Proofs in LaTeX

Alexander W. Kocurek

June 8, 2019 (version 3)

What follows is a brief guide to writing proofs, in a variety of proof systems, using LaTeX. Proof systems covered include:

- Fitch proofs (§ 1)
- Sequent calculi and natural deduction trees (§ 2)
- Lemmon proofs (§ 3)
- Truth trees (§ 4)

To typeset in some of these systems, you may need to install some `.sty` files that do not come preinstalled in your TeX distribution. There are two ways to do this: locally and globally. The local way is to place a copy of the `.sty` file in the same folder as your main `.tex` file. This is fine as far as it goes, but it may be tedious to include separate copies of this file each time you start a new `.tex` document. The global way is to place just one copy of the `.sty` file in a globally accessible location, so that any `.tex` file you create has the ability to load that package automatically. The folder where you want to place the `.sty` file will vary based on your operating system.

Windows: `C:\Users\`⟨*user name*⟩`\texmf\tex\latex\local\`

Mac: `/Users/`⟨*user name*⟩`/Library/texmf/tex/latex/local/`

Linux: `~/texmf/tex/latex/local/`

Contents

1 Fitch Proofs

There are three main packages for Fitch proofs: `fitch`, `fitch`, and `lplfitch`. Yes, there are two `fitch` packages, one by Johan Klüwer another by Peter Selinger.

1.1 `fitch` (by Johan Klüwer)

I've placed a copy of Klüwer's `fitch.sty` here. Note I've slightly edited this copy to not load `mdwtab`, which redefines `tabular`; you can find Klüwer's original version here.

1.1.1 Basics

Fitch proofs are typed within the `fitch` environment:

```
\begin{fitch}
...
\end{fitch}
```

By default, the `fitch` will automatically number each line. To remove line numbers, use `fitch*` instead:

```
\begin{fitch*}
...
\end{fitch*}
```

There are three essential commands for typing lines in a proof:

- `\fa`: derived line
- `\fh`: hypothesis line
- `\fj`: hypothesis line (only for the last main premise)

You can increase the nesting depth by iterating these commands. So, `\fa` makes a derived line in the main proof, `\fa \fa` makes a derived line in a subproof, `\fa \fa \fa` makes a derived line in a subsubproof, etc. Each line should end with `\\` like they do in tables (the `fitch` is essentially just a table). Everything is automatically in math mode.

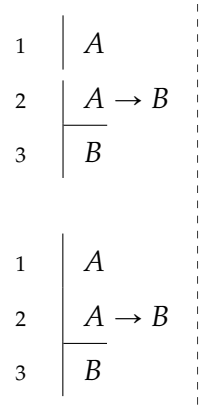
Example 1.1: Basic Fitch Proof

1 | A
2 | | B
3 | | A
4 | $B \rightarrow A$
5 | $A \rightarrow (B \rightarrow A)$

```
\begin{fitch}
  \fj  A \\
  \fa \fh B \\
  \fa \fa A \\
  \fa  B \rightarrow A \\
A \rightarrow (B \rightarrow A) \\
\end{fitch}
```

For spacing purposes, `\fj` is used on the last main premise line. Otherwise, use `\fh` for the hypothesis lines.

Example 1.2: Failing to Use `\fj`



```
\begin{fitch}
\fa A \\
\fh A \rightarrow B \\ % Ewww
\fa B
\end{fitch}
```

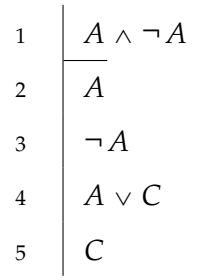


```
\begin{fitch}
\fa A \\
\fj A \rightarrow B \\ % Nice!
\fa B
\end{fitch}
```

1.1.2 Rules

You can cite rules in a separate, nicely aligned column with & like in a table. Everything to the right of & is in text mode.

Example 1.3: Rules



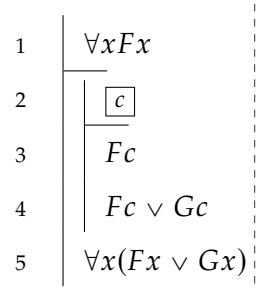
```
\begin{fitch}
\fj A \wedge \neg A  &  \\
\fa A                &  $\wedge$-Elim (1) \\
\fa \neg A           &  $\wedge$-Elim (1) \\
\fa A \vee C         &  $\vee$-Intro (2) \\
\fa C                &  $\vee$-Elim (3, 4) \\
\end{fitch}
```

1.1.3 Variable and Modal Subproofs

You can use `\fw` to create a line introducing a boxed item as a witnessing variable (for quantifier rules):

`\fw{`⟨*boxed item*⟩`}`

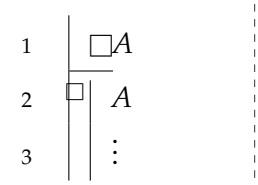
Example 1.4: Using \fw



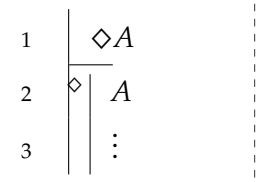

```
\begin{fitch}
\fj \forall x Fx \\
\fa \fh \fw{c} \\
\fa \fa Fc \\
\fa \fa Fc \vee Gc \\
\fa \forall x (Fx \vee Gx) \\
\end{fitch}
```

One can also use \fn for box modal proofs and \fp for diamond modal proofs:

Example 1.5: Modal Proofs with \fn and \fn



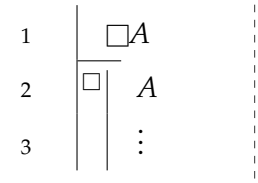

```
\begin{fitch}
\fj \Box A \\
\fa \fn A \\
\fa \fa \vdots \\
\end{fitch}
```




```
\begin{fitch}
\fj \Diamond A \\
\fa \fp A \\
\fa \fa \vdots \\
\end{fitch}
```

Notice the spacing here is not very good. You can adjust the spacing by redefining the length \fitchindent (the default is 0.7em). Note, however, that by increasing this length, you will also push all formulas away from the vertical lines.

Example 1.6: Modal Proofs with \fn and \fn

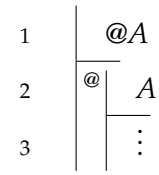



```
\setlength{\fitchindent}{1em}
\begin{fitch}
\fj \Box A \\
\fa \fn A \\
\fa \fa \vdots \\
\end{fitch}
```

If you want to insert your own operator, you can use:

\fitchmodal{⟨*operator*⟩}
\fitchmodalh{⟨*operator*⟩} (for hypotheses)

Example 1.7: Using \fitchmodalh

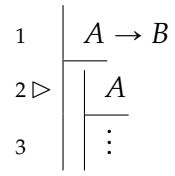



```
\setlength{\fitchindent}{1em}
\begin{fitch}
\fj @A \\
\fa \fitchmodalh{@} A \\
\fa \fa \vdots \\
\end{fitch}
```

1.1.4 Markers

You can put a marker in the proof to draw attention to a particular line using \fr (or \fs, if you don't want to include a vertical line).

Example 1.8: Using \fr




```
\begin{fitch}
\fj A \rightarrow B \\
\fr \fh A \\
\fa \fa \vdots \\
\end{fitch}
```

You can include multiple markers if you wish. To keep the marker from overlapping the vertical line, \fr should come before the other line commands.

