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

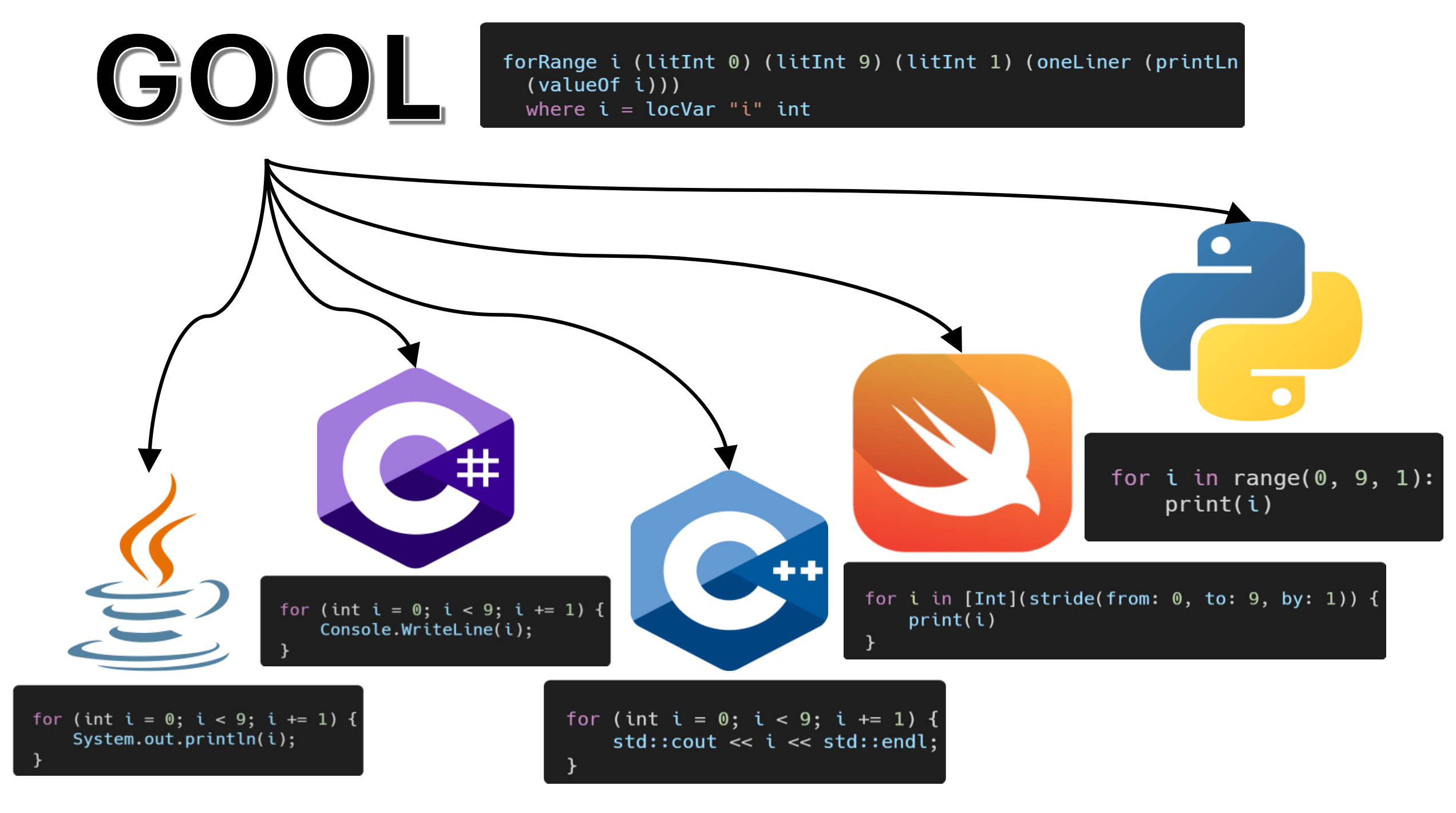
Introduction

The Task

We want to add Julia to the list of programming languages supported by GOOL. Julia is a relatively new language with a focus on scientific computing (SC) and data science [1]. Adding Julia to GOOL will align well with GOOL's primary use case – the Drasil project, which focuses on SC [2].

What is GOOL?

GOOL is our Generic Object-Oriented Language, which we use to generate code in a variety of Object-Oriented (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 renderers from GOOL to five different OO languages [3].



