IJCAI'99 Workshop on
Intelligent Information Integration

Practical Knowledge Representation for the Web

Frank van Harmelen
Dept. of Maths & CS
Vrije Universiteit Amsterdam
frankh@cs.vu.nl

Dieter Fensel AIFB
University of Karlsruhe
dieter.fensel@aifb.uni-karlsruhe.de

Abstract

The lack of *semantic markup* is a major barrier to the development of more intelligent document processing on the Web. Current HTML markup is used only to indicate the structure and lay-out of documents, but not the document semantics.

Unfortunately, proposals from the AI community for Web-based knowledge -representation languages can hardly expect wide acceptance on the Web. Even if unpalatable for the AI community, the question should instead be how well AI concepts can be fitted into the markup languages that are widely supported on the Web, either now or in the foreseeable future.

We provide a survey and analysis of traditional, new, and arising Web standards and show how they can be used to represent machine-processable semantics of Web sources.

The results of this paper should help AI researchers and practitioners to apply their results to real Web documents, instead of basing themselves on AI specific representations that have no chance of becoming widely used on the Web.

1 Introduction

Currently, the World Wide Web (WWW) contains around 300 million static objects providing a broad variety of information sources [Bharat and Broder, 1998]. The early question of whether a certain piece of information is on the Web has become the problem of how to find and extract it. The problem will become even more serious when the growth of the Web maintains its high speed as expected by the W3C (the standardization committee of the WWW).

Artificial Intelligence has a strong tradition in developing methods, tools and languages for structuring knowledge and information. Therefore it is quite natural to apply its techniques to tackle the above problems. However, applying AI techniques directly to (semistructured) natural language documents is still not very promising. Employing the power of automated reasoning to guide access to information sources requires machine-processable representations of the semantics of these sources. In consequence, meta-data annotation of Web sources is essential for applying AI techniques on a large and successful scale. Taking a step in this direction is the purpose of our paper. Complementary to [Calvanese *et al.*, 1998] who look for a very expressive Description Logics for modeling semistructured data we rather take the opposite point of view. We provide a survey and analysis of traditional, new, and arising Web standards and show how they can be used to represent machine-processable semantics of Web sources having in mind that this area may become one of the killer applications of AI.

The importance of AI techniques in this area stems from the fact that finding the right piece of information is only one problem among serious other ones. In fact, four types of problems arise when dealing with large amounts of semistructured information:

— *Searching information*: Existing keyword-based search retrieves irrelevant information that uses a certain word in a different meaning or it may miss information where different words about the desired content are used.

— *Extracting information*: Currently human browsing and reading is required to extract relevant information from information sources since automatic agents miss all common sense knowledge required to extract such information from textual representations, and they fail to integrate information spread over different sources.

— *Maintaining* weakly structured text sources difficult and time consuming activity when such sources become large. Keeping such collections consistent, correct, and up-to-date requires mechanized representation of semantics and constraints that help to detect anomalies.

— *Automatic document generation* [Perkowitz and Etzioni, 1997] discuss the usefulness of adaptive Web sites which enable their dynamic reconfiguration according to user profiles or other aspects of relevance. Such generating of semistructured information presentations from semistructured data requires a machine-accessible representation of the semantics of these information sources.

In general, two alternative and complementary strategies are available to achieve this goal. First, one can enrich information sources *declaratively* with annotations that provide their semantics in a machine accessible manner. Second, one can write programs (filters, wrappers, extraction programs) that *procedurally* extract such semantics of Web sources. Clearly the declarative and the procedural approaches are complementary. The Procedural approach can be used to generate annotations for Web sources and existing annotations make procedural access to information much easier. In this paper, we will focus on the first approach, i.e., on declarative representations of semantics, and refer the reader to [Muslea, 1998] for a survey on wrapper generation and other procedural information extraction techniques.

