GIT101

Contributing to gitMechanics

Basics of LaTeX, git, and How You Can Help

Arizona State University

Brian Chevalier

Updated: November 27, 2019

Contents

Welcome to the gitMechanics Team!

This document is a beginners introduction to the information needed to contribute to gitMechanics. We will cover the basics of git, LaTeX, and we'll end on the steps you need to take to make a contribution.

All suggestions, corrections, or new material is very welcome, no matter how small. The *beauty* of open-source is that all contributions make a permanent difference. If you have no experience with any of these things don't be intimidated! Feel free to reach out for any clarification or help you need. I'm so happy you've even considered helping.

1 Introduction to git

Git is an open-source, distributed *version control system* originally built by Linus Torvalds for managing contributions to Linux. Git is now one of the world's most used version control systems (all of Windows is now managed with git), one of the primary places git repositories are stored is on github.com (which is where this document lives). At it's core all git does is keep track of changes in files, but particularly plain text files. Plain text files can have many file extensions such as txt, md, tex, etc. We'll talk about these more later.

Github offers many courses on using the service. They features videos, text content, and interactive learning. You can check them out here:

https://lab.github.com/courses

Git can be used through two primary ways: via a graphical user interface (GUI) or through the command line. Git is available on all platforms (even iOS, Android, and chromeOS. On iOS I highly recommend Working Copy). Before we talk about actually using git we'll look at some basic terminology.

1.1 Vernacular

There are a few concepts that you'll want to understand in general before getting started with the particulars of using git. Some of these words can be very confusing for beginners so we're going to lay out some words that you'll want to be familiar with.

Repository Also called a "repo". This is a directory that stores your source files. There is a special folder called ".git" which contains the metadata about the repo. Folders that start with a dot will be hidden in most files systems.

Clone When you see a project on github you can clone that project. This is a full copy of a repository. Every repo is a clone which is why git is a distributed version control system.

Fork A fork is a copy of a repo that is associated with your github account. This gives you a repo that you can edit and push changes to. Typically you will *fork* someone's repo on github.com, then *clone* that repo to your local

machine. In effect, all repos are clones and forks because it is a distributed network.

Commit After you make changes to a repo you'll want to commit those changes. This is essentially like a bookmark in your repo that holds the state of your repo. Every commit must be made with a message (make sure it's useful!) and optional additional comments.

Stage Before you actually make a commit you will will have to "stage" or decide which files will be included with the commit. If you're using a GUI git tool this may just be a step while making a commit. On the command line you have to stage the files you are going to commit (this is done with git add filename or git add -a for all files in the repo to be added) and then commit them.

Fetch When you're working with a repo you typically have a version of that repo on a server that you want to route your changes through. If someone else has made changes to that repo or you made changes on another device you have to fetch changes, then merge them into your repo and push those changes back to the server.

Push You push changes to a server. You can push multiple commits at a time, so you don't have to push every time you make a commit or change a file.

Pull A pull is a combination of fetch and merge. You'll typically want to avoid pulling changes into your repo because you want to manually fetch and then merge in changes.

Pull Request This is a request made by someone to suggest changes to a repo that they do not have write access to. This is very common for open source projects so changes to the canonical source code is vetted by project maintainers.

Branch A repo starts with one branch called the master. You can create as many branches as you want. For instance, you may want to make a new branch that includes

Merge Conflict This is the error that occurs when two that git attempts to combine have different text on the same line number that it cannot reconcile. This requires manual fixing but does not often happen.

Diff A diff is to show the differences between two or more files. Depending on your knowledge, a GUI is likely better for seeing diffs since it will be easier to understand.

Checkout The power of git is that you can checkout previous commits of a repo. As long as all the files in your repo are already committed, you can checkout another branch or another commit. This eliminates the fear of making destructive changes. As long as you commit changes regularly and make descriptive commit messages you will always be able to return to a known change.

Cherrypick Sometimes when you're working with multiple branches you'll want to choose a commit from one branch and apply it onto another. This is what a cherrypick does. You will not likely need to do this.

