Axessibility: Creating PDF documents with accessible formulae

D. Ahmetovic, T. Armano, C. Bernareggi, M. Berra, A. Capietto, S. Coriasco, N. Murru, A. Ruighi

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 LaTeX package `axessibility.sty` that we have developed alleviates this issue, allowing one to create PDF documents where the formulae are read by these assistive technologies, since it automatically generates hidden comments in the PDF document (using the `/ActualText` attribute) in correspondence to each formula. This actual text is hidden in the PDF document, but the screen readers JAWS, NVDA and VoiceOver read it correctly. Moreover, we have created NVDA and Jaws dictionaries (in English and in Italian) that provide reading in the natural language in case the user does not know LaTeX commands. The package does not generate PDF/UA.

1 Introduction

In this paper, we describe `axessibility.sty`, a LaTeX package which allows automatic generation of a PDF document with formulae accessible by assistive technologies for visually impaired people. The package was first introduced in [3] and we took the material presented there as the starting point for the results presented here. Assistive technologies (screen readers and braille displays) perform satisfactorily with regard to digital documents containing text, but they still have a long way to go as far as formulae and graphs are concerned. A comprehensive overview of this problem can be found in [1, 2, 4].

Many studies have been conducted in order to improve the accessibility of digital documents with mathematical content. For instance, MathPlayer ensures accessibility of formulae inserted by using MathType in Word documents [13]. Another way to create accessible mathematical documents is given by the MathML language (see [7] for further information). However, accessibility of such documents is heavily affected by the versions of browsers, operating systems and screen readers, making this solution very unstable.

A system used by blind people for reading and writing mathematics is the LAMBDA system (Linear Access to Mathematics for Braille Device and Audiosynthesis). Mathematical language in LAMBDA is designed so that every symbol can be directly translated into words. For further details on LAMBDA we refer to [6]. Unfortunately, this system does not help to spread accessible digital documents, since it is used only by visually impaired people and is not a standard for the realization of documents by sighted people. Regarding LaTeX, assistive technologies can directly manage LaTeX documents. In this case, visually impaired people need to learn LaTeX in order to understand the commands. However, there is software which facilitates LaTeX comprehension and usability; one such is BlindMath [12]. Moreover, some converters from LaTeX to braille exist, see, e.g., [5] and [11].

In general, the most widespread digital documents are in PDF format. However, in the case of mathematical contents, they are not accessible at all, since formulae are usually unreadable by screen readers because they are bidimensional as images. None of the above systems directly support production of accessible formulae in PDF documents. This could be possible only performing specific tasks. For instance, using the Word editor, if each formula is manually tagged by the author (by using the alternative text), such a comment will be kept when the corresponding PDF file is generated and it will be read by the screen reader. However, this procedure does not help to improve the presence of accessible PDF documents, since it is a tedious and time-consuming method. It is difficult to imagine an author or editor performing these actions for the realization, e.g., of a book.

Currently, a standard and fast method for inserting accessible formulae into PDF documents is still lacking, despite it being a crucial issue for spreading accessible digital scientific documents. In [14], standard guidelines for accessibility of PDF documents are presented. Moreover, in [8], [9] and [10], an overview about accessibility of PDF documents is provided with a focus on mathematical contents.

In this paper, we describe the features of our package `axessibility.sty`, which provides the first method for an automated production of accessible PDF documents with mathematical contents. We would like to highlight that this package does not produce fully tagged PDF, such as the standard PDF/UA, but it does allow obtaining a PDF where formulae are described using the `/ActualText` attribute.

2 Problem statement

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., actual text, similar to the case of web pages or Word documents.

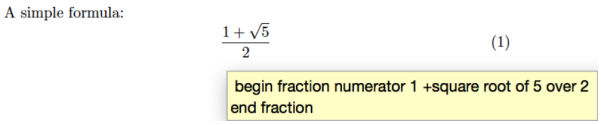

Figure 1: PDF document generated using the package `pdfcomment.sty`

This can be made, e.g., by using the LaTeX package `pdfcomment.sty` or using an editor for PDF files like Adobe Acrobat Pro. In any case, this task must be manually performed by the author and thus is surely inefficient, since the author must write the formulae and, in addition, insert a description for each formula. Note also that the package `pdfcomment.sty` does not allow insertion of special characters like *backslash*, *brace*, etc., in the comment. Moreover, with these solutions, the reading is bothersome, since the screen reader first incorrectly reads the formula and then the comment provided for the formula. In Figure 1, we show the PDF document generated from the following LaTeX code containing a simple formula with a manually inserted comment:

```
\documentclass{article}
\usepackage{pdfcomment}
\begin{document}
A simple formula:
\begin{equation}
\pdftooltip{
        \frac{1 + \sqrt{5}}{2}
        }{
        begin fraction numerator 1 +
    square root of 5 over 2 end fraction
        }
\end{equation}
\end{document}
```

When the screen reader accesses the PDF document, the formula will be read

> square root 1 plus 5 2 begin fraction numerator 1 plus square root of 5 over 2 end fraction

i.e., before reading the correct comment

> begin fraction numerator 1 plus square root of 5 over 2 end fraction

the screen reader reads incorrectly the formula

> square root 1 plus 5 2.

