Conversational Mathematics

Independent Study Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Bachelor of Arts in the
Mathematics and Computer Science at The College of Wooster

by
Dylan Orris

The College of Wooster
2019

Advised by:

Dr. Fox (Mathematics and Computer Science)

Abstract

Include a short summary of your thesis, including any pertinent results. This section is *not* optional for the Mathematics and Computer Science or Physics Department ISs, and the reader should be able to learn the meat of your thesis by reading this (short) section.

This work is dedicated to the future generations of Wooster students.

Acknowledgments

I would like to acknowledge Prof. Lowell Boone in the Physics Department for his suggestions and code.

Contents

List of Figures

List of Tables

List of Listings

CHAPTER 1

Introduction

ver the past decade, natural language processors have become more and more commonplace. I n late 2011, Apple introduced their digital assistant, Siri, to iOS. Siri was the successo r to voice control, an iOS feature that allowed the user to interact with their phone usin g their voice, though in both an extremely limited fashion and rather unreliably. Siri was powered by a much better voice recognition model, and not only could process many command s into action, but could even ask for clarification based on context. Since this time, Goo gle introduced the Google Assistant to its Android and ChromeOS platforms, Amazon released smart home products powered by Alexa, and Siri has been ported to MacOS and WatchOS. All of these digital assistants rely on understanding the commands of users to be useful, and the only way this is possible is through natural language processing.

From Q2 2017 to Q2 2018, smart speakers, the quintessential aspect of the smart home to ma ny, had shipments grow 187%. [**?**] t-speaker-shipments-grew-187-year-on-year-in-q2-2018-with-china-the-fastest-growing-market

This growth is expected to continue in the coming years, with smart speakers and the smart home technology they may control becoming an enormous industry. As this industry expands, there will be greater and greater access to these systems among the general population. W hat if it was possible for a student to ask their smart screen why a mathematical claim wa s true, and that system could respond by displaying the proof? Smart devices offer more th an simply convenience, they could become an amazing source of educational opportunities wh ich are not otherwise available for many.

While we will not be using smart devices in this project, we will be creating a front end interface for students to more easily investigate mathematical questions, without the barr ier of language.

1.1 Problem Statement

Automated theorem proving systems are a powerful tool which can allow a student to check t heir work or a researcher to prove a claim which could be extremely time consuming if done by hand. Unfortunately, these systems are not trivial to use, with the use of any such sy stem requiring the user to learn the proper method for input, which will also differ betwe en different such systems. As a result of this, it can be easy to be put off of these syst ems, and once one learns one such system, it is easy to be stuck using what is already kno wn, even if better options may be available. A wonderful solution to this problem would be leveraging computational power to convert a human language request, i.e. "Prove that the length of the third side of a triangle is determined by the two other sides and the angle between them" into a language the proving system will understand.

1.2 Project Aims

This project has two separate but interconnected goals; the production of a natural langua ge processor, and the creation of a basic automated prover. The processor will:

- Serve as a front end for any automated prover.
- Be fully functional without any understanding of symbolic notation or programming
- Be modular enough to become the front end of any automated proving system.
- Accurately translate standard English to symbolic notation.

The prover will have far more modest aims:

- Understand basic questions in a limited scope, such as geometry or arithmetic.
- Take symbolic input, and return the truth value of said statement.
- Display the steps of the proof to the end user.

As the two pieces are separate, the processor will not be limited by the prover, and may i nstead be joined with a more robust system by specifying what said system is. This will be necessary to convert the request properly, so that the processor will know which symbolic language to translate to.

1.2.1 Natural Language Processor

The processor will take a user's input in English, and determine the goal of the user base d on the content of that input. The user will not be required to know anything, other than that the system is intended for mathematical queries and will be able to do nothing else. Once the input has been analyzed, the processor will convert it to the proper symbolic la nguage for the automated prover.

1.2.2 Automated Theorem Prover

The prover will take a mathematical claim, in a symbolic notation, and apply axioms, theor ems, and other knowledge as necessary to prove or disprove the claim. If the prover is abl e to use theorems in its proofs, it will be barred from using them in such a way as to giv e the appearance of circular logic. For example, when a user asks for a proof of the Pytha gorean Theorem, the prover will be disallowed from using the Pythagorean Theorem as its ju stification for the truth of the theorem. ing single step proofs except in the case of an appeal to an axiom! Should the user ask for a proof of an axiom, it will be permissible to simply point to the axiom. The prover should not and will not be expected to explain an axiom to a user.

CHAPTER 2

Previous Work

2.1 Natural Language Processors

Natural language processing has been an area of interest in computer science for the past 60 years. Initially, the subject was limited to machine translation, with the Georgetown e xperiment in 1954 being an early foray into the field. In this experiment over sixty Russi an sentences were translated to English entirely automatically. The results were promising enough that the researchers anticipated automatic translation being solved within the nex t five years. [**?**] Progress was, obviously, much slower than this.

Funding was cut when the program ran over a decade, and progress in the field slowed. Two successful systems were created in the 1960s, SHRDLU and ELIZA.

2.1.1 SHRDLU

SHRDLU was a querying system which created a small "Blocks World" which was populated by cones, spheres, cubes, and other geometric shapes of various sizes and colors. [**?**] The user was able to instruct SHRDLU on how to move these objects around simply by s pecifying the shape in addition to its color or size. SHRDLU also had a simple memory syst em, allowing for reference to recently interacted with objects. This memory also allowed f or SHRDLU to be queried on what she had previously done. SHRDLU's world contained basic ph ysics, allowing for the program to describe what was possible in the world and what was no t. The final major feature was to remember the name a user gave an object or collection of objects. From this, it was possible to more easily instruct SHRDLU.

2.1.2 ELIZA

Include a sample dialogue, and a flowchart of ELIZA's function

CHAPTER 3

Working with figures and tables

3.1 Getting a simple figure in the document

In this chapter we want to talk about including figures and tables in the document. To insert a simple figure you can enter something like

```
\begin{figure}[!ht]
\begin{center}
\woopic{picture3}{.8}
\end{center}
\caption{Our first
 picture}\label{first}
\end{figure}
```

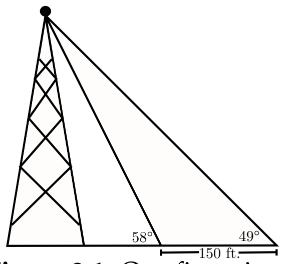


Figure 3.1: Our first picture

The `!ht` tell LaTeX to try and place the figure here no matter what or at the top of the next page. The `\woopic` command takes the name of the picture as the first argument and the scaling factor as the second argument. The scaling factor must be between zero and one and the figure name must have *no spaces*. Your figures can be in one of three formats: jpg, tif, or pdf. Captions are placed below the figure and your label should be placed after the caption.

In the next example we are using the woosterthesis option `picins` to typeset a picture inside a paragraph and have the text wrap around the figure. This option loads the `wrapfig` package. One thing to note is that the figures placed in this manner do not float with the other figures and as such numbering could get out of sequence. Keep an eye out for such behavior. This technique should be used sparingly in your thesis.

```
\newcommand{\sample}{Some text that is reused over and over
 again in the example. }
\begin{wrapfigure}{r}{2.2in}
\woopic{picture2}{.4}
\caption{Conchoid.}
\end{wrapfigure}
\sample\sample\sample\sample
```

Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example.

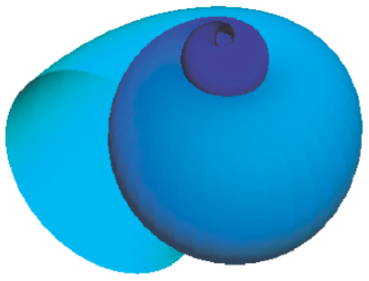


Figure 3.2: Conchoid.

