"payload":"allShortcutsEnabled":true,"fileTree":"":"items":["name":"README.md","path":"README.md'
05-11T03:37:41.000-04:00","ownerAvatar":"https://avatars.githubusercontent.com/u/198281?v=4","public":tru

Untitled Lean Thesis

Logan Johnson

Invalid Date

Table of contents

Preface

I will not do my homework today.

```
sum(4, 7, 3)
```

```
[1] 14
```

Hello World

Making Chapters

I am using this section to figure out how to incorporate a table of contents and different sections/chapters of the paper. This should prove useful in the final thesis and allow readers to quickly jump to important or interesting sections.

Incorporating Some Code

I will also be able to use some LaTeX equations within the document which could halp to make the paper look quite nice.

If $a < b$ and $c \leq d$, prove that $a + c \leq b + d$. It just so happens that I was able to prove this using lean!

```
example (a b c d : ) (h1: a < b) (h2 : c  d) : a + c < b + d := by
  by_cases h3 : c = d
  rw [h3]
  apply add_lt_add_right h1
  push_neg at h3
  have h4 : c < d := by
    apply Ne.lt_of_le h3 h2
  apply add_lt_add h1 h4
  done
```

Now to display the benefits of the lean infoview!

Showing the Infoview with Picture Sequences

Not going to do this now as I think the columns are far superior.

Showing Infoview with Columns

Lean File

```
example (a b c d : ) (h1: a < b) (h2 : c  d) : a + c < b + d := by

  done
```

Infoview

```
R: Type u_1
inst: Ring R
abcd:
h1: a < b
h2: c  d
 a + c < b + d
```

As we can clearly see, this is the first step of the code and it can now be explained with great ease. Now onto the next step!

Lean File

```
example (a b c d  ) (h1 a  b) (h2  c  d)  a  c  b  d = by
  by_cases h3  c = d

  done
```

Infoview

```
case pos
R: Type u_1
inst: Ring R
abcd:
h1: a < b
h2: c  d
h3: c = d
```

```
 a + c < b + d
case neg
R: Type u_1
inst: Ring R
abcd:
h1: a < b
h2: c  d
h3: ¬c = d
 a + c < b + d
```

We can see here that lean is smart enough to split our goal into two different goals, where each one uses a different case by our assumption.

I believe that displaying the code in unevaluated R chunks could prove useful as it looks nicer than just being a basic preformatted chunk and it takes less work than doing css work or something.

Lean File

```
theorem Example_3_2_4_v2 (P Q R : Prop)
    (h : P → (Q → R)) : ¬R → (P → ¬Q) := by
  assume h2 : ¬R
  assume h3 : P
  done
```

Tactic State in Infoview

```
P Q R : Prop
h : P → Q → R
h2 : ¬R
h3 : P
 ¬Q
```

About This Book

This book is intended to accompany my book *How To Prove It: A Structured Approach, 3rd edition* (henceforth called *HTPI*), which is published by Cambridge University Press. Although this book is self-contained, we will sometimes have occasion to refer to passages in *HTPI*, so this book will be easiest to understand if you have a copy of *HTPI* available to you.

HTPI explains a systematic approach to constructing mathematical proofs. The purpose of this book is to show you how to use a computer software package called *Lean* to help you master the techniques presented in *HTPI*. Lean is free software that is available for Windows, MacOS, and Unix computers. To get the most out of this book, you will need to download and install Lean on your computer. We will explain how to do that below.

The chapters and sections of this book are numbered to match the sections of *HTPI* to which they correspond. The first two chapters of *HTPI* cover preliminary topics in elementary logic and set theory that are needed to understand the proof techniques presented in later chapters. We assume that you are already familiar with that material (if not, go read those chapters in *HTPI*!), so Chapters 1 and 2 of this book will just briefly summarize the most important points.

Those chapters are followed by an introduction to Lean that explains the basics of using Lean to write proofs. The presentation of proof techniques in *HTPI* begins in earnest in Chapter 3, so that is where we will begin to discuss how Lean can be used to master those techniques.

