Writing my thesis with TEX

A report from the battlefields

Sharing your TEX-setup feels a little like airing your dirty laundry in public. The projects I know are often a mess of haphazardly cobbled up solutions and last-minute changes, duct-taped together with a bunch of macros copy-pasted from random pages on the internet.

Nevertheless (or possibly precisely because of this), sharing these setups and project structures can help other — especially more novice — users to streamline their workflow and lower the threshold to customize their own system. At least, write-ups like these have helped me.[1]

I will follow with a general overview of the setup that I used to write my thesis last year, with an audience in mind that would include myself before I started this thesis. It did get me that sought-after diploma, so you might try your risk with something similar.

First warning: always remember that content goes above style. Don't let the pleasure derived from your TEX-journeys obscure the thing that matters: that which you are trying to convey to the reader. Not even the best design or most comfortable workflow can mend a broken text — or a lack of text, for that matter.

Second warning: I'm not a very capable TEX-user by any means. If anything in the following seems old-fashioned, clumsy or plain wrong: you're probably right. Blame it on a routine built up from anonymous internet advice coming from several different decades and from questionable sources. (You could also take it as the indication of the TEX-proficiency level of an average user — which suggests that better documentation of currently available options is more useful than adding even more complex features to them.)

Background

The thesis I wrote was the conclusion of my bachelor in mathematics, which I followed at the University of Amsterdam (UvA). In the second year of the program, the course 'Computeralgebra / LATEX' is taught as an introduction to typesetting mathematics with LATEX, combined with some programming in Mathematica. Students are then expected to be sufficiently comfortable in LATEX to use it for research projects, an occasional homework assignment, and their thesis (this expectation is met to a certain degree). Also, presentations are expected to be prepared with `beamer`.

For the thesis, we are supplied with a class file with some basic scaffolding: a title page with institute names, supervisors and the logo of the university, and a place to put our abstract.[2] The rest (including design choices) is left to the students.

My thesis was in the area of mathematical logic, specifically set theory. This requires some extra funny symbols, but my setup should work for any scientific thesis or larger project. For an idea of the size: in the end, my thesis was 42 pages long, included 6 chapters and some 10.700 words (according to `texcount`).

Editor

Let's start with the main interface — the editor. I'm using Sublime Text 3 on a MacBook Pro. In the course mentioned earlier, Mac users were advised to use TeXShop since it came bundled with the MacTeX-distribution. I used it for a while, but never liked its design and default settings (it uses a proportional width font for code, and no syntax coloring). It's possible to tweak this to a certain extent, but I already used Sublime Text for other programming tasks, so it was more natural to switch to that.

The Sublime Text-plugin `LaTeXTools`[3] offers various necessary functionalities. Most importantly, it adds a configurable build pipeline, so you can compile while staying within Sublime Text. It also installs syntax highlighting, autofill-options for a.o. citations and references, and command completion. With this enabled, editing becomes very intuitive and swift.

A PDF-previewer with auto-reload (so that the preview reloads every time the PDF is compiled) makes the 'write-compile-read'-cycle even smoother. Unfortunately, Preview.app (the default PDF-viewer on Mac) doesn't offer this functionality. I use the open source alternative Skim instead.[4]

The choice of your editor is usually a very personal one, and discussions about it usually get very heated. I do not claim that Sublime Text is the most suitable editor for everyone — far from it. But in general, picking a good editor and getting comfortable with it and setting it up

```
/project
 ├── img
 │   ├── figure.tikz
 │   ├── logo.png
 │   └── ...
 ├── tex
 │   ├── 1-introduction.tex
 │   ├── 1.1-state-of-the-art.tex
 │   ├── 1.2-research-goal.tex
 │   ├── 2-experiment-design.tex
 │   ├── 2.1-lab-settings.tex
 │   └── ...
 ├── .gitignore
 ├── project.bib
 ├── project.tex
 ├── project.sty
 └── style.sty
```

Figure 1. An outline of the directory structure. For the function of `.gitignore`, see section 'Version control'.

properly (learning the most useful shortcuts, installing the right packages, configuring proper file exclusion patterns so your workspace doesn't get cluttered) makes the whole process a lot smoother.

Project organization

At a certain point, keeping all your text in one file becomes unworkable. I break everything up in smaller files quite aggressively — usually, every section gets its own file. Then there is another file per chapter, which collects all its sections using `\input`. These are all organized in a directory called `tex`. Finally, a main file is kept in the root directory. Using `\includes`, it collects all the chapters in `/tex`. This file can then be compiled into a PDF.

