DO NOT INCLUDE THESE PAGES IN YOUR THESIS! THIS IS FOR INSTRUCTION ONLY!

The main file for this thesis is `thesis.tex`. All other files are `included` from `thesis.tex` or another `included` file!

The `.tex` files used in this example thesis are:

- `thesis.tex`: The main thesis file from which everything is included.
- `header.tex`: Includes the packages, styles, and definitions used in this example.
- `algorithms-and-listings.tex`: Included from `thesis.tex` — Examples of typesetting algorithms and code listings.
- `citations.tex`: Included from `thesis.tex` — Examples of citations using LaTeX's `natbib` package.
- `figures-and-tables.tex`: Included from `thesis.tex` — Examples of figures, tables and sub-floats.
- `fine-points.tex`: Included from `thesis.tex` — Some fine points of typesetting, to give your thesis even more of a polished touch.
- `formatting.tex`: Included from `thesis.tex` — Examples of formatting and spacing issues.
- `getting-started.tex`: Included from `thesis.tex` — A few notes on how to get started using LaTeX according to your chosen operating system.
- `graphics-info.tex`: Included from `thesis.tex` — Information about including graphics in your thesis.
- `introduction.tex`: Included from `thesis.tex` — A short introduction.
- `lists.tex`: Included from `thesis.tex` — Information about how you create lists of things.
- `packages.tex`: Included from `thesis.tex` — Describes the various packages used in this example.

- `preliminaries.tex`: Included from `thesis.tex` — The file from which the thesis template draws personal details about the thesis (e.g., thesis title, you name, your advisor, etc.).
- `sections.tex`: Included from `thesis.tex` — How to use the sectioning features of LaTeX.
- `websites.tex`: Included from `thesis.tex` — A few web sites that you may find useful.

DO NOT INCLUDE THESE PAGES IN YOUR THESIS! THIS IS FOR INSTRUCTION ONLY!

This document was last updated November 9, 2016.

(PUT YOUR THESIS TITLE IN PRELIMINARIES.TEX)

by

(PUT YOUR FULL NAME IN preliminaries.tex)

Thesis submitted in partial fulfillment of the

requirements for the Degree of

Bachelor of (PUT YOUR DEGREE IN preliminaries.tex) with

Honours in (PUT YOUR DEPT IN preliminaries.tex)

Acadia University

(March) (20XX)

This thesis by (PUT YOUR FULL NAME IN preliminaries.tex)
is accepted in its present form by the
(Department or School) of (PUT YOUR DEPT IN preliminaries.tex)
as satisfying the thesis requirements for the degree of
Bachelor of (PUT YOUR DEGREE IN preliminaries.tex) with Honours

Approved by the Thesis Supervisor

(Dr. Your Supervisor)	Date

Approved by the Head of the Department

(Dr. The Head or Director)	Date

Approved by the Honours Committee

(The honours committee person)	Date

I, (PUT YOUR FULL NAME IN preliminaries.tex), grant permission to the University Librarian at Acadia University to reproduce, loan, or distribute copies of my thesis in microform, paper or electronic formats on a non-profit basis. I, however, retain the copyright in my thesis.

_____

Signature of Author

_____

Date

Acknowledgements

Place any acknowledgments you might want to make here.
Don't forget to be formal and professional.

Contents

Abstract **xvii**

1 Getting Started **1**

1.1 Using Linux . . . 2

1.2 Using MS Windows . . . 4

1.2.1 IDE . . . 4

1.2.2 Support in MS-Windows . . . 4

1.2.3 Command Line Support . . . 4

Option 1: *pdflatex* . . . 4

Option 2: *texify* then *dvipdfm* . . . 5

Option 3: Using *texify* then *GhostView* . . . 5

1.3 Using Macs . . . 6

1.4 Tutorials, etc. . . . 6

1.5 Quick Start Guide . . . 7

2 Introduction to the Acadia Style **9**

3 Formatting Points of Interest **11**

3.1 One or Two Sides? . . . 11

3.2 Emphasis . . . 11

3.3 Spacing . . . 13

3.4 Controlling automatic spacing and hyphenation . . . 15

4 Sectioning **17**
4.1 A Sample Section . . . 17
4.1.1 A sample SUBSECTION . . . 18
And a SUBSUBSECTION . . . 18
PARAGRAPH 1 . . . 18
SUBPARAGRAPH Label . . . 18
4.2 Custom "Environments" . . . 18
4.3 Cross-Referencing Sections . . . 20

5 Acronyms and Lists **23**
5.1 Defining an Acronym Command . . . 23
5.2 Examples of Lists . . . 24

6 Citations **27**

7 Creating and Including Graphics **31**
7.1 Types of Graphics . . . 31
7.2 Including External Graphics . . . 32
7.2.1 Graphics File Types . . . 33
7.3 Internal Graphics . . . 34

8 Figures and Tables **39**
8.1 Floats . . . 39
8.2 Figures and Tables . . . 40

9 Some Fine Points of Typesetting **43**
9.1 Non-Breakable Spaces . . . 43
9.2 Inter-Sentence Spacing . . . 44
9.3 Hyphens and Dashes . . . 44
9.4 Clubs and Widows . . . 44

10 Algorithms and Listings **47**
10.1 Algorithms . . . 47
10.2 Listings . . . 49

11 Packages **53**

A Web sites **55**

Bibliography **57**

List of Tables

6.1 Citation commands and results, in parenthetical or textual style. 28
6.2 Multiple citations. 29
6.3 Citations with Optional Text. 29

8.1 Table of values for $F(\chi) = \chi + 4$. 41
8.2 Table of values for two other $F()$s. 42
8.3 Table of values for $F(\chi) = \chi + 4$. 42

List of Figures

7.1 The Petersen Graph. 38

8.1 An example image. 41
8.2 Making Peanut Butter Cookies. 41

Abstract

This is a "quick-start" guide to help you use LaTeX to produce your thesis. It by no means covers all issues, but it should give you a solid start.

TeX is a system developed by Donald Knuth with the goal of doing beautiful typesetting, particularly for documents which use mathematics. It is one of the first significant programs made available to the world at large *for free* by its author. By design, Knuth has dictated that TeX won't change any more, which means that TeX documents won't become unusable because of incompatible updates. (However, other people are extending TeX, so it is not a "dead" system.)

LaTeX is a set of additions to TeX which arguably make it easier to produce documents. This sample thesis will assume you are going to use LaTeX but if you later to decide to try Knuth's "plain" TeX (or LaTeX's "competitor", ConTeXt) you will find that the concepts are fairly similar.

This document was initially written by Brian Demmings; almost any "I" found in this document refers to Brian. Jim Diamond has updated it occasionally since Brian graduated, and hopes he has only improved things with his changes. Alex Sanford took the masters thesis template and transmogrified it into the honours thesis template, fixing the many problems with the former version.

The "source code" of this document should be in the same directory as the nicely-formatted PDF file you are reading. If you want to see exactly how anything here is produced, just look in the appropriate "source code" file. (For example... if you read the source code, you will see that we used "`\dots`" rather than "`...`", since the spacing of "…" is better than "...".)

Write *your* abstract here (in the file `preliminaries.tex`). That is, using your favourite text editor (see Chapter 2), edit the file `preliminaries.tex`. Delete all of the text below

the "`\prefacesection{Abstract}`" line down to the following "`%----------`" line; then type your abstract into that location.

Note: your abstract should not go on anywhere near as long as this did!

BEWARE: Some of the techniques (e.g., the "Table of Algorithms/Listings") may *not* be permitted. When in doubt check with RGS (masters students), the registrar (honours students) or your advisor.

Chapter 1

Getting Started

The basic idea of LaTeX (and "plain" TeX) is that you create one or more *plain text* "source files" using any text editor (but not word processor!) of your choice, and then "compile" your source files into a nicely-formatted document (normally a PDF file). This work flow separates the writing of your ideas from the formatting and style details, allowing you to concentrate on what you want to say, without simultaneously worrying about getting the look "just so". In particular, this thesis template allows you to create a document meeting Acadia's thesis formatting requirements without having to worry about all the details of the front pages, page numbering, lists of figures and tables, and so on... it just happens automatically for you. But, if some time later you wish to publish your writing elsewhere (e.g., a journal or conference), you can change the few lines which specify the formatting, select the parts of your work you want to publish, and re-compile.

To use LaTeX (or plain TeX) to produce your thesis (or other documents or presentations) you need to have a TeX "distribution" installed on your computer, or you need to have a so-called "live TeX DVD". The former is preferable to the latter, unless you are extremely short of disk space.

As mentioned in the abstract, and given the existence of this thesis template, the quickest way to get going with the TeX system is to use the LaTeX package. LaTeX can be used to create so-called "`.dvi`" files, from which you can subsequently create PostScript or Adobe PDF files. However, LaTeX can also directly create PDF files, and this route is strongly recommended for the most users. (There are two reasons for this recommendation. First, since the library wants a PDF file, there are less steps involved if you just create a PDF file

from the get-go. Second, it is easier to include *most* graphics made with other programs (photographs, plots, ...) using the "flavour" of LaTeX which creates PDF files.)

You can download (for free!) a TeX distribution from `http://www.tug.org/texlive/` for many different systems. However, if your operating system supplier provides an up-to-date TeX distribution, you may (or may not!) find it easier to just use that. The way you use LaTeX varies a bit, depending on which operating system you use on your computer, as well as which TeX system you installed on your computer. The next three subsections give some operating system-specific details.

You can use an ordinary text editor to create or edit your "source files", but there are some editors which are specifically designed for creating TeX or LaTeX documents. (In fact, these "editors" tend to be more than plain editors; a computer scientist might call them integrated development environments (IDEs)). Three such editors, all available for free, are

- TeXstudio, available at `http://texstudio.sourceforge.net/`,
- Texmaker, available at `http://www.xm1math.net/texmaker/`, and
- TeXworks, available at `http://www.tug.org/texworks/`.

