Enhancing LaTeX to automatically produce tagged and accessible PDF[*]

Frank Mittelbach, Ulrike Fischer

This paper was initially presented at the 5th International Workshop on "Digitization and E-Inclusion in Mathematics and Science 2024" (DEIMS 2024) 15–17 February 2024, at Nihon University, Tokyo, Japan. This version contains some minor updates.

The complete program and material on all presentations can be found at the workshop website [1]. A video of the talk and the demonstration is available at `https://youtu.be/7FnZv5FhmRg&?t=9869`.

Abstract

At the TUG 2020 online conference the LaTeX Project Team announced the start of a multi-year project to enhance LaTeX so that it will fully and naturally support the creation of structured document formats, in particular the "tagged PDF" format as required by accessibility standards such as PDF/UA.

In this talk we present the current achievements of this project[1] and the issues we encountered along the way. We also outline open areas of research and the future steps that we shall take to automatically produce well-tagged PDF that supports accessible standards (in particular, the recently finalized PDF/UA-2) as well as general reuse and further conversions. This will be achieved by embedding in the PDF a comprehensive description of the document structure.

Contents

[*] This article is a fully tagged and accessible PDF produced by the project software it describes.

[1] This project is carried out by a small number of developers. Besides the authors, the following individuals from the LaTeX Project Team are actively involved: Chris Rowley, David Carlisle, Joseph Wright, Marcel Krüger, and Phelype Oleinik.

We also wish to acknowledge the contributions from various members of the TeX community and beyond; these have been made through comments and suggestions, and more recently through the testing of the new functionality made available in the form of prototype implementations. Without such feedback it would be difficult to finish this project with satisfactory results.

1 General overview

For over 30 years now, the LaTeX system has been used, widely and successfully, for document production in the STEM world and also in other places where high-quality output is required; but until recently its focus was solely on page-oriented output for print (on paper) or as paged output using the PDF format. Therefore, the structural information about the document that was present in the LaTeX source did not get incorporated into the PDF output. Rather, this information was discarded as soon as possible during the processing; this was necessary so as to conserve the limited computer resources (memory and storage) that were typically available at that time (when the core of the LaTeX processing model was first designed).

As long as the intention is only to print a document on a physical medium, then this is all that is required. However, for quite a while now other uses of documents have been increasing in importance so that nowadays many documents are never printed, or printed only as a secondary consideration.

Coming into the 21st century, for many reasons great interest has arisen in the production of PDF documents that are "accessible", in the sense that they contain information to assist screen reading software, etc., and, more formally, that they adhere to the PDF/UA (Universal Accessibility) standard [3, 6], which is explained further in [2].

At present, all methods for producing such "accessible PDFs", including the use of LaTeX, require extensive manual labor[2] during either the preparation of the source or the post-processing of the PDF (maybe even at both stages); and these labors often have to be repeated after making even minimal changes to the (LaTeX or other) source.

1.1 The goals of the multi-year "LaTeX Tagged PDF" project

The main goal of the project is to enhance LaTeX so that it can *automatically* produce tagged PDF without the need to add additional data or commands to the LaTeX source, or to do any of the post-processing work necessary in other workflows.

[2] If not using the already existing code extensions to LaTeX provided by the project.

If it remains necessary to alter substantially, or to extend, each individual document in order to provide tagged PDF that conforms to some accessibility standard, then we shall see very few document authors willing to go through the pain of making such additions (unless they are forced to). It is therefore of utmost importance that the generation of tagged PDF be done essentially behind the scenes, with the only cost to the authors being a somewhat longer compilation time.

Another important aim is to make already existing documents accessible by simply recompiling them without the need to alter the source in any substantial way.[3]

The project will support the PDF 2.0 standard [4] (with the very widely supported PDF 1.7 as a fallback solution), because the PDF 2.0 standard offers a more comprehensive tag set, and it supports associated files and many other important features; its use is also a requirement of the new PDF/UA-2 standard [6].

