The philprop package, version 1.0

Jason Turner

February 29, 2008

Contents

1 Overview

Philosophers tend to use numbered (or sometimes named) propositions in their writing — numbered or named, extracted text that they can later refer back to. The propositions are generally numbered consecutively.

LaTeX's `list` environments do a nice job of setting such propositions; for instance

```
\begin{itemize}
\item[(1)] This is an important thesis I will be talking about
later.
\end{itemize}
```

inside of a document will be set as:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc sit amet arcu sed enim consequat dapibus.

(1) This is an important thesis I will be talking about later.

Praesent ac dui eget urna placerat elementum. Suspendisse ac sapien. Vestibulum vestibulum volutpat orci. Nullam magna. Donec sollicitudin laoreet dui.

Unfortunately, the automatic numbering of these list environments resets each time the environment `end`s. This means that authors have to put in each proposition number 'by hand', as it were, by telling the `\item` command exactly what label to use (as in the example above). This means the proposition labels cannot be captured with the `\label` and `\ref` commands, so authors have to keep track of just what numbers were used for which propositions, and go through and change all those numbers if a proposition is added to or deleted from the middle of a paper. Likewise (since `\label` and `\ref` only keep track of counters, not written-out item labels), authors have to keep track of named propositions as well — so if they decide to change the name of a proposition, they have to change every place to which it is referred, also.

This package aims to provide tools for authors who use numbered or named propositions — tools that make LaTeXkeep track of these labels so that authors don't have to.

2 The Package

The heart of the philprop package is a new list environment, `prop`. All numbered or labeled propositions are produced using this environment, and it keeps track of proposition numbering throughout the document.

2.1 Environment lenghts

The `prop` environment uses the following lengths, corresponding to the lengths used by the `list` environment:

`list` **length**	`prop` **length**	**what it does**
`\topsep`	`\proptopsep`	amount of extra vertical space at top of list
`\partopsep`	`\proppartopsep`	extra length at top if environment is prececed by a blank line (it should be a rubber length)
`\itemsep`	`\propitemsep`	amount of extra vertical space between item
`\parsep`	`\propparsep`	amount of vertical space between paragraphs within an item
`\leftmargin`	`\propleftmargin`	horizontal distance between the left margins of the environment and the list; must be nonnegative
`\rightmargin`	`\proprightmargin`	horizontal distance betwen the right margins of the enviroment and the list; must be nonnegative
`\listparindent`	`\propparindent`	amount of extra space for paragraph indent after the first in an item; can be negative
`\itemindent`	`\propitemindent`	indentation of first line of an item; can be negative
`\labelsep`	`\proplabelsep`	separation between end of the box containing the label and the text of the first line of an item
`\labelwidth`	`\proplabelwidth`	normal width of the box containing the label; if the actual label is bigger, the natural width is used, extending into the space for the first line of the item's text

2.2 Counters

Within the `prop` environment, the `propositions` counter is used to produce numbers for the numbered propositions. However, a second counter — `allprops` — keeps track of the number *between* environments, and when a new `prop` environment is started, tells `propositions` where to start counting from. Thus

```
\begin{prop}
\item This is one item.
\item This is another.
\end{prop}
And there you see two items\ldots
\begin{prop}
\item Here is yet a third.
\end{prop}
```

produces the text:

> (1) This is one item.
>
> (2) This is another.
>
> And there you see two items...
>
> (3) Here is yet a third.

Although the `propositions` counter is used within the `prop` environment, since list environments by default reset their counters when they begin, the separate counter, `allprops`, is used outside of the environment to keep running track of the proposition number to be started whenever a new `prop` environment is called. So if you want to reset the numbering at some point in your document (at the beginning of a new chapter, for instance), it is the `allprops` counter that you need to reset.

2.3 Item and ref commands

You can use the `\item` command for items within a `prop` environment. But the `philprop` package comes with a number of specific `\item=`like commands

for use with the `prop` environment, and an integrated reference command, `\pref`, for cross-referencing these propositions throughout the document.

Each of the specialized `\item`-like commands creates (or refers to) a *key* — a (case sensitive) string of alpha-numerics that you can use to cross-reference the proposition later. This key is like that you create with the `\label{`⟨*key*⟩`}` command (and, in fact, in some cases is driven by the `\label` command), and if you create keys for each proposition, you can do cool things with them later.

