SpeLaTeX — Speech-enabled LaTeX

Walter Daems (walter.daems@uantwerpen.be) and Paul Levrie

v0.91— 2024/06/21

1 Preface

1.1 Background

At our institute, the University of Antwerp in Belgium, the number of students with reading and/or writing disorders (or at least aware of these disorders) is increasing. Approximately 5% of the students are registered with such a disorder. Probably there's an additional number of people opting not to register their disorder.

A large portion of the study materials we offer to students is still written material. The authors believe that this will keep on being so, even given the multimedia and AI options that have become mainstream. Let's not go into this debate for now. However, written course texts cannot be but suboptimal for students with reading disorders.

For small texts, reading them out loud and recording them using a voice recorder to create an aid for our target group, is still feasible. We have taken that route at our institute for exam assignments. However, for bigger texts (like course syllabi) this is beyond the time a teacher can afford spending on this small a group of students. Yes, economic laws also govern teaching!

1.2 State of the art

1.2.1 In general

Therefore, reverting to readily available text-to-speech software is an obvious choice. Nowadays, special software exists that provides the functionality of reading out loud the contents of electronic documents, e.g. NVDA [1] or SprintPlus [2]. Moreover, more and more standard PDF readers (such as Acrobat Reader [3] have the facility of performing text-to-speech). For nontechnical subjects, this works fine. However, when it comes to technical course syllabi that are loaded with non-trivial mathematics, the standard text-to-speech packages fail to meet our expectations. In addition, they cannot read a sensible textual description of figures or tables.

The issue with reading mathematics will be solved in the future, by enforcing mathematical equations to follow a specific standard that can be parsed and converted, not only into a visual representation, but also into proper text. MathML will be one of the candidates for that format.

The issue with figures and tables can be solved by using the tag infrastructure of PDF. The format provides a system of tags that allow you to provide extra information about the content of a document, in much the same way as you can specify an 'alt' key for an image in HTML. This tag could contain a proper textual description of the figure or the table.

1.2.2 For LaTeX

Currently, the LaTeXproject is investing in enabling LaTeX to partially automate the tagging of PDFs with the `tagpdf` package [4], such that the user only has to do a minimal job (adding tags for figures and tables). The goal is to maximize the accessibility of LaTeX-produced documents.

So all is well? Not quite. Though I'm confident that with enough time the community will solve the issue completely, there are still some gaps:

- The tagpdf package is still not a part of the mainstream LaTeX-kernel.
- MathML reading support in PDF readers is still in its infancy.
- Many PDF readers do not fully support tags yet.

1.3 This package

This package aims to overcome these problems in the meantime and also to contribute to the longterm goal: making perfectly tagged PDFs that are read by any PDF-reader.

In the *first phase* of this package (i.e. the version you are looking at right now), this package reads your LaTeX source, converts the text and the formulas to audio files and equips your PDF with hyperlinks to these files, such that with a few clicks you can listen to your document. The audio files are external files that should be packaged with your PDF to allow a reader to use the document with the available audio. How does it read the formulas? It parses your LaTeX constructs and makes the best of it. This will probably be the part that might survive up until the very last phase of this package.

In a *second phase* of this package, the audio files will be embedded into the PDF. Currently, there are not enough PDF readers that support this feature. Therefore, we decided to keep using the external audio files for now.

In a *third phase*, the audio files *may be* abandoned alltogether, fully switching to tags. And we do think that this should be the end goal. The reasons why we say *'may be'*, are:

- The voices of the current PDF-readers are still not of the same quality as the ones available online. And quality does matter.

- The better voices may require cloud access, and probably will not be free, therefore (me) paying for them at document creation time, makes more sense to me than having my students pay for these voices whenever they read the document.
- The industry standard Adobe reader is not easily available on open-source operating systems (like UNIX/BSD/Linux-derived platforms). You might consider using emulation using wine [5], but in that case you can forget about audio. Free access to software is something we consider to be a must-have, rather than a nice-to-have. In addition, Walter feels miserable when he's forced to use anything else than GNU/Linux, because then he keeps moaning that his productivity is ruined.

Admittedly, in this first phase, using the S*D*eLaTeX package causes a significant overhead in writing your text. Therefore we also provide you the S*D*eLbox extension for the Org-mode [6] of GNU Emacs [7]. To satisfy your curiosity: S*D*eLbox stands for "Speech-Enabled LaTeX By Org-mode eXport".

Will S*D*eLaTeX become obsolete in the future? Undoubtedly so. But for the time being, it answers our desire to provide our students with good audio support when studying their engineering courses. That is now, not only in five years time.

We hope that you enjoy using our software, or that — if you are not pleased with it — it triggers you to give us feedback or to come up with a better solution. We especially would like to thank Ulrike Fischer (of the LaTeX-project and the maintainer of the tagpdf package) for trying to use this package and reaching out to give us feedback. One of her suggestions (not to use big hyperlink-areas) was almost instantly implemented and has been adopted as the standard way of linking.

You are free to use our software but kindly ask you to provide a mention "The audio materials of this text have been prepared with S*D*eLaTeX/S*D*eLbox" in the section treating copyrights, bibliographic data or any other spot that is suited. We'd also welcome a short mail of yours telling that you make use of the package. The pleasure of receiving such an e-mail makes our day.

You are free to modify this LaTeX-package, keeping a reference to our original package intact, provided that your package is subject to the LPPL license, as is S*D*eLaTeX. However, contributing to our package might be a better way to go, in order to bundle the efforts for a better speech-enabled LaTeX.

2 Introduction

2.1 Target audience

S*D*eLaTeX is primarily intended for persons with a reading disorder. This may be:

- persons suffering dyslexia
- visually impaired persons

– persons who still can recognize the basic parts of a book, i.e. are able to operate a PDF viewer and click on the individual parts.
– persons who can't recognize the basic parts of a book (e.g., blind persons): they can listen to the automatic playback of the ordered chain of audio fragments.

But also people who want to multitask, e.g., gardening while listening to a technical book, can benefit from SpeLaTeX.[1] Personally, we often use SpeLaTeX to proofread texts as we hear language errors more easily than we spot them while reading.

2.2 The magic under the hood

2.2.1 The SpeLbox-ecosystem

The SpeLaTeX-package is one of the parts of the SpeLbox ecosystem. It consists of three components:

- An Org-mode exporter called `ox-spel` [8]
- This LaTeX-package SpeLaTeX [9]
- A script embeded in the Perl module `SpeL::Wizard` [10]

2.2.2 The overall picture

SpeLaTeX equips the PDF that is generated by LaTeX with hyperlinks to audio files that contain the spoken equivalent of the original text, equations, figures and tables

Let's look at this in detail in Fig. 1. For now, ignore the two top boxes, and let's assume you are composing the file `text.tex` as your document. We will return to the two top boxes later.

By loading the `spel.sty` package in your source document, LaTeX will produce a PDF file that references audio files that will be generated later (see below). In addition, it generates text chunks (i.e. small portions of your text) in separate files (`.tex`) and a spel index file (`.spelidx`) referencing them in sequence.

Together with the `.aux`-file (needed by SpeLaTeX for labels and citations), these are the inputs to the SpeLaTeX-engine (`spel-wizard.pl`) that parses the text chunks, writes a full text version of them as spel files (with extension `.spel`) and controls the text-to-speech engine to generate audio files of them.[2]

To avoid excessive text to speech conversion (i.e. an expensive step) the SpeLaTeX engine derives a finger print of them and compares it to previously generated fingerprints for the chunk. If the finger print has changed, the audio file is overwritten (or created the first time), otherwise it is left untouched.

[1] Note that multitasking is not reserved for persons without visual impairment. Also visually impaired persons can benefit from listening to an audiobook while doing other things.
[2] In the figure, Ogg Vorbis has been chosen as format, but this can be any audio format.

As a cherry on top, the SpeLaTeX-engine also creates a playlist, such that you may use the audio files for a PODcast-like listening experience.
Something that has not been indicated on the figure, is that for reading out loud entire (sub)sections, the PDF-file also references m3u playlists that gather all chunks belonging to the (sub)section.

