Symbolic Logic and LaTeX

David W. Agler

June 21, 2013

1 Introduction

This document introduces some features of LaTeX, the special symbols you will need in Symbolic Logic (PHIL012), and some reasons for why you should use LaTeX over traditional word processing programs. This video also accompanies several video tutorials on how to use LaTeX in the Symbolic Logic (PHIL012) course at Penn State.

2 LaTeX is not a word processor

LaTeX is a typesetting system that was originally designed to produce documents with special symbols. LaTeX is not a word processor which coordinates text with styling simultaneously. Instead, the LaTeX system separates text from styling as it consists of (i) a text document (or *.tex file) which is text that does not contain any formatting and (ii) a compiler that takes the *.tex file and turns it into a readable and professionally stylized document. In other words, the production of a document using LaTeX begins with the specification of *what you want to say* and *the structure of what you want to say* and this is then processed by a compiler which styles your document. To get a clearer idea of what some of this means, let's look at a very simple LaTeX source document:

```
\documentclass[12pt]{article}
\begin{document}
\title{Symbolic Logic and \LaTeX\ }
\author{David W. Agler}
```

```
\date{\today}
\maketitle
\section{Introduction}
Hello Again World!
\end{document}
```

The above specifies the class (or kind) of document you want to produce (an article, rather than a book or letter), commands for begining the document, the title of the article, the author of the article, the date, a command to make the title, a section that will automatically be numbered, some content that will appear as text ("Hello World"), and finally a command to end the document.

What we see then is that the empsource document for LaTeX consists of not only content ("Hello World") but also the structure of the document.

2.1 Advantages to LaTeX

The first question you might have about LaTeX is this: *I already know how to use a word processor (e.g. WORD), what does LaTeX give me that I don't already have?* There are several reasons why you might want to use LaTeX in this course and beyond:

1. LaTeX gives you more control over mathematical symbols, formulas, and your document in general.

   But I'm not a mathematician, so why would I ever want to use LaTeX?

2. Some word-processors involve auto-formatting for lists, paragraphs, margins, and sections that are hard to override. Other times you want to create structured documents that contain sections, titles, a table of contents, and an index. LaTeX gives you more control over these functionalities.

   But even if your document doesn't have a lot of structure, there are some other reasons to use LaTeX!

3. LaTeX is cross-platform. Some word processing programs are platform specific.

4. There is a lot of user-generated documentation on how to use LaTeX. If you have a question, just google it!

5. LaTeX allows you to sharply separate the content of your document from its form. Once you've finished writing *what* you want to say, then you can apply a template to design that content.

6. LaTeX is free!

2.2 Disadvantages to LaTeX

Despite there being several benefits to using LaTeX there is a disadvantage. LaTeX has a learning curve. So, if you have a term paper due tonight, now is not the time to learn LaTeX. The learning curve is mostly due to the fact that LaTeX separates form from content and so you will need to learn specific commands to get LaTeX to display symbols and format your document the way you want it. For example, suppose you wanted to write the following formula in logic:

$(\exists x)(Px \rightarrow Lx)$

To write the above equation in LaTeX you would need to write the following:

```
$(\exists x)(Px \rightarrow Lx)$
```

In other word processors, to do the same thing, you will have to go searching through the different fonts available, create a shortcut (or hotkey) for these symbols, and then use these shortcut commands to input them into your document. This is not terribly difficult to do but (i) there are some serious limitations are limited in how you can control your equations and (ii) sometimes your formulas will disappear when you try to convert it to PDF or won't appear if you send it to a friend.

But, there are two things to be said concerning this:

1. Simple articles are pretty easy to write and so as long as you don't need some special features, learning how to use LaTeX isn't terribly difficult. All you need are a few commands and the LaTeX package.

2. The complexity of LaTeX isn't for complexity's sake. Separating the content and structure of LaTeX from styling is beneficial as the complexity of LaTeX is due to its immense flexibility. You can use it to create graphics, tables, chemical diagrams, and even music.

If you think that LaTeX is not for you, don't worry. This course has several tutorials on how to do logic in WORD. Or, you can even do your work in HTML.