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

```
for (int i = 0; i < 9; i += 1) {
    System.out.println(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 i in [Int](stride(from: 0, to: 9, by: 1)) {
    print(i)
}
```

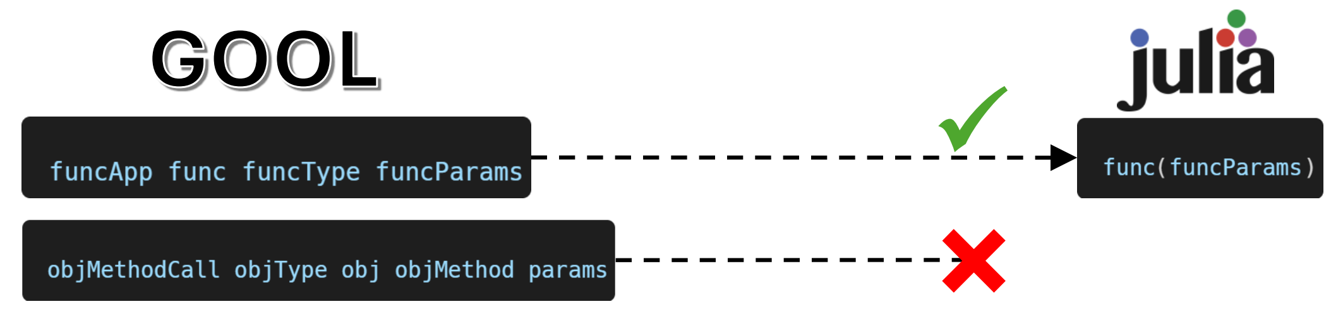
```
for i in range(0, 9, 1):
    print(i)
```

Figure 1: GOOL code and its renders in different languages.

The Problem

Julia is not OO

Because of this, not everything we can express in GOOL can be expressed in Julia.



```
funcApp func funcType funcParams
```

```
func(funcParams)
```

```
objMethodCall objType obj objMethod params
```

Figure 2: Some GOOL constructs make sense in Julia; others do not.

Solution: A New Generic Language!

Since GOOL is too OO-specific, 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 and most of GProc hold the same features, both can inherit from a 'shared' generic language with typeclasses that both can use.
- Problem: many of GOOL's typeclasses are a mix of features that can be shared and features that are OO-Only.
  For example, types:
- Solution: split up Mixed typeclasses into Shared and OO-Only components.

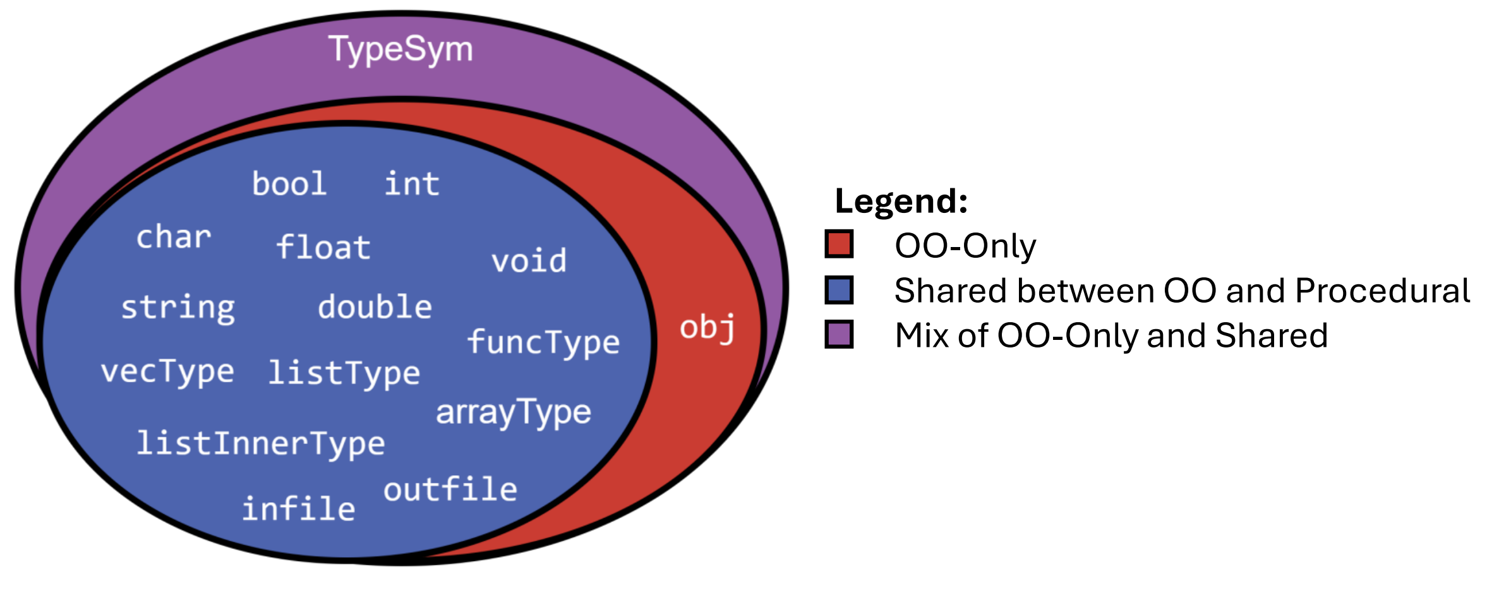


Figure 3: Most types are can be shared between different types of languages; objects are OO-only.