You might wonder: where are the links? Well, there are three variants:

- *area links*, which are almost exclusively used for equations, tables or figures. These links make an entire rectangle active, linking to the corresponding audio file.
- *group links*, indicated by small right-pointing triangles next to section headers. These cause all blocks in the section to be read one by one.
- *hidden links*, wich are, as the name suggests, hidden to avoid visual disturbance of the text; every paragraph, footnote and list is activated by a link on the whitespace before the first line of the chunk. Try it! Once you are aware that these regions are active, you'll find them easily. In addition, not using the full paragraph as an active region, allows existing hyperlinks (like for citations, references or URLs) to still function.

2.2.3 Implicit spelchunks

Generating the text chunks to be read out loud requires us to use special LaTeX-macros. For all pieces of text that are within an existing macro (e.g. `\title`, `\author`, `\section`, `\caption`, `\footnote`, `\thanks`), these macros have been redefined by the SpeLaTeX-package to execute the magic without any further hassle. We call these LaTeX-fragments *implicit spelchunks*.

However, not all LaTeX-constructs can be dealt with in an automatic way. This is true for any `\item` you put inside a list. You need to replace that with a `\spelitem` that takes the text that follows as a first explicit argument, i.e. `\item blabla` should be replaced by `\spelitem{blabla}`. We call these LaTeX-fragments *defunct implicit spelchunks*. They should have been implicit, but we could not get that to work without problems. Therefore you need to mark them explicitly as `\spelitem` constructs.

2.2.4 Explicit spelchunks

One would hope that displaymath environments are also implicit spelchunks. However, overriding environments in LaTeX is a tricky business. In view of this, the SpeLaTeX package keeps away from this, and we've chosen to treat displaymath environments (like `equation`, `eqnarray`, `gather`, `align`, `alignat`) the same way as tables or figures, i.e. you need to embed the in a `spelchunk` environment. The only difference between math environments and figures an tables is that displaymath environments can be automatically read out loud by the `spel-wizard.pl` script, while you will have to provide a text description for tables and figures manually, using a subsequence `spelchunkad` environment (the suffix ad stands for *audio description*).

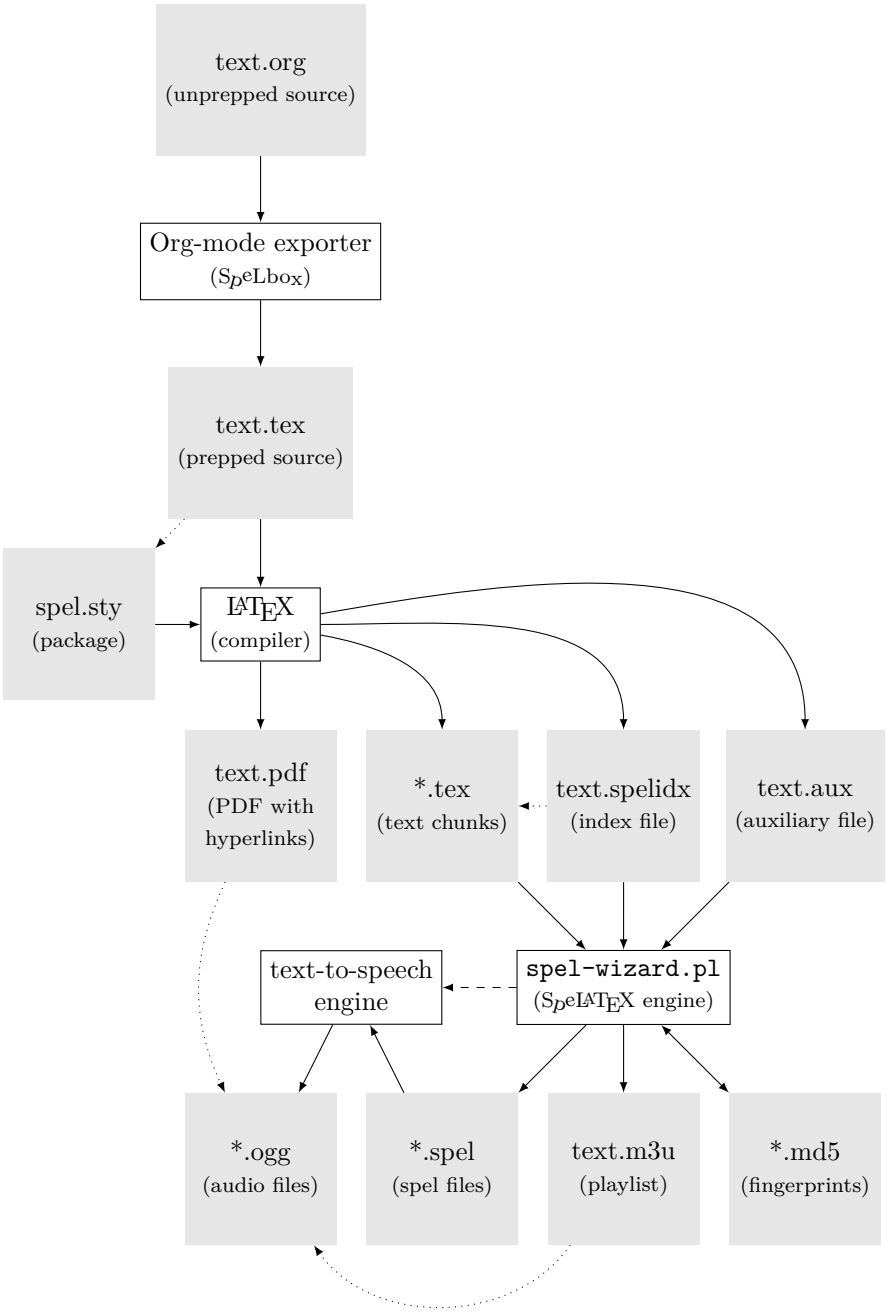

Figure 1: Basic tool setup of SpeLbox of which the SpeLaTeX package is an integral part; filled boxes indicate files, outlined boxes indicate tools; solid lines indicates use or creation of files, dotted lines indicate references, dashed lines indicate a control relationship.

To keep the terminology clear, we label them as automatic and manual spelchunks respectively:

- automatic: `equation`, `eqnarray`, `gather`, `align`, `alignat`, …
- manual: `figure`, `table`, `tikzpicture`, …

The similarity between both categories is that you both embed them in a `spelchunk` environment, but that you provide the manual textual description using a `spelchunkad` environment right after the chunk: in this way, the `spelchunkad` environment provides an alternative text for the chunk. Note that this way of working also enables you to provide an overriding text for an equation, if you think that `spel-wizard.pl` is doing a bad job. You do me a favor if you submit this subpar behavior as a bug report to me!

It is worthwile to make good descriptions for a figure or a table. It's here that you can create true added value to your manuscript, even for readers without impairment!

2.2.5 Paragraphs

However, there is the elephant in the room that we did not yet talk about, the paragraph. Indeed, it is desirable to split plain text according to the paragraphs in the document. Alas, paragraphs are one of the silent features that are not easily accessible from within a LaTeX-package. Therefore paragraphs (or smaller text chunks) are considered to be explicit spelchunks and need to be embedded in `spelchunk` environments. This environment will cause its contents to be hyperlinked to a separate piece of audio.

Encasing all your paragraphs with `spelchunk` environments manually is a pain. There is a reason why in LaTeX paragraphs can be created by a double newline: convenience. The `spelchunk`-environment encasing ruins this totally!

2.2.6 Emacs Orgmode to the rescue!

However, we provide an exporter for the Org-mode [6] of Emacs [7], such that you can prepare your manuscript in Org-mode and not worry about all these exlicit tricks with `spelchunk` and `spelitem` constructs.

This brings us to the two top boxes in the diagram of Fig. 1. If you prepare your manuscript in Org-mode, and use the Org-mode exporter, you don't have to worry about explicit spelchunks, or defunct implicit spelchunks. Therefore, we encourage you to use Emacs as your editor. The learning curve is steep, but the rewards regarding ease-of-use are huge. In that way, you can focus on providing textual descriptions for the non textual portions of your manuscript, e.g. giving a good description of a graph.

Note that this very document has been prepared manually to make it independent of Org-mode. It also shows that if you don't want to use Emacs and Org-mode, you will suffer some considerable pain in equipping your manuscript with an overload

of `spelchunk` environments. We have only one advice: don't use the SpeLaTeX outside SpeLbox. Outside a proper boxed confinement, spells easily become curses…

2.3 Extra math commands

Some of the ways to specify mathematical expressions in LaTeX is very liberal, what makes converting them to text quite difficult. Therefore, we also provide some extra constructs that make life easier for both parties: you as a user and `spel-wizard.pl` as a parser.

