The proofpack package*

David W. Agler
https://github.com/davidagler/proofpack

Released 2023/07/09

1 Introduction

There are several packages that provide support for typesetting natural deduction proofs. These include lplfitch by John Etchemendy, Dave Barker-Plummer, and Richard Zach, logicproof by Alan Davison, natded by Ajalloeian and Pelletier, synproof by Yuri Robbers and Paul Isambert, `fitch` by Johan Klöwer, another fitch by Peter Selinger, and bussproofs by Samuel R. Buss.

`proofpack` is another package to typeset natural deduction proofs (in mostly a Fitch-style way). proofpack is most similar to fitch.sty by Johan W. Klawer (June 10, 2001) and edited by Alexander W. Kocurek (June 8, 2019). It deviates from this package in several ways, but notably it allows for simpler input. The simpler input makes it more attractive to those new to LaTeX and for students learning logic.

2 Usage

The way to create a natural deduction proof using `proofpack` is to use one of the three environments: `sdeduce`, `mdeduce`, or `ldeduce`. Proofs are formatted in the same way you would create a table using `tabular` except you don't need to specify the number of columns and, at least for two of the environments, the numbering of the lines is automatic. Within the environments, you can use the commands `\hyp`, `\hypl`, and `\sub` to create assumptions and subproofs.

2.1 Environments and Commands

The main environments for creating proofs are `sdeduce`, `mdeduce`, and `ldeduce`. These are short for "short deduction", "manual deduction", and "long deduction", respectively. Here is a brief description of each environment.

`sdeduce` (*env.*) This environment creates proofs that do not break across the page. Proofs are automatically numbered. Usage: `\begin{sdeduce}` ... `\end{sdeduce}`.

`mdeduce` (*env.*) This environment creates proofs that do not break across the page. Proofs are manually numbered. Usage: `\begin{mdeduce}` ... `\end{mdeduce}`.

*This document corresponds to proofpack v0.2, dated 2023/07/09 .

`ldeduce` (*env.*) This environment creates proofs that break across the page. Proofs are automatically numbered. Usage: `\begin{ldeduce}` ... `\end{ldeduce}`.

Without a package, natural deduction proofs are super easy to make so long as you don't need to make any assumptions: you can make the proof in a `tabular` environment.

1	$P \rightarrow Q$	P	
2	P	P	
3	Q	$1, 2 \rightarrow E$	

```
\begin{tabular}{lll}
1&$P\rightarrow Q$ &P\\
2&P                &P\\
3&Q                &$1,2 \rightarrow E$\\
\end{tabular}
```

The headache of typesetting proofs in LaTeX using tables comes when you need to make an assumption and start a subproof. To solve this problem, `proofpack` makes use of three commands. The primary commands creating and reasoning in subproofs are `\hyp`, `\hypl`, and `\sub`. These are short for "hypothesis", "hypothesis with a line", and "subproof, respectively. The commands are used in the `sdeduce`, `mdeduce`, and `ldeduce` environments. Here is a brief description of each of these commands:

`\hyp` The `\hyp[`⟨*int*⟩`]` command is used in a `sdeduce`, `mdeduce`, or `ldeduce` environment to make an assumption / hypothesis (hyp). The hypothesis does not have a right line underneath like you see in many textbooks. Example: `\hyp P\rightarrow Q`. The command takes an optional argument to set the depth of the subproof. Example: `\hyp[3] P\rightarrow Q`. This is a major improvement on earlier fitch.sty as it results in less code. You don't need to write `\hyp\hyp\hyp P\rightarrow Q`. The default depth is 1.

`\hypl` The `\hypl[`⟨*int*⟩`]` command is used in a `sdeduce`, `mdeduce`, or `ldeduce` environment to make an assumption / hypothesis (hyp) with a line underneath the assumption. Example: `\hypl P\rightarrow Q`. The command takes an optional argument to set the depth of the subproof. Example: `\hypl[3] P\rightarrow Q`. This sets the depth of the formula $P \rightarrow Q$ at 3. The default depth is 1.

`\sub` The `\sub[`⟨*int*⟩`]` command is used in a `sdeduce`, `mdeduce`, or `ldeduce` environment to set a formula at a certain depth within a subproof. Example: `\sub P\rightarrow Q`. The command takes an optional argument to set the depth of the subproof. Example: `\sub[3] P\rightarrow Q`. This sets the depth of the formula $P \rightarrow Q$ at 3. The default depth is 1.

