The bigfoot bundle for critical editions

David Kastrup
dak@gnu.org

Abstract

The LaTeX package bigfoot and supporting packages solve many of today's problems occurring in the contexts of single and multiple blocks of footnotes, and more. The main application is with philological works and publications, but simpler problems can be solved painlessly as well without exercising all of the package's complexities. For other problems not yet tackled in this area, a solid framework is provided.

Introduction

Footnotes in TeX are a problematic area. One reason is that TeX's insertion mechanism is far too basic to cope with more complicated usage patterns. Insertions are not subjected to the usual optimization methods of TeX, but instead are fitted on the page with a greedy algorithm at the time they are encountered. At that time, they may also be split or floated to the next page. A split does not take into account any mandatory following material on the vertical list: infinite values of `\widowpenalty` coupled with footnotes anchored in the next to last line will not be split at the correct point, and thus will have to get moved over to the next page.

Another deficiency is that when splitting a footnote, shrinkability is considered by TeX while doing the split, fitting more material on the page. However, at the time of the page break decision, the information about the shrinkability used for the insertion split gets lost, and consequently the page can appear overfull.

Since TeX does not even get the cases right for which it was designed, more complicated footnote schemes like those for critical editions have to be implemented mostly manually.

The bigfoot addresses a number of deficiencies and replaces the normal footnote mechanism.

Features

So what are the features that bigfoot provides?

- Multiple footnote apparatus[1] are possible.[2]
- Footnotes can be nested.[3]
- Footnotes are numbered in the order they appear on the page, and numbering may start from 1[†] on each page. In each apparatus, the footnotes are arranged in numerical order identical to page order. This does not sound exciting at all until you consider the implications of footnotes being nested: if the main text has some footnote[4] and then the publisher comments the main text with a footnote,[c] the logical order of footnotes (in which they appear in the source text) would have been to let footnote e appear before footnote c. The footnotes instead will be reordered to page order.[5]
- Footnotes may contain `\verbatim` commands[6] and similar, and they will just work as expected. This is achieved in a manner similar to the `\footnote` command of plain TeX.
- Footnotes can be broken across pages.[7]

[1] An apparatus is one block of contiguous footnotes forming a logical and physical unit. Separate apparatus[a] can be independently broken to the next page.

[2] Actually, manyfoot already provides this functionality[b] but it fails to address a number of intricacies inherent to this sort of setup, a few of which follow.

[3] You can anchor footnotes for some apparatus in the main text[d].

[†] or whatever the first footnote symbol happened to be

[4] such as shown in this example footnote[e]

[5] The style file perpage has been extended with additional functionality for reordering such numbers.

[6] even stuff like `\verb-\iffalse-`

[7] While this does not sound like something excitingly new, it must be noted that TeX does not do a satisfactory job at splitting insertions, the underlying mechanism for split footnotes. In particular, TeX only manages to find a split when no material whatsoever is added to the page after the occurence of the split footnote. This might include another footnote in a different apparatus, or simply a line tied to the

[a] Yes, this is the correct plural form.

[b] and is loaded by bigfoot

[c] This is a subsequent comment to the main text.

[d] or any apparatus preceding it on the page

[e] which happens to have a comment attached to it. Notice that bigfoot will prefer to leave this smaller footnote block intact, as breaking it will not help fitting the above footnote block on the page.

Polska Grupa Użytkowników Systemu TeX, 2005 (http://www.GUST.org.pl)

