The logicproof package

Alan Davidson
alan.davidson@gmail.com

20 March 2014

1 Introduction

This package provides support for Fitch-style box proofs, intended to be used for proofs in predicate logic and first order logic. In this proof style, each statement of the proof is accompanied by a justification, and subproofs or lemmata within a larger proof are enclosed in boxes to separate them from the rest of the proof.

2 Usage

`logicproof` The `logicproof` environment takes a single argument, which should be a non-negative integer describing the maximum number of nested subproofs that this proof will contain. Each line of the proof consists of two main columns: the proof statement of the current line, followed by an ampersand (`&`), followed by the justification for the statement. Lines should be separated from each other by a `\\`. The column of proof statements defaults to math mode, while the justification column defaults to normal text.

If you add a `\label` within a proof, the corresponding `\ref` will be the current line number.

`subproof` Within these logic proofs, it is often necessary to make subproofs which begin by making some sort of assumption that does not necessarily hold in the broader proof. To delineate the scope of these assumptions, use the `subproof` environment, which will draw a box around the statements of the proof for which these assumptions should hold. Subproofs can be nested within each other, up to the maximum level provided as an argument to the `logicproof` environment.

Within a subproof, the format of each line is exactly the same as in the `logicproof` environment: a proof statement, followed by an ampersand (`&`), followed by a justification for the statement, with `\\` at the end of every line except the last one (which instead ends with `\end{subproof}`).

If a statement has an empty justification, it is still important to put in the `&` to separate out the column in which the justification would go, or else the right sides of the subproof boxes will be misaligned for this statement.

The previous line before a subproof begins should end in either `\\` or `\end{subproof}`.

A warning for advanced LaTeX users: although most environments make their own groups (such that changes made within an environment go out of scope at the end of it), the `subproof` environment does not! Any changes made within a `subproof` environment will not go out of scope until the enclosing `logicproof` environment ends. This was done because groups cannot cross alignment tabs, but the `subproof` environment needs to use the same alignment as the enclosing `logicproof` environment does (or else they won't line up together). An inconvenient side effect of this is that when you forget to put in a `\end{subproof}`, the error message about mismatched environments points to the line number where the enclosing `logicproof` environment started, rather than the offending `subproof` environment.

3 Example

This example proves the validity of the sequent $(p \lor q) \lor r \vdash p \lor (q \lor r)$. There are many different styles that can be used for the justification of each step; we are using the style found in *Logic in Computer Science: Modelling and Reasoning about Systems* by Huth and Ryan.

```
\begin{logicproof}{2}
  (p\lor q)\lor r & premise\\
  \begin{subproof}
    (p\lor q) & assumption\\
    \begin{subproof}
      p & assumption\\
      p\lor (q\lor r) & $\lor\mathrm{i}_1$, 3
    \end{subproof}
    \begin{subproof}
      q & assumption\\
      q\lor r & $\lor\mathrm{i}_1$, 5\\
      p\lor (q\lor r) & $\lor\mathrm{i}_2$, 6
    \end{subproof}
    p\lor (q\lor r) & $\lor$e, 2, 3--4, 5--7
  \end{subproof}
  \begin{subproof}
    r & assumption\\
    q\lor r & $\lor\mathrm{i}_2$, 9\\
    p\lor (q\lor r) & $\lor\mathrm{i}_2$, 10
  \end{subproof}
  p\lor (q\lor r) & $\lor$e, 1, 2--8, 9--11
\end{logicproof}
```

This compiles into the following:

1.	$(p \vee q) \vee r$	premise
2.	$(p \vee q)$	assumption
3.	p	assumption
4.	$p \vee (q \vee r)$	$\vee \mathrm{i}_1$, 3
5.	q	assumption
6.	$q \vee r$	$\vee \mathrm{i}_1$, 5
7.	$p \vee (q \vee r)$	$\vee \mathrm{i}_2$, 6
8.	$p \vee (q \vee r)$	$\vee$e, 2, 3–4, 5–7
9.	r	assumption
10.	$q \vee r$	$\vee \mathrm{i}_2$, 9
11.	$p \vee (q \vee r)$	$\vee \mathrm{i}_2$, 10
12.	$p \vee (q \vee r)$	$\vee$e, 1, 2–8, 9–11

4 Advanced Configuration

There are two lengths that can be configured manually if desired.

`\subproofhorizspace` The horizontal distance between the sides of nested boxes is the length stored in `\subproofhorizspace`. If you have many nested subproofs, it might be desirable to make this space larger so that it is easier to distinguish between them.