3.1.1 Minipages

You can also create minipages in your documents to accomplish more complicated formatting. For example you could try the following which produces Figure 3.3.

```
\begin{minipage}[t][3 in][t]{1 in}
This is a minipage which is 3 in tall and 1 in wide.
 Top Text Text Text Text.\end{minipage}\hfill
\begin{minipage}[t][3 in][c]{1 in}
This is a minipage which is 3 in tall and 1 in wide.
 Center Text Text Text Text.\end{minipage}\hfill
\begin{minipage}[t][3 in][b]{1 in}
This is a minipage which is 3 in tall and 1 in wide.
 Bottom Text Text Text Text.\end{minipage}
```

In the example above, the syntax `\begin{minipage}[t][3 in][t]{1 in}` follows the convention

`\begin{minipage}[minipageposition][height][textposition]{width}`

3.1.1.1 How to get more than one picture in the same figure

You can use minipages to put more than one picture in a figure. Here is an example of how to do this.

```
\begin{minipage}[!ht]{6cm}
\woopic{picture1}{.4}
```

This is a minipage which is 3 in tall and 1 in wide. Top Text Text Text Text.

This is a minipage which is 3 in tall and 1 in wide. Center Text Text Text Text.

This is a minipage which is 3 in tall and 1 in wide. Bottom Text Text Text Text.

Figure 3.3: Minipage example

```
\par
\caption[What goes in the List of Figures]{Left}
\end{minipage}
\hfill
\begin{minipage}[!ht]{6cm}
\woopic{picture2}{.4}
\end{picture}\par
\caption{Right}
\end{minipage}
```

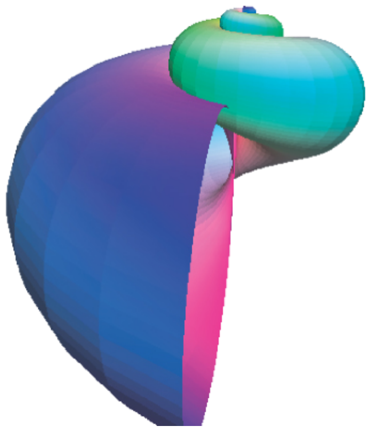


Figure 3.4: Left

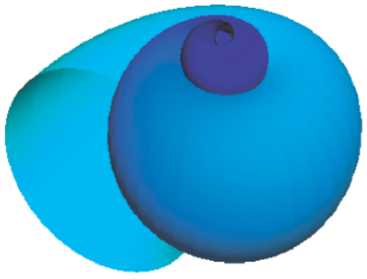
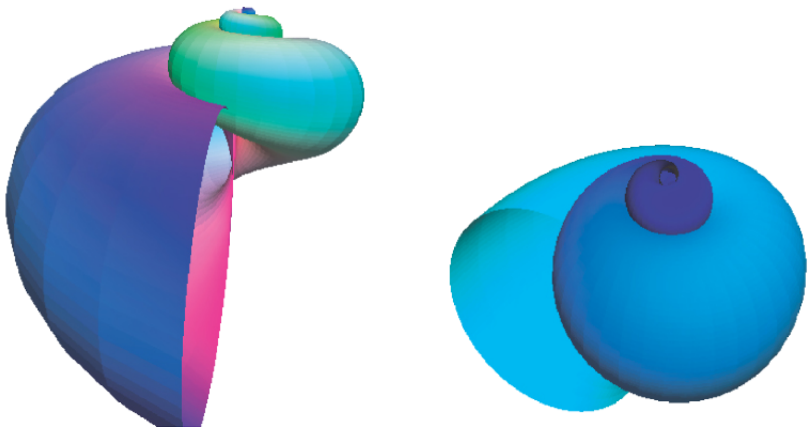


Figure 3.5: Right

You can also use the `subfigure` package to do this.

```
\begin{figure}[!ht]\centering
\subfigure[What goes in the List][Large conchoid]
{\woopic{picture1}{.4}\label{fig3:left}}
\qquad
\subfigure[What goes in the List][Small conchoid]
{\woopic{picture2}{.4}\label{fig3:right}}
\caption{Two pictures in one figure}\label{fig3}
\end{figure}
```



(a) Large conchoid (b) Small conchoid

Figure 3.6: Two pictures in one figure

We should now be able to refer to either Figure 3.6 (a) or Figure 3.6 (b) using the labels we gave to the left and right images.

The reader is referred to Chapters 8, 9, and 16 of **?**] or to Chapters 6 and 10 of **?**] for a complete discussion of figures and graphics.

3.2 Tables

Tables are fairly easy to set up. Here is a simple table

```
\begin{table}[!ht]
\begin{center}
\begin{tabular}{r l}
  $\underline{\textnormal{District}}$ &
  $\underline{\textnormal{Population}}$\\
   Applewood & 8280 \\
   Boxwood & 4600  \\
   Central & 5220
   \end{tabular}\caption{Our first table}
   \end{center}
\end{table}
```

District	Population
Applewood	8280
Boxwood	4600
Central	5220

Table 3.1: Our first table

In `\begin{tabular}{r l}` the two "r" and "l" indicate that we have two columns with right and left aligned entries and no lines dividing cells or around the table. I can make the table look more like a spreadsheet by doing

```
\begin{table}[!ht]
\begin{center}
\begin{tabular}{|r|l|}
\hline
  {\textnormal{District}} &
  {\textnormal{Population}}\\ \hline
   Applewood & 8280 \\ \hline
   Boxwood & 4600  \\ \hline
   Central & 5220\\ \hline
   \end{tabular}\caption{Our first table again}
   \end{center}
\end{table}
```

District	Population
Applewood	8280
Boxwood	4600
Central	5220

Table 3.2: Our first table again

Here is a more complicated example of a table.

```
\begin{table}[!ht]
\centerline{
\begin{tabular}{|l||r|r|r|r|} \hline
\emph{Reprojection} & \multicolumn{3}{|c|}{\emph{Largest
 Reduction of Curvature}}
  & \emph{Average} \\ \cline{2-4}
\emph{Method} & \emph{Original} & \emph{Reprojected} &
 \emph{at} &
  \emph{Reduction} \\
 & \emph{Curvature} & \emph{Curvature} &
  \emph{Rotation} & \emph{of Curvature} \\
  \hline \hline
ZEEL & 0.0358 & 0.0245 &
 $\degree{45}$ & 0.0050 \\ \hline
ZEEL ext.\ & 0.0358 & 0.0245 &
 $\degree{45}$ & 0.0059 \\ \hline
Regridding & 0.0428 & 0.0166 &
 $\degree{75}$ & 0.0159 \\ \hline
Block & 0.0358 & 0.0103 &
 $\degree{45}$ & 0.0163 \\ \hline
\end{tabular}}
\caption{Reduction of curvature by each
 reprojection method\label{tbl:kreduce}}
\end{table}
```

Reprojection Method	*Largest Reduction of Curvature*			*Average Reduction of Curvature*
	Original Curvature	*Reprojected Curvature*	*at Rotation*	
ZEEL	0.0358	0.0245	45°	0.0050
ZEEL ext.	0.0358	0.0245	45°	0.0059
Regridding	0.0428	0.0166	75°	0.0159
Block	0.0358	0.0103	45°	0.0163

Table 3.3: Reduction of curvature by each reprojection method

Please refer to Chapter 6 of **?**] for a complete discussion of tables and tabular environments.