A comparison of these (and others) can be found at `http://en.wikipedia.org/wiki/Comparison_of_TeX_editors`.

1.1 Using Linux

Regardless of which Linux distribution you use, you should be able to download and install one or more packages to get TeX and LaTeX running on your system (if the TeX system is not already installed). Look through the list of packages available for download and installation for your distribution. For example, in Ubuntu-type systems you can use your GUI package manager to install the `tex` package, or from the command line you can type "`sudo apt-get install tex`". This is a big download; make sure your network connection is good and you have more than a minute or two for the download to complete.

If you are unable to find anything specific for your Linux distribution, the TeX Live distribution is quite easy to install and should work on virtually any Linux distribution. The installation instructions are found at the web site given above.

TeXlive for Linux does not provide (what a computer scientists might call) an integrated development environment (IDE), but some people like the combination of `emacs`

and `auctex`. Alternatively, you might like to try out one of the free TEX editors mentioned above. However, the command-line usage of LATEX is very straightforward, so you might decide that you don't need to spend time trying to find a suitable IDE.

To compile any LATEX document from the command line, you can either use the program `latex`, which produces "`.dvi`" files, or you can use `pdflatex`, which produces "`.pdf`" files. All things being equal, the latter is probably the best choice. Of course, most people (as well as the library) will be able to use a PDF file, but if you produce a `.dvi` file, you will need to use a post-processor to create a PDF or PostScript file. The `.dvi` approach is probably only of interest to you if you want to use the power of the PostScript language to do tricky things.

Having created one or more LATEX source files using your favourite text editor, the next step is to "compile" your document into a PDF file. This can be done with the following steps. First, assuming that your document is in a file called `thesis.tex` (or that `thesis.tex` includes all of the files comprising your document), run

```
pdflatex thesis.tex
```

Note that under Linux you do not need to type the "`.tex`", `pdflatex` will add that for free. Assuming you are using `bibtex` to process your citations, next run

```
bibtex thesis
```

Finally, to ensure that all of the references in your document are correct, re-run `pdflatex` twice (yes, twice):

```
pdflatex thesis
```

You can, of course, type all these on one line, as follows:

```
pdflatex thesis && bibtex thesis && pdflatex thesis && pdflatex thesis
```

If for some reason you want to go the `.dvi` route, use `latex` instead of `pdflatex` in the examples above. Then, you can create a PostScript file by running

```
dvips thesis.dvi -o thesis.ps
```

and if you want to create a PDF file from that you can run the command

```
ps2pdf thesis.ps
```

Alternatively, a PDF file can be created directly from the `.dvi` file with the command

```
dvipdfm thesis.dvi
```

There is a "`Makefile`" in the thesis template directory with the other files. You can just type `make` to create a new PDF version of your thesis whenever you have made some

changes. The Makefile is a bit heavy-handed, since it runs `pdflatex` three times and `bibtex` once, even if one run of `pdflatex` would suffice. Ho hum.

1.2 Using MS Windows

1.2.1 IDE

A (MS Windows-only) IDE which many people use to write LaTeX is called TeXnic Center; at time of writing it can be found at: `http://www.texniccenter.org/`.

You can use any text editor to edit `.tex` files, but TeXnic Center provides useful extras. Another alternative, `Texmaker`, which is also available for Linux and OS X, can be found at `http://www.xm1math.net/texmaker/`.

1.2.2 Support in MS-Windows

TeXnic Center relies on MiKTex to perform the actual compiling into DVI, PS, PDF, etc. MiKTex can be found at `http://www.miktex.org/`. You'll need to install MiKTex *before* installing TeXnic Center.

1.2.3 Command Line Support

You may choose to type commands in to a command window, rather than using the pointy-clicky features of TeXnic Center or `Texmaker`. The next subsection describes how to do that. But if you like the pointy-clicky method, you can skip the rest of this subsection.

Option 1: *pdflatex*

To compile a `.tex` document that uses citations in `bibtex` format (see Section 6) we must compile the document in a specific order. For example, using the `thesis.tex` file in used to produce this document:

First run:

```
pdflatex -interaction=nonstopmode thesis.tex
```

which will compile the document, tabulate any citations and cross-references and include images.

Next run:

```
bibtex.exe thesis
```

which extracts bibliographic data from `./bibliograph/thesisbib.bib` and combines it with the `thesis.aux` file produced in the first step.

Then run:

```
pdflatex -interaction=nonstopmode thesis.tex
```

again. This step combines the bibliographic data from the first step with the document and tabulates the bibliography of the document. At this point, everything is done *except* the numbering of the items in the bibliography. To fix this run the previous step again, and you will be left with a PDF containing a cross-referenced thesis.

Option 2: *texify* then *dvipdfm*

You have options other than option 1. However, getting a PDF document out of the process can be more difficult.

For example, the `texify` command, runs all of the previous commands in one call, automatically tabulating everything. The downside is that a `.dvi` ("**D**e **V**ice **I**ndependent") file is produced instead of a PDF. You then must convert the `.dvi` to a PDF if you desire. First, run

```
texify thesis.tex
```

which produces the `dvi` file. Then run:

```
dvipdfm thesis
```

to convert to PDF. Unfortunately, this process does NOT preserve hyper-linking in the document, so you must use Option 3 if you wish to have this feature (which you really should!).

Option 3: Using *texify* then *GhostView*

If you'd like to take advantage of `texify` and its compile-one interface, you may be disappointed that `dvipdfm` doesn't preserve your hyper-linking. However, (on MS Windows) you can use another program that will preserve hyper-linking: GhostView. To start, go out and download GhostScript from:

```
http://www.cs.wisc.edu/~ghost/
```

Now use `texify` to compile your document. Then take your `.dvi` file and convert it into a PS (PostScript) file using:

```
dvips thesis.dvi
```

which will produce a `thesis.ps` file. Now convert the PS file into a PDF using the command-line interface of GhostScript. Assuming GhostScript is installed at:

```
C:\Program Files\gs\gs8.54\bin\gswin32.exe
```

run (all on one line) the following

```
C:\Program Files\gs\gs8.54\bin\gswin32.exe -dBATCH -dNOPAUSE
    -q -sDEVICE=pdfwrite -sOutputFile=thesis.pdf thesis.ps
```

to produce a beautiful document with cross-referencing, hyper-linking and so on.

HINT: You can also automate this process in TeXnicCenter by defining your own "output profile". The `texify` command becomes your primary command and then tack the DVI-to-PS and PS-to-PDF conversions as post-processors (in that order).

1.3 Using Macs

Aside from TEX Live, there is a Mac-only system called `TeXShop` which can be obtained from `http://www.uoregon.edu/~koch/texshop/`. If you use `TeXShop` and want to expand this section for the benefit of other Mac users, please e-mail Jim Diamond to discuss getting your contribution into this document.

1.4 Tutorials, etc.

Finally, there are *a lot* of tutorials on the web explaining how to write in LATEX/TEX. Try searching for `latex tutorial` using your favourite internet search engine and take a look at the first few results. These sites move around, making it difficult to keep a list of them up to date (so we are going to make you look for yourself).

There is a USENET newsgroup called `comp.text.tex` inhabited by TEX and LATEX wizards (as well as mere mortals). You can get help there for tough questions, but before submitting questions to that newsgroup you should search the newsgroup archives first, since it is unlikely you are the first problem to have any given TEX- or LATEX-related problem.

At the web site `http://tex.stackexchange.com/` you can find answers to many common (and uncommon) TEX and LATEX questions.

1.5 Quick Start Guide

This is a long document; it started out somewhat shorter, but as sections were added to give more explanation and help, it grew. You may be concerned that you have to read the whole thing before you can start typing up your thesis, but that's not true. You can get going with just the information in this chapter and the next one. When you want to learn a bit more, you can use the table of contents of this document to zoom in on what you want, or you can skim through the whole document from beginning to end when you have a spare half hour; later, when you have a specific need, you can carefully re-read the corresponding chapter or sections.

In summary, to get going:

1. Install a TEX system on your computer.
2. If you want to use a GUI interface, download and install one of the suggested TEX GUIs (or maybe you will find another one not described here).
3. Copy all the files from the same place you found this PDF file into a directory ("folder") on your computer. At time of writing, all the files are in `http://cs.acadiau.ca/~jdiamond/tex-reference-material/honours-thesis-template/`.
4. As described in Chapter 2, edit the file `preliminaries.tex` to personalize it for your name, your department or school, your thesis title, and so on.
5. (Optional, but you really should.) Put your name, thesis title and keywords in the "hypersetup" in `header.tex`. When you create a PDF file, these pieces of information are put into the internal PDF document description.
6. Edit the file `thesis.tex` to do two things:
   (a) delete the lines between "START NOTE" and "END OF START NOTE" (that text produces the first two pages of the sample thesis you are reading); and
   (b) replace the lines like `\include{gettingstarted}` with lines which include the files with *your* writing.
7. Write your thesis.

TEX and LATEX are extremely powerful and sophisticated tools. Many students have used this template to produce a nicely-prepared thesis without learning any more about

TEX or LATEX. Should you decide to use LATEX to write papers, letters, or other documents in the future, you will discover that there are many packages for LATEX to help you do these things. And if you are in a department that requires you to stand up at the front of a room and present your thesis, you may want to learn about the LATEX "`beamer`" package, which takes input very similar to what you will write for your thesis, but instead formats it for presentations.

Finally, we hope this document provides 99% of the information needed by a beginner to create his or her thesis using LATEX. If you decide there is something else we should really talk about, please e-mail Jim Diamond (or go and see him in person). Grammatical improvements from English majors are also welcome.

Chapter 2

Introduction to the Acadia Style

The Acadia thesis style relies on the file `acadia-hon-thesis.sty`. This file needs to be included where LaTeX can find it. The simplest thing to do is to put it in the same directory as the rest of your thesis.

