GlossTEX 0.4

Volkan Yavuz*

1997/12/13

Abstract

GlossTEX is a tool for the preparation of glossaries, lists of acronyms or sorted lists in general. It greatly simplifies this task. One or more glossary-definition files serve as databases which contain descriptions of terms. These terms are identified through labels. Based upon labels set into the TEX-source, GlossTEX determines which entries have to appear in the typeset list. GlossTEX uses MakeIndex for the sorting of the lists. References to the place where a term appears in the text can be set in the list. A term consists of a label which is used to identify it, an optional item describing the typeset output, an optional long-form and the actual text representing it. There are many ways to access each of these fields within the document. It is also possible to generate cross-references to another term. GlossTEX is well customizable in respect to the produced output.

Contents

*e-mail: yavuzv@rumms.uni-mannheim.de

List of Tables

List of Acronyms

ASCII . . . . . American Standard Code for Information Interchange A character encoding. See also EBCDIC.

EBCDIC . . . Extended Binary Coded Decimals Interchange Code A character encoding exclusively used on mainframes. See also ASCII.

1 Introduction

1.1 Purpose

GlossTEX is a tool for the automatic preparation of glossaries, lists of acronyms, nomenclature and sorted lists in general. Based upon the labels set into the TEX-source, GlossTEX determines which entries from a glossary-definition file have to be processed to generate the list. GlossTEX then creates an intermediate file that has to be processed by MakeIndex for sorting. The output of MakeIndex is then included into the TEX-source for typesetting. With each term associated is an item representing the typeset output, an optional long-form if it's an acronym and an optional descriptive text. These elements can all be accessed within the document in many ways.

1.2 History

I created GlossTEX because there were no tools for the preparation of glossaries that fit my needs. GlossTEX is mainly a combination of the features of the packages acronym, nomencl and GloTEX. The commands starting with `\ac` are taken more or less directly from acronym and the way GlossTEX handles page references is almost identical to the way nomencl does. The use of glossary databases is inspired by GloTEX.

1.3 Legalise

GlossTEX is provided "as is" and comes with absolutely no warranty. It is covered by the GNU General Public License (see the file `COPYING` that comes with this package).

2 Building and Installation

First, you need to build glosstex from its C sources. Note that you need an ANSI C compiler. If you have MAKE and a decent OS (speak UNIX), typing `make` from the shell should build the binary (have a look at the various makefiles, namely `Makefile` (which is my rather complex development version that may not run on your system setup), `Makefile.unx` (which is the most generic one) and `Makefile.os2`). Also have a look at the start of the makefiles where you will find some variables to be customized. The most important ones may be `CC` and `SHELL`. You may also want to have a look at `config.h`.

If you don't have MAKE get it right now or just issue these commands:

```
cc -c database.c -o database.o
cc -c error.c -o error.o
cc -c labels.c -o labels.o
cc -c list.c -o list.o
cc -c main.c -o main.o
cc -c version.c -o version.o
cc database.o error.o labels.o list.o main.o version.o -o glosstex
```

Now just move the resulting binary to a directory where your other binaries live. After that you need to unpack `glosstex.dtx`. Do this by invoking

```
latex glosstex.ins
```

which will produce some more files. You will have to move them to their proper places according to your particular TEX-installation. The instructions that will appear on the screen should give you a good start.

2.1 A Goodie for teTEX User

If you are using THOMAS ESSER's fantastic teTEX system, GlossTEX has some good news for you. GlossTEX comes with a kpathsea shell-wrapper which supports path searching. For this to work, rename `glosstex` to `glosstex.bin` and `glosstex.sh` to `glosstex`. Put them somewhere convenient. Now add a line of the form

```
GDFINPUTS = $KPSE_DOT:$TEXMFS/glosstex//
```

in you `texmf.cnf` file. That's it!

