LaTeX for Logicians

bussproofs.sty

A User Guide

Contents

The latest version of `bussproofs.sty` is 1.1, July 2011.

Guide version 1.1 March 2012. Corrections/suggestions to `peter_smith@me.com`
LaTeX for Logicians is at `http://www.logicmatters.net`

1 Introduction

Sam Buss's powerful and flexible package `bussproofs.sty` offers macros for setting natural deduction and sequent proofs in two different styles. The key difference between the two styles is illustrated by the difference between

$$\dfrac{\dfrac{\Gamma, A, B \vdash C}{\Gamma, A \vdash (B \to C)}}{\Gamma \vdash (A \to (B \to C))}$$

and

$$\dfrac{\dfrac{\Gamma, A, B \vdash C}{\Gamma, A \vdash (B \to C)}}{\Gamma \vdash (A \to (B \to C))}$$

In one style, the sequents above and below the inference line are centered; in the other, the sequents are placed so as to align their deducibility signs.

We can divide `bussproofs.sty`'s commands for building proofs into four types:

- *the structural commands* for producing proof trees in the *centering* style;
- *more structural commands* for producing proof trees in the *sequent-aligning* style;
- *additional commands* for adding labels, and changing the setting of inference lines (selecting double, dashed lines, etc.);
- *fine-tuning commands* for adjusting the spacing and layout of the proof.

`bussproofs.sty` also provides an additional command which enables some very laconic shorthand for the most commonly used commands. We describe these in Section 6; but we avoid using shorthand until the full commands have been explained and made familiar.

2 'Centered' proofs: the structural commands

2.1 The commands

There are four basic proof-building commands for producing proofs in the centered style:

`\AxiomC{`*form*`}`
`\UnaryInfC{`*form*`}`
`\BinaryInfC{`*form*`}`
`\TrinaryInfC{`*form*`}`

where '*form*' holds the place for a formula or sequent. Note

- The use of the surrounding '`{}`'s is mandatory.
- By default, *form* is set in text mode: if you want to set some or all of *form* in math mode, then you need to use `$`'s in the usual way.

Chaining a series of those commands, however, is not enough to cause a proof to be set. To display a proof you must use one of

`\DisplayProof`, a command placed at the end of a list of proof-building commands, which produces an in-line proof at the current insertion point.

`\begin{prooftree} ... \end{prooftree}`, an environment which contains a list of proof-building commands, and which produces a proof in a centered display.

Again, note

- The `\DisplayProof` command allows you to put proofs anywhere normal text might appear; for example, in a paragraph, in a table, in a tabbing environment, etc.
- To generate a proof in a centered display, do *not* use `$$ ... $$` or `\[ ... \]`.

2.2 Usage

It will be a *lot* clearer if we start with examples rather than begin with an abstract description of the principles for chaining proof-building commands! So here goes ...

```
\AxiomC{A}
\UnaryInfC{D}
\DisplayProof
```

produces the one-step proof $\dfrac{\mathrm{A}}{\mathrm{D}}$ in the current line. While

```
\begin{prooftree}
\AxiomC{A}
\AxiomC{B}
\BinaryInfC{D}
\end{prooftree}
```

displays the proof

$$\dfrac{\mathrm{A}\qquad\mathrm{B}}{\mathrm{D}}$$

centered and set off from the surrounding text. So it will be no surprise to learn that

```
\begin{prooftree}
\AxiomC{A}
\AxiomC{B}
\AxiomC{C}
\TrinaryInfC{D}
\end{prooftree}
```

produces the proof

$$\dfrac{\mathrm{A}\qquad\mathrm{B}\qquad\mathrm{C}}{\mathrm{D}}$$

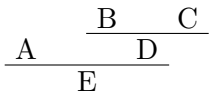
Two comments:

- You'll get an error message if there's a mismatch between the arity of the `\...aryInfC` command and the number of inputs.
- `bussproof.sty` does not allow arbitrary numbers of premisses appearing as separate *form* items above the inference line.