To customize your thesis with your own name and the name of your supervisor, convocation date, etc., simply rely on the predefined commands in the style. Your information is inserted into your document by editing the file `preliminaries.tex`. Once you have filled in your details the template will format it appropriately. NOTE: if you *must* update the template you will have to edit the `acadia-hon-thesis.sty` file. And if you have to edit it because there is a problem or Acadia regulations have changed, please let Jim Diamond know, so he can fix up the public version.

We now discuss the relevance of the contents of the `preliminaries.tex` file.

`\title{Thesis Title}`: insert the title of your thesis between the open and close braces of the `\title` command.

`\author{NAME IN FULL}`: use this to set your name.

`\dept{Your department}`: use this to set the department or school (e.g., "Computer Science", "Mathematics", ...)

`\deptOrSchool{whichever it is}`: use this to indicate whether you are in a "Department" or a "School". Don't forget to capitalize! (And don't use quotation marks.)

`\submissionMonth{month name}`: enter the month of your thesis submission ("March" for most honours students).

`\submissionYear{20XX}`: Enter the year of your thesis submission.

`\copyrightYear{20XX}`: the copyright year of your thesis. (Just in case it is not the same as the submission year for some reason.)

`\supervisor{Dr.~Your Supervisor}`: your supervisor's name. Use a "~", **not** a space, between your supervisor's title and first name (as well as the following dignitaries), for reasons explained later in this document.

`\cosupervisor{Dr.~Your Other Supervisor}`: if you required lots and lots of supervision, enter your co-supervisor's name.

`\headOrDirector{Dr.~The Head or Director}`: the current head of your department or director of your school. If he or she is an "acting director" or "acting head", remove the `%` from the line "`% \justActing`" in the file `preliminaries.tex`.

`\honoursCommittee{Dr.~The Honours Committee Rep}`: the current honours committee person, whose name appears on the signature page of your thesis. Part of your research effort is figuring out who this is. Seriously.

Next we begin the sections that create some output for you and your readers to admire.

`\firstThreePages`: this command instructs the template to output the first three pages (duh) of your thesis.

`\Acknowledgments`: this begins the acknowledgments section. Be eloquent and yet concise.

`\tocAndSuch`: this instructs the template to output the table of contents, the list of figures, the list of tables, and so on.

`\prefacesection{Abstract}`: this begins your abstract.

`\afterpreface`: this instructs the template to (1) output the table of contents, and, as appropriate, the list of figures, list of tables, list of algorithms and list of listings, (2) change the page numbering from Roman numerals to Arabic, and (3) prepare to start Chapter 1.

That's it! Once you have filled in this template you can start typing your thesis and the messy details are taken care of for you.

Chapter 3

Formatting Points of Interest

3.1 One or Two Sides?

You might have noticed that the left margin on odd pages is much larger than the left margin on even pages. This positions the text on the page to allow the thesis to be printed on both sides of the paper and bound.

If for some reason you wish the margins to be the same on both sides of the page, look in `header.tex` for the line `\documentclass[12pt,twoside,openright]{report}` and change it to `\documentclass[12pt]{report}`.

You may have also noticed that chapters (as well as the abstract, the list of figures, and so on) always start on a right-hand (i.e., odd-numbered) page. If you want a two-sided document but you are OK with having a chapter start on a left-hand page (seriously??), remove "`,openright`" from the "`\documentclass`" line.

If, when creating a two-sided document, you want the signature page to be on the back of the title page (which is just so wrong), you will need to break (sic) things in the `\firstThreePages` command in the file `acadia-hon-thesis.sty`.

3.2 Emphasis

In LaTeX and TeX spacing and text "prettying" (e.g., *italics*, **bold**, etc.) are integral parts of properly typesetting a document. In a word processor, such issues are often left up to

the user, however, it is often advantageous to let the typesetter do it for you. For example, given the text (from MacBeth):

> They met me in the day of success: and I have
> learned by the perfectest report, they have more in
> them than mortal knowledge...

how would you *emphasize* a certain word? In a word processor you might choose to *italicise* it using `\textit{success}`:

> They met me in the day of *success*: and I have
> learned by the perfectest report, they have more in
> them than mortal knowledge...

But what if the quotation was instead written

> *They met me in the day of success: and I have*
> *learned by the perfectest report, they have more in*
> *them than mortal knowledge...*

Now *italics* has no clear emphasis at all. In LaTeX, we use the `\emph{success}` command to indicate emphasis. If we emphasize *success* in the previous quotations we get:

> They met me in the day of *success*: and I have
> learned by the perfectest report, they have more in
> them than mortal knowledge...

but if the quotation was itself italicised

> *They met me in the day of* success*: and I have*
> *learned by the perfectest report, they have more in*
> *them than mortal knowledge...*

`emph{...}` actually *de-italicises* the enclosed text, which ensures that it nonetheless stands out. In a word processor we would have to manually change the style of emphasis, whereas LaTeX takes care of it for us automagically.

3.3 Spacing

Spacing is another contentious issue in LaTeX. The compiler assumes that most spacing is simply used to aid in reading the LaTeX code, rather than to provide whitespace in the typeset document. For example, the simple text (where we use “␣” to represent a space):

```
‘‘We␣write␣text␣on
␣␣␣␣multiple␣lines␣with␣␣␣␣random␣spacing␣␣␣␣␣␣between
␣words,␣and␣\LaTeX␣does␣the␣right
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣thing␣and␣groups␣it
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣all␣together.’’
```

results in

“We write text on multiple lines with random spacing between words, and LaTeXdoes the right thing and groups it all together.”

How about the space after “\LaTeX”? Well, this raises an important point. Sometimes LaTeX does *not* seem to do the right thing. In this instance, it has “eaten” the space after the command. In cases like this we must enforce the space using \␣, where “␣” indicates a single space. We can also use this to force the compiler to leave a number of spaces. Let’s revisit our example, with this in mind:

```
‘‘We␣write␣text␣on
␣␣␣␣multiple␣lines␣with\␣\␣\␣\␣random␣spacing␣␣␣␣␣␣between
␣words,␣and␣\LaTeX\␣does␣the␣right
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣thing␣and␣groups␣it
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣all␣together.’’
```

becomes:

“We write text on multiple lines with   random spacing between words, and LaTeX does the right thing and groups it all together.”

Observe that there is now a proper amount of space after “LaTeX”. (Note: using constructs like “\␣\␣\␣\␣” is usually not the right way to leave some space in the middle of a line. A much better way is to use something like “`dog\hspace{1 in}cat`”, which produces “dog   cat”, keeping the dog exactly one inch from the cat.)

LaTeX and TeX were not designed to be WYSIWYG (What You See Is What You Get) systems, and with some experience using these systems most people realize that is a Good Thing. But what if we had actually wanted to put a line-break in this text? We can either use the line-break delimiter \\, or put two carriage returns in the text, depending on exactly what we want. Both LaTeX and TeX use the convention that two consecutive carriage returns (i.e., a blank line) denotes the start of a new paragraph. Depending on the style of your document (such as the Acadia thesis style), the first line of a new paragraph may be indented, and there may be extra (vertical) space between the two paragraphs. (And, in some styles which normally indent the first line of a paragraph, a paragraph immediately following a section heading is not indented. If you want to know why this would be, talk to your local typographer.)

For example, in:

```
``We␣write␣text␣on
␣␣␣␣multiple␣lines␣with\␣\␣\␣\␣\␣random␣spacing␣␣␣␣␣␣between

words,␣and␣\LaTeX\␣does␣the␣right
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣thing␣and␣groups␣it
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣all␣together.''
```

the blank line after "between" prompts the compiler to indent the next line beginning with "words…":

> "We write text on multiple lines with   random spacing between
> words, and LaTeX does the right thing and groups it all together."

To indicate that you don't want to indent the first line of a new paragraph, use the `\noindent` command to start the new paragraph:

```
``We␣write␣text␣on
␣␣␣␣multiple␣lines␣with\␣\␣\␣\␣\␣random␣spacing␣␣␣␣␣␣between

\noindent
␣words,␣and␣\LaTeX\␣does␣the␣right
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣thing␣and␣groups␣it
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣all␣together.''
```

becomes:

"We write text on multiple lines with random spacing between
words, and LaTeX does the right thing and groups it all together."

Some people get confused about when to use a blank line and when to use \\. Simply, use a blank line if you are logically starting a new paragraph, and use \\ if you want to start a new line within the current paragraph. Do NOT use \\ at the end of a paragraph to get extra space. There are many ways to get extra vertical space; an example of one such command is "`\vspace{0.5 in}`" which terminates the current paragraph (if you give this command inside a paragraph) and then puts out 0.5 inches of vertical space.

Having said that, there may be an unusual situation in which you want to start a new line in the middle of a paragraph, and in such cases you can use "\\". Also, when creating tables it is not uncommon to use "\\" (see Chapter 8).

3.4 Controlling automatic spacing and hyphenation

Sometimes you want to make sure that the spacing in your code is the spacing that is used during the actual typesetting. For example, suppose we have a very long name that we don't want to be broken at the end of a line (hyphenated):

"... the building blocks of life are often said to be the chemicals Deoxyribonucleic acid (DNA) and Ribonucleic acid (RNA), but actually amino acids are used to build DNA and RNA."

Let's say we want to ensure that no matter what happens, "Deoxyribonucleic acid (DNA)" and "Ribonucleic acid (RNA)" are not split over lines. To do this we use the non-breakable space "~" and a command to temporarily turn off hyphenation as follows:

```
``\dots\ the building blocks of life are often said
to be the chemicals {\hyphenchar\font=-1 Deoxyribonucleic~acid~(DNA)}
and  {\hyphenchar\font=-1 Ribonucleic~acid~(RNA)}, but actually
amino acids are used to build DNA and RNA.''
```

which produces

"... the building blocks of life are often said to be the chemicals Deoxyribonucleic acid (DNA) and Ribonucleic acid (RNA), but actually amino acids are used to build DNA and RNA."