If you get problems, the following notes may help you. First check whether

```
kpsetool -v -n glosstex '$GDFINPUTS'
```

and

```
kpsetool -v -n tex '$GDFINPUTS'
```

do what you want, i.e. their output should be equal. These won't find your teTEX `texmf` tree (where your `texmf.cnf` file lives) if `glosstex` is not in something like `teTeX/bin` or `teTeX/bin/i386-linux`. In these cases, replace

```
TETEXDIR = $SELFAUTOPARENT
```

in `texmf.cnf` with something like

```
TETEXDIR = /usr/local/lib/teTeX.
```

This makes sure that the proper `texmf` tree is found, even if `glosstex` is not in the teTEX `texmf` tree but in your local `texmf` tree.

You also need symbolic links from e. g. `/usr/local/lib/localTeX/texmf.cnf` to your teTEX `texmf.cnf`. This is because kpathsea looks for `texmf.cnf` in the tree where the binary was found.

These instructions should apply to teTEX version 0.4 or newer.

3 Usage

The LATEX-macros needed by GlossTEX have to be included into the source using

```
\usepackage[[⟨options⟩]]{glosstex}
```

Valid ⟨*options*⟩ are listed in table 1.

Option	Discussed in section
`refpage`, *`norefpage`*	3.3
`itemfirst`, `longfirst`	3.4.1
`text`, `footnote`	3.4.2
`roundparen`, `squareparen`	3.4.3

Table 1: Package options. Defaults are set in *italics*.

`\glosstex` Whenever you want a term to appear in the glossary, you insert

```
\glosstex[(⟨list⟩)][[⟨pageref-mode⟩]]{⟨label⟩}
```

into the text. ⟨*label*⟩ references the entry and the optional arguments ⟨*pageref-mode*⟩ and ⟨*list*⟩ determine the mode for page references[1] and the list the referenced term should appear in.

But you can create n independent lists and make each ⟨*label*⟩ appear in any ⟨*list*⟩, each appearance independent of the other ones. To facilitate the creation of multiple ⟨*list*⟩s and especially make it easy to achieve the most likely usage, there are actually two sets of commands. One set starting with `\gl` defaults to ⟨*list*⟩ `glo` (which stands for glossary) and ⟨*pageref-mode*⟩ `p`, the other set starting with `\ac` defaults to ⟨*list*⟩ `acr` (list of acronyms) and ⟨*pageref-mode*⟩ `n`. The following is the equivalent to `\glosstex` for acronyms:

`\acronym`

```
\acronym[(⟨list⟩)][[⟨pageref-mode⟩]]{⟨label⟩}
```

These two commands can also be called as `\glosstex{*}` and `\acronym{*}` to include all terms found in the `.gdf`-files into the corresponding ⟨*list*⟩. This

[1] Page references are described in section 3.3. ⟨*pageref-mode*⟩ works in conjunction with the optional arguments `refpage` and `norefpage` to the package and defaults to `p`. ⟨*list*⟩ defaults to `glo`.

is similar to the command `\nocite{*}` in BibTeX and is primarily meant for debugging purposes. But do *not* use something like `\gls{*}` (see below).

There is another set of commands that produce both an entry in the list (this is optional) and typeset output.

`\gls`
`\gls*`

`\gls[*][(`⟨*list*⟩`)][[`⟨*pageref-mode*⟩`]]{`⟨*label*⟩`}`

`\gls` typesets ⟨*item*⟩ and produces a list entry (⟨*list*⟩ defaults to `glo`). The starred version `\gls*` just produces typeset output without a list-entry.[2]

`\ac`
`\ac*`
`\acs`
`\acs*`
`\acl`
`\acl*`
`\acf`
`\acf*`

`\ac[*][,`⟨*lparen*⟩`,`⟨*rparen*⟩`,][<`⟨*form*⟩`>][(`⟨*list*⟩`)][[`⟨*pageref-mode*⟩`]]{`⟨*label*⟩`}`
`\acs[*][(`⟨*list*⟩`)][[`⟨*pageref-mode*⟩`]]{`⟨*label*⟩`}`
`\acl[*][(`⟨*list*⟩`)][[`⟨*pageref-mode*⟩`]]{`⟨*label*⟩`}`
`\acf[*][,`⟨*lparen*⟩`,`⟨*rparen*⟩`,][<`⟨*form*⟩`>][(`⟨*list*⟩`)][[`⟨*pageref-mode*⟩`]]{`⟨*label*⟩`}`

