glossary.sty v 2.24: LaTeX 2ε Package to Assist Generating Glossaries

Nicola L.C. Talbot

27th September 2005

Contents

1 Introduction

The glossary package is provided to assist generating a glossary. It has a certain amount of flexibility, allowing the user to customize the format of the glossary, and define new glossary-style objects.

2 Installation

You need to make sure you have downloaded the following three files:

```
glossary.ins
glossary.dtx
README
```

To extract the code from the `glossary.dtx` file, you will need to run the installation file through LaTeX:

```
latex glossary.ins
```

This will create the following files:

```
glossary.sty
glossary.perl
makeglos.pl
makeglos.bat
```

along with several sample files. The file `glossary.sty` should be placed somewhere in the LaTeX path, e.g. `texmf/tex/latex/glossary/`. The file `glossary.perl` is a LaTeX2HTML style file, and should be placed in the LaTeX2HTML style file directory (usually `latex2html/styles/`). The file `makeglos.pl` is a Perl script which calls `makeindex`. If you are using UNIX or Linux, you will need to set the permissions so that you can execute the file:

```
chmod a+x makeglos.pl
```

You should then place this file somewhere on your path. (You may also need to edit the first line of this file, if `perl` is located in a directory other than `/usr/bin/`[1].)

If you are not using UNIX or Linux, you may have to explicitly load the file into Perl, so you would need to do `perl makeglos.pl` instead of just `makeglos.pl`. If you are using Windows, a batch file, `makeglos.bat` is provided which will run Perl on `makeglos.pl`. Both `makeglos.pl` and `makeglos.bat` should be placed somewhere specified by the PATH environment variable. (For example, put them both in the same directory as `makeindex`, which will probably be in `\texmf\miktex\bin\`).

If you don't have Perl installed on your system, you can just use `makeindex`, only you will have to remember all the command line switches, and you won't be able to merge entries that have the same name, but different descriptions.

3 Generating Glossary Information

`\makeglossary` The standard LaTeX command `\makeglossary` (analogous to `\makeindex`) should

[1] and you can also remove the `.pl` extension which isn't to everyone's liking.

be placed in the document preamble. If this command is omitted, glossary information will be ignored. Glossary entries are generated using the command

\glossary

`\glossary{`⟨*key-val list*⟩`}`. This command is a slightly modified version of the standard `\glossary` command, in order to separate out the information into ⟨*entry-name*⟩ and ⟨*entry-description*⟩. The argument to `\glossary` must be a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs. The following keys are available:

Key	**Value**
`name`	The entry name
`description`	A description about the entry
`sort`	How to sort the entry. (Entry name used if sort omitted)
`format`	How to format the page number

For example:

```
\glossary{name={singular matrix},
          description={A matrix with zero determinant}}
```

The following example sorts on the text `U` instead of `$\mathcal{U}$`:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}
```

The page format for individual entries can be changed using the `format` key. This should be the name of a LaTeX formatting command without the preceding `\` (as with the `|` operator in `\index`.) For example:

```
\glossary{name={$\mathbb{R}$},
          description={The set of real numbers},
          sort=R,
          format=textbf}
```

In addition, the following formats are also available:

`hyperrm`	The number is a hyper link in roman
`hypersf`	The number is a hyper link in sans-serif
`hypertt`	The number is a hyper link in typewriter font
`hyperbf`	The number is a hyper link in bold
`hyperit`	The number is a hyper link in italic

If the `hyper` option has not been set, `hyperrm` etc are equivalent to `textrm` etc. You should use `\hyperrm` instead of `\hyperpage`, as `\hyperpage` won't work on a list or range of numbers in the glossary [2].

As with the `\index` command, care must be taken if you want to use the special characters: `@ | "` and `!`. These characters should be preceded by the double quote character. For example:

```
\glossary{name={$"|\mathcal{S}"|$,
description=The cardinality of the set \mathcal{S}}}
```

There is no provision for sub-entries, as these are generally only applicable in an index, and not in a glossary.

\xglossary

As from version 2.14, there is an additional command available:

[2] This is because the list and number ranges are delimited using `\delimR` and `\delimN` instead of explicitly using a comma or en-dash.

\xglossary{⟨*gls-entry*⟩ }{⟨*text*⟩}

This is equivalent to⟨*text*⟩\glossary{⟨*gls-entry*⟩}, where ⟨*text*⟩ will be made a hyperlink to the relevant entry in the glossary, if hyper links are supported.

3.1 Storing Glossary Information

\storeglosentry The following command:

\storeglosentry[⟨*gls-type*⟩]{⟨*label*⟩}{⟨*gls-entry*⟩}

can be used to store glossary information. That information can then be used later with any of the following commands:

\useglosentry
\useGlosentry
\gls

\useglosentry[⟨*opt*⟩]{⟨*label*⟩}
\useGlosentry[⟨*opt*⟩]{⟨*label*⟩}{⟨*text*⟩}
\gls[⟨*opt*⟩]{⟨*label*⟩}

\useglosentry adds the stored glossary entry ⟨*gls-entry*⟩ to the appropriate glossary, \useGlosentry adds the stored glossary entry, and makes ⟨*text*⟩ a hyperlink to that entry (if hyperlinks are supported). The third command, \gls, is like \useGlosentry, but forms ⟨*text*⟩ from the name given in ⟨*gls-entry*⟩.

Returning to an earlier example, instead of typing:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}
```

every time you want to add this entry to the glossary, you can instead store the information:

```
\storeglosentry{glos:U}{name={$\protect\mathcal{U}$},
          description={The universal set},
          sort=U}
```

(Note the use of \protect.) Now, instead of continually copying and pasting the glossary command for this entry (which can have quite a large description field), you can use either:

```
\useglosentry{glos:U}
```

which is equivalent to:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}
```

or you can use:

```
\useGlosentry{glos:U}{text}
```

which is equivalent to:

```
\xglossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}{text}
```

or you can use:

```
\gls{glos:U}
```

which is equivalent to:

```
\xglossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U}{$\mathcal{U}$}
```

The optional argument ⟨*gls-type*⟩ indicates the glossary type (see section 7 to find out how to define new glossary types). If omitted, the standard glossary is used.

The optional argument ⟨*opt*⟩ allows you to add additional information to the glossary entry, for example:

```
\useglosentry[format=textbf]{glos:U}
```

is equivalent to:

```
\glossary{name={$\mathcal{U}$},
          description={The universal set},
          sort=U,
          format=textbf}
```

The identifying label, ⟨*label*⟩, should not contain any special characters, and since `\storeglosentry` is fragile, care should be taken when using commands within its argument. If in doubt, use `\protect`.

4 makeglos.pl

The glossary information (as given by the commands `\glossary` and `\xglossary`) is saved in the file with the extension `glo` (unless the `\makeglossary` command is omitted, in which case the glossary information is simply ignored.) A `makeindex` style file (`ist`) is also created, which is customized for the document, and can be passed to `makeindex`.

For example, suppose your document is called `mydoc.tex`, the glossary will be saved in the file `mydoc.glo`, and the `makeindex` style file `mydoc.ist` will be created. These files can then be passed to `makeindex` as follows:

```
makeindex -s mydoc.ist -t mydoc.glg -o mydoc.gls mydoc.glo
```

which generates the output file `mydoc.gls`, with transcript written to `mydoc.glg`.

The Perl script `makeglos.pl` provided with this package allows you to use `makeindex` without having to remember all the command line options. The command

```
makeglos.pl mydoc.glo
```

will perform the command:

```
makeindex -s mydoc.ist -t mydoc.glg -o mydoc.gls mydoc.glo
```

The `glo` extension may be omitted. In addition, `makeglos.pl` also takes the option `-m` which can be used to collate entries where the same name has multiple descriptions.

`makeglos.pl` has the following syntax:

```
makeglos.pl [-ilqrgm] [-s sty] [-o gls] [-t log] [-p num] <filename>
```

where all switches, apart from `-m` are the same as those for `makeindex`. In addition, if the extension to the input file name is omitted, `glo` will be assumed. If there are multiple glossary types (see section 7) and the file extension is omitted, `makeglos.pl` will iterate through each glossary type.

`\istfilename`

The name of the `ist` file can be changed by redefining the command `\istfilename`. For example:

```
\renewcommand{\istfilename}{foo.ist}
```

However, you will need to explicitly pass the name of this file to `makeglos.pl` using the `-s` switch.

`\noist`

Creation of the `ist` file can be suppressed by issuing the command `\noist` before `\makeglossary`. It will also be suppressed when the command `\nofiles` is used, or if the command `\makeglossary` is omitted.

It should be noted that there are a few packages that can cause problems with the creation of the `ist` file, for example ngerman. If you encounter problems when LaTeX is processing the `\makeglossary` command, or if you get errors from `makeindex` complaining about the style file, this is the most probable cause. See section 12, item 15 for information on how to fix this.

5 Displaying the Glossary

`\printglossary`

Once the `gls` file has been created by `makeindex` (as described in the previous section) the glossary can then be included in the document with the command `\printglossary`. If chapters are defined, the glossary will start with

