Exploring `Underfull` or `Overfull` boxes and `badness` calculations

Overleaf Support Team

May 2020

NOTE: The calculations in this article will no longer be valid if you change the document text font.

1 Introduction

This article explores the meaning of `Underfull` and `Overfull` boxes and examines the calculation of `badness` using an `\hbox` with *finite* and *infinite* glues. We have tried to provide a range of examples to demonstrate various aspects of TeX engines' box-construction behaviour and the "diagnostic messages" issued by TeX engines during the box-creation process. It is unlikely that we have covered every possible scenario but hope to have provided sufficient material to "demystify" some aspects of TeX-based typesetting. We also hope to be forgiven for using various raw TeX commands within the examples `:-)`.

2 Background: fitting "stuff" into boxes

Before going on vacation most of us dust-down our suitcases to fill them with a collection of materials we need for the trip. Of course, a suitcase is a three-dimensional (3D) box with a (mostly) fixed width, height and depth. Into that 3D box we might try to squeeze as much as we need (or think we need...). If we're lucky, and can travel light, our "stuff" might not completely fill the case, so it will have excess space and be `Underfull`—or we may try to squeeze too much into it, and fail, because the suitcase is `Overfull`.

TeX engines face a not too dissimilar problem when tasked with packaging typeset material into the boxes it creates—TeX's boxes are restricted to 2D but with the vertical dimension subdivided into the box's depth and height. Deep inside TeX engines the process of creating boxes is referred to as *packaging*: in effect, it involves taking a list of typeset items and packing them into a container to provide a "boundary area" which defines how those items will be arranged on the typeset page. The resultant packaged-up content container (a box) produces an `\hbox` or one of the vertical variants: `\vbox`, `\vtop` or `\vcenter`.

3 Introduction to boxes and glue

All TeX engines (PdfTeX, XeTeX, LuaTeX *etc.*) typeset LaTeX documents using Donald Knuth's so-called boxes and glue model: your typeset text or mathematics is assembled into a "box" and a TeX engine uses flexible space called *glue* to adjust the positioning of material within the box, helping to achieve a desirable typeset result. For example, when breaking a paragraph into lines of typeset text a TeX engine uses so-called *interword glue* to adjust spacing between individual words to achieve an optimal linebreak.

The term "glue" is probably not ideal and it may have been preferable to describe TeX's flexible spacing as behaving like a spring—with its ability to stretch or shrink. Usage of the term glue was adopted early in the history of TeX's development—people quickly got accustomed to using "glue" so it soon became too late give it a better name, such as spring.

3.1 A little more about glue

If you are new to LaTeX you may not have encountered the lower-level concept of TeX glue so here an introduction to the key ideas. Because glue is a form of flexible spacing it is specified using three components: a fixed amount plus an amount it can stretch and an amount by which it can shrink. Glues can be specified to allow finite or infinite amounts of stretching and/or shrinking.

You can specify glue using commands such as `\hskip` or `\vskip`:

- horizontal glue: **`\hskip`** `<fixed amount>` **`plus`** `<amount to stretch>` **`minus`** `<amount to shrink>`
- vertical glue: **`\vskip`** `<fixed amount>` **`plus`** `<amount to stretch>` **`minus`** `<amount to shrink>`

where

- **`plus`** and **`minus`** are keywords that TeX engines understand
- `<fixed amount>` is referred to as the *normal* component of the glue
- `<amount to stretch>` is referred to as the *stretch* component of the glue
- `<amount to shrink>` is referred to as the *shrink* component of the glue.

`<fixed amount>` is required but `<amount to stretch>` and `<amount to shrink>` are both optional—if either, or both, are missing they are set to a value of 0.

In addition to `\hskip` and `\vskip` TeX engines provide other primitive (built-in) glue-related commands that we won't address here. Those commands include:

- horizontal glue: `\hfil`, `\hfill`, `\hfilneg`, `\hss`
- vertical glue: `\vfil`, `\vfill`, `\vfilneg`, `\vss`

together with `\mskip` for inserting glue in math expressions.

3.2 Finite and infinite glue units

For glues specified with `\hskip` and `\vskip` the normal component of glue (`<fixed amount>`) has to be expressed in “real world” units; however, TEX engines allow for the shrink component (`<amount to shrink>`) and the stretch component (`<amount to stretch>`) to be expressed in *two* types of unit:

- *finite* units: these are “real word” quantities such as points (`pt`), millimetres (`mm`), inches (`in`) *etc.*
- *infinite* units: these are “TEX world” quantities called `fil`, `fill` or even `filll`

TEX engines use those strange “infinite units” to create glue components that are infinitely stretchable or infinitely shrinkable.

