S1.1: A Whirlwind Tour of OCaml

CSci 2041:

Advanced Programming Principles

University of Minnesota,
Prof. Van Wyk,
Spring 2022

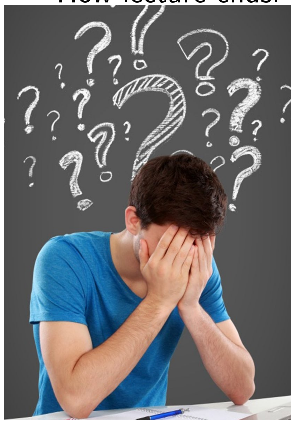
After the principles...

- We've said that many of the principles in which we are interested are more directly elucidated in a language like OCaml.
- So we should learn a bit of OCaml before we do much else.
- But we'd also like to see, today, some parts of OCaml that make it appropriate for this class, and in my opinion, general use.

A whirlwind tour! We'll go fast!

How lecture starts.

How lecture ends.



Goal for today: be able to read and understand most things

Goal for Monday and onwards: be able to read, understand, and write OCaml.

First, some simple stuff, nothing really “new”

- We’ll write three functions: `square`, `sumTo`, and `fib`.
- We’ll develop these in class, so be attentive to that discussion.
- The programs (like all that we’ll develop this term) will end up in the ‘Sample-Programs‘ directory, under ‘Course-Resources‘ in the public repository.
- These three will be written in `whirlwind_basic.ml`.

Next, some of the interesting bits

- symbolic (inductive) data, and recursive functions over it
- parametric polymorphism
- higher order functional programming
  - functions that take functions as arguments
  - function literals
  - functions to “map over” and “fold up” data

Support for “good software”

- On Wednesday we discussed principles in support of
  - correctness,
  - efficiency,
  - re-usability, and
  - transparency

  in software.
- Today we begin to see these themes play out.

Symbolic data

- Consider a binary tree with the values 2, 3, 4, 5, and 6.
- We might view it as follows:

```
    3
   / \
  2   5
     / \
    4   6
```

- We might write functions to sum up the values, determine if a value is in the tree, or create a new tree with an additional element.

Symbolic data in OCaml

- How do we
  - define a tree data structure?
  - construct new trees?
  - inspect and deconstruct trees?
  - write recursive functions over them?
- This code will be written in `whirlwind_tree.ml`.
- After class, download this file and experiment a bit.

Parametric Polymorphism, Higher Order Functions

- What about a tree of strings, or key/value pairs?
- Functions like `sum` may no longer be relevant, but `insert` and `elem` might be.
- How can we reuse a version of `elem` for trees of different types?
- Here we see some higher-order functional programming
  - functions that take functions as arguments
  - writing function literals
- Can we map functions over trees? Can we “fold up” trees?
- This code will be written in `whirlwind_parametric_tree.ml`.

Sequences or lists can be structured data too

- Traditionally arrays are "flat", they don't have the "hierarchical structure" we saw in trees.
- Lists in OCaml are not flat - they are trees that slant to one side or have only one child.
- For example, the list `1, 2, 3, 4, 5` can be seen as

  ```
  1
   \
    2
     \
      3
       \
        4
         \
          5
  ```

  Each node has only 1 child.
- This code will be written in `whirlwind_list.ml`.

Some reading

Some reading, in Functional Programming in OCaml

- Chapter 1, 2, and 3.1

You do not need to digest every detail, but be aware of main points to use it as a reference later.

We went quite fast today - we'll take it from the beginning on Monday so that you can write the kinds of programs instead of just read them.