`sref.sty`: Semantic Cross-Referencing in LaTeX*

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

November 17, 2015

Abstract

The `sref` package is part of the STEX collection, a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM).

The `sref` package supplies an for semantic cross-referencing over multiple documents.

Contents

---

*Version v1.2 (last revised 2015/10/27)

1   Introduction

The automatic computation of cross-references is one of the traditional strong points of LaTeX. However, cross-referencing is limited to labels in the current document only. Cross-referencing between multiple documents in a jointly developed document collection is not easy to achieve in the LaTeX processing model, which reads files sequentially and lacks a path concept.

The `sref` package is mainly aimed at package developers. It supplies the internal macros that can be used to make document structuring elements cross-referencable. The general strategy here is to equip the document structuring `id` macros with an `id` key, so that the author can specify meaningful ones, but to let the transformation give default ones if the author did not. The value of the `id` key can also be used for cross-referencing like the `\label`/`\ref` mechanism in LaTeX. We implement an independent referencing mechanism, since the referencing model is geared more towards referencing text fragments than text fragment labels like section numbers. Therefore we let the referenced fragments define the reference text much like the `\autoref` macro from `\hpyerref`.

2   The User Interface

This package is currently mainly meaningful in the context of the sTeX collection, since all cross-referencable macros and environments must be extended to know about their referencing mechanism. We explain the user interface in Section 2.2. To port the functionality to other LaTeX classes and packages, they have to be upgraded via the API in Section 2.3.

2.1   Package Options

`extrefs` The `sref` package has the `extrefs` package option, which can be set to activate
`showmeta` multi-file support (see Section 2.4). If the `showmeta` is set, then the metadata keys are shown (see [Koh15] for details and customization options).

2.2   Cross-Referencing

`\sref` The `\sref{`⟨*id*⟩`}` macro is the main cross-referencing macro, see Figure 1 for an example. Depending on the whether macro or environment marking up the respective document fragment carries the key/value pair `id=`⟨*id*⟩ the cross-reference will expand to "Section 2.1" or "this remark", both carrying hyper-references. The `\sref` macro takes an optional key/value argument that allows to customize its behavior: The `linktext` key can be used to specify a link text that overrides the auto-generated one and the `fallback` allows to give the fallback text when the label ⟨*id*⟩ is not defined. The `pre` and `post` keys allow to specify text that is pre/postpended to the generated label. This is particularly useful if combined with the `fallback` option as in the last `\sref` in Figure 1, where the generated "in Section. . . " is equivalent to the fallback "above".

```
\mysection[id=foo]{#2}
... \sref{foo} ...
... \sref[linktext=this section]{foo} ...
... \sref[pre={in~},fallback=above]{foo} ...
```
Example 1: Semantic Crossreferencing

\sreflabel   The `\sreflabel`[1] macro is a variant to the `\label` macro provided by LaTeX proper. It takes two arguments, the first one is a classification (used in `\sref`) and the second one the identifier.

\srefs   The `\srefl{`$\langle id^1\rangle$`}{`$\langle id^2\rangle$`}` is a variant it `\sref`, only that it allows to reference two semantic objects and expands to "$\langle reference^1\rangle$ and $\langle reference^2\rangle$".

\srefl   `\srefl{`$\langle id^1\rangle$`}{`$\langle id^n\rangle$`}` is similar, but for ranges; it expands to "$\langle reference^1\rangle$ to $\langle reference^n\rangle$". Its use should be restricted to cases, where the types of objects references are homogenous.

\spageref   Finally, there is a variant `\spageref` that only outputs the page number of the referenced object. It can be used in cases where no hyper-referencing is present.

\sref@page@label   It uses the macro `\sref@page@label` for styling the page reference. Redefining this will allows to customize this. The default setting is

```
\newcommand\sref@page@label[1]{p.~{#1}}
```

2.3 An API for Package Authors

To make use of the `sref` package, the package must define the document structuring infrastructure using the `sref` internal macros. The STEX packages already does this, so we make an example here for a slightly upgraded sectioning command in Figure 2. The first three lines define the keys for the keyval attribute of the `\mysection` command using the infrastructure supplied by the `omd` package [Koh15] (remember the `\RequirePackage{metakeys}`). The first two just initialize the keys to save the key values in internal macros, and the `\metasetkeys` activates the keys when reading the keyval argument. The `\srefaddidkey` macro