1.2 Git on the Command Line

If you choose to use a command line tool for git it is very important you understand what these words mean as many

of them are the names of commands you'll want to use when managing your repo. The most common are clone, add, commit, fetch and push. The typical steps you make when working on a repo is: clone someone else's repo. Make changes. Fetch changes from the server to see if any changes have been made there and merge in the changes. Add your files to a commit, then make the commit. Finally, you can push your files to the server. This can be accomplished by the following commands:

```
git clone [URL]
git fetch
git merge
git add --all
git commit -m "commit message"
git push
```

Note that you'll want to replace [URL] with the URL of the repo that you are cloning. You will enter each line for each part of the process.

Here is a great video on YouTube with an in depth example on using git from the command line. Even if you don't use the command line it is great for understanding how git works.

https://youtu.be/0fKg7e37bQE

1.3 GUI git Clients

Many text editors include git integration out of the box including MATLAB and VSCode.

Github provides a GUI client for users on macOS and Windows:

https://desktop.github.com

and provides full guides on using their application:

https://help.github.com/desktop/guides/

1.4 Making a Pull Request

Now we will take a look at making a *pull request*. A pull request is your method of suggesting changes to someone's work. Typically, big open-source projects are not just a Wikipedia-style free-for-all. Only project maintainers have *write access* to the canonical version of Python, for instance. However, community members *are* able to fork a copy of the repository to their own github. This allows write access to your own version of the repo. After you make changes to your code, you can then make a pull request on the original repo. This ensures that the code can be vetted by automated testing (i.e. continuous integration), and further tested by humans.

The following video shows how to fork a public repo, make changes to that repo where you have write access, then suggest changes via a pull request to a project.

https://youtu.be/YTbRzhQju4c

2 Introduction to LaTeX

LaTeX (pronounced lay-teck) is a free, open-source high-quality document preparation system (this document is written and compiled with LaTeX!). TeX was written and designed by Donald Knuth in 1978. LaTeX builds on top of the work of Knuth providing more functionality through a more advanced set of macros.

2.1 Vernacular

Macro A macro is a LaTeX command that can take input and does something with it. You can define any number of custom macros (also called commands) that can speed up your document preparation and make your style more consistent.

Package A package is a set of predefined LaTeX macros packaged up for your use. Users can create their own packages. gitMechanics relies on a few custom packages.

Document class There are many document classes that you can use, and you can even define your own classes. Classes include: article, standalone, beamer (slide presentation format), report, book, etc. Each class provides a set of commands to typeset your work.

Environments Text in an environment is processed in a particular way by the compiler. Inside an environment particular types of commands are available for use. The entire body of the document is wrapped in the document environment. There are also list environments, math environments, floating environments, and of course you can define custom environments.

WYSIWYG Pronounced wizzy-wig. This stands for what you see is what you get. This is what GUI editors do. Some people refer to LaTeX as what you see is what you want.

2.2 Getting LaTeX on Your Machine

Your first step is to install a distribution of LaTeX. This will be different depending on your platform, however it is available on all platforms (you can use the Linux distribution on ChromeOS. On iOS you can get a dedicated app, I reccomend TeX Writer). The following URL includes links on where to install the different distributions.

https://www.latex-project.org/get/

Most also come with a GUI editor and built in methods for compiling your documents so you don't have to touch the command line. Each editor will be slightly different so you'll have to spend a while getting to know your editor. Note that you can also opt to use your own text editor and compile from the command line using pdflatex. Regardless, make sure your text editor has good code completion support for LaTeX otherwise it will make learning commands a nightmare.

2.3 Compiling Your First Document

The most basic document needs two things. First you have to define your "document class". As discussed before, there are many classes. Document classes can also take optional inputs in square brackets before the actual document class. This document is an article class document and the options are landscape, twocolumn, and 12pt. The options must be separated by commas. The example code below typsets a blank document. Note that the article declaration is on the second line, which is completely valid, but is also done to fit within the column margins. Also note that your document will be within the document environment, and that the % sign indicates a comment.

```
\documentclass[landscape, twocolumn, 12pt]
{article}

\begin{document}
% Your document goes here
\end{document}
```

2.4 Document Structure

Documents are split into different pieces. This includes (from highest level, to lowest): part, chapter, section, subsection, subsubsection, paragraph, subparagraph. The highest level in this document is a section and then subsection. Table 1 shows the various header types available. Note that "part" is only available in books and reports.