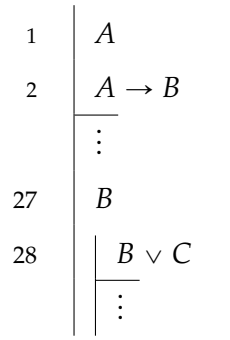
1.1.5 Custom Tags/Skipping Numbers

If you want to modify what appears to the left of the outermost vertical line (e.g., to change/remove line numbers), you can do this manually using:

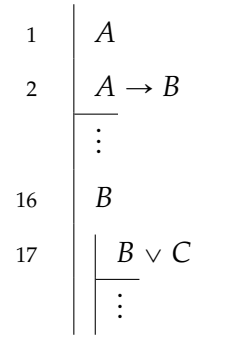
\ftag{⟨*lefthand side*⟩}{⟨*righthand side*⟩}

To skip a line number, the ⟨*lefthand side*⟩ argument should be ~. You can also just reset fitchcounter to manually adjust the line numbers.

Example 1.9: Changing/Removing Line Numbers




```
\begin{fitch}
\fa A \\
\fj A \rightarrow B \\
\ftag{~}{\fa \vdots} \\
\ftag{\scriptsize 27}{\fa B} \\
\ftag{\scriptsize 28}{\fa \fh B \vee C} \\
\ftag{~}{\fa \fa \vdots} \\
\end{fitch}
```

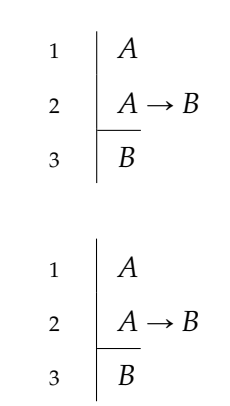



```
\begin{fitch}
\fa A \\
\fj A \rightarrow B \\
\ftag{~}{\fa \vdots} \setcounter{fitchcounter}{15} \\
\fa B \\
\fa\fh B \vee C \\
\ftag{~}{\fa \fa \vdots} \\
\end{fitch}
```

1.1.6 Shorter Top Line

If you think the top of the main vertical line extends too far, you can use `\fb` instead of `\fa` for the first premise. (It's hard for me to tell the difference.)

Example 1.10: Comparing Vertical Line Length with `\fb`




```
\begin{fitch}
\fb A \\ % Right length
\fj A \then B \\
\fa B
\end{fitch}
```



```
\begin{fitch}
\fa A \\ % A bit too long
\fj A \then B \\
\fa B
\end{fitch}
```

1.1.7 An Illustrative Example

Example 1.11: Illustrating `fitch`

1	$\forall x(Fx \to Gx)$	
2	$\exists x Fx$	
3	$\boxed{c} : Fc$	
4	$Fc \to Gc$	$\forall$-Elim (1)
5	Gc	$\to$-Elim (3, 4)
6	$\exists x Gx$	$\exists$-Intro (5)
7	$\exists x Gx$	$\exists$-Elim (2, 3–6)
8	$\exists x Fx \to \exists x Gx$	$\to$-Intro (2–7)
	$\triangleright\ \forall x(Fx \to Gx) \to (\exists x Fx \to \exists x Gx)$	$\to$-Intro (1–8)

```
\begin{fitch}
\fj \forall x (Fx \rightarrow Gx)  &  \\ %1
\fa \fh \exists x Fx               &  \\ %2
\fa \fa \fh \fw{c}: Fc             &  \\ %3
\fa \fa \fa Fc \rightarrow Gc      &  $\forall$-Elim (1) \\ %4
\fa \fa \fa Gc                     &  $\rightarrow$-Elim (3, 4) \\ %5
\fa \fa \fa \exists x Gx           &  $\exists$-Intro (5) \\ %6
\fa \fa \exists x Gx               &  $\exists$-Elim (2, 3--6) \\ %7
\fa \exists x Fx \rightarrow \exists x Gx  &  $\rightarrow$
-Intro (2--7) \\ %8
\ftag{~}{\fs \forall x (Fx \rightarrow Gx) \rightarrow (\exists x Fx
\rightarrow \exists x Gx)}  &  $\rightarrow$-Intro (1--8) \\ %End
\end{fitch}
```

1.2 `fitch` (by Peter Selinger)

You can find Selinger's version of `fitch.sty` here. I've also placed a copy of the package here.

1.2.1 Basics

Proofs are constructed in the `nd` environment, which must be used in math mode.

```
\begin{nd}
...
\end{nd}
```

There are four basic commands for typing lines in a fitch proof:

- `\hypo{`⟨*line label*⟩`}{`⟨*formula*⟩`}`: line with horizontal bar
- `\have{`⟨*line label*⟩`}{`⟨*formula*⟩`}`: line without horizontal bar
- `\open`: opens a subproof
- `\close`: closes a subproof

Example 1.12: Basic Fitch Proof

1 | | A	`\begin{align*}`
2 | | | B	`\begin{nd}`
3 | | | A	`  \open`
4 | | $B \to A$	`    \hypo{prem}{A}`
5 | $A \to (B \to A)$	`    \open`
	`      \hypo{hyp}{B}`
	`      \have{reit}{A}`
	`    \close`
	`  \have{cond}{B \rightarrow A}`
	`  \close`
	`  \have{conc}{A \rightarrow (B \rightarrow A)}`
	`\end{nd}`
	`\end{align*}`

Note that the first argument of `\hypo` and `\have` are *labels*. This is so you can refer back to them when citing rules.

1.2.2 Rules

Rules are placed after `\hypo` and `\have` commands. There is a generic rule command `\by`:

`\by{`⟨*rule*⟩`}{`⟨*line labels*⟩`}`

The ⟨*line labels*⟩ argument takes a list of labels, *not* numbers (though you could use numbers as your labels).

There are also a bunch of pre-defined rules, taking a single argument for the line labels. See Table 1.

Command	Output
\r	reiteration
\ai	$\wedge$ introduction
\ae	$\wedge$ elimination
\oi	$\vee$ introduction
\oe	$\vee$ elimination
\ii	$\rightarrow$ introduction
\ie	$\rightarrow$ elimination

Command	Output
\ni	$\neg$ introduction
\ne	$\neg$ elimination
\be	$\bot$ elimination
\nne	double negation elimination
\Ai	$\forall$ introduction
\Ae	$\forall$ elimination
\Ei	$\exists$ introduction
\Ee	$\exists$ elimination

Table 1: Rule commands in `fitch`.

Example 1.13: Rules

1	$A \wedge \neg A$	
2	A	$\wedge$E, 1
3	$\neg A$	$\wedge$E, 1
4	$A \vee C$	$\vee$I, 2
5	C	DS, 3, 4

```
\begin{align*}
\begin{nd}
  \hypo{contra}{A \wedge \neg A}
  \have{A}{A}          \ae{contra}
  \have{nA}{\neg A}    \ae{contra}
  \have{AvC}{A \vee C} \oi{A}
  \have{conc}{C}       \by{DS}{nA,AvC}
\end{nd}
\end{align*}
```

Notice the use of labels for the first argument in \hypo and have. You can refer to these labels outside of the nd environment.

1.2.3 Changing Line Numbers

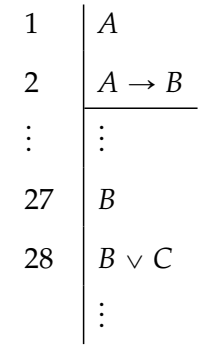
To change the line numbers, you need to use optional arguments for \hypo and \have.