CHAPTER 4

Working with bibliographies and indicies

I would highly recommend that you use BibTEX to create your bibliography. BibTEX processes a special .bib file. The .bib file is where you enter your bibliographic information. A sample entry looks something like

```
@article{feu02,
author= {Thomas~Feuerstack},
title= {Introduction to pdf{\TeX{}}},
journal= {TUGboat},
volume= {23},
pages= {329--334},
number= {3/4},
url= {http://www.tug.org/TUGboat/Articles/tb23-3-4/tb75feu.pdf},
year= 2002}
```

or

```
@book{mgbcr04,
author= {Frank~Mittelbach and Michel~Goossens and
Johannes~Braams and David~Carlisle and Chris~Rowley},
title= {The \LaTeX\ Companion},
publisher= {Addison Wesley Professional},
edition= {2nd},
address= {New York},
year= 2004}
```

For a Web site I would recommend the following

```
@misc{brei04,
author =  {Jon~Breitenbucher},
title =  {{W}ooster related {L}a{T}e{X} files},
url =  {http://jbreitenbuch.wooster.edu/~jonb/latex/},
howpublished= {World Wide Web},
year= 2004,
note =  {Accessed on 03/11/2004}}
```

You can make a reference by typing `\citet{mgbcr04}` to produce **?**]. Other forms for citation include `\citep{mgbcr04}` or `\citeauthor {mgbcr04}` to produce [**?**] or **?** respectively. You can consult **?**] or **?**] to find out how to format entries in the .bib file and what options each reference type has.[1]

Indicies are also relatively easy to create. If I wanted to have Wooster show up in the index, I would enter `Wooster\index{Wooster}` in my source file. I could create a subentry for User Services by entering `User Services\index{Wooster!User Services}`. A subsubentry for Help Desk would be entered as `\index{Wooster!User Services!Help Desk}`.

To create the index one needs to make sure to uncomment the `\makeindex` command in the `username.tex` file. One also needs to uncomment the makeidx entry in the `styles/packages.tex` file and then run the Makeindex program. Consult **?**] or **?**] for further information.

[1]You could also use footnotes if your department called for that.

APPENDIX A

Typesetting Mathematical Formulae

This appendix is taken from **?**] under the GNU open source documentation license. This appendix addresses the main strength of TEX: mathematical typesetting. But be warned, this appendix only scratches the surface. While the things explained here are sufficient for many people, don't despair if you can't find a solution to your mathematical typesetting needs here. It is highly likely that your problem is addressed in AMS-LATEX[1] or some other package.

A.1 General

LATEX has a special mode for typesetting mathematics. Mathematical text within a paragraph is entered between `\(` and `\)`, between `$` and `$` or between `\begin{math}` and `\end{math}`.

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
$c^{2}=a^{2}+b^{2}$
```

Add $a$ squared and $b$ squared to get $c$ squared. Or, using a more mathematical approach: $c^{2}=a^{2}+b^{2}$

```
\TeX{} is pronounced as
$\tau\epsilon$.\\[6pt]
100~m$^{3}$ of water\\[6pt]
This comes from my $\heartsuit$
```

TEX is pronounced as $\tau\epsilon$.

100 m$^{3}$ of water

This comes from my $\heartsuit$

It is preferable to *display* larger mathematical equations or formulae, rather than to typeset them on separate lines. This means you enclose them in `\[` and `\]` or between `\begin{displaymath}` and `\end{displaymath}`. This produces formulae which are not numbered. If you want LATEX to number them, you can use the equation environment.

[1]`CTAN:/tex-archive/macros/latex/packages/amslatex`

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
\begin{displaymath}
c^{2}=a^{2}+b^{2}
\end{displaymath}
And just one more line.
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach:

$$c^2 = a^2 + b^2$$

And just one more line.

You can reference an equation with `\label` and `\ref`

```
\begin{equation} \label{eq:eps}
\epsilon > 0
\end{equation}
From (\ref{eq:eps}), we gather
\ldots
```

$$\epsilon > 0 \tag{A.1}$$

From (A.1), we gather …

Note that expressions will be typeset in a different style if displayed:

```
$\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}$
```

$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$

```
\begin{displaymath}
\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}
\end{displaymath}
```

$$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$$

There are differences between *math mode* and *text mode*. For example in *math mode*:

1. Most spaces and linebreaks do not have any significance, as all spaces either are derived logically from the mathematical expressions or have to be specified using special commands such as `\,`, `\quad`, or `\qquad`.

2. Empty lines are not allowed. Only one paragraph per formula.

3. Each letter is considered to be the name of a variable and will be typeset as such. If you want to typeset normal text within a formula (normal upright font and normal spacing) then you have to enter the text using the `\textrm{...}` commands.

```
\begin{equation}
\forall x \in \mathbf{R}:
\qquad x^{2} \geq 0
\end{equation}
```

$$\forall x \in \mathbf{R}: \qquad x^2 \geq 0 \tag{A.2}$$

```
\begin{equation}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbf{R}
\end{equation}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbf{R} \tag{A.3}$$

Mathematicians can be very fussy about which symbols are used: it would be conventional here to use 'blackboard bold', bold symbols which is obtained using `\mathbb` from the package `amsfonts` or `amssymb`.

The last example becomes

```
\begin{displaymath}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbb{R}
\end{displaymath}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbb{R}$$

A.2 Grouping in Math Mode

Most math mode commands act only on the next character. So if you want a command to affect several characters, you have to group them together using curly braces: `{...}`.

```
\begin{equation}
a^x+y \neq a^{x+y}
\end{equation}
```

$$a^x + y \neq a^{x+y} \tag{A.4}$$

A.3 Building Blocks of a Mathematical Formula

In this section, the most important commands used in mathematical typesetting will be described. Take a look at **?**] for a detailed list of commands for typesetting mathematical symbols.

Lowercase Greek letters are entered as `\alpha`, `\beta`, `\gamma`, . . . , uppercase letters are entered as `\Gamma`, `\Delta`, . . .[2]

```
$\lambda,\xi,\pi,\mu,\Phi,\Omega$
```

$\lambda, \xi, \pi, \mu, \Phi, \Omega$

Exponents and Subscripts can be specified using the `^` and the `_` character.

[2]There is no uppercase Alpha defined in LaTeX 2$_\varepsilon$ because it looks the same as a normal roman A. Once the new math coding is done, things will change.

```
$a_{1}$ \qquad $x^{2}$ \qquad
$e^{-\alpha t}$ \qquad
$a^{3}_{ij}$\\
$e^{x^2} \neq {e^x}^2$
```

$a_1 \qquad x^2 \qquad e^{-\alpha t} \qquad a^3_{ij}$
$e^{x^2} \neq {e^x}^2$

The **square root** is entered as `\sqrt`, the n^{th} root is generated with `\sqrt[`n`]`. The size of the root sign is determined automatically by LaTeX. If just the sign is needed, use `\surd`.

```
$\sqrt{x}$ \qquad
$\sqrt{ x^{2}+\sqrt{y} }$
\qquad $\sqrt[3]{2}$\\[3pt]
$\surd[x^2 + y^2]$
```

$\sqrt{x} \qquad \sqrt{x^2+\sqrt{y}} \qquad \sqrt[3]{2}$
$\surd[x^2 + y^2]$

The commands `\overline` and `\underline` create **horizontal lines** directly over or under an expression.

```
$\overline{m+n}$
```

$\overline{m+n}$

The commands `\overbrace` and `\underbrace` create long **horizontal braces** over or under an expression.

```
$\underbrace{ a+b+\cdots+z }_{26}$
```

$\underbrace{a+b+\cdots+z}_{26}$

To add mathematical accents such as small arrows or tilde signs to variables, you can use the commands given in **?**]. Wide hats and tildes covering several characters are generated with `\widetilde` and `\widehat`. The `'` symbol gives a prime.

```
\begin{displaymath}
y=x^{2}\qquad y'=2x\qquad y''=2
\end{displaymath}
```

$$y = x^2 \qquad y' = 2x \qquad y'' = 2$$

Vectors often are specified by adding a small arrow symbol on top of a variable. This is done with the `\vec` command. The two commands `\overrightarrow` and `\overleftarrow` are useful to denote the vector from A to B.

