# The pf2 Package

Leslie Lamport

7 October 2011

## Contents

## 1 Introduction

The `pf2` style provides commands for typesetting the kind of hierarchically structured proofs described in [1, 2]. Here is a tiny nonsensical example of such a proof and the commands that produced it. The vertical spacing of the typeset proof has been exagerated to make it easy to compare the proof and the LaTeX source.

1. $2 = 8/4$

   PROOF: By Thm. 1.2.

2. $8 = 2\sqrt{16}$

   2.1. $8 = 4 + 4$

   PROOF: Obvious.

   2.2. $4 = \sqrt{16}$

   2.2.1. $4 \cdot 4 = 16$

   PROOF: Step 2.1.

   2.2.2. Q.E.D.

   PROOF: By 2.2.1.

   2.3. Q.E.D.

   PROOF: 2.2 and 1

3. Q.E.D.

   PROOF: By 2

```
\begin{proof}
\step{1}{$2 = 8/4$}
  \begin{proof}
  \pf\ By Thm.\ 1.2.
  \end{proof}
\step{1a}{$8 = 2\sqrt{16}$}
  \begin{proof}
  \step{1-1}{$8=4+4$}
    \begin{proof}
    \pf\ Obvious.
    \end{proof}
  \step{1-2}{$4=\sqrt{16}$}
    \begin{proof}
    \step{1-2-a}{$4\cdot4=16$}
      \begin{proof}
      \pf\ Step \stepref{1-1}.
      \end{proof}
    \qedstep
      \begin{proof}
      \pf\ By \stepref{1-2-a}.
      \end{proof}
  \end{proof}
  \qedstep
     \begin{proof}
     \pf\ \stepref{1-2} and \stepref{1}
     \end{proof}
  \end{proof}
\qedstep
  \begin{proof}
  \pf\ By \stepref{1a}
  \end{proof}
\end{proof}
```

Observe that a `\begin{proof}` command starts a new level of proof that is ended by the corresponding `\end{proof}` command. A numbered step is produced by a `\step` command, whose first argument is an arbitrary label and whose second argument produces the text of the step. A reference to the step number of the step with label `1a` is produced by the command `\stepref{1a}`. A `\qedstep` command produces a numbered Q.E.D. step. (The Q.E.D. step has no label because one never refers to its number in a proof.) The last step in any (sub)proof should be a Q.E.D. step, but this is not enforced by the `pf2` commands. A step need not have a proof.

The proof above uses the long numbering style of proofs. It is specified by the `\pflongnumbers` declaration. The short numbering style, which is the default, is specified by the `\pfshortnumbers` declaration. It produces the following numbering.

$\langle 1\rangle 1.$ $2 = 8/4$
  PROOF: By Thm. 1.2.
$\langle 1\rangle 2.$ $8 = 2\sqrt{16}$
  $\langle 2\rangle 1.$ $8 = 4 + 4$
    PROOF: Obvious.
  $\langle 2\rangle 2.$ $4 = \sqrt{16}$
    $\langle 3\rangle 1.$ $4 \cdot 4 = 16$
      PROOF: Step $\langle 2\rangle 1$.
    $\langle 3\rangle 2.$ Q.E.D.
      PROOF: By $\langle 3\rangle 1$.
  $\langle 2\rangle 3.$ Q.E.D.
    PROOF: $\langle 2\rangle 2$ and $\langle 1\rangle 1$
$\langle 1\rangle 3.$ Q.E.D.
  PROOF: By $\langle 1\rangle 2$

It would impossible to keep track of all the step labels in a proof if you had to make them up yourself. The easiest thing to do is to let the step labels be the actual printed step numbers. Typing those numbers yourself would be an impossible task, since they change as you modify the proof. However, it's easy to do with the *pfnum* program. When you write the proof of step 2, start numbering the steps in some simple way—for example: 2.1, 2.2, 2.3. When you find that you need to add a step between steps 2.2 and 2.3, label it something like 2.2$a$. Running *pfnum* on your document will renumber the steps so they are the same as in the printed output, making the appropriate changes to `\stepref` commands. More precisely, you have to tell the *pfnum* program whether to use long or short numbers; the sensible thing to do is to tell it to number them the same way you tell LaTeX to print the numbers.

The *pfnum* program can be downloaded from:

http://research.microsoft.com/en-us/um/people/lamport/latex/pfnum.html

Note that short numbers are not unique within a proof, $\langle 2\rangle 3$ being the number of the third step of any level-2 proof. At any point in the proof, it is legal to refer to at most one step numbered $\langle 2\rangle 3$. LaTeX will generate a warning if a `\stepref` command refers to a step label where no step with that label can legally be referenced. (The same step label can be used to label different steps if it's never legal to refer to both steps from the same place in the proof.)