So much for one-step proofs. Now things get a little more fun .... Consider

```
\begin{prooftree}
     \AxiomC{A}
     \AxiomC{B}
     \AxiomC{C}
     \BinaryInfC{D}
     \BinaryInfC{E}
\end{prooftree}
```

This produces the proof



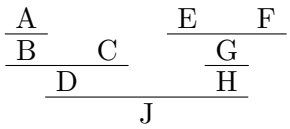
Why? Think of the first `\BinaryInfC` command as 'using up' as premisses the two axioms immediately above it. Then the second `\BinaryInfC` command takes as premisses 'D' immediately above it and then searches up to the next 'unused' item, i.e. 'A'. The inputs to a particular `\BinaryInfC` command are then set right-to-left in the order in which the command finds them as it looks up the series of commands. Which means, as we want, that the inputs to a particular inference are set left-to-right in the same order as they occur in the chain of commands.

It probably helps if we set out the proof-building commands like this:

```
\begin{prooftree}
   \AxiomC{A}
                \AxiomC{B}
                          \AxiomC{C}
                     \BinaryInfC{D}
         \BinaryInfC{E}
\end{prooftree}
```

That pretty much replicates the layout of the tree we want. This indeed is one of the *very* nice features of `bussproofs.sty` – the instructions to build a proof-tree can be set out to look very like the proof we want.

Suppose then that we want to set a tree that is arranged like this:



The following does the trick:

```
\begin{prooftree}
   \AxiomC{A}
   \UnaryInfC{B}
                \AxiomC{C}
         \BinaryInfC{D}
                                \AxiomC{E}
                                             \AxiomC{F}
                                    \BinaryInfC{G}
                                    \UnaryInfC{H}
                       \BinaryInfC{J}
\end{prooftree}
```

In sum, then, the rule to produce a given tree is

> *Set out the commands for the left-hand sub-proof first; and then within a sub-proof, do its left-hand sub-sub-proof first; and so on.*

Two more points about general structure:

- The command `\noLine` can be inserted before any inference command to suppress the drawing of an inference line.
- Null axioms are allowed!

Hence, the commands

```
\begin{prooftree}
   \AxiomC{$A \lor B$}
         \AxiomC{[$A$]}
         \noLine
         \UnaryInfC{$C$}
              \AxiomC{[$B$]}
              \noLine
              \UnaryInfC{$C$}
         \TrinaryInfC{$C$}
\end{prooftree}
```

yield

$$\dfrac{A \lor B \qquad \begin{array}{c}[A]\\ C\end{array} \qquad \begin{array}{c}[B]\\ C\end{array}}{C}$$

And we can use the fact that null axioms are allowed to generate natural deduction proofs with formulae overlined to indicate the discharge of premisses (by treating the formula to be overlined as a unary inference from a null axiom). For example:

```
\begin{prooftree}
   \AxiomC{$P$}
                    \AxiomC{}
                    \UnaryInfC{$\neg P$}
            \BinaryInfC{$\bot$}
            \UnaryInfC{$\neg\neg P$}
\end{prooftree}
```

generates the proof

$$\dfrac{\dfrac{P \qquad \overline{\neg P}}{\bot}}{\neg\neg P}$$

2.3 More than three premisses

So far we've only considered inference steps that take one premiss (like double negation), two premisses (like modus ponens), or three inputs (like or-elimination). But Buss proofs in fact supplies – should you want it – the capacity to deal with four or five premisses. The syntax is as you would expect. We have the two commands

