The covington Package
Macros for Linguistics

Michael A. Covington Jürgen Spitzmüller*

Version 2.3, June 21, 2019

Abstract

This package, initially a collection of Michael A. Covington's private macros, provides numerous minor LaTeX enhancements for linguistics, including multiple diacritics on the same letter, interlinear glosses (word-by-word translations), Discourse Representation Structures, example numbering, and semantic markup.

Contents

*Current maintainer. Please report issues via `https://github.com/jspitz/covington`

1 Introduction

This is the documentation for version 2.3 of covington (June 21, 2019), which is a LaTeX package providing macros for typing some special notations common in linguistics.[1]

To use covington with LaTeX 2ε, load the package as usual by adding the command `\usepackage{covington}` to your document preamble. The package has the following options:

force: Force the redefinition of environments that have already been defined by other packages or the class.

This applies to the **example**, **examples**, **subexamples** and **exercise** environments, which are by default not touched if they are already defined before covington is loaded. See sec. 3.2, 3.3, 3.4 and 8 for details.

keeplayout: Do not tweak the layout.

Covington sets `\raggedbottom` and redefines the value of the `\textfloatsep` length. This just follows the preferences of the original package author and is not necessary for the package's functionality. Yet for backwards compatibility reasons, we cannot change this. Thus, we provide the option described here to opt out this presetting.

noglossbreaks: If this option is set, covington will try hard to prevent page breaks within glosses.

If this option is not set, page breaks can occur between interlinearized text and free translation of a gloss, as well as between gloss preamble and interlinearized text, which is usually not what you will want. Nonetheless the option is not set by default. This is for backwards compatibility reasons (in order to not change page breaking of existing documents). Note that page breaks might still occur in some cases even if the option is set. In order to prevent them definitely, you can put the gloss in a parbox or minipage.

owncounter: Use an own counter for numbered examples.

By default, covington uses LaTeX's equation counter for example numbering, so that if you use equations and numbered examples in the same paper, you get a single continuous series of numbers. While some people (including the original author of this package) consider this a feature, others might prefer to number equations and linguistic examples separately. If you count to the latter sort, use this option.

Please note the following package-related caveats:

- If you are using covington and the uga (University of Georgia thesis style) package together, you should load uga before covington.

[1] The package has a long history. It started off as a collection of private macros back in the LaTeX 2.09 days and was initially released as `covingtn.sty` (following the old 8.3 FAT file name limit). In emTeX under MS-DOS, the file was distributed as `covingto.sty`. Eventually, it has been renamed to covington and adapted to LaTeX 2ε.

- If you are using covington with beamer-article, you should load beamer-article before covington.
- If you are using covington with the drs package, you should load drs before covington. See sec. 7.

In what follows we presume that you know how to use LaTeX and have access to LaTeX manuals.

2 Stacked diacritics

LaTeX provides a generous range of diacritics that can be placed on or below any letter, such as:

x̀ x́ x̂ ẍ x̃ x̄ x̋ x̌x x̧ x̣ ẋ x̲

which are typed, respectively, as:

```
\‘{x} \’{x} \^{x} \"{x} \~{x} \={x} \H{x} \t{xx} \c{x} \d{x} \b{x}
```

Out of the box, however, LaTeX doesn't give you a convenient way to put *two* diacritical marks on the same letter. To fill this gap, covington provides the following macros:

`\twodias{<upper diac.>}{<lower diac.>}{<char>}`
to combine any two diacritics, e. g., `\twodias{\~}{\=}{a}` = ã̄

`\acm{...}` for acute over macron, e. g., `\acm{a}` = ā́
`\grm{...}` for grave over macron, e. g., `\grm{a}` = ā̀
`\cim{...}` for circumflex over macron, e. g., `\cim{a}` = ā̂

The first of these is the general case[2] and the latter three are special cases that are often used in Greek transcription. Now you can type *Koinḗ* with both accents in place.

The vertical distance between the two diacritics can be adjusted via the macro **`\SetDiaOffset{<length>}`** which lets you increase or decrease the vertical space that is currently in effect. If you'd use `\SetDiaOffset{-0.25ex}`, the above examples would come out as

`\twodias{<upper diac.>}{<lower diac.>}{<char>}`
to combine any two diacritics, e. g., `\twodias{\~}{\=}{a}` = ã̄

`\acm{...}` for acute over macron, e. g., `\acm{a}` = ā́
`\grm{...}` for grave over macron, e. g., `\grm{a}` = ā̀
`\cim{...}` for circumflex over macron, e. g., `\cim{a}` = ā̂

with a slightly better matching distance for the font used here.

Note that not all accent macros work in the `tabbing` environment. Use the `Tabbing` package or refer to [6] for alternative solutions.

[2] Alternatively, there's also the old syntax **`\twoacc[<upper diac.>|<char with lower diacr.>],`** e. g. `\twoacc[\~|\={a}]` to the same effect, which is however discouraged due to its rather odd form.

3 Numbered examples

Linguistic papers often include numbered examples. With covington, generating those is straightforward. In this section, we describe how you can typeset a self-stepping example number (see section 3.1), a single numbered example (sec. 3.2), a consecutive range of numbered examples (sec. 3.3), and alpha-numerically labeled sub-examples (sec. 3.4). All numbered examples can be referred to in the text via `\label` and `\ref` as usual (see sec. 3.6 for details).

