The repltext package*

Scott Pakin
scott+repl@pakin.org

September 25, 2020

1 Introduction

The repltext packages exposes to LaTeX a relatively obscure PDF feature: replacement text. When replacement text is specified for a piece of text, it is the replacement text, not the typeset text that is copied and pasted. Try selecting and copying the following sentence and pasting it into some other document:

I love to eat celery.

If your PDF reader supports replacement text—Adobe Acrobat and Acrobat Reader currently appear to be the only ones—the pasted text will say "ice cream" instead of "celery". Why is this useful? The PDF specification presents the following usage model [1]:

> Just as alternate descriptions can be provided for images and other items that do not translate naturally into text ... replacement text can be specified for content that does translate into text but that is represented in a nonstandard way. These nonstandard representations might include, for example, glyphs for ligatures or custom characters, or inline graphics corresponding to letters in an illuminated manuscript or to dropped capitals.

Hence, one might imagine using replacement text in a sentence such as,

Have you ever seen ♱ perform live in concert?

although in the context of LaTeX, mathematical typesetting may present a more practical use for replacement text:

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

*This document corresponds to repltext v1.1, dated 2020/09/25.

2 Usage

The repltext package works only with pdfLaTeX or LuaLaTeX as it requires certain primitives that only those TeX engines provide. To use repltext, simply include `\usepackage{repltext}` in your document's prologue. There are currently no package options.

`\repltext {`⟨*replacement text*⟩`} {`⟨*general text*⟩`}`

This is the main command defined by repltext. It takes two arguments: the replacement text that goes into the copy buffer and some arbitrary LaTeX code to which the replacement text corresponds. That LaTeX code is typeset as normal; only its behavior when copied and pasted is different.

The replacement text is included verbatim; no characters have any special meaning to LaTeX with the exception that curly braces must appear as properly nested, open and close pairs. For example, `\repltext{$7 or 50%}{seven bucks or half the total}`, produces

seven bucks or half the total ⇒ `$7 or 50%`

Because of ⟨*replacement text*⟩'s verbatim-like nature, if the "`$`" or "`%`" in the preceding `\repltext` command were backslashed, the replacement text would also include a literal backslash.

3 Limitations

repltext's most severe limitation is the dearth of PDF readers that support replacement text: as far as I know, only Adobe Acrobat and Acrobat Reader. A secondary limitation is that, in the Adobe programs, a selection backed by replacement text extends only to the first space or kern. This can be somewhat disconcerting to a person trying to select a piece of text. A workaround is to try to use only word-for-word substitutions and to prevent kerning by inserting "`{}`" at kerning points (which, alas, results in inferior typesetting). Contrast selecting "`Zippy drinks \repltext{fine wines}{Valvoline}.`"

Zippy drinks Valvoline.

with "`Zippy drinks \repltext{fine wines}{V{}alv{}oline}.`"

Zippy drinks Valvoline.

Yet another limitation, again in the Adobe programs, is that spurious spaces are sometimes, but not always, introduced into the copy buffer following a `\repltext`. I have yet to determine the source of those spaces and whether a workaround exists.

Finally, like `\verb`, `\repltext` does not behave as expected within an argument to another macro. `\repltext` does not currently detect and warn if it is being used as an argument, but the replacement text may be altered in surprising and undesirable ways. Consider, for example, `\fbox{\repltext{### BAD_BAD_BAD ###}{terrible}}`:

terrible ⇒ `###### BADBADBAD ######`

4 Implementation

This section presents the documented source code for repltext. Most users can ignore this section.

`\@ifdefined` I find `\@ifdefined` a more natural construct than `\@ifundefined`.

```
\providecommand{\@ifdefined}[3]{\@ifundefined{#1}{#3}{#2}}
```

`\repl@literal` repltext needs a mechanism to insert literal PDF code. This varies by TEX engine so we have to define `\repl@literal` on an engine-by-engine basis.

```
\@ifdefined{pdfliteral}{%
```

pdfLATEX

```
  \let\repl@literal=\pdfliteral
}{%
  \@ifdefined{pdfextension}{%
```

LuaLATEX

```
    \protected\def\repl@literal{\pdfextension literal}%
  }{%
```

Other

```
    \PackageError{repltext}{Unrecognized TeX engine}{%
      The repltext package currently works only with pdfLaTeX and\MessageBreak
      LuaLaTeX.\space\space Please use of those engines to build your document.%
    }%
  }%
}%
```

repltext additionally needs a mechanism to escape backslashes and parentheses in a string to make it usable as a PDF string. This is easy in pdfLATEX, which provides `\pdfescapestring`. Unfortunately, to my knowledge, no other TEX engine provides equivalent functionality. Rather than write our own function, we leverage hyperref's `\Hy@pstringdef`, which provides similar functionality.

```
\RequirePackage{etoolbox}
\AtEndPreamble{%
  \@ifpackageloaded{hyperref}{}{\RequirePackage{hyperref}}%
}
```

The \repl@text@ii macro invokes \scalebox, which is defined by the graphicx package.

```
\RequirePackage{graphicx}
```

Replacement text is stored temporarily in \repl@text@toks.

```
\newtoks\repl@text@toks
```

\repltext
\do

The \repltext user macro nominally takes as arguments the replacement text and its visual representation. However, we want the first argument to be interpreted as text, not as LaTeX code. For this to work, \repltext must in fact take no arguments. Instead, it marks all characters with category code 12 ("other") except for curly braces, which retain their original roles. It ends by storing all characters from "{" to "}" (i.e., what appears to the document author to be the first argument to \repltext) in \repl@text@toks and transferring control to the \repl@text@i macro.

```
\newcommand{\repltext}{%
  \bgroup
  \let\do=\@makeother
  \dospecials
  \catcode`\{=1
  \catcode`\}=2
  \afterassignment\repl@text@i
  \global\repl@text@toks=%
}
```

\repl@text@i

The \repl@text@i macro ends the group begun by \repltext. Doing so restores all characters to their previous category code. It then transfers control to the \repl@text@ii macro.

```
\newcommand{\repl@text@i}{%
  \egroup
  \repl@text@ii
}
```

\repl@text@ii
\repl@escaped

\repl@text@ii is the macro that finally outputs something. It takes ordinary LaTeX code as its argument and writes a PDF marked-content sequence that uses an ActualText entry to indicate that \repltext's first argument (currently stored in \repl@text@toks) is the replacement text for \repltext's second argument (\repl@text@ii's #1 argument).

```
\newcommand{\repl@text@ii}[1]{%
  \Hy@pstringdef\repl@escaped{\the\repl@text@toks}%
  \repl@literal{
    /Span << /ActualText (\repl@escaped) >>
    BDC
  }%
  #1%
```

It seems that either Adobe Acrobat (or perhaps the PDF specification itself; I'm not sure) requires the spanned item to include true PDF text, not just graphics. We

therefore include a microscopic piece of text to satisfy that requirement without being noticeable.

```
\scalebox{0.000001}{-}%
\repl@literal{EMC}%
}
```

`\prevrepl` For the author's convenience we define `\prevrepl` to refer to the previous first argument of `\repltext`, interpreted as ordinary LaTeX code. It is intended to be used in the second argument of `\repltext` to present typeset text that can be copied and pasted like LaTeX source, as in `\repltext{$\sum_{i=1}^n \frac{1}{n^2}$}{\prevrepl}` (result: $\sum_{i=1}^{n} \frac{1}{n^2}$).

Because `\prevrepl` requires `\scantokens`, this macro requires ε-TeX. Fortunately, all modern pdfLaTeX and LuaLaTeX engines include ε-TeX support.

`\prevrepl` is not currently documented in the author documentation because I'm not sure it's a sufficiently useful macro to retain in repltext. For the time being, I'm leaving it in on the off chance that someone requests a feature like what `\prevrepl` provides.

```
\newcommand{\prevrepl}{%
  \expandafter\scantokens\expandafter{\the\repl@text@toks}%
}
```

References

[1] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from `http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf`.

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.