leipzig package documentation*

Natalie Weber
natalie.a.weber@gmail.com

2017/07/02

Abstract

The leipzig package indexes and prints all linguistic gloss abbreviations used in a document. It provides a set of macros for standard glossing abbreviations, with options to redefine these or to create new ones. leipzig interfaces with the glossaries package for added functionality.

Basic Usage

Authors define gloss abbreviations which can be used in a document. (A standard set of abbreviations is pre-defined by leipzig.) These are accessed via 'shortcut' macros that take no argument such as `\Nom{}` or `{\Nom}` (short for nominative), which is typeset as NOM.

The abbreviations are indexed in a glossary which can then be printed in an inline or block format using `\printglossaries` or `\printglosses`. To print an inline glossary, e.g. in a footnote, use the schema in (1).

(1)
```
\documentclass{article}

\usepackage{leipzig}
\makeglossaries

\begin{document}
  Use at least one macro like
  {\Nom}.\footnote{\printglossaries} % or \printglosses
\end{document}
```

*This document corresponds to leipzig v2.1, dated 2017/07/02.

The default output looks like the following. (The occasional space before a comma or period is *not* normal behavior, and is due to some clash with the doc class used to typeset the package documentation.)

Use at least one macro like NOM.[a]

[a] 1 = first person , 2 = second person , 3 = third person, COP = copula, DU = dual, INS = instrumental, NOM = nominative, PL = plural, SG = singular , SG = singular, VB = verbalizer .

To print a block glossary, e.g. in a manuscript, use package option [block] or [mcolblock], as in (2). The option [mcolblock] is like [block] but prints in a `multicol` environment (default number of columns = 2).

(2)
```
\documentclass{article}
\usepackage{multicol}

\usepackage[mcolblock]{leipzig} % or option: [block]
\makeglossaries

\begin{document}
  Use at least one macro like {\Nom}.
  \printglossaries % or \printglosses
\end{document}
```

The default output looks like the following.

Use at least one macro like NOM.

Abbreviations

1	first person	NOM	nominative
2	second person	PL	plural
3	third person	SG	singular
COP	copula	SG	singular
DU	dual	VB	verbalizer
INS	instrumental		

Block glossary names print in either a chapter header (for classes like book where `\chapter` is defined) and in a section header otherwise, but this default behavior can be changed via the [section] package option.

Contents

1 Setup **4**
- 1.1 Installation . . . 4
- 1.2 Document structure with glossaries . . . 5

2 Package options **6**

3 Core functionality **7**
- 3.1 Defining gloss abbreviations . . . 8
- 3.2 Pre-defined abbreviations . . . 9
- 3.3 Font and display . . . 10

4 Abbreviation display with glossaries **11**
- 4.1 Acronym style . . . 11
- 4.2 Hyperlinks to the glossary . . . 13
- 4.3 Explicitly formatting individual abbreviations . . . 14

5 Displaying the glossary **15**
- 5.1 Glossary style . . . 15
- 5.2 Display options for all styles . . . 16
- 5.3 Options to modify pre-defined styles . . . 17
  - 5.3.1 Inline glossaries . . . 17
  - 5.3.2 Block glossaries . . . 18

6 Multiple glossaries **20**

7 FAQ **21**

References **21**

A Pre-defined abbreviations **22**

B Example of multiple glossaries **24**

C The Code **25**
- C.1 Global conditionals and definitions . . . 25
- C.2 Package options . . . 26
- C.3 Global settings used with glossaries . . . 27
- C.4 Formatting abbreviations . . . 30
- C.5 Glossary styles . . . 32
  - C.5.1 leipzigalttree style . . . 33

C.5.2 leipzigmcolalttree style 35
C.5.3 inline style 36
C.6 Creating gloss abbreviations 39
C.7 leipzig.tex 42

1 Setup

1.1 Installation

Download the leipzig package from CTAN and save it somewhere where LaTeX can find it (usually in $TEXMFHOME/tex/latex/leipzig/).

leipzig will automatically load the glossaries package unless you use the [noglossaries] package option. You will need glossaries v3.02 (2012/05/21) or later, because version 3.02 comes bundled with the glossary-inline package, which leipzig relies on for its inline glossary style.

If you do not already have glossaries, then download it and save it somewhere LaTeX can find it. Run `latex glossaries.ins` to generate the style files, if need be. Be sure to run `texhash` or `maketexlsr` in terminal after installing both packages. To test whether glossaries was installed correctly, you can use the minimal example file `minimalgls.tex` that comes bundled in the glossaries package. In terminal or the command shell, run:

```
pdflatex minimalgls.tex
makeglossaries minimalgls
pdflatex minimalgls.tex
```

Notes:
- `makeglossaries` takes a filename with *no* `.tex` extension.

`makeglossaries` is a perl script which comes bundled with glossaries and which does some smart business of determining whether to build the glossaries using `makeindex` or `xindy`. If your system doesn't recognise the command `perl` then it's likely you don't have Perl installed. Although it is possible to run `makeindex` directly instead, this creates more work on your part in setting various parameters, and some options in glossaries may be unavailable. I highly recommend that you download `perl` and use `makeglossaries`. For Windows, I recommend http://strawberryperl.com/.

If you use TeXWorks or the like for text editing, you can add `makeglossaries` as a menu option instead of running the script in terminal.

See http://tex.stackexchange.com/questions/13152/how-to-makeglossaries-with-texworks.

If you have any trouble installing glossaries, refer to the installation instructions in `glossaries-user.pdf`. If you are daunted by the size of the manual, try starting off with the (much shorter!) guide for beginners (`glossariesbegin.pdf`).

1.2 Document structure with glossaries

There are **five** main parts to any document that utilizes glossaries. If you are missing any of these, then your glossaries will not print.

1. **Load the package** If you use hyperref, you should load hyperref *before* leipzig, contrary to the normal practice of loading hyperref last. (It's because leipzig loads glossaries, which must be loaded after hyperref.)

```
\documentclass{article}
\usepackage{hyperref}
\usepackage{leipzig}% leipzig loads glossaries
```

2. **Generate the glossaries** Any new glossaries must be defined before `\makeglossaries`. No glossaries are created without this line!

```
\makeglossaries
```

3. **Define glossary entries** Best practice is to define new terms in the preamble after `\makeglossaries`.

```
\newleipzig{verbz}{vbz}{verbalizer}
```

4. **Use the glossary entries** somewhere in your document.

```
\begin{document}
 Use {\Verbz} or {\Nom}. %{\Nom} pre-defined by leipzig
```

5. **Print the glossaries** `\printglossaries` will print all defined glossaries. `\printglosses` is provided by leipzig and will just print the glossary that holds the gloss abbreviations.

```
\printglossaries % or \printglosses
\end{document}
```

Once you have those five elements, don't forget to run these commands. (The last run of `pdflatex` typesets the table of contents, if there is one.)

```
pdflatex minimalgls.tex
makeglossaries minimalgls
pdflatex minimalgls.tex
pdflatex minimalgls.tex
```

2 Package options

- **[glossaries]** (default) leipzig will load glossaries if it exists (as well as the bundled packages glossary-inline, glossary-tree, glossary-mcols).
- **[noglossaries]** Prevents glossaries from being loaded. This option basically reduces the package to a list of shortcut macros like `{\Acc}` which expand to `\textsc{acc}`, plus a method to create new shortcut macros, plus a switch to control the font.
- **[nostandards]** The set of standard pre-defined glosses from the Leipzig Glossing Rules [1] can be used but will be prevented from printing in the glossary.
- **[inline]** (default) The glossary containing the gloss abbreviations will be typeset in an inline style. (It uses the `inline` style, which leipzig has modified from the `inline` style in glossary-inline.sty.)
- **[block]** The glossary containing the gloss abbreviations will be typeset in a block style. (It uses the `leipzigalttree` (alias: `block`) style, which leipzig has modified from the `alttree` style in glossary-tree.sty.)
- **[mcolblock]** The glossary containing the gloss abbreviations will be typeset in a multicolumn block format. (It uses the `leipzigmcolalttree` (alias: `mcolblock`) style, which leipzig has modified from the `mcolalttree` style in glossary-mcols.sty.)
- **[leipzignohyper]** (default) Abbreviations created with `\newleipzig` will not have hyperlinks to the glossary.
- **[leipzighyper]** Abbreviations created with `\newleipzig` will have hyperlinks to the glossary.
- **[glosses]** By default, gloss abbreviations are put into the main glossary. Use this option to separate out gloss abbreviations into a separate glossary (type=leipzig). An alias option [leipzig] is also provided.

The leipzig package passes all unknown options to glossaries, so you can use any of the glossaries package options as well. Some useful ones are:

- **[nostyles]** Prevents glossaries from loading several pre-defined glossary styles. If you are only using the styles that are defined in leipzig, then you can safely use [nostyles]. If you use this option but still want to use one of glossaries' pre-defined glossary styles, you will have to load the package that contains that style before setting the style, e.g.

```
\usepackage[nostyles]{leipzig}
\usepackage{glossary-long}
\setglossarystyle{long}
\makeglossaries
```

- **[indexonlyfirst]** - If you have many interlinear glossed examples in your paper, compiling may be slow because every instance of a glossary entry must be written to the glossary. This option causes only the first usage to be written to the glossary. This option affects all glossary entries–not just gloss abbreviations.
- **[acronym]** or **[acronyms]** Separates acronyms from the main glossary. The acronyms glossary can be printed with `\printacronyms`.
- **[symbols]** Creates a new 'Symbols' glossary which can be printed using `\printsymbols`.
- **[toc]** Adds glossaries to the table of contents.
- **[section=⟨*value*⟩]** By default, glossaries begin with a chapter header if you are using a documentclass that defines chapters, else with a section header. You can set this explicitly. If you use section without a value, it means section=section. You can also reset the section type in the document with `\setglossarysection`. (Glossaries using the `inline` styles never begin with a section header.)
- **[nonumberlist]** By default, gloss entries will be followed by a number list that links back to the pages the gloss entries were used on. You can suppress this for all glossaries by using this option.