3 Examples

Now that we have a sense of the key environments and commands, let's see some examples. In our first example, we'll use the `sdeduce` environment to create a non-breaking proof. Notice that the proof is automatically numbered, that the middle column is in math mode so you do not need to write `$P\rightarrow Q$` but the rightmost column is not in math mode so you need to write `$\rightarrow E$`. Just as with the `tabular` environment, you can use the `&` command to move to the next column and the `\\` command to move to the next row.

1	$P \to Q$	P	
2	P	P	
3	Q	$1, 2 \to E$	

```
\begin{sdeduce}
P\rightarrow Q   &P\\
P                &P\\
Q                &$1,2 \rightarrow E$\\
\end{sdeduce}
```

To typeset a proof without automatic numbering, use the `mdeduce` environment. With `mdeduce` you need to add a column for the line numbers. The line numbers are automatically formatted.

1	$P \to Q$	P	
2	P	P	
3	Q	$1, 2 \to E$	

```
\begin{mdeduce}
1& P\rightarrow Q   &P\\
2& P                &P\\
3& Q                &$1,2 \rightarrow E$\\
\end{mdeduce}
```

Assumptions and subproofs are created using the `hyp` and `hypl` commands. Both commands take one optional argument: the depth of the subproof. The default depth is 1. The `hyp` command creates an assumption / hypothesis (hyp) without a line underneath. The `hypl` command creates an assumption / hypothesis (hyp) with a line underneath. To create a subproof, simply write `\hyp[int]` or `\hypl[int]` before the formula you want to set as an assumption, where `int` is a positive integer.

1	Q	P	
2	$\vdash P$	A	

```
\begin{sdeduce}
Q      &P\\
\hypl[1] P  &A\\
\end{sdeduce}
```

Let's look at at an example where we use `hyp`.

1	Q	P	
2	$\vert\ P$	A	

```
\begin{sdeduce}
Q      &P\\
\hyp[1] P  &A\\
\end{sdeduce}
```

To move deeper into the subproof, we can change the optional argument of the `hyp` and `hypl` commands. The default depth is 1. The following example shows how to use the optional argument to move deeper into the subproof.

1	Q	P	
2	$\vert\ P$	A	
3	$\vert\ \vert\ Q$	A	

```
\begin{sdeduce}
Q      &P\\
\hypl[1] P &A\\
\hypl[2] Q &A\\
\end{sdeduce}
```

Whereas `hyp` and `hypl` are commands for moving deeper into (starting) a subproof, `sub` is a command for moving within a subproof of the same level. Just write `\sub[int]` where `int` is the depth of the subproof expressed as an integer. The following example shows how to use the `sub` command.

1	Q	P	
2	$\vert\ P$	A	
3	$\vert\ Q$	R	

```
\begin{sdeduce}
Q      &P\\
\hypl[1] P   &A\\
\sub[1] Q    &$R$\\
\end{sdeduce}
```

Here is a proof illustrates the use of `hypl` and `sub` together.

1 Q P
2 | P A
3 | Q R
4 | | Q A
5 | | Q R

```
\begin{sdeduce}
Q               &P\\
\hypl[1] P      &A\\
\sub[1] Q       &R\\
\hypl[2] Q      &A\\
\sub[2] Q       &R\\
\end{sdeduce}
```

When the proof is manually numbered, you want to place the `hyp`, `hypl`, and `sub` commands after the column that has the numbering. The following example shows how to use the `hyp`, `hypl`, and `sub` commands in the `mdeduce` environment and that manual numbering of lines is not restricted to integers.

1 $P \to Q$ P
2 P P
3 Q $1, 2 \to E$
n+k Q $1, 2 \to E$
n+k+1 | Q $1, 2 \to E$

```
\begin{mdeduce}
1& P\rightarrow Q  &P\\
2& P               &P\\
3& Q               &$1,2 \rightarrow E$\\
n+k & Q           &$1,2 \rightarrow E$\\
n+k+1 &\hyp[1] Q  &$1,2 \rightarrow E$\\
\end{mdeduce}
```

You can place a box around formulas or the justification in the proof. This may be useful if you want to highlight certain lines in a proof. If you put a formula in a box, you need to use mathmode.