Unfortunately, even though PDF 2.0 has already existed for six years, it has yet to be adopted for industry solutions; e.g., most viewers and other applications are still incapable of making correct use of the new PDF 2.0 features. This is largely a chicken-and-egg problem: because nobody produced 2.0 files, no application was specifically extended to enable processing such files; and due to the fact that no viewer could handle such files, the developers of PDF writers saw no need to invest in the technology to produce PDF 2.0 files.

As a result, LaTeX is one of the first authoring applications that can produce PDF 2.0 files automatically and in large quantities. In particular, LaTeX is capable of producing documents compliant with PDF/UA-2, the new standard for Universal Accessibility [6] that was finalized in 2023 and will be officially released in early 2024.[4] No doubt other suppliers will follow our lead when there is sufficient demand for the production and processing of PDF 2.0 and PDF/UA-2 conformant files.

The document entitled "LaTeX Tagged PDF Feasibility Evaluation" [10], available from the LaTeX Project website [8], explains in detail both the project goals and the tasks that need to be undertaken, concluding with the project plan that is currently being executed.

For the time being the project will focus primarily on PDF output (generated either directly by the TeX engine or through a DVI-based workflow). However, as a bonus outcome of the design approach, the implemented solution will make it easy to add other such output formats to the workflow by simply replacing the output (backend) module. Instead of PDF output, HTML5 or some other format can thus be written. As of now, such alternative backends are not part of the project coverage, but once LaTeX is able, using well-defined interfaces, to pass structure information to a backend, we expect that support for other structured output formats will follow. Such work may be undertaken by us or by other teams, possibly in parallel to later phases of the project.

1.2 Current status and achievements

As mentioned earlier, LaTeX was originally designed, as was essential 40 years ago, to be very economical with computer resources; the implementation therefore worked very hard to discard information as soon as it was no longer needed for the compilation of a document. For print output, which was all that was produced back then, these discards included most of the structural information since this was no longer useful once the visual representation had been determined. An important part of the early work on this project was therefore to alter LaTeX's inner workings by adding code that preserves this structural information from the source and adds it to the PDF.

Another part of this early "background" work was to standardize (and often to provide, for the first time) code interfaces into which extension packages can safely hook. The use of these interfaces, rather than directly overwriting internal LaTeX functions (as was commonly done in the past), avoids the problem that such packages would often break when used in certain combinations, or break when LaTeX internals changed. Moreover, it means that these packages can automatically benefit from the existence of extended workflows (such as those which produce tagged PDF).

Most of these interfaces are now in place in the LaTeX kernel. What remains (as a huge task) is to upgrade many of the core extension packages so that they make use of the new functionalities; this will enable the retirement of some of the existing code that directly overwrites LaTeX internals, or that makes assumptions (about those internals) that will become invalid in the future.

[3] Of course, required data that is not part of the document source (such as alternative text for figures or additional metadata) will need to be manually added, so as to ensure that the document is compliant with PDF/UA. But even if this work is not undertaken, the fact that the document gets automatically tagged will mean that it can be easily navigated and consumed in ways that were impossible before.

[4] At the moment we can only claim that the project software is capable of producing documents that comply with the latest draft of the imminent PDF/UA-2 standard.

The next large phase of the project was to provide automatic tagging for a subset of LaTeX documents. This task is largely finished and therefore most documents that are restricted to using only the commands and environments described in Leslie Lamport's "LaTeX Manual" [7] can be automatically tagged by adding a single configuration line at the top of the document. We say "largely finished" because a few such elements, or element combinations, are not yet covered at the time of writing.

On the other hand, a number of extension packages that go beyond Lamport are already supported, most importantly much of `amsmath` (providing extended math capabilities) and `hyperref` (enhancing LaTeX with interactive hyperlinking features). Also already supported are some of the major bibliography packages, such as `natbib` and `biblatex`.

The project is thus by now capable of producing PDF 2.0 documents that conform to the new PDF/UA-2 standard. In fact, after correcting a small number of issues (not directly related to tagging) in the class file for this conference we have been able to deliver this article as a fully tagged PDF 2.0 document.

