The axessibility package

Dragan Ahmetovic, Tiziana Armano, Cristian Bernareggi,
Michele Berra, Anna Capietto, Sandro Coriasco, Nadir Murru,
Alice Ruighi, Eugenia Taranto
Dipartimento di Matematica "G. Peano"
Università degli Studi di Torino
<anna.capietto at unito.it>,<sandro.coriasco at unito.it>

January 8, 2019

Abstract

PDF documents containing formulae generated by LaTeX are usually not accessible by assistive technologies for visually impaired people (i.e., by screen readers and braille displays). The package manages this issue, allowing to create a PDF document where the formulae are read by these assistive technologies, since it automatically generates hidden comments in the PDF document (by means of the /ActualText attribute) in correspondence to each formula. The package does not generate PDF/UA.

Contents

1 Introduction

This package focuses on the specific problem of the accessibility of PDF documents generated by LaTeX for visually impaired people. When a PDF document is generated starting from LaTeX, formulae are not accessible by screen readers and braille displays. They can be made accessible by inserting a hidden comment, i.e., an ActualText, similarly to the case of web pages. This can be made, e.g., by using the LaTeX package pdfcomment.sty. In any case, this task must be manually performed by the author and it is surely inefficient, since the author should write the formulae and, in addition, insert a description for each formula. Note also that the package pdfcomment.sty does not allow to insert special characters like 'backslash', 'brace', etc, in the comment. Moreover, with these solutions, the reading is bothered since the screen reader reads incorrectly the formula and then the correct comment of the formula. There are also some LaTeX packages that try to improve the accessibility of PDF documents produced by LaTeX. In particular, the packages accsupp.sty and accessibility_meta.sty have been developed in order to obtain tagged PDF documents. However, both packages do not solve the problem of the accessibility of formulae. The package accsupp.sty develops some interesting tools for commenting formulae using also special characters (possibility that is not available in the pdfcomment.sty package). Moreover, this is not an automatized method, since the comment must be manually inserted by the author. The package accessibility_meta.sty is an improved version of the package accessibility.sty. This package allows the possibility of inserting several tags for sections, links, figures and tables. However, even if these tags are recognized by the tool for checking tags of Acrobat Reader Pro, they are not always recognized by the screen readers. Moreover, this package does not manage formulae. Our package automatically produces an ActualText corresponding to the LaTeX commands that generate the formulae. This ActualText is hidden in the PDF document, but the screen reader reads it without reading any incorrect sequence before.

2 License

3 Prerequisites

The package **axessibility** requires the following packages: **accsupp, amsmath, amssymb, xstring**.

4 Package specification

If you use LaTeX2$_\epsilon$ simply add the following line in the preamble:

\usepackage{axessibility}

The package includes the following features:

- The commands

  \pdfcompresslevel=0
  \pdfoptionpdfminorversion=6

  that produce an uncompressed PDF document.

- The command

  \BeginAccSupp

  contained in the package **accsupp**, has been redefined so that the screen readers access the ActualText created by this command.

- The new commands

  \wrap#1
  \wrapml#1

  allow to store their input into an ActualText in the PDF document (e.g., the LaTeX commands for generating a formula), for single line and multiple line formulae environments, respectively

- The environments

  \begin{equation} ... **\end{equation}**
  \begin{equation*} ... **\end{equation*}**
  \[... \]
  \(... \)

  have been redefined. In each environment listed above, the command `\wrap` is inserted, together with the command `\collect@body`, so that all the content of the environment is automatically stored into an ActualText in the PDF document. The following multiline formula environments, defined in the **amsmath** package,

  \begin{align} ... **\end{align}**
  \begin{align*} ... **\end{align*}**
  \begin{alignat} ... **\end{alignat}**
  \begin{alignat*} ... **\end{alignat*}**
  \begin{flalign} ... **\end{flalign}**
  \begin{flalign*} ... **\end{flalign*}**
  \begin{gather} ... **\end{gather}**

\begin{gather*} ... \end{gather*}
\begin{xalignat} ... \end{xalignat}
\begin{xalignat*} ... \end{xalignat*}
\begin{xxalignat} ... \end{xxalignat}

have been similarly redefined, using the command `\wrapml`. The content of these environments, too, is now stored into an ActualText in the PDF document. The support for more multiline environments will be added in future versions of the package.

5 Usage

An author that wants to create an accessible PDF document for visually impaired people can add this package and use the above environments for inserting the formulae. The LaTeX code of the inserted formulae will be added as hidden comments in correspondence to the location of the formulae in the text. This will allow the user to access the formula code with the screen reader and with the braille refreshable display. Additionally, the package enables to copy the formula LaTeX code from the PDF reader and paste it elsewhere.

Note that, to preserve the compatibility with Acrobat Reader, our package discourages the use of the underscore character (_), which is not correctly read using screen readers in combination with this PDF reader. Alternatively, we suggest to use the equivalent command `\sb`.

Inline and displayed mathematical modes encoded by means of $ and $$ are not supported by the package. However, external scripts, provided as companion software and described in the following section, can address, at least partly, these cases. Moreover, provided that also the package **eqnalign** is added, the (old) multiline formula environments

\begin{eqnarray} ... \end{eqnarray}
\begin{eqnarray*} ... \end{eqnarray*}

will automatically generate the corresponding hidden ActualText.

6 External scripts and screen reader integration

In addition to the package, we also provide scripts that complement package functionalities.

6.1 Preprocessing scripts

While we warmly suggest to follow the indications provided in the usage guide (suggested commands and environments), it is also possible to apply our package to an already existing LaTeX document. In this case, it is necessary to preprocess

the document in order to replace some of the unsupported commands and environments with the suggested ones. We provide a preprocessing script to handle some of these cases at our Github repository[1].

6.2 Expansion of user macros

Note that custom macros used by the author within the formulae are copied as-is into the ActualText in the hidden comment. This macros may bear no meaning for other readers, so it may be more meaningful to expand those macros into the original LaTeX commands. We provide a script that can parse LaTeX document and replace all the user macros within the formulae with their expanded definitions. You can download this script at our Github repository[1].

6.3 Screen reader dictionaries

LaTeX commands that are included as ActualText in the hidden comments corresponding to formulae may appear awkward when read by the screen reader. We provide dictionaries for JAWS and NVDA screen readers that convert LaTeX commands into natural language. Please note that the braille refreshable display will still show the formulae in their original LaTeX representations. The dictionaries can be downloaded at our Github repository[1].

7 Implementation

Standard file identification.

```
%
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{axessibility}
%[2019/01/08 v2.0: Accessibility support by marked content for inline,
%                  displayed, and various multiline formulae]
\RequirePackage{accsupp}
\RequirePackage{amsmath}
\RequirePackage{amssymb}
\RequirePackage{xstring}
%
```

PDF compression/unicode settings.

```
%
\pdfcompresslevel=0
\pdfoptionpdfminorversion=6
\input{glyphtounicode}
\pdfgentounicode=1
%
```

[1] www.integr-abile.unito.it/axessibility/?repository

Tokens used for the treatment of multiline formula environments.

```
%
\newtoks\@mltext
\newtoks\@mltexttmp
%
```

Renewed command `\BeginAccSupp`, originally defined in the package **accsupp**, to add the string `\S` before `\span`. This makes the formula readable by voiceover technologies.

```
%
\makeatletter
\renewcommand*{\BeginAccSupp}[1]{%
  \begingroup
    \setkeys{ACCSUPP}{#1}%
    \edef\ACCSUPP@span{%
      /S/Span<<%
        \ifx\ACCSUPP@Lang\relax
        \else
          /Lang\ACCSUPP@Lang
        \fi
        \ifx\ACCSUPP@Alt\relax
        \else
          /Alt\ACCSUPP@Alt
        \fi
        \ifx\ACCSUPP@ActualText\relax
        \else
          /ActualText\ACCSUPP@ActualText
        \fi
        \ifx\ACCSUPP@E\relax
        \else
          /E\ACCSUPP@E
        \fi
      >>%
    }%
    \ACCSUPP@bdc
    \ACCSUPP@space
  \endgroup
}
\makeatother
%
```

The next command creates a blank space to avoid clash with references (it appears to be a `\protect`...). Refer to https://tex.stackexchange.com/questions/57151/how-do-i-prevent-conflicts-between-accsupp-and-hyperref for possible handling of such issues.)

```
%
\newcommand{\auxiliaryspace}{ }
%
```

The next one is the actual wrapper. Takes the body of a formula environment and wraps it in AccSupp commands, to make the math-text available in comments. `\detokenize` allows the formula to be parsed and read as a string. `\expandafter` there applies to the token "{" and allow `\detokenize` to be applied after argument #1 is passed to `\BeginAccSupp`.

```
%
\makeatletter
\long\def\wrap#1{
\BeginAccSupp{method=escape,ActualText=\detokenize\expandafter{#1}}
 #1
\EndAccSupp{}%
}
\makeatother
%
```

The next wrapper, similar to the previous one, is used to handle multiline formula environments. Here some additional step is needed to obtain the desired content, to be stored via `\BeginAccSupp`.

```
%
\makeatletter
\long\def\wrapml#1{
\def\@mltext{\detokenize\expandafter{#1}}
\def\@mltexttmp{}
\StrBehind[5]{\@mltext}{ }[\@mltexttmp]
\StrGobbleRight{\@mltexttmp}{1}[\@mltext]
\BeginAccSupp{method=escape,ActualText=\auxiliaryspace\@mltext}
 #1
\EndAccSupp{}%
}
\makeatother
%
```

The next function redefines `\equation` by calling the above wrapper to its argument. This makes `\equation` accessible.

```
%
\makeatletter
\renewenvironment{equation}{%
 \incr@eqnum
  \mathdisplay@push
  \st@rredfalse \global\@eqnswtrue
 \mathdisplay{equation}%
  \collect@body\wrap\auxiliaryspace}{%
  \endmathdisplay{equation}%
  \mathdisplay@pop
  \ignorespacesafterend
}
\makeatother
%
```

The next function redefines `\equation*` by calling the above wrapper to its argument. This makes `\equation*` accessible.

```
%
\makeatletter
\renewenvironment{equation*}{%
  \mathdisplay@push
  \st@rredtrue \global\@eqnswfalse
  \mathdisplay{equation*}%
  \collect@body\wrap\auxiliaryspace}{%
  \endmathdisplay{equation*}%
  \mathdisplay@pop
  \ignorespacesafterend
}
\makeatother
%
```

The next function redefines `\[ \]`, using the above redefinition of `\equation*`

```
%
\makeatletter
\protected\def\[#1\]{\begin{equation*}#1\end{equation*}}
\makeatother
%
```

The next function redefines `\( \)` by means of a (temporary) math environment that calls the wrapper defined above.

```
%
\makeatletter
\newenvironment{tempenv}{%
  \relax\ifmmode\@badmath\else$\fi%
 \collect@body\wrap}{%
  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}
\protected\def\(#1\){\begin{tempenv}#1\end{tempenv}}
\makeatother
%
```

The next functions redefine the environments align, align*, alignat, alignat*, flalign, flalign*, gather, gather*, xalignat, xalignat*, xxalignat, originally defined in the package **amsmath**, by calling the above multiline wrapper to their argument. The structure, as for the original macros, is essentially the same for all of them.

```
%
\makeatletter

\renewenvironment{alignat}{%
  \collect@body\wrapml\auxiliaryspace
  \start@align\z@\st@rredfalse
}{%
  \endalign
}
```

```
\renewenvironment{alignat*}{%
  \collect@body\wrapml\auxiliaryspace
  \start@align\z@\st@rredtrue
}{%
  \endalign
}
\renewenvironment{xalignat}{%
  \collect@body\wrapml\auxiliaryspace
  \start@align\@ne\st@rredfalse
}{%
  \endalign
}
\renewenvironment{xalignat*}{%
  \collect@body\wrapml\auxiliaryspace
  \start@align\@ne\st@rredtrue
}{%
  \endalign
}
\renewenvironment{xxalignat}{%
  \collect@body\wrapml\auxiliaryspace
  \start@align\tw@\st@rredtrue
}{%
  \endalign
}

\renewenvironment{align}{%
  \collect@body\wrapml\auxiliaryspace
 \start@align\@ne\st@rredfalse\m@ne
}{%
  \math@cr \black@\totwidth@
  \egroup
  \ifingather@
    \restorealignstate@
    \egroup
    \nonumber
    \ifnum0=`{\fi\iffalse}\fi
  \else
    $$%
  \fi
  \ignorespacesafterend
}

\renewenvironment{align*}{%
  \collect@body\wrapml\auxiliaryspace
  \start@align\@ne\st@rredtrue\m@ne
}{%
  \endalign
}

\renewenvironment{flalign}{%
```

```
  \collect@body\wrapml\auxiliaryspace
  \start@align\tw@\st@rredfalse\m@ne
}{%
  \endalign
}

\renewenvironment{flalign*}{%
  \collect@body\wrapml\auxiliaryspace
  \start@align\tw@\st@rredtrue\m@ne
}{%
  \endalign
}

\renewenvironment{gather}{%
  \collect@body\wrapml\auxiliaryspace\auxiliaryspace
  \start@gather\st@rredfalse
}{%
  \math@cr \black@\totwidth@ \egroup
  $$\ignorespacesafterend
}

\renewenvironment{gather*}{%
  \collect@body\wrapml\auxiliaryspace\auxiliaryspace
  \start@gather\st@rredtrue
}{%
  \endgather
}

\makeatother
%
```

8 History

[2018/07/09: v1.0]

- First version

[2019/01/08: v2.0]

- Added support for environments align, align*, alignat, alignat*, flalign, flalign*, gather, gather*, xalignat, xalignat*, and xxalignat, from the package **amsmath**