What Objects for the Semantic Web?

Hassan Aït-Kaci

ILOG & Simon Fraser University

Outline

- Motivation
- A review of existing Web objects
- [In?]Digestion of all-of-the-above
- Tracking the ideal WOLF
- Conclusion

Motivation: seek a meaningful Web Object Logic Formalism

The advent of the “*Web*” and the sudden recent proliferation at breakneck speed of “*Web technologies*” warrants a pause of reflexion.

⇒ What is (or should be) a “*Web Object*”?

... and what can (or should) we do with it?

Goal: review notions of objects in extant web technologies and attempt to distill therefrom a WOLF capable of conveying meaning over the Web.

A review of extant Web objects

The web of confusion: *You want objects ... ?*

... *here they are !*

But what are these *good* for?

A rough categorization:

⇒ Document-oriented objects

⇒ Data-oriented objects

⇒ Computation-oriented objects

⇒ Semantics-oriented objects

Document-oriented objects

Originally, the most basic web data is structured document.

Viz., the nature of HTML is text annotation (inspired from SGML), and this has been inherited by its descendant—XML.

Basic structure of an XML element ⇔ that of a syntax tree.

XML is the *de facto* standard for web data representation.

ergo ...

A web object is an XML element

Document-oriented objects: XML

```
<!DOCTYPE family [ <!ELEMENT family (person)*>
                   <!ELEMENT person (name)>
                   <!ELEMENT name (#PCDATA)>
                   <!ATTLIST person id       ID     #REQUIRED
                                    mother   IDREF  #IMPLIED
                                    father   IDREF  #IMPLIED
                                    children IDREFS #IMPLIED>
]>

<family>
   <person id="jane" mother="mary" father="john">
      <name>Jane Doe</name>
   </person>
   <person id="john" children="jane jack">
      <name>John Doe</name>
   </person>
</family>
...
```

Document-oriented objects

Understanding XML building blocks—1st approx:

XML	⟷	**PL Concept**
elements	⟷	data structures
document type definitions	⟷	grammars
schemas	⟷	types
namespaces	⟷	modules

... but it ain’t so simple!

Document-oriented objects

What is (really) an XML schema? ... a DTD++?

W3C XML Schema Requirements

> “The purpose of a schema is to define and describe a class of XML documents by using these constructs to constrain and document the meaning, usage and relationships of their constituent parts: datatypes, elements and their content, attributes and their values, entities and their contents and notations. Schema constructs may also provide for the specification of implicit information such as default values. Schemas document their own meaning, usage, and function.
>
> Thus, the XML schema language can be used to define, describe and catalogue XML vocabularies for classes of XML documents.”

[http://www.w3.org/TR/NOTE-xml-schema-req]

Document-oriented objects

Many proposals...

- XtrML Schema Definition Language - W3C XML Schema Working Group
- XML-Data [Reduced] (XDR)
- Document Content Description (DCD)
- Schema for Object-oriented XML (SOX)
- Document Definition Markup Language (DDML) - p.k.a. XSchema
- XML Structure Validation Language using Patterns in Trees (Schematron)
- Datatypes for DTDs (DT4DTD)
- REgular LAnguage description for XML (RELAX)
- Document Structure Description (DSD)
- Tree Regular Expressions for XML (TREX)

...but no winner!

Document-oriented objects

XML has no semantics! ... It is pure syntax.

So what’s good about this?

⇒ one parser does it all

⇒ universal data representation

⇒ including meta data

Does this sound familiar? ... LISP all over again!

Document-oriented objects

OK - syntactic structure, but what about *semantics*?

For documents,

$$\boxed{\text{meaning = layout styling}}$$

Hence,

- ▶ Style interpretation of XML is done with a stylesheet language; *e.g.*, XSL.
- ▶ XSL is a pattern-directed rule-based language for XML⟶HTML translation.

XSL	=	labeled tree pattern matching	([quasi] r.e. path exprs—XPath)
	+	tree traversal	(structural recursion)

- ▶ XSLT: general (*e.g.*, XML⟶XML) structure transformations

Document-oriented objects

XSLT example: business card

```
<card type="simple">
  <name>John Doe</name>
  <title>CEO, Widget Inc.</title>
  <email>john.doe@widget.com</email>
  <phone>(202) 456-1414</phone>
</card>
```

XHTML rendering semantics with an XSLT stylesheet:

```
<html xmlns="http://www.w3.org/1999/xhtml"><title>business card</title>
<body><h1>John Doe</h1><h3><i>CEO, Widget Inc.</i></h3>
<p>email: <a href="mailto:john.doe@widget.com"><tt>john.doe@widget.com</tt></a></p>
<p>phone: (202) 456-1414</p>
</body></html>
```