1 $\boxed{P \to Q}$ P
2 $\boxed{Q}$ $\boxed{\text{P}}$
3 $Q \vee Z$ $2 \vee I$

```
\begin{sdeduce}
\bxdwff{$P\rightarrow Q$} &P\\
\bxdwff{$Q$}              &\bxdwff{P}\\
Q\vee Z                     &$2\vee I$\\
\end{sdeduce}
```

You can also color anything in mathmode in the proof. Usage: `\cwff[color]{formula}` This may be useful if you want to highlight certain lines in a proof.

1 $P \to Q$ P
2 P $\boxed{\text{P}}$
3 Q $\wedge E$

```
\begin{sdeduce}
P\rightarrow Q           &P\\
\cwff[magenta]{P}   &\bxdwff{P}\\
\cwff[green]{Q}     &\cwff[blue]{$\wedge E$}\\
\end{sdeduce}
```

The `ldeduce` environment takes three optional arguments 1 is the alignment (options include "l" for left, "c" for center, and "r" for right; the default is l), 2 is the caption (default is empty), and 3 is the label (default is empty). The following example shows how to use the optional arguments.

1 $\phi \to \psi$ P
2 $(\exists x)(\exists y)Pxy$ P

Proof 2: A centered proof

1 $\phi \to \psi$ P
2 $(\exists x)(\exists y)Pxy$ P

1 $\phi \to \psi$ P
2 $(\exists x)(\exists y)Pxy$ P

For long proofs or for proofs you want to break across pages, use the `ldeduce` environment. This environment will automatically number the lines of the proof.

1 $\phi \to \psi$ P

2	$(\exists x)(\exists y)Pxy$	P
3	$\vert\ P \rightarrow Q$	$1, 2 \vee E$
4	$\vert\ \Box P$	$1, 2 \vee E$
5	$\vert\ \Diamond\neg P$	$1, 2 \rightarrow E$
6	$\Diamond\phi \leftrightarrow \phi$	Magic rule
7	$\vert\ \underline{Q\quad\quad}$	New Assumption!
8	$\vert\ \boxed{Q \rightarrow Q}$	$1, 2 \vee E$
9	$\vert\ \vert\ (\exists x)Px$	A, 2nd level
10	$\vert\ \vert\ \phi \wedge \psi\triangleleft$	2nd level
11	$\vert\ \vert\ \triangleright\phi \wedge \psi$	2nd level
12	$\vert\ \vert\ \vert\ \underline{(\forall x)Sx}$	A, 3rd level
13	$\vert\ \vert\ \vert\ q$	$1, 2 \rightarrow E$
14	$\vert\ \vert\ Q$	$1, 2 \rightarrow E$
15	q	$1, 2 \rightarrow E$
16	e	$1, 2 \rightarrow E$
17	r	$1, 2 \rightarrow E$
18	Γ	$1, 2 \rightarrow E$
19	Σ	$1, 2 \rightarrow E$
20	Q	$1, 2 \rightarrow E$
21	Q	$1, 2 \rightarrow E$
22	Q	$1, 2 \rightarrow E$
23	Q	$1, 2 \rightarrow E$
24	Q	$1, 2 \rightarrow E$
25	Q	$1, 2 \rightarrow E$
26	Q	$1, 2 \rightarrow E$
27	Q	$1, 2 \rightarrow E$
28	Q	$1, 2 \rightarrow E$
29	Q	$1, 2 \rightarrow E$
30	Q	$1, 2 \rightarrow E$
31	Q	$1, 2 \rightarrow E$
32	Q	$1, 2 \rightarrow E$
33	Q	$1, 2 \rightarrow E$
34	Q	$1, 2 \rightarrow E$
35	Q	$1, 2 \rightarrow E$
36	Q	$1, 2 \rightarrow E$
37	Q	$1, 2 \rightarrow E$
38	Q	$1, 2 \rightarrow E$
39	Q	$1, 2 \rightarrow E$
40	Q	$1, 2 \rightarrow E$
41	Q	$1, 2 \rightarrow E$
42	Q	$1, 2 \rightarrow E$

```
\begin{ldeduce}
\phi\rightarrow \psi                    &P\\
(\exists x)(\exists y)Pxy              &P\\
\hyp \cwff[magenta]{P\rightarrow Q}  &$1,2\vee E$\\
\sub[1] \Box P                          &$1,2 \vee E$\\
\sub[1] \Diamond \neg P                 &$1,2 \rightarrow E$\\
\Diamond\phi\leftrightarrow\phi         &Magic rule\\
\hypl[1] Q                              &New Assumption!\\
```

```
\sub[1] \bxdwff{$Q\rightarrow Q$}  &$1,2 \vee E$\\
\hyp[2] (\exists x)Px               &$A$, 2nd level\\
\sub[2] \phi\wedge\psi \rightmarker  &2nd level\\
\sub[2] \leftmarker \phi\wedge\psi   &2nd level\\
\hypl[3] (\forall x)Sx              &$A$, 3rd level\\
\sub[3] q                           &$1,2 \rightarrow E$\\
\sub[2] Q                           &$1,2 \rightarrow E$\\
q                                   &$1,2 \rightarrow E$\\
e                                   &$1,2 \rightarrow E$\\
r                                   &$1,2 \rightarrow E$\\
\Gamma                              &$1,2 \rightarrow E$\\
\Sigma                              &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
Q                                   &$1,2 \rightarrow E$\\
\end{ldeduce}
```