2.3 How to Get LaTeX

There are several ways to get LaTeXẆhat you will need is the LaTeX system and a text editor. If you are unsure about whether you want to use the LaTeX system, a safe bet is to start with the online version at `www.scribtex.com`. Once you are at the site, click the button that reads "Go to our new LaTeX editor", from there you will need to sign up and then can start working from a *.tex file which can be easily compiled into a viewable PDF.

If you have decided that LaTeX is the way to go and are going to install it on your computer, here are the best sites to download the LaTeX system:

for Mac: `www.tug.org/mactex`

for Windows: `http://miktex.org`

While all of the above come with a default text editor, I really like the following text editors (and so highly recommend downloading it in addition to the above):

TeXstudio: `http://texstudio.sourceforge.net/`
TeXworks: `http://www.tug.org/texworks/`

The reason I recommend Texstudio is because the editor contains some buttons that allow for the easy creation and modification of tables.

3 Special Symbols

3.1 Propositional Logic

These symbols are created using the following LaTeX commands:

Sign	LaTeX Command	Name of Sign
$\neg$	`$\neg$`	unary not, negation
$\wedge$	`$\wedge$`	caret
$\vee$	`$\lor$`	the vee or wedge
$\leftarrow$	`$\leftarrow$`	left arrow
$\rightarrow$	`$\rightarrow$`	right arrow
$\leftrightarrow$	`$\leftrightarrow$`	double arrow
$\vdash$	`$\vdash$`	right turnstile "therefore"
$\dashv$	`$\dashv$`	left turnstile
$\models$	`$\models$`	models or semantic entailment

If you are using a different system, here are several other operators:

Sign	LaTeX Command	Name of Sign
$\sim$	`$\mathord{\sim}$`	unary not (the 'tilde')
&	`\& (in tables)`	ampersand
$\supset$	`$\supset$`	horseshoe
$\equiv$	`$\equiv$`	tribar
$\otimes$	`$\otimes$`	circled X

You may also want to subscript integers to propositional letters as follows:

Sign	LaTeX Command
P	`$P_1$`
P_1	`$P_1$`
P_2	`$P_2$`
P_{12}	`$P_{12}$`

3.2 Predicate Logic Symbols

In moving from predicate logic to propositional logic, you will need the following two symbols:

Existential Quantifier: $\exists$
Universal Quantifier: $\forall$

These are created using the following commands in LaTeX :

```
Existential Quantifier: $\exists$
```

```
Universal Quantifier: $\forall$
```

3.3 Modal Logic Symbols

In moving from propositional logic to modal logic, you will need the following two symbols:

$\Box$: modal 'box'
$\Diamond$: modal 'diamond'

4 Tables

Truth tables, trees, and proofs can be created using tables.

4.1 How To Create a Table

To create a table, the first thing you will need to do is open the table environment using the following command:

1	R	Premise 1
2	S	Premise 2
.	.	.
.	.	.
.	.	.
n	Conclusion	Justification

This table is created by writing

```
"\begin{tabular}{lll}" where "\begin{tabular}
```

opens up the table environment and where the number of rows is specified by the number of letters in the curly braces (in our case three). Here is the entire table above in LaTeX

```
\begin{tabular}{lll}
1 & R &Premise 1 \\
2 & S &Premise 2 \\
. & . &.\\
. & . &.\\
```

```
. & . &.\\
n & Conclusion & Justification \\
\end{tabular}
```

We can specify the justification of these columns by our choice of letter in the curly braces:

r column is right justified

c column is centered

l column is centered.

For example, changing "l" to "c" in the above gives us the following table:

1	R	Premise 1
2	S	Premise 2
.	.	.
.	.	.
.	.	.
n	Conclusion	Justification

Occasionally, we want to add vertical and horizontal lines. To create a vertical line we separate the letters in the braces with a pipe, which is the following symbol:

```
|
```

On many QWERTY keyboards, this symbol is located above the "Enter" key and typically is shared by the backslash key.