The content of the paper is organized as follows. In section 2, we describe existing languages for annotating Web sources with semantics. We analyze HTML (the ⟨META⟩-tag and the ⟨SPAN⟩-tag), style-sheet mechanisms, XML and RDF. In section 3 we analyse syntactic features of these languages, such as the possibility to avoid information duplication and to exploit scoping. Section 4 takes a more KR point of view and analyzes the modeling primitives of these languages for factual knowledge, terminological knowledge and inferential knowledge. Section 5 concludes the paper by providing a summary and an outlook.

2 Existing semantic markup-languages

In this section we will discuss different ways in which semantic markup can be added to Web-pages using W3C technology.

2.1 HTML-based semantic markup

HTML ⟨META⟩-tags

Historically the first attempt at representing semantic aspects inside Web-documents are the HTML META-tags. Their intended use is limited to stating global properties that apply to the entire document, for example:

```
⟨HEAD⟩
  ⟨META NAME="AUTHOR" CONTENT="FRANK"⟩
⟨/HEAD⟩
```

This expresses that the author of the entire document is `Frank`.

Although unintended, the META-tag mechanism can be stretched to allow statements about specific *parts* of the text, instead of only properties applying globally to the *entire* text. This relies on using the anchor mechanism of HTML (underline added for emphasis only):

```
⟨HEAD⟩
  ⟨META NAME="AUTHOR"   CONTENT="#L0"⟩
  ⟨META NAME="LOCATION" CONTENT="#L1"⟩
  ⟨META NAME="TEL"      CONTENT="#L2"⟩
  ⟨META NAME="ROOM"     CONTENT="#L3"⟩
⟨/HEAD⟩
⟨BODY⟩
  This page is written by
  ⟨SPAN ID="L0"⟩Frank van Harmelen⟨/SPAN⟩.
  ⟨SPAN ID="L1"⟩
    His tel.nr. is ⟨SPAN ID="L2"⟩47731⟨/SPAN⟩,
    room nr. ⟨SPAN ID="L3"⟩T3.57⟨/SPAN⟩
  ⟨/SPAN⟩
⟨/BODY⟩
```

This states that contents of the type indicated by the NAME attribute of the ⟨META⟩-tag (`AUTHOR`, `LOCATION`, etc) can be found at the specified location in the document.

This is stretching the META-tag mechanism beyond its original limits: the above use of anchors in META-tags is not standardised. It can be exploited in software if one wishes to, but it cannot be relied upon to be treated by standard Web-browsers, search-engines, etc.

The SHOE research project [Luke *et al.*, 1997] proposes essentially an extension of the HTML ⟨META⟩-tag concept. The fact that SHOE expressions can occur in both ⟨HEAD⟩ and ⟨BODY⟩ of a document is unimportant; what matters is that (like HTML ⟨META⟩-tags), SHOE expressions are separate from the contents of a document, and apply to the entire document[1]. Whereas HTML ⟨META⟩-tags are limited to attribute-value pairs, SHOE expressions include arbitrary relations between instances:

```
⟨INSTANCE KEY="HTTP://WWW.CS.VU.NL/~FRANKH"⟩
  ⟨RELATION NAME="FULLNAME"⟩
   ⟨ARG POS=TO VALUE="FRANK VAN HARMELEN"⟩
  ⟨/RELATION⟩
  ⟨RELATION NAME="AUTHOR-OF"⟩
   ⟨ARG POS=TO VALUE="HTTP://..URL OF THIS DOC.."⟩
  ⟨/RELATION⟩
  ⟨RELATION NAME="TEL"⟩
   ⟨ARG POS=TO VALUE="47731"⟩
  ⟨/RELATION⟩
  ⟨RELATION NAME="ROOM"⟩
   ⟨ARG POS=TO VALUE="T3.57"⟩
  ⟨/RELATION⟩
⟨/INSTANCE⟩
This page is written by Frank van Harmelen.
His tel.nr. is 4773, room nr. T3.57.
```

The URL `http://www.cs.vu.nl/ frankh` is the identifier for the person Frank and used to describe his properties. The syntax `POS=TO` states that the relations apply to the current instance (i.e. the person Frank).