3 Core functionality

The elements discussed in this section work regardless of whether you have loaded glossaries or not.

3.1 Defining gloss abbreviations

The main part of the package provides a method to define new gloss abbreviations and shortcut macros to print the short forms of the abbreviations. These functions work regardless of whether glossaries is loaded or not.

\newleipzig
\renewleipzig

Gloss abbrevations are defined (or re-defined) with `\newleipzig` and `\renewleipzig`, respectively. For best results, gloss abbreviations should be defined in the preamble after `\makeglossaries` and before `\begin{document}`. (The code for `\renewleipzig` is courtesy TEX.SX user egreg.)

(3) `\newleipzig [`⟨*options*⟩`] {`⟨*label*⟩`} {`⟨*short*⟩`} {`⟨*long*⟩`}`

(4) `\renewleipzig [`⟨*options*⟩`] {`⟨*label*⟩`} {`⟨*short*⟩`} {`⟨*long*⟩`}`

When glossaries is loaded, `\newleipzig` defines a new glossary entry using `\newacronym[type=\leipzigtype]`, where `\leipzigtype` expands to the glossary that houses the gloss abbreviations (default is `main`) .

- The optional argument ⟨*options*⟩ is a key=value list which is passed to `\newacronym` after `[type=\leipzigtype]`. It is ignored if glossaries is not loaded. A list of recognized keys is in chapter 4 of the glossaries documentation.
- ⟨*label*⟩ must be a unique label with which to identify the entry. It cannot contain any nonexpandable commands or active characters, nor can it contain numbers. The reason for this restriction is that leipzig uses the label to construct a shortcut macro, so the label must be able to expand to a valid control sequence name.
- ⟨*short*⟩ is the short form of the abbreviation. This needs to be lowercase so that `\textsc{}` will work. (You cannot make capital letters into smallcaps with `\textsc{}`.)
- ⟨*long*⟩ is the long version of the acronym. I recommend also typing this argument in lowercase letters, and using the glossaries package to format the glossary such that the first letter of all long forms are consistently uppercase or lowercase. You may need to specify hyphenation explicitly in the long form using `\-`.

To redefine a macro, use `\renewleipzig` with the ⟨*label*⟩ of the abbreviation you want to redefine. The ⟨*short*⟩ and ⟨*long*⟩ arguments should be the new forms that you want to change to.

\Label `\newleipzig` creates a shortcut macro by capitalizing the first letter of ⟨*label*⟩ to form `\Label`. This format was chosen because it is short and mnemonic, it stands out visually when editing interlinear gloss texts (IGTs) in your .tex file, and uppercase macros are less likely to be defined than lowercase ones. Shortcut macros take no arguments and gobble a following space, so they require braces; you can type either `\Label{}` or `{\Label}`.[1]

Shortcut macros print the short form of the abbreviation. If glossaries is loaded, they essentially expand to either `\acrshort{`⟨*label*⟩`}` or `\acrshort*{`⟨*label*⟩`}`, depending on whether they include a link to the glossary or not. If glossaries is not loaded, the macros print the short form of the abbreviation in `\leipzigfont`, which is initialized to `\textsc`. Here is a simple example:

(5) `\newleipzig{verbz}{vbz}{ver\-ba\-liz\-er}`

a. creates the shortcut: `\Verbz{}` or `{\Verbz}`

b. which expands to: VB

3.2 Pre-defined abbreviations

The leipzig package pre-defines (in leipzig.tex) the gloss abbreviations from the appendix to the Leipzig Glossing Rules [1]. These are pre-defined to save the end-user time, and also to encourage standardization within the field. (However, they can be re-defined using `\renewleipzig` and the same label that they were originally defined with.)

Whenever possible, the label for the pre-defined abbreviations is the same as the short form. Hence, the shortcut macros are a capitalized version of the short form; i.e. `\Cop{}` will typeset COP, `\Ins{}` will typeset INS, etc. Exceptions to this rule are given in Table 1. In these cases the label and short form differ because the shortcut macro that would have otherwise been created was either already defined in LaTeX or would have begun with a number. (Macro names in LaTeX cannot begin with a number.)

The package also defines macros for common person-number combinations, like 1SG and 3PL. Abbreviations for first person begin with 'F' (not

[1] If you want to be able to type `\Label` without braces, you will need to load the xparse package. leipzig does not load xparse because there are some issues with it, and it remains wholly untested with leipzig. See David Carlisle's explanation: http://tex.stackexchange.com/questions/86565/drawbacks-of-xspace.

Table 1: Unexpected macro names

Shortcut	Label	Short	Long
`\Aarg{}`	aarg	A	agent
`\Parg{}`	parg	P	patient
`\Sarg{}`	sarg	S	argument of intransitive verb
`\First{}`	first	1	first person
`\Second{}`	second	2	second person
`\Third{}`	third	3	third person

Table 2: Abbreviations for persons and number

`\Fsg{}`	1SG	`\Ssg{}`	2SG	`\Tsg{}`	3SG
`\Fdu{}`	1DU	`\Sdu{}`	2DU	`\Tdu{}`	3DU
`\Fpl{}`	1PL	`\Spl{}`	2PL	`\Tpl{}`	3PL

'1'), abbreviations for second person begin with 'S' (not '2'), and abbreviations for third person begin with 'T' (not '3'). These abbreviations are shown in Table (2).

The macros that print the person and number abbreviations were defined using `\newcommand`, not `\newleipzig`, as in (6). The reason is that each glossary entry should correspond to one component of a fusional gloss, not the entire fusional gloss itself. That way a glossary will have entries like 1, 2, 3, SG, DU, PL, etc., but *not* 1SG, 1DU, 1PL, 2SG, 2DU, etc.

(6) a. `\newleipzig{first}{1}{first person} % {\First} prints 1`

b. `\newleipzig{sg}{sg}{singular}  % {\Sg} prints` SG

c. `\newcommand{\Fsg}{{\First}{\Sg}}  % {\Fsg} prints` 1SG

This strategy can be used for any fusional gloss. I usually use a command name that begins with a capital letter like the shortcut macros.

3.3 Font and display

`\leipzigfont` The short form of the abbreviation is typeset using `\leipzigfont`, which takes the short form as its only argument. This is initialized to expand to `\textsc{#1}`, but you can always re-define it. For instance, to print all short forms in an uppercase italic font, use:

(7) `\renewcommand{\leipzigfont}[1]{\textit{\MakeUppercase{#1}}}`

This can occur anywhere in your document and it affects the short forms of any following abbreviations, e.g. further instances of `{\Verbz}` from (5) will expand to *VB*.

Note that the name of each glossary entry in the glossary will also be printed using the final redefinition of `\leipzigfont`. You can redefine `\glsnamefont` to control how glossary names are displayed. See section 5 for more information on glossary display.

4 Abbreviation display with glossaries

If glossaries is loaded, there are many more options for display refinement.

4.1 Acronym style

`\newleipzig` is built on `\newacronym` from the glossaries package. The glossaries package allows you to set an acronym display style. This is a global setting, and it affects all acronyms that are defined using `\newacronym` (not only gloss abbreviations).

`\setacronymstyle`

leipzig sets a custom acronym style called `long-lpz-short` that is identical to the `long-short` style defined in glossaries, except that the short form of the abbreviation is printed using `\firstleipzigfont` (on first use) and `\leipzigfont` (on subsequent uses). It is set in the package using:

(8) `\setacronymstyle{long-lpz-short}`

You can use the same command in the preamble to set a different acronym style. However, this style will affect all acronyms, even if they are not gloss abbreviations. See the glossaries user manual for details.

`\gls`

Using `\gls{`⟨*label*⟩`}` instead of `\Label` will display the abbreviation according to the acronym style, which may be useful in-text when discussing particular grammatical glosses. The following example uses the abbreviation defined in (5).

(9) a. `\gls{verbz} prints:` verbalizer (VB) `% first time use`

b. `\gls{verbz} prints:` VB `% subsequent uses`

`\firstleipzigfont`

The command `\firstleipzigfont` controls how the short form is displayed the first time it is displayed using `\gls` or other `\gls`-like commands.

Table 3: Short, long, and full acronym formats

Command	Prints
`\acrshort{verbz}`	VB
`\Acrshort{verbz}`	VB
`\ACRshort{verbz}`	VB
`\acrlong{verbz}`	verbalizer
`\Acrlong{verbz}`	Verbalizer
`\ACRlong{verbz}`	VERBALIZER
`\acrfull{verbz}`	verbalizer (VB)
`\Acrfull{verbz}`	Verbalizer (VB)
`\ACRfull{verbz}`	VERBALIZER (VB)

It is initialized to `\leipzigfont`, but can be changed in the preamble. For example, the command in (10) will make the short form in the first usage be displayed in bold and `\leipzigfont`.

(10)
```
\renewcommand{\firstleipzigfont}[1]{
    \textbf{\leipzigfont{#1}}}
```

a. `\gls{verbz} prints:` verbalizer (**VB**) `% first time use`

b. `\gls{verbz} prints:` VB `% subsequent uses`

`\glsreset(all)` `\glsunset(all)`

`\gls{}` and other `\gls`-like commands access and modify the first use flag, which determines if the abbreviation has been used before or not. To reset the first use flag so that the next use of `\gls` will diplay the abbreviation as if on first use, use `\glsreset{`⟨*label*⟩`}` for an individual abbreviation, or `\glsresetall` to reset all abbreviations. There are also analogous `\glsunset{`⟨*label*⟩`}` and `\glsunsetall` commands, which cause `\gls` to never use the first use display. See chapter 14 of the glossaries user documentation.