- \hypo[⟨*symbol*⟩][⟨*offset*⟩]{⟨*line label*⟩}{⟨*formula*⟩}
- \have[⟨*symbol*⟩][⟨*offset*⟩]{⟨*line label*⟩}{⟨*formula*⟩}

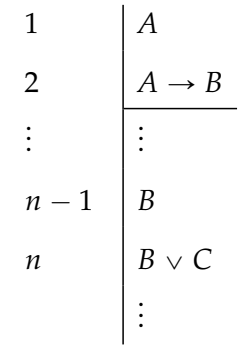
The symbol is just if you want the line number to be something like "n". Each line will automatically increment by one, so the next lines would be $n+1$, $n+2$, etc. The offset can be used to start at a different place (e.g., if you want to start at $n-1$, the offset argument would just be -1.

If you just want the line number to be something specific (e.g., 14), leave the symbol argument blank and just enter the number in the offset. To remove a line number entirely, you need to use ~ for the symbol argument.

Example 1.14: Changing/Removing Line Numbers



```
\begin{align*}
\begin{nd}
  \have{a}{A}
  \hypo{atob}{A \rightarrow B}
  \have[\vdots]{skip1}{\vdots}
  \have[][27]{b}{B}
  \have{bvc}{B \vee C}
  \have[~]{cont}{\vdots}
\end{nd}
\end{align*}
```

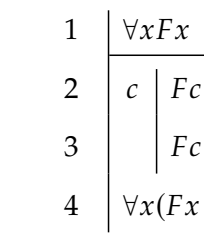


```
\begin{align*}
\begin{nd}
  \have{a}{A}
  \hypo{atob}{A \rightarrow B}
  \have[\vdots]{skip1}{\vdots}
  \have[n][-1]{b}{B}
  \have{bvc}{B \vee C}
  \have[~]{cont}{\vdots}
\end{nd}
\end{align*}
```

1.2.4 Variable and Modal Subproofs

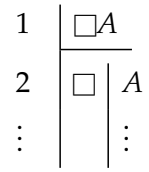
You can add a variable or operator next to a vertical line opening a subproof using an optional argument for `open`.

Example 1.15: Variable Proofs



```
\begin{align*}
\begin{nd}
  \hypo{1}{\forall x Fx}
  \open[c]
    \have{2}{Fc}
    \have{3}{Fc \vee Gc}
  \close
  \have{4}{\forall x (Fx \vee Gx)}
\end{nd}
\end{align*}
```

Example 1.16: Modal Proofs




```
\begin{align*}
\begin{nd}
  \hypo{1}{\Box A}
  \open[\Box]
    \have{2}{A}
    \have[\vodts]{3}{\vdots}
  \close
\end{nd}
\end{align*}
```

1.2.5 An Illustrative Example

Example 1.17: Illustrating `fitch`

1	$\forall x(Fx \to Gx)$	
2	$\exists xFx$	
3	c | Fc	
4	$Fc \to Gc$	$\forall$E, 1
5	Gc	$\Rightarrow$E, 3, 4
6	$\exists xGx$	$\exists$I, 5
7	$\exists xGx$	$\exists$I, 2, 3–6
8	$\exists xFx \to \exists xGx$	$\Rightarrow$I, 2–7

```
\begin{align*}
\begin{nd}
  \hypo{1}{\forall x (Fx \rightarrow Gx)}
  \open
    \hypo{2}{\exists x Fx}
    \open[c]
      \hypo{3}{Fc}
      \have{4}{Fc \rightarrow Gc}   \Ae{1}
      \have{5}{Gc}                  \ie{3,4}
      \have{6}{\exists x Gx}        \Ei{5}
    \close
    \have{7}{\exists x Gx}          \Ei{2,3-6}
  \close
```

```
  \have{8}{\exists x Fx \rightarrow \exists x Gx}   \ii{2-7}
\end{nd}
\end{align*}
```

1.3 lplfitch

The `lplfitch` package is the Fitch proof package used in *Language, Proof, and Logic*. Installing `lplfitch.sty` is a bit roundabout. So to make your life easy, I've just placed the most up-to-date version as of January 2017 here. To download the most recent version, if it has changed, you need to first go to its CTAN page and download the `.zip` archive. You then need to run `lplfitch.ins` (just typeset as normal), and it will generate the `lplfitch.sty` file.

1.3.1 Basics

Instead of placing your fitch proof in an environment, you simply place it inside the command:

`\fitchprf{`⟨*hypotheses*⟩`}{`⟨*deduction*⟩`}`

The first argument is a list of your hypotheses, while the second argument is your deduction.

To add a line in a deduction, use the command:

`\pline[`⟨*left of formula*⟩`]{`⟨*formula*⟩`}[`⟨*rule*⟩`]`

You can separate lines using `\\`. (In the examples that follow, I'll be generous with spacing to make it easier to read. But you do not have to be so liberal.)

Example 1.18: Basic Fitch Proof

A
A → B

B

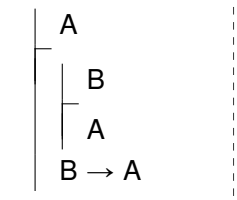
```
\fitchprf{
  \pline{A} \\
  \pline{A \rightarrow B} }
{ \pline{B} }
```

To start a subproof, all you need to do is use the command:

`\subproof{`⟨*hypotheses*⟩`}{`⟨*deduction*⟩`}`

which works exactly like `\fitchprf`. You do not need to put `\\` at the end of `\subproof` to move on to the next line.

Example 1.19: Subproofs

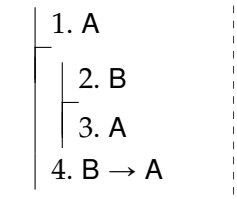



```
\fitchprf{
  \pline{A} }
{ \subproof{\pline{B}}
  { \pline{A} }
  \pline{B \rightarrow A} }
```

1.3.2 Line Numbers

Unlike `fitch.sty`, there's no good way to automatically number lines. You can do this manually via the optional argument for `\pline`.

Example 1.20: Numbered Lines

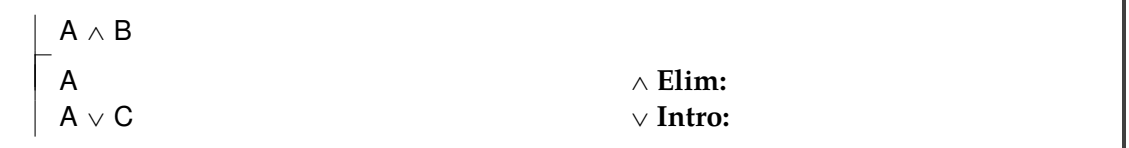



```
\fitchprf{
  \pline[1.]{A} }
{ \subproof{\pline[2.]{B}}
  { \pline[3.]{A} }
  \pline[4.]{B \rightarrow A} }
```

1.3.3 Rules

The `\pline` command's second optional argument allows you to cite rules in your proofs.