If you are reading this book online, then at the end of the title in the left margin you will find an icon that is a link to a pdf version of the book. Below that is a search box, which you can use to search for any word or phrase anywhere in the book. Below the search box is a list of the chapters of the book. Click on any chapter to go to that chapter. Within each chapter, a table of contents in the right margin lists the sections in that chapter. Again, you can go to any section by clicking on it. At the end of each chapter there are links to take you to the next or previous chapter.

About Lean

Lean is a kind of software package called a *proof assistant*. What that means is that Lean can help you to write proofs. As we will see over the course of this book, there are several ways in which Lean can be helpful. First of all, if you type a proof into Lean, then Lean can check the correctness of the proof and point out errors. As you are typing a proof into Lean, it will keep track of what has been accomplished so far in the proof and what remains to be done to finish the proof, and it will display that information for you. That can keep you moving in the right direction as you are figuring out a proof. And sometimes Lean can fill in small details of the proof for you.

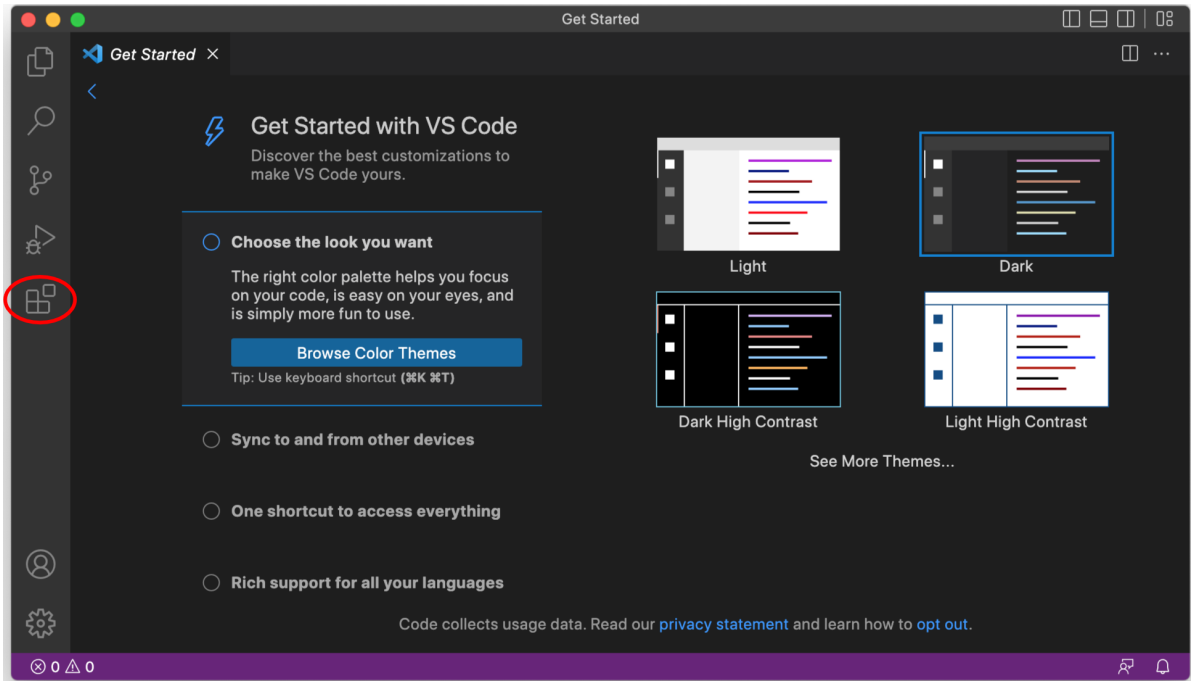
Of course, to make this possible, you must type your proof in a format that Lean understands. Much of this book will be taken up with explaining how to write a proof so that Lean will understand it.