1.3 Ongoing and future project tasks

At present, tagging support for the core document elements in a LaTeX document is still at the prototype level, which means that it works for the standard LaTeX classes and for the document elements provided by the LaTeX kernel, but it may or may not work with extension packages or classes that alter the implementation of these document elements, or that provide completely new elements.

To make further progress, some of the interfaces for tagging will first need to be finalized. Then all major extension packages, as well as all important third-party document classes, will need analyzing and possibly updating.[5] The tasks here are to identify all of the legacy low-level code for which the kernel now provides tagging-aware replacements, and then, in cooperation with their maintainers, to make the necessary updates to all these packages and classes.

In addition, any packages and classes that provide new document elements will need to specify how these elements are supposed to be tagged. Some interfaces already exist to help with this process, but it is likely that most of these will require further refinement when tested in the field.

[5] The number of widely used packages, e.g., those described in *The LaTeX Companion, third edition* [9], amounts to roughly 500, so this evaluation and code adjustment forms a substantial part of the remaining project work, and most likely will require additional volunteer support.

The remaining phases of the project, as outlined in the "Feasibility Evaluation Study" [10], cover further support for other PDF standards, and an improved interface to comprehensive metadata. There are also a number of research problems that need to be solved in order for authors to easily generate high-quality tagged PDF documents from their LaTeX sources. These are outlined below.

2 Specific aspects of the project work

We now take a look at a few specific aspects of the project work that are related to challenging problems and pose interesting research questions. These topics are: the development of a more granular tag set; the handling of formulas; the need for an extended table specification syntax; and the handling of language and script related requirements.

2.1 The existing tag set support in PDF

When PDF (already in version 1.3) first introduced a structure tree into the format, to support the inclusion of the document's logical structure, it used only a fairly minimal set of structure tags that were largely modeled after the basic HTML tag set.

For example, for mathematical formulas there was only the `<Formula>` tag itself, with no possibility to add further structure within the formula. For accessibility, all that was available was an "alt" attribute in which one could add a textual description of the formula's content. In a similar manner, other areas of document structures were (over)simplified in the tag set: e.g., for all types of floating elements there is only the tag `<Aside>` that they must share with margin notes (and even that tag is available only in PDF 2.0). For code elements, whether they are small snippets or long, commented listings, there is only a single `<Code>` tag, and there is no option to accurately describe the handling of spaces and new lines within code listings. There is a tag (again only in PDF 2.0) to denote footnotes; but if the document contains several types of structured (and possibly nested) notes, then there is no way to adequately describe this without losing possibly crucial information.

As is also the case with HTML, the relationships between these tags define a fairly simple document model that is not sufficiently rich, so that it cannot express (or not correctly express) many real-life documents; this is often due to the fact that certain elements appear in such documents with nesting relationships that are not permitted by the inclusion rules defined in ISO 32005 [5].

All this means that, when preparing a PDF to be PDF/UA-2 or PDF/UA-1 compliant, compromises have to be made and some of the structural information may thus get lost.

As part of the project we are therefore developing an extended tag set (currently called the "LaTeX namespace") that describes the logical structure of (complex) documents in more granular detail; this will help PDF processors (such as viewers) that understand this namespace to make better use of a document's structure. Ideas from this development may also prove useful in conjunction with future HTML5 developments.

2.2 The LaTeX namespace

LaTeX is an open system that allows for structural extensions (and even changes to structures) in every direction. It is therefore not possible to define a fixed (definitive) document model that is both valid and comprehensive for each and every conceivable LaTeX document.

However, it is possible to define a document model which captures the majority of LaTeX documents that are out there in the real world. If this is combined with methods to extend (and possibly alter) the document model whenever necessary for special structural extensions or changes, we are confident that a comprehensive solution can eventually be provided.

As part of the project we are therefore developing a "standard namespace" that fully describes the LaTeX document model (in the sense outlined above). This tag set will thus be noticeably more detailed and comprehensive than those offered by PDF 2.0 and HTML5. We are working with the PDF Association [11] and various application producers to ensure that this namespace will, when complete, become a recognized resource (preferably acknowledged in future revisions of the PDF standard); it may also be more generally useful as an XML schema. This will, for example, allow PDF and other applications to directly use the extended tag set it provides; and this will enable such applications to make better use of the information contained in the document, whether for accessibility support or for other purposes.