Each of `fil`, `fill` or `filll` represent “different orders” of infinite flexibility, with each one being more flexible than the next:

`fil < fill < filll`

Glue components expressed in finite units are said to have *glue order* 0 but components expressed in infinite units have glue order 1, 2 or 3, as shown in the following table:

Units	**Glue order**
`pt`, `mm`, `in`, *etc.*	0
`fil`	1
`fill`	2
`filll`	3

Putting this all together:

- the normal component of glue (`<fixed amount>`) is mandatory and has to be expressed in “real world” units, so it is always glue order 0. Note that the normal component can have zero width by specifying 0pt, 0mm *etc.*
- the shrink component (`<amount to shrink>`) and the stretch component (`<amount to stretch>`) of glue are both optional and can be expressed in finite units (glue order 0) or infinite units of `fil`, `fill` `filll` with glue order 1, 2 or 3 respectively

Just to add another term, the value of a glue’s normal component is also called its *natural width* which is the amount of space occupied by that glue if no stretching or shrinking takes place. Interested readers can find more information and a detailed worked example in the Overleaf article How TEX Calculates Glue Settings in an `\hbox`.

3.2.1 Example: glue with finite components

Glue can be specified to have stretch or shrink components using any combination of finite or infinite units but we’ll start with an example of glue with finite stretch and shrink.

If we specify a glue example such as `\hskip2pt plus 1pt minus 0.5pt` this glue is comprised of the following:

- The *normal component* value is 2pt (the normal component is always a finite value). This glue is also said to have a *natural width* of 2pt.
- The `plus` keyword specifies the amount of *stretch*. Here, the glue has a finite stretch component value of 1pt.
- The `minus` keyword specifies the amount of *shrink*. Here, the glue has a finite shrink component value of 0.5pt.

By specifying this glue we are providing the TEX engine with "flexible space" that is normally 2pt wide but we *recommend*, to the TEX engine, this glue can stretch by up to 1pt or shrink by up to 0.5pt. So, ideally, this glue (flexible space) can shrink to become as small as 2pt − 0.5pt = 1.5pt or stretch up to 2pt + 1pt = 3pt.

If the TEX engine needed to fill a particular box it could use our glue to create some space which occupies anything from 1.5pt up to 3pt. If, for whatever reason, the TEX engine did not need to stretch or shrink this glue it would be used at its *natural width* of 2pt—its normal component value.

Caveat: As we'll see below, TEX engines *will over-stretch finite glue* beyond our recommended value for `<amount to stretch>` but they *will not over-shrink finite glue* below the amount specified by our `<amount to shrink>`. TEX engines consider over-shrinking to be a "very bad thing indeed!".

3.2.2 Example: glue with infinite components

We can rewrite our previous example with infinite glue using `\hskip2pt plus 1fil minus 0.5fil` which comprises the following:

- The *normal component* value is 2pt (the normal component is always a finite value). This glue is also said to have a *natural width* of 2pt.
- The `plus` keyword specifies the amount of *stretch*. Here, it has an infinite stretch component value of `1fil`.
- The `minus` keyword specifies the amount of *shrink*. Here, the glue has an infinite shrink component value of 0.5pt.

Our glue has "infinite" stretch and "infinite" shrink (in units of `fil`) so it can stretch or shrink to any desired value.

3.2.3 Mixing finite and infinite glue components

You can specify glue which has a mixture of finite and infinite components, such as `\hskip2pt plus 1fil minus 0.5pt`. For this particular glue we would say it has an infinite stretch component (from the `plus 1fil`) but finite shrink component (from the `minus 0.5pt`). This glue can stretch out to any desired value but can only shrink down to a minimum of 1.5pt.

3.2.4 Examples of glue errors

If you try to use infinite glue as the normal component and write something like

```
\hskip 1fil plus 2pt minus3pt
```

you'll receive an error:

```
! Illegal unit of measure (pt inserted).
<to be read again>
                   f
l.57 \hskip 1f
          il plus 2pt minus3pt
Dimensions can be in units of em, ex, in, pt,
pc, cm, mm, dd, cc, nd, nc, bp, or sp;
but yours is a new one!
I'll assume that you meant to say pt, for printer's points.
...
```

If you try to omit the normal component completely and write something like

```
\hskip plus 2pt minus3pt
```

you'll see errors such as these:

```
! Missing number, treated as zero.
<to be read again>
                  p
l.93 \hskip p
             lus 2pt minus3pt
A number should have been here; I inserted `0'.
(If you can't figure out why I needed to see a number,
look up `weird error' in the index to The TeXbook.)

! Illegal unit of measure (pt inserted).
<to be read again>
                  p
l.93 \hskip pl
          us 2pt minus3pt
Dimensions can be in units of em, ex, in, pt,
pc, cm, mm, dd, cc, nd, nc, bp, or sp;
but yours is a new one!
I'll assume that you meant to say pt, for printer's points.
...
```

4 Back to boxes

We noted that the task of creating boxes is referred to as *packaging*: fitting a list of typeset items into a container which provides a "boundary area" defining how those items will be arranged on the typeset page. A vital part of the packaging process is to determine how the *glues* inside that box will behave: will they stretch or shrink?

