Through the \parshape, and what Joseph found there

Joseph Wright

1 Paragraph shape

The shape of the paragraph defines how text looks on the page. To talk about paragraph shape, we first need to think about how it relates both to the text itself and to the outer 'container' in which we are placing the paragraph (see Figure 1). Text in paragraphs is placed in vertical containers, 'galleys', which are themselves then placed on the page. The galley edges (thick black lines in the figure) may be separated by a margin from the edges of the paragraph (the light grey box in the figure). The paragraph shape itself may be a simple rectangle, as illustrated, or may be a more complex shape, as we will see below. Within that shape, the text itself is placed on a line-by-line basis. Not all of those lines of text will necessarily use the full width of the paragraph shape: the last line is often ended short of the margin, while in many styles the first line of a paragraph has a marker indent.

TEX provides us with a variety of primitives which are in some way linked to the shape of a paragraph. *TEX by Topic* (Eijkhout, 1992) lists seven primitives in the 'Paragraph Shape' chapter:

- \parindent
- \hsize
- \leftskip
- \rightskip
- \hangindent
- \hangafter
- \parshape

The \parindent primitive can be thought of as controlling appearance *within* the paragraph shape, whilst \hsize controls what the shape has to fit within. Both \leftskip and \rightskip are linked more to justification than to paragraph shape. This leaves \hangindent, \hangafter and \parshape to set up the shape of the paragraph itself. And the two \hang... primitives are in fact special cases of \parshape, so if we want to think about paragraph shape in TEX primitive terms we must focus on \parshape.

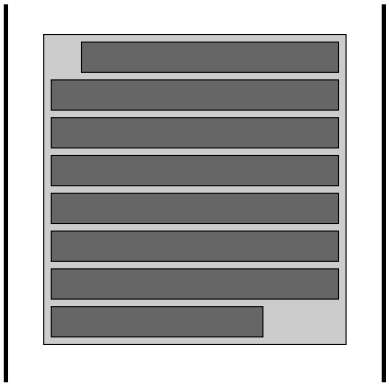

Figure 1: Paragraph shape

2 Views of \parshape

Matching up the TEX view of paragraph shape with the ways we can think about design requires us to think carefully about manipulating \parshape. Different design elements interact; thus, creating complex layouts by hand is time-consuming. Rather than do this, an alternative approach is to conceptualise these different design aspects and to provide interfaces (and data structures) for each of them. We can then construct the necessary \parshape programmatically, freeing us to describe design in a more natural way (at the cost of the effort in creating the underlying logic).

As part of the experimental LATEX3 galley module (The LATEX3 Project, 2015), the team has been exploring how we can achieve this separation. Some of the concepts are easy to implement, whilst others are more challenging, particularly given the nature of the underlying TEX model. Here, I will survey the design concepts we have identified and how we have currently tackled each one. The focus is very much on ideas rather than code: for the latter, readers are encouraged to consult l3galley.pdf and xgalley.pdf, both available on CTAN.

2.1 Margins

The paragraph shape will have left and right margins separating it from the galley edges. (These margins may of course be of zero length.) This is by far the most straightforward part of the design description of a paragraph. We can add margins to a paragraph either by specifying the absolute distance from the edge of the galley, or by describing a margin relative to any existing margin. An interface for both of these requires only three data items

- The left margin
- The right margin
- Whether to apply these on a relative or absolute basis

A suitable \parshape can then be created to implement these requirements: existing margins can be tracked separately or can be recovered from the existing \parshape using the ε-TEX \parshapeindent and \parshapelength primitives.

2.2 Shapes

Within the margins, the next design concept we can identify is the most obvious one of all: an actual shape applying to the paragraph. Such shapes are commonly seen in lists, which may use either a first-indented or first-hanging design. To allow maximum flexibility in this area it is useful to minimise the number of fixed decisions, which leads to an interface which requires

- A number of 'normal' (unmodified) lines
- A list of indents from the left margin
- A list of indents from the right margin
- A flag to indicate if the normal margins should resume after the last modified line

2.3 Cutouts

In contrast to margins and fixed paragraph shapes, which typically apply to a particular part of a document (for example, all quotations, all headers, and so on), the third view of paragraph shape is used on a one-off basis. A 'cutout' is a section of the paragraph which is (normally) indented to allow space for the insertion of an independent element: almost always a figure, with the text wrapping around it.

The current interface for this element is based on

- The side of the paragraph to cut
- The number of normal lines to leave
- A list of indents to apply to altered lines

In contrast to shaping a paragraph, there is no question here that the normal line length will resume: a cutout applies to a strictly fixed number of lines.

2.4 Combinations

On their own, each of the three different design views for `\parshape` are clear. The challenge at a code level is allowing fluid combinations of one or more of them to occur without the user needing to know that they are implemented using a single primitive.

By separating out the design elements and more importantly by tracking them in appropriate data structures, this is achievable. Notably, whilst the expectation is that margins and shapes obey TEX groupings (in LATEX terms, they are tied to environments), cutout parts need to act globally within the galley they apply to.

3 Challenges

By far the most challenging concept in the design of paragraph shape is handling cutouts in a 'complete' sense. These constructs are unused in many document designs but where they are used, they throw up a wide range of issues.

The `\parshape` primitive is a rare example of a grid-like approach to typesetting in the TEX engine. The primitive works in terms of lines of text, an approach which only makes complete sense if the baseline-to-baseline distance is known. For simple designs this will be no issue, but once a rich mix of display-like elements (maths, headings, *etc.*) is included, creating cutout shapes which work reliably becomes much more challenging. At the same time, describing cutout elements is likely more naturally done using distances than numbers of lines: 'leave a space for a figure which is 5 cm high', for example. Accommodating these more complex design descriptions requires both more code and, more importantly, a more thorough focus on user expectations.

The link between `\parshape` and number of lines also shows in the fact that we can talk about unaltered lines at the start of a paragraph but not at the end. A short consideration of the implementation shows why this is: to deal with a paragraph 'bottom up' means breaking the lines to some fixed length, then finding if any of the lengths need altering, then re-breaking, *etc.* This contrasts with the 'top down' algorithm we have in TEX: for each line to break, we know the length allowed as part of the first pass.

Cutout parts are usually used in cases where they need to be applied as a single block: it is no good leaving part of the space for a wrapped figure at the bottom of one page and the remainder at the top of the next! Handling this requires some information from the page-breaking system, and a definition of a cutout which includes a floating element.

However, the biggest single challenge in using the new code is that `xgalley` requires full control of the `\par` primitive, and in manipulating `\parshape` we require that no other code modifies it. As such, whilst at present the `xgalley` code works well in controlled tests, it is likely to break badly with 'real life' documents. This is an area we are currently addressing.

References

Eijkhout, Victor. *TEX by Topic*. Addison-Wesley, Wokingham, United Kingdom, 1992. `http://www.eijkhout.net/texbytopic`.

The LATEX3 Project. "The `xgalley` package". Available on CTAN: `macros/latex/l3experimental/xgalley`, 2015.

◇ Joseph Wright
Morning Star
2, Dowthorpe End, Earls Barton
Northampton NN6 0NH UK
`joseph.wright (at) morningstar2.co.uk`