4 Implementation

<*package>

packages Some packages are required for the package to work.

```
\RequirePackage{lmodern, amsmath, tikz}
\RequirePackage{newfloat}
\RequirePackage{amssymb, mathtools, etoolbox, array, longtable}
\RequirePackage{xparse}
```

counters Several counters are used to keep track of the line numbers in the proof.

```
\newcounter{deducecounter}
\setcounter{deducecounter}{0}
\newbool{resetdeducecounter}
```

```
\booltrue{resetdeducecounter}
\newbool{increasededucecounter}
\booltrue{increasededucecounter}
```

more counters Several counters are used to keep track of the line numbers in the proof. When the user sets line counter, `userlinecounter` switches to true and the counter is manually set.

```
\newcounter{userlinecounternum}
\setcounter{userlinecounternum}{0}
\newbool{userlinecounter} % manually set line counter
\boolfalse{userlinecounter}
```

`\formatdeducecounter` Formats the line numbers in the proof.

```
\newcommand{\formatdeducecounter}[1]{\scriptsize \arabic{#1}}
```

`\deducecounter` Increments the line counter in the proof. If the user sets the line counter, then the counter is set by the user. Otherwise, the counter is incremented by 1. Also formats the line counter. See `\formatdeducecounter` if you want to change the font style of the line counter.

```
\newcommand{\deducecounter}{%
\ifbool{increasededucecounter}{\addtocounter{deducecounter}{1}}{}
\ifbool{userlinecounter}{\setcounter{deducecounter}{
  \value{userlinecounternum}}}{}%
\formatdeducecounter{deducecounter}}
```

`\pplineht` Formats the line heights in the proof.

```
\newlength{\pplineht}
\setlength{\pplineht}{1.5\baselineskip}
```

`\ppindent` Horizontal indent between proof levels, used by the `\hyp`, `\hypl`, and `\sub` commands. Increase or decrease to move the subproofs further or closer to each other. This length is used to define several other lengths.

```
\newlength{\ppindent}
\setlength{\ppindent}{0.4em}
```

`\ppnumsep` Horizontal space between leftmost numbers (1st column) and the main body of the proof. Increasing this value will create more space between numbered lines and the formulas in the main body of the proof. You might considering decreasing the value to create more compact proofs. Simply replace `\ppindent` with a smaller value, e.g. 0.3em.

```
\newlength{\ppnumsep}
\setlength{\ppnumsep}{\ppindent}
```

`\ppsep` Horizontal space

```
\newlength{\ppsep}
\setlength{\ppsep}{\ppindent}
```

`\ppnumwd` Line number width. Increasing this value will push the proofs to the right.

```
\newlength{\ppnumwd}
\setlength{\ppnumwd}{1em}
```

`\ppjustindent` Indent for justification column. By increasing the value of `\ppjustindent`, you are increasing the distance between the formula in the main body of the proof and the rightmost column that contains the justification. Increase for more space, decrease for more compact proofs.

```
\newlength{\ppjustindent}
\setlength{\ppjustindent}{2em}
```

