Milestone #3 – report

PytHon

Gocaml: Go to Python compiler

Why Python? Let's be honest, the main reason Python was chosen as a target language is because Olivier is a Python developer. Although, there are also significant advantages choosing Python:

- *Its flexibility adapts well to multiple constraints from another language*
- *Often uses a same type to represent multiple Go types, making implementation simpler*
- *No need to declare variables, allowing to drop a big part of Go's code after typecheck*
- *Python is fun, honestly*

Issues with Python. In the particular case of Go, some issues do arise from choosing Python. Here are the main issues and the way we worked around them.

- *There is no multiple-statement lambda functions in Python*

  This turned out to be the most challenging issue that was faced up to this point of the code generation. There turned out to be no other way than to alias each declared lambda function and to declare it right before the first expression calling it. The aliasing is done during the weeder phase and the must be a constant look-ahead in code generation to recover lambdas before expressions. To imitate the behavior of lambdas in term of memory use, each function mimicking a lambda is set to None right after its appearance.
- *Go's switch is Python's elif*

  Python documentation clearly indicates that elif is meant to be the Pythonic equivalent of switch statements, which means Python does not have the flexibility of writing its else statement, corresponding to the default, anywhere in the code flow. Except for pushing the default case to the end, this turned out to be a minor issue.

- *Overly soft Python typecheck*

  Python turned to lose information which needed to be restored manually. For example, a Go rune type is simply a string in Python, thus Python must be reminded that a rune, say c, must be evaluated as chr(c). Luckily Python's isinstance(object, type) function allows to patch the issue.
- *Indentation*

  While minor as well since we reused the indentation coded in the pretty printer, the fact that Python forces indentation is worth mentioning

Code generation progress. The code generation is actually implemented at 80%. The main features remaining are lambda function aliasing and representation of struct as object instances. We were able to compile and run basic programs using loops, if statements, switchs and operations.

Although, the typechecker is still a work in progress as it now run but some section have been deactivated due to bugs.

Who is doing what? Most of the typechecking is still being done by Terrence who masters the code better. Code generation is mainly handled by Olivier who is more efficient in Python.