Design of an inference engine for the semantic web

This document is a proposal for a Master Thesis made as a part of my Master Degree in Computer Science Education (Software Systems) at the Open University of the Netherlands. The work will be done in collaboration with the research department of the company Agfa in Mortsel Belgium.

Student data

Name	Guido Naudts
Student number	831708907
Address	Secretarisdreef 5
	2288 Bouwel
	Belgium
Telephone work	0030-2-542.76.01
Home	0030-14-51.32.43
E-mail	Naudts_Vannoten@yahoo.com

Introduction

What is the semantic web? Very short: the semantic web is the automation of the internet. What does this mean? On the internet an enormous amount of information is available mostly in the form of HTML or other formats that can be represented in a human readable way. The semantic interpretation of the material can only be performed by a human being as it cannot be done by a computer. Consequently, the internet pages are only to a small degree accessible for computer manipulation (something can be done by interpreting e.g. the embedded links).

Now to make these pages available for processing by a computer there are two possibilities:

- a computer can analyse the contents of the pages. This is the domain of artificial intelligence. For the purpose of this thesis this possibility can be eliminated.
- information can be added to the pages that is meant for usage by a computer. This information is called meta-information. Obviously, standards are needed to make this information useful. So W3C devised the standard RDF. RDF stands for Resource Description Framework and W3C stands for World Wide Web Consortium, an organisation charged with the task of devising standards for the internet. RDF creates the possibility to add declarative meta-information to the web pages. In this proposal the second possibility will be explored.

The standard representation of RDF is in XML format. I will not make use of this representation but, for convenience sake, I will use the RDF representation named N3. This syntax for RDF is more easy to use and work with than the XML syntax.

With the mere use of RDF alone much can be done for the automation of the internet. But it is not enough. Let's try to build a system, layer by layer. Figure 1 gives a picture of the complete system. Till now I discussed the first three layers of fig. 1 (but not RDF Schema). With RDFS (= RDF Schema) RDF is extended with notions like:

class, property, constraint,... Another extension is in preparation (in the Webont working group of the W3C) where an ontology for the Semantic Web will be specified [WEBONT]. RDF gives the possibility to use subjects, properties and objects; an ontology says which subjects, properties and objects can be used.

But again this is not enough: on top of these four layers (see fig. 1) a logical layer is needed, so rules to be used by inference engines can be made. Sometimes a proof is needed, e.g. rule 1 says: x is the author of y. Rule 2 (on another website) says: the author of y is director of w3c. Then x is director of w3c. Therefore a proof layer is added. Finally on top is the trust layer, which defines what can be trusted and what not.

The logic layer and the proof layer will be the subject of my investigation.

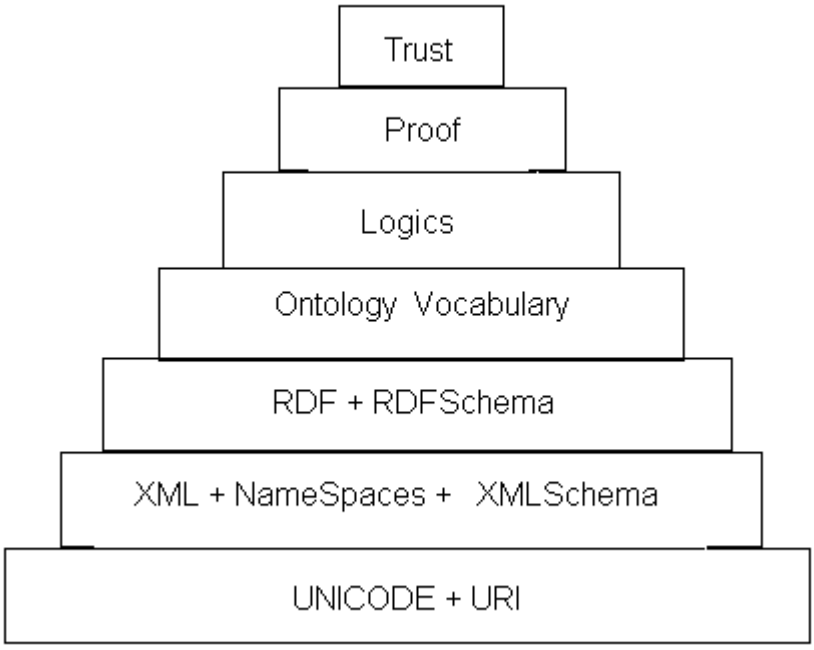

Fig. 1 The different layers composing the semantic web
Explanation of figure 1:
At the bottom are the basic mechanisms to define and use information on the internet : unicode and URI = Universal Resource Locator.
The second layer is the XML (Extensible Markup Language) layer with its namespaces and XMLSchema, the successor of DTD (= Document Type Definition), a language for describing XML objects.
The third layer is the RDF (Resource Description Framework) [RDFMS] and RDFSchema (see text) [RDFSC] layer.
The fourth layer is the ontology layer. Where RDF offers a syntax to describe a document, the ontology will offer the specific terms needed to describe it.
The fifth layer is the logic layer. This layer will make it possible to generate rules and make queries based on these rules.
The sixth layer is the proof layer. Proof here is to be understood as proof validation, not as proof generation (see [DESIGN]) e.g. the proof that text X really has been written by person Y.