3.1 Example numbers

The macro **`\exampleno`** generates a new example number, stepped by 1. It can be used anywhere you want the number to appear. For example, to display a sentence with a number at the extreme right, do this:

```
\begin{flushleft}
This is a sentence. \hfill (\exampleno)
\end{flushleft}
```

Here's what you get:

This is a sentence. (1)

If you want to output the (current) number without stepping it, the starred form **`\exampleno*`** will do that.

Normally, however, you do not need to manually place **`\exampleno`** yourselves, as in the example above. For the common case where example numbers in parentheses are placed left to the example, covington provides more convenient solutions. These are described in turn.

3.2 The `example` environment

The **`example`** environment (alias **`covexample`**) displays a single example with a generated example number to the left of it. If you type

```
\begin{example}
This is a sentence.
\end{example}
```

or

```
\begin{covexample}
This is a sentence.
\end{covexample}
```

you get:

(2) This is a sentence.

The example can be of any length; it can consist of many lines (separated by `\\`), or even whole paragraphs.

If you need more space between the example number and the text, you can increase it by means of the length **`\examplenumbersep`** (which is preset to `0pt`). Doing `\setlength\examplenumbersep{1em}`, for instance, will increase the space by 1 em (negative values will decrease the space accordingly).

Note that, as of version 1.1, covington checks if there is already an **`example`** environment defined (e. g., by the class). If so, covington does not define its own one. However, there is always the alias environment **`covexample`** which can be used in order to produce `covington's` example. If you use the package option **`force`**, covington will override existing **`example`** environments. In any case, the package will issue a warning if **`example`** is already defined (this is the case, for instance, if you use covington with the `beamer` class).

One way to number sub-examples is to use `itemize` or `enumerate` within an example, like this:

```
\begin{example}
\begin{itemize}
\item[(a)] This is the first sentence.
\item[(b)] This is the second sentence.
\end{itemize}
\end{example}
```

This prints as:

(3) (a) This is the first sentence.
(b) This is the second sentence.

However, the **`subexamples`** environment, described in sec. 3.4, is usually more convenient for this task.

3.3 The `examples` environment

To display a series of examples together, each with its own example number, use **`examples`** (or **`covexamples`**) instead of **`example`** or **`covexample`**. The only difference is that there can be more than one example, and each of them has to be introduced by `\item`, like this:

```
\begin{examples}
\item This is the first sentence.
\item This is the second sentence.
\end{examples}
```

or, respectively:

```
\begin{covexamples}
\item This is the first sentence.
\item This is the second sentence.
\end{covexamples}
```

This prints as:

(4) This is the first sentence.

(5) This is the second sentence.

As for **example**, covington checks if there is already an **examples** environment defined, and if this is the case, covington does not define its own one. The alias environment **covexamples** is always available as a fallback. If you use the package option **force**, covington will override existing **examples** environments. The package will issue a warning if **examples** is already defined (this is the case, for instance, if you use covington with the `beamer` class), telling you how it has dealt with the situation.

3.4 The `subexamples` environment

Sometimes a set of (paradigmatic) sub-examples gets only one main example number with alphabetic sub-numbering, as in (6 a). To achieve this most conveniently, covington provides the **subexamples** (or **covsubexamples**) environment. The difference to **examples**/**covexamples** is the numbering:

```
\begin{subexamples}
\item This is the first sentence.
\item This is the second sentence.
\end{subexamples}
```

or, respectively:

```
\begin{covsubexamples}
\item This is the first sentence.
\item This is the second sentence.
\end{covsubexamples}
```

prints as:

(6) (a) This is the first sentence.

(b) This is the second sentence.

Again, covington checks if there is already an **subexamples** environment defined, and if this is the case, covington does not define its own one. The alias environment **covsubexamples** is always available as a fallback. If you use the package option **force**, covington will override existing **subexamples** environments. The package will issue a warning if **subexamples** is already defined.

The **subexamples** environment provides the following option:

preamble={⟨arbitrary text⟩} Arbitrary text that is inserted in the first line (after the main number and before the first sub-example, which then follows in a new line). This might be useful, for instance, to give context information, to specify the language or the source in case of cited sub-examples. You can globally set the markup of this preamble text (see sec. 3.5).

For instance,

```
\begin{subexamples}[preamble={Here are two sentences}]
\item This is the first sentence.
\item This is the second sentence.
\end{subexamples}
```

or, respectively:

```
\begin{covsubexamples}[preamble={Here are two sentences}]
\item This is the first sentence.
\item This is the second sentence.
\end{covsubexamples}
```

prints as:

(7) Here are two sentences

  (a) This is the first sentence.

  (b) This is the second sentence.

The distance between example number and subnumber (letter) can be changed via the length **\examplenumbersep** (which is preset to `0pt`). The distance between example subnumber and text can be changed via the length **\subexamplenumbersep** (preset to `0pt` as well). In both cases, a positive value will increase, a negative value will decrease the respective distance. Doing

```
\setlength{\examplenumbersep}{-0.5em}
\setlength{\subexamplenumbersep}{0.5em}
```

for instance, will come out like this:

(8) (a) This is the first sentence.

