UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group Silly Walks

Bachelorarbeit

Submitted to the Silly Walks Research Group
in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Science

Your Fancy Title Goes Here

—

Your Title Can Even Span Multiple Rows of Text

by
Eugen Shulimov

Thesis Supervisor:
Prof. Dr. Square Jumper

Paderborn, July 1, 2024

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum	Unterschrift

Abstract. We present a full documentation of the Paderborn University Computer Science thesis template (UPB-CS-TT) and how to use it. This document also serves as a demonstrator to show what documents UPB-CS-TT produces. Have fun!

Contents

1 Introduction **1**
- 1.1 Getting Started . . . 1
- 1.2 Organisation: Directories . . . 2
  - 1.2.1 Main Directory . . . 2
  - 1.2.2 Chapters Directory . . . 3
  - 1.2.3 Figures Directory . . . 3
  - 1.2.4 Pretext Directory . . . 3
  - 1.2.5 Appendices Directory . . . 3

2 Hintergrund **5**
- 2.1 C-Präprozessor . . . 5
- 2.2 Variabilität Umsetzung mit C-Präprozessor . . . 6

3 The upb_cs_thesis Document Class **7**

4 Packages and Commands **13**
- 4.1 Class Packages . . . 13
- 4.2 Auxiliary Packages . . . 14
- 4.3 Custom Commands . . . 15

5 A Short Guide **17**
- 5.1 Style . . . 17
  - 5.1.1 Math Mode . . . 17
  - 5.1.2 Footnotes . . . 18
  - 5.1.3 Acronyms . . . 18
- 5.2 References . . . 18
- 5.3 Using Graphics . . . 19
- 5.4 Tables . . . 20
- 5.5 Bibliographies . . . 22

6 Makefiles **25**
- 6.1 Makefile Troubleshooting . . . 25
- 6.2 Main Makefile . . . 26
- 6.3 figures's Makefile . . . 26

Bibliography **28**

1 Introduction

This is the documentation of the Paderborn University Computer Science thesis template (UPB-CS-TT). In this chapter, you find everything you need to know to get started with your thesis — or at least the LaTeX side of it. The other chapters of this document provide you with further details of the template. Besides a full documentation of the document class file that UPB-CS-TT uses and the auxiliary files UPB-CS-TT supplies, we also provide you with a short guide on certain aspects of LaTeX, and particularly the LaTeX packages that UPB-CS-TT loads by default. However, it is assumed that you know some LaTeX already. If you are unfamiliar with LaTeX you should consult a proper introduction first, for example the LaTeX guide by Oetiker et al. [OPHS16].

Organization In the remainder of this chapter, we discuss how to get started with the UPB-CS-TT template, and what directory structure the template provides and assumes.

In Chapter 3, we provide a documentation of the *upb_cs_thesis* document class provided by the UPB-CS-TT template. The template loads a number of LaTeX packages by default. Chapter 4 provides an overview.

Chapter 5 provides a guide to both beginners and experts on aspects of packages that UPB-CS-TT loads by default for you to use. Particularly, the chapter discusses some style aspects of good writing, references, including bibliographic references, graphics, and tables. Finally, in Chapter 6, we provide a documentation of and troubleshooting guide for the *Makefiles* that come with this template.

1.1 Getting Started

In this section, we briefly discuss how to set up your thesis to work with the UPB-CS-TT template. This is a four step program:

1. choosing a language,
2. setting up the title page of your thesis,
3. writing its abstract, and
4. writing the thesis.

The UPB-CS-TT template can set up your thesis to use either English or German language options. By default, English is used. If you want to use German, edit the line of *thesis.tex* that reads `\documentclass[]{upb\_cs\_thesis}` and write *german* (lower case!) into the pair of brackets.

What you need to set up the title page are a *title*, your *own name*, the *type* of the thesis and the academic *degree* you aim for, the name of your *supervisor* and his or her *research group's name*, as well as a submission date. Setting up your thesis's title page, by editing the main file *thesis.tex*. In the section marked "your thesis title, [...]," you find certain commands to which you pass the information listed above. Pass your thesis's title to the `\thesis` command, and your own name to the `\author` command. The type of your thesis is passed to the `\thesistype` command. You can comment in or out any of the example types given in the file, or pass a string of your own choice. Similarly, for the academic degree choose any of the given examples or pass your own choice. Your supervisor's research group's name is passed to the `\researchgroup` command, while his or her name goes into the `\supervisor` command. The submission date of your thesis is passed to the `\submissiondate` command; you can pass the current date via the `\today{}` command, i. e. not changing the template file.

If you already know your thesis's abstract, write it down in file abstract.tex. You can leave the file unchanged, in which case this template documentation's abstract is used instead until you change file abstract.tex.

Finally, start writing your thesis. Your texts should go into the chapters directory, as described in Section 1.2.2.

1.2 Organisation: Directories

The UPB-CS-TT template assumes a simple directory structure. It has a main directory that holds all files relevant to the template itself. Three complementary directories hold your texts, figures and appendices.

1.2.1 Main Directory

Your main directory contains the four sub-directories

1. `chapters`,
2. `figures`,
3. `pretext`, and
4. `appendices`.

These directories are further described in separate sections. The main directory also contains a couple of files:

`abstract.tex` This file holds the abstract of your thesis.

`appendix.tex` This file groups your appendices together. The file is `\input` into the main file thesis.tex. The appendices themselves should go into the `appendices` folder, and be included into appendix.tex via the `\input` command.

`body.tex` This file contains the main text of your thesis. The file is `\input` into the main file thesis.tex. The texts themselves should go into the `chapters` folder, and be included into body.tex via the `\input` command.