Of course in this example the results are rather ugly because "...Deoxyribonucleic acid (DNA)..." now hangs over the end of a line, but you get the point. You might want to re-word your sentence (in the final version of your thesis) to avoid this problem.

Chapter 4

Sectioning

LaTeXprovides a great deal of flexibility in creating and naming sections; for instance, we used "`\chapter{Sectioning}`" command to create this section. After the main chapters of your thesis have been included, you can use the command `\appendix` to indicate that all subsequent chapters should be called appendices. For example,

```
\chapter{A chapter}
Chapter contents...
\appendix
\chapter{An Appendix}
Appendix Contents...
```

will ensure that the chapter titled "An Appendix" will be referred to as Appendix A in the document. See Appendix A, for example.

We can also create many other types of sections, as seen here:

4.1 A Sample Section

Sectioning is an important organization tool in your thesis. In LaTeX and TeX (with "eplain") there is built-in support for sectioning along with appropriate section numbering.

For example, this section was introduced with `\section{A Sample Section}`.

4.1.1 A sample SUBSECTION

Subsection text, which was introduced with `\subsection{A sample SUBSECTION}`.

And a SUBSUBSECTION

Note that in the Acadia thesis style, there is no number for a sub-sub-section. This is the text of the sub-sub-section, which was introduced with the somewhat longer command `\subsubsection{And a SUBSUBSECTION}`.

PARAGRAPH 1 We can also create paragraphs with paragraph titles. This was introduced with `\paragraph{PARAGRAPH 1}`.

SUBPARAGRAPH Label And we can even do a sub-paragraph, which was introduced with `\subparagraph{SUBPARAGRAPH Label}`. Yet more subparagraph text.

4.2 Custom "Environments"

You can also create so-called "environments" which are treated specially by LaTeX. Later in this document you can find information about (for example) figures and tables. These environments are typeset specially, are numbered, and can be referenced from other parts of the document.

This section has a demonstration of how you can set up a "definition" environment. Mathematicians (or others, for that matter) might want to create a lemma environment, a corollary environment, and so on. The idea explained here can be used in a wide variety of ways.

For example, two definitions of baked goods (specifically, cookies) follow. In order to number these definitions (just as figures and tables are numbered) and to be able to reference the definitions from other places in your thesis, you can define a "definition" environment.

To create the custom "definition" environment we included the following code in the `header.tex` file:

```
\theoremstyle{definition}
\newtheorem{definition}{Definition}[chapter]
```

The command `\newtheorem{definition}{Definition}[chapter]` creates an environment called `definition` which uses the text `Definition` to introduce a definition, where the definition numbers are reset every chapter, and the formatting is derived from the "theorem" environment (which is itself defined by the `amsthm` package).

The command `\theoremstyle{definition}` is used to indicate that the following `\newtheorem` command will use the "definition" style. (To find more about these things read the `amsthm` documentation.) Note that if you are creating (say) a corollary environment, you might add the following two lines to `header.tex`:

```
\theoremstyle{definition}
\newtheorem{corollary}{Corollary}[chapter]
```

In particular, note that the `\theoremstyle` is still "definition".

Having set that up once in `header.tex`, we can now create definitions as shown below. Notice that the first one includes the command `\label{def:COOKIE}` so that the second definition (or other parts of your document) can refer to it by using `\ref{def:COOKIE}`. This is *much* better than explicitly using "4.1" in your text; if you revise your document later and add a new definition before this one, LaTeX will automagically renumber not only your definitions, but also the references to the definitions. The careful reader will also notice that instead of carelessly writing the obvious "`(Definition \ref{def:COOKIE})`" we instead took the extra tiny amount of time to write "`(Definition~\ref{def:COOKIE})`". The "`~`" is a *non-breakable* space; this means that the word "Definition" won't appear at the end of one line, leaving "4.1" all by itself at the beginning of the next line.

Our first usage:

```
\begin{definition}
    \label{def:COOKIE}
        A \emph{cookie} is a round baked good that is tasty,
        sugary, and usually bad for you. See~\citep{COOKIE}.
\end{definition}
```

creates the example:

Definition 4.1. A *cookie* is a round baked good that is tasty, sugary, and usually bad for you. See [2].

The code:

```
\begin{definition}
    \label{def:PBCOOKIE}
        A \emph{peanut butter cookie} is a \emph{cookie}
        (Definition~\ref{def:COOKIE}) which contains, at a minimum,
        the ingredients peanut butter, flour, sugar and eggs.  Peanut
        Butter cookies are said to be \emph{delicious}.
\end{definition}
```

creates:

Definition 4.2. A *peanut butter cookie* is a *cookie* (Definition 4.1) which contains, at a minimum, the ingredients peanut butter, flour, sugar and eggs. Peanut Butter cookies are said to be *delicious*.

4.3 Cross-Referencing Sections

The keen observer will have noticed that we used a cross-reference when defining what a Peanut Butter cookie was: "A *peanut butter cookie* is a *cookie* (Definition 4.1)..." this feature is extremely useful since it allows us to refer to other sections of the document using section numbering.

Let's say we want to reference our section on sections. We have defined the section using the command:

```
\section{SECTION}
\label{sec:ONE}
```

which creates a "section" called "SECTION" and associates a label "`sec:ONE`" with this section. Now I can reference this label from elsewhere in the document. For example, the input

```
As shown in Section~\ref{sec:ONE}, we can create different\dots
```

would produce:

As shown in Section 4.1, we can create different...

in the document.

Finally, the numbering of a cross-reference reflects the numbering of the actual label in the document. Therefore, if the document's number is updated, the cross-reference will also be updated to suit. This is *much* easier than trying to maintain cross-referencing manually.

Chapter 5

Acronyms and Lists

5.1 Defining an Acronym Command

TEX and LATEX have very powerful facilities for creating your own commands. So powerful, in fact, that discussing them is well out of the scope of this document. However, we will present one simple example here which you may find useful.

Suppose you have some long word or phrase which is used frequently in your thesis (such as "deoxyribonucleic acid"), but you don't want to type it over and over. If you put the following command in your document header (`header.tex` in this sample thesis)

```
\def\dna{deoxyribonucleic acid}
```

then you can just type `\dna` and LATEX will expand that into what you really wanted. For example, if you now have this text

```
I try to eat some \dna\ every day.
I like \dna.
```

you will get this output:

I try to eat some deoxyribonucleic acid every day. I like deoxyribonucleic acid.

There are LATEX add-on packages to do all sorts of things for you, and one such package handles acronyms in a much more flexible way. For example, you might want to define an acronym so that the first time it is used you get something like "Turing Machine (TM)" and

from then on the acronym expands to just "TM". If this possibly interests you, try doing a search at the Comprehensive TeX Archive Network (CTAN) (`http://www.ctan.org`).

One final note about this... as mentioned elsewhere, LaTeX "eats" spaces following commands such as "`\dna`". In order to get a space, the first "`\dna`" needs to be followed by something which causes the space to be preserved. In the second "`\dna`", it is immediately followed by punctuation ("`.`" in this case) and so there is no need to do anything special. If you have a hard time remembering this and don't mind a bit of extra typing, you could always write "`{}`" after commands, as follows:

```
I try to eat some \dna{} every day.
I like \dna{}.
```

which will produce this output:
I try to eat some deoxyribonucleic acid every day. I like deoxyribonucleic acid.

5.2 Examples of Lists

There are numerous examples of places that we might want to use a list. We might use a described list, which is begun with `\begin{description}`, ended with `\end{description}`, and where each item is introduced with `\item[...text...]`. The "`...text...`" for the first item below is "`First Item`".

First Item This thesis talks about SPECIALACRONYM.

Second This thesis talks about XML [3].

Final Point This thesis talks about the future!

Or we might want to use a bulleted list, started with `\begin{itemize}`, and each `\item` then has no `[...text...]`.

- This thesis talks about SPECIALACRONYM.
- This thesis talks about XML [3] by Head et al. 3.
- This thesis talks about the future!

Or we might use a numbered list, where each item is introduced with `\begin{enumerate}`, and each `\item` again has no `[...text...]`.

1. This thesis talks about SPECIALACRONYM.
2. This thesis talks about XML [3] by Head et al. 3.
3. This thesis talks about the future!

The point is that we have a great deal of flexibility.

One thing that you might not like about the above lists is that there are lots of blank lines. If we add `\vspace{-\topsep}` *before* the `\begin{itemize}` line and *after* the `\end{itemize}` line and

```
\setlength{\itemsep}{0pt}
\setlength{\parskip}{0pt}
\setlength{\parsep}{0pt}
```

after the `\begin{itemize}` line, the above bulleted list will look like this:

- This thesis talks about SPECIALACRONYM.
- This thesis talks about XML [3] by Head et al. 3.
- This thesis talks about the future!

that is, with no extra space before, after, or between list items.

An alternative approach is to define your own list environment, which takes care of the space before the list and between items.

```
\newenvironment{nicelist}
{
    \vspace{-\topsep}
    \vspace{-\parskip}
    \begin{itemize}
        \setlength{\topsep}{0pt}
        \setlength{\itemsep}{0pt}
        \setlength{\parskip}{0pt}
        \setlength{\parsep}{0pt}
}
{
    \end{itemize}
}
```

With this definition, the following code

```
That produces the following text:
\begin{nicelist}
    \item This is the first item in my list.
    \item This is the second.  It is more than one line long.
    Not that it is an interesting item, but there you go.
    \item Third item.
\end{nicelist}
```

That produces the following text:

- This is the first item in my list.
- This is the second. It is more than one line long. Not that it is an interesting item, but there you go.
- Third item.