- When footnotes are broken across pages, the color stack is maintained properly. Color is handled in LaTeX with the help of specials that switch the color (and, in the case of `dvips`, restoring it afterwards with the help of a color stack). Restarting the footnote on the next page with the proper color is something that has never worked in LaTeX. Now it simply does.
- Footnotes may be set in a compact form in one running paragraph.[8]
- Split footnotes will not get jumbled in the presence of floats. `bigfoot` is not afflicted by this bug in LaTeX's output routine since it does not delegate the task of splitting footnotes to TeX in the first place. While the faulty output routine of LaTeX may still jumble the order of footnotes in that particular case (when one footnote gets held over as an insertion 'floated' at infinite cost), `bigfoot` will sort the jumbled footnotes back into order before processing them.
- Each footnote apparatus can have its own private variant of `\@makefntext` and a few other macros and parameters responsible for formatting a footnote block. The default is to use what the class provides, but special versions can be defined, for example,

  ```
  \FootnoteSpecific{variants}%
  \long\def\@makefntext#1{...
  ```

  for the footnote block called "variants".

Drawbacks

What about current drawbacks?

- ε-TeX is used throughout. After it became clear that the implementation of the package would not be possible without using some of ε-TeX's features, its features were extensively employed: rewriting the package to get along without ε-TeX would be very hard, even if you came up with ideas for those cases where I could find no other solution. Free TeX distributions have come with ε-TeX for a long time by now (in fact, ε-TeX is now the recommended engine for LaTeX, and actually used as the default in the latest TeX Live), but proprietary variants may lack ε-TeX support. The same holds for quite a few **Ω** versions.
- The licence is not the LPPL, but the GPL. In my book, I consider this an advantage: the functionality of the package is quite important, and it is in its infancy yet. I would not like to encourage a market of proprietary offsprings directly competing with it. While with sufficient financial incentive I might feel confident enough to have the means to reimplement whatever noteworthy extension somebody else might come up with, at the current time I prefer this way of ensuring that the free development does not fall behind and that there is no incentive to turn to

---

current line with an infinite penalty, for example because of a respective setting of `\widowpenalty`. In contrast, `bigfoot` breaks footnotes properly in such circumstances, and it uses a backtracking algorithm (with early pruning of branches that can't beat the current optimum) for finding the best split positions for several footnote apparatus in parallel. The fill level of the page is taken into account as well as the costs of the individual splits. A split footnote is penalized with a penalty of 10000 (which is pretty similar to what TeX itself does when dealing with footnotes), so that in general TeX will tend to avoid splitting more than a single footnote whenever possible. One complication is that if the parts broken to the next page contain footnotes themselves, those have to be moved to the next page completely and adapted to the numbering of footnotes there[a]. This rather intricate and complicated mechanism leads to results that look simple and natural.

[8] While `manyfoot` and `fnpara` also offer this arrangement, `bigfoot` offers a superior solution in several respects:

- The line breaking can be chosen much more flexibly: with appropriate customization, it is possible to fine-tune quite well when and where stuff will be placed in the same line, and when starting a new line will be preferred.
- In-paragraph footnotes can be broken across pages automatically, just like normal footnotes. They will only be broken after the last footnote in the block has started.
- Pages will not become over- or underfull because of misestimating of the size of in-paragraph footnotes. Also the total width of such footnotes is not restricted to `\maxdimen` (which sounds generous at something like 6 m or 19 ft, until you realize that a few pages of text suffice to burst that limit, and a few pages of text are reached easily with longer variants of the main text). While TeX will accumulate boxes exceeding this size without problem, it panics at its own audacity if you actually ask about the total width of the acquired material. While one may still not have material exceeding a total *vertical* size of `\maxdimen` accumulate in one footnote block, one would usually need a few dozen pages for that, and so *this* limitation is much less noisome than the corresponding restriction on the horizontal size.
- The decision of whether to make a footnote in-paragraph or standalone can be changed for each footnote apparatus at any time, including on mid-page. In fact, you can make this decision for each footnote separately. Since display math requires vertical mode footnotes, this is convenient.
- `bigfoot` will make a good-faith effort to adapt the normal footnote layout provided by the document class with the `\@makefnmark` and `\@makefntext` macros to in-paragraph footnotes.

[a] which can be completely different!

developers with no qualms about creating proprietary versions.