An example of this are sets. We provide two commands to define a set. As we want these commands to blend in with general LaTeX, we did not equip them with a prefix `spel`. Therefore, we made activating them conditional to specifying the package option `extramath`.

`\setenum` a command to define a set that consists of comma or semicolon separated elements

`\setdesc` a command to define a set that is specified using a description

2.4 Added value

Why would it make sense to use SpeLaTeX? We think there are many selling points. We can mention a few:

Minimal overhead Preparing a LaTeX manuscript for use with SpeLaTeX requires a minimal amount of work, if you are using Org-mode in Emacs.

Free for the content provider If you are using a freeware text-to-speech engine (like for example festival [11] or balabolka [12]) and a royalty-free audio format and player (like for example ogg-vorbis), generating audio-enabled documents only requires the effort of preparing your manuscript. There are no license costs involved.
You could also consider to use an online paying text-to-speech service. As an example, We incorporated a connection to Amazon's Polly [14].
In addition, if your user has a better-quality (maybe commercial) text-to-speech system, he/she can reconvert the text files him-/herself, equiping your document with a voice they like and are used to, without you having to worry about license costs. They might even use an AI-generated copy of their own voice!

Free for your audience In addition, the user of your audio-enabled document doesn't need to buy a license for text-to-speech software. Only a PDF-viewer and standard audio-player program are required.

Math capable Try some of the equations in this manuscript. We are quite confident you'll be convinced fairly soon.

3 Installation

3.1 The SpeLaTeX package

If you are a package manager then you'll know how to prepare an installation package for SpeLaTeX.

If you are a normal user then you have two options. First, check if there is a package that your favorite LaTeX distributor has prepared for you. Most of the major distributions (like e.g. Tex Live or MikTeX) do so.
Second, grab the TDS package from CTAN [15] (spel.TDS.zip) and unzip it somewhere in your own TDS tree, regenerate your filename database and off you go. In any case, make sure that LaTeX finds the file `spel.sty`.

The SpeLaTeX package uses a number of auxiliary packages, fetch them from CTAN [15] if your TeX distributor does not provide them. The ones used are: `expl3`, `hyperref`, `ifthen`, `fancyvrb`, `newfile`, `rotating`, `babel`, `kvoptions`.

3.2 The `spel-wizard.pl` speech generator

3.2.1 The script

You can install the wizard assuming you have a working Perl interpreter installed. Assuming you're on GNU/Linux or MAC, you should be able to find an installation package using the package manager for your distribution. If you are on MS-Windows, look for Strawberry perl or ActiveState perl.

The only thing to do is to install the `SpeL::Wizard` module. You can do this with the perl pacakge manager for your interpreter.

Open a terminal or command window, and then enter on the command line (the dollar represents your prompt):

On GNU/Linux and MAC: `$ cpanm SpeL::Wizard`
On Strawberry perl: `$ cpan SpeL::Wizard`
On ActiveState perl: `$ ppm install SpeL-Wizard`

The script `spel-wizard.pl` will be installed on your system. Make sure it is on your search path.

3.2.2 The configuration file

Finally, you need to provide `spel-wizard.pl` with an appropriate config-file, named `tts.conf` that sets up the text-to-speech conversion. Below you can find a setup for Festival [11]:

```
[engine]
tts=festival

[languagetags]
```

```
dutch=nl
english=en-gb

[voices]
dutch=nl1_mbrola
english=en1_mbrola
```

And additionally, for the Microsoft users a setup for Balabolka [12]:

```
[engine]
tts=balabolka

[languagetags]
english=en-us

[voices]
english=Zira
```

The tts configuration parameter defines the speech engine to use. The language-tags section defines how the babel languages are mapped to internationalization codes (also known as locales). The voices section specifies what voice to use for a specific language.

An environment variable can specify where your config file is located, e.g., on GNU/Linux:

```
$ export SPELWIZARD_CONFIG=/home/wdaems/.config/tts.conf
```

If that variable is not set, the script will look in a subdirectory `.spel_wizard` of your home-folder (or `%userprofile%` in MS-Windows), or it will take the default that came with the `SpeL::Wizard` module.

Be aware that you need to install your text-to-speech tool yourself according to the documentation provided by the tool provider. In addition, make sure it the executable is in your search path. In case you are using an online text-to-speech service provider you will need to get an account on their cloud platform and setup credentials and whatever is needed to get going. Providing assistance for this is beyond the aim of this manual.

3.3 The PDF viewer

You need to make sure you have a PDF-viewer that supports links containing 'run' tags. E.g., xpdf [17] does not, evince [18] complains about security risks, but okular [19] and Adobe Reader [3] do. So does PDF-XChange Viewer [20].

3.4 The media player

When clicking a SpeLaTeX-enabled item in your PDF-file, your media player is started to play the `.ogg` or `.m3u`-file. On GNU/Linux most media players work fine (SoX, totem, vlc, …).

On windows, we recommend using vlc. It works out of the box. When usign the stock Windows Media Player, you will need to add every folder that contains a PDF you'd like to have read, to your Media Player library. Search the internet to find instructions on that and be prepared: in line with Microsoft's standard practice it is well hidden in the interface.

4 Usage

4.1 Preparing your document source

Using the SpeLaTeX package is very simple. Just load the package's style file using an appropriate `\usepackage{spel}`.

In case you are not using Org-mode, there are 5 things to do:

1. Treat the defunct implicit spelchunks
2. Treat the explicit spelchunks
3. Manually provide text to read when needed
4. Provide audiodescriptions or preprocessing instructions for your typesetting macros
5. Provide audiodescriptions or preprocessing instructions for your typesetting environments

Steps 1 and 2 can be done automatically for you by using Org-mode in Emacs and using the proper `spel` exporter. Steps 3, 4 and 5 are up to you. In this section, we assume you are executing all 5 steps manually and therefore, we will explain all required macros.

4.1.1 Treat the implicit spelchunks

The texts of chapter, (sub)section titles, a.s.o. will be formatted automatically such that they are hyperlinked to the appropriate audio file. Therefore, this step was not mentioned above. It is done automatically for you by using the SpeLaTeX package.

You only need to cover your *defunct implicit spelchunks*:

`\spelitem` Use this macro instead of the `\item` macro to make sure your list environments are converted to speech chunks appropriately.

Example:

```
We like
\begin{itemize}
  \spelitem{apples,}
  \spelitem{pears, and}
  \spelitem{oranges.}
\end{itemize}
```

Another example:

```
If you don't know these fruits:
\begin{description}
  \spelitem[apple]{a green round fruit}
  \spelitem[pears]{a green pointy-shaped fruit}
  \spelitem[orange]{an orange round fruit}
\end{description}
```

Note that the spel exporter in Org-mode takes care of all this boilerplate work.

4.1.2 Treat the explicit spelchunks

`spelchunk` (*env.*) Use this environment to embed the chunks of text in that you want to generate audio for. In case the content is an equation, a figure or a table, we recommend specifying `arealink` as the optional argument to the `spelchunk` environment. It makes the entire equation an active hyperlink.

Example:
(note: the example below is not SpeLaTeX-enabled because it generates internal package problems)

```
\begin{spelchunk}
  An orderinary paragraph must be embedded in this environment.
  The same holds for equations! However, then we recommend using
  the |arealink| option, as that makes the full area of the
  equation clickable and avoids an empty white line before the
  equation.
\end{spelchunk}
\begin{spelchunk}[arealink]
  \begin{align}
    E &= m c^2\\
    e^{j\pi} &= -1
  \end{align}
\end{spelchunk}
```

4.1.3 Manually provide text to read when needed

`spelchunkad` (*env.*) If you want a different text to be used for the previous spelchunk environment, this environment allows you to specify it. For plain text or math environments, this is also your generic escape route in case the `spel-wizard.pl` parser does not work as you'd like it to.
Just have your `spelchunk` environment followed by a `spelchunkad` environment that specifies the correct text to read out loud. However, please, file a bug report, such that we can improve the tool.