HTML ⟨SPAN⟩-elements

According to the HTML 4.0 specification, the ⟨SPAN⟩ element "is a generic container of any text element offering a generic mechanism for adding structure to documents" Using the standard `CLASS` attribute, the same semantic markup as above can now be written as follows:

```
⟨BODY⟩
  This page is written by
  ⟨SPAN CLASS="AUTHOR"⟩Frank van Harmelen⟨/SPAN⟩.
  ⟨SPAN CLASS="LOCATION"⟩
    His tel.nr. is ⟨SPAN CLASS="TEL"⟩47731⟨/SPAN⟩,
    room nr. ⟨SPAN CLASS="ROOM"⟩T3.57⟨/SPAN⟩
  ⟨/SPAN⟩
⟨/BODY⟩
```

Although intended for specifying layout, the HTML-4.0 reference document already suggests the use of the ⟨SPAN⟩-tag to express semantic structure of a document, so this use of the ⟨SPAN⟩-tag should not be considered as inappropriate

The markup-scheme used in Ontobroker [Decker *et al.*, 1999] is based on the same idea as the HTML ⟨SPAN⟩-tag approach, but uses the HTML anchor tag ⟨A⟩ instead of the ⟨SPAN⟩-tag for the same purpose.

Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) aim to separate the structure of a document from a specification of the layout of the doc-

[1] OML (http://wave.eecs.wsu.edu/CKRMI/OML.html) is the encoding of (a suitably modified) SHOE in XML. One main difference is that the distributed SHOE markup needs to be gathered and extracted, since it is embedded in HTML pages; whereas the OML files only point to HTML files, and can either be distributed or centralized. Since OML is separate, the legacy HTML need not be modified.

ument. Particular document elements can be formatted as specified in style information:

```
⟨HEAD⟩
  ⟨STYLE⟩ P.TECHNOTE {font-size: 50%} ⟨/STYLE⟩
⟨/HEAD⟩
⟨BODY⟩
  ⟨P CLASS="TECHNOTE"⟩a note in a small font⟨/P⟩
⟨/BODY⟩
```

specifies that paragraphs from the class TECHNOTE should be set in a smaller font.

Although originally intended for layout information, the ⟨STYLE⟩-mechanism can also be used (abused?) for adding semantic information:

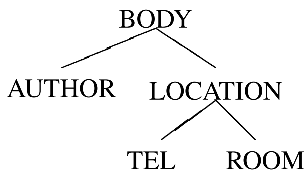
```
⟨HEAD⟩
  ⟨STYLE⟩
   SPAN.L0  {contents: author}
   SPAN.L1  {contents: location}
   SPAN.L2  {contents: tel}
   SPAN.L3  {contents: room}
  ⟨/STYLE⟩
⟨/HEAD⟩
⟨BODY⟩
  This page is written by
  ⟨SPAN CLASS="L0"⟩Frank van Harmelen⟨/SPAN⟩.
  ⟨SPAN CLASS="L1"⟩
    His tel.nr. is ⟨SPAN CLASS="L2"⟩47731⟨/SPAN⟩,
    room nr.⟨SPAN CLASS="L3"⟩T3.57⟨/SPAN⟩
  ⟨/SPAN⟩
⟨/BODY⟩
```

2.2 XML

One of the results of a general push towards more semantic structure on the Web has been the development of the XML markup language[2]. XML allows Web-page creators to use their own set of markup-tags. These tags can be chosen to reflect the domain specific semantics of the information, rather than merely its lay-out.

```
⟨BODY⟩
  This page is written by
  ⟨AUTHOR⟩Frank van Harmelen⟨/AUTHOR⟩.
  ⟨LOCATION⟩
    His tel.nr. is ⟨TEL⟩47731⟨TEL⟩,
    room nr. ⟨ROOM⟩T3.57⟨/ROOM⟩
  ⟨/LOCATION⟩
⟨/BODY⟩
```