`\QuaternaryInfC{`*form*`}`
`\QuinaryInfC{`*form*`}`

These aren't pointless! After all, you might want to illustrate the ω-rule thus:

```
\begin{prooftree}
\AxiomC{$\varphi(0)$}
\AxiomC{$\varphi(1)$}
\AxiomC{$\varphi(2)$}
\AxiomC{$\varphi(3)$}
\AxiomC{$\ldots$}
\QuinaryInfC{$\forall n\varphi(n)$}
\end{prooftree}
```

to give

$$\dfrac{\varphi(0) \qquad \varphi(1) \qquad \varphi(2) \qquad \varphi(3) \qquad \ldots}{\forall n \varphi(n)}$$

3 'Sequent-aligned' proofs: the structural commands

3.1 The commands

There are four basic proof-building commands for producing proofs in the second style:

`\Axiom$` *form* `\fCenter` *form* `$`
`\UnaryInf$` *form* `\fCenter` *form* `$`
`\BinaryInf$` *form* `\fCenter` *form* `$`
`\TrinaryInf$` *form* `\fCenter` *form* `$`

where '*form*' holds the place for a formula or formulae, and `\fCenter` marks the point of alignment.

- This time, the use of the surrounding '`$`'s is mandatory; and material between the '`$`'s is set in math mode.
- The '`\fCenter`' is by default just a place-marker; however, it can be redefined to be printed character such as a sequent arrow or turnstile.

And for multi-premiss usage, we also have the additional commands:

`\QuaternaryInf$` *form* `\fCenter` *form* `$`
`\QuinaryInf$` *form* `\fCenter` *form* `$`

3.2 Usage

The basic usage is illustrated by the following:

```
\begin{prooftree}
\Axiom$A, B, C, D,\fCenter\ E, F$
\UnaryInf$A, B,\fCenter\ C,  D, E, F$
\end{prooftree}
```

$$\frac{A, B, C, D,\ E, F}{A, B,\ C, D, E, F}$$

Now let's redefine `\fCenter` to be the turnstile surrounded by spaces. Then

```
\begin{prooftree}
\def\fCenter{\ \vdash\ }
\Axiom$A, B, C, D \fCenter E, F$
\UnaryInf$A, B \fCenter C,  D, E, F$
\end{prooftree}
```

produces

$$\frac{A, B, C, D\ \vdash\ E, F}{A, B\ \vdash\ C, D, E, F}$$

However, it is in general preferable to redefine `\fCenter` using an `\mbox` as in the next example (this allows more options):

```
\def\fCenter{\mbox{\Large$\rightarrow$}}
\begin{prooftree}
\Axiom$\Gamma, A, B\ \fCenter\ B$
\UnaryInf$\Gamma, A\ \fCenter\ (B \to C)$
\UnaryInf$\Gamma\ \fCenter\ (A \to (B \to C))$
\end{prooftree}
```

produces

$$\frac{\dfrac{\Gamma, A, B \longrightarrow B}{\Gamma, A \longrightarrow (B \to C)}}{\Gamma \longrightarrow (A \to (B \to C))}$$

And this time, since the `\def\fCenter{...}` command precedes the following `prooftree` environment, it will stay in effect until overridden later in the document.

- The chaining of multiple sequent-aligning commands to build complex proofs works just as with the earlier centering-style commands.
- It should go without saying that the point of the `\BinaryInf$...$` command is not to align the sequent turnstile, (arrow, or whatever) in the conclusion of the two-premiss inference with the turnstiles in the premisses, but to provide a new alignment point for later sequents.
- Mixing and matching sensible combinations of `C{...}`-type centering commands and `$...$`-type sequent-aligning commands is in fact legitimate.

Thus, to illustrate the last point,

```
\begin{prooftree}
\def\fCenter{\mbox{\ $\vdash$\ }}
\AxiomC{$\Gamma \vdash A$}
\AxiomC{$A, B \vdash C$}
\BinaryInf$\Gamma, B \fCenter C$
\UnaryInf$\Gamma \fCenter B \to C$
\end{prooftree}
```

is acceptable, and yields

$$\dfrac{\dfrac{\Gamma \vdash A \qquad A, B \vdash C}{\Gamma, B \vdash C}}{\Gamma \vdash B \to C}$$

4 Additional layout commands

Conventionally, the proof

$$\dfrac{\dfrac{P \qquad \dfrac{}{\neg P}}{\bot}}{\neg\neg P}$$

needs labelling. This section describes how to add labels to proofs, and also how to change the style of inference-line (to double lines, etc.).