`commands.tex` Your custom LaTeX commands should go into this file. The file is `\input` into the preamble of the main file thesis.tex. By default, commands.tex defines commands for common abbreviations, and some environments. See Section 4.3 for further information.

`literature.bib` Your bibliographic references should go into this file. See Section 5.5 for further information.

`Makefile` This file can make your life easier. It provides means for simple and complex compilation processes. For details, see Section 6.2.

`packages.tex` This is where you include LaTeX packages. The file is `\input` as the first file into the preamble of the main file thesis.tex. By default, it loads the auxiliary packages as described in Section 4.2.

`thesis.tex` This is the main file for your thesis. It defines the document class, includes the preamble, and loads files body.tex and appendix.tex. It also sets up the title page, legalities, and the table of contents.

`upb_cs_thesis.cls` This is defines the document class used by the UPB-CS-TT template. See Chapter 3 for details.

1.2.2 Chapters Directory

This directory is where the .tex files of your thesis go. It is advisable to have one file per chapter. For long chapters, you should create additional files for smaller bits of text, e. g. on the section or even subsection level. The low level files should then be `\input` into the respective chapter's .tex file, while the chapter's .tex file should be `input` in main directory's body.tex.

1.2.3 Figures Directory

This directory is where your graphics go. See Section 5.3 on how to include graphics in your thesis. By default, this directory contains the university's logo (`uni-logo.pdf`), a `Makefile`, and the `template_files` sub-directory.

The university's logo is used on the title page. The *Makefile* of the *figures* directory converts some types of raw data into graphics; see Section 6.3 for details. Finally, the *template_files* sub-directory contains the `tikz_template.tex` file, that can be used to create tikz[1] graphics that the figures directory's *Makefile* can properly process.

1.2.4 Pretext Directory

This directory contains two files `erklaerung.tex` and `titlepage.tex`. The former contains a legal statement, the latter defines the layout of your titlepage. Typically, you do not need to edit these files. You fill in the details of your title page as described in Section 1.1.

1.2.5 Appendices Directory

This directory is where your thesis's appendices go. The files should be `\input` into your thesis via the appendix.tex file. By default, this directory is empty.

[1] TeX Ist Kein Zeichenprogramm, a LaTeX package for programming graphics

2 Hintergrund

2.1 C-Präprozessor

C-Präprozessor ist ein Tool, das den Quellcode vor dem Kompilieren manipuliert [**?**]. Dieses Tool bietet Möglichkeiten zur bedingte Kompilierung, zur Dateieinbindung und zur Erstellung lexikalische Makros [**?**]. Eine C-Präprozessor-Direktive beginnt mit # und geht bis zum ersten Whihespace-Zeichen weiter, optional kann nach der Direktive Argument im Rest der Zeile stehen. Der C-Präprozessor hat solche Anweisungen wie, #include zum Einbinden von Dateien um zum Beispiel Header-Dateien wiederzuverwenden. Wie das Aussehen kann ist in der Abbildung 2.1 Zeile 1 zu sehen(Abb.2.1 Z1). Mit den Anweisungen #if(Abb.2.1 Z6), #else(Abb.2.1 Z10), #elif(Abb.2.1 Z8), #ifdef(Abb.2.1 Z18), #ifndef(Abb.2.1 Z2), und #endif(Abb.2.1 Z4) wird die bedingte Kompilierung erzeugt. Dabei funktionieren #if, #else, #elif, und #endif vergleichbar mit dem was man aus Programmiersprachen und Pseudocode gewohnt ist. #ifdef ist ähnlich zu #if, wird aber nur dann Wahr wenn der drauf folgender Makros definiert ist. #ifndef ist die Negation von #ifdef. Die Makros werden durch die Anweisung #define(Abb.2.1 Z3) erstellt. Der Präprozessor ersetzt dann wehrend seiner Arbeit, dem Makronamen durch seine Definition. Während dieser Arbeit kann ein Makros definiert, umdefiniert und undefiniert werden. Der C-Präprozessor hat noch weitere Anweisungen, auf die wir nicht weiter eingehen. Der C-Präprozessor kann in anderen Programmiersprachen verwendet werden, wenn diese Sprachen syntaktisch ähnlich zu C sind. Beispiel für solchen Sprachen sind C++, Assemblersprachen, Fortran und Java. Der Grund dafür ist, dass der C-Präprozessor ist unabhängig von der zugrundeliegenden Programmiersprache ist. Eine so ähnliche Vorverarbeitungsmöglichkeit ist in vielen anderen Programmiersprachumgebungen.

```c
#include <stdio.h>
#ifndef N
#define N 10
#endif

#if N > 10
#define A "^-^"
#elif N == 10
#define A ";)"
#else
#define A ":("
#endif

        int main()
        {
                int i;
                puts("Hello world!");
#ifdef N
                for (i = 0; i < N; i++)
                {
                        puts(A);
                }
#endif

                return 0;
        }
```

Figure 2.1: Beispiel für C Code mit Präprozessor Anweisungen

2.2 Variabilität Umsetzung mit C-Präprozessor

Der C-Präprozessor ist eine Möglichkeit, Variabilität zu erzeugen [**?**]. Um die Variabilität mithilfe des C-Präprozessors zu erzeugen, brauchen wir dessen Möglichkeit zur bedingten Kompilierung [**?**]. Dabei können beliebige Aussageformeln über Features im Quellcode mit den C-Präprozessor-Anweisungen #if, #ifdef und, #ifndef abgebildet werden [**?**] .

3 The upb_cs_thesis Document Class

In this section we discuss what the *upb_cs_thesis* class as used by the UPB-CS-TT template does, and what commands and environments it provides. We go through the file in chunks of lines that belong together and provide descriptions afterwards.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{ubp_cs_thesis}[2017/12/18 initial version]
```

The UPB-CS-TT template is requires the LaTeX 2_ε and provides the *upb_cs_thesis* documents class

```
\newif\ifgerman\germanfalse
\DeclareOption{german}{\germantrue}
```

The class supports English and German language options. While the German language option (lower case: *german*!) must be explicitly requested, the class defaults to English.

