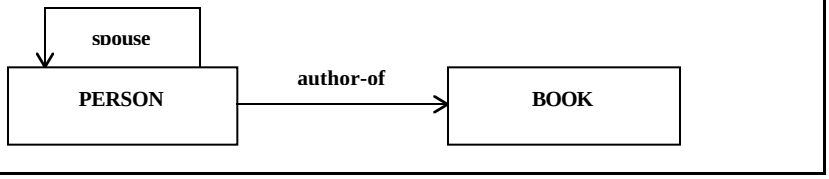
The final point is not immediately obvious, but requires consideration of existing library information systems, in which there is only one class of object, a publication, available to the general user. Although modern card catalog information systems allow for nested attributes (attributes of attributes) that lead to slightly more expressive queries, the only type of thing that can be retrieved in a search is a publication. In web-based search, in which statistical methods increase the potential efficacy of keyword-based queries, the only result of a search is documents. We believe this to be a critical flaw as libraries become digital and provide access not just to the traditional card catalog type information, *but to the contents of the texts* themselves. We are proposing expanding library information systems to include knowledge of authors, their institutions, etc., as well as information relating to the contents of documents such as character names, places, etc., and even such structures as paragraphs, or nouns and verbs, such that these objects can be the potential results of a search. In later sections we give more specific examples of this notion that motivate it better.

In order to achieve the proposed expansion of library information systems, extensive ontological analysis is required.

3.2 Formal Ontology

"Ontology" is one of those unfortunate words whose usage has become somewhat disassociated with its original meaning by those who use it carelessly. In philosophy, ontology refers to the study of the state of being and the identification of the kinds and nature of things that exist. This rather vague notion has been captured by the knowledge representation community and slightly adapted to mean "describing what things are." A *formal* ontology is a complete symbolic description of what things in some domain *can be.* This description is specified in a formal notation, such as a logic or calculus, in which all the possibilities are either enumerated or stated inductively. In other words, the ontology is the definition of the classes, relations, constraints, and rules of inference a knowledge-based system will use.

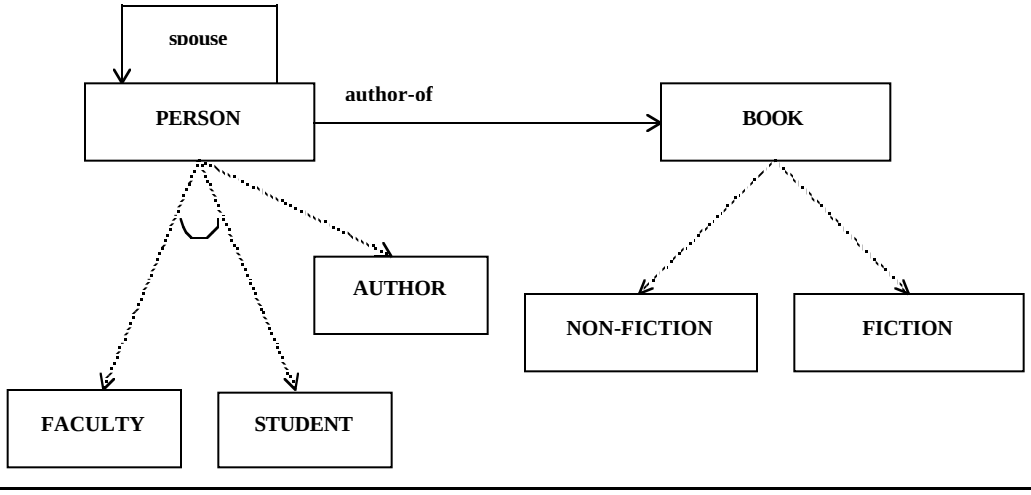
Figure 2 A simple ontology.



The process of analyzing a domain in order to formally specify an ontology requires several steps, one of the most important of which is identifying the object types, specifying their attributes, and specifying the kinds of relationships they can have with each other. When the specification language of an ontology is relational, this step can frequently be accomplished using a graphical notation; however, it is important to distinguish this type of *terminological*

graph, such as the one shown in Figure 2, from the type of *assertional* graph shown in Figure 1. An assertional graph denotes actual data and links; a terminological graph specifies no actual data, only *what the data can be*. In Figure 2 we have stated that there can be two types of objects, *PERSON* and *BOOK*, and that an instance of a person can have a *SPOUSE* relationship with another instance of *PERSON*, and any number of *AUTHOR-OF* relationships with instances of *BOOK*. In this terminological picture, or ontology, we refer to no actual people or books, merely describe (and constrain) how that kind of data can be specified.

Figure 3 An ontology with taxonomic relationships.



Another important step in ontological analysis that is supported by most formal languages (although not by simple relational databases is the identification of taxonomic relationships. Taxonomic structure is rapidly growing in importance as a mechanism for narrowing search spaces (Welty [1998]). We could easily imagine our ontology from Figure 2 being expanded by providing two *disjoint* subclasses of person, *FACULTY* and *STUDENT*, and another non-disjoint subclass of person *AUTHOR*. object could be an instance of *FACULTY*, or *STUDENT* (but not both at the same time), as well as an instance of *AUTHOR*. We have also added the subclasses *FICTION* and *NON-FICTION* for books. The extended ontology is shown in Figure 3.

The taxonomic relationship is typically unique in ontological analysis in that it is the only purely terminological relationship (with the notable exception of the part/whole relationship, which is still not fully understood; see Artale et al. [1996]). In other words, taxonomic relationships exist between classes, not between the instances.

3.3 Description Logics

Description logics are the class of representation languages that derive from KL-ONE (Brachman and Schmolze [1985]), a knowledge representation language that first tried to formalize the notion of a *frame*, as described in Minsky [1981]. Description logics are for the most part *less* expressive than First-Order Logic (Borgida [1998]), with a syntax that enables rules of inference that are sound, complete, decidable, and allows for the taxonomy to be exploited for tractability.[1] While the types of these languages vary, attempts have been made to formalize a core functionality that characterizes them (Patel-Schneider and Swartout [1993]), and

[1] A reasoning system is *sound* if no unprovable inferences can be made, *complete* if anything that can be proved is inferred, and *decidable* if a procedure exists that will make all inferences in finite time. *Tractability* refers to the time it takes to compute the inferences, where more tractable implies less time.

they all share two basic features: subsumption and terminological reasoning (explained below). We use the CLASSIC system (Brachman, et al. [1989]), a description logic developed at AT&T Bell Laboratories (now AT&T Labs – Research).

