Semantic verification of Web sites using Natural Semantics

Thierry Despeyroux **Brigitte Trousse**

Inria - B.P. 93 - 06902 Sophia Antipolis Cedex, France
(Thierry.Despeyroux Brigitte.Trousse)@inria.fr

Abstract

The huge amount of information and knowledge available on the Web leads to the fact that it is more and more difficult to manage this information. Two different ways are commonly explored: giving a syntactical structure to Web sites, and annotating their content to facilitate Web mining. In this paper we explore a different approach inherited from software engineering: specifying the semantics of Web sites, allowing semantic verifications that will help both the conception and the maintenance of Web sites. To achieve this goal, we have experimented with the application of Natural Semantics (traditionally used to specify the semantics of programming languages) to Web sites specification and verification.

1 Introduction

Conceiving and maintaining a Web site is a difficult task. It is far simpler to discover inconsistent information than a well maintained site on the Internet. Our goal is to study and construct the tools that are necessary to design, produce, and maintain complex and coherent Web sites. Most efforts done in this domain concern the syntactical structure of Web sites, leading to XML. But only a little part of the semantics can be handle by syntactical constraints, and we would like to address all the semantics of a Web site.

After introducing the current possibilities of representing semantics of Web pages and more generally Web sites (using HTML and XML), we present our main objective related to supporting the designers and the web-masters in specifying and verifying semantically Web sites. This problem of adding semantics is more and more addressed by the different working groups of the W3C (see RDF (W3C-RDF, 1999; W3C-RDF-Schema, 1999), XML Schema (W3C-XML-Schema, 2000)...) and also by the ontological approach issued from AI researchers (SHOE (Heflin, Hendler, and Luke, 1999), On2broker (Fensel et al., 1998b)). But the main motivation of such works is to improve information retrieval providing a better indexing.

Our motivation is slightly different as we want to help in designing, specifying and checking Web sites. Very few works address the semantic verification of web pages. Two of them are WebMaster (van Harmelen and van der Meer, 1999; van Harmelen and Fensel, 1999), and works by Psaila and Crespi-Reghizzi (1999) that uses attribute grammars.

Our approach is inspired from previous works done in semantics of programming languages, drawing a parallel between the syntax of programming languages and the structure of Web sites (or semi-structured documents), and between the semantics of programs and the semantics of Web sites, applying some notions of types and semantic rules to documents on the Web. To achieve this goal, we have used the Centaur system (a generic programming environment generator, http://www.inria.fr/croap/centaur/centaur.html) and its semantics specification formalism Typol to construct a prototype of a Web site verification system by means of inference rules using natural semantics (Despeyroux, 1987; Kahn, 1987; Despeyroux, 1988; Borras et al., 1988).

We illustrate this method by applying it on two examples of Web sites, a thematic directory (like Yahoo) and an institutional site. The use of natural semantics shows clearly the difference between syntactical checking (for example verifying a page against a DTD, like in an XML validator) that is context free, and a semantical computation that is context dependent. The example of thematic directory shows the possibility of using external resources tools (thesauri, ontologies).

2 Semantics of Web Sites

Specifying semantics of Web sites or adding semantics to semi-structured documents is an effort that is widely recognised to be very useful to the community for facilitating information retrieval but more recently for the design of Web sites. However, what is semantics in such a context ?

Semantics appear in fact at different levels. First, a Web site is a way of presenting some information. To present this information, one will use a computer language like HTML or XML. This language has its own semantics. Second, one will want to have a model of knowledge (as ontologies), that can be represented by means of a computer language. Again this language has its own semantics. Third, one can annotate a Web document by a reference to a model of knowledge, and again will use a specific language and its semantics. The hole picture is darkened by the fact that we tend to use a universal language, namely XML, to represented very different things. More than an extensible language, XML can be viewed as a meta-language that allows to create new languages. If one has the possibility of defining these new languages using DTDs, the real problem is to define their semantics. This is not done in general and we want to address this specific problem.

