The `philex` Package
version 0.3

Peter Pagin

October 21, 2007, 01:09

The `philex` package is a small addition to Wolfgang Sternefeld's `linguex` package and requires `linguex`. As of version 0.2 it also requires `calc`. `philex` will load `linguex` if it is installed. The purpose of `philex` is to add functions for more flexible cross-referencing, for repeating and embedding named or numbered sentences and also a little bit for formatting.

As of version 0.3 `philex` creates a separate file in the same directory as the main file. If the typeset document is *myfile*.tex, the file created by `philex` will be *myfile*-px.tex. `philex` writes its command definitions to this file at typesetting, and reads from the created file at the beginning (AtBeginDocument) of the next round of typesetting. The purpose of this device is to allow forward cross-referencing to named sentences, in analogy to the label-reference system of LaTeX itself.

`philex` is called by `\usepackage{philex}`. Its basic command is `\lb{}{}`, which takes two obligatory arguments. The first argument is the label of the sentence, and the second is the sentence itself. `lb` uses the `\ex.` formatting of `linguex`, and thereby the `ExNo` counter. Type

```
\lb{nice}{This is a nice day}
```

and the result will be

(1) This is a nice day.

in case it is numbered sentence no 1 in your document (we will return in a second to the full stop issue).

In `linguex` you leave a blank line after sentence to close the environment. The blank line gives a `\par`-command to TeX. In `philex` the `\par`-command is built into the top-level environments. So you leave a blank line after an `\lb{}{}`-environment only if you want to start a new paragraph. The reason for this change, is that only this way will the `philex` package work together with the `extract` package by Hendri Adriaens, which can be used for extracting material, e.g. for generating a handout, from documents making use of `philex`. `linguex` comes together with Sternefeld's `linguho` package, which has the function of generating a handout from documents using `linguex`. `linguho` does not work

with `philex`, since the `linguex` environments are not explicit in the document. The corresponding function, only more flexible, can be served with the `extract` package.

For cross-referencing a numbered sentence you can use to ordinary `\ref{}`-command, or the command `\rf{}` provided by `philex`. Thus `\rf{nice}` produces the `ExNo` number of (1). The command takes an optional argument. In case you want to add a suffix in the cross-reference, you provide it as the optional argument. `\rf[a]{nice}` produces '(1a)'. This is normally not needed, since the usual reasons for doing this are accommodated by other means (see below).

A cross-reference with `\rn{}` will remove the brackets. `\rn{nice}` gives '1'. This is equivalent to using the label itself as a command name: `\nice` gives '1'.

You might want to repeat the sentence later. This can be done with the `\rp{}`-command. Thus `\rp{nice}` produces 'This is a nice day'. With the `\ml{}`-command you can repeat the sentence with the initial letter made lowercase: `\ml{nice}` produces 'this is a nice day'. This is useful in case you want to embed the sentence in a larger sentence, such as

(2) Elsa thinks that this is a nice day, but Alfred does not.

This also illustrates the insertion of the full stop. You do not normally type in a full stop a the end of the named sentence, but let it be put in by the `\lb`-command. This makes the call by means of the `\ml`-command fully embeddable. The full-stop insertion can be turned off with `\q` and turned on again with `\s`, and also transformed into a comma with `\km`. These commands shall go before the relevant `philex` environment. The analogous functions for the first subordinate level are served by `\qt` (changing to nothing), `\stp` (changing to full stop), and `\kmt` (changing to comma).

In case you want to repeat the sentence with its name or number you can use `\rff{}`. `\rff{nice}` produces

(1) This is a nice day.

In many cases you will want to produce a variant of the original sentence, which will have the same name or number as the original, but with an added suffix, like a letter or a prime symbol. This is achieved with the `\lbu{}{}{}{}`-command. It takes four obligatory arguments. The first is the label of the new sentence, the second is the label of the sentence you are giving a variant of, the third is the suffix you want to add, and the fourth is the new sentence itself.

```
\lbu{nicy}{nice}{\('\)}{But that was even nicer}
```

produces

(1′) But that was even nicer.

And a cross-reference will preserve the name: `\rf{nicy}` produces '(1′)'.

You will also want to have named principles, such as

(W) Life is wonderful!.

These are produced with the `\lbp{}{}{}`-command which makes use of the optional argument to `\ex.[]` in `linguex`. Cross-references with `\rf`, like to (W), and updates with `\lbu`, like

(W^+) Life is terrific!

work as before. If you later change the name of the principle, or the formulation, the changes are reflected in the cross-references and repetitions after the next typesetting, as long as you do not change the labels. In the last example, the full stop function was turned off by means of `\q` inside the last argument, so as not to produce an extra full stop after the exclamation mark.

In case you don't want the brackets in the principle name, turn them off with `\broff`, as in L (exemplifying forward cross-reference):

L Has a lot to offer!.

To turn them on again, type `\bron`. It is also possible to change the brackets with `\renewcommand{\lebrack}{}` and `\renewcommand{\ribrack}{}` (put in what you want), for the left and right bracket, respectively.

The cross-reference and repeat and update functions also work for subordinate clauses, corresponding to `\a.` and `\b.` of `linguex`. For these purposes, `philex` has `\lba{}{}`, `\lbb{}{}` and `\lbz{}{}`. To illustrate, the code

```
\lbp{clauses}{PP}{Some main words, followed by
\lba{first}{Time flies} \lbb{second}{Like an arrow}
\lbz{last}{And much too fast}}
```

produces

(PP) Some main words, followed by

a. Time flies.
b. Like an arrow.
c. And much too fast.

