How Students Progress Through Functional Programming Assignments

Jerry Nyoike

McGill University

COMEPLS

Tuesday 22, 2023

Introduction

Chuqin Geng, Wenwen Xu, Yingjie Xu, Brigitte Pientka, and Xujie Si. Identifying Different Student Clusters in Functional Programming Assignments: From Quick Learners to Struggling Students. (SIGCSE 2023)

- Grade is usually considered a key measure of how well a student is doing.
- Can be misleading - especially with access to autograders.
- By considering other features, like number of static errors, authors identify 4 clusters of students (quick-learning, hardworking, satisficing and struggling).
- Authors analyze how work habits, range of errors and ability to fix errors impact the different student clusters.
- Study provides a nuanced picture of student behaviours.

Introduction

Will Crichton and Shriram Krishnamurthi. Profiling Programming Language Learning. (OOPSLA 2024)

- Gather data to understand what makes language learning difficult.
- Use the data to improve language learning.
- Modify The Rust Programming Language book to include learner profiles and quizzes.
- Provide hints to poorly performed quesitons and check if it improves student performance.
- Most readers drop off after encountering difficult language concepts (Rust Ownership).
- Statistics on questions help instructors identify questions that are too difficult.

Introduction

Chris Kerslake. Stump-the-Teacher: Using Student-generated Examples during Explicit Debugging Instruction (SIGCSE 2024)

- Debugging instruction - often a just-in-time support for specific problems
- 2 classroom activities to introduce explicit debugging instruction:
  1. Students purposefully introduce bugs to working code and teacher shows how to fix (stump the teacher).
  2. Students compare how they would have approached the same scenario.
- Bugs are mostly syntax related.
- Students pick up debugging skills faster.

Objectives

Can we:

- split student submissions by question and analyse the questions separately?
- establish if students spend more time maximizing their grade or fixing type errors?
- establish what question a student is working on when they make a submission?
- establish when students switch between questions?
- establish when students get stuck and if they get stuck on similar type/logical errors?

Experiment Setup
Data

- Source - Introductory Functional Programming course at McGill University.
- OCaml code from student submission log data: LearnOCaml grade events, compile events and eval events.
- Chuqin et. al. groups students in dataset into 4 categories quick learning, hardworking, satisficing and struggling.
- Our work builds on this prior classification and does a **question-by-question** analysis on the different categories.

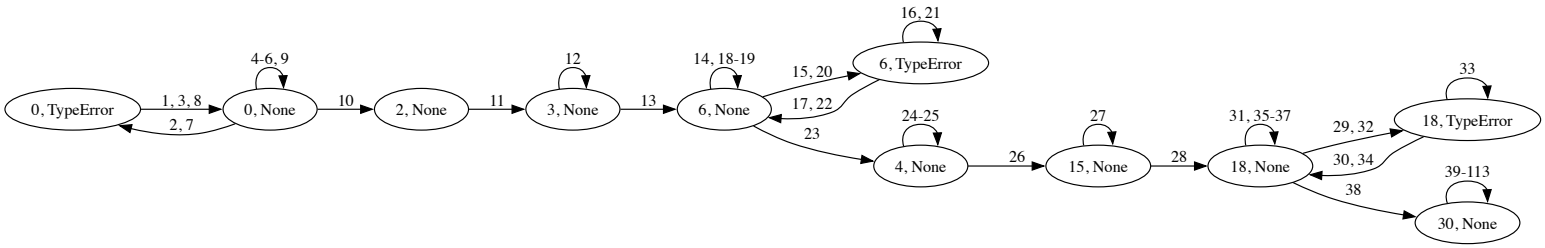
Experiment Setup
Data Preprocessing - Question Split

- Split student submissions by question using OCaml compiler libs.
- Leave out tests in the question split - studies show students tend not to use test driven development.
- Grade questions individually and record type errors, grade, submission time and code included in the submission.
- Using data from the grading we build state machines which we perform analysis on.

Experiment Setup
State Machines

- Combine eval, compile and grade events chronologically.
- Only take unique events into consideration: use unix diff to check if the code is the same between 2 submissions.
- Type error states take the same grade as the last well typed state before the error occurs.
- Each node has the grade and whether the code is well typed or ill-typed.

State Machine Example



Ratio	Entire State Machine	First 3/4	Last 1/4
WT/IT	2.66	4	2
LE/IT	2.3	3	1

Figure: Comparison of different ratios

WT - Well Typed (includes 100% grade) IT - Ill Typed

LE - Logical Error (doesn't include 100% grade)

Conclusion

Through further analysis we aim to test more hypotheses:

- Students struggle with similar type errors.
- Students struggle with type errors throughout the development of their solution to homework problems.

THANK YOU!