If so, which ones and by how much? The placement of individual items within the enclosing box area/dimensions is largely determined by stretching or shrinking of glues available in that box.

4.1 Packing boxes

Each time a TEX engine is asked to package a list of typeset items into a new box it starts out by setting that box's `badness` value to 0—only if it needs to, will TEX later calculate an actual `badness` value for that box. The TEX engine proceeds to examine all the individual items contained in the list supplied for packaging—those items might be characters (glyphs), rules, glue, penalties, other boxes and so forth. Based on the nature of each item, TEX will determine how each of them might affect the dimensions of the box it is constructing. The goal is to determine the *natural size* of the box which is then compared to the user's *desired size* so that TEX can work out how to stretch or shrink the glue.

TEX engines take particular care with any glue values detected in the list to be packaged. Only the normal component (natural width) of glue directly contributes to the *natural size* dimensions of the box; the glue stretch and glue shrink components do not. Of course, a TEX engine does not completely ignore the stretch and shrink component values of glues in the list being packaged. The TEX engine keeps a running total of how much glue stretch and glue shrink it has seen for each *order* of glue. Recall that finite units of glue have glue order 0 whereas infinite units of fil, fill and filll have glue order 1, 2 and 3 respectively.

After the TEX engine has examined all items in the list to be packaged-up it will have calculated the natural size of the box. Recall that for glue items their natural size is their normal component (the `<fixed amount>`).

TEX next compares the natural size with any desired size set by the user—for example, you set the desired size to `<dimension>` when writing `\hbox to <dimension>{...}`. Note, if you do not specify a size for a box and write `\hbox{...}` or `\vbox{...}` those boxes will be have the natural size of the components within them and the glues will not need to shrink or stretch.

Clearly, the user can set a desired size which is the same as, greater than or less than the box's natural size. If we denote the box's natural size by S_N and the user's desired size by S_D then there are three possibilities:

1. $S_N > S_D$: the natural size is greater than the desired size, so the box contents (i.e., the glues) need to *shrink*.
2. $S_N < S_D$: the natural size is less than the desired size, so the box contents (i.e., the glues) need to *stretch*.
3. $S_N = S_D$: the natural size is the same as the desired size, so the glues do not need to shrink or stretch. They will use their normal width value (their natural width arising from the `<fixed amount>` value).

After examining S_N and S_D the TeX engine knows if the glues need to stretch, shrink or assume their natural width.

We noted that during box construction TeX engines keep a running total of the amount of stretch and shrink—for each glue order—seen in the list of items being packaged into a box. To select which glues will provide the necessary stretching or shrinking, a TeX engine checks the values of total stretch or total shrink *for each glue order* and picks the highest order that that has a non-zero value: that is how the glues are selected for each box—the highest glue order wins the race. For example, if a particular box needed to stretch its glue and the sum total of all stretch components was as follows:

Units	**Glue order**	**Total (stretch)**
`pt`, `mm`,`in`, *etc.*	0	50 (`pt`)
`fil`	1	5 (`fil`)
`fill`	2	0 (`fill`)
`filll`	3	1 (`filll`)

TeX would choose the glue(s) with `filll` because it is the highest glue order that has a non-zero total. In this box, only glues with glue order 3 (`filll`) will be "activated" to take part in stretching: all glue(s) of a lower glue order will adopt a size determined by their natural width.

Further details and a fully worked example can be found in the Overleaf article How TeX Calculates Glue Settings in an `\hbox`

5 Boxes and their glue settings

Once constructed, each box will have a certain width, height and depth. In addition, just after creating a new `\hbox`, `\vbox`, `\vtop` or `\vcenter` the TeX engine will assign three glue-related parameter values to that freshly-minted box to record the stretching or shrinking behaviour of any glue within the list of items "inside" that box. Those parameters are used to "set" the glue in that box: making the glues occupy a particular amount of space by stretching or shrinking to the values necessary to arrange the items as required. Those three parameters are called the *glue set ratio*, *glue order* and *glue sign*.

- *glue order*: An integer which determines the order of glue to be activated inside this box—finite glue (order 0) or one of the infinite glues fil, fill or fill with glue order 1, 2 or 3 respectively.
- *glue sign*: An integer (0, 1 or 2) which determines whether glues will neither shrink or stretch (0), only stretch (1) or only shrink (2)
- *glue set ratio*: A floating point number which is a multiplier applied to glue components and determines by how much each glue will stretch or shrink.