If we add a pipe after the first letter in the curly baces as follows:

```
\begin{tabular}{l|ll}
1 & R &Assumption 1 \\
2 & S & . \\
. & . &.\\
. & . &.\\
. & . &.\\
n & Conclusion & Justification \\
\end{tabular}
```

what we get is a line that descends between the first column and the second column:

1	R	Assumption 1
2	S	.
.	.	.
.	.	.
.	.	.
n	Conclusion	Justification

To create a horizontal line, we add the following command where we want the horizontal line to appear

```
\hline command
```

For example, if we want to add a horizontal line after the second row and above the last line, we would write:

```
\begin{tabular}{lll}
1 & R &Premise 1 \\
2 & S &Premise 2 \\
\hline
. & . &.\\
. & . &.\\
. & . &.\\
\hline
n & Conclusion & Justification \\
\end{tabular}
```

This would give use the following table:

1	R	Premise 1
2	S	Premise 2
.	.	.
.	.	.
.	.	.
n	Conclusion	Justification

4.2 How To Create a Truth Table

Suppose we have the formula $P \wedge R$ and we want to create a truth table for this formula. To do this we begin by creating a table with 5 columns (one for each propositional letter and one for each operator:

`\tablular{cc|cccc} environment`

Next, we will want to create 5 rows with $P \wedge R$ at the top separated by a horizontal line:

```
\begin{tabular}{cc|ccc}
P & R & P & $\wedge $ & R \\
\hline
 &  &  &  & \\
 &  &  &  & \\
 &  &  &  & \\
 &  &  &  &
\end{tabular}
```

Next, we insert the T's and F's between the ampersands:

P	R	P	$\wedge$	R
T	T	T	T	T
T	F	T	F	F
F	T	F	F	T
F	F	F	F	F

And, of course, if we needed more columns, we would add more letters to the

`\begin{tabular}{*****}`

command. While if we needed more rows, we would add them as follows:

```
row 1 & * & * & * & *\\
row 2 & * & * & * & *\\
.     & * & * & * & * \\
.     & * & * & * & * \\
.     & * & * & * & * \\
row n & * & * & * & *
```

4.3 How To Create a Truth Tree

Creating truth trees is the trickiest part of using LaTeX for logic. Before we begin, it will be necessary to add the following packages to your *.tex document:

```
\usepackage{tikz}
\usepackage{tikz-qtree}
```

In the table mode, we begin by creating a table with however many columns we might think we need. It is not necessary we know ahead of time, so it can be helpful to allot for more than we think we will need. Let's consider the following set of propositions:

$$A \wedge B, C, A \vee \neg C$$

As both of these will use certain notation for creating the tree, let's look at that notation:

1. The tree begins with the following commands:

   ```
   \Tree
   [.ROOT ]
   ```

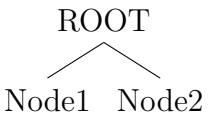
2. The root of the tree or a root of the subtree always begins with a period "." Thus, the above tree produces:

   ROOT

3. Each subtree in the tree is indicated by backets "[ ]". It is important that there is a space to the left of every closing bracket). For example, the following commands:

   ```
   \Tree
   [.ROOT [.Node1 ] [.Node2 ] ]
   ```

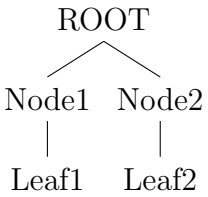
   produce the tree below:

4. Items descending from the node (sometimes called "leaf nodes") are expressed by their labels. These are expressed by typing a name after a node within the brackets of a node. The following commands:

```
\Tree
[.ROOT [.Node1 Leaf1 ] [.Node2 Leaf2 ] ]
```

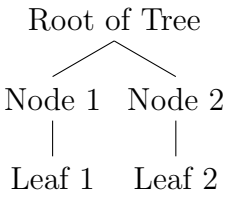
produce the tree below:



5. In order to have a label that contains space, e.g. "Node 1" rather than "Node1", curly braces are used. The following commands:

```
\Tree
[.{Root of Tree} [.{Node 1} {Leaf 1} ] [.{Node 2} {Leaf 2} ] ]
```

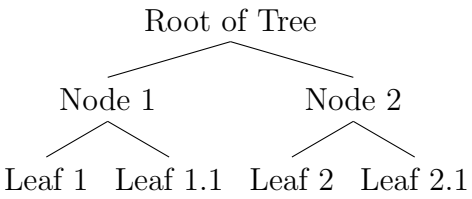
produces the tree below:



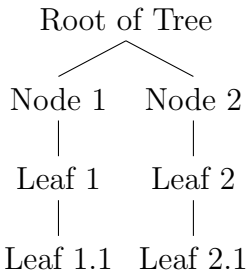
6. Adding space and a label to the right of a node will make that leaf node descend from the parent node. Subsequent spaces and labels will just create additional leafs under the parent node; it won't create leafs under leafs. Thus, if we write:

```
\Tree
[.{Root of Tree} [.{Node 1} {Leaf 1} {Leaf 1.1} ]
[.{Node 2} {Leaf 2}  {Leaf 2.1} ] ]
```

We won't have a tree where Node 1 descends from the Root, Leaf 1 descends from Node 1, and Leaf 1.1 descends from Leaf 1. Rather, we will have a tree where both Leaf 1 and Leaf 1.1 will descend from Node 1.



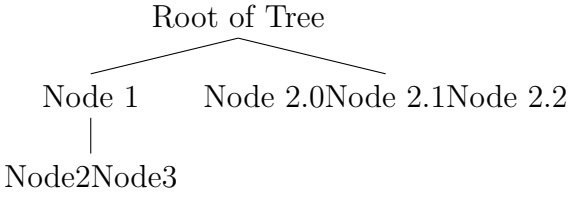
7. In order to indicate descendence, you must use the [.] structure. Thus, if we wanted to create the following tree:



we need to make Leaf 1 a node by prefixing [. before it and a closing bracket after Leaf 1.1. That is, we must use the following commands:

```
[.{Root of Tree}
[.{Node 1} [.{Leaf 1} {Leaf 1.1} ] ]
[.{Node 2} [.{Leaf 2}  {Leaf 2.1} ] ]
]
```

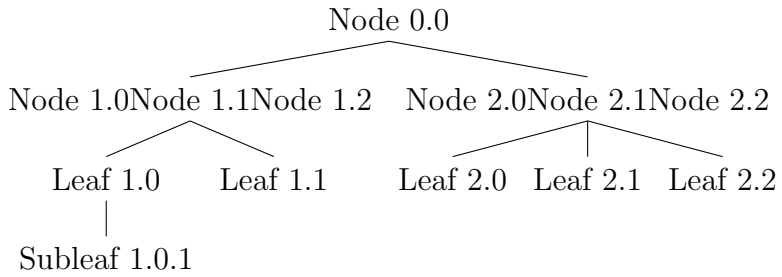
8. Finally, we can stack labels or formulas in a node by using the double backslash. That is,



The above is produced by using the following commands:

```
[.{Root of Tree}
[.{Node 1.0 \\ Node 1.1 \\ Node 1.2 } ]
[.{Node 2.0 \\ Node 2.1 \\ Node 2.2 } ]
]
```

We can use the principles above to create increasingly complex trees:



Using LaTeX for truth trees can be somewhat challenging as it is difficult to see how the linear commands of LaTeX correspond to the tree diagram. First, it is helpful to begin every tree by giving the root and its closing bracket their own lines:

```
\Tree
[.{Node 0.0}
]
```

Next, remember the following:

SPACES are for descendence

SLASHES are for stacking

BRACKETS are for grouping.

It can be helpful to configure your code by using putting every open bracket on a separate line. In other words, rather than have your code look like this:

```
\Tree
[.{Node 0.0} [.{Node 1.0 \\ Node 1.1 \\ Node 1.2 } [.{Leaf 1.0}
{Subleaf 1.0.1} ] {Leaf 1.1} ] [.{Node 2.0 \\ Node 2.1 \\
Node 2.2 } {Leaf 2.0} {Leaf 2.1} {Leaf 2.2} ]
```

it is more perspicuous if it looks like this:

```
\Tree
[.{Node 0.0}
[.{Node 1.0 \\ Node 1.1 \\ Node 1.2 }
[.{Leaf 1.0} {Subleaf 1.0.1} ] {Leaf 1.1} ]
[.{Node 2.0 \\ Node 2.1 \\ Node 2.2 } {Leaf 2.0} {Leaf 2.1}
{Leaf 2.2} ]
]
};
```

