SHOE: A Blueprint for the Semantic Web

Jeff Heflin
James Hendler
Sean Luke

February 21, 2003

Abstract

The term *Semantic Web* was coined by Tim Berners-Lee to describe his proposal for "a web of meaning," as opposed to the "web of links" that currently exists on the Internet. To achieve this vision, we need to develop languages and tools that enable machine understandable web pages. The SHOE project, begun in 1995, was one of the first efforts to explore these issues. In this paper, we describe our experiences developing and using the SHOE language. We begin by describing the unique features of the World Wide Web and how they must influence potential Semantic Web languages. Then we present SHOE, a language which allows web pages to be annotated with semantics, describe its syntax and semantics, and discuss our approaches to handling the problems of interoperability in distributed environments and ontology evolution. Finally, we provide an overview of a suite of tools for the Semantic Web, and discuss the application of the language and tools to two different domains.

1 Introduction

The World Wide Web is a vast repository of information, but its utility is restricted by limited facilities for searching and integrating this information. The problem of making sense of the Web has engaged the minds of numerous researchers from fields such as databases, artificial intelligence, and library science; and these researchers have applied numerous approaches in an attempt to solve it. Tim Berners-Lee, inventor of the Web, has coined the term *Semantic Web* to describe a vision of the future in which the "web of links" is replaced with a "web of meaning." In this paper, we examine the thesis that the "the Semantic Web can be achieved if we describe web resources in a language that makes their meaning explicit."

Any language for the Semantic Web must take into account the nature of the Web. Let's consider some of the issues that arise:

- **The Web is distributed.** One of the driving factors in the proliferation of the Web is the freedom from a centralized authority. However, since the Web is the product of many individuals, the lack of central control presents many challenges for reasoning with its information. First, different communities will use different vocabularies, resulting in problems of synonymy (when two different words have the same meaning) and polysemy (when the same word is used with different meanings). Second, the lack of editorial review or quality control means that each page's reliability must be questioned. An intelligent web agent simply cannot assume that all of the information it gathers is correct and consistent. There are quite a number of well-known "web hoaxes" where information was published on the Web with the intent to amuse or mislead. Furthermore, since there can be no global enforcement of integrity constraints on the Web, information from different sources may be in conflict. Some of these conflicts may be due to philosophical disagreement; different political groups, religious groups, or nationalities may have fundamental differences in opinion that will never be resolved. Any attempt to prevent such inconsistencies must favor one opinion, but the correctness of the opinion is very much in the "eye of the beholder."

- **The Web is dynamic.** The web changes at an incredible pace, much faster than a user or even a "softbot" web agent can keep up with. While new pages are being added, the content of existing pages is changing. Some pages are fairly static, others change on a regular basis and still others change at unpredictable intervals. These changes may vary in significance: although the addition of punctuation, correction of spelling errors or reordering of a paragraph does not affect the semantic content of a document, other changes may completely alter meaning, or even remove large amounts of information. A web agent must assume that its data can be, and often will be, out of date.

The rapid pace of information change on the Internet poses an additional challenge to any attempt to create standard vocabularies and provide formal semantics. As understanding of a given domain changes, both the vocabulary may change and the semantics may be refined. It is important that such changes do not adversely alter the meaning of existing content.

- **The Web is massive.** Recent estimates place the number of indexable web pages at over 2 billion and this number is expected to double within a year. Even if each page contained only a *single* piece of agent-gatherable knowledge, the cumulative database would be large enough to bring most reasoning systems to their knees. To scale to the size of the ever growing Web, we must either restrict expressivity of the language or use incomplete reasoning algorithms.
- **The Web is an open world.** A web agent is not free to assume it has gathered all available knowledge; in fact, in most cases an agent should assume it has gathered rather little available knowledge. Even the largest search engines have only crawled about 25% of the available pages. However, in order to deduce more facts, many reasoning systems make the closed-world assumption. That is, they assume that anything not entailed in the knowledge base is not true. Yet it is clear that the size and evolving nature of the Web makes it unlikely that any knowledge base attempting to describe it could ever be complete.

In an attempt to deal with these issues, we have designed a language named SHOE, for Simple HTML Ontology Extensions. SHOE is one of the first languages that allows ontologies to be designed and used directly on the World Wide Web [25]. In this paper we describe work that influenced SHOE, present an overview of the language, describe its syntax and semantics, and discuss how SHOE addresses the issues posed in this introduction. We then discuss the problem of implementing a system that uses SHOE, describe some tools that enhance the language's usability, and discuss the application of these tools to two different domains. Finally, we provide an overview of related work and some concluding remarks.

2 Background

The success of the Web was made possible by the Hypertext Markup Language (HTML). With HTML, people can easily author sharable documents and link to related documents that might exist on different systems. However, HTML is mostly concerned with presentation (i.e., how a document is displayed in a web browser), and it is difficult to automatically extract content from an HTML document.

In 1998, the World Wide Web Consortium (W3C) officially released the Extensible Markup Language (XML) [6], a simplified version of the Standard Generalized Markup Language (SGML) [18] for the Web. XML allows special codes, called *tags*, to be embedded in a text data stream in order to provide additional information about the text, such as indicating that a certain word should be emphasized. Usually, each tag has a corresponding end-tag, with arbitrary text or other tags contained between them. The combination of a start-tag, end-tag and the data between is called an *element*. Some tags have additional properties, called *attributes*.

Unlike HTML, which precisely defines the structure and usage of a specific set of elements, XML allows users to define their own elements and attributes. Thus, users can create a document using content-specific as opposed to presentation-specific tags. If a document conforms to basic rules of XML, then it is said to be well-formed. If a document conforms to a common grammar, as specified by a document type definition (DTD), then is said to be valid. A DTD specifies valid elements, the contents of these elements, and which attributes may modify an element. Thus a DTD provides a syntax for an XML document, but the semantics of a DTD are implicit. That is, the meaning of an element in a DTD is either inferred by a human due to the name assigned to it, is described in a natural-language comment within the DTD, or is described in a document separate from the DTD. Humans can then build these semantics into tools that are used to interpret or translate the XML documents, but software tools cannot acquire these semantics independently. Thus, an exchange of XML documents works well if the parties involved have agreed to a DTD beforehand, but becomes problematic when one wants to search across the entire set of DTDs or to spontaneously integrate information from multiple sources [17].

A number of approaches including information retrieval, wrappers, semi-structured databases, machine learning, and natural language processing have been applied to the problem of querying and understanding HTML and/or XML web pages. However, the lack of semantics in the sources and the lack of common human knowledge in the tools greatly limits the quality of the techniques. We support an alternative approach: that authors should explicitly associate semantics with the content they provide.

In order to provide meaning for data, the knowledge must be represented in some way. Knowledge representation is a sub-field of artificial intelligence concerned with such matters. The goal of knowledge representation is to provide structures that allow information to be stored, modified, and reasoned with, all in an efficient manner. Over time, numerous knowledge representation languages with different properties have evolved, from early languages such as KL-ONE [4] and KRL [3], to more recent languages such as LOOM [26] , Classic [5], and CYC-L [23].

One of the oldest knowledge representation formalisms is semantic networks. A semantic net represents knowledge as a set of nodes connected by labeled links. In such a representation, meaning is implied by the way a concept is connected to other concepts. Frame systems are another representation that is isomorphic to semantic networks. In the terminology of such systems, a frame is a named data object that has a set of slots, where each slot represents a property or attribute of the object. Slots can have one or more values; these values may be pointers to other frames.