4.1 Adding labels

There are two commands for decorating a proof with labels:

```
\LeftLabel{text}
\RightLabel{text}
```

These put `text` as a label to the left/right of the next inference line (and they work even if `\noLine` is used and line isn't actually drawn).

So, for example, the commands

```
\begin{prooftree}
         \AxiomC{$\Gamma, A \vdash B$}
    \LeftLabel{Conditional Proof:}
         \UnaryInfC{$\Gamma \vdash A \rightarrow B$}
\end{prooftree}
```

will generate

$$\text{Conditional Proof:}\ \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

You can insert spacing commands and other text commands into the label text, so e.g. substituting `\LeftLabel{Conditional Proof:\quad}` will generate

$$\text{Conditional Proof:}\quad \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

However, there is also a global command for adjusting the position of labels, and in general it might be better to use that.

You can reduce the size of the label font, so that

```
\begin{prooftree}
   \AxiomC{$P$}
      \AxiomC{}
  \RightLabel{\scriptsize(1)}
```

```
                \UnaryInfC{$\neg P$}
        \BinaryInfC{$\bot$}
        \RightLabel{\scriptsize(1)}
        \UnaryInfC{$\neg\neg P$}
\end{prooftree}
```

will set the conventionally annotated proof

$$\dfrac{\dfrac{P \qquad \dfrac{}{\neg P}\,(1)}{\bot}}{\neg\neg P}\,(1)$$

4.2 Varieties of inference lines

The inference line between premiss and derived conclusion defaults, as we've seen, to a single solid line. But this can be changed, either locally or more globally. The local change commands are

```
\noLine
\singleLine
\doubleLine
\solidLine
\dottedLine
\dashedLine
```

Each applies to the next inference, and line-*number* and line-*type* commands can be combined: so, for example,

```
\begin{prooftree}
     \AxiomC{$\Gamma, A \vdash B$}
     \doubleLine\dashedLine
     \UnaryInfC{$\Gamma \vdash A \rightarrow B$}
\end{prooftree}
```

produces the following:

$$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}$$

There are corresponding global commands that can be placed anywhere in a proof and will affect all succeeding lines unless and until overridden.

```
\alwaysNoLine
\alwaysSingleLine
\alwaysDoubleLine
\alwaysSolidLine
\alwaysDottedLine
\alwaysDashedLine
```

Here 'succeeding lines' means 'succeeding in the (vertical) order that commands are given' not 'logically succeeding'. (Note: Sam Buss comments that `\alwaysNoLine` is 'untested': but it seems to work fine on my trials.)

So, for example, the commands

```
\begin{prooftree}
     \alwaysNoLine
     \AxiomC{$\Pi_1$}
     \UnaryInfC{$A$}
               \AxiomC{[$A$]}
               \UnaryInfC{$\Pi_2$}
               \UnaryInfC{$B$}
               \alwaysSingleLine
               \UnaryInfC{$A \to B$}
          \BinaryInfC{$B$}
\end{prooftree}
```

produces the following:

$$\dfrac{\Pi_1 \quad A \qquad \dfrac{\begin{array}{c}[A]\\ \Pi_2 \\ B\end{array}}{A \to B}}{B}$$

4.3 Turning a proof upside down

Suppose you want to set a proof-tree the other way up, i.e. get a *downward* branching tree with the root at the *top*. Then you need the command `\rootAtTop`. Note, this applies only to the current proof. If you want to make this behaviour persistent, then use `\alwaysRootAtTop`. (The default behaviour can be restored using `\rootAtBottom` and `\alwaysRootAtBottom`.)

