



GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Master's Thesis

submitted in partial fulfillment of the requirements
for the course “Applied Computer Science”

My title

Jane Doe


Institute of Computer Science


Bachelor's and Master's Theses
at Georg-August-Universität Göttingen
ISSN 1612-6793


2025-November-13


Georg-August-Universität Göttingen
Institute of Computer Science

Goldschmidtstraße 7
37077 Göttingen
Germany

 +49 (551) 39-172000

 +49 (551) 39-14403

 office@informatik.uni-goettingen.de

 www.informatik.uni-goettingen.de

First Supervisor: My first supervisor

Second Supervisor: My second supervisor

I hereby declare that I have written this thesis independently without any help from others and without the use of documents or aids other than those stated. I have mentioned all used sources and cited them correctly according to established academic citation rules.

Göttingen, 2025-November-13

Abstract

Here comes the abstract...

Contents

1	First steps	1
2	How to structure your thesis	1
2.1	Introduction	1
2.2	Basics / Foundations	2
2.3	Analysis / Related Work	2
2.4	Design / Approach / Methods	2
2.5	Implementation / Case Study / Evaluation / Results / Discussion	2
2.6	Conclusion	3
3	Using typst and this template to write your thesis	3
3.1	Document structure	3
3.1.1	Formatting text	4
3.1.2	Lists, Tables, Formulas & more	5
3.1.3	Grids and Tables	5
3.1.4	Figures and including graphics	6
3.1.5	Citing	6
3.1.6	Footnotes	6
3.1.7	Math	6
3.1.8	Acronyms	6
3.1.9	PDF/A validation	7
A	Bibliography	8

1 First steps

1. Read the following chapters
2. Configure this template. Put your thesis data into the `main.typ`.
3. Decide on a chapter structure, comment out this file in `main.typ` and put your chapters in there.

2 How to structure your thesis

We first discuss the structure of your thesis and then give you an introduction into `typst` and this template.

Structuring your thesis can be confusing at first but it is actually easier than you think. No matter where you write your thesis, the structure is always very similar. Different disciplines just use different titles and have a slightly different focus. The kind of work you do, e.g. proof, case study, benchmark, also has an impact on wording and focus.

A thesis is generally written in present tense with a few exceptions. We typically present an evaluation and/or case study in past tense, referring to *what happened*. Similarly abstract, parts of the introduction and the conclusion may also other tenses. We can write a thesis in different persons. There is no consensus on what is best but the options are generally:

- the scientific “we”: Modern and increasingly common, also in papers. It might feel awkward to use if you write the thesis by yourself, but you get used to this rather quickly and then it is (subjectively) easy to read.
- “I”: This is the most accurate but often also feels rather personal. Some people think this is too personal and you should put the your thesis in the center instead of yourself.
- passive: Definitely puts the content in the center but also often seems uncommitted and like you don’t want to take responsibility for your work and your decisions. It is also often harder to understand.

You should probably discuss this with your supervisors.

2.1 Introduction

Every thesis has an introduction. The introduction typically has multiple uses:

- it introduces the reader to the topics of the thesis
- it *motivates* the work (not your personal motivation but rather why the topic is interesting/relevant)
- it defines *goals* and depending on your field also either *research questions* or *problems* that the work is attempting to solve
- it gives an overview over the *structure* of the document

We generally start by introducing the research area and why it is relevant. Then we explain the specific problem/challenge. If there are existing/common solutions in that area, we briefly mention them and especially their limitations. Following this, we propose our new solution/approach. Depending on your area, we also summarize how we evaluated this approach

2.2 Basics / Foundations

The basics chapter presents foundational knowledge on which the work is based. It also provides the opportunity to define terminology and introduce the reader to domain knowledge that might not be common knowledge.

This chapter is also sometimes named “Foundations”.

2.3 Analysis / Related Work

The analysis chapter typically does what it says: Problem analysis. This also includes presenting and discussion other research that is related to the thesis’ topics.

This chapter is sometimes called “Related work” and only presents / relates other research in the area of your topic. If it is called “Related work”, a part of the analysis is usually shifted into the next chapter or a dedicated chapter. You might also have a section on “Related Work” in you analysis chapter.

2.4 Design / Approach / Methods

This chapter presents the design of your research. This *can* include software design but the focus is on **research design** i.e. how your research solves problems / answers it’s research questions.

The design chapter is also often called “Approach” and presents how you approach reaching your goal / answering your research questions. It is also sometimes called “Methods” and discusses the research methodology of your work.