Advanced semantic networks and frame systems typically include the notion of abstraction, which is represented using *is-a* and *instance-of* links. An *is-a* link indicates that one class is included within another, while an *instance-of* link indicates that a concept is a member of a class. These links have correlations in basic set theory: *is-a* is like the subset relation and *instance-of* is like the element of relation. The collection of *is-a* links specifies a partial order on classes; this order is often called a taxonomy or categorization hierarchy. The taxonomy can be used to generalize a concept to a more abstract class or to specialize a class to its more specific concepts. As demonstrated by the popularity of Yahoo and the Open Directory, taxonomies clearly aid users in locating relevant information on the Web.

Many researchers in knowledge representation have become interested in the use of ontologies [14]. The term ontology, which is borrowed from philosophy, is defined as "a particular theory about being or reality." As such, an ontology provides a particular perspective onto the world, or some part there of. Where a knowledge representation system specifies how to represent concepts, an ontology specifies what concepts to represent and how they are interrelated. Most researchers agree that an ontology must include a vocabulary and corresponding definitions, but it is difficult to achieve consensus on a more detailed characterization. Typically, the vocabulary includes terms for classes and relations, while the definitions of these terms may be informal text, or may be specified using a formal language like predicate logic. The advantage of formal definitions is that they allow a machine to perform much deeper reasoning; the disadvantage is that these definitions are much more difficult to construct.

Numerous ontologies have been constructed, with varying scopes, levels of detail, and viewpoints. Noy and Hafner [28] provide a good overview and comparison of some of these projects. One of the more prominent themes in ontology research is the construction of reusable components. The potential advantages of such components are that large ontologies can be quickly constructed by assembling and refining existing components, and integration of ontologies is easier when the ontologies share components. One of the most common ways to achieve reusability is to allow the specification of an inclusion relation that states that one or more ontologies are included in the new theory [12, 23]. If these relationships are acyclic and treat all elements of the included ontology as if they were defined locally then an ontology can be said to extend its included ontologies.

One attempt to apply ideas from knowledge representation to the Web is the Resource Description Framework (RDF) [22]. The RDF data model is essentially a semantic network without inheritance: it consists of nodes connected by labeled arcs, where the nodes represent web resources and the arcs represent attributes of these resources. RDF can be embedded in Web documents using an XML serialization syntax, although its designers emphasize this is only one of many possible representations of the RDF model.

To allow for the creation of controlled, sharable, extensible vocabularies (i.e., ontologies) the RDF working group has developed the RDF Schema Specification [7]. This specification defines a number of properties that have specific semantics. RDF Schema defines properties that are equivalent to the *instance-of* and *is-a* links commonly used in knowledge representation. It also provides properties for describing properties, including the specification of a property's domain and range, as well as any properties of which it is a subproperty.

Although RDF is an improvement over HTML and XML, it is insufficient for a Semantic Web language [17]. In particular, it provides a very small set of semantic primitives and has relatively weak mechanisms for managing schema evolution. It is desirable that a Semantic Web language provides semantics that allow inferences beyond what is capable in RDF, but it is also important that the reasoning procedures for the language can scale to the volumes of data available on the Internet.

The field of deductive databases deals with combining inferential capabilities with the scalability of database systems. The datalog model is commonly used as the basis for describing deductive databases. It is similar to Prolog in that it consists entirely of Horn clauses, but differs in that it does not allow function symbols and is a strictly declarative language.[1] Datalog is based on the relational model but defines two types of relations: *extensional database* (EDB) relations are those predicates which are physically stored in the database, while *intensional database* (IDB) relations are those that can be computed from a set of logical rules.

Datalog restricts its Horn clauses to be *safe*, meaning that all of its variables are *limited*. Datalog defines "limited" as follows: variables are limited if they appear in an ordinary predicate of the rule's body, appear in an '=' comparison with a constant, or appear in an '=' comparison with another limited variable. Datalog's Horn clauses may depend on each other recursively. Datalog allows negation in a limited form called *stratified negation*, which we will not discuss here.

Datalog is relevant to the design of a Semantic Web language because it allows important classes of rules to be expressed while inference algorithms such as *magic sets*, which combine the best features of forward and backward chaining, provide efficient reasoning. Additionally, it seems reasonable to expect that the Web will consist of a large EDB and a comparatively small IDB, which is an ideal situation for a deductive database system.

3 The SHOE Language

SHOE combines features of markup languages, knowledge representation, datalog, and ontologies in an attempt to address the unique problems of semantics on the Web. It supports knowledge acquisition on the Web by augmenting it with tags that provide semantic meaning. The basic structure consists of *ontologies*, which define rules that guide what kinds of assertions may be made and what kinds of inferences may be drawn on ground assertions, and *instances* that make assertions based on those rules. As a knowledge representation language, SHOE borrows characteristics from both predicate logics and frame systems.

SHOE can be embedded directly in HTML documents or used in XML documents. There are a number of advantages to using an XML syntax for SHOE. First, although more standard knowledge representation syntaxes, such as first-order logic or S-expressions, could be embedded between a pair of delimiting tags, Web authors are comfortable with XML-like, tag-based languages. Second, the XML syntax allows SHOE information to be analyzed and processed using the Document Object Model (DOM), allowing software that is XML-aware, but not SHOE-aware to still use the information in more limited but nevertheless powerful ways. For example, some web browsers are able to graphically display the DOM of a document as a tree, and future browsers will allow users to issue queries that will match structures contained within the tree. The third reason for using an XML syntax is that SHOE documents can then use the XSLT stylesheet standard [8] to render SHOE information for human consumption. This is perhaps the most important reason for an XML syntax, because it can eliminate the redundancy of having a separate set of tags for the human-readable and machine-readable knowledge.

In this section, we provide a brief description of the syntax of the language followed by a formal model and a discussion of SHOE's key features. The interested reader can find a detailed description of SHOE's syntax in the SHOE Specification [24].

3.1 SHOE Ontologies

SHOE uses ontologies to define the valid elements that may be used in describing entities. Each ontology can reuse other ontologies by extending them. An ontology is stored in an HTML or XML file and is made available to document authors and SHOE agents by placing it on a web server. This file includes tags that identify the ontology, state which ontologies (if any) are extended, and define the various elements of the ontology. Figure 1 shows an example of a SHOE ontology.

In SHOE syntax, an ontology appears between the tags <ONTOLOGY ID=*id* VERSION=*version*> and </ONTOLOGY>, and is identified by the combination of the *id* and *version*. An ontology can define categories, relations and other components by including special tags for these purposes.

The tag <DEF-CATEGORY> can be used to make *category definitions* that specify the categories under which various instances could be classified. Categories may be grouped as subcategories under one or more supercategories, essentially specifying the *is-a* relation that is commonly used in semantic networks and frame systems. The use of categories allows taxonomies to be built from the top down by subdividing known classes into smaller sets. The example ontology defines many categories, including Chair, which is a subcategory of Faculty.

The tag <DEF-RELATION> (which is closed by a </DEF-RELATION> tag) can be used to make *relational definitions* that specify the format of n-ary relational claims that may be made by instances regarding instances and other data. One of the relationships defined by the example ontology is advises, which is between an instance of category Faculty and an instance of category Student. A relation argument can also be one of four basic types (string, number, date, or boolean value) as is the case with the second argument of the relationship hasGPA.