`\intersubproofvertspace` An extra vertical space of length `\intersubproofvertspace` is added after each statement in the proof. The intention here is to have this much space between consecutive subproofs. This is to say, when one subproof ends and the next subproof begins immediately, this is the distance between the bottom edge of the ending subproof and the top edge of the beginning subproof. This space is inserted after each line, however, so that the vertical spacing between them is consistent regardless of whether subproofs are begun or ended. If you want more compact proofs and you never use consecutive subproofs, it might be useful to reduce or remove this extra space.

5 Constraints

The environments in this package might not work properly if any of the following constraints are violated. These should not be difficult burdens; they are listed here mainly for completeness.

- Each proof and each subproof must be at least 1 statement long.
- Each subproof must start with at least 1 statement before containing another subproof inside itself. You should not have a subproof that immediately begins with a nested subproof.
- The end of each subproof must have at least 1 statement (or a new subproof) following it; a single proof statement cannot end multiple subproofs at once.

Consequently, the last statement of the top-level proof must be outside of all subproofs. It is okay to immediately start a new subproof after ending a previous subproof.

6 Summary of Implementation

The `logicproof` environment is built on the `tabular` environment. It has a column for the line number, a column for each possible nested subproof, a column for the statement, a column for the justification, and then a second column for each possible nested subproof. The columns corresponding to nested subproofs either contain a `\vline` if the subproof is currently being used, or nothing if the subproof is not being used. When a subproof is begun or ended, a `\cline` is used to draw horizontal lines between the columns corresponding to that subproof. In order to prevent multiple `\cline`'s from overlapping when one subproof is ended and another is immediately begun, each statement in the proof actually ends with a negative vertical space backing up to the previous line, then a space down `\intersubproofvertspace` and a redrawing of the subproof lines again to cover the extra space.

7 Implementation

```
\RequirePackage{array}
\RequirePackage{ifthen}
```

`\subproofhorizspace` `\intersubproofvertspace` We allow the user to configure the horizontal and vertical distance between the edges of the boxes.

```
\newlength{\subproofhorizspace}
\setlength{\subproofhorizspace}{0.5em}
\newlength{\intersubproofvertspace}
\setlength{\intersubproofvertspace}{0.333em}
```

We use a variety of counters to keep track of the state.

```
\newcounter{lp@line}% Current line number on the proof
\newcounter{lp@nested}% Number of nested subproofs we're currently in
\newcounter{lp@total@nests}% Maximum number of nested subproofs allowed
\newcounter{lp@cline@1}% Used to draw horizontal lines in subproofs
\newcounter{lp@cline@2}% Also used to draw horizontal lines in subproofs
\newcounter{lp@temp}% Temporary storage counter
```

`logicproof` Use this to make a propositional logic proof. The argument it takes is the maximum number of nested subproofs you will use.

```
\newenvironment{logicproof}[1]{%
  \setcounter{lp@line}{0}%
  \setcounter{lp@nested}{0}%
  \setcounter{lp@total@nests}{#1}%
  \setlength{\tabcolsep}{0mm}%
```

When using the array package, the tabular environment contains the statement `\let\\\@arraycr` (note that with the array package, `\@tabularcr` is replaced with `\@arraycr` even within the tabular environment). So, to modify the behavior of `\\`, we're actually going to modify `\@arraycr`. Save a copy of the original definition first, so that we can use it inside our new definition. Remember that when the logicproof environment finishes, this redefinition will go out of scope and revert to the previous version, so we won't ruin any future uses of the tabular environment.

```
\let\lp@orig@arraycr\@arraycr%
\renewcommand{\@arraycr}{\lp@cr}%
```

Get labels to work in proofs by defining `\@currentlabel` to always be the line number, regardless of where we are in the proof. Note that the usual approach of using `\refstepcounter{lp@line}` doesn't work because it goes out of scope by the time we get to the next cell in the tabular environment.

```
\renewcommand{\@currentlabel}{\p@lp@line\thelp@line}%
```

If the maximum number of nested subproofs is 0, we need a slightly different column format, because the array environment doesn't like it when you repeat a formatting group 0 times.

```
\ifthenelse{%
  0=#1%
}{%
  \def\lp@tab@format{{r@{~~~}>{$}l<{$}@{~~~~}l}}%
}{%
```

Although we could use the `array` package's `>{...}` and `<{...}` features to have automatic placement of the vertical lines on the sides of subproofs, we would not be able to get the horizontal lines at the tops and bottoms of the subproofs to line up properly. Consequently, we go with the "old school" approach of putting subproofs in their own columns, so that we can use `\cline` to put the horizontal lines in their proper places.

```
  \def\lp@tab@format%
      {{r@{~~~}*{#1}{l}@{~}>{$}l<{$}@{~~~~}l@{~}*{#1}{r}}}
}%
\center%
```

We use the `tabular` environment instead of the `array` environment because we want to be able to have labels on individual lines. Since the entirety of the `array` environment is in math mode, it does not support more than one label per array.

```
\expandafter\tabular\lp@tab@format%
\lp@start@proof@line%
}{%
\lp@stop@proof@line%
\endtabular%
\endcenter%
```