Besides `\gls`, there are other acronym-specific macros to print components of the acronym without modifying the first use flag. These can take the same optional arguments as other `\glstext`-like commands. A few examples are shown in Table 3; see Chapter 13 of the glossaries documentation for more options.

Any modifications to `\firstleipzigfont` will affect any of the `\acrfull`-type macros; `\acrshort`-type macros will be unaffected. The shortcut

macros like {\Verbz} are based on \acrshort, so \firstleipzigfont will not affect them.

4.2 Hyperlinks to the glossary

If you load the hyperref or html packages prior to loading the glossaries package, the \gls-like and \glstext-like commands will automatically have hyperlinks to the relevant glossary entry, unless the hyper option has been switched off. There are many ways to affect the hyper option.

Note: if you use hyperref, then leipzig must be loaded *after* hyperref.

\glsdisablehyper
\glsenablehyper

To disable or enable all hyperlinks to glossaries, use \glsdisablehyper or \glsenablehyper. The effect can be localised by placing the commands within a group.

[nohypertypes]

To disable hyperlinks only for certain glossaries, use the package option (from glossaries) [nohypertypes]. This is a key that takes a list of comma-separated glossary names as a value. Make sure you enclose the value in braces if it contains any commas. The values must be fully expanded, so something like [nohypertypes=\leipzigtype] *won't* work. Instead use [nohypertypes=main] if the [glosses] option has not been used, or [nohypertypes=leipzig] if the [glosses] option has been used. Instead of or in addition to the package option [nohypertypes], you can also use \GlsDeclareNoHyperList{⟨*list*⟩}.

[nohyperfirst]

To disable hyperlinks only on the first use, use the package options (from glossaries) [nohyperfirst]. This is a key that take a list of comma-separated glossary names as a value. Make sure you enclose the value in braces if it contains any commas.

\gls*
\gls+

The \gls-like and \glstext-like commands all take a first optional argument that is a comma-separated list of ⟨*key*⟩=⟨*value*⟩ options, including hyper=false and hyper=true. They also have a star-variant (e.g. \gls*) which uses hyper=false, and a plus-variant (e.g. \gls+) which uses hyper=true. These optional arguments override global settings unless the hyperlinks have been suppressed using \glsdisablehyper.

\leipzighypertrue
\leipzighyperfalse

Regardless of the global settings you use for hyperlinks in \gls-like or \glstext-like entries, the shortcut macros like \Label are defined by default to not link to the glossary. Use the package option [leipzighyper] to make shortcut macros link to the glossary. You can also switch hyperlinks on and off for particular glosses by using \leipzighypertrue or \leipzighyperfalse in the preamble *before* you define the abbreviation with \newleipzig.

4.3 Explicitly formatting individual abbreviations

Occasionally a subset of grammatical glosses should be typeset in some font other than smallcaps (or whatever `\leipzigfont` has been (re)defined to). For instance, when number and gender markers are frequent, some authors use non-capitalized shortened forms like 1s instead of 1SG for first singular person. The optional argument to `\(re)newleipzig` can be used to set the short form explicitly with a font. (The repeated entries in the glossary are, again, *not* normal, and is probably due to some clash with the `doc` program used to typeset the documentation.)

(11) `\renewleipzig[short={\glstextup{s}}]{sg}{s}{singular}`

a. `{\Sg} prints:` s

b. `{\Fsg} prints:` 1s

c. Glossary looks like: 1 = first person , 2 = second person , 3 = third person, COP = copula, DU = dual, INS = instrumental, NOM = nominative, PL = plural, S = singular , s = singular, VB = verbalizer .

The definition in (11) sets the short form of the singular abbreviation to a lowercase upright s. All short forms are printed in `\leipzigfont`, which is initialized to `\textsc`. The glossaries-internal `\glstextup` is used to cancel the effect of that `\textsc`. This defaults to `\textulc`, if defined, otherwise `\textup`. This will affect any command which calls `{\Sg}`, such as the definition for first singular person in `{\Fsg}` (Table 2). Setting the short form explicitly will also determine how the abbreviation is displayed in the glossary, because the glossary prints the `name` field of the glossary entry, which defaults to be the same as the `short` field.

The optional argument to `\re(new)leipzig` can also be used to explicitly set the name and description of a gloss abbreviation, which is useful if these should have a different form than the short and long form in the text. For example, say that you are glossing a Bantu language and you want to identify class markers in the IGTs with C1, C2, C3, etc. Furthermore, you want the glossary to display C#, so that it is clear that C is always followed by a number. Then you could set the fields as in (12).

(12)
```
\newleipzig[%
  name={c\#},%
  description={class marker (\# = class number)}
]{class}{c}{class marker}
```

This defines an abbreviation macro `{\Class}`, which prints the short form C. The short (C) and long (class marker) forms are called in-text via commands like `\acrfull`,[2] but the glossary entry will be printed with the name (c#) and description (class marker (# = class number)) fields. An example is shown in (13).

(13)
```
Use \acrfull{class} or {\Class} in
  document.\footnote{\printglossaries} % or \printglosses
```

The default output looks like the following.

Use class marker (C) or C in document.[a]

[a] c# = class marker (# = class number) .

5 Displaying the glossary

`\printglossaries` `\printglosses`

To display the glossaries, put `\printglossaries` (or the leipzig-defined `\printglosses`) in your document where you would like the glossaries to display. `\printglossaries` prints all glossaries in the order they were defined, while `\printglosses` prints only the `\leipzigtype` glossary.

5.1 Glossary style

The style of the `\leipzigtype` glossary can be controlled in two ways.

`[mcolblock]` `[block]` `[inline]`

The simplest is to use one of three leipzig package options: [mcolblock] sets the `leipzigmcolalttree` style, [block] sets the `leipzigalttree` style, and [inline] sets the `inline` style. (The `inline` style is used by default if no options are specified.) Using a package option ensures that the `\leipzigtype` glossary will display using the intended style regardless of whether you use `\printglossaries` or `\printglosses`.

You can also specify the style in an optional argument to `\printglosses` (or any of the `\printglossary` commands). For example, the code in (14) will print just the `\leipzigtype` glossary and will use the block style. If you want to use your own style defined via `\newglossarystyle`, then you *must* use this method.

[2] If you specify the name field, then `\gls` will no longer print the long and short forms. Use `\acrfull` instead.

(14) `\printglosses[style=block] % same as [style=leipzigalttree]`

If you have multiple glossaries (e.g. glossaries other than gloss abbreviations), the style of the other glossaries is controlled in the "normal" fashion for the glossaries package: either use `\setglossarystyle` before `\makeglossaries`, or you can use the package option [style=⟨*style*⟩]. The latter option only works for styles that are pre-loaded by glossaries. See section 6 for more information on defining multiple glossaries.

5.2 Display options for all styles

`\leipzigname` To change the name of the glossary that houses the gloss abbreviations, redefine `\leipzigname`. It defaults to 'Abbreviations', as below.

(15) `\renewcommand{\leipzigname}{Abbreviations}`

The `\leipzigname` is used in `\glossarysection`, which is called just before the glossary itself. (Note that `\glossarysection` is redefined to print nothing in the `inline` style.)

`\glossarypreamble` Information can be added to the start of the glossary (after the glossary name and before the main body of the glossary) by redefining `\glossarypreamble`. For example:

(16)
```
\renewcommand{\glossarypreamble}{%
  Abbreviations are as follows:}
```

`\glossarypostamble` There is an analogous command which prints at the end of each glossary called `\glossarypostamble`.

`\ifleipzigdesccapitalize` By default the description for gloss abbreviations in the glossary is not capitalized. It will print with whatever case you used to define it. If you would like to instead force-capitalize the descriptions, put `\leipzigdesccapitalizetrue` in your preamble. This will only affect glossaries that are displayed with the styles `inline`, `block` (`leipzigalttree`), or `mcolblock` (`leipzigmcolalttree`).

`\ifleipzignonumbers` By default abbreviations in the `\leipzigtype` glossary will not have a list of page numbers with links back to the pages where the abbreviation was used. If you would like this feature, put `\leipzignonumberstrue` in your preamble. This will only affect the `\leipzigtype` glossary.

`\ifglsnogroupskip` By default there is no vertical space between groups of abbreviations, where each group is the set of gloss abbreviations that begin with the

same letter. (Note that this conditional is set to true for any other defined glossary.) To print a vertical space between groups of abbreviations, put `\glsnogroupskiptrue` in your preamble. This will affect all glossaries.

`\ifglsnopostdot` By default there is no final period at the end of descriptions in the `\leipzigtype` glossary. To print final periods, put `\glsnopostdotfalse` in your preamble. This will affect all glossaries.

5.3 Options to modify pre-defined styles

5.3.1 Inline glossaries

leipzig loads the `glossary-inline` package, which comes bundled with glossaries. This package defines an inline glossary style, which leipzig then redefines to look like a style often used in linguistics papers. This style is set by default when you load leipzig, or by using the [inline] package option.