For applications that do not (yet) understand this new namespace, we provide role-mapping back into PDF 2.0 (or PDF 1.7) as necessary; but of course, in that case the more granular information provided by the tags in the new namespace will get at least partially lost.

Additionally, we will be providing interfaces that allow package or class developers who extend the standard LaTeX structures to specify how their new commands or environments map into the LaTeX namespace (and from there, if necessary, are role-mapped back to the PDF tag set).

2.3 Formulas in STEM documents

LaTeX is well-known and appreciated for its ability to describe and format mathematical or other formulas with a high degree of flexibility and unsurpassed quality. This is one of the reasons why we see a huge proportion of the documents in STEM disciplines such as mathematics, physics, and computer science being produced using LaTeX.

As described by Neil Soiffer in his keynote for the DEIMS 2021 conference [12], there are basically three methods for making such formulas accessible in a tagged PDF. One option is to use static text that can be attached as "alternative" natural language text to the formula structure, which is then read by an "assistive technology" (AT) application. While rather easy to implement, this method has various drawbacks: it does not allow for braille generation or for exploration of the equation; and the text often must be hand-crafted to avoid problems with reading software whose heuristic usually ignores certain symbols such as punctuation or braces.

A second option is to add marked-content operators to the PDF stream and then build a MathML structure tree that references this marked content. This method leads to a large structure tree with many objects, since this tree will be very fine-grained.

There are a number of problems with this second approach. It is difficult for LaTeX (without the help of a real programming language) to generate correct and useful MathML while building the TeX math list. Furthermore, when the still widely used pdfTeX engine is producing the PDF, there is a high chance that the combined processes (of simultaneously adding the necessary tagging-related material to the content stream while formatting the formula) will alter the spacing of the formula and thus render the visual representation invalid. There may be technical solutions to circumvent these issues and this is an area of active research. However, it is likely that this option can only be implemented successfully if the LuaTeX engine is used, because then a suitable programming language (i.e., Lua) is available and, furthermore (because of extended functionality in LuaTeX), it becomes possible to delay adding the necessary extra material to the content stream until after LaTeX has completed the formatting of the formula with the correct spacing.

The last option is to make use of so-called "associated files" (AF) that were introduced in PDF 2.0: these are files directly embedded into the PDF that can be attached to a structure element.[6] Each such "embedded AF" can contain, for example, a MathML representation for a formula, or its LaTeX source or some additional commentary text; more than one of these can be attached to each structure. The AF approach is simpler and easier to implement, and it also allows the use of MathML representations that do not closely follow the visual output; but it has the drawback that the MathML in the AF file is associated only to the formula as a whole and it is therefore not possible to synchronize parts of the MathML representation with the corresponding parts of the formula in the typeset document, as is necessary to support navigation of formulas and highlighting them. This method may require that the AT software overlays the printed output with its own rendering of the MathML (which may differ substantially from the original rendering).

The two last options, MathML in the structure tree or in associated files, both suffer from a lack of support in current PDF viewers and AT software: Neil Soiffer's optimistic statement in 2021 that "*Adobe's API will likely incorporate this ability in the future*" has not yet come true.

Because of this, LaTeX currently follows a threefold strategy in the prototype for math tagging: it incorporates the LaTeX source as alternate text for the formula, under the assumption that the LaTeX syntax is understandable to most readers of mathematics; and it also embeds the LaTeX source as an associated file. Additionally, an external file can be constructed in which, for all (or a selection of) the formulas, a MathML representation is provided that can be embedded in the PDF as associated files. Such an external file can be created, for example, with the help of `tex4ht` or with the LuaTeX engine. At this point in time there is no fully automatic workflow implemented for this, but with only a few adjustments it was already possible to add MathML associated files to all the formulas in the `amsmath` user documentation.