To ensure that no one tries using the subproof environment outside of the logicproof environment, set the maximum number of nested subproofs to 0.

```
\setcounter{lp@total@nests}{0}%
```

Finally, make sure that all open subproofs have been closed. We do this last because if a subproof is still open, we need to set `\@currenvir` properly for `\end` to check and throw errors on, but previous commands have `\endgroup`'s that make it revert to previous definitions.

```
\ifthenelse{%
  0=\value{lp@nested}
}{% All is well.
}{% There are still open subproofs.
  \def\@currenvir{subproof}%
}
}
```

`subproof` This environment puts a box around the lines of the proof within it. It should come right after either a `\\` or a `\end{subproof}`.

```
\newenvironment{subproof}{%
```

Make sure we don't start more nested subproofs than the current logicproof environment can handle.

```
  \ifthenelse{%
    \value{lp@total@nests}>\value{lp@nested}%
  }{% All is well; don't do anything.
  }{%
    \PackageError{logicproof}{Too many nested subproofs!}{%
      Increase the maximum number of nested subproofs allowed
      in the current logicproof environment.%
    }%
  }%
```

The `\begin` and `\end` parts of an environment start and end a group, so that macros defined within them have local scope. However, a group cannot cross alignment tabs (`&`'s), which means that this subproof environment, which must cross them, needs to get rid of those extra groups first. So, we immediately end the group that `\begin` created before going on with the subproof. Note that this means any redefinitions of any macros we might have will persist outside this subproof and will not go out of scope until the entire logicproof environment is over. Note also that this approach is slightly brittle: if the implementation of `\begin` and `\end` ever changes, this subproof environment is likely to break.

```
  \endgroup%
  \lp@stop@proof@line%
```

Ideally, we'd use `\lp@extend@space` here. However, we first need to end the current line, which means putting in `\lp@orig@arraycr` and then going up an extra line in the tabular environment via `\lp@add@space`.

```
  \lp@orig@arraycr%
  \lp@add@space%
  \lp@go@up@a@line%
  \stepcounter{lp@nested}%
  \lp@cr@clines%
```

The current line number was added in before this subproof was started. Do not add it in again now; just skip over the line number entry and go straight on to the subproof-drawing stuff.

```
  &%
  \lp@continue@proof@line%
}{%
```

If we try ending a subproof that has not yet begun, we will run into trouble with `\cline` trying to draw a horizontal line to a column past the end of the tabular environment. This happens before `\end` actually checks whether we're ending the right environment. In order to get a more useful error message, we first check that there is at least 1 open subproof.

```
  \ifthenelse{%
    0<\value{lp@nested}%
  }{% All is well; don't do anything.
  }{%
    \PackageError{logicproof}{Cannot end a subproof before it begins}{%
      You must have a \protect\begin{subproof} before you can use %
      \protect\end{subproof}.%
    }%
  }%
  \lp@stop@proof@line%
  \lp@cr@clines%
  \addtocounter{lp@nested}{-1}%
  \lp@extend@space%
  \lp@start@proof@line%
```

Now that we're done with the subproof, we need to create a group because `\end` is expecting us to still be in the group that was started in `\begin`. We also must redefine `\@currenvir` within that group, or the error-checking in `\end` will think we've ended the wrong environment (its previous redefinition went out of scope when we ended the group created by `\begin`).

```
  \begingroup%
  \def\@currenvir{subproof}%
}
```

`\lp@cr` This is what the `\\` will be defined as inside the logicproof environment.

```
\newcommand{\lp@cr}{%
  \lp@stop@proof@line%
  \lp@orig@arraycr%
  \lp@extend@space%
  \lp@start@proof@line%
}
```

`\lp@go@up@a@line` This moves up one entire line in the proof.

```
\newcommand{\lp@go@up@a@line}{%
  \vspace{-\ht\@arstrutbox}%
  \vspace{-\dp\@arstrutbox}%
  \vspace{-\intersubproofvertspace}%
}
```

`\lp@add@space` Extends the vertical lines at the sides of the proof down slightly, so that the horizontal lines at the end of the previous subproof and the start of the next one don't overlap. This basically inserts a blank row in the proof (no line number, no statement, no justification; just the subproof lines), then backs up part of a line.

```
\newcommand{\lp@add@space}{%
  \lp@extend@space%
```

The `\@arstrutbox` is a box containing the minimum array height. Remember that the height of the strut is spread between the height above the baseline and the depth below it!

```
  \vspace{-\ht\@arstrutbox}%
  \vspace{-\dp\@arstrutbox}%
```