In essence, XML allows us to structure Web-pages as labelled trees[3], where the labels can be chosen by the information provider to reflect as much of the documents semantics as is required. The labelled tree for the above XML-code is shown below:



Although XML allows the use of any tags as long as they are properly nested in the document, it is possible to define restrictions on the set of tags that can be used in document. This is done in a Document Type Definition (DTD), which expresses in a grammar-like formalism which allowed sequences and nestings of tags are allowed in a document.

2.3 RDF(S)

The third and final W3C-supported semantic markup-scheme that we will discuss is RDF (currently a W3C proposed recommendation).

XML provides semantic information as a by-product of defining the structure of the document. XML prescribes a tree structure for documents and the different leaves of the tree have a well-defined tag and context the information can be understood with. That is, structure and semantics of document are interwoven.

The *Resource Description Framework* RDF [Lassila and Swick, 1998] provides a means for adding semantics to a document without making any assumptions about the structure of the document. It is an XML application (i.e., its syntax is defined in XML) customized for adding meta information to Web documents. It is currently under development as a W3C standard for content descriptions of Web sources and will be used by other standards such as PICS-2, P3P, and DigSig.

The data model of RDF provides three object types: resources, property types, and statements[4]

- A *resource* is an entity that can be referred to by a address at the WWW (i.e., by an URI). Resources are the elements that are described by RDF statements.
- A *property* defines a binary relation between resources and/or atomic values provided by primitive datatype definitions in XML.
- A *statement* specifies for a resource a value for a property. That is, statements provide the actual characterizations of the Web documents.

A simple example is

```
Author(http://www.cs.vu.nl/~frankh) = Frank[5]
```

This states that the author of the named Web document is Frank. Values can also be structured entities:

```
Author(http://www.cs.vu.nl/~frankh) = X
Name(X) = Frank
Email(X) = frankh@cs.vu.nl
```

where X denotes an actual (i.e., the homepage of Frank) or a virtual URI. In addition, RDF provides bags, sequence, and alternatives to express collections of Web sources. Finally,

[2]Strictly speaking, XML is a markup meta-language, but we will follow common practice and ignore this difference.

[3]Using shared identifiers in the XML attribute/value mechanism it is possible to encode arbitrary graphs as XML trees, but this does not change the fact that the lexical structure of an XML document remains a tree.

[4]In the most recent RDF drafts resource are called *subjects*, properties are *predicates*, and statements are *objects*.

[5]We skip the awkward syntax of RDF because simple tooling could easily present it in a more common format such as shown here.

RDF can be used to make statements about RDF-statements, i.e. it provides meta-level facilities:

```
Claim(Dieter)=(Author(http://www.cs.vu.nl/~frankh)
                 =
                 Frank)
```

states that Dieter claims that Frank is the author of the named resource.

[Brickley *et al.*, 1998] provide a basic type schema for RDF (called RDFS during the following) based on core classes, core property types and core constraints. Three core classes are provided by the RDF Schema machinery: *Resource* (i.e., the class of all objects), *Property Type* (i.e., the class of all binary relations), and *Class* (i.e., the class of all types). Two core property types are provided: *instanceOf* and *subClassOf*. *instanceOf* defines a relation between a resource and an element of Class and *SubClassOf* defines a relationship between two elements of Class. *SubClassOf* is supposed to be transitive. *Constraint* is a subclass of *Property Type* and has the two core instances *range* and *domain* applicable to property types having a class as value. *Range* and *domain* define the range and domain of property types respectively.

3 A Symbol-level comparison

In this section we discuss some syntactic and pragmatic requirements which Web-based markup languages must satisfy in order to be a practical basis for Knowledge Representation on the Web. We will also indicate how well each of the markup-schemes described above scores on these requirements.