The form of the final solution for formulas, and whether or not it is necessary to offer customizable alternatives — to cater for different reader deficiencies or different user preferences — are questions that need active research to understand how to best serve consumers given the currently limited functionality of AT tools with respect to "associated files", etc.

There is another aspect of common LaTeX usage that affects all three of these method, and is also found in many other areas beyond formulas and STEM: the inclination of authors to invent new symbols, notation systems, and command names. This is nowadays exacerbated by the widespread failures to take accessibility into account. Such ad hoc extensions make it difficult to fully automate any tagging process. Overcoming this will need both technical support for such extensions and also, perhaps more importantly, encouragement of authors to keep accessibility in mind when writing documents.

The approach that will most likely be adopted to deal with this issue is as follows: by default, assume that new commands are simply abbreviations and that, by recursively replacing each of these with its definition, we eventually get to something that can be automatically tagged by using standard methods. For cases where this does not work, there will be interfaces with which the author of the document (or the package developer, if the command is defined there) can specify how the command should be interpreted when providing tagged output (e.g., MathML).

2.4 Extensions to LaTeX's table handling

In most cases, the LaTeX source will contain all the necessary information about the logical structure of a document, so that it is possible to automatically transform the source into richly tagged PDF output. There is one noticeable exception: LaTeX's handling of tabular data. This arises since standard LaTeX, and most extension packages, do not describe table data through structural information; rather, they do this in a purely visual fashion, describing only the content that should go into each cell. Thus no information is supplied concerning important relationships between cells, such as which are the header or sub-header cells, or to which cells some header cell applies.

Thus, while it is fairly trivial to tag tables as simply consisting of table rows and table data cells, determining the header cells can be done only by the use of heuristics (e.g., cell formatting changes done through `\multicolumn` are likely to represent header cells, or certain rules in a table may indicate header rows). However, any such heuristic will have a noticeable number of counterexamples.

It is therefore an important task to develop good heuristics that correctly cover a large proportion of the tables in legacy documents; and in addition to develop a syntax extension for LaTeX that allows authors to specify such logical structure explicitly in case the heuristics fail or they wish to specify explicitly the logical structure of the table. This syntax extension has to be done in a lightweight way, i.e., without putting an unnecessary burden onto

[6] Note that "associated files" do not exist as separate physical entities at the operating system level; thus they are not in fact "files" in the normal sense of the word.

the authors. Furthermore, it should be upwardly compatible with the existing syntax so that it is easily possible to enhance documents with only small alterations to the original source.

It is also important to develop methods that enable authors to easily check LaTeX's interpretation of the logical structure of a table without the need to examine the final PDF, so that they can overwrite the heuristics when necessary. This is an area of active research.

2.5 Support for different scripts and languages

Historically, the TeX engine and LaTeX were developed for ASCII-based, English documents and then (with TeX 3.0) extended to support other languages and scripts — at first, because of the restrictions to 8-bit codepages, mostly for languages using Latin scripts, but later also to non-Latin scripts, such as Greek or Cyrillic, as well as more diverse scripts. Initially, the solutions for all such scripts required complex font setups (as done, e.g., by the `CJK` package), special processors to handle transliterations, and engine extensions to handle, for example, right-to-left scripts or special input encodings.

The advent of Unicode and the Unicode-aware engines (XeTeX, LuaTeX and upTeX) led to the existence of simpler, and much more powerful, setups; therefore, most scripts are now well supported in LaTeX — perhaps with the exception of scripts that change the writing direction, since this isn't part of the original LaTeX design and thus often requires overwriting many standard commands.

The project currently concentrates on documents that use Latin scripts or scripts with similar characteristics. The correct tagging conventions to use with other types of script are not yet known by us: e.g., how to deal with direction changes or ruby characters. When using scripts (such as Latin) that typically use "whitespace" to delimit "words", tagged PDF has a requirement that even within the typeset content stream these words must remain delimited by an explicit "whitespace character" [4, §14.8.2.6.2]. This conflicts with the normal practice of TeX typesetting engines since they do not naturally add such delimiter characters; however, both pdfTeX and LuaTeX have been modified to provide workarounds for this.[7] We also do not yet know to what extent the many external packages supporting diverse scripts and languages will need to be adapted for the support of tagging. To research these topics, help from users and developers with in-depth knowledge of such scripts will be needed.

