RDFa Metadata in LaTeX*

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

October 28, 2010

Abstract

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

Contents

Experimental! do not use!

*Version v0.2 (last revised 2010/09/01)

1 Introduction

The `rdfmeta` package allows mark up extensible metadata in sTEX documents, so that that it can be harvested by automated tools or exported to PDF; see [KKL10] for an application. The main idea is that metadata are annotated as key value pairs in the semantic environments provided by sTEX. In most markup formats, the metadata vocabularies are fixed by the language designer. In sTEX, the `rdfmeta` package allows the user to extend the metadata vocabulary.

```
\importmodule[../ontologies/cert]{certification}
...
\section[id=userreq,hasState=$\statedocrd{\tuev}$]{User Requirements}
...
```

```
<imports from="../ontologies/cert.omdoc#certification"/>
...
<omgroup xml:id="userreq" xmlns:cert="../ontologies/cert.omdoc#certification">
  <metadata>
    <link rel="cert:hasState">
      <dc:title>User Requirements</dc:title>
      <resource rel="cert:statedocrd" resource="cert:tuev"/>
    </link>
  </metadata>
...
</omgroup>
```

Example 1: Metadata for Certification

Take, for instance, the case where we want to use metadata for the certification status of document fragments. In Figure 1 we use the `hasState` key to say that a section has been approved by the TÜV, a specific certification agency. There are two concerns here. First, the `hasState` key has to be introduced and given a meaning, and same for the (complex) value `\statedocrd{\tuev}`. This meaning is given in the `certifiation` ontology which we imported via the `\importmodule` command. The ontology can be marked up in sTEX (see Figure 2), with the exception that we use the `\keydef` macro for the definition of the `hasState` relation so that it also defines the key. For the details of this see the next section.

2 User Interface

We now document the specifics of the environments and macros provided by the `rdfmeta` package from a user perspective.

2.1 Package Options

`showmeta` The `rdfmeta` package takes the option: `showmeta`. If this is set, then the metadata keys are shown (see [Koh10a] for details and customization options).

`sectioning` The remaining options can be used to specify metadata upgrades of standard keys. The `sectioning` option upgrades the `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph` macros (and of course their starred variants).

2.2 Extending Macros and Environments by Metadata Keys

\keydef

The main user-visible feature of the `rdfmeta` package is the `keydef` macro.

EdNote(1)

It takes two arguments, a key set identifier[11] and a key name. Semantically, `\keydef{`⟨*keyset*⟩`}{`⟨*key*⟩`}` defines a symbol just like the `\symdef` macro from the

EdNote(2)

`modules` package [KGA10][2]. But it also extends the syntax of SᴛEX itself: it adds a key ⟨*key*⟩ to ⟨*keyset*⟩, which allows to state the corresponding metadata as a key/-value pair in the SᴛEX macro or environment. Following the ideas from [LK09], the metadata is transformed to RDFa metadata [Adi+10] in OMDoc, where the identifiers of relations are exactly the symbols introduced by the corresponding `\keydef`.

```
\documentclass{omdoc}
\usepackage{stex,rdfmeta,amstext}
\begin{document}
\begin{module}[id=certification]
  \metalanguage[../owl2onto/owl2]{OWL2}
  \keydef{omtext}{hasState}
  \symdef{statedocrd}[1]{rd. #1}
  \symdef{tuev}{\text{T\"UV}}
  \begin{definition}[for=hasState]
    A document {\definiendum[hasState]{has state}} $x$, iff
    the project manager decrees it so.
  \end{definition}
  \begin{definition}[for=statedocrd,hasState=$\statedocrd\tuev$]
    A document has state \definiendum[statedocrd]{rd. $x$},
    iff it has been submitted to $x$ for certification.
  \end{definition}
  \begin{definition}[for=tuev,hasState=$\statedocrd\tuev$]
    The $\tuev$ (Technischer \"Uberwachungs Verein) is a well-known certification agency in
    Germany.
  \end{definition}
\end{module}
\end{document}
```

Example 2: A simple Ontology on Certification

In our example in Figure 2 we have defined a key `hasstate` in the `omtext` key set[2] and a symbol `hasstate` via `\addkey{omtext}{hasstate}`. Furthermore, we have defined the meaning of the relation expressed by the `hasstate` symbol informally and specified some possible objects for the relation (that could of course

EdNote(3)

have been done in other documents as well). We have made use[3] of this metadata ontology and the new key `hasstate` in the example in Figure 1.

[1] In a nutshell, every SᴛEX command that takes metadata keys comes with a "key set identifier" that identifies the set of admissible keys; see [Koh10a] for details on this concept

[1] EdNote: introduce this in the `metakeys` package also give an overview over the keysets in SᴛEX

[2] EdNote: as I have not been able to make the keyset a list of keysets, we have to have a non-defining version of this macro probably `\keydef*`.