`\glossarysection` Note that `inline` defines `\glossarysection` to print nothing. Therefore, if you set this style for all glossaries by using `\setglossarystyle {inline}`, then no glossaries will have a glossary section header.

```
\renewcommand*{\glossarysection}[2][]{}
```

`\glossaryheader` `\glsgroupheading` `\glsgroupskip` These commands control what to print at the beginning of the the glossary, what to print at the beginning of a group, and how much space to print between groups. Groups divide a glossary into groups, e.g. based on the first letter of the acronym (all 'A's grouped together, all 'B's grouped together, etc.) It is not generally desired for gloss abbreviations unless the list is very long. These three commands are all initialized to do nothing:

```
\renewcommand*{\glossaryheader}{}
\renewcommand*{\glsgroupheading}[1]{}
\renewcommand*{\glsgroupskip}{}
```

`\glsinlineseparator` `\glsinlinesubseparator` Each full glossary entry is separated by a comma and a space, regardless of whether that entry is a a subentry (linked to a parent entry) or not. `\glsinlineseparator` controls the separator between (parent) entries, and `\glsinlinesubseparator` controls the separator between child entries. The defaults are as follows:

```
\renewcommand*{\glsinlineseparator}{,\space}
\renewcommand*{\glsinlinesubseparator}{,\space}
```

\glsinlineparentchildseparator
\glsinlinepostchild

A set of child entries may be linked to a parent entry by setting `[parent=⟨label⟩]` in the optional argument given to `\(re)newleipzig`. The child or children entries will be grouped together and follow the parent entry. For the inline glossary style, the parent and first child are separated by `\glsinlineparentchildseparator` and the last child is followed by `\glsinlinepostchild`. The defaults are:

```
\renewcommand*{\glsinlineparentchildseparator}{:\space}%
\renewcommand*{\glsinlinepostchild}{}%
```

\glspostinline

The entire inline glossary is concluded with `\glspostinline`. This is initialized to print a period and a space. Default:

```
\renewcommand*{\glspostinline}{.\space}
```

\glsinlinedescformat
\glsinlinesubdescformat
\glsinlineemptydescformat

Short and long forms of abbreviations are separated by `\glsinlinedescformat`. The default is to print an equals sign with small spaces on either side, and to encourage linebreaks *after* the equals sign so that new lines do not begin with an equals sign. The short and long forms of child entries are separated by `\glsinlinesubdescformat`, which is defined the same as `\glsinlinedescformat`. Finally, `\glsinlineemptydescformat` controls what happens if there is no description at all. Defaults:

```
\renewcommand*{\glsinlinedescformat}[3]{\,=\,\linebreak[1]##1}
\renewcommand*{\glsinlinesubdescformat}[3]{\,=\,\linebreak[1]##1}
\renewcommand*{\glsinlineemptydescformat}[2]{}
```

\glsinlinenameformat
\glsinlinesubnameformat

The name of entries and subentries prints without extra formatting, which means it will print using `\leipzigfont`. You could change this behavior by redefining `\glsinlinenameformat` and `\glsinlinesubnameformat`.

```
\renewcommand*{\glsinlinenameformat}[2]{\glstarget{##1}{##2}}
\renewcommand*{\glsinlinesubnameformat}[2]{\glstarget{##1}{##2}
```

5.3.2 Block glossaries

leipzig defines the styles `leipzigaltree` and `leipzigmcolaltree` for block glossaries. These names are cumbersome, so the aliases `block` and `mcolblock` are also provided. These two styles are based on `alttree` and `mcolalttree`, from the glossary-tree and glossary-mcols packages, respectively, both of which come bundled with glossaries. They are identical except that `leipzigmcolalttree` sets the glossary in a `multicol` environment. These styles can be easily set for the `\leipzigtype` glossary by

using the [block] or [mcolblock] package options, or by setting the style for individual glossaries, e.g.:

(17) `\printglosses{(mcol)block}`

`\glsmcols` The number of columns for `leipzigmcolalttree` is two by default, but this can be changed by redefining `\glsmcols`. (Note that multicol will not accept a number less than two.)

(18) `\renewcommand*{\glsmcols}{2}`

`\glsfindwidesttoplevelname` `\glssetwidest` The indentation for the description at each level is as wide as the widest abbreviation defined in leipzig, which happens to be COMPL. You can set it explicitly by redefining `\glssetwidest` in the preamble. This command sets the indentation to the width of the text in its argument.

(19) `\glssetwidest{xxxxxx}`

Alternatively, you can calculate the widest abbreviation name by passing a list of glossary names to `\glsfindwidesttoplevelname`. If the optional argument is omitted, all glossaries are assumed.

(20) a. `\glsfindwidesttoplevelname % or`

b. `\glsfindwidesttoplevelname[main,leipzig]`

`\glspostnamespace` The glossary names are followed by `\glspostnamespace`, which is initialized to `\space` but can be redefined.

(21) `\renewcommand*{\glspostnamespace}{\space}`

`\glstreenamefmt` The abbreviation name in the glossary prints without extra formatting, which means it will print using `\leipzigfont`. You could change this behavior by redefining `\glstreenamefmt`.

(22) `\renewcommand*{\glstreenamefmt}[1]{##1}`

6 Multiple glossaries

See Appendix B for an example of a book-like document with multiple glossaries.

By default, gloss abbreviations are put into the `main` glossary. (Actually they are put into the `\glsdefaulttype` glossary, which is usually set to `main`.) The [glosses] option creates a new glossary named `leipzig`, via:

```
\newglossary[lpz]{leipzig}{lzs}{lzo}{\leipzigname}
```

In addition, `\leipzigtype` is redefined to `leipzig`. Since all entries defined via `\newleipzig` have `type=\leipzigtype`, they will be printed in the `leipzig` glossary instead of `main`.

`\newglossary` You can create as many glossaries as you would like before `\makeglossaries`, either by using certain glossaries package options like [acronyms] and [symbols], or by using `\newglossary` directly. (See Chapter 12 of the glossaries user manual.)

`\printglossaries` `\printglossary` `\printglossaries` will print all glossaries in the order they were defined. If you want to print the glossaries in a different order, or if you would like to assign different styles to each, you should use `\printglossary` for each glossary and set the style explicitly using the optional argument.

(23) `\printglossary[`⟨*options*⟩`]`

For example, `\printglossary[type=main,style=list]` would print the `main` glossary using the `list` style defined by glossaries. (If no type is given, then `\printglossary` will print the `main` glossary.)

Recall that `\printglosses` is defined as `\printglossary[type=\leipzigtype]` and is also defined to print with the style specified by leipzig package options. If you want a different style, you can specify that in the optional argument to `\printglosses`:

(24) `\printglosses[style=list]`

You can reset the glossary preamble between each call of `\printglossary` or `\printglosses`.

`[style=`⟨*options*⟩`]` The [style=⟨*options*⟩] option in glossaries is set to define the style for all glossaries *except* the `\leipzigtype` glossary. If you truly want to set one style for all glossaries, you will need to use `\setglossarystyle` before `\makeglossaries`. A linguistics paper might have several different

glossaries of abbreviations, so if you wanted to print all of them using the `leipzigalttree` style, you could do the following:

(25)
```
\usepackage{leipzig}
\setglossarystyle{leipzigalttree}
\makeglossaries
```

Note: if you use this method to set `inline` as the style for every glossary, then no glossary will have a section header.

7 FAQ

Q: Why don't the abbreviations display in smallcaps?

A: Did you define abbreviations using ALL CAPS for the short form? The short form is displayed in `\leipzigfont`, which uses `\textsc`, but `\textsc` cannot make smallcaps out of capital letters: `\textsc{abc}` produces ABC, but `\textsc{ABC}` produces ABC. Solution: change the `\newleipzig` definitions to use lowercase letters in the second argument.

A: Not all font families contain a smallcaps font. For instance, only some version of Times New Roman contain a smallcaps font; the versions on Windows XP and Mac OS X do not. Solution: try changing the smallcaps font, or at least using `\usepackage[T1]{fontenc}` in your preamble.

References

[1] Bickel, Balthasar, Bernard Comrie, and Martin Haspelmath. (2008). "The Leipzig Glossing Rules. Conventions for Interlinear Morpheme by Morpheme Glosses." Revised version of February 2008. Department of Linguistics, Max Plank Institute for Evolutionary Anthropology. Retreived 30 June 2012: http://www.eva.mpg.de/lingua/resources/glossing-rules.php.

A Pre-defined abbreviations

Command	Short	Long
`\First{}`	1	first person
`\Second{}`	2	second person
`\Third{}`	3	third person
`\Abl{}`	ABL	ablative
`\Abs{}`	ABS	absolutive
`\Acc{}`	ACC	accusative
`\Adj{}`	ADJ	adjective
`\Adv{}`	ADV	adverbial
`\Aarg{}`	A	agent
`\Agr{}`	AGR	agreement
`\All{}`	ALL	allative
`\Antip{}`	ANTIP	antipassive
`\Appl{}`	APPL	applicative
`\Art{}`	ART	article
`\Aux{}`	AUX	auxiliary
`\Ben{}`	BEN	benefactive
`\Caus{}`	CAUS	causative
`\Clf{}`	CLF	classifier
`\Com{}`	COM	comitative
`\Comp{}`	COMP	complementizer
`\Compl{}`	COMPL	completive
`\Cond{}`	COND	conditional
`\Cop{}`	COP	copula
`\Cvb{}`	CVB	converb
`\Dat{}`	DAT	dative
`\Decl{}`	DECL	declarative
`\Defn{}`	DEF	definite
`\Dem{}`	DEM	demonstrative
`\Det{}`	DET	determiner
`\Dist{}`	DIST	distal
`\Distr{}`	DISTR	distributive
`\Du{}`	DU	dual
`\Dur{}`	DUR	durative
`\Erg{}`	ERG	ergative
`\Excl{}`	EXCL	exclusive
`\F{}`	F	feminine

`\Foc{}`	FOC	focus
`\Fut{}`	FUT	future
`\Gen{}`	GEN	genitive
`\Imp{}`	IMP	imperative
`\Incl{}`	INCL	inclusive
`\Ind{}`	IND	indicative
`\Indf{}`	INDF	indefinite
`\Inf{}`	INF	infinitive
`\Ins{}`	INS	instrumental
`\Intr{}`	INTR	intransitive
`\Impf{}`	IMPF	imperfective
`\Irr{}`	IRR	irrealis
`\Loc{}`	LOC	locative
`\M{}`	M	masculine
`\N{}`	N	neuter
`\Neg{}`	NEG	negative
`\Nmlz{}`	NMLZ	nominalizer
`\Nom{}`	NOM	nominative
`\Obj{}`	OBJ	object
`\Obl{}`	OBL	oblique
`\Pass{}`	PASS	passive
`\Parg{}`	P	patient
`\Pfv{}`	PFV	perfective
`\Pl{}`	PL	plural
`\Poss{}`	POSS	possessive
`\Pred{}`	PRED	predicative
`\Prf{}`	PRF	perfect
`\Prs{}`	PRS	present
`\Prog{}`	PROG	progressive
`\Proh{}`	PROH	prohibitive
`\Prox{}`	PROX	proximal
`\Pst{}`	PST	past
`\Ptcp{}`	PTCP	participle
`\Purp{}`	PURP	purposive
`\Q{}`	Q	question particle
`\Quot{}`	QUOT	quotative
`\Recp{}`	RECP	reciprocal
`\Refl{}`	REFL	reflexive
`\Rel{}`	REL	relative
`\Res{}`	RES	resultative

