QED.sty — right-justified end-of-proof mark

Paul Taylor

1994

1 Introduction

Euclid established a style in his *Elements* that, for better or worse, has come down to us as the definitive one for presenting mathematical proofs. A *Theorem* or *Problem* (that is, something to be *proved* or *constructed*, respectively) is first stated, with the heading “Theorem” or “Problem”. After the proof or construction has been given, the original statement is re-iterated, followed by the letters “QED” (or “QEI” for a Problem).

QED stood for either *quod erat demonstrandum* or *quod est demonstatum*, which respectively mean “which was to be shown” and “which has been shown”. Modern custom is to use a little square instead of the letters QED.

It is also customary to put this square on the far right of the page, which is where this macro package comes in. It is surprisingly difficult to persuade TEX to do this. The obvious solution,

```
and so $2 n = p + q$ as required.  \hfill $\square$
```

works 80% of the time, but fails if the preceding text happens to fill the line, in which case the square appears at the *left* of the next line.

This happens because, when TEX breaks the line at `\hfill`, it discards this stretchable space (called “glue”) [TEXbook, page 97]. The reason for this behaviour is that, when a word occurs at the beginning or end of a line, we like to have it printed hard against the margin, so the space that would otherwise occur between words is lost when the line is broken there.

TEX and LATEX have been in standard use in the mathematical sciences for two decades now, so you would expect this problem to have been solved, and solved many times. However, on examining the documents and macro files that have come my way, I find that they either employ the naïve solution above, which doesn’t work, or they use code that is mine.

With the help of Chris Thompson of the Cambridge University Computing Service, I implemented the following solution for my Ph.D. thesis in 1986. As well as keeping the square on the right (though possibly on the next line), it also stops it from being left over (as a “widow”) on the next *page*.

```
\def\qed {{%       set up
  \parfillskip=0pt        % so \par doesnt push \square to left
```

```
  \widowpenalty=10000     % so we dont break the page before \square
  \displaywidowpenalty=10000  % ditto
  \finalhyphendemerits=0  % TeXbook exercise 14.32
 %
 %                 horizontal
  \leavevmode             % \nobreak means lines not pages
  \unskip                 % remove previous space or glue
  \nobreak                % don't break lines
  \hfil                   % ragged right if we spill over
  \penalty50              % discouragement to do so
  \hskip.2em              % ensure some space
  \null                   % anchor following \hfill
  \hfill                  % push \square to right
  $\square$%              % the end-of-proof mark
 %
 %                   vertical
  \par}}                  % build paragraph
```

If you already have a macro called `\qed`, `\endproof` or `\pushright` which is laid out in a similar way to this, then it is almost certainly my code.

Please use this macro package in place of this fragment of my code.

Besides the basic case of the little square at the end of a piece of *text*, it also works with displays (`$$` or the `displaymath` environment) and has hooks for compatibility with other situations.

2 QED as a macro

The basic use of the package is as follows:

```
\documentclass{article}
\title{A Proof of Goldbach's Conjecture}
\author{Paul Taylor}

\usepackage[open-square,define-standard-theorems]{QED}
\usepackage{amssymb}

\begin{document}
\maketitle

\begin{Theorem}
    Every even number may be expressed as a sum of two primes.
\end{Theorem}

\Proof Trivial.  $2n = p+q$.  \qed
```

```
\end{document}
```

In principle, the `\qed` macro goes logically where you want the little square to appear in the text. **But this still doesn't work** (and nor can it) because of the way TeX works. Briefly, the `\qed` macro alters the way in which the paragraph is formatted, but commands like `\end{itemize}` close the paragraph themselves, leaving `\qed` to the next paragraph, and therefore the next line.

The rule is that `\qed` has to be put *inside* the innermost environment instead, for example

```
\begin{itemize}
\item This case is easy too.
\item In the final case we have
  $$  2n = p + q      \qed  $$
\end{itemize}
```

If you still have to put `\qed` outside the environment, it may be appropriate to add

```
\vskip-\baselineskip
```

before `\qed`, unless of course this would over-print. There is, however, still a danger that TeX will break the page here (which seems to be unavoidable if the environment ends with a `\penalty`).

Even if you don't need to put `\qed` inside an inner environment to avoid a blank *line*, it's still a good idea to do so, because otherwise it's possible that intervening end-of-environment material may allow TeX to put the square on the next *page*.

3 Options

The package may be customised using the following options in the `usepackage` declaration:

[`black-square`] Use ■ for `\qed`.

[`open-square`] Use □ for `\qed`.

[`define-standard-theorems`] Define Answer, Axiom, Assumption, Conjecture, Construction, Convention, Corollary, Counterexample, Definition, Example, Exercise, Fact, Lemma, NB, Notation, Note, Observation, Problem, Proposition, PS, Remark, Question, Theorem, Thesis, Warning and their plurals.

[`roman-theorems`] Print the theorem statements in roman (upright).

[`italic-theorems`] Print the theorem statements in *italic* (default).

