CPSC-354 Report

Sarah Yoon
Chapman University

September 8, 2024

Abstract

Contents

1 Introduction **1**

2 Week by Week **1**
2.1 Week 1 1
2.2 Week 2 3
2.3 7

3 Lessons from the Assignments **7**

4 Conclusion **7**

1 Introduction

2 Week by Week

2.1 Week 1

Notes

Learned about some tactics and theorems

rfl: a tactic that proves theorems that take the form of X = X
rw: a tactic that rewrites a proof
one_eq_succ_zero: a theorem that proves 1 = succ 0 (there are also other similar existing theorems like two_eq_succ_one and so on)
add_zero: a theorem that proves a + 0 = a.
add_succ: a theorem that proves a + succ b = succ(a + b)
succ_eq_add_one: a theorem that proves succ a = a + 1

Homework

Problem 5:
a b c are in the set of natural numbers.
Prove that both sides are equal to each other.

a + (b + 0) + (c + 0) = a + b + c
rw [add_zero] - uses the add_zero theorem to prove that b + 0 = b
This is rewritten to:
a + b + (c + 0) = a + b + c
rw [add_zero] - this is done again to prove that c + 0 = c
This is rewritten to:
a + b + c = a + b + c
rfl - this proves that both sides that look the same are equal to each other

Problem 6:
This is the same problem as 5 but will be approached in a different manner.
a + (b + 0) + (c + 0) = a + b + c
rw [add_zero c] - specifically applies the add_zero theorem to c, making c + 0 into c
This is rewritten to:
a + (b + 0) + c = a + b + c
rw [add_zero b] - specifically applies the add_zero theorem to b, making b + 0 into b
This is rewritten to:
a + b + c = a + b + c
rfl - this proves that both sides that look the same are equal to each other

Problem 7:
n is in the set of natural numbers.
Prove that both sides are equal to each other.
succ n = n + 1 rw[one_eq_succ_zero] - rewrite 1 into successor 0 This is rewritten to:
succ n = n + succ 0
rw[add_succ] - uses the add_succ theorem to change n + succ 0 into succ(n + 0)
This is rewritten to:
succ n = succ (n + 0)
rw[add_zero] - uses the add_zero theorem to prove that n + 0 = n
This is rewritten to:
succ n = succ n
rfl - this proves that both sides that look the same are equal to each other

Problem 8:
Prove that both sides are equal to each other.
2 + 2 = 4
rw[two_eq_succ_one] - rewrites 2 into succ 1
This is rewritten to:
succ 1 + succ 1 = 4
rw[one_eq_succ_zero] - rewrites 1 into succ 0
This is rewritten to:
succ (succ 0) + succ (succ 0) = 4
rw[four_eq_succ_three] - rewrites 4 into succ 3
This is rewritten to:
succ (succ 0) + succ (succ 0) = succ 3
rw[three_eq_succ_two] - rewrites 3 into succ 2
This is rewritten to:
succ (succ 0) + succ (succ 0) = succ (succ 2)
rw[two_eq_succ_one] - rewrites 2 into succ 1
This is rewritten to:

succ (succ 0) + succ (succ 0) = succ (succ (succ 1))
rw[one_eq_succ_zero] - rewrites 1 into succ 0
This is rewritten to:
succ (succ 0) + succ (succ 0) = succ (succ (succ (succ 0)))
rw[add_succ] - changes succ (succ 0) + succ (succ 0) into succ (succ (succ 0) + succ 0)
This is rewritten to:
succ (succ (succ 0) + succ 0) = succ (succ (succ (succ 0)))
rw[add_succ] - changes succ (succ (succ 0) + succ 0) into succ (succ (succ (succ 0) + 0))
This is rewritten to:
succ (succ (succ (succ 0) + 0)) = succ (succ (succ (succ 0)))
rw[add_zero] - changes succ (succ 0) + 0 into
This is rewritten to:
succ (succ (succ (succ 0))) = succ (succ (succ (succ 0)))
rfl - this proves that both sides that look the same are equal to each other

For level 5: add_zero is a Lean proof that a + 0 = a (a representing any number). In mathematics, there are laws for arithemic. One of them is called the identity which applies to addition and multiplication. For addition, it states that m + 0 = m = 0 + m. This is the exact same as the Lean proof, a + 0 = a, which can also be written as a = 0 + a.

