CPSC-354 Report

Rohm Tandon
Chapman University

07/01/2024

Abstract

This report contains assignments throughout the fall 2024 semester and is intended for the purpose of documenting my work and showing my progress in CPSC 354 - Programming Languages, taught by Jonathan Weinberg.

Contents

1 Introduction

This report, prepared for CPSC 354 - Programming Languages at Chapman University, is a comprehensive account of my academic voyage over the semester. It includes a detailed compilation of my notes, homework solutions, and critical reflections on the coursework. This report serves as a bridge between the theoretical knowledge imparted in lectures and the practical skills essential for future pursuits in both graduate studies and the software industry.

2 Week by Week

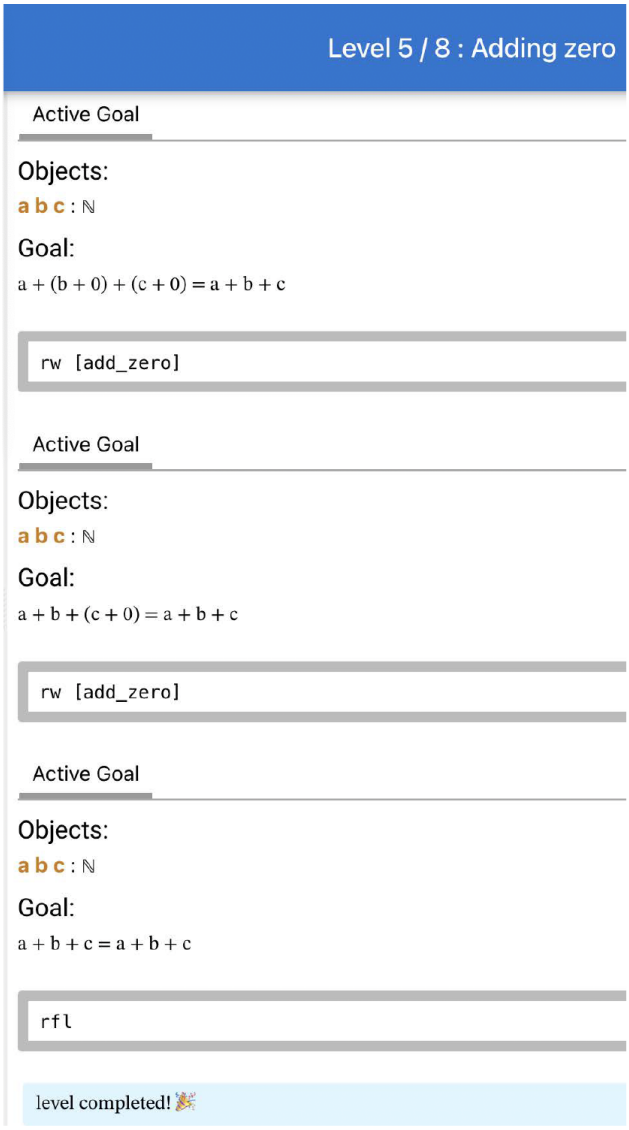
2.1 Week 1

Notes

In week 1 we learnt about Lean as a programming language and its correlation to discrete math. We also learnt about other proof assistants. We then shifted our focus to the NNG tutorial world as you can see below.

Homework

Tutorial world



Level 5:

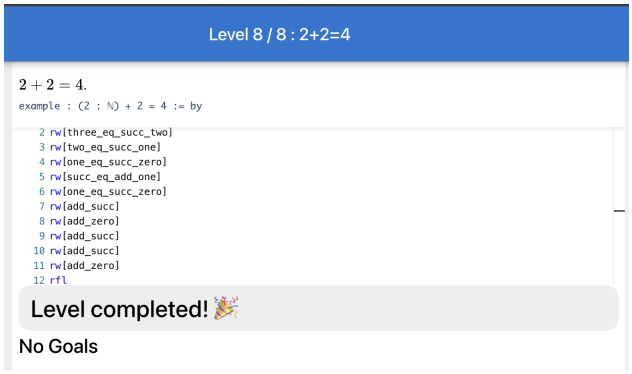
Discrete math's lemmas tell us that anything added to 0 will give the result of that number itself. So; A+0=A. Using this we can bring the left hand side down to a+b+c. From here we can use the property of reflexivity to show that both sides are equal, hence solving the puzzle.

Level 6:


Level 6 / 8 : Precision rewriting
Objects:
a b c : ℕ
Goal:
a + (b + 0) + (c + 0) = a + b + c
rw [add_zero c]
Active Goal
Objects:
a b c : ℕ
Goal:
a + (b + 0) + c = a + b + c
rw [add_zero b]
Active Goal
Objects:
a b c : ℕ
Goal:
a + b + c = a + b + c
rfl
level completed! 🎉

Level 7:

Level 8:



Comments and Questions

(Delete and Replace:) Here you should write your own critical reflection on the content of the week. If you can surprise me with something I have not seen before, you are on the right track.

Ask at least one **interesting question**[1] on the lecture notes. Also post the question on the Discord channel so that everybody can see and discuss the questions.

[1] It is important to learn to ask *interesting* questions. There is no precise way of defining what is meant by interesting. You can only learn this by doing. An interesting question comes typically in two parts. Part 1 (one or two sentences) sets the scene. Part 2 (one or two sentences) asks the question. A good question strikes the right balance between being specific and technical on the one hand and open ended on the other hand. A question that can be answered with yes/no is not an interesing question.