\srefaddidkey   is a variant of `\addmetakey` macro supplied by the `sref` package that sets up the

\sref@id   keys to set the `\sref@id` register for later use by the `sref` infrastructure. Note that the `\srefaddidkey` macro uses the `prefix` key to systematically construct prefixed identifiers. This can be useful in particular for sectioning commands.

```
\addmetakey{sec}{short}
\addmetakey[black]{sec}{color}
\srefaddidkey[prefix=sec.]{sec}
\newcommand\mysection[2][]{\metasetkeys{#1}\sref@target\color{\sec@color}
\section[\sec@short]{#2}\sref@label@id{Section \thesection}}
```
Example 2: A slightly upgraded sectioning command

---

[1]It would have been more natural to name the macro `slabel`, but this is overwritten by other packages without warning.

In this situation, the `\mysection` macro processes the optional argument with
`\metasetkeys` and then sets the color of the section. The `\sref@target` sets
up the hyper-target for the `hyperref` package to use. Then we use the regular
`\section` command, and we use the `\sref@label@id` macro to define the label
that the `\sref` macro will use for cross-referencing.

Note that the use of the straight use of the label "Section", which will be
written into the auxiliary files is bad practice since it is not configurable. It
would be much better to make it configurable via a presentation macro like
`\my@section@label` in Figure 3. Then translators or even the user could redefine
the `\my@section@label` to adapt them to their needs.

```
\newcommand\my@section@label[1]{Section~{#1}}
\newcommand\mysection[2][]{\metasetkeys{#1}\sref@target\color{\sec@color}
\section[\sec@short]{#2}\sref@label@id{\my@section@label\thesection}}
```

Example 3: A Sectioning Command with Configurable Label

2.4   Inter-Document Cross-Referencing

`sref.sty` provides inter-document cross-referencing. The use case is simple: we
want to have a document collection (e.g. a book with conference proceedings),
but also want to be able format the individual documents separately, and still
have meaningful cross-references. To show off the possibilities, let us assume that
we have a book with two separate papers, which we put into separate directories
`idc` and `scr` to minimize interference between the authors Jane Doe and John
Bull. To achieve this, we would set up paper driver files `main.tex` like the one
in Figure 4 in the two directories. These use the `\makeextrefs` macro, which
causes the `sref` package to generate a *external references file* `main.refs`. Note
that the `\makeextrefs` macro reads the previous `main.refs` file so that forward-
referencing is possible (in the pass after a reference was labeled).

```
\documentclass{article}
\usepackage[extrefs]{sref}
\makeextrefs{idc}
\inputrefs{scr}{../scr/main}
\extrefstyle{scr}{\cite[\protect{\theextref}]{Doe09}}
\title{Inter-Document Crossreferencing}
\author{John Bull\\...}
\begin{document}\maketitle\input{paper}\end{document}
```

Example 4: A document driver `idc/main.tex` for a paper

The external references file can be read by other documents; in Figure 4, we
read the references file of Jane Doe's paper via the `\inputrefs` macro. This
allows John Bull to use[2] references like `\extref{scr}{foo}` to reference doc-

---

[2]Note that the external references file is updated every time LATEX is run, so that references
may be off by one version.

4

ument fragments in Jane Doe's paper she has labeled with the *reference prefix* `\sreflabel{foo}` (assuming that she has added `\makeextrefs{scr}` in the preamble of her paper). Note that just as the `\sref` macro `\extref` takes an optional first argument that allows to specify the link text. Here, John Bull uses the `\extrefstyle` macro to specify how the external references are to be formatted, in this case he decided to use a LaTeX citation. Generally, first argument of the `\extrefstyle` macro is the reference prefix which should be configured, and the second is the format, where the `\theextref` macro expands to the cross-reference. In this case, John chose to use a bibTeX citation (he has an entry `Doe09` in his database) for the reference to the external paper.