Description logics have three basic syntactic elements: *concepts*, *roles*, and *individuals*, which correspond to classes, relations, and instances (or data objects). The language of description logics is centered on specifying information in such a way that it is possible to automatically determine when one concept *subsumes* another. For example, the set of objects that are blue or green subsumes the set of objects that are blue.

While many languages also provide for representing classes of objects, description logics actually allow reasoning and expression using them. In most systems, most notably database or object-oriented systems, classes are static, membership in a class is stated explicitly, and all queries and any reasoning occur at the assertional level (on the instances). Description logics, on the other hand, do allow for reasoning and queries at the terminological level (on the classes). This distinction is at once subtle and crucial.

The simplest case of terminological reasoning is computing subsumption relationships between concepts. For example, imagine that there are three concepts defined as follows:

```
(define-concept book)
(define-concept nonfiction-book book)
(define-concept biography nonfiction-book)
```

This sets up a taxonomy of types of books. We can now define several more concepts:

```
(define-concept person)
(define-concept author
   (AND person
        (ALL author-of book)
        (AT-LEAST 1 author-of)))
```

This defines an author as a PERSON with at least one value in its AUTHOR-OF role, and all those values must be individuals of BOOK. In other words, an AUTHOR is a PERSON who is the author-of at least one BOOK. Now we can define another concept:

```
(define-concept nonfiction-author
   (AND person
        (ALL author-of nonfiction-book)
        (AT-LEAST 1 author-of)))
```

CLASSIC is capable of computing that AUTHOR subsumes NONFICTION-AUTHOR, since it can be proven that all individuals in any interpretation that satisfy the definition of NONFICTION-AUTHOR will also satisfy the definition of AUTHOR. Note that while this may be intuitively obvious, in the definition above it is not explicit: NONFICTION-AUTHOR is defined to be a sub-concept of PERSON, not a sub-concept of AUTHOR. Only computational systems capable of reasoning at the terminological level would be capable of recognizing the implicit subsumption between these two concepts. The important point here is that the reasoning is being performed over the *concepts* (or classes), and in this case the taxonomy is being automatically restructured. At this time, only description logics are capable of this reasoning, and the importance of this capability is that it allows us to exploit the taxonomy for increased efficiency.

This was only a simple example and only one use of terminological reasoning. In the next section we discuss a more complex example that demonstrates other uses of terminological reasoning for semi-structured and incomplete data, which is a common problem in the representation of old documents.

4. Semi-structured and Incomplete Data

We have been working with several groups involved in large, ongoing encoding efforts. The Brown Women Writers Project (Flanders, 1998) and the Model Editions Partnership (Chesnutt 1995) are two such efforts focused on encoding of humanities data, including manuscripts, letters, diaries, and other document sources. Many of the sources are more than 100 years old, and others date back to the sixteenth century. A large proportion of these materials exists in manuscript (i.e., hand-written) form only.

These documents present difficulties for most representation systems because the data are *incomplete* and *semi-structured*. Because of the condition of some of the documents (e.g., manuscript stains), or due to illegibility, some of the data are incomplete. Semi-structured data, which does not consistently adhere to a preconceived template, is also characteristic of these data: for example, some diary entries include dates while others do not, letters may or may not include recipient addresses (possibly depending upon whether or not they were hand-delivered), etc. Such omissions could be regarded as cases of incomplete data: e.g., a diary entry can be assumed to have a date, which is missing in some of the entries. However, unlike the database community, we distinguish semi-structured and incomplete data: incomplete data is data that was there or should be there(e.g., *every* letter has a recipient, although in some cases the recipient may not be discernible), whereas a letter in which no recipient address is specified is semi-structured, since the address is not a necessary part of it.

4.1 Incomplete Data

Knowledge Representation and Reasoning has much to offer in domains in which data are often incomplete and/or semi-structured. The purpose of this section is to provide a realistic yet simple example of how the representation relates to the markup, and how our research can exploit the power of KR tools to deal with this rich and unique kind of data.

Let us suppose that our library has a special collection of original letters from a well-known person, Abigail Adams. These letters are being entered into the library in electronic form and marked up. Part of the DTD for marking up these letters includes, among others, the tags *SENDER, RECIPIENT, RECIPIENT-ADDRESS,* and *SENDER-ADDRESS.*

One could well imagine the following to be an excerpt from a marked up letter:

```
<head>
<sender>Abigail Adams</sender>
<sender-address>Boston, MA</sender-address>
<recipient>John Adams</recipient>
<recipient-address>Philadelphia, PA</recipient-address>
</head>
```

The goal of this markup, again, is to enable retrieval of information in a far more robust way than was previously possible. This idea goes well beyond keyword searches or substring matches, because the presence of the tags provides additional semantic information. This is not just a document with the string "Abigail Adams" in it; it is a letter from Abigail Adams, distinct from a letter *to* Abigail Adams or a letter that *mentions* Abigail Adams. Given the wealth of information fully digital libraries will contain in the future, such differentiations could mean the difference between a query returning a few items and a query returning thousands of items.

One can imagine that some of the marked up information from digital texts in general is being automatically extracted and entered into a card catalog database, in order to facilitate access to the documents. For this to happen, classes (or database schemas) need to be created that correspond to the object types that will be extracted. These objects correspond to marked-up elements within electronic texts, so that the data extracted from the marked-up excerpt above and entered into the database would be:

```
{OBJECT-0213::
  Type: Letter
  Sender: OBJECT-9876
  Recipient: OBJECT-7654
}

{OBJECT-9876::
  Type: Person
  Name: "Abigail Adams"
  Address: "Boston, MA"
}

{OBJECT-7654::
  Type: Person
  Name: "John Adams"
  Address: "Philadelphia, PA"
}
```

We will, for the moment, ignore how the extraction process determines the correspondence between a person's name in the marked up text and an object whose name slot has that value in the database.