3.1 Supported by Web technology

No matter how nice any KR representation language is as proposed by the AI community, the real Web can hardly wait until Netscape and Microsoft decide to support such a language. Even if unpalatable for the AI community, the order of precedence is the other way round: how well can AI concepts be fitted into the markup languages that are widely supported on the Web, either now or in the foreseeable future.

Unfortunately, this requirement disqualifies a lot of current research aimed at applying AI techniques to the Web. Instead, many of the markup-schemes described above, are already (or will soon be) widely supported, with the exceptions of SHOE and Ontobroker (both of which are syntactic varieties of schemes that are supported).

3.2 Avoiding duplication

A basic tenet of information modelling is that redundancy inevitably leads to inconsistency. It is therefore unfortunate that some of the above markup-schemes enforce a duplication between information for semantic purposes and information for rendering. Of course, no syntax would be able to avoid the *possibility* of stating redundant information, but we would prefer a syntax that does at least not *necessitate* such redundancy. Consider the following HTML ⟨META⟩-tag scheme:

```
⟨HEAD⟩
  ⟨META NAME="AUTHOR" CONTENT="FRANK"⟩
⟨/HEAD⟩
⟨BODY⟩
 This page written by Dieter.
⟨/BODY⟩
```

Although this can be avoided by the use of anchors as shown above, this goes at the cost of browser support for that non-standard mechanism.

The SHOE markup-scheme suffers from the same drawback: meta-information is stated separately and duplicates the rendered contents of the document.

XML is more attractive in this respect, since it uses the *same* information for both rendering and semantic purposes:

```
⟨BODY⟩
 This page written by
 ⟨AUTHOR⟩Dieter⟨/AUTHOR⟩.
⟨/BODY⟩
```

The same effect is obtained using HTML ⟨SPAN⟩-tags:

```
 This page written by
 ⟨SPAN CLASS="AUTHOR"⟩Dieter⟨/SPAN⟩.
```

and similarly for style-sheets. Ontobroker also manages to avoid duplication, but at the price of using non-standard HTML:

```
 This page written by
 ⟨A ONTO="TAG[AUTHOR=BODY]"⟩Dieter⟨/A⟩.
```

where `body` is a reserved word referring to the text contained in the ⟨A⟩-tag (ie. `Dieter`).

A strength of RDF is the *decoupling* of the structure of the document and the structure of the meta-information. RDF makes no strong assumptions on the internal structure of the document that it provides meta-data for (unlike XML, which assumes that the document itself is structured as a labelled tree). As a result, RDF is forced to duplicate information, since it cannot assume that the meta-information is already present in the document itself. XML can avoid this duplication at the price of interleaving document structure and -contents with meta-information.

3.3 Allowing nesting

Nesting of expressions is a familiar device in language design to achieve scoping. For example, the following

```
Frank's tel. nr is 47731,
Dieter's tel. nr is 6921.
```

tells us not only that we are dealing with two names and two telephone numbers, but also which number belongs to which name. Such nesting can be trivially expressed in XML:

```
⟨PERSON⟩
  ⟨NAME⟩Frank⟨/NAME⟩'s tel. nr is
  ⟨TEL⟩47731⟨/TEL⟩
⟨/PERSON⟩,
⟨PERSON⟩
  ⟨NAME⟩Dieter⟨/NAME⟩'s tel. nr is
  ⟨TEL⟩6921⟨/TEL⟩
⟨/PERSON⟩.
```

and similarly by using HTML ⟨SPAN⟩-tags. Because the markup with style-sheet exploits the ⟨SPAN⟩-tag, nesting is also possible with that approach.

The standard use of HTML ⟨META⟩-tags cannot express such nesting. The Ontobroker markup can also not express such nesting, since it exploits the ⟨A⟩-tag, which cannot be nested.