Installing Lean

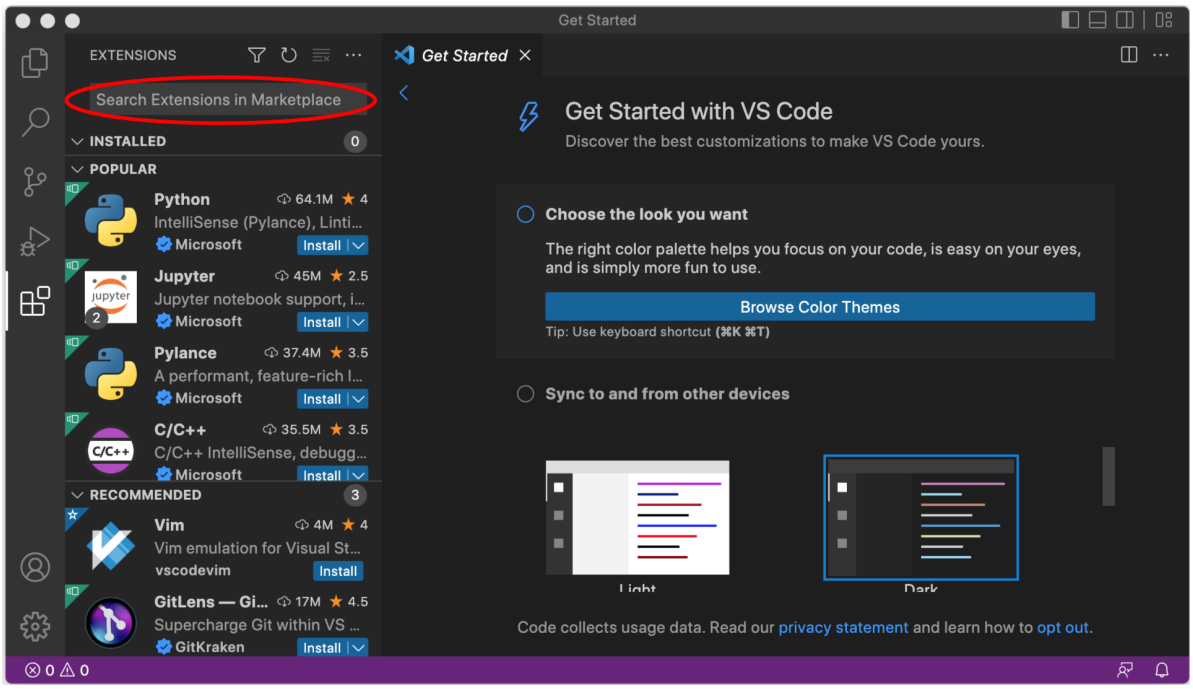
We will be using Visual Studio Code to run Lean, so you will need to install VS Code first. VS Code is free and can be downloaded here.

You will also need the Lean package that accompanies this book, which can be downloaded from https://github.com/djvelleman/HTPILeanPackage. After following the link, click on the green "Code" button and, in the pop-up menu, select "Download ZIP". Open the downloaded zip file to create a folder containing the HTPI Lean package. You can put this folder wherever you want on your computer.

Now open VS Code. You should see a window that looks something like this:

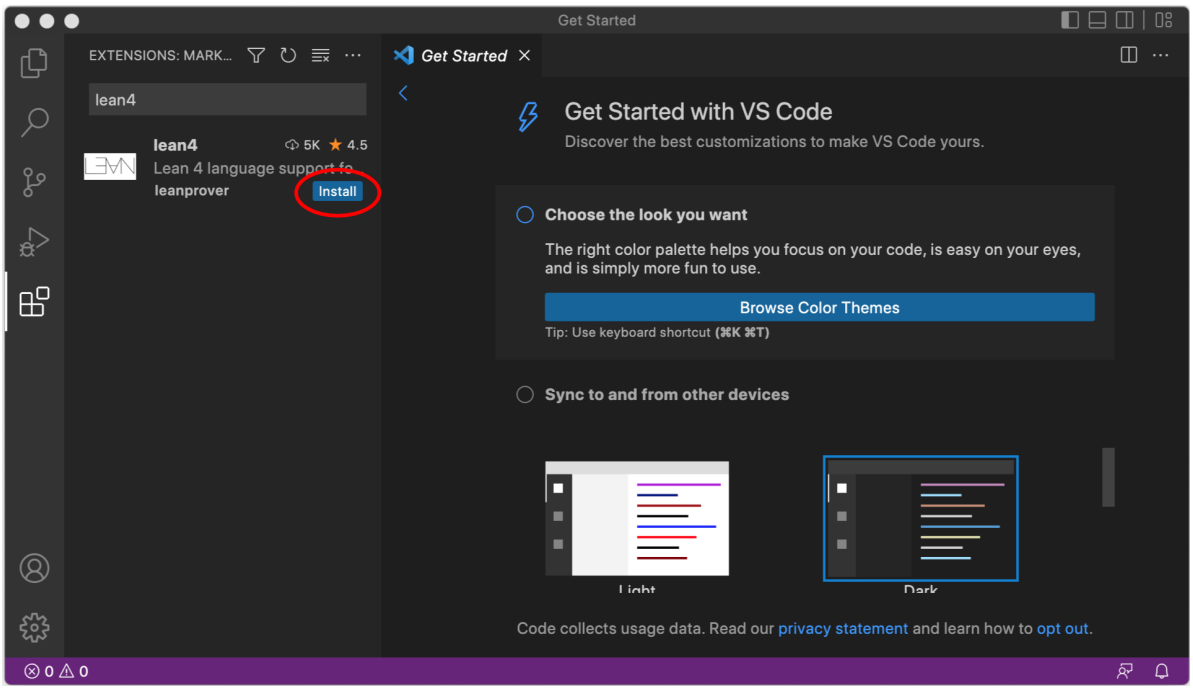


Click on the *Extensions* icon on the left side of the window, which is circled in red in the image above. That will bring up a list of available extensions:



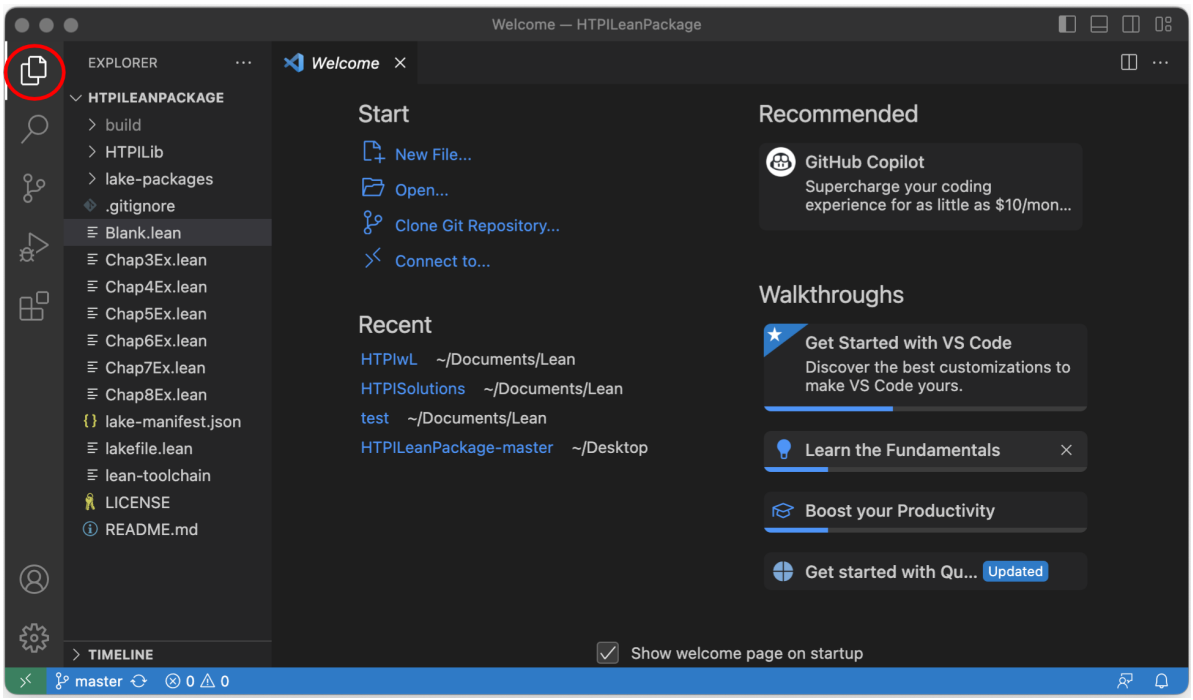
In the *Search Extensions in Marketplace* field, type "lean4". VS Code should find the Lean 4