- bigfoot requires twice as many box registers[9] as manyfoot: one set in the form of an insertion for each footnote apparatus, one set as mere boxes.
- It can't handle more footnotes in a single block per page than the group nesting limit of TEX, and that is usually hardwired at 255.[†]
- Since it meddles considerably with the output routine's workings, interoperation with other packages doing the same might be problematic. Considerable effort has been spent on minimizing possibly bad interactions, but the results might not always be satisfactory and, at the very least, might depend on the load order of packages.
- It slows things down. This is not much of a concern, and usually the package is astonishingly fast.
- The complexity of the package makes it more likely for things to go wrong in new ways.[10]

Additional new packages

The bundle provides some more packages: perpage is used for the sort of renumbering games mentioned before, and suffix is used for defining augmented commands.

As an example of use for those packages we had previously a few examples where numbers like 7[‡] and 255[§] were given footnotes, and in order not to confuse this with powers as the following 666[11] is in danger of, we have switched to per-page numbering of footnotes with symbols for that purpose. The source code simply uses

```
like~7\footnote'{a lucky number}
```

namely a variant footnote command. How is that achieved? Just with

```
\newcounter{footalt}
\def\thefootalt{\fnsymbol{footalt}}
\MakeSortedPerPage[2]{footalt}
\WithSuffix\def\footnotedefault'{%
  \refstepcounter{footalt}%
  \Footnote{\thefootalt}}
```

A new counter is created, its printed representation is set to footnote symbols, the counter is made to start from 2 on each page (since symbol 1[¶] is a bit ugly), and then a variant of \footnotedefault is defined which will step the given counter and use it as a footnote mark.[12]

That's all. One can define several suffixes, the resulting commands are robust[13], and one can use arguments and other stuff. For example,

```
\WithSuffix\long\def\footnotedefault
   [#1]{#2}{...
```

would augment the macro \footnotedefault by a variant accepting an optional argument.

Some internals

Basic operation The package uses most of the interfaces of manyfoot for its operation. While it uses TEX's insertions for managing the page content, the material collected in those insertions is in a pretty raw state and its size is always overestimated.[14] The actual material that goes onto the finished page is generated from the insertions at \output time.

Material that is put into insertions is prewrapped into boxes without intervening glue.[15] The box dimensions are also somewhat special: while the total height (height+depth) corresponds to the actual size of the footnote, the depth contains a unique id that identifies the last footnote in each box (of which there usually is just one, unless we are dealing with the remnants of an in-paragraph footnote apparatus broken across pages). The width is set to a sort key that is used for rearranging the various footnotes into an order corresponding to their order of appearance on the page.

The boxes are sorted by unvboxing them and then calling the comparatively simple sorting routine (a straight insertion sort):

```
\def\FN@sortlist{{\setbox\z@\lastbox
  \ifvoid\z@ \else \FN@sortlist
  \FN@sortlistii \fi}}
```

[9] Since ε-TEX has an ample supply of box registers (32767 instead of 256), this is not really much of an additional limitation. If you find yourself running out of insertions, etex offers the \reserveinserts command.

[†] This limit seems sufficient at first glance, but one could use the various mechanisms available in connection with in-paragraph footnotes to make sure that a footnote will be broken across the page at a point closely related to the main text's breakpoint (for example, if you are doing an interlinear translation in a footnote). In that case, this limit might become problematic.

[10] Most of those problems should arise under requirements that could not possibly be met without the package, so this would be reason for improving rather than not using the package.

[‡] a lucky number [§] well, almost as lucky [11] strange, yes?

[¶] which is ∗

[12] manyfoot defines a two-argument command \Footnote that takes a footnote mark and corresponding footnote text.

[13] as long as their suffixes are so as well

[14] bigfoot simply sets each footnote, even those that should be typeset with others in one block, separately in its own paragraph for estimating its size, which should be a safe upper limit for the size a footnote can take when set in a paragraph with others.