The glue order and stretching/shrinking of glue is utilized when a box is physically written out to the typeset page in the PDF file, enabling the box contents to be arranged on the output page as required.

Trivia: Calculation of glue set ratio is the only place that TEX engines use floating point calculations.

5.1 *glue order*: an important parameter

The *glue order* assigned to a box is key to understanding the origin of `Underfull` and `Overfull` box warnings. The glue order determines which glues in a box are going to participate in any stretching or shrinking activities. For example, if a particular box is assigned a glue order of 2 it means that only those glues with stretch or shrink components in units of `fill` are to be used—all glues with a lower glue order do not take part and will occupy an amount of space determined by their natural width: the value of their *normal component*. The *glue sign* value determines whether those `fill` glue components are going to stretch or shrink.

5.2 Glues and quality control: underfull or overfull?

During creation of boxes destined to contain your typeset material, a TEX engine examines the resulting box's "quality" by checking how much "certain glues" (see below) need to stretch or shrink to produce your box. The result of those checks are reported to you using `Underfull` or `Overfull` box diagnostic messages which tell you the box's "`badness`" value, or an amount by which your content exceeds the given size (`<dimension>`) of that box. Note that is only during the initial box-creation process that TEX engines issues `Underfull` or `Overfull` box diagnostic messages.

Recall that within each box *only* those glues which have a particular *order*—as identified by the box's *glue order*—will participate in stretching or shrinking to align or arrange the contents of the box. The key point to note is that TEXwill *only* report `Underfull` or `Overfull` boxes if, during the box's creation, the TEX engine identifies that glues of order 0 need to be used for stretching or shrinking–the "certain glues" referred to above is glues of order 0.

If a box has glue order 1, 2 or 3 it means that glues containing infinite shrink or stretch components (in units of `fil`, `fill` or `filll`) are to be used (or "activated") for stretching or shrinking. Consequently, TEX *will not* report `Underfull` or `Overfull` warnings for such boxes because infinite glues are *specifically designed* to stretch or shrink to any desired value—TEX will not consider them to be `Underfull` or `Overfull`.

You can typeset the `badness` value for the most recently constructed box by writing `\the\badness` *immediately after* that box is created. As soon as another box is constructed the `badness` value for the previous box is lost because it is not stored or "attached" to the box which produced it. `badness` is a transient value calculated on-the-fly during box construction—but only if TEX needs to do so (i.e., when the glue order is 0).

"Box-fitting" processes are an intrinsic aspect of all TEX engines—such as during line-breaking where individual lines of a typeset paragraph are converted to an `\hbox` of a fixed width, or the content of each page is fitted into a `\vbox` of a fixed height. During linebreaking or page construction TEX engines will report `Underfull` or `Overfull` box diagnostic messages which result from boxes that you have not specified explicitly. At first, it can be confusing to see numerous `Underfull` or `Overfull` `\hbox` warnings which arise from linebreaking processes and calculations, not from boxes that *you* have directly specified in your TEX or LATEX code. Hopefully, this project will help you to better understand TEX engines' `Underfull` or `Overfull` box messages, irrespective of their source.

5.3 Using an \hbox to explore badness

To obtain the width of a piece of text it is customary to put that text in an `\hbox` and measure the width of that `\hbox` using `\the\wd\<boxnumber>`. For example, if we do

```
\setbox100=\hbox{Overleaf}\the\wd100
```

TEX informs us that the word "Overleaf" has width 34.97989pt (using the current font). What happens if we tell TEX to fit "Overleaf" into an `\hbox` that is just a tiny bit wider, say 0.00989pt, than we actually need to accommodate "Overleaf". We will ask the TEX engine to use an `\hbox` of width 34.98978pt compared to 34.97989pt, which is actual width of the text. That's really not so bad… is it?

If we write

```
\setbox100=\hbox to 34.98978pt{Overleaf}
```

we get a badness value of 10000!

What happened? Why did TEX report a badness value of 10000 for such a tiny increment in the width of the `\hbox`? The reason is that our `\hbox` has absolutely no glue that TEX can use to absorb even the tiny extra width of 0.00989pt so it assigns a value of 10000 to the badness of `\hbox100`.

Conversely, if we tell TEX to fit "Overleaf" into an `\hbox` that is 0.00989pt *narrower* than we actually need to accommodate "Overleaf", i.e.,

```
\setbox100=\hbox to 34.97pt{Overleaf}
```

then we obtain a badness value of 1000000—which is *significantly* higher than `badness` of 10000 reported for `\hbox 100` when it was 0.00989pt too wide! In essence, TEX engines *deplore* the use of boxes for which the content is too large for a given box and there is insufficient glue to provide the amount of shrinking required to absorb the extra space required by your content—hence TEX engines assign the special `badness` value of 1000000.