```
\chapter*{\glossaryname}
```

If not, it will start with

```
\section*{\glossaryname}
```

The format of the main body of the glossary depends on the options passed to the package.

6 Package Options

The package options must be specified as a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs. Available options are:

`style` The glossary style. Values:

- `list` use description environment in the glossary
- `altlist` modified version of `style=list`. The description starts on the line following the name of the term being defined.
- `super` use supertabular environment in the glossary
- `long` use longtable environment in the glossary (Default)

`header` Glossary header. Values:

- `none` The glossary doesn't have a heading (Default)

`plain` The glossary has a heading

`border` Glossary border. Values:

`none` The glossary doesn't have a border (Default)

`plain` Border around the main body of the glossary

`cols` Number of columns. Values:

`2` The entry name and description are in two separate columns with the associated page numbers in the same column as the description. (Default)

`3` The entry name, description and associated page numbers are in three separate columns.

`number` Associated number corresponding to each entry. Values:

`page` Each entry has the corresponding page number(s) where the entry is defined. (Default)

`section` Each entry has the corresponding section number(s) where the entry is defined.

`none` The corresponding numbers are suppressed.

`toc` Boolean variable:

`true` Add glossary to table of contents

`false` Omit glossary from table of contents (Default)

Note that if you specify this option, you will need to run LaTeX twice after generating the glossary.

`hypertoc` Boolean variable. This is similar to the package option `toc`, but if you are using the hyperref package, `hypertoc` will generate a link to the beginning of the page, whereas `toc` will have a hyperlink to jst after the glossary title. Note that you can not use both `toc=true` and `hypertoc=true`. Default value: `hypertoc=false`.

`hyper` Boolean variable:

`true` Make associated numbers in the glossary a hypertext link, and also make acronyms, and the text given by `\xglossary` have a hyperlink to their corresponding entries in the glossary.

`false` Don't make associated numbers a hypertext link

If the hyperref or html package has been loaded prior to loading `glossary.sty`, `hyper=true` is set, otherwise the default is `hyper=false`. Note that this package option now encompasses the old `hyperacronym` option.

`section` Boolean variable:

`true` Make the glossary an unnumbered section, even if chapters are defined

`false` Only make glossary an unnumbered section if chapters are not defined (default).

`acronym` Boolean variable:

`true` Make the list of acronyms separate from the main glossary.

`false` The acronyms will all be placed in the main glossary. (Default)

The `border`, `header` and `cols` options should not be used in conjunction with `style=list` or `style=altlist`, as they only make sense with one of the tabular-style options. The value for the boolean variables can be omitted if they are to be set. For example `toc` is equivalent to `toc=true`. Note that the `altlist` style is better suited to glossaries with long entry names.

You can set up your own preferred defaults in a configuration file . The file must be called `glossary.cfg` and should be placed somewhere on the TEX path. In this file you can use the command `\glossarypackageoptions{`⟨*option-list*⟩`}` where ⟨*option-list*⟩ is a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs, as passed to the glossary package. Note that this command may only be used in the configuration file.

`\glossarypackageoptions`

6.1 Examples

Suppose the document has the following `\glossary` commands:

Page	Command
1	`\glossary{name=diagonal matrix, description=Matrix whose only non-zero entries are along the leading diagonal}`
2	`\glossary{name=identity matrix, description=Diagonal matrix with 1s along the leading diagonal}`
4	`\glossary{name=singular matrix, description=Matrix with zero determinant}`

Variations:

1. If `style=list` is chosen, the glossary will look like:

   diagonal matrix Matrix whose only non-zero entries are along the leading diagonal, 1

   identity matrix Diagonal matrix with 1s along the leading diagonal, 2

   singular matrix Matrix with zero determinant, 4

2. If `style=altlist` is chosen, the glossary will look like:

   diagonal matrix
   Matrix whose only non-zero entries are along the leading diagonal, 1

   identity matrix
   Diagonal matrix with 1s along the leading diagonal, 2

   singular matrix
   Matrix with zero determinant, 4

3. If `style=list,number=none` is chosen, the glossary will look like:

diagonal matrix Matrix whose only non-zero entries are along the leading diagonal

identity matrix Diagonal matrix with 1s along the leading diagonal

singular matrix Matrix with zero determinant

4. If `style=long`,`border=none`, `header=none`,`number=page` is chosen (default), the glossary will look like:

diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal, 1
identity matrix	Diagonal matrix with 1s along the leading diagonal, 2
singular matrix	Matrix with zero determinant, 4

5. If `style=long`,`border=plain`, `header=none` is chosen, the glossary will look like:

diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal, 1
identity matrix	Diagonal matrix with 1s along the leading diagonal, 2
singular matrix	Matrix with zero determinant, 4

6. If `style=long`,`border=plain`, `header=plain` is chosen, the glossary will look like:

Notation	**Description**
diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal, 1
identity matrix	Diagonal matrix with 1s along the leading diagonal, 2
singular matrix	Matrix with zero determinant, 4

7. If `style=long`,`border=none`, `header=plain`,`cols=3` is chosen, the glossary will look like:

Notation	Description	
diagonal matrix	Matrix whose only non-zero entries are along the leading diagonal	1
identity matrix	Diagonal matrix with 1s along the leading diagonal	2
singular matrix	Matrix with zero determinant	4

7 Defining New Glossary Types

`\newglossarytype` A new type of glossary can be defined using the command:

`\newglossarytype[`⟨*log-ext*⟩`]{`⟨*name*⟩`}{`⟨*out-ext*⟩`}{`⟨*in-ext*⟩`}`

For example, suppose you want your document to have a separate index of terms and index of notation, you could use `\makeglossary`, `\glossary`, `\xglossary` and `\printglossary` for the first glossary, and define a new type of glossary called say, `notation`, using

```
\newglossarytype[nlg]{notation}{not}{ntn}
```

which will create the analogous commands: `\makenotation`, `\notation`, `\xnotation` and `\printnotation` which can be used for the second glossary.

The command `\newglossarytype` should only occur in the preamble. The new commands `\make`⟨*name*⟩, `\`⟨*name*⟩, `\x`⟨*name*⟩ and `\print`⟨*name*⟩ all have the same format as their "glossary" counter-parts.

The glossary information will be saved to a file with the extension given by ⟨*out-ext*⟩ (analogous to `glo`), which can then be passed to `makeindex` either directly or via `makeglos.pl`, and the file to be read in (i.e. the file created by `makeindex`) will have the extension ⟨*in-ext*⟩ (analogous to `gls`).

The optional argument ⟨*log-ext*⟩ indicates the extension for the `makeindex` log file, if omitted the extension `glg` is used. This is not used by LaTeX, however `makeglos.pl` reads in this information from the LaTeX auxiliary file and passes it to `makeindex`.

For the above `notation` example, if your document is called, say, `mydoc.tex`, you will need to do the following:

```
latex mydoc
makeglos.pl mydoc
latex mydoc
```

(You may need to do an extra `latex mydoc` to get cross-references up-to-date.) Note that if you don't specify the file extension when using `makeglos.pl`, it will check the transcript file from the LaTeX run to determine all the glossary types, so, for this example,

```
makeglos.pl mydoc
```

is equivalent to:

```
makeglos.pl mydoc.glo
makeglos.pl mydoc.not
```

since `makeglos.pl` has read in the information for the `notation` glossary type from the file `mydoc.log`.

If you don't have Perl installed on your system, or for any other reason are unable to use `makeglos.pl`, you can call `makeindex` explicitly:

```
latex mydoc
makeindex -s mydoc.ist -t mydoc.glg -o mydoc.gls mydoc.glo
makeindex -s mydoc.ist -t mydoc.nlg -o mydoc.ntn mydoc.not
latex mydoc
```

Note that you can use the command `\printglossary[`⟨*name*⟩`]` instead of `\print`⟨*name*⟩. These two commands have the same effect when using LaTeX, however, they have a slightly different effect when using LaTeX2HTML (see section 11).

If the command `\`⟨*glossary-type*⟩`name` is defined, (e.g. `\notationname` in the above example) this will be used as the title for the specified glossary. If this command is not defined, `\glossaryname` will be used instead. If the command `\short`⟨*glossary-type*⟩`name` is defined, (e.g. `\shortnotationname` in the above example) this will be used for the table of contents entry, otherwise `\`⟨*glossary-type*⟩`name` will be used instead. For example:

```
\newglossarytype[nlg]{notation}{not}{ntn}
\newcommand{\notationname}{Index of Notation}
\newcommand{\shortnotationname}{Notation}
```

8 Acronyms

`\newacronym` The glossary package provides the command:

`\newacronym[`⟨*cmd-name*⟩`]{`⟨*acronym*⟩`}{`⟨*long*⟩`}{`⟨*glossary entry*⟩`}`

which can be used to define acronyms. The argument ⟨*long*⟩ is the full name, the argument ⟨*acronym*⟩ is the acronym for ⟨*long*⟩ and ⟨*glossary entry*⟩ is the glossary information in the form used by the `\glossary` command. If the optional argument ⟨*cmd-name*⟩ is missing, `\newacronym` will create a command called `\`⟨*acronym*⟩, otherwise it will create a command called `\`⟨*cmd-name*⟩ (henceforth denoted `\`⟨*acr-name*⟩). This command can then be used throughout the text. The first instance of this command is equivalent to:

⟨*long*⟩ `(\xacronym{name=`⟨*long*⟩ `(`⟨*acronym*⟩`),`⟨*glossary entry*⟩`}{`⟨*acronym*⟩`})`

subsequent instances will be equivalent to:

`\xacronym{name=`⟨*long*⟩ `(`⟨*acronym*⟩`),`⟨*glossary entry*⟩`}{`⟨*acronym*⟩`}`

The command `\`⟨*acr-name*⟩ also has a starred version, which will make the first letter of ⟨*long*⟩ uppercase (for use at the start of a sentence).

Note that if you want to change the format of the acronym, for example, if you want the acronym to appear in small caps, you will need to not only use the optional argument, but you will also need to use the `sort` key, otherwise you will get an error. For example:

```
\newacronym[SVM]{\textsc{svm}}{Support Vector Machine}%
```

```
{description=Statistical pattern recognition
technique,sort=svm}
```

If the package option `acronym` is not set (default) `\xacronym`, is a synonym for `\xglossary`. If the package option `acronym=true` is specified, a new glossary type called `acronym` will be defined as:

```
\newglossarytype[alg]{acronym}{acr}{acn}
\newcommand{\acronymname}{List of Acronyms}
```

You will then need to use the commands `\makeacronym` and `\printacronym` to make the list of acronyms appear. You will also need to run the `acr` file through `makeindex` (or `makeglos.pl`). For example:

```
makeindex -s mydoc.ist -t mydoc.alg -o mydoc.acn mydoc.acr
```

alternatively:

```
makeglos.pl mydoc
```

Note that the package option `acronym=true` is only appropriate if you want both a glossary and a separate list of acronyms.

The `name` key is not used in ⟨*glossary entry*⟩, as it is constructed from ⟨*long*⟩ and ⟨*acronym*⟩. By default this will be in the form: ⟨*long*⟩ (⟨*acronym*⟩), however the format can be overridden using the command:

`\setacronymnamefmt`

`\setacronymnamefmt{`⟨*format*⟩`}`

Within ⟨*format*⟩ the following commands may be used to represent ⟨*long*⟩ and ⟨*acronym*⟩: `\glolong` and `\gloshort`. For example, suppose you just want the acronym to appear in the glossary entry, and not its full length name, then you would need to do:

`\glolong`
`\gloshort`

```
\setacronymnamefmt{\gloshort}
```

Given an acronym named ⟨*acr-name*⟩ (the command name associated with the acronym as defined in `\newacronym` without the preceding backslash), the following commands are also available:

`\useacronym` `\useacronym[`⟨*insert*⟩`]{`⟨*acr-name*⟩`}`

This command can be used instead of `\`⟨*acr-name*⟩. `\useacronym` also has a starred version equivalent to `\`⟨*acr-name*⟩`*`. The optional argument ⟨*insert*⟩ allows you to insert text after ⟨*long*⟩, if this is the first occurrence of the acronym, or after the acronym on subsequent occurrences.

`\resetacronym` `\resetacronym{`⟨*acr-name*⟩`}`

This command will cause the next use of `\`⟨*acr-name*⟩ to produce the long version. To reset all acronyms do `\resetallacronyms`.

`\resetallacronyms`

`\ifacronymfirstuse` `\ifacronymfirstuse {`⟨*acr-name*⟩ `}{`⟨*true text*⟩`}{`⟨*false text*⟩`}`

This will test if the acronym has been used yet. If it has been used, ⟨*true text*⟩ will be implemented, otherwise ⟨*false text*⟩ will be implemented.

The long and short forms of an acronym can be produced explicitly without a corresponding glossary entry, using the commands:

\acrln `\acrln{`⟨*acr-name*⟩`}`
\acrsh `\acrsh{`⟨*acr-name*⟩`}`

Or, alternatively:

`\`⟨*acr-name*⟩`long`
`\`⟨*acr-name*⟩`short`

The first two commands (`\acrln` and `\acrsh`) have a starred form that makes the first letter uppercase. The other two commands, simply contain ⟨*long*⟩ and ⟨*acronym*⟩.

Note that since these four commands do not generate glossary entries they will therefore not contain any hyperlinks, even if you have specified the `hyper` package option. They are provided for use in situations where the associated glossary command may cause problems (e.g. in the table of contents entry.)

Note that, as with all LaTeX commands, spaces following command names are ignored so if, for example, you defined a new acronym called, say, SVM, then the `\SVM` will ignore any spaces following it. To force a space, you can either place an empty set of braces after the command name (e.g. `\SVM{}`) or use `\␣` i.e. a backslash followed by a space (e.g. `\SVM\ `). Alternatively, as from version 2.22, if you load the xspace package before loading the glossary package, spaces will be put in automatically using `\xspace`.

\acronymfont If you want the acronym to appear in a particular font, for example, small caps, you can redefine the command `\acronymfont`. For example:

```
\renewcommand{\acronymfont}[1]{\textsc{#1}}
```

The default definition of `\acronymfont` is:

```
\newcommand{\acronymfont}[1]{#1}
```

8.1 Examples

```
\newacronym{SVM}{Support Vector Machine}{description=Statistical
pattern recognition technique}
```

This will define the command `\SVM`. The first time this command is used will display the text: Support Vector Machine (SVM). Subsequent use will simply display: SVM. The next example uses the optional argument ⟨*cmd-name*⟩ since the acronym contains a non-alphabetical character:

```
\newacronym[KSVM]{K-SVM}{Kernel Support Vector
Machine}{description=Statistical pattern recognition
technique using the ``kernel trick''}
```

This will define the command `\KSVM`. The first time this command is used will display the text: Kernel Support Vector Machine (K-SVM). Subsequent use will simply display: K-SVM.

To test whether or not an acronym has been used:

```
\ifacronymfirstuse{SVM}{a}{an} \SVM\ is \ldots
```

If the acronym has not been used, the following text will be produced:

a Support Vector Machine is ...

otherwise it will produce:

an SVM is ...

To expand the acronym a second time:

```
\chapter{An overview of the \protect\SVM}
\resetacronym{SVM}
The \SVM\ \ldots
```

Note the use of `\protect` (see note 14 on page 22.) In fact, in this situation it would be better to do:

```
\chapter[An overview of the \SVMlong]{An overview of the \protect\SVM}
\resetacronym{SVM}
The \SVM\ \ldots
```

Now suppose you want the text: support vector machine, instead of Support Vector Machine (i.e. you don't like the uppercase letters). You can define the acronym as follows:

```
\newacronym{SVM}{support vector machine}{description=Statistical
pattern recognition technique}
```

however, if the command `\SVM` occurs at the start of the sentence, you would clearly want the first letter as an uppercase letter. This can be done using `\SVM*` instead of `\SVM`. For example:

```
\SVM*\ techniques are widely used \ldots
```

This will then come out as: Support vector machine (SVM) techniques are widely used ... (Assuming this is the first use of either `\SVM` or `\SVM*`.)

Alternatively, `\useacronym{SVM}` can be used instead of `\SVM`. For example:

```
\useacronym*[s]{SVM} are widely used in the area of pattern
recognition.
```

If this is the first use of the acronym SVM, it will produce the following text:

Support vector machines (SVM) are widely used in the area of pattern recognition.

If this is not the first use of this acronym, it will produce the following text:

SVMs are widely used in the area of pattern recognition.

9 Customizing the Glossary

The glossary package provides commands which can be redefined to customize the glossary. The following name commands are defined by this package:

Command	Default Value
\glossaryname	Glossary
\shortglossaryname	\glossaryname
\entryname	Notation
\descriptionname	Description

\glossaryname
\entryname
\descriptionname

\shortglossaryname The command \shortglossaryname is used for the page headers and table of contents entry. Any text required before or after the glossary can be added by redefining the commands \glossarypreamble and \glossarypostamble. For example.

\glossarypreamble
\glossarypostamble

```
\renewcommand{\glossarypreamble}{Page numbers in
italic indicate the main definition\par}
```

By default, \glossarypreamble and \glossarypostamble do nothing.

\setglossary The command \setglossary{⟨*key-val list*⟩} can be used to modify some of the glossary settings. The argument ⟨*key-val list*⟩ is a comma-separated list of ⟨*key*⟩=⟨*value*⟩ pairs. Available keys are:

`type` This is the glossary type. If it is omitted, the standard glossary is assumed.

`glsnumformat` This is the name of the command, *without* the preceding backslash[3], to format the entry numbers. For example, to make all the entry numbers italic, do:

```
\setglossary{glsnumformat=textit}
```

To suppress numbering altogether, you can do:

```
\setglossary{glsnumformat=ignore}
```

`glodelim` This specifies what to do after the entry description and before the page numbers. The default value is a comma, unless the `cols=3` option is specified, in which case it has the value `&`, or if `style=altlist`, in which case it is simply a space[4]. If the package option `number=none` is specified, `glodelim` will have an empty value (unless `cols=3` is specified, where, again, it will have the value `&`.) This setting corresponds to the `delim_0` key in the `makeindex` style file.

`delimN` The delimiter to be inserted between two page numbers for the same entry. (This corresponds to the `delim_n` key in the `makeindex` style file.) By default, this has the value `,␣` (comma followed by a space). If the package option `number=none` is chosen, the value is set to empty.

`delimR` The delimiter to be inserted between the starting and ending page number range for the same entry. (This corresponds to the `delim_r` key in the `makeindex` style file.) By default, this has the value `--`. If the package option `number=none` is chosen, the value is set to empty.

[3]Note, you should no longer try redefining the command `\glsnumformat`, as this now takes an optional argument, allowing for different glossary types

[4]This is because the `altlist` style is intended for use with long descriptions that will look better ending with a full stop which the user can add if desired.

Note that:

```
\setglossary{glsnumformat=ignore}
```

is equivalent to

```
\setglossary{glsnumformat=ignore,delimN={},delimR={}}
```

The number associated with each entry is, by default, the page number where that entry was defined. If the package option `number=section` has been chosen, this will be the section number instead. You can choose a different counter by changing the value of the command `\theglossarynum`. For example, suppose you have the following code:

`\theglossarynum`

```
\begin{equation}
\Gamma(z) = \int_{0}^{\infty}e^{-t}t^{z-1}\,dt
\end{equation}
\glossary{name=$\Gamma(z)$,description=Gamma function,sort=Gamma}
```

and you want the glossary to refer to the equation number, instead of the page or section number, this can be done by redefining `\theglossarynum` as follows:

```
\renewcommand{\theglossarynum}{\theequation}
```

If the equation number depends on another counter (such as `chapter`) you will need to specify that the number is in the form ⟨*NUM*⟩.⟨*num*⟩ (e.g. 3.24). This can be done by redefining the `\pagecompositor` command:

`\pagecompositor`