There are some differences in what this chapter aims to achieve. A case study design often discusses the *approach*, empirical studies generally discuss *methods* and a focus on problem solving or software design usually make the *design* the center of this chapter.

2.5 Implementation / Case Study / Evaluation / Results / Discussion

Depending on the focus in the previous chapter and the type of work, this chapter might actually be multiple chapters or contain multiple sections in the following order:

- **Implementation:** when software design or a reference implementation / proof of concept is important for your work.
 - Although *implementation* seems to imply a focus on software engineering, this is **only true in very few edge cases**. Depending on your area source code snippets may be required or highly undesirable!
- **Case Study:** then the work includes a case study.
 - This may present how your study was conducted be written in past tense (while all other chapters typically use present tense).
 - The core is: usage/application of your approach for answering the research

question.

- This may include presenting results and discussing their validity.
- **Evaluation / Results:** You may present your data / results here.

- *How* you evaluate is part of the previous chapter!
- **Discussion / Interpretation:** Discuss your results, critically analyze your approach and discuss limitations.

There is a lot of variation in here. An example: A case study where users use a software and answer a questionnaire about that. We

- might have just the “Case Study” chapter which tells the story of conducting the case study (incl. the actual procedure and the results). The discussion might be put into **Conclusion**.
- might have a separate chapter on the (software) **Implementation**, the **Case Study**, presenting **Results** and a **Discussion**.

Having the optimal structure here is not the most important thing but there should always be:

- a clear separation between design/approach and case study/implementation and
- a clear separation between presenting results and discussing/interpreting results

2.6 Conclusion

This chapter actually has three different functions:

- summarize the results
- finally answer your research questions or summarizing that, depending on the previous chapters
- discuss the validity and limitations of your work
- give an outlook (also called future work) how your work could/should be continued

It is possible to make separate sections or chapters for some of these.

3 Using typst and this template to write your thesis

The rest of this introduction chapter shows basic usage of some `latex` typst features that can be used in the document. Of course there is a lot more to typst than what can be covered here.

The document structure is defined by `main.typ` which applies the template in `template.typ`. When you apply the template, you should enter your data there. You also have the choice between two styles:

- `legacy` looks like the classic LaTeX template and is optimized for printing and binding your thesis.
- `modern` looks more modern and is optimized for on computer screens.

3.1 Document structure

Depending on your preference, you can distribute your chapters over multiple files or put everything into one file. Instead of one compact file with all chapters, you can e.g. create one file per chapter. A tip for later: The convention is that a file is not closed with a `#pagebreak()` but leaves this to the importing file. You can import files with a command like this (see `main.typ`):

```
#include "content/content.typ"
```

You can create headings (chapter, section, subsection, ...) like this:

```
= A Chapter
== A Section
=== A Subsection
==== A Subsubsection
```

These shorthand commands are equivalent to calling the `heading` function. By default all headings are numbered and outlined. The `heading` function takes arguments, e.g.:

- `depth`: e.g. 1 for chapter and 3 for subsection.
- `numbering`: How the heading should be numbered or `none` if numbering should be omitted for this heading.
- `outlined`: `true` or `false`, whether the heading should appear in the table of contents

The `heading` function also takes `content` (the heading's title) as a *positional* argument:

```
#heading(depth: 2, outlined: false, numbering: none, "Invisible Section")
```

Many typst functions take content and passing content along as a bare string is limited so there is a shorthand for providing functions with content:

```
#heading(depth: 2, outlined: false, numbering: none)[Invisible Section]
```

You can use the same syntax as in `.typ` files inside `[]`.

3.1.1 Formatting text

A common example for passing `content` into a function is centering:

This text appears centered on the page.

Typst comes with a simple markdown-like syntax for many common problems. Text can be made ****bold****, *`__italic__`* or ``monospaced``. Formatting text is generally exposed via typst's `text` function that **can do a lot**:

```
#text(weight: "thin", size: 12pt, font: "Comic Neue", fill: red)[can do a lot]
```

The function that generated monospaced content is called `raw` and can also display syntax-highlighted code. It also has a shorthand for multiline code-like content:

```
```c
#include<stdio.h>

int main() {
 printf("Hello World\n");
 return 0;
}
```
... becomes this code block when rendered:
#include<stdio.h>

int main() {
    printf("\nHello World\n\n");
    return 0;
}
```

The function `link` is used to generate clickable hyperlinks. By default links look like regular text and we could manually wrap each link in a `#underline[...]` but there is a better way. Typst has a flexible system for styling content.

