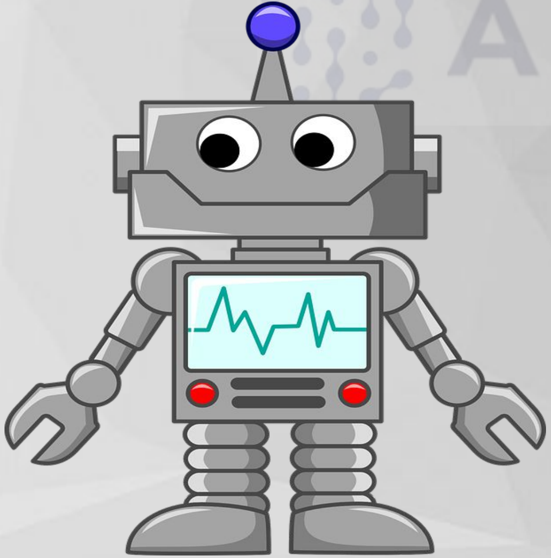