Following the parallel with programming languages, one will traditionally speak of the static semantics of a programming language to express the constraints that a legal program must verify (this will very often correspond to compile-time verifications) in opposition with the dynamic semantics that express what happens at run-time. In this paper, we are concerned only by the static semantics of Web sites i.e. we want to be able to define the semantic constraints that a particular Web site must follow to be able to verify that some documents are not only well formed and valid but also correct in some sense. The dynamic semantics of a Web site could be defined as the way users navigate in the site, speaking of the behaviour of users (Trousse, Jaczynski, and Kanawati, 1999; Trousse, 2000), and is not in the scope of this paper.

We try now to give a survey of different languages and systems that are available and take into account the semantics for Web documents.

2.1 Representing Web Documents

In HTML, surely the most popular language on the Web, one manipulates some structured formated text but mixing in some way the structure of texts and their presentation. Semantics may appear, as we will see later, in meta-data attached to the documents, or be disseminated in the text by the use of `<SPAN>` tags, as explained in (van Harmelen and van der Meer, 1999; van Harmelen and Fensel, 1999).

XML (W3C-XML, 1998) not only gives the possibility to extend the formalisms of tags, but gives a way to clearly separate the syntactical structure of a document (its tree form) from the presentation by means of style-sheets. Trees can be attributed by text. By adding DTD[1] to XML, one add syntactical constraints concerning the structure of the document itself. So we can

[1]DTD : Data Type Document http://www.w3.org/TR/2000/WD-xml-c14n-20000119.html

manipulate annotated typed trees.

2.2 Annotating Web Documents

RDF (W3C-RDF, 1999) and the RDF Schema language (W3C-RDF-Schema, 1999), as recently proposed by the World Wide Web Consortium (W3C), provide a powerful framework to formalise meta-data as acyclic directed labelled graphs where nodes represent resources and arcs represent named properties.

SHOE (Heflin, Hendler, and Luke, 1999) is an HTML(XML)-based knowledge representation language which adds the tags necessary to embed arbitrary semantic data into Web pages. SHOE tags are divided into two categories. First, there are tags for constructing ontologies i.e sets of rules which define what kinds of assertions SHOE documents can make and what these assertions mean. Second, there are tags for annotating Web documents.

On2broker (Fensel et al., 1998; Fensel et al., 1998b) uses formal ontologies to extract, reason and possibly generate meta-data in the format of the Resource Description Framework (RDF).

2.3 Manipulating Web Documents

We can distinguish at least two ways of creating coherent Web documents. A first one is to generate documents from a data base or an other document, the second one is to edit documents that are then checked against some declarations. We focus here on the second possibility.

XSL (W3C-XSL, 2000), initially designed to format document, is in fact more powerful than that as it can be used to translate documents from one XML syntax to an other one. It works on attributed typed trees, with access to the context (mainly information accessible from the root of the tree to the "current" point) and gives the possibility to construct new attributed typed trees.

In a few words, what is mainly possible today for manipulating Web documents is context-free, expressing semantics by use of syntactical constraints. For example, in XML a DTD validator verifies that an XML document respects the structure specified in a DTD, if this one is present.

However others works try to go a step further proposing context-dependant document manipulation for semantic verification of Web documents. If we pursue our comparison between a Web site and a program, we can specify the syntax of the language that is used (a DTD is a context-free grammar), we can define pretty printers and syntactical translators, but we cannot perform global computations. The need for global computation as been studied in WebMaster by van Harmelen and van der Meer (1999) and also by Psaila and Crespi-Reghizzi (1999) who extend a DTD to create attributes that are manipulated by an attributed grammar evaluator.

Webmaster (van Harmelen and van der Meer, 1999) addresses semantic verification of Web sites and proposes a constraint language for representing integrity constraints for HTML or XML documents (for example, a publication on a page of a member of the group must also be included in the publication list of the entire group).