Example 1.21: Rules




```
\fitchprf{
  \pline{A \wedge B} }
{ \pline{A}[$\wedge$-Elim] \\
  \pline{A \vee C}[$\vee$-Intro] }
```

As you can see, for some reason, the formulas are placed ridiculously far away from their rule citations. The easiest fix for me has been to manually reset the width of fitch proofs by adding the following line of code before the proof:

```
\setlength{\fitchprfwidth}{⟨length⟩}
```

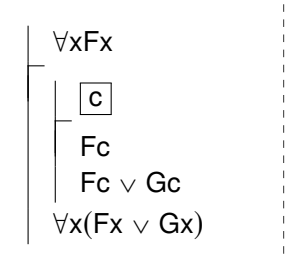
1.3.4 Boxes

For quantifier proofs, boxed subproofs may be added using the command:

```
\boxedsubproof[⟨left of boxed item⟩]{⟨boxed item⟩}{⟨hypotheses⟩}{⟨deduction⟩}
```

This works exactly like `\fitchprf` and `\subproof`. The optional argument allows you to add the line number to the left of the box, instead of to the left of the formula. (Note: don't use `\pline` in the ⟨*hypothesis*⟩ argument.)

Example 1.22: Boxed Subproof




```
\fitchprf{
  \pline{\forall x Fx}
}
{ \boxedsubproof{c}{}
  { \pline{Fc} \\
    \pline{Fc \vee Gc} }
  \pline{\forall x (Fx \vee Gx)} }
```

1.3.5 Symbols

`lplfitch` provides a small library of predefined commands for common logic symbols. They are listed in Table 2. The difference between `\lall` and `\uni` is in spacing. Compare:

`\lall x Fx`	⇒	$\forall x Fx$	`\lall x \lis y Rxy`	⇒	$\forall x \exists y Rxy$
`\uni{x} Fx`	⇒	$\forall x\, Fx$	`\uni{x} \exi{y} Rxy`	⇒	$\forall x\, \exists y\, Rxy$

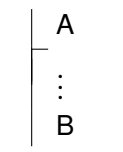
1.3.6 Formatting

`lplfitch` introduces a `\formula{`⟨*formula*⟩`}` command, so that you can indicate what is and is not a formula. By default, `\formula` puts its argument into sans serif. Everything inside `\pline` is inside the scope of `\formula`. You can change the formatting however you like by redefining the `\formula` command. For instance, to just have everything appear in normal math mode font, you can insert the following line of code in the preamble:

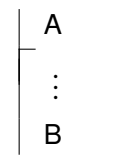
```
\renewcommand{\formula}[1]{\ensuremath{#1}}
```

If one introduces ellipses, one should use `\ellipsesline` in place of `\pline{\vdots}` to get the spacing right. Compare:

`\vdots`



`\ellipsisline`



Command	Output
\lnot	$\neg$
\land	$\wedge$
\lor	$\vee$
\lif	$\rightarrow$
\liff	$\leftrightarrow$
\lfalse	$\bot$
\lall	$\forall$
\lis	$\exists$
\uni{x}	$\forall x$
\exi{x}	$\exists x$

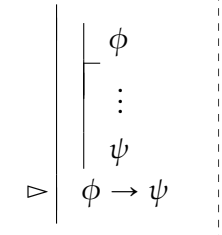
Command	Output
\lnoti{1}	$\neg$ **Intro:** 1
\lnote{2}	$\neg$ **Elim:** 2
\landi{3}	$\wedge$ **Intro:** 3
\lande{4}	$\wedge$ **Elim:** 4
\lori{5}	$\vee$ **Intro:** 5
\lore{6}{7}{8}	$\vee$ **Elim:** 6, 7, 8
\lifi{9}	$\rightarrow$ **Intro:** 9
\life{10}	$\rightarrow$ **Elim:** 10,
\liffi{11}	$\leftrightarrow$ **Intro:** 11,
\liffe{12}	$\leftrightarrow$ **Elim:** 12,
\lfalsei{13}	$\bot$ **Intro:** 13,
\lfalsee{14}	$\bot$ **Elim:** 14
\lalli{15}	$\forall$ **Intro:** 15
\lalle{16}	$\forall$ **Elim:** 16
\lexii{17}	$\exists$ **Intro:** 17
\lexie{18}{19}	$\exists$ **Elim:** 18, 19
\reit{20}	**Reit:** 20
\eqi	= **Intro**
\eqe{21}{22}	= **Elim:** 21, 22

Table 2: Symbol commands in `lplfitch`.

1.3.7 `fitchctx`

`lplfitch` provides another command for making Fitch proofs in place of `\fitchprf`: `\fitchctx`. This command, among other things, allows you to remove the first horizontal hypothesis line. Unlike `\fitchprf`, `\fitchctx` only takes one argument.

Example 1.23: Removing the First Hypothesis Line




```
\fitchctx{
  \subproof{ \pline{\phi} }
  { \ellipsesline \\
    \pline{\psi} }
    \fpline{\phi \rightarrow \psi} }
```

Notice the last line's use of `\fpline` instead of `\pline`. This places the marker on that line. *Note that* `\fpline` *is only defined in* `fitchctx`. You can redefine the marker used by `\fpline` by redefining the command `\slider`.

1.3.8 An Illustrative Example

Example 1.24: Illustrating `lplfitch`

1. ∀x (Fx → Gx)	
2. ∃x (Fx)	
3. c Fc	
4. Fc → Gc	∀ **Elim:** 1
5. Gc	→ **Elim:** 3, 4,
6. ∃x (Gx)	∃ **Intro:** 5
7. ∃x (Gx)	∃ **Elim:** 2, 3–6
8. ∃x (Fx) → ∃x (Gx)	→ **Intro:** 2–7

```
\setlength{\fitchprfwidth}{5cm}

\fitchprf{ \pline[1.]{\uni{x} (Fx \lif Gx)} }
{ \subproof{ \pline[2.]{\exi{x} (Fx)} }
  { \boxedsubproof[3.]{c}{ Fc }
    { \pline[4.]{Fc \lif Gc}[\lalle{1}] \\
      \pline[5.]{Gc}[\life{3, 4}] \\
      \pline[6.]{\exi{x} (Gx)}[\lexii{5}] }
    \pline[7.]{\exi{x} (Gx)}[\lexie{2}{3--6}] }
  \pline[8.]{\exi{x} (Fx) \lif \exi{x} (Gx)}[\lifi{2--7}] }
```

2 Proof Trees

The main way to type proof trees is with `bussproofs`. I've placed a copy of `bussproofs.sty` here. The official website with full documentation can be found here. You can find a more complete documentation of `bussproofs` with more formatting tricks here.

2.1 Basics

To make a proof tree, one starts with the `prooftree` environment.

```
\begin{prooftree}
...
\end{prooftree}
```

There are two ways to type proof trees in `bussproofs`: the *automatic* way and the *manual* way.

2.1.1 Automatic Alignment

Each node of a proof tree will be enclosed in one of the following commands (the `C` stands for "centered"):

- `\AxiomC{`⟨*node*⟩`}`: the beginning, or "leaf", of a tree
- `\UnaryInfC{`⟨*node*⟩`}`: a node that follows from 1 previous node
- `\BinaryInfC{`⟨*node*⟩`}`: a node that follows from 2 previous nodes
- `\TrinaryInfC{`⟨*node*⟩`}`: ... 3 previous nodes
- `\QuaternaryInfC{`⟨*node*⟩`}`: ... 4 previous nodes
- `\QuinaryInfC{`⟨*node*⟩`}`: ... 5 previous nodes

The ⟨*node*⟩ is in text mode. The ordering of the commands determines their location on the tree. Inference lines are placed directly below the most recently created nodes. So for instance, suppose my first three nodes are:

```
\AxiomC{$A_1$}
\AxiomC{$A_2$}
\AxiomC{$A_3$}
```

If my next line were `\UnaryInfC{$B$}`, B would be placed directly below A_3. If my next line were `\BinaryInfC{$B}$`, B would be placed directly below A_2 and A_3. And so on. If I want to draw an inference after just A_2, say, then I need to write `\UnaryInfC{$B$}` after `\AxiomC{$A_2$}`. This rule applies to all nodes, including those derived via inferences.