3 Some TeXnical details

In this final section we take a brief look at two technical aspects of the project work. Both will be covered in more depth during the demonstration session at the conference.

The first subsection explains how to set up a document (such as this one) so that it will automatically produce tagged PDF. This should, we hope, enable you to immediately experiment with the addition of such tags to your own works. If you want to provide feedback on any issues that you encounter, or to provide suggestions for improvements, we suggest adding them to the project repository `https://github.com/latex3/tagging-project`, using either the `issues` or the `discussions` page, as appropriate.

This is followed by some background information on the experiments we are currently conducting to automatically include in the PDF MathML representations of all the formulas in a document. We expect this work to become available for public testing during 2024/Q2.

3.1 How to enable tagging

Until recently there was no dedicated location in LaTeX documents to declare settings that affect the document as a whole. Settings had to be placed somewhere in the preamble or as class options, or sometimes even as package options. For some such settings this was problematic, e.g., setting the PDF version is only possible if the PDF output file has not yet been opened, which can be caused by loading one or another package. For the "LaTeX Tagged PDF project" [10, p. 17] further metadata about the whole document (and its processing) needs to be specified, and again all this data should be placed in a single well-defined place.

For this reason we introduced (in June 2022) the command `\DocumentMetadata` so as to unify all such settings in one place. This command takes one argument that should contain a key/value list specifying all the document metadata for the current document.[8] This should be placed at the very beginning of the document, i.e., *before* `\documentclass`; it will produce an error if found later.

The `\DocumentMetadata` command also loads the LaTeX PDF management bundle, which provides various PDF-related commands that are needed to

[7] Engines, such as XeTeX, that do not offer this workaround can therefore not be used to produce PDF/UA documents involving scripts that separate words with spaces.

[8] At this point in time only a few keys are accepted, e.g., to set the PDF version, the language, a PDF standard and to load a color profile.

```
<div>
<h2>\mml 65</h2>
<p>\begin{math}\sqrt [\beta ]{k}\end{math}</p>
<p>656E4D3BB4F29D20A1B2CBCB35C35E7E</p>
<math xmlns="http://www.w3.org/1998/Math/MathML" display="inline">
<mroot>
<mi>k</mi>
<mi>&#x03b2;</mi>
</mroot>
</math>
</div>
```

Figure 1: Sample entry with MathML data for associated file (AF)

create a tagged PDF. It also accepts the `testphase` key, which is of a temporary nature since it is needed only while new functionality is being introduced for testing. This key is used to load specific tagging support: this article, for example, uses the following:

```
\DocumentMetadata{
   testphase={phase-III,table},
   pdfversion=2.0,
   pdfstandard=a-4,
}
```

which loads the tagging support from `phase-III` (basic document elements) and `table` (newly developed prototype code for tagging of `tabular`-like environments not yet integrated in any test phase). In addition, the PDF version is set to 2.0 and it specifies that the PDF should be compliant with the PDF/A-4 standard. This is all that was necessary to produce the tagged version of this article.[9]

Eventually, the testphase code will move (once all components are considered stable) into the LaTeX kernel itself and the `testphase` key will vanish. Tagging will continue to require a `\DocumentMetadata` declaration, but will then use a simple `tagged=true` key (name to be decided).

3.2 Inclusion of external MathML

As outlined in section 2.3 on page 905 we are currently experimenting with a scheme in which externally provided MathML is embedded in the PDF as AFs. The MathML for the formulas is provided in an external file containing one or more `\mml` commands with the format shown in figure 1 (i.e., surrounded by HTML tags so that it can be proofread in a browser).

The first argument to `\mml` is a label (e.g., a number) to that uniquely identifies this MathML snippet; the second argument contains the LaTeX source for the MathML. The third argument is the MD5 hash of the LaTeX source. Its use ensures that the PDF file will contain only one AF for any formula, even if a formula is repeated several times: for example, if the LaTeX source document repeatedly uses `$\beta$`, then each repetition of exactly this formula will reference the same embedded AF, which means that the PDF file does not become unnecessarily large.