2.2 Week 2

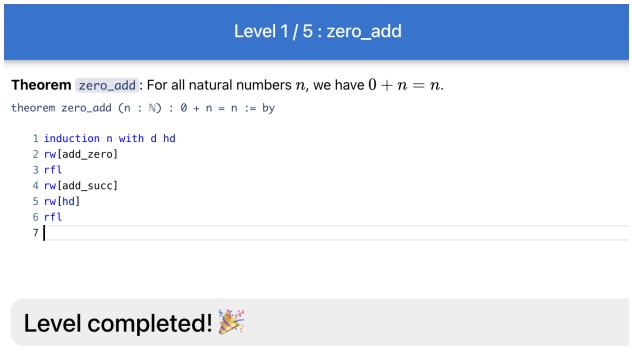
Notes

In week 2 we learnt about recursion and its application in other problems such as the Towers of Hanoi game we played. We also learnt about its various benefits such as breaking down complexity of problems and being more concise.

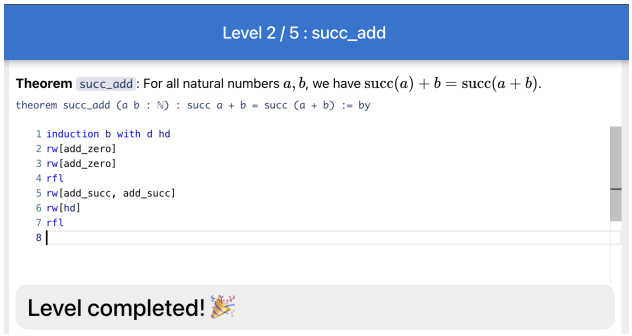
Homework

Addition world

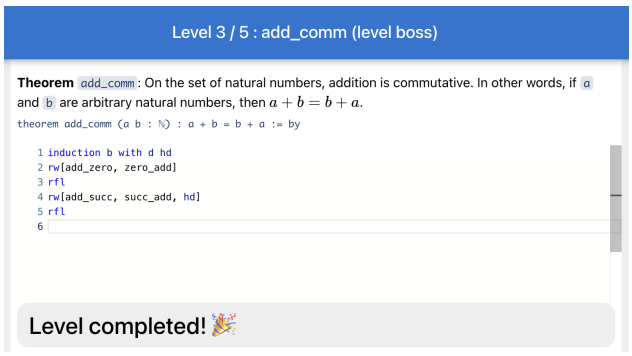
Level 1:



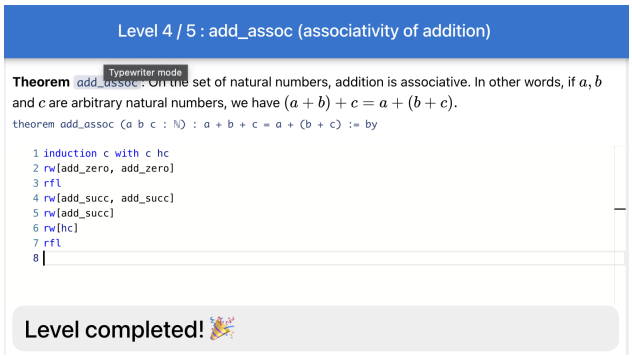
Level 2:



Level 3:

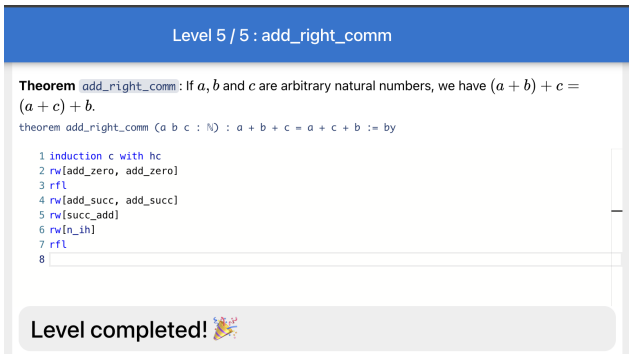


Level 4:



Using induction on c we can initially create an easier medium to use reflexivity to solve for a+b. Then solving the other side we just use the mathematical definiton of a successor function, until we can use the induction again to get the equation to the point where we can use reflexivity to prove it. This is a clear example of mathematical prrofs by induction.

Level 5:



Once again this is proof by mathematical induction similar to the previous one. This time we add the zeroes and then use reflexivity for the first part of the proof. Then to prove the second part we use the successor function until bringing back the induction we used like the previous question. Then to complete the proof we use reflexivity again.

-

Discord Question: Since recursion has so many benefits and also breaks down the complexity of problems, why aren't we taught to use it as our primary method? In other words, why isn't it the first method of problem solving we're taught?

2.3 ...

...

3 Lessons from the Assignments

(Delete and Replace): Write three pages about your individual contributions to the project.

On 3 pages you describe lessons you learned from the project. Be as technical and detailed as possible. Particularly valuable are *interesting* examples where you connect concrete technical details with *interesting* general observations or where the theory discussed in the lectures helped with the design or implementation of the project.

Write this section during the semester. This is approximately a quarter of apage per week and the material should come from the work you do anyway. Just keep your eyes open for interesting lessons.

Make sure that you use LaTeX to structure your writing (eg by using subsections).

4 Conclusion

(Delete and Replace): (approx 400 words) A critical reflection on the content of the course. Step back from the technical details. How does the course fit into the wider world of software engineering? What did you find most interesting or useful? What improvements would you suggest?

References

[BLA] Author, Title, Publisher, Year.