Since I want to reuse design and package decisions in later projects, I separated them into two `.sty`-files, that are imported in the main `.tex`-file by `\usepackage`. One of them (`style.sty`) contains general customizations; the other (`project.sty`) contains project-specific choices. See fig. 1 for how this all comes together.

Version control

Although it seems this is a rare opinion among other students (who often use the free version of online LaTeX collaboration tools such as Overleaf — which miss any history tools), to me version control is indispensable. I use Git to save the complete history of my project. Using 'commits', I can add another state of my files in the history tree. This allows me to edit my texts quite ruthlessly, since I know that if I ever change my mind about a part or two, I can always revert it to a previous state, or pick out specific sentences (in practice, I rarely ever actually need this — knowing that the option is there is enough, apparently).

Git offers 'tags', with which you can annotate commits. I use it to mark the commits from which I compiled particular versions — such as the ones I discussed with my supervisor. It's also possible to use 'branches' — particular diversions from the main file history, which I used a couple of times to try out certain new ideas or formats, without messing up the main history. To give an idea: I used 73 commits, 8 tags and 2 branches (besides the 'master' branch).

If you use Git, make sure to properly configure your `.gitignore`-file, which has file and folder exclusion patterns that Git uses (so you don't save all intermediate `.aux` and `.log` files in your history). There is a TeX-specific list on Github that is a good starting point.[5]

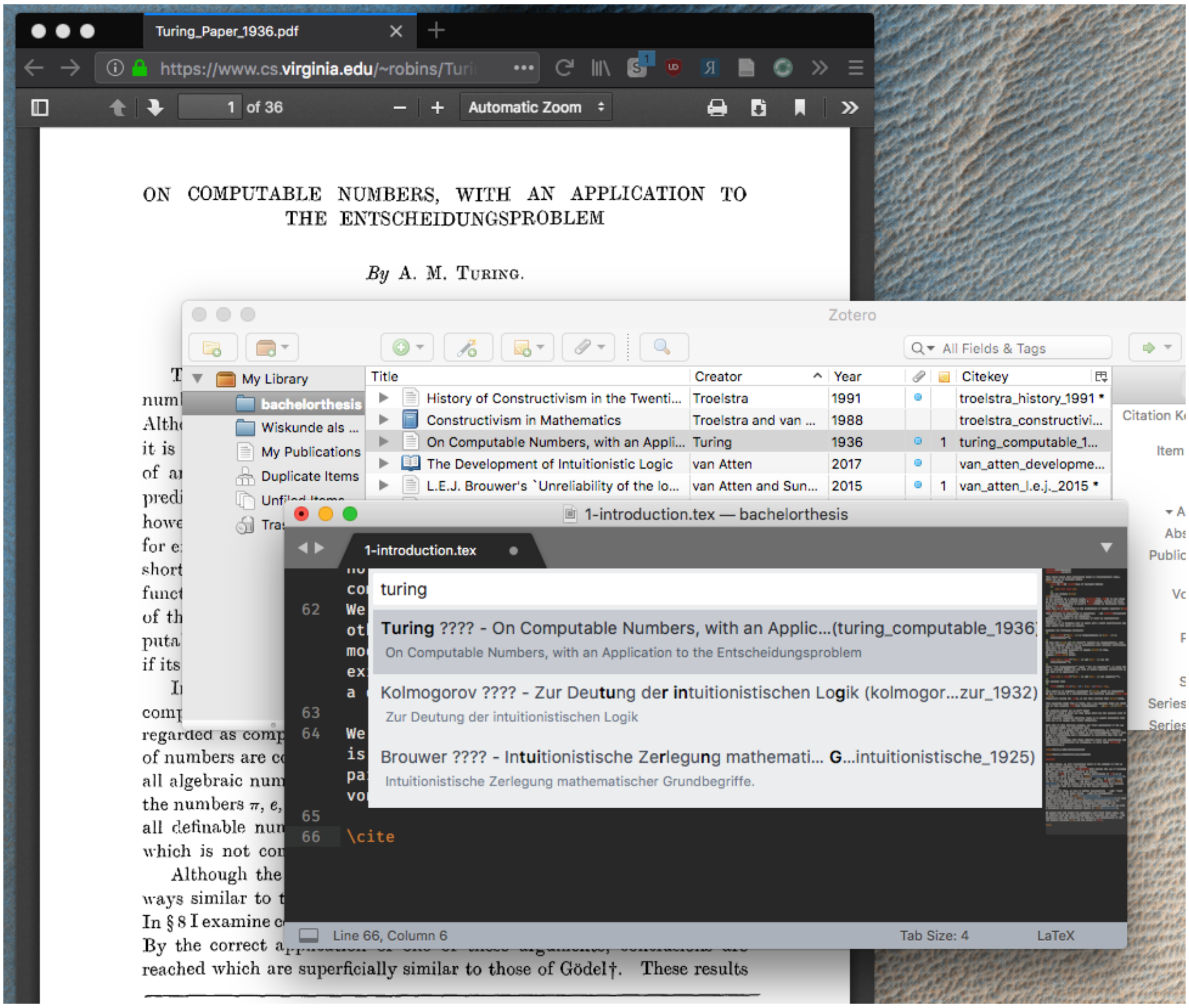
Speaking of Github: it is *the* place to synchronize your Git-repository with. If you are working on a project by yourself, it also functions as an external backup (although be sure that you have an extra copy or two saved on some external disks). And if you are collaborating on a project, Github can function as a central server to host your repository. All the participants then have a local copy of the repository, in which they can edit. Afterwards, they send their edits (new commits) back to Github for others to see.