Trivia: Inside the source code of TEX the `badness` value of 10000 is referred to as "infinitely bad" but no name is given to the `badness` value of 1000000—maybe that value is so bad that it leaves TEX speechless…

5.4 Adding a small but *finite* glue

5.4.1 But first: `\hbadness`

TEX engines provide the `\hbadness<threshold>` command which you can use to set a minimum `<threshold>` value for reporting the `badness` value of `\hboxes`. Any `\hbox` whose `badness` below (or equal to) the `<threshold>` (integer) value will not be reported as `Underfull`. Here, we'll assume LATEX's standard `\hbadness` value of 1000—which LATEX sets using `\hbadness=1000`. Note that the "=" sign is optional, you can also set `\hbadness` by writing `\hbadness1000`. There is an equivalent setting for `\vboxes`: `\vbadness`.

The current value of `\hbadness` can be typeset using `\the\hbadness`: the current value of `\hbadness` is 1000.

5.4.2 ... back to the glue

We'll build on our earlier example `\hbox` which was 0.00989pt *wider* than we need to accommodate "Overleaf". Here, we'll add some glue to our `\hbox` which provides the TEX engine with some "flexible space" it can use to absorb the additional 0.00989pt of space inside our `\hbox`.

Assuming LATEX's standard `\hbadness` value of 1000, we can add a tiny amount of flexible, but *finite*, `\hskip` glue to our `\hbox`—`\hskip0pt plus`δ where we'll vary δ from 0.01pt down to 0.002pt:

```
\setbox100=\hbox to 34.98978pt{Overleaf\hskip0pt plusδ}
```

In the following examples we see very different values of `badness` resulting from the values of δ, even though differences in the glue are extremely small:

δ = 0.01pt gives `badness` = 96. This is < `\hbadness` (1000) and not reported.

δ = 0.007pt gives `badness` = 281. This is < `\hbadness` (1000) and not reported.

δ = 0.006pt gives `badness` = 446. This is < `\hbadness` (1000) and not reported.

δ = 0.005pt gives `badness` = 768. This is < `\hbadness` (1000) and not reported.

δ = 0.004pt gives `badness` = 1509. This is > `\hbadness` (1000) and is reported.

δ = 0.003pt gives `badness` = 3547. This is > `\hbadness` (1000) and is reported.

δ = 0.0025pt gives `badness` = 6157. This is > `\hbadness` (1000) and is reported.

δ = 0.002pt gives `badness` = 10000. This is > `\hbadness` (1000) and is reported.

The last 4 examples with δ = 0.004pt, 0.003pt, 0.0025pt, 0.0020pt all produce `Underfull` box warnings with increasingly large badness values—even though the differences in δ are minute. Note that between δ = 0.003pt and δ = 0.002pt the `badness` rises rapidly—but that is due to the `badness` calculation having a "discontinuity" designed by Knuth: any calculated badness above $2^{13} = 8192$ is reported as 10000 which TEX considers "infinitely bad".

If we now set `\hbadness=1509` (or use `\hbadness1509`) and repeat one of our experiments with $\delta = 0.004$pt our box still has the same `badness` value but is no longer reported as `Underfull`:

$\delta = 0.004$pt gives `badness` = 1509. This is equal to `\hbadness` (1509) and not reported.

5.5 Infinitely flexible glue

In the above examples, TeX reported non-zero badness values because we had used a small, but *finite*, glue after "Overleaf". If we repeat these experiments but switch to using *infinite* glue, an `Underfull` box is not produced and the badness is always 0. If we now add tiny amounts of flexible `\hskip` glue but this time with *infinite* flexibility—`\hskip0pt plus`δ where we'll vary δ from 0.01`fil` down to 0.002`fil`—we see `badness` is 0 in every case.

```
\setbox100=\hbox to 34.98978pt{Overleaf\hskip0pt plusδ}
```

$\delta = 0.01$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

$\delta = 0.007$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

$\delta = 0.006$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

$\delta = 0.005$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

$\delta = 0.004$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

$\delta = 0.003$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

$\delta = 0.0025$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

$\delta = 0.002$`fil` gives `badness` = 0. This is < `\hbadness` (1000) and not reported.

The reason for the different behaviour of glue with *finite* stretch (e.g., $\delta = 0.01$pt *etc.*), compared to glue with *infinite* stretch (e.g., $\delta = 0.01$`fil` *etc.*) is that TeX engines *only* calculate `badness` for boxes which have to stretch or shrink glue with order 0. Here, the glue in the `\hbox` has infinite stretch in units of `fil`, so it will always have `badness` of 0 and is not considered to be `Underfull`—`fil` glue is designed to stretch/shrink to any size.