  (b) This is the second sentence.

3.5 Customizing number display and example markup

You can change the display of the example number by redefining (via `\renewcommand*`) the macro **\covexnumber**, which has the following default definition:

```
\newcommand*\covexnumber[1]{(#1)}
```

with the variable `#1` representing the number.

In the same vein, you can customize the display of the subexample letter by redefining the macro **\covsubexnumber**, which has the following default definition:

```
\newcommand*\covsubexnumber[1]{(#1)}
```

To remove the parentheses from the subexample letter, for instance, to this:

```
\renewcommand*\covsubexnumber[1]{#1}
```

You can also customize the markup of the example sentences by redefining the macro **\covexamplefs** (which is empty by default). To have all example sentences italicized, for instance, do:

```
\renewcommand*{\covexamplefs}{\itshape}
```

Note that this does, deliberately, not include the numbers, since those are controlled by another font setting macro, **\covexamplenofs**, which defaults to `\normalfont`. Of course you are free to redefine this as well, if you wish do to so.

Finally, you can customize the markup of the sub-example preamble text by redefining the macro **\subexpreamblefs**, which also defaults to `\normalfont`. To have it italicized, analogously do:

```
\renewcommand*{\subexpreamblefs}{\itshape}
```

3.6 Referring to examples

References to examples and sub-examples can be made the usual way via the **\ref** command (which refers to a **\label** that is placed in the respective example paragraph). The references do not have parentheses by default, i. e., a reference to the example in section 3.2 would be printed as 2, a reference to the sub-example in section 3.4 as 6 a. For convenience, though, covington provides a command **\pxref** that also prints the parentheses, as in (2) and (6 a). It is defined as follows and can be redefined if needed:

```
\providecommand*\pxref[1]{(\ref{#1})}
```

4 Glossing sentences word-by-word

To gloss a sentence is to annotate it word-by-word. Most commonly, a sentence in a foreign language is followed by a word-for-word translation (with the words lined up vertically) and then a smooth translation (not lined up), like this:

Dit	*is*	*een*	*Nederlands*	*voorbeeld*
This	is	a	Dutch	example

'This is an example in Dutch.'

Covington provides different ways to typeset such glosses. The most convenient way is via gloss macros (see sec. 4.1), an alternative (and the traditional) way is via a set of commands (see sec. 4.2). Both are described in turn.

4.1 Gloss macros

Covington provides two gloss macros:

- **\digloss[**⟨**options**⟩**]{**⟨**gloss line 1**⟩**}{**⟨**gloss line 2**⟩**}{**⟨**free translation**⟩**}** typesets two-line glosses with a translation line (the macro name puns on Greek *diglossía*, lit. 'two tongues', 'bilingualism')

- **\trigloss[⟨options⟩]{⟨gloss l. 1⟩}{⟨gloss l. 2⟩}{⟨gloss l. 3⟩}{⟨free tr.⟩}** typesets three-line[3] glosses with a translation line (cf. Greek *triglossía*, lit. 'three tongues', 'trilingualism')

The example given above would thus be typed as:

```
\digloss{Dit is een Nederlands voorbeeld}
        {This is a Dutch example}
        {This is an example in Dutch.}
```

Note that:

- The ⟨free translation⟩ argument can be left empty. In this case, no translation line is added (and no extra space taken).
- The macros automatically markup the lines. By default, the first gloss line is in italics, subsequent lines are set upright, and the free translation in single quotation marks (using the language-sensitive csquotes [5] macros if this package is loaded). This can be customized, though, via the macro options or globally (for the latter, see sec. 4.3).
- By default, page breaks might occur within glosses. In order to prevent it, the option **noglossbreaks** (see sec. 1) will help in many cases. If it doesn't, you can wrap the whole gloss into a minipage or parbox.
- The words do not have to be typed lining up; TEX counts and aligns them.
- On the other hand, multiple blanks are ignored, so you can, but do not need to, use the space key to line up the words to your liking in the TEX file. The example above could thus also have been input like this, with no change to the output:

```
\digloss{Dit  is een Nederlands voorbeeld}
        {This is a   Dutch      example}
        {This is an example in Dutch.}
```

- If the words in the two languages do not correspond one-to-one, you can use curly brackets to group words and a pair of empty curly brackets to mark null forms. For example, to print

Dit	*is*	*een*	*voorbeeldje*	*in*	*het*	*Nederlands*
This	is	a	little example	in		Dutch

  'This is a little example in Dutch.'

  you would type:

```
\digloss{Dit  is een voorbeeldje      in het Nederlands}
        {This is a   {little example} in {}  Dutch}
        {This is a little example in Dutch.}
```

[3] The additional line can be useful for instance to gloss cited forms, morphology, or an additional translation. See sec. 4.4 for examples.