This doesn't take care of automatically removing the extra space after the list, in the case where your paragraph continues after the last list item. Budding LaTeX gurus may prefer to read about and then use the `enumitem` package, which makes configuring the look of your lists a lot easier to do.

Chapter 6

Citations

Scholarly citations can be a bit of an ugly issue, in that different disciplines have different ideas about how references should appear both in the body of a document as well as in its bibliography. And, for that matter, whether there should be one bibliography at the end of the document, or whether there should be one at the end of each section or chapter.

Here are some reference styles you may come across; there are many more.

- [12]
- Aardvark et al. [12]
- Aardvark et al. (2007)
- (Aardvark et al. 2007)
- Aardvark et al. (2007, Chap. 22)
- Aardvark, Bloggins, Crumpet and Domingo [12]
- [ABCD07]
- (Zhang 1990, Zhu 1991, Zhao and Zou 1992, and Zhong, Zheng and Zeng 2001)

You could just type the citation reference in the body of your thesis wherever you want it, but there are two problems with that. First, if you use "[12]" and later introduce another citation which comes before that one in the bibliography, you will have to re-number everything. Second, if you wish to submit your work to a journal or conference, changing all of the references by hand is tedious and error-prone.

To help deal with this, we can use a tool called BibTeX to manage the complexity. This thesis template is set up to use this tool is a fairly simple way; if, in the future, you start

publishing many documents with the use of BibTeX, you will want to spend a few minutes learning more of its capabilities.

To get going quickly, you can put your cited works in the file `thesisbib.bib`; the format of this file is somewhat strict, and you might need to do an on-line search to see how to format a citation for a particular document type, if you can't work your situation out from any of the examples found therein.

This file is searched when LaTeX looks for a citation. (Actually, it is processed when the `bibtex` command is run, but that's a fine point you probably don't need to worry about.) You may keep all of your bibliographic entries in this file, even if you aren't using them for your thesis; unused entries will be silently ignored.

The "normal" citation capabilities of LaTeX can be lacking in functionality. For instance, [2, TEST] doesn't tell us anything about the "cookie" article within the paper and we have to manually track authors, etc. To deal with this, this thesis template uses the `natbib` citation package, combined with the `plainnat` style. (Do a search for `natbib.pdf` to find out more information on this package.)

By default, LaTeX uses `\cite` to put a citation into the body of the document, but `natbib` also provides `\citep` (parenthetical) and `\citet` (textual). Some variations on these are shown in Table 6.1. In Table 6.2 examples of citing multiple references (in the same place) is shown. In Table 6.3 we show how the citations can be "dressed up" with some surrounding commentary.

Table 6.1: Citation commands and results, in parenthetical or textual style.

Command	Result
`\cite{Purchase}`	[5]
`\citep{Purchase}`	[5]
`\citep[chap. 2]{Purchase}`	[5, chap. 2]
`\citep[see][]{Purchase}`	[see 5]
`\citep[see][chap. 2]{Purchase}`	[see 5, chap. 2]
`\citep*{Purchase}`	[5]
`\citet{Purchase}`	Purchase et al. [5]
`\citet[chap. 2]{Purchase}`	Purchase et al. [5, chap. 2]
`\citet*{Purchase}`	Purchase, Cohen, and James [5]

With the reference style used by this sample thesis, the items in the bibliography are given meaningless numbers and show up in your text like "[42]", which probably tells the reader nothing, forcing him or her to look to the back, should they want to see

Table 6.2: Multiple citations.

Command	Result
`\cite{Purchase,HHG2TG}`	[5, 1]
`\citep{Purchase,HHG2TG}`	[5, 1]
`\citep{Purchase,Kaufmann}`	[5, 4]
`\citep{Purchase,Purchase2}`	[5, 6]
`\citet{Purchase,HHG2TG}`	Purchase et al. [5], Adams [1]
`\citet{Purchase,Kaufmann}`	Purchase et al. [5], Kaufmann and Wagner [4]
`\citet{Purchase,Purchase2}`	Purchase et al. [5, 6]

Table 6.3: Citations with Optional Text.

Command	Result
`\cite[chap. 2]{Purchase}`	[5, chap. 2]
`\cite[see][chap. 2]{Purchase}`	[see 5, chap. 2]
`\citep[chap. 2]{Purchase}`	[5, chap. 2]
`\citep[see][]{Purchase}`	[see 5]
`\citep[see][chap. 2]{Purchase}`	[see 5, chap. 2]
`\citet[chap. 2]{Purchase}`	Purchase et al. [5, chap. 2]
`\citet[see][chap. 2]{Purchase}`	Purchase et al. [see 5, chap. 2]

whose work is being referenced. However, by changing "`\bibliographystyle{plaintat}`" to "`\bibliographystyle{alpha}`" the above reference might show up as "[Ada79]". If you want to experiment with different styles, just change the settings in `header.tex` and recompile the document.

Normally with LaTeX when you use citations such as these, in the PDF output you will see green boxes around links to the bibliography, red boxes around other "internal" links and blue (or is it cyan?) boxes around external links. Jim Diamond thinks these are horribly ugly and turned this mis-feature off in the Acadia thesis styles. Notice that you can discover whether something which looks like it could be a link is, indeed, a link, by hovering the mouse cursor over the text for a second or two. If for some reason you wish to uglify your PDF output by including these, you can change it to look how you like. Since that is such a bad idea, we are not going to tell you how to do it.

Chapter 7

Creating and Including Graphics

They say a picture is worth 1000 words. That is probably true of a good picture, but you don't want to go overboard with pictures just to fill up space. After a while, your readers will catch on to the scam you are trying to perpetrate.

Before we discuss how to do get pictures, graphs, and other graphical items into a LaTeX document, a tiny bit of background will be useful.

7.1 Types of Graphics

There are essentially two types of graphics you can include: bitmaps and vector graphics. Bitmaps are found in files with extensions such as `.jpeg`, `.png`, `.gif`, and the laughably horrid `.bmp`. In bitmap images, the information has already been chopped up into individual units called pixels. When you show a bitmap image on the screen, or print a bitmap image on a printer, if there is not an exact one-to-one correspondence between the image pixels and the output device pixels, the image has to be scaled up or down, which typically reduces the quality of the image and gives poor results. You can not always avoid these image types, but it is a good idea to avoid them whenever possible.

The compression used in `.jpeg` files is designed for photographic (and similar) images, and does not work well for line drawings or textual material. If you must use a bitmap image, use JPEGs for photos and PNGs for other material.

Unlike bitmap images, vector graphics are lists of "instructions" telling the displaying program where to draw lines, curves and points. Typically these will scale up and down

very well, and therefore these are usually a much better choice than bitmap graphics. For X-Y plots (for example), you should use some tool which creates vector graphics, if at all possible.

The inventor of TEX (Donald Knuth) made a conscious decision to NOT include graphics features directly into TEX. He knew that such an approach would be doomed to failure, since better, more convenient graphics facilities would be created in the future. But he realized that graphics are important, so TEX has a feature to allow graphics from other systems to be included. More recently some people have designed add-on packages to TEX and LATEX which gives you some powerful graphics capabilities from within TEX and LATEX. Some examples are shown below.

7.2 Including External Graphics

If you have created a graphic which you want to include in your thesis, assuming you have `\usepackage{graphicx}` in (for example) your header file, you can then type

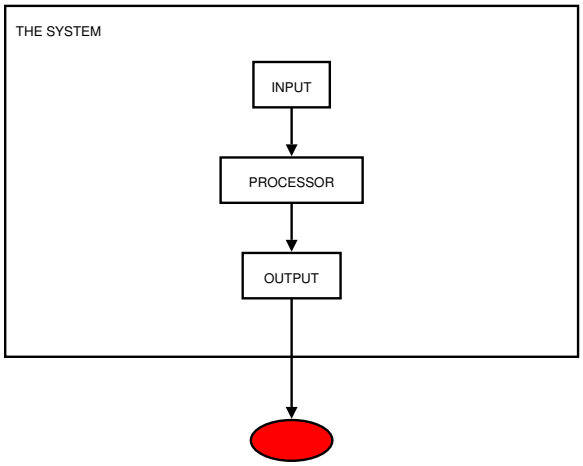
```
\includegraphics[width=WIDTHOFGRAPHIC]{GRAPHIC_LOCATION}
```

Note that the `[width=WIDTHOFGRAPHIC]` option adjusts the width of the graphic to be `WIDTHOFGRAPHIC` in the document. For example, `[width=0.80\textwidth]` will set the width to be 80% of the width of the text at that point in the document. You could also write `width=5in` to make the graphic exactly 5 inches wide. (There is also a `[height=...]` option, should that be more convenient for you.)

The code

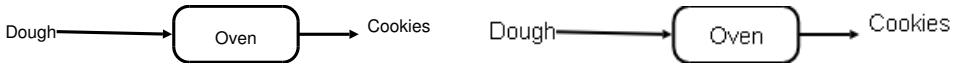
```
\begin{center}
    \includegraphics[width=.50\textwidth]{figures/example}
\end{center}
```

produces the following image:



This image looks rather lonely sitting there by itself. Instructions for attaching a figure number and a caption to such an image are found in Chapter 8.

To show you why you should never, ever, ever use a bitmap image if you have a vector graphics image available, here is an original vector graphics image and the same image converted to a bitmap. Need we say more?



(OK, one of us can't resist: "Just don't do it"; "Friends don't let friends use bitmap images"; "Just say no".) Even if you think your spreadsheet plot looks good on the screen, do not export it as a bitmap or use a screen capture program to grab it.

7.2.1 Graphics File Types

Unfortunately, LaTeX can not use every type of graphics known. Specifically, `latex` can use encapsulated PostScript (`.eps`) files, and `pdflatex` can use `.png`, `.jpg`/`.jpeg` and `.pdf` files.