More surprisingly perhaps, even a supposedly sophisticated and carefully designed language like RDF is incapable of expressing this nesting in a natural way. RDF only provides binary relations, and anything else (n-ary relations, hierarchies, etc) must all be simulated using binary relations. This quickly becomes very cumbersome. Even the trivial example of nesting given above becomes hard when simulated with binary relations only:

```
⟨A NAME="PERSON1"⟩Frank's tel. nr is 47731⟨/A⟩,
⟨A NAME="PERSON2"⟩Dieter's tel. nr is 6921⟨/A⟩.

Name(#person1) = Frank
Tel(#person1)  = 47731
Name(#person2) = Dieter
Tel(#person2)  = 6921
```

The markup-scheme of SHOE (also based on binary relations) suffers from the same problems.

3.4 Summary

The above considerations can be summarised in the following table. (The first group concerns traditional Web technology, the second group new Web technology, and the third group the proposals originating from AI).

	Web support	avoiding duplication	allowing nesting
HTML ⟨META⟩	+	-	-
HTML ⟨SPAN⟩	+	+	+
CSS	+	+	+
XML	+	+	+
RDF (S)	+	-	-
SHOE	-	-	-
Ontobroker	-	+	-

4 A knowledge-level comparison

Besides the syntactic and pragmatic requirements investigated in the previous section, we can also analyse the various markup-schemes on their underlying modelling primitives and their expressive power. This is the purpose of the current section.

4.1 Factual Knowledge: Data models

The data-models underlying the various markup schemes vary greatly:

— HTML ⟨META⟩-tags only provide a basic **attribute-value mechanism** .

— Both XML and HTML ⟨SPAN⟩-tags (and therefore style-sheets because they rely on the ⟨SPAN⟩-tag) take **labelled trees** for their basic data-model. (Although we have not shown this in our examples, attribute-value pairs can be associated with each node in such a labelled tree).

— In Ontobroker the situation is somewhat complicated: The value of Ontobroker's non-standard `onto`-attribute attribute is itself an expression in another language, namely an **expression in F-logic** [Kifer *et al.*, 1995]. As a result, Ontobroker's has access to F-logic's rich data model, consisting of classes, attributes with domain and range definitions, is-a hierarchies with set inclusion of subclasses and multiple attribute inheritance.

— RDF's data model is based on **binary relations**, enhanced with a reification mechanism to enable relations between relations. RDFS uses this basic data model to build a basic object-oriented type schema on top of RDF.

— SHOE's data model is similar to that of RDFS, but can express **n-ary relations** instead of only binary relations. It does not include the reification mechanism of RDF. Although not shown in our example above, SHOE allows the specification of classes with attributes, with multiple inheritance of attributes between classes.

Although RDFS, Ontobroker and SHOE all provide an object-oriented type schema, there is however an important problem with RDF/RDFS, when compared to Ontobroker and SHOE: Contrary to object-oriented and frame-based approaches RDFS is *property centric*. Properties are not defined as attributes of object classes but as relations that link two object classes. This has the consequence, that properties have a global domain and global range definition whereas object-oriented systems may refine the domain and range definitions of an attribute by a subclass that inherits this properties and adds additional type constraints. Also it is not possible that different object classes use the same property name with different domain and value restrictions.

4.2 Terminological knowledge: ontologies

Modern Knowledge Representation and Knowledge Engineering advocates the use of explicit ontologies CYC [Lenat and Guha, 1990], KIF [Genesereth, 1991], Ontolingua [Gruber, 1993]). Ontologies are a specification of the conceptualisation and the corresponding vocabulary used to describe a domain. Roughly, ontologies correspond to generalised database schemas. However, ontologies can be used to describe the structure of semantics of much more complex objects than common databases and are therefore well-suited for describing heterogeneous, distributed and semistructured information sources such as found on the Web.

It is therefore important that any semantic markup-scheme for the Web supports the notion of an explicitly specified ontology.