## 2 Text-Producing Commands

### 2.1 Types of Steps

The following commands produce particular kinds of proof steps. They are most often used as the second argument of a `\step` command, but you might want to use some of them in a proof outside any step. See the **Let** construct for an example.

#### Assume / Prove

2.3. ASSUME: There is a smallest purple number $n$.
PROVE: $n+1$ is puce.

```
\step{2.3}{ \assume{There is
                          ... $n$.}
            \prove{$n+1$ ... } }
```

#### Suffices

2.3. SUFFICES: There is a smallest purple number $n$.

```
\step{2.3}{ \suffices{There is
                          ... $n$.} }
```

#### Suffices Assume / Prove

2.3. SUFFICES ASSUME: $n > 0$
PROVE: $n+1 > 1$

```
\step{2.3}{ \sassume{$n > 0$}
            \prove{$n+1>1$} }
```

#### Case

2.3. CASE: There is no smallest purple number.

```
\step{2.3}{ \case{There ...
                     number.} }
```

**Let**

| | |
|---|---|
| 2.3. LET: Let $n$ be the smallest purple number. | `\step{2.3}{ \pflet{Let ... number.} }` |

Here is an example of the **Let** construct used outside a step.

| | |
|---|---|
| 2.3. $y > 17$ | `\step{2.3}{$y > 17$}` |
| LET: $n = 2\sqrt{y}$ | `\pflet{$n = 2\sqrt{y}$\\` |
| $m = n + 1$ | `$m = n+1$ }` |
| 2.4. $n + m > 17$ | `\step{2.4}{$n+m>17$}` |

**Define**

| | |
|---|---|
| 2.3. DEFINE: $n = 2\sqrt{y}$ | `\step{2.3}{ \define{$n=...$\\` |
| $m = n + 1$ | `$m=...$} }` |

## 2.2 Keywords

The `pf2` package defines commands for producing some commonly used "keywords" in proofs—for example, in Section 1, you saw that the `\pf` command produces the keyword "PROOF:". It's good to use these commands rather than just typing the keywords because you can change the appearance of all of them by just changing the keyword style. Here are the commands and what they produce in the default style.

| | |
|---|---|
| `\pf` | PROOF: |
| `\pfsketch` | PROOF SKETCH: |
| `\qed` | □ |
| `\pick` | PICK |
| `\pfnew` | NEW |

Section 4.3 explains how to change the style of the keywords.

## 2.3 Proof Numbers

Sometimes you might want to refer to a step number from somewhere in the document where you can't use a `\stepref` command. The two common cases are:

- When discussing the proof in text that lies outside the proof itself.

- When preceding a proof with a proof sketch, as in the following example. (Vertical space has been added to the typeset proof to make the correspondence between the output and the source easier to see.)

$\vdots$

⟨1⟩4. $A = A$

Proof sketch: The key step in this proof is ⟨2⟩3.

$\vdots$

⟨2⟩3. $A \neq \neg A$

```
  ⋮
\step{<1>3a}{$A=A$}
\begin{proof}
  \pfsketch\ The key step
   ... is \pfref{lab3a.3}.
    ⋮
  \step{<2>3}{$A \neq \lnot A$}
  \pflabel{lab3a.3}
```

A `\pflabel` command can be placed after the `\step` command or in its second argument. The command refers to the name of the most recent step at the same or higher proof level as the place where the command appears.

Because short numbers like ⟨2⟩3 are not unique within a proof, you can't use them to refer to proof steps from outside the proof. The `\pfsidenumbers` declaration can be used to print the long step numbers as *side numbers*—for example:

$\vdots$

2 ⟨1⟩2. $A = A$

Proof sketch: The key step in this proof is ⟨2⟩3.

$\vdots$

2.3 ⟨2⟩3. $A \neq \neg A$

The syntax of the command is `\pfsidenumbers{`$n$`}{`$d$`}`, where $n$ is an integer and $d$ is a length. This command prints the long numbers of all proof steps of all proof levels $n$ and higher, left-aligned a distance $d$ to the left of the left margin. This example was typeset with the command `\pfsidenumbers{1}{1em}`, though a first argument of $-42$ would have produced the same effect. The `\pfsidenumbers` declaration has the usual scoping rules, so if you put it inside a subproof it applies only to steps in that subproof. The declaration `\pfnosidenumbers` (the default) causes side numbers not to be printed.

The `\pflonglabel` command allows you to refer to these side numbers symbolically. It is the same as the `\pflabel` command, except a corresponding `\pfref` command always produces the long step number, regardless of the numbering style used in the proof.

