linguex.sty Documentation

Wolfgang Sternefeld
Version 3.4 — January 2002

Abstract

`linguex.sty` is a LaTeX-tool written for the lazy linguist. Its handling of example numbering, indentations, indexed brackets, and grammaticality judgments reduces the effort of formatting linguistic examples to a minimum. It also allows for automatic extraction of a handout.

The macro runs with LaTeX 2_ε only and requires two additional style files: `xspace.sty` from the base of and `cgloss4e.sty` (a modification of the midnight gloss macro) for glosses.

The Three Basic Commands

The linguex macro defines three basic commands: `\ex.`, `\a.`, and `\b.`. The first two generate list environments. The third functions basically like an `\item`. Here is an example. Type setting

```
\ex. This is the first level of embedding
    \a. This is the second level
    \b. This is still the second level, but:
        \a. This is the third level
        \b. This is not the end.
        \b. This is the end
```

will print:

(1) This is the first level of embedding
  a. This is the second level
  b. This is still the second level, but:
    (i) This is the third level
    (ii) This is not the end.
    (iii) This is the end

The list environment created by `\ex.` must be closed by a single blank line (a `\par`). The text following this list will not be indented. In order to get a `\parindent`, a *second* blank line must be added, immediately following the first.

The commands `\c.`, `\d.`, `\e.`, and `\f.` are equivalent copies of `\b.`; they were defined to produce a kind of WYSIWYG-effect.

The blank line at the end of (1) simultaneously closes the three lists of (1) (opened by `\ex.`, `\a.`, and the embedded `\a.`). To close just one of the embedded lists (introduced by `\a.`) put `\z.` at the point of transition from level n of the embedding to level $n-1$. This is exemplified by the following example:

(2) a. *Government:*
  A governs B iff
  (i) A is a governor;
  (ii) A m-commands B;
  (iii) no barrier intervenes between A and B
 b. Governors are lexical nodes and tensed I.
 (from Haegeman 1991)

```
This is exemplified by the following example:
\ex.\a. {\it Government:}\\
        A governs B iff
         \a. A is a governor;
         \b. A m-commands B;
         \c. no barrier intervenes between A and B
         \z.
    \b.  Governors are lexical nodes and tensed I.
    \z.
    (from Haegeman 1991)
```

Cross References

can be handled by `\label{...}` and `\ref{...}` in the usual manner. For example, a label between `\ex.` and `\a.` stores the main example number for further reference; the same command following `\a.` or `\b.` stores the label of a (sub-)subexample for further reference.

For cross references in the immediate vicinity of an example, you need not `\label` the examples. `\Next` refers to the following example number, and `\NNext` to the next but one. Reference to the previous example is provided by `\Last`; the penultimate one is referred to with `\LLast`.

These shorthands always refer to the first level of embedding. Reference to subexamples can only be obtained by adding an (optional) argument to the above Next- and Last-commands. For example, saying `\NNext[g-ii]` right now yields (4-g-ii). The dash between 4 and g is defined as `\refdash`. It can be suppressed by `\renewcommand{\refdash}{}`.

Footnotes

Inside a footnote, the `\Next`-command presupposes that the next example is still inside the footnote rather than within the main body of the document. Say `\TextNext` to refer to the next example within the main text. (When you intend to make endnotes instead of footnotes, all cross references from footnote to text must be handled by `\label{...}` and `\ref{...}`.)

In case an `\ex.`-environment contains a footnote that contains a `\par` (= a blank line — this will necessarily be the case with a footnote containing itself an `\ex.`), the `\par` will be misinterpreted as the end of the main text example. It is therefore necessary to split the command `\footnote{...}` into `\footnotemark` and `\footnotetext{...}` (see LaTeX-manual). The footnotetext must be placed outside the `\ex.`-environment of the main text.

Some style files (e.g. endnote.sty) modify the definition of footnotes (as linguex.sty itself does so in order to let `\ex.` know whether or not it is inside a footnote). Such style files must be accommodated to linguex.sty by making sure that `\if@noftnote` is set false at the beginning of each footnote (by saying `\@noftnotefalse` in the modified footnote definition); otherwise you will get the arabic style of example numberings (as being used in the main text) rather than the roman numbers (being used inside footnotes).

Glosses

require `cgloss4e.sty`, which is input by `linguex.sty`. Instead of writing `\gll` (as would be required by the gloss macro), one should append a 'g' to the last letter of an example command, as shown below:

```
\ex.\a. No gloss
     \bg. This is a first gloss\\
         Dies ist eine erste Glosse\\

\exg.  Dies ist nicht die erste Glosse\\
         This is not the first gloss\\
```

(3) a. No gloss
b. This is a first gloss
Dies ist eine erste Glosse

(4) *Dies ist nicht die erste Glosse
This is not the first gloss

Using `\gll` still works in principle, but should be avoided, for reasons explained in the next section.

