

The expex-acro package*

Florian Matter
florianmatter@gmail.com

April 28, 2022

Contents

1	Introduction	1
2	Referring to examples	1
3	Glossing abbreviations	2
4	Other commands useful for linguistic documents	2

1 Introduction

expex-acro, as the name suggests, combines `expex` (for typesetting linguistic examples) with `acro` (for acronyms). The two main functionalities are commands to refer to examples, and to handle glossing abbreviations.

2 Referring to examples

<code>\exref</code>	Refer to examples, e.g. <code>\exref{kwaza-3}</code> . Use <code>\exref[a]{b}</code> to get (Xa-b). 1 <code>\providecommand{\exref}[2][]{% 2 \ifthenelse{\equal{#1}{}}{% 3 {(\getfullref{#2})}% 4 {(\getfullref{#1}--\getref{#2})}% 5 }</code>
<code>\exrefnil</code>	Refer to examples without explicit numbers. 6 <code>\providecommand{\exrefnil}[2][]{% 7 \ifthenelse{\equal{#1}{}}{(\getref{#2})}{(\getref{#1}--\getref{#2})}% 8 }</code>
<code>\mexref</code>	For multiple, non-adjacent examples. 9 <code>\providecommand{\mexref}[2][,]{% 10 (% 11 \def\nextitem{\def\nextitem{#1}}% Separator 12 \renewcommand*{\do}[1]{\nextitem\getfullref{##1}}% How to process each item</code>

*This document corresponds to expex-acro v0.0.1, dated 2022/04/28.

```

13 \docsvlist{#2}% Process list
14 )%
15 }

```

3 Glossing abbreviations

`\gl` Glossing abbreviations (pre-defined or custom), which will occur in the list. For example, `\gl{erg}` yields ERG.

```

16 \providecommand{\gl}[1]{\acs{#1}}

```

`\newGlossingAbbrev` Define a new glossing abbreviation: `\newGlossingAbbrev{occ}{occultive}`.

```

17 \providecommand{\newGlossingAbbrev}[2]{
18   \DeclareAcronym{#1}{
19     short=#1,
20     long=#2,
21     short-format=\scshape,
22   }
23 }

```

`\glossingAbbrevsList` Print the list of glossing abbreviations.

```

24 \newcommand{\glossingAbbrevsList}{
25   \printacronyms[
26     template=glossinglist,
27     name=Glossing abbreviations,
28     heading=none
29   ]
30 }

```

4 Other commands useful for linguistic documents

The big advantage of using something like `\obj` rather than `\textit` is that you can change how object language is displayed at any time, instead of hardcoding italics.

`\obj` Object language.

```

31 \providecommand{\obj}[1]{\textit{#1}}

```

`\qu` Translations.

```

32 \providecommand{\qu}[1]{‘#1’}

```

`\rc` Reconstructed forms.

```

33 \providecommand{\rc}[1]{*\textit{#1}}

```

`\ort` Orthographic forms.

```

34 \providecommand{\ort}[1]{\langle$#1$\rangle$}

```

`\pnt` Phonetic brackets.

```

35 \providecommand{\pnt}[1]{[#1]}

```

`\pnm` Phonemic slashes.
36 `\providecommand{\pnm}[1]{/#1/}`

`\dbqu` Double quotation marks.
37 `\providecommand{\dbqu}[1]{\#1"}`

`\ungr` Grammatically incorrect forms.
38 `\providecommand{\ungr}[1]{*\textit{#1}}`

`\bad` Grammatically questionable forms.
39 `\providecommand{\bad}[1]{?\textit{#1}}`

`\lxm` Lexemes.
40 `\providecommand{\lxm}[1]{\textsc{#1}}`

`\glosstilde` Nice-looking tildes for reduplication.
41 `\providecommand{\glosstilde}{\char'\sim\kern-1ex}`