The following ⟨**options**⟩ (key-value pairs) are provided for the two gloss macro:[4]

ex=⟨**true|false**⟩ Default: *false*. Wraps the gloss in an example environment (i. e., it is numberered).

tlr=⟨**true|false**⟩ Default: *false*. If set to true, the translation line (content of the ⟨free translation⟩ argument) is set right to the gloss lines, rather than into a new line below. Since the gloss itself is set in a box, this means the ⟨free translation⟩ will appear lined up with the first line of the gloss. This can be useful when no translation, but an aligned number or something similar, is to be inserted right to the gloss (please refer to sec. 4.4 for an example).

fsi={⟨**font settings**⟩**}** Adjusts the font settings of the first gloss line. Valid values are LaTeX font switches such as `\itshape`, `\bfseries` etc.

fsii={⟨**font settings**⟩**}** Adjusts the font settings of the second gloss line. Valid values are LaTeX font switches such as `\itshape`, `\bfseries` etc.

fsiii={⟨**font settings**⟩**}** Adjusts the font settings of the third gloss line. Valid values are LaTeX font switches such as `\itshape`, `\bfseries` etc.

preamble={⟨**arbitrary text**⟩**}** Arbitrary text that is inserted on an own line before the interlinearized gloss. This might be useful, for instance, to give context information, to specify the language or the source in case of cited glosses.

The advantages over just adding a line manually above the gloss are that you can globally set the markup (see sec. 4.3), and that such lines are kept on the same page than the gloss with the option **noglossbreaks** (at least as long as preamble does not exceed one line). Furthermore, this option is the only way to add such text when the **ex** option is used.

If given as optional arguments to a **\digloss** or **\trigloss** macro, the options will only apply to this very gloss. If you want to make a permanent change, you can use the macro

- **\setglossoptions{**⟨**options**⟩**}**

and pass either of the above options to it. This will apply to all subsequent glosses (until further global change and unless the setting is altered locally via macro option).

4.2 Glossing with low-level commands

The gloss macros described above build on low-level commands[5] which can also be used directly (this was actually the only way up to covington 2.0 which introduced the macros). Low-level commands can be useful if you want to do fancy things in a gloss. Note, though, that using them also has limitations: Some commands cannot be used inside macros (such as footnotes), and the markup is not done automatically in

[4] Please consult sec. 4.4 below for examples that showcase these options.

[5] The commands are adapted with permission from `gloss.tex`, by Marcel R. van der Goot.

all cases (as documented in what follows); furthermore, you cannot make use of the options the macros have. Thus, we strongly suggest to use the macros, unless you have very good reasons not to do that.

The following is a complete list of all low-level glossing commands:

\gll introduces two lines of words vertically aligned, and activates an environment very similar to `flushleft`. The two lines are separated by a normal line break (carriage return). This is possible since the command makes the line-ending character active. As a consequence, however, this command does not work inside macros (such as `\footnote`).

\glll is like **\gll** except that it introduces *three* lines of lined-up words.

\xgll is similar to **\gll** except that it does not make the line ending active. It thus works inside macros such as footnotes but requires explicit gloss line termination via **\xgle**.

\xglll is similar to **\glll** except that it does not make the line ending active. It thus works inside macros such as footnotes but requires explicit gloss line termination via **\xgle**.

\xgle is a gloss line ending marker to be used with **\xgll** and **\xglll**.

\glt ends the set of lined-up lines and introduces a line (or more) of translation. Note that this command does not markup the translation line (no automatic enquoting). Also, it outputs an empty line if no text follows.

\gln is like **\glt** but does not start a new line (useful when no translation follows but you want to put a number on the right).

\glot{⟨**free translation**⟩**}** is an alternative to **\glt** and a smarter way to insert a translation line. Other than **\glt**, it marks up (by default: enquotes) the translation line. Also, it does not add an empty line if the translation is empty. This command has been introduced in covington 2.0.

\glosspreamble{⟨**arbitrary text**⟩**}** lets you enter text that is printed immediately before the interlinearized gloss (on a line of its own). This might be useful, for instance, to give context information, to specify the language or the source in case of cited glosses. The advantages over just adding a line manually above the gloss are that you can globally set the markup (see sec. 4.3) and that such lines are kept on the same page than the gloss with the option **noglossbreaks**. Note, however, that page breaks might occur if this text spans multiple lines. In this case, you can wrap the whole gloss into a minipage. This command has been introduced in covington 2.1.

\glend ends the special `flushleft`-like environment.

Using the low-level commands, the examples given in the previous section would be coded as follows:

```
\gll Dit is een Nederlands voorbeeld.
     This is a Dutch example.
\glt ‘This is an example in Dutch.’
\glend
```

```
\gll Dit is een voorbeeldje     in het Nederlands
     This is a {little example} in {}  Dutch
\glt ‘This is a little example in Dutch.’
\glend
```

Observe that you need to markup (i. e., enquote) the translation line yourself if you use **\glt**. This is not so if you use **\glot**:

```
\gll Dit is een Nederlands voorbeeld
     This is a Dutch example
\glot{This is an example in Dutch.}
\glend
```

The advantage of **\glot** is that you can easily customize the translation markup globally. Also, **\glot** uses the language-sensitive csquotes [5] macros if this package is loaded (see sec. 4.3).

