Generating Julia:
Finding Commonalities between OO and Procedural Languages

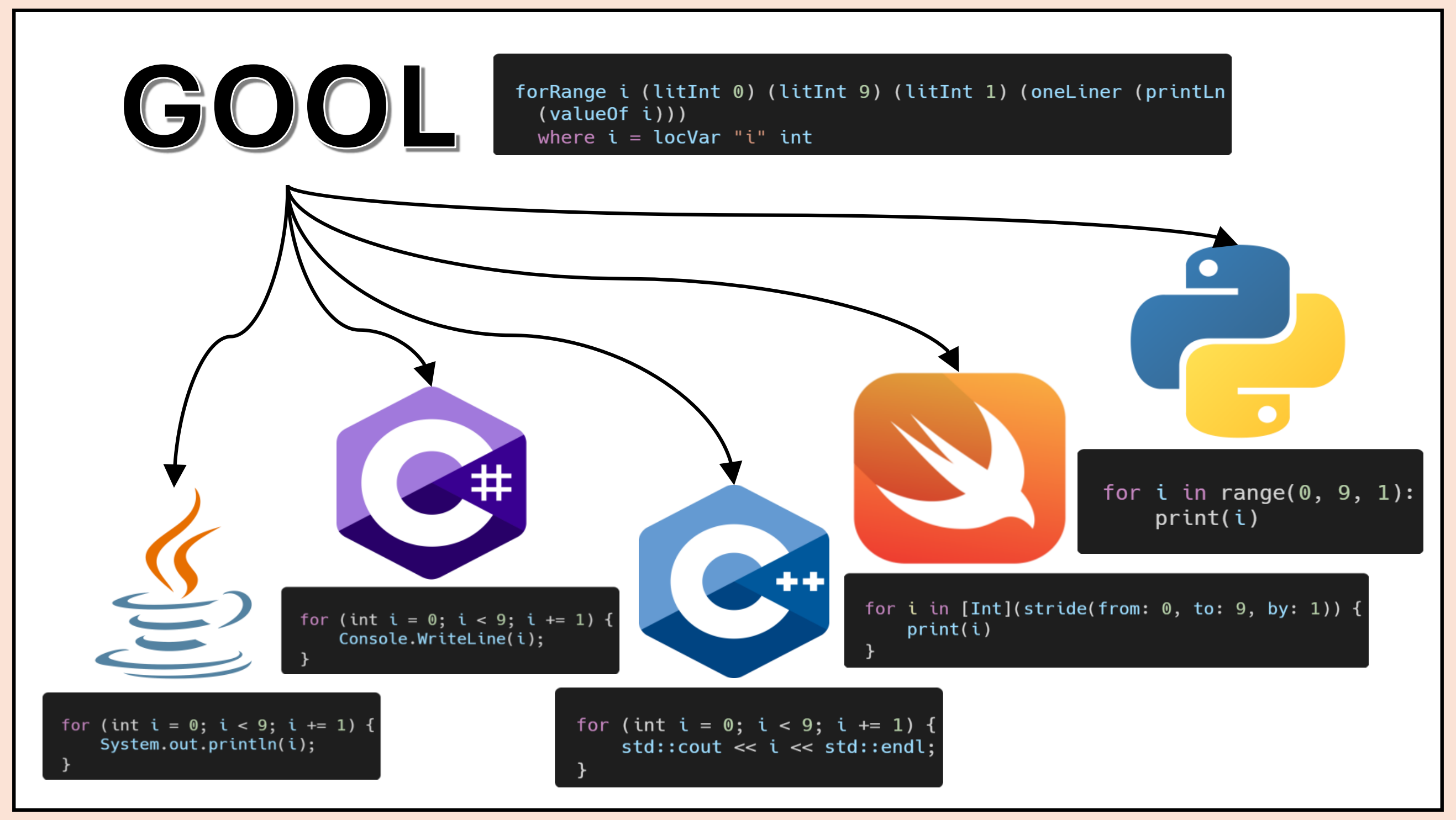
Brandon Bosman, Dr. Jacques Carette, Dr. Spencer Smith
Department of Computer and Software, McMaster University

McMaster University | Computing & Software

Introduction

What is GOOL?