extension and display it:

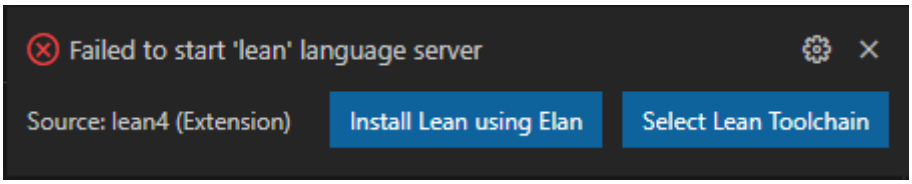


Click on "Install" to install the Lean 4 extension.

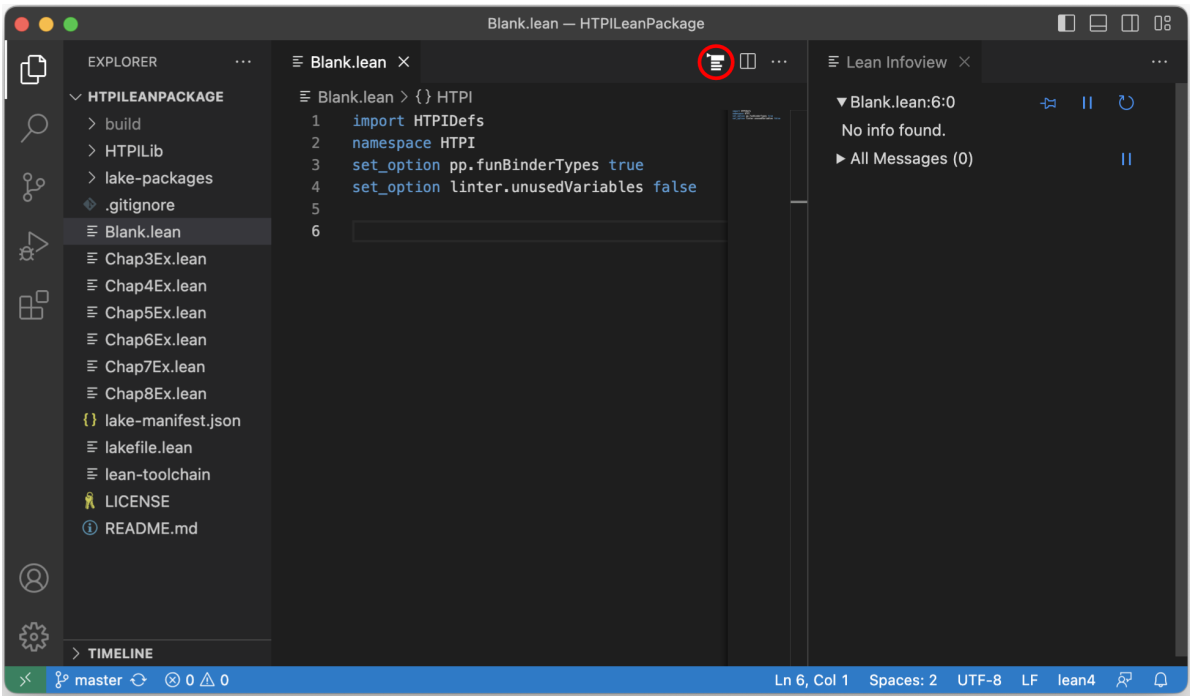
Next, in VS Code, select "Open Folder . . . " from the File menu and open the folder containing the HTPI Lean package that you downloaded earlier. Under the heading "Explorer" on the left side of the window, you should see a list of the files in the package. (If you don't see the list, try clicking on the *Explorer* icon, circled in red below.)