Once the data have been extracted, a query such as "FIND all letters from Abigail Adams to John Adams" is possible. Clearly, such a query could not be expressed in any current card catalog systems or using keyword other indexing approaches.

We are now ready to consider how terminological representation comes into the play. The examples presented thus far have been entirely assertional. The terminological part of the representation in a database comprises the class or schema definitions, which are static and used only as type checking mechanisms. Consider, however, the case of another letter in the collection that has been damaged over time, resulting in the loss of the recipient's name, although the recipient's address is still intact. The markup might be as follows:

```
<head>
<sender>Abigail Adams</sender>
<sender-address>Boston, MA</sender-address>
<recipient-address>Philadelphia, PA</recipient-address>
</head>
```

Outside of description logics, there is no way to represent the fact that this letter is from Abigail Adams and to "someone in Philadelphia," without actually creating a new object as a sort of place holder for that person. In other words, the closest database representation would be:

```
{OBJECT-0214::
  Type: Letter
  Sender: OBJECT-9876
  Recipient: OBJECT-7655
}

{OBJECT-7655::
  Type: Person
  Name: "Unknown"
  Address: "Philadelphia, PA"
}
```

We have created a new, "dummy" Person object (OBJECT-7655) whose address is Philadelphia. The problem here is that the existence of this unknown person object implies that the recipient of the letter is *not* John Adams, because John Adams is represented by another object (OBJECT-7654), and this is not necessarily the case. The recipient may not be John Adams, but then again it may be – we just don't know. All we do know is that the recipient is in Philadelphia.

In a description logic, we can allow for terminological descriptions of attributes without giving them concrete assertional values. The four objects in question would be represented as follows:

```
(define-individual OBJECT-0213
  (AND Letter
       (FILLS Sender OBJECT-9876)
       (FILLS Recipient OBJECT-7654)))


(define-individual OBJECT-9876
  (AND Person
       (FILLS Name "Abigail Adams")
       (FILLS Address "Boston, MA")))


(define-individual OBJECT-7654
  (AND Person
       (FILLS Name "John Adams")
       (FILLS Address "Philadelphia, PA")))


(define-individual OBJECT-0214
  (AND Letter
       (FILLS Sender OBJECT-9876)
       (ALL Recipient (AND Person
                           (FILLS Address "Philadelphia, PA")))))
```

The first three objects (or *individuals* in a description logic) are precisely the same as in a database approach, but the final individual (OBJECT-0214) is different, and there is no dummy object. Instead, the recipient of the letter is described using a new class, "a person whose address is Philadelphia." This is a *class* , not a new individual; in a description logic a class (or *concept*) is a description of a set of individuals, just as a schema is in a database. By using a concept that describes all the possible values for the recipient, we convey all the information without excluding the possibility that John Adams is the recipient, since OBJECT-7654 matches that description.

The important point in this example is that by creating this new, more specific class, we enable the Abigail Adams letter to be retrieved by a query, "find all letters sent to people in Philadelphia". However, because there is no value in the recipient slot of the object, we have constrained it further. This does not mean that all objects that match the constraint are considered values of the recipient slot; it simply means that the recipient of the letter is a person who lives in Philadelphia. The slot itself is not filled, and therefore the letter would *not* be retrieved by the query, "find all letters to John Adams."

4.2 Semi-structured Data

Description logics are also well suited for handling semi-structured data. The main obstacle for most systems in dealing with semi-structured data is that there are no precise schemas. In a database (and, for that matter, in an object oriented design), the main purpose of a schema is to define which attributes a particular data type can have. A problem that arises with semi-structured data is that instances of the same type may have different attributes.

In a description logic, all attributes (or roles) are actually global. Any piece of data can have any of the roles that have been previously defined. The concepts (which, again, correspond to schemas) do not define the exclusive list of roles for their individuals. In other words, given the concept below:

```
(define-concept letter
   (AND document
        (ALL sender person)
        (AT-LEAST 1 sender)
        (ALL recipient person)))
```

which says that, for all individuals of LETTER there must be at least one value in the sender role, all the fillers must be individuals of PERSON, and that all the fillers for the recipient role must be individuals of PERSON as well. It does *not* say that individuals of LETTER have any restrictions on any other roles. An individual of LETTER could have a NAME role (if such a role has been defined), or fillers for any other defined role.

It is also possible to restrict individuals of certain concepts from having fillers for particular roles. For example, we might define the concept UNSENT LETTER as a letter without a recipient:

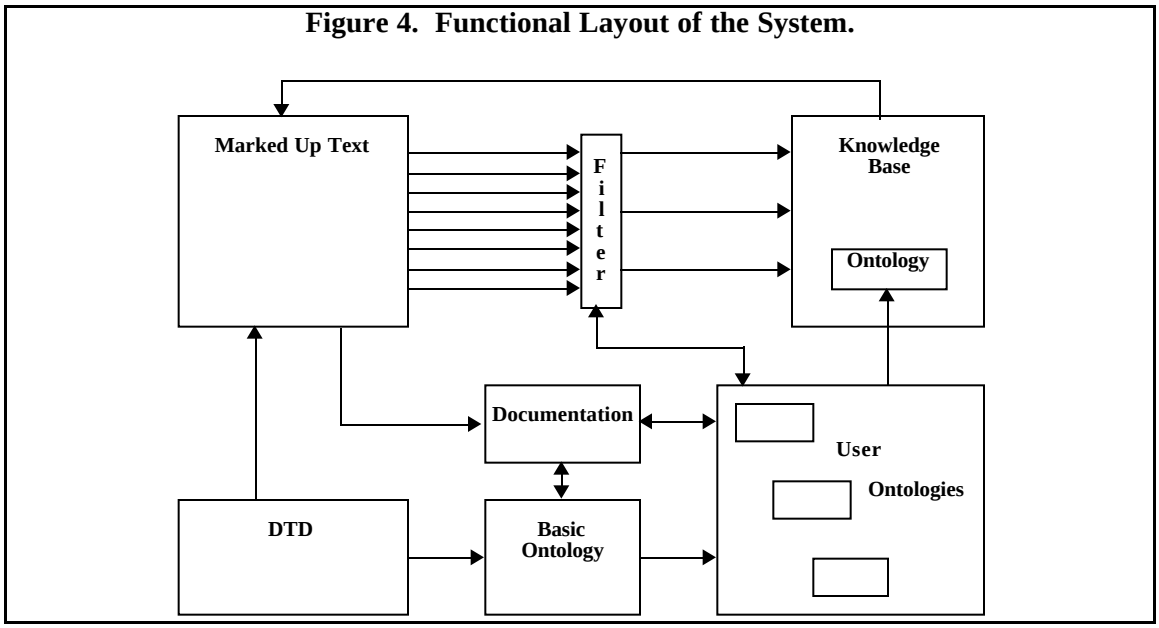
```
(define-concept unsent-letter
   (AND letter
        (at-most 0 recipient)))
```