SHOE uses inference rules, indicated by the <DEF-INFERENCE> tag, to supply additional axioms. A SHOE inference rule consists of a set of antecedents (one or more subclauses describing claims that entities might make) and a set of consequents (consisting of one or more subclauses describing a claim that may be inferred if all claims in the body are made). The <INF-IF> and <INF-THEN> tags indicate the antecedents and consequents of the inference, respectively. There are three types of inference subclauses: category, relation and comparison. The arguments of any subclause may be a constant or a variable, where variables are indicated by the keyword VAR. Constants must be matched exactly and variables of the same name must bind to the same value. The ontology in the example includes a rule stating that the head of a department is a chair.

As is common in many ontology efforts, such as Ontolingua and Cyc, SHOE ontologies build on or extend other ontologies, forming a lattice with the most general ontologies at the top and the more specific ones at the bottom. Ontology extension is expressed in SHOE with the <USE-ONTOLOGY> tag, which indicates the id and version number of an ontology that is extended. An optional URL attribute allows systems to locate the ontology if needed and a PREFIX attribute is used to establish a short local identifier for the ontology. When an ontology refers to an element from an extended ontology, this prefix and a period is appended before the element's name. In this way, references are guaranteed to be unambiguous, even when two ontologies use the same term to mean different things. By chaining the prefixes, one can specify a path through the extended ontologies to an element whose definition is given in a more general ontology.

Sometimes an ontology may need to use a term from another ontology, but a different label may be more useful within its context. The <DEF-RENAME> tag allows the ontology to specify a local name for a concept from any extended ontology. This local name must be unique within the scope of the ontology in which the rename appears. Renaming allows domain specific ontologies to use the vocabulary that is appropriate for the domain, while maintaining interoperability with other domains.

3.2 SHOE Instances

Unlike RDF, SHOE makes a distinction between what can be said in an ontology and what can be said on an arbitrary web page. Ordinary web pages declare one or more instances that represent SHOE entities, and each instance describes itself or other instances using categories and relations. An example of a SHOE instance is shown in Figure 2. The syntax for instances includes an <INSTANCE> element that has an attribute for a KEY that uniquely identifies the instance. We recommend that the URL of the web page be used as this key, since it is guaranteed to identify only a single resource. An instance commits to a particular ontology by means of the <USE-ONTOLOGY> tag, which has the same function as the identically named element used within ontologies. To prevent ambiguity in the declarations, ontology components are referred to using the prefixing mechanism described earlier. The use of common ontologies makes it possible to issue a single logical query to a set of data sources and enables the integration of related domains. Additionally, by specifying ontologies the content author indicates exactly what meaning he associates with his claims, and does not need to worry that an arbitrary definition made in some other ontology will alter this meaning.

An instance contains ground *category claims* and *relation claims*. A category claim is specified with the <CATEGORY NAME=*y* FOR=*x*> tag, and says that the instance claims that an instance x is an element of a category y. If the FOR attribute is omitted, then the category claim is about the instance making the claim. In the example, the instance http://univ.edu/jane/ claims that http://univ.edu/jane/ is a Chair and http://univ.edu/john/ is a Student.

A relational claim is enclosed by the <RELATION NAME=*foo*> and </RELATION> tags, and says that the instance claims that an n-ary relation *foo* exists between some n number of appropriately typed arguments consisting of data or instances. In the example, the instance http://univ.edu/jane/ claims that there exists the relation advises between http://univ.edu/jane/ and her student http://univ.edu/john/ and that that the name of http://univ.edu/jane/ is Jane Smith.

3.3 SHOE's Semantics

In order to describe the semantics of SHOE, we will extend a standard model-theoretic approach for definite logic with mechanisms to handle distributed ontologies. For simplicity, this model intentionally omits some minor features of the SHOE language.

We define an ontology O to be a tuple $< V, A >$ where V is the vocabulary and A is the set of axioms that govern the theory. Formally, V is a set of predicate symbols, each with some arity ≥ 0 and distinct from symbols in other ontologies,[2] while A is a set of definite program clauses that have the standard logical semantics.[3] We now discuss the contents of V and A, based upon the components that are defined in the ontology:

A <USE-ONTOLOGY> statement adds the vocabulary and axioms of the specified ontology to the current ontology. Due to the assumption that names must be unique, name conflicts can be ignored.

A <DEF-RELATION> statement adds a symbol to the vocabulary and for each argument type that is a category, adds an axiom that states that an instance in that argument must be a member of the category. If the tag specifies a name R and has n arguments then there is an n-ary predicate symbol R in V. If the type of the ith argument is a category C, then $[R(x_1, ..., x_i, ...x_n) \rightarrow C(x_i)] \in A$. This rule is a consequence of SHOE's open-world policy: since there is no way to know that a given object in a relation claim is *not* a member of a category appropriate for that relation, it is better to assume that this information is yet undiscovered than it is to assume that the relation is in error. However, when arguments are basic data types, type checking is performed to validate the relation. Basic data types are treated differently because they *are* different: they have syntax which can be checked in ways that category types cannot, which allows us to impose stringent input-time type checking on basic data types.

A <DEF-CATEGORY> adds a unary predicate symbol to the vocabulary and possibly a set of rules indicating membership. If the name is C, then $C \in V$. For each super-category P_i specified, $[C(x) \rightarrow P_i(x)] \in A$.

A <DEF-INFERENCE> adds one or more axioms to the theory. If there is a single clause in the <INF-THEN>, then there is one axiom with a conjunction of the <INF-IF> clauses as the antecedent and the <INF-THEN> clause as the consequent. If there are n clauses in the <INF-THEN> then there are n axioms, each of which has one of the clauses as the consequent and has the same antecedent as above.

A <DEF-RENAME> provides an alias for a non-logical symbol. It is meant as a convenience for users and can be implemented using a simple pre-processing step that translates the alias to the original, unique non-logical symbol. Therefore, it can be ignored for the logical theory.

A formula F is well-formed with respect to O if 1) F is an atom of the form $p(t_1, ..., t_n)$ where p is a n-ary predicate symbol such that $p \in V$ or 2) F is a Horn clause where each atom is of such a form. An ontology is well-formed if every axiom in the ontology is well-formed with respect to the ontology.

Now we turn our attention to data sources, such as one or more web pages, that use an ontology to make relation and category claims. Let $S =< O_S, D_S >$ be such a data source, where $O_S =< V_S, A_S >$ is the ontology and D_S is the set of claims. S is well-formed if O_S is well-formed and each element of D_S is a ground atom that is well-formed with respect to O_S. The terms of these ground atoms are constants and can be instance keys or values of a SHOE data type.

We wish to be able to describe the meaning of a given data source, but it is important to realize that on the Web, the same data could have different meanings for different people. An agent may be able to draw useful inferences from a data source without necessarily agreeing with the ontology intended by the data's author. A common case would be when an agent wishes to integrate information that depends on two overlapping but still distinct ontologies. Which set of rules should the agent use to reason about this data? There are many possible answers, and we propose that the agent should be free to choose. To describe this notion, we define a *perspective* $P =< S, O >$ as a data source $S =< O_S, D_S >$ viewed in the context of an ontology $O =< V, A >$. If $O = O_S$ then P is the *intended perspective*, otherwise it is an *alternate perspective*. If there are elements of D_S that are not well-formed with respect to O, these elements are considered to be irrelevant to the perspective. If W_S is the subset of D_S that is well-formed with respect to O, then P is said to result in a definite logic theory $T = W_S \cup A$.

Finally, we can describe the semantics of a perspective P using a model theoretic approach. An interpretation of the perspective consists of a domain, the assignment of each constant in S to an element of the domain, and an assignment of each element in V to a relation from the domain. A model of P is an interpretation such that every formula in its theory T is true with respect to it. We define a query on P as a Horn clause with no consequent that has the semantics typically assigned to such queries for a definite logic program T.