Example:
(note: the example below is not SpeLaTeX-enabled because it generates internal package problems)

```
\begin{spelchunk}
  An orderinary paragraph must be embedded in this environment.
\end{spelchunk}
\begin{spelchunkad}
  Do not forget to embed ordinary paragraphs in this environment.
\end{spelchunkad}
```

For non-textual material such as figures or tables, this allows you to specify a sensible text that acts as an audio description for that material.
Just have your `spelchunk` environment that surrounds your figure or table, followed by a `spelchunkad` environment that provides the audio description for the non-textual material.

Example:
(note: the example below is not SpeLaTeX-enabled because it generates internal package problems)

```
\begin{spelchunk}
  \includegraphics{engine.jpg}
\end{spelchunk}
\begin{spelchunkad}
  The image shows a turbo-fan engine of an aircraft. One can
  clearly see the silver blades of the fan, and the housing. Note
  how little spacing there is between the blades and the housing.
\end{spelchunkad}
```

4.1.4 Provide descriptions for typesetting macros

`\spelmacad` Often, recurring constructs are being typeset using a dedicated macro, defined by the user. For example, to consistently typeset input voltages for arbitrary pins, one might have defined the macro:

```
\newcommand\vin[2][IN]{\ensuremath{v_{\mathit{#1},#2}}}
```

This allows easy specification of

```
\vin{1} = \sin 20 t
```

resulting in $v_{IN,1} = \sin 20t$.

However one might want this line to be read as 'the input voltage at pin 1 equals sine 20 t'.
To this end, one can provide an description for this macro using the `spelmacad` macro.

Example:

```
\spelmacad{vin}[1][IN]{the #1put voltage at pin #2}
```

Note that the audio description in this case will only be acceptable, for arguments IN and OUT. One clearly has to take the audio description into account when defining LaTeX-macros.

4.1.5 Descriptions for typesetting environments

\spelenvad Often, recurring constructs are being typeset using a dedicated environment, defined by the user. For example, to consistently typeset a proof or illustration one might have defined the environment:

```
\newenvironment{proof}[2][Proof]{
  \textbf{#1: #2}\\
}
{
  \hfill$\blacksquare$\\
}
```

This allows easy specification of an illustration as:

```
\begin{proof}[Illustration]{solving a quadratic equation}
  blabla
\end{proof}
```

However one might want this environment to be read as 'Illustration of solving a quadratic equation: blabla. This concludes this illustration.'
To this end, one can provide an description for this macro using the `spelenvad` macro.

Example:

```
\spelenvad{proof}[1][Illustration]
{#1 of #2:}
{This concludes this #1.}
```

4.1.6 Using the i18n features of `spel-wizard.pl`when describing your macros and environments

Sooner rather than later you will feel the need to provide reading alternatives for your constructs that are language dependent. In that case you can call the i18n features that are built into `spel-wizard.pl`. We illustrate this with an example.

Assume you've made your own command to raise numbers to a power, and you provide and description for your macro.

```
\newcommand\numtopower[2]{#1^{#2}}
\spelmacad{numtopower}[2]{#1 to the power of #2}
```

The problem with this solution is, that it only works for one language. The solution is to use an i18n expression in your description:

```
\spelmacad{numtopower}[2]{#1 @{i18n(Power,#2)}}
```

This will call the maketext function (See Locale::Maketext) on the Lexicon provided in SpeL::Wizard::I18n, as:

```
$SpeL::Wizzard::I18n::lh->maketext( 'Power', "#2")
```

to read your macro.

4.1.7 The extra math commands

Note that these commands are only available if you provide the package option `extramath`.

`\setenum` This macro typesets an enumeration set and makes sure `spel-wizard.pl` can read it properly.

```
\begin{equation}
  P = \setenum{ 2, 3, 5, 7, 11, 13, \ldots }
\end{equation}
```

`\setdesc` This macro typesets an enumeration set and makes sure `spel-wizard.pl` can read it properly.

```
\begin{equation}
  P = \setdesc{ n \in N \mid n \text{~is prime} }
\end{equation}
```

4.2 Going through the flow

Once your document source has been prepared, you are ready for the regular SpeLaTeX-flow. It consists of 3 steps.

1. Create a `jobname-spel` subdirectory in the working directory your LaTeX source document is in (replace jobname with the basename of your latex file, the final `-spel` is a literal).
2. Run your document 3-times through your {pdf,Xe,Lua}LaTeX-compiler to get all the references right.
3. Run the `spel-wizard.pl` speech generator (see scripts directory or the wrapper provided by the package manager), by launching it with the base name of your document as command-line argument.
   E.g.: `spel-wizard.pl -v example`
   The `-v` argument causes the script to be somewhat more verbose.

The result of this will be a PDF file equipped with links to audio files in the 'speech' subdirectory. Alas your PDF file has been become a little less portable, as it now requires the 'speech' subdirectory to be complete. You might want to package the ensemble into a tar-file or zip-archive.

5 Example

Below, you can find a simple example to give you a head-start. In order not to spoil the fun for you, the embedded version here is not speech-enabled.

```
\documentclass{article}

\usepackage[dutch,english]{babel}   % load babel before spel to avoid
                                    % option clash!
\selectlanguage{english}
\usepackage[format=ogg]{spelatex}

\newrobustcmd\CTAN{CTAN}
\spelmacad{CTAN}{see-tan}
\newrobustcmd\CPAN{CPAN}
\spelmacad{CPAN}{see-pan}

\title{\spelatex{} Example}
\author{Walter Daems and Paul Levrie}
\date{2011/04/12}
\setlength\parindent{0em}
\setlength\parskip{1ex}

\begin{document}

\maketitle

\section{Introduction}

\begin{spelchunk}
  This file is just a simple showcase of the features of \spelatex.
  Below, you'll find examples of:
\end{spelchunk}

\begin{itemize}
  \spelitem{a simple equation}
  \spelitem{a more complex equation}
\end{itemize}

\section{A simple equation}
\label{eqn:simple}
\begin{spelchunk}
  Consider the following simple definition of a polynomial function and
  check its spoken version by clicking on it.
\end{spelchunk}
\begin{spelchunk}[arealink]
  \begin{equation}
    f(x) = x^{5}- x^4 + 7 x^3 + 3 x^2 - 8 x + 23
```

```
  \end{equation}
\end{spelchunk}
\begin{spelchunk}
  This seems a simple equation, however, it is not so straightforward
  for an automated reader, to read it correctly.
\end{spelchunk}

\section{A more complex equation}
\newcommand\xx[2]{\ensuremath{#1_{#2}}}
\spelmacad{xx}[2]{#1 #2}

\label{eqn:complex}
\begin{spelchunk}
  For a lightray that hits the parabola at the point
  $P(t,9-\frac{t^2}{4})$, the reflected ray has slope $\tan 2\alpha$.
  Since the slope of the tangent to the parabola at $P$ is
  equal to $\tan\alpha = -\frac{t}{2}$, the equation of the
  reflected ray is given by
\end{spelchunk}
\begin{spelchunk}[arealink]
  \[
  y-9+\frac{t^2}{4} = -\frac{4t}{4-t^2} \cdot (x-t)
  \]
\end{spelchunk}

\selectlanguage{dutch}
\section{Een andere taal}
\begin{spelchunk}
\spelatex{} is ook volledig babel-actief, wat wil zeggen dat de
voorleesstem de geselecteerde taal zal volgen.
\end{spelchunk}

\begin{spelchunk}[arealink]
  \[
  y-9+\frac{t^2}{4} = -\frac{4t}{4-t^2} \cdot (x-t)
  \]
\end{spelchunk}

\selectlanguage{english}
\section{And some extras}
\subsection{Citations}
\begin{spelchunk}
Two excellent repositories are \CPAN{} \cite{CPAN} and \CTAN{} \cite{CTAN}.
\end{spelchunk}

\subsection{References to labels}
\begin{spelchunk}
Section~\ref{eqn:simple} contains an illustration of a simple
equation. For a more complex equation, we refer the user to
section~\ref{eqn:complex}.
```

```
\end{spelchunk}

\bibliographystyle{alpha}

\begin{thebibliography}{99}

\bibitem{CTAN}
The Comprehensive \TeX{} Archive Network.
\newblock \url{http://www.ctan.org}.
\newblock online, accessed in August 2021.

\bibitem{CPAN}
The Comprehensive Perl Archive Network.
\newblock \url{http://www.cpan.org}.
\newblock online, accessed in August 2021.

\end{thebibliography}

\end{document}
```

6 Demo

The examples below have been composed and used to test the math reading capabilities of SpeLaTeX and `spel-wizard.pl`. The source code has not been made visible in this document. If you'd like to see the source code, check the original `.dtx`-file that was used to generate this PDF-file.