SiRLi, developed in the Ontobroker[2] project, is a logic-based RDF interpreter, able to reason with meta-data in the XML serialisation of RDF. Ontobroker, as mentioned in (Fensel et al., 1998b) could support weakly maintenance of structured text sources and detect incorrectness i.e. inconsistencies among documents and external sources. Such a support could be offered by integrating the inference of Webmaster or by using the existing inference engine in a different

[2]http://ontobroker.aifb.uni-karlsruhe.de/

way (and the type system). Such a tool could suggest to add some new meta-data according to the ontology specification.

Psaila and Crespi-Reghizzi (1999) propose the use of attributed grammars to make semantic computations on Web documents. Attributes of the grammar are saved in XML attributes, implying a modification of the syntax.

As Webmaster or the attributed grammar approach, we want to allow global computations overpassing some of their limitations as we want to be able to manipulate a context containing external or computed information.

3 Applying Natural Semantics to Structured Documents

In this section, we first give a quick overview of Natural Semantics, before explaining how we can apply it to Web sites.

3.1 What is Natural Semantics?

Our goal in this section is not to give a complete description of Natural Semantics, but to give a sufficient number of elements to show how it can be applied to Web sites.

Natural Semantics (Kahn, 1987) has been developed to specify the semantics of programming languages to allow mechanical manipulations of programs and generation of compilers. It aims at two main goals: providing an executable semantic specification formalism (Despeyroux, 1987) and allowing proofs (Despeyroux, 1986).

Programs are structured documents that can be represented as trees. The set of syntactical constraints that legal programs must respect is called the abstract syntax of the programming language. If we thought of an XML document as a tree, we can identify the notion of abstract syntax with the notion of DTD, even if the expressiveness of declaring an element is less powerful that giving the signature of an abstract syntax operator (Abstract syntaxes traditionally define operators and types, while only operators, called elements, are defined in a DTD). In both cases these constraints are context free.

Natural Semantics has its origin in the structural operational semantics developed by Plotkin (1981). Only the purely natural deduction part of it has been retained, and the style of the definition is closed to the natural deduction style, using Gentzen's sequents (Szabo, 1969). The natural semantics of a programming language or more generally of a computer formalism is given by one (or more) formal systems (sets of axioms and inference rules) which define sets of valid theorem (theories). Basically, these inference rules express a way to navigate in a program or in a document, checking that some properties (called consequents) are true. But, in opposition to syntactical constraints, these properties can most of the time only been checked in a given environment that contains more information that in what is generally called a context. In the case of the static semantics of a programming language, this environment contains the so called symbol table which contains the set of visible variables or objects together with their types, while the dynamic semantics contains a memory state that is a mapping from variable names to values. In this context, performing a computation (as type-checking a program) is proving that a particular predicate (this program is correctly typed) holds in the theory. Natural Semantics is declarative and does not define the tactic used to build proofs.

We try now to give a more precise idea of Natural Semantics, giving some examples of rules.

$$\frac{\emptyset \vdash Decls \rightarrow \rho \qquad \rho \vdash Stats}{\vdash \textbf{declare } Decls \textbf{ in } Stats}$$

This rule explains what could be the top rule of the static semantics of a programming language in which statements use objects defined in a preceding declarative part. The declarations *Decls* are verified in an empty environment and produce a new environment containing declared objects and their associated types that is used to verify the statements *Stats*.

Most of the rules express the fact that the verification process is recursive and how the environment is propagated. Modifications or accesses to the environment very often appear at the leaves of the tree as in the following axiom (i.e. an inference rule without premises).

$$\rho \vdash \textbf{var } X : T\textbf{;} \rightarrow \{X : T\} \cdot \rho \textbf{ (provided } X \textbf{ not in } \rho\textbf{)}$$

This rule explains that a declaration must by added to the environment only when the declared object does not already appear in it. X and T are variables that will be unified with actual variables names and type expressions.