The final argument contains the corresponding MathML. In our current experiments the MathML is generated from the LaTeX source by processing it with `tex4ht`, with some further processing to add the MD5 hash values and with some manual corrections to improve the resulting MathML. One advantage of using an external file at this stage is to allow the MathML to be validated before being embedded as an AF in the PDF. The MathML could potentially be generated by other TeX to MathML conversion programs such as `latexml` or `luamml`, which would allow experimentation with different pipelines to construct associated files containing MathML.

This file is then input at the beginning of the document and each MathML (with a unique hash value) is embedded in the PDF as the content stream of an AF. The LaTeX code to produce tagged PDF then checks, for each math formula in the document, whether an associated file containing MathML for this formula has already been added to the PDF, and, if so, a reference to this MathML associated file is added to the `<Formula>` structure element being constructed. Therefore, if the same math expression occurs more than once (as a complete formula) then each occurrence will reference this same MathML AF.

Currently, the generation of the file of MathML fragments requires some manual editing and explicit execution of conversion programs. The next step will be to create scripts that will: run directly in a LuaTeX compilation; fully automate the generation of the MathML fragments from the LaTeX source; and validate this output.

[9] This paper does not contain any tabular material, thus `table` is actually unnecessary for tagging this article. The setting was added to show how the interface can be used when new functionality is made available.

References

[1] DEIMS 2024. Website of the *5th International Workshop on Digitization and E-Inclusion in Mathematics and Science 2024*, Nihon University, Tokyo, Japan, February 2024. `workshop.sciaccess.net/deims2024/`.

[2] Olaf Drümmer and Bettina Chang. *PDF/UA in a Nutshell — Accessible documents with PDF*. PDF Association, August 2013. `pdfa.org/resource/pdfua-in-a-nutshell/`.

[3] *ISO 14289-1:2014; Document management applications — Electronic document file format enhancement for accessibility — 1: Use of ISO 32000-1 (PDF/UA-1)*, 2nd edition, 2014. `www.iso.org/standard/64599.html`.

[4] ISO. *ISO 32000-2:2020(en); Document management — Portable document format — Part 2: PDF 2.0*, 2nd edition, 2020. `iso.org/en/contents/data/standard/07/58/75839.html`.

[5] *ISO/TS 32005:2023; Document management — Portable Document Format — PDF 1.7 and 2.0 structure namespace inclusion in ISO 32000-2*, 1st edition, 2023. `iso.org/en/contents/data/standard/04/58/45878.html`.

[6] *ISO/FDIS 14289-2; Document management applications — Electronic document file format enhancement for accessibility — Part 2: Use of ISO 32000-2 (PDF/UA-2)*, 1st edition, 2024. `www.iso.org/standard/82278.html`.

[7] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison Wesley, 2nd edition, 1994.

[8] LaTeX Project Team. Website of the LaTeX Project. `latex-project.org/`.

[9] Frank Mittelbach and Ulrike Fischer. *The LaTeX Companion*. Addison-Wesley, Boston, MA, USA, third edition, 2023.

[10] Frank Mittelbach, Ulrike Fischer, and Chris Rowley. *LaTeX Tagged PDF Feasibility Evaluation*. LaTeX Project, September 2020. `latex-project.org/publications/indexbyyear/2020/`.

[11] PDF Association (PDFA). Website of the PDF association. `pdfa.org/`.

[12] Neil Soiffer. Accessible PDF: $2 \not> 1$. In *The 4th International Workshop on Digitization and E-Inclusion in Mathematics and Science 2021*. The DEIMS2021 Organizing Committee, 2021. `workshop.sciaccess.net/deims2021/DEIMS2021_Proceedings.zip`.

◇ Frank Mittelbach
Mainz, Germany
`https://www.latex-project.org`

◇ Ulrike Fischer
Bonn, Germany
`https://www.latex-project.org`