\Sbj{}	SBJ	subject
\Subj{}	SUBJ	subjunctive
\Sg{}	SG	singular
\Sarg{}	S	argument of intransitive argument
\Top{}	TOP	topic
\Tr{}	TR	transitive
\Voc{}	VOC	vocative

B Example of multiple glossaries

```
\documentclass{book}%

\usepackage[mcolblock,% style for gloss abbreviations
     symbols,%          creates glossary of symbols
     glosses,%          creates glossary of gloss abbreviations
     acronyms,%         creates glossary of acronyms that
                                % are not symbols or glosses
]{leipzig}

 % Glossaries other than \leipzigtype use this style:
\setglossarystyle{block}

 % Change default name of \leipzigtype:
\renewcommand{\leipzigname}{Grammatical glosses}

\makeglossaries

 % Define some entries.

 % These will go into the acronym glossary.
 \newacronym{acr1}{BPS}{Bare Phrase Structure}
 \newacronym{acr2}{OT}{Optimality Theory}

 % These will go into the main glossary.
 \newglossaryentry{gls1}{name={Richness of the Base},%
      description={This is a description of Richness of the Base}}
 \newglossaryentry{gls2}{name={Emergence of the Unmarked},%
      description={This is a description of the
          Emergence of the Unmarked}}

 % These will go into the symbols glossary.
 \newacronym[type=symbols,%
    symbol=\textiota{},%
```

```
   sort={iota}]{iphrase}{IPh}{intonational phrase}%

\newacronym[type=symbols,%
   symbol=\textphi{},%
   sort={phi}]{pphrase}{PPh}{phonological phrase}%

\newacronym[type=symbols,%
   symbol=\textomega{},%
   sort={omega}]{pword}{PWd}{prosodic word}%

\glsaddall % if you want to see what the output looks like

\begin{document}

% The main glossary will print after its own chapter heading,
%    because we didn't change the [section] option.
% We will print the main glossary in yet a third style:
\printglosses[style=list]

% Now we will print all of the remaining glossaries as
%      individual sections together in one chapter.
\chapter*{Abbreviations}
\setglossarysection{section}

\printglosses % prints using mcolblock style
\printacronyms % prints using block style
\printsymbols % prints using block style

\end{document}
```

C The Code

C.1 Global conditionals and definitions

`\ifleipzig@glossaries` `\ifleipzig@noglossaries` Some booleans to determine whether the glossaries package is loaded or not. The default is to load glossaries.

```
\newif\ifleipzig@glossaries\leipzig@glossariestrue
\newif\ifleipzig@noglossaries\leipzig@noglossariesfalse
```

`\leipzig@sepglossestrue` Boolean to determine whether to typeset glosses in a separate glossary. Default is false, because we assume that the most common use case is to have only one glossary, in which case glosses can go in the main glossary.

```
\newif\ifleipzig@sepglosses\leipzig@sepglossesfalse
```

\ifleipzig@nostandards
\ifleipzigdonotindex

These two booleans prevent the set of standard set of abbreviations defined in the Leipzig Glossing Rules from being indexed or printed in the glossary. Users are still able to use the shortcut macros defined with the package, like `\Acc{}`. Default is to include standards in the glossary. `\leipzigdonotindex` can be used by authors to prevent particular abbreviations from being indexed.

```
\newif\ifleipzig@nostandards\leipzig@nostandardsfalse
\newif\ifleipzigdonotindex\leipzigdonotindexfalse
```

\@leipzig@default@style

`\@leipzig@default@style` holds the style that will be used in the definition for `\printglossary[type=main]`. We default to an inline style because this is the most common use case. This can be reset by the package options [block], [mcolblock], and [inline] (same as default).

```
\newcommand*{\@leipzig@default@style}{inline}
```

\ifleipzighyper

Boolean switch to make abbreviations link to the glossary. Default is false.

```
\newif\ifleipzighyper\leipzighyperfalse
\newif\ifleipzignonumbers\leipzignonumberstrue
```

\leipzigfont

Some user hooks to change how abbreviations are displayed in-text on first and subsequent uses. These default to smallcaps. `\leipzigfont` works regardless of whether glossaries is loaded or not.

```
\newcommand{\leipzigfont}[1]{\textsc{#1}}%
```

C.2 Package options

[glossaries]
[noglossaries]

Option [glossaries] will load the glossaries package, if it exists (default behaviour). The package option [noglossaries] prevents leipzig from loading glossaries; if glossaries was loaded before leipzig then some of that package's functionality will still exist.

```
\DeclareOption{glossaries}{\leipzig@glossariestrue}
\DeclareOption{noglossaries}{%
  \leipzig@glossariesfalse
  \leipzig@noglossariestrue
}%
```

[glosses]
[leipzig]

These options create a new glossary named `leipzig` and redefines `\leipzigtype` to `leipzig`.

```
\DeclareOption{glosses}{\leipzig@sepglossestrue}
\DeclareOption{leipzig}{\leipzig@sepglossestrue}
```

[nostandards] This option prevents the set of standard abbreviations from being indexed in a glossary.

```
\DeclareOption{nostandards}{\leipzig@nostandardstrue}
```

[block] [mcolblock] [inline] leipzig defines three glossary styles which can be used, which can be set via these package options. The `\leipzigtype` glossary will display using this style, regardless of whether `\printglossaries` or `\printglosses` is used.

```
\DeclareOption{block}{%
  \renewcommand*{\@leipzig@default@style}{leipzigalttree}%
}%
\DeclareOption{mcolblock}{%
  \renewcommand*{\@leipzig@default@style}{leipzigmcolalttree}%
}%
\DeclareOption{inline}{%
  \renewcommand*{\@leipzig@default@style}{inline}%
}%
```

[leipzighyper] [leipzignohyper] Two package options to control whether gloss abbreviations are hyperlinked to the glossary. Default is false.

```
\DeclareOption{leipzighyper}{\leipzighypertrue}
\DeclareOption{leipzignohyper}{\leipzighyperfalse}
```

Pass all options to the glossaries package and end.

```
\DeclareOption*{%
    \PassOptionsToPackage{\CurrentOption}{glossaries}%
}%
\ProcessOptions\relax
```

C.3 Global settings used with glossaries

Unless the [noglossaries] option was declared, determine if glossaries was loaded; if not, then load glossaries, glossary-inline, glossary-tree, and glossary-mcols, if they exist. If glossaries cannot be found, throw an error.

```
\ifleipzig@noglossaries\relax
\else
  \ifleipzig@glossaries
    \@ifpackageloaded{glossaries}%
      {\relax}%
      {\IfFileExists{glossaries.sty}{%
        \RequirePackage{glossaries}%
        }%
        {\PackageError{leipzig}%
```

```
      {glossaries.sty not found.}%
      {Install glossaries.sty or use [noglossaries] option.}%
    }%
    \IfFileExists{glossary-tree.sty}{%
      \RequirePackage{glossary-tree}%
    }%
    {\PackageWarning{leipzig}%
      {glossary-tree.sty not found.}%
      {Install glossary-tree.sty to use leipzigalttree style.}%
    }%
    \IfFileExists{glossary-mcols.sty}{%
      \RequirePackage{glossary-mcols}%
    }%
    {\PackageWarning{leipzig}%
      {glossary-mcols.sty not found.}%
      {Install glossary-mcols.sty to use leipzigmcolalttree style.}%
    }%
    \IfFileExists{glossary-inline.sty}{%
      \RequirePackage{glossary-inline}%
    }%
    {\PackageWarning{leipzig}%
      {glossary-inline.sty not found.}%
      {Install glossary-inline.sty to use inline style.}%
    }%
  }%
 \fi
\fi
```

From here on out we use `\@ifpackageloaded{glossaries}`, whichis true either if someone used the [glossaries] option, or if they used [noglossaries] but loaded glossaries before leipzig. Possibly unexpected behavior, but this command avoids nested conditionals, which are problematic.

`\leipzigtype` If glossaries was loaded, define some stub macros and conditionals. All glossary entries created with `\newleipzig` have type=`\leipzigtype`. It is initialized here to `\glsdefaulttype`, so that gloss abbreviations are put into the main glossary. The [glosses] package option redefines it to expand to `leipzig`.