3.2 Current Implementation of Natural Semantics

Natural Semantics has been implemented by means of a specification formalism called Typol (Despeyroux, 1988). It is part of a programming environment generator called Centaur (Borras et al., 1988) that allows the definition of the concrete and abstract syntaxes of programming languages, pretty printing rules, and semantic rules. The environment offers support for creating a good man-machine interface. While most of the system is written in Lisp, semantic rules (and possibly parsers and un-parsers) are translated into Prolog code.

This implementation includes two features that are used to construct real tools: an error recovery system that allows to specify special rules that are used when an error occurs providing error messages, and the ability to keep track of occurrences in the source tree and to indicate precisely where errors occur.

3.3 Application to Markup Language Based Documents

Markup based languages are good candidates to specify their semantics using Natural Semantics. An XML validator checking that an XML page complies with a particular DTD is part of what can be generated with Centaur if the static semantics of XML is specified in Typol. However, this is not exactly our goal as we want to give a semantic to Web sites. In the following we are considering sites implemented in XML, but the method can be easily extended to HTML. We can consider that the real implementation language of a site is not XML but a language defined by a set of XML tags. The usage of these tags can be constrained by some syntactical rules: a DTD. To define completely the language, we have now to find a way of defining semantic rules.

In our running examples, DTDs have been manually translated to AS (Despeyroux, 1996), a Centaur formalism to define modular abstract syntaxes (this should be quite easily performed automatically). This is sufficient for Centaur to generate a syntactical editor for our sites. The next step is to define the static semantics in Typol and to get advantage of the Centaur interface to derive an executable version of this semantics. Executable in this context means that, once we have compiled the semantic rules, we get a tool that can check that a particular site complies with the semantic rules, providing error messages or warnings if necessary.

3.4 Maintaining Web Sites

In our vision, the role of the creator or the administrator of a Web site is quite more important that the traditional web-master one as it must take care of the semantics of the site.

We can here get the advantage of using an executable semantics: from a specification we can produce such tools as validators, compilers or debuggers.

We reach here the main distinction between checking a program, where all concepts (types, numbers etc.) are very precisely defined in a mathematical way, and checking a Web site where we manipulate natural languages and some representation of knowledge that cannot be manipulated by a theorem prover. Hopefully, the way Natural Semantics is conceived and implemented makes it possible to delegate some parts of the deductive process to external tools.

In the context of a Web site, two parameters may evolve: the contents, and the semantic constraints. In the last case, one have to regenerate the compiler itself (i.e. the site validator) before checking that the site is still correct. The Centaur environment generates a generic debugger than can be used to debug the semantics itself.

3.5 Examples

In this section we give two examples of Web site static specifications that have been done with the help of Natural Semantics.

Specifying a Thematic Directory A thematic directory is a way of classifying documents found on the Web and presenting them to users in an organised manner. They can be found as services provided by portails such as Yahoo[3] or Voilà[4]. Topics are presented in a hierarchical manner as categories and sub-categories. Most of the time, documents indexed in such a directory are selected and classified manually. Such directories must evolve frequently, as new documents are very often available and because it is some time helpful to reorganise the hierarchy of categories. After each modification in such a directory, one would like to check the integrity of it.

The whole directory is a tree that contains two different types of node. The first type of nodes concerns categories that come with a list of sub-categories. For example the category `Travel` may contains the sub-categories `Boating`, `Backpacking`, etc. In this case, one want to check that sub-categories `make sense` in the context of the upper level category, by reference to a thesaurus or an ontology. The second type of nodes concerns leaf categories that come with a list of documents, or more exactly with a list of descriptions and links to documents. In the hypothesis in which a list of keywords is provided for each document, one would like to check that this list `is relevant` in the context in which the document is referred. The structure of the site (what we call its abstract syntax) can be described as a DTD while the site itself is an XML document.
To specify the coherence of such a site, one will affect a type to each node. A node has a type if and only if all subtrees are typed. Type expressions are made of a list of names that should appear in the thesaurus. Sub-categories must get a type that is "more precise" that the type of the upper category. For example, it is possible to say that the type `Travel.Boating` is more precise than the type `Travel` because it makes sense to speak of `Boating` in the context of `Travel` according to our favourite thesaurus.

