An Abstract Intelligent Machine for the Semantic Web

Yannick Naudet,
Patrick Plichart, Simon Renault, Thibaud Latour

Centre de Recherche Public Henri Tudor,
Centre d’Innovation par les Technologies de l’Information

29, Avenue JFK
L-1855 Luxembourg – Kirchberg

yannick.naudet@tudor.lu
Content

- The Semantic Web vision
  - Understanding Web informations – Humans vs Machines
  - Structuring Meta-Knowledge in the SW

- DRP: an Ontology model
- The Semantic Web Processor (SWP), as an implementation of DRP
  - The SWP as a learning machine
  - Learning RDFS
  - Rule example

- The SWP as an architecture
- Usage example

- Conclusion & Perspectives
The Semantic Web vision

Intelligent machines with inference capabilities can reason on semantic data. They interact with the semantic layer and also with the classical layer of the Web. The human user interacts with all levels:
- with intelligent machines
- with the semantic layer
- with the classical layer

The human user interacts with the Web: access, visualisation, modification. Machine does what it is required to do, without knowing what it does...

html, doc, pdf, svg, xml, avi, mpeg, wav, mp3, ...

- Application
- Agent
- Web Service

Predetermined Interaction
Intelligent Interaction

November 15-18, 2004 - LUXEMBOURG

The success of the web is mainly due to its simplicity

- At a conceptual level:
  - A protocol → HTTP
  - A language → HTML
  - An access tool → browser

- At a usage level
  → Anyone can say something on the web and can access anything easily

It can be assumed that the SW will be successful under the same conditions

Current status:
- HTTP
- XML for syntax, RDF for semantics
- Access → browser, for humans
  → What about machines?
Understanding Web informations

- Humans vs Machines

- The Web: an **open and unbounded environment**, non-deterministic, non-predictable...
- The Web is **human-readable**, machine processing stays ‘stupid’
- Human process the Web using meta-knowledge they acquire from processing other source of knowledge
- Machines do not possess such meta-knowledge, unless:
  - They have been programmed to do so → meta-knowledge is hard-coded in the processing unit and depends strongly on data structure
  - The meaning of data and the way they can be processed is provided with data → the **Semantics**
- Semantic Web is a web that is machine processable
  - So that **machine is more adapted to the Web true nature**
  - So that **machine can better serve humans**
Structuring Meta-Knowledge in the SW

- Semantics is expressed in the form of ontologies (most often)
- Ontologies are expressed in formal languages,
- But so far:
  1. Only structural aspects are considered
  2. The meaning of the language primitives is not expressed in a machine-processable formalism (rdfs:subClassOf meaning is written in full text in the recommendation)
  3. Procedural knowledge and executable functions or processes, which are linked to the ontology concepts manipulation are not considered
What do we use when building an ontology?

- In documentation (text)
- In code of RDF-, RDFS-, or OWL-compliant applications

Ex. What are the consequences on properties and instances when a rdfs:Class is the rdfs:subClassOf another rdfs:Class?
Structuring Meta-Knowledge in the SW

What’s happening when building my ontology?

- In documentation (text)
- In rdfs:comment
- Should I write a MY-compliant applications?

- In documentation (text)
- In code of RDF-, RDFS-, or OWL-compliant applications

Primitive semantics  
MY  
Primitives

MeaningOf

Primitive semantics  
OWL  
Primitives

ExpressedIn

Primitive semantics  
RDFS  
Primitives

ExpressedIn

Primitive semantics  
RDF  
Primitives

ExpressedIn
Structuring Meta-Knowledge in the SW

First problem: semantics of a given language

- Is this enough?
- What can a machine do, if I give it an ontology containing concepts and their meaning?
  - Reasoning and deductions if it has inferences capabilities
  - Execution?
    - Is a description of a process explaining how to send a mail enough for a machine to actually send a mail?
      - NOT with current SW languages
      - YES if the description uses primitives the machine knows, and that are linked with actions the machine can execute.

Second problem: There is also a need for expressing processes or executable functions, accounting for actions
There might be an infinity of languages and ontologies which requires expressing some specific behaviour.

This behaviour might concern knowledge structure as well as actions to execute.

They must be processable, using only the explicit knowledge that defines them self-consistently.

What we need is a system which is NOT “some language”-compliant.
Structuring Meta-Knowledge in the SW

→ There must exist some base language(s?) in which all others are describable, in terms of *structure* (primitive concepts), in terms of *behaviour* (what kind of knowledge manipulation is implied by the use of a concept), and in terms of *actions* allowed by the language.

→ What we need is a generic processor which relies on a basic language, capable of processing any other language which has been formally described.
DRP: an Ontology model

- A Universe description as a set of three subsets:

\[ U = \{D^u, R^u, P^u\} \]

- **The Domain** \((D)\):
  - The set of facts describing concepts related to the universe components, their properties and their relationships.

- **The Rules** \((R)\):
  - Axioms that define or constraint the universe
  - Rules rule the way the universe behaves, maybe reacting modifications of the universe, or to interactions with the external world

- **The Processes** \((P)\):
  - Structured sequence of actions which transforms the universe or the external world
The Semantic Web Processor (SWP) as an implementation of DRP

- We assume that the basic $L_D$ is RDF
- We assume that the basic $L_R$ is SWRL
- We assume that the basic $L_P$ could be PSL, BPML, “SWPL”, ...

