CMSC330: The Expression Problem

Chris Kauffman

Last Updated:
Tue Nov 28 12:55:51 AM EST 2023

The Expression Problem (Extensibility Problem)

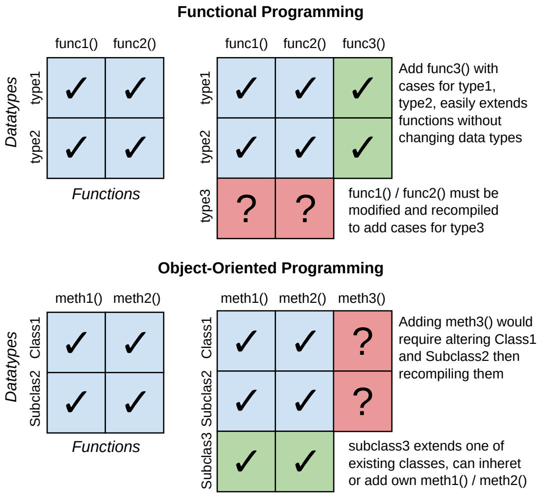
Q: How well can a programming language do these two tasks

(1) Extend Functions

Add a function that works on existing data types without modifying those datatypes

(2) Extend Types

Add a datatype that works with existing functions without changing those functions



Functional Programming

Datatypes

	func1()	func2()
type1	✓	✓
type2	✓	✓

Functions

	func1()	func2()	func3()
type1	✓	✓	✓
type2	✓	✓	✓
type3	?	?	

Add func3() with cases for type1, type2, easily extends functions without changing data types

func1() / func2() must be modified and recompiled to add cases for type3

Object-Oriented Programming

Datatypes

	meth1()	meth2()
Class1	✓	✓
Subclas2	✓	✓

Functions

	meth1()	meth2()	meth3()
Class1	✓	✓	?
Subclas2	✓	✓	?
Subclas3	✓	✓	

Adding meth3() would require altering Class1 and Subclass2 then recompiling them

subclass3 extends one of existing classes, can inheret or add own meth1() / meth2()

A: Traditional Statically Typed Functional and OO Languages favor one or the other task and suffer for the other

Expression Problem in Statically Typed Languages

- Java, OCaml suffer classic symptoms of the Expression Problem
- Haskell's Type Classes partially solve the Expression Problem[1]
- Rust DOES NOT fully solve the expression problem as it forbids adding `impl` for datatypes outside of the crate in which they are defined (see `extend_string_fail.rs` for an example)
- Likely there are other approaches but the absence of widely known solutions means this may be a limitation of statically typed system

It feels like if Rust lifted the `impl`-within-crate restriction they'd have a full solution but they must have reasons for it...

[1] The inspiration for the grid-based diagram comes from Eli Bendersky's Post about the Expression Problem which provides additional code and detail

Expression Problem in Dynamically Typed Languages

- Most dynamic languages dodge the Expression Problem as data is open, no compiler to satisfy, allow for dynamic behavior
- Example: Python “Monkey Patching” allows runtime alteration of functions withing classes, addition of new functions, etc.
- **Julia** is Dynamically typed but has many properties similar to Statically Typed languages, features Multiple Dispatch to solve the Expression Problem
- **Clojure** is a dynamically typed language but provides 2 distinct solutions to the Expression Problem: Multimethods and Protocols