The main difficulty now becomes how to number the rows of the tree on the lefthand side and justify the lines of the tree on the righthand side. What we will do is create three different trees, each in its own column. [1]The columns for numbering

```
\begin{tikzpicture}[sibling distance=.5cm]
\begin{scope}
   Here is where the commands for the first column will go
\end{scope}

\Tree  [.{Here is where the main tree will go}

\begin{scope}
   Here is where the commands for the third column will go
\end{scope}
\end{tikzpicture}
```

The left and right columns will be positioned in relation to the main tree. First, let's focus on the left column.

```
\begin{scope}[xshift=-1.5in] %Column #1 For Numbering
\tikzset{edge from parent/.style={edge from parent path={(\tikzparentnode)
-- (\tikzchildnode)}},every tree node/.style={text width=5em,align=left}}
\Tree [.1
         [.2
        [.3 ]]]
\end{scope}
```

[1]I had some difficulty figuring out how to line trees up with columns; as such, I drew the following from `http://tex.stackexchange.com/questions/48758/how-to-draw-custom-nodes-attached-to-tikz-qtree`.

This will give us a tree with three numbered rows:

1

2

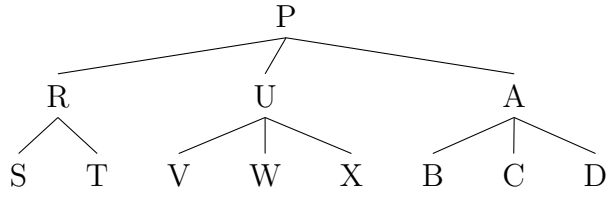
3

We can adjust the position of these by changing the "xshift" value from "-1.5" to some other number and we can add additional numbers by adding more open and closed brackets with numbers, i.e. [.n]. Next, let's add the following tree to our numbered column:

```
\Tree
   [.P
       [.R
        [.S ] [.T ]  ]
       [.U
        [.V ] [.W ] [.X ] ]
       [.A
        [.B ] [.C ] [.D ] ]
   ]
```

Adding the above command will produce:

1 2 3

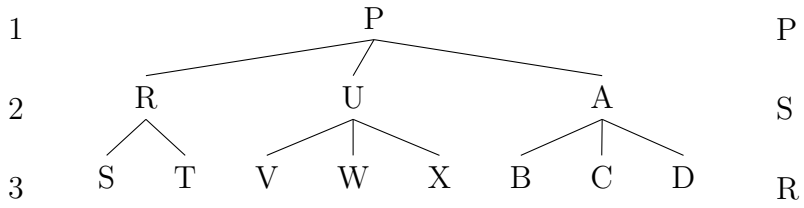


Finally, let's add the justification column. To do this, we will use the same commands we used to create the numbered columns except that we will need to adjust the "xshift" from "-1.5in" to "2.5in". This will ensure that the justification column is pushed to the right of the main tree. Here is what we will add to the existing commands:

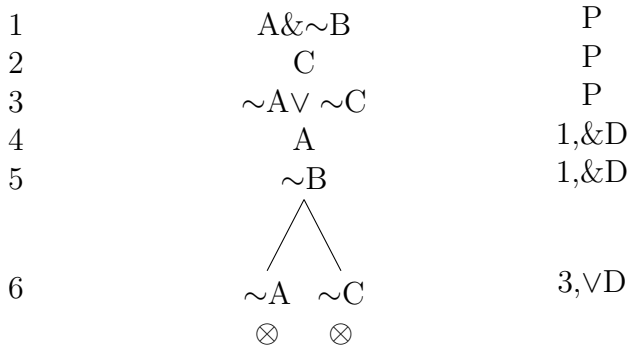
```
\begin{scope}[xshift=2.5in] %Column #3 For Justification
\tikzset{edge from parent/.style={edge from parent path={(\tikzparentnode)
-- (\tikzchildnode)}},every tree node/.style={text width=5em,align=left}}
\Tree [.{P}
```

```
        [.{S}
      [.{R} ]]]
\end{scope}
```

Putting everything together, we get the following tree:



The major difficulty we now face is how to STACK propositions on top of each other without the use of a line connect parent to sibiling. The trick will be to (i) make sure the anchor is south, (ii) toggle the level distance to coordinate



4.4 How To Create a Proof

In order to create a proof, we will need to install a the fitch.sty package created by Johan W. Kluwer (`http://folk.uio.no/johanw/FitchSty.html`). Here is how to do this:

1. 1. In your C: drive, create a directory as follows:

   ```
   C:\\Local TeX Files\tex\latex\misc
   ```

2. click on (`http://folk.uio.no/johanw/FitchSty.html`) and save the fitch.sty file to that directory.