```
\ProcessOptions
\LoadClass[a4paper, 11pt, twoside, openright]{book}
```

The class is based on the standard book class provided by LaTeX.

```
\ifgerman
        \RequirePackage[ngerman]{babel}
\else
        \RequirePackage[english]{babel}
\fi
```

The class loads language packs depending on the language options passed to the class.

```
\RequirePackage[T1]{fontenc}
\RequirePackage[utf8]{inputenc}
\RequirePackage{lmodern}
\RequirePackage{geometry}
\RequirePackage{graphicx}
\RequirePackage{hyperref}
\RequirePackage{csquotes}
\RequirePackage{tikz}
\RequirePackage{calc}
\RequirePackage[explicit]{titlesec}
```

A couple of standard packages are loaded, for details on the packages, see Section 4.1.

```
\reversemarginpar
\geometry{a4paper, twoside, left=30mm, right=20mm, top=30mm, bottom=30mm}
```

The general page layout has 3cm margins on the top and bottom of an A4 page. The inner margin is 3cm wide, while the outer one is 2cm wide. The first page is to be considered a right page, i. e. inner margins on the left hand side of the page.

```
\pagestyle{empty}
\pagenumbering{roman}
```

The first few pages of a thesis are not to show any headers or page numbers. This behavior is altered in the `\tableofcontents` command. Until then, page numbers are counted in Roman numbers, although only displayed on the table of contents page.

```
\newcommand{\@upbchaptername}{}
\newcommand{\@upbsectionname}{}
\newcommand{\ps@upb}{%
  \renewcommand{\@oddhead}{\leftmark\hfill}%
  \renewcommand{\@evenhead}{\hfill\rightmark}%
  \renewcommand{\@oddfoot}{\hfill\thepage\hfill}%
  \renewcommand{\@evenfoot}{\hfill\thepage\hfill}%
}
```

This class provides the *upb* page style that prints the page number centered et the bottom of the page and chapter or section number and name on odd and even pages, respectively.

```
\newcommand*{\emptypage}{\newpage\null\thispagestyle{empty}\newpage}
```

The class provides the `\emptypage` command that takes no argument and adds an empty page to the document.

```
\newcommand{\thetitle}{undefined}
\newcommand{\theauthor}{undefined}
\let\oldtitle=\title
\let\oldauthor=\author
\renewcommand{\title}[1]{\renewcommand{\thetitle}{#1}\oldtitle{#1}}
\renewcommand{\author}[1]{\renewcommand{\theauthor}{#1}\oldauthor{#1}}
\newcommand{\thethesistype}{undefined}
\newcommand{\thedegree}{undefined}
\newcommand{\theresearchgroup}{undefined}
\newcommand{\thesupervisor}{undefined}
\newcommand{\thesubmissiondate}{undefined}
\newcommand{\thesistype}[1]{\renewcommand{\thethesistype}{#1}}
\newcommand{\degree}[1]{\renewcommand{\thedegree}{#1}}
\newcommand{\researchgroup}[1]{\renewcommand{\theresearchgroup}{#1}}
\newcommand{\supervisor}[1]{\renewcommand{\thesupervisor}{#1}}
\newcommand{\submissiondate}[1]{\renewcommand{\thesubmissiondate}{#1}}
```

The class provides seven commands for setting data for a thesis title page:

1. author: `\author`,

2. title: `\title`,
3. thesis type: `\thesistype`,
4. academic degree: `\degree`,
5. supervisor's research group name: `\researchgroup`,
6. supervisor's name: `\supervisor`, and
7. thesis submission date: `\submissiondate`.

These values passed to these commands can be accessed via commands

1. `\theauthor`,
2. `\thetitle`,
3. `\thethesistype`,
4. `\thedegree`,
5. `\theresearchgroup`,
6. `\thesupervisor`, and
7. `\thesubmissiondate`, respectively.

Since `\author` and `\title` commands are provided by LaTeX by default, we change their behavior slightly to make the values passed to the commands available in a manner consistent with the other commands, but we also preserve their original behavior.

These commands are used to typeset the title page as defined in pretext/titlepage.tex

```
\newenvironment{abstract}{
    \begin{center}
        \begin{minipage}{.9\textwidth}
            \ifgerman
                \textbf{Zusammenfassung.} \hspace*{0.10pt}
            \else
                \textbf{Abstract.} \hspace*{0.10pt}
            \fi
}{
        \end{minipage}
    \end{center}
}
```

The class provides the language sensitive `abstract` environment for typesetting an abstract for your thesis.

```
\let\@upbtocold=\tableofcontents
\renewcommand{\tableofcontents}{
  \pagestyle{plain}
  \@upbtocold
  \cleardoublepage
  \setcounter{page}{0}
```

```
  \pagestyle{upb}
  \pagenumbering{arabic}
  \renewcommand{\chaptermark}[1]{
    \markboth{\rm\chaptername\ \thechapter.\ #1}{}
  }
  \renewcommand{\sectionmark}[1]{
    \markright{\rm\thesection\ #1}{}
  }
}
```

Starting with the table of contents, the thesis page numbers are displayed (Roman numbers). After the table of contents, page numbers are reset to 0, displayed in Arabic, and headers (lowercase upright chapter and section names) are shown. For this the behavior of `\tableofcontents` is altered, but its original behavior is preserved.

```
\newlength{\@upbchapternumberwidth}
\newlength{\@upbtitlemaxwidth}
\newlength{\@upbtitletextwidth}
```

These are some dimensions used by the upcoming commands to layout chapter openings.