6.1 Numbers

$$\pi \tag{1}$$

$$-31415 \tag{2}$$

$$1.25 \tag{3}$$

$$-0.34 \times 10^{4} \tag{4}$$

$$12 - j3 \tag{5}$$

$$-31415.23 + .45i \tag{6}$$

6.2 Fractions

6.2.1 A fraction only containg numbers

$$x = -\frac{1}{2} \tag{7}$$

$$y = -\sqrt{\frac{\pi}{2}} \tag{8}$$

6.2.2 A fraction with a little more under the hood

$$u = -\frac{x^2 + 35}{\sqrt{12}} \tag{9}$$

$$v = -\frac{\sqrt{\frac{\pi}{2}}}{-3x^2 + 3} \tag{10}$$

6.3 Simple expressions

6.3.1 A polynomial function

$$f(x) = x^5 - x^4 + 7x^3 + 3x^2 - 8x + 23 \tag{11}$$

6.3.2 Some more complex equations

Here's de Moivre's formula:

$$(\cos x + j \sin x)^n = \cos(nx) + j \sin(nx) \tag{12}$$

Euler's relationship:

$$e^{j\phi} = \cos\phi + j \sin\phi \tag{13}$$

Euler's identity:

$$e^{j\pi} + 1 = 0 \tag{14}$$

6.3.3 A rather well-known definite integral

$$\int_{-\infty}^{\infty} e^{-x^2}\,\mathrm{d}x = \sqrt{\pi} \tag{15}$$

6.4 Sets

Let's check the two set commands this package provides: `\setenum` and `\setdesc`:

$$P = \{2, 3, 5, 7, 11, 13, \ldots\} \tag{16}$$
$$P = \{n \in N \mid n \text{ is prime}\} \tag{17}$$

6.5 Matrices

How about some linear algebra?

$$\begin{bmatrix} 3 & 4 \\ 7 & 2 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{18}$$
$$\begin{vmatrix} 3 & 4 \\ 7 & 2 \end{vmatrix} = -22 \tag{19}$$

6.6 Figures and Tables

6.6.1 Figures

The example Fig. 2 illustrates the voice-aid that can be added to figures.

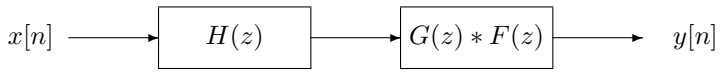

Figure 2: A block diagram of the filter system

6.6.2 Tables

Food	Sweet	Bitter
apple	•	
unsweetened coffee		•
cake	•	
chocolate	•	•

6.7 A parabola tale

For a lightray that hits the parabola at the point $P(t, 9 - \frac{t^2}{4})$, the reflected ray has slope $\tan 2\alpha$. Since the slope of the tangent to the parabola at P is equal to $\tan\alpha = -\frac{t}{2}$, the equation of the reflected ray is given by

$$y - 9 + \frac{t^2}{4} = -\frac{4t}{4 - t^2} \cdot (x - t) \tag{20}$$

The x-coordinate of the point of intersection of the reflected ray with a fixed line $y = u$ satisfies:

$$u - 9 + \frac{t^2}{4} = -\frac{4t}{4 - t^2} \cdot (x - t) \tag{21}$$

We calculate the minimal value of this x for varying t, by differentiating (21) with respect to t and assuming that $\frac{\mathrm{d}x}{\mathrm{d}t} = 0$:

$$\frac{t}{2} = -\frac{4(4 + t^2)}{(4 - t^2)^2}(x - t) - \frac{4t}{4 - t^2} \cdot (-1) \quad \Leftrightarrow \quad x = 3\frac{t}{2} - \frac{t^3}{8}$$

Inserting in the equation containing u gives us the relation between t and u:

$$u = 9 - 3\frac{t^2}{4}$$

This leads to a system of parametric equations for the caustic:

$$\begin{cases} x = 3\frac{t}{2} - \frac{t^3}{8} \\ y = 9 - 3\frac{t^2}{4} \end{cases} \quad \Leftrightarrow \quad \begin{cases} x = \frac{t}{2} \cdot (3 - \frac{t^2}{4}) \\ y = 3(3 - \frac{t^2}{4}) \end{cases}$$

It is now easy to eliminate the parameter t. As you can see, $t = \frac{6x}{y}$. Inserting into the equation for y gives us the equation of Tschirnhausen's cubic.

7 Implementation

To ease the implementation work and because raw LaTeX code is difficult to read on itself, We took the liberty of not providing this section with speech chunks (except for this introduction text).

7.1 Design principles

SpeLaTeX has been developed using the following main targets in mind. Some of them are common sense design principles, some of them are specific for this application.

- minimal effort in preparing a LaTeX manuscript for use with SpeLaTeX

- maximal compatibility with existing LaTeX packages
- no (or minimal) compromise mathematical reading capabilities for mathematical constructs
- user extensible audio preprocessor
- minimal use of processing power for text to speech conversion

7.2 Auxiliary Packages

The SpeLaTeX package uses some basic auxiliary packages to make life easy.

```
\RequirePackage{expl3}
\RequirePackage{hyperref}
\RequirePackage{xcolor}
\RequirePackage{ifthen}
\RequirePackage{fancyvrb}
\RequirePackage{newfile}
\RequirePackage{rotating}
\RequirePackage{babel}
\hypersetup{backref=true,
            breaklinks=true,
            colorlinks=true,
            citecolor=black,
            filecolor=black,
            hyperindex=true,
            linkcolor=black,
            pageanchor=true,
            pagebackref=true,
            pagecolor=black,
            pdfpagemode=UseOutlines,
            bookmarksopen=true,
            urlcolor=black}
\RequirePackage{kvoptions}
\RequirePackage{xkeyval}
\RequirePackage{marginnote}
```

7.3 Options

```
\SetupKeyvalOptions{
  family=spel,
  prefix=spel@
}
\DeclareStringOption[ogg]{format}
\DeclareBoolOption[false]{disabled}
\DeclareBoolOption[false]{extramath}
\DeclareBoolOption[false]{propermath}
\ProcessKeyvalOptions*
```

7.4 Logos

Vanity is everything, so let's make some logoware.

`\spelatex` This is the official SpeLaTeX logo.

```
\DeclareRobustCommand{\spelatex}{S\kern-0.3ex\raisebox{-0.1ex}{\rotatebox{-15}{p}}\kern-0.25ex\raisebox{0.1ex}{\rotatebox{10}{e}}\kern-0.1ex\LaTeX}
```

`\spelbox` This is the official SpeLbox logo.

```
\DeclareRobustCommand{\spelbox}{S\kern-0.3ex\raisebox{-0.1ex}{\rotatebox{-15}{p}}\kern-0.25ex\raisebox{0.1ex}{\rotatebox{10}{e}}\kern-0.1exLbo\raisebox{-0.2ex}{x}}
```

`\spelpl` This is the official `spel-wizard.pl` logo.

```
\DeclareRobustCommand{\spelpl}{\texttt{spel-wizard.pl}}
```

7.5 The speech stream

The basic structural elements of a document (title, chapters, sections, ...) are written to the speech index stream. This is a textfile that has the same base name as your LaTeX job and has extension `.spelidx`.

It is the index to the chunks of text that are written to the speech directory.

The `.spelidx` file requires postprocessing by the `spel-wizard.pl` script in order to obtain the required audio files.

The speech stream needs to be open before the preamble's title, author and date.

```
\newoutputstream{chunk}
\newoutputstream{spelidx}
\openoutputfile{\jobname.spelidx}{spelidx}
```

The stream needs to be closed upon termination of the document.

```
\AtEndDocument{
  \closeoutputstream{spelidx}%
}
```

To begin with, we write the standard locations for audio and chunk data to the `.spelidx` file.

```
\newcommand\audiodir{\jobname-spel}
\newcommand\chunkdir{\jobname-spel}
\addtostream{spelidx}{format|\spel@format}
\addtostream{spelidx}{audiodir|\audiodir}
\addtostream{spelidx}{chunkdir|\chunkdir}
```

To ease writing to the speech index stream, we define a `\spelidxwrite` function to take care of appropriate formatting.