Since **\gll** and **\glll** locally activate the end of line in glosses in order to identify the different lines of the gloss (via category code change), they do not work inside macros (e. g., if the gloss is in a footnote). To work around this, special versions of the **\gll** and **\glll** commands are provided that do without the character activation: **\xgll** and **\xglll**, respectively. These can also be used in macro arguments; however, the end of each gloss line needs to be explicitly specified by the **\xgle** command in this case. If you want to put the above gloss in a footnote, thus, you would type:

```
\xgll Dit is een voorbeeldje     in het Nederlands.\xgle
      This is a {little example} in {}  Dutch.\xgle
\glt ‘This is a little example in Dutch.’
\glend
```

Note, again, that the macros described in sec. 4.1 do not have this problem.

To sum up: With low-level commands, every glossed sentence begins with either **\gll**, **\xgll**, **\glll** or **\xglll**, then contains either **\glt** or **\gln**, and ends with **\glend**. Layout is critical in the part preceding **\glt** or **\gln**, and fairly free afterward.

4.3 Customization

The font settings of each gloss line can be customized globally by way of the global options macro **\setglossoptions** and the **fsi**, **fsii** or **fsiii** key, respectively (see sec. 4.1). Alternatively, you can also redefine these macros:

```
\newcommand*\glosslineone{\normalfont\itshape}% font settings 1st gloss line
\newcommand*\glosslinetwo{\normalfont\upshape}% font settings 2nd gloss line
\newcommand*\glosslinethree{\normalfont\upshape}% font settings 3rd gloss line
```

The markup of the translation line (if the **\digloss** or **\trigloss** macro or the **\glot** low-level command is used) can be customized by redefining the following macro.

```
\newcommand*\glosslinetrans[1]{\covenquote{#1}}
```

Note that for **\covenquote**, as used in the default definition, covington checks at document begin whether the csquotes [5] package is loaded. If so, it uses its language-sensitive **\enquote*** macro for enquoting the translation. If not, a fallback quotation (using English single quotation marks) is used. The usage of csquotes is highly recommended!

The markup of the preamble line (which is not marked up at all by default) can be customized by redefining the macro:

```
\newcommand*\glosslinepreamble[1]{#1}
```

4.4 Examples

This section gives some further examples. First, a sentence with three lines aligned, instead of just two:

Hoc	*est*	*aliud*	*exemplum*
N.SG.NOM	3SG	N.SG.NOM	N.SG.NOM
This	is	another	example

'This is another example.'

In order to produce this, we use the **\trigloss** macro for a three-line gloss and pass the **fsii** option with the respective font switches in order to get small capitals in the second line:

```
\trigloss[fsii={\normalfont\scshape}]
        {Hoc est aliud exemplum}
        {n.sg.nom 3sg n.sg.nom n.sg.nom}
        {This is another example}
        {This is another example.}
```

Next, an example with a gloss but no translation, with an example number at the right:

Hoc	*habet*	*numerum*	(9)
This	has	number	

That one was typed using the option **tlr**:

```
\digloss[tlr]{Hoc habet numerum}
            {This has number}
            {\hfill (\exampleno)}
```

Third, we'll put a glossed sentence inside the `example` environment, which is a very common way of using it:

(10) | *Hoc* | *habet* | *numerum* | *praepositum* |
|---|---|---|---|
| This | has | number | preposed |

'This one has a number in front of it.'

This last example was, of course, typed as:

```
\digloss[ex]{Hoc habet numerum praepositum}
           {This has number preposed}
           {This one has a number in front of it.}
```

although you could also construct it manually as:

```
\begin{example}
\digloss{Hoc habet numerum praepositum}
       {This has number preposed}
       {This one has a number in front of it.}
\end{example}
```

And finally, an example that uses the *Leipzig glossing rules* ([1], cited example: p. 2) and also exemplifies the use of **`preamble`**:

(11) Lezgian (Haspelmath 1993:207)
Gila *abur-u-n* *ferma* *hamišaluǧ* *güǧüna* *amuq'-da-č.*
now they-OBL-GEN farm forever behind stay-FUT-NEG
'Now their farm will not stay behind forever.'

This has been input as follows:

```
\digloss[ex,preamble={Lezgian (Haspelmath 1993:207)}]
       {Gila abur-u-n             ferma hamišalu\v{g} gü\v{g}üna amuq'-da-\v{c}.}
       {now  they-\textsc{obl-gen} farm  forever       behind     stay-\textsc{fut-neg}}
       {Now their farm will not stay behind forever.}
```

Of course, you would use `\cite` in a real document for the citation. Also, if you adhere to the *Leipzig glossing rules*, you might want to check out the leipzig LaTeX package [7] that facilitates the use of the gloss abbreviations that have been entered and marked-up manually here.

5 Phrase structure rules

To print phrase structure rules such as $S \rightarrow NP\ VP$ you can use `covington's` macro **`\psr{`**⟨**`constituent`**⟩**`}{`**⟨**`sub-constituents`**⟩**`}` (for the given example, `\psr{S}{NP~VP}`).

6 Feature structures

To print a feature structure such as

$$\begin{bmatrix} case : nom \\ person : P \end{bmatrix}$$

you can type:

```
\fs{case:nom \\ person:P}
```

The feature structure can appear anywhere – in continuous text, in a displayed environment such as `flushleft`, or inside a phrase-structure rule, or even inside another feature structure.