In the examples above, no file extension was explicitly used, which means that `latex` or `pdflatex` will pick an appropriate choice. How this choice is made is left as a small research project for the diligent student.

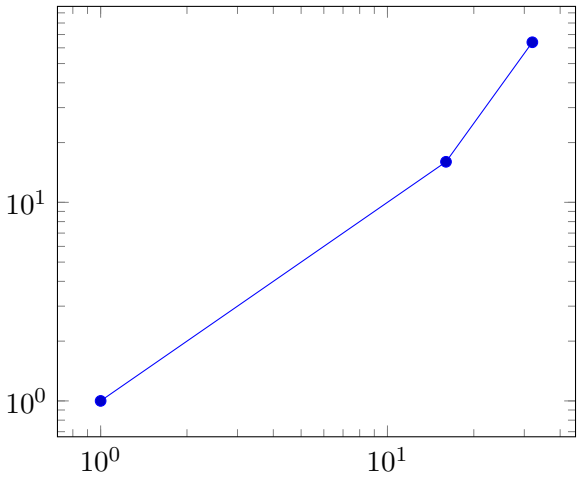
7.3 Internal Graphics

There are two very powerful add-on packages for TEX and LATEX: PS Tricks and TikZ/PGF. PS Tricks is extremely powerful, but can only be used (directly) with `latex`; TikZ/PGF is also very powerful, and can be used with both `latex` and `pdflatex`.

Unfortunately, both of these have a learning curve which is a bit steep. However, just to whet your appetite, here are a few examples using TikZ/PGF. Note that there are tutorials and collections of samples to be found on the internet; for example, look at `http://www.texample.net/tikz/examples/` to see a gallery of sample figures produced using TikZ/PGF.

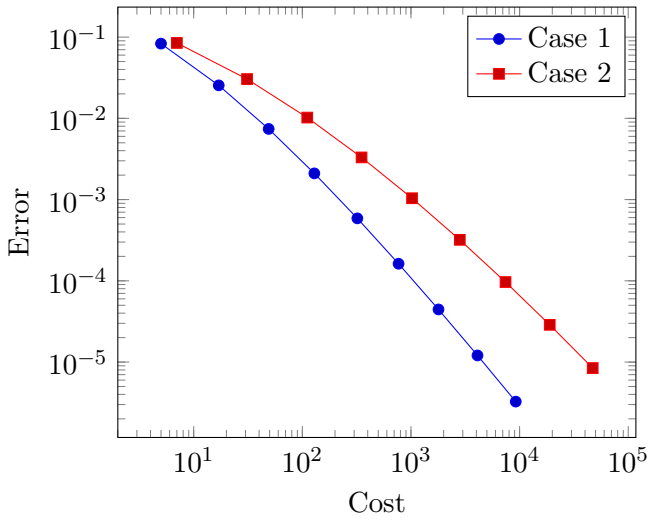
Here is a sample of a log-log plot created with the following code:

```
\begin{tikzpicture}
  \loglogaxis
  \addplot coordinates {
      (1,1)
      (16,16)
      (32,64)
  };
  \endloglogaxis
\end{tikzpicture}
```



One of the nice things about using TikZ/PGF is that you don't have to worry about external graphics files becoming separated from the rest of your document when you move your files from one place to another.

Here is another slightly more complicated example:



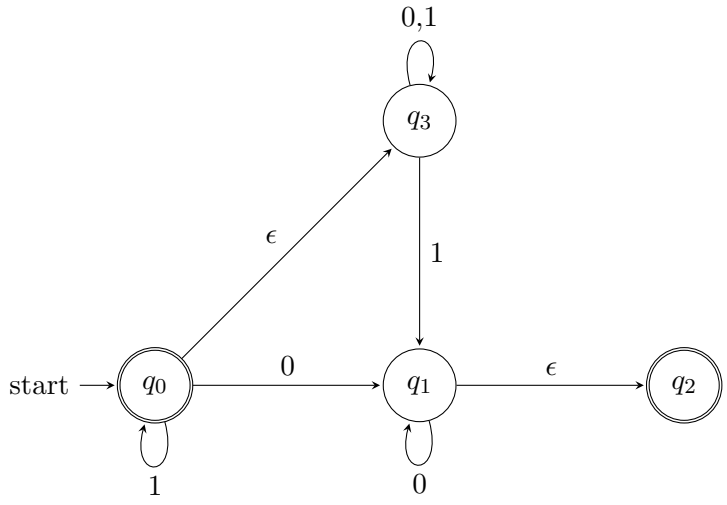
The code to produce that is the following:

```
\begin{tikzpicture}
  \loglogaxis[
      xlabel=Cost,
      ylabel=Error]
  \addplot coordinates {
      (5,     8.31160034e-02)
      (17,    2.54685628e-02)
      (49,    7.40715288e-03)
      (129,   2.10192154e-03)
      (321,   5.87352989e-04)
      (769,   1.62269942e-04)
      (1793, 4.44248889e-05)
      (4097, 1.20714122e-05)
      (9217, 3.26101452e-06)
  };
  \addplot coordinates {
```

```
        (7,     8.47178381e-02)
        (31,    3.04409349e-02)
        (111,   1.02214539e-02)
        (351,   3.30346265e-03)
        (1023,  1.03886535e-03)
        (2815,  3.19646457e-04)
        (7423,  9.65789766e-05)
        (18943, 2.87339125e-05)
        (47103, 8.43749881e-06)
    };
    \legend{Case 1,Case 2}
    \endloglogaxis
  \end{tikzpicture}
```

As you can see, if you have large amounts of data, you might not want to massage it into the format required for TikZ/PGF.

Aside from X-Y plots, TikZ/PGF is very nice for creating graphs such as the following finite automaton:



That was produced with the following code:

```
\usetikzlibrary{automata}
\makeatletter
```

```
\def\tikz@double@width@distance{1.6pt}  % default 0.6
\makeatother

\begin{tikzpicture} [shorten >=1pt,>=stealth,node distance=3.5cm,auto]
    %\draw[help lines] (0,0) grid (3,2);
    \node[state,initial,accepting]     (q_0) {$q_0$};
    \node[state]                       (q_1) [right of=q_0] {$q_1$};
    \node[state,accepting]             (q_2) [right of=q_1] {$q_2$};
    \node[state]                       (q_3) [above of=q_1] {$q_3$};
    \path[->]
        (q_0) edge node {0} (q_1)
              edge [loop below] node {1} (q_0)
              edge node {$\epsilon$} (q_3)
        (q_1) edge node {$\epsilon$} (q_2)
              edge [loop below] node {0} (q_1)
        (q_3) edge [loop above] node {0,1} (q_3)
              edge node {1} (q_1);
\end{tikzpicture}
```

One of the nice things about using a package like TikZ/PGF is that, while there is a learning curve involved, and there is no instantaneous visual feedback as when using a programs such as `xfig`, `dia`, or (*gag*) `visio`, there is also no endless fussing trying to get the lines and circles arranged in a visually pleasing, consistent manner. For example, note that the edges in the automaton graph all point to the centers of their corresponding nodes, rather than just pointing to some random spot on the edge of the node.

A second nice thing about using TikZ/PGF can be observed by reviewing the figures in Section 7.2; the text in these figures does not match the text in the rest of this chapter (both the typeface and the size are wrong). This is a recurring problem when you includes figures or other graphics created using other programs. Depending on the program creating the figure, you may find it either difficult or impossible to make the text consistent. When possible, it is nice to avoid this ugly-ism. Since TikZ/PGF graphics use the same fonts as your document, there won't be these style and/or size mismatches when such graphics are used.

There is a **very** extensive user manual for this package; on a Linux system run the command "`texdoc tikz`" to see the documentation.

As a final example, in Figure 7.1 an example of a (graph-theoretic) graph drawn using TikZ/PGF is shown; note that the textual labels of the graph perfectly match the rest of the text. If you want to see the commands used to create Figure 7.1, look in the `graphics-info.tex` input file. If you need to perk your interest a bit more, look at the examples in the afore-mentioned web site `http://www.texample.net/tikz/examples/`.

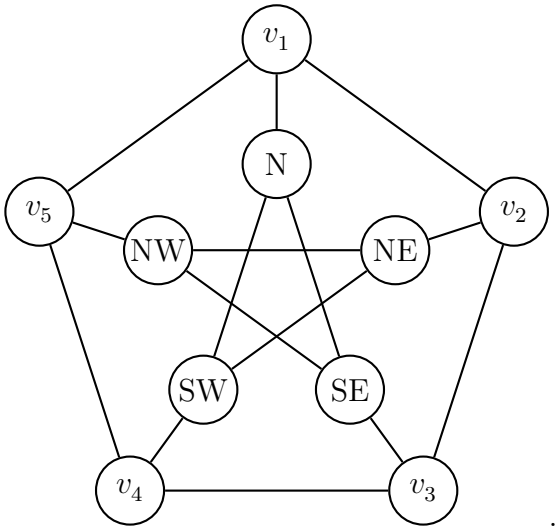


Figure 7.1: The Petersen Graph.

Aside from the matching text size and typeface in Figure 7.1, note the symmetry and precision of the placement of the circles and the fact that the lines are positioned exactly as they should be. Learning a bit of TikZ/PGF to create figures will take a while, but it is worth it. (Conversely, doing a mediocre job with a GUI drawing program is easy, but doing something worthy of your thesis using most GUI tools is much trickier.)

Chapter 8

Figures and Tables

8.1 Floats

A *float* is an object in a document that should not be broken across multiple pages. The term "float" signifies that LaTeX may move a floating object within the document to preserve the flow of the document (e.g., to ensure that the bottom of a page doesn't have a gaping hole simply because the figure does not fit on the page) and to improve the alignment of the object if more than one can fit horizontally on the page.