Result:

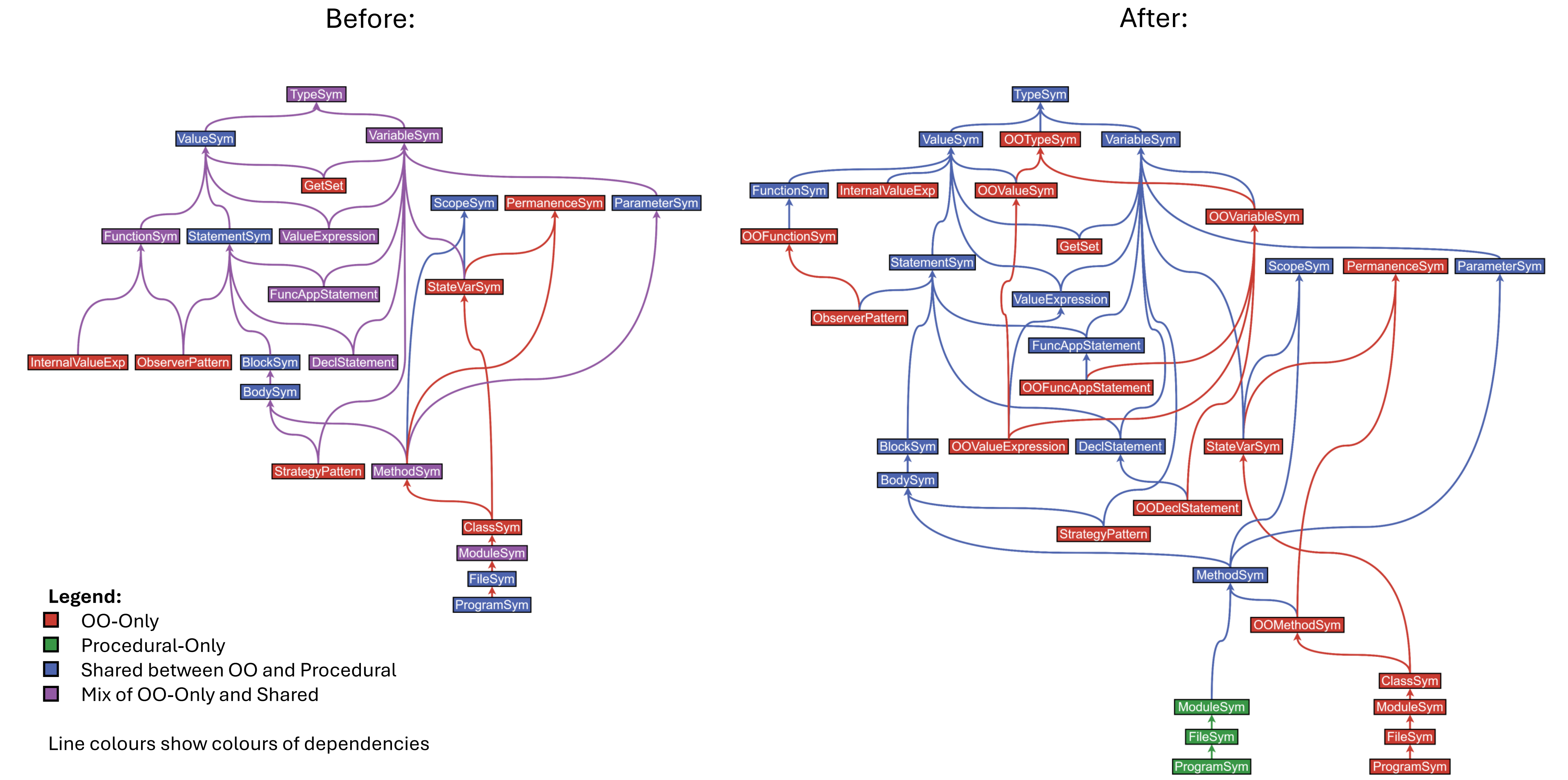


Figure 4: The types and dependencies of GOOL's central typeclasses before and after GProc was created.