These all produce typeset output. `\acs` typesets ⟨*item*⟩, `\acl` typesets ⟨*long-form*⟩ and `\acf` typesets ⟨*full*⟩ (see section 3.4). `\ac` works like `\acf` at it's first invocation and like `\acs` on all subsequent ones. Using `\ac` you can make sure that an acronym is always spelled out at least once in your document while safely using the short form whenever possible. Note that the use of `\acf` has no effect on subsequent invocations of `\ac`. The starred versions `\ac*`, `\acs*`, `\acl*` and `\acf*` just produce typeset output. Table 2 gives an overview of all this.

defaults	list	list and text	text	output
`(glo)[p]`	`\glosstex`	`\gls`	`\gls*`	⟨*item*⟩
`(acr)[n]`	`\acronym`	`\ac`	`\ac*`	⟨*item*⟩ or ⟨*full*⟩
`(acr)[n]`		`\acs`	`\acs*`	⟨*item*⟩
`(acr)[n]`		`\acl`	`\acl*`	⟨*long-form*⟩
`(acr)[n]`		`\acf`	`\acf*`	⟨*full*⟩

Table 2: Overview of the different sets of commands.

In the most simple case, ⟨*full*⟩ will look like "⟨*item*⟩ `(`⟨*long-form*⟩`)`". You can switch the order of ⟨*item*⟩ and ⟨*long-form*⟩ by using the options `itemfirst` and `longfirst`. If you rather like either in a footnote, try the option `footnote` instead of the default `text`. If you don't like the round parentheses, just use the optional arguments ⟨*lparen*⟩, ⟨*rparen*⟩ which are also described in section 3.4.

3.1 The Glossary Definition File

A glossary-definition file (suffix `.gdf`) is needed which serves as a database for GlossTeX, holding the actual descriptions of all terms. You can have m `.gdf`-files that contain the definitions to the ⟨*label*⟩s you reference in your documents. Entries have the form

`@entry{`⟨*label*⟩`[,`⟨*item*⟩`[,`⟨*long-form*⟩`]]}` [⟨*text*⟩]

[2] To be honest, a list-entry is produced, but it never appears in the output. The starred version implicitly produces entries with ⟨*list-mode*⟩ `n` (never appear in typeset ⟨*list*⟩) while the unstarred versions set ⟨*list-mode*⟩ to `a` (always appear). There is no other way of specifying the ⟨*list-mode*⟩ of an ⟨*label*⟩.

where ⟨*label*⟩ is used to identify the entry and ⟨*text*⟩ may contain any amount of TEX-source, being the actual definition of the item. You should know that ⟨*label*⟩ is used to construct TEX-macros, so it shouldn't contain funny characters or you will most likely get funny errors. The optional argument ⟨*item*⟩ describes the appearance of the item in the produced list. If omitted, it defaults to ⟨*label*⟩. It can be used when some special form of typesetting is wanted. ⟨*item*⟩ can contain any TEX-construct, as long as each "`{`" has a corresponding "`}`" or GlossTEX will get confused. The same applies to ⟨*long-form*⟩, except that it defaults into an empty string if not specified.

Following is the `.gdf`-file used for this documentation. Note the use of "`~`" as the quote-character. Also note that all lines until the first line starting with `@entry{` are ignored. Additionally, all lines starting with "`%`" are ignored, too. Thus they can serve as comments.

```
⟨*gdf⟩
% -*- latex -*-

This is a database file for GlossTeX.

@entry{ist-file, \texttt{.ist}-file} Style file for \MakeIndex{},
describing the input and output format of read and written files.

@entry{gdf-file, \texttt{.gdf}-file} This file is the database file
containing definitions for Gloss\TeX{}.

@entry{gxs-file, \texttt{.gxs}-file} Intermediate file produced by
\GlossTeX{} to be processed by \MakeIndex{}.

@entry{glx-file, \texttt{.glx}-file} This file contains the sorted
lists, ready to be read by \LaTeX{}.

@entry{gxg-file, \texttt{.gxg}-file} This is the log-file produced by
the \GlossTeX{}-run. See also \glxref{glg-file}.

@entry{glg-file, \texttt{.glg}-file} This is the log-file produced by
the \MakeIndex{}-run. See also \glxref{gxg-file}.

@entry{ASCII, ASCII, American Standard Code for Information Interchange}
A character encoding. See also \glxref{EBCDIC}.

@entry{EBCDIC, EBCDIC, Extended Binary Coded Decimals Interchange Code}
A character encoding exclusively used on mainframes. See also
\glxref{ASCII}.
⟨/gdf⟩
```