Comments and Questions

Learning the root of mathematics is very eye-opening, and I am confident it will be the same for programming languages. It provides another perspective for elementary functions like 2 + 2 equals 4, which is different from just knowing it through memorization. I feel as though this is why people have been able to expand mathematically. This makes me wonder: how can looking through the core of programming help us better current languages (e.g. python, rust)?

2.2 Week 2

Notes

Recursion as a concept using the Towers of Hanoi: It is broken down into: moving a tower of n disks from x to y moving a tower of n+1 disks when it is already known how to move a tower of n disks The algorithm is made up of a bunch of "pushs" and "pops" The logic overall is a bunch of back and forth movement of the disks

Lean: induction proof with: induction n with d hd succ_add: proves that succ a + b = succ (a + b) add_comm x y: proves that x + y = y + x add_assoc: proves that a + b + c = a + (b + c) add_right_comm a b c: proves that a + b + c = a + c + b

Homework

Problem 1:
n is in the natural number set
Prove 0 + n = n.
induction n with d hd - starting a proof by induction
Now our first goal is:
0 + 0 = 0
rw[add_zero] - proves that 0 + 0 = 0
This is rewritten to:
0 = 0
rfl - this proves that both sides that look the same are equal to each other

Now, we prove our second goal
hd: 0 + d = d
0 + succ d = succ d
rw[add_succ] - proves that 0 + succ d = succ (0 + d)
This is rewritten to:
succ (0 + d) = succ d
rw[hd] - this replaces 0 + d with d
This is rewritten to:
succ d = succ d
rfl - this proves that both sides that look the same are equal to each other

Problem 2:
a b is in the set of natural numbers
Prove succ a + b = succ (a + b)
inductin b with d hd - starting a proof by induction
Now our first goal is:
succ a + 0 = succ (a + 0)
rw[add_zero] - proves that succ a + 0 = succ a
This is rewritten to:
succ a = succ(a + 0)
rw[add_zero] - proves that succ (a + 0) = succ a
This is rewritten to:
succ a = succ a
rfl - this proves that both sides that look the same are equal to each other
Now, we prove our second goal
hd: succ a + d = succ (a + d)
succ a + succ d = succ (a + succ d)
rw[add_succ] - proves that succ a + succ d = succ (succ a + d)
This is rewritten to:
succ (succ a + d) = succ (a + succ d)
rw[hd] - this replaces succ a + d with succ (a + d)
This is rewritten to:
succ (succ (a + d)) = succ (a + succ d)
rw[add_succ] - proves that succ (a + succ d) = succ (succ (a + d))
This is rewritten to:
succ (succ (a + d)) = succ (succ (a + d))
rfl - this proves that both sides that look the same are equal to each other

Problem 3:
a b is in the set of natural numbers
Prove a + b = b + a
induction b with hd - starting a proof by induction
Now our first goal is:
a + 0 = 0 + a
rw[zero_add] - proves that 0 + a = a
This is rewritten to:
a + 0 = a
rw[add_zero] - proves that a + 0 = a
This is rewritten to:
a = a

rfl - this proves that both sides that look the same are equal to each other
Now, we prove our second goal
n_ih: a + hd = hd + a
a + succ hd = succ hd + a
rw[add_succ] - proves that a + succ hd = succ (a + hd)
This is rewritten to:
succ (a + hd) = succ hd + a
rw[succ_add] - proves that succ hd + a = succ (hd + a)
This is rewritten to:
succ (a + hd) = succ (hd + a)
rw[n_ih] - replaces succ (a + hd) with succ (hd + a)
This is rewritten to:
succ (hd + a) = succ (hd + a)
rfl - this proves that both sides that look the same are equal to each other