5 Fine-tuning the proof layout

`bussproofs.sty` allows a lot of user-modification of the default style of proof lay-out, either by global modifications of parameters, or by local tweakings within a proof. We'll divide these commands into

- commands for determining *where a whole proof is set*;
- commands for *adjusting the 'geometry' within a proof*;
- commands for *redefining line styles*.

5.1 Where a proof is set

i The amount of *space above and below* a proof-tree display is set by the parameter

```
\proofSkipAmount
```

Its default setting is `{\vskip.8ex plus.8ex minus.4ex}`. This can be modified, and negative values of `\vskip` are accepted.

To change the space above the tree (which you'll want to do if the top lines are null axioms, which generate additional white space), use `\def\proofSkipAmount{\vskip` *new-skip*`}` *before* the opening command `\begin{prooftree}`; the size of the skip after the proof can then be reset by another `\def\proofSkipAmount{...}` command before the proof closes.

ii Whole proofs set using the proof-tree environment can be *nudged sideways* by using the command `\hskip` *nudge* (no brackets!) placed e.g. immediately after the command `\begin{prooftree}`. A positive value of *nudge* such as `0.5in` shifts the proof to the right, a negative value such as `-20pt` shifts the proof left.

iii For *vertical positioning of in-line proofs*, compare $\dfrac{\phi \wedge \psi}{\phi}$ with $\dfrac{\phi \wedge \psi}{\phi}$ and $\dfrac{\phi \wedge \psi}{\phi}$. The first is set vertically centered within the current line (the default position), the second is set rather lower so (at least with a one-step proof) the inference line is on a level with the foot of the current line, and the third is set with the conclusion aligned to the current line. To produce the second tree, use the command `\centerAlignProof`; to produce the third tree, use `\bottomAlignProof`; to restore the default positioning use `\normalAlignProof`.

5.2 Layout within a proof

In this section, we consider the use of

`\ScoreOverhang` (controls how far inference lines 'overhang' the premisses/conclusion)
`\extraVskip` (controls the extra space above and below lines)
`\labelSpacing` (controls horizontal space separating labels and inference lines)
`\defaultHypSeparation` (controls horizontal spread of proof tree)
`\insertBetweenHyps` (can be used to adjust position of hypotheses in next inference)
`\kernHyps` (nudges hypotheses left or right)

i The default setting of `\ScoreOverhang` is `{4pt}`. This produces e.g. the tree

$$\dfrac{\dfrac{\phi \wedge \psi}{\psi} \quad \dfrac{\phi \wedge \psi}{\phi}}{\psi \wedge \phi}$$

Inserting, for example, the line `\def\ScoreOverhang{1pt}` at the beginning of the chain of proof-commands changes (improves!?) the look of the tree by reducing the 'overhang' of inference lines:

$$\dfrac{\dfrac{\phi \wedge \psi}{\psi} \quad \dfrac{\phi \wedge \psi}{\phi}}{\psi \wedge \phi}$$

ii The default setting of `\extraVskip` is `{2pt}`, which inserts 2 pt spacing above and below an inference line, as in the last two proofs. To increase this throughout the proof, use e.g. the line `\def\extraVskip{3pt}` at the beginning of the chain of proof-commands to get

$$\dfrac{\dfrac{\phi \wedge \psi}{\psi} \quad \dfrac{\phi \wedge \psi}{\phi}}{\psi \wedge \phi}$$

You can also change the interline skip on the fly, during a proof, to improve alignments. Thus compare

$$\dfrac{\Pi_1 \quad \dfrac{\begin{array}{c}[A]\\ \Pi_2 \\ B\end{array}}{A \to B}}{B} \qquad\qquad \dfrac{\Pi_1 \quad \dfrac{\begin{array}{c}[A]\\ \Pi_2 \\ B\end{array}}{A \to B}}{B}$$