Basic item types

Every item created with one of the packages' specialized `\item`-like commands is assigned one of two types: **named** or **numbered**. The primary commands for these are `\pitem{`⟨*key*⟩`}`, for numbered propositions, and `\nitem{`⟨*key*⟩`}{`⟨*label*⟩`}` for named ones. FA third command uses no tag, because it produces no label: the `\qitem` command produces a 'quiet', unlabelled item.

For instance, the code

```
\begin{prop}
\pitem{First} This is the first numbered proposition.
\nitem{FirstNamed}{F} This is the first named proposition.
\pitem{Second} This is the second numbered proposition.
\qitem This is a 'quiet' item.
\end{prop}
```

produces the output:

(1) This is the first numbered proposition.

(F) This is the first named proposition.

(2) This is the second numbered proposition.

This is a 'quiet' item.

The keys *First*, *FirstNamed*, and *Second* are each now tagged to the labels (1), (F), and (2), respectively.

\pref

The `\pref{`⟨*key*⟩ command produces, in the text, the proposition label associated with the given key. For instance,

```
The first numbered proposition was \pref{First}, the second was
\pref{Second}, and the first named proposition was \pref{FirstNamed}.
```

produces

> The first numbered proposition was (1), the second was (2), and the first named proposition was (F).

Note that the cross-referencing is done via the production of an .aux file, *jobname*`.prp`; in order to get the cross-references right, a document will need to be LaTeXed at least twice. Also, assigning a single key to multiple propositions will prompt the package to give you a warning and make the `\pref` command unreliable.

Referencing propositions

Sometimes you will want to produce a new proposition with the same name or number as an old one. In this case, the `\refitem` command can be used to produce an item that re-creates the label of an earlier item. For instance, the code

```
Recall the earlier proposition:
\begin{prop}
\refitem{First} This is the first numbered proposition.
\end{prop}
```

produces the text:

> Recall the earlier proposition:
>
> (1) This is the first numbered proposition.

This can be useful — if you later add a proposition before *First* to your document, the label will change automatically in both places. (Note that this works just as well for named propositions as well.) There is a more useful form of this command, `\refprop`, that will be discussed below.

A similar command, `\primeitem`, lets you produce variants of old propositions. It is sometimes useful to 'modify' an earlier proposition, and to mark the modification by production of a proposition with the same name and some sort of marker, such as an asterisk or a prime. The `\primeitem` command, syntax

`\primeitem[`⟨*symbol*⟩`]{`⟨*oldkey*⟩`}{`⟨*newkey*⟩`}`

produces a new item based on an old proposition, where *symbol* is the symbol used to mark the proposition's label (asterisk, prime, etc.), *oldkey* is the key of the old proposition being modified, and *newkey* is the key of the new proposition created. (The *symbol* argument is optional; if left out, the command produces a prime symbol, '′'.) For instance, the code:

```
\begin{prop}
\pitem{One} This is a rather important proposition.
\primeitem[$^{\ast}$]{One}{OnePrime} This is a \emph{quite}
important proposition.
\end{prop}
```

produces the text

> (3) This is a rather important proposition.
>
> (3*) This is a *quite* important proposition.

Note two things. First, the new 'prime' proposition gets its own key, so that it can (a) be referenced later by `\pref{`⟨*newkey*⟩`}`. Second, the new key can itself be sent to a `\primeitem` command, for making 'doubly prime' items (e.g., (3**).) And the original key can be used to make a different `\primeitem` too (e.g., `One` can be used with another `\primeitem` to create a (3′) proposition.)

The new primeitem inherits the item-type associated with the ⟨*oldkey*⟩ used to create it. So (3*) from above will be typed as numbered. On the other hand, a `\primeitem` created from a key set by a `\nitem` command will iself by typed as named.

2.4 Item label appearance

By default, the `prop` environment and the `\pref` command are set to produce item labels between round braces '(' and ')'. This behavior can be altered.

The philprop package keeps track of four sets of braces: those to appear around the label of a numbered item in the prop environment, those to appear around the label of a named item in the prop environment, those to appear around an in-text reference (produced by \pref) to a numbered item, and hose to appear around an in-text reference to a named item.

The braces are set by the following four commands:

\pitembraceset{⟨*lbrace*⟩}{⟨*rbrace*⟩}	braces around numbered items in prop environment.
\nitembraceset{⟨*lbrace*⟩}{⟨*rbrace*⟩}	braces around named items in prop environment.
\pitembraceoutset{⟨*lbrace*⟩}{⟨*rbrace*⟩}	braces around \pref references to numbered items.
\nitembraceoutset{⟨*lbrace*⟩}{⟨*rbrace*⟩}	braces around \pref references to numbered items.