We also introduce one additional piece of terminology that will be used later in the paper. If every ground atomic logical consequence of perspective P is also a ground atomic logical consequence of perspective P' then P' is said to *semantically subsume* P. In such cases, any query issued against perspective P' will have at least the same answers as if the query was issued against P. If two perspectives semantically subsume each other, then they are said to be equivalent.

3.4 Interoperability in Distributed Environments

SHOE was designed specifically with the needs of distributed internet agents in mind. A key problem in distributed systems is interoperability; SHOE attempts to maximize interoperability through the use of shared ontologies, prefixed naming, prevention of contradictions, and locality of inference rules. This section discusses each of these in turn.

Figure 3 shows how the ontology extension and renaming features of the language promote interoperability. When two ontologies need to refer to a common concept, they should both extend an ontology in which that concept is defined. In this way, consistent definitions can be assigned to each concept, while still allowing communities to customize ontologies to include definitions and rules of their own for specialized areas of knowledge. These methods allow the creation of high-level, abstract unifying ontologies extended by often-revised custom ontologies for specialized, new areas of knowledge. There is a trade-off between trust of sources far down in the tree (due to their fleeting nature) and the ease of which such sources can be modified on-the-fly to accommodate new important functions (due to their fleeting nature). In a dynamic environment, an ontology too stable will be too inflexible; but of course an ontology too flexible will be too unstable. SHOE attempts to strike a balance using simple economies of distribution.

The problems of synonymy and polysemy are handled by the extension mechanism and <DEF-RENAME> tag. Using this tag, ontologies can create aliases for terms, so that domain-specific vocabularies can be used. For example, in Figure 3, the term DeptHead in univ-ont2 means the same thing as Chair in univ-ont due to a <DEF-RENAME> tag in univ-ont2. Although the extension and aliasing mechanisms solve the problem of synonymy of terms, the same terms can still be used with different meanings in different ontologies. This is not undesirable, a term should not be restricted for use in one domain simply because it was first used in a particular ontology. As shown in Figure 3, in SHOE different ontologies may also use the same term to define a different concept. Here, the term Chair means different things in univ-ont and furn-ont because the categories have different ancestors. To resolve any ambiguity that may arise, ontological elements are always referenced using special prefixes that define unique paths to their respective enclosing ontologies. Instances and ontologies that reference other ontologies must include statements identifying which ontologies are used and each ontology is assigned a prefix which is unique within that scope. All references to elements from that ontology must include this prefix, thereby uniquely identifying which definition is desired.

Recall that each SHOE instance must be assigned a key, and that this key is often the URL of the web page describing the instance. In SHOE, it is assumed that each key identifies exactly one entity, but no assumptions are made about whether two distinct keys might identify the same entity. This is because many different URLs could be used to refer to the same page (due to the facts that a single host can have multiple domain names and operating systems may allow many different paths to the same file). To solve these problems in a practical setting, a canonical form can be chosen for the URL; an example rule might be that the full path to the file should be specified, without operating systems shortcuts such as '~ ' for a user's home directory. Even then, there are still problems with multiple keys possibly referring to the same conceptual object. At any rate, this solution ensures that the system will only interpret two objects as being equivalent when they truly are equivalent. Ensuring that two object references are matched when they conceptually refer to the same object is an open problem.

In distributed systems, a contradiction cannot be handled by simply untelling the most recent assertion, otherwise the system would give preference to those authors who provided their information first, regardless of whether it was true, false, or a matter of opinion. Rather than delve into complex procedures for maintaining consistency, we chose to keep SHOE easy to understand and implement. Therefore, we have carefully designed the language to eliminate the possibility of contradictions between agent assertions. SHOE does this in four ways:

1. SHOE only permits assertions, not retractions.
2. SHOE does not permit logical negation.
3. SHOE does not have single-valued relations, that is, relational sets which may have only one value (or some fixed number of values).
4. SHOE does not permit the specification of disjoint classes.

Although this restricts the expressive power of the language, in our practical experience, we have not yet found it to be a significant problem. It should be noted that SHOE does not prevent "contradictions" that are not logically inconsistent. If claimant A says $father(Mark, Katherine)$ and claimant B says $father(Katherine, Mark)$, the apparent contradiction is because one claimant is misusing the $father$ relation. However, this does not change the fact that A and B made those claims.

A similar problem may occur in an ontology where an inference rule derives a conclusion whose interpretation would be inconsistent with another ontology. Therefore, it is the ontology designer's responsibility to make sure that the ontology is correct and that it is consistent with all ontologies that it extends. It is expected that ontologies which result in erroneous conclusions will be avoided by users, and will thus be weeded out by natural selection.

Yet another problem with distributed environments is the potential interference of rules created by other parties: a rule created by one individual could have unwanted side-effects for other individuals. For these reasons, SHOE only allows rules to be defined in ontologies, and the only rules that could apply to a given claim are those which are defined in the ontologies used by the instance making claim. Since rules can only be expressed in ontologies, the process of determining when a rule is applicable is simplified, and page authors can use this to control the side-effects of their claims. If a user wishes to view an instance in a different context or use it in ways originally unintended by the author, then the user can use an alternate perspective for the instance that is based on a different, but compatible ontology.

3.5 Ontology Evolution

The Web's changing nature means that ontologies will have to be frequently changed to keep up with current knowledge and usage. Since physically revising an ontology can invalidate objects that reference it for vocabulary and definitions, it is useful to think of a revision as a new ontology that is a copy of the original ontology with subsequent modifications. In fact, this is exactly what SHOE does: each version of an ontology is a separate Web resource and is assigned a unique version number, while all references to an ontology must denote a specific version. How then, is a revision different from an ontology with a different identifier? The answer is that a revision can specify that it is backwardly-compatible with earlier version (using the backward-compatible-with attribute of the ontology), which allows interoperability between sources that use different versions of an ontology.

Before we define backward-compatibility, we will first characterize and compare different types of revisions using the formal model developed in Section 3.3. To be succinct, we will only discuss revisions that add or remove components; the modification of a component can be thought of as a removal followed by an addition. In the rest of this section, O will refer to the original ontology, O' to its revision, P and P' to the perspectives formed by these respective ontologies and an arbitrary source $S =< O, D_S >$, and T and T' to the respective theories for these perspectives.

If a revision O' adds an arbitrary rule to ontology O, then for any source S, the perspective P' semantically subsumes P. Since the revision only adds a sentence to the corresponding theory $T' \supset T$, and since first-order logic is monotonic any logical consequence of T is also a logical consequence of T'. Thus, when a revision that adds rules provides an alternate perspective of a legacy data source, there may be additional answers that were not originally intended by the author of the data. Similar reasoning is used to ascertain that if the revision removes rules, then P semantically subsumes P'.

If O' consists of the removal of categories or relations from O, then P semantically subsumes P'. This is because there may be some atoms in S that were well-formed w.r.t. O that are not well-formed w.r.t. O'. Informally, if categories or relations are removed, predicate symbols are removed from the vocabulary. If the ground atoms of S depended on these symbols for well-formedness then when the symbols are removed the sentences are no longer well-formed. Thus, $T' \subseteq T$ and due to the monotonicity of definite logic, every logical consequence of T' is a logical consequence of T. Revisions of this type may mean that using the revised ontology to form a perspective may result in fewer answers to a given query.