In general, a concept describes both the necessary and sufficient conditions for membership. This is the main reason description logics handle semi-structured data so easily: most other representation systems provide only for the specification of necessary conditions. With necessary conditions, once you know a piece of data is an instance of a specific type, that data **must** obey the restrictions described by that type. Conversely, with sufficient conditions, once a piece of data obeys the restrictions described by a specific type, it is an instance of that type.

The power of description logics lies in the expressiveness of terminological representations. Again, while several description logic systems exist, we have been using the CLASSIC description logic (Brachman, et al. [1989]). CLASSIC provides a subset of the full description logic specification described in Patel-Schneider and Swartout [1993], lacking features such as disjunction in concept descriptions, role instantiation, and SAME-AS on general roles. These features have been excluded intentionally to provide speed and tractability, as well as manageable memory requirements.

5. System Overview

Figure 4. Functional Layout of the System.



Our system has several major components, as shown in Figure 4. The arrows show dependencies or flow of information (or both). In some cases the information flow is formal (i.e. automatic), and in other cases informal (human generated). The figure itself does not distinguish these types; they are defined below. All the examples used in this section are based on work we are doing using the CES DTD.

5.1 DTD and Basic Ontology

All efforts begin with the DTD itself, which in many cases is already in place. In general there should be a separate ontology effort for each DTD, although the obvious overlaps between DTDs should result in significant re-use of ontology components. From the DTD, the basic ontology is generated automatically, which is similar in concept to generating database object models from DTDs, as in Simons (1997). The basic ontology simply consists of each element in the DTD, and the taxonomy will be generated based on the use of entities and some TYPE attributes. For example, the CES DTD includes a simple taxonomy of paragraph-like elements that are specified using entities. These entities become concepts in the ontology that subsume all the elements contained in the entity definition. The NAME element in the CES DTD is used consistently with the TYPE attribute to indicate a one-level deep taxonomy that includes place names, person names, organization names, etc.

The basic ontology is then augmented manually to reflect any other formal semantics that apply generally. For example, in our basic ontology, there is a concept DATE that corresponds to the DATE element in the CES DTD. We have found it useful to distinguish between a date in the header of a document and a date in the body. Since each of these has identical syntactic structure, there has traditionally been no reason to create more than one tag; however, semantically there is a difference between a date used in the header (which will contain meta-data about the document or the markup), and a date that appears as part of the document's marked-up text. In order to create these two semantic categories, we have included general concepts for header elements and body elements, and then two concepts, MARKUP-DATE and CONTENT-DATE. The former is a concept subsumed by both HEADER-ELEMENT and DATE, and the latter by BODY-ELEMENT and DATE. These new concepts can be thought of as "virtual tags," since they don't appear explicitly in the document but can still be retrieved.

Although the addition of these concepts to the ontology is done manually, classification of

specific marked-up regions of text is done automatically via subsumption reasoning. This is an important point, because we feel that encoders will not spare the time to go back through their documents and add new tags that match new elements specified in the ontology. Adding concepts to the ontology is, on the other hand, fairly easy to do. The resulting ability to apply these new concepts to all documents that use the DTD makes the effort involved in using our approach easily worth the gains of increased accessibility.