3.1.1.1 Changing styling behavior

Typst has two kinds of rules:

- `set` rules can globally set some parameters, e.g. `#set align(center)` makes everything centered (until another alignment is set)
- `show` rules can change how something is rendered

In the case of styling hyperlinks we can use a `show` rule to make all hyperlinks underlined and italic:

```
#show link: l => text(style: "italic", underline(l))
```

From now on all Links are italic and underlined:

<https://en.wikipedia.org/wiki/Hyperlink>

3.1.2 Lists, Tables, Formulas & more

Typst has unordered list:

- | | |
|---------|---------|
| - One | • One |
| - Two | • Two |
| - Three | • Three |

And ordered lists:

- | | |
|---------|----------|
| + One | 1. One |
| + Two | 2. Two |
| + Three | 3. Three |

Of course lists can also be nested:

- | | |
|---------|----------|
| + One | 1. One |
| + alpha | 1. alpha |
| + beta | 2. beta |
| + Two | 2. Two |
| + Three | 3. Three |

3.1.3 Grids and Tables

Other forms of lists can often be done as a grid:

```
#grid(columns: 2, column-gutter: .5em, row-gutter: .5em, align: (right, left),
  [First:], [One],
  [Second:], [Two],
  [Third:], [Three],
)
```

| |
|--------------|
| First: One |
| Second: Two |
| Third: Three |

And if you have looked at the source code to this document, you have seen that `grid` can also be used to do much more e.g. figures with a two-column layout.

The `table` function act similarly to `grid` and can be used to create a simple table. It is usually best to wrap tables and images into a `figure` with a caption.

```
#figure(kind: table,
  caption: "A simple centered table with
some of my favorite numbers",
  align(center)[#table(
    columns: 4,
    align: (left,center,left,right,),
    [1], [2], [3], [4],
    [5], [6], [7], [8],
    [9], [10], [11], [12],
  )]
)
```

| | | | |
|---|----|----|----|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

Table 1: A simple centered table with some of my favorite numbers

3.1.4 Figures and including graphics

The `image` function can be used to include graphics in the document. The vector-based graphic formats i.e. `svg` (but unfortunately not yet `pdf`) are preferred, while the pixel-based `jpg` and `png` can also be included.

```
#figure(caption: "An outdated univervity logo",
  image("../images/goe_logo_small.jpg", width: 50pt)
)
<fig:old_uni_logo>
```



Figure 1: An outdated univervity logo

Relative units (`em`, `pt` are relative to text size and `%` is relative to the available space) are generally preferred for `width`. If you include labels like `<fig:old_uni_logo>` with your figures, you can easily reference e.g. as “[Figure 1](#)” by writing `@fig:old_uni_logo`.

3.1.5 Citing

One of the most important things in scientific work is the citing. Citing, incl. numbering and formatting is easy since everything is automatically managed by `latex typst`. For example (look into the source code):

John Doe^[1] proposes in his paper a new approach on XYZ.

Meyer et al.^[2, p. 100] suggest a new method to XYZ.

The `cite` function is used to reference to an item by its key but we also have shorthands for that. The references are stored in a separate `references.bib` file that contains all references in the `bibtex` format.

The bibliography is also automatically generated.

For good practices in scientific writing you can make use of the following documents:

- https://www.hochschulverband.de/fileadmin/redaktion/download/pdf/resolutionen/Gute_wiss_Praxis_Fakultaetentage.pdf
(only in german)
- <https://www.sub.uni-goettingen.de/en/learning-teaching/academic-work-tools-and-methods/academic-writing/>

3.1.6 Footnotes

Footnotes can be easily inserted using the `footnote` function¹.

```
#footnote[Make sure to use footnotes but only when necessary.]
```

3.1.7 Math

Typst can display math formulas. The syntax is similar but also a little different from latex:

<https://typst.app/docs/reference/math/>

3.1.8 Acronyms

There are packages for making acronyms easy and fun, e.g.:

<https://typst.app/universe/package/acrostiche/>

¹Make sure to use footnotes but only when necessary.

3.1.9 PDF/A validation

The thesis has to be in PDF/A format for archiving. This template automatically generates your file according to the PDF/A standard. It can be that this is violated by changing the document, for example by inserting non-compliant graphics. Please check, whether your final version of the document is PDF/A compliant, e.g., by using a freely available tool like *veraPDF*².

²<https://verapdf.org/home/>

A Bibliography

- [1] J. Doe, "A very interesting article," *Famous Journal*, vol. 2, no. 3, pp. 10–15, June 2013.
- [2] B. Meyer, S. Smith, and M. Green, *The Book*. PublishMe Ltd., 2014.