[2] For the `\omtext` environment and key set see [Koh10d]

[3] EdNote: this does not quite work yet, since we do not know where the prefix `cert:` comes from. I guess we need to provide a variant of `importmodule` for certification that allows to specify the prefix. Of course this only affects the latexml transformation

2.3 Redefinitions of Common LaTeX Macros and Environments

The `rdfmeta` package redefines common LaTeX commands (e. g. the sectioning macros) so that they include optional KeyVal arguments that can be extended by `\keydef` commands. With this extension, we can add RDFa metadata to any existing LaTeX document and generate linked data (XHTML+RDFa documents) via the LaTeXML translator. The only thing needed[4]

EdNote(4)

2.4 Extending Packages with `rdfmeta`

The `rdfameta` package also exposes its internal infrastructure for extending the redefinitions. Note that the upgrade macros can only be used in LaTeX packages, as the macro names contain `@`. Consequently, this section is only addressed at package developers who want to extend existing (i.e. not written by them) packages with flexible metadata functionality.

`\rdfmeta@upgrade` `\rdfmeta@upgrade` is the basic upgrade macro. It takes an optional keyval argument an a command sequence ⟨*cseq*⟩ as a proper argument and (if that is defined), redefines \⟨*cseq*⟩ to take a keyval argument. There is a variant

`\rdfmeta@upgrade*` `\rdfmeta@upgrade*` that has to be used to upgrade macros that have a starred form (e.g. `\section` and friends). Note that `\rdfmeta@upgrade*` upgrades both forms (e.g. `\section` and `\section*`).

`\rdfmeta@upgrade` uses four keys to specify the behavior in the case the the macro to be upgraded already has an optional argument. For concreteness, we introduce them using the `\section` macro from standard LaTeX as an example. `\section` has an optional argument for the "short title", which

`optarg` will appear in the table of contents. The `optarg` key can be used to specify a key for the existing optional argument. Thus, after upgrading it via `\rdfmeta@upgrade*[optarg=short]{section}`, we can use the updated form `\section[short=`⟨*toctitle*⟩`]{`⟨*title*⟩`}` instead of the old `\section[`⟨*toctitle*⟩`]{`⟨*title*⟩`}`. Actually, this still has a problem: the `\section*` would also be given the `short` key and would be passed an optional argument (which it does not accept). To rem-

`optargstar` edy this we can set the `optargstar` key to `no`. In summary, the correct upgrade command for `\section` and `\section*` would be

```
\rdfmeta@upgrade*[optarg=short,optargstar=no]{section}
```

The `\rdfmeta@upgrade*` macro also initializes a metadata key-group (a named set of keys and their handlers; see [Koh10b] for details) for the section macro with an `id` key for identification (see [Koh10e] for details). Often, the name of the key-group is the same as the command sequence, so we take this as the default, if we want to specify a different metadata key-group name, we can do so with the `keygroup` key in `\rdfmeta@upgrade*`.

BegOP(5)

`label` If `idlabel` is set to ⟨*prefix*⟩, then the LaTeX label is set to the value

[4] EdNote: continue

[5] Old Part: not implemented or tested yet.

⟨*prefix*⟩.⟨*id*⟩, where ⟨*id*⟩ is the value given in the RDFa `id` key. This allows to use the normal LaTeX referencing mechanism in addition to the semantic referencing mechanism provided by the `sref` package [Koh10f].

EndOP(5)

2.5 Limitations

1. Currently the coverage of the redefinitions of standard commands in the `rdfmeta` package is minimal; we will extend this in the future.
2. The `\rdfmeta@upgrade` macro only works with single arguments, this should be easy to fix with `\case` for the argument string.
3. I am not sure `\rdfmeta@upgrade` works with environments.

3 The Implementation

The `sref` package generates two files: the LaTeX package (all the code between ⟨*package⟩ and ⟨/package⟩) and the LaTeXML bindings (between ⟨*ltxml⟩ and ⟨/ltxml⟩). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

We first set up header information for the LaTeXML binding file.

```
⟨*ltxml⟩
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
⟨/ltxml⟩
```

3.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).[6]

EdNote(6)

```
⟨*package⟩
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\newif\if@rdfmeta@sectioning\@rdfmeta@sectioningfalse
\DeclareOption{sectioning}{\@rdfmeta@sectioningtrue}
\ProcessOptions
⟨/package⟩
⟨*ltxml⟩
DeclareOption('showmeta','');
DeclareOption('sectioning','');
⟨/ltxml⟩
```