```
\begin{displaymath}
\vec a\quad\overrightarrow{AB}
\end{displaymath}
```

$$\vec a \quad \overrightarrow{AB}$$

Names of log-like functions are often typeset in an upright font and not in italic like variables. Therefore LaTeX supplies the following commands to typeset the most important function names:

```
\arccos   \cos    \csc   \exp   \ker     \limsup  \min   \sinh
\arcsin   \cosh   \deg   \gcd   \lg      \ln      \Pr    \sup
\arctan   \cot    \det   \hom   \lim     \log     \sec   \tan
\arg      \coth   \dim   \inf   \liminf  \max     \sin   \tanh
```

```
\[\lim_{x \rightarrow 0}
\frac{\sin x}{x}=1\]
```

$$\lim_{x\rightarrow 0}\frac{\sin x}{x}=1$$

For the modulo function, there are two commands: `\bmod` for the binary operator "$a \bmod b$" and `\pmod` for expressions such as "$x \equiv a \pmod{b}$."

A built-up **fraction** is typeset with the `\frac{...}{...}` command. Often the slashed form 1/2 is preferable, because it looks better for small amounts of 'fraction material.'

```
$1\frac{1}{2}$~hours
\begin{displaymath}
\frac{ x^{2} }{ k+1 }\qquad
x^{ \frac{2}{k+1} }\qquad
x^{ 1/2 }
\end{displaymath}
```

$1\frac{1}{2}$ hours

$$\frac{x^{2}}{k+1}\qquad x^{\frac{2}{k+1}}\qquad x^{1/2}$$

To typeset binomial coefficients or similar structures, you can use either the command `\binom`{*num*}{*denom*} or `\genfrac`{*ldelim*}{*rdelim*}{*thickness*}{*style*}{*num*}{*denom*}. The second command can be used to produce customized fraction like output and more information can be found in **?**].

```
\begin{displaymath}
\binom{n}{k}\qquad
\genfrac{}{}{0pt}{}{x}{y+2}
\end{displaymath}
```

$$\binom{n}{k}\qquad\genfrac{}{}{0pt}{}{x}{y+2}$$

The **integral operator** is generated with `\int`, the **sum operator** with `\sum`. The upper and lower limits are specified with `^` and `_` like subscripts and superscripts.

```
\begin{displaymath}
\sum_{i=1}^{n} \qquad
\int_{0}^{\frac{\pi}{2}} \qquad
\end{displaymath}
```

$$\sum_{i=1}^{n}\qquad\int_{0}^{\frac{\pi}{2}}\qquad$$

For **braces** and other delimiters, there exist all types of symbols in TEX (e.g. $[\ \langle\ \|\ \updownarrow$). Round and square braces can be entered with the corresponding keys, curly braces with `\{`, all other delimiters are generated with special commands (e.g. `\updownarrow`). For a list of all delimiters available, check **?**].

```
\begin{displaymath}
{a,b,c}\neq\{a,b,c\}
\end{displaymath}
```

$$a, b, c \neq \{a, b, c\}$$

If you put the command `\left` in front of an opening delimiter or `\right` in front of a closing delimiter, TEX will automatically determine the correct size of the delimiter. Note that you must close every `\left` with a corresponding `\right`, and that the size is determined correctly only if both are typeset on the same line. If you don't want anything on the right, use the invisible '`\right .`'!

```
\begin{displaymath}
1 + \left( \frac{1}{ 1-x^{2} }
    \right) ^3
\end{displaymath}
```

$$1+\left(\frac{1}{1-x^2}\right)^3$$

In some cases it is necessary to specify the correct size of a mathematical delimiter by hand, which can be done using the commands `\big`, `\Big`, `\bigg` and `\Bigg` as prefixes to most delimiter commands.[3]

```
$\Big( (x+1) (x-1) \Big) ^{2}$\\
$\big(\Big(\bigg(\Bigg($\quad
$\big\}\Big\}\bigg\}\Bigg\}$\quad
$\big\|\Big\|\bigg\|\Bigg\|$
```

$\Big((x+1)(x-1)\Big)^{2}$

$\big(\Big(\bigg(\Bigg($ $\quad$ $\big\}\Big\}\bigg\}\Bigg\}$ $\quad$ $\big\|\Big\|\bigg\|\Bigg\|$

To enter **three dots** into a formula, you can use several commands. `\ldots` typesets the dots on the baseline, `\cdots` sets them centered. Besides that, there are the commands `\vdots` for vertical and `\ddots` for diagonal dots. You can find another example in section A.5.

```
\begin{displaymath}
x_{1},\ldots,x_{n} \qquad
x_{1}+\cdots+x_{n}
\end{displaymath}
```

$$x_{1},\ldots,x_{n} \qquad x_{1}+\cdots+x_{n}$$

[3]These commands do not work as expected if a size changing command has been used, or the `11pt` or `12pt` option has been specified. Use the exscale or amsmath packages to correct this behaviour.

A.4 Math Spacing

If the spaces within formulae chosen by TeX are not satisfactory, they can be adjusted by inserting special spacing commands. There are some commands for small spaces: `\,` for $\frac{3}{18}$ quad (␣), `\:` for $\frac{4}{18}$ quad (␣) and `\;` for $\frac{5}{18}$ quad (␣). The escaped space character `\␣` generates a medium sized space and `\quad` (␣) and `\qquad` (␣) produce large spaces. The size of a quad corresponds to the width of the character 'M' of the current font. The `\!` command produces a negative space of $-\frac{3}{18}$ quad (␣).

```
\newcommand{\rd}{\mathrm{d}}
\begin{displaymath}
\int\!\!\!\int_{D} g(x,y)
  \, \rd x\, \rd y
\end{displaymath}
instead of
\begin{displaymath}
\int\int_{D} g(x,y)\rd x \rd y
\end{displaymath}
```

$$\int\!\!\!\int_{D} g(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

instead of

$$\int\int_{D} g(x,y)\mathrm{d}x\mathrm{d}y$$

Note that 'd' in the differential is conventionally set in roman.

AMS-LaTeX provides another way for fine tuning the spacing between multiple integral signs, namely the `\iint`, `\iiint`, `\iiiint`, and `\idotsint` commands. With the `amsmath` package loaded, the above example can be typeset this way:

```
\newcommand{\rd}{\mathrm{d}}
\begin{displaymath}
\iint_{D} \, \rd x \, \rd y
\end{displaymath}
```

$$\iint_{D} \,\mathrm{d}x\,\mathrm{d}y$$

See the electronic document testmath.tex (distributed with AMS-LaTeX) or Chapter 8 of "The LaTeX Companion"[4] for further details.

A.5 Vertically Aligned Material

To typeset **arrays**, use the `array` environment. It works somewhat similar to the `tabular` environment. The `\\` command is used to break the lines.

[4] available at `CTAN:/tex-archive/info/ch8.*`.

```
\begin{displaymath}
\mathbf{X} =
\left( \begin{array}{ccc}
x_{11} & x_{12} & \ldots \\
x_{21} & x_{22} & \ldots \\
\vdots & \vdots & \ddots
\end{array} \right)
\end{displaymath}
```

$$\mathbf{X} = \left(\begin{array}{ccc} x_{11} & x_{12} & \ldots \\ x_{21} & x_{22} & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

The `array` environment can also be used to typeset expressions which have one big delimiter by using a "`.`" as an invisible right delimiter:

```
\begin{displaymath}
y = \left\{ \begin{array}{ll}
 a & \textrm{if $d>c$}\\
 b+x & \textrm{in the morning}\\
 l & \textrm{all day long}
  \end{array} \right.
\end{displaymath}
```

$$y = \left\{ \begin{array}{ll} a & \textrm{if $d>c$}\\ b+x & \textrm{in the morning}\\ l & \textrm{all day long} \end{array} \right.$$

For formulae running over several lines or for equation systems, you can use the environments `eqnarray`, and `eqnarray*` instead of `equation`. In `eqnarray` each line gets an equation number. The `eqnarray*` does not number anything.

The `eqnarray` and the `eqnarray*` environments work like a 3-column table of the form `{rcl}`, where the middle column can be used for the equal sign or the not-equal sign. Or any other sign you see fit. The \\ command breaks the lines.