Grammaticality Judgments

composed out of *, ?, #, and %, are automatically prefixed at the very beginning of a new list. A standard example like (5) looks like this (to increase readability I also removed mathmode from subscripting):

```
\catcode`_=\active
\def_#1{\ifmmode\sb{#1}\else$\sb{#1}$\fi}
\exg. \%*Wen_i liebt_k seine_i Mutter t_i t_k?\\
        Whom  loves  his   mother\\
        `Who does his mother love?'
```

(5) %*Wen$_i$ liebt$_k$ seine$_i$ Mutter t$_i$ t$_k$?
Whom loves his mother
'Who does his mother love?'

Automatic prefixing implies that **nothing** intervenes between the list opening command and the judgment. In particular, labels must *follow* the grammaticality judgment! Likewise, writing `\ex.\gll` instead of `\exg.` will have the effect of not prefixing the grammaticality judgment.

Labelled Brackets

`\exi.` identifies "words" by looking for space characters between them; it then checks whether the word following a space starts with one of the brackets "[" or "]". The material between such a bracket and the next space character will be subscripted. A standard example would be the following:

```
\exi. *[CP Wen_i [C$'$ liebt_j [IP [NP seine_i Mutter ]%
      [VP t_i t_j ]]]]
```

(6) *[$_{\mathrm{CP}}$ Wen$_i$ [$_{\mathrm{C'}}$ liebt$_j$ [$_{\mathrm{IP}}$ [$_{\mathrm{NP}}$ seine$_i$ Mutter][$_{\mathrm{VP}}$ t$_i$ t$_j$]]]]

Note that above a blank space between `[IP` and `[NP` is crucial, otherwise the bracket of `[NP` would be subscripted as well. Note also that a space is required before the grammaticality judgment, otherwise the beginning of subscripting cannot be properly identified.

As with the gloss macro, grouping surpresses subscripting. Consider the effects of spacing and grouping in (7):

```
\exi.\a.  [[NP Fritz ][ snores ]]S
    \b.   [[NP Fritz][snores]]S
    \c.   [ [NP Fritz ][snores]]S
    \d.   [[NP Fritz {][}snores ]]S
    \e.  {[[+N,--V]} Fritz ][VP snores ] \hfill{} [NP Structure]
    \f. *[{}[+N,--V] Fritz ][VP snores ] \hfill [NP-Structure]
```

(7)
a. [[$_{\mathrm{NP}}$ Fritz][snores]]$_{\mathrm{S}}$
b. [[$_{\mathrm{NP}}$ Fritz][snores]]S
c. [[$_{\mathrm{NP}}$ Fritz][$_{\mathrm{snores]]S}}$
d. [[$_{\mathrm{NP}}$ Fritz][snores]]$_{\mathrm{S}}$
e. [[+N,–V] Fritz][$_{\mathrm{VP}}$ snores] [$_{\mathrm{NP}}$ Structure]
f. *[$_{\mathrm{[+N,-V]}}$ Fritz][$_{\mathrm{VP}}$ snores] [NP-Structure]

Grouping works the same way as in the gloss macro where it protects from glossing. E.g., a group such as `{][}` in (7-d) is ignored for subscripting so that the following material is protected from automatic subscripting (but note that something like the opposite happens in (7-e) after `\hfill`!).

Note that commands that look ahead and extend over more than one word might not work properly unless two or more words are put into a group. Suppose we are inside an `\exi.`-environment and we want to say something like `\b.[Principle C]`. What will happen? The space between "e" and "C" will make the subscripting device read `\b.[Principle` as a "word." This, unfortunately, prevents the argument of `\b.` from being interpreted correctly. Solutions are to write `\b.[Principle~C]`, `\b.[{Principle C}]`, or `{\b.[Principle C]}`.

Subscripting outside the `\exi.`-environment can be done by using `\I` as shown below:

`\I[NP Text` $\Longrightarrow$ $[_{\mathrm{NP}}$ Text,
`\I($\alpha$ Text` $\Longrightarrow$ $(_{\alpha}$ Text,
`\I{$\alpha$}NP Text` $\Longrightarrow$ α_{NP} Text,
`\I{Whom}ACC does John\ldots` $\Longrightarrow$ Whom$_{\mathrm{ACC}}$ does John...

For combining glosses with labelled brackets, say `\exig.` or `\exgi.`. Note that although the list-generating commands `\ex.` and `\a.` can be combined in any order, all commands that introduce automatic subscripting can be used only at the top level, i.e. cannot be embedded!!!. Note also that n nested `\ex.`-commands need n consecutive blank lines (`\par`s) to properly identify the end of all lists.