As the content of the respective paper is input from a file `paper.tex` in the individual papers, we can re-use these in the book. To do this we set up a book driver file like the one in Figure 5. This one does not use the `extrefs` option, so the references are written to the `.aux` file. Furthermore `\extref` is redefined to act like `\sref` disregarding the first required argument. Thus all references work like they should.

```
\documentclass{book}
\usepackage{sref}
\title{Cross-Referencing in {\LaTeX}}
\author{Elder Reseacher}
\begin{document}
\maketitle
\chapter{Semantic Crossreferencing (Jane Doe, ...)}
\input{scr/paper}\newpage
...
\chapter{Inter-Document Crossreferencing (John Bull, ...)}
\input{idc/paper}\newpage
\end{document}
```

Example 5: A document driver for the book assembling the papers

This example has been carried through (without the separation of chapters in to subdirectories) in the files accompanying the source distribution of the `sref` package. They are used for testing the package.

2.5 Semantic Versions of Commonly used Referencing Commands

The `sref` package defines semantically referencable versions of commonly used LaTeX environments and command sequences.[3]

The `sequation` environment takes an optional key/value argument that allows to specify an identifier and unifies the behavior of the `equation` (if an `id` key is given) and `displaymath` (else) environments. So the markup

----

[3]This section will be extended by need, so if you miss some semantic environment, please contact the package author, or (better) file an issue at [**sTeX:online**])

```
 A semantic equation with id
 \begin{sequation}[id=foo]
   e^{mc}=-1
 \end{sequation}
 and another one without id
 \begin{sequation}
   e^{mc}=-1
 \end{sequation}
 now, we reference the first equation: \sref{foo}
```

yields the result:

A semantic equation with id

$$e^{mc} = -1 \tag{1}$$

and another one without id

$$e^{mc} = -1$$

now, we reference the first equation: equation (1)

Example 6: Semantic Equation

2.6 Semantic Citations

bibTeX [Pat] and bibLaTeX [Leh10] provide a semi-semantic way of referencing
literature. If we look at the current practice of citing from an RDF stand-
point [LS99]which views links as subject/predicate/object triples, then the treat-
ment of the predicate and object are semantic, but the subject is hinted at by mere
\withcite juxtaposition in the text. The `sref` package helps out here via the macro for short
subjects (in the second argument) that are postfixed by the citation (key in the
first argument). For instance the occurrence at the beginning of this paragraph
was created by

```
\withcite{Patashnik:b88}{bib\TeX}
```

withcitation The general case is covered by the `withcitation` environment for long subjects.
\citeit In the latter, the citation can be placed by the . For instance, the second sentence
was marked up as

```
If we look at the
\begin{withcitation}{LasSwi:rdf99}
   current practice of citing from an RDF standpoint which views links as
   subject/predicate/object triples,
\end{withcitation}
then the treatment of the predicate ...
```

The advantage of this treatment is that the meaning of the reference is fully marked
up and can be taken advantage of in the OMDoctransformation, from which RDF
triples can then be harvested for a linked open data treatment.

3 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 sTeX GitHub repository [sTeX].

1. currently only the `\sref` macro has a fallback argument. The others `\srefs` and `\srefl` and their external variants should also have them, but I am not clear what the adequate invocation pattern would be.

4 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.

```
 1 ⟨*ltxml⟩
 2 # -*- CPERL -*-
 3 package LaTeXML::Package::Pool;
 4 use strict;
 5 use LaTeXML::Package;
 6 ⟨/ltxml⟩
```

4.1 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).[1]

EdN:1

```
 7 ⟨*package⟩
 8 \newif\ifextrefs\extrefsfalse
 9 \DeclareOption{extrefs}{\extrefstrue}
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{metakeys}}
11 \ProcessOptions
12 ⟨/package⟩
13 ⟨*ltxml⟩
14 DefConditional('\if@extrefs');
15 DeclareOption('extrefs', sub{Digest(T_CS('\@extrefstrue')); });
16 DeclareOption(undef,sub {PassOptions('metakeys','sty',ToString(Digest(T_CS('\CurrentOption'))))
17 ProcessOptions();
18 ⟨/ltxml⟩
```

Then we need to set up the packages by requiring the `metakeys` package [Koh15] to be loaded (in the right version).

```
19 ⟨*package⟩
20 \RequirePackage{metakeys}
21 \RequirePackage{xspace}
22 \RequirePackage{etoolbox}
23 ⟨/package⟩
24 ⟨*ltxml⟩
25 RequirePackage('metakeys');
26 ⟨/ltxml⟩
```