```
\begin{eqnarray}
f(x) & = & \cos x     \\
f'(x) & = & -\sin x   \\
\int_{0}^{x} f(y)dy &
 = & \sin x
\end{eqnarray}
```

$$\begin{array}{rcl} f(x) & = & \cos x \qquad (A.5) \\ f'(x) & = & -\sin x \qquad (A.6) \\ \int_{0}^{x} f(y)dy & = & \sin x \qquad (A.7) \end{array}$$

Notice that the space on either side of the the equal signs is rather large. It can be reduced by setting `\setlength\arraycolsep{2pt}`, as in the next example.

Long equations will not be automatically divided into neat bits. The author has to specify where to break them and how much to indent. The following two methods are the most common ones used to achieve this.

```
{\setlength\arraycolsep{2pt}
\begin{eqnarray}\notag
\sin x & = & x -\frac{x^{3}}{3!}
     +\frac{x^{5}}{5!}-{}
                   \\\notag
 & & {}-\frac{x^{7}}{7!}+{}\cdots
\end{eqnarray}}
```

$$\begin{aligned}\sin x \;&=\; x - \frac{x^3}{3!} + \frac{x^5}{5!} - \\ & \quad - \frac{x^7}{7!} + \cdots\end{aligned}$$

```
\begin{eqnarray}\notag
\lefteqn{ \cos x = 1
     -\frac{x^{2}}{2!} +{} }
                   \\\notag
 & & {}+\frac{x^{4}}{4!}
     -\frac{x^{6}}{6!}+{}\cdots
\end{eqnarray}
```

$$\begin{aligned}&\cos x = 1 - \frac{x^2}{2!} + \\ &\qquad + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots\end{aligned}$$

The \notag command causes LaTeX to not generate a number for this equation.

It can be difficult to get vertically aligned equations to look right with these methods; the package amsmath provides a more powerful set of alternatives.

A.6 Math Font Size

In math mode, TeX selects the font size according to the context. Superscripts, for example, get typeset in a smaller font. If you want to typeset part of an equation in roman, don't use the \textrm command, because the font size switching mechanism will not work, as \textrm temporarily escapes to text mode. Use \mathrm instead to keep the size switching mechanism active. But pay attention, \mathrm will only work well on short items. Spaces are still not active and accented characters do not work.[5]

```
\begin{equation}
2^{\textrm{nd}} \quad
2^{\mathrm{nd}}
\end{equation}
```

$$2^{\textrm{nd}} \quad 2^{\mathrm{nd}} \tag{A.8}$$

Nevertheless, sometimes you need to tell LaTeX the correct font size. In math mode, the font size is set with the four commands:

displaystyle (123), textstyle (123), scriptstyle ($\scriptstyle 123$) and scriptscriptstyle ($\scriptscriptstyle 123$).

Changing styles also affects the way limits are displayed.

[5]The AMS-LaTeX package makes the textrm command work with size changing.

```
\begin{displaymath}
\mathop{\mathrm{corr}}(X,Y)=
 \frac{\displaystyle
   \sum_{i=1}^n(x_i-\overline x)
   (y_i-\overline y)}
  {\displaystyle\biggl[
 \sum_{i=1}^n(x_i-\overline x)^2
\sum_{i=1}^n(y_i-\overline y)^2
\biggr]^{1/2}}
\end{displaymath}
```

$$\mathop{\mathrm{corr}}(X,Y)=\frac{\displaystyle\sum_{i=1}^n(x_i-\overline x)(y_i-\overline y)}{\displaystyle\biggl[\sum_{i=1}^n(x_i-\overline x)^2\sum_{i=1}^n(y_i-\overline y)^2\biggr]^{1/2}}$$

This is one of those examples in which we need larger brackets than the standard `\left[ \right]` provides.

A.7 Theorems, Laws, . . .

When writing mathematical documents, you probably need a way to typeset "Lemmas", "Definitions", "Axioms" and similar structures. LaTeX supports this with the command

newtheorem{*name*}[*counter*]{*text*}[*section*]

The *name* argument, is a short keyword used to identify the "theorem". With the *text* argument, you define the actual name of the "theorem" which will be printed in the final document.

The arguments in square brackets are optional. They are both used to specify the numbering used on the "theorem". With the *counter* argument you can specify the *name* of a previously declared "theorem". The new "theorem" will then be numbered in the same sequence. The *section* argument allows you to specify the sectional unit within which you want your "theorem" to be numbered.

After executing the newtheorem command in the preamble of your document, you can use the following command within the document.

`\begin`{*name*}[*text*]
This is my interesting theorem
`\end`{*name*}

This should be enough theory. The following examples will hopefully remove the final remains of doubt and make it clear that the `\newtheorem` environment is way too complex to understand.

```
% definitions for the document
% preamble
\newtheorem{law}{Law}
\newtheorem{jury}[law]{Jury}
%in the document
\begin{law} \label{law:box}
Don't hide in the witness box
\end{law}
\begin{jury}[The Twelve]
It could be you! So beware and
see law~\ref{law:box}\end{jury}
\begin{law}No, No, No\end{law}
```

Law 1. *Don't hide in the witness box*

Jury 2 (The Twelve). *It could be you! So beware and see law* 1

Law 3. *No, No, No*

The "Jury" theorem uses the same counter as the "Law" theorem. Therefore it gets a number which is in sequence with the other "Laws". The argument in square brackets is used to specify a title or something similar for the theorem.

```
\flushleft
\newtheorem{mur}{Murphy}[section]
\begin{mur}
If there are two or more
ways to do something, and
one of those ways can result
in a catastrophe, then
someone will do it.\end{mur}
```

Murphy A.7.1. *If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.*

The "Murphy" theorem gets a number which is linked to the number of the current section. You could also use another unit, for example chapter or subsection.

A.8 Bold symbols

It is quite difficult to get bold symbols in LaTeX; this is probably intentional as amateur typesetters tend to overuse them. The font change command `\mathbf` gives bold letters, but these are roman (upright) whereas mathematical symbols are normally italic. There is a `\boldmath` command, but *this can only be used outside mathematics mode*. It works for symbols too.

```
\begin{displaymath}
\mu, M \qquad \mathbf{M} \qquad
\mbox{\boldmath $\mu, M$}
\end{displaymath}
```

$$\mu, M \qquad \mathbf{M} \qquad \boldsymbol{\mu, M}$$

Notice that the comma is bold too, which may not be what is required.

The package `amsbsy` (included by `amsmath`) makes this much easier as it includes a `\boldsymbol` command.

