Documenting Ontologies the Mathematical Way

Christoph Lange `<ch.lange>` — *foaf:maker*, *foaf:img*

Michael Kohlhase `<m.kohlhase>` — *foaf:maker*, *foaf:img*

foaf:knows

Jacobs University, Bremen, Germany `<jacobs-university.de>` — *foaf:member*, *foaf:member*

JACOBS UNIVERSITY

flexible granularity

inferred vs. declared facts

Add documentation to an ontology – but how?

context-sensitive presentation

Model ontologies as mathematical theories like in textbooks!

flexible expressivity

An ontology is written as a *theory*, containing *statements*.

FOAF references entities from other ontologies. OMDoc tools can identify missing imports.

Imports can carry morphisms.

Classes, properties, and individuals are declared as *symbols*, having a *type*.

Informal sections and descriptions can be included right into the ontology document.

Formal and informal content of a statement can be given in parallel and ☞cross-linked.

Custom symbol notations:
- $owl{:}disjointWith(A, B) \vdash A \sqcap B = \bot$
- $foaf{:}member(g, m) \vdash g \ni_{member} m$.

We use compound property types to declare range and domain.

A lot about inverse properties can be inferred from the original property and the inverseness declaration, using the OWL axioms.

Proofs can be given, but are optional.

We can add axioms that refine imported concepts.

By importing other logics, we can exceed OWL's expressivity.

FOAF Basics

The basic idea is pretty simple. If people publish information in the FOAF document format, … So, what is the 'FOAF document format'? … For example, one interesting relationship type is ☞*foaf:depiction*. …

Theory (FOAF)

Imports: *OWL*, *First Order Logic*

WordNet, Dublin Core ($dc{:}creator \mapsto maker$), …

Classes

Symbol (Project): $Project{:}\ owl{:}Class$

The ☞*Project* class represents the ☞*class* of things that are 'projects'. These may be formal or informal, collective or individual. It is often useful to indicate the ☞*homepage* of a ☞*Project*.

Axiom: $Project \sqsubseteq wordnet{:}Project$

We reuse and specialize WordNet's project class.

Axiom: $Project \sqcap Document = \bot$

A ☞*Project* is not a ☞*Document*.

Properties

Symbol (made):

$made{:}\ owl{:}ObjectProperty(Agent \to owl{:}Thing)$

The ☞*made* property relates an ☞*Agent* to something ☞*made* by it.

Symbol (maker): *maker* (no declared type; see below)

Axiom: $made = maker^{-}$

Type Assertion: $maker{:}\ owl{:}ObjectProperty(owl{:}Thing \to Agent)$

Proof: We prove this using the declared type of ☞*foaf:made*, using axiom …, and the OWL direct semantics of ☞*owl:inverseOf*.

Lemma: $maker = made^{-}$

Proof:

1. We know that $made = maker^{-}$
2. Interpreted using the OWL semantics, this means that $made^{\mathcal{I}} = (maker^{-})^{\mathcal{I}} = (maker^{\mathcal{I}})^{-}$
3. Now we apply the inverse on both sides, eliminate double inverses, and obtain $(made^{\mathcal{I}})^{-} = ((maker^{\mathcal{I}})^{-})^{-} = maker^{\mathcal{I}}$
4. This is the interpretation of $maker = made^{-}$, which we had to prove. □

Axiom: $\forall t, m, n.\ maker(t, m) \wedge name(m, n) \Rightarrow dc{:}creator(m, n)$

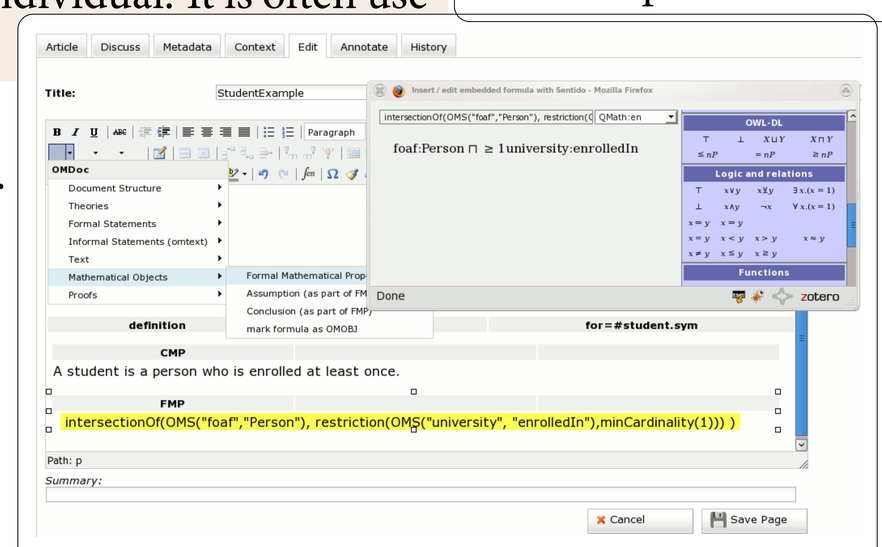
Symbol (membershipClass): The ☞*membershipClass* property relates a ☞*Group* to an RDF class representing a sub-class of ☞*Agent* whose instances are all the agents that are a ☞*member* of the ☞*Group*.

Axiom: $\forall m, g, C.\ g \ni_{member} m \wedge membership Class(g, C) \Rightarrow m :_{rdf:type} C$

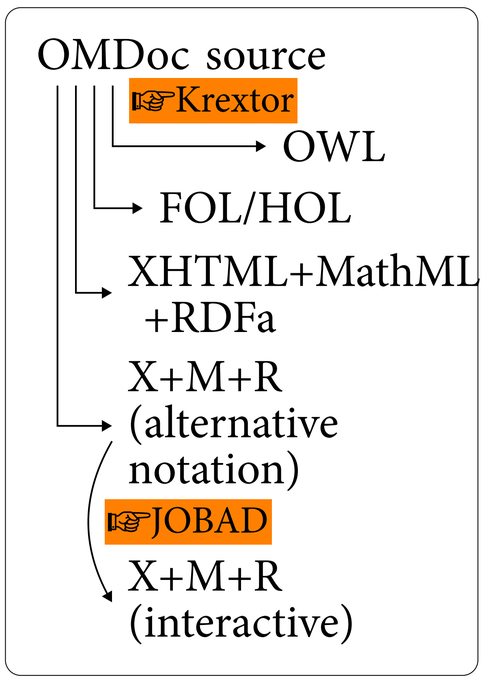
Write ontologies in ☞OMDoc (Open Math. Documents).

We have formalized RDF, RDFS, and OWL (symbols and partial semantics) as theories.

From a single OMDoc source, we can obtain:
- various formal representations
- various human-readable presentations



Editor of the ☞SWiM semantic wiki



Related work:
- CASL/Hets (modular, heterogeneous, no documentation)
- XHTML+RDFa (emerging for ontologies, we also generate it)

Links

```
http://www.omdoc.org
http://kwarc.info/projects/swim/
http://jomdoc.omdoc.org/wiki/JOBAD
http://kwarc.info/projects/krextor/
```