3. Now, you will need to make the *.sty file known to your LaTeK system. In the case of MiKTeX, click "Start/ Programs/ MiKTeX 2.x/" then access "Maintenance" as the Administrator, then click "Settings" to access the MiKTeX options, then click Root, add the *.sty file, and finally click "General", and click "Refresh FNDB".

   For further assistance with this, see `http://tex.stackexchange.com/questions/2063/how-can-i-manually-install-a-package-on-miktex-windows`

4. Once MiKTeK recognizes the fitch.sty, you will need to write

   ```
   \usepackage{fitch1}
   ```

   under the document class at the top of your *.tex file.

Let's see how this pacakge works.
First, in order to begin a proof you will need to write the following commands:

```
\begin{equation*}
  \begin{fitch}
[proof will go here]
  \end{fitch}
\end{equation*}
```

The proof will automatically number lines and so the focus will be on (i) providing the wff in the main line of the proof and (ii) providing its justification

1	P	P
2	Q	P
3	S	P

To make an assumption and enter a subproof, use the following command:

```
\fa
```

To make another assumption in order to enter a nested subproof, use the above command iteratively. So,

```
\begin{equation*}
  \begin{fitch}
\fa P & A \\
\fa\fa Q & A \\
\fa\fa P\land Q & 1,2$\land$ I \\
  \end{fitch}
\end{equation*}
```

Would give us:

$$
\begin{array}{l|l|ll}
1 & P & & \text{A} \\
2 & \quad Q & & \text{A} \\
3 & \quad P \land Q & & 1{,}2 \land \text{I}
\end{array}
$$

fitch.sty also allows for several other commands dealing with assumptions. For example, the following:

$$
\begin{array}{l|l}
1 & \forall y \neg P(y) \\
\hline
2 & \quad \exists x P(x) \\
\hline
3 & \quad \overset{u}{|} \; Pa \\
\hline
4 & \quad\quad \neg Pa \\
5 & \quad\quad \bot \\
6 & \quad \bot \\
7 & \neg \exists x P(x)
\end{array}
$$

was typset using the following code:

```
\begin{equation*}
  \begin{fitch}
    \fh \forall y\lnot P(y)   \\
    \fa\fh \exists xP(x)  \\
    \fa\fa\fitchmodalh{u}  Pa \\
   \fa\fa\fa \lnot Pa  \\
    \fa\fa\fa \bot \\
    \fa\fa\bot  \\
```

```
    \fa  \lnot\exists xP(x)
  \end{fitch}
\end{equation*}
```

5 LaTeX Resources

1. A Great LaTeX Blog: `<http://texblog.net/>`
2. LaTeX templates: `http://www.latextemplates.com/`
3. LaTeX package 1 for Fitch-style proofs: `http://folk.uio.no/johanw/FitchSty.html`
4. LaTeX package 2 for Fitch-style proofs: `http://www.mathstat.dal.ca/~selinger/fitch/`
5. LaTeX for Linguistics `http://www.essex.ac.uk/linguistics/external/clmt/latex4ling/`
6. On Nodes: `http://en.wikibooks.org/wiki/LaTeX/PGF/TikZ#Nodes`
7. More on Nodes: `http://stuff.mit.edu/afs/athena/contrib/tex-contrib/beamer/pgf-1.01/doc/generic/pgf/version-for-tex4ht/en/pgfmanualse12.html`
8. Tikz-Qtree: `http://ctan.mackichan.com/graphics/pgf/contrib/tikz-qtree/tikz-qtree-manual.pdf`
9. LaTeX for Logicians: `http://www.logicmatters.net/latex-for-logicians/`