Here, the proof on the left was created by using the default settings, and the one on the right was created by the following commands (with the two proofs being set in the same line in a `\begin{center} ... \end{center}` environment):

```
\alwaysNoLine
\AxiomC{$\Pi_1$}
\def\extraVskip{1pt}
\UnaryInfC{$A$}
          \AxiomC{[$A$]}
          \def\extraVskip{2pt}
          \UnaryInfC{$\Pi_2$}
          \UnaryInfC{$B$}
          \def\extraVskip{2.5pt}
          \alwaysSingleLine
          \UnaryInfC{$A \to B$}
     \BinaryInfC{$B$}
\DisplayProof
```

iii The default setting of `\labelSpacing` is `{3pt}`, which inserts 3 pt between inference lines and their associated labels. To change this, use the command `\def\labelSpacing{`*new-space*`}` at the beginning of a chain of proof-commands. For example

$$\dfrac{\dfrac{\dfrac{(\phi \wedge \psi)}{\psi} \qquad \dfrac{}{\neg\psi}\ (1)}{\bot}}{\neg(\psi \wedge \phi)}\ (1)$$

is set with `\def\labelSpacing{8pt}`.

iv The default setting of `\defaultHypSeparation` is `{\hskip.2in}`. Increase this to spread out a spread out a proof horizontally, decrease it to squeeze the proof. Thus compare

$$\dfrac{\dfrac{\phi \wedge \psi}{\psi} \quad \dfrac{\phi \wedge \psi}{\phi}}{\psi \wedge \phi} \qquad\qquad \dfrac{\dfrac{\phi \wedge \psi}{\psi} \qquad\qquad \dfrac{\phi \wedge \psi}{\phi}}{\psi \wedge \phi}$$

where the proof commands for the left tree start `\def\defaultHypSeparation{\hskip .1in}`, and for the right tree start `\def\defaultHypSeparation{\hskip .5in}`.

v Use the command `\insertBetweenHyps{`*stuff*`}` before a binary or trinary inference command to insert *stuff* between the hypotheses of that inference. The obvious use of this is to adjust the spacing between the hypotheses of that inference, by using e.g. `\insertBetweenHyps{\hskip -4pt}`.

vi Use the command `\kernHyps{`*kern*`}` before an inference command to nudge the hypotheses of that inference right by the distance *kern*, e.g. `4pt`. Negative values of *kern* nudge the hypotheses to the left.

5.3 Inference-line styles

In this section, we consider the use of

`\ruleScoreFiller` (sets the style of solid rules)
`\dottedScoreFiller` (sets the style of dotted rules)
`\dashedBuildScore` (sets the style of dashed rules)

i The default setting of `\ruleScoreFiller` is `{\hrule}`, and in LaTeX an unqualified `\hrule` command produces a horizontal line of thickness 0.4 pt. To change the rule's thickness, use the command `\def\ruleScoreFiller{\hrule height` *new-thickness*`}`.

ii The default setting of `\dottedScoreFiller` is `{\hbox to4pt{\hss.\hss}}`. The default for `\dashedBuildScore` is

```
{\hbox to2.8mm{\hss\vrule width1.4mm height0.4pt depth0.0pt\hss}}
```

Both defaults can be altered to produce any other desired type of line.

6 Abbreviating the commands

Using the command `\EnableBpAbbreviations` enables some laconic shorthand for various commands:

`\AX` and `\AXC` abbreviate `\Axiom` and `\AxiomC`
`\UI` and `\UIC` abbreviate `\UnaryInf` and `\UnaryInfC`
`\BI` and `\BIC` abbreviate `\BinaryInf` and `\BinaryInfC`
`\TI` and `\TIC` abbreviate `\TrinaryInf` and `\TrinaryInfC`
`\DP` abbreviates `\DisplayProof`

The enabling of these short abbreviations is optional, and users should guard against conflicts with commands from other macro packages or their own defined abbreviations.