---

[1]EDNOTE: need an implementation for LaTeXML

4.2 Crossreferencing

The following user-level macros just use the `\sref@hlink` macros in various ways

for internal referencing.[2]

`\sref` First we declare the keys for `\sref`. `\sref@part` will be globally defined when `\makeextrefs` is called.

```
27 ⟨*package⟩
28 \addmetakey{sref}{linktext}
29 \addmetakey{sref}{fallback}
30 \addmetakey{sref}{pre}
31 \addmetakey{sref}{post}
32 \newcommand\sref[2][]{%
33   \metasetkeys{sref}{#1}%
34   \ifcsundef{sref@part}{%
35     \sref@hlink[\sref@linktext]{#2}{\sref@fallback}{\sref@pre}{\sref@post}%
36   }{%
37     \sref@hlink[\sref@linktext]{\sref@part @#2}{\sref@fallback}{\sref@pre}{\sref@post}%
38   }%
39   \xspace%
40 }%
41 ⟨/package⟩
42 ⟨*ltxml⟩
43 DefKeyVal('sref','linktext','Semiverbatim');
44 DefKeyVal('sref','fallback','Semiverbatim');
45 DefKeyVal('sref','pre','Semiverbatim');
46 DefKeyVal('sref','post','Semiverbatim');
47 sub withhash {'#';}
48 DefConstructor('\sref OptionalKeyVals:sref {}[]',
49  "<omdoc:oref ?&GetKeyVal(#1, 'part')(href='&GetKeyVal(#1, 'part')&withhash()#2')(href='&withha
50 ."?&GetKeyVal(#1, 'fallback')(<omdoc:fallback>&GetKeyVal(#1, 'fallback')</omdoc:fallback>)()"
51 ."?&GetKeyVal(#1, 'pre')(<omdoc:pre>&GetKeyVal(#1, 'pre')</omdoc:pre>)()"
52 ."?&GetKeyVal(#1, 'post')(<omdoc:post>&GetKeyVal(#1,'post')</omdoc:post>)()"
53 ."</omdoc:oref>");
54 Tag('omdoc:oref',afterOpen=>\&numberIt,afterClose=>\&locateIt,autoClose=>1);
55 ⟨/ltxml⟩
```

`\srefs`

```
56 ⟨*package⟩
57 \newcommand\srefs[3][]{%
58   \def\@test{#1}%
59   \ifx\@test\@empty%
60     \sref{#2} and \sref{#3}%
61   \else%
62     #1%
63   \fi%
64 }%
```

---

[2]EDNOTE: they need implementation in LaTeXML, the ones here only are stubs to make the error messages shut up.

```
65 ⟨/package⟩
66 ⟨∗ltxml⟩
67 DefConstructor('\srefs[]{}',
68   "<omdoc:oref href='&withhash()#2'/>");
69 ⟨/ltxml⟩
```

\srefl

```
70 ⟨∗package⟩
71 \newcommand\srefl[3][]{%
72   \def\@test{#1}%
73   \ifx\@test\@empty%
74     \sref{#2} to \sref{#3}%
75   \else%
76     #1%
77   \fi%
78 }%
79 ⟨/package⟩
80 ⟨∗ltxml⟩
81 DefConstructor('\srefl[]{}',
82   "<omdoc:oref href='&withhash()#2'/>");
83 ⟨/ltxml⟩
```

EdN:3   ³

\spageref

```
84 ⟨∗package⟩
85 \newcommand\spageref[1]{%
86   \ifcsundef{sref@part}{%
87     \sref@pageref{#1}%
88   }{%
89     \sref@pageref{\sref@part @#1}%
90   }%
91 }%
92 ⟨/package⟩
93 ⟨∗ltxml⟩
94 DefConstructor('\spageref{}',
95   "<omdoc:oref href='&withhash()#1'/>");
96 ⟨/ltxml⟩
```

4.3 An API for Package Authors

We find out whether the hyperref package is loaded, since we may want to use
it for cross-references, for which we set up some internal macros that gracefully
degrade if hyperref is not loaded.