— the HTML-based approaches fall short in this respect: neither the plain attribute-value data-model of ⟨META⟩-tags nor the labelled trees underlying the ⟨SPAN⟩-tags allow that their data-schema is explicitly and separately specified. Style-sheets provide an explicit listing of the available ontological categories, but such a flat list of category-names is not a full-blown ontology.

— HTML-derived approaches such as SHOE and Ontobroker do provide explicit ontologies, albeit in very different ways. In SHOE, ontologies can be defined by information-providers themselves inside their own HTML pages (using again a special-purpose extension to HTML). Such an ontology contains a class-lattice and possible relations between instances of these classes. Ontobroker ontologies are similar in nature (a class-hierarchy, attributes with domain and range definitions, and multiple attribute inheritance), but an essential difference is that Ontobroker relies on a single centrally defined ontology, whereas SHOE allows for local definitions of ontologies (or local extensions of central ontologies). The merits of these different approaches are unclear: obviously a central ontology will quickly become a bottle-neck in Web-based distributed information modelling; on the other hand, unchecked creating, extending and mixing of ontologies will just as quickly create the same problems on the ontological level that now exist on the level of the information itself.

— The closest thing that XML offers for ontological modelling is the Document Type Definition (DTD) which defines the legal nestings of tags in a document. At first sight, the nesting of tags as illustrated in section 2.2 would seem to coincide with the notion of an ontological hierarchy, but this is in fact not the case: a DTD specifies the legal *lexical* nesting in a document, which may or may not coincide with any *ontological* hierarchy (subclass or part-of) of a given domain. For example, an XML DTD may state that ⟨AUTHOR⟩ may be nested inside ⟨BOOK⟩ or the other way round, but no ontological relationship between authors and books can be inferred from either nesting. What is represented in an XML-tree are the attributes defined for classes (as can be seen from the same figure), but this is done in a very weak way: no range restrictions on attribute values can be stated, and because of the absence of a class-hierarchy, the usual inheritance mechanism is also missing. Work on XML-schema [Malhotra and Maloney, 1999] may well contribute to bridging the gap between DTD's and ontologies.

— RDFS is not directly an annotation formalism but rather provides the vocabulary used for annotation. That is, it can be used to describe what is called an ontology in SHOE and Ontobroker. In RDFS, properties are defined globally and are not encapsulated as attributes in class definitions. Therefore, an ontology expressed in Ontobroker can only be expressed in RDFS by reifying the property names with class name suffixes. This is a rather disappointing feature which ignores all of the lessons from object-oriented modelling in the past decade or more.

4.3 Inferential knowledge

In this section we analyse the extent to which inferential knowledge can be expressed in the various markup-schemes. As a simple example of such inferential knowledge, we can take the subsumption relationship between authorship and co-authorship. From the following document:

```
This document is written by ⟨AUTHOR⟩Frank⟨/AUTHOR⟩
together with ⟨CO-AUTHOR⟩Dieter⟨CO-AUTHOR⟩
```

any human reader will infer that Dieter is also author of the document, since any co-author is also an author. For truly intelligent Web-applications, it is necessary that this knowledge is available in machine accessible form.

Of all the markup-schemes discussed above, only SHOE and Ontobroker (precisely the languages originating from AI research groups) allow to express such inferential knowledge. SHOE allows to state pure Horn rules inside local Web pages, while Ontobroker only states this inferential knowledge centrally (similar to the respective decisions on the terminological knowledge). Ontobroker allows a larger fragment of first-order logic to be used, namely exactly the fragment which can be translated to stratified normal logic programs via the Lloyd-Topor transformations [Lloyd and Topor, 1984]

4.4 Summary

The comparison on knowledge-level features can be summarised (in a very abbreviated form) in the following table.

	facts	terminology	inference
HTML ⟨META⟩	-	-	-
HTML ⟨SPAN⟩	+	-	-
CSS	+	-	-
XML	+	±	-
RDF (S)	+	±	-
SHOE	++	++	+
Ontobroker	++	++	++

5 Conclusions

We have provided a survey and analysis of traditional, new, and arising Web standards and show how they can be used to represent machine-processable semantics of Web sources.