GOOL is our Generic Object-Oriented Language, which we use to generate code in a variety of OO languages. It consists of a series of Haskell typeclasses which together represent an abstract OO language. At the start of the summer, we had 5 renderers from GOOL to different languages.



```
forRange i (litInt 0) (litInt 9) (litInt 1) (oneLiner (printLn
  (valueOf i)))
  where i = locVar "i" int
```

```
for i in range(0, 9, 1):
    print(i)
```

```
for i in [Int](stride(from: 0, to: 9, by: 1)) {
    print(i)
}
```

```
for (int i = 0; i < 9; i += 1) {
  Console.WriteLine(i);
}
```

```
for (int i = 0; i < 9; i += 1) {
    std::cout << i << std::endl;
}
```

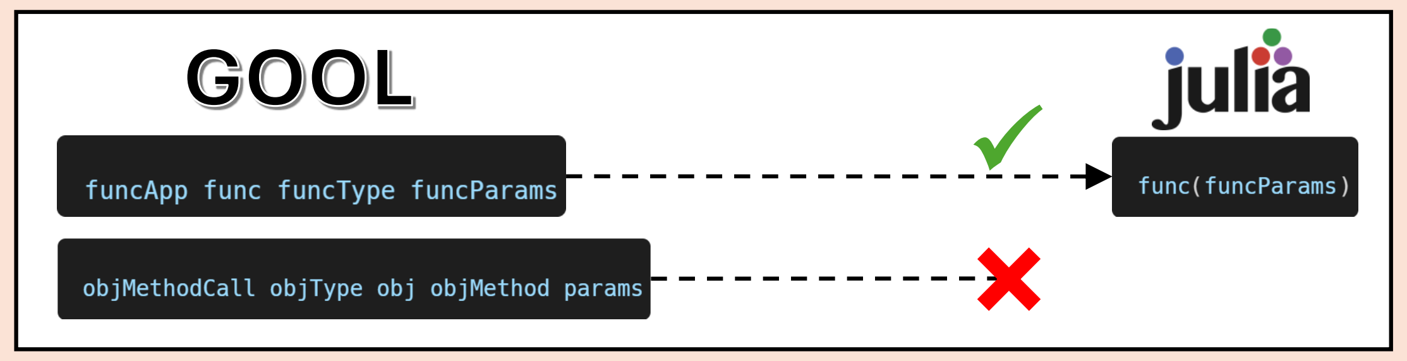
```
for (int i = 0; i < 9; i += 1) {
    System.out.println(i);
}
```

The Task

We want to be able to generate Julia code from GOOL. This will align well with GOOL's primary use case – the Drasil project, which is well-suited towards Scientific Computing.

Problem: Julia is not OO

Julia is not OO, so not everything we can express in GOOL can be expressed in Julia.



```
funcApp func funcType funcParams
```

```
func(funcParams)
```

```
objMethodCall objType obj objMethod params
```

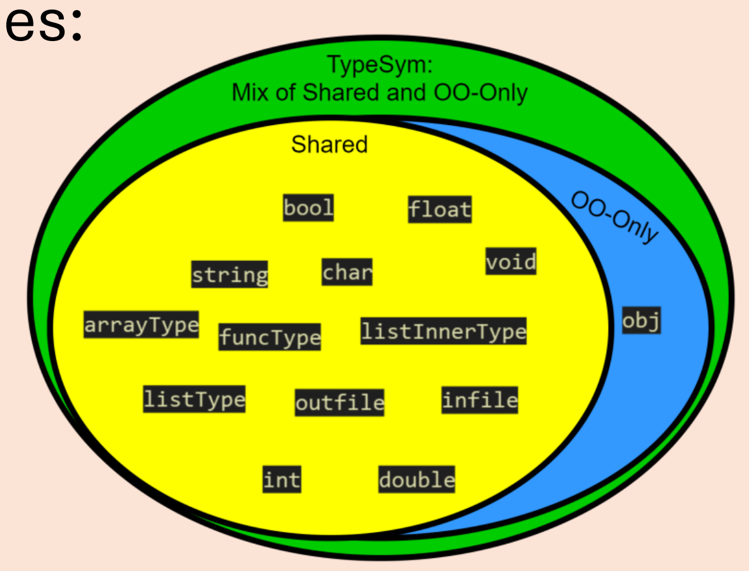
Solution: A New Generic Language!