```
\begin{displaymath}
\mu, M \qquad
\boldsymbol{\mu}, \boldsymbol{M}
\end{displaymath}
```

$$\mu, M \qquad \boldsymbol{\mu}, \boldsymbol{M}$$

A.9 List of Mathematical Symbols

In the following tables, you find all the symbols normally accessible from *math mode*.

To use the symbols listed in Tables A.12–A.17,[6] the package `amssymb` must be loaded in the preamble of the document and the AMS math fonts must be installed, on the system. If the AMS package and fonts are not installed, on your system, have a look at

`CTAN:/tex-archive/macros/latex/required/amslatex`

Table A.1: Math Mode Accents.

$\hat{a}$	`\hat{a}`	$\check{a}$	`\check{a}`	$\tilde{a}$	`\tilde{a}`	$\acute{a}$	`\acute{a}`
$\grave{a}$	`\grave{a}`	$\dot{a}$	`\dot{a}`	$\ddot{a}$	`\ddot{a}`	$\breve{a}$	`\breve{a}`
$\bar{a}$	`\bar{a}`	$\vec{a}$	`\vec{a}`	$\widehat{A}$	`\widehat{A}`	$\widetilde{A}$	`\widetilde{A}`

Table A.2: Lowercase Greek Letters.

α	`\alpha`	θ	`\theta`	o	`o`	υ	`\upsilon`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	ϕ	`\phi`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	φ	`\varphi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	χ	`\chi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	ψ	`\psi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ω	`\omega`
ζ	`\zeta`	ν	`\nu`	ς	`\varsigma`		
η	`\eta`	ξ	`\xi`	τ	`\tau`		

Table A.3: Uppercase Greek Letters.

Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		

[6]These tables were derived from `symbols.tex` by David Carlisle and subsequently changed extensively as suggested by Josef Tkadlec.

Table A.4: Binary Relations.

You can produce corresponding negations by adding a `\not` command as prefix to the following symbols.

$<$	`<`	$>$	`>`	$=$	`=`
$\leq$	`\leq` or `\le`	$\geq$	`\geq` or `\ge`	$\equiv$	`\equiv`
$\ll$	`\ll`	$\gg$	`\gg`	$\doteq$	`\doteq`
$\prec$	`\prec`	$\succ$	`\succ`	$\sim$	`\sim`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\simeq$	`\simeq`
$\subset$	`\subset`	$\supset$	`\supset`	$\approx$	`\approx`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\cong$	`\cong`
$\sqsubset$	`\sqsubset` [a]	$\sqsupset$	`\sqsupset` [a]	$\Join$	`\Join` [a]
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\bowtie$	`\bowtie`
$\in$	`\in`	$\ni$	`\ni` , `\owns`	$\propto$	`\propto`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$\models$	`\models`
$\mid$	`\mid`	$\parallel$	`\parallel`	$\perp$	`\perp`
$\smile$	`\smile`	$\frown$	`\frown`	$\asymp$	`\asymp`
$:$	`:`	$\notin$	`\notin`	$\neq$	`\neq` or `\ne`

[a]Use the `latexsym` package to access this symbol

Table A.5: Binary Operators.

$+$	`+`	$-$	`-`	$\triangleleft$	`\triangleleft`
$\pm$	`\pm`	$\mp$	`\mp`	$\triangleright$	`\triangleright`
$\cdot$	`\cdot`	$\div$	`\div`	$\star$	`\star`
$\times$	`\times`	$\setminus$	`\setminus`	$\ast$	`\ast`
$\cup$	`\cup`	$\cap$	`\cap`	$\circ$	`\circ`
$\sqcup$	`\sqcup`	$\sqcap$	`\sqcap`	$\bullet$	`\bullet`
$\vee$	`\vee` , `\lor`	$\wedge$	`\wedge` , `\land`	$\diamond$	`\diamond`
$\oplus$	`\oplus`	$\ominus$	`\ominus`	$\uplus$	`\uplus`
$\odot$	`\odot`	$\oslash$	`\oslash`	$\amalg$	`\amalg`
$\otimes$	`\otimes`	$\bigcirc$	`\bigcirc`	$\dagger$	`\dagger`
$\bigtriangleup$	`\bigtriangleup`	$\bigtriangledown$	`\bigtriangledown`	$\ddagger$	`\ddagger`
$\lhd$	`\lhd` [a]	$\rhd$	`\rhd` [a]	$\wr$	`\wr`
$\unlhd$	`\unlhd` [a]	$\unrhd$	`\unrhd` [a]		

Table A.6: BIG Operators.

$\sum$	`\sum`	$\bigcup$	`\bigcup`	$\bigvee$	`\bigvee`	$\bigoplus$	`\bigoplus`
$\prod$	`\prod`	$\bigcap$	`\bigcap`	$\bigwedge$	`\bigwedge`	$\bigotimes$	`\bigotimes`
$\coprod$	`\coprod`	$\bigsqcup$	`\bigsqcup`			$\bigodot$	`\bigodot`
$\int$	`\int`	$\oint$	`\oint`			$\biguplus$	`\biguplus`

Table A.7: Arrows.

$\leftarrow$	`\leftarrow` or `\gets`	$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`
$\rightarrow$	`\rightarrow` or `\to`	$\longrightarrow$	`\longrightarrow`	$\downarrow$	`\downarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`	$\updownarrow$	`\updownarrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\Uparrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Downarrow$	`\Downarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`	$\nearrow$	`\nearrow`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`	$\searrow$	`\searrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`	$\swarrow$	`\swarrow`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`	$\nwarrow$	`\nwarrow`
$\rightleftharpoons$	`\rightleftharpoons`	$\iff$	`\iff` (bigger spaces)	$\leadsto$	`\leadsto` [a]

[a]Use the `latexsym` package to access this symbol

Table A.8: Delimiters.

$($	`(`	$)$	`)`	$\uparrow$	`\uparrow`	$\Uparrow$	`\Uparrow`
$[$	`[` or `\lbrack`	$]$	`]` or `\rbrack`	$\downarrow$	`\downarrow`	$\Downarrow$	`\Downarrow`
$\{$	`\{` or `\lbrace`	$\}$	`\}` or `\rbrace`	$\updownarrow$	`\updownarrow`	$\Updownarrow$	`\Updownarrow`
$\langle$	`\langle`	$\rangle$	`\rangle`	$\vert$	`|` or `\vert`	$\Vert$	`\\|` or `\Vert`
$\lfloor$	`\lfloor`	$\rfloor$	`\rfloor`	$\lceil$	`\lceil`	$\rceil$	`\rceil`
$/$	`/`	$\backslash$	`\backslash`		`.` (dual. empty)		

Table A.9: Large Delimiters.

$\lgroup$	`\lgroup`	$\rgroup$	`\rgroup`	$\lmoustache$	`\lmoustache`	$\rmoustache$	`\rmoustache`
$\arrowvert$	`\arrowvert`	$\Arrowvert$	`\Arrowvert`	$\bracevert$	`\bracevert`		

Table A.10: Miscellaneous Symbols.

$\dots$	`\dots`	$\cdots$	`\cdots`	$\vdots$	`\vdots`	$\ddots$	`\ddots`
$\hbar$	`\hbar`	$\imath$	`\imath`	$\jmath$	`\jmath`	ℓ	`\ell`
$\Re$	`\Re`	$\Im$	`\Im`	$\aleph$	`\aleph`	$\wp$	`\wp`
$\forall$	`\forall`	$\exists$	`\exists`	$\mho$	`\mho` [a]	∂	`\partial`
$'$	`'`	$\prime$	`\prime`	$\emptyset$	`\emptyset`	∞	`\infty`
∇	`\nabla`	$\triangle$	`\triangle`	$\Box$	`\Box` [a]	$\Diamond$	`\Diamond` [a]
$\bot$	`\bot`	$\top$	`\top`	$\angle$	`\angle`	$\surd$	`\surd`
$\diamondsuit$	`\diamondsuit`	$\heartsuit$	`\heartsuit`	$\clubsuit$	`\clubsuit`	$\spadesuit$	`\spadesuit`
$\neg$	`\neg` or `\lnot`	$\flat$	`\flat`	$\natural$	`\natural`	$\sharp$	`\sharp`

[a]Use the `latexsym` package to access this symbol

Table A.11: Non-Mathematical Symbols.

These symbols can also be used in text mode.

†	`\dag`	§	`\S`	©	`\copyright`
‡	`\ddag`	¶	`\P`	£	`\pounds`

Table A.12: AMS Delimiters.

$\ulcorner$	`\ulcorner`	$\urcorner$	`\urcorner`	$\llcorner$	`\llcorner`	$\lrcorner$	`\lrcorner`

Table A.13: AMS Greek and Hebrew.

$\digamma$	`\digamma`	$\varkappa$	`\varkappa`	$\beth$	`\beth`	$\daleth$	`\daleth`	$\gimel$	`\gimel`