Click on the file "Blank.lean" in the file list. You should see a warning that VS Code failed to start the 'lean' language server:



Click on the "Install Lean using Elan" button, and the Lean server should be installed. Then Lean should "build" the HTPI Lean package. This may take a while, but it only has to be done once. If anything goes wrong, try quitting VS Code and restarting. Eventually your window should look like this:



If you don't see the Infoview pane on the right side of the window, click on the icon circled in red in the image above, and the Infoview pane should appear.

Your installation is now complete.

About the HTPI Lean Package

For each chapter, starting with "Introduction to Lean," the package has a file containing all Lean definitions and theorems from that chapter. There are also files containing all the exercises. In the exercise file for a chapter, all definitions and theorems from that chapter and all earlier chapters are available for use in solving the exercises.

The chapter files are inside the folder "HTPILib". There is also one other file in that folder. All of these files are needed to make the package work. *Do not edit the files in the HTPILib folder*, or you will need to re-download the package.

License

This book is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. This license allows you to share or adapt the book. In any adaptation, you must identify Daniel J. Velleman as the author, and you must also acknowledge that excerpts from

Acknowledgments

A number of people have provided advice, encouragement, and feedback about this project. In particular, I would like to thank Jeremy Avigad, Clayton Cafiero, François Dorais, Charles Hoskinson, Heather Macbeth, Pietro Monticone, and Ketil Wright.

1 Real Analysis

Symbol	Meaning
$\neg$	not
$\wedge$	and
$\vee$	or
$\rightarrow$	if ... then
$\leftrightarrow$	iff (that is, if and only if)

Name			
De Morgan's Laws	$\neg(P \wedge Q)$	is equivalent to	$\neg P \vee \neg Q$
	$\neg(P \vee Q)$	is equivalent to	$\neg P \wedge \neg Q$
Double Negation Law	$\neg\neg P$	is equivalent to	P
Conditional Laws	$P \rightarrow Q$	is equivalent to	$\neg P \vee Q$
	$P \rightarrow Q$	is equivalent to	$\neg(P \wedge \neg Q)$
Contrapositive Law	$P \rightarrow Q$	is equivalent to	$\neg Q \rightarrow \neg P$

$A \cap B = \{x \mid x \in A \wedge x \in B\} =$ the *intersection* of A and B,

$A \cup B = \{x \mid x \in A \vee x \in B\} =$ the *union* of A and B,

$A \setminus B = \{x \mid x \in A \wedge x \notin B\} =$ the *difference* of A and B,

$A \triangle B = (A \setminus B) \cup (B \setminus A) =$ the *symmetric difference* of A and B.

2 Functional Programming

$\forall x\, P(x)$ means "for all x, $P(x)$,"

Quantifier Negation Laws		
$\neg\exists x\, P(x)$	is equivalent to	$\forall x\, \neg P(x)$
$\neg\forall x\, P(x)$	is equivalent to	$\exists x\, \neg P(x)$

3 Lean as a Theorem Prover

3.1 & 3.2. Proofs Involving Negations and Conditionals

To prove a goal of the form `P → Q`:

1. Assume `P` is true and prove `Q`.
2. Assume `Q` is false and prove that `P` is false.

Tactic State Before Using Strategy

```
P → Q
```

Tactic State After Using Strategy

```
h : P
 Q
```

Q	¬Q	(Q	→	False)
F	T	F	T	F
T	F	T	F	F

Lean File

```
theorem Example_3_2_4_v2 (P Q R : Prop)
    (h : P → (Q → R)) : ¬R → (P → ¬Q) := by
  assume h2 : ¬R
  assume h3 : P
  done
```

Tactic State in Infoview

```
P Q R : Prop
h : P → Q → R
h2 : ¬R
h3 : P
 ¬Q
```

Exercises

Fill in proofs of the following theorems. All of them are based on exercises in *HTPI*.