[15] That way, there is never a legal breakpoint in an insertion.

```
\def\FN@sortlistii{\setbox\tw@\lastbox
  \ifvoid\tw@\else
    \ifdim\wd\tw@<\wd\z@
      {\FN@sortlistii}%
    \fi
    \nointerlineskip \box\tw@ \fi
  \nointerlineskip \box\z@}
```

and then all consecutive runs of hboxes are joined into vboxes. The desirability of breaking between two in-paragraph footnotes depends on their respective size, on whether this would save lines when typesetting, on whether a footnote apparatus can be shrunk by more than a certain factor in this manner, and whether the ratio of allowable joints between footnotes[16] to the number of footnotes exceeds a certain ratio.[17] The criteria are configurable per apparatus or globally.

There are some footnotes where a vertical arrangement is mandatory,[18] and the footnote must not be set into a hbox to start with. This is the case, for example, for footnotes containing display math. Placing a `+` sign before the opening brace of the footnote text will achieve that, and similarly a `-` sign can be used for switching in an otherwise vertically arranged footnote apparatus to horizontal arrangement.

`bigfoot` hooks into the output routine and does its accounting work before the main output routine gets a chance to get called. This work involves sorting the various contributions to a single insertion, joining together all in-paragraph footnotes into a single paragraph, measuring the resulting boxes, and gathering more material from the page in case that this produces an underfull box. As the insertions `bigfoot` uses are unsplittable, this will often lead to an overfull box. In that case, various footnote blocks get split to an optimum size before the real output routine gets called, and if this results in an underfull box again, more material gets called in again.

Dissecting `\@makefntext` The footnote layout of document classes is given by `\@makefntext`. This macro receives one argument, the body of the footnote. We'll now discuss several problems we want to tackle in the context of using `\@makefntext` for implementing the layout prescribed by the class file.

Robust footnotes One problem with LaTeX's footnotes is that they scan their arguments prematurely. We want them to behave more like those of plain TeX, to forestall complaints when `\verb` and its catcode mongering cousins fail to work in footnotes. The trick is to have the macro argument of the `\footnote` macro not really be a macro argument, but the content of an `\hbox` or `\vbox` command, and have subsequent code do its work with `\aftergroup`, once the command finishes.

This means that we have to cut `\@makefntext` into parts before and after its argument. It turns out that cutting the part before it starts processing its argument is rather easy:

```
\@makefntext \iffalse \fi
```

will do that. It executes and expands `\@makefntext` until it comes to the point where it would process its argument, which happens to be `\iffalse`, and then kills the rest of `\@makefntext`. At least as long as the argument `#1` does not happen to be in itself inside of a conditional, in which case bad things will happen. Very bad things. But a pretty thorough sampling of `\@makefntext` variants on TeX Live did not turn up such code.

Much more problematic is getting hold of the second part of `\@makefntext`. It turns out that about 95% of the variations out there in different class files will work with

```
\expandafter \iffalse \@makefntext \fi
```

which looks rather similar to the above. Unfortunately, it is not quite equivalent, since in the upper code, `\@makefntext` is cut into two once it has been expanded up to its macro parameter, whereas in the lower version it is cut into two before any parts of it get expanded. If any of the closing braces that follow `#1` in the definition of `\@makefntext` happen to belong to the argument of a macro starting before `#1`, they will cause spurious closing groups.

Getting the closing part at the end of the footnote without any remaining macro braces is more tricky, inefficient and error prone. One possibility is starting another instance of `\@makefntext` inside of a box to be discarded later. Then as its macro argument you use code that will repeatedly be closing opened groups until the outer group level is reached again and the box can be discarded. ε-TeX's grouping status macros (`\currentgrouplevel` and `\currentgrouptype`) make it possible to know how to close the current group and whether it is the last involved one. After everything that has been opened has been discarded again, the remaining tokens in

---

[16] where both footnotes around the breakpoint are considered potentially horizontal material

[17] A footnote apparatus in which there are just few horizontally arranged footnotes would appear inconsistent.