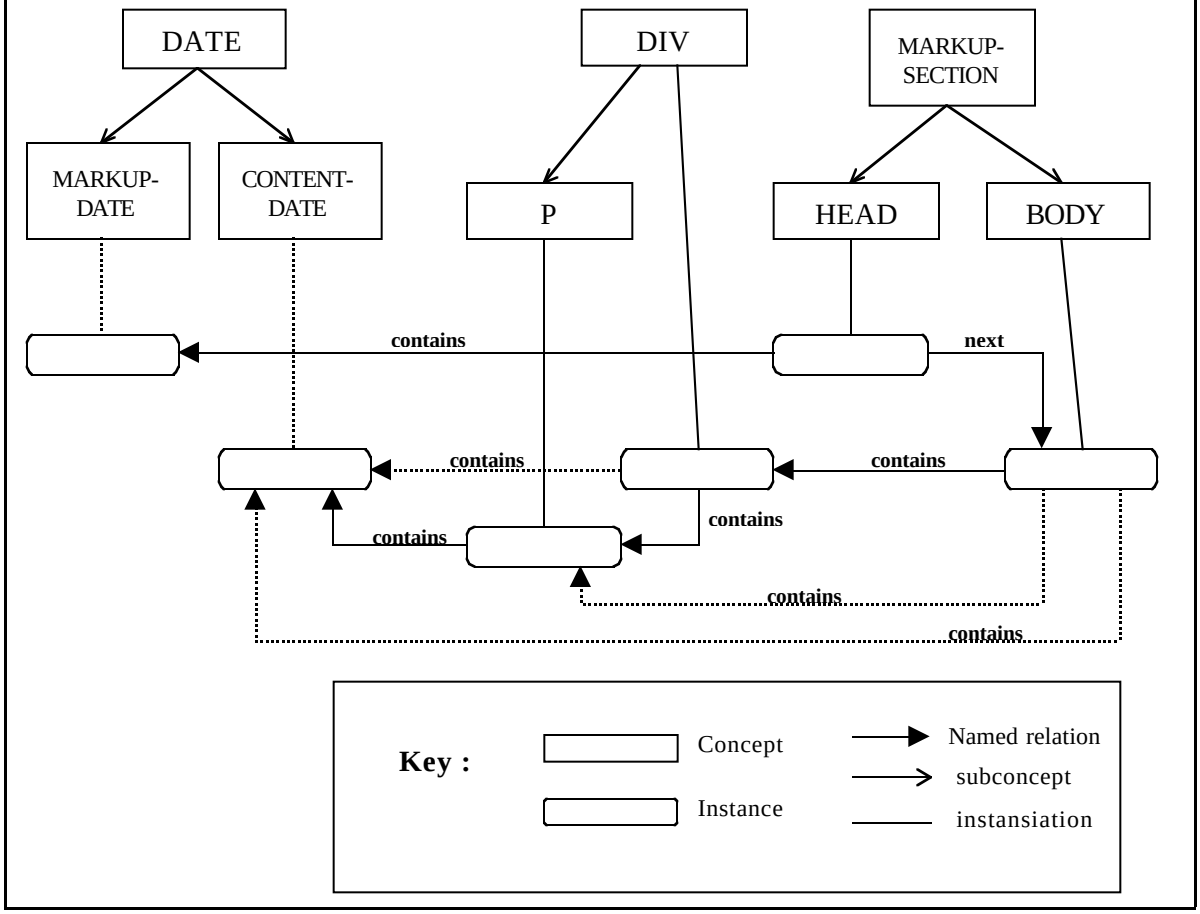
The automatic recognition of, e.g., occurrences of the virtual CONTENT-DATE tag proceeds as follows: the actual tags are extracted from a marked-up document and used to populate a large knowledge base. Each occurrence of a tag in the document corresponds to an instance in the knowledge base, and structure-preserving relations between these instances maintain the parse tree of the document. In particular, the relation *contains* is used between an element that contains other elements and those elements, and this relationship is defined to be transitive, i.e. if A *contains* B and B *contains* C, then A *contains* C. Therefore a text marked up roughly as follows:

```
<head> … <date>1/1/1998</date> … </head>
<body> … <div> … <p> … <date>7/4/1776</date> … </p> … </div> … </body>
```

will lead to the creation of the instances shown in Figure 5. The dashed links show the information that is not explicitly in the markup, i.e. that has been added via inference. Note that the two date elements from the text are each classified appropriately.

5.2 User Specific Ontologies

Figure 5. Automatic Classification of Dates



Another important aspect of our work is the notion that different users will require different *views* of the data. We support different views by allowing users to specify their own ontologies, or (more likely) choose from a set of pre-defined ontologies that best suits their retrieval needs.

The goal of multiple views is to support a wide range of different users. Scholars in different fields, e.g. linguists and historians, may have different semantics, and thus different uses, for the same tags. In addition, different views give our system the ability to handle large amounts of data more efficiently, and in some cases, enable handling this data at all.

For example, for a scholar doing linguistic analysis, a name's primary relevance may be that it is a proper noun, and so the concept NAME appears in a taxonomy of concepts below NOUN. To a historian, the name itself is not as important as the person it denotes, and so NAME would be a concept whose instances could fill the "name" role of an individual of the concept PERSON. In this example, the difference between the two meanings of the same tag are not mutually exclusive, and both can be retained if desired; the only reason to exclude one would be to eliminate unnecessary information and thus reduce the size of the knowledge-base being searched. However, it is possible for tag semantics to conflict across views, thus necessitating the exclusion of one when the other is present. Description Logics are also useful for detecting these sorts of inconsistencies, although we have not come across specific examples so far in practice.

The most significant contribution of the approach we propose here is the ontologies themselves and the ontological analysis required to build them. More specific examples of these benefits are outlined in Section 6.

5.3 Filters

A user ontology defines a view of the data that is manifested as a filter. The filter serves to keep the size of the knowledge base limited to the data that is relevant to the needs of the user. Returning to our linguist vs. historian example, for a linguist the names, places, and events included in the documents may not be relevant information, and therefore they can be excluded from the knowledge base. On the other hand, for a historian the fact that a word is a noun, verb, or other part of speech may not be relevant.

A filter component enables eliminating certain pieces of information from the knowledge base, thus enabling a specific view of the data, and at the same time reducing its size. Size is a serious issue at the moment, in particular for description logics (which are primary storage-based) in comparison with database systems (which are secondary storage-based). We believe this problem to be ephemeral, however. Hardware researchers are already testing 64GB flash memory cards, and experts predict that within the next ten years disk drives will be relegated to archival backups (Newton [1997]).

Filters also help focus the information being searched so that the scale for human users is reduced as well. The less information being searched, the less likely that irrelevant information will be found. Furthermore, tailoring the information in the knowledge base to conform to the user's own view of its contents and avoiding overwhelming the user with large amounts of extraneous information significantly enhances usability.

5.4 The Knowledge-Base and Marked Up Text

Another important goal of this research has been to avoid, as much as possible, changes to the way text encoders currently mark up documents. Our knowledge-based approach is designed to naturally *complement* the efforts that are already on-going.