Example 2.1: Basic Proof Tree

$$\dfrac{\dfrac{A_1}{A_2}\qquad\dfrac{B_1\qquad B_2}{B_3}}{AB}$$

```
\begin{prooftree}
\AxiomC{$A_1$}
  \UnaryInfC{$A_2$}
\AxiomC{$B_1$}
\AxiomC{$B_2$}
  \BinaryInfC{$B_3$}
    \BinaryInfC{$AB$}
\end{prooftree}
```

2.1.2 Manual Alignment

Nodes on the automatic way are centered. But often, you may want to align nodes based to a symbol, e.g., $\vdash$ or $\Rightarrow$. The manual way can help.

The commands are the same as before, except without the `C` or the curly brackets `{...}`. Furthermore, your arguments must be in math mode, and must include the command `\fCenter` so that `bussproofs` knows where to align the node. Compare the two methods:

Example 2.2: Automatic vs. Manual

$$\frac{A \vdash B}{A, A', A'' \vdash B}$$

```
\begin{prooftree}
\AxiomC{$A \vdash B$}
  \UnaryInfC{$A, A', A'' \vdash B$}
\end{prooftree}
```

$$\frac{A \vdash B}{A, A', A'' \vdash B}$$

```
\begin{prooftree}
\Axiom$A \fCenter\vdash B$
  \UnaryInf$A, A', A'' \fCenter\vdash B$
\end{prooftree}
```

By default, `\fCenter` just produces a space. To make your life easy, you could define it as a turnstile or an arrow. So for instance, if we insert this into the preamble:

```
\renewcommand{\fCenter}{\vdash}
```

then `\fCenter` will output $\vdash$; so the code above could be replaced by:

Example 2.3: Redefined `\fCenter`

$$\frac{A \vdash B}{A, A', A'' \vdash B}$$

```
\begin{prooftree}
\Axiom$A \fCenter B$
  \UnaryInf$A, A', A'' \fCenter B$
\end{prooftree}
```

You can use both manual and automatic methods in the same proof tree. So there's no reason to feel like you must stick with one method or the other.

2.2 Rules and Lines

2.2.1 Rules

You can add rule labels with `\LeftLabel{`⟨*label*⟩`}` and `\RightLabel{`⟨*label*⟩`}`. These should be placed before the inference line is drawn.

Example 2.4: Right Label

$$\frac{A \vdash C}{A, B \vdash C}\ \text{Weakening}$$

```
\begin{prooftree}
\AxiomC{$A \vdash C$}
  \RightLabel{Weakening}
  \UnaryInfC{$A, B \vdash C$}
\end{prooftree}
```

2.2.2 Lines

You can also change the appearance of an inference line. On the one hand,

```
\noLine
\singleLine
\doubleLine
```

will draw zero, one, or two lines respectively for the next inference.

Example 2.5: Drawing No Lines

$$\dfrac{A \vee B \qquad \begin{array}{c}[A]\\ \vdots \\ C\end{array} \qquad \begin{array}{c}[B]\\ \vdots \\ C\end{array}}{C}$$

```
\begin{prooftree}
\AxiomC{$A \vel B$}
\AxiomC{$[A]$}
  \noLine
  \UnaryInfC{$\vdots$}
  \noLine
  \UnaryInfC{$C$}
\AxiomC{$[B]$}
  \noLine
  \UnaryInfC{$\vdots$}
  \noLine
  \UnaryInfC{$C$}
    \TrinaryInfC{$C$}
\end{prooftree}
```

Example 2.6: Drawing Double Lines

$$\dfrac{\dfrac{A, B \vdash C}{}}{A \vdash B \to C}$$

```
\begin{prooftree}
\AxiomC{$A, B \vdash C$}
  \doubleLine
  \UnaryInfC{$A \vdash B \rightarrow C$}
\end{prooftree}
```

On the other hand,

```
\solidLine
\dashedLine
\dottedLine
```

will draw solid, dashed, or dotted lines respectively for the next inference.

Example 2.7: Drawing Dashed Line

$$\dfrac{A \vdash C}{A, B \vdash C}$$

```
\begin{prooftree}
\AxiomC{$A \vdash C$}
  \doubleLine
  \dashedLine
  \UnaryInfC{$A, B \vdash C$}
\end{prooftree}
```

As the above example illustrates, you can mix and match these style options.

If you want all of your inference lines to be a certain style, the following global commands will change the default style accordingly:

```
\alwaysNoLine
\alwaysSingleLine
\alwaysDoubleLine
\alwaysSolidLine
\alwaysDashedLine
\alwaysDottedLine
```

2.3 Abbreviations

If your fingers get tired, you can utilize the abbreviated versions of these commands. Simply insert `\EnableBpAbbreviations` in your document. Then you can make use of the following abbreviated commands:

Abbrevation	**Abbreviates**	**Abbrevation**	**Replaces**
`\AX`	`\Axiom`	`\AXC`	`\AxiomC`
`\UI`	`\UnaryInf`	`\UIC`	`\UnaryInfC`
`\BI`	`\BinaryInf`	`\BIC`	`\BinaryInfC`
`\TI`	`\TernaryInf`	`\TIC`	`\TernaryInfC`

Table 3: Abbreviations provided by `bussproofs`.

I've also defined some abbreviated commands, summarized in Table 4. If you want to use them, add these lines to your preamble (note this requires the `xargs` and `xifthen` packages):

```
\newcommand{\ife}[4]{\ifthenelse{\equal{#1}{#2}}{#3}{#4}}
\newenvironment{tree}{\begin{prooftree}}{\end{prooftree}}
\newcommand{\ax}[1]{\AxiomC{\EMX{#1}}}
\renewcommandx{\inf}[2][1=1, usedefault]{
\ife{#1}{1}
  {\UnaryInfC{\EMX{#2}}}
  {\ife{#1}{2}
    {\BinaryInfC{\EMX{#2}}}
    {\ife{#1}{3}
      {\TrinaryInfC{\EMX{#2}}}
      {\ife{#1}{4}
        {\QuaternaryInfC{\EMX{#2}}}
        {\ife{#1}{5}
          {\QuinaryInfC{\EMX{#2}}}
          {}
        }
      }
    }
  }
}
\newcommand{\LL}[1]{\LeftLabel{#1}}
\newcommand{\RL}[1]{\RightLabel{#1}}
\newcommand{\binf}[1]{\inf[2]{#1}}
```

Abbreviation	**Abbreviates**
`tree`	`prooftree`
`\ax`	`\AxiomC`
`\inf`	`\UnaryInf`
`\binf`	`\BinaryInfC`
`\LL`	`\LeftLabel`
`\RL`	`\RightLabel`

Table 4: Abbrevations provided by me.