`\leipzigname` The name of the `\leipzigtype` glossary is initialized to 'Abbreviations'. This prints in the section header before the glossary, if there is one.

```
\@ifpackageloaded{glossaries}{%
  \newcommand*{\leipzigtype}{\glsdefaulttype}%
  \newcommand{\leipzigname}{Abbreviations}%
```

\printglosses
\printleipzig

\printglosses will print the \leipzigtype glossary. The macro \printleipzig is provided as an alias.

```
\providecommand*{\printglosses}[1][]{}%
\providecommand*{\printleipzig}[1][]{}%
```

\firstleipzigfont

This controls how the short form is formatted on first use.

```
\newcommand{\firstleipzigfont}[1]{\leipzigfont{#1}}%
```

\ifleipzigdesccapitalize

Switch to capitalize the description of the abbreviation in the glossary. Default is false.

```
\newif\ifleipzigdesccapitalize\leipzigdesccapitalizefalse
```

\leipzig@glossentrydesc

\leipzig@glossentrydesc replaces original \glossentrydesc in the glossary styles leipzigalttree, leipzigmcolalttree, and inline. If the author has used \leipzigdesccapitalizetue, then we redefine \leipzig@glossentrydesc to use \Glossentrydesc. This throws an error if the description is simply \nopostdesc, so we stick with \glossentrydesc in that case.

```
\newcommand*{\leipzig@glossentrydesc}[1]{%
  \ifleipzigdesccapitalize
    \expandafter\ifx#1\nopostdesc
      \glossentrydesc{#1}%
    \else\Glossentrydesc{#1}%
    \fi
  \else
    \glossentrydesc{#1}%
  \fi
}%
```

\glspostnamespace

This is used in the leipzigalttree and leipzigmcolalttree styles. In glossaries the space after a glossary entry name was hard-coded as \space, but authors might want this to be more flexible.

```
\newcommand*{\glspostnamespace}{\space}%
```

End if glossaries was loaded.

```
}{}%
```

If an author needs more than one glossary, the package option [glosses] will create a new glossary called leipzig with name \leipzigname (default: 'Abbreviations') and will switch \leipzigtype to point to there.

```
\@ifpackageloaded{glossaries}{%
  \ifleipzig@sepglosses
```

```
\newglossary[lpz]{leipzig}{lzs}{lzo}{\leipzigname}%
\renewcommand*{\leipzigtype}{leipzig}%
```

Define hook to set the toc title when translator is in use.

```
\newcommand*{\gls@tr@set@leipzig@toctitle}{%
  \translatelet{\glossarytoctitle}{Abbreviations}%
}%
```

Otherwise, gloss abbreviations will sit in the `main` glossary, because we initialized `\leipzigtype` to `\glsdefaulttype` (usually 'main'). We want the main glossary name to always expand to whatever the user defines `\leipzigname` as. This is the most likely expected behavior from the user's end, because if a linguistics document has a glossary at all, it is likely to be a list of gloss abbreviations. Therefore, we `\renewcommand` the `\glossaryname` to return `\leipzigname`.

```
\else
  \renewcommand*{\glossaryname}{\leipzigname}
\fi
```

C.4 Formatting abbreviations

Declare the `\leipzigtype` glossary as an acronym list.

```
\DeclareAcronymList{\leipzigtype}%
```

`long-lpz-short` We create a new acronym style, modeled off of the glossaries code for the `long-sc-short` style. It copies the `long-short` style, but changes the font of the short form to be `\leipzigfont`.[3] The short form will be used both in running text and in the glossary. (This default can be overridden by setting the short forms explicitly via options to `\newglossaryentry` or `\newleipzig`.)

```
\newacronymstyle{long-lpz-short}%
{%
  \GlsUseAcrEntryDispStyle{long-short}%
}%
{%
  \GlsUseAcrStyleDefs{long-short}%
  \renewcommand{\acronymfont}[1]{\leipzigfont{##1}}%
  \renewcommand{\firstacronymfont}[1]{\firstleipzigfont{##1}}%
  \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
```

[3] Unfortunately, all abbreviations will use this style, even if they are not gloss abbreviations. The glossaries-extra provides a way to set a different style for each category of abbrevations.

```
}%
```

Set acronym style. (Must be set after `\DeclareAcronymList`).

```
\setacronymstyle{long-lpz-short}
```

Several redefinitions need to happen at the beginning of the document, to take into account any changes in the preamble.

```
\AtBeginDocument{%
```

If glossaries option [nostyles] has been used, then `\@glossary@default@style` is set to `\relax` and no glossary style is set! This is a problem if the author has at least one glossary besides `\leipzigtype`, so we set the glossary style to `\@leipzig@default@style` as a last resort. `\@leipzig@default@style` is initialized to `inline` but could have been redefined by the user via the package options [block], [mcolblock], and [inline].

```
\if\@glossary@default@style\relax
  \setglossarystyle{\@leipzig@default@style}%
\fi
```

`\printglosses` `\printleipzig` Provide shortcut `\printglosses` (and alias `\printleipzig`) to print abbreviations in the appropriate style. We wait until `\AtBeginDocument` in case the author has changed the style in the preamble.

`\printglosses` basically expands to `\printglossary[type=\leipzigtype]`. If `\@leipzig@default@style` is defined, we also set a style. If the author used `\leipzignonumbersfalse`, then we also include a number list.

```
\ifx\@leipzig@default@style\relax
  \renewcommand*{\printglosses}[1][]{%
    \printglossary[type=\leipzigtype,#1]%
  }%
\else
  \ifleipzignonumbers
    \renewcommand*{\printglosses}[1][]{%
      \printglossary[type=\leipzigtype,%
        style=\@leipzig@default@style,%
        nonumberlist,#1]
    }%
  \else
    \renewcommand*{\printglosses}[1][]{%
      \printglossary[type=\leipzigtype,%
        style=\@leipzig@default@style,#1]
    }%
  \fi
```

```
    \fi
    \let\printleipzig\printglosses
```

We also redefine `\printglossaries` so that when the `\leipzigtype` glossary is printed, it uses `\printglosses` with the correct style specifications instead of `\printglossary`.

```
    \renewcommand*{\printglossaries}{%
```

`\forallglossaries` defines the first argument (`\@@glo@type`) as the name of the current glossary in the loop.

```
      \forallglossaries{\@@glo@type}{%
```

If `\@@glo@type` and `\leipzigtype` expand to the same thing, then we use `\printglosses`. We have to fully expand both macros before comparison.

```
        \edef\tempa{\@@glo@type}%
        \edef\tempb{\leipzigtype}%
        \ifx\tempa\tempb%
          \printglosses%
        \else
          \printglossary[type=\@@glo@type]%
        \fi%
```

End `\forallglossaries`.

```
      }%
```

End `\printglossaries` redefinition.

```
    }%
```

End `\AtBeginDocument`.

```
  }%
```

End `\@ifpackageloaded{glossaries}` true text. If not, `\relax`.

```
  }%
  {\relax}%
```

C.5 Glossary styles

Set some defaults for the `\leipzigtype` glossary. These can be overridden by the author.

```
  \@ifpackageloaded{glossaries}{%
    \forallglossaries[\leipzigtype]{\@@glo@type}{%
      \glsnogroupskiptrue
      \glsnopostdottrue
```

```
  }%
}{\relax}%
```

The leipzig package pre-defines three styles.

C.5.1 leipzigalttree style

```
\@ifpackageloaded{glossary-tree}{%
```

leipzigalttree If the glossary-tree package is loaded, then define a new block glossary style, `leipzigalttree`, modified from the `alttree` style with only minimal changes.

```
\newglossarystyle{leipzigalttree}{%
  \setglossarystyle{alttree}%
    \renewenvironment{theglossary}%
    {\def\@gls@prevlevel{-1}%
    \mbox{}\par
    }%
    {\par}%
```

Removed default bolding from `\glstreenamefmt`. That way, the short form prints with the default formatting imposed by the `long-lpz-short` acronym style. This defaults to `\leipzigfont`, unless the author used the `short` key in the gloss entry definition.

```
\renewcommand*{\glstreenamefmt}[1]{##1}
```

The definitions for `\glossentry` and `\subglossentry` are identical to `alttree`, except that I made the space after the widest name adjustable with `\glspostnamespace`, which defaults to `\space`. I also printed the description using `\leipzig@glossentrydesc`, which defaults to `\glossentrydesc`, but will be redefined to use `\Glossentrydesc` if `\leipzigdesccapitalize` is set to true by the author.

```
\renewcommand{\glossentry}[2]{%
  \ifnum\@gls@prevlevel=0\relax
  \else
     \settowidth{\glstreeindent}{%
       \glstreenamefmt{\@glswidestname\glspostnamespace}%
     }%
  \fi
  \hangindent\glstreeindent
  \parindent\glstreeindent
  \makebox[0pt][r]{%
    \glstreenamebox{\glstreeindent}{%
      \glsentryitem{##1}\glstreenamefmt{%
```

```
          \glstarget{##1}{\glossentryname{##1}}%
        }%
      }%
    }%
    \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
    \leipzig@glossentrydesc{##1}\glspostdescription\space ##2\par
    \def\@gls@prevlevel{0}%
  }%
  \renewcommand{\subglossentry}[3]{%
    \ifnum##1=1\relax
      \glssubentryitem{##2}%
    \fi
    \ifnum\@gls@prevlevel=##1\relax
    \else
      \@ifundefined{@glswidestname\romannumeral##1}{%
        \settowidth{\gls@tmplen}{%
          \glstreenamefmt{\@glswidestname\glspostnamespace}}%
        }{%
        \settowidth{\gls@tmplen}{\glstreenamefmt{%
          \csname @glswidestname\romannumeral##1\endcsname
          \glspostnamespace}}%
        }%
      \ifnum\@gls@prevlevel<##1\relax
         \setlength\glstreeindent\gls@tmplen
         \addtolength\glstreeindent\parindent
         \parindent\glstreeindent
      \else
         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
           \settowidth{\glstreeindent}{\glstreenamefmt{%
              \@glswidestname\glspostnamespace}}}{%
           \settowidth{\glstreeindent}{\glstreenamefmt{%
              \csname @glswidestname\romannumeral\@gls@prevlevel
                 \endcsname\glspostnamespace}}}%
         \addtolength\parindent{-\glstreeindent}%
         \setlength\glstreeindent\parindent
      \fi
    \fi
    \hangindent\glstreeindent
    \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
      \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
    \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%
    \leipzig@glossentrydesc{##2}\glspostdescription\space ##3\par
    \def\@gls@prevlevel{##1}%
  }%
}%
```

The style name `leipzigalttree` is hard to remember. We will create an alias `block`. This is easier to remember and matches the analogous package option that sets this style, but it runs the risk of being so common that some other package (or glossaries itself) overwrites it.

```
\newglossarystyle{block}{%
  \setglossarystyle{leipzigalttree}%
}%
```