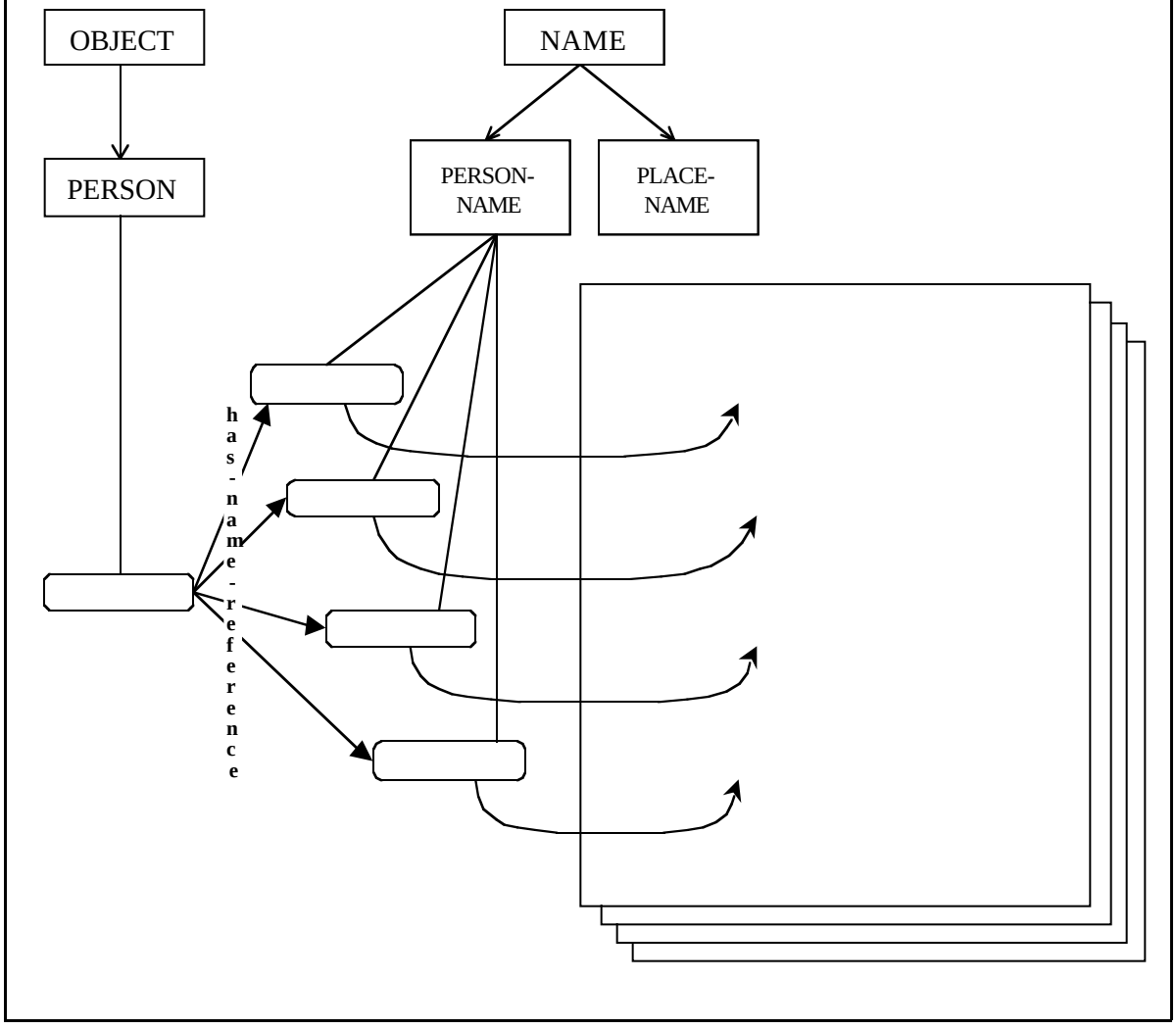
As mentioned above, The knowledge-base is generated automatically from marked-up texts by applying a set of filters. The marked-up text is scanned and for each tag, and if its type appears in the selected ontology (filter), an instance is created in the knowledge-base along with associated instances its existence implies. This latter point is important because it is the key to some of the benefits we discuss in the next section

In addition to representing the individual tags in a marked-up text, we also represent the background knowledge that ties the *meaning* of these tags together. For example, consider a text in which the following tag appears:

```
<name type=person>George Washington</name>
```

A base ontology could be generated from the DTD that includes a NAME concept with a sub-concept called PERSON-NAME. In addition, a user-specific ontology could be added that specifies a relationship between instances of PERSON-NAME and instances of PERSON. That is, a rule can be included a rule that says, "For every instance of PERSON-NAME, find an instance of PERSON who has that name. If no such PERSON instance is found, create one. Then create a relationship between this instance of PERSON-NAME (the tag) and the instance of PERSON." In fact, the description logic specification of this rule is far more concise than the English. An illustration of the process is given in Figure 6.

Figure 6. Automatic creation of background information.



The searching power apparent in this simple example should be clear. During retrieval, a user is not typically searching through a database of tags, but through a knowledge-base that includes information that is common to the documents. knowledge-base generated from a document (or set of documents) in which there are many references, say, to the name "George Washington" would include only one instance representing the person, and many links from that instance to the instances of NAME that reference it. While representing the background information in this manner simplifies searching dramatically, it is during browsing that this technique has the most profound advantages.

5.5 Documentation

Documentation is generated manually from a DTD by the DTD designers, with natural language descriptions of the intended usage and rendering of the tags. We know from our experience, validated also by existing software engineering practice, that development of documentation and augmentation of the basic ontology influence each other, since the formal descriptions in the

ontology often clarify or force further specification in the documentation, and existing documentation is used to assist in developing the formal semantics.

6. Benefits

We believe our approach offers benefits to the text encoding community that naturally complement existing tools and techniques. These benefits include improved search and dramatically improved browsing, semantic consistency checking, multiple views of the data, and more expressive manipulation of the DTD and marked-up texts. In this section we use specific examples to demonstrate some of the benefits of using our approach.

6.1 Searching and Browsing

We believe the principal advantage of our approach is improvement of searching and browsing for information retrieval. The added knowledge, resulting from both a principled approach to ontology development and the background information that ties the meanings of the tags together, makes it possible to more precisely specify a query. Furthermore, we perceive a new paradigm in retrieval in which users will use searches to support *browsing* of the information in the knowledge-base for scholarly research.

Imagine, for example, a scholar who is motivated by current events to research how commonplace it was, for government officials to mention government business in their personal correspondence during the civil war period in American history. While a truly intelligent library system may one day be able to answer such a question directly, we are interested in enabling such a search to the extent that today's technology allows.

The historian would begin by finding a digital library that includes marked-up versions of historical documents, such as the Model Editions Partnership's Civil-War era documents [Chesnutt (1995)]. He would then enter the query, "Find all government officials during the years 1860-1865," and be would then be presented with a list of all the government officials the system knows about. The system knows about government officials because the markup includes tags that identify government documents, and in one of our historical ontologies we would have a rule that says, "the author of a government document is a government official." There are similar rules for names that appear in tags that signify senders, recipients, signatures, etc. The system can also infer dates of service for these officials from the dates of the documents.