`\spel@idxwrite` This is an internal macro, used to write information to the `.spelidx` file and to a correspondig chunk file.

```
\ifspel@disabled\newcommand{\spel@idxwrite}[2]{}\else
```

```
\newcommand{\spel@idxwrite}[2]{%
  \typeout{spel: Generating #1 - #2}%
  \addtostream{spelidx}{#1|#2}%
}
\fi
```

To ease writing speech chunk, we define a `\spel@chunkwrite` function.

`\spel@chunkwrite` This is an internal macro, used to write information to the speech chunk files.

```
\ifspel@disabled\newcommand{\spel@chunkwrite}[2]{}\else
\newcommand{\spel@chunkwrite}[2]{%
  \openoutputfile{\audiodir/#1.tex}{chunk}%
  \addtostream{chunk}{#2}%
  \closeoutputstream{chunk}%
}
\fi
```

7.6 Create missing counters

As we need to be able to fully identify every speech chunk, we need to provide some missing counters for the starred versions of the sectioning commands.

`spel@spart` counter

```
\newcounter{spel@spart}
\renewcommand\thespel@spart{\@arabic\c@spel@spart}
\setcounter{spel@spart}{0}
```

`spel@schapter` counter

```
\ifx\c@chapter\@undefined
\else
\ifx\c@part\@undefined
\newcounter{spel@schapter}
\else
\newcounter{spel@schapter}[part]
\fi
\renewcommand\thespel@schapter{\@arabic\c@spel@schapter}
\setcounter{spel@schapter}{0}
\fi
```

`spel@ssect` counter

```
\ifx\c@chapter\@undefined
\newcounter{spel@ssect}
\else
\newcounter{spel@ssect}[chapter]
\fi
\renewcommand\thespel@ssect{\@arabic\c@spel@ssect}
\setcounter{spel@ssect}{0}
```

In addition, some elements that are not canonically numbered require a unique and monotonous numbering.

`spel@footnote` counter

```
\newcounter{spel@footnote}
\renewcommand\thespel@footnote{\@arabic\c@spel@footnote}
\setcounter{spel@footnote}{0}
```

`spel@chunk` counter

```
\newcounter{spel@chunk}[subparagraph]
\renewcommand\thespel@chunk{\@arabic\c@spel@chunk}
\setcounter{spel@chunk}{0}
```

7.7 Setting up the language

We want to make sure that babel communicates the switching of langauges to spel, such that it can take not of it. This allows the spel engine to select an appropriate language-capable voice when generating the spoken text.

```
\AddBabelHook{informspel}{write}{\spel@idxwrite{language}{\languagename}}
\EnableBabelHook{informspel}
```

7.8 Generating speech chunks — implicitly

7.8.1 Auxiliary macros

We define a macro to generate wrappers for single-line text elements. The `\spel@registerelement` macro does the job. The user can even use the macro for his own custom single-line text elements (e.g., for a subtitle, a version string).

`\spel@registerelement` generic macro to register single-line text elements

```
\ifspel@disabled\newcommand{\spel@registerelement}[1]{}\else
\newcommand{\spel@registerelement}[1]{%
  \expandafter\let\csname spel@@#1\expandafter\endcsname\csname #1\endcsname
  \expandafter\gdef\csname #1\endcsname##1{%
    \spel@chunkwrite{#1}{##1}
    \csname spel@@#1\endcsname{\href{run:\audiodir/#1.\spel@format}{##1}}
  }
  \expandafter\AtBeginDocument{
    \spel@idxwrite{#1}{#1}
  }
}
\fi
```

7.8.2 Title elements

By redefining the title elements, `\title`, `\author` and `\date` we avoid having to chunk them.

Using this macro, we can easily take care of all title-like elements, using:

```
\spel@registerelement{title}
\spel@registerelement{date}
\spel@registerelement{author}
```

7.8.3 Table of contents

```
\ifspel@disabled\else
\let\spel@@addcontentsline\addcontentsline
\renewcommand\addcontentsline[3]{%
  \let\spel@@href\href%
  \renewcommand\href[2]{#2}%
  \spel@@addcontentsline{#1}{#2}{#3}%
  \let\href\spel@@href%
}
\providecommand{\tableofcontents}{}
\renewcommand\tableofcontents{%
    \if@twocolumn
      \@restonecoltrue\onecolumn
    \else
      \@restonecolfalse
    \fi
    \@ifclassloaded{article}{\section*{\contentsname}}{\chapter*{\contentsname}}
        \@mkboth{%
           \MakeUppercase\contentsname}{\MakeUppercase\contentsname}%
    \@starttoc{toc}%
    \if@restonecol\twocolumn\fi
    }
\fi
```

7.8.4 Sectioning commands

`\@part` This is a simple wrapper around the regular `\@part` macro.

```
\ifspel@disabled\else
\let\spel@@part\@part
\def\@part[#1]#2{%
  \setcounter{spel@chunk}{0}% need this because counter resetting fails
  \spel@@part[#1]{\href{run:\audiodir/\spel@@optpart.\spel@format}{#2}}%
  \spel@idxwrite{part \thepart}{\spel@@optpart}%
  \spel@chunkwrite{\spel@@optpart}{#2}%
}
\fi
```

`\@spart` This is a simple wrapper around the regular `\@spart` macro.

```
\ifspel@disabled\else
```

```
\let\spel@@spart\@spart
\def\@spart#1{%
  \stepcounter{spel@spart}%
  \setcounter{spel@chunk}{0}% need this because counter resetting fails
  \spel@@spart{%
    \href{run:\audiodir/\spel@@optpart star-\thespel@spart.\spel@format}{#1}}%
  \spel@idxwrite{part}{\spel@@optpart star-\thespel@spart}%
  \spel@chunkwrite{\spel@@optpart star-\thespel@spart}{#1}%
}
\fi
```

`\@chapter` This is a simple wrapper around the regular `\@chapter` macro. It is defined conditionally on the existence of the `\chapter` macro.

```
\ifspel@disabled\else
\ifx\chapter\@undefined\else
\let\spel@@chapter\@chapter
\def\@chapter[#1]#2{%
  \setcounter{spel@chunk}{0}% need this because counter resetting fails
  \spel@@chapter[#1]{%
    \href{run:\audiodir/\spel@@optpart\thechapter.\spel@format}{#2}}%
  \spel@idxwrite{chapter \thechapter}{\spel@@optpart\thechapter}%
  \spel@chunkwrite{\spel@@optpart\thechapter}{#2}%
}
\fi
\fi
```

`\@schapter` This is a simple wrapper around the regular `\@schapter` macro. It is defined condionally on the existence of the `\schapter` macro.

```
\ifspel@disabled\else
\ifx\schapter\@undefined\else
\let\spel@@schapter\@schapter
\def\@schapter#1{%
  \stepcounter{spel@schapter}%
  \setcounter{spel@chunk}{0}% need this because counter resetting fails
  \spel@@schapter{%
    \href{run:\audiodir/\spel@@optpart star-\thespel@schapter.\spel@format}{#1}}%
  \spel@idxwrite{chapter}{\spel@@optpart star-\thespel@schapter}%
  \spel@chunkwrite{\spel@@optpart star-\thespel@schapter}{#1}%
}
\fi
\fi
```

`\@sect` This is a simple wrapper around the regular `\@sect` macro.

```
\ifspel@disabled\else
\let\spel@@sect\@sect
\def\@sect#1#2#3#4#5#6[#7]#8{%
  % correct default tex behavior
  \ifnum #2>\c@secnumdepth%
  \stepcounter{#1}%
  \fi%
```

```
  \setcounter{spel@chunk}{0}% need this because counter resetting fails
  \spel@@sect{#1}{#2}{#3}{#4}{#5}{#6}[#7]{%
    \href{run:\audiodir/\spel@@optpart\thesubparagraph.\spel@format}{#8}\hfill%
    \href{run:\audiodir/\spel@@optpart\thesubparagraph.m3u}{\textcolor{black!25}{$\triangleri
  \def\spel@@label{\ifnum #2>\c@secnumdepth\else#1 \csname the#1\endcsname\fi}
  \spel@idxwrite{\spel@@label}{\spel@@optpart\thesubparagraph}%
  \spel@chunkwrite{\spel@@optpart\thesubparagraph}{#8}%
}
\fi
```

