Creating accessible pdfs with LaTeX

Ulrike Fischer

Abstract

This article describes the current state and planned actions to improve the accessibility of pdfs created with LaTeX as it is currently undertaken by the LaTeX Team.

Sommario

Questo articolo descrive lo stato attuale e le azioni pianificate per migliorare l'accessibilità dei pdf creati con LaTeX così come garantito dal LaTeXTeam.

1 Accessibility of pdf

The pdf language is at its core a *page description* programming language. It describes very accurately how text and graphical elements look and where they are placed on a page. But it doesn't describe the semantical meaning of the elements and the reading order: by looking at the code there is no way to know if a text is a section heading or some watermark or a footnote or if it belongs to a tabular. You can't know where a sentence has its continuation on another page or how many words a text contains, and sometimes it is even impossible to identify the characters: you only see some glyph index number and copy & paste can give gibberish.

This all isn't a problem as long as the pdf is merely meant for printing or viewing but it restricts its use for digital processing like copy & paste, automatic extraction of billing data, reflowing or using the pdf with a screen reader. For such uses you need accessible, structured, extractable content.

«Accessibility» as a standard is described in PDF/UA. It is also included in other standards like PDF/A-1a and PDF/A-2a (the «a» stands for accessible). The standards contain a number of requirements that should help retrieving the content for further processing: for example, that every character has a unicode representation (no gibberish when copy & pasting), that word spaces are correctly marked up (so that reflowing works), that the language of the document and the text is declared (so that a screen reader can guess the pronunciation), that pictures have sensible alternative descriptions. And most importantly: that the document is *tagged*. This last requirement is responsible for adding structure information to the content. It marks up content as section or tabular cell or list item. This improves navigation in the document with, for example, a screen reader, but also exporting to other formats like xml.

TUG maintains a webpage with various links to relevant standards, articles and packages, (see TUG, 2019).

2 Creating a tagged pdf

Tagging consists of two main tasks: at first in the *stream object* of a page every bit of content must be marked and labelled with a number `MCID` n.

The following listing shows a small example. The `BDC` and the `EMC` lines are the start and end markers needed for tagging. The `/H1` indicates that the content is part of a sectioning element.

```
stream
/H1 <<MCID 0>> BDC
BT
/F17 14.3462 Tf 124.802 706.129
 Td [(0.1)-1100(Section)]TJ
ET
EMC
```

In the next step a number of pdf objects must be created to describe the *structure tree*. Every object contains references to the parent `/P` and to one or more kid elements `/K`. The leaf nodes are the `MCID` n created in the first step. A typical object looks roughly like this:

```
5 0 obj
<<
   /Type /StructElem
   /S /H1
   /P 4 0 R
   /K <</Type /MCR /MCID 0>>
>>
endobj
```

This is a structure element of the type `/H1` (and so a sectioning element) and it has one kid element, the text of the section marked above.

Beside this a number of additional settings and objects must be added to the pdf for crossreferencing and «administration».

3 Changing LaTeX

Measured in computer time LaTeX is quite old. LaTeX is not only a format: it was always meant to be extended by packages and classes and over the time many people contributed to LaTeX. It has a quite large user base with *very* varied demands regarding stability, features and development. LaTeX is still used with a variety of engines: pdfTeX, XeTeX, LuaTeX, (u)pTeX and backends (dvips,

dvipdfmx). One could compare LaTeX to an old city: lots of houses built at different times in different styles by various people, some modern, some older, some are in a good state, other are falling apart but nevertheless home to someone.

This means that changing LaTeX is not easy: We can't break lots of packages and old documents even if the reward is accessible pdfs. And we have to consider that documents must be compilable in TeX systems of varying age for example when uploading them to some journal.

A very important aspect of the project long term is to develop a change strategy and manage the integration of core support across the LaTeX universe.

4 First Steps towards tagging

Tagging pdf with LaTeX has been on the agenda for quite some time. Babett Schalitz wrote a thesis about it in 2007, Ross Moore had a number of talks and articles at TUG since then too. When I considered to work on the topic some time ago I got code from both and decided rather quickly that at first some work on the basics was needed. Tagging should in my opinion not be done by creating a package that patches all sorts of commands in other packages: this is much too fragile. It needs proper support in the LaTeX kernel and proper support in the main classes and packages. I also thought that to identify the needed support and to test implementations and interfaces concrete code was needed. So I wrote the package tagpdf. The package offers core commands to tag a document and to activate some of the other requirements needed to make a pdf accessible. The low-level code to mark up a text as a section looks roughly like this:

```
\tagstructbegin{tag=H1}
  \tagmcbegin{tag=H1}
    Section
  \tagmcend
\tagstructend
```

The `\tagstructXX` commands create the structure, while the `\tagmcXX` commands add the `MCID` marks to the page stream.

The tagpdf package currently works with pdfLaTeX and LuaLaTeX – with lualatex the results are the best as one doesn't have to worry about the behaviour at page breaks – but with the help of the work on the pdfresources project described below it should be possible to extend it to other engines and backends.

5 LaTeX-dev

Another important step towards accessible pdfs was the implementation of the latex-dev format by the LaTeX team and the maintainers of TeXLive and MiKTeX: latex-dev is a pre-release of LaTeX from the development branch and made available on CTAN. It allows users of a current TeX distribution to test their documents and code against the upcoming LaTeX release by simply using a latex-binary with the addition `-dev` attached.

latex-dev has not been created solely with tagging in mind but it will help us to coordinate and test changes with package and class authors and so it is an important part of the project.