Handouts

Since the surface implementation of the list environment does not use the begin-end-format, style files that suppress printing between the end and the beginning of certain environments (e.g. `xcomment.sty`) could unfortunately not be adopted to produce a handout (I didn't see a way of doing so, probably because I didn't understand what goes on inside xcomment.sty).

This deficiency might be compensated for by using `linguho.sty`, which removes all examples and headings (of sections and subsections) from the LaTeX-document and stores them in a file whose extension is .han (namely *jobname*.han). Putting `\makehandout` somewhere at the end of the document (e.g. before the bibliography) will print out *jobname*.han at the position of `\makehandout`. If `\section*{...}` and `\subsection*{...}` should also go into the handout, say `\usepackage[*]{linguho}`. Use `\maketitle` to get the title into the handout.

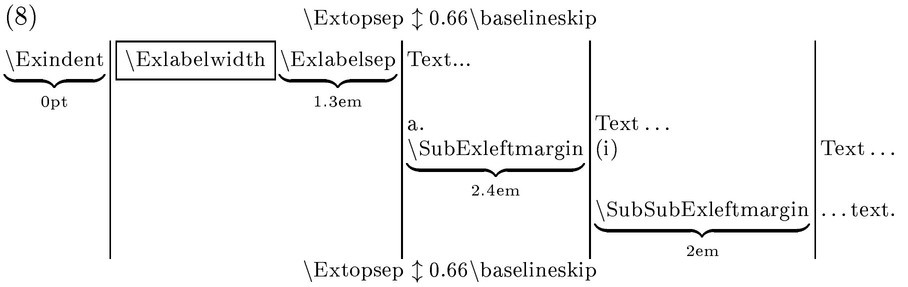
Note that all of LaTeX's `\setlength` and `\settowidth` instructions automatically go into *jobname*.han and are executed only there. (The main body of text before `\makehandout` is considered irrelevant, but I didn't find a way to suppress writing it into the dvi-file. Any help to solve this problem will be appreciated; note that I am not a TeX-wizzard, only a lazy linguist).

In case you have defined `\active` characters (e.g. Umlaut in German) these will most likely turn out troublesome (i.e. these commands should normaly be protected). For a general solution, inspect the definition of `\MkOthersSpecial` in linguho.sty. Uncommenting any of the lines there will solve the problem for the active character contained in the respective line (sometimes it might in addition be necessary to use t1enc.sty or germanb.sty instead of german.sty).

Customizing Lengths and Margins

The user may modify any of the lengths displayed in the following scheme, which also shows their predefined default values:

(8)



The value of `\Exlabelwidth` is determined by the width of the current label. Note that for displaying the table as shown in (8), the leftmargin of the list must be zero; accordingly, both `\Exlabelwidth` and `\Exlabelsep` were set to zero. To restore all default values of the lenghts shown in (8) simultaneously, say `\resetExdefaults`. The exact behavior of `\Exlabelwidth` is explained further below. With some non standard fonts the default spacing can be defined only `\AtBeginDocument` (for reasons I do not understand), therefore the default values for the above lengths are declared `\AtBeginDocument`. In consequence, any deviation from the default must be specified after `\begin{document}`

Some journals require a different arrangement of margins, e.g. one which aligns the left edges of examples with the left edges of subexamples, as shown in (9) and (10):

(9) First example:
  a. Subexample1
  b. Subexample2
    (i) Subsubexample1 ...

By saying `\alignSubExtrue` before `\begin{document}`, the sublabels (a. and b. in (9)) are given a negative indentation. This is controled by the lengths `\alignSubExnegindent` and `\Exlabelsep`, which can be customized by `\setlength`. after `\begin{document}`.

Customizing Labels

The labels of the examples are created by the counters `ExNo`, `SubExNo`, and `SubSubExNo` for the three levels of embedding respectively; `FnExNo` is used for the first level inside footnotes. These counters are the default labels of the list items. They can be changed anywhere in the document, by using `\setcounter` as usual.

There are a number of further possibilities to generate labels that differ from the default. One is to use `\a.` without embedding it into `\ex.`. This will yield the expected result, namely a list that uses the `SubEx` counter at the top-level:

a. First line
b. Second line
c. Third line

Moreover, any of the list commands can take an optional argument, whose specification replaces the default label:

Principle C (Chomsky[81]): An R-expression is free only

α) with respect to potential binders in A-positions,
β) within the domain of its chain.

```
\a.[{\bf Principle C} (Chomsky{[81]}): ]
      An R-expression is free only
      \a.[$\alpha$)] with respect to potential binders
             in A-positions,
      \b.[$\beta$)] within the domain of its chain.
```

The optional argument must immediately follow ".", so don't leave space between `\ex.` or `\a.` and "[" !!!. By contrast, it will often be the case that an example starts with a (labelled) bracket like $[_{\mathrm{NP}}$. If so, it is obligatory to put a space between "." and "[" .