5.6 Calculating badness

If the amount of available *finite* glue is non-zero, TeX calculates `badness` using an *approximation* to the formula

$$\texttt{badness} = 100 \times \left(\frac{\text{amount of space to fill or absorb}}{\text{amount of finite glue available}} \right)^3$$

A TeX engine is satisfied with an approximate value for `badness` because it is used as an heuristic—it isn't necessary to calculate an *exact* value, allowing TeX to keep the calculation fast and efficient. Note that in the source code of the TEX program Knuth writes

> "Any badness of 2^{13} or more is treated as infinitely bad, and represented by 10000."

where $2^{13} = 8192$. Knuth also observes that his approximation to computing the value of the `badness` function

> "... is capable of computing at most 1095 distinct values, but that is plenty."

Taking one of our examples above, where we had to fill an additional space of 0.00989pt but only had 0.004pt of (*finite*) glue available, TEX reported a `badness` value of 1509 but an exact calculation of `badness` would be

$$\texttt{badness} = 100 \times \left(\frac{0.00989}{0.004}\right)^3 = 1511.502...$$

but 1509 is close enough for TEX's purposes.

5.7 Over-stretching but never over-shrinking of *finite* glue

Let's use `\hbox to 16pt{a<glue>b}` with various values of `<glue>` between a and b.

Using `\setbox101=\hbox{ab}\the\wd101` we can determine that the width of the text typeset ab (allowing for kerns) is 9.49998pt.

Just out of interest, we can check for kerning by calculating the width of the individual characters and testing if their combined individual widths has the same value as the `\hbox{ab}`—i.e., when they are typeset together. Width of character 'a' is given by `\setbox101=\hbox{a}\the\wd101=` 4.56999pt, and the width of 'b' is given by `\setbox101=\hbox{b}\the\wd101=`4.93pt, giving a combined width of 9.49999pt—which, up to 4th decimal place, is the same as the width when typeset together: 9.49998pt, so we can assume there is no kerning. The difference of 0.0001pt is a rounding error—it would produce rather invisible kerning!

5.7.1 Over-stretching of *finite* glue

`\hbox to 16pt{a b}` produces a   b with `badness` =3271. But where did the stretchy glue come from? The answer is the space character between a and b: TEX converted that space character to a "glob" of interword glue—the value of which is specific to each font (and font size). Here, TEX had to over-stretch the glue (arising from the space character) resulting in a `badness` value of 3271.

`\hbox to 16pt{a\hskip 1pt plus 2pt b}` uses an explicit `\hskip` glue with only `2pt` of stretch (via the `plus 2pt`) and produces a result similar to using a space character: it results in a   b with `badness`=2073.

We can work out why. The desired width of our box is 16pt but what is the natural width of the components? We know that the width of 'ab' is 9.49999pt but what about the glue? Recall that the glue's normal component contributes to the natural width, so a glue specified using `\hskip 1pt plus 2pt` will contribute 1pt to the natural width of the box, giving a total natural width of 9.49999 + 1 = 10.49999pt, meaning that the glue has to stretch to fill 16 − 10.49999 = 5.50001pt of space. Our glue has a total 2pt of finite stretch but needs to fill up 5.50001pt of space in this box. Applying our `badness` calculation

$$\texttt{badness} = 100 \times \left(\frac{\text{amount of space to fill or absorb}}{\text{amount of finite glue available}}\right)^3$$

we can plug in our values to calculate

$$\texttt{badness} = 100 \times \left(\frac{5.50001}{2}\right)^3 = 2079.698\ldots$$

but 2073 is close enough for TEX's purposes.

Because we used small finite glue values—from the interword space or the 2pt in our explicit `\hskip` glue—TEX was forced to significantly over-stretch those finite glues, which resulted in the `badness` values of 3271 and 2073.

If we add more stretch by using `\hbox to 16pt{a\hskip 1pt plus 5pt b}` the `\hskip` glue has `5pt` of stretch (via the `plus 5pt`) and produces the result a b with `badness`=132. Again, we can check this by calculating

$$\texttt{badness} = 100 \times \left(\frac{5.50001}{5}\right)^3 = 133.100\ldots$$

which TEX approximates to a `badness` value of 132. The glue has much more stretch with which to fill the box, hence the smaller `badness` value.