End `\@ifpackageloaded{glossaries}`. Else, `\relax`.

```
}%
{\relax}%
```

C.5.2 leipzigmcolalttree style

`leipzigmcolalttree`

The `leipzigmcolalttree` style technically only requires glossary-tree to be loaded. We also require glossary-mcols; this anticipates future updates where leipzig might redefine several more `mcol*` styles.

```
\@ifpackageloaded{glossary-tree}{%
  \@ifpackageloaded{glossary-mcols}{%
    \newglossarystyle{leipzigmcolalttree}{%
      \setglossarystyle{leipzigalttree}%
      \renewenvironment{theglossary}%
      {%
        \begin{multicols}{\glsmcols}%
```

I added `\raggedcolumns`. Otherwise if one column has fewer entries than the others, the space between entries in that column is increased so that the bottoms of both columns are equal, which ruins the tabular-look.

```
        \raggedcolumns
        \def\@gls@prevlevel{-1}%
        \mbox{}\par
```

I hardcoded a negative `\vspace` because the glossary was beginning with a blank line that `\removelastskip` would not remove.

```
        \vspace{-\baselineskip}%
        }%
        {\par
        \end{multicols}}%
    }%
```

The style name `leipzigmcolalttree` is hard to remember. We will create an alias `mcolblock`. This is easier to remember and matches the analogous

package option that sets this style, but it runs the risk of being so common that some other package (or glossaries itself) overwrites it.

```
\newglossarystyle{mcolblock}{%
  \setglossarystyle{leipzigmcolalttree}%
}%
```

End if glossary-mcols and glossary-tree loaded.

```
    }%
    {\relax}%
  }%
  {\relax}%
```

C.5.3 inline style

`inline` If the glossary-inline package is loaded, then redefine the existing `inline` glossary style.

```
\@ifpackageloaded{glossary-inline}{% if glossary-inline loaded
```

We need the functionality of `\renewglossarystyle`, so define it if it is undefined.

```
\providecommand{\renewglossarystyle}[2]{%
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries}{Glossary style `#1' isn't already defined}{}%
  }%
  {%
    \csdef{@glsstyle@#1}{#2}%
  }%
}
```

The first version of glossary-inline is from glossaries v3.02. There were major additions in 3.03 though, so we include a slightly different version of `inline` for systems with glossaries v3.03+ (2012/05/22 or later).

```
\@ifpackagelater{glossaries}{2012/05/22}{%
\renewglossarystyle{inline}{%
  \renewenvironment{theglossary}%
    {%
      \def\gls@inlinesep{}%
      \def\gls@inlinesubsep{}%
      \def\gls@inlinepostchild{}%
    }%
    {\glspostinline}%
 \renewcommand*{\glossaryheader}{}%
 \renewcommand*{\glsgroupheading}[1]{}%
```

The gloss and subgloss format are changed to replace `\glossentrydesc` with `\leipzig@glossentrydesc`.

```
\renewcommand{\glossentry}[2]{%
  \glsinlinedopostchild
  \gls@inlinesep
  \glsentryitem{##1}%
  \glsinlinenameformat{##1}{%
    \glossentryname{##1}%
  }%
 \ifglsdescsuppressed{##1}%
 {%
    \glsinlineemptydescformat
    {%
       \glossentrysymbol{##1}%
    }%
    {%
      ##2%
    }%
  }%
  {%
    \ifglshasdesc{##1}%
    {\glsinlinedescformat{\leipzig@glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2
    {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
  }%
  \ifglshaschildren{##1}%
  {%
     \glsresetsubentrycounter
     \glsinlineparentchildseparator
     \def\gls@inlinesubsep{}%
     \def\gls@inlinepostchild{\glsinlinepostchild}%
  }%
  {}%
  \def\gls@inlinesep{\glsinlineseparator}%
}%
\renewcommand{\subglossentry}[3]{%
  \gls@inlinesubsep%
  \glsinlinesubnameformat{##2}{%
     \glossentryname{##2}}%
  \glssubentryitem{##2}%
  \glsinlinesubdescformat{\leipzig@glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3
  \def\gls@inlinesubsep{\glsinlinesubseparator}%
}%
```

We also change some of the default settings.

```
  \renewcommand*{\glsgroupskip}{}%
  \renewcommand*{\glsinlineseparator}{,\space}% original: ;\space
  \renewcommand*{\glsinlinesubseparator}{,\space}%
  \renewcommand*{\glsinlineparentchildseparator}{:\space}%
  \renewcommand*{\glsinlinepostchild}{}%
  \renewcommand*{\glspostinline}{.\space}% original: \glspostdescription\space
  \renewcommand*{\glsinlinenameformat}[2]{\glstarget{##1}{##2}}%
  \renewcommand*{\glsinlinedescformat}[3]{%
    \,=\,\linebreak[1]##1}% original: \space#1
  \renewcommand*{\glsinlineemptydescformat}[2]{}%
  \renewcommand*{\glsinlinesubnameformat}[2]{%
    \glstarget{##1}{##2}}%original: \glstarget{#1}{}
  \renewcommand*{\glsinlinesubdescformat}[3]{%
    \,=\,\linebreak[1]##1}%original: #1
```

We redefine `\glossarysection` *within* the glossary style definition. This means that any glossary that uses this style will *not* have a section header. Other glossaries will not be affected unless the author uses `\setglossarystyle{inline}` before `\makeglossaries`.

```
    \renewcommand*{\glossarysection}[2][\@gls@title]{}% no section header
    }%
  }%
```

For systems with glossaries v3.02, leipzig added a separator between the long and short forms with `\glsinlineshortlongseparator`. (This functionality was added in glossaries v3.03 with `\glsinline(sub)descformat`.

```
  {%
    \renewglossarystyle{inline}{%
      \renewenvironment{theglossary}%
        {%
          \def\gls@inlinesep{}%
          \def\gls@inlinesubsep{}%
          \def\gls@inlineshortlongsep{%
              \glsinlineshortlongseparator}% added this v1.0
        }%
        {\glspostinline}%
      \renewcommand*{\glossaryheader}{}%
      \renewcommand*{\glsgroupheading}[1]{}%
      \renewcommand*{\glossaryentryfield}[5]{%
        \gls@inlinesep
        \glsentryitem{##1}\glstarget{##1}{##2}%
        \def\glo@desc{##3}%
        \def\@no@post@desc{\nopostdesc}%
        \ifx\glo@desc\@no@post@desc
```

```
      \else
        \ifstrempty{##3}{}{\gls@inlineshortlongsep##3}%
      \fi
      \ifglshaschildren{##1}%
      {%
         \glsresetsubentrycounter
         \glsinlineparentchildseparator
         \def\gls@inlinesubsep{}%
      }%
      {}%
      \def\gls@inlinesep{\glsinlineseparator}%
    }%
    \renewcommand*{\glossarysubentryfield}[6]{%
      \gls@inlinesubsep%
      \glstarget{##2}{}%
      \glssubentryitem{##2}##4%
      \def\gls@inlinesubsep{\glsinlinesubseparator}%
    }%
  \renewcommand*{\glsgroupskip}{}%
  \newcommand*{\glsinlineshortlongseparator}{%
    \,=\,\linebreak[1]}% added this v1.0
  \renewcommand*{\glsinlineseparator}{,\space}% original: ;\space
  \renewcommand*{\glsinlinesubseparator}{,\space}%
  \renewcommand*{\glsinlineparentchildseparator}{:\space}%
  \renewcommand*{\glspostinline}{.\space}% changed from .
 % \renewcommand{\glsnamefont}[1]{\textsc{##1}}% removed v2.0
  \renewcommand*{\glossarysection}[2][\@gls@title]{}% added v1.0
  }%
}%
```

Else, if glossary-inline is not loaded, `\relax`.

```
}{\relax}%
```

C.6 Creating gloss abbreviations

`\newleipzig` Define `\newleipzig` to do the right thing whether glossaries is loaded or not.

```
\@ifpackageloaded{glossaries}{%
```

If glossaries is loaded, `\newleipzig` calls `\newacronym` with type `\leipzigtype`. It then calls `\@newleipzig`, which creates a shortcut macro.

```
  \newcommand{\newleipzig}[4][]{%
    \bgroup
    {\newacronym[type=\leipzigtype,#1]{#2}{#3}{#4}}%
    \@newleipzig#2\@nil%
```

```
}%
```

`\newleipzig` passes the glossary label (parameter #2) to `\@newleipzig`, which actually reads in the first token of the label separately than the remainder of the label.

```
\def\@newleipzig#1#2\@nil{%
```

A globally-defined macro is created from the label after capitalizing the first token. If the `\leipzigdonotindex` boolean is toggled true, then this macro is defined to print the short form of the abbreviation in `\leipzigfont`, which is initialized to smallcaps.

```
\ifleipzigdonotindex
  \uppercase{\expandafter\gdef\csname #2}#2\endcsname{%
    \leipzigfont{\glsentryshort{#1#2}}%
  }%
```

If `\leipzigdonotindex` is false, then the macro is defined to call `\acrshort` or `\acrshort*`, depending on whether the abbreviation should include a hyperlink to the glossary or not.