Business card example XSLT rule & template:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
                xmlns="http://www.w3.org/1999/xhtml">

  <xsl:template match="card[@type=’simple’]">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <title>business card</title><body>
        <xsl:apply-templates select="name"/>  <xsl:apply-templates select="title"/>
        <xsl:apply-templates select="email"/> <xsl:apply-templates select="phone"/>
    </body></html>
  </xsl:template>

  <xsl:template match="card/name">
    <h1><xsl:value-of select="text()"/></h1>
  </xsl:template>

  <xsl:template match="email">
    <p>email: <a href="mailto:{text()}"><tt><xsl:value-of select="text()"/></tt></a></p>
  </xsl:template>

  ...
</xsl:stylesheet>
```

NB: XSL(T) can be the basis for an XML Schema... (*e.g.*, Schematron)

Data-oriented objects

Semi-structured data

- ▶ Notion from the DB world:
  - – heterogenous data
  - – inconsistent structure
- ▶ Examples:
  - – biological data
  - – web data
- ▶ Object Exchange Model (OEM):
  - – Edge-labeled (multi-)Graph

Data-oriented objects

Semi-structured Data Model [Abiteboul, Buneman, Suciu; ’98–’00]

```
bib : &o1 { paper : &o12 { ... },
            book : &o24 { ... },
            paper : &o29 { author : &o52 "Abiteboul",
                           author : &o96 { firstname : &978 Victor",
                                           lastname : &741 "Vianu" },
                           title : &21
                                   "Regular path queries with constraints",
                           cites : &o12,
                           cites : &o24,
                           pages : &o73 { start : &o712 122,
                                          end : &o862 133
                                        }
                         }
          }
```

Data-oriented objects

SSD structures resemble XML structures...

- edge-labeled trees
- graph structure via oid references
- multiple occurrences of components

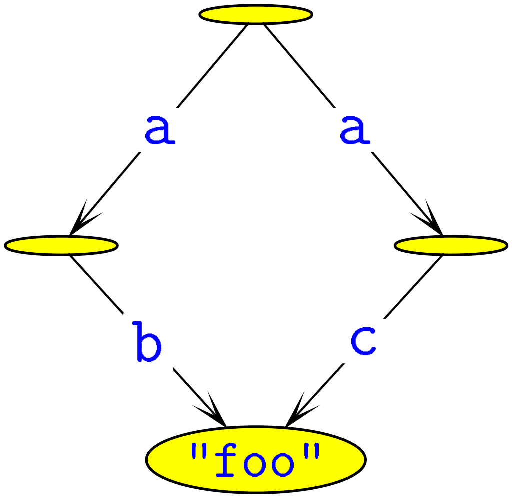
...but do they really?

- component order matters in XML elements—not in SSD
- XML attributes are not ordered, but limited to strings (CDATA)
- structure sharing is awkward to encode in XML

Data-oriented objects

How to encode SSD graph structure in XML:

```
{ a : { b : &o "foo" } , a : { c : &o } }
```



```
<a><b id="&o">foo</b></a>
<a c="&o"/>
```

?or?