It is in general not necessary to to switch from `\lbb` to `\lbz` for the last subordinate clause.

Notice that the subordinate environments are put *into* the main `\lbp` environment. This is necessary, since the main environment has a `\par` command built in, which ends the `\ex.` environment of `linguex`. Also, there are no linebreaks in the code (even though that cannot be seen above), and that is again for the sake of the `extract` package, which extracts given command environments by command line. In these subordinate environments the full stop is put in. The closing one, `\lbz`, turns off the the full stop that may come from the the main environment, so as not to produce doubles.

We can now cross-reference a subordinate clause, as with `\rf{second}`, to produce '(PPb)'. We can also repeat, as with `\rff{last}`:

(PPc) And much too fast.

Similarly update.

There are some formatting options. The command `\subformat{}{}{}` takes three arguments. With the first argument you can control the numbering format. Default is lowercase alphabetic. Setting the first argument to 'A' gives uppercase alphabetic, to 'r' gives lowercase roman, to 'R' gives uppercase roman, and to '1' gives arabic. Other argument values have no effect. The second argument controls the prefix. For instance, inserting a left bracket will produce left-bracket enumeration. The last argument produces the suffix. So, `\sub{1}{}{)}` will produce arabic enumeration with a closing bracket after the numeral. The `\subformat`-command should be put in before the first `\lba`-command.

There are some shorter fast-switching commands for this purpose: `\sa` produces lowercase alphabetic with dot (which is `linguex` default), `\sab` produces lowercase alphabetic followed by right bracket, and `\sr` produces lowercase roman followed by right bracket. If these commands, just as the -verb+ ommand, are put within the main`philex` environment (before the first subordinate clause), the changes are only local to that environment. Note that the longer command works by *redefining* the `\alph`-command which normally gives lowercase alphabetic, while the fast switching commands redefine `\theSubExNO` without redefining `\alph`. So if `\alph` is locally redefined to mean `\roman`, the command `\sa` will produce a roman numeral with dot in that local context.

Repeating an `\lba` command after a subordinate clause produces a new subordinate level, as designed in `linguex`. We can have

(3) Much to do
  a) So little time.
    (i) And one is always late.

Clause (3ai) is worth thinking about. Produced with the following code:

```
\lb{numclauses}{Much to do \subformat{a}{}{)}
\lba{more}{So little time}
\lba{late}{And one is always late}}
Clause \rf{late} is worth thinking about.
```

Some formatting of the second subordinate level can be done by means of `\subsubformat{}{}{}`. At present, however, only the second and third argument, controlling prefix and suffix, are operative. The enumeration style is not affected.

In case you wish a list of independently numbered principles with a shared name stem, use one of the `\lbpx`-commands, where `x` is one of `a,b,c,d`. It takes two arguments, for label and content sentence. You first set the enumeration style and and the stem with `\bpxformat{}{}`, where again `x` has the corresponding value. The first argument sets the numbering style, as above, and the second the stem. Then the `\lbpx`-command picks it up:

```
\bpaformat{1}{T}
```

```
\lbpa{kno}{Alfred is in the know.}
\lbpa{hu}{Elsa is, too.}
\lbpa{kne}{John agrees.}
```

produces

(T1) Alfred is in the know.
(T2) Elsa is, too.
(T3) John agrees.

These enumerations are independent of `ExNO` and of each other, and the counters (`bpx`) are reset with the relevant `\bpxformat`. They still work with sub-clauses, cross-references and the rest of the apparatus. Note that these enumerations do not have to be contiguous:

(T4) But he shouldn't.

The top-level commands, `\lb`, `\lbp`, `\lbpx`, and `\lbu`, take an optional argument. If that argument is set to 'c', as in `\lb[c]{xx}{yy}`, the content sentence will be centered, as in

$$P(\gamma(t) \geq \delta)|\mathbf{A}) = (0.5)^{\alpha(t)} \sum_{i=0}^{\epsilon(t)} \binom{\alpha(t)}{i} \tag{4}$$

.

This is designed for equations and figures. It works both with `\(\)` and, as here, with `\[\]`, which produces more space above and below.

Positioning text (left, center or right) on lower lines in a `philex`-environment can be done by means of line-break and a `\makebox[][]{}`-command on the new line. The first optional argument sets the width of the box. With the command `\hcentro`, provided by `philex`, the length is set to `\textwidth` (the current width of the entire text column) minus the `linguex` inset (width of the label + with of the separation between label and text). The length command `\centro` deducts that quantity twice. This second alternative is used for centering with respect to the entire text column (exemplifed in (4) above), while the `\hcentro` command is used for left-, center- or right positioning within the interior text column of `philex/linguex`. The positioning is chosen by the LaTeXvalues `l`, `c` and `r` in the second optional argument.

Finally, `philex` provides an additional counter, `bna`, for informal enumerations in the main text. The command `\bn` advances the counter and puts in a number with a quad space. The command `\bns` resets the counter to zero.

1. The result is exemplified with first with this sentence.
2. And then it is followed up here.

There are more possibilities in `linguex`, described in its manual.