[18] like footnotes containing

- list environments
- display math like

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{n} = \log \frac{1}{2}$$

the input stream should form a perfect complement to the tokens that the initial `\iffalse` trick has discarded at the start of the footnote.

One other mechanism probably worth playing with is the use of alignment templates, since they provide a natural way of having TEX switch input contexts across groups. The best approach in that regard would seem to parse the content of the footnote within a `\noalign` group of a `\valign`, but that still suffers from the problem that no automatic discretionaries are generated for explicit hyphens.

But since most of the the `\@makefntext` variants out in the field are covered with the simple variant (basically, this is the case for all definitions that do not use `#1` within a macro argument itself), `bigfoot` for now has not added any of the more complicated versions. The group discarding trick might perhaps be made available with a separate package option at a later time, if there is sufficient demand for it.

But it may be easier in most cases simply to rewrite the culprits: after all, `\@makefntext` is rarely complicated. Most notably, the `\@makefntext` of the `ltugboat` class is so ridiculously contorted that the automated analysis of it fails. (It has been replaced with an equivalent for this article.)

`\@makefntext` in 'para' footnotes is really a bit out of place: the 'para' footnote style sets all footnotes within one continuous running paragraph, a manner of operation quite different from the original intent of `\@makefntext`. Single footnotes are first collected in horizontal mode, and at `\output` time the relevant footnotes making it to the current page are pasted together. This has several problems: for one, `\@makefntext` will set paragraph breaking parameters. We need these at the time that we assemble the footnotes into one paragraph. But `\@makefntext` also generates the footnote mark, so we need to call it for each footnote.

So even when we set `\@thefnmark`[19] equal to an empty string at footnote assembly time, the assembled footnote mark will likely take up some additional space. This is not the end of our worries: while the formatting will be right for standard footnotes, it does not cater for 'para' footnotes. If we want to have a reasonably looking turnout, here are the conditions we have to meet:

1. At the beginning of the footnote block, or if a footnote starts right after a line break, the specified formatting should be used.
2. Within the line, we shall keep the spacing between footnote mark and footnote text correct. However, most styles right-justify the footnote mark within a box of fixed size. If we keep this sort of formatting, we will end up with a large space before short footnote marks, and a small one before longer marks. Since the amount of whitespace inside of a line should not be so large as to cause unsightly white holes, nor so small to make the footnote mark confused to be a part of the preceding footnote, we want a fixed spacing before the footnote mark.

The solution to these problems is to do a few measurements: we measure the width that an empty footnote mark would cause in the footnote box (and start our assembled footnotes with a negative space compensating that), and we typeset the footnote mark once on its own with `\@makefntext`, fishing with `\unskip` and `\lastbox` for the footnote mark box and resetting it to its natural size (which will kill the particular justification prevalent in the majority of class files doing justification). The difference in box size gets recorded separately until the time that the footnote gets set, and then the interfootnote glue is calculated accordingly.[20]

Maintaining the color stack is not nice.[21]

What is the color stack, anyway? LATEX's `color` package provides color selection commands that will change the current text color until the end of the group, where it will be restored.

The involved macros are

`\color@begin@group` is called at the start of each 'movable' box: material that does not necessarily appear right away. Without color support loaded, this does nothing. With color support loaded, it is usually equal to `\begingroup`.

`\color@end@group` is the corresponding macro at the end of 'movable' boxen. Any color restoration initiated with `\aftergroup` in the box will happen right here, still within the scope of the box, instead of outside where it would not move with the box.

`\set@color` will be called for setting the current color. It will also use `\aftergroup` in order to insert a call to `\reset@color` when the group ends.

`\reset@color` will restore the current color to what it was before the current group.

How will the color be restored? We have two different models:

dvips restores colors by making use of a color stack: dvips can 'push' a new color onto the stack, and pop the previous color back. Consequently, `\reset@color` inserts a special that tells dvips to pop the stack once.

