RDFa Metadata in \LaTeX\*

Michael Kohlhase
FAU Erlangen-Nürnberg
http://kwarc.info/kohlhase

March 20, 2019

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

Contents
1 Introduction 2
2 User Interface 2
  2.1 Package Options .............................................. 2
  2.2 Extending Macros and Environments by Metadata Keys 3
  2.3 Redefinitions of Common \LaTeX\ Macros and Environments 4
  2.4 Extending Packages with \texttt{rdfmeta} .................. 5
  2.5 Limitations ................................................. 5
3 The Implementation 6
  3.1 Package Options .............................................. 6
  3.2 Key Definitions ............................................. 7
  3.3 RDFa upgrade Facilities .................................. 7
  3.4 Redefinitions ............................................... 8

Experimental!
do not use!

*Version v0.2 (last revised 2019/03/20)
1 Introduction

The \texttt{rdfmeta} package allows mark up extensible metadata in \LaTeX{} documents, so that it can be harvested by automated tools or exported to PDF. It is also intended to support the new metadata infrastructure for the OMDoc format [Koh06 \textsuperscript{1}] introduced in OMDoc1.3 [Koh10 \textsuperscript{2}] (see [LK09 \textsuperscript{3}] for the relevant ideas and [KKL10 \textsuperscript{4}] for an application).

Metadata are annotated as key value pairs in the semantic environments provided by \LaTeX{}. In most markup formats, the metadata vocabularies are fixed by the language designer. In \LaTeX{}, the \texttt{rdfmeta} package allows the user to extend the metadata vocabulary.

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

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

\textbf{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 \ref{fig:.certification} we use the \texttt{hasState} key to say that a section has been approved by the TÜV, a specific certification agency. There are two concerns here. First, the \texttt{hasState} key has to be introduced and given a meaning, and same for the (complex) value \texttt{statedocrd\{\text{tuev}\}}. This meaning is given in the \texttt{certification} ontology which we imported via the \texttt{\importmodule} command. The ontology can be marked up in \LaTeX{} (see Figure \ref{fig:ontology}), with the exception that we use the \texttt{\keydef} macro for the definition of the \texttt{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 \texttt{rdfmeta} package from a user perspective.

2.1 Package Options

\texttt{showmeta} The \texttt{rdfmeta} package takes the option: \texttt{showmeta}. If this is set, then the metadata
keys are shown (see [Koh16a] for details and customization options).

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 \texttt{rdfmeta} package is the \texttt{keydef} macro. It takes two arguments, a “key group identifier” and a key name. In a nutshell, every \texttt{\LaTeX} command that takes metadata keys comes with a “key group identifier” that identifies the set of admissible keys; see [Koh16a] for details on this concept. Figure 1 gives an overview over the key groups and their identifiers in \texttt{\LaTeX}.

Semantically, \texttt{\keydef{(keygroup)}{(key)}} defines a symbol just like the \texttt{\symdef} macro from the \texttt{modules} package [KGA16]. But it also extends the syntax of \texttt{\LaTeX} itself: it adds a key \texttt{(key)} to \texttt{(keygroup)}, which allows to state the corresponding metadata as a key/value pair in the \texttt{\LaTeX} macro or environ-
ment. 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}
  \keydef{omgroup}{hasState}
  \symdef{hasState}{\text{hasState}}
  \symdef{statedocrd}[1]{rd. \#1}
  \symdef{tuev}{\text{T"UV}}
  %begin{omgroup}[id=foo,hasState=test]{Definitions}
  \begin{definition}[for=hasState]
  A document \defii{has}{state} $x$, iff the project manager decrees it so.
  \end{definition}
  \begin{definition}[for=statedocrd,hasState=$\statedocrd\tuev$]
  A document has state \defi{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 national certification agency in Germany.
  \end{definition}
\end{omgroup}
\end{module}

%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:

\textbf{Example 2: A simple Ontology on Certification}

In our example in Figure 2 we have defined a key \texttt{hasstate} in the \texttt{omtext} key group\footnote{For the \texttt{omtext} environment and key group see [Koh16d]} and a symbol \texttt{hasstate} via \texttt{\addkey{omtext}{hasstate}}. Furthermore, we have defined the meaning of the relation expressed by the \texttt{hasstate} symbol informally and specified some possible objects for the relation (that could of course have been done in other modules as well). We have made use of this metadata ontology and the new key \texttt{hasState} in the example in Figure 1.