The first measure is to ensure that the right packages are loaded.

```
⟨*package⟩
\RequirePackage{sref}
```

[6] EdNote: need an implementation for LaTeXML

```
⟨/package⟩
```

3.2 RDFa upgrade Facilities

We first define the keys for the `\rdfmeta@upgrade` macro.

```
⟨*package⟩
\def\@yes@{yes}
\addmetakey*{upgrade}{idlabel}
\addmetakey*{upgrade}{optarg}
\addmetakey*[yes]{upgrade}{optargstar}
\addmetakey*{upgrade}{keygroup}
```

`\rdfmeta@upgrade` This upgrade macro gives extended functionality according to the optional keys. The top-level invocation just differentiates on whether a star is following:

```
\def\rdfmeta@upgrade{\@ifstar\rdfmeta@upgrade@star\rdfmeta@upgrade@nostar}
```

Both cases are almost the same, they only differ in the third line where they call `\rdfmeta@upgrade@base` or `\rdfmeta@upgrade@base@star` defined above. In particular, both take the arguments originally intended for `\rdfmeta@upgrade`.

```
\newcommand\rdfmeta@upgrade@nostar[2][]{\metasetkeys{upgrade}{#1}%
\ifx\upgrade@keygroup\@empty\def\@@group{#2}\else\def\@@group{\upgrade@keygroup}\fi
\rdfmeta@upgrade@base{#2}{\@nameuse{\@@group @\upgrade@optarg}}}
```

They set the metakeys from the second argument, then set `\@@group` to be the intended group (if the `keygroup` key was specified, it takes precedence over the default `#2`).

```
\newcommand\rdfmeta@upgrade@star[2][]{\metasetkeys{upgrade}{#1}%
\ifx\upgrade@keygroup\@empty\def\@@group{#2}\else\def\@@group{\upgrade@keygroup}\fi
\rdfmeta@upgrade@base@star{#2}{\@nameuse{\@@group @\upgrade@optarg}}}
⟨/package⟩
⟨*ltxml⟩
⟨/ltxml⟩
```

`\rdfmeta@upgrade@base` This auxiliary macro and is invoked as `\rdfmeta@upgrade@base{`⟨*cseq*⟩`}{`⟨*optarg*⟩`}`, where ⟨*cseq*⟩ is a command sequence name. It checks if `\`⟨*cseq*⟩ is defined (if not it does nothing), saves the old behavior of `\`⟨*cseq*⟩ as `\rdfmeta@`⟨*cseq*⟩`@old`, and then redefines `\`⟨*cseq*⟩ to take a keyval argument and passes ⟨*optarg*⟩ as the optional argument.

```
⟨*package⟩
\newcommand{\rdfmeta@upgrade@base}[2]{\@ifundefined{#1}{}%
{\srefaddidkey{#1}%
\expandafter\let\csname rdfmeta@#1@old\expandafter\endcsname\csname #1\endcsname%
\expandafter\renewcommand\csname #1\endcsname[2][]%
{\metasetkeys{#1}{##1}\@nameuse{rdfmeta@#1@old}[#2]{##2}}
\addmetakey*\@@group{\upgrade@optarg}}}
```

`\rdfmeta@upgrade@base@star` This is a variant of `\rdfmeta@upgrade@base`, which also takes care of the starred variants of a macro.

```
\newcommand{\rdfmeta@upgrade@base@star}[2]{\@ifundefined{#1}{}%
```

```
{\srefaddidkey{#1}%
\expandafter\let\csname rdfmeta@#1@old\expandafter\endcsname\csname #1\endcsname%
```

In this case, we cannot just use `\newcommand` for dealing with the optional argument because the star is between the command sequence and the arguments. So we make a case distinction on the presence of the star. `\rdfmeta@`⟨*cseq*⟩`@old`.

```
\expandafter\def\csname #1\endcsname%
{\@ifstar{\@nameuse{rdfmeta@#1@star}}{\@nameuse{rdfmeta@#1@nostar}}}%
```

the macros `\rdfmeta@`⟨*cseq*⟩`@star` and `\rdfmeta@`⟨*cseq*⟩`@nostar` that are defined in terms of `\rdfmeta@`⟨*cseq*⟩`@old` handle the necessary cases. The second one is simple:

```
\expandafter\newcommand\csname rdfmeta@#1@nostar\endcsname[2][]%
{\metasetkeys{#1}{##1}\@nameuse{rdfmeta@#1@old}[#2]{##2}}%
```