To put a category label at the top of the feature structure, like this,

$$\begin{array}{c} N \\ \left[\begin{array}{l} case : nom \\ person : P \end{array}\right] \end{array}$$

here's what you type:

```
\lfs{N}{case:nom \\ person:P}
```

And here is an example of a PS-rule made of labeled feature structures:

$$\begin{array}{c} S \\ \left[\, tense : T \,\right] \end{array} \quad \rightarrow \quad \begin{array}{c} NP \\ \left[\begin{array}{l} case : nom \\ number : N \end{array}\right] \end{array} \quad \begin{array}{c} VP \\ \left[\begin{array}{l} tense : T \\ number : N \end{array}\right] \end{array}$$

which was obviously coded as:

```
\psr{\lfs{S}{tense:T}}
    {\lfs{NP}{case:nom \\  number:N}
     \lfs{VP}{tense:T \\ number:N} }
```

7 Discourse Representation Structures

Several macros in covington facilitate display of *Discourse Representation Structures* (DRSes) in the box notation introduced by Hans Kamp [4]. The simplest one is **\drs**, which takes two arguments: a list of discourse variables joined by ~, and a list of DRS conditions separated by \\. Nesting is permitted.

Note that the **\drs** macro itself does not give you a displayed environment; you must use `flushleft` or the like to display the DRS.

Here are some examples:

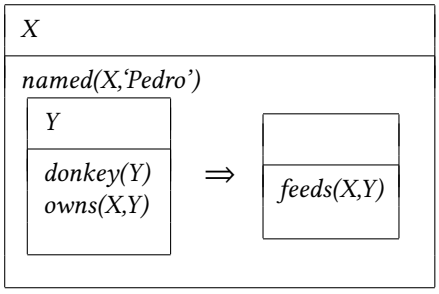
```
\begin{flushleft}
  \drs{X}
      {
       donkey(X)\\
       green(X)
      }
\end{flushleft}
```

prints as:

X
donkey(X) *green(X)*

```
\begin{flushleft}
  \drs{X}
      {
       named(X,‘Pedro’)\\
       \drs{Y}
           {
            donkey(Y)\\
            owns(X,Y)
           }
       ~~{\large $\Rightarrow$}~
       \drs{~}
           {feeds(X,Y)}
      }
\end{flushleft}
```

comes out as:



Note that the alignment of the input is fairly free, so you can also write the two arguments of **\drs** in one line, like:

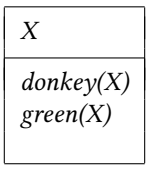
```
\drs{X}{donkey(X)\\green(X)}
```

To display a sentence above the DRS, use **\sdrs**, which has one extra argument for this purpose, as in:

```
\begin{flushleft}
  \sdrs{A donkey is green.}
       {X}
       {donkey(X)\\green(X)}
\end{flushleft}
```

which prints as:

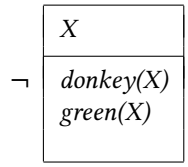
A donkey is green.



Some DRS connectives are also provided (normally for forming DRSes that are to be nested within other DRSes). The macro **\negdrs** forms a DRS preceded by a negation symbol, so

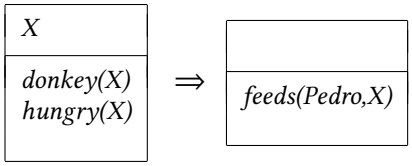
```
\negdrs{X}
       {
        donkey(X)\\
        green(X)
       }
```

comes out as

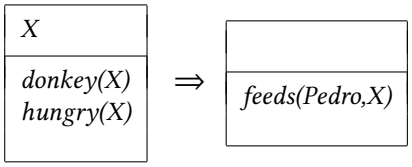


Finally, **\ifdrs** forms a pair of DRSes joined by a big arrow, like this:

```
\ifdrs{X}
      {
       donkey(X)\\
       hungry(X)
      }
      {~}
      {feeds(Pedro,X)}
```



If you have an "if"-structure appearing among ordinary predicates inside a DRS, you may prefer to use **\alifdrs**, which is just like **\ifdrs** but shifted slightly to the left for better alignment:



Note that for more extended DRS representations, dedicated packages are meanwhile available, most notably the drs [2] and the sdrt [3] package. Both packages actually draw on covington, add some additional features and, in some cases, tweak the layout to (what strikes those package authors) the better. If the rather basic DRS macros provided by covington do not suit you, please check if one of those packages does.

Note, though, that while sdrt introduces new (capitalized) macro naming which lets the package peacefully coexist with covington, drs simply re-uses covington's macro names, which makes the two packages incompatible. In order to fix this, covington checks whether the DRS macros are already defined when it is loaded; if so, it does not define its own ones. So if you want to use the DRS macros of the drs package together with covington's non-DRS features, you can do so, provided that drs is loaded *before* covington. In that case, covington's own DRS macros are disabled.

8 Exercises

The **exercise** environment (alias **covexercise**) generates an exercise numbered according to chapter, section, and subsection (suitable for use in a large book; in this example, the subsection number is going to come out as 0). Here is an example:

Exercise 8.0.1 (Project) *Prove that the above assertion is true.*

This was coded as

```
\begin{exercise}[Project]
Prove that the above assertion is true.
\end{exercise}
```

The argument (`[Project]` in the example) is optional.