```
\renewcommand{\pagecompositor}{.}
```

(By default this value is `-`, unless the package option `number=section` has been chosen, in which case this command has the value `.` (full stop), as in the example above). The `\pagecompositor` command must be redefined before `\makeglossary` for it to have any effect. If the package option `number=none` is used, the numbering will be suppressed, regardless of the value of `\theglossarynum`. This command corresponds to the `page_compositor` key in the `makeindex` style file.

The start and end of the main body of the glossary is given by the commands: `\beforeglossary` and `\afterglossary`. If the `style=list` or `style=altlist` package options are chosen these commands simply begin and end the description environment, otherwise these commands begin and end the longtable or supertabular environment with argument specified by `\glossaryalignment`.

`\beforeglossary`
`\afterglossary`
`\glossaryalignment`

The above conflicts with the array package, so as from version 2.1, if the array package has been loaded *prior* to loading the glossary package, the new column type `G` will be defined and used instead of `\glossaryalignment`.

The command `\gloitem` indicates what to do at the start of each glossary entry. This command takes one argument, which will be the text specified by the `name` key in the `\glossary` command. In the case of the `style=list` option, `\gloitem{`⟨*text*⟩`}` will do

`\gloitem`

`\item[`⟨*text*⟩`]`

or if `style=altlist`:

`\item[`⟨*text*⟩`]\mbox{}\par`

otherwise it will do

⟨*text*⟩ `&`

if it's the first item of the page or

`\\` ⟨*text*⟩ `&`

otherwise. This command corresponds to the `item_0` key in the `makeindex` style file.

`\gloskip` The command `\gloskip` specifies what to do between groups. If `style=list` or `style=altlist` this has the value `\indexspace`, otherwise it creates a blank line in the longtable or supertabular environment. This command corresponds to the `group_skip` key in the `makeindex` style file.

`\glossaryheader` The command `\glossaryheader` (if `style=long` or `style=super` is selected) indicates what to do at the start of the longtable or supertabular environment. If `border=none`, it is defined as:

```
\bfseries\entryname & \bfseries \descriptionname\\
```

So, if you want the `\descriptionname` to be centred, you could do:

```
\renewcommand{\glossaryheader}{\bfseries\entryname &
 \hfil\bfseries\descriptionname\\}
```

`\glosstail` The command `\glosstail` indicates what to do at the end of the longtable or supertabular environment.

`\descriptionwidth` The width of the second column for the tabular-type styles is given by the length `\descriptionwidth`. This value can be changed using the `\setlength` command (the default value is `0.6\textwidth`).

10 Sample Documents

This package comes with the following sample documents:

- `sampleSec.tex` — This document uses the options: `style=altlist`, `toc` and `number=section`. It also loads the hyperref package before loading the glossary package, so the glossary has hyperlinks to the section numbers. Experimenting with different package options, will illustrate the different glossary styles. You will need to do:

  ```
  pdflatex sampleSec
  makeglos.pl sampleSec
  pdflatex sampleSec
  pdflatex sampleSec
  ```

  If you don't want to use `makeglos.pl`, you will need to do

  ```
  makeindex -s sampleSec.ist -t sampleSec.glg -o sampleSec.gls sampleSec.glo
  ```

- `sampleNtn.tex` — This has a glossary and defines a new glossary type called `notation`. The glossary has associated page numbers, but the new glossary type doesn't. You will need to do:

  ```
  latex sampleNtn
  makeglos.pl sampleNtn
  latex sampleNtn
  latex sampleNtn
  ```

If you don't want to use `makeglos.pl`, you will need to do

```
makeindex -s sampleNtn.ist -t sampleNtn.glg -o sampleNtn.gls sampleNtn.glo
makeindex -s sampleNtn.ist -t sampleNtn.nlg -o sampleNtn.ntn sampleNtn.not
```

- `sampleNtn2.tex` —This is similar to `sampleNtn.tex`, but uses `\storeglosentry`.
- `sampleEq.tex` — This has a glossary where the numbers in the glossary refer to the equation number rather than the page number. The `\pagecompositor`, `\theglossarynum`, `\glossaryname` and `\glossaryheader` are all redefined to customize the glossary. You will need to do:

  ```
  latex sampleEq
  makeglos.pl sampleEq
  latex sampleEq
  ```

  If you don't want to use `makeglos.pl`, you will need to do

  ```
  makeindex -s sampleEq.ist -t sampleEq.glg -o sampleEq.gls sampleEq.glo
  ```

- `sampleEqPg.tex` — This is a modified version of `sampleEq.tex`. This example has one glossary, where some of the entry numbers refer to the corresponding page number, and some of the entry numbers refer to the corresponding equation number. You will need to do:

  ```
  latex sampleEqPg
  makeglos.pl sampleEqPg
  latex sampleEqPg
  ```

  If you don't want to use `makeglos.pl`, you will need to do

  ```
  makeindex -s sampleEqPg.ist -t sampleEqPg.glg -o sampleEqPg.gls sampleEqPg.glo
  ```

- `sampleAcr.tex` — This has a glossary containing acronyms. It uses the style `altlist` as this is better suited to glossaries with long names. It also uses the hyperref package, so the page numbers in the glossary will automatically be hyperlinks, and the acronyms within the text will have hyperlinks to their corresponding entry in the glossary. You will need to do:

  ```
  pdflatex sampleAcr
  makeglos.pl sampleAcr
  pdflatex sampleAcr
  pdflatex sampleAcr
  ```

  If you don't want to use `makeglos.pl`, you will need to do

  ```
  makeindex -s sampleAcr.ist -t sampleAcr.glg -o sampleAcr.gls sampleAcr.glo
  ```

- `sample.tex` — This has a glossary entry with two different definitions of the same name. If you just use `makeindex`, the two entries will be treated separately, however, if you want them concatenated, you can use `makeglos.pl` with the `-m` switch. You will need to do:

  ```
  pdflatex sample
  makeglos.pl -m sample
  pdflatex sample
  pdflatex sample
  ```

(Depending on the configuration of your system, you may have to do `perl makeglos.pl` instead of just `makeglos.pl`)

If you don't want to use `makeglos.pl`, you will need to do

```
makeindex -s sample.ist -t sample.glg -o sample.gls sample.glo
```

however, the entries with the same name but multiple descriptions will not be merged.

11 LaTeX2HTML Style File

A LaTeX2HTML Perl script, `glossary.perl`, is provided with this package for those wishing to use the glossary package with the LaTeX2HTML translator. The file `glossary.perl` should be extracted along with `glossary.sty` when you run the installation script (`glossary.ins`) through LaTeX.

11.1 Limitations

- The only package options supported are: `style=altlist`, `hyper=true`, `toc=true`, `acronym=true` and `acronym=false`.
- If you have more than one glossary type, the secondary glossaries will occur in the same segment as the primary glossary if you use the command `\print`⟨*name*⟩ instead of `\printglossary[`⟨*name*⟩`]`, where ⟨*name*⟩ is the name of the glossary type.
- The command `\setglossary` must be placed in the preamble to have an effect.
- The `\storeglosentry` commands must be in the document environment to have an effect. (They don't seem to work in the preamble, I don't know why.)
- If you place a `\glossary` command inside an environment not translated by LaTeX2HTML (for example, inside a mathematics environment), it will not be entered into the glossary.
- The combinations `""`, `"|`, `"!` and `"@` will be correctly translated, unless they occur within a maths environment. This is because the maths environment is translated before being passed to `\glossary`. You can overcome this by doing, e.g.:

```
\begin{latexonly}
\glossary{name=$"|\mathcal{S}"|$,description=cardinality of set
$\mathcal{S}$,sort=cardinality}
\end{latexonly}
\begin{htmlonly}
\glossary{name=$|\mathcal{S}|$,description=cardinality of set
$\mathcal{S}$,sort=cardinality}
\end{htmlonly}
```

Alternative, you can use `\mid` instead:

```
\glossary{name=$\mid\mathcal{S}\mid$,description=cardinality of
set $\mathcal{S}$,sort=cardinality}
```

- Glossary items with the same names but different definitions will not be merged.
- The command `\theglossarynum` is ignored by LaTeX2HTML, as page numbers have no meaning within HTML files.
- The configuration file `glossary.cfg` is ignored.

12 Troubleshooting

1. My glossary hasn't appeared.

   Check the following:

   - Have you included the command `\makeglossary` in the preamble?
   - Have you put the command `\printglossary` where you want the glossary to appear?
   - Have you used `makeglos.pl` or `makeindex`, and if you did, did it successfully create the `gls` file? (Check the transcript `glg` file.)
     - If you used `makeindex` directly, did you specify the `ist` file created by `\makeglossary`, and did you remember to specify the output file with the extension `gls`?
     - When `makeindex` scans the `ist` file, it should generate the message:
       `9 attributes redefined, 0 ignored`
       If you have a number other than 0 ignored, then there is something wrong with the `ist` file. Some packages can cause problems with the creation of this file, see item 15 below.
   - Have you remembered to LaTeX your document again after using `makeglos.pl` or `makeindex`?
   - Have you used `\glossary` or `\xglossary`?
   - If you have used `\storeglosentry`, have you also used `\useglosentry`, `\useGlosentry` or `\gls`?

   If you have defined a new glossary type, have you checked all the analogous commands to the above?

2. My list of acronyms hasn't appeared.

   Have you used the `acronym=true` package option? If no, check the answers to the previous item, if yes, make sure you have used `\makeacronym` and `\printacronym`. Have you used any of the acronyms you have defined? Remember that `\acrsh`, `\acrln`, `\`⟨*acr-name*⟩`short` and `\`⟨*acr-name*⟩`long` don't generate entries in the list of acronyms, where `\`⟨*acr-name*⟩ is the name of an acronym command.

3. I get an error when using the command `\saveglosentry`.

   Don't use this command it's obsolete, use `\storeglosentry` instead.

4. One of more of my glossary entries hasn't appeared.

   Check the following

   - If you defined the entry using `\storeglosentry` have you used either `\useglosentry`, `\useGlosentry` or `\gls`?
   - Have you remembered to `\protect` commands such as `\mathcal` within `\storeglosentry`?
   - Have you used the characters `@ ! | "`? If so, have you preceded them with a double quote character?

   Check the `makeindex` log file to see if there are any error messages.

5. My glossary has duplicate entries on separate lines.

   LaTeX treats multiple spaces equivalent to a single space, but `makeindex` takes spaces into account when determining whether two entries are identical. For example:

   ```
   \glossary{name=Identity␣matrix,
   description=diagonal␣matrix␣with␣1's␣along␣the␣diagonal}
   ```

   and

   ```
   \glossary{name=Identity␣␣matrix,
   description=diagonal␣matrix␣with␣1's␣along␣the␣diagonal}
   ```

   will be treated as different entries by `makeindex`, because the first has only one space between 'Identity' and 'matrix' and the second has two. The easiest way to ensure consistency is to use `\storeglosentry` together with `\useglosentry`, `\useGlosentry` or `\gls`.

6. I had an error, fixed it, but I keep getting the same error message.

   Suppose you've made an error in the `\glossary` command. For example:

   ```
   \glossary{name=Java,description=A programming language,format=texbf}
   ```

   In this case `textbf` has been mis-spelt. This error will be copied to the `glo` file, which in turn will be copied to the `gls` file by `makeindex`. A subsequent run of LaTeX will read this error in. If you fix the error in your main document, the error will still be read in from the `gls` file. The best thing to do is to delete the `gls` file, and try again.

7. My glossary has ended up wider than my page.

   This may occur if you have long entry names, and you are using either the `style=long` or `style=super` options. The width of the description column is proportional to the text width (in fact, it's `0.6\textwidth`) but the first column is as wide as the widest entry name. You can either redefine `\glossaryalignment` to change the column specifications, or use one of the list-type styles.

8. The page numbers in my glossary don't match up with the actual page numbers where the entry was defined.

   You may need to LaTeX your document again (just as you have to do with `\tableofcontents`, `\listoffigures` etc).

9. I'm getting a `keyval` error.

   The glossary package uses the keyval package to extract the information from $\langle key \rangle = \langle value \rangle$ comma separated lists. You need to make sure the syntax is correct. If your $\langle value \rangle$ contains a comma, you will need to enclose $\langle value \rangle$ in curly braces. See the keyval documentation for further information[5].

10. I've used the `hyper` option, but nothing happens when I click on the numbers in the glossary.

    Check the following:

    (a) Have you remembered to use PDFLaTeX instead of LaTeX, or used a driver that understands hyperlinks?

    (b) Have you remembered to use the hyperref or html package?

    (c) Have you remembered to use a formatting command which uses `\hyperlink`? (E.g. using `hyperbf` instead of `textbf`)? Remember to check the `format` key in your `\glossary` commands, and the `glsnumformat` key in the `\setglossary` command.

    (d) What application are you using to view the PDF file? Ghostview can display a PDF file, but ignores the links. If you are using Windows, try using Adobe's Acrobat Reader, or if you are using UNIX or Linux, try using `xpdf` or `acroread`.

11. The glossary package conflicts with the datetime package.

    This has been fixed in version 2.01.

12. I get an error when using certain commands, such as `\cite` or `~` in `\newacronym`.

    This has been fixed in version 2.1.

13. I get the following error:

    ```
    ! Package array Error: Illegal pream-token (\glossaryalignment): `c' used.
    ```

    The glossary package used to conflict with the array package. This was fixed in version 2.1, however, you must load the array package *prior* to loading the glossary package.

14. I get the following error:

    ```
    Use of \@chapter doesn't match its definition
    ```

    or

    ```
    ! Argument of \@sect has an extra }
    ```

[5] This should be in the directory `texmf/doc/latex/graphics/`

If you want to use an acronym command in a moving argument (such as a chapter heading) you need to `\protect` it first. Note that if you do put an acronym in a chapter etc heading, it will be expanded for the first time in the table of contents, not in the chapter heading. The best way to get around this is to use the optional argument, e.g.