The arguments are in math mode by default (except for `\LL` and `\RL`). The optional argument of `\inf[`⟨*number*⟩`]{`⟨*label*⟩`}` determines the number of nodes above that inference line. The default option is 1. So the effects are summarized as follows:

```
\inf[1]  ⇒  \UnaryInfC
\inf[2]  ⇒  \BinaryInfC
\inf[3]  ⇒  \TernaryInfC
\inf[4]  ⇒  \QuaternaryInfC
\inf[5]  ⇒  \QuinaryInfC
\inf[?]  ⇒  nothing
```

2.4 An Illustrative Example

Example 2.8: Illustrating `bussproofs`

$$\dfrac{\dfrac{(A \to B) \wedge (A \to C) \vdash A \to B}{(A \to B) \wedge (A \to C), A \vdash B} \to\text{E} \quad \dfrac{(A \to B) \wedge (A \to C) \vdash A \to C}{(A \to B) \wedge (A \to C), A \vdash C} \to\text{E}}{\dfrac{(A \to B) \wedge (A \to C), A \vdash B \wedge C}{(A \to B) \wedge (A \to C) \vdash A \to (B \wedge C)} \to\text{I}} \wedge\text{I}$$

```
\renewcommand{\fCenter}{\vdash}

\begin{prooftree}

% Left Branch
\Axiom$(A \rightarrow B) \wedge (A \rightarrow C) \fCenter A \rightarrow B$
\LeftLabel{$\rightarrow$E}
  \UnaryInf$(A \rightarrow B) \wedge (A \rightarrow C), A \fCenter B$

% Right Branch
\AxiomC{$(A \rightarrow B) \wedge (A \rightarrow C) \vdash A \rightarrow C$}
\RightLabel{$\rightarrow$E}
\dashedLine
  \UnaryInfC{$(A \rightarrow B) \wedge (A \rightarrow C), A \vdash C$}

% Combining Branches
\RightLabel{$\wedge$I}
\BinaryInfC{$(A \rightarrow B) \wedge (A \rightarrow C), A \vdash B \wedge C$}
\RightLabel{$\rightarrow$I}
\doubleLine
\UnaryInfC{$(A \rightarrow B) \wedge (A \rightarrow C) \vdash A \rightarrow
(B \wedge C)$}

\end{prooftree}
```

3 Lemmon Proofs

The easiest way to type Lemmon proofs is with `ND`, though I typically end up drawing these proofs using TikZ (see § 4.2). You can find a copy of `ND` here. A good, quick guide can be found here.

Typing Lemmon proofs are relatively straightforward. Lemmon proofs are typed inside the `ND` environment:

```
\begin{ND}[⟨title⟩][⟨label⟩][⟨premise width⟩][⟨rule width⟩][⟨total width⟩]
...
\end{ND}
```

The easiest way to control spacing is, I think, to set the total width first, and make adjustments from there if necessary.

Lines in a proof are made using the `\ndl` command:

```
\ndl{⟨premises⟩}{⟨formula⟩}{⟨rule⟩}
```

One can label lines of a proof to automatically refer back to them when you cite rules. However, this feature seems to conflict with the parameters for `hyperref` package used in this guide. So unfortunately, I can't typeset a proof with labels here. Again, see this guide for details. Otherwise, using `ND` is pretty straightforward:

Example 3.1: Illustrating `ND`

Lemmon Style

1	(1)	A	Premise
2	(2)	$A \rightarrow B$	Premise
3	(3)	$B \rightarrow C$	Premise
1, 2	(4)	B	→-Elim (1, 2)
1, 2, 3	(5)	C	→-Elim (3, 4)

```
\begin{ND}[Lemmon Style][][][][0.7\linewidth]

\ndl{1}{$A$}{Premise}
\ndl{2}{$A \rightarrow B$}{Premise}
\ndl{3}{$B \rightarrow C$}{Premise}
\ndl{1,2}{$B$}{$\rightarrow$-Elim (1, 2)}
\ndl{1,2,3}{$C$}{$\rightarrow$-Elim (3, 4)}

\end{ND}
```

4 Truth Trees

I only know of two ways to type truth trees in LaTeX. The first is to use `qtree`, which is simple and easy to learn, but has very low functionality and is designed for constructing syntax trees rather than truth trees. The second way is to use TikZ, which is much more flexible and still fairly straightforward to use, but requires a bit more effort to learn.

4.1 `qtree`

I've placed a copy of the `qtree` package here. The official site is here.

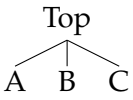
4.1.1 Basics

To type a tree, you simply use the command:

```
\Tree[.{⟨parent⟩} {⟨child1⟩} {⟨child2⟩} ... ]
```

Each child is separated by a space.

Example 4.1: Basic Tree

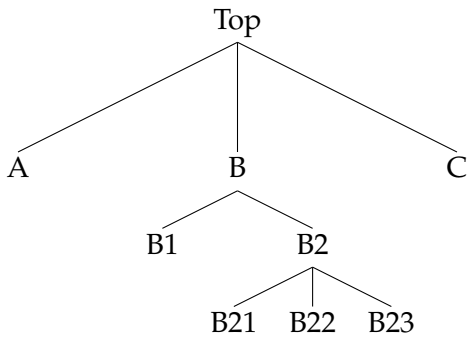



```
\Tree[.{Top} {A} {B} {C} ]
```

The general pattern for including more descendent nodes is to enclose the nodes in square brackets and to place a period in front of the parent node. You may include the period and parent node either after the left or right square bracket. Thus, either of these patterns is acceptable, and produces the same output:

```
[.{⟨parent⟩} {⟨child1⟩} {⟨child2⟩} ... ]
[ {⟨child1⟩} {⟨child2⟩} ... ].{⟨parent⟩}
```

Example 4.2: Further Branching



```
\Tree[.{Top} {A} [.{B} {B1} [.{B2} {B21} {B22} {B23} ] ] {C} ]
```

```
\Tree[ {A} [ {B1} [ {B21} {B22} {B23} ].{B2} ].{B} {C} ].{Top}
```

4.1.2 Alignment

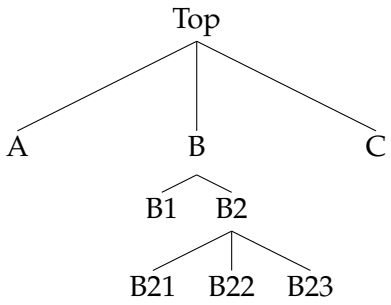
By default, trees are always centered on the page. You can change this globally by adding the option `nocenter` when you load the package. You can also toggle automatic centering locally via the following commands:

```
\qtreecentertrue
\qtreecenterfalse
```

4.1.3 Spacing

To adjust the inter-node distance, you can use the command `\qsetw{`⟨*width*⟩`}`. This adjusts the width of (sub)tree whose last node appears to the left of the command (do not forget the `!` before `\qsetw`).