The two following rules are equivalent

[3] http://www.yahoo.com/
[4] http://www.voila.fr/

$$\frac{\mathbf{makesense}(Name, \tau) \qquad \vdash \mathit{List} : \tau \cdot Name}{\vdash \mathit{category}(\mathit{name}\ Name, \mathit{List}) : \tau}$$

$$\frac{\vdash \mathit{List} : \tau \cdot Name}{\vdash \mathit{category}(\mathit{name}\ Name, \mathit{List}) : \tau}\ \textbf{(provided makesense}(Name, \tau)\textbf{)}$$

The expression $\mathit{category}(\mathit{name}\ Name, \mathit{List})$ uses the Typol syntax to describe a schema in which $Name$ and List are variables. It matches pieces of XML text (or tree) like the following one:

```
<category>
  <name>...</name>
  ...
</category>
```

A category appears in a context in which some category nodes have already been traversed (except for the top ones for which the context is empty). The predicate **makesense** is an external predicate that may call a thesaurus (or a ontology) browser to verify that the relation between categories and sub-categories is respected. The order of premises in the upper part of the rules is not theoretically important. All premises must be proved to allow the inference. When a premise can be viewed as a test or a function call, the second form may be preferred. The second form makes clear that it is a conditional recursive rule.

A reference may be affected the type τ if it contains a sufficient number of keywords that make sense for this type. We can call this a `confidence` degree.

$$\frac{\mathbf{confidence}(Keywords, \tau, N)}{\vdash reference(Title, Keywords, Uri) : \tau}\ \textbf{(provided}\ N > 50\textbf{)}$$

In this rule, we can use again an external predicate that calculates a rating number that must say what percentage of the list of keywords is relevant in the current context (category). It is the choice of the web-master to impose this ratio.

In the above rules, we have chosen to call an external predicate that is viewed as an axiom from the theory defined by the rules. It should be possible to define the thesaurus as a term, defining also access procedures to the thesaurus in Typol. In this case, the thesaurus should be considered as a parameter (an hypothesis) in the rules, and the sequents should have the form $\rho \vdash X : \tau$, saying that X has type τ in the context of a particular thesaurus ρ.

When the proof that, according to the specification, the Web site is correct cannot be constructed, some recovery rules are used to provide error messages or warning.

Specifying the Coherence of an Institutional Site Verifying an institutional site as a research institute one is closer to type-checking a program than the previous example. The hypothesises are the following: an authority, may be the staff, is in charge of the general Web site presentation that is the official view of the institution; then some subparts of the site are edited by other groups of persons, may be some researchers in charge of a particular project. We can see this form of site as containing a declarative part, and an other one than must follow these declarations, even when the first "declarative" part is updated. For example, each team may produce a list of other teams with which they cooperate.

In the following figure, the inner box must be considered as a page accessible by the link A.

Here is the list of teams; click on one of them to get details on research topics.

team A	Team A research topics focus on... We cooperate with teams B and M.
teams B	...
teams C	...

As some teams may disappear, it is reasonable to check that a particular team does not pretend having some collaboration with a team that does not exist any more.

We are facing the usual problem of forward declarations in programming languages: each team-page must be checked in an environment containing the list of all teams. Three possibilities are available using natural semantics: the more operational one consists of a two-passes process (one to construct a list of teams, a second to check the team pages), a second possibility consists in constructing a fix point, a third one consists of considering the whole site as a context as shown in the following rules.

$$\frac{ListofTeams \vdash ListofTeams}{\vdash teamspresentation(Texte, ListofTeams)}$$

Each time a reference to a team exists, we must check that it is in the legal list:

$$\frac{\mathbf{appears}(Name, ListofTeams)}{ListofTeams \vdash team(Name)}$$

This approach supposes that references to teams of the institute are correctly tagged in the document.