3.2 Invocation

After the first run of LATEX, the `.aux`-file contains all necessary information for the preparation of the glossary. GlossTEX is then invoked to read one or more `.gdf`-files and outputs all definitions that are referenced in the `.aux`-file. The output of GlossTEX is then processed by MAKEINDEX for sorting.

GlossTEX is invoked in a UNIX-like environment using the following command

`glosstex` ⟨*aux-file*⟩ ⟨*gdf-file*⟩ [⟨*gdf-file*⟩[...]] [`-v`[`0...5`]]

This produces 2 files as output, one `.gxs`-file to be input into MakeIndex and a log-file with extension `.gxg` which contains more detailed information. The `-v` option selects how verbose GlossTEX should be when writing the log-file. `-v` is equal to `-v4` and `-v2` is the default. `-v0` makes GlossTEX shut his mouth and only report errors and `-v5` makes GlossTEX really talkative.

MakeIndex has to be invoked in this way

`makeindex` ⟨*gxs-file*⟩ `-o` ⟨*glx-file*⟩ `-s` glosstex.ist [`-t` ⟨*glg-file*⟩]

The commands

```
glosstex thesis thesis.gdf master.gdf
makeindex thesis.gxs -o thesis.glx -s glosstex.ist
```

`\printglosstex` produce the final `.glx`-file which is then included by

`\printglosstex`[(⟨*list*⟩)][[⟨*pageref-mode*⟩]]

during the next LaTeX-run. The argument ⟨*pageref-mode*⟩ supersedes the one given to the entries individually for each ⟨*list*⟩ and defaults to `p`. You can turn on page references unconditionally for each ⟨*list*⟩ individually by using ⟨*pageref-mode*⟩ `a` and turn it off by using ⟨*pageref-mode*⟩ `n` as argument to `\printglosstex`. See also table 3 for an overview of these options.

Note that, no matter of how many ⟨*list*⟩s you produce, there is always exactly *one* `.glx`-file which contains the entries of *all* produced lists.

Depending on whether you use ⟨*item*⟩ or ⟨*long-form*⟩ in your text or you have cross-references, it may be necessary to run LaTeX and GlossTEX up to 4 times until all references are resolved. Watch out for warnings from glosstex during a LaTeX run. Messages about unresolved ⟨*label*⟩s from GlossTEX are caused by missing definitions in the `.gdf`-files.

3.3 Page References

You may want a reference in the list to the place where the term first appears in the text. This can be done using the optional argument ⟨*pageref-mode*⟩ on ⟨*item*⟩-level and ⟨*list*⟩-level. These arguments in combination with the option to `\usepackage` control page references. Table 3 gives an overview of all possible combinations of these 3 arguments.

list	`refpage`			`norefpage`		
entry	`a`	`p`	`n`	`a`	`p`	`n`
`a`	×	×	–	×	×	–
`p`	×	×	–	×	–	–
`n`	×	–	–	×	–	–

Table 3: Options controlling the appearance of page references. A "×" indicates that a reference is produced.

One possible usage of this feature: while debugging a document, turn on page references by using the option `refpage` to the package. Every entry included with the modes `a` (always) or the default `p` (package) will contain a reference. After debugging, remove the option `refpage` and only those entries that were included with mode `a` will still have a reference.

3.4 The Appearance of ⟨*full*⟩

⟨*full*⟩ stands for the appearance of the typeset term in the text. The appearance of ⟨*full*⟩ is controlled through the optional arguments ⟨*lparen*⟩, ⟨*rparen*⟩ and ⟨*form*⟩ to `\ac` and `\acf`. Additionally, these arguments work together with defaults set forth on ⟨*list*⟩-level and package level. The following sections shall shed some light on this topic.