```
\titleformat{\chapter}{
  \normalfont\huge\bfseries
}{}{0em}{
  \newcommand*{\@upbprintscaledchapternumber}{
    \resizebox{!}{15mm}{\thechapter}
  }
  \settowidth{\@upbchapternumberwidth}{
    \@upbprintscaledchapternumber
  }
  \setlength{\@upbtitlemaxwidth}{
    .9\textwidth - \@upbchapternumberwidth
  }
  \settowidth{\@upbtitletextwidth}{#1}
  \flushright
  \rlap{
    \parbox[b]{\textwidth + 23mm}{
      \parbox[b]{\@upbtitlemaxwidth}{
        \ifdim\@upbtitletextwidth<\@upbtitlemaxwidth
          \hfill #1
 \else
          #1\hbadness=10000
        \fi
      }
      \hspace{12mm}
      \makebox[\@upbchapternumberwidth][b]{
        \raisebox{12mm}{\@upbprintscaledchapternumber}
      }
      \hspace{-\@upbchapternumberwidth}
      \hspace{-12mm}
      \begin{tikzpicture}
```

```
        \fill[black] (0mm, 0mm) rectangle
           (\@upbchapternumberwidth +23mm, 10mm);
      \end{tikzpicture}
    }
  }
}
```

These lines define the layout of a numbered chapter opening: the chapter name to the left of a black box, the chapter number atop the bar. The first few lines compute the width of the printed chapter number, and the text available for the chapter name. The width of the black box depend on the width of the chapter number, so we account for arbitrary digit numbers. If the chapter name is wide than the space available, line breaks are automatically applied, the name becomes left aligned, and the base line of its lowest line is the bottom line of the black box. If there is sufficient space, the text is right aligned.

```
\titleformat{name=\chapter,numberless}{
  \normalfont\huge\bfseries
}{}{0em}{
  \settowidth{\@upbchapternumberwidth}{
    \resizebox{!}{15mm}{0}
  }
  \setlength{\@upbtitlemaxwidth}{
    .9\textwidth - \@upbchapternumberwidth
  }
  \settowidth{\@upbtitletextwidth}{#1}
  \flushright
  \rlap{
    \parbox[b]{\textwidth + 23mm}{
      \parbox[b]{\@upbtitlemaxwidth}{
        \ifdim\@upbtitletextwidth<\@upbtitlemaxwidth
          \hfill
 \else
          #1
        \fi
      }
      \hspace{8mm}
      \begin{tikzpicture}
        \fill[black] (0mm, 0mm) rectangle
            (\@upbchapternumberwidth +23mm, 10mm);
      \end{tikzpicture}
    }
  }
}
```

This is a version of the chapter opening for unnumbered chapters.