`\ppvline` Creates a shorter line to start subproof. To do this, we set the height and depth of the line using the `strut` and set the width using `\arrayrulewidth`. Some explanation of how everything works. First, `\makeatletter` and `\makeatother` allow for the use of `@` in command names. The `\vrule` draws the vertical line. If we simply drew the vertical line without any adjustments, the line would be too high. So, we want to adjust it to the level of the strut. The strut height is the maximum height of uppercase letters and its depth is the maximum depth of lowercase letters. We want the height of the line to be the height of the strut and the depth of the line to be the depth of the strut plus 2pt. This 2pt makes the line a little longer so it connects up with the vertical line below it. To set the height of the `vrule` we set its height using `@height` and set it to the height of the strutbox using `\ht\strutbox`. We do the same thing for the depth but we use `\dimexpr` to add 2pt to the depth of the strutbox. The `\dimexpr` command allows for the use of `+` and `-` in the command. Finally, we need to set the width or thickness of the line. The width is set to `\arrayrulewidth`, which is the width of lines in array environments. If you wanted a thicker line, you could replace `\arrayrulewidth` with another measurement, e.g. `\@width 3pt`. The `\relax` is used to end the command.

```
\makeatletter
\newcommand\ppvline{%
  \vrule\@height\ht\strutbox\@depth\dimexpr
  \dp\strutbox + 2pt\@width\arrayrulewidth\relax
}
\makeatother
```

`\hyp` The `\hyp[`⟨*int*⟩`]` command is used in the proof environment to make an assumption / hypothesis (hyp). The hypothesis does not have a right line underneath like you see in many textbooks. Example: `\hyp P\rightarrow Q`. The command takes an optional argument to set the depth of the subproof. Example: `\hyp[3] P\rightarrow Q`. This is probably the main reason you would use `proofpack` over the earlier `fitch.sty` as it results in less code. You don't need to write `\hyp\hyp\hyp P\rightarrow Q`. The default depth is 1.

```
\newcounter{counthyp} % counter for number of hypotheses
\setcounter{counthyp}{1} % set counter to 1
\newcommand{\hyp}[1][1]{%
    \setcounter{counthyp}{1}
    \whileboolexpr{test{\ifnumcomp{#1}{>}{\value{counthyp}}}}
```

```
    {
        \hspace*{\ppsep}\vline\hspace*{\ppindent}\stepcounter{counthyp}% True
    }
    \hspace*{\ppsep}\ppvline\hspace*{\ppindent}% False
}
```

`\hypl` Similar to hypothesis `hyp` except `hypl[`⟨*int*⟩`]` is "hypothesis with a line" (hence `hypl`). The optional argument`[`⟨*int*⟩`]` sets the depth. The line is a right line under the assumption. Here is how it works. First, the `counthyp` is set to 1. The `whileboolexpr` checks whether the user has input a value greater than 1 as the optional argument. If it is, then it sets the counter to the optional argument and draws a vertical line in the normal way (without a right horizontal line). It then increments the counter. If the optional argument is not greater than 1, then it draws the vertical line with a right horizontal line. Once the the conditional in the while loop is false, then the right horizontal line is drawn. This is done by creating horizontal space, drawing the `ppvline` rather than a `vline`. The right line is drawn using `\rule`. You can adjust its placement using `\hspace*{4.0pt}` and its thickness using `\arrayrulewidth`.

```
\newcommand{\hypl}[1][1]{%
    \setcounter{counthyp}{1}
    \whileboolexpr{test{\ifnumcomp{#1}{>}{\value{counthyp}}}}
    {
      \hspace*{\ppsep}\vline\hspace*{\ppindent}\stepcounter{counthyp}% True
    }
    \hspace*{\ppsep}\ppvline\hspace*{-\dimexpr\ppsep}\raisebox{-1.25ex}
    {\hspace*{4.0pt}\rule[.5ex]{2.35em}{\arrayrulewidth}}
    \hspace*{\dimexpr \ppindent - 2.35em}% False
}
```

`\sub` The `\sub[`⟨*int*⟩`]` command creates a line in the proof at a certain depth but the line is not the assumption/hypothesis that starts the subproof. Technically, you could use it to start the subproof, but `sub` does not make use of the adjusted vertical line like `hyp` and `hypl`. The command takes an optional argument to set the depth of the line. Example: `\sub[3]` is a subproof with a depth of 3. The default depth is 1.

```
\newcounter{sub}
\setcounter{sub}{0}
\newcommand{\sub}[1][1]{
    \foreach \x in {1,...,#1} {%
    \hspace*{\ppsep}\vline\hspace*{\ppindent}
    }
    \stepcounter{sub}%
}
```

`markers` Use these to add markers to left and right of lines to highlight important steps in the proof. This was in the original fitch.sty. My implementation of this needs some additional work.

```
\newcommand{\rightmarker}{%
\makebox[1.75\ppindent][r]{${\lhd}$}}
```

```
\newcommand{\leftmarker}{%
\hspace*{\ppsep}\makebox[0pt][r]{${\rhd}$}}
```