Table 1: Header levels

Header	Level
part	-1
chapter	0
section	1
subsection	2
subsubsection	3
paragraph	4
subparagraph	5

New headers can be added similar to the following:

```
\section{section name}
\subsection*{subsection name}
```

This code adds a new numbered section with the name "section name" and unnumbered subsection that does not show up in the table of contents called "subsection name".

Note: subsubsections are used in this document, however they are suppressed from the table of contents since it would make the table of contents too long. You can choose the level shown in the table of contents by the following:

```
\setcounter{tocdepth}{2}
```

where the number is the lowest depth the table of contents goes to.

2.5 Text Commands

LaTeX has a markup syntax similar to what you might be familiar with HTML. In a text environment (document environment, itemize, etc.) You can use:

```
\textit{}
```

to *italicize* text. You can use

```
\textbf{}
```

to write **bold face** text.

2.6 Text Environments

There are many text environments available in LaTeX to typeset lists. You can typeset a basic bullet-point list with the following itemize environment:

```
\begin{itemize}
\item Item 1
\item Item 2
\end{itemize}
```

Note that each item must be prepended with the item command followed by a space then the item. You can typeset a numbered list in the enumerate environment:

```
\begin{enumerate}
\item Item 1
\item Item 2
\end{enumerate}
```

You can nest lists by adding the environment you want nested. You can also customize the bullet points and how the list is typeset.

You can typeset a list of definitions using the *description* environment. Each item is the optional input item in the item command like the following:

```
\begin{description}
\item[Word] Description 1
\item[Phrase] Description 2
\end{description}
```

2.7 Loading Packages

Standard LaTeX is good, however, you will likely need to load additional packages to extend the language. You can import a package like the following:

```
\usepackage{amsmath}
```

You will want to typically use the AMSMath package since it adds many commonly used math symbols. You can find full documentation on the package at:

https://www.ctan.org/pkg/amsmath

The Comprehensive TeX Archive Network (CTAN) has many packages and documentation for packages that are typically already included with your LaTeX distribution. Most packages have the full documentation in PDF format available on CTAN.

2.8 Equations in LaTeX

If there's one thing LaTeX is known for it's typesetting math and equations. We're going to get into that now. For math to be added to your document you have to be in some sort of math mode. This means you could be in a math environment or have your math inline with delimiters.

Math environments include: align, equation, or a few more. Equations in this environment will automatically be centered and numbered. You can suppress numbering of these environments by adding "*" after the environment name. The following equation is typeset using the equation environment:

$$\int_0^L f(x)dx \tag{1}$$

```
\begin{equation}
\int_{0}^{L} f(x) dx
\end{equation}
```

Note that the braces around the limits of integration are not strictly needed for single character limits, however it is best practice to keep them in braces to avoid confusion and be consistent for more complicated equations.

The next important math environment is the align environment. The align environment allows typesetting multiple consecutive equations aligned. The following is an example of that:

$$y = mx + b \tag{2}$$
$$= \frac{\Delta y}{\Delta x}x + b \tag{3}$$

The syntax looks like:

```
\begin{align}
y &= mx + b\\
  &= \frac{\Delta y}{\Delta x}x + b
\label{Eq:ex1}
\end{align}
```

Things to note: the equations are aligned based on the location of the "&" and new lines are entered with "\". Greek letters and other commands also require a space after the command name otherwise it will throw an error.

You can insert any equation *inline* anywhere in your document simply by wrapping the equation in dollar signs. For instance, you can reference equation 3 and say the variable Δ means the change in the following variable.

