UNTITLED LEAN THESIS

A Thesis Submitted to the Faculty of

Georgetown College

In Partial Fulfillment of the Requirements for the

Honors Program

By

Logan Johnson

Georgetown, Kentucky

May 2024

Abstract

UNTITLED LEAN THESIS

Logan C. Johnson

idk dr burch?? maybe dr white???

Here is the text of your abstract. It goes on and on and on. It goes on like this for about 150 words, so it should all fit on this page. Note that the Abstract comes before the title page and has no page number. The rest of this paragraph is a filler. It goes on like this for about 150 words, so it should all fit on this page. Note that the Abstract comes before the title page and has no page number. It goes on like this for about 150 words, so it should all fit on this page. Note that the Abstract comes before the title page and has no page number. It goes on like this for about 150 words, so it should all fit on this page. Note that the Abstract comes before the title page and has no page number. It goes on like this for about 150 words, so it should all fit on this page. Note that the Abstract comes before the title page and has no page number. It goes on like this for about 150 words, so it should all fit on this page. Note that the Abstract comes before the title page and has no page number. It goes on like this for about 150 words, so it should all fit on this page. Note that the Abstract comes before the title page and has no page number. If your abstract is more than 250 words, consider shortening it.

APPROVED BY THE DIRECTOR OF HONORS THESES:

_____

Dr. Homer White, Department of Mathematics

APPROVED BY THE HONORS PROGRAM:

_____

Dr. Barbara Burch, Director

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 : ℝ) (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

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!
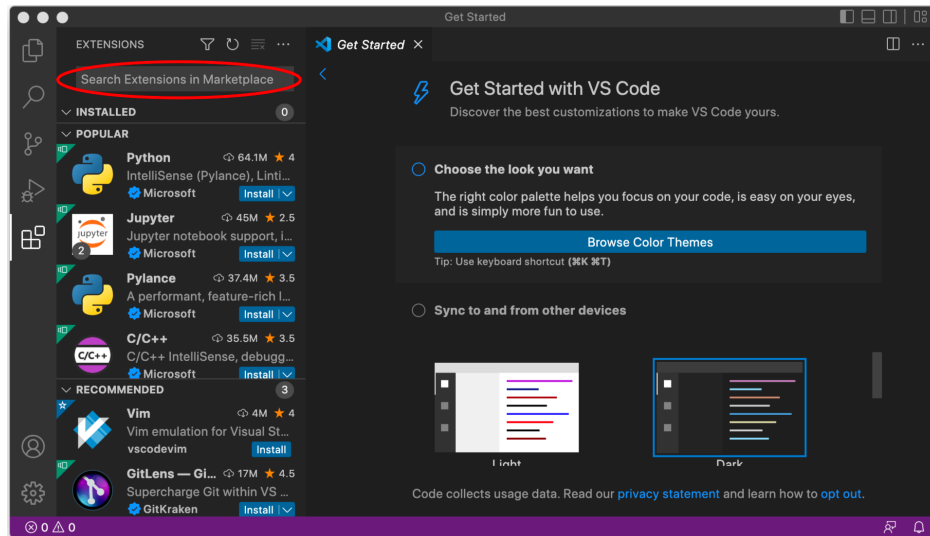
<u>Lean File</u>   <u>Tactic State in Infoview</u>

```
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
```

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

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:

Acknowledgments

1 Real Analysis

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

Name	Equivalence		
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	$\neg Q \rightarrow \neg P$

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

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

$A \setminus B = \{x \mid x \in A \land 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

Differences From Paragraph Style Proofs

Despite the incredible power that lean could provide in the verification of mathematical proofs, this does pose some difficulties, namely the ease with which the aforementioned proofs can be written up. Typically, proofs are simply written up in a paragraph style, where the steps being taken and the theorems being applied are laid out in plain terms so that it can be easily understood by fellow mathematicians. There are often times when mathematicians will take things for granted or skip over steps that they think the reader will either already know to be fact or can easily reason out for themselves when writing out typical proofs. This lax approach for conveying information simply does not work when trying to communicate with technology, and a much more specific and methodical approach must be adopted in order to take advantage of the logical verification benefits. Thankfully, lean has a community working to create libraries of previously proven theorems that can be applied to speed up the writing and verification of future proofs. This thankfully means that all proofs do not need to be taken all the way back to basic axioms: Users can save time by avoiding proving adjacent theorems and instead focus only on the immediately relevant steps of their proof.

Inequality Addition

Step 1 Putting the initial theorem we want to show in lean code and observing the final goal in the infoview.