5.7.2 No over-shrinking of *finite* glue

The previous example showed that a TEX engine will, if necessary, over-stretch finite glues by far more that we specify (recommend). In the above example we specified `\hskip 1pt plus 2pt`, allowing 2pt of finite stretch but the TEX engine ignored our *recommendation* because it was forced to stretch it to 5.50001pt to fill the `\hbox`. However, although TEX engines will over-stretch finite glue, *they will not over-shrink* finite glues.

5.7.3 An example with larger font size

To make this example easier to discuss we'll use a very large point size of 50pt to display our text (also in blue):

```
\setbox100=\hbox{{\color{blue}\fontsize{50}{50}\selectfont ab}}
```

The width of ab is 47.49997pt.

Next we'll use `\hbox to 40pt{a\hskip 0pt minus 7.49997pt b}` which asks the TEX engine to typeset an `\hbox` that is exactly 7.49997pt smaller than the width of the text 'ab' (typeset at 50pt).

It produces ab with `badness=100`. We used an explicit `\hskip` glue with only `7.49997pt` of shrink (via the `minus 7.49997pt`). TEX uses that `minus 7.49997pt` to "squeeze" our characters ab into this box—but only just, resulting in a "perfect fit `badness`" value of 100. To understand the `badness` value of 100 we only need to refer back to the calculation of `badness`:

$$\texttt{badness} = 100 \times \left(\frac{\text{amount of space to fill or absorb}}{\text{amount of finite glue available}}\right)^3$$

In our box we had 7.49997pt as the amount of finite glue available and 7.49997pt of space to fill (or in our case, absorb), hence:

$$\texttt{badness} = 100 \times \left(\frac{7.49997}{7.49997}\right)^3 = 100 \times (1)^3 = 100$$

What happens if we now reduce the width of the `\hbox` to, say, 30pt via

```
\hbox to 30pt{a\hskip 0pt minus 7.49999pt b}
```

asking TEX to typeset an `\hbox` that is exactly 17.49997pt smaller than the width of the text ab in 50pt type. We still use an explicit `\hskip` glue with only `7.49997pt` of shrink (via the `minus 7.49997pt`) but this time the requested `\hbox` is now 17.49997pt too small: will TEX over-shrink our glue of 7.49997 to absorb the 17.49997pt—squashing the a and b closer together to absorb the extra 10pt reduction in the size of `\hbox` we have requested?

It produces ab with `badness=1000000`.

TEX honoured our request for a 30pt wide `\hbox` as you can see because the text of "with `badness` ..." overlaps the blue text. However, the ab in the 30pt-wide `\hbox` are typeset with exactly the same overlap as the 40pt-wide `\hbox`: the glue between them was not shrunk by more than the 7.49997pt because TEX simply will not shrink finite glue below the minimum size stipulated in its specification. This is, of course, unlike the finite stretch component of glues which TEX will happily over-stretch whenever necessary. Here, the `\hskip 0pt minus 7.49997pt b}` will absorb up to 7.49997pt but no more.

We see that TEX has reported `Overfull \hbox(10.0pt too wide) detected at line...` as you can also see in Overleaf. It also assigned this box the "special maximum" `badness` of 1000000 because there was insufficient shrinkable glue to absorb the excess space of 10pt.

If we increase the shrink to 17.49997pt and write

```
\hbox to 30pt{{\color{blue}\fontsize{50}{50}\selectfont
a\hskip 0pt minus 17.49997pt b}}
```

we duly get ab with `badness`=100. A 30pt `\hbox` with characters 'ab' overlapping.

So, TEX will happily overlap the characters ab but *only* if there is sufficient glue shrink to absorb the excess space occupied by the text ab—here, accommodating a 30pt box size request by shrinking the space between a and b by 17.49997pt.

5.7.4 Trying a tiny bit of infinite shrink

We'll try reducing the `\hbox` right down to 20pt wide by using a tiny amount of infinitely shrinkable glue (`0.001fil`) to see what happens.

```
\hbox to 20pt{{\color{blue}\fontsize{50}{50}\selectfont
a\hskip 0pt minus 0.001fil b}}
```

we duly get ab with `badness`=0.

It has badness=0 *and* is no longer reported as being overfull. The difference in behaviour arises because we used infinitely shrinkable glue (`0.001fil`).

We can go even further and create a zero width `\hbox`:

```
\hbox to 0pt{{\color{blue}\fontsize{50}{50}\selectfont
a\hskip 0pt minus 0.001fil b}}
```

we duly get ba with `badness`=0.

The `0.001fil` glue has shrunk to absorb the entire width of the box, a total of 47.49997pt. In effect, it typesets the 'a' then "backspaces" by 47.49997pt and typesets the 'b'.

We'll fake this (using red text) with a `\kern-47.49997pt`:

```
\hbox{{\color{red}\fontsize{50}{50}\selectfont
a\kern-47.49997pt b}}
```

ba

Result typeset with `\kern-47.49997pt`.

ba

Result typeset with `0.001fil` glue shrink.