2.8.1 Lableing and Referencing

One of the benefits with LaTeX is that it has automated equation numbering, referencing and hyperlinking. When you write an equation, within the equation environment you need to provide a label with a name for that object like the following:

```
\label{label-name}
```

Once you have labeled an equation you can reference that equation number in your text with the following:

```
\ref{label-name}
```

It is best practice to begin an equation label with "Eq:" since labels include table and figure labels (same for "Tab" and "Fig"). The process for labeling and referencing figures and tables is the same. Add a label within the respective environment, then reference the object somewhere in your text.

Note that you may have to run LaTeX more than once for the object number to appear in your text. This is because LaTeX builds a file in your current working directory with the information about your labeled information. Just run LaTeX again and it will update your text. The same is true for the table of contents.

All internal references can automatically become hyperlinked by importing the "hyperref" package. This will make the

table of contents and any referenced object clickable and will redirect to the referenced object. This is already imported by default in gitMechanics documents since the custom TitlePage package uses the hyperref package so there's no need to add that to your preamble.

2.8.2 Greek Letters

Most Greek letters can be included in math mode typically by typing "\" followed by the name of the letter such as typesetting the letter beta: β. The following PDF lists many Greek letters and math mode symbols.

https://wch.github.io/latexsheet/latexsheet.pdf

2.8.3 Operators

There are many common math operators such as trigonometric functions:

```
\sin \csc
\cos \sec
\tan \cot
```

2.8.4 Typsetting Tables

Tables in LaTeX can be difficult to produce and debug. This would be a great reason to employ a WYSIWYG table generator such as:

https://tablesgenerator.com

I have my own custom iOS extension that converts .csv files into LaTeX markup (available upon request, maybe one day I'll write a blog post).

3 Drawing in LaTeX

LaTeX produces truly stunning PDF output, but as the great philosopher, Beyoncé once said, "Pretty Hurts".

GUI Options There are some options to avoid manually marking up drawings. One method is to simply produce figures and drawings in another application, and export as a PDF, or if you like the pixelated compression artifacts, JPEG. You can then save the file in the Figures subdirectory of your project and include something like the following:

```
\begin{figure}[H]
\centerline{
\includegraphics[width=\columnwidth]
{Figures/FIGNAME}}
\caption{CAPTION}
\label{fig:LABEL}
\end{figure}
```

You can also use a GUI editor that will output to markup formats and include those directly into your source code. You can export to TikZ format which we will discuss in the next section. A great option for this is GeoGebra:

https://www.geogebra.org

3.1 TikZ

While including external images works, it can get out of control pretty quickly if you have many figures. It also eliminates the possibility of using a set of macros to keep drawings consistent throughout many documents. Using generated TikZ still runs into this problem. Therefore, the only good option is to learn TikZ markup. For gitMechanics, I have already begun building a library of macros that will hopefully make the processs more straight-forward and consistent between documents.

The following site has many example TikZ examples:

http://www.texample.net/tikz/examples/

3.1.1 TikZ Math

The TikZ Math library is a great way to use math to better geometrically describe your diagrams. The direct link to the documentation can be found at:

http://ctan.mirrors.hoobly.com/graphics/pgf/base/doc/pgfmanual.pdf#page640

You must load the tikzmath library using the following:

```
\usetikzlibrary{math}
```

Then you can define variables and evaluate math expressions in the tikzmath command. Each statement **must** be terminated by a semi-colon, and the variable should not be an existing variable used in your LaTeX file or used packages.

4 Making your Contribution

Before you begin, you will need: a way to use git (command line or GUI option), a way to compile LaTeX documents (command line or GUI option) and a github account.

The following steps are the steps you need to take to make your first contribution to the gitMechanics repository:

- ☐ Fork the main gitMechanics repo
- ☐ Clone the fork from your account to your local machine
- ☐ Make changes to the source code and recompile your PDFs as needed
- ☐ Commit changes to your local repo and push changes to your github server repo
- ☐ Once you want to make a change to the website, go to github and make a pull request on the main repo

4.1 How the repo is organized

Figure 1 shows the typical directory of a courses files and will be used to explain how the repo is organized.