Finally, if the revision only adds categories or relations, the corresponding perspective P' is equivalent to P. Since $T' \supset T$ it is easy to show that P' semantically subsumes P. The proof of the other direction depends on the nature of the axioms added: $R(x_1, ..., x_i, ...x_n) \rightarrow C(x_i)$ for relations and $C(x) \rightarrow P_i(x)$ for categories. It also relies on the fact that due to the definitions of categories and relations, the predicate of each antecedent is a symbol added by the new ontology and must be distinct from symbols in any other ontology. Therefore any atoms formed from these predicates are not well-formed with respect to any preexisting ontology. Thus, there can be no such atoms in S, since S must be well-formed with respect to some ontology $\neq O'$. Since the antecedents cannot be fulfilled, the rules will have no new logical consequences that are ground atoms. Since P semantically subsumes P' and vice versa, P and P' are equivalent. This result indicates that we can safely add relations or categories to the revision, and maintain the same perspective on all legacy data sources.

We can now define backward-compatibility: an ontology revision O' can be said to be backward-compatible with an ontology O if for any data source $S =< O, D_S >$, the perspective $P' =< S, O' >$ semantically subsumes the perspective $P =< S, O >$. Put simply, if every logical consequence of the original is also a consequence of the revision, then the revision is backward-compatible. By our analysis above, if a revision only adds categories, relations, or rules then it is backward compatible with the original, while if it removes any of these components then it is not backward compatible.

With this notion of backward compatibility, agents can assume with some degree of confidence that a perspective that uses the backward compatible revision will not alter the original meaning of the data source, but instead supplement it with information that was originally implicit. Agents that don't wish to assume anything, may still access the original version because it still exists at the original URL. However, it should be noted that this versioning mechanism is dependent on the compliance of the ontology designers. Since an ontology is merely a file on a web server, there is nothing to prevent its author from making changes to an existing ontology version. This is the price we pay for have having a system that is flexible enough to cope with the needs of diverse user communities while being able to change rapidly. However, we presume that users will gravitate towards ontologies from sources that they can trust and ontologies that cannot be trusted will become obsolete.

Although, ideally integration in SHOE is a byproduct of ontology extension, a distributed environment in which ontologies are rapidly changing is not always conducive to this. Even when ontology designers have the best intentions, a very specialized concept may be simultaneously defined by two new ontologies. To handle such situations, periodic ontology integration must occur. Ontologies can be integrated using a new ontology that maps the related concepts using inference rules, by revising the relevant ontologies to map to each other, or by creating a new more general ontology which defines the common concepts, and revising the relevant ontologies to extend the new ontology. We discuss each of these solutions in more detail in [15].

3.6 Scalability

The scalability of a knowledge representation depends on the computational complexity of the inferences that it sanctions. We intentionally omitted from SHOE features such as disjunction and negation that typically make knowledge representation systems intractable. Since SHOE is essentially a set of Horn clauses, a naive forward-chaining inference algorithm can be executed in polynomial time and space in the worst case. Of course, the expected size of an extensional database built from the Web makes this an undesirable option.

We carefully chose SHOE's set of semantic primitives so that it could be reduced to datalog, thus allowing the use of optimized algorithms such as magic sets [33]. The semantics of SHOE categories can be easily described using a datalog rule. For example, category membership such as the fact that a Person is a Mammal may be expressed by using unary predicates and a rule of the form:

```
Mammal(X) :- Person(x)
```

Since SHOE's inferential rules are basically Horn clauses, they also map directly to datalog. Furthermore, SHOE's more restrictive variable join rule ensures that all SHOE inference rules are safe. Thus, SHOE is equivalent to safe datalog without negation.

Obviously, SHOE systems can benefit from the deductive database research, but the massive size of the resulting KBs may at times yield unacceptable performance. Therefore SHOE has a modular design that allows systems to cleanly provide differing degrees of inferential capability. For example, a system may choose only to implement transitive category membership, or may choose to implement no inference at all, thus providing only access to the extensional database. Although such systems might be incomplete reasoners with respect to the intended perspective (see section 3.3), they can be complete reasoners with respect to an alternate perspective (i.e., one that mirrors the intended perspective but omits all inference rules and/or category definitions).

4 Implementation

In the previous section we described the semantics of SHOE and discussed its treatment of some of the specific challenges for a Semantic Web language. In this section we describe a suite of tools and methods for using the language in practice, and describe its application to the domains of computer science departments and food safety.

Although there are many possible ways to use the SHOE language, the simplest is one that parallels the way the Web works today. There are numerous tools that can be used to produce content, and this content is published on the Web by making it accessible via a web server. A web-crawler can then gather relevant pages and store it in a repository, which can then be queried by some type of user interface. The key component of a SHOE system is that both the content and the ontologies which provide semantics for the content are published on the Web. Since this information is structured and has semantics, the repository should be a knowledge base rather than an information retrieval system. This basic architecture is shown in Figure 4.

The first step in using SHOE is to locate or design an appropriate ontology. To assist in this process, there should be a central repository of ontologies. A simple repository could be a set of web pages that categorize ontologies, while a more complex repository may associate a number of characteristics with each ontology so that specific searches can be issued. A web-based system that uses the later approach is described in [34]. If it is determined that no suitable ontology is available, any newly created ontology should always extend available ontologies that contain related concepts.

The process of adding semantic tags to a web page is called annotation. There are a number of tools that can be used in this process, from simple text editors, to GUI-based editors, to semi- or fully-automated techniques. Text editors have the advantage that they are common place, but require that users become familiar with the syntax of SHOE. We have developed a GUI editor for SHOE called the Knowledge Annotator that allows the user to create markup by selecting from lists and filling in forms. However, there are cases where the user may want to generate markup that corresponds to information from large lists or tables in pre-existing web pages. For such situations, our Running SHOE tool can be used to quickly create a wrapper for an existing document by specifying tokens that indicate records and fields, and then mapping these records and fields to classes and relations from some ontology. This tool can be used to generate a large set of semantic markup from regular documents in minutes. Other approaches to generating markup can include machine learning, information extraction, and, in limited domains, natural language processing techniques.

Once the necessary content and ontologies documents are published on the Web, they can be harvested by a web-crawler. Exposé, the SHOE web-crawler, searches for web pages with SHOE markup and stores the information in a knowledge base. Whenever a page commits to an ontology unknown to the system, this ontology is also retrieved and stored in the knowledge base. The biggest drawback to a web-crawler approach is that the information is only as recent as the last time the crawler visited the source web page. In certain applications, such as comparison shopping, this time period may be unacceptable. An interesting direction for future research is the development of focused crawlers that seek answers to particular questions in real-time.

As mentioned above, the web-crawler needs a knowledge base in which to store the results of its efforts. A knowledge base provides permanent storage for information and the ability to use knowledge to draw inferences from that which was explicitly stored in it. An important trade-off for candidate knowledge base systems is the degree to which it can make inferences sanctioned by the language (completeness) and the response time to queries (performance). We believe that users of SHOE systems should be able to specify their preferences with regard to this scale. Thus, many different knowledge base systems could be used for SHOE, with the appropriate tradeoffs in mind.

At the completeness end of the scale sit systems such as XSB [31], a logic programming and deductive database system. XSB is more expressive than datalog and can thus be used as a complete reasoner for SHOE. At the performance end of the scale sit relational database management systems (RDBMS), which have been traditionally used for querying enormous quantities of data. However, a relational database provides no automated inferential capability, and thus can only answer queries using the information that was explicitly stored. In between the two extremes, sit systems such as Parka [11, 32], a high-performance knowledge representation system whose roots lie in semantic networks and frame systems, but which makes use of certain database technology such as secondary storage and indexing. Parka has better inferential capabilities than a RDBMS, but less than XSB, while being faster than XSB but slower than an RDBMS.