3.4.1 The ⟨*order*⟩ of ⟨*item*⟩s

`\glxitemorderdefault` The options `itemfirst` and `longfirst` control which one of ⟨*item*⟩ and ⟨*long-form*⟩ should be typeset first. On ⟨*list*⟩-level, ⟨*item-order*⟩ can be set using

`\glxitemorderdefault{`⟨*list*⟩`}{`⟨*item-order*⟩`}`

⟨*item-order*⟩ may be either "`i`", "`l`" or empty. `i` first typesets ⟨*item*⟩, `l` first typesets ⟨*long-form*⟩. Omitting ⟨*item-order*⟩ lets the defaults take effect. Table 4 gives an overview over each combination of the options for ⟨*item-order*⟩.

list	`itemfirst`			`longfirst`		
entry	`i`	–	`l`	`i`	–	`l`
`i`	`i`	`i`	`l`	`i`	`i`	`l`
–	`i`	`i`	`l`	`i`	`l`	`l`
`l`	`i`	`l`	`l`	`i`	`l`	`l`

Table 4: Options controlling ⟨*item-order*⟩.

3.4.2 Placement of footnotes and ⟨*form*⟩

`\glxitemplacementdefault` The options `text` and `footnote` control where the second typeset output should go, either into the text or into a footnote.

`\glxitemplacementdefault{`⟨*list*⟩`}{`⟨*item-placement*⟩`}`

sets this option on ⟨*list*⟩-level. ⟨*item-placement*⟩ may be either `t`, `f` or empty. `t` typesets both parts into the text, `f` puts one part into a footnote. Omitting ⟨*item-placement*⟩ lets the defaults for ⟨*item-placement*⟩ take effect. See also table 5 for an overview of all possible combinations.

list	`text`			`footnote`		
entry	`t`	–	`f`	`t`	–	`f`
`t`	`t`	`t`	`f`	`t`	`t`	`f`
–	`t`	`t`	`f`	`t`	`f`	`f`
`f`	`t`	`f`	`f`	`t`	`f`	`f`

Table 5: Options controlling ⟨*item-placement*⟩.

The aforementioned ⟨*form*⟩ is built by simply concatenating ⟨*item-order*⟩ and ⟨*item-placement*⟩. You supply the ⟨*form*⟩ argument to `\ac` and `\acf` by enclosing it in angle brackets, just like `\ac<if>{foo}` or `\ac<t>{bar}`. Please note that ⟨*item-order*⟩ is specified *before* ⟨*item-placement*⟩.

3.4.3 Encapsulation

\glxparendefault
\glxparenlistdefault

When using `t` for ⟨*item-placement*⟩, the second part is encapsulated within ⟨*lparen*⟩ on the left and ⟨*rparen*⟩ on the right.

`\glxparendefault{`⟨*list*⟩`}{`⟨*lparen*⟩`}{`⟨*rparen*⟩`}`

sets this option on package level. The options `roundparen` and `squareparen` may be used as well.

`\glxparenlistdefault{`⟨*list*⟩`}{`⟨*lparen*⟩`}{`⟨*rparen*⟩`}`

sets this option on ⟨*list*⟩-level. On ⟨*item*⟩-level these are supplied to `\ac` and `\acf` just like `\ac,(,),{foo}` or `\acf,--,--,<lt>{bar}`. It may look strange, but I was out of parentheses.

3.5 Cross-References

\glxref
\glxref*

`\glxref`[`*`]`{`⟨*item*⟩`}`

It may be useful to use cross-references in entries. Assume you have referenced `\glosstex{ascii}` which describes the term ASCII (American Standard Code for Information Interchange). You may also want to include EBCDIC (Extended Binary Coded Decimals Interchange Code) as an example for another character encoding. To achieve this, write this into the definition of ASCII

```
See also \glxref{ebcdic}.
```

and GlossTEX then produces "See also EBCDIC" and also includes the definition for EBCDIC into the same list ASCII appears in. Note that `\glxref` is only available within the ⟨*text*⟩ argument in the `.gdf`-file since it only makes sense within a ⟨*list*⟩. There is also a starred version `\glxref*` that doesn't produce typeset output.