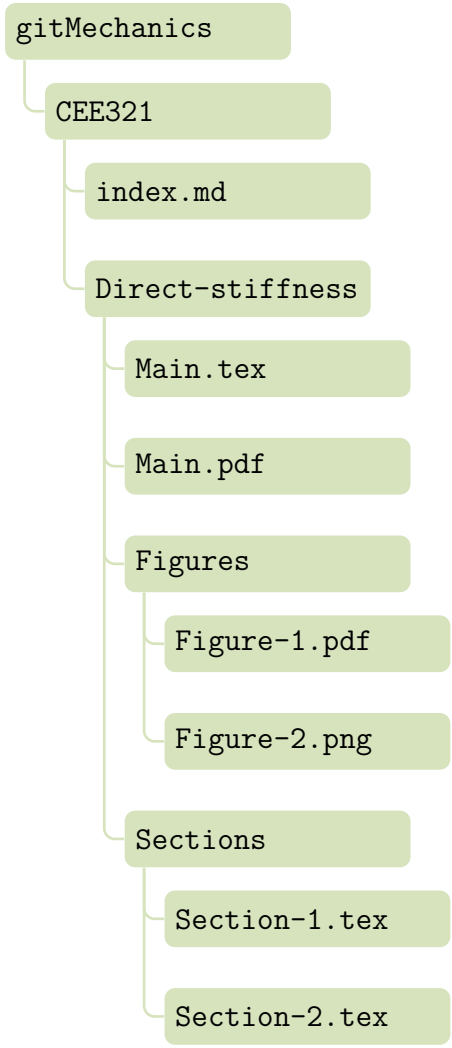

Figure 1: Directory of a typical course

Appendices

A How the Website is Compiled

Jekyll is a blog-aware static site generator. Github supports hosting websites with custom domains using the Jekyll blog-

ging platform. The major benefit to this (other than that it is free), is that anytime the master branch of the github repo is updated via a commit, github automatically rebuilds the site using Jekyll and makes it live. This is great because all of the source documents can be stored in the same repository as the source files that build the website. It also has the added benefit of automatically rebuilding itself if someone makes a pull request on the repo and it is accepted.

B LaTeX Symobls

α	\alpha	θ	\theta	o	o
β	\beta	ϑ	\vartheta	π	\pi
γ	\gamma	γ	\gamma	ϖ	\varpi
δ	\delta	κ	\kappa	ρ	\rho
ϵ	\epsilon	λ	\lambda	ϱ	\varrho
ε	\varepsilon	μ	\mu	σ	\sigma
ζ	\zeta	ν	\nu	ς	\varsigma
η	\eta	ξ	\xi		
Γ	\Gamma	Λ	\Lambda	Σ	\Sigma
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon
Θ	\Theta	Π	\Pi	Φ	\Phi
τ	\tau	υ	\upsilon	ϕ	\phi
φ	\varphi	χ	\chi	ψ	\psi
ω	\omega	Ψ	\Psi	Ω	\Omega

Table 2: Greek Letters

$\hat{a}$	\hat{a}	$\acute{a}$	\acute{a}
$\check{a}$	\check{a}	$\grave{a}$	\grave{a}
$\dot{a}$	\dot{a}	$\breve{a}$	\breve{a}
$\ddot{a}$	\ddot{a}	$\tilde{a}$	\tilde{a}
$\bar{a}$	\bar{a}	$\mathbf{a}$	\vec{a}

Table 3: Math mode accents

$\pm$	\pm	$\cap$	\cap	$\diamond$	\diamond
$\mp$	\mp	$\cup$	\cup	$\triangle$	\bigtriangleup
$\times$	\times	$\uplus$	\uplus	$\triangledown$	\bigtriangledown
$\div$	\div	$\sqcap$	\sqcap	$\triangleleft$	\triangleleft
$*$	\ast	$\sqcup$	\sqcup	$\triangleright$	\triangleright
$\star$	\star	$\vee$	\vee	$\lhd$	\lhd^b
$\circ$	\circ	$\wedge$	\wedge	$\rhd$	\rhd^b
$\bullet$	\bullet	$\setminus$	\setminus	$\unlhd$	\unlhd^b
$\cdot$	\cdot	$\wr$	\wr	$\unrhd$	\unrhd^b
$+$	+	$-$	-	$\amalg$	\amalg
$\oplus$	\oplus	$\ominus$	\ominus	$\otimes$	\otimes
$\oslash$	\oslash	$\odot$	\odot	$\bigcirc$	\bigcirc
$\dagger$	\dagger	$\ddagger$	\ddagger		