\sref@*@ifh

```
97 ⟨∗package⟩
98 \newif\ifhref\hreffalse%
```

³EDNOTE: it is not clear what we want in the LaTeXML implementation of spageref

```
 99 \AtBeginDocument{%
100   \@ifpackageloaded{hyperref}{%
101     \hreftrue%
102   }{%
103     \hreffalse%
104   }%
105 }%
106 \newcommand\sref@href@ifh[2]{%
107   \ifhref%
108     \href{#1}{#2}%
109   \else%
110     #2%
111   \fi%
112 }%
113 \newcommand\sref@hlink@ifh[2]{%
114   \ifhref%
115     \hyperlink{#1}{#2}%
116   \else%
117     #2%
118   \fi%
119 }%
120 \newcommand\sref@target@ifh[2]{%
121   \ifhref%
122     \hypertarget{#1}{#2}%
123   \else%
124     #2%
125   \fi%
126 }%
```

Then we provide some macros for sTeX-specific crossreferencing

\sref@target   The next macro uses this and makes an target from the current `sref@id` declared
by a `id` key.

```
127 \def\sref@target{%
128   \ifx\sref@id\@empty%
129     \relax%
130   \else%
131     \edef\@target{sref@\ifcsundef{sref@part}{}{\sref@part @}\sref@id @target}%
132     \sref@target@ifh\@target{}%
133   \fi%
134 }%
```

The next two macros are used for setting labels, it is mainly used for enabling
forward references, to do this, it is written into ⟨*jobname*⟩.`aux` or ⟨*jobname*⟩.`refs`.

\@sref@def   This macro stores the value of its last argument in a custom macro for reference.

```
135 %\newcommand\@sref@def[3]{\expandafter\gdef\csname sref@#1@#2\endcsname{#3}}
136 \newcommand\@sref@def[3]{\csgdef{sref@#1@#2}{#3}}
```

The next step is to set up a file to which the references are written, this is normally the `.aux` file, but if the `extref` option is set, we have to use an `.ref` file.

```
137 \ifextrefs%
138   \newwrite\refs@file%
139 \else%
140   \def\refs@file{\@auxout}%
141 \fi%
```

`\sref@def` This macro writes an `\@sref@def` command to the current aux file and also executes it.

```
142 \newcommand\sref@def[3]{%
143   %\@sref@def{#1}{#2}{#3}%
144   \protected@write\refs@file{}{\string\@sref@def{#1}{#2}{#3}}%
145 }%
```

`\srefaddidkey` `\srefaddidkey[`⟨*keyval*⟩`]{`⟨*group*⟩`}` extends the metadata keys of the group ⟨*group*⟩ with an `id` key. In the optional key/value pairs in ⟨*keyval*⟩ the `prefix` key can be used to specify a prefix. Note that the `id` key defined by `\srefaddidkey[`⟨*keyval*⟩`]{`⟨*group*⟩`}` not only defines `\sref@id`, which is used for referencing by the `sref` package, but also `\`⟨*group*⟩`@id`, which is used for showing metadata via the `showmeta` option of the `metakeys` package.

```
146 \addmetakey{srefaddidkey}{prefix}
147 \newcommand\srefaddidkey[2][]{%
148   \metasetkeys{srefaddidkey}{#1}%
149   \@metakeys@ext@clear@keys{#2}{sref@id}{}% id cannot have a default
150   \metakeys@ext@clear@keys{#2}{id}{}%
151   \metakeys@ext@showkeys{#2}{id}%
152   \define@key{#2}{id}{%
153     \edef\sref@id{\srefaddidkey@prefix ##1}%
154   %\expandafter\edef\csname #2@id\endcsname{\srefaddidkey@prefix ##1}%
155     \csedef{#2@id}{\srefaddidkey@prefix ##1}%
156   }%
157 }%
158 ⟨/package⟩
```

4.4   Inter-Document Crossreferencing

`\makeextrefs`

```
159 ⟨*package⟩
160 \newcommand\makeextrefs[1]{%
161   \gdef\sref@part{#1}%
162   \makeatletter%
163   \IfFileExists{\jobname.refs}{\input{\jobname.refs}}{}%
164   \immediate\openout\refs@file=\jobname.refs
165   \makeatother%
166 }%
167 ⟨/package⟩
168 ⟨ltxml⟩DefConstructor('\makeextrefs{}','');
```