Problem 4:
a b c is in the set of natural numbers
Prove a + b + c = a + (b + c)
induction a with hd - starting a proof by induction
Now our first goal is:
0 + b + c = 0 + (b + c)
rw[zero_add] - proves that 0 + b = b
This is rewritten to:
b + c = 0 + (b + c)
rw[zero_add] - proves that 0 + (b + c) = b + c
This is rewritten to:
b + c = b + c
rfl - this proves that both sides that look the same are equal to each other
Now, we prove our second goal
n_ih: hd + b + c = hd + (b + c)
succ hd + b + c = succ hd + (b + c)
rw[succ_add] - proves that succ hd + b + c = succ (hd + b) + c
This is rewritten to:
succ (hd + b) + c = succ hd + (b + c)
rw[succ_add] - proves that succ (hd + b) + c = succ (hd + b + c)
This is rewritten to:
succ (hd + b + c) = succ hd + (b + c)
rw[n_ih] - replaces succ (hd + b + c) with succ (hd + (b + c))
This is rewritten to:
succ (hd + (b + c)) = succ hd + (b + c)
rw[succ_add] - proves succ hd + (b + c) = succ (hd + (b + c))
This is rewritten to:
succ (hd + (b + c)) = succ (hd + (b + c))
rfl - this proves that both sides that look the same are equal to each other

Problem 5:
a b c is in the set of natural numbers
Prove a + b + c = a + c + b
induction c with hd - starting a proof by induction
Now our first goal is:

a + b + 0 = a + 0 + b
rw[add_zero] - proves that b + 0 = b
This is rewritten to:
a + b = a + 0 + b
rw[add_zero] 0 proves that a + 0 = a
This is rewritten to:
a + b = a + b
rfl - this proves that both sides that look the same are equal to each other
Now, we prove our second goal
n_ih: a + b + hd = a + hd + b
a + b + succ hd = a + succ hd + b
rw[add_succ] - proves that a + b + succ hd = succ (a + b + hd)
This is rewritten to:
succ (a + b + hd) = a + succ hd + b
rw[add_succ] - proves that a + succ hd + b = succ (a + hd) + b
This is rewritten to:
succ (a + b + hd) = succ (a + hd) + b
rw[succ_add] - proves that succ (a + hd) + b = succ (a + hd + b)
This is rewritten to:
succ (a + b + hd) = succ (a + hd + b)
rw[n_ih] - replaces a + b + hd with a + hd + b
This is rewritten to
succ (a + hd + b) = succ (a + hd + b)
rfl - this proves that both sides that look the same are equal to each other

Problem 5 Proof in Mathematics:
a + b + c = a + (b + c)
0 + b + c = 0 + (b + c) - Basis
b + c = 0 + (b + c) - Addition Identity
b + c = b + c - Addition Identity
Inductive Step:
k + b + c = k + (b + c)
The goal is to prove that Sk + b + c = Sk + (b + c)
S(k + b + c) = Sk + (b + c) - Definition of Addition
S(k + b + c) = S(k + (b + c)) - Definition of Addition
S(k + (b + c)) = S(k + (b + c)) - Inductive Hypothesis
Therefore, by the Axiom of induction a + b + c = a + (b + c) for all a in the natural numbers set

Math to Lean
Basis: induction a with hd
Addition Identity: zero_add
Definition of Addition: succ_add
Inductive Hypothesis: n_ih

Comments and Questions

The Towers of Hanoi reminded me of solving certain problems by simply using recursion. I also remember applying this method to the Fibonacci sequence. This makes me wonder how it transfers to math. How does recursion appear in mathematics or, specifically, in Lean?

2.3 ...

...

3 Lessons from the Assignments

4 Conclusion

References

[BLA] Author, Title, Publisher, Year.