```
\titlespacing*{\chapter}{0pt}{30pt}{25pt}
```

This command sets the spacing around chapter openings; it is required by the *titlesec* package.

4 Packages and Commands

This chapter provides a short overview of the packages used in the template. We first have a brief look at the packages directly referenced in the class file upb_cs_thesis.cls. We then discuss the complementary packages included via file packages.tex. Lastly, we also provide a short list of custom commands introduced in file commands.tex.

For all LaTeX packages listed in this chapter, you can find documentation in the Comprehensive TeX Archive Network (CTAN) [The17].

4.1 Class Packages

This section deals with the packages introduced vi the document class file upb_cs_thesis.cls.

babel: english or ngerman. This package loads hyphenation patterns, language-specific commands and characters, etc — whether the *english* or *ngerman* option is used depends on the language option passed to the document class.

fontenc: T1. This package enables font encodings required by most western European languages.

inputenc: utf8. This package is used to interpret input files as utf8 encoded, so unicode characters can be used in the source code. This file encoding is pretty much standard on all operating systems. Typically, you do not need to concern yourself with the functionality of this package.

lmodern. This package loads the *Latin Modern* font.

geometry. This package allows fine-grained and comfortable control over all aspects dealing with paper dimensions and text placements. Typically, you do not need to concern yourself with the functionality of this package.

graphicx. This package allows the inclusion of a wide range of image formats, e. g. .png, .jpg and .pdf, via the `\includegraphics[\dots]{\dots}` command, as shown in Section 5.3.

hyperref. This package enables clickable links within the documents as well as to other resources. See Section 5.2 on how references work.

csquotes. This package loads the correct quotation marks, depending on the language passed to the babel package.

tikz This package is used to draw fancy images and make simple plots with LaTeX. While these graphics look really great, creating them can be time consuming. The package's functionality can be greatly extended by loading additional libraries.

calc. This package allows for some more intuitive programming of computations in certain contexts.

titlesec. This package allows for the redefinition of chapter and section headings, etc.

4.2 Auxiliary Packages

This section provides a list of packages loaded from file packages.tex. Preferably, if you want to use your own packages, you add them to this file.

amsmath, amssymb, amsfonts. These packages introduce a load of mathematical symbols and environments.

amsthm. This package introduces the ability of comfortable adding environments for definitions, theorems, etc. A `proof` environment is provided by default. See the commands.tex file for example environments, and the marked examples in this document for how these environments are rendered, e. g. Example 5.1.

todonotes. This package allows you to mark open tasks (ToDos) by passing them to the `\todo` command.

Like this ToDo.

Or like this with the `[inline]` variant to the command.

You can create a comprehensive list of open ToDos via `\listoftodos`.

subfig. This package allows you to combine multiple figures into one, with each sub-figure getting its own caption. See Section 5.3 on how to use this feature.

url. This package is used to render URLs correctly via command `\url`. In conjunction with the *hyperref* package, which is loaded by default, the URL becomes a clickable link.

longtable. Normally, LaTeX tables only cover a single page. With this package, long multipage tables become possible. See Section 5.4 for further information.

booktabs. This packages improves the default look of LaTeX tables. See Section 5.4 for further information.

acronym. This package allows you to define acronyms. You can use the acronyms without needing to keep track about whether you have formally introduced them before, or not. See Section 5.1.3 for further information.

4.3 Custom Commands

LaTeX allows you to easily add your own commands. Your own commands should go into file commands.tex, which also introduces some commands by default.

The first blocks of commands, sets up some environments according to the *amsthm* package. Environments for theorems, lemmas, corollaries, properties and definitions are provided. The environments are set up such that they share a counter, i. e. you will not find Theorem 2.1 after Lemma 2.15; instead, the theorem that follows Lemma 2.15 will be Theorem 2.16, unless some other of these environments is used in between.

The commands.tex file also provides some commands for common abbreviations with correct spacing, so you will not find the *e.* and *g.* of the e. g. in different rows of your text. Remember to use them with a pair of braces, as is good practice. Otherwise you may find that space is missing between the abbreviation and the following word. The abbreviations provided are e. g., i. e., c. f., and et al.

The file also provides an acronym definition for the *acronym* package, as well as a math operator for correct from the *ams* packages.

5 A Short Guide

In this chapter, we discuss how to use the UPB-CS-TT. Although we mainly present information every versed LaTeX user should know, even an expert may gain some new knowledge. Particularly, we discuss simple but important style choices that you must make when writing LaTeX. We also discuss how to set clickable references to different parts of your thesis, such as chapters, figures, tables or full bibliographic references. Figures, tables and bibliographies are also discussed in this section.

5.1 Style

This is just a simple guide on how to use the UPB-CS-TT, so we do not get into the details of how to write a good scientific text. However, we give a short overview on the means that LaTeX and UPB-CS-TT provide to make your life easier, and achieve your goal of writing a high quality thesis. In this section, we discuss proper use of LaTeX's math mode and the footnotes.

5.1.1 Math Mode

Whenever you use the language of mathematics in your text, you should tell LaTeX! Do so using the sequence `\(` at the start of the formula or expression, and `\)` at the end. These commands render the mathematical expression in-line. For long, complex, or important expressions, LaTeX offers the similar `\[` and `\]` commands.[1]

The various *ams* packages leaded by default in file packages.tex provide a number of environments that also switch to math mode, but introduce additional functionality, like equation counting and multi-row formulas with custom alignment of content.

Whenever you use the language of mathematics in your text, you should tell LaTeX! There really is a difference between x and x! If you have a variable name that consists of multiple letters, you should pass the name as an argument to command `\mathit`, as to avoid confusion between abc and abc, where the latter is short for $a \cdot b \cdot c$.[2]

When you use your own mathematical operators, similar to sin, you should explicitly tell LaTeX that it is an operator. Do so by using the `\DeclareMathOperator` command. Command

[1] The use of `$` and `$$` is simply bad practice in LaTeX. Using the same symbol as the start and stop marking makes reading and understanding things really hard. It also is confusing to some syntax highlighters in certain circumstances. However, switching to math mode via `$` may still be used in certain contexts in which the LaTeX equivalents do not work, for example in `\item[$foo$]` commands inside lists.

[2] If you haven't noticed, the spacing and letters are slightly different.

`DeclareMathOperator` works similar to `\newcommand`, but gets the spacing right; compare

$$1\,\mathrm{top}\,23 \quad \text{and} \quad 1\mathrm{top}23,$$

as produced by `1 \testop 23` and `1 \mathrm{top} 23`.[3]

5.1.2 Footnotes

In many cases, you may[4] be inclined to use footnotes to provide additional information via the `\footnote` command. Don't! Footnotes break the flow of reading. This means, you divert the reader's attention to some information you do not consider to be sufficiently important to add it to the main text body. As a result, using the footnote achieves the exact opposite of what it is supposed to achieve.[5] You have probably experienced this yourself while reading this section.

5.1.3 Acronyms

When writing your thesis, you may discover that some terms are repeated very often. As a result you want to abbreviate the terms. This is normal and completely fine. However, you have to take care that you do not start using an abbreviation before you formally introduce it. Otherwise your readers may be unable to understand your text.

Unfortunately, you are so used to the term and its abbreviation that you will probably not notice that you use an abbreviation that has not been introduced yet, particularly when you edit your text later on, e. g. after proof reading. Fortunately, there is a package to help you: the *acronym* package that is loaded by default via the packages.tex file. *acronym* allows you to define the term you use, the abbreviation and the long form as `\newacro{myacro}[MA]{my acronym}`, where `myacro` is the term you use, `MA` is the abbreviation and `my acronym` represents the long form. You can insert the abbreviation called `myacro` by using `\ac{myacro}`, which is rendered as "my acronym (MA)" for its first use, and as "MA" for each subsequent use. The *acronym* package provides variants of the `\ac` command to use the plural form of the abbreviated term or to explicitly use the unabbreviated form.

5.2 References

LaTeX allows you to insert insert clickable references to other parts of the document into your thesis. These references help your readers avoid tedious scrolling to get to a specific page or searching for some Figure 17, that can be located anywhere.

The clickable references are enabled from the *hyperref* package that is loaded by default via file thesis.tex. With *hyperref* loaded, LaTeX puts references into the table of contents, so the reader can jump to the right section without a hassle. However, it is good practice to insert such references yourself wherever needed.

For example, your thesis's introduction will typically contain a paragraph that provides an overview of the thesis's structure. This is where you would typically add references to the sections that you talk about, e. g. the foundations of your work are presented in Chapter 4. The reference to Chapter 4 is invoked by using command `\ref{ch:packages}`. Of course, you need to tell LaTeX what you mean by "ch:packages." For that, you put a `\label{ch:packages}` right at the start of the chapter called "Packages and Commands," like you can see in the code for this document, also replicated in Example 5.1.

[3] `\testop` was defined via `\DeclareMathOperator{\testop}{top}`.
[4] read as: probably are
[5] That is: provide additional, less important information that can be skipped without harm.

Example 5.1. *This is the code for starting a new chapter and setting a label/jump mark:*