\sref@label The `\sref@label` macro writes a label definition to the auxfile.

```
169 ⟨∗package⟩
170 \newcommand\sref@label[2]{%
171   \sref@def{\ifcsundef{sref@part}{}{\sref@part @}#2}{page}{\thepage}%
172   \sref@def{\ifcsundef{sref@part}{}{\sref@part @}#2}{label}{#1}%
173 }%
174 ⟨/package⟩
```

\sreflabel The `\sreflabel` macro is a semantic version of `\label`, it combines the categorization given in the first argument with LaTeX's `\@currentlabel`.

```
175 ⟨∗package⟩
176 \newcommand\sreflabel[2]{\sref@label{#1 \@currentlabel}{#2}}
177 ⟨/package⟩
```

\sref@label@id The `\sref@label@id` writes a label definition for the current `\sref@id` if it is defined.

```
178 ⟨∗package⟩
179 \newcommand\sref@label@id[1]{%
180   \ifx\sref@id\@empty%
181     \relax%
182   \else%
183     \sref@label{#1}{\sref@id}%
184   \fi%
185 }%
186 ⟨/package⟩
```

EdN:4   Finally we come to the user visible macro `\sref` which is used for referencing.[4]

\sref@hlink `\sref@hlink[⟨alt⟩]{⟨label⟩}{⟨fallback⟩}{⟨pre⟩}{⟨post⟩}` creates an error message if the target specified by ⟨label⟩ is not defined (but uses ⟨fallback⟩ if provided), and otherwise generates a hyperlinked reference whose link text is ⟨alt⟩ (if the optional argument is given) and the label generated by object specified by ⟨label⟩ otherwise.

```
187 ⟨∗package⟩
188 \newcommand\sref@hlink[5][]{%
189   \edef\@linktext{#1}%
190   \edef\@fallback{#3}%
191   \ifcsundef{sref@#2@label}{%
192     \ifx\@fallback\@empty% warn about undefined links and show a substitute
193       \protect\G@refundefinedtrue\@latex@warning{reference #2 undefined}%
194       \ifx\@linktext\@empty%
195         ?#2?%
196       \else%
197         \@linktext%
198       \fi%
199     \else%
200       \protect\G@refundefinedtrue\@latex@warning{using fallback for undefined reference #2}%
```

---

[4]EDNOTE: The LaTeXML does not take into account the optional argument yet.

```
201        \@fallback%
202      \fi
203    }{%
204      \edef\@link{#4\@nameuse{sref@#2@label}#5}% retrieve the reference label and prepost it
205      \ifx\@linktext\@empty%
206        \sref@hlink@ifh{sref@#2@target}{\@link}%
207      \else%
208        \sref@hlink@ifh{sref@#2@target}{\@linktext}%
209      \fi%
210    }%
211 }%
212 ⟨/package⟩
```

\sref@page@label   This macro styles a page reference.

```
213 ⟨*package⟩
214 \newcommand\sref@page@label[1]{p.~{#1}}
215 ⟨/package⟩
```

\sref@pageref   The next macro creates an error message if the target is not defined, and otherwise
                 generates a page reference.

```
216 ⟨*package⟩
217 \newcommand\sref@pageref[1]{%
218    \ifcsundef{sref@#1@page}{%
219      \protect\G@refundefinedtrue\@latex@warning{reference #1 undefined}\sref@page@label{??}%
220    }{%
221      \sref@hlink@ifh{sref@#1@target}{\sref@page@label{\@nameuse{sref@#1@page}}}%
222    }%
223 }%
224 ⟨/package⟩
```

\sref@href   The next macro creates an error message if the target is not defined, and otherwise
              generates a hyperlinked reference.

```
225 ⟨*package⟩
226 \newcommand\sref@href[3][]{%
227    \edef\@linktext{#1}%
228    \ifcsundef{sref@#2@label}{%
229      \protect\G@refundefinedtrue\@latex@warning{reference #2 undefined}??%
230    }{%
231      \ifcsundef{sref@#3@URI}{%
232        \protect\G@refundefinedtrue\@latex@warning{external refs of type  #3 undefined}??%
233      }{%
234        \edef\@uri{\@nameuse{sref@#3@URI}.pdf\#sref@#2@target}%
235        \edef\@label{\ifx\@linktext\@empty\@nameuse{sref@#2@label}\else\@linktext\fi}%
236        \sref@href@ifh\@uri\@label%
237      }%
238    }%
239 }%
240 ⟨/package⟩
```