Bibliography

Biblatex seemed the most modern of all options for managing citations, so that was my choice. It comes closest to working 'out of the box'.

One of the most important parts in my workflow is managing the references. You could edit the `.bib`-file directly, but that is quite error-prone and time-consuming. A solution which worked very well for me, is to use the open source Zotero.[6] It is a capable reference manager, and together with the plugin 'Better Bibtex'[7] you can easily exportcitations into a `.bib`-file and keep it updated automatically. If you use the browser plugin 'Zotero Connector' on top of that, you get a workflow that is so seamless it feels slightly magical (see fig. 2):

1. find a reference online
2. import it to Zotero with one click on the Zotero Connector (this automatically saves the PDF of the article, and fills in metadata — usually quite accurately)
3. Zotero automatically updates the `.bib`-file
4. Sublime Text recognizes the new reference, and it is immediately available in the autofill suggestions



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

Title	Creator	Year			Citekey
History of Constructivism in the Twenti...	Troelstra	1991	●		troelstra_history_1991 *
Constructivism in Mathematics	Troelstra and van ...	1988			troelstra_constructivi...
On Computable Numbers, with an Appli...	Turing	1936	●	1	turing_computable_1...
The Development of Intuitionistic Logic	van Atten	2017	●		van_atten_developme...
L.E.J. Brouwer's `Unreliability of the lo...	van Atten and Sun...	2015	●	1	van_atten_l.e.j._2015 *

```
turing
Turing ???? - On Computable Numbers, with an Applic...(turing_computable_1936)
  On Computable Numbers, with an Application to the Entscheidungsproblem
Kolmogorov ???? - Zur Deutung der intuitionistischen Logik (kolmogor...zur_1932)
  Zur Deutung der intuitionistischen Logik
Brouwer ???? - Intuitionistische Zerlegung mathemati... G...intuitionistische_1925)
  Intuitionistische Zerlegung mathematischer Grundbegriffe.

\cite

Line 66, Column 6                Tab Size: 4        LaTeX
```

reached which are superficially similar to those of Gödel†. These results

Figure 2. The Zotero workflow: open article in browser; add to Zotero with Zotero Connector (the 'page'-icons in the top right of the browser); Sublime suggest references as soon as you type `\cite`. (Note that there is a bug where Sublime doesn't properly extract the year of the references from the `.bib`-file.)