As illustrated above, the topmost `\a.`-command picks up the `\Exlabelwidth` of the preceding example. By contrast, using `\ex.[{\it Principle C} ...]` instead of `\a.[{\it Principle C} ...]` has the effect of adjusting the indentation of the following text to the width of the optional argument, as shown below:

Principle C (Chomsky[81]): An R-expression is free only

α) with respect to potential binders in A-positions,
β) within the domain of its chain.

Assuming that regular example numbers do not exceed (999), this departure from the default behavior only occurs for optional arguments whose width exceeds that of (999).

Optional arguments can also be used to simulate other environments. E.g., LaTeX-style itemizing at the top level can be simulated as follows:

```
\ex.[\hfill$\bullet$\hfill] Line 1
\b. Line 2
\c. Line 3
```

- Line 1
- Line 2
- Line 3

Temporarily redefining `\alph`, as shown in the following example, yields an enumeration:

```
\let\oldalph=\alph\let\alph=\arabic
\a.  Text 1
\b.  Text 2
\c.  Text 3 \global\let\alph=\oldalph
```

1. Text 1
2. Text 2
3. Text 3

The stylesheet of *English Language and Linguistics* requires examples of the following form:

(10) [$_{\mathrm{v}}$ [$_{\mathrm{v}}$ broke$_i$ $\emptyset$][$_{\mathrm{VP}}$ the vase [$_{\mathrm{V}'}$ t$_i$ into pieces]]]

(11) (a) It kept warm
(b) She kept it warm

Apart form specifying `\alignSubExtrue` before `\begin{document}` this requires to redefine the brackets of sublabels, as shown below:

```
\renewcommand{\SubExLeftBracket}{(}
\renewcommand{\SubExRightBracket}{)}
\setlength{\Exlabelsep}{2em}
\exi.  [v [v broke_i $\emptyset$ ][VP the vase [V$'$ t_i into
    pieces~]]]

\ex.\a.  It kept warm
    \b.  She kept it warm
```

In addition, `\renewcommand{\refdash}{}` changes cross references from eg. (11-b) to (11b).

`\Exlabelwidth`

By default, the value of `\Exlabelwidth` is determined by the width of the example numbering. More precisely, for labels between (1) and (9) the labelwidth is by default that of (11); accordingly, the actual labelwidth is a bit larger than the natural size of the label. Similarly, `\Exlabelwidth` has the width of (110) for `\theExNo`'s between (10) and (99). Finally, `\Exlabelwidth` has the width of (1100) for example numbers between (100) and (999). For even wider labels (usually specified by an optional argument, cf. above), the label retains its natural size.

Since the space between the `\Exlabelwidth` and the text remains the same, the transition from (9) to (10) and from (99) to (100) will cause a small change of indentation, which might look ugly, particularly in handouts. In order to supress the default behavior of `\Exlabelwidth`, it must be assigned a particular length. For example, saying `\settowidth{\Exlabelwidth}{(110)}` will cause the labelwidth of all examles from (1) to (99) to be identical.

Since wide labels should still retain their natural size, the user's specification of `\Exlabelsep` should not be wider than (1100), otherwise the default mechanism is still active, and the value of `\Exlabelsep` is ignored. E.g., saying `\setlength{\Exlabelwidth}{3em}` will most likely cause the list declaration to take up its default behavior, because "3em" is wider than "(1100)". (The default value of `\Exlabelwidth` is 4em).

cgloss4e.sty

I found two problems with cgloss4e.sty. The first relates to AMS-TEX. Writing a translation immediated below the glossed material normally behaves as one would expect, but with AMS-Tex the translation becomes indented, for reasons I do not understand. To remidy this, one has to (re-)define the list indentation, e.g. as shown below (from LATEX):

```
\makeatletter
\def\@listi{\leftmargin\leftmargini
    \parsep 5\p@  \@plus2.5\p@ \@minus\p@
    \topsep 10\p@ \@plus4\p@   \@minus6\p@
    \itemsep5\p@  \@plus2.5\p@ \@minus\p@}
\let\@listI\@listi
\@listi
\makeatother
```

The second problem concerns the use of `\gll` at a position other than the beginning of a list. Then the gloss starts a new paragraph and puts unwanted vspace above and below the list. One way to suppress this is to change the flushleft commands used within cgloss4e.sty into raggedright by saying `\renewenvironment{flushleft}{\raggedright}{}`.