Once information is stored in a knowledge base, it can be queried by a variety of front ends. SHOE Search [16] is a generic tool gives users a new way to browse the web by allowing them to submit structured queries and open documents by clicking on the URLs in the results. The user first chooses an ontology against which the query should be issued and then chooses the class of the desired object from a hierarchical list. After the system presents a list of all properties that could apply to that object and the user has typed in desired values for one or more of these properties, the user issues a query and is presented with a set of results in a tabular form. If the user double-clicks on a binding that is a URL, then the corresponding web page will be opened in a new window of the user's web browser.

In order to evaluate the tools and techniques for SHOE, we have used them in two different domains. The first domain is that of computer science departments. We started by creating a simple computer science department ontology[4] by hand. The scope of this ontology includes departments, faculty, students, publications, courses, and research, for a total of 43 categories and 25 relations. The next step was to annotate a set of web pages (i.e., add SHOE semantic markup to them). Every member of the Parallel Understanding Systems (PLUS) Group marked up their own web pages. Although most members used the Knowledge Annotator, a few added the tags using their favorite text editors. To get even more SHOE information, we used the Running SHOE tool on the faculty, users, courses and research groups web pages from the web sites of various computer science departments. Exposé was used to acquire the SHOE knowledge from the web pages. This resulted in a total of 38,159 assertions, which were stored in both Parka and XSB. Although the KB for this domain is very small when compared to the scale of the entire Web, the initial results are promising. For example, a query of the form $\mathsf{member}(\mathsf{http://www.cs.umd.edu},\ x) \wedge \mathsf{instance}(\mathsf{Faculty},\ x)$ takes Parka less than 250 milliseconds to answer.

The possible benefits of a system such as this one are numerous. A prospective student could use it to inquire about universities that offered a particular class or performed research in certain areas. Or a researcher could design an agent to search for articles on a particular subject, whose authors are members of a particular set of institutions, and were published during some desired time interval. Additionally, SHOE can combine the information contained in multiple sources to answer a single query. For example, to answer the query "Find all papers about ontologies written by authors who are faculty members at public universities in the state of Maryland" one would need information from university home pages, faculty listing pages, and publication pages for individual faculty members. Such a query would be impossible for current search engines because they rank each page based upon how many of the query terms it contains.

The SHOE technology was also applied to the domain of food safety. The Joint Institute for Food Safety and Applied Nutrition (JIFSAN), a partnership between the Food and Drug Administration (FDA) and the University of Maryland, is working to expand the knowledge and resources available to support risk analysis in the food safety area. One of their goals is to develop a website that will serve as a clearinghouse of information about food safety risks. This website must serve a diverse group of users, including researchers, policy makers, risk assessors, and the general public, and thus must be able to respond to queries where terminology, complexity and specificity may vary greatly. This is not possible with keyword based indices, but can be achieved using SHOE.

The initial TSE ontology was fleshed out in a series of meetings that included members of JIFSAN and a knowledge engineer. The ontology focused on the three main concerns for TSE Risks: source material, processing, and end-product use. Currently, the ontology has 73 categories and 88 relations.[5] Following the creation of the initial ontology, the team annotated web pages. There are two types of pages that this system uses. Since the Web currently has little information on animal material processing, we created a set of pages describing many important source materials, processes and products. The second set of pages are existing TSE pages that provide general descriptions of the disease, make recommendations or regulations, and present experimental results. Early annotations were difficult because the original ontology did not have all of the concepts that were needed. When the initial set of pages was completed, we ran Exposé, using Parka as the knowledge base system. Since the TSE Ontology currently does not define inference rules, Parka is able to provide a complete reasoning capability for it. The Parka KB can be queried using SHOE Search as discussed earlier, but JIFSAN also wanted a special purpose tool to help users visualize and understand the processing of animal materials.

To accommodate this, we built the TSE Path Analyzer, a graphical tool that can be used to analyze how source materials end up in products that are eventually consumed by humans or animals. This information is extremely valuable when trying to determine the risk of contamination given the chance that a source material is contaminated. It is expected that information on each step in the process will be provided on different web sites (since many steps are performed by different companies), thus using a language like SHOE is essential to integrating this information. The TSE Path Analyzer allows the user to pick a source, process and/or end product from lists that are derived from the taxonomies of the ontology. The system then displays all possible pathways that match the query. Since these displays are created dynamically based on the semantic information in the SHOE web pages, they are kept current automatically, even when the SHOE information on some remote site is changed.

5 Related Work

In recent years, there has been work to use ontologies to help machines process and understand Web documents. Fensel et al. [13] have developed Ontobroker, which proposes minor extensions to the common anchor tag in HTML. The theoretical basis for Ontobroker is frame logic, a superset of Horn logic that treats ontology objects as first class citizens. However, this approach depends on a centralized broker, and as a result, the web pages cannot specify that they reference a particular ontology, and agents from outside the community cannot discover the ontology information. Kent [20] has designed the Ontology Markup Language (OML) and the Conceptual Knowledge Markup Language (CKML), which were influenced by SHOE, but are based on the theories of formal concept analysis and information flow. However, the complexity of these theories make it unlikely that this language will be accepted by the majority of existing web developers and/or users. The Ontology Interchange Language (OIL) [10] is a new web ontology language that extends RDF and RDF Schema with description logic capabilities. Jannink et al. [19] suggest a different approach from creating web ontology languages and annotating pages; they propose that an ontology should be built for each data source, and generalization is accomplished by integrating these data sources. In this way, the data dictates the structure of the ontology rather than the other way around.

Querying the Web is such an important problem that a diverse body of research has be directed towards it. Some projects focus on creating query languages for the Web [1, 21], but these approaches are limited to queries concerning the HTML structure of the document and the hypertext links. They also rely on index servers such as AltaVista or Lycos to search for words or phrases, and thus suffer from the limitations of keyword search. Work on semistructured databases [27] is of great significance to querying and processing XML, but the semistructured model suffers the same interoperability problems as XML. Even techniques such as data guides will be of little use when integrating information developed by different communities in different contexts. Another approach involves mediators (or wrappers), custom software that serves as an interface between middleware and a data source [35, 29, 30]. When applied to the Web, wrappers allow users to query a page's contents as if it was a database. However, the heterogeneity of the Web requires that a multitude of custom wrappers must be developed, and it is possible that important relationships cannot be extracted from the text based solely on the structure of the document. Semi-automatic generation of wrappers [2] is a promising approach to overcoming the first problem, but is limited to data that has a recognizable structure.

In order to avoid the overhead of annotating pages or writing wrappers, some researchers have proposed machine learning techniques. Craven et al. [9] have trained a system to classify web pages and extract relations from them in accordance with a simple ontology. However, this approach is constrained by the time-consuming task of developing a training set and has difficulty in classifying certain kinds of pages due to the lack of similarities between pages in the same class.

6 Conclusion

In this paper, we have described many of the challenges that must be addressed by research on the Semantic Web and have described SHOE, one of the first languages to explicitly address these problems. SHOE provides interoperability in distributed environments through the use of extensible, shared ontologies, the avoidance of contradictions, and localization of inference rules. It handles the changing nature of the Web with an ontology versioning scheme that supports backward-compatibility. It takes steps in the direction of scalability by limiting expressivity and allowing for different levels on inferential support. Finally, since the Web is an "open-world," SHOE does not allow conclusions to be drawn from lack of information.