The power of our approach comes from the fact that it is fairly easy to specify such rules and then capitalize on the data already in the marked-up documents. Thus, rather than enumerating all the government officials during the Civil War, we specify the rules and let the system gather that information for us. Of course, the information may be incomplete; for example, if the library does not include documents in which a particular government official is mentioned in the right context (i.e., as content of a author, recipient, etc. tag), the system will not know that person is a government official.

The user's next step is to pick a person from the list of government officials and ask the system to display the information known about that person. This will show the user the kind of information the system keeps about government officials in the form of the labeled links to this information, e.g., "author-of", "recipient-of", "sender-of", "member-of", etc. The historian can ask to follow any of these links, such as the "author-of" link, after which he will be presented with a list of all the documents of which this person is the author. The documents are listed along with their most specialized parent (i.e. the parent concept lowest in the concept taxonomy), so the user will see a list of document titles and document types; for example:

```
PERSON-102: Andrew Johnson AUTHOR-OF:

DOCUMENT-23: Public Address to Baltimore
MEMO-54: Message to Lincoln
LETTER-32: Letter to Mrs. Johnson
```

If the user selects LETTER-32, he sees all the information that the system keeps about such a document, including links such as "has-author" and "has-recipient," and all the parent concepts (the list view above reveals only one parent, but most instances will have many parent concepts inherited down through the taxonomy from all its immediate parents). In this example, the concept is an instance of a LETTER, PERSONAL-DOCUMENT, HISTORICAL-DOCUMENT, DOCUMENT, MEP-DOCUMENT, and several others.

Now, assume the user selects the PERSONAL-DOCUMENT concept. All concepts in the ontology have brief natural language descriptions, so the user will see something like this (slightly abridged here due to space constraints):

```
Concept PERSONAL-DOCUMENT:
  Comments: "A Personal-Document is a document, usually a letter or diary
entry, that was considered part of the author's personal life.  This
concept was originally disjoint with PROFESSIONAL-DOCUMENT, however we have
found several exceptions and removed that restriction."
  Parents: DOCUMENT
  Ancestors: DOCUMENT OBJECT LIBRARY-THING
```

The user now has learned all the information he needs to form a more specific query, which is essentially, "Find all the PERSONAL-DOCUMENTs written by GOVERNMENT-OFFICIALs in the years 1860-1865 whose recipients were not GOVERNMENT-OFFICIALS." The result will provide all documents the user is looking for, and finishing the research will involve reading through these documents.

In the future, collaborations with user-interface groups will lead to the ability to deliver an entire document composed of the results of the query. Such a dynamic document could be pruned on-line as the user discovers parts (individual letters) that do not match his criteria.

Eventually the scenario above would be augmented by a powerful interface that assists users with the query language. Until that time, and probably even after, we expect that trained librarians who understand the representation and the query language will assist users with this system. Note that, of course, modern communication technology, from phones to chat rooms to 3D web spaces, imply that the human assistance does not require proximity.

6.2 Semantic Consistency Checking

While SGML-based tools for checking the syntax of the markup in documents exist, our tools add the ability to check the semantic consistency of the markup.

Our ontology can express certain constraints beyond those that are purely syntactic. For example, we have a knowledge-based constraint that says, "Fictional characters must have names of type FICTIONAL." Such a simple yet useful rule could catch the following inconsistency in the markup:

```
....<name type=fictional>Asmodeus</name>...
....<name type=person>Asmodeus</name>...
```

This sequence of tags in a marked-up text will generate an error during translation to the knowledge-base, because it would imply that the instance represetning the character Asmodeus is to be classified as both a PERSON and a FICTIONAL-CHARACTER. Because these concepts are defined to be disjoint in the ontology, an error condition is raised. This type of simple inconsistency occurs frequently in marked-up texts, especially when large amounts of text are encoded by different people. Such inconsistencies cannot be detected using SGML validation tools. As a result, thee errors often go undetected, or are left to be found and corrected via hand-validation. Detection of these kinds of errors at the time of text entry helps to ensure the accuracy of the markup and reduce the time and cost of hand validation.

Other types of errors, such as misspelled names, not detected automatically, but rather are made more obvious because of the way the data is organized and accessed. For example, using the rule mentioned previously that says, roughly, "The sender and recipients of all government documents are government officials," and a document with the recipient name spelled wrong, i.e.:

```
...<sender><name type=person>Andrew Jonson</name></sender>...
```

a new government official named "Andrew Jonson" is be created. During browsing, the results of a query to search for “all government officials during the years 1860-1865,” would include an instance representing Andrew Jonson. By tracing the source of the person (easy to do because of the links in the knowledge base—see Figure 6), it is easy to detect that the name has been misspelled.

6.3 DTD and Markup Manipulation

The use of a class structure and the ability to automatically classify concepts makes it possible to manipulate entire documents or document sets to reflect systematic changes to the DTD, i.e. to actually alter the markup. For example, in the previous section we introduced a “virtual tag” called GOVERNMENT-OFFICIAL: this is not a tag defined in the DTD, but rather a new sub-category of PERSON introduced in the knowledge-based representation of a marked-up document. If it were later decided to update the DTD to add this new tag, it would be simple to regenerate all the marked-up documents to include the proper usage of the updated DTD.

The use of the class hierarchy also makes it possible to manipulate sets of tags as one group. For example, since there is a hierarchy of paragraph-like elements in our CES base ontology, we could request that all paragraph-like tags be removed from the markup in order to generate a minimally marked-up document.

The existence of a class hierarchy corresponding to the tags also has implications for integrating texts that were encoded using different DTDs. When there is a correspondence between two tags (such as the <PERSNAME> and <NAME TYPE=PERSON>), a parent class can be created that subsumes each. Queries over sets of documents with different encodings can thus be unified by having the queries use the subsuming class. For example, a PERSON-NAME-TAG class might be created that subsumes (is the parent of) the concept PERSNAME (generated from one DTD) and the concept NAME-PERSON (generated from the second DTD). Searches for instances of PERSON-NAME-TAG will then find instances of either.