### 2.4 Lists

The `pfenum` environment is like the `enumerate` environment, except that there is no extra space between items, and item numbers are flush-left against the prevailing left margin. It can be used to list assumptions in a proof step, as in:

1.2. ASSUME: 1. $x \in S$
                    2. $y > x$
       PROVE: $y \in S$

```
\step{1.2}{\assume{\begin{pfenum}
                     \item $x \in S$
                     \item $y > x$
                   \end{pfenum}
           \prove {$y \in S$}}
```

Nested `pfenum` environments work properly for structured assumptions. Note that the `\label` and `\ref` commands do the right thing for referring to such assumptions.

1. 1. a. $x \in S$
          b. $x > 0$
    2. $y > x$
    PROOF: The key is assumption 1b.

```
\step{1}{
 \begin{pfenum}
  \item \begin{pfenum}
         \item $x \in S$
         \item $x > 0$  \label{the.key}
        \end{pfenum}
  \item $y > x$
 \end{pfenum}}
  \begin{proof}
  \pf\ ... assumption \ref{the.key}.
  \end{proof}
```

The `pfenum*` environment is like the `pfenum` environment except that items are indented. It is appropriate for use in paragraph proofs, where you want items to be indented but without the space between them produced by the normal `enumerate` environment.

PROOF: We use
  1. Fact 1
  2. Fact 2
as follows ...

```
\begin{proof}
  \pf\ We use
     \begin{pfenum*}
       \item Fact 1
       \item Fact 2
     \end{pfenum*}
  as follows ...
```

The amount of indentation of item numbers in the `pfenum*` environment is determined by the length parameter `\pfenumindent`.

## 3 Showing Part of a Proof

### 3.1 Writing Two Versions of a Proof

It's often useful to have two versions of a proof: a more detailed *main* proof whose purpose is to convince the reader that the theorem is correct, and a less detailed *short*. For example, the shorter one might be published in a conventional journal and the longer one put on the Web.

In a two-version proof, each subproof that has two versions is produced with a `proof*` environment instead of a `proof` environment. In that environment, a `\mainproof` command separates the short proof from the main one. For example, a step having two proofs might be written as follows.

```
\step{<2>3}{$2+2=4$}
  \begin{proof*}
    \pf\ By my pocket calculator.
  \mainproof
    \step{<3>1}{$1+1=2$}
      ⋮
  \end{proof*}
```

The default is to show the main proof, so this portion of the proof produces:

$\langle 2\rangle 3.\ 2+2=4$
  $\langle 3\rangle 1.\ 1+1=2$
    ⋮

A `\useshortproofs` declaration causes the same source text to produce:

$\langle 2\rangle 3.\ 2+2=4$
  PROOF: By my pocket calculator.

A `\usemainproofs` declaration causes the main proof to be printed in subsequent proofs.

It makes no sense to nest a `proof*` environment inside another `proof*` environment; don't do it.

### 3.2 Printing Parts of a Proof

Sometimes you may want to show part of a proof. A partial proof you are unlikely to write is:

$\langle 2\rangle 3.\ 2+2=4$
  $\langle 3\rangle 1.\ 1+1=2$

To understand how this partial proof is produced, let's first write a proof with the smallest possible number of steps that includes the part of the proof we want to show. In this case, it might be.

```
\begin{proof}
  \step{<1>1}{Some step.}
    \begin{noproof}
      \step{<2>1}{Some step.}
      \step{<2>2}{Some step.}
      \step{<2>3}{$2+2=4$}
        \begin{proof}
          \step{<3>1}{$1+1=2$}
        \end{proof}
    \end{noproof}
\end{proof}
```

Then, for each step you don't want to show, replace the command `\step{`*lbl*`}{...}` with `\nostep{`*lbl*`}`. (A `\stepref{`*lbl*`}` in the part of the proof that is printed will produce the expected step number.) For each subproof that you don't want indented, replace the `proof` environment with a `noproof` environment.

```
\begin{proof}
  \nostep{<1>1}
    \begin{noproof}
      \nostep{<2>1}
      \nostep{<2>2}
      \step{<2>3}{$2+2=4$}
        \begin{proof}
          \step{<3>1}{$1+1=2$}
        \end{proof}
    \end{noproof}
\end{noproof}
```

In the `pf2` package, the outermost proof is not indented, so you can use either a `proof` or `noproof` environment for it.

## 3.3 Hiding Lower Levels

The declaration `\pfhidelevel{`$n$`}` causes only the first $n$ levels of subsequent proofs to be printed. Thus, `\pfhidelevel{0}` suppresses the printing of all `proof` and `proof*` environments, and `\pfhidelevel{999}` will cause the printing of complete proofs.

Within the scope of a `\pfhidelevel{`$n$`}` declaration, an `\unhideqedproof` declaration causes the top-most level of the proofs of level-$n$ Q.E.D. steps to be printed. A `\hideqedproof` declaration restores the default of not printing those level-$(n+1)$ proofs.

Suppressing lower levels of a proof in this way can be useful for hiding irrelevant details while you are writing the proof. It could also be useful in producing slides for a talk. If you distribute the LaTeX source of your papers, readers might insert `\pfhidelevel` commands to help them read the proofs hierarchically.