To demonstrate SHOE's features, we have described applications that show the use of SHOE. We've developed a freely available ontology for computer science pages, and we've also worked with biological epidemiologists to design an ontology for a key food safety area. These applications show that SHOE can exist on the web, and that tools using SHOE can be built and used.

Although we believe SHOE is good language that has practical use, we do not mean to suggest that it solves all of the problems of the Semantic Web. We are at the beginning of a new and exciting research field and there is still much work to do. Further research must be performed to determine how SHOE's method for interoperability scales to the thousands of ontologies that will be created on the Web and the problem of ontology evolution must be studied more closely. Additionally, the appropriate level of expressivity for a semantic web language must be explored. For example, are negation and cardinality constraints necessary, and if so, how can they be used in a decentralized system such as the Web? Finally, more user-friendly tools need to be developed, so that use of the semantic web can become routine for the layperson.

As early "pioneers," we hope that our experience with SHOE can inspire and inform others. A key goal of this project is to raise the issues that are crucial to the development of the Semantic Web and encourage others to explore them. To this end, we have made SHOE freely available on the Web, including the Java libraries and our prototype tools. Interested readers are urged to explore our web pages at *http://www.cs.umd.edu/projects/plus/SHOE/* for the full details of the language and the applications.

Acknowledgments

This work was supported by the Army Research Laboratory under contract number DAAL01-97-K0135 and the Air Force Research Laboratory under grants F306029910013 and F306020020578.

References

[1] G. Arocena, A. Mendelzon and G. Mihaila, Applications of a Web Query Language, in: *Proceedings of ACM PODS Conference* Tuscon, AZ (1997).

[2] N. Ashish and C. Knoblock, Semi-automatic Wrapper Generation for Internet Information Sources, in: *Proceedings of the Second IFCIS Conference on Cooperative Information Systems (CoopIS)* Charleston, SC (1997).

[3] D. Bobrow and T. Winograd, An overview of KRL, a knowledge representation language, *Cognitive Science* 1(1) (1977).

[4] R. Brachman and J. Schmolze, An overview of the KL-ONE knowledge representation system, *Cognitive Science*, 9(2) (1985).

[5] R. Brachman, D. McGuinness, P.F. Patel-Schneider, L. Resnick,and A. Borgida, Living with Classic: When and how to use a KL-ONE-like language, in: J. Sowa, ed., *Explorations in the representation of knowledge* (Morgan-Kaufmann, CA, 1991).

[6] T. Bray, J. Paoli and C. Sperberg-McQueen, Extensible Markup Language (XML), W3C (World Wide Web Consortium), at: http://www.w3.org/TR/1998/REC-xml-19980210.html. (1998)

[7] D. Brickley and R.V. Guha, Resource Description Framework (RDF) Schema Specification (Candidate Recommendation), W3C (World-Wide Web Consortium) (2000). (At http://www.w3.org/TR/2000/CR-rdf-schema-20000327)

[8] J. Clark, XSL Transformations (XSLT) W3C (World-Wide Web Consortium) (1999). (At http://www.w3.org/TR/1999/REC-xslt-19991116)

[9] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigram, and S. Slattery, Learning to Extract Symbolic Knowledge From the World Wide Web, in: *Proceedings of the Fifteenth American Association for Artificial Intelligence Conference (AAAI-98)* (AAAI/MIT Press, 1998).

[10] S. Decker, D. Fensel , F. van Harmelen , I. Horrocks, S. Melnik , M. Klein, and J. Broekstra, Knowledge Representation on the Web, in: *Proceedings of the 2000 International Workshop on Description Logics (DL2000)* (Aachen, Germany, August 2000).

[11] M. Evett, W. Andersen and J. Hendler, Providing Computational Effective Knowledge Representation via Massive Parallelism, in: L. Kanal, V. Kumar, H. Kitano, and C. Suttner, eds., *Parallel Processing for Artificial Intelligence*, (Elsevier Science, Amsterdam, 1993).

[12] A. Farquhar, R. Fikes and J. Rice The Ontolingua Server: A tool for collaborative ontology construction, *International Journal of Human-Computer Studies* 46(6) (1997) 707-727.

[13] D. Fensel, S. Decker, M. Erdmann, and R. Studer, Ontobroker: How to enable intelligent access to the WWW, in: *AI and Information Integration, Papers from the 1998 Workshop*, Technical Report WS-98-14 (AAAI Press, Menlo Park, CA, 1998).

[14] T. Gruber, A Translation Approach to Portable Ontology Specifications, in: *Knowledge Acquisition* 5 (1993) 199-220.

[15] J. Heflin, and J. Hendler, Dynamic Ontologies on the Web, in: *Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)* (AAAI/MIT Press, Menlo Park, CA, 2000) 443-449.

[16] J. Heflin and J. Hendler, Searching the Web with SHOE, in: *Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01* (AAAI Press, Menlo Park, CA, 2000) 35-40.

[17] J. Heflin and J. Hendler, Semantic Interoperability on the Web, in: *Proc. of Extreme Markup Languages 2000* (Graphic Communications Association, Alexandria, VA, 2000) 111-120.

[18] ISO (International Organization for Standardization) *ISO 8879:1986(E). Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML)* (International Organization for Standardization, Genevea, 1986).

[19] J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold., G. Encapsulation and Composition of Ontologies, in: *AI and Information Integration, Papers from the 1998 Workshop*, Technical Report WS-98-14 (AAAI Press, Menlo Park, CA, 1998) 43-50.

[20] R.E. Kent. Conceptual Knowledge Markup Language: The Central Core, in: *Twelfth Workshop on Knowledge Acquisition, Modeling and Management* (1999).

[21] D. Konopnicki and O. Shemueli, W3QS: A Query System for the World Wide Web, in: *Proceedings of the 21st International Conference on Very Large Databases* (Zurich, Switzerland, 1995).

[22] O. Lassila, Web Metadata: A Matter of Semantics, *IEEE Internet Computing* 2(4) (1998) 30-37.

[23] D. Lenat and R. Guha, *Building Large Knowledge Based Systems*, (Addison-Wesley, MA, 1990).

[24] S. Luke and J. Heflin, *SHOE 1.01, Proposed Specification*, at: http://www.cs.umd.edu/projects/plus/SHOE/spec.html (2000).

[25] S. Luke, L. Spector, D. Rager, and J. Hendler, Ontology-based Web Agents, in: *Proceedings of the First International Conference on Autonomous Agents* (Association of Computing Machinery, New York, NY, 1997) 59-66.

[26] R. MacGregor, The Evolving Technology of classification-based knowledge representation systems, in: J. Sowa, ed., *Explorations in the representation of knowledge* (Morgan-Kaufmann, CA, 1991).

[27] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, Lore: A Database Management System for Semistructured Data, *SIGMOD Record*, 26(3) (1997) 54-66.

[28] N. Noy and C. Hafner, C. The State of the Art in Ontology Design. *AI Magazine* **18**(3) (1997) 53-74.

[29] Y. Papakonstantinou, et al., A Query Translation Scheme for Rapid Implementation of Wrappers, in: *Proceedings of the Conference on Deductive and Object-Oriented Databases(DOOD)* Singapore (1995).

[30] M. Roth, and P. Schwarz, Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources, in: *Proceedings of 23rd International Conference on Very Large Data Bases* (1997).

[31] K. Sagonas, T. Swift, and D. S. Warren, XSB as an Efficient Deductive Database Engine, in: R. T. Snodgrass and M. Winslett, editors, *Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94)* (1994) 442-453.