Since GOOL is too flexible, we need a new set of typeclasses to express what can be expressed in procedural languages. We created GProc, the GOOL of procedural programs.

Designing GProc

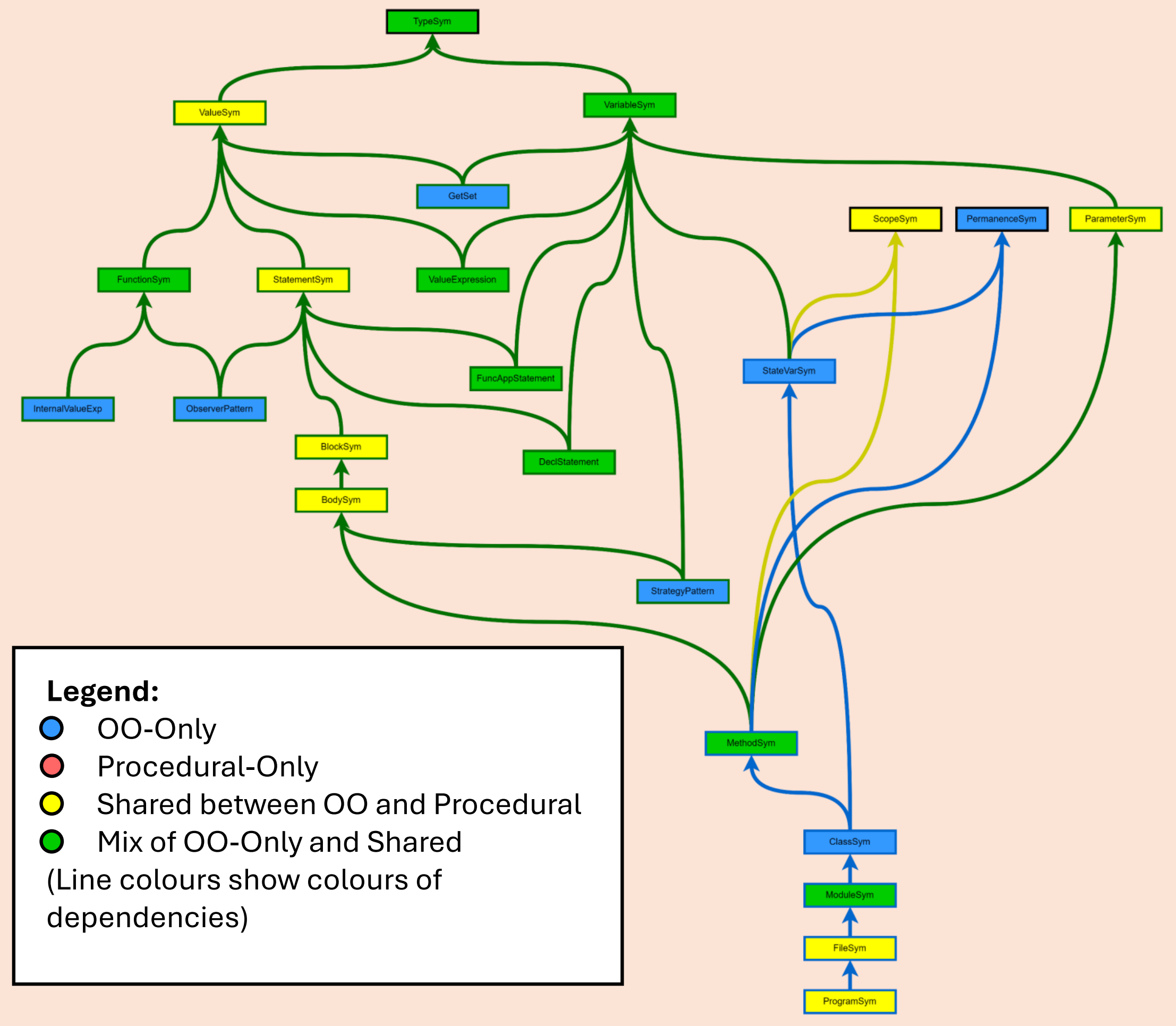
Goal: minimize code duplication between GOOL and GProc.