4 Comparison with Related Works

In this section, we focus on the two main already cited works addressing what we have called "static semantics" of Web sites and based on a context-dependent computation : the work of Psaila and Crespi-Reghizzi (1999) from the software engineering community and Webmaster (van Harmelen and van der Meer, 1999) from the AI community.

We can compare our method with the one used by Psaila and Crespi-Reghizzi that use attributed grammars to semantically evaluate Web documents. It has been shown that Natural Semantics is a generalisation of attributed grammars (Attali, 1992), and that some Typol programs can be compiled into attributed grammars. However our method is not limited to XML documents, as we can also use it for HTML documents (like WebMaster) and, at the difference, we do not need to modify an actual DTD to apply it as we do not use XML attributes.

Even if our goals are very closed to those of WebMaster, our approach is quite different. We are closed to traditional software engineering, trying to take Web documents as input and producing error messages or warning related to them, whereas WebMaster tries to classify documents following the type of documents then the type of errors that have been detected in them. With our approach, the output lists all the errors and their point of occurrence for each Web document, that, according to our point of view, corresponds more to the needs of Web sites designers.

Our method is based on a more powerful expressiveness of semantics representation: Webmaster addresses only semantic constraints between pages whereas we can address both syntactical and semantic constraints, being able to specify any semantic constraint between any entity such as pages but also any fragment of pages.

5 Conclusion and Future Work

In this paper, we first made a parallel between the syntax of programming languages and the structure of Web sites (or semi-structured documents), and between the semantics of programs and the semantics of Web sites, applying some notions of types and semantic rules to documents on the Web. Then we argued that Natural Semantics (traditionally used to specify the semantics of programming languages) was a powerful tool to address the problem of Web site (static) semantics specification i.e. semantic verifications which is crucial for supporting the design and the maintenance of Web sites. A test implementation of our ideas has been done using the Centaur system using its semantics specification formalism Typol to construct a prototype of a Web site verification system by means of inference rules using natural semantics (Kahn, 1987; Despeyroux, 1988; Borras et al., 1988; Jacobs and Rideau-Gallot, 1992).

This test example has been done for two classes of XML-based Web sites: thematic directories and institutional sites. The use of natural semantics shows clearly the difference between syntactical checking (for example verifying a page against a DTD, like in an XML validator) that is context-free, and a semantic computation that is context-dependent. The example of thematic directory shows also the possibility of using external resources tools (thesauri, ontologies).

The overall experience is positive and needs to be confirmed with more real applications. However, two main problems must be addressed in the future: first, the Typol language is heavy to use in such a context (some rules must be given for each node of the abstract syntax tree i.e. each tag, and every action on the environment must be specified in great details), it seems necessary to be able to get default or generic rules, to define a more intuitive syntax and to offer a way for capitalising experiments; second, as we need to perform global computation on Web sites we need to get a less monolithic implementation of the system.

Acknowledgments We want to thank Hacene Cherfi (student from Paris-XVII University), for its participation at the first steps of this work.

References

Attali, I. 1992. Incremental evaluation of natural semantics specifications. In *Proceedings of Programming Language Implementation and Logic Programming*. Springer-Verlag LNCS, August.

Borras, P., D. Clement, Th. Despeyroux, J. Incerpi, G. Kahn, B. Lang, and V. Pascual. 1988. Centaur: the system. In *Proceedings of the 3rd Symp. on Software Development Environments*, Boston, USA, November. also Inria research report 777, France, December 1987.