```
<a b="&o"/>
<a><c id="&o">foo</b></a>
```

Data-oriented objects

SSD object structures are subject to O/RDB queries (SQL, OQL)...

⇒... XML Query Languages have been proposed.

e.g.,

- Quilt
- UnQL
- XDuce (types for TREX)
- XML-QL
- XPath (path expressions for XSL/T)
- XQL
- YatL

Data-oriented objects

All these XML Query Languages share the same basic characteristics

- underlying SSD object model
- standard O/RDB operations (proj., select., iter., join, ...)
- object type/schema declarations
- path constraints (reg. exp. matching)
- structural recursion
- functions/methods

Recently an (excellent!) unifying contribution has proceeded to give a rigorous formal basis: the XML Query Algebra.

`http://www.w3.org/TR/query-algebra/`

Computation-oriented objects

Ambient Calculus & Ambient Logic

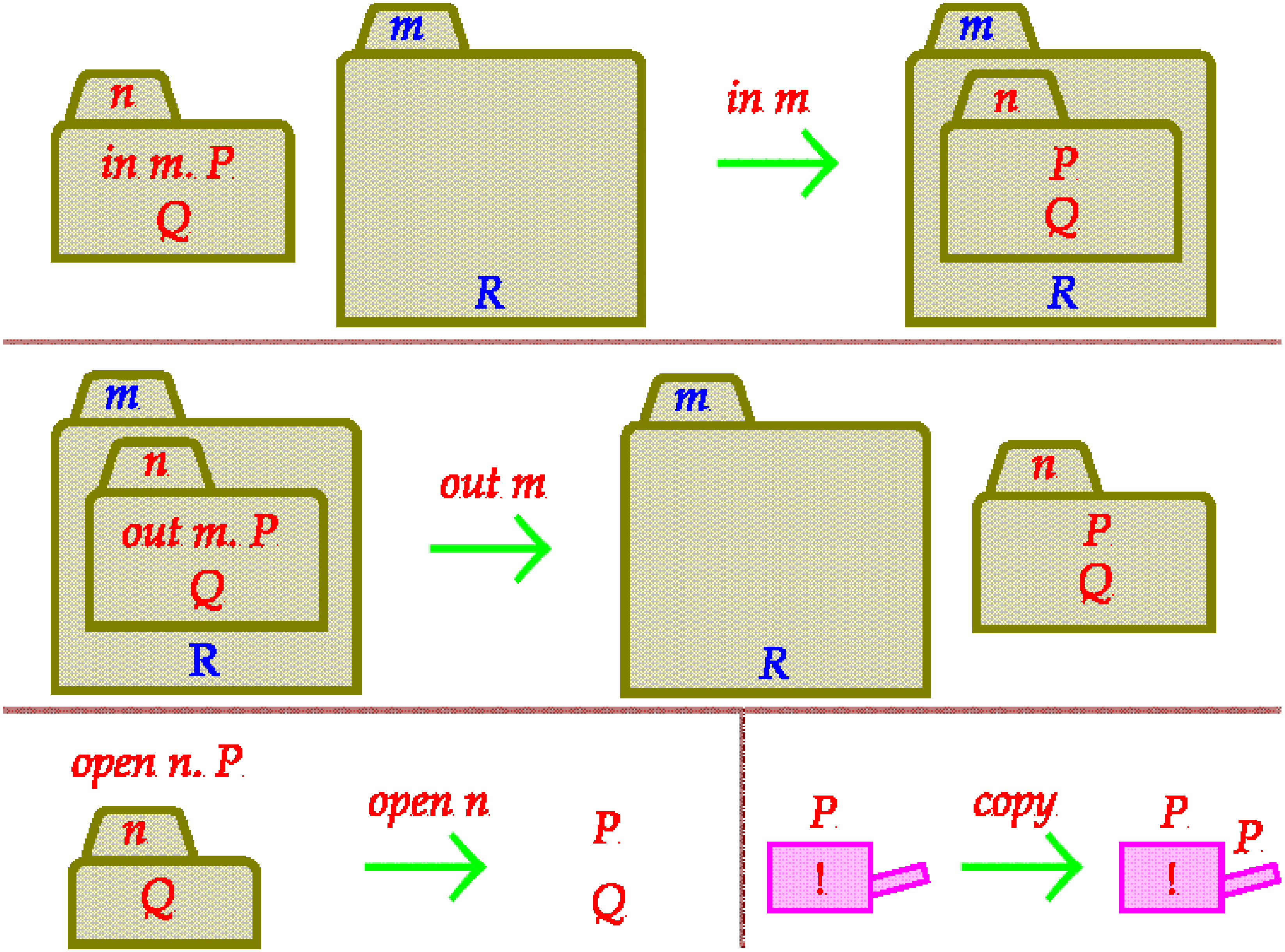
[Cardelli and Ghelli, '98–'00]

- simple and elegant calculus of mobile computing (firewalls, security)
- surprising powerful ($\sim \pi$-calculus, join calculus, CHAM)
- possesses an associated modal logic—*Ambient Logic*—that describes spatial and temporal states of computation
- unexpected connection with SSD structures:

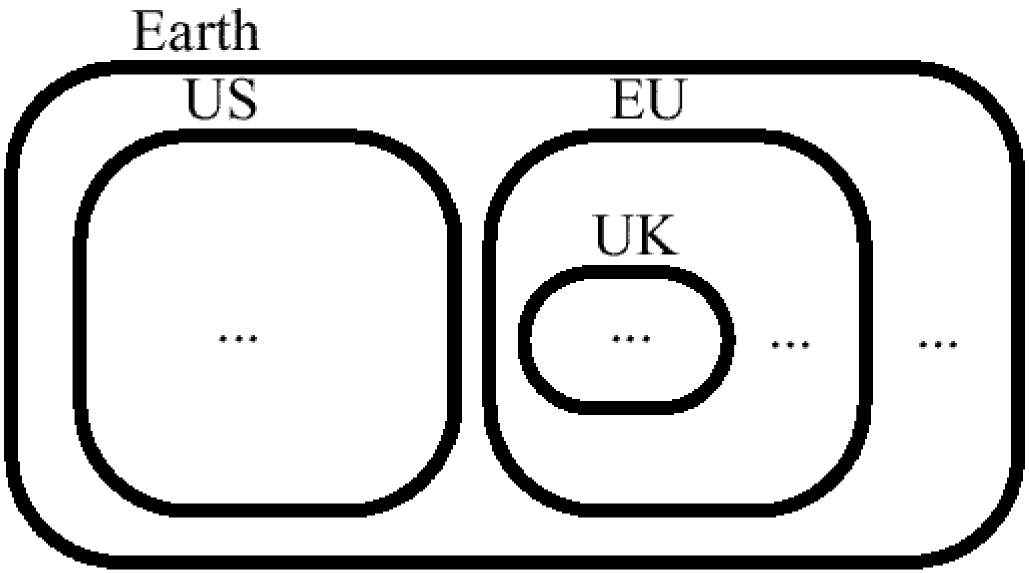
  ⇒ ambients as SSD trees

  ⇒ SSD querying as Ambient Logic satisfaction (QTL)

Metaphor: The Folder Calculus



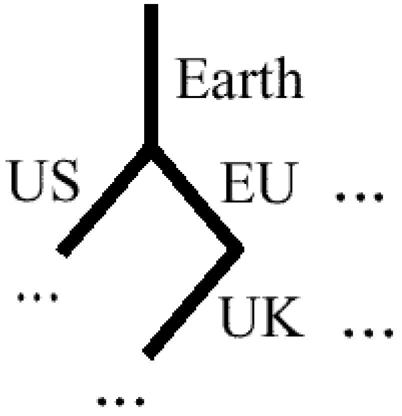
Geographical maps



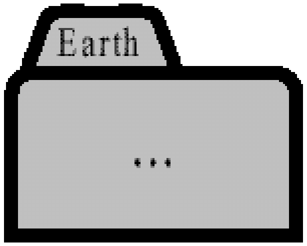
Expressions

Earth[US[...] | EU[UK[...] | ...] ...]

Edge-labeled trees



Folders



Semantics-oriented objects

- Simple HTML Ontology Extension (SHOE)
- Resource Description Format (RDF)
- Ontology Inference Layer (OIL)
- RDF-related logic systems:
    - MetaLog
    - SWI-Prolog
    - RDF(S) encoding of (F-)Logic