Note that, as of version 1.1, covington checks if there is already an **exercise** environment defined (e. g., by the class). If so, covington does not define its own one. However, there is always the alias environment **covexercise** which can be used in order to produce `covington's` exercise. If you use the package option **force**, covington will override existing **exercise** environments. In any case, the package will issue a warning if **exercise** is already defined.

9 Reference Lists

To type a simple LSA-style hanging-indented reference list, you can use the **reflist** environment. (*Note:* **reflist** is not integrated with BibTEX in any way.[6]) For example,

```
\begin{reflist}
Barton, G. Edward; Berwick, Robert C.; and Ristad, Eric Sven.  1987.
Computational complexity and natural language.  Cambridge,
Massachusetts: MIT Press.

Chomsky, Noam.  1965.  Aspects of the theory of syntax.  Cambridge,
```

[6] For BibTEX, there are several options: the LSA style, as used in the journal *Language*, can be obtained by means of the style files `lsalike.bst` (`http://www.icsi.berkeley.edu/ftp/pub/speech/jurafsky/lsalike.bst`) or `language.bst` (`http://ron.artstein.org/resources/language.bst`); the latter uses `natbib`. The so-called *Unified Style Sheet for Linguistics*, as proposed by the CELXJ (*Committee of Editors of Linguistics Journals*), which slightly differs from the LSA style, is followed by the style file `unified.bst` (available at `http://celxj.org/downloads/unified.bst`). A `biblatex` style file for the unified style is available at `https://github.com/semprag/biblatex-sp-unified` or on CTAN as part of the univie-ling bundle.

```
Massachusetts: MIT Press.

Covington, Michael.  1993.  Natural language processing for Prolog
programmers.  Englewood Cliffs, New Jersey: Prentice-Hall.
\end{reflist}
```

prints as:

Barton, G. Edward; Berwick, Robert C.; and Ristad, Eric Sven. 1987. Computational complexity and natural language. Cambridge, Massachusetts: MIT Press.

Chomsky, Noam. 1965. Aspects of the theory of syntax. Cambridge, Massachusetts: MIT Press.

Covington, Michael A. 1993. Natural language processing for Prolog programmers. Englewood Cliffs, New Jersey: Prentice-Hall.

By default, the references have a hanging indentation of 3 em. This can be globally changed by altering the length **\reflistindent**. Doing `\setlength\reflistindent{1.5em}`, for instance, will shorten the indentation by half. Likewise, the length **\reflistitemsep** (6 pt by default) and **\reflistparsep** (ca. 4 pt by default) can be adjusted to alter the vertical separation (`\itemsep` and `\parsep`, for that matter) of reference entries.

Notice that within the reference list, "French spacing" is in effect — that is, spaces after periods are no wider than normal spaces. Thus you do not have to do anything special to avoid excessive space after people's initials.

10 Semantic markup

The macro **\sentence** displays an italicized sentence (it is a combination of `flushleft` and `itshape`). If you type

```
\sentence{This is a sentence.}
```

you get:

This is a sentence.

The font shape can be modified by redefining the following macro:

```
\newcommand*\sentencefs{\itshape}
```

The following macros provide further markup options common in linguistics:

- **\lexp**{word} is used to mark word forms (italic by default, as in *word*)
- **\lcon**{concept} is used to mark concepts (small caps by default, as in CONCEPT)
- **\lmean**{meaning} is used to mark meaning (single quotes by default, as in 'meaning')

Note that for **\lmean**, covington checks at document begin whether the csquotes [5] package is loaded. If so, it uses its language-sensitive **\enquote*** macro for quoting. If not, a fallback quotation (using English single quotation marks) is used. The usage of csquotes is highly recommended!

Here are the definitions of the macros. They can be redefined via **\renewcommand**:

```
\providecommand*\lexp[1]{\textit{#1}}
\providecommand*\lcon[1]{\textsc{#1}}
\providecommand*\lmean[1]{\covenquote{#1}}
```

11 Big curly brackets (disjunctions)

Last of all, the two-argument macro **\either** expresses alternatives within a sentence or PS-rule:

`the \either{big}{large} dog` = the $\left\{\begin{array}{c}\text{big}\\ \text{large}\end{array}\right\}$ dog

`\psr{A}{B~\either{C}{D}~E}` $= A \rightarrow B \left\{\begin{array}{c}C\\ D\end{array}\right\} E$

That's all there is for now. Suggestions for improving covington are welcome, and bug reports are actively solicited (via `https://github.com/jspitz/covington`). Please note, however, that this is free software, and the authors make no commitment to do any further work on it.

12 Release history

2.4 (forthcoming)

- Fix definition of covexercise theorem when no subsection counter is defined.

2.3 (2019 June 21)

- Add preamble option to subexamples environment. See sec. 3.4.
- Allow to use covington together with the drs package.
- Documentation fixes and restructuring.

2.2 (2019 June 4)

- Add new option **owncounter** that makes covington use an own counter for examples (rather than the equation counter).
- Add starred **\exampleno*** command that outputs the current example number value without stepping it. See sec. 3.1.

- Add macros **\covexamplefs** and **\covexamplenofs** for global setting of example text markup. See sec. 3.5.