Understanding what LaTeX will do with a float is important because the system will make decisions about where to place a float automatically; the position of the float in your *source code* may or may not be the same as the position of the float in the document. Learning to trust LaTeX and TeX to do The Right Thing is the first hurdle.

There is much more to know about floats, since packages can influence their placement. A basic overview along with lots more information about figures and tables can be found at `http://www.andy-roberts.net/misc/latex/latextutorial6.html`, but to be sure read the documentation for your particular package to be sure.

For example, notice below that LaTeX moved Figure 8.1 and Table 8.1 to follow the paragraph starting with "We can also create", even though the code for those preceded that paragraph in the `figuresandtables.tex` input file. That is because there is not room for them to fit on that page, so LaTeX "floats" them to a future page where there is room to fit them, without leaving a (possibly) big, sucking, ugly gap at the bottom of the page.

8.2 Figures and Tables

We can use the `figure` environment in order to add a caption and label to a figure so that it is picked up in the "List of Figures". We can also use the label to reference the figure in the text. For example, Figure 8.1 and Table 8.1 were produced using the commands

```
\begin{figure}[ht]
    \centering
    \includegraphics[width=.50\textwidth]{figures/example}
    \caption{An example image.}
    \label{fig:EXAMPLE1}
\end{figure}
```

and

```
\begin{table}[ht]
    \centering
    \caption{Table of values for $F(\chi)=\chi+4$.}
    \begin{tabular}[ht]{r|l}
        \hline
        $\chi$ & $F(\chi)=\chi+4$\\
        \hline
        $1$ & $5$\\
        $2$ & $6$\\
        $3$ & $7$\\
        $4$ & $8$\\
        \multicolumn{2}{l}{$\ldots$}\\
        \hline
    \end{tabular}
    \label{tab:EQUATION1_VALUES}.
\end{table}
```

We can also create *sub*-figures (Figures 8.2a and 8.2b) and *sub*-tables (Tables 8.2a and 8.2b). Look at the source for this chapter to see how it was done, should you want to do it in your thesis.

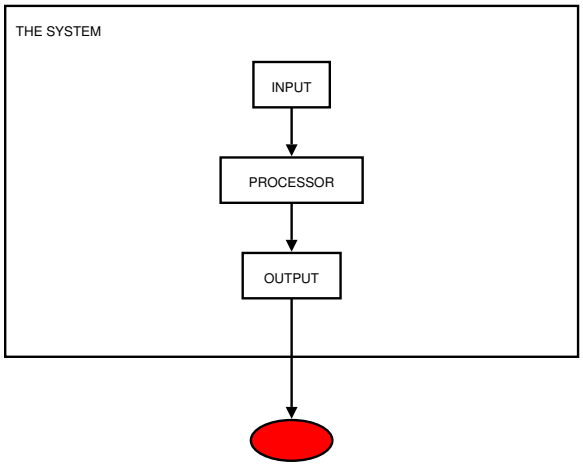
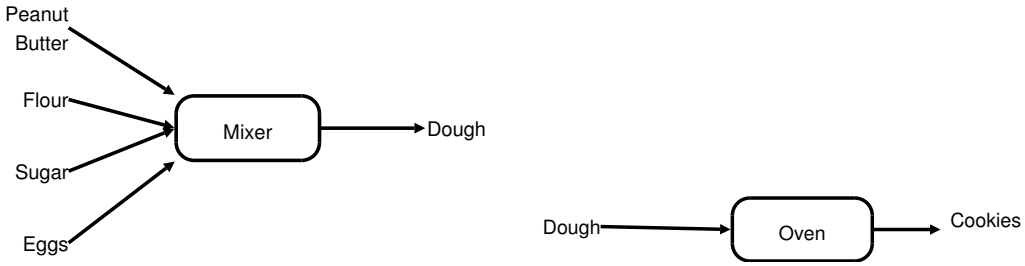


Figure 8.1: An example image.

Table 8.1: Table of values for $F(\chi) = \chi + 4$.

χ	$F(\chi) = \chi + 4$
1	5
2	6
3	7
4	8
...	



(a) Make the Dough (b) Bake the Cookies

Figure 8.2: Making Peanut Butter Cookies.

You may have thought there wasn't enough space above and below the line with $F(\chi)$ in Table 8.1. JD certainly doesn't think there is enough space there, and he feels that the fact that LaTeX's tabular environment doesn't automagically provide enough space is a bit of a flaw. JD uses plain TeX rather than LaTeX, and thus doesn't know The One True LaTeX Way to fix this, but he would solve the problem by putting the definition

Table 8.2: Table of values for two other $F()$s.

(a) $F(z) = x + y$

x	y	$F(z) = x + y$
1	2	3
1	3	4
1	4	5
1	5	6
1	6	7

(b) $F(z) = x + 2y$

x	y	$F(z) = x + 2y$
1	2	5
1	3	7
1	4	9
1	5	11
1	6	13

.

```
\def\tallstrut{\vrule width 0pt depth 5.5 pt height 12 pt}
```

before the definition of the table and then changing "`$\chi$ & $F(\chi)=\chi+4$\\`" in the table definition to read "`\tallstrut $\chi$ & $F(\chi)=\chi+4$\\`". This produces the result shown in Table 8.3. The `depth 5.5pt` and `height 12pt` tell TEX to make sure there is at least 5.5 pt. of space below the typesetting baseline and 12 pt. of space above the baseline. If you decide to go to this length to make your thesis look good (and why shouldn't you?), keep in mind that you may wish to change these numbers, depending on what characters are actually used on the line you are prettying up.

Table 8.3: Table of values for $F(\chi) = \chi + 4$.

χ	$F(\chi) = \chi + 4$
1	5
2	6
3	7
4	8
...	

.

Finally, you should note that figures and tables should **always** go **after** the first reference to them in the text. The "`[ht]`" in the "`\begin{table}`" line indicates that you are happy with the figure going `h`ere or at the `t`op of a following page. (You can also use "`b`" for the `b`ottom of the page.) Sometimes LATEX puts the float further away than necessary; if this happens to you try changing "`[ht]`" to "`[!ht]`". If you examine the source code for this chapter you will see this was done in a couple of places.

Chapter 9

Some Fine Points of Typesetting

No-one expects an honours or masters thesis to be a world-class example of typography (unless, I suppose, you are studying that discipline). However, that doesn't mean you shouldn't put a tiny bit more effort in to make it look as nice as possible. TEX already takes care of many of the details for you, but you can help TEX out every now and then.

9.1 Non-Breakable Spaces

In some situations you may want to tell TEX to *not* put a line-break between two particular words. For example, you would not want "Section" to be at the end of one line and "3.2.1" to be at the beginning of the next line. Similarly you don't want "Figure", "Table", "Algorithm", "Listing", and so on at the end of one line and the corresponding number at the beginning of the next line. (Especially if the next line is at the top of the next page!)

Any time you want to keep to words together, use a "~" instead of a space between the two words. For example, earlier you saw "`Definition~\ref{def:COOKIE}`"; the "~" ensures that the reference number for COOKIE (4.1) does not end up at the beginning of a line.

To avoid having the last word in a paragraph on a line by itself, but a ~ between the last two words in the paragraph. Caution: with a **very** short paragraph, this may do more harm than good. But you should avoid very short paragraphs anyway (in most situations).

9.2 Inter-Sentence Spacing

In English typography, it is customary to have more space between the word at the end of a sentence and the word at the beginning of the next sentence. Look at the first paragraph in this chapter and you can see what I mean. TEX has a heuristic to automagically add this extra space for you.

Namely, normally a "`.`", "`?`", or "`!`" ends a sentence when it is followed by white space (or it is at the end of a line). However, this rule is not always correct. For example, in "I saw H.M. Schmoe downtown." the period after "M" is not a sentence-ending period. TEX assumes this is the case because the letter before the period is upper-case.

However, sometimes you might end a sentence after an abbreviation: "Hamiltonian Cycle is in NP." To tell TEX that this is a sentence-ending period, you can separate the upper case letter from the period as follows: "`NP\null.`".

On the other hand, if you say "My dog's name is Mr. Smith" you do not want TEX to treat "Mr." as the end of a sentence. In this case if you write "`Mr.~Smith`" then TEX will use the normal amount of inter-word space (and, as a bonus, you won't end up with "Mr." at the end of a line and "Smith" at the beginning of the next line).

9.3 Hyphens and Dashes

There are four separate typographic symbols which you might be tempted to use the same keyboard character for: "-", "–", "—" and "−". The first is a hyphen, used for (wait for it...) hyphenated words, such as "multi-faceted". The second is known as an "en-dash", and should be used when writing a numeric range such as "40–45". The third is known as an em-dash, and is used — according to some styles — to set off parts of a sentence. Finally, − is an arithmetic operator. These four symbols are obtained with "`-`", "`--`", "`---`"and "`$-$`", respectively.

9.4 Clubs and Widows

A *widow* is a single line at the top of a page. A *club* is a single line at the bottom of a page. (Other people use different names, including *orphan*. Moving on...) I told you that so I can tell you this.

Widows and clubs are typographically displeasing, and so by default the TEX typesetting engine has a moderate dislike for widows and clubs and will try to break paragraphs (when spitting out a page) to avoid them. However, if you have lots of figures, tables, listings, or other such non-textual stuff, or if you like to write short paragraphs, you may find that TEX/LATEX is not able to automatically avoid them.

This is something you should only worry about when you are very close to the final copy of your thesis, since minor edits can cause the content of many following pages to move around. When you are almost done, if you want to deal with clubs and widows, read the blurb in `acadia-hon-thesis.sty` which gives you some information about going this last typographic mile. (Note that the blurb is a comment in that file, so it doesn't appear anywhere in this PDF file.)