Table 4: Binary Operation Symbols

$\leq$	\leq	$\geq$	\geq	$\equiv$	\equiv
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq
$\ll$	\ll	$\gg$	\gg	$\asymp$	\asymp
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong
$\sqsubset$	\sqsubset^b	$\sqsupset$	\sqsupset^b	$\neq$	\neq
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\doteq$	\doteq
$\in$	\in	$\ni$	\ni	$\propto$	\propto
$\vdash$	\vdash	$\dashv$	\dashv	$<$	<
$:$	:	$\models$	\models	$\perp$	\perp
$\mid$	\mid	$\parallel$	\parallel	$\bowtie$	\bowtie
$\Join$	\Join^b	$\smile$	\smile	$\frown$	\frown
$=$	=	$>$	>		

Table 5: Relation Symbols

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown
$\rightleftharpoons$	\rightleftharpoons	$\leadsto$	\leadsto^b
$\uparrow$	\uparrow	$\Updownarrow$	\Updownarrow
$\Uparrow$	\Uparrow	$\nearrow$	\nearrow
$\downarrow$	\downarrow	$\searrow$	\searrow
$\Downarrow$	\Downarrow	$\swarrow$	\swarrow
$\updownarrow$	\updownarrow	$\nwarrow$	\nwarrow

Table 6: Arrow Symbols

$\widetilde{abc}$	\widetilde{abc}	$\widehat{abc}$	\widehat{abc}
$\overleftarrow{abc}$	\overleftarrow{abc}	$\overrightarrow{abc}$	\overrightarrow{abc}
$\overline{abc}$	\overline{abc}	$\underline{abc}$	\underline{abc}
$\overbrace{abc}$	\overbrace{abc}	$\underbrace{abc}$	\underbrace{abc}
$\sqrt{abc}$	\sqrt{abc}	$\sqrt[n]{abc}$	\sqrt[n]{abc}
f'	f'	$\frac{abc}{xyz}$	\frac{abc}{xyz}

Table 7: Some other constructions

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

Table 8: Variable-sized Symbols

$\ldots$	\ldots	$\cdots$	\cdots	$\vdots$	\vdots
$\ddots$	\ddots	∞	\infty	$\triangle$	\triangle
$\aleph$	\aleph	$\prime$	\prime	$\forall$	\forall
$\hbar$	\hbar	$\emptyset$	\emptyset	$\exists$	\exists
$\imath$	\imath	∇	\nabla	$\neg$	\neg
$\jmath$	\jmath	$\surd$	\surd	$\flat$	\flat
ℓ	\ell	$\top$	\top	$\natural$	\natural
$\wp$	\wp	$\bot$	\bot	$\sharp$	\sharp
$\Re$	\Re	$\|$	|	$\backslash$	\backslash
$\Im$	\Im	$\angle$	\angle	∂	\partial
$\mho$	\mho[b]	.	.	$\|$	|

Table 9: Miscellaneous Symbols

$($	(	$)$	)	$\uparrow$	\uparrow
$\Uparrow$	\Uparrow	$\Downarrow$	\Downarrow	$\Updownarrow$	\Updownarrow
$[$	[	$]$	]	$\downarrow$	\downarrow
$\{$	\{	$\}$	\}	$\updownarrow$	\updownarrow
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil
$\langle$	\langle	$\rangle$	\rangle	$/$	/
$\|$	|	$\|\|$	\\|	$\rceil$	\rceil
$\backslash$	\backslash				

Table 10: Delimiters