

To use this CCG style, please include the following line somewhere in the \LaTeX preamble:

```
\usepackage{cgc-latex}
```

The style file `cgc-latex.sty` neither loads nor requires packages or fonts. Kerning controls for typesetting the CCG slashes, the LF's colon, and rule decorations in CCG derivations are in the pt unit, without the `\!` command. This is because the eye-pleasing slash distance seems to be independent of font size, at least to my eyes.

If you feel like changing it, just look for the `\kern` command.

All math is introduced by `\ensuremath` in `cgc-latex.sty`; there is no `$. . $` unless you introduce it manually. I don't recommend using math without `\ensuremath` in `cgc-latex.sty`. The short form `\mymath{. .}`, for 'bare math', is for that.

I demonstrate examples with increasingly high-level `cgc-latex` code, from lowest level `\cgex` to `\begin{cgc}` and `\begin{cgg}`.

An Example with the `\cgex{n}{derivations}` command. The `\lf{}` is already in math mode. Lexical assumption lines are drawn by one command.

$$\begin{array}{c}
 \text{John} \qquad \text{likes} \qquad \text{Mary} \\
 \hline
 \text{S}/(\text{S}\backslash\text{NP}) \quad (\text{S}\backslash\text{NP}_{3s})/\text{NP} \quad (\text{S}\backslash\text{NP})\backslash((\text{S}\backslash\text{NP})/\text{NP}) \\
 : \lambda p.p \text{john}' : \lambda x \lambda y. \text{like}' xy \qquad : \lambda p.p \text{mary}' \\
 \hline
 \text{S}\backslash\text{NP} : \lambda y. \text{like}' \text{mary}' y \\
 \hline
 \text{S} : \text{like}' \text{mary}' \text{john}'
 \end{array}$$

The `cgc-latex` code is:

```

\cgex{3}{John & likes & Mary}\
\cglines{3}\
\cgf{S\fs(S\bs NP)} & \cgf{(S\bs\cgs{NP}{3s})\fs NP}
& \cgf{(S\bs NP)\bs((S\bs NP)\fs NP)}\
\lf{\lambda p.p\,\so{john}}
& \lf{\lambda x \lambda y.\so{like}xy} & \lf{\lambda p.p\,\so{mary}}\
&\cgline{2}{\cgba}\
&\cgres{2}{S\bs NP \lf{\lambda y.\so{like}\so{mary}y}}\
\cgline{3}{\cgfa}\
\cgres{3}{S \lf{\so{like}\so{mary}\so{john}}}
}

```

You will notice that the result of ‘likes Mary’ was typeset in a different font. That’s because `\cgres` command typesets its input in default font. Use the `\cat` command inside it to avoid that (or `\cgf`):

$$\begin{array}{ccc}
 \text{John} & \text{likes} & \text{Mary} \\
 \hline
 S/(S\backslash NP) & (S\backslash NP_{3s})/NP & (S\backslash NP)\backslash((S\backslash NP)/NP) \\
 : \lambda p.p\text{john}' & : \lambda x\lambda y.\text{like}'xy & : \lambda p.p\text{mary}' \\
 \hline
 & S\backslash NP:\lambda y.\text{like}'mary'y & \\
 \hline
 S:\text{like}'mary'\text{john}' & &
 \end{array}$$

The `cgg-latex` code is:

```

\cgex{3}{John & likes & Mary\\
\cglines{3}\\
\cgf{S\fs(S\bs NP)} & \cgf{(S\bs\cgs{NP}{3s})\fs NP}
& \cgf{(S\bs NP)\bs((S\bs NP)\fs NP)}\\
\lf{\lambda p.p\,\,\so{john}}
& \lf{\lambda x\lambda y.\so{like}xy} & \lf{\lambda p.p\,\,\so{mary}}\\
& \cglines{2}{\cgba}\\
& \cgres{2}{\cat{S\bs NP}\lf{\lambda y.\so{like}\so{mary}y}}\\
% note that \cgres is by default in \cgf font
\cglines{3}{\cgfa}\\
\cgres{3}{\cat{S}\lf{\so{like}\so{mary}\so{john}}}
}

```

Same example with lower-level ccg-latex commands for lexical assumption lines, and with shorthanded type-raising notation using subscript and superscript. NB. they are typeset in math mode without needing $\$$. $\$$.

$$\begin{array}{ccc}
 \text{John} & \text{likes} & \text{Mary} \\
 \hline
 \text{NP}_{3s}^{\uparrow} & (\text{S} \backslash \text{NP}_{3s}) / \text{NP} & \text{S} \backslash (\text{S} / \text{NP}) \\
 \text{:}\lambda p.p\text{john'} & \text{:}\lambda x\lambda y.\text{like'}xy & \text{:}\lambda p.p\text{mary'} \\
 \hline
 \text{S/NP:}\lambda x.\text{like'}x\text{john'} & \xrightarrow{\text{B}} & \\
 \hline
 \text{S:like'mary'john'} & \xrightarrow{\hspace{1.5cm}} &
 \end{array}$$

The ccg-latex code is:

```

\cgex{3}{John & likes & Mary\\
% uses the alias \cat rather than \cgf above--same result
\cgul & \cgul & \cgul\\
% manually repeats the columns for comparison with \cglines above
\cat{\cgss{NP}{3s}{\uparrow}}
& \cat{(S\bs\cgs{NP}{3s})\fs NP} & \cat{S\bs(S\fs NP)}\\
\lf{\lambda p.p\,\,\so{john}}
& \lf{\lambda x\lambda y.\so{like}xy} & \lf{\lambda p.p\,\,\so{mary}}\\
\cgline{2}{\cgfc}\\
\cgres{2}{\cat{S\fs NP}\lf{\lambda x.\so{like}x\so{john}}}\
% note that \cgres is by default in \cgf font
\cgline{3}{\cgfa}\\
\cgres{3}{\cat{S} \lf{\so{like}\so{mary}\so{john}}}\
% using \cat inside \cgres is nae problem
}

```

Although the top line produced by \cgex command seems like it is meant for lexical assumptions only, because of its font, you can override that for example using the \cat command:

$$\begin{array}{c}
 X/Y \quad Y \\
 \hline
 X \xrightarrow{\hspace{1cm}}
 \end{array}$$

Here is the ccg-latex code for it:

```

\cgex{2}{\cat{X\fs Y} & ~~~\cat{Y}}\\
\cgline{2}{\cgfa}\\
\cgres{2}{\cat{X}}
}

```

An example with double slashes (for morphology, etc.)

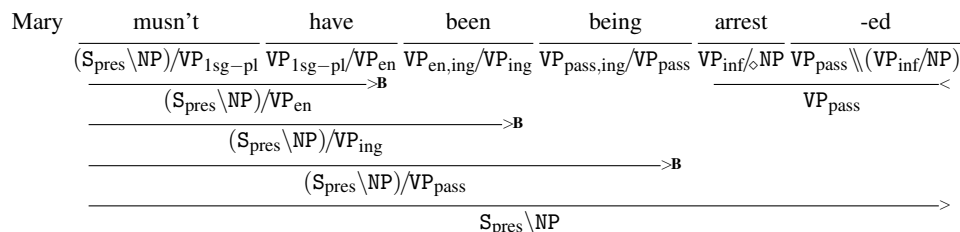
$$\frac{\text{dismiss} \qquad \text{-ed}}{\frac{\text{VP}_{\text{inf}}/\text{NP} : \lambda x \lambda y. \text{dismiss}' xy \ ((S \backslash \text{NP}_{\text{agr}})/\text{NP}) \backslash \backslash (\text{VP}_{\text{inf}}/\text{NP}) : \lambda p \lambda x \lambda y. \text{past}'(Pxy)}{(S \backslash \text{NP}_{\text{agr}})/\text{NP} : \lambda x \lambda y. \text{past}'(\text{dismiss}' xy)} <$$

The ccc-latex code is below (using `\cgf` instead of `\cat`, which do the same, and native latex math for LF, which does the same as `\lf{. .}`). NB. empty subscripts with no effect, if you keep changing the categories as I do).

The last `\cgres` is a painful reminder that if you put inline math in it using `$.. $`, you will get a warning from L^AT_EX. Use `\lf{. .}` or `\mymath{. .}` instead.

```
\cgex{3}{dismiss& -ed\\
\cglines{2}\\
\cgf{\cgs{VP}{inf}\fs\cgs{NP}{}}
$: \lambda x \lambda y. \so{dismiss}\,x\,y$&
\cgf{((\cgs{S}{})\bs\cgs{NP}{agr})\fs NP)\bss(\cgs{VP}{inf}\fs NP)}
$: \lambda p \lambda x \lambda y. \so{past}(P\,xy)$\\
\cgline{2}{\cgba}\\
\cgres{2}{\cgf{(\cgs{S}{})\bs\cgs{NP}{agr})\fs\cgs{NP}{}}
\lf{\lambda x \lambda y. \so{past}(\so{dismiss}\,x\,y)}
}
```

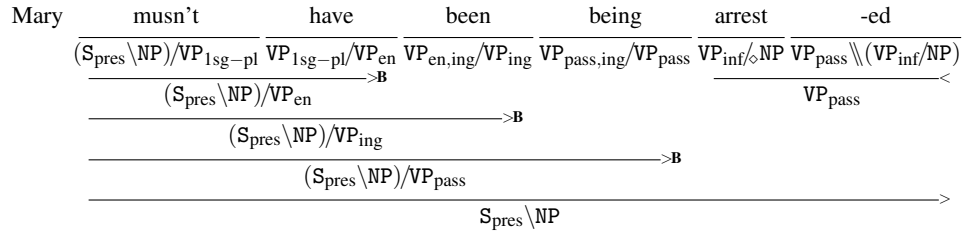
Here is one example with features galore, from Emmon Bach:



Here is the ccg-latex code:

```
{\footnotesize
Mary \cgex{6}{musn't & have & been & being & arrest & -ed\\
\cglines{6}{\\
\cgf{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{1sg-pl}}
& \cgf{\cgs{VP}{1sg-pl}\fs\cgs{VP}{en}}
& \cgf{\cgs{VP}{en,ing}\fs\cgs{VP}{ing}}
& \cgf{\cgs{VP}{pass,ing}\fs\cgs{VP}{pass}}
& \cgf{\cgs{VP}{inf}\fds NP}
& \cgf{\cgs{VP}{pass}\lds(\cgs{VP}{inf}\fs NP)}\\
\cglines{2}{\cgfc} &&&\cglines{2}{\cgba}\\
\cgres{2}{\cat{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{en}}}}
&&&\cgres{2}{\cgs{VP}{pass}}\\
\cglines{3}{\cgfc}\\
\cgres{3}{\cat{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{ing}}}}\\
\cglines{4}{\cgfc}\\
\cgres{4}{\cat{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{pass}}}}\\
\cglines{6}{\cgfa}\\
\cgres{6}{\cat{\cgs{S}{pres}\bs NP}}}
}}
```

Same example with `\begin{ccg}{n}{data}{derivations}\end{ccg}`.
No gloss line, and lexical assumption lines are drawn by default.



Here is the ccg-latex code:

```
{\footnotesize
Mary
\begin{ccg}{6}{musn't & have & been & being & arrest & -ed}
  {\cggf{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{1sg-pl}}
    & \cggf{\cgs{VP}{1sg-pl}\fs\cgs{VP}{en}}
    & \cggf{\cgs{VP}{en,ing}\fs\cgs{VP}{ing}}
    & \cggf{\cgs{VP}{pass,ing}\fs\cgs{VP}{pass}}
    & \cggf{\cgs{VP}{inf}\fds NP}
    & \cggf{\cgs{VP}{pass}\lds(\cgs{VP}{inf}\fs NP)}\\
    \cgline{2}{\cgfc} &&& \cgline{2}{\cgba}\\
\cgres{2}{\cat{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{en}}}
&&& \cgres{2}{\cgs{VP}{pass}}\\
    \cgline{3}{\cgfc}\\
\cgres{3}{\cat{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{ing}}}\backslash
\cgline{4}{\cgfc}\\
\cgres{4}{\cat{(\cgs{S}{pres}\bs NP)\fs\cgs{VP}{pass}}}\backslash
\cgline{6}{\cgfa}\\
\cgres{6}{\cat{\cgs{S}{pres}\bs NP}}
}
\end{ccg}
}
```

Another example, to show glossing in the beginning. The end gloss is typeset by `\mc`, for multi-column, centered.

It uses `\begin{ccgg}{n}{data}{gloss}{derivations}\end{ccgg}`.

ver-dir give-caus	-t -caus	-ti. -past
$\text{VP}_{\text{inf}} \backslash \text{NP}_{\text{dat}} \backslash \text{NP}_{\text{dat}} \backslash \text{NP}_{\text{acc}}$ $:\lambda x \lambda y \lambda z. \text{give}'_{yxz}$	$(S \backslash \text{NP}_{\text{nom}} \backslash \text{NP}_{\text{case}}) \backslash \text{VP}_{\text{inf}}$ $:\lambda p \lambda x \lambda y. \text{cause}'(px)y$	
$\text{S} \backslash \text{NP}_{\text{nom}} \backslash \text{NP}_{\text{dat}} \backslash \text{NP}_{\text{dat}} \backslash \text{NP}_{\text{dat}} \backslash \text{NP}_{\text{acc}}$ $:\lambda x_1 \lambda x_2 \lambda x_3 \lambda x_4 \lambda x_5. \text{cause}_{1,2}'(\text{cause}'_{1,2}(\text{give}'_{x_1 x_2 x_3})x_4)x_5$ ‘made to let give’, from Turkish		

Here is the ccg-latex code:

```
\begin{ccgg}{3}{ver-dir & -t & -ti.}{give{-caus} & {-caus} & {-past}}
{
\cgf{\cgs{VP}{inf}\bs\cgs{NP}{dat}\bs\cgs{NP}{dat}\bs\cgs{NP}{acc}}
& \cgf{(S\bs\cgs{NP}{nom}\bs\cgs{NP}{case})\bs\cgs{VP}{inf}}\\
\lf{\lambda x \lambda y \lambda z. \so{give}_{yxz}}
& \lf{\lambda p \lambda x \lambda y. \so{cause}(px)y}\\
\cglines{2}{\cgb{3}}\\
\cgres{2}{\cat{S\bs\cgs{NP}{nom}\bs\cgs{NP}{dat}\bs
\cgs{NP}{dat}\bs\cgs{NP}{dat}\bs\cgs{NP}{acc}}}\
\cgres{2}{\lf{\lambda x_1 \lambda x_2 \lambda x_3 \lambda x_4 \lambda x_5.
\so{cause}_{1,2}}
(\so{cause}_{1,2}(\so{give}_{x_1 x_2 x_3})x_4)x_5}}\[1ex]
\mc{3}{‘made to let give’, from Turkish}
}
\end{ccgg}
```

Note that subscripted semantic objects such as $\text{cause}_{1,2}'$ are better typeset as $\text{cause}'_{1,2}$ using `\sos` (for ‘semantic object, subscripted’), rather than `\so`.

An example with slash modalities and the non-derivability indicator (*):

$$\frac{*I \text{ will} \quad \text{give} \quad \text{flowers} \quad \text{my very heavy friends.}}{\frac{(VP/NP)/_{\diamond}NP \quad VP \backslash (VP/NP)}{* < \mathbf{B}_x} \quad VP \backslash (VP/NP)}$$

Code:

```
\begin{ccg}{4}{*I~will& give& flowers& my~very~heavy~friends.}
{
&\cat{(VP\fs NP)\fds NP}&\cat{VP\bs(VP\fs NP)}&\cat{VP\bs(VP\fs NP)}\\
&\badline{2}{\cgbx}
}
\end{ccg}
```


Some examples to show slash distancing by pt unit for categories in subscript:

If you use `ccg-latex` for category name spacing you will get:

John [should]_{(S\NP)//IV} walk the dog.

John [wants]_{(S\NP)/VP} to walk the dog.

Mary [believes]_{(S\NP)/S} that John walked the dog.

And some big ones by `ccg-latex`:

$(S\backslash_{\times}NP)/_{\diamond}IV$

$(S\backslash\backslash NP)/VP$

$(S\backslash NP)//S$

and bigger ones:

$(S\backslash_{\times}NP)/_{\diamond}IV$

$(S\backslash\backslash NP)/VP$

$(S\backslash NP)//S$

Here is the native \LaTeX math rendering of above without `ccg-latex` to compare spacing in various sizes:

$(S\backslash_{\times}NP)/_{\diamond}IV$

$(S\backslash\backslash NP)/VP$

$(S\backslash NP)//S$

John [should]_{(S\NP)//IV} walk the dog.

John [wants]_{(S\NP)/VP} to walk the dog.

Mary [believes]_{(S\NP)/S} that John walked the dog.

Check the code:

```
{\Large
$(S\backslash\_times NP)/\_{{\diamond}IV}$\medskip

$(S\backslash\backslash NP)/VP$\medskip

$(S\backslash NP)// S$}\bigskip

John [should]$_{\scriptstyle (S\backslash NP)//IV}$ walk the dog.

John [wants]$_{\scriptstyle (S\backslash\backslash NP)/ VP}$ to walk the dog.

Mary [believes]$_{\scriptstyle (S\backslash NP)/\_{{\diamond}S}}$ that John walked the dog.
```

It generates.

$$(S \backslash \times NP) / \diamond IV$$

$$(S \backslash \backslash NP) / VP$$

$$(S \backslash NP) // S$$

John [should]_{(S \ NP) // IV} walk the dog.

John [wants]_{(S \ \ NP) / VP} to walk the dog.

Mary [believes]_{(S \ NP) / \diamond S} that John walked the dog.

Not pretty.