Recognition of general classes that subsume similar tags from multiple DTDs is *not* automatic. The similarities must be identified and reasonable choices for subsuming tags must be made. We are just beginning to explore this area in depth.

7. Conclusion

The representation of SGML documents in a knowledge representation system such as CLASSIC offers the potential to provide considerably more powerful query and retrieval capabilities than have previously been available. In particular, it will enable the manipulation of and access to elements within documents on the basis of semantic rather than purely syntactic (structural) properties. Classes of elements can be accessed, and knowledge of the DTD is not essential for constructing queries. Further, CLASSIC's inferencing capabilities can provide access to information that is not directly retrievable from the document structure, upon which all current systems rely.

The representation of SGML documents in CLASSIC may also have repercussions for DTD design. We have found that wide variations in the kinds of information represented by elements, attributes, and tag content often occur, even within the same DTD. However, the formal representation of elements, attributes, and content as CLASSIC objects demands consistency in their use within the DTD. The development of a set of principles for DTD design is a desideratum among the encoding community; we are looking into the ways in which formalization of DTDs in CLASSIC can contribute to this development.

At the same time, the use of a system such as CLASSIC allows for greater flexibility in tagging text. For example, for names of people, the encoder can use the general NAME tag--or even more generally, RS (referring string)--or provide a very precise encoding using PERSNAME with AME, LASTNAME, etc. elements inside. Once represented in CLASSIC, these objects can be both recognized as members of the class person-name and accessed and manipulated as such. This frees the encoder to choose an encoding for each name that is appropriate; there is no need for absolute consistency to enable the semantic identity of the two elements to be recognized. More generally, it allows precise tag semantics to be instantiated in a system external to the encoded text.

Finally, one of the principal benefits of a formal system is that violations of the formalism are easy to detect automatically. SGML has always provided a formal representation for the syntax (content models) of tag sets, and violations of this syntax are immediately flagged, which provides some assistance in catching errors during markup. However, SGML provides no support for detecting semantic errors. As a result, groups using DTDs as the basis for markup of documents have found it necessary to adopt software engineering principles for documenting the proper usage of tags by encoders (see, for example, Flanders [1998]). This results in the creation of extensive documentation with which encoders must be familiar in order to make full use of the desired semantics. This support is informal, relying on human memory and other cognitive factors relating to the way the usage information is indexed. As a result, encoders do not always use the right tags. By augmenting the formal syntax of the DTD in SGML with a formal semantics in CLASSIC, some of these problems can be alleviated. Much of the intended usage for a tag can be represented in such a way that consistency checking will detect many cases of inappropriate or under-specified tags. This is no panacea, but it does go a long way toward providing some much-needed support for encoders.

Acknowledgments

The present research has been partially funded by US NSF RUI grant IRI-9413451 and AT&T Bell Laboratories. The authors would like to acknowledge the contribution of Greg Priest-Dorman and Tim McGraw to our work, and thank Elli Mylonas and Allen Renear for their helpful comments and patience. Finally, we would like to thank the anonymous reviewers for their helpful and in-depth comments.

References

Artale, A., Franconi, E., Guarino, N., Pazzi, L. (1996) Part-Whole Relations in Object-Centered Systems: An Overview. *Data and Knowledge Engineering Journal.* **20**. pp 347-383. Elsevier.

Blake, G.E., Consens, M., Davis, I.J., Kilpelainen, P., Kuikka, E, Larson, P.-A., Snider, T., Tompa, F.W. (1997). *Text/Relational database management systems: Overview and proposed SQL extensions*. Available at http://solo.uwaterloo.ca/trdbms/.

Borgida, A. (1998) On the Relative Expressiveness of Description Logics and Predicate Logics. To appear, *Artificial Intelligence Journal.* Available at ftp://cs.rutgers.edu/pub/borgida/dl-vs-fol.dvi.Z.

Brachman, R. (1983) What is-a is and isn't. *IEEE Computer*. October, 1983. pp 30-36.

Brachman, R., Schmolze, J. (1985). An overview of the KL-ONE Knowledge Representation System. *Cognitive Science.* **9**(2). Pp. 171-216.

Brachman, R, Borgida, A., McGuinness, D., and Resnick, L. The CLASSIC Knowledge Representation System (1989). *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan-Kaufman.

Chesnutt, D. (1995) The Model Editions Partnership. *D-Lib Magazine.* November, 1995. Available at http://www.dlib.org/.

Flanders, J. (1998)*The Brown University Womens Writers Project*. http://www.wwp.brown.edu/

Harie, S., Ide, N., Le Maitre, J., Murisasco, E., Véronis, J. (1996). SgmlQL - An SGML Query Language. *Proceedings of SGML'96,* 127.

Ide, N. (1998a). *Corpus Encoding Standard: SGML Guidelines for Encoding Linguistic Corpora.* Proceedings of the First International Language Resources and Evaluation Conference (LREC), Granada, Spain, 463-470. CES Documentation and DTDs available at http://www.cs.vassar.edu/CES/.

Ide, N. (1998b) *Encoding Linguistic Corpora.* Proceedings of the Sixth Workshop on Very Large Corpora (WVLC6), Montréal, Canada, 9-17.

Minsky, M. (1981). A Framework for Representing Knowledge. *Mind Design.* MIT Press, 95-128.

Patel-Schneider, P., Swartout, B. (1993) *Description Logic Knowledge Representation System Specification*. Fromthe KRSS group of the ARPA Knowledge Sharing Effort, available at http://dl.kr.org/dl/.

Simons, G. (1997). Using architectural forms to map TEI data into an object-oriented database. *Proceedings of TEI-10.*

Welty, C. (1996). Intelligent Assistance for Navigating the Web. *Proceedings of the 1996 Florida AI Research Symposium*. AAAI Press.

Welty, C.. (1998). The Ontological Nature of Subject Taxonomies. *Proceedings of the 1998 International Conference on Formal Ontology in Information Systems*. IOS Press “Frontiers in Artificial Intelligence and Applications” series.