To see exactly what is going on, let's look at the placement of text.

Width of large red a is 22.84998pt

Width of large red b is 24.65pt

When TeX typesets ba it first moves forward by a (+22.84998pt) then moves back by −47.49997pt (via `\kern-47.49997pt` or `0,001fil`) then typesets b moving forward again by (+24.65pt), giving a total movement of 0pt (within TeX's rounding errors!).

5.8 Understanding `\hfuzz`

Here, we'll take a look at the primitive (built-in) TeX command `\hfuzz` which can be used to influence TeX's decision as to whether it considers a box to be `Overfull` and reported as such.

Let's go back to our first example, `\setbox100=\hbox{Overleaf}\the\wd100` where TeX informs us that (using the current font) the word "Overleaf" has width 34.97989pt.

Putting "Overleaf" into an `\hbox` that is exactly 34pt:

```
\setbox100=\hbox to 34pt{Overleaf}
```

we get a corresponding badness value of 1000000 and a message `Overfull \hbox (0.97989pt too wide) detected at line...`, which is not surprising.

If we now set `\hfuzz=0.97989pt` and repeat the experiment `\setbox100=\hbox to 34pt{Overleaf}` we still get an `\hbox` with badness of 1000000 but `\hbox` 100 is no longer reported as `Overfull` .

In the next section we'll take a closer look at `\hfuzz`.

5.8.1 Adding shrink glue: effect on Overfull warning and `\hfuzz`

When TEX reports an `Overfull` `\hbox` or `\vbox` the amount (in points, pt) by which the `\hbox` is too wide or the `\vbox` is too high is calculated using difference between the excess width and the total amount of zero order shrink glue in the box. We'll work through some examples to explain this in more detail.

If we write `\hbox to 20pt{\kern25pt}\the\badness` to create an empty box we see space and badness of 1000000. The `\hbox` is set to 20pt but the `\kern` of 25pt is 5pt wider than the `\hbox` and there is no shrinkable glue available to absorb the 5pt. As a result, TEX sets the "special badness" value of 1000000 and reports `Overfull \hbox(5pt too wide) detected at line...`.

If we now add some flexible glue by writing

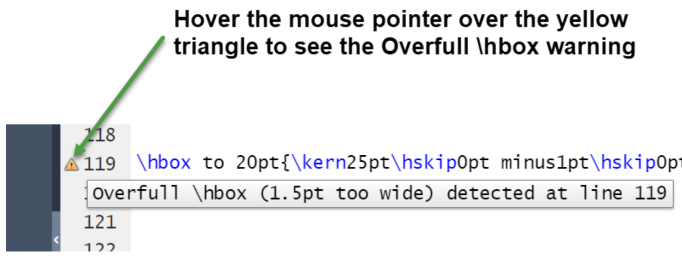
```
\hbox to 20pt{\kern25pt\hskip0pt minus1pt}
```

we see some empty space: and the same "special badness" of 1000000—also due to insufficient shrinkable glue being available. However, TEX now reports `Overfull \hbox(4pt too wide) detected at line...`. The box is no longer 5pt too wide but 4pt because TEX has accounted for the 1pt of flexible glue in its report of an `Overfull \hbox`.

Going further, if we now add more flexible glue by writing

```
\hbox to 20pt{\kern25pt\hskip0pt minus1pt\hskip0pt minus2.5pt}
```

there is a total flexible glue of 1pt + 2.5pt = 3.5pt. Now, the box is only 5pt − 3.5pt = 1.5pt too wide, as you will also see reported by Overleaf.


Hover the mouse pointer over the yellow triangle to see the Overfull \hbox warning

```
\hbox to 20pt{\kern25pt\hskip0pt minus1pt\hskip0pt minus2.5pt}
Overfull \hbox (1.5pt too wide) detected at line 119
```

This example demonstrates that the amount by which a box is reported as `Overfull` takes into account the total finite shrink glue available in the box—i.e., it is not just the excess width of the non-glue content. In our example, the `\kern25pt` exceeded the desired width of 20pt by 5pt. The total flexible shrink is now capable of absorbing 3.5pt but that still leaves 1.5pt that cannot be absorbed by shrinking the glue—1.5pt is the amount by which the `\hbox` is reported as `Overfull`. Recall that TEX engines *will not* shrink finite glue below the minimum size specified by its shrink component value.

If we want to stop TEX reporting this `Overfull \hbox` we need to set `\hfuzz=1.5pt`, after which TEX will no longer report this space created via an empty `\hbox` as being `Overfull`. `\vfuzz` is the equivalent command for `\vbox`es.