---

[19] the mark as displayed in the footnote

[20] A few classes work with `\parshape` or `\hangindent`, either directly or with a `list` environment, and this is also taken into consideration as far as possible.

[21] The main philosophy for work on the color stack has been summarized well by David Carlisle: "It's not my fault."

pdftex instead restores colors by reinstating the color stored in `\current@color` after closing the group.[22]

It is clear that the pdftex model is insufficient to even keep the color of the main text across page breaks, since on the next page there is no special after the page break that could switch back to the text color after the page footer[23] from the last page and headers from the current page have been placed with a default color.[24]

But in the context of footnotes, the problem is severely exacerbated: a footnote can be broken right in the middle of a sequence of color changes. The technically sound solution would be to switch to a different color stack for each footnote block. Since dvips does not offer multiple color stacks (and pdftex does not even offer a single one), we have to revert to trickery.

At each color change, the complete state of the color stack gets recorded in a mark. When the footnote is broken, we use the information in the mark in order to unwind the color stack to the state on the page before the footnote was entered. When the footnote is continued on the next page, the unwound color stack is reinstated again. Whenever `\color@begin@group` is called, the whole recording and restoration business is stopped (since a new context has been started), the record of the color stack essentially restored to empty, and only resumed when the corresponding group has ended.

In order to keep these proceedings fit for consumption by the general public, the reader is referred to the actual code for further details.

Outlook

At the time this article was written, quite a few tasks remained to be done. Further improvements in the footnote breaking decisions and their scoring metrics are needed. Flushing footnotes out in the middle of the page for short successive works would be nice. Amending footnotes with marginals (including line numbers) in a manner consistent with the main text would seem desirable. Additional footnote arrangements apart from the existing basic two styles should be easily implementable on top of the general scoring and breaking mechanisms.

[22] Of course this means that if we are at the end of a movable box, the restored color will be that at the time the box was assembled, not at the time it was used.

[23] and footnotes

[24] Heiko Oberdiek's `pdfcolmk` package tries to deal with that particular problem.

Conclusion

It is hoped and expected that this bundle will become a basic building block for critical typesetting applications. While there are other packages available for that purpose, `bigfoot` (with its companions) offers the following important features:

- It is completely layout-neutral: while most solutions for critical typesetting are provided in the form of document classes, `bigfoot` does not make layout decisions but instead just uses the layout provided by a base class.
- Footnote arrangement and balancing is vastly superior to and more flexible than any of the available solutions.
- Color works.
- The interfaces for creating new functionality focused around footnotes are reasonably simple.

At the time this article was written, not all interfaces have yet been cast into stone. However, `bigfoot` can be mostly used as an upwards-compatible drop-in replacement of `manyfoot`.

One can define a `plain` footnote style in the manner of `manyfoot`, and then the default footnotes will get replaced by this footnote style. In fact, if one does not redefine the `plain` style, `bigfoot` will do so itself. Thus just loading it without any further action on behalf of the user will cater for the most common problems in connection with footnotes.

At the current point of time, still problems remain to be tackled: the accounting of page space and page splits was modeled after TeX's insertion mechanism and suffers from the same problem with regard to shrinkability, so in this paper, shrinkability has been removed from footnotes, a bad temporary hack. Page breaks currently are calculated by looping inside of the output routine instead of restarting it. In consequence, the headlines are not correct when material gets pushed to the next page. In a similar vein, floats like tables and figures might appear too soon. This will get solved with the next iteration of the package, after which a regular release should be possible.

It is not entirely clear how to deal satisfactorily with floats: if the first page size calculation results in a float being moved to the next page, and then it is determined that enough space on the current page is available for placing the float, doing so will significantly *reduce* the available space for the main text.

References

[1] `http://sarovar.org/projects/bigfoot`
(developer site and CVS instructions)

[2] `CTAN:macros/latex/contrib/bigfoot`
(released packages)