Finally, the seventh layer is the trust layer. In order for people and computers to trust each other a web of trust must be stablished. This must be based on rules i.e.logic and cryptographic mechanisms. Besides trust there are other applications at this level.

Research question

The problem

Regarding the semantic web experiments have been done with two programming systems that are inference engines: CWM (abbreviation for Closed World Machine) [SWAP/CWM] and Euler [DEROO]. These programs are based on the RDF and RDFS standards and on top of these are using logical enhancements and ontology features. However these program systems are experimental and were developed ad-hoc. Now, as Tim Berners-Lee points out [DESIGN], there is a need for an inference engine that is adaptable, well specified and as simple as possible.
In my view the following restrictions when building an inference engine are imposed by the structure of the internet:

- the data are distributed. A consequence of this is the difficulty to avoid inconsistencies.
- heterogeneity of goals and purposes must be taken into account.
- there is a relatively slow transmission medium.
- the scale of everything can vary from small to extensive.

A well defined engine can help a lot in guaranteeing consistency and procuring efficiency. Indeed it is difficult to reason about something which is not well specified.

So there really is a need to develop an inference engine (or family of engines) founded on a more formal basis. So the fundamental question arises:
What is the best way for realising an inference engine so that the restrictions that are imposed by the structure of the World Wide Web on the Semantic Web are met ?

There are other arguments. First there is the argument of evolvability (a general requirement for software development: see [GHEZZI]). The internet is not something static neither is the semantic web. So an inference engine deployed on the scale of the internet will be confronted with continuous development and changes. There is the argument of variability. In a system with millions of users it is to be expected that not all user communities have the same needs and look for the same properties in an inference engine. This implies that the engine must be built in a modular way. This again leads to the development of a basic engine which is kept as small as possible. This basic engine is enhanced with a set of logic specifications comprising facts and rules. However there has to be a formal specification of the engine too. Such a specification can be made with a functional language.

In order to be able to reason logically about the inference engine a specification of it has to be made. One way of doing this is by using meta-logical frameworks. Hence our research question can be reformulated: *can meta-logical frameworks be used to specify the inference engine of the semantic web?*

A start for the study of the metalogical frameworks can be made by looking at the following systems: Twelf, ALF, GANDALF, lambda-prolog and Otter (see the bibliography of references).

Once a description on a meta-level has been made it is possible to reconsider the two points mentioned above, namely: optimisation and checking of inconsistency. Different mechanisms might exist for optimising a proof process. One is especially worth mentioning: reordering (this can be done as well on the inter-clause level (reordering of the sequence of clauses) as on the intra-clause level (reordering of the premises)). Other principles might exist (e.g. failure should be detected as soon as possible). *So the question is: what optimisation techniques can be used?*

Consistency checks are also important on the internet. To a certain degree this interferes with optimisation in this sense that inconsistencies should be detected as soon as possible. Inconsistencies can be defined by logical rules. However there is another possibility: if the inference engine is defined by a metalogical specification, the correctness of the proofs can be metalogically checked. Thus inconsistencies will be detected. *The question is : how can inconsistencies best be avoided?*

When a system on internet scale is involved, some assumptions concerning logic can readily be made:

- no closed world: when files are collected and merged on the internet, it is not to be expected that the information will be complete.
- paraconsistent logics: when there are contradictions, strange results would be the result if the rule is accepted that from a contradiction everything can be deduced.
- should reasoning be monotonic?

The question is: which system of logic should be followed on the internet?

Another question is: what is the interpretation of the logic, i.e. what are its semantics?

Solution framework:

I propose the following framework as a guiding structure for arriving at an answer to the questions posed above:

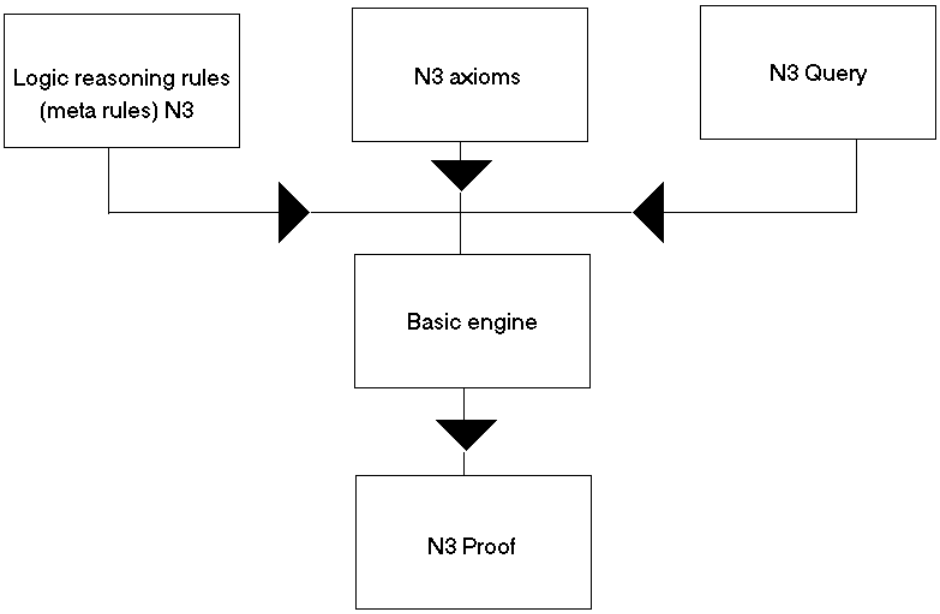

Fig.2. Guiding framework for the thesis

The basic engine is based on the Robinson resolution algorithm and consists basically of unification and backtracking. The engine is kept as simple as possible. Functions which are not considered to be basic will be defined in the meta-rules in N3-format. An example of this is the definition of a property as being a transitive property in RDFSchema. Possibly also optimisations and strategies for the engine can be defined here?

What must be built in and what not?
Should the meta-part be treated by the engine in the same way as the N3 axioms or not?

The axioms represent input data and rules. They themselves and the query, as well as the proof, are also in N3-format.

Aim of the thesis

The aim is to specify and test a basic inference engine in a functional language; to investigate how the engine can be extended with meta-statements, especially meta-statements that are concerned with the efficiency of the deductive process and the consistency of the input data; test the engine using test cases of the semantic web. An answer will be given to the questions asked above: although the answer cannot be complete, I will strive to highlight the most important aspects of the problem.

Eventually a working engine will be built in Python that can use the meta-rules in N3 syntax.

Planning

Specification of a parser and an inference engine in a functional language (presumably Haskell): 300 hours.

Testing the use cases and building a set of meta-statements for the engine: 200 hours.

Study of the literature: 150 hours.

Writing the text of the thesis: 150 hours.

Coaching and graduation committee

Chairman: prof. dr. J. Jeuring, Open University of the Netherlands
Secretary : ir. F.J. Wester, Open University of the Netherlands
Coach: ir. J. De Roo, Agfa Gevaert Mortsel Belgium

Bibliography of references

[AIT] Hassan Aït-Kaci, *WAM A tutorial reconstruction*. Still a good intro in Prolog engines .
[BENTHEM] Van Benthem e.a., *Logica voor informatici,* Addison Wesley 1991. A very good introduction in logic. In Dutch.
[DESIGN] http://www.w3.org/DesignIssues/ *Tim Berners-Lee's site with his design-issues articles*.
[DEROO] http://www.agfa.com/w3c/jdroo *The site of the Euler program.*
[GANDALF] http://www.cs.chalmers.se/~tammet/gandalf *Gandalf Home Page.*
[GHEZZI] Ghezzi e.a., *Fundamentals of Software Engineering,* Prentice-Hall 1991.
[LAMBDA] http://www.cse.psu.edu/~dale/lProlog/ *Lambda prolog home page.*
[MCGUINESS] Deborah McGuiness, *Explaining reasoning in description logics,* 1966 Ph.D.Thesis. A very readable text .
[OTTER] http://www.mcs.anl.gov/AR/otter/ .*Otter Home Page.*
[RDFM] *RDF Model Theory* ,Editor: Patrick Hayes, http://www.w3.org/TR/rdf-mt/ .Semantics of rdf.
[RDFMS] *Resource Description Framework (RDF) Model and Syntax Specification,* <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222>
[RDFSC] *Resource Description Framework (RDF) Schema Specification 1.0* <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>
[SWAP/CWM] http://www.w3.org/2000/10/swap http://infomesh.net/2001/cwm .CWM is another inference engine for the web.
[TBL01] Berners-Lee e.a., *The semantic web,* Scientific American May 2001.
[TWELF] http://www.cs.cmu.edu/~twelf *Twelf Home Page*
[UNCERT] Hin-Kwong Ng e.a.,*Modelling uncertainties in argumentation,* Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong , http://www.se.cuhk.edu.hk/~hkng/papers/uai98/uai98.html Some ideas about handling uncertainties.
[WEBONT] Ed. Heflin e.a.,*Requirements for a Web Ontology Language,* W3C Working Draft 07, March 2002, http://www.w3.org/TR/2002/WD-webont-req-20020307/

[WESTER] Wester e.a., *Concepten van programmeertalen,* Open Universiteit Eerste druk 1994. Important for a thorough introduction in Gofer. In Dutch.

Abbreviations

ALF : Algebraic Logic Functional Programming Language
CWM : Closed World Machine
An experimental inference engine for the semantic web
DTD : Document Type Definition , a language for defining XML-objects.
HTML : Hypertext Markup Language
RDF : Resource Description Framework
RDFS : RDF Schema
W3C : World Wide Web Consortium
WAM : Warren Abstract Machine
Probably the first efficient implementation of prolog .
XML : Extensible Markup Language.