Design

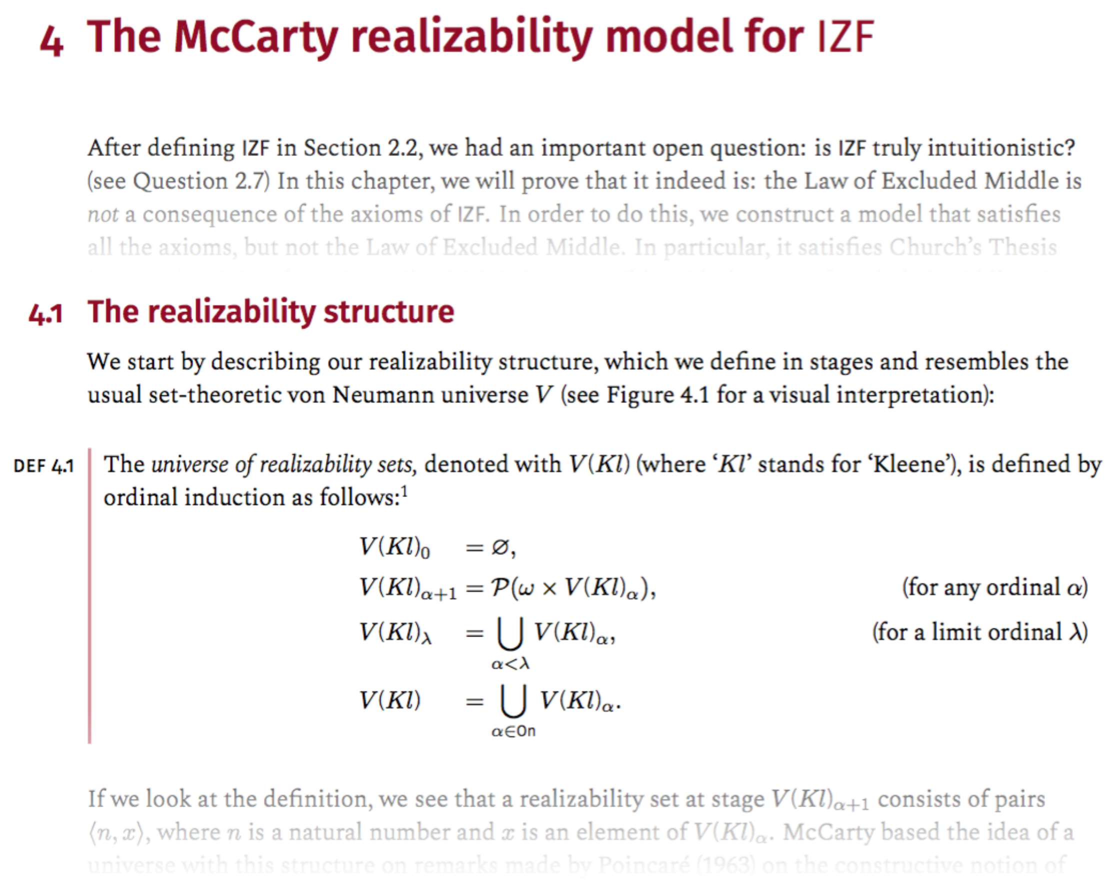
After staring at mathematics articles in the default LATEX-layout for so long, I got a bit bored with the default design and tried my hand at customizing it. The main part of this was choosing different fonts. My eye fell on the nice Spectral as a body font, and Fira Sans for titles and captions.[8] To easily use these fonts and customize settings, I switched to the XeTEX-engine. It is then as simple as

```
\setmainfont{Spectral Light}
```

to switch to the Light variant of Spectral as the body font (provided you have the fonts installed on your computer).

For mathematics, the `amsthm`-theorem environments are indispensable. To give them a more unique look, I gave them a line on the left side. This also separates them more clearly from the running text, leading to an interesting vertical typographical dynamic. I used the package `mdframed` to achieve this. This package is usually used to draw boxes around theorems, but you can also use it to draw a line on just one side. Once you have set this package up with the correct line widths and spacings, it is as simple as replacing `\newtheorem` with `\newmdtheoremenv`.

Since the margins of a document printed on A4 are quite wide, I decided to pull the numbering of theorems and titles into the side margins. This can be achieved with a quite clever hack with 'negative phantoms'. Usu-



4 The McCarty realizability model for IZF

After defining IZF in Section 2.2, we had an important open question: is IZF truly intuitionistic? (see Question 2.7) In this chapter, we will prove that it indeed is: the Law of Excluded Middle is *not* a consequence of the axioms of IZF. In order to do this, we construct a model that satisfies all the axioms, but not the Law of Excluded Middle. In particular, it satisfies Church's Thesis

4.1 The realizability structure

We start by describing our realizability structure, which we define in stages and resembles the usual set-theoretic von Neumann universe V (see Figure 4.1 for a visual interpretation):

DEF 4.1 The *universe of realizability sets,* denoted with $V(Kl)$ (where 'Kl' stands for 'Kleene'), is defined by ordinal induction as follows:[1]

$$\begin{aligned} V(Kl)_0 &= \varnothing, \\ V(Kl)_{\alpha+1} &= \mathcal{P}(\omega \times V(Kl)_\alpha), && \text{(for any ordinal } \alpha) \\ V(Kl)_\lambda &= \bigcup_{\alpha<\lambda} V(Kl)_\alpha, && \text{(for a limit ordinal } \lambda) \\ V(Kl) &= \bigcup_{\alpha\in\mathsf{On}} V(Kl)_\alpha. \end{aligned}$$