3.6 GlossTEX and nomencl

It is possible to use nomencl and GlossTEX in one document without problems. The following commands show how to deal with documents using both GlossTEX and nomencl.

```
latex thesis
glosstex thesis thesis.gdf
makeindex thesis.gxs -o thesis.glx -s glosstex.ist
makeindex thesis.glo -o thesis.gls -s nomencl.ist
latex thesis
```

4 Customizing

4.1 Global

GlossTEX can be customized by using the file `glosstex.cfg` which is automatically loaded if it is present. The file `glosstex.std` is the default configuration file

that is absolutely mandatory to GlossTEX's proper working. It shows all aspects that are meant to be customized, so let's discuss it now.

Each term that gets typeset either goes through `\GLX@output@short` or `\GLX@output@long`, depending whether it's the ⟨*item*⟩ or ⟨*long-form*⟩. These macros each take 3 arguments and get called this way:

```
\GLX@output@short{⟨label⟩}{⟨list⟩}{⟨item⟩}
\GLX@output@long{⟨label⟩}{⟨list⟩}{⟨long-form⟩}
```

`\GLX@output@short` `\GLX@output@long` We just output ⟨*item*⟩ as it is. See section 4.2 for some more elaborate implementation of this macro.

```
⟨*std⟩
\newcommand{\GLX@output@short}[3]{#3}
\newcommand{\GLX@output@long}[3]{#3}
```

`\glosstexpage` This is used to typeset the page at the end of a definition. It uses `\pagename` so that should be defined elsewhere.

```
\newcommand{\glosstexpage}[1]{\nobreak{\itshape\pagename~#1}\nobreak}
```

`\glxgldefault` `\glxacdefault` These macros set the defaults for ⟨*list*⟩ and ⟨*pageref-mode*⟩ which the commands starting with `\gl` (except for `\glxref`, of course) and `\ac` are using.

```
\glxgldefault{glo}{p}
\glxacdefault{acr}{n}
```

`\glxitemorderdefault` `\glxitemplacementdefault` These macros set the defaults concerning ⟨*item-placement*⟩ and ⟨*item-order*⟩ in ⟨*list*⟩. We specify no default on ⟨*list*⟩-level.

```
\glxitemorderdefault{glo}{}
\glxitemplacementdefault{glo}{}
\glxitemorderdefault{acr}{}
\glxitemplacementdefault{acr}{}
```

`\glxparendefault` `\glxparenlistdefault` These macros set the defaults concerning ⟨*lparen*⟩ and ⟨*rparen*⟩ on package and ⟨*list*⟩-level, respectively. To set ⟨*paren*⟩ on package level, you may as well use the options `roundparen` and `squareparen`. Because `roundparen` is the default option, this command is not used. As long as there is no ⟨*paren*⟩-default on ⟨*list*⟩-level, the default on package-level is used. So therefor, `\glxparenlistdefault` is not used here as well.

```
%\glxparendefault{(}{)}
%\glxparenlistdefault{glo}{-*}{*-}
```

`\GLX@benv@glo` `\GLX@eenv@glo` `\GLX@item@glo` `\GLX@benv@acr` `\GLX@eenv@acr` `\GLX@item@acr` Each ⟨*list*⟩ gets embedded into `\GLX@benv@`⟨*list*⟩ and `\GLX@eenv@`⟨*list*⟩, so these macros should provide a reasonable environment. Each line itself is typeset using `\GLX@item@`⟨*list*⟩ which gets called with 7 arguments.

```
\GLX@item@⟨list⟩{⟨label⟩}{⟨list⟩}{⟨long-form⟩}{⟨text⟩}{⟨list⟩}{⟨list-mode⟩}{⟨page-stuff⟩}
```

```
\newcommand{\GLX@benv@glo}{\begin{description}}
\newcommand{\GLX@eenv@glo}{\end{description}}
\newcommand{\GLX@item@glo}[7]{%
  \item[#2]\ifx#3\empty\else\emph{#3}\space\fi#4\space#7}
\newcommand{\GLX@item@acr@label}[1]{\mbox{#1}\dotfill}
\newcommand{\GLX@benv@acr}{
  \begin{list}{}{%
      \renewcommand{\makelabel}{\GLX@item@acr@label}%
```

```
      \setlength{\labelwidth}{7em}%
      \leftmargin\labelwidth \advance\leftmargin by \labelsep}}
\newcommand{\GLX@eenv@acr}{\end{list}}
\newcommand{\GLX@item@acr}[7]{%
  \item[\textsc{#2}]%
    \ifx#3\empty\else#3\quad\fi\ifx#4\empty\else#4\space\fi#7}
```

