The axessibility package

Dragan Ahmetovic[*], Tiziana Armano[*], Cristian Bernareggi[*], Anna Capietto[*], Sandro Coriasco[*], Boris Doubrov[†], Alexander Kozlovskiy[†], Nadir Murru[*]

<anna.capietto at unito.it>,<sandro.coriasco at unito.it>,
<boris.doubrov at duallab.com>

January 8, 2020

Abstract

PDF documents containing formulae generated by LaTeX are usually not accessible by assistive technologies for people with special educational needs and 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 a PDF/UA document.

Contents

[*]Dipartimento di Matematica "G. Peano", Università degli Studi di Torino, 10123, Italy
[†]Dual Lab, Ottignies-Louvain-la-Neuve 1340, Belgium

1 Introduction

This package focuses on the specific problem of the accessibility of PDF documents generated by LaTeX for visually impaired people and people with special educational needs. 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 first reads incorrectly the formula and then, only as a second step, provides 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, accessibility_meta.sty and tagpdf have been developed in order to obtain tagged PDF documents. The package accsupp.sty develops some interesting tools for commenting formulae using also special characters (possibility that is not available, e.g., in the pdfcomment.sty package). The package tagpdf widely further developed tagging functionalities, along the most recent specifications for PDF documents accessibility. However, all of the above are not automatized methods, since the comment and tags 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. Additional functionalities, implemented in this version, are available when the typeset is done by means of lualaTeX(see below).

2 License

project.org/lppl.txt and version 1.3 or later is part of all distributions of LaTeX version 2005/12/01 or later.

3 Prerequisites

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

4 Package specification

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

- for the usage based on the **tagpdf** package,

  usepackage{**axessibility**}

  or, equivalently,

  usepackage[**tagpdf**]{**axessibility**}

- for the usage based on the **accsupp** package,

  usepackage[**accsupp**]{**axessibility**}

The package includes the following features:

- In the **accsupp** version, the commands

  \pdfcompresslevel=0
  \pdfoptionpdfminorversion=6

  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.

- In the **tagpdf** version, the commands

  \tagpdfsetup{tabsorder=structure,uncompress,activate-all,interwordspace=true}
  \tagpdfifpdftexT
  {
  \pdfcatalog{/Lang (en-US)}
  \usepackage[T1]{fontenc}

\input glyphtounicode
\pdfgentounicode=1
}
\tagpdfifluatexT
{
\pdfextension catalog{/Lang (en-US)}
\RequirePackage{fontspec}
\RequirePackage{luacode}
\newfontface\zerowidthfont{freeserif}
\directlua{
require("axessibility.lua")
}
}

produce an uncompressed PDF document, directing appropriately the typesetting, either via pdfLaTeX or luaLaTeX. Equations (and other structures) are tagged by means of the commands defined in the **tagpdf** package, so that screen readers access the ActualText created by them. When typeset via luaLaTeX, additional functionalities, implemented in the file axessibility.lua, can be activated (see Section 6 below).

- The new commands

  \wrap#1
  \wrapml#1
  \wrapmlstar#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}**
\begin{multline} ... **\end{multline}**
\begin{multline*} ... **\end{multline*}**

have been similarly redefined, using the commands `\wrapml` and `\wrapmlstar`. 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, or people with special educational needs, 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 (concerning this feature, please see also Section 7 below).

Inline and displayed mathematical modes encoded by means of $ and $$ are not supported by the package. However, external scripts, or a command **\doreplacement{true}** when typesetting with luaLaTeX, implement the automatic replacement of these TeX markers by their LaTeX equivalents \(\) and \[\]. The external scripts are provided as companion software and described in the following section.

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 and other resources that complement its 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, if pdfLaTeX is employed, 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]. Namely, the underscore characters have to be substituted as indicated above when employing the **accsupp** mode, while this is not necessary when the **tagpdf** mode is selected. $ and $$ markers must be replaced when typesetting with pdfLaTeX both in the **accsupp** and **tagpdf** mode.

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 the 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].

6.4 Automatic replacement of $ and $$ markers in Lua mode

Lua mode implements the LaTeX command **doreplacement** to switch on/off the automatic replacement of $ and $$ by `\(` `\)` and `\[` `\]` environments, so that external scripts are no longer required. This option is disabled by default and can be switched on or off by the call `\doreplacement{true}` or `\doreplacement{false}` respectively. When enabled, the replacement is applied to every input line, which might have undesired effects in verbatim mode or other environments, where $

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

and $$ are used as regular symbols. So, it is recommended to switch off the automatic replacement functionality in such cases. The Lua code implementing the replacement is contained in the second main source file axessibility.lua. Please see the comments within the file itself for further explanations.

7 Known issues

Note that, to preserve the compatibility with Acrobat Reader when employing the **accsupp** mode, 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`. The underscore character works correctly when the PDF file is produced in the **tagpdf** mode.

The typeset of documents with **axessibility** by means of luaLaTeX does not work corrrectly when selecting the **accsupp** mode.

The copy and paste feature described in Section 5 above does not work correctly at times. We observed that it behaves as expected when the NVDA screenreader is active, while it can produce multiple copies of the formula code when this software is not running.

The preprocessing scripts do not cover all the possible character/environment combinations, so some errors can be generated, at times, when they are employed to perform the underscore, $ and $$ substitutions.

8 Implementation

Standard file identification.

```
%
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{axessibility}
%[2019/11/01 v3.0: Accessibility support by marked content for inline,
%                  displayed single line, and various displayed multiline formulae]

%% 'tagpdf' or 'accsupp' option
\newif\iftagpdfopt

\DeclareOption{accsupp}{
  \tagpdfoptfalse
}

\DeclareOption{tagpdf}{
  \tagpdfopttrue
}

\ExecuteOptions{tagpdf}

\ProcessOptions\relax
```

```
21
22 \RequirePackage{amsmath}
23 \RequirePackage{amssymb}
24 \RequirePackage{xstring}
25
26 %%%
27 % to avoid errors in if constructs
28 %%%
29 \makeatletter
30 \long\def\@macronestedifalign{
31 \ifingather@
32     \restorealignstate@
33     \egroup
34     \nonumber
35     \ifnum0=`{\fi\iffalse}\fi
36   \else
37     $$%
38   \fi
39   }
40
41 \long\def\@macronestedifmultline{
42   \iftagsleft@ \@xp\lendmultline@ \else \@xp\rendmultline@ \fi
43 }
44 %
45 \makeatother
46 %
47 %
48 \iftagpdfopt
49  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
51  % tagpdf option code (default)   %
52  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54 %\def\messaggio{option tagpdf} %debug
55 %
```

Setup of the **tagpdf** package.

```
56 %
57 \RequirePackage{tagpdf}
58 \tagpdfsetup{tabsorder=structure,uncompress,activate-all,interwordspace=true}
59 \tagpdfifpdftexT
60  {
61   %set language / can also be done with hyperref
62   \pdfcatalog{/Lang (en-US)}
63   \usepackage[T1]{fontenc}
64   \input glyphtounicode
65   \pdfgentounicode=1
66  }
67 \tagpdfifluatexT
68  {
```

```
  %set language / can also be done with hyperref
  \pdfextension catalog{/Lang (en-US)}
  \RequirePackage{fontspec}
  \RequirePackage{luacode}
  \newfontface\zerowidthfont{freeserif}
\directlua{
require("axessibility.lua")
}
}
%
```

Tokens used for the treatment of multiline formula environments.

```
%
\makeatletter

\newtoks\@mltext
\newtoks\@mltexttmp

%
```

The command **\doreplacmeent** with boolean argument switches on or off $ and $$ replacement by LaTeX environments `\( \)` and `\[ \]`. This command works only in Lua mode and allows to avoid the use of external substitution script. It is switched off by default.

```
%
\newcommand{\doreplacement}[1]{
\tagpdfifluatexT
\directlua { replace_dls_and_double_dls(#1) }
}
%
```

Automatic tagging at the document level.

```
%
\let\begin@document=\document
\let\end@document=\enddocument
\renewcommand{\document}{\begin@document\tagstructbegin{tag=Document}}
\renewcommand{\enddocument}{\tagstructend\end@document}
\makeatother
%
```

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

```
%
\makeatletter
\newenvironment{temp@env}{%
  \relax\ifmmode\@badmath\else$\fi%
 \collect@body\wrap}{%
  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}
\protected\def\(#1\){\begin{temp@env}#1\end{temp@env}}
\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. It takes the body of a formula environment and wraps it in the **tagpdf** package tagging 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 allows `\detokenize` to be applied after argument #1 is passed to the tagging commands.

```
%
\makeatletter
\long\def\wrap#1{
\tagstructbegin{tag=P,alttext-o=\detokenize\expandafter{#1},
                  actualtext-o=\detokenize\expandafter{#1}}
 \tagmcbegin{tag=P,alttext-o=\detokenize\expandafter{#1},
                  actualtext-o=\detokenize\expandafter{#1}}
 #1
 \tagmcend
\tagstructend
}
\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 wrappers, similar to the previous one, are used to handle multiline formula environments. Here some additional step is needed to obtain the desired content, to be stored via the tagging commands.

```
%
\makeatletter

\long\def\wrapml#1{
\def\@mltext{\detokenize\expandafter{#1}}
\def\@mltexttmp{}
\StrBehind[6]{\@mltext}{ }[\@mltexttmp]
\StrGobbleRight{\@mltexttmp}{1}[\@mltext]
\tagstructbegin{tag=P,alttext-o=\detokenize\expandafter{\@mltext},
                  actualtext-o=\detokenize\expandafter{\@mltext}}
 \tagmcbegin{tag=P,alttext-o=\detokenize\expandafter{\@mltext},
                  actualtext-o=\detokenize\expandafter{\@mltext}}
 #1
}

%
% This one should be \wrapml parametrized \StrBehind[5]
%
\long\def\wrapmlstar#1{
\def\@mltext{\detokenize\expandafter{#1}}
\def\@mltexttmp{}
\StrBehind[5]{\@mltext}{ }[\@mltexttmp]
\StrGobbleRight{\@mltexttmp}{1}[\@mltext]
\tagstructbegin{tag=P,alttext-o=\detokenize\expandafter{\@mltext},
                 actualtext-o=\detokenize\expandafter{\@mltext}}
 \tagmcbegin{tag=P,alttext-o=\detokenize\expandafter{\@mltext},
                 actualtext-o=\detokenize\expandafter{\@mltext}}
```

```
 #1
}

%
% This one should be \wrapml parametrized = \wrapmlstar
%
\long\def\wrapmlalt#1{
\def\@mltext{\detokenize\expandafter{#1}}
\def\@mltexttmp{}
\StrBehind[5]{\@mltext}{ }[\@mltexttmp]
\StrGobbleRight{\@mltexttmp}{1}[\@mltext]
\tagstructbegin{tag=P,alttext-o=\detokenize\expandafter{\@mltext},
                actualtext-o=\detokenize\expandafter{\@mltext}}
 \tagmcbegin{tag=P,alttext-o=\detokenize\expandafter{\@mltext},
                actualtext-o=\detokenize\expandafter{\@mltext}}
 #1
}

\makeatother
%
```

The next functions redefine the environments align, align*, alignat, alignat*, flalign, flalign*, gather, gather*, xalignat, xalignat*, xxalignat, multline, multline*, 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{align}{%
  \collect@body\wrapml\auxiliaryspace
 \start@align\@ne\st@rredfalse\m@ne
}{%
  \math@cr \black@\totwidth@
  \egroup
  \@macronestedifalign
  \ignorespacesafterend
  \tagmcend
  \tagstructend
}

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

\renewenvironment{alignat}{%
  \collect@body\wrapml\auxiliaryspace\auxiliaryspace
```

```
  \start@align\z@\st@rredfalse
}{%
  \endalign
}

\renewenvironment{alignat*}{%
  \collect@body\wrapmlstar\auxiliaryspace
  \start@align\z@\st@rredtrue
}{%
  \endalign
}

\renewenvironment{xalignat}{%
  \collect@body\wrapmlalt\auxiliaryspace
  \start@align\@ne\st@rredfalse
}{%
  \endalign
}

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

\renewenvironment{xxalignat}{%
  \collect@body\wrapmlalt\auxiliaryspace
  \start@align\tw@\st@rredtrue
}{%
  \endalign
}

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

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

\renewenvironment{gather}{%
  \collect@body\wrapmlalt\auxiliaryspace\auxiliaryspace
  \start@gather\st@rredfalse
```

```
}{%
  \math@cr \black@\totwidth@ \egroup
  $$\ignorespacesafterend
  \tagmcend
  \tagstructend
}

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

\renewenvironment{multline}{%
  \collect@body\wrapmlalt\auxiliaryspace\auxiliaryspace
  \start@multline\st@rredfalse
}{%
%  \iftagsleft@ \@xp\lendmultline@ \else \@xp\rendmultline@ \fi
  \@macronestedifmultline
  \ignorespacesafterend
  \tagmcend
  \tagstructend
}

\renewenvironment{multline*}{
    \collect@body\wrapmlstar\auxiliaryspace\auxiliaryspace
    \start@multline\st@rredtrue
}{
  \endmultline
}

\makeatother
%
```

End of tagpdf option code

```
%
\else
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %           accsupp option code          %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%\def\messaggio{option accsupp} %debug

\RequirePackage{accsupp}

%\RequirePackage{amsmath}
%\RequirePackage{amssymb}
%\RequirePackage{xstring}
```

```
% \noindent PDF compression/unicode settings.


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

Tokens used for the treatment of multiline formula environments.

```
%

\makeatletter

\newtoks\@mltext
\newtoks\@mltexttmp

\makeatother
%
```

Renewed command `\BeginAccSupp`, originally defined in the package **accsupp**, to add the string `\S` before `\span`. This makes the formula readable by screenreading 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. It 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 allows `\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, multline, multline*, 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{align}{%
  \collect@body\wrapml\auxiliaryspace
 \start@align\@ne\st@rredfalse\m@ne
}{%
  \math@cr \black@\totwidth@
  \egroup
  \@macronestedifalign
  \ignorespacesafterend
}

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

\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{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
}

\renewenvironment{multline}{%
  \collect@body\wrapml\auxiliaryspace\auxiliaryspace
  \start@multline\st@rredfalse
}{%
  %\iftagsleft@ \@xp\lendmultline@ \else \@xp\rendmultline@ \fi
  \@macronestedifmultline
  \ignorespacesafterend
}

\renewenvironment{multline*}{
  \collect@body\wrapml\auxiliaryspace\auxiliaryspace
  \start@multline\st@rredtrue
}{
  \endmultline
}
```

```
%%}
%%
\makeatother
%%%%
% End of accsupp option code
%%%%
\fi
%
```

For the automatic $ and $$ replacement code, please see the second main source file axessibility.lua.

9 History

[2018/07/09: v1.0]

- First version (with Michele Berra, Alice Ruighi, and Eugenia Taranto).

[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** (with Michele Berra, Alice Ruighi, and Eugenia Taranto).

[2020/01/08: v3.0]

- Added selection option, to choose between tagging via accsupp or tagpdf packages. Added support for environment multline and multline* from the package **amsmath**. Added the second main source file axessibility.lua, containing code that can be activated when typesetting with luaLaTeX.