```
\chapter{Packages and Commands}
\label{ch:packages}
```

The `\ref` command is also used to create references to other parts of your thesis, e. g. figures, tables, equations, or the code example that you find above.

Typically, the references are rendered as numbers, like 5.1, potentially including letters if the reference points to the appendix of your document. However, you can tell LaTeX to make other text into a reference as well. For example, the code example above can also be referenced via `\hyperref[ex:code_label]{code example}`.[6] If you want to point to specific equations inside an equation environment, you should use `\eqref` instead of `\ref`, so the reference renders correctly: surrounded by a pair of parentheses.

Two other classes of references are enabled by the packages UPB-CS-TT loads by default: URLs and bibliographic references. The *url* package in conjunction with *hyperref* makes URL clickable, and if your PDF viewer supports it, clicking such a URL accesses the URL using your standard web browser. Such references are created using the `\url{URL}` command.

Bibliographic references. Bibliographic references point to specific entries in the bibliography section of your thesis, assuming you use the default setup of this template. However, for bibliographic references you do not manually put jump marks/labels into your document. The marks are instead taken from the labels provided in the .bib file, as discussed in Section 5.5. You can create bibliographic references using the `\cite{label}` command. The `\cite` command correctly puts bibliographic references in square brackets. Note that the `\cite` command can take an optional argument that is typically used if you want to give further information, e. g. if you talk about Chapter 27 of the book referenced as `FancyBookRef`, you should put the reference as `\cite[Chapter~27]{FancyBookRef}`.

5.3 Using Graphics

LaTeX together with the *graphicx* package provides a simple means to add graphics to your thesis. Use the `\includegraphics` command to include a graphic.[7] The command takes the file name of the graphics (relative to the main file) as its paremeter. As optionaly parameters you can pass key=value pairs that determine scaling factors, e. g. `[width=10cm]`.

If you use graphics compiled from .svg via inkscape, as described in Section 6.3, they are included via

```
\def\svgwidth{10cm}
\input{pdf_tex file}
```

Here, the scaling factor is given as the parameter to `\svgwidth` and the path to the graphics file (a .pdf_tex file), given via `\input`(!) must be relative tou your thesis's main file.

Typically, you want to put your graphics in context, e. g. add a caption and a label. For this, the *figure* floating environment is used, see Example 5.2

Example 5.2. *An example of a* figure *environment.*

[6] Note that with `\hyperref`, the label is the optional argument!

[7] only some file types are supported. With the default UPB-CS-TT template, .pdf, .png and .jpg files work just fine.

```
\begin{figure}[tbph]
        \includegraphics[...]{...}
        \caption{Some descriptive text}
        \label{fig:name}
\end{figure}
```

The optional parameter passed to the environment determines its preferred placement on a page, while the order of the letters determine the options' precedence. In this case, LaTeX first attempts to put the figure on the top (`t`*) of a page, the bottom (*`b`*) of the page is the second option. Putting the figure on a special page (*`p`*) for floating environments is the third option, and as a last resort, LaTeX will put the figure at the point of its definition (relative to the source code).*

Inside the environment, first the graphics file is included, then a caption is given, and finally a label is defined. Note *that the label must be given after the caption for it to function properly. Also* node *that with figures, the caption is given after the graphics.*

In some cases a figure consists of sub-figures. For this, you use the `\subfloat` command as provided by the *subfig* package. You put the `\subfloat` inside the *figure* environment and pass the `\includegraphics[...]{...}` command as a parameter. The sub-figure can have an individual caption an label: the caption is passed to the `subfloat` as an optional parameter (no `\caption` command!) that also contains the respective label (passed as a `\label` command).

5.4 Tables

Tables are a means to display a lot of information in a comprehensible manner. LaTeX offers many options to create and structure tables. Unfortunately, given the options, many people have a tendency to put too many structuring elements into their tables. As a result, what is intended to improve the readers understanding has quite the opposite effect.

Basic advice on what not to do in good tables typically includes:

- never use vertical bars,
- use horizontal bars rarely, e. g. for grouping multiple entries, and
- if there is a real possibility for readers to confuse entries from multiple rows as belonging together, use color coding (with decent(!) colors) to make boundaries clearer (there are LaTeX packages for that, google them if necessary).

Following this advice, we present you with the basics on how to produce tables that look good.

Tables are produced in a *tabular* environment, such as in Example 5.3.

Example 5.3. *This is a code example of a simple table.*

