

owl2onto.cls: Marking up OWL2 Ontologies in S^TE^X.^{*}

Michael Kohlhase
Jacobs University, Bremen
<http://kwarc.info/kohlhase>

November 17, 2015

Abstract

The `owl2onto` class allows mark up OWL2 Ontologies in S^TE^X and generate OWL2-XML from them via the L^AT_EX_{ML} system.

Contents

1	The User Interface	2
1.1	Package Options	2
1.2	Ontologies	2
1.3	Declarations	2
1.4	Properties	3
2	Limitations	3
3	The Implementation	4
3.1	The <code>owl2onto</code> Class	4
3.2	Classes and Properties	5
3.3	Using Ontologies	7
3.4	Providing IDs	7
3.5	Finale	7

^{*}Version ? (last revised ?)

Experimental!

do not use!

1 The User Interface

The `owl2onto` package allows mark up ontology-based Metadata in \LaTeX documents that can be harvested by automated tools or exported to PDF.¹

The main idea behind the `owl2onto` class and package is to think of (documented) ontologies as documents, which present the knowledge behind ontology informally to the (human) reader and at the same time contain enough (hidden) information so that a formal ontology can be generated from them.

1.1 Package Options

`showmeta` The `owl2onto` package takes a single option: `showmeta`. If this is set, then the metadata keys are shown (see `[Kohlhase:metakeys:ctan]` for details and customization options).

1.2 Ontologies

`ontology` The `owl2onto` class provides the `ontology` environment environment to declare OWL classes; it takes the place of the usual `document` environment¹, but its optional argument provides `KeyVal` attributes that allow to mark up the semantic aspects of the class, see Figure 3. In this example, we only have a single class declaration.

```
\documentclass{owl2onto}
\begin{ontology}[id=foaf,baseURI=http://xmlns.com/foaf/0.1/]
\declaration[type=Class,name=Agent,cseq=agent]
{An agent (eg. person, group, software or physical artifact).}
\end{ontology}
```

Class: Agent

An agent (eg. person, group, software or physical artifact).

Example 1: A simple Ontology in \LaTeX and its presentation

1.3 Declarations

`\declaration` In general the `\declaration` macro can be used to declare the objects of various types in an ontology. The `type` key is used to specify this, its values range over

¹EdNOTE: continue

¹Admittedly, it is somewhat unconventional to use the document environment for this, but this is the best way to ensure that we an OWL/XML document with a single document root.

EdN:2

the set². The `name` attribute specifies the name of the declared object. This information is used in the XML generation; see Figure 2 for the result of generating XML from Figure 3. Finally, the `cseq` key allows to specify a command sequence for the object, which can be used in properties.

```
<?xml version="1.0"?>
<!--This OWL2 ontology is generated from an sTeX-encoded one via LaTeXML,
you may want to reconsider editing it.-->
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
  xmlns:stex="http://kwarc.info/ns/sTeX"
  xmlns:omdoc="http://omdoc.org/ns"
  ontologyIRI="http://xmlns.com/foaf/0.1/"
  xml:base="http://xmlns.com/foaf/0.1/"
  xml:id="foaf">
  <Import>owl2.omdoc#OWL2</Import>
  <Declaration stex:srcref="test.tex#textrange(from=5;0,to=5;64)">
    <Class IRI="Agent"/>
  </Declaration>
  <AnnotationAssertion>
    <AnnotationProperty IRI="http://www.w3.org/2000/01/rdf-schema#comment"/>
    <IRI>Agent</IRI>
    <Literal>An agent (eg. person, group, software or physical artifact).</Literal>
  </AnnotationAssertion>
</Ontology>
```

Example 2: The OWL/XML generated from Figure 3

1.4 Properties

EdN:3

The properties of the declared objects can be stated via the `\axiom` macro. Its first argument is an OWL formula marked up in prefix notation³, and the second one a natural language explanation.

```
\begin{document}


---


Axiom: agent||document
Agents are not Documents


---


<DisjointClasses>
  <Annotation>
    <AnnotationProperty IRI="http://www.w3.org/2000/01/rdf-schema"/>
    <Literal>Agents are not Documents</Literal>
  </Annotation>
  <Class IRI="Agent"/>
  <Class IRI="Document"/>
</DisjointClasses>
```

Example 3: An Axiom

2 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently

²EdNOTE: give a list of all of them.
³EdNOTE: explain this better

discussed in the \LaTeX GitHub repository [[sTeX:github:on](#)].