\extref The next macros use `\sref@href` with the respective prefix for external referencing if external references are used as indicated by the `extrefs` option; otherwise it

disregards the first required macro and uses internal referencing.[5]

```
241 ⟨*package⟩
242 \ifextrefs%
243   \newcommand\extref[3][]{%
244     \def\theextref{\sref@href[#1]{#2@#3}{#2}}%
245     \csname doextref@#2\endcsname%
246   }%
247 \else%
248   \newcommand\extref[3][]{\sref[#1]{#3}}%
249 \fi%
250 ⟨/package⟩
251 ⟨*ltxml⟩
252 DefConstructor('\extref[]{}{}',
253   "<omdoc:oref href='#2@#3'/>");
254 DefConstructor('\theextref','');
255 ⟨/ltxml⟩
```

\extpageref The next macros use `\sref@pagref` with the respective prefix for external referencing if external references are used as indicated by the `extrefs` option; otherwise

it disregards the first required macro and uses internal referencing.[6]

```
256 ⟨*package⟩
257 \ifextrefs%
258   \newcommand\extpageref[3][]{%
259     \def\theextref{\sref@pageref{#2@#3}}%
260     \csname doextpageref@#2\endcsname%
261   }%
262 \else%
263   \newcommand\extpageref[3][]{\spageref{#3}}%
264 \fi%
265 ⟨/package⟩
266 ⟨*ltxml⟩
267 DefConstructor('\extpageref[]{}{}',
268   "<omdoc:oref href='#2@#3'/>");
269 DefConstructor('\theextref','');
270 ⟨/ltxml⟩
```

\extrefstyle This user macro defines an internal macro that is used for internal styling; for instance `\extrefstyle{foo}{\theextref in bar}` defines the macro `\doextref@foo` which evaluates to ⟨*the reference*⟩ in `bar`. This is used in the `\extref` macro.

```
271 ⟨*package⟩
272 %\newcommand\extrefstyle[2]{\expandafter\gdef\csname doextref@#1\endcsname{#2}}%
273 \newcommand\extrefstyle[2]{\csgdef{doextref@#1}{#2}}%
274 ⟨/package⟩
275 ⟨*ltxml⟩
```

---

[5]EDNOTE: This needs to be implemented on the LaTeXML side.
[6]EDNOTE: This needs to be implemented on the LaTeXML side.

```
276 DefConstructor('\extrefstyle{}{}',"");
277 ⟨/ltxml⟩
```

\extpagerefstyle   This is analogous to \extrefstyle

```
278 ⟨∗package⟩
279 %\newcommand\extpagerefstyle[2]{\expandafter\gdef\csname doextpageref@#1\endcsname{#2}}%
280 \newcommand\extpagerefstyle[2]{\csgdef{doextpageref@#1}{#2}}%
281 ⟨/package⟩
282 ⟨∗ltxml⟩
283 DefConstructor('\extrefstyle{}{}',"");
284 ⟨/ltxml⟩
```

\inputrefs   If the external references file exists, it is read (under the protection of \makeatother) otherwise an error message is displayed.

```
285 ⟨∗package⟩
286 \newcommand\inputrefs[2]{%
287 %\@namedef{sref@#1@URI}{#2}%
288   \csdef{sref@#1@URI}{#2}%
289   \extrefstyle{#1}{\theextref}\extpagerefstyle{#1}{\theextref}%
290   \makeatletter%
291   \IfFileExists{#2.refs}{%
292     \message{Reading external references: #2.refs}\input{#2.refs}%
293   }{%
294     \PackageError{sref}{Reference file #2.refs does not exist}%
295     {Maybe you have to run LaTeX on #2.tex first}}%
296   \makeatother%
297 }%
298 ⟨/package⟩
299 ⟨∗ltxml⟩
300 DefConstructor('\inputrefs{}{}','');
301 ⟨/ltxml⟩
```