```
\begin{tabular}{lcp{3cm}r}
  \toprule
  This & is & an & example \\
  \midrule
  Hello & World & I really have no idea what to write & here \\
  so & I just write & some nonsense & text. \\
  \bottomrule
\end{tabular}
```

The code tells LaTeX to produce a table (tabular environment) with four columns (arguments `l`*,* `c`*,* `p{3cm}` *and* `r` *to the environment. The four arguments state how to align the text in the respective column.*

- `l`*: Left aligned text; the column's width is automatically determined by the widest entry.*
- `c`*: Center aligned text; the column's width is determined as for Option* `l`*.*
- `p{3cm}`*: Block aligned text; the column's with is set manually to 3cm.*
- `r`*: Right aligned text; the column's width is determined as for Option* `l`*.*

Inside the environment, we start with a `\toprule`*, a thick-ish horizontal bar to indicate the beginning of the table. The header of the table follows, the change of columns indicated by the* `&` *character; the row is finished with the* `\\` *mark. The end of the header section is visually marked by another (thin) bar created by* `\midrule`*. In the example, two more rows of contents follow, each ended by the* `\\` *mark, again with the* `&` *character to separate cells. The table's end is visually marked by another thick-ish horizontal bar.*

The table is rendered as follows:

This	*is*	*an*	*example*
Hello	*World*	*I really have no idea what to write*	*here*
so	*I just write*	*some nonsense*	*text.*

Note that the commands `\toprule`, `\midrule` and `\bottomrule` are provided by the *booktabs* package that is loaded from file packages.tex. Without these, LaTeX's `\hline` can be used, but messes up the spacing between rows, resulting in ugly tables.

Typically, you do not want to surround your tables with context, e. g. add a caption to it and be able to point to it from any part of the document. You can achieve this by using the *table* floating environment as shown in Example 5.4. As a floating environment, it may be placed where it fits, rather than where (relative to the source code) it is defined.

Example 5.4. *An example of the* table *environment.*

```
\begin{table}[tbhp]
  \caption{Some descriptive text}
  \label{tab:name}
  \begin{tabular}{...}
    ...
  \end{tabular}
\end{table}
```

The optional parameter passed to the environment determines its preferred placement. The order of the letters gives the precedence. In this case, placement of the table at the top (`t`*) of a page is preferred, followed by a placement on the bottom (*`b`*) of a page. If neither is possible, LaTeX is to attempt to put the table where it is defined (*`h`*, relative to the source code), and if everything else fails, the table will be put onto a special page (*`p`*) that may only hold floating environments. This last option is one of the reasons, why the caption should be some descriptive text that a reader can understand without having read the main text.*

Inside the environment, you find the `caption` *of the table, followed by its* `label`*. The label does not work properly if it precedes the caption. Finally, the table environment holds the relevant table environment.* Note *that, unlike with figures, with tables, the caption precedes the table!*

Sometimes you may have tables that are too long to fin onto a single page. This is problematic, since LaTeX does not put page breaks into tables. The *longtable* package comes to rescue. It provides the *longtable* environment that puts page breaks into tables, repeats the table header

automatically on each page that holds a portion of the table, and indicates that the table is continued on the next page if necessary. Example 5.5 shows how to create long tables. *Note* that with long tables, you do not need/want a floating environment. As compensation the `\caption` and `label` commands can be used with the *longtable* environment directly.

Example 5.5. *This is a code example of a long table*

```
\begin{longtable}{l}
  \caption{Some descriptive text}
  \label{tab:name} \\

  \toprule
  HEADER \\
  \midrule
  \endfirsthead

  \midrule
  SECOND_HEADER
  \midrule
  \endhead

  \midrule
  FOOTER
  \endfoot

  \bottomrule
  \endlastfoot

  Table contents
\end{longtable}
```

The longtable *environment takes a parameter just like the* tabular *environment that tells LaTeX about the number of columns end text alignment. The* `\caption` *and* `\label` *commands follow as in the* table *environment. This part is ended with the* `\\` *mark.*

What follows is a definition of the first header `HEADER` *of the table, its end is marked by* `\endfirsthead`*. The secondary header* `SECOND_HEADER` *is the header that gets repeated after each page break. The end of its definition is marked by* `\endhead`*. The footer* `FOOTER` *gets repeated just before every page break that occurs in the table. Its definition is ended by* `\endfoot` *The footer that marks the end of the table gets defined last (in this case a simple* `\bottomrule`*). The end of its definition is marked by* `\endlastfoot`*. Finally, the tables contents are put into* longtable *environment, just like you would put in* inside *the* tabular *environment.*

5.5 Bibliographies

Typically you do not want to manage you bibliographic references yourself. LaTeX and its helper program *bibtex* manage them for you. You just need to tell them what references you are going to use. You do that by putting a bibliographic reference — called *bib entry* — into file bibliography.bib.

The bib entries are as shown in Example 5.6. The key=value pairs depend on the type of the bib entry. Certain keys must be present in an entry of a given type, while others can be missing.

Example 5.6. *This is an example bib entry of type "inproceedings." It starts with an @ symbol, immediately followed by its type and an openeing curly brace and a label (LABEL in this example) and a comma. What follows is a comma-separated list of key=value pairs, with the values being surrounded by curly braces. The bib entry closes with a final curly brace.*