`\bxdwff` Command to draw a box around a wff. Use `\bxdwff{wff}` where wff is the formula you want to surround in a box. Example: `\bxdwff{$P\rightarrow Q$}`

```
\newcommand{\bxdwff}[1]{%
  \tikz[baseline=(content.base)] \node[draw, inner sep=2pt] (content) {#1};%
}
```

`\cwff` Command to color a wff. `\cwff` or "colored well-formed formula" (wff). Usage: `\cwff[`⟨*colorname*⟩`]{formula}`. The color name defaults to blue.

```
\newcommand{\cwff}[2][blue]{%
  \textcolor{#1}{#2}%
}
```

`\sdeduce` Environment for proof that does not break (autonumbered). Usage: `\begin{sdeduce}` ... `\end{sdeduce}`

```
\newenvironment{sdeducenum}%
{\ifbool{resetdeducecounter}{\setcounter{deducecounter}{0}}{}
\begin{tabular}{!{\makebox[\ppnumwd][r]{\deducecounter}
    \hspace{\ppnumsep-\ppsep}}>{$}l<{$}@{\hspace{\ppjustindent}}l}}
{\end{tabular}}
%
\newenvironment{sdeduce}{%
  \renewcommand{\arraystretch}{1.25}%
  \begin{sdeducenum}%
}{%
  \end{sdeducenum}%
}
```

`\mdeduce` Environment for proof that does not break (manually numbered). Usage: `\begin{mdeduce}` ... `\end{mdeduce}`

```
\newenvironment{mdeducenum}{%
  \begin{tabular}{>{\scriptsize}r@{\hspace{\ppnumsep}}
    >{$}l<{$}@{\hspace{\ppjustindent}}l}%
}{%
  \end{tabular}%
}
\newenvironment{mdeduce}{\renewcommand{\arraystretch}
{1.25}\begin{mdeducenum}}{\end{mdeducenum}}
```

`\ldeduce` The `\ldeduce[`⟨*l,c,r*⟩`][`⟨*caption*⟩`][`⟨*label*⟩`]` environment is for proofs that break across the page. Takes an optional argument for alignment (argument is l, c, or r). The option defaults to l (left aligned). Usage: `\begin{ldeduce}[l]` ... `\end{ldeduce}`

```
\setlength{\LTpost}{0pt} % remove space after longtable !!!!!!!!!
\setlength{\LTpre}{0.5em} % remove space before longtable !!!!!!!!!
\makeatletter
```

```
\renewcommand{\LT@makecaption}[3]{%
  \LT@mcol\LT@cols c{\hbox to\z@{\hss\parbox[t]\LTcapwidth{% \reset@font
    \sbox\@tempboxa{#1{#2: }#3}%
    \ifdim\wd\@tempboxa>\hsize
      #1{#2: }#3%
    \else
      \hbox to\hsize{\hfil\box\@tempboxa\hfil}%
    \fi
    \endgraf}%
  \hss}}}
\makeatother

\NewDocumentEnvironment{ldeducenum}{O{l} O{} O{}}{%
\renewcommand\tablename{Proof}
  \ifbool{resetdeducecounter}{\setcounter{deducecounter}{0}}{}%
  \ifblank{#2}{%
    \begin{longtable}[#1]{!{\makebox[\ppnumwd][r]{\deducecounter}\hspace{\ppnumsep-\ppsep}}>{$l
  }{
      \begin{longtable}[#1]{!{\makebox[\ppnumwd][r]{\deducecounter}\hspace{\ppnumsep-\ppsep}}>
    }%
  }%
{%
  \end{longtable}
}
\newenvironment{ldeduce}{%
  \begin{ldeducenum}%
}{%
  \end{ldeducenum}%
}

\RequirePackage{tabularray}
\UseTblrLibrary{counter}
\newenvironment{ltdeducenum}{
  \ifbool{resetdeducecounter}{\setcounter{deducecounter}{0}}{}
  \begin{tblr}{
    colspec={Q[l]Q[l,mode=math]Q[l]},
    baseline=t,
    long,
    column{1}={font=\deducecounter},
  }
}{
  \end{tblr}
}
\DeclareFloatingEnvironment[name=Proof, placement=!htb]{proof}
\newenvironment{ltdeduce}{%
  \begin{ltdeducenum}%
}{%
  \end{ltdeducenum}%
}
```

<*package>