Note that `\glsentryshort` and `\acrshort` both display the short form of the acronym. Neither changes the first use flag. The only differences are that `\acrshort` indexes the acronym for use in the glossary and is formatted according to the short form of the acronym style `long-lpz-short`, but `\glsentryshort` does not index the acronym nor format the text.

```
\else
  \ifleipzighyper
    \uppercase{\expandafter\gdef\csname #1}#2\endcsname{%
    \acrshort{#1#2}%
  }%
  \else
    \uppercase{\expandafter\gdef\csname #1}#2\endcsname{%
      \acrshort*{#1#2}%
    }%
  \fi
\fi
\egroup
```

End `\def\@newleipzig`.

```
}%
```

`\renewleipzig` This macro is provided to allow authors to change previously-defined gloss abbreviations, such as the pre-defined abbreviations that come bundled with

leipzig. This command is courtesy egreg, from http://tex.stackexchange.com/questions/277292/redefining-existing-abbreviations-for-leipzig-sty.

```
\newcommand{\renewleipzig}[2][]{%
  \if@leipzig@defined{#2}
    {%
     \renew@leipzig{#1}{#2}%
    }%
    {%
     \PackageError{leipzig}
       {Abbreviation `#2' undefined}
       {No `#2` abbreviation is defined, use \string\newleipzig}%
     \@gobbletwo
    }%
}
\def\if@leipzig@defined#1{%
  \uppercase\expandafter{\expandafter\ifcsname\@car#1\@nil}\@cdr#1\@nil\endcsname
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
}
\def\renew@leipzig#1#2{%
  \ifcsname glo@\glsdetoklabel{#2}@name\endcsname
    \csundef{glo@\glsdetoklabel{#2}@name}
  \fi
  \if\relax\detokenize{#2}\relax
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi
  {\newleipzig{#2}}{\newleipzig[#1]{#2}}%
}%
```

If glossaries not loaded, then the code for `\newleipzig` and `\renewleipzig` is much shorter.

```
}{%
  \newcommand{\newleipzig}[4][]{\@newleipzig(#3)#2\@nil}%
  \newcommand{\renewleipzig}[4][]{%
    \if@leipzig@defined{#2}
      {%
       \@newleipzig(#3)#2\@nil%
      }%
      {%
       \PackageError{leipzig}
```

```
        {Abbreviation ‘#2’ undefined}
        {No ‘#2‘ abbreviation is defined, use \string\newleipzig}%
      \@gobbletwo
     }%
  }
  \def\@newleipzig(#1)#2#3\@nil{%
    \uppercase{\expandafter\gdef\csname #2}#3\endcsname{\leipzigfont{#1}}
  }%
}
```

Load the list of standard abbreviations.

```
\@ifpackageloaded{glossaries}{%
  \loadglsentries{leipzig.tex}%
  }%
  {\input{leipzig.tex}}%
```

C.7 leipzig.tex

The following is the list of pre-defined standards which are stripped and written to leipzig.tex.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  This is a simple list of newcommands which create shortcuts for    %%
%%  standard linguistic glosses (see the Leipzig Glossing rules,       %%
%%  http://www.eva.mpg.de/lingua/resources/glossing-rules.php         %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Format of examples: %
%
%  All leipzig gloss macros are of the form {\Label}
%
% where Label is a capitalized version of the gloss label, which is the first
% argument passed to \newleipzig. In most cases, the label has been chosen
% to match the abbreviations suggested in the Leipzig glossing rules.
%
% That is, \Acc{} and {\Acc} will print \textsc{acc}, etc.
%
% In a few cases, the label does not match the abbreviation, because the
% macro that would have been created is already defined in LaTeX.
```

If the [nostandards] option was used, then do not index the following.

```
\makeatletter\ifleipzig@nostandards\leipzigdonotindextrue\fi\makeatother

\newleipzig{abl}{abl}{ab\-la\-tive} %ablative
```

```
\newleipzig{abs}{abs}{ab\-so\-lu\-tive} %absolutive
\newleipzig{acc}{acc}{ac\-cusa\-tive} %accusative
\newleipzig{adj}{adj}{ad\-jec\-tive} %adjective
\newleipzig{adv}{adv}{ad\-ver\-bial} %adverb(ial)
\newleipzig{aarg}{a}{agent} %agent-like argument of
%canonical transitive verb
\newleipzig{agr}{agr}{agreement} %agreement
\newleipzig{all}{all}{al\-la\-tive} %allative
\newleipzig{antip}{antip}{anti\-pas\-sive} %antipassive
\newleipzig{appl}{appl}{ap\-plica\-tive} %applicative
\newleipzig{art}{art}{article} %article
\newleipzig{aux}{aux}{aux\-il\-iary} %auxiliary
\newleipzig{ben}{ben}{bene\-fac\-tive} %benefactive
\newleipzig{caus}{caus}{causative} %causative
\newleipzig{clf}{clf}{clas\-si\-fi\-er} %classifier
\newleipzig{com}{com}{comi\-ta\-tive} %comitative
\newleipzig{comp}{comp}{com\-ple\-men\-ti\-zer} %complementizer
\newleipzig{compl}{compl}{com\-ple\-tive} %completive
\newleipzig{cond}{cond}{con\-di\-tion\-al} %conditional
\newleipzig{cop}{cop}{cop\-u\-la} %copula
\newleipzig{cvb}{cvb}{con\-verb} %converb
\newleipzig{dat}{dat}{da\-tive} %dative
\newleipzig{decl}{decl}{declarative}     %declarative
\newleipzig{def}{def}{definite}          %definite
\newleipzig{dem}{dem}{demonstrative}     %demonstrative
\newleipzig{det}{det}{determiner}        %determiner
\newleipzig{dist}{dist}{dis\-tal}          %distal
\newleipzig{distr}{distr}{dis\-tri\-bu\-tive}  %distributive
\newleipzig{du}{du}{dual}                %dual
\newleipzig{dur}{dur}{dur\-ative}          %durative
\newleipzig{erg}{erg}{erg\-ative}          %ergative
\newleipzig{excl}{excl}{ex\-clu\-sive}       %exclusive
\newleipzig{f}{f}{feminine}              %feminine
\newleipzig{foc}{foc}{focus}             %focus
\newleipzig{fut}{fut}{future}            %future
\newleipzig{gen}{gen}{gen\-i\-tive}          %genitive
\newleipzig{imp}{imp}{imperative}        %imperative
\newleipzig{incl}{incl}{inclusive}       %inclusive
\newleipzig{ind}{ind}{indicative}        %indicative
\newleipzig{indf}{indf}{indefinite}      %indefinite
\newleipzig{inf}{inf}{in\-fini\-tive}        %infinitive
\newleipzig{ins}{ins}{instrumental}      %instrumental
\newleipzig{intr}{intr}{in\-tran\-si\-tive}    %intransitive
\newleipzig{ipfv}{ipfv}{im\-per\-fec\-tive}    %imperfective
\newleipzig{irr}{irr}{ir\-real\-is}          %irrealis
```

```
\newleipzig{loc}{loc}{loc\-ative}          %locative
\newleipzig{m}{m}{masculine}             %masculine
\newleipzig{n}{n}{neuter}                %neuter
\newleipzig{neg}{neg}{negative}          %negation, negative
\newleipzig{nmlz}{nmlz}{nom\-i\-nal\-iz\-er}    %nominalizer/nominalization
\newleipzig{nom}{nom}{nom\-in\-ative}        %nominative
\newleipzig{obj}{obj}{object}            %object
\newleipzig{obl}{obl}{ob\-lique}           %oblique
\newleipzig{pass}{pass}{passive}        %passive
\newleipzig{parg}{p}{patient}            %patient
\newleipzig{pfv}{pfv}{per\-fec\-tive}        %perfective
\newleipzig{pl}{pl}{plural}              %plural
\newleipzig{poss}{poss}{possessive}      %possessive
\newleipzig{pred}{pred}{pred\-i\-ca\-tive}     %predicative
\newleipzig{prf}{prf}{perfect}           %perfect
\newleipzig{prs}{prs}{present}           %present
\newleipzig{prog}{prog}{progressive}     %progressive
\newleipzig{proh}{proh}{prohibitive}     %prohibitive
\newleipzig{prox}{prox}{prox\-i\-mal}        %proximal/proximate
\newleipzig{pst}{pst}{past}              %past
\newleipzig{ptcp}{ptcp}{participle}      %participle
\newleipzig{purp}{purp}{pur\-po\-sive}       %purposive
\newleipzig{q}{q}{question particle}     %question particle/marker
\newleipzig{quot}{quot}{quot\-ative}       %quotative
\newleipzig{recp}{recp}{recip\-ro\-cal}      %reciprocal
\newleipzig{refl}{refl}{reflexive}       %reflexive
\newleipzig{rel}{rel}{relative}          %relative
\newleipzig{res}{res}{re\-sul\-ta\-tive}       %resultative
\newleipzig{sbj}{sbj}{subject}           %subject
\newleipzig{subj}{subj}{sub\-junc\-tive}     %subjunctive
\newleipzig{sg}{sg}{singular}            %singular
\newleipzig{sarg}{s}{argument of intransitive verb}
                                        %single argument of intransitive verb
\newleipzig{top}{top}{topic}             %topic
\newleipzig{tr}{tr}{tran\-si\-tive}          %transitive
\newleipzig{voc}{voc}{voc\-ative}          %vocative

%%  Define short versions of person + number:
\newleipzig{first}{1}{first person}%
\newleipzig{second}{2}{second person}%
\newleipzig{third}{3}{third person}%

\newcommand{\Fsg}{{\First}{\Sg}}%
\newcommand{\Fdu}{{\First}{\Du}}%
\newcommand{\Fpl}{{\First}{\Pl}}%
```

```
\newcommand{\Ssg}{{\Second}{\Sg}}%
\newcommand{\Sdu}{{\Second}{\Du}}%
\newcommand{\Spl}{{\Second}{\Pl}}%
\newcommand{\Tsg}{{\Third}{\Sg}}%
\newcommand{\Tdu}{{\Third}{\Du}}%
\newcommand{\Tpl}{{\Third}{\Pl}}%

\makeatletter
```

If glossaries is loaded, calculate the widest name based on this set of predefined abbreviations. It is probably vaguely accurate; if the author needs to, they can always redefine this in the preamble.

```
\@ifpackageloaded{glossaries}{% if glossary-tree loaded
    \glsfindwidesttoplevelname[\leipzigtype]%
}{\relax}%
```

Turn indexing back on if it was off.

```
\leipzigdonotindexfalse
\makeatother
```