With these changes in place, we were able to integrate Julia into the Drasil project and generate **545** lines of Julia code.

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.

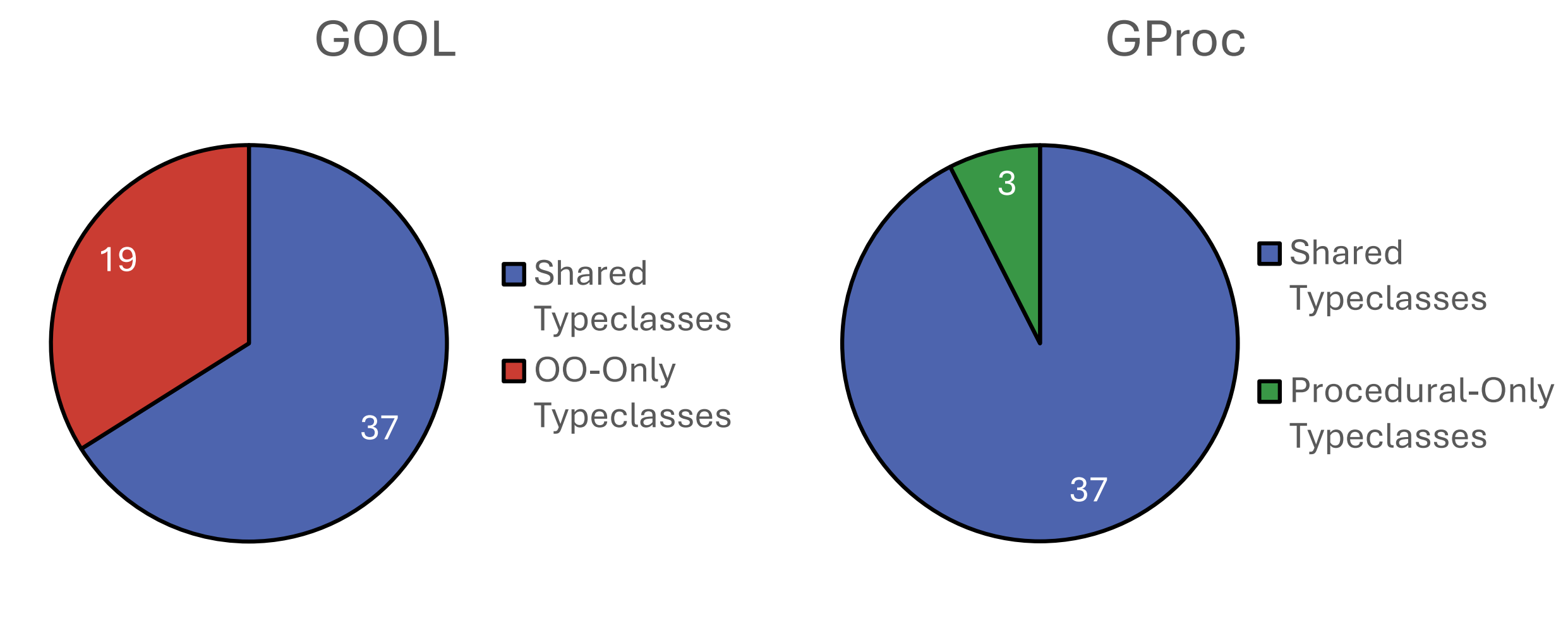


Figure 5: Number of typeclasses shared vs exclusive to GOOL and GProc.

Next steps:

- Add struct support to GProc. This will allow for better data-bundling techniques, which are currently lacking in GProc.
- Improve GProc integration within the Drasil project.

References and Acknowledgements

References:

- [1] JuliaLang.org contributors. (2024). *The Julia Programming Language*. https://julialang.org/
- [2] Carette, J., Smith, S, Balaci, J. (2023). Generating Software for Well-Understood Domains. *OASIcs, 109*. https://arxiv.org/abs/2302.00740
- [3] Carette, J., MacLachlan, B., Smith, S. (2020). GOOL: A Generic Object-Oriented Language. *PEPM '20*. https://dl.acm.org/doi/10.1145/3372884.3373159

Acknowledgements:

- I would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding this position.
- I would also like to acknowledge the support of Dr. Carette and Dr. Smith in making this opportunity available to me and being available for guidance throughout the summer.
- Lastly, I would like to acknowledge the people I worked with over the summer. Thank you to Jason Balaci and Sam Crawford for taking the time to help get us up to speed and providing day-to-day advice and assistance. Thank you to Mohammad Bilal, Noah Cardoso, and Xinlu Yan for being great coworkers and sharing lots of helpful tips that you learned over the summer.