If we look at the definition, we see that a realizability set at stage $V(Kl)_{\alpha+1}$ consists of pairs $\langle n, x\rangle$, where n is a natural number and x is an element of $V(Kl)_\alpha$. McCarty based the idea of a universe with this structure on remarks made by Poincaré (1963) on the constructive notion of

Figure 3. An example of the design: note the custom fonts, the hanging section and definition numbers, and the line besides the definition (some content faded out for clarity).

ally, `\hphantom{text}` is used to insert a space exactly the width of 'text' if it would have been rendered at that place. This can be used to fine-tune the position of mathematical formulae, which helps in aligning them in more complex cases (where `align`-environments are not enough). This trick can be extended to use a 'negative phantom', which places a negative space at that spot. A common definition (don't ask me where I got it — it was floating somewhere on the internet) is the following:

```
\newcommand{\nhphantom}[1]%
  {\sbox0{#1}\hspace{-\the\wd0}}
```

If you then include this in the definition of your `theoremstyle`, you can pull the numbering of theorem-environments as far into the margins as you want — that is, precisely the length of the numbering itself.

A similar procedure works for chapter and section titles, with the aid of the package `titlesec`. I also used the extra options offered by this package to change the default spacing of titles.

For an example of how this all comes together, see fig. 3.

General remarks

Some more general wisdom that appeared to me during this process (which might be obvious to some, but very helpful to others[9])

- Use `\label`'s and `cleveref` for automatic referencing of elements like theorems, chapters and figures. It is as easy placing a label at e.g. a new chapter:

  ```
  \chapter{The best chapter}
  \label{ch:best-chapter}
  ```

  and then elsewhere in the text use this as:

  ```
  Earlier in \cref{ch:best-chapter},
    we discussed [...]
  ```

  Then, `cleveref` (the package which supplies the `\cref`-command) will automatically expand this to

> Earlier in chapter 1, we discussed [...]

If you also include the `hyperref`-package, it also changes 'chapter 1' into a link in the PDF that brings you to the right page.

Note that I add the `ch:`-prefix to all chapter-labels. I do something similar for theorems (`th:`), sections (`sc:`), and so on. It helps to remember what you actually referenced, when you find all your `\cref`'s scattered throughout the code.

- Make commands for every special symbol. I was working with set theory and logic, and had to use a lot of special notation. Because I made commands for every new symbol (e.g. `\newcommand{\proves}{\vdash}` for a specific proof-symbol), I could apply the symbols consistently in my text. And if I changed my mind about which symbol to use for which purpose, I only had to update one definition to change all occurrences.
- Add plenty of comments, at least in your `.sty`-files. Some definitions and commands can be quite obscure, and comments help to later remember why you added the code in the first place. For the same reason, when I find a solution for some problem online, I often include the link to the forum post or webpage where I found it.
- When writing, I often leave **todo** notes in the text to remind myself which parts I have to fill in later on. I use the aptly named package `todonotes` for this.
- If possible, consider a dual-screen monitor setup. This allows you to have a window with your editor on one screen, and a preview of the compiled PDF in the other.
- For any new text: write a draft on paper before you dive into your editor. Even though LATEX-syntax can seem relatively easy once you are used to it, it still slows down your creative process while writing. If I am not careful, I will waste time tweaking small typographic details, instead of using that time to write new text.

Notes

1. Specifically, I picked up advice from a couple of interesting discussions on workflows on the TEX Stackexchange (besides the huge number of solutions to more specific problems that I found there — it's a fantastic resource). For examples, check out the questions tagged 'workflow': `tex.stackexchange.com/questions/tagged/workflow`.
2. The class is available at `github.com/UvA-FNWI/uvamath`.
3. `github.com/SublimeText/LaTeXTools`
4. `skim-app.sourceforge.io`
5. Check the `gitignore`-repository at `github.com/github/gitignore`. You can also use this as an exclude list for your editor.
6. `zotero.org`
7. `github.com/retorquere/zotero-better-bibtex`
8. Both fonts are open source and available on Github (`github.com/productiontype/spectral` and `github.com/mozilla/Fira`)
9. relevant xkcd: `xkcd.com/1053`

Rens Baardman