There are also some LaTeX packages that try to improve the accessibility of PDF documents produced by LaTeX. Specifically, the packages `accsupp.sty` (`ctan.org/pkg/accsupp`) and `accessibility.sty` (`github.com/AndyClifton/AccessibleMetaClass`) have been developed in order to obtain tagged PDF documents. However, neither package solves the problem of accessibility of formulae.

3 `axessibility.sty` LaTeX package

Our package, named `axessibility.sty`, solves the problem described in the previous section. It achieves this by inserting a hidden comment in the PDF file corresponding to any given formula. This comment, named `/ActualText`, contains the original LaTeX commands used to generate the formula. The hidden comment is read by screen readers and braille displays instead of the ASCII representation of the formula, which is often incorrect. We tested our package using Acrobat Reader together with the screen readers JAWS and NVDA for Windows and the native VoiceOver on macOS and iOS.

3.1 Usage

To create an accessible PDF document for visually impaired people, authors need only include the package `axessibility.sty` in the preamble of their LaTeX project. Mathematical environments automatically produce the `/ActualText` content and include it in the produced PDF file.

We handle the most common environments for inserting formulae, i.e., `equation`, `equation*`, `\[`, `\(`. Hence, any formula inserted using one of these environments is accessible in the corresponding PDF document. Additionally, the package enables copying the formula's LaTeX code from the PDF reader and pasting it elsewhere.

To preserve 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. We suggest using the equivalent command `\sb`.

In-lined and display mathematical modes (`$`, `$$`) are not supported in this version of the package. However external scripts provided as companion software can also address these use cases.

If we use the package `axessibility.sty` applied to the previous example, we obtain the following LaTeX code:

```
\documentclass{article}
\usepackage{axessibility}
\begin{document}
    A simple formula:
    \begin{equation}
        \frac{1 + \sqrt{5}}{2}
    \end{equation}
\end{document}
```

We observe that, in this case, the author has to write the formula without adding anything else. Moreover,

inside the source code of the PDF file, we find an `/ActualText` tag with the LaTeX code, automatically generated by the `axessibility.sty` package.

```
/S/Span<</ActualText(\040\040\\frac
\040{1\040+\040\\sqrt
\040{5}}{2}\040)
>>
BDC
```

The screen reader reads correctly the LaTeX command `\frac{1+\sqrt{5}}{2}`. Moreover, we have created JAWS and NVDA dictionaries that provide the reading in the natural language in the case that the user does not know the LaTeX commands.

3.2 Technical overview

`axessibility.sty` first defines a pair of internal commands (`\BeginAxessible` and `\EndAxessible`) modelled on `\BeginAccSupp` and `\EndAccSupp` from the `accsup` package as follows:

```
\newcommand*{\BeginAxessible}[1]{%
  \begingroup
    \setkeys{ACCSUPP}{#1}%
    \edef\ACCSUPP@span{%
     /S/Formula<<%
        \ifx\ACCSUPP@Alt\relax
        \else
          /Alt\ACCSUPP@Alt
        \fi
        \ifx\ACCSUPP@ActualText\relax
        \else
          /ActualText\ACCSUPP@ActualText
        \fi
      >>%
    }%
    \ACCSUPP@bdc
    \ACCSUPP@space
  \endgroup
}
```

Specifically, as seen, `\BeginAxessible` adds a hidden comment that starts with `/S/Formula` instead of `/Span`. Then, to close:

```
\newcommand*{\EndAxessible}{%
  \begingroup
    \ACCSUPP@emc
  \endgroup
}
```

The second building block of this package is the wrapper. This routine takes the LaTeX code inside the formula, removes the tokens and passes it to `\BeginAxessible`:

```
\long\def\wrap#1{%
\BeginAxessible{method=escape,
ActualText=\detokenize\expandafter{#1},
Alt=\detokenize\expandafter{#1}}%
#1%
\EndAxessible
}
```

Finally, using the wrapper, we can redefine the mathematical environments using the command above. Here is an example using `equation`:

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

4 Conclusions and future work

We have developed a LaTeX package that automatically generates comments for formulae when the PDF document is produced by LaTeX. The comments are hidden in the PDF document and they contain the LaTeX commands that generate the formulae. In this way, an accessible PDF document containing formulae is generated. Indeed, screen readers are able to access the comment when processing a formula and reading it. Moreover, we have created JAWS and NVDA dictionaries that provide for reading in natural languages in case the user does not know LaTeX commands.