Despeyroux, J. 1986. Proof of translation in Natural Semantics. In *Proceedings of the first ACM-IEEE Symp. on Logic In Computer Science, Cambridge, MA, USA*, pages 193–205, June. also Inria research report 514, April 1986.

Despeyroux, Th. 1987. Executable Specification of Static Semantics. In *Semantics of Data Types, Lecture Notes in Computer Science,Vol. 173*, June.

Despeyroux, Th. 1988. Typol: a formalism to implement Natural Semantics. Technical Report 94, Inria, France, March.

Despeyroux, Th. 1996. As, for abstract syntax - manual - v1.0. Technical Report 197, Inria, september.

Fensel, D., R. Decker, M. Erdman, and R. Studer. 1998. Ontobroker : the Very High Idea. In *Proceedings of the 11th Internation FLAIRS Conference (FLAIRS-98)*, May.

Fensel, Dieter, Jürgen Angele, Stefan Decker, Michael Erdmann, Hans-Peter Schnurr, Rudi Studer, and Andreas Witt. 1998b. On2broker: Lessons Learned from Applying AI to the Web. Technical report, Institute AIFB.

van Harmelen, F. and D. Fensel. 1999. Practical Knowledge Representation for the Web. In D. Fensel, editor, *Proceedings of the IJCAI'99 Workshop on Intelligent Information Integration*.

van Harmelen, F. and J. van der Meer. 1999. Webmaster: Knowledge-based Verification of Web-pages. In *Twelfth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE'99*.

Heflin, J., J. Hendler, and S. Luke. 1999. SHOE: A Knowledge Representation Language for Internet Applications. Technical report, University of Maryland at College Park. Technical Report CS-TR-4078 (UMIACS TR-99-71), Dept. of Computer Science,http://www.cs.umd.edu/projects/plus/SHOE/.

Jacobs, I. and L. Rideau-Gallot. 1992. A Centaur tutorial. Technical Report 140, Inria.

Kahn, G. 1987. Natural Semantics. In *Proceedings of the Symp. on Theorical Aspects of Computer Science, TACS*, Passau, Germany. LNCS 247, Springer-Verlag, Berlin. also Inria Research Report 601, February 1987.

Plotkin, G.D. 1981. A Structural Approach to Operational Semantics. Technical report, Aarhus, DAIMI FN-19.

Psaila, G. and S. Crespi-Reghizzi. 1999. Adding Semantics to XML. In *Second Workshop on Attribute Grammars and their Applications, WAGA'99*.

Szabo, E. 1969. *The Collected Papers of Gerhard Gentzen*. Amsterdam: North-Holland.

Trousse, B. 2000. Evaluation of the Prediction Capability of an On-line Adaptative Service Approach Based on User Behaviour Similarity. In *Proceedings of the 6th RIAO Conference - Content-Based Multimedia Information Access*, Paris, France, april.

Trousse, B., M. Jaczynski, and R. Kanawati. 1999. Using User Behavior Similarity for Recommandation Computation : The Broadway Approach. In *Proceedings of 8th international conference on human computer interaction (HCI'99)*. Lawrence Erlbaum Associates, august. Munich.

W3C-RDF. 1999. Ressource Description Framework (RDF) Model and Syntax Specification, March. W3C http://www.w3.org/TR/REC-rdf-syntax-19990222/.

W3C-RDF-Schema. 1999. Ressource Description Framework (rdf) Schema Specification, March. W3C http://www.w3.org/TR/1999/PR-rdf-schema-19990303/.

W3C-XML. 1998. Extensible Markup Language (XML) 1.0, February. W3C Recommandation http://www.w3.org/TR/1998/REC-xml-19980210.

W3C-XML-Schema. 2000. XML Schema Part 1: Structures; XML Schema Part 2: Datatypes, February. W3C Working Draft http:http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/.

W3C-XSL. 2000. Extensible Styleshet language (XSL) version 1.0, January. W3C Working Draft http://www.w3c.org/TR/2000/WD-xsl-20000112/.