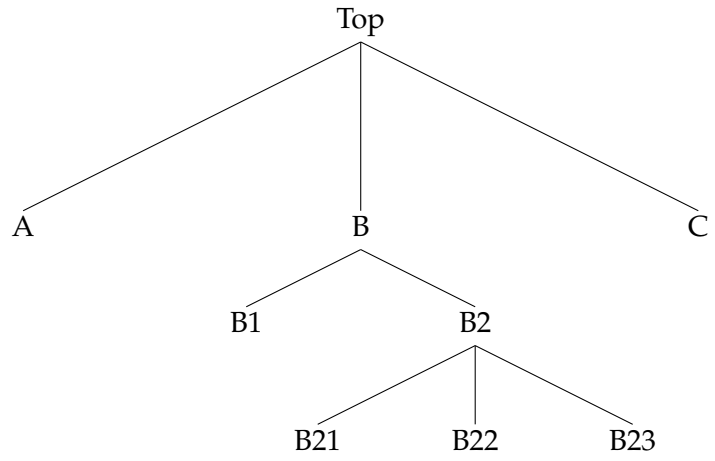
Example 4.3: Adjusting Node Distance




```
\Tree[.{Top} {A} [.{B} {B1} [.{B2} {B21} {B22} {B23} ] !\qsetw{1cm} ] {C} ]
```

Notice also that the subtrees gradually get smaller. To avoid this, you can use `\qbalance`.

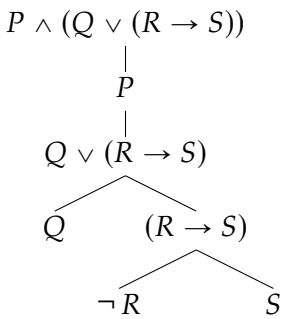
Example 4.4: Balancing Tree




```
\Tree[.{Top} {A} [.{B} {B1} [.{B2} {B21} {B22} {B23} !{\qbalance} ]  ] {C} ]
```

4.1.4 An Illustrative Example

Example 4.5: Illustrating `qtree`




```
\Tree[.{$P \wedge (Q \vee (R \rightarrow S))$} [.{$P$} [.{$Q \vee (R
\rightarrow S)$} {$Q$} [.{$(R \rightarrow S)$} {$\neg R$} {$S$} !{\qbalance} ]
!\qsetw{3cm} ] ] ]
```

4.2 TikZ

While TikZ can be a bit unweildly to learn in its full scope (see the 1161-page manual!), using it for simple things like truth trees is rather easy and flexible once things are set up. In what follows, we won't try to give a general tutorial for TikZ; we'll only set things up just enough to do truth trees. See my guide to TikZ here for more on how to use TikZ to create diagrams in LaTeX. For a more complete tutorial, I would recommend reading the

first few chapters of the manual, which is quite helpful in walking you through examples step-by-step.

4.2.1 Set-up

Include the following in your preamble:

```
\usepackage{tikz}
\usetikzlibrary{positioning}
```

While TikZ offers a wide variety of other libraries which may help customize all sorts of things, this suffices for truth trees.

4.2.2 Nodes

We'll show how to draw a complete proof tree one step at a time. The first thing to do is to figure out how to draw the nodes of the truth tree.

Start with the following:

```
\begin{tikzpicture}[node distance=1ex]
...
\end{tikzpicture}
```

The optional parameter isn't necessary, and you can adjust the distance however you like, though 1ex seems appropriate.

A line of code corresponding to a node will look generally like this:

```
\node (⟨name⟩) [⟨options⟩] {⟨formula⟩};
```

Don't forget the semicolon at the end! You'll get errors if you do (I make this mistake a lot. . .).

The ⟨*name*⟩ corresponds to a unique name for that node (no spaces). Usually, for ease of reference, I try to have it mimic the formula. For instance, if $A \to B$ is your formula, you could have its name be `A->B`, or `AtoB`.

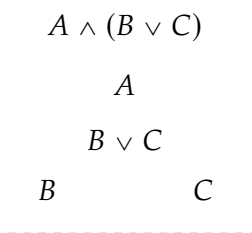
The ⟨*options*⟩ will help position the formula appropriately relative to the other formulas. The most common options for truth trees include:

```
below=of ⟨name⟩
below left=of ⟨name⟩
below right=of ⟨name⟩
```

The ⟨*formula*⟩ is where you can write your formula. You can also just put ordinary text there. Whatever you fill in for this argument will go in the center of the node.

In the examples below, I put spaces between these different arguments for readability. But they are not necessary.

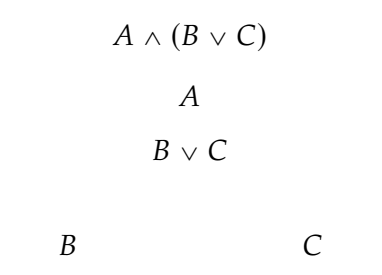
Example 4.6: Drawing Nodes




```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                      {$A \wedge (B \vee C)$};
  \node  (A)     [below=of A&BvC]     {$A$};
  \node  (BvC)   [below=of A]         {$B \vee C$};
  \node  (B)     [below left=of BvC]  {$B$};
  \node  (C)     [below right=of BvC] {$C$};
\end{tikzpicture}
```

If you want to adjust the spacing between nodes, you can put a length before `of` in the optional parameters.

Example 4.7: Adjusting Length




```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                          {$A \wedge (B \vee C)$};
  \node  (A)     [below=of A&BvC]         {$A$};
  \node  (BvC)   [below=of A]             {$B \vee C$};
  \node  (B)     [below left=1cm of BvC]  {$B$};
  \node  (C)     [below right=1cm of BvC] {$C$};
\end{tikzpicture}
```

If you want to manually shift the position of a node, you can do so via `xshift` and `yshift`.

Example 4.8: Adjusting Position

$$A \wedge (B \vee C)$$

$$A$$

$$B \vee C$$

$$B \qquad C$$

```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                                  {$A \wedge (B \vee C)$};
  \node  (A)    [below=of A&BvC]                     {$A$};
  \node  (BvC)  [below=of A]                         {$B \vee C$};
  \node  (B)    [below left=of BvC, xshift=5mm]      {$B$};
  \node  (C)    [below right=of BvC, xshift=-5mm]    {$C$};
\end{tikzpicture}
```

4.2.3 Paths

To draw lines connecting formulae, insert a line of code like the following:

```
\path  (⟨name of start node⟩)  edge[-]  (⟨name of end node⟩);
```

Example 4.9: Drawing Paths

$$A \wedge (B \vee C)$$

$$A$$

$$B \vee C$$

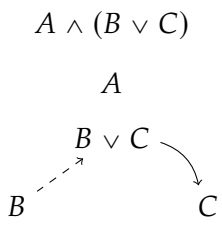
$$B \qquad\qquad C$$

```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                              {$A \wedge (B \vee C)$};
  \node  (A)     [below=of A&BvC]             {$A$};
  \node  (BvC)   [below=of A]                 {$B \vee C$};
  \node  (B)     [below left=5mm of BvC]      {$B$};
  \node  (C)     [below right=5mm of BvC]     {$C$};

  \path  (BvC)  edge[-]  (B);
  \path  (BvC)  edge[-]  (C);
\end{tikzpicture}
```

Next to the -, you can add optional arguments to change the shape or bend of the line.

For instance, you can make it `dashed` or `dotted`, or you could `bend right` or `bend left`. You can also change `-` to an arrow `->`.

Example 4.10: Shaping Paths

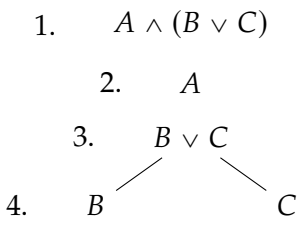



```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                          {$A \wedge (B \vee C)$};
  \node  (A)    [below=of A&BvC]          {$A$};
  \node  (BvC)  [below=of A]              {$B \vee C$};
  \node  (B)    [below left=5mm of BvC]   {$B$};
  \node  (C)    [below right=5mm of BvC] {$C$};

  \path  (BvC)  edge[<-, dashed]  (B);
  \path  (BvC)  edge[->, bend left=45]  (C); % Bend in degrees
\end{tikzpicture}
\end{tikzpicture}
```

4.2.4 Numbers

Line numbers can themselves be treated as nodes.