6 Pdf resource management

When tagging a pdf one has to add a number of settings to pdf dictionaries which can be described as «global resources». As already mentioned in an answer (Oberdiek, 2015), LaTeX has no interfaces for this:

> Unhappily, the LaTeX format has overslept the PDF development quite entirely. Managing global resources is the prime task for an OS, format in TeX speek. Because of the missing resource manager, both [tikz and transparent] packages do what most packages do, they think they are alone and add their stuff to the resource, ...

With tagging entering the scene it was clear that something needed to be done to remedy this problem and so the pdfresource project in the LaTeX github was created: it contains a (still quite experimental) expl3-style which offers commands to add contents to pdf resources in a controlled way. It also offers backend independent interfaces to a number of core commands needed when writing objects to a pdf. The package works with the main engines (pdfTeX, LuaTeX and XeTeX) and backends (dvipdfmx and – more or less – dvips).

The main task for the next months is to test the code, to integrate it into the kernel and to adapt existing packages to use it. The number of packages which should use the pdf resource manager is not very large but it includes important packages like hyperref, tikz, media9, pdfx.

7 Adapting the engines

Another open issue that emerged during the last year was missing functionalities in engines and backends. For example pdfTeX is not ready for pdf 2.0: it has no command to set a major pdf version. As pdf 2.0 adds important features needed for accessibility (the concept of associated files) this is clearly something that should be changed. It would be also useful if pdfTeX could execute code at shipout time as it can be done with luatex with `\latelua`. The dvipdfmx backend and dvips are missing additional color stacks.

8 Adding hooks

As already shown in section 2 and 4, tagging a pdf requires adding quite a number of commands. Obviously all the standard structures should if possible add the needed code automatically. For this hooks are needed at the right places. The «right place» has firstly a technical meaning: with the exception of LuaTEX, the tagging code inserts *whatsits*; this means it can change the output if used in the wrong place (as sometimes anchors set by hyperref do).

But more importantly the «right place» means that we need to identify the *owner* of the code which should insert the tagging code. For example sections are generally created with `\@startsection`. So this kernel command looks like a natural place to insert hooks for tagging commands. On the other hand chapters and parts have special commands created by the classes. Does it make sense if the kernel handles the one part and the classes the other? Another example are bibliographies and glossaries: packages like biblatex and glossaries look like the natural owner here – and both packages have already lots of hooks which make it easy to implement tagging – but both also use standard structures like lists or tabulars and additions to this generic environments could clash with their needs.

This means that beside a pdf resource manager we also need a hook management. And we need lots of real use cases and examples to be able to investigate the various dependencies.

9 Mathematics

How to tag maths is still an open problem. There are quite a number of possibilities to make it accessible.

- One is to attach the LATEX source code either as file or verbatim with `/Actualtext` to the math structure. For a number of environments this can be automated quite well as the axessibility package demonstrate (but it is difficult for inline math input with `$..$`). The usability with a screen reader is not bad – even if not every word was correctly read aloud in my tests – but it requires that the user understands LATEX input syntax and with large equations and complicated grouping it can be quite difficult to follow and to navigate through subequations. The usability can be improved if one invests the time to manually split the math and add explaining words.

- Another possibility is to mark all the maths bits with mathml structure names. At least with LuaTEX this can probably be done more or less automatically – proof of concepts are the ConTEXt format and TEX4ht. But it is unknown wether screen readers or other applications can actually use the information.

- A third possibility is to convert the equation to mathml, for example with mathjax, and attach it as associated file to the structure. But here too it is unclear how such a mathml can be processed by the pdf consumer. It is also unknown which flavour of mathml should be used in this case.

The pdf standard requires that glyphs and symbols are mapped to unicode. Here too variants are possible. a could be mapped to U+1D44E (Mathematical Italic Small A) or U+0061 (Latin Small Letter A), $\int$ could mapped to U+222B (Integral) or to `\int` (as it is done by the package mmap). The first alternative sounds more unicode-like but actually the screen readers don't seem to know what to do with the symbols.

The main task here is to get more information to be able to decide about which route to follow.

10 Contacts

Quite a number of questions and projects circle around the pdf specification, the needs of users and of pdf consumer applications. To get tagging working it is not enough to know how TEX works. So one important part of the tagging project is to get in contact with people having inside knowledge about pdf and pdf consumer applications in various pdf related organizations and to promote the project in the TEX world to get user feedback.

11 Summary

Adding tagging facilities to LATEX is a large project with many aspects. Happily it doesn't have to be done in one large jump: with the tagpdf package is it already possible for adventurous users with a bit of knowledge in TEX programming to tag quite large documents. Despite the clear warning in the documentation that it isn't meant for production I already got a number of feedbacks of successful uses. This gives hope that it can evolve to a stable and usable system.

References

Oberdiek, Heiko (2015). «tikz and transparent incompatibility». https://tex.stackexchange.com/a/253417/2388.

TUG (2019). «Pdf accessibility and pdf standards». https://www.tug.org/twg/accessibility/.

▷ Ulrike Fischer
LATEX Project
Mönchengladbach
fischer at troubleshooting-tex dot de