Table A.14: AMS Binary Relations.

$\lessdot$	`\lessdot`	$\gtrdot$	`\gtrdot`	$\doteqdot$	`\doteqdot` or `\Doteq`
$\leqslant$	`\leqslant`	$\geqslant$	`\geqslant`	$\risingdotseq$	`\risingdotseq`
$\eqslantless$	`\eqslantless`	$\eqslantgtr$	`\eqslantgtr`	$\fallingdotseq$	`\fallingdotseq`
$\leqq$	`\leqq`	$\geqq$	`\geqq`	$\eqcirc$	`\eqcirc`
$\lll$	`\lll` or `\llless`	$\ggg$	`\ggg` or `\gggtr`	$\circeq$	`\circeq`
$\lesssim$	`\lesssim`	$\gtrsim$	`\gtrsim`	$\triangleq$	`\triangleq`
$\lessapprox$	`\lessapprox`	$\gtrapprox$	`\gtrapprox`	$\bumpeq$	`\bumpeq`
$\lessgtr$	`\lessgtr`	$\gtrless$	`\gtrless`	$\Bumpeq$	`\Bumpeq`
$\lesseqgtr$	`\lesseqgtr`	$\gtreqless$	`\gtreqless`	$\thicksim$	`\thicksim`
$\lesseqqgtr$	`\lesseqqgtr`	$\gtreqqless$	`\gtreqqless`	$\thickapprox$	`\thickapprox`
$\preccurlyeq$	`\preccurlyeq`	$\succcurlyeq$	`\succcurlyeq`	$\approxeq$	`\approxeq`

Table A.15: AMS Binary Relations Continued.

$\curlyeqprec$	\curlyeqprec	$\curlyeqsucc$	\curlyeqsucc	$\backsim$	\backsim
$\precsim$	\precsim	$\succsim$	\succsim	$\backsimeq$	\backsimeq
$\precapprox$	\precapprox	$\succapprox$	\succapprox	$\vDash$	\vDash
$\subseteqq$	\subseteqq	$\supseteqq$	\supseteqq	$\Vdash$	\Vdash
$\Subset$	\Subset	$\Supset$	\Supset	$\Vvdash$	\Vvdash
$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\backepsilon$	\backepsilon
$\therefore$	\therefore	$\because$	\because	$\varpropto$	\varpropto
$\shortmid$	\shortmid	$\shortparallel$	\shortparallel	$\between$	\between
$\smallsmile$	\smallsmile	$\smallfrown$	\smallfrown	$\pitchfork$	\pitchfork
$\vartriangleleft$	\vartriangleleft	$\vartriangleright$	\vartriangleright	$\blacktriangleleft$	\blacktriangleleft
$\trianglelefteq$	\trianglelefteq	$\trianglerighteq$	\trianglerighteq	$\blacktriangleright$	\blacktriangleright

Table A.16: AMS Arrows.

$\dashleftarrow$	\dashleftarrow	$\dashrightarrow$	\dashrightarrow	$\multimap$	\multimap
$\leftleftarrows$	\leftleftarrows	$\rightrightarrows$	\rightrightarrows	$\upuparrows$	\upuparrows
$\leftrightarrows$	\leftrightarrows	$\rightleftarrows$	\rightleftarrows	$\downdownarrows$	\downdownarrows
$\Lleftarrow$	\Lleftarrow	$\Rrightarrow$	\Rrightarrow	$\upharpoonleft$	\upharpoonleft
$\twoheadleftarrow$	\twoheadleftarrow	$\twoheadrightarrow$	\twoheadrightarrow	$\upharpoonright$	\upharpoonright
$\leftarrowtail$	\leftarrowtail	$\rightarrowtail$	\rightarrowtail	$\downharpoonleft$	\downharpoonleft
$\leftrightharpoons$	\leftrightharpoons	$\rightleftharpoons$	\rightleftharpoons	$\downharpoonright$	\downharpoonright
$\Lsh$	\Lsh	$\Rsh$	\Rsh	$\rightsquigarrow$	\rightsquigarrow
$\looparrowleft$	\looparrowleft	$\looparrowright$	\looparrowright	$\leftrightsquigarrow$	\leftrightsquigarrow
$\curvearrowleft$	\curvearrowleft	$\curvearrowright$	\curvearrowright		
$\circlearrowleft$	\circlearrowleft	$\circlearrowright$	\circlearrowright		

Table A.17: AMS Negated Binary Relations and Arrows.

$\nless$	`\nless`	$\ngtr$	`\ngtr`	$\varsubsetneqq$	`\varsubsetneqq`
$\lneq$	`\lneq`	$\gneq$	`\gneq`	$\varsupsetneqq$	`\varsupsetneqq`
$\nleq$	`\nleq`	$\ngeq$	`\ngeq`	$\nsubseteqq$	`\nsubseteqq`
$\nleqslant$	`\nleqslant`	$\ngeqslant$	`\ngeqslant`	$\nsupseteqq$	`\nsupseteqq`
$\lneqq$	`\lneqq`	$\gneqq$	`\gneqq`	$\nmid$	`\nmid`
$\lvertneqq$	`\lvertneqq`	$\gvertneqq$	`\gvertneqq`	$\nparallel$	`\nparallel`
$\nleqq$	`\nleqq`	$\ngeqq$	`\ngeqq`	$\nshortmid$	`\nshortmid`
$\lnsim$	`\lnsim`	$\gnsim$	`\gnsim`	$\nshortparallel$	`\nshortparallel`
$\lnapprox$	`\lnapprox`	$\gnapprox$	`\gnapprox`	$\nsim$	`\nsim`
$\nprec$	`\nprec`	$\nsucc$	`\nsucc`	$\ncong$	`\ncong`
$\npreceq$	`\npreceq`	$\nsucceq$	`\nsucceq`	$\nvdash$	`\nvdash`
$\precneqq$	`\precneqq`	$\succneqq$	`\succneqq`	$\nvDash$	`\nvDash`
$\precnsim$	`\precnsim`	$\succnsim$	`\succnsim`	$\nVdash$	`\nVdash`
$\precnapprox$	`\precnapprox`	$\succnapprox$	`\succnapprox`	$\nVDash$	`\nVDash`
$\subsetneq$	`\subsetneq`	$\supsetneq$	`\supsetneq`	$\ntriangleleft$	`\ntriangleleft`
$\varsubsetneq$	`\varsubsetneq`	$\varsupsetneq$	`\varsupsetneq`	$\ntriangleright$	`\ntriangleright`
$\nsubseteq$	`\nsubseteq`	$\nsupseteq$	`\nsupseteq`	$\ntrianglelefteq$	`\ntrianglelefteq`
$\subsetneqq$	`\subsetneqq`	$\supsetneqq$	`\supsetneqq`	$\ntrianglerighteq$	`\ntrianglerighteq`
$\nleftarrow$	`\nleftarrow`	$\nrightarrow$	`\nrightarrow`	$\nleftrightarrow$	`\nleftrightarrow`
$\nLeftarrow$	`\nLeftarrow`	$\nRightarrow$	`\nRightarrow`	$\nLeftrightarrow$	`\nLeftrightarrow`

Table A.18: AMS Binary Operators.

$\dotplus$	`\dotplus`	$\centerdot$	`\centerdot`	$\intercal$	`\intercal`
$\ltimes$	`\ltimes`	$\rtimes$	`\rtimes`	$\divideontimes$	`\divideontimes`
$\Cup$	`\Cup` or `\doublecup`	$\Cap$	`\Cap` or `\doublecap`	$\smallsetminus$	`\smallsetminus`
$\veebar$	`\veebar`	$\barwedge$	`\barwedge`	$\doublebarwedge$	`\doublebarwedge`
$\boxplus$	`\boxplus`	$\boxminus$	`\boxminus`	$\circleddash$	`\circleddash`
$\boxtimes$	`\boxtimes`	$\boxdot$	`\boxdot`	$\circledcirc$	`\circledcirc`
$\leftthreetimes$	`\leftthreetimes`	$\rightthreetimes$	`\rightthreetimes`	$\circledast$	`\circledast`
$\curlyvee$	`\curlyvee`	$\curlywedge$	`\curlywedge`		

Table A.19: AMS Miscellaneous.

$\hbar$	`\hbar`	$\hslash$	`\hslash`	$\Bbbk$	`\Bbbk`
$\square$	`\square`	$\blacksquare$	`\blacksquare`	$\circledS$	`\circledS`
$\vartriangle$	`\vartriangle`	$\blacktriangle$	`\blacktriangle`	$\complement$	`\complement`
$\triangledown$	`\triangledown`	$\blacktriangledown$	`\blacktriangledown`	$\Game$	`\Game`
$\lozenge$	`\lozenge`	$\blacklozenge$	`\blacklozenge`	$\bigstar$	`\bigstar`
$\angle$	`\angle`	$\measuredangle$	`\measuredangle`	$\sphericalangle$	`\sphericalangle`
$\diagup$	`\diagup`	$\diagdown$	`\diagdown`	$\backprime$	`\backprime`
$\nexists$	`\nexists`	$\Finv$	`\Finv`	$\varnothing$	`\varnothing`
$\eth$	`\eth`	$\mho$	`\mho`		

Table A.20: Math Alphabets.

Example	Command	Required package
ABCdef	`\mathrm{ABCdef}`	
ABCdef	`\mathit{ABCdef}`	
$\mathnormal{ABCdef}$	`\mathnormal{ABCdef}`	
$\mathcal{ABC}$	`\mathcal{ABC}`	
$\mathscr{ABC}$	`\mathcal{ABC}`	`eucal` with option: or
	`\mathscr{ABC}`	`eucal` with option: mathscr
$\mathfrak{ABCdef}$	`\mathfrak{ABCdef}`	`eufrak`
$\mathbb{ABC}$	`\mathbb{ABC}`	`amsfonts` or `amssymb`

APPENDIX B

Examples of Java Code

Here are some examples of Java source using the `listings` package. I have entered the following before any code examples to format the code as shown.