- Thus we have as native processor U’s
  - $U^{RDF} = \{ D^{RDF}, R^{RDF}, P^{RDF} \}$
  - $U^{SWRL} = \{ D^{SWRL}, R^{SWRL}, P^{SWRL} \}$
  - $U^{SWPL} = \{ D^{SWPL}, R^{SWPL}, P^{SWPL} \}$

- Implementation of these depends on the processor implementation language
The SWP as a learning machine

- If the processor is implemented in Prolog (the core language):
  - $D$: rdf(S,P,O).
  - $R$: rdf(S,P,O) :- {rdf(S,P,O)}.
  - $P$: rdf(S,P,O) :- {rdf(S,P,O)},{Built-in predicates}.

- Basic internal processes are
  - Assert
  - Retract
  - interpret
The SWP as a learning machine
Learning RDFS

How to describe RDFS?

\[ \mathcal{D}_{RDFS} = \]

```xml
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)


- <rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
  <rdfs:isDefinedBy
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#" />
  <rdfs:label xml:lang="en">Resource</rdfs:label>
  <rdfs:comment>The class resource,
    everything.</rdfs:comment>
</rdfs:Class>

- <rdf:Property rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
  <rdfs.isDefinedBy
    rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
  <rdfs:label xml:lang="en">type</rdfs:label>
  <rdfs:comment>Indicates membership of a
    class</rdfs:comment>
  <rdfs:range
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource" />
</rdf:Property>

- <rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-
```

---


November 15-18, 2004 - LUXEMBOURG

Avec le soutien du FEDER
Learning RDFS

How to describe RDFS?

\[
\mathcal{R}_{\text{RDFS}} = \quad \text{SWRL interpreter will interpret into:}
\]

\[
\text{triple}(X_1, \text{rdfs:subClassOf}, \text{rdfs:Resource}) \\
\text{ :- triple}(X_1, \text{rdf:type}, \text{rdfs:Class}).
\]
Rule Exemple

```
<rdf:RDF xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-nmtoken"
    xmlns:srule="http://www.w3.org/2003/11/swrl#">
  <srule:Variable rdf:ID="X1" />  
  <srule:Variable rdf:ID="X2" />  
  <srule:Variable rdf:ID="X3" />
  <srule:Implies>
    <srule:Body rdf:parseType="Collection">
      <srule:IndividualPropertyAtom>
        <srule:propertyPredicate rdf:resource="#hasParent" />
        <srule:argument1 rdf:resource="#X1" />
        <srule:argument2 rdf:resource="#X2" />
      </srule:IndividualPropertyAtom>
      <srule:IndividualPropertyAtom>
        <srule:propertyPredicate rdf:resource="#hasSibling" />
        <srule:argument1 rdf:resource="#X2" />
        <srule:argument2 rdf:resource="#X3" />
      </srule:IndividualPropertyAtom>
      <srule:IndividualPropertyAtom>
        <srule:propertyPredicate rdf:resource="#hasSex" />
        <srule:argument1 rdf:resource="#X3" />
        <srule:argument2 rdf:resource="#male" />
      </srule:IndividualPropertyAtom>
    </srule:Body>
    <srule:Head rdf:parseType="Collection">
      <srule:IndividualPropertyAtom>
        <srule:propertyPredicate rdf:resource="#hasUncle" />
        <srule:argument1 rdf:resource="#X1" />
      </srule:IndividualPropertyAtom>
    </srule:Head>
  </srule:Implies>
</rdf:RDF>
```

interpret

```
triple(X1, '#hasUncle', X3)
:-
  triple(X1, '#hasParent', X2),
  triple(X2, '#hasSibling', X3),
  triple(X3, '#hasSex', #male')
```
The SWP as an architecture

WORLD: The Semantic Web

Ontologies

Ontology Manager

Query Engine

Inference Engine

Rule Manager

Process & Built-in Manager

Process Engine

I/O Dispatcher

Volatile Knowledge Base

Knowledge Base

RDF-based description or query

World alteration

SWP

Facts

Rules

Processes

Built-ins
Usage example

Mail desc.: from, to, subject, body, ...
L: SendMail description (D,R,P):
* D: description of the domain of L,
* R: description of the rules of L,
* P: L_SendMail(Mail) ->
rdf(Y, rdf:type, SWP:TCP_Socket),
SWP_TCP_Connect(Y), ...

Mail instance: rdf(mail, rdf:type, L:Mail)
Process call: L_SendMail(mail)

Processes
Built-in Functions
Proc. Lang. (SWPL)
Rule Lang. (SWRL)
RDF

Interpreter modules
SWRL
SWPL
Native lang

SWP

agent

Mail sent

1

2

3

4a

4b

New process: L_SendMail

Knowledge Base

SWRL_Exec
SWPL_Exec
SWP:TCP_Socket
SWP_TCP_Connect
Conclusion

- With the DRP model, Ontologies incorporates processing capabilities and allow defining languages with an infinity of abstraction levels, self-consistently.

- SWP provides a browser for machines, allowing for Web meta-knowledge understanding.

- It is assumed that the proposed SWP will be able to process any ontology built using any language provided that there exist a formal description of the language somewhere.
Perspectives

● SWP will allow users to design their own abstract intelligent machine, use other's machine in a distributed way and ultimately say something to the Web in any arbitrary language that can be processed by any processor.

● Processors can be realized as
  ● Agents,
  ● Web services,
  ● Applications,
  ● Etc…