[`theorems-as-commands`] Use *commands* like `\Theorem` to introduce a theorem statement, instead of wrapping it in a `\begin{Theorem}` ... `\end{Theorem}` environment.

[`auto-QED`] Make `\end{Theorem}` insert a `\qed` automatically.

[`explicit-QED`] Suppress automatic `\qed` in `\end{Theorem}` (default).

[`number-theorems`] (Default).

[`dont-number-theorems`]

Beware that the □ and ■ symbols come from the AMS symbols fonts, not the standard TeX ones. This is why you need to `\usepackage{amssymb}`.

Besides the `open-square` and `black-square` options, you can also change what `\qed` prints by redefining, for example,

```
\def\qedsymbol{\bigcirc}
```

to get a big circle. The macros `\QED`, `\QEI` and `\QEF` are also provided for classically minded mathematicians, and produce these three-letter acronyms (I and F stand for inveniendum and faciendum — to be found or done).

The `\Proof` macro, of course, produces the word "**Proof**", in bold, as a new unindented paragraph. You can change this by redefining, for example,

```
\def \TheWordProof {\kern\parindent\textsc{D\'emonstration}: }
```

to print it in SMALL CAPITALS with indentation and a colon.

4 Proof environments

Given that a proof is enclosed in the pair `\Proof` ... `\qed`, you may like to use a LaTeX environment instead:

```
\begin{Theorem}
   Every even number is the sum of two primes.
\end{Theorem}
\begin{Proof}
   Easy.
\end{Proof}
```

The problem with this is that `\end{Proof}` comes *after* any `$$` or `\end{itemize}` that might finish the proof, which is too late for `\qed` to do its stuff.

In this case, you have to insert `\qed` yourself within `$$` or `\end{itemize}` as in Section 2. It will then print in the correct place. `QED.sty` will not repeat the square when `\end{Proof}` is executed.

In order to achieve this effect, and also to allow for nested `Proof` environments (for example the Lemmas leading up to a Theorem), `QED.sty` has to maintain an internal switch saying whether to print the square or not. This switch is turned on by `\Proof` or `\begin{Proof}` and off when `\qed` is executed. Nested `Proof` environments save the outer value of the switch. As a result of this,

`\qed` which doesn't follow `\Proof` will not print the square!

Each properly nested `\begin{Proof}` ... `\end{Proof}` pair gets a single square at the end, on the far right hand side.

5 Theorem styles and environments

You may feel that it is logical to put the `Proof` environment *within* the `Theorem` environment. `QED.sty` supports this. In the case where the `Proof` is absent (because it is obvious or because the main Theorem is stated after its proof), you can arrange for the square to be generated by `\end{Theorem}`, by selecting the `auto-QED` option in `\usepackage`.

I, on the other hand, regard the words **Theorem** *etc.* as *headings*, and prefer not to wrap my enunciations in LaTeX environments at all. I just write

```
\Theorem $2n = p+q$.
\Proof Obvious. \qed
```

This behaviour is declared with the `theorems-as-commands` option in `\usepackage`. This also suppresses *italic* typesetting of the statement, because there is no `\end{Theorem}` to turn it off.

You can also suppress italicisation or numbering with the `\usepackage` options

`roman-theorems` and `dont-number-theorems`

If you want to use Mittelbach's `theorem.sty` package, you have to load that first, then `QED.sty`. This is because of the way in which the two packages interact, in particular regarding the internal macro `\@thm`. The options `theorems-as-commands`, `roman-theorems` and `dont-number-theorems` will not work in this case.

6 Control of page breaking

This is not really concerned with theorems and proofs, but this problem seems to arise in the same context.

Whilst TeX is extremely good at breaking paragraphs into lines nicely, it often makes a mess of page breaking. This is because the program was written in the early 1980s, when computers had kilobytes rather than megabytes of memory, so TeX generates pages of DVI as soon as it can.

Its mechanisms for breaking both lines and pages involve `\penalty` items in its horizontal and vertical lists. The LaTeX commands `\nobreak` and `\goodbreak` generate TeX `\penalty` commands, which are supposed to control (line and) page breaking.

In my experience, these don't work.

The macros `\goalbreak` and `\allowlines` fix this problem by testing how full the page already is, and generating a `\newpage` if necessary.

- `\goalbreak{50pt}` or `\goalbreak{2cm}` ensure that this much space is available on the current page.

- `\allowlines{5}` measures the space in lines instead (using `\baselineskip`).

You may want to use this just before a short Theorem statement in order to keep it on the same page as its proof. (In fact, `QED.sty` uses a small `\penalty` to encourage this anyway.)

The macro `\closeupaline` is useful after `\Proof` when displayed material follows immediately:

```
\Proof\closeupaline
$$ 2n = p + q $$
```

You may want to use this before a short theorem statement, to keep it one the same page as its proof.

7 Modifying other LaTeX environments

This section is for TeXperts.