```
\lstset{language=java}
\lstset{backgroundcolor=\color{white},rulecolor=\color{black}}
\lstset{linewidth=.95\textwidth,breaklines=true}
\lstset{commentstyle=\textit,stringstyle=\upshape,showspaces=false}
\lstset{frame = trbl, frameround=tttt}
\lstset{numbers=left,numberstyle=\tiny,basicstyle=\small}
\lstset{commentstyle=\normalfont\itshape,breakautoindent=true}
\lstset{abovecaptionskip=1.2\baselineskip,xleftmargin=30pt}
\lstset{framesep=6pt}
```

I have included the code by entering

```
\begin{singlespace}
\lstinputlisting[caption=Clock Code,label=clock]{source/Clock.java}
\end{singlespace}
```

Listing B.1: Clock Code

```
// file: Clock.java
public class Clock extends UpdateApplet {
    public void paint( java.awt.Graphics g ) {
        g.drawString( new java.util.Date().toString(  ), 10, 25 );
    }
}
```

Listing B.2: Consumer

```java
// file: Consumer.java
import java.util.Vector;

public class Consumer implements Runnable
{
    Producer producer;

    Consumer( Producer producer ) {
        this.producer = producer;
    }

    public void run() {
        while ( true ) {
            String message = producer.getMessage();
            System.out.println("Got message: "+ message);
                        try {
                                Thread.sleep( 2000 );
                        } catch ( InterruptedException e ) { }
        }
    }

    public static void main(String args[]) {
        Producer producer = new Producer();
        new Thread( producer ).start();
        Consumer consumer = new Consumer( producer );
        new Thread( consumer ).start();
    }
}
```

Listing B.3: EvilEmpire Code

```java
// file: EvilEmpire.java
import java.net.*;

public class EvilEmpire {
  public static void main(String[] args) throws Exception{
    try {
      Socket s = new Socket("???.???.???.???", 80);
      System.out.println("Connected!");
    }
    catch (SecurityException e) {
      System.out.println("SecurityException: could not connect.");
    }
  }
}
```

APPENDIX C

C++ Examples

This appendix demonstrates the `listings` packages ability to format C++ code.

Listing C.1: Motion Class

```
#include "Motion.h"


Motion::Motion(int _steps) : TimeSeries(_steps) {}

Motion::Motion(Noise2 *_noise) : TimeSeries(_noise->GetSteps()) {
  noise = _noise;
}

Motion::~Motion() {
  delete noise;
}

void Motion::SyncWithNoise() {
  if (noise != NULL) {
    this->Initialize();
    double sum = 0;
    int getsteps = this->GetSteps();
    for (int i = 0; i < getsteps; i++) {
      sum += noise->GetData(i);
      this->SetData(i, sum);
    }
  } else {
    fprintf(stderr, "%s\n", MOTION_NOISE_ERR);
  }
}
```

Listing C.2: Plotter Class

```
#include <unistd.h>
#include "Plotter.h"


void Plotter::MakePlot(char *filename) {
  ofstream fout(FILE_PLOT);
  fout << "set data style linespoints" << endl
       << "plot \"" << filename << "\"" << endl;
  fout.close();

  int pid, status;
  pid = fork();
  if (pid >= 0) {
    if (pid == 0) {
      execl(FILE_GNUPLOT, "gnuplot", "-persist", FILE_PLOT, NULL);
      fprintf(stderr, "%s \"gnuplot\"", EXEC_ERR);
      exit(0);
    } else {
      wait(status);
    }
  } else {
    fprintf(stderr, "%s \"gnuplot\"", FORK_ERR);
  }

  /*  pid = fork();
  if (pid >= 0) {
    if (pid == 0) {
      execlp("rm", FILE_PLOT, NULL);
      fprintf(stderr, "%s \"rm\"", EXEC_ERR);
      exit(0);
    } else {
      wait(status);
    }
  } else {
    fprintf(stderr, "%s \"rm\"", FORK_ERR);
    }*/

}
```

Listing C.3: Simulation Class

```
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

Listing C.4: Simulation Class

```cpp
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

Listing C.5: Simulation Class

```cpp
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

Listing C.6: Simulation Class

```cpp
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

Listing C.7: Simulation Class

```cpp
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

Listing C.8: Simulation Class

```cpp
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

Listing C.9: Simulation Class

```cpp
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

Afterword

So how does a LaTeX session work? LaTeX loads the document class with any specified options and uses the information in the document class to decide on how the document will be formatted. At this point LaTeX loads any packages that the user has specified. Packages extend the basic LaTeX commands and formatting for special situations. `woosterthesis` loads a number of packages by default and it is assumed you have these installed on your system. They are: `ifpdf`, `textpos`, `geometry`, `amsthm`, `amssymb`, `amsmath`, `setspace`, `fancyhdr`, `graphicx`, `eso-pic`, `listings`, `natbib`, `makeidx`, `verbatim`, `lettrine`, `alltt`, `fontenc`, `pxfonts`, `floatflt`, `float`, `caption`, `subfigure`, and `ifthen`. The `woosterthesis` class assumes you are using pdfTeX (support for postscript based TeX has been dropped as of 2006/17/11).

The `hyperref` package will make your thesis a linked document. `amsthm` is for altering the Theorem environments. `amsmath` implements almost all of the mathematical symbols. `amssymb` adds the mathematical symbols not present in `amsmath`. `graphicx` and `eso-pic` are used to place graphics files in the thesis. `geometry` is used to set up the margins for the thesis. `setspace` is used to alter spacing by allowing a `singlespace`, `doublespace`, and `onehalfspace` environments. `natbib` formats references in parentheses with author and year. Documentation is included for some of the packages in the `doc` folder.

These packages should all be installed with a full installation of TeXLive on OS X or XP. On OS X one can use the the MacTeX installer as i-Installer is no longer supported as of 2007/1/1. On XP/Vista one can use MikTeX to install all available packages which will install all of the above. By default the MikTeX install does a minimal installation. You will need to run the updater to make your MikTeX installation aware of all the new packages.

There is also a new TeX engine called XeTeX which allows one to use the native fonts on your system as text fonts in the document. More information can be found at the XeTeX homepage. If using XeTeX you will also need `fontspec` and `xltxtra` which should be installed with XeTeX .

Once the packages are loaded, LaTeX begins to process the commands contained between the `document` tags. As it processes the commands, a number of auxiliary files are created. These files contain information needed for things like the Bibliography, Table of Contents, List of Figures, etc. We then process the file a second time to allow LaTeX to use its auxiliary files to fill in information. Some information may require three passes before it is displayed. Once LaTeX is done you are presented with a PDF of the output.

References

Index