Example 4.11: Line Numbers




```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                          {$A \wedge (B \vee C)$};
  \node  (A)     [below=of A&BvC]         {$A$};
  \node  (BvC)   [below=of A]             {$B \vee C$};
  \node  (B)     [below left=5mm of BvC]  {$B$};
  \node  (C)     [below right=5mm of BvC] {$C$};
```

```
  \path  (BvC)  edge[-]  (B);
  \path  (BvC)  edge[-]  (C);

  \node  (1)  [left=5mm of A&BvC]  {1.};
  \node  (2)  [left=5mm of A]      {2.};
  \node  (3)  [left=5mm of BvC]    {3.};
  \node  (4)  [left=5mm of B]      {4.};
\end{tikzpicture}
```

However, that doesn't look very nice. To align the numbers vertically, we need to consider an expansion of the `\node` command:

`\node (⟨name⟩) at (⟨position⟩) [⟨options⟩] {⟨formula⟩};`

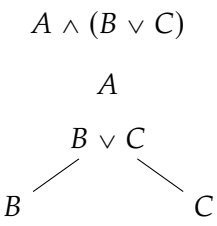
A lot of things could be specified with ⟨*position*⟩. But one in particular makes use of `|-` and `-|`. If one writes before the optional parameters

`at (⟨name$_v$⟩ |- ⟨name$_h$⟩)`

then the resulting node will be *vertically* aligned to ⟨$name_v$⟩ and *horizontally* aligned to ⟨$name_h$⟩. The result is the same if one writes instead:

`at (⟨name$_h$⟩ -| ⟨name$_v$⟩)`

Example 4.12: Line Numbers Aligned



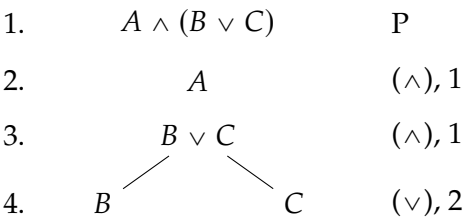

```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                          {$A \wedge (B \vee C)$};
  \node  (A)     [below=of A&BvC]         {$A$};
  \node  (BvC)   [below=of A]             {$B \vee C$};
  \node  (B)     [below left=5mm of BvC]  {$B$};
  \node  (C)     [below right=5mm of BvC] {$C$};

  \path  (BvC)  edge[-]  (B);
  \path  (BvC)  edge[-]  (C);
```

```
  \node  (1)  [left=1cm of A&BvC]  {1.};
  \node  (2)  at (1 |- A)          {2.};
  \node  (3)  at (1 |- BvC)        {3.};
  \node  (4)  at (1 |- B)          {4.};
\end{tikzpicture}
```

4.2.5 Rules

Rule citations can be done in the same way that numbers are done. However, to ensure that the rule citations are aligned properly, I would recommend putting the label in the optional parameters rather than as the formula.

Example 4.13: Line Numbers Aligned

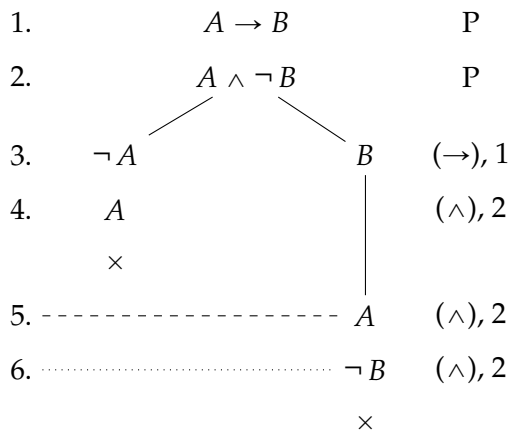



```
\begin{tikzpicture}[node distance=1ex]
  \node  (A&BvC)                          {$A \wedge (B \vee C)$};
  \node  (A)    [below=of A&BvC]          {$A$};
  \node  (BvC)  [below=of A]              {$B \vee C$};
  \node  (B)    [below left=5mm of BvC]   {$B$};
  \node  (C)    [below right=5mm of BvC]  {$C$};

  \path  (BvC)  edge[-]  (B);
  \path  (BvC)  edge[-]  (C);

  \node  (1)  [left=1cm of A&BvC]  {1.};
  \node  (2)  at (1 |- A)          {2.};
  \node  (3)  at (1 |- BvC)        {3.};
  \node  (4)  at (1 |- B)          {4.};

  \node  (r1) [right=1cm of A&BvC, label=right:{P}]          {};
  \node  (r2) at (r1 |- A)   [label=right:{($\wedge$), 1}]   {};
  \node  (r3) at (r1 |- BvC) [label=right:{($\wedge$), 1}]   {};
  \node  (r4) at (r1 |- B)   [label=right:{($\vee$), 2}]     {};
\end{tikzpicture}
```

You can do the same with numbers, to ensure that they're aligned correctly. But unless you have a lot of numbers, this usually isn't a problem.

4.2.6 An Illustrative Example

Example 4.14: Illustrating `tikzpicture`




```
\begin{tikzpicture}[node distance=1ex]
  \node    (A->B)                                  {$A \rightarrow B$};
  \node    (A&-B)    [below=of A->B]               {$A \wedge \neg B$};
  \node    (-A)      [below left=7mm of A&-B]      {$\neg A$};
  \node    (A)       [below=of -A]                 {$A$};
  \node    (x-A)     [below=of A]                  {$\times$};
  \node    (B)       [below right=7mm of A&-B]     {$B$};
  \node    (A2)      at (x-A -| B) [yshift=-7mm]   {$A$};
  \node    (-B)      [below=of A2]                 {$\neg B$};
  \node    (x-B)     [below=of -B]                 {$\times$};

  \path    (A&-B)    edge[-]    (-A);
  \path    (A&-B)    edge[-]    (B);
  \path    (B)       edge[-]    (A2);

  \node    (1)    [left=2cm of A->B]    {1.};
  \node    (2)    at (1 |- A&-B)        {2.};
  \node    (3)    at (1 |- -A)          {3.};
  \node    (4)    at (1 |- A)           {4.};
  \node    (5)    at (1 |- A2)          {5.};
  \node    (6)    at (1 |- -B)          {6.};

  \path    (5)    edge[-,dashed]    (A2);
  \path    (6)    edge[-,dotted]    (-B);

  \node    (r1)   [right=2cm of A->B, label=right:{P}]                {};
  \node    (r2)   at (r1 |- A&-B) [label=right:{P}]                   {};
```

```
  \node    (r3)   at (r1 |- -A)   [label=right:{($\rightarrow$), 1}]  {};
  \node    (r4)   at (r1 |- A)    [label=right:{($\wedge$), 2}]       {};
  \node    (r5)   at (r1 |- A2)   [label=right:{($\wedge$), 2}]       {};
  \node    (r6)   at (r1 |- -B)   [label=right:{($\wedge$), 2}]       {};
\end{tikzpicture}
```