1.
```
theorem Exercise_3_2_1a (P Q R : Prop)
    (h1 : P → Q) (h2 : Q → R) : P → R := by

  done
```

2.
```
theorem Exercise_3_2_1b (P Q R : Prop)
    (h1 : ¬R → (P → ¬Q)) : P → (Q → R) := by

  done
```

3.
```
theorem Exercise_3_2_2a (P Q R : Prop)
    (h1 : P → Q) (h2 : R → ¬Q) : P → ¬R := by

  done
```

4.
```
theorem Exercise_3_2_2b (P Q : Prop)
    (h1 : P) : Q → ¬(Q → ¬P) := by

  done
```

3.3. Proofs Involving Quantifiers

Let `x` stand for an arbitrary object of type `U` and prove `P x`. If the letter `x` is already being used in the proof to stand for something, then you must choose an unused variable, say `y`, to stand for the arbitrary object, and prove `P y`.

`quant_neg` Tactic		
`¬ (x : U), P x`	is changed to	`(x : U), ¬P x`
`¬ (x : U), P x`	is changed to	`(x : U), ¬P x`
`(x : U), P x`	is changed to	`¬ (x : U), ¬P x`
`(x : U), P x`	is changed to	`¬ (x : U), ¬P x`

Theorem. *Suppose B is a set and $\mathcal{F}$ is a family of sets. If $\bigcup \mathcal{F} \subseteq B$ then $\mathcal{F} \subseteq \mathscr{P}(B)$.*

Proof. Suppose $\bigcup \mathcal{F} \subseteq B$. Let x be an arbitrary element of $\mathcal{F}$. Let y be an arbitrary element of x. Since $y \in x$ and $x \in \mathcal{F}$, by the definition of $\bigcup \mathcal{F}$, $y \in \bigcup \mathcal{F}$. But then since $\bigcup \mathcal{F} \subseteq B$,

$y \in B$. Since y was an arbitrary element of x, we can conclude that $x \subseteq B$, so $x \in \mathscr{P}(B)$. But x was an arbitrary element of $\mathcal{F}$, so this shows that $\mathcal{F} \subseteq \mathscr{P}(B)$, as required. □

Theorem 3.4.7. *For every integer n, $6 \mid n$ iff $2 \mid n$ and $3 \mid n$.*

Proof. Let n be an arbitrary integer.

($\rightarrow$) Suppose $6 \mid n$. Then we can choose an integer k such that $6k = n$. Therefore $n = 6k = 2(3k)$, so $2 \mid n$, and similarly $n = 6k = 3(2k)$, so $3 \mid n$.

($\leftarrow$) Suppose $2 \mid n$ and $3 \mid n$. Then we can choose integers j and k such that $n = 2j$ and $n = 3k$. Therefore $6(j - k) = 6j - 6k = 3(2j) - 2(3k) = 3n - 2n = n$, so $6 \mid n$. □

For the next exercise you will need the following definitions:

4 Conclusions

$$[x]_R = \{y \in A \mid yRx\}.$$

The set whose elements are all of these equivalence classes is called A *mod* R. It is written A/R, so

$$A/R = \{[x]_R \mid x \in A\}.$$

Note that A/R is a set whose elements are sets: for each $x \in A$, $[x]_R$ is a subset of A, and $[x]_R \in A/R$.

5 Works Cited

This work had been formatted and styled from the book *How To Prove It With Lean*, written by Daniel J. Velleman. *How To Prove It With Lean* contains short excerpts from *How To Prove It: A Structured Approach, 3rd Edition*, by Daniel J. Velleman and published by Cambridge University Press.

```
example : square1 = square2 := by rfl

#eval square1 7     --Answer: 49
```

6 Additional space

Extra chapter to write more things if needed!!