<div style="display: flex; justify-content: space-between;">

Lean File

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

  done
```

Tactic State in Infoview

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

</div>

Step 2

Lean File

```
example (a b c d : ℝ) (h1: a < b)
    (h2 : c ≤ d) : a + c < b + d := by
  by_cases h3 : c = d

  done
```

Tactic State in Infoview

```
R: Type u_1
inst†: Ring R
abcd: ℝ
h1: a < b
h2: c ≤ d
h3: c = d
⊢ a + c < b + d
```

Step 3

Lean File

```
example (a b c d : ℝ) (h1: a < b)
    (h2 : c ≤ d) : a + c < b + d := by
  by_cases h3 : c = d
  rw [h3]

  done
```

Tactic State in Infoview

```
R: Type u_1
inst†: Ring R
abcd: ℝ
h1: a < b
h2: c ≤ d
h3: c = d
⊢ a + d < b + d
```

Step 4

Lean File

```
example (a b c d : ℝ) (h1: a < b)
    (h2 : c ≤ d) : a + c < b + d := by
  by_cases h3 : c = d
  rw [h3]
  apply add_lt_add_right h1

  done
```

Tactic State in Infoview

```
R: Type u_1
inst†: Ring R
abcd: ℝ
h1: a < b
h2: c ≤ d
h3: ¬c = d
⊢ a + c < b + d
```

Step 5

Lean File

```
example (a b c d : ℝ) (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

  done
```

Tactic State in Infoview

```
R: Type u_1
inst†: Ring R
abcd: ℝ
h1: a < b
h2: c ≤ d
h3: c ≠ d
⊢ a + c < b + d
```

Step 6a We can see here that the lean infoview is now displaying my new hypothesis as the current goal.

Lean File   Tactic State in Infoview

```
example (a b c d : ℝ) (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

  done
```

```
R: Type u_1
inst†: Ring R
abcd: ℝ
h1: a < b
h2: c ≤ d
h3: c ≠ d
⊢ c < d
```

Step 6b

Lean File   Tactic State in Infoview

```
example (a b c d : ℝ) (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

  done
```

```
No goals
```

Step 7

Lean File

```
example (a b c d : ℝ) (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
```

Tactic State in Infoview

No goals

Absolute Convergence

I will now display how I used to lean to demonstrate that a sequence converges if and only if the absolute value of that sequence converges.

Laying out the goal

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by

  done
```

Tactic State in Infoview

```
s1: ℕ → ℝ
⊢ ConvergesTo s1 0 ↔ ConvergesTo |s1|
```

Step 1

Lean File

```lean
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]

  done
```

Tactic State in Infoview

```
s1: ℕ → ℝ
⊢ (∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ),
  ∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ), n
```

Step 2

Lean File

```lean
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by

    done

  done
```

Tactic State in Infoview

```
s1: ℕ → ℝ
x: ℕ
⊢ |s1 x| = |abs s1 x|
```

Step 3

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done

  done
```

Tactic State in Infoview

```
s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
⊢ (∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ),
  ∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ), n
```

Step 4

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards

  · --Reverse

  done
```

Tactic State in Infoview

```
s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
⊢ (∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ),
  ∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ), n
```

Step 5

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards
    intro h1

  · --Reverse

  done
```

Step 6

Tactic State in Infoview

```
s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
h1: ∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ),
⊢ ∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ), n
```

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards
    intro h1
    simp

  · --Reverse

  done
```

Step 7

Tactic State in Infoview

```
s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
h1: ∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ),
⊢ ∀ (ε : ℝ), 0 < ε → ∃ N, ∀ (n : ℕ), N
```

Absolute Convergence

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards
    intro h1
    simp
    simp at h1

  · --Reverse

  done
```

Step 8

Tactic State in Infoview

```
s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
h1: ∀ (ε : ℝ), 0 < ε → ∃ N, ∀ (n : ℕ),
⊢ ∀ (ε : ℝ), 0 < ε → ∃ N, ∀ (n : ℕ), N
```

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards
    intro h1
    simp
    simp at h1
    simp [← h3]

  · --Reverse

  done
```

Step 9

Tactic State in Infoview

```
s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
h1: ∀ (ε : ℝ), 0 < ε → ∃ N, ∀ (n : ℕ),
⊢ ∀ (ε : ℝ), 0 < ε → ∃ N, ∀ (n : ℕ), N
```

Absolute Convergence

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards
    intro h1
    simp
    simp at h1
    simp [← h3]
    apply h1
  · --Reverse

  done
```

Step 10

Tactic State in Infoview

s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
⊢ (∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ),
  ∀ (ε : ℝ), ε > 0 → ∃ N, ∀ (n : ℕ), n

Lean File

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards
    intro h1
    simp
    simp at h1
    simp [← h3]
    apply h1
  · --Reverse
    intro h1
    simp
    simp at h1
    simp_rw [h3]

  done
```

Step 11

Tactic State in Infoview

s1: ℕ → ℝ
h3: ∀ (x : ℕ), |s1 x| = |abs s1 x|
h1: ∀ (ε : ℝ), 0 < ε → ∃ N, ∀ (n : ℕ),
⊢ ∀ (ε : ℝ), 0 < ε → ∃ N, ∀ (n : ℕ), N

Lean File

Tactic State in Infoview

```
example (s1 : ℕ → ℝ) : ConvergesTo s1 (0 : ℝ) ↔
    ConvergesTo (abs s1) (0 : ℝ) := by
  rw [ConvergesTo]
  rw [ConvergesTo]
  have h3 (x : ℕ) : |s1 x| = |abs s1 x| := by
    simp [abs]
    done
  apply Iff.intro
  · --Forwards
    intro h1
    simp
    simp at h1
    simp [← h3]
    apply h1
  · --Reverse
    intro h1
    simp
    simp at h1
    simp_rw [h3]
    apply h1
  done
```

No goals

Perhaps a Third Great Example??

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.

Perhaps a Third Great Example??

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. $\square$

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 $6 \mid n$. $\square$

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!!