[32] K. Stoffel, M. Taylor and J. Hendler, Efficient Management of Very Large Ontologies, in: *Proceedings of American Association for Artificial Intelligence Conference (AAAI-97)* (AAAI/MIT Press, 1997).

[33] J. Ullman, Principles of Database and Knowledge-Base Systems, (Computer Science Press, MD, 1988).

[34] J. Vega, A. Gomez-Perez, A. Tello, and Helena Pinto, How to Find Suitable Ontologies Using an Ontology-Based WWW Broker, in: *International Work-Conference on Artificial and Natural Neural Networks, IWANN'99, Proceeding, Vol. II*, Alicante, Spain (1999) 725-739.

[35] G. Wiederhold, Mediators in the Architecture of Future Information Systems, *IEEE Computer* 25(3) (1992).

Notes

[1]Prolog is not strictly declarative because the order of the rules determines how the system processes them.

[2]In actuality, SHOE has a separate namespace for each ontology, but one can assume that the symbols are unique because it is always possible to apply a renaming that appends a unique ontology identifier to each symbol.

[3]A definite program clause is a Horn clause that has at least one antecedent and exactly one consequent.

[4]This ontology is located at *http://www.cs.umd.edu/projects/plus/SHOE/onts/cs1.0.html*

[5] Those interested in the details of the ontology can view it at *http://www.cs.umd.edu/projects/plus/SHOE/onts/tseont.html*

```
<!-- Declare an ontology called "university-ont". -->
<ONTOLOGY ID="university-ont" VERSION="1.0">

<!-- Borrow some elements from an existing ontology, prefixed with a "g." -->
   <USE-ONTOLOGY ID="general-ont" VERSION="1.0" PREFIX="g"
                 URL="http://www.ontology.org/general1.0.html">

<!-- Create local aliases for some terms -->
   <DEF-RENAME FROM="g.Person" TO="Person">
   <DEF-RENAME FROM="g.Organization" TO="Organization">
   <DEF-RENAME FROM="g.name" TO="name">

<!-- Define some categories and subcategory relationships -->
   <DEF-CATEGORY NAME="Faculty" ISA="Person">
   <DEF-CATEGORY NAME="Student" ISA="Person">
   <DEF-CATEGORY NAME="Chair" ISA="Faculty">
   <DEF-CATEGORY NAME="Department" ISA="Organization">

<!-- Define some relations; these examples are binary, but relations can be n-ary too -->
   <DEF-RELATION NAME="advises">
      <DEF-ARG POS="1" TYPE="Faculty">
      <DEF-ARG POS="2" TYPE="Student">
   </DEF-RELATION>

   <DEF-RELATION "hasGPA">
      <DEF-ARG POS="1" TYPE="Student">
      <DEF-ARG POS="2" TYPE=".NUMBER">
   </DEF-RELATION>

<!-- Define a rule that states that the head of a Department is a Chair -->
   <DEF-INFERENCE>
      <INF-IF>
         <RELATION NAME="g.headOf">
            <ARG POS="1" VALUE="x" USAGE="VAR">
            <ARG POS="2" VALUE="y" USAGE="VAR">
         </RELATION>
         <CATEGORY NAME="Department" FOR="y" USAGE="VAR">
      </INF-IF>
      <INF-THEN>
         <CATEGORY NAME="Chair" FOR="x" USAGE="VAR">
      </INF-THEN>
   </DEF-INFERENCE>

</ONTOLOGY>
```

Figure 1: An example ontology.

```
<INSTANCE KEY="http://univ.edu/jane/">

<!-- Use the semantics from the ontology "university-ont", prefixed with a "u." -->
   <USE-ONTOLOGY ID="university-ont" VERSION="1.0" PREFIX="u"
                 URL="http://www.ontology.org/univ1.0.html">

<!-- Claim some categories for this instance and others. -->
   <CATEGORY NAME="u.Chair">
   <CATEGORY NAME="u.Student" FOR="http://univ.edu/john/">

<!-- Claim some properties and relationships  -->
   <RELATION NAME="u.name">
      <ARG POS="TO" VALUE="Jane Smith">
   </RELATION>

   <RELATION NAME="u.advises">
      <ARG POS="TO" VALUE="http://univ.edu/john/">
   </RELATION>

</INSTANCE>
```

Figure 2: An example instance.

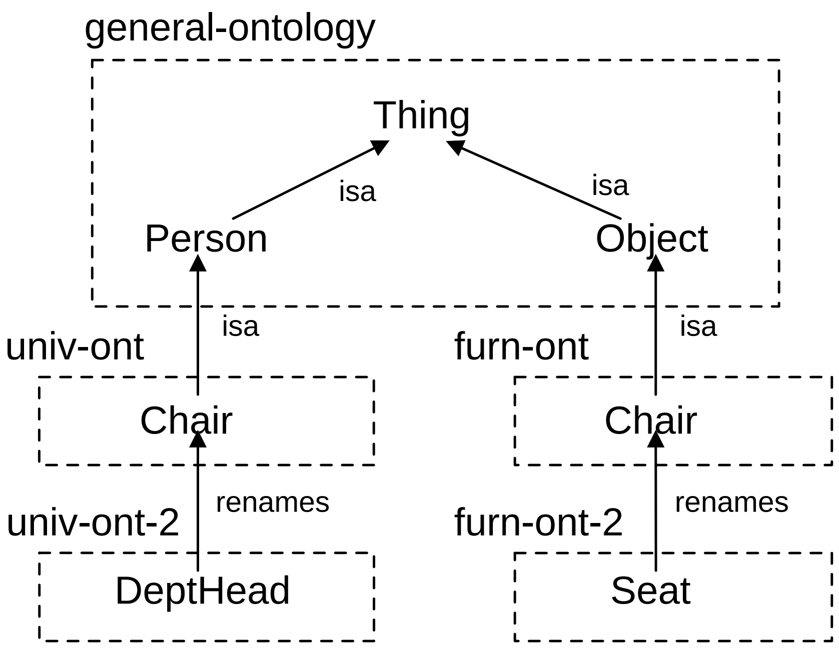

Figure 3: Interoperability is based on shared ontologies.

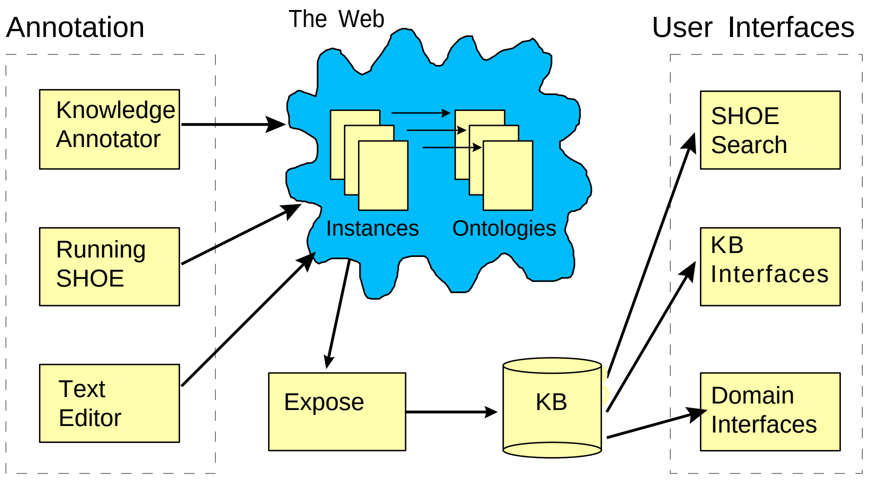

Figure 4: The SHOE architecture.