For example, suppose I wanted the labels of numbered items to be followed by a period, the labels of named items to appear as-is, the in-text references to numbered items to be surrounded by round brackets, and the in-text references to named items to be surrounded by square brackets. Then I could use

```
\pitembraceset{}{.}
\nitembraceset{}{}
\pitembraceoutset{(}{)}
\nitembraceoutset{[}{]}
\begin{prop}
\pitem{numbered} This is a numbered proposition.
\nitem{named}{NP} This is a named proposition.
\end{prop}
We can refer to these as \pref{numbered} and \pref{named},
respectively.
```

to produce

4. This is a numbered proposition.

NP This is a named proposition.

We can refer to these as (4) and [NP], respectively.

These commands can be called anywhere in the document, and will change the behavior of the braces from that point in the document onward. Note also that, since `\primeitem`s inherit their proposition type, the same type of braces that appear around an item's label will appear around the label of a `\primeitem` created from it.

2.5 Proposition commands

Along with the five item types (`\pitem`, `\nitem`, `\quitem`, `\primeitem`, `\refitem`), the package provides five corresponding proposition commands:

`\pprop{`⟨*key*⟩`}{`⟨*proposition text*⟩`}`
`\nprop{`⟨*key*⟩`}{`⟨*label*⟩`}{`⟨*proposition text*⟩`}`
`\qprop{`⟨*proposition text*⟩`}`
`\primeprop[`⟨*symbol*⟩`]{`⟨*oldkey*⟩`}{`⟨*newkey*⟩`}{`⟨*label*⟩`}`
`\pprop{`⟨*key*⟩`}{`⟨*proposition text*⟩`}`

The various proposition commands work just like the corresponding item commands, but for two exceptions. First, they can be used outside of the `prop` environment. (In fact, the command calls the environment and the environment does not nest well, so they should *not* be used inside of it.) Second, the proposition commands link the text of the proposition to the key, which makes possible the use of a *further* command,

`\fullrefprop{`⟨*key*⟩`}`

which reproduces the original proposition created with that key. Example:

```
\pprop{remember}{For every proposition this very important, it
will be extremely important to be able to reproduce it later on.}
Then we go on to say some other stuff here\ldots And now we remind
ourselves:
\fullrefprop{remember}
```

will produce:

(5) For every proposition this very important, it will be extremely important to be able to reproduce it later on.

Then we go on to say some other stuff here... And now we remind ourselves:

(5) For every proposition this very important, it will be extremely important to be able to reproduce it later on.

The command, of course, works just as well with propositions made using `\nprop` and `\primeprop`.

2.6 Keeping propositions with long labels flush.

The default behavior of `philprop` pushes long labels into the first line of the text, like so:

(KETSUP) The Keep Every Tree Set Up Properly principle: Make sure every tree you've got is properly set up.

Some find this ugly, and want the entire proposition indented flush with the first line, as follows:

(KETSUP) The Keep Every Tree Set Up Properly principle: Make sure every tree you've got is properly set up.

The latter behavior can be achieved with the `\longprop` command. (In the example, it was produced with

```
\longprop{LongKetsup}{KETSUP}{The Keep Every Tree Set Up
Properly principle: Make sure every tree you've got is properly
set up.}
```

There are also long versions of the `refprop` and `fullrefprop` commands: `\longpropref` and `\longpropfullref`, respectively, which work just like `\refprop` and `\fullrefprop` but for the margins. (Note, however, that these commands will only work for *named* propositions; the package may behave unpredictably if you combine these commands with keys for numbered propositions.)

2.7 The .prp file

The package allows for references that occur before their keys are set, e.g.:

```
We will soon define a proposition named \pref{early}.
\nprop{early}{EK}{This proposition was referred to before it was
produced.}
```

produces

> We will soon define a proposition named (EK).
>
> (EK) This proposition was referred to before it was produced.

The references are stored in the auxiliary file *jobname*`.prp`, and if are in advance of their declaration are not set properly on the first run after a change. You will need to LaTeXthe document (at least) twice in order to ensure that all of the `propref` references are set right.

3 Hyperref support

Version 1.0 does not include any support for the `hyperref` package (except for a little bit natively embedded by the package's use of LaTeX's `\label` and `\ref` commands. I hope to incorporate `hyperref` support in the near future.