UC San Diego
Jacobs School of Engineering

Type Inference from Test Cases

Jiani Huang, Soomin Lee, Yijun Zhang
UCSD CSE Early Research Scholars Program (ERSP)

UCSDCSE
Computer Science and Engineering

Problem Statement

It is well-known in community that functional programming languages are challenging, but less is understood about why. Our goal is to understand how beginners learn OCaml and propose a way to make their learning process more efficient

Background

Syntax difference - Java vs. OCaml:

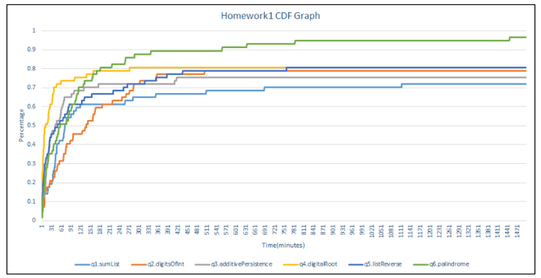
```
double average (double a, double b)
{
  return (a + b) / 2;
}
```

```
let average a b =
    (a + .b) /. 2.0;;
```

In order to track students' progress, we collected data from homework assignments from CSE 130, which included snapshots of their entire file every thirty seconds. Collected as JSON files, they included the following:

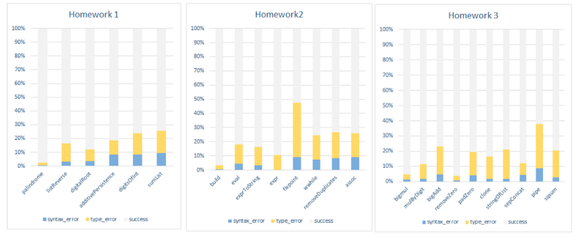
- Unix timestamp
- Body of file
- Section of code that was send to compiler
- Output error message, if any

Initial data analysis was done to see how much time the students spend on each homework problem to find out which concepts they struggle with the most.



Methods

From our collected data, we were able to find out that type error was the main issue for the beginners:



After looking at type errors, in order to see how the student's code changes over time, we tracked the progress of each function, and labeled them as bad-fix pairs:

```
fix:
let rec sumList xs = match xs with
| [] -> 0
| x::xs' -> x + sumList xs'

bad:
let rec sumList xs = match xs with
| [] -> []
| (x::xs') -> x sumList xs'
```

Then we proceeded to see if the actual fixed location matches the error location that compiler outputs and found that the error messages are largely misleading.

From top-level, typing an OCaml function generates its type annotation. It prints out what it thinks are the types of declared function. The general format is:

```
f : arg1 -> arg2 -> ... -> argn -> return_type
```

For example,

```
# let average a b =
    (a +. b) /. 2.0;;
val average : float -> float -> float = <fun>
```

And we used this information to annotate each function and its fixed version, to see if this would lead to OCaml compiler outputting more accurate error location:

```
annotated:
let rec sumList : int list -> int = fun  xs  ->  match xs with
| [] -> []
| (x::xs') -> x+1 sumList xs'

annotated_fix:
let rec sumList : int list -> int = fun  xs  ->  match xs with
| [] -> 0
| x::xs' -> x + sumList xs'
```

Results

Discussion

Acknowledgements

We would like to thank