```
\chapter[Introduction to Kernel Support Vector Machines]{Introduction
to \protect\KSVM}
```

You will also need to do this if you are using bookmarks in a PDF document.

Alternatively, you can do:

```
\resetacronym{KSVM}
\chapter{Introduction to \protect\KSVM}
```

or if you are using PDFLaTeX:

```
\resetacronym{KSVM}
\chapter{Introduction to \texorpdfstring{\protect\KSVM}{KSVM}}
```

15. The glossary package conflicts with ngerman.

    This problem is caused by the fact that ngerman redefines the effect of the double quote character, but this character is used in the creation of the `ist` `makeindex` style file. Try one of the following methods:

    (a) Include the ngerman package after the `\makeglossary` command:

    ```
    \usepackage{glossary}
    \makeglossary
    \usepackage{ngerman}
    ```

    (b) First omit the ngerman package and include `\makeglossary` then LaTeX your document. This will create the `ist` file. Then include the ngerman package, and insert `\noist` before the `\makeglossary` command, this will prevent further attempts to generate the `ist` file.

    ```
    \usepackage{ngerman}
    \usepackage{glossary}
    \noist\makeglossary
    ```

    (c) Use `\noist`, as above, and create the `ist` file in an ordinary text editor. The file should contain the following lines:

    ```
    keyword "\\glossaryentry"
    preamble "\\begin{theglossary}"
    postamble "\n\\end{theglossary}\n"
    group_skip "\\gloskip "
    item_0 "\n\\gloitem "
    delim_0 "\n\\glodelim "
    page_compositor "-"
    delim_n "\\delimN "
    delim_r "\\delimR "
    ```

    It is possible that there may be other packages which will also cause a problem, if so, try any of the above.

16. `makeglos.pl` gives the following error message:

    ```
    unable to extract name from glossary item:
    ```

    You are using an old version of `makeglos.pl` with a new version of the glossary package. You will need to update your version `makeglos.pl`.

Let me know if you encounter any other problems or if you have any comments regarding this package.

13 Obsolete Commands

The commands described in this section are now obsolete, but are currently still provided for backwards compatibility. Their use is deprecated.

`\saveglosentry` `\saveglosentry{⟨name⟩}{⟨description⟩}`
This command has now been replaced by `\storeglosentry`.

`\glsprimaryfmt` The command `\glsprimaryfmt` is no longer used in `\newacronym` as you can end up with duplicate page numbers, however the command is currently still defined (although it may be removed in future versions.)

The package option `hyperacronym` is now superseded by the package option `hyper`. This option was implemented prior to the introduction of the command `\xglossary`. Since the acronyms now use `\xglossary`, there is no difference between the `hyperacronym` and `hyper` options. This option has a boolean value:

`true` Make acronyms link to their corresponding entry in the glossary

`false` Acronyms don't have a hyperlink.

If the hyperref package has been loaded prior to loading `glossary.sty` or if `hyper=true` is set, `hyperacronym=true` otherwise `hyperacronym=false`.

14 Contact Details

Dr Nicola Talbot
School of Computing Sciences
University of East Anglia
Norwich. Norfolk
NR4 7TJ. United Kingdom.
http://theoval.cmp.uea.ac.uk/~nlct/

15 Acknowledgements

I would like to thank the following people for their suggestions: Jens Happe, Dmitry Katsubo, Markus Lazanowski, Slava Pestov and Dario Teixeira

Change History

Index