The `\qed` macro will only work in those environments which provide access to the right hand side of the page. In other words, wherever you think "`\hfill\square`" ought to work then `\qed` will. (Of course it is just because that naïve method doesn't work that this macro file is needed at all.) The macro was designed to solve the problem in the case of "unrestricted horizontal mode" (*i.e.* at the end of a paragraph); it also works in `$$` displays by pretending to be an equation number. In most other cases it basically does "`\hfill\square`" or "`\hfill$\square$`" and hopes for the best.

Unfortunately some environments (we shall use `picture` as an example) do not allow this at all. In this case, the (re)designers of the environment may wish to handle the placing of the little square for themselves.

Each individual environment (both the standard ones and those defined by the user) needs to co-operate in order to make `\qed` work. It is not the business of this package to rewrite LaTeX in this way, but the following mechanism may be inserted into other environments.

- Put `\WillHandleQED` in the `\begin` part of the environment definition. This notifies `QED.sty` that the environment is going to ...
- put `\HandleQED` anywhere in the `\end` part of the environment definition (even inside dollars, braces or another environment), where you would like the square to be printed if `\qed` is used.

For example:

```
\newenvironment{qedpicture}%
 {\begin{picture}\WillHandleQED}%
 {\put(0,0){\HandleQED}\end{picture}}
```

defines a version of the `picture` environment in which the `\qed` command causes a square to be printed at the origin of `picture`'s coordinate system — *wherever* `\qed` occurs within the body.

The width of the saved `\qed` symbol is available for testing as the dimension `\wd\QEDbox`.

To make this compatible with *not* loading `QED.sty`, add the following to macro files which use `\WillHandleQED` and `\HandleQED`:

```
\ifx\WillHandleQED\undefined
   \let\WillHandleQED\relax
   \let\HandleQED\relax
   \let\UnHandleQED\relax % if you use it
   \let\QEDbox\voidb@x    % must have \makeatletter here!
\fi
```

Also, `\UnHandleQED` restores normal processing; this may be useful for the `minipage` environment, which has its own notion of "right hand side" of the page.

8 Implementation

Underlying `\qed` with its switchable behaviour and configurable symbol is a macro

`\pushright{text}`

that always prints its argument, on the far right. The argument to `\pushright` is in text mode, so `\qed` invokes it with `{$\qedsymbol$}`. It does the following case analysis:

- outer horizontal text mode (in a paragraph) uses `\parag@pushright{#1}`
- inner horizontal (text or maths) mode (`\hbox`) uses `\hbox@pushright{#1}`
- outer maths mode (display) uses TeX's `\eqno` mechanism but has to suppress a LaTeX error check.
- outer vertical mode (between paragraphs on the current page) uses `\page@pushright{#1}`;
- inner vertical mode (`\vbox`) uses `\vbox@pushright{#1}`;

The interested reader can follow the comments in the file `QED.sty` itself to see how these cases are handled.

Why can't we do all of this with `\endProof`? Because by that stage the damage has been done: the environments, paragraphs or displays have already been closed and their `\hbox`'es added to the current vertical list, and cannot be removed for alteration if this is the current page.

Ideally we should be able to add the symbol to the last line after the paragraph, display or whatever other structure has been completed and we've returned to vertical mode.

The reason why this isn't possible is that TeX forbids the use of `\lastbox` and similar commands (which can be used to take `\hbox` and `\vbox` lists apart) on the current page (outer vmode). It does this because its "galley" implementation (shipping complete pages to the DVI file as soon as possible) might output the item before we have a chance to remove it from the page.

In principle there is a more sophisticated method, which could be used in inner vertical mode (*i.e.* inside TeX's `\vbox`, `\vadjust` or `\insert` or LaTeX's `\footnote`, `\parbox`, `minipage`, `figure` or `table`). In these circumstances `\lastbox` is allowed.

We would have to strip and restore glue and penalties (`\insert`, `\mark` and `\write` are obstacles to this) to get access to the last line (box), which may still not have the correct baseline. We could check its natural width against the page width, but if we re-set it, centering and (right) justification might be disturbed.

This method could even used to implement the `Proof` environment by collecting the entire proof in its own `\vbox`, modifying that and `\unvbox`'ing it to the current page. This approach may lead to the following problems:

- footnotes, floats, marginpars, labels and index items may be lost;
- overflow of TEX's main memory, especially if `\end{Proof}` is missing;
- I don't know how page-breaking would be handled.

I don't think this is worth the trouble. The package has, since January 1994, included an error message

`QED.sty` could be improved in this case: please ask

that appears when `\qed` is used in inner vertical mode, but nobody has contacted me as a result of it.

9 Availability, compatibility, copyright and (no) warranty

The package may be obtained from the Comprehensive TEX Archive Network at

`www.ctan.org`

or one of its numerous mirror sites. Within the directory structure there, the file is called

`macros/generic/proofs/taylor/QED.sty`

but on your own computer you may call it `QED.sty`, `QED.tex`, `qed.sty` or `qed.tex`.