\glossaryname
\listacronymname
\pagename

These are defined to contain some default strings if they're not already defined (\pagename e.g. is defined trough babel).

```
\ifx\glossaryname\@undefined
  \def\glossaryname{Glossary}
\fi
\ifx\listacronymname\@undefined
  \def\listacronymname{List of Acronyms}
\fi
\ifx\pagename\@undefined
  \def\pagename{page}
\fi
```

\glxheading

Each ⟨*list*⟩ starts with an appropriate heading which is defined by

\glxheading[⟨*list*⟩]{⟨*definition*⟩}.

```
\ifx\chapter\@undefined
  \glxheading{glo}{\section*{\glossaryname}}
  \glxheading{acr}{\section*{\listacronymname}}
\else
  \glxheading{glo}{\chapter*{\glossaryname}}
  \glxheading{acr}{\chapter*{\listacronymname}}
\fi
⟨/std⟩
```

4.2 Local

Now we need a local configuration file for this document. This is done by using a file named glosstex.cfg. In \GLX@output@short we test whether we are in the list of acronyms (GLX@acdef@list) and typeset ⟨*item*⟩ with caps and small caps if we are. Otherwise, just typeset ⟨*item*⟩. We also define a call to \index in \GLX@output@short. As sort-key, we use ⟨*label*⟩. Instead of the usual @ we use = for designating the appearance part of the index entry.

```
⟨*cfg⟩
\renewcommand{\GLX@output@short}[3]{%
  \ifthenelse{\equal{#2}{\GLX@acdef@list}}{%
    \textsc{#3}}{#3}\index{#1=#3}}
⟨/cfg⟩
```

5 Some Details

While reading the .aux-file, GlossTEX only considers the first appearance of one ⟨*item*⟩ for each ⟨*list*⟩. All subsequent entries are silently ignored. (Almost silently, because the .gxg-file will contain detailed information about this, and more.) But if the first entry says not to produce a page reference and a following one

says to do so, then the latter will supersede the former. The same applies if a term is referenced without the option not to generate a list-entry (all commands containing a `*`, e.g. `\gl*`), but a following tells to do so.

While reading one or more `.gdf`-files, only the first definition is used, all other entries are ignored. This fact can be utilised in some way. Assume you have a `master.gdf` which contains general terms and a file `thesis.gdf` which only contains terms that are intended for use in your thesis. Whenever an entry is present in both `.gdf`-files, the one from `thesis.gdf` should be taken. To achieve this, specify `thesis.gdf` *before* `master.gdf`.

Additionally, see the file `TODO` in this package for known bugs (also called features) and not yet implemented features (also called bugs).

6 Acknowledgments

I would like to thank these people who have contributed to the development of GlossTEX:

Daniel Courjon, Stefan A. Deutscher, Michael Friendly, Olaf Michelsson

Glossary

`.gdf`-file This file is the database file containing definitions for GlossTEX. *page 5*

`.glg`-file This is the log-file produced by the MakeIndex-run. See also `.gxg`-file. *page 12*

`.glx`-file This file contains the sorted lists, ready to be read by LaTeX. *page 7*

`.gxg`-file This is the log-file produced by the GlossTEX-run. See also `.glg`-file. *page 7*

`.gxs`-file Intermediate file produced by GlossTEX to be processed by MakeIndex. *page 7*