```
@inproceedings{LABEL,
  author    = {First Author and Second Author and ...},
  editor    = {A List of Fancy People},
  title     = {Example Title},
  booktitle = {Proceedings of Conference 1468},
  pages     = {123--456},
  publisher = {Publisher's Name},
  year      = {1468},
}
```

Note that you can enforce a certain type of letter capitalization, i. e. all caps, by putting the respective term inside an extra pair of curly braces, e. g. `publisher = {ACM},` *may be rendered as Acm, while* `publiser = {{ACM}},` *is always rendered as ACM.*

Using *bibtex*, you can compile a nicely looking list of bibliographic references. The list is included via the *biblatex* package at the place indicated by the `\printbibliography` command. You can refer to the individual list entries by using the label (in the example LABEL) of the corresponding bib entry in a `\cite` command as described in Section 5.2. Hence, it is useful to derive a bib entry's label from the information provided by the entry itself, e. g. the title.

While *bibtex* provides means to manage your bibliographic references, they do not help you with obtaining all relevant, and particularly complete, bib entries. You should create bib entries yourself, unless all other options fail. Typically, publishers of computer science literature provide bib entries on their web pages. Also Google Scholar[8] provides bib entries. However, both sources sometimes provide very incomplete bib entries. Complete entries can typically obtained from DBLP.[9]

[8] `https://scholar.google.com`
[9] `http://dblp.org/search/index.php`

6 Makefiles

This UPB-CS-TT template comes with two Makefiles to make your life easier. One of the files is located inside the template's main folder, the other one is located in the figures sub-folder. The main Makefile can be used to compile the thesis's source files into a PDF file, while the figures directory's Makefile is used for creating graphics out of certain types of raw data. In this chapter, we discuss what both files do.

You can compile your thesis without these Makefiles. However, the Makefiles make sure the compilation process happens in the right order. For a simple compilation, type "make" into your console and hit Enter. This starts the simple compilation process (calling pdflatex twice). Type "make complete" and hit Enter, in order to compile the source files, and include compiled images and bibliographic references. This latter option may pose problems in very specific circumstances. Therefore we provide a section on troubleshooting next.

6.1 Makefile Troubleshooting

The Makefiles introduce certain dependencies, i. e. applications they need to properly function. However, they will typically not pose a problem. The issues that may occur are:

1. your thesis's bibliography file is empty, or
2. dependencies are not installed, but relevant files exist.

Empty bibliography file. The compilation process may fail if the bibliography.bib file does not contain a reference. In this case, either add a reference, or change Line 8 of the main Makefile from `complete: images lit short` to `complete: images short`, or comment out Lines 12 and 13 of the main Makefile by prepending them with a hashtag (#). The first option is to be preferred if you already know at least one bibliographic reference that you will use, the second option is reasonable if you will not have any bibliographic reference (very unlikely), and the third option is more or less a quick and dirty hack.

Missing dependencies. The compilation process may also fail, if you have .svg files in your figures directory, but *inkscape* is not installed, or you have .tex files in the figures directory but either *pdflatex* or *pdfcrop* is not installed. In either case, you can install the relevant dependencies, or remove the problematic files, or comment out the problematic lines of the figures directory's Makefile by prepending them with a hashtag (#). Which lines to comment out depend on the file type that causes the problems; see Section 6.3 for details.

6.2 Main Makefile

The main Makefile is designed to compile the LaTeX source files of your thesis into the final PDF file. The Makefile provides two main recipes for obtaining the PDF, a short and a long version. The short version is invoked via "make" or "make short," the long version is called via "make complete."

We now have a look at the Makefile itself. The first line stores the file name (without the file extension) of your thesis's main LaTeX file in variable `THESIS` Typically, this is thesis.tex, and THESIS store the value "thesis," accordingly.

Lines 4 to 6 are the recipe for the short compilation process, they simply call pdflatex twice on your thesis's main source file.

Line 8 contains the recipe for the long compilation process. However, rather than containing the process itself, it depends on three other recipes that are called automatically: "images," "lit" and "short." Hence (see below and above) the long compilation process created image files from raw data in the figures directory, processes bibliographic references and finally puts the PDF together.

Lines 10 to 12 process bibliographic references. First, *pdflatex* is invoked on your thesis's main file in order to collect the bibliographic references that you actually use, and then *bibtex* for processing bibliographic references is called.

Lines 14 to 15 compile raw image data from the figures directory. The do so by changing into the directory and executing the directory's Makefile; for details, see the next section.

6.3 figures's Makefile

The figures sub-folder's Makefile is designed to process raw data and transform them into useful images. It handles vector graphics in the .svg format, as well as graphics described via tikz[1] in .tex files. The reasoning behind putting tikz graphics in separate files is that compiling the code into graphics is somewhat time consuming, so recompiling graphics that do not change should be avoided.

We now go through the Makefile and describe its purposes. The first line collects all .svg files in the "figures" directory, and stores their file names in variable `SVG_FILES`. The second line creates a variable `INKSCAPE_FILES` and initializes it with all the file names stored in `SVG_FILES` with the .svg file extension replaced by .pdf_tex. Similarly, Lines 4 and 5 collect the .tex files and corresponding .pdf files in variables `TEX_FILES` and `TIKZ_FILES`.

Line 7 says that, by default, we want to create the .pdf_tex files from .svg files, and .pdf files from .tex files. The following lines say how this transformation process works. However, the transformation is only applied to source files that have changed recently, i. e. if the target file does not exist yet, or source file's last modified date is newer than the target file's last modified date.

Lines 9 and 10 are the recipe for creating .pdf_tex files out of .svg files. For this, the application *inkscape* is used. The recipe calls inkscape in command line mode and asks it to export the source file's drawing and text as two separate files. The first output file is a .pdf and contains the drawing itself. The second file is the desired .pdf_tex that is a LaTeX file in disguise.[2] It contains LaTeX code that puts embeds the .pdf file into your thesis, but also contains the text from the .svg file and overlays the .pdf graphic with the text. As a result, the text uses the same font as your document, and all text from the .svg file gets interpreted as LaTeX code. *Note* that due to some bug in the inkscape export, you may observe LaTeX errors

[1] TeX Ist Kein Zeichenprogramm, a LaTeX package.

[2] The .pdf_tex file extension helps with avoiding confusion with other .tex files.

during compilation if your .pdf file contains more than a single page, i. e. if the .svg file was created with inkscape in the first place, you must put all graphic elements onto the same layer before exporting.

Lines 12 to 14 are the recipe for getting .pdf files from .tex files. For that, the source file is compiled using pdflatex, as expected. Afterwards, the resulting .pdf file is cropped to the drawing area, with some surrounding margins. This adds the applications *pdflatex* and *pdfcrop* as dependencies.

Bibliography

[OPHS16] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. The Not So Short Introduction To LaTeX 2_ε, 2016. Checked 2017-12-18.

[The17] The CTAN Team. CTAN Comprehensive TeX Archive Network, 2017. Checked 2017-12-18.