Chapter 10

Algorithms and Listings

10.1 Algorithms

Algorithms can be typeset in LaTeX using the `algorithm` and `algorithmic` packages. Once they have been included typesetting an algorithm is a snap. For example, the code:

```
\begin{algorithmic}[1]
\FOR{$i \leftarrow \left[0, 10 \right]$}
   \PRINT $i$ to the console.
\ENDFOR
\end{algorithmic}
```

produces the output:

1: **for** $i \leftarrow [0, 10]$ **do**
2: **print** i to the console.
3: **end for**

In this example the `[1]` argument to the algorithmic environment tells the package to place Arabic numbers at the beginning of each line.

Like figures and tables, algorithms can be labelled and captioned. For example, the code:

```
%----------------------------------------
% Algorithm: Packet size : PACKETSIZE_ALG
%----------------------------------------
\begin{algorithm}[h]
    \centering
    \small
    \caption{Prints out each odd number from $0\rightarrow n-1$ in
unary, and $n$ in decimal.}
    \begin{algorithmic}[1]
        \REQUIRE
    $n$, an integer number where $n \geq 1$
\ENSURE
    Prints out each odd number from $0 \rightarrow n-1$ in unary.\\
    Prints out $n$ in decimal.
\FOR{$i \leftarrow \left[0, n \right]$}
    \IF{$i = n$}
\PRINT Emit $i$ to the stream.
    \ELSIF{$i \% 2 != 0$}
\STATE
    \COMMENT{Then $i$ is odd.}
\STATE
     $j \leftarrow i$
\WHILE{$j \geq 0$}
    \STATE
    Emit $1$ to the stream.
    \STATE
    $j \leftarrow j - 1$
\ENDWHILE
    \ELSE[Then $i$ is even.]
\STATE
\COMMENT{Print nothing!}
    \ENDIF
\ENDFOR
```

```
    \end{algorithmic}
    \label{alg:EXAMPLE_ALG}
\end{algorithm}
```

produces the following typeset algorithm, which can be referenced as Algorithm 10.1 by typing `Algorithm~\ref{alg:EXAMPLE_ALG}`:

Algorithm 10.1 Prints out each odd number from $0 \rightarrow n-1$ in unary, and n in decimal.

Input: n, an integer number where $n \geq 1$
Output: Prints out each odd number from $0 \rightarrow n-1$ in unary.
Prints out n in decimal.

```
 1: for i ← [0, n] do
 2:     if i = n then
 3:         print  Emit i to the stream.
 4:     else if i%2! = 0 then
 5:         /*Then i is odd.*/
 6:         j ← i
 7:         while j ≥ 0 do
 8:             Emit 1 to the stream.
 9:             j ← j − 1
10:         end while
11:     else /*Then i is even.*/
12:         /*Print nothing!*/
13:     end if
14: end for
```

10.2 Listings

Listings in LaTeX can be typeset using the `listings` package. We can include code within the text using:

```
\begin{centering}
\lstinputlisting[float=h,language=FILE_LANGUAGE,
                 caption=A CAPTION,label=THE_LABEL]{FILE_TO_LIST}
\end{centering}
```

For example, to include a centered listing of the contents of the Java file located at `./examples/example.java`, with a caption "Example Java File." and a label `lst:JAVA`, we would write:

```
\begin{centering}
    \lstinputlisting[float=h,language=Java,caption=Example Java File.,
                     label=lst:JAVA]{./examples/example.java}
\end{centering}
```

which would produce Listing 10.1.

Listing 10.1: Example Java File.

```
1  /**
2   *
3   */
4  package treestore.testing;
5
6  /**
7   * @author An Author!
8   *
9   */
10
11 public class MathTester
12 {
13     public static void main(String [] args) throws Throwable
14     {
15         int a = 1;
16         int b = 3;
17         int c = 100;
18         System.out.println("a='"+a+"', b='"+b+"', c='"+c+"'");

19         System.out.println("1:a-b:"+(a-b));
20         System.out.println("2:a-b*c:"+(a-b*c));
21         System.out.println("3:a-b*c-1:"+(a-b*c+1));
22     }
23 }
```

But what if we only want to show a few of the lines of the Java program?

In this case we can tell the package to only include certain lines, for example, lines $10 \rightarrow 11$ in file `./examples/example.java` would be written

```
\begin{centering}
    \lstinputlisting[float=h,firstline=10,lastline=11,language=Java,
        caption=Partial Java Listing.,
        label=lst:SOMEJAVA]{./examples/example.java}
\end{centering}
```

and would produce Listing 10.2.

Listing 10.2: Partial Java Listing.

```
1
2 public class MathTester
```

The `listings` package is capable to working with a large number of languages; you can find the documentation at: `http://www.ctan.org/tex-archive/macros/latex/contrib/listings/listings-1.3.pdf`.

For example, to typeset an XML file using a "footnote-sized" font we would use the code:

```
\begin{centering}
    \lstinputlisting[float=h,basicstyle=\footnotesize,language=XML,
        caption=Example XML File.,
        label=lst:XML]{./examples/simple_example_xml.xml}
\end{centering}
```

which would produce Listing 10.3.

Listing 10.3: Example XML File.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <first>
3    <second>
4      <z>Simple content</z>
5      <e attrib="1"/>
6    </second>
7    <third>
8      <e attrib="2"/>
9    </third>
10 </first>
```

Chapter 11

Packages

You might find some or all of the following packages useful for writing your thesis. To find out more options about these packages, you can read the corresponding manual. On a Linux system you should be able to read (for example) the hyperref documentation by executing the command `texdoc hyperref`.

textcomp Useful for some extra symbols.

inputenc Specifies the encoding format of inputs. Generally you would use `latin1` for a thesis written in English.

amsmath Many useful math mode bits.

amsfonts Fonts for use in math mode (e.g., subscript sized Greek letters).

amssymb Extra math symbols (not necessarily useful).

amsth Allows you to define "theorem-like" sections (e.g., the definition used earlier).

graphicx Package for graphics functionality and quality.

soul Provides ability to "**s**trike **o**ut" and **u**nder**l**ine text (e.g., useful at editing time).

hyperref Provides hyper-linking within the PDF output file. At time of writing, the options given to the hyperref package in this sample thesis' `header.tex` cause the table of contents entries, figure and tables numbers, bibliographic references and various other things to be hyperlinks. That is, if you see (for example) "Figure 8.1"

and you click on the "8.1", your PDF viewer should take you to that figure. (Try it out right now.) Note that in `header.tex` there are a few fields you should edit by hand to register the author and title of your thesis inside the PDF file.

listings A very powerful listings package to make life easier.

subfig Allows you to create sub-figures and subtables (using `\subfloat`) within figures and tables. This replaces the old `subfigure` package which is deprecated.

verbatim Allows for the use of the `verbatim` environment which makes it easier to include multi-line text verbatim in the document.

alltt Allows for including text that is typed in typewriter font. The text is interpreted as code, so be careful of the TEX and LATEX commands you use.

natbib Better control over your bibliography, including citation style within the document (e.g., saying Head et al. [3] versus just [3]). Numerous customization parameters.

acadia-hon-thesis The "hopefully-done (for 2015/16, anyway)" Acadia honours style package which was used for this sample thesis.

algorithm Adds powerful/clean algorithm functionality.

algorithmic Adds an algorithmic environment to your documents. Allows for labeling, captioning, etc.

tikz This is a very powerful package for producing many different kinds of plots, diagrams, graphs, and so on. Including this package gives you the basic capability; to get additional capabilities you need additional `\usepackage` commands (put them in `header.tex`). For example, the data plots shown in Chapter 7 require the use of `\usepackage{pgfplots}`.

Appendix A

Web sites

Here are a few web sites that you may find useful. Note that these links are all "click-able" from a PDF file because they were entered with the "`\url{...}`" command. There is also the "`\href{mylink}{mytext}`" command, which creates a link to `mylink`, but puts the `mytext` as the visible text in your document.

- `http://www.google.com`. (Duh.)
- TEXlive: `http://tug.org/texlive/`. One of the free TEX systems, and maybe the best.
- Citations Site: `http://merkel.zoneo.net/Latex/natbib.php`.
- CTAN (the Comprehensive TEX Archive Network): `http://www.ctan.org/`. This site contains most freely-available TEX resources (including LATEX itself and many contributed packages).
- A FAQ: `http://www.tex.ac.uk/`.
- MiKTeX: `http://www.miktex.org/`.
- Many Tikz and PGF graphics examples: `http://www.texample.net/`. If you want to use PGF/TikZ, this is a good place to look for a plot, picture, graph or diagram similar to the one you want. You can peruse the output images and then download the corresponding source code.

- The (allegedly) comprehensive list of LaTeX symbols: `http://tug.ctan.org/info/symbols/comprehensive/symbols-letter.pdf`. At the time of writing, a (*cough*) bonus feature of this document is that it utilizes the really ugly coloured boxes around the links, as described at the end of Chapter 6.

Bibliography

[1] Douglas Adams. *The Hitchhiker's Guide to the Galaxy*. Pan Books, UK, 1979.

[2] Wikipedia Community. Cookie. Wikipedia, 2007. URL `http://en.wikipedia.org/wiki/Cookie`.

[3] Michael R. Head, Madhusudhan Govindaraju, Robert van Engelen, and Wei Zhang. Grid scheduling and protocols—benchmarking xml processors for applications in grid web services. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 121, New York, NY, USA, 2006. ACM Press. ISBN 0-7695-2700-0. doi: http://doi.acm.org/10.1145/1188455.1188581.

[4] Michael Kaufmann and Dorothea Wagner. *Drawing Graphs: Methods and Models*. Springer, Berlin, 2001.

[5] Helen C. Purchase, Robert F. Cohen, and Murray I. James. An Experimental Study of the Basis for Graph Drawing Algorithms. *Journal of Experimental Algorithmics*, 2(4), 1997.

[6] Helen C. Purchase, Robert F. Cohen, and Murray I. James. Silly putty experiments. *Journal of Experimental Putty*, 2(4), 1999.