4.5   Semantic Versions of Commonly used Referencing Commands

sequation

```
302 ⟨∗package⟩
303 \srefaddidkey{sequation}%
304 \def\sref@sequation@heading{equation}%
305 \newenvironment{sequation}[1][]{%
306   \metasetkeys{sequation}{#1}%
307   \ifx\sref@id\@empty%
308     \begin{displaymath}%
309   \else% no id, using equation*
310     \begin{equation}%
311     \sref@target\sref@label@id{\sref@sequation@heading~(\theequation)}%
312   \fi%
313 }{%
314   \ifx\sref@id\@empty%
```

```
315     \end{displaymath}%
316   \else%
317     \end{equation}%
318   \fi%
319 }%
320 ⟨/package⟩
321 ⟨∗ltxml⟩
322 DefEnvironment('{sequation} OptionalKeyVals',
323       "<ltx:equation "
324           .       "?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')' "
325           .       "refnum='#refnum')(xml:id='#id')>"
326   .   "<ltx:Math mode='display'>"
327   .     "<ltx:XMath>#body</ltx:XMath>"
328   .   "</ltx:Math>"
329   . "</ltx:equation>",
330       mode=>'display_math',
331       properties=> sub { RefStepCounter('equation') },
332       locked=>1);
333 ⟨/ltxml⟩
```

seqnarray

```
334 ⟨∗package⟩
335 \newenvironment{seqnarray}[1][]{%
336   \metasetkeys{sequation}{#1}%
337   \begin{eqnarray*}%
338   \sref@target%
339   \sref@label@id{\sref@sequation@heading~(\theequation)}%
340 }{%
341   \end{eqnarray*}%
342 }%
343 ⟨/package⟩
344 ⟨∗ltxml⟩
345 DefMacro('\seqnarray OptionalKeyVals','\begin{eqnarray*}');
346 DefMacro('\endseqnarray','\end{eqnarray*}');
347 ⟨/ltxml⟩
```

4.6   Semantic Citations

withcite

```
348 ⟨∗package⟩
349 \newcommand\withcite[2]{#2~\cite{#1}}%
350 ⟨/package⟩
351 ⟨∗ltxml⟩
352 DefMacro('\withcite{}{}','\begin{withcitation}{#1}#2\citeit\end{withcitation}');
353 ⟨/ltxml⟩
```

withcitation

```
354 ⟨∗package⟩
355 \newenvironment{withcitation}[1]{\def\citeit{\cite{#1}}}{}%
```

```
356 ⟨/package⟩
357 ⟨∗ltxml⟩
358 DefConstructor('\citeit',"<omdoc:citation/> ",
359 afterConstruct => sub {
360   my ($document,$whatsit) = @_;
361   # LibXML acrobatics, since we can't talk about the xml:id prior to construction's end
362   # (and please do correct me if this is inaccurate)
363   my $node = $document->getNode;
364   my ($citenode) = $document->findnodes('preceding-sibling::omdoc:citation',$node);
365  my ($phrase_parent) = $document->findnodes('ancestor::ltx:text[@xml:id]',$node);
366   return unless (defined $phrase_parent) && (defined $citenode);
367   my $id = $phrase_parent->getAttribute('xml:id');
368   my $refs = $phrase_parent->getAttribute('citeit-refs');
369   $phrase_parent->removeAttribute('citeit-refs');
370   $citenode->setAttribute('for',$id);
371   $citenode->setAttribute('refs',$refs);
372 });#$
373 DefEnvironment('{withcitation}{}',
374   "<ltx:text citeit-refs='#1'>#body</ltx:text>");
375 ⟨/ltxml⟩
```

4.7   Finale

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

```
376 ⟨∗ltxml⟩
377 1;
378 ⟨/ltxml⟩
```

References

[Koh15]   Michael Kohlhase. *metakeys.sty: A generic framework for extensible Metadata in LATEX*. Tech. rep. Comprehensive TEX Archive Network (CTAN), 2015. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/metakeys/metakeys.pdf`.

[Leh10]   Philipp Lehmann. *The biblatex Package*. Tech. rep. CTAN: Comprehensive TEX Archive Network, 2010. URL: `http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/doc/biblatex.pdf`.

[LS99]   Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. World Wide Web Consortium (W3C), 1999. URL: `http://www.w3.org/TR/1999/REC-rdf-syntax`.

[Pat]   Oren Patashnik. *bibTEXing*. URL: `http://www.ctan.org/get/biblio/bibtex/contrib/doc/btxdoc.pdf` (visited on 12/14/2009).

[sTeX]   *KWARC/sTeX*. URL: `https://svn.kwarc.info/repos/stex` (visited on 05/15/2015).