1. **Go Backend (Analysis)**: The Go backend reads the Monke source code from a file, scans it to generate tokens, and parses these tokens into an **Abstract Syntax Tree (AST)** or **Parse Tree**. This process involves **traversing the tree in pre/postorder** and possibly **multiple passes** for different analysis stages.
2. **Intermediate Representation (IR)**: The Go backend then traverses the AST or Parse Tree to generate an **IR** (Intermediate Representation). This IR is a series of Variable, Expression, and Statement objects (as defined in the .proto file) that represent the semantics of the original Monke code.
3. **gRPC or JSON Service**: The Go frontend then uses gRPC or JSON to send this IR to the Python backend. In the case of gRPC, the Go backend would call the Translate method of the MonkeTranslator service, passing the IR as an argument. In the case of JSON, Monke would serialize the IR to a JSON string and send this string to Paw over HTTP or some other protocol.
4. **Python (Code Generation)**: The Paw receives the IR and deserializes it back into Variable, Expression, and Statement objects. It then traverses these objects to generate corresponding Python code. This could involve multiple passes and different traversal orders for optimization purposes.
5. **Python Code**: Paw returns the generated Python code as a PythonCode object (as defined in the .proto file). This code can then be written to a file, executed, or further processed as needed.

## **Folder Structure**

**Go Frontend (Monke):**

1. main.go: This is your main application file where you’ll handle the command-line arguments, read the Monke source code file, and start the analysis process.
2. lexer.go: This file will contain the implementation of your lexer which will generate tokens from the Monke source code.
3. parser.go: This file will contain the implementation of your parser which will generate an Abstract Syntax Tree (AST) from the tokens.
4. ir.go: This file will contain the code to generate the Intermediate Representation (IR) from the AST.
5. client.go: This file will contain the gRPC client code that will send the IR to the Python backend.

**Python Backend (Paw):**

1. main.py: This is your main application file where you’ll start the gRPC server.
2. server.py: This file will contain the gRPC server code that will receive the IR from the Go frontend.
3. codegen.py: This file will contain the code to generate Python code from the IR.

**Shared .proto file:**

1. monke.proto: This is the Protocol Buffers file that defines the IR, Variable, Expression, Statement, and PythonCode messages and the MonkeTranslator service. This file is shared between the Go frontend and the Python backend.