`\@sect` This is a simple wrapper around the regular `\@ssect` macro.

```
\ifspel@disabled\else
\let\spel@@ssect\@ssect
\def\@ssect#1#2#3#4#5{%
  \stepcounter{spel@ssect}%
  %\setcounter{spel@chunk}{0}% need this because counter resetting fails
  \spel@@ssect{#1}{#2}{#3}{#4}{%
    \href{run:\audiodir/\spel@@optpart\thesubparagraph-star-\thespel@ssect.\spel@format}%
         {#5}}%
  \spel@idxwrite{section}{\spel@@optpart\thesubparagraph-star-\thespel@ssect}%
  \spel@chunkwrite{\spel@@optpart\thesubparagraph-star-\thespel@ssect}{#5}%
}
\fi
```

7.8.5 Notes

`\@footnotetext` This is a simple wrapper around the regular `\$footnotetext` macro. We use a `spelfootnote` counter to keep track of the individual footnotes.

```
\ifspel@disabled\else
\let\spel@@fntext\@footnotetext
\long\def\@footnotetext#1{%
  \stepcounter{spel@footnote}%
  \settowidth\spel@mptboxwidth{\usebox\spel@mptbox}%
  \spel@@fntext{%
    \hspace*{-\spel@mptboxwidth}\href{run:\audiodir/footnote-\thespel@footnote.\spel@format}{
  \spel@idxwrite{footnote}{footnote-\thespel@footnote}%
  \spel@chunkwrite{footnote-\thespel@footnote}{#1}%
}
\fi
```

7.8.6 Itemizations/Enumerations

`\spelitem` This macro is to be used inside an enumerate, itemize, description environment to automatically cause the generation of a speech chunk.

```
\ifspel@disabled\newcommand{\spelitem}{\item}\else
\newcommand{\spelitem}{%
  \@ifnextchar[{\spelitem@opt}{\spelitem@intone}
}
\fi
```

This macro uses a number of auxiliary macros.

`\spelitem@opt` This is an internal macro intended to deal with the `\item`'s options.

```
\def\spelitem@opt[#1]{\spelitem@inttwo{#1}}
```

`\spelitem@opt` This is an internal macro intended to deal with an `\spelitem` without options.

```
\def\spelitem@intone#1{%
  \stepcounter{spel@chunk}%
  \settowidth\spel@mptboxwidth{\usebox\spel@mptbox}%
  \spel@idxwrite{item}{\spel@@optpart\thesubparagraph-\thespel@chunk}%
  \spel@chunkwrite{\spel@@optpart\thesubparagraph-\thespel@chunk}{#1}%
  \item \hspace*{-\spel@mptboxwidth}\href{run:\audiodir/\spel@@optpart\thesubparagraph-
\thespel@chunk.\spel@format}{\usebox\spel@mptbox}#1}
```

`\spelitem@inttwo` This is an internal macro intended to deal with an `\spelitem` with options.

```
\def\spelitem@inttwo#1#2{%
  \stepcounter{spel@chunk}%
  \settowidth\spel@mptboxwidth{\usebox\spel@mptbox}%
  \spel@idxwrite{item}{\spel@@optpart\thesubparagraph-\thespel@chunk}%
  \spel@chunkwrite{\spel@@optpart\thesubparagraph-\thespel@chunk}{#1 . #2}%
\item[#1] \hspace*{-\spel@mptboxwidth}\href{run:\audiodir/\spel@@optpart\thesubparagraph-
\thespel@chunk.\spel@format}{\usebox\spel@mptbox}#2}
```

`\caption` This is a redefinition of the `\caption` macro such that it becomes alive.

```
\ifspel@disabled\else
\let\spel@@caption\caption
\renewcommand\caption[2][]{%
  \stepcounter{spel@chunk}%
  \spel@idxwrite{caption}{\spel@@optpart\thesubparagraph-\thespel@chunk}%
  \spel@chunkwrite{\spel@@optpart\thesubparagraph-\thespel@chunk}{#2}%
  \spel@@caption[#1]{\protect\href{run:\audiodir/\spel@@optpart\thesubparagraph-
\thespel@chunk.\spel@format}{#2}}
}
\fi
```

7.9 Generating speech chunks — explicitly

7.9.1 Spel chunks to be parsed by `spel-wizard.pl`

`spelchunk` (*env.*) The `spelchunk` environment is used to define explicit speech chunks.

```
\newlength\spel@mptboxwidth
\newsavebox\spel@mptbox
\savebox\spel@mptbox{\textcolor{black!25}{$\qquad$}}
\newif\ifspel@chunkarealink
\define@key{spelchunk}{arealink}[]{\spel@chunkarealinktrue}
\ifspel@disabled\def\spelchunk{}\else
\def\spelchunk{%
```

```
  \catcode`\^^M=\active%
  \stepcounter{spel@chunk}%
  \spel@idxwrite{chunk}{\spel@@optpart\thesubparagraph-\thespel@chunk}%
  \@ifnextchar[{\catcode`\^^M=5\spelchunk@opt}{\catcode`\^^M=5\spelchunk@int}}%
\fi
\ifspel@disabled\def\endspelchunk{}\else
\def\endspelchunk{%
  \end{VerbatimOut}%
  \catcode`\^^M=5\relax%
  \ifspel@chunkarealink%
  \href{run:\audiodir/\spel@@optpart\thesubparagraph-\thespel@chunk.\spel@format}{\input{./\c
\thespel@chunk}}%
  \else%
  \settowidth\spel@mptboxwidth{\usebox\spel@mptbox}%
  \hspace*{-\spel@mptboxwidth}\href{run:\audiodir/\spel@@optpart\thesubparagraph-
\thespel@chunk.\spel@format}{\usebox\spel@mptbox}\input{./\chunkdir/\spel@@optpart\thesubpara
\thespel@chunk}%
  \fi%
  \spel@chunkarealinkfalse%
}%
\fi
```

The environment above checks if it is called with optional arguments or not.

`\spelchunk@opt` This is macro that deals with the optional arguments of the `spelchunk` envronment.

```
\def\spelchunk@opt[#1]{\setkeys{spelchunk}{#1}\spelchunk@int}
```

`\spelchunk@int` This is an internal macro to start the `VerbatimOut` environment embedded in the `spelchunk` environment.

```
\def\spelchunk@int{%
  \VerbatimEnvironment
  \begin{VerbatimOut}{\chunkdir/\spel@@optpart\thesubparagraph-\thespel@chunk.tex}}
```

7.9.2 Explicit spelchunks

`spelchunkad` (*env.*) The `spelchunkad` environment is used to override a previous speech chunk. In this way you can provide your own text.

```
\def\spelchunkad{%
  \catcode`\^^M=\active
  \@ifnextchar[{\catcode`\^^M=5\spelchunk@opt}{\catcode`\^^M=5\spelchunk@int}}
\def\endspelchunkad{%
  \end{VerbatimOut}
  \catcode`\^^M=5\relax
}

\AtBeginDocument{
  \newcommand\spel@@optpart{}
}
```

7.10 Helping the wizard to read our chunks

7.10.1 Listing macros that are to be preprocessed

Some LaTeX or TeXcommands are only for layout purposes and are totally not content related. They do not contribute to what must be read. On the contrary, they make it hard for the `spel-wizard.pl` parser to convert the texts flawlessly to what can be read by the text-to-speech engines. Examples of these layout-only commands are `\sf`, `\it`, `\tt`, `\bf` and `\displaystyle` that are to be discarded, but also macro's like e.g. `\fbox` for which only the content is to be retained.

As you might also make your own macros that are pure typesetting oriented, it makes sense to provide a macro that registers them as pure type-setting macros and use that macro to cover the examples mentioned above.