- Since most of GOOL is the same as most of GProc, we can reuse many of the typeclasses, as well as their implementations in different renderers.
- Problem: many typeclasses are a mix of elements that can be shared and things that are OO-Only. E.g., types:



- Solution: Split up Mixed typeclasses into Shared and OO-Only components.
- OO typeclasses can inherit from Shared typeclasses.
- Result:

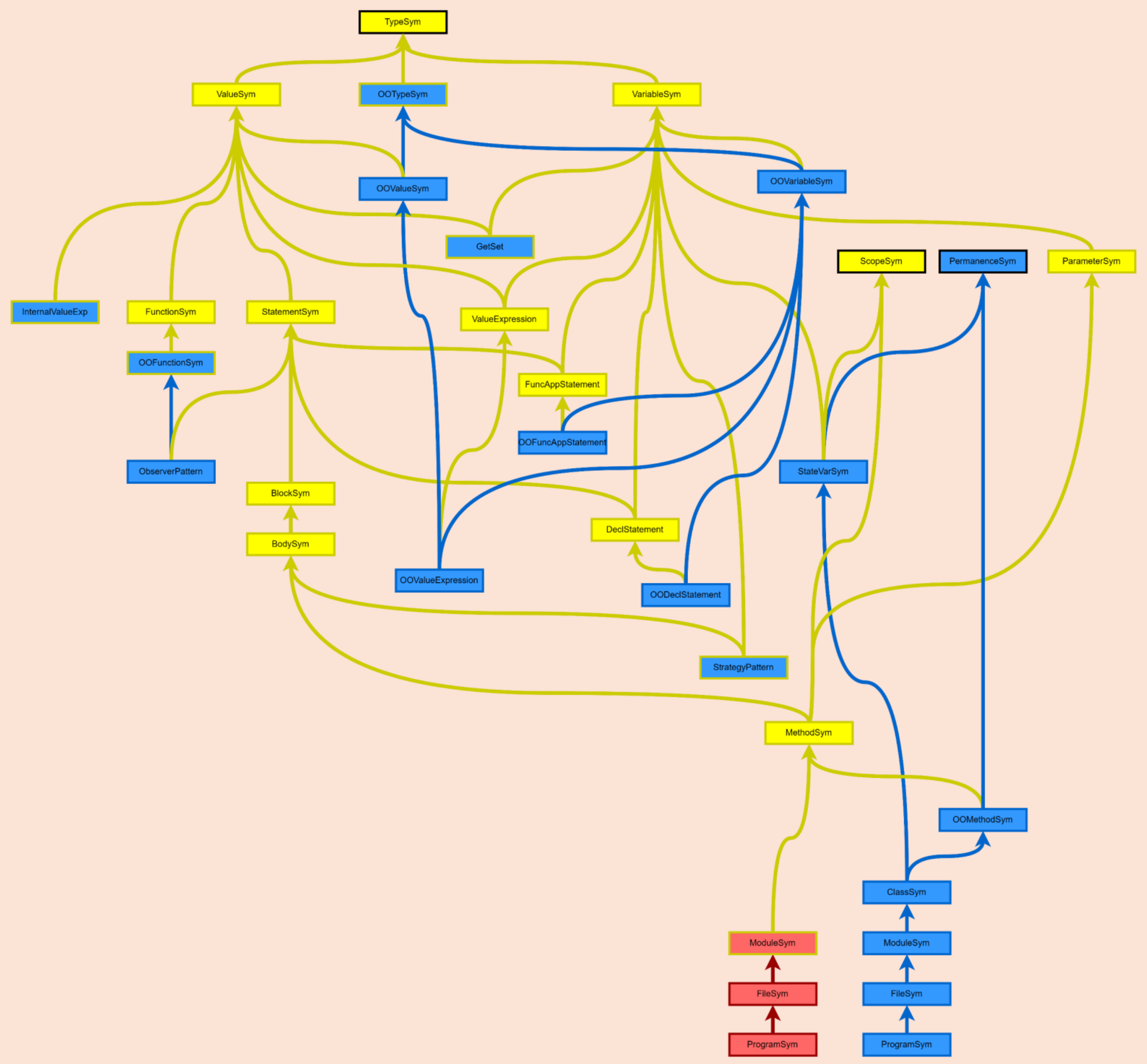
Before:



Legend:
- 🟦 OO-Only
- 🟥 Procedural-Only
- 🟨 Shared between OO and Procedural
- 🟩 Mix of OO-Only and Shared

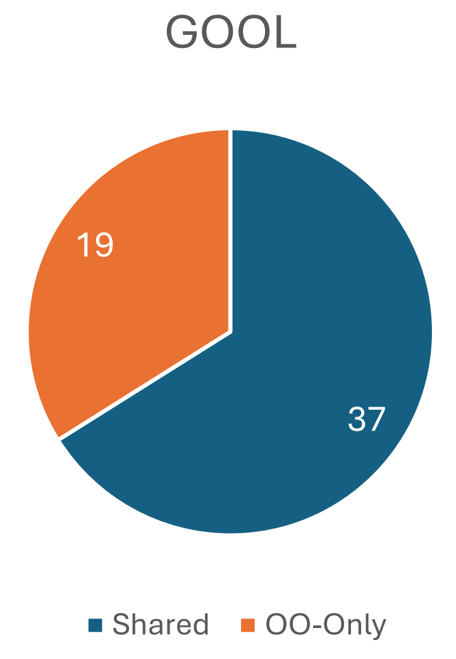
(Line colours show colours of dependencies)

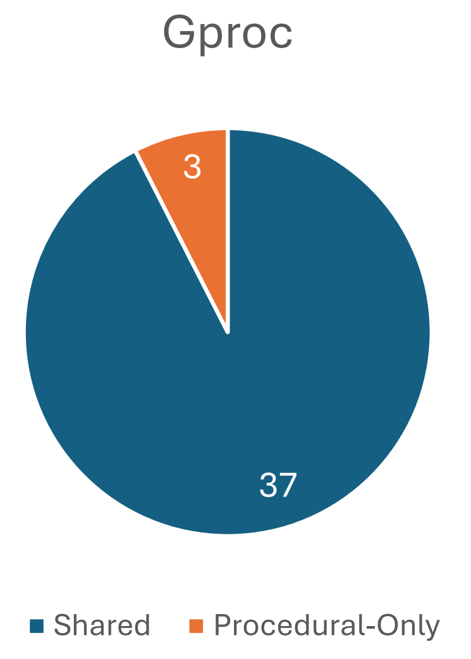
After:



Conclusions and Future Work

- OO and procedural languages have a lot in common:
  - 37 typeclasses are shared.
  - 19 typeclasses are OO-Only, making GOOL 66% shared.
  - 3 typeclasses are Procedural-Only, making GProc 93% shared.
- Next steps:
  - Add struct support to GProc. This will allow for better data-bundling techniques, which are currently lacking in GProc.


GOOL
Shared
OO-Only


Gproc
Shared
Procedural-Only