Semantics-oriented objects—SHOE

SHOE is (*indeed!*) a simple HTML extension that enables:

- ▶ defining ontologies:
  - ⇒ taxonomies of conceptual categories (is-a)
  - ⇒ relations schemas typed by these categories
  - ⇒ inference rules as Horn clauses over these relations (Datalog)
- ▶ using ontologies to annotate web pages with semantic content by
  - ⇒ associating (unique) instances to individuals assimilated to URI's
  - ⇒ declaring categories for these instances
  - ⇒ specifying facts using relations defined by ontologies
  - ⇒ importing ontologies

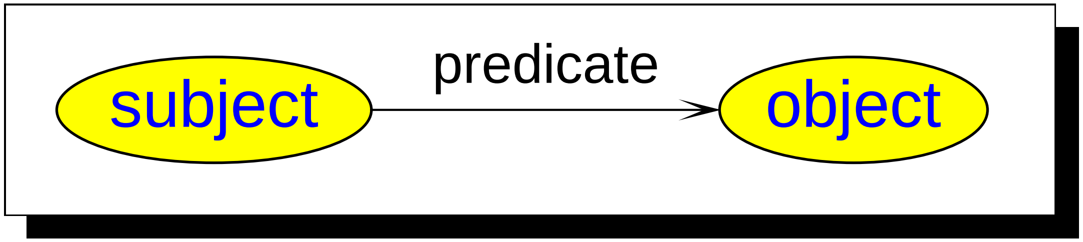
NB: SHOE does not use XML nor RDF—it just extends HTML

Semantic contents expressed in SHOE is meant to enable "intelligent" Web agents.

Semantics-oriented objects—RDF

RDF is a notation for meta-description about data (metadata) using (edge- and node-) labeled graphs.

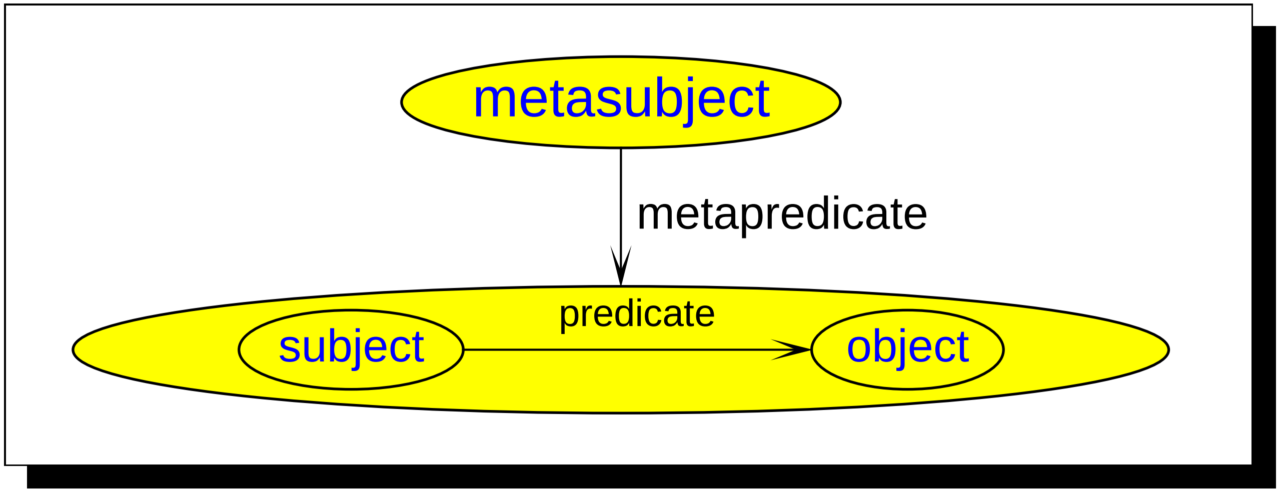
- Basic building block: “triple” labeled by “resources”—*i.e.*, URI’s.
- A triple consists of a resource (the subject), linked through a resource (the predicate) to another resource (the object).
- A triple states that the subject has a property, denoted by the predicate, whose value is the object:



- The information carried by a triple is called a “statement.”

Semantics-oriented objects—RDF

- RDF statements can be reified and be denoted as resources—hence, RDF's metalinguistic nature:



- RDF uses XML for its serialized syntax.
- RDF enables the definition of vocabularies which can be shared over the Web thanks to XML namespaces (*e.g.*, Dublin Core).
- RDF Schema (RDFS) is a meta-description of RDF in RDF; it defines a vocabulary for RDF.

OIL

OIL is a major effort by a large group of AI researchers whose aim is to design a language for declaring and using ontological knowledge over the Web.