For `\rdfmeta@`⟨*cseq*⟩`@star` we have to take care of the optional argument of the old macro: if the `optargstar` key was set, then we pass the second argument of `\rdfmeta@upgrade@base` as an optional argument to it as above.

```
\ifx\upgrade@optargstar\@yes@%
\expandafter\newcommand\csname rdfmeta@#1@star\endcsname[2][]%
{\metasetkeys{#1}{##1}\@nameuse{rdfmeta@#1@old}*[#2]{##2}}%
\else%
\expandafter\newcommand\csname rdfmeta@#1@star\endcsname[2][]%
{\metasetkeys{#1}{##1}\@nameuse{rdfmeta@#1@old}*{##2}}%
\fi%
\addmetakey*\@@group{\upgrade@optarg}}}
⟨/package⟩
```

3.3 Key Definitions

`\keydef` The `\keydef` macro is rather simple, we just add a key to the respective environment and do a `\symdef`.

```
⟨*package⟩
\newcommand\keydef[2]{\addmetakey{#1}{#2}\symdef{#2}{\text{#2}}}
⟨/package⟩
⟨*ltxml⟩
⟨/ltxml⟩
```

3.4 Redefinitions

If the `sectioning` macro is set, we redefine the respective commands

```
⟨*package⟩
\if@rdfmeta@sectioning
\rdfmeta@upgrade*[optarg=short,optargstar=no]{part}
\rdfmeta@upgrade*[optarg=short,optargstar=no]{chapter}
\rdfmeta@upgrade*[optarg=short,optargstar=no]{section}
\rdfmeta@upgrade*[optarg=short,optargstar=no]{subsection}
\rdfmeta@upgrade*[optarg=short,optargstar=no]{subsubsection}
\rdfmeta@upgrade*[optarg=short,optargstar=no]{paragraph}
```

71 \fi
72 ⟨/package⟩
73 ⟨*ltxml⟩
74 ⟨/ltxml⟩

3.5 Finale

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

75 ⟨ltxml⟩1;

References

[Adi+10] Ben Adida et al. *RDFa Core 1.1. Syntax and processing rules for embedding RDF through attributes*. W3C Working Draft. World Wide Web Consortium (W3C), Aug. 3, 2010. URL: http://www.w3.org/TR/2010/WD-rdfa-core-20100803/.

[KGA10] Michael Kohlhase, Deyan Ginev, and Rares Ambrus. *modules.sty: Semantic Macros and Module Scoping in sTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: http://www.ctan.org/get/macros/latex/contrib/stex/modules/modules.pdf.

[KKL10] Andrea Kohlhase, Michael Kohlhase, and Christoph Lange. "sTeX – A System for Flexible Formalization of Linked Data". In: *Proceedings of the 6th International Conference on Semantic Systems (I-Semantics) and the 5th International Conference on Pragmatic Web*. Ed. by Adrian Paschke et al. ACM, 2010. ISBN: 978-1-4503-0014-8. arXiv: 1006.4474v1.

[Koh10a] Michael Kohlhase. *metakeys.sty: A generic framework for extensible Metadata in LaTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: http://www.ctan.org/tex-archive/macros/latex/contrib/stex/metakeys/metakeys.pdf.

[Koh10b] Michael Kohlhase. *metakeys.sty: A generic framework for extensible Metadata in LaTeX*. Self-documenting LaTeX package. 2010. URL: https://svn.kwarc.info/repos/stex/trunk/sty/metakeys/metakeys.pdf.

[Koh10c] Michael Kohlhase. *metakeys.sty: A generic framework for extensible Metadata in LaTeX*. Self-documenting LaTeX package. 2010.

[Koh10d] Michael Kohlhase. *omtext: Semantic Markup for Mathematical Text Fragments in LaTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: http://www.ctan.org/tex-archive/macros/latex/contrib/stex/omtext/omtext.pdf.

[Koh10e] Michael Kohlhase. *sref.sty: Semantic Crossreferencing in LaTeX*. Self-documenting LaTeX package. 2010. URL: https://svn.kwarc.info/repos/stex/trunk/sty/sref/sref.pdf.

[Koh10f] Michael Kohlhase. *sref.sty: Semantic Crossreferencing in LaTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: http://www.ctan.org/tex-archive/macros/latex/contrib/stex/sref/sref.pdf.

[Koh10g] Michael Kohlhase. *sref.sty: Semantic Crossreferencing in LaTeX*. Self-documenting LaTeX package. 2010.

[LK09] Christoph Lange and Michael Kohlhase. "A Mathematical Approach to Ontology Authoring and Documentation". In: *MKM/Calculemus Proceedings*. Ed. by Jacques Carette et al. LNAI 5625. Springer Verlag, July 2009, pp. 389–404. URL: https://svn.omdoc.org/repos/omdoc/trunk/doc/blue/foaf/mkm09.pdf.