2.3 Redefinitions of Common \LaTeX\ Macros and Environments

The \texttt{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 \LaTeXXML translator.

2.4 Extending Packages with rdfmeta

The rdfmeta 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.

\texttt{\backslash rdfmeta@upgrade} is the basic upgrade macro. It takes an optional keyval argument an a command sequence \texttt{(cseq)} as a proper argument and (if that is defined), redefines \texttt{\langle cseq \rangle} to take a keyval argument. There is a variant \texttt{\backslash rdfmeta@upgrade*} that has to be used to upgrade macros that have a starred form (e.g. \texttt{section} and friends). Note that \texttt{\backslash rdfmeta@upgrade*} upgrades both forms (e.g. \texttt{section} and \texttt{section*}).

\texttt{\backslash rdfmeta@upgrade} uses four keys to specify the behavior in the case the macro to be upgraded already has an optional argument. For concreteness, we introduce them using the \texttt{section} macro from standard \LaTeX as an example. \texttt{section} has an optional argument for the “short title”, which will appear in the table of contents. The \texttt{optarg} key can be used to specify a key for the existing optional argument. Thus, after upgrading it via \texttt{\backslash rdfmeta@upgrade*[optarg=short]{section}}, we can use the updated form \texttt{\langle section\short\langle toctitle \rangle\rangle{\langle title \rangle}} instead of the old \texttt{\langle section\langle toctitle \rangle\rangle{\langle title \rangle}}.

Actually, this still has a problem: the \texttt{section*} would also be given the \texttt{short} key and would be passed an optional argument (which it does not accept). To remedy this we can set the \texttt{optargstar} key to \texttt{no}. In summary, the correct upgrade command for \texttt{section} and \texttt{section*} would be

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

The \texttt{\backslash rdfmeta@upgrade*} macro also initializes a metadata key-group (a named set of keys and their handlers; see [Koh16a] for details) for the section macro with an \texttt{id} key for identification (see [Koh16b] 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

\texttt{\backslash keygroup key in \backslash rdfmeta@upgrade*}.

\texttt{idlabel} is set to \texttt{(prefix).\langle id \rangle}, where \texttt{(id)} is the value given in the RDFa \texttt{id} key. This allows to use the normal \LaTeX referencing mechanism in addition to the semantic referencing mechanism provided by the \texttt{sref} package [Koh16c].

2.5 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
discussed in the \TeX\ GitHub repository \texttt{sTeX}.

1. Currently the coverage of the redefinitions of standard commands in the \texttt{rdfmeta} package is minimal; we will extend this in the future.

2. The \texttt{\textbackslash rdfmeta@upgrade} macro only works with single arguments, this should be easy to fix with \texttt{\textbackslash case} for the argument string.

3. I am not sure \texttt{\textbackslash rdfmeta@upgrade} works with environments.

4. It would be convenient, if we had a macro \texttt{\textbackslash keydefs}, which takes a list of keygroups, so that we can define keys in multiple groups in one go, e.g. \texttt{\textbackslash keydefs\{omtext,omgroup\}\{hasState\}} in Figure 2. But the obvious “solution”

\begin{verbatim}
\newcommand\keydefs[2]{\@for\@I:=#1\do{\keydef{#1}{#2}}}
\end{verbatim}

does not work for me.

\section{The Implementation}

\subsection{Package Options}

We declare some switches which will modify the behavior according to the package options. Generally, an option \texttt{xxx} will just set the appropriate switches to true (otherwise they stay false).\footnote{\textit{EdNote}: need an implementation for \LaTeXXML.}

\begin{verbatim}
\newif\if@rdfmeta@sectioning\@rdfmeta@sectioningfalse
\DeclareOption{sectioning}{\@rdfmeta@sectioningtrue}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{sref}}
\PassOptionsToPackage{\CurrentOption}{modules}}
\ProcessOptions
\end{verbatim}

The first measure is to ensure that the right packages are loaded. From the \TeX\ collection, we need the \texttt{sref} package (see \texttt{\cite{Koh16}}) for handling keys, the \texttt{modules} package for exporting the \texttt{\keydef} (see \texttt{\cite{KGA16}}).