OIL derives its essence from three roots:

- ▶ Description Logics
  formal semantics and reasoning support
- ▶ Frame-based systems
  epistemological modeling primitives
- ▶ Web languages XML-based and RDF-based syntax

[In?]Digestion

Much of the above is taking advantage of:

- a standard syntax for a Web *Lingua Franca*—XML
- XML inherent labeled graph structures to model *everything*
- OODB and AI technology for KR and inference

with the hope of achieving widespread semantic information interchange.

Confused notion of Web object—derived from document processing, data bases, and logic.

The danger is that the failed ambitions of the past be recast in the new Web Esperanto (as opposed to old LISP-based vernacular).

Lessons from the past ⇒ substantial mileage can be covered!

The Ideal WOLF

- ▶ Start with OSF Logic—*i.e.*, LIFE
  - ⇒ Subsumes the largest set comprising most other object models
  - ⇒ Designed from the start as a powerful Labeled Graph Algebra
- ▶ What it offers
  - ⇒ Flexible knowledge/data model
    - ∗ Graph unification / matching
    - ∗ Object structure theories
  - ⇒ Deductive power over taxonomic knowledge
  - ⇒ Inductive power (*cf.*, RHB+ [Sasaki *et al.*])
  - ⇒ Residuation as “optimistic” mobile computation (*cf.*, E)
- ▶ What it needs
  - ⇒ Regular feature path constraints
  - ⇒ Collection type unification
  - ⇒ CSP

Conclusion

Ours is an exciting time...

> The diverse approaches of object in various (Web, DB, AI, PL, CP) communities seem to converge toward a coherent and powerful *usable* notion that has substantial declarative and procedural power to enable a truly *Semantic Web*.

... and our future unfathomably more thrilling!