2.1 (2019 May 12)

- Add new option **noglossbreaks** that tries to prevent page breaks within glosses.
- Add **\glosspreamble** command and **preamble** gloss macro option for arbitrary text preceding glosses.

2.0 (2018 December 10)

- Add new gloss macros (**\digloss** and **\trigloss**) for a more convenient, flexible and robust gloss insertion. See sec. 4.1.
- Add **\glot** command for customizable gloss translation line, together with customization possibilities. See sec. 4.2.
- Add possibility to customize **\sentence** font setting. See sec. 10.
- Add **\lexp**, **\lcon** and **\lmean** markup macros. See sec. 10.

1.8 (2018 December 7)

- Fix font markup of second gloss line (do not force rm).
- Add possibility to customize gloss line font setting. See sec. 3.5.
- Add possibility to customize example number display. See sec. 3.5.

1.7 (2018 September 8)

- Fix alignment in **subexamples**.
- Improve manual.

1.6 (2018 September 7)

- Introduce new environment **subexamples** (see sec. 3.4).
- Introduce new command **\pxref** (see sec. 3.6).

1.5 (2018 August 24)

- Introduce new option **keeplayout** which allows to opt-out the layout presettings covington does (`\raggedbottom`, `\textfloatsep`).

1.4 (2017 May 23)

- Introduce a new macro **\twodias** that supersedes the rather odd **\twoacc** (which is kept for backwards compatibility). See sec. 2 for details.
- Introduce macro **\SetDiaOffset** for more convenient setting of vertical distance in stacked diacritics. See sec. 2 for details.
- LaTeX 2.09 is no longer officially supported (it might continue to work, but is not tested).

1.3 (2017 April 5)

- Gloss variants **\xgll** and **\xglll** that work inside macros (such as footnotes) but require explicit gloss line end markers (**\xgle**). See sec. 4 for details.
- New lengths **\reflistindent**, **\reflistparsep** and **\reflistitemsep** to globally adjust the indentation or vertical space, respectively, of reflist items. See sec. 9 for details.

1.2 (2016 August 26)

- New length **\examplenumbersep** to adjust (increase) the horizontal space between example number and example text. See sec. 3.2 for details.
- Add some more info about bibliography generation.

1.1a (2016 July 7)

- Fix encoding problem in documentation and some typos. No change in functionality.

1.1 (2016 July 6)

- The package now uses NFSS font commands if available (fallback for LaTeX 2.09 is still provided).
- Work around clash with classes/packages that define their own **example** and **examples** environments (most notably the `beamer` class) as well as **execise** environments. The covington package no longer blindly attempts to define these environments. By default, it does not define them if they are already defined (`covington's` own environments, however, are still available via aliases). By means of a new package option, a redefinition can also be forced. See sec. 3.2 and 3.3 for details.
- New length **\twoaccsep** allows for the adjustment of the distance between stacked accents (see sec. 2).
- Update manual.

- New maintainer: J. Spitzmüller.
- License has been changed to LPPL (in agreement with M. Covington)
- Introduce version numbers. Arbitrarily, we start with 1.1.

2014 May 16

- Patches by Robin Fairbairns:
  - Setting of `\textfloatsep` uses `\setlength` rather than `\renewcommand`
  - Style file converted to un*x line endings

2001 March 27

- It is no longer necessary to type `\it` to get proper italic type in feature structures.
- Instructions have been rewritten with LaTeX 2_ε users in mind.

Older versions

- Multiple accents on a single letter (e. g., $\acute{\tilde{a}}$) are supported.
- This package is now called covington (with the o) and is compatible with LaTeX 2_ε and NFSS as well as LaTeX 2.09.
- The vertical placement of labeled feature structures has been changed so that the category labels line up regardless of the size of the structures.

References

[1] Bickel, Balthasar, Bernard Comrie, and Martin Haspelmath: *The Leipzig glossing rules: Conventions for interlinear morpheme by morpheme glosses*. Revised version of February 2008. Department of Linguistics, Max Plank Institute for Evolutionary Anthropology. `http://www.eva.mpg.de/lingua/resources/glossing-rules.php`.

[2] Dimitriadis, Alexis: *drs – Typeset Discourse Representation Structures (DRS)*. June 10, 2010. `https://ctan.org/pkg/drs`.

[3] Isambert, Paul: *sdrt – Macros for Segmented Discourse Representation Theory*. May 13, 2007. `https://ctan.org/pkg/sdrt`.

[4] Kamp, Hans: A Theory of Truth and Semantic Representation. In Jeroen A. G. Groenendijk, Theo M. V. Janssen, and Martin J. B. Stokhof (eds.): Formal Methods in the Study of Language. Amsterdam: Mathematics Center, 1981, 277–322.

[5] Lehman, Philipp and Joseph Wright: *csquotes: Context sensitive quotation facilities*. April 4, 2018. `https://www.ctan.org/pkg/csquotes`.

[6] Pakin, Scott: The Comprehensive LaTeX Symbol List. November 30, 2015. https://www.ctan.org/pkg/comprehensive.

[7] Weber, Nathalie: *leipzig: Typeset and index linguistic gloss abbreviations*. June 16, 2017. https://ctan.org/pkg/leipzig.