\begin{verbatim}
\RequirePackage{sref}
\RequirePackage{modules}
\end{verbatim}

and we define a macro \texttt{\textbackslash rdfmeta@loaded} just for the purpose of determining whether the \texttt{rdfmeta} package is loaded.

\begin{verbatim}
\newcommand\rdfmeta@loaded{yes}
\end{verbatim}

And another macro \texttt{\textbackslash rdfmeta@sectioning} to determine whether the sectioning macros have been redefined.
\section*{3.2 Key Definitions}

\texttt{\keydef} The \texttt{\keydef} macro is rather simple, we just add a key to the respective environment and extend the export token register for the current module by an \texttt{\addmetakey} instruction.

\begin{verbatim}
\newcommand\keydef[2]{\addmetakey{#1}{#2}%
\expandafter\g@addto@macro\this@module{\addmetakey{#1}{#2}}}
\end{verbatim}

\texttt{\listkeydef} The \texttt{\listkeydef} macro is analogous, but uses \texttt{\addmetalistkey} instead.

\begin{verbatim}
\newcommand\listkeydef[2]{\addmetalistkey{#1}{#2}%
\expandafter\g@addto@macro\this@module{\addmetalistkey{#1}{#2}}}
\end{verbatim}

\section*{3.3 RDFa upgrade Facilities}

We first define the keys for the \texttt{\rdfmeta@upgrade} macro.

\begin{verbatim}
\def\@yes@{yes}
\addmetakey*{upgrade}{idlabel}
\addmetakey*{upgrade}{optarg}
\addmetakey*[yes]{upgrade}{optargstar}
\addmetakey*{upgrade}{keygroup}
\end{verbatim}

\texttt{\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:

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

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

\begin{verbatim}
\newcommand\rdfmeta@upgrade@base[2][\@nameuse{\@@group @\upgrade@optarg}]
\end{verbatim}

This auxiliary macro and is invoked as \texttt{\rdfmeta@upgrade@base{\langle cseq\rangle}{\langle optarg\rangle}}, where \texttt{\langle cseq\rangle} is a command sequence name. It checks if \texttt{\langle cseq\rangle} is defined (if not it does nothing), saves the old behavior of \texttt{\langle cseq\rangle} as \texttt{\rdfmeta@\langle cseq\rangle@old}, and then redefines \texttt{\langle cseq\rangle} to take a keyval argument and passes \texttt{\langle optarg\rangle} as the optional argument.
This is a variant of \texttt{\textbackslash rdfmeta@upgrade}@\texttt{base}, which also takes care of the starred variants of a macro.

In this case, we cannot just use \texttt{\textbackslash 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. \texttt{\textbackslash rdfmeta@\langle cseq \rangle@old}.

the macros \texttt{\textbackslash rdfmeta@\langle cseq \rangle@star} and \texttt{\textbackslash rdfmeta@\langle cseq \rangle@nostar} that are defined in terms of \texttt{\textbackslash rdfmeta@\langle cseq \rangle@old} handle the necessary cases. The second one is simple:

For \texttt{\textbackslash rdfmeta@\langle cseq \rangle@star} we have to take care of the optional argument of the old macro: if the \texttt{optargstar} key was set, then we pass the second argument of \texttt{\textbackslash rdfmeta@upgrade}@\texttt{base} as an optional argument to it as above.

3.4 Redefinitions

If the \texttt{sectioning} macro is set, we redefine the respective commands

\texttt{\textbackslash if@\textbackslash rdfmeta@sectioning}

\texttt{\textbackslash message\{redefining sectioning commands!\}}

\texttt{\textbackslash rdfmeta@Upgrade*[optarg=short, optargstar=no]\{part\}}

\texttt{\textbackslash rdfmeta@Upgrade*[optarg=short, optargstar=no]\{chapter\}}

\texttt{\textbackslash rdfmeta@Upgrade*[optarg=short, optargstar=no]\{section\}}

\texttt{\textbackslash rdfmeta@Upgrade*[optarg=short, optargstar=no]\{subsection\}}
References


[sTeX] KWARC/sTeX. URL: https://github.com/KWARC/sTeX (visited on 05/15/2015).