Uncommenting this would line things up exactly where they started.

```
  %\vspace{-\intersubproofvertspace}%
}
```

`\lp@extend@space` This extends the vertical lines at the sides of the subproofs down by an extra `\intersubproofvertspace`. This is done so that two subproofs in a row don't have their horizontal lines overlap each other.

```
\newcommand{\lp@extend@space}{%
  \vspace{-\ht\@arstrutbox}%
  \vspace{-\dp\@arstrutbox}%
  \vspace{\intersubproofvertspace}%
```

Now, insert a row that has vertical lines for the subproofs but no line number, proof statement, or justification.

```
  &%
  \lp@continue@proof@line%
  &%
  \lp@stop@proof@line%
  \lp@orig@arraycr%
}
```

`\lp@amper` You can't have a `&` in a `\whiledo` loop, but this works instead.

```
\newcommand{\lp@amper}{&}
```

`\lp@start@proof@line` This macro does everything on a proof line before the statement itself: it increments and prints the line number, then calls `\lp@continue@proof@line` to put in the vertical lines of any subproofs we're currently in.

```
\newcommand{\lp@start@proof@line}{%
```

We use `\stepcounter` instead of `\refstepcounter` here because the changes made by `\refstepcounter` to how labels get made would go out of scope by the time we got to the next cell of the tabular environment (i.e., 2 lines from here). Instead, we redefined `\@currentlabel` at the beginning of the environment to always contain the current value of the `lp@line` counter.

```
  \stepcounter{lp@line}%
  \arabic{lp@line}.%
```

```
  &%
  \lp@continue@proof@line%
}
```

`\lp@continue@proof@line` This macro makes the vertical lines of the subproof boxes on the left side of the proof (i.e., the ones that come between the line numbers and the proof statements). We use `\lp@amper` here because the raw `&` token doesn't play well with `\whiledo` loops.

```
\newcommand{\lp@continue@proof@line}{%
  \setcounter{lp@temp}{0}%
  \whiledo{\value{lp@temp}<\value{lp@nested}}{%
    \vline%
    \hspace*{\subproofhorizspace}%
    \lp@amper%
    \stepcounter{lp@temp}%
  }%
  \whiledo{\value{lp@temp}<\value{lp@total@nests}}{%
    \hspace*{\subproofhorizspace}%
    \lp@amper%
    \stepcounter{lp@temp}%
  }%
}
```

`\lp@stop@proof@line` This macro makes the vertical lines of the subproof boxes on the right side of the proof (i.e., the ones that come after the justifications for each step). It basically does the same thing as `\lp@continue@proof@line`, but in reverse order. We use `\lp@amper` here because the raw `&` token doesn't play well with `\whiledo` loops.

```
\newcommand{\lp@stop@proof@line}{%
  \whiledo{\value{lp@temp}>\value{lp@nested}}{%
    \addtocounter{lp@temp}{-1}%
    \lp@amper%
    \hspace*{\subproofhorizspace}%
  }%
  \whiledo{\value{lp@temp}>0}{%
    \addtocounter{lp@temp}{-1}%
    \lp@amper%
    \hspace*{\subproofhorizspace}%
    \vline%
  }%
}
```

`\lp@subtract@from@counter` Subtraction by a value in another counter is annoying, but with this we can use `\expandafter` and make it easier.

```
\newcommand{\lp@subtract@from@counter}[2]{%
  \addtocounter{#2}{-#1}%
}
```

`\lp@set@clines` This macro sets up where the horizontal bars go for subproofs.

```
\newcommand{\lp@set@clines}{%
```

lp@cline@1 = lp@nested + 1

```
  \setcounter{lp@cline@1}{\value{lp@nested}}%
  \stepcounter{lp@cline@1}%
```

lp@cline@2 = 2 * lp@total@nests + 4 - lp@nested

```
  \setcounter{lp@cline@2}{\value{lp@total@nests}}%
  \addtocounter{lp@cline@2}{\value{lp@total@nests}}%
  \addtocounter{lp@cline@2}{4}%
```

Subtracting one counter from another is tricky. We need to expand the value of the counter being subtracted first.

```
  \expandafter\lp@subtract@from@counter\expandafter{%
      \value{lp@nested}}{lp@cline@2}%
}
```

`\lp@cr@clines` This macro goes to the next line of the tabular environment and puts in a horizontal line for the beginning or end of the current subproof.

```
\newcommand{\lp@cr@clines}{%
  \lp@set@clines%
```

We put the `\lp@orig@arraycr` here instead of in `\lp@stop@proof@line` because `\cline` doesn't seem to work properly unless it's right after the carriage return. Even moving it up above `\lp@set@clines` on the previous line messes it up.

```
  \lp@orig@arraycr%
  \cline{\value{lp@cline@1}-\value{lp@cline@2}}%
}
```