## 4 Controling the Style of Proofs

### 4.1 Spacing and Indentation

#### Indentation

The default indentation of proof steps, produced by a `\pfshortindent` declaration, is to indent each level of proof by a constant amount. That amount is can be changed by changing the length `\pfindent`.

The `\pflongindent` longindent causes steps to be indented as shown by these examples:

$\langle 1\rangle 1.$ $2 = 8/4$
PROOF: By Thm. 1.2.
$\langle 1\rangle 2.$ $8 = 2\sqrt{16}$
$\langle 2\rangle 1.$ $8 = 4 + 4$
PROOF: Obvious.
$\langle 2\rangle 2.$ $4 = \sqrt{16}$
$\langle 3\rangle 1.$ $4 \cdot 4 = 16$
PROOF: Step $\langle 2\rangle 1$.
$\langle 3\rangle 2.$ Q.E.D.
PROOF: By $\langle 3\rangle 1$.
$\langle 2\rangle 3.$ Q.E.D.
PROOF: $\langle 2\rangle 2$ and $\langle 1\rangle 1$
$\langle 1\rangle 3.$ Q.E.D.
PROOF: By $\langle 1\rangle 2$

1. $2 = 8/4$
PROOF: By Thm. 1.2.
2. $8 = 2\sqrt{16}$
2.1. $8 = 4 + 4$
PROOF: Obvious.
2.2. $4 = \sqrt{16}$
2.2.1. $4 \cdot 4 = 16$
PROOF: Step 2.1.
2.2.2. Q.E.D.
PROOF: By 2.2.1.
2.3. Q.E.D.
PROOF: 2.2 and 1
3. Q.E.D.
PROOF: By 2

**Vertical Spacing**

Vertical spacing in a proof is determined by three length parameters:

**beforePfSpace** The space inserted above a `proof` or `proof*` environment.

**beforePfSpace** The space inserted below a `proof` or `proof*` environment.

**interStepSpace** The space inserted before a step that is preceded by another step without an intervening proof.

The length of these vertical spaces can be specified separately for each proof level. For example, the command

```
\beforePfSpace{1ex, .5ex, 0ex}
```

declares that the space above a level-1 (outermost) `proof` environment is $1ex$, the space above a level-2 `proof` environment is $.5ex$, and the space above all higher-level `proof` environments is $0ex$ (no added space). The default declarations are:

```
\beforePfSpace{1ex, 0pt}
\afterPfSpace{1ex, 0pt}
\interStepSpace{0pt}
```

Two or more `\end{proof}` commands can come with no intervening steps, so multiple levels of proof end at the same point. In that case, the maximum applicable *afterPfSpace* value is used.

## 4.2 Numbering

The declarations

```
\pfshortnumbers  \pflongnumbers
\pfsidenumbers  \pfnosidenumbers
```

are explained in Section 2.3. There is one additional declaration relevant to step numbering. The command `\pfmixednumbers{`$n$`}` specifies that long step numbers are used for steps at all levels less than $n$, and short step nunbers for higher levels. I find such a mixing of number styles to be weird and have never used it, but perhaps someone will like it.

### 4.3 Keywords

The command `\pfkeywords{`*sty*`}` specifies that the style *sty* of keywords is to be used in proofs. There are currently two styles defined:

`default` The style used in this the examples in this document.

`tla` The style used in pretty-printing TLA$^+$ proofs [2]. Here's an example of the `tla` style:

1. suffices assume $x > 1$
   prove $x > 0$

You can define your own style by redefining the LaTeX commands that produce the keywords. Here are the commands and the keywords they produce.

| | |
|---|---|
| `\qedstepPfkwd` | Q.E.D. |
| `\assumePfkwd` | Assume: |
| `\provePfkwd` | Prove: |
| `\sufficesPfkwd` | Suffices: |
| `\asufficesPfkwd` | Suffices (produced by the `\asuffices` command) |
| `\casePfkwd` | Case: |
| `\letPfkwd` | Let: |
| `\definePfkwd` | Define: |
| `\pickPfkwd` | Pick |
| `\pfnewPfkwd` | New |
| `\proofPfkwd` | Proof: |
| `\proofsketchPfkwd` | Proof sketch: |
| `\qedPfkwd` | □ (produced by the `\qed` command) |
| `\pfdot` | . (the "." in long step numbers.) |

## References

[1] Leslie Lamport. How to write a proof. In Karen Uhlenbeck, editor, *Global Analysis in Modern Mathematics*, Houston, 1992. Publish or Perish Press. Also appeared in *American Mathematical Monthly 102*, 7 (August-September 1995), 348–369.

[2] Leslie Lamport. How to write a 21st century proof. To appear, 2011.