Our comparison, summarised in the two tables above, is not meant to suggest that we are hoping for a single language that will solve all problems at all of the above levels in an satisfactory way. On the contrary, we expect that different languages will emerge that will together provide appropriate solutions, each with its own specific intended use. Instead, the above comparison is meant as an inventory on which aspects each of the currently available languages on the Web scores well or not.

The main conclusions we can draw from this are as follows:

Looking at the syntactic design of the various languages, it is rather surprising to see how well HTML ⟨SPAN⟩-tags compare with more novel approaches such as XML. One of the surprises to us when writing this paper was that the HTML ⟨SPAN⟩-mechanism already provided much of the functionality now so loudly advertised for XML.

Furthermore, it is rather disappointing to see that RDF ignores a few basic lessons in language design.

Looking at the semantic side of these languages, it is no surprise that traditional technologies (⟨META⟩, ⟨SPAN⟩) are not rich enough in this respect, but it is rather disappointing that also the new Web technologies (XML, RDF) fail to deliver, with little support for ontologies, and no support for inference.

For applying AI in *realistic*, *large-scale* Web applications, all this implies the following: from a syntactic and technological point of view, we can use the well-supported HTML ⟨SPAN⟩-tag, possibly gradually migrating to XML when support for it grows.

From a semantic perspective, RDF is unfortunately not going to provide us with what is required, and more input from the AI community is needed in the development of future Web-standards, in particular concerning the representation of ontological and inferential knowledge.

Comparing the two summary tables, the two markup-schemes from an AI background score lower on symbol-level design, but they are much stronger on knowledge-level features. It would seem that a combination of features is called for.

References

[Bharat and Broder, 1998] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public Web search engine. In *7th Int. WWW Conf.*, pg. 379–388.

[Brickley *et al.*, 1998] D. Brickley, R. Guha, A. Layman (eds.). Resource description framework (RDF) schema specification. W3C Working Draft, August 1998. http://www.w3c.org/TR/WD-rdf-schema

[Calvanese *et al.*, 1998] D. Calvanese, G. De Giacome, and M. Lenzerini. What can knowledge representation do for semi-structured data. In *AAAI'98*.

[Decker *et al.*, 1999] S. Decker, M. Erdmann, D. Fensel, and R Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al., editor, *Semantic Issues in Multimedia Systems*. Kluwer, Boston, 1999.

[Genesereth, 1991] M. Genesereth. Knowledge Interchange Format. In *KR'91*, pages 238–249.

[Gruber, 1993] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

[Lassila and Swick, 1998] O. Lassila and R. Swick. Resource description framework (RDF). W3C proposed Recommendation, January 1999. http://www.w3c.org/TR/WD-rdf-syntax.

[Lenat and Guha, 1990] D. B. Lenat and R. V. Guha. *Building large knowledge-based systems. Representation and inference in the Cyc project*. Addison-Wesley, Reading, Massachusetts, 1990.

[Lloyd and Topor, 1984] J.W. Lloyd and R.W. Topor. Making Prolog more expressive. *J. Logic Programming*, 1(3):225–240, 1984.

[Luke *et al.*, 1997] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web agents. In *First Int. Conf. on Autonomous Agents (AA'97)*, 1997.

[Malhotra and Maloney, 1999] Ashok Malhotra and M. Maloney. XML-Schema Requirements. W3C Note, February 1999. http://www.w3.org/TR/NOTE-xml-schema-req.

[Muslea, 1998] I. Muslea. Extraction patterns: from information extraction to wrapper generation. Technical report, ISI-USC, 1998.

[Perkowitz and Etzioni, 1997] M. Perkowitz and O. Etzioni. Adaptive Web sites: an AI challenge. In *IJCAI'97*.