`\spelmacpp`

```
\ExplSyntaxOn
\NewDocumentCommand{\spelmacpp}{moom}
{
  \addtostream{spelidx}{macpp|#1|#2|#3|#4}
}
\ExplSyntaxOff
```

Now let's register some standard macros that are to be ignored.

```
\spelmacpp{sf}{}
\spelmacpp{it}{}
\spelmacpp{tt}{}
\spelmacpp{bf}{}
\spelmacpp{HUGE}{}
\spelmacpp{Huge}{}
\spelmacpp{huge}{}
\spelmacpp{LARGE}{}
\spelmacpp{Large}{}
\spelmacpp{large}{}
\spelmacpp{normalsize}{}
\spelmacpp{small}{}
\spelmacpp{footnotesize}{}
\spelmacpp{scriptsize}{}
\spelmacpp{tiny}{}
\spelmacpp{minuscule}{}
\spelmacpp{textsf}[1]{keep}
\spelmacpp{textit}[1]{keep}
\spelmacpp{texttt}[1]{keep}
\spelmacpp{textbf}[1]{keep}
\spelmacpp{quad}{}
\spelmacpp{qquad}{}
\spelmacpp{displaystyle}{}
\spelmacpp{relax}{}
\spelmacpp{strut}{}
\spelmacpp{mathstrut}{}
\spelmacpp{label}[1]{}
```

And let's register a macro for which only the contents is to be preserved:

```
\spelmacpp{fbox}[1]{keep}
```

7.10.2 Listing environments that are to be ignored

Some LaTeX or TeXenvironments are only for layout purposes and are totally not content related. They do not contribute to what must be read. On the contrary, they make it hard for the `spel-wizard.pl` parser to convert the texts flawlessly to what can be read by the text-to-speech engines. An examples of such a layout-only environment is the `center` environment.

As you might also make your own environments that are pure typesetting oriented, it makes sense to provide a macro that registers them as pure type-setting macros and use that macro to cover the examples mentioned above.

`\spelenvpp`

```
\ExplSyntaxOn
\NewDocumentCommand{\spelenvpp}{moom}
{
  \addtostream{spelidx}{envpp|#1|#2|#3|#4}
}
\ExplSyntaxOff
```

Now let's register some standard macros that are to be ignored:

```
\spelenvpp{center}{keep}
```

7.10.3 Audio descriptions for typesetting macros

`\spelmacad` This macro allows specifying how to treat macros (with arguments) that appear in the chunks to read out loud. The arguments are in order:

1. (mandatory) name of the macro (without leading backslash)
2. (optional) number of arguments of the macro
3. (optional) default for optional (first) argument
4. (mandatory) text to read (with macro parameters in them) You can use the special syntax `@{i18n(keyword,#1,#2)}` to trigger a call to the internationalization (i18n) features built in the `spel-wizard.pl` script. This will help to read your commands in an appropriate way. If you miss some features in the i18n list of `spel-wizard.pl`, please contact the author to help you out. If you are fluent in Perl, you might also want to change the i18n list of `spel-wizard.pl` yourself. It's not that hard.

```
\ExplSyntaxOn
\NewDocumentCommand{\spelmacad}{moom}
{
```

```
  \addtostream{spelidx}{macad|#1|#2|#3|#4}
}
\ExplSyntaxOff
```

We immediately provide some standard constructs, which are to be ignored:

```
\spelmacad{spelatex}{spee-lay-tech}
\spelmacad{spelbox}{spel-box}
\spelmacad{spelpl}{spel wizzard dot pl}
\spelmacad{LaTeX}{lay-tech}
\spelmacad{TeX}{tech}
\spelmacad{textsf}[1]{#1}
\spelmacad{texttt}[1]{#1}
\spelmacad{textit}[1]{#1}
\spelmacad{emph}[1]{#1}
\spelmacad{underline}[1]{#1}
\spelmacad{mbox}[1]{#1}
\spelmacad{text}[1]{#1}
\spelmacad{nobreakspace}{#1}
\spelmacad{textasciitilde}[1]{ }
\spelmacad{textbackslash}{backslash}
\spelmacad{footnote}[1]{}
\spelmacad{pm}{@{i18n(plusminus)}}
\spelmacad{ldots}{...}
```

Some more that don't seem ignorable - and they are not indeed - they are treated differently by `spel-wizard.pl`. However, by registering them here, `spel-wizard.pl` knows there signature:

```
\spelmacad{cite}[1]{}
\spelmacad{ref}[1]{}
\spelmacad{pageref}[1]{}
```

7.10.4 Audio descriptions for typesetting environments

`\spelenvad` This macro allows specifying how to treat environments (with arguments) that appear in the chunks to read out loud. The arguments are in order:

1. (mandatory) name of the macro (without leading backslash)
2. (optional) number of arguments of the macro
3. (optional) default for optional (first) argument
4. (mandatory) text to read (with macro parameters in them)

```
\ExplSyntaxOn
\NewDocumentCommand{\spelenvad}{moomm}
{
  \addtostream{spelidx}{envad|#1|#2|#3|#4|#5}
}
\ExplSyntaxOff
```

We immediately provide some standard constructs, which are to be ignored:

```
\spelenvad{center}{}{}
```

7.11 Extra math commands

The commands are only loaded if the package option `extramath` is provided:

```
\ifspel@extramath
```

`\setenum` This macro typesets a set defined by enumeration:

```
\DeclareRobustCommand{\setenum}[1]{\left\{#1\right\}}
\spelmacad{setenum}[1]{@{i18n(Setenum,#1)}}
```

`\setdesc` This macro typesets a set defined by description:

```
\DeclareRobustCommand{\setdesc}[1]{\left\{#1\right\}}
\spelmacad{setdesc}[1]{@{i18n(Setdesc,#1)}}
```

Note that these two macro's are identical! However, the fact that they have a different name is of great value to `spel-wizard.pl`.

The conditional loading ends here:

```
\fi
```

8 TODO

As long as there are things on my todo list, We have a reason to live.

- provide enable/disable switch to disable certain ranges in text, e.g. the implementation range in this document
- enable bibliography and citation stuff

References

[1] NVDA from NV Access, empowering lives through non-visual access to technology `https://www.nvaccess.org` online, accessed in June 2024.

[2] SprintPlus, helping people with dyslexia `https://www.sprintplus.be/en` online, accessed in June 2024.

[3] Adobe Reader, a PDF reader from Adobe. `https://get.adobe.com/` online, accessed in May 2024.

[4] TagPDF - Tools for experimenting with tagging using pdfLaTeX and LuaLaTeX. https://ctan.org/pkg/tagpdf online, accessed in June 2024.

[5] Wine - Wine Is Not an Emulator - running windows applications on POSIX-compliant systems. https://www.winehq.org online, accessed in June 2024.

[6] Org Mode — your life in plain text. https://orgmode.org/. online, accessed in May 2024.

[7] The Emacs An extensible, customizable, free text editor — and more. https://www.gnu.org/software/emacs/. online, accessed in May 2024.

[8] Org Mode SpeLaTeX Exporter — SpeLaTeX made easy. https://www.melpa.org/#/ox-spelatex (not yet online)

[9] SpeL — Speech-enabled LaTeX. https://ctan.org/pkg/spel online, accessed in June 2024.

[10] SpeL::Wizard — Incantating LaTeX into natural lanuage https://metacpan.org/pod/SpeL::Wizard online, accessed in June 2024.

[11] The Festival TTS-program. http://www.cstr.ed.ac.uk/projects/festival. online, accessed in May 2024.

[12] The Balabolka TTS-program. http://www.cross-plus-a.com/balabolka.htm. online, accessed in May 2024.

[13] FreeTTS — A speech synthesizer in Java. https://freetts.sourceforge.io/docs/index.php. online, accessed in May 2024.

[14] Amazon Polly — An online text-to-speech engine. https://aws.amazon.com/polly online, accessed in May 2024.

[15] The Comprehensive TeX Archive Network. http://www.ctan.org. online, accessed in May 2024.

[16] The Comprehensive Perl Archive Network. http://www.cpan.org. online, accessed in May 2024.

[17] xpdf, a simple and very fast PDF reader on GNU/Linux. http://www.xpdfreader.com/. online, accessed in May 2024.

[18] evince, a PDF reader, part of the Gnome environment. https://help.gnome.org/users/evince/stable/. online, accessed in May 2024.

[19] okular, a PDF reader, part of the KDE environment. https://okular.kde.org. online, accessed in May 2024.

[20] PDF XChange Viewer, a PDF reader from Tracker Software. https://www.pdf-xchange.com/ online, accessed in May 2024.

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

T

U

V