1. none reported yet

3 The Implementation

The functionality is spread over the `owl2onto` class and package. The class provides the `document` environment and the `ontology` element corresponds to it, whereas the package provides the concrete functionality.

`owl2onto.dtx` generates four files: `owl2onto.cls` (all the code between `<*cls>` and `</cls>`), `owl2onto.sty` (between `<*package>` and `</package>`) and their \LaTeX ML bindings (between `<*ltxml.cls>` and `</ltxml.cls>` and `<*ltxml.sty>` and `</ltxml.sty>` respectively). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

3.1 The owl2onto Class

We first define the `owl2onto` class, which on the \LaTeX side just calls the article class.

```

1 <*cls>
2 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
3 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
4 \ProcessOptions
5 \LoadClass{article}
6 \RequirePackage{modules}
7 \RequirePackage{owl2onto}
8 </cls>
9 <*cltxml>
10 # -*- CPERL -*-
11 package LaTeXML::Package::Pool;
12 use strict;
13 use LaTeXML::Global;
14 use LaTeXML::Package;
15 DeclareOption('showmeta',sub {PassOptions('metakeys','sty',ToString(Digest(T_CS('\CurrentOption
16 DeclareOption(undef,sub {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))));
17 ProcessOptions();
18 LoadClass('article');
19 RequirePackage('modules');
20 RequirePackage('owl2onto');
21 </cltxml>

```

Now, we also need to register the namespace prefixes for \LaTeX ML to use.

```

22 <*cltxml>
23 RegisterNamespace('owl'=>"http://www.w3.org/2002/07/owl#");
24 RegisterNamespace('stex'=>"http://kwarc.info/ns/sTeX");
25 RegisterNamespace('ltx'=>"http://dlmf.nist.gov/LaTeXML");
26 RegisterNamespace('omdoc'=>"http://omdoc.org/ns");
27 </cltxml>

```

Since we are dealing with a class, we need to set up the document type in the L^AT_EX_{ML} bindings.

```
28 <*cltxml>
29 RelaxNGSchema('owl2+ltxml',
30     '#default'=>"http://www.w3.org/2002/07/owl#",
31     'omdoc'=>"http://omdoc.org/ns",
32     'ltx'=>"http://d1mf.nist.gov/LaTeXML",
33     'stex'=>"http://kwarc.info/ns/sTeX");
34 </cltxml>
```

The `owl2onto` class also supplies the top-level `document` environment, which we redefined so that we can provide KeyVal arguments.

EdN:4

document The `document` environment is redefined to allow a `baseURI`⁴

```
35 <*cls>
36 \srefaddidkey{ontology}
37 \addmetakey{ontology}{baseURI}
38 \newcommand\ontology[1][]{\document\metasetkeys{ontology}{#1}%
39 \importmodule[owl2]{OWL2}%
40 \ifx\sref{id}@empty\begin{module}\else\begin{module}[id=\sref{id}]\fi}
41 \newcommand\endontology{\end{module}\enddocument}
42 </cls>
43 <*cltxml>
44 DefEnvironment('{ontology} OptionalKeyVals:omdoc',
45     "<owl:Ontology "
46     . "    "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id'))() "
47     . "    "?&GetKeyVal(#1,'baseURI')(xml:base='&GetKeyVal(#1,'baseURI'))() "
48     . "    "?&GetKeyVal(#1,'baseURI')(ontologyIRI='&GetKeyVal(#1,'baseURI'))()>"
49     . "    "#body"
50     . "</owl:Ontology>",
51     beforeDigest=> sub { AssignValue(inPreamble=>0); },
52     afterDigest=> sub { $_[0]->getGullet->flush; return; });#$
53 </cltxml>
```

3.2 Classes and Properties

Before we provide the core functionality, we need to ensure that the `modules` and `presentation` packages are loaded. For L^AT_EX_{ML} we also initialize the package inclusions.

```
54 <*package>
55 \RequirePackage{amstext}
56 \RequirePackage{presentation}
57 </package>
58 <*sltxml>
59 # -*- CPERL -*-
60 package LaTeXML::Package::Pool;
```

⁴EDNOTE: @Deyan, need to remember the `baseURI` in a keyword, so that we can use it in the class and property environments

```

61 use strict;
62 use LaTeXML::Global;
63 use LaTeXML::Package;
64 </sltxml>

```

We first set up a utility macro that allows us to export values

`\exportvalue`

```

65 <*package>
66 \def\exportvalue#1#2{%
67 \expandafter\def\csname\declaration@cseq #2\expandafter\endcsname\expandafter{#1}
68 \expandafter\g@addto@macro\csname module@defs@mod@id\expandafter\endcsname\expandafter%
69 {\expandafter\def\csname\declaration@cseq #2\expandafter\endcsname\expandafter{#1}}
70 </package>

```

`\declaration`

```

71 <*package>
72 \addmetakey{declaration}{id}
73 \addmetakey{declaration}{cseq}
74 \addmetakey{declaration}{type}
75 \addmetakey{declaration}{name}
76 \newcommand\declaration[2][\metasetkeys{declaration}{#1}%
77 \ifx\declaration@cseq@empty\else
78 \expandafter\exportvalue{\ontology@baseURI\declaration@name}{@URI}
79 \expandafter\exportvalue{\expandafter\text\declaration@name}{@}
80 \fi
81 \noindent\textbf{\declaration@type: \declaration@name}\par\noindent #2\par\noindent}
82 </package>
83 <sltxml>
84 DefMacro(' \declaration OptionalKeyVals:declaration {}', sub {
85 my ($self,@args)=@_;
86 my $name = ToString(KeyVal($args[0],'name')) if $args[0];
87 my $cseq = ToString(KeyVal($args[0],'cseq')) if $args[0];
88 DefConstructor("\\".$cseq."<XTok name='$name' />",
89 properties=>{mode=>'math',requiremath=>1})
90 if ($cseq && $name);
91 (Invocation(T_CS(' \declaration'),@args)->unlist);
92 });
93 DefConstructor(' \declaration OptionalKeyVals:declaration {}', sub {
94 my ($document, $keyval, $arg, %properties) = @_;
95 my $type = ToString(KeyVal($keyval,'type'));
96 my $name = ToString(KeyVal($keyval,'name'));
97 $document->openElement('owl:Declaration');
98 $document->insertElement('owl:'.$type,undef,(IRI=>$name)) if $type;
99 $document->closeElement('owl:Declaration');
100 if (defined $arg) {
101 $document->openElement('owl:AnnotationAssertion');
102 $document->insertElement('owl:AnnotationProperty',undef,(IRI=>'http://www.w3.org/2000/01/rdf-schema#'));
103 $document->insertElement('owl:IRI',$name);
104 $document->insertElement('owl:Literal',$arg);

```

```

105     $document->closeElement('owl:AnnotationAssertion');
106   }
107   });#$
108 </sltxml>

```

\axiom

```

109 <*package>
110 \addmetakey{axiom}{id}
111 \addmetakey{axiom}{cseq}
112 \addmetakey{axiom}{type}
113 \addmetakey{axiom}{name}
114 \newcommand\axiom[3] [] {\metasetkeys{axiom}{#1}%
115 \noindent\textbf{Axiom:} #2\par\noindent #3\par\noindent}
116 </package>
117 <*sltxml>
118 DefConstructor('\axiom OptionalKeyVals:axiom {}{}',
119               "<owl:Axiom>#2"
120               ".?#3(<owl:Annotation>"
121               .      "<owl:AnnotationProperty IRI='http://www.w3.org/2000/01/rdf-schema#comment' />"
122               .      "<owl:Literal>#3</owl:Literal>"
123               .      "</owl:Annotation>")()"
124               . "</owl:Axiom>");
125 </sltxml>

```

3.3 Using Ontologies

useontology

```

126 <*package>
127 \def\useontology#1#2{\input}
128 </package>

```

3.4 Providing IDs

To provide default identifiers, we tag all OMDoc elements that allow `xml:id` attributes by executing the `numberIt` procedure from `omdoc.sty.ltxml`.

```

129 <*ltxml>
130 Tag('owl:Ontology',afterOpen=>\&numberIt,afterClose=>\&locateIt);
131 Tag('owl:Declaration',afterOpen=>\&numberIt,afterClose=>\&locateIt);
132 Tag('owl:AnnotationAssertion',afterOpen=>\&numberIt,afterClose=>\&locateIt);
133 Tag('owl:AnnotationProperty',afterOpen=>\&numberIt,afterClose=>\&locateIt);
134 Tag('owl:IRI',afterOpen=>\&numberIt,afterClose=>\&locateIt);
135 Tag('owl:Literal',afterOpen=>\&numberIt,afterClose=>\&locateIt);
136 Tag('owl:Axiom',afterOpen=>\&numberIt,afterClose=>\&locateIt);
137 </ltxml>

```

3.5 Finale

Finally, we need to terminate the file with a success mark for perl.

138 $\langle \text{cltxml} \mid \text{sltxml} \rangle 1;$