There are a few issues that are yet to be solved with a pure LaTeX solution. Namely,

- Math environments delimited with `$`, `$$`.
- User-defined macros.
- Multi-line environments such as `\align` and `\eqnarray`.
- Semantic description of formulae.
- PDF/UA.

We address the first two problems using an external script — `axesscleaner.py`, from `github.com/integr-abile/axesscleaner` — coded in Perl and Python. We successfully "cleaned" two entire books using it. The script also replaces all underscore characters `_` with `\sb`. Using this solution we are now able to apply `axessibility.sty` to entire textbooks that were written without using the package in the first place. Multi-line environments are going to be

treated using a LaTeX solution that is currently in the test phase.

Concerning the last two problems, more in-depth research is in order. The authors are currently initiating the investigation to address these issues in future work.

The authors are also aware that the use of `/S/Formula` is in conflict with the internal structure of the PDF — the document is fully readable but it does not pass the so-called pre-flight test. This is not intended to be a final solution as the authors' goal is to create a PDF/UA with accessible formulae. However, we point out that this is an operative and reproducible solution, which automatically creates scientific material that is successfully used by people with visual impairment.

5 Acknowledgements

The authors wish to thank 'Fondazione Cassa di Risparmio di Torino', LeoClub (Biella, Italy) and the several volunteers with visual impairment who provided their fundamental contribution. We are grateful to U. Fisher for pointing us in the right direction concerning the use of the package `accsupp`. Specifically, she suggested not redefining `BeginAccSup`, but rather using a new environment. We also thank R. Moore for the fruitful discussion on the PDF structure and the suggestions to improve the manuscript.

References

[1] D. Ahmetovic, T. Armano, et al. Axessibility: A LaTeX package for mathematical formulae accessibility in PDF documents. In *Conference on Computers and Accessibility*. ACM, 2018.

[2] D. Archambault, B. Stöger, et al. Access to scientific content by visually impaired people. *Upgrade*, 2007.

[3] T. Armano, A. Capietto, et al. An automatized method based on LaTeX for the realization of accessible PDF documents containing formulae. In *Computers Helping People with Special Needs*, vol. 1089 of *Lecture Notes in Computer Science*. Springer, 2018.

[4] T. Armano, A. Capietto, et al. An overview on ICT for the accessibility of scientific texts by visually impaired students. In *SIREM-SIE-L Conference*, 2014.

[5] M. Batusic, K. Miesenberger, and B. Stöger. Labradoor, a contribution to making mathematics accessible for the blind. In *International Conference on Computers Helping People with Special Needs*. Springer, 1998.

[6] C. Bernareggi. Non-sequential mathematical notations in the LAMBDA system. In *Computers Helping People with Special Needs*. Springer, 2010.

[7] C. Bernareggi and D. Archambault. Mathematics on the Web: Emerging opportunities for visually impaired people. In *Conference on Web accessibility*. ACM, 2007.

[8] M. Borsero, N. Murru, and A. Ruighi. Il LaTeX come soluzione al problema dell'accesso a testi con formule da parte di disabili visivi. *ArsTEXnica* 22:12–18, Oct. 2016. `guitex.org/home/images/ArsTeXnica/AT022/murru-2016.pdf`

[9] R. Moore. Ongoing efforts to generate tagged PDF using pdfTeX. *TUGboat* 30(2):170–175, 2009. `tug.org/TUGboat/tb30-2/tb95moore.pdf`

[10] R. Moore. PDF/A-3u as an archival format for accessible mathematics. In S. Watt et al., eds., *CICM*. Springer, 2014.

[11] A. Papasalouros and A. Tsolomitis. Direct TeX-to-braille transcribing method. *Science Education for Students with Disabilities*, 2017.

[12] A. Pepino, C. Freda, et al. "BlindMath", a new scientific editor for blind students. In *Computers Helping People with Special Needs*. Springer, 2006.

[13] N. Soiffer. MathPlayer: Web-based math accessibility. In *Conference on Computers and Accessibility*. ACM, 2018.

[14] A. Uebelbacher, R. Bianchetti, and M. Riesch. PDF accessibility checker (PAC 2): The first tool to test PDF documents for PDF/UA compliance. In *Computers Helping People with Special Needs*. Springer, 2014.

◇ D. Ahmetovic
T. Armano
Dipartimento di Matematica
"G. Peano", Università di Torino
`dragan.ahmetovic` ,
`tiziana.armano`
`(at) unito.it`

◇ C. Bernareggi
Dipartimento di Informatica,
Università di Milano
`cristian.bernareggi (at)`
`unimi.it`

◇ M. Berra
A. Capietto
S. Coriasco
N. Murru
A. Ruighi
Dipartimento di Matematica
"G. Peano", Università di Torino
`michele.berra` ,
`anna.capietto` ,
`sandro.coriasco` ,
`nadir.murru` ,
`alice.ruighi`
`(at) unito.it`