Drs.sty, a LaTeX package for Discourse Representation Structures

Alexis Dimitriadis

Version 1.1

June 10, 2010

This package draws Discourse Representation Structures (DRSs). It can draw embedded DRSs, if-then conditions and quantificational "duplex conditions" (with a properly scaled connecting diamond). Formatting parameters allow the user to control the appearance and placement of DRSs, and of DRS variables and conditions.

You should use this package together with *pict2e.sty,* which allows better diagonal lines to be drawn. It is not automatically loaded by *drs.sty,* but will be utilized if it you do load it.

1 History

This package is derived from the DRS macros in the style *covington.sty,* written by Michael A. Covington. I am grateful to him for allowing modification and redistribution of his code. The present style retains the basic design (command syntax, use of the `tabular` environment to construct the DRSs) and is intended to be backward compatible. It adds commands for duplex conditions and logical connectives, formatting options, better spacing, and an absurdly elaborate algorithm for calculating the size of the diamond used in duplex conditions.

Please direct suggestions, bug reports, or other comments to Alexis Dimitriadis, *a.dimitriadis @uu.nl.* As usual, I cannot promise this package will do what you want it to, but I will try to help if there is a problem.

2 Box-building commands

These commands can be embedded in each other as desired.

`\drs`{***variables***}{***conditions***}

Typesets a DRS in a box according to the formatting options in effect (see below). Conditions should be separated by `\\`. If the variables argument is completely empty (given as `{}`), no space is provided for variables at the top of the box. To leave an empty line, give a space as the list of variables (`{ }` or `{~}`).

```
\drs{x y}{Jones(x) \\ Ulysses(y) \\ x owns y}
```

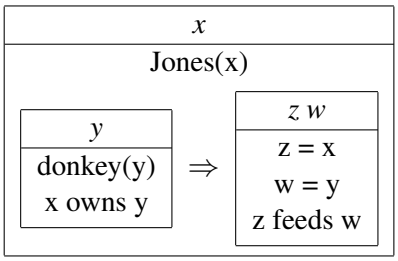
x y
Jones(x)
Ulysses(y)
x owns y

\drs*{*variables*}{*conditions*}

Top-level DRSs are positioned on the page according to a number of formatting parameters (see next section). To generate a DRS without the extra spacing, define it using `\drs*` rather than `\drs`.

\ifdrs{*lvar*}{*lcond*}{*rvar*}{*rcond*} Forms two DRSs joined by an arrow. This command is just a special case of `\condrs`.

```
\drs{x}{Jones(x) \\
  \ifdrs{y}{donkey(y)\\x owns y}
        {z w}{z = x\\ w = y\\ z feeds w}}
```



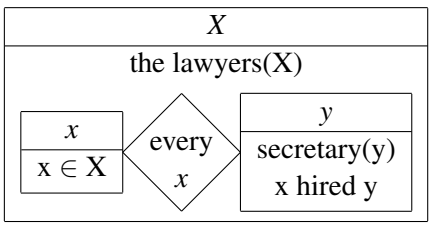
\condrs{*lvar*}{*lcond*}{*connective*}{*rvar*}{*rcond*}

Connects two DRSs with the middle argument, presumably a logical connective like `$\vee$` or `$\wedge$`. (You must manually switch to math mode if it's needed). This is provided as a separate command because the spacing needs certain adjustments. The connective is always put in size `\large`, which is appropriate for normal-sized text. If the text of your DRS is in a different size, manually resize the connective to be one step larger; for example, `{\LARGE $\vee$}` if the DRS text is `\Large`.

\qdrs{*lvar*}{*lcond*}{*quantifier*}{*qvar*}{*rvar*}{*rcond*}

Forms a duplex condition. The arguments *quantifier* and *qvar* are placed in the diamond connecting the left and right DRSs, which is sized to fit them. The *quantifier* argument may contain newlines (`\\`), if necessary.

```
\drs{X}{ the lawyers(X) \\
    \qdrs{x}{x $\in$ X}
         {every}{x}
         {y}{secretary(y) \\ x hired y}}
```



\negdrs{***variables***}{***conditions***}

Produces a DRS preceded by the negation symbol.

\drsdiamond{***quantifier***}{***variable***}

Produces the automatically scaling diamond used by `\qdrs`. Perhaps it is useful to call it directly for specialized uses. An absurdly fancy scaling algorithm draws a diamond of the right size and places the text at the right height. The algorithm is optimized for cases where the variable is narrower than the quantifier text, but works well for a variety of shapes, including multi-line quantifier names (use `\\` to break lines, as in: `at least\\one`). If it does the wrong thing for you, try padding the text with non-breaking spaces (`~`).

For compatibility with the *covington.sty* macros, the following commands are also supported:

\sdrs{***sentence***}{***variables***}{***conditions***}

Just like `\drs`, but the first argument is a sentence to be displayed above the DRS.

\alifdrs{***lvar***}{***lcond***}{***rvar***}{***rcond***}

Just like `\ifdrs`, but shifted to the left so that the conditions in it, if left-adjusted, will line up with unembedded DRS conditions.

3 Formatting parameters

The following commands may be redefined to control the appearance of the DRS boxes.

\drscondfont The font used for the DRS conditions. Default: `\rm`. If necessary, it can be redefined to a macro that takes an argument, e.g.:

```
\renewcommand\drscondfont[1]{\emph{#1}}
```

The font properties selected with this macro (e.g., italics) will also apply to the DRS variables, except as overridden by `\drsvarfont`.

\drsvarfont Additional font settings used for the DRS variables. This macro modifies the font style already in effect from `\drscondfont`. Default: `\it`. If necessary, it can be redefined to a macro that takes an argument.

\drsseparator The separator between variables and conditions. By default it is `\hline`, but can be redefined to suppress the line (or to draw other line styles, if appropriate). To suppress the line, put the following in the preamble of your document:

```
\renewcommand\drsseparator{\relax}
```

\drsalignment Alignment of the contents of the DRS, in the form of a {tabular} alignment parameter. The default is `c` (centered) but can be changed to `l` (or even `r` if you wish).

\drsboxalignh Horizontal alignment of the top-level DRS box in the surrounding text. You can ask for left-aligned (`l`) or centered boxes. The default is `c`.

\drsboxalignv Vertical alignment of the top-level DRS box with the baseline of the text. The default is `c` (centered on the baseline); `t` and `b` are also available. Embedded DRS boxes are always centered on the baseline.

\drsarraystretch This parameter gives the value of `\arraystretch` used when making DRS boxes. Undefined by default, which is equivalent to the value *1.0*. You can change it to other numbers (without a dimension) to stretch or compress linespacing.

\drslinewidth The thickness of the lines used to draw the DRS boxes. The default is the current value of `\arrayrulewidth` (which controls the width of table borders), normally 0.4pt. If you redefine this you should definitely load the package *pict2e.sty* (recommended anyway), which will allow LaTeX to set the diagonal lines of the quantifier diamond to the same thickness.

3.1 Compatibility with *covington.sty*

Although this package should accept DRSs written for the *covington.sty* macros, the appearance of the output is not the same by default. To match that style, specify the following formatting parameters:

```
\renewcommand\drscondfont{\it}
\renewcommand\drsalignment{l}
\renewcommand\drsboxalignh{l}
```

4 Other packages

I recommend that you use *drs.sty* in conjunction with *pict2e.sty,* which improves LaTeX's picture drawing capabilities. Your DRS diamonds will look a lot better if you do. *Drs.sty* is also compatible with *eepic.sty,* which has similar function but only works for DVI output (with PostScript enhancements). *Drs.sty* will not automatically load either package, but will take advantage of their capabilities if one is loaded.

I have tried to make *drs.sty* work well with doublespacing. Because of shortcomings of the doublespacing macros, the boxes might still be placed too close to each other. Perhaps some day this will be corrected in future versions of *setspace.sty* (but I requested the fix years ago, so don't hold your breath). Until then, I have provided a work-around which you can use if you are using *setspace.sty.* Place the command `\drshacksetspace` in your document preamble, somewhere after both packages have been included, to redefine certain *setspace* internals so that boxes can be properly spaced. This is turned off by default since it is a hack and might cause unexpected problems.

Starting with version 1.1, *drs.sty* is compatible with *memoir.cls* and the reimplementation of the *tabular* and *array* environments used by *memoir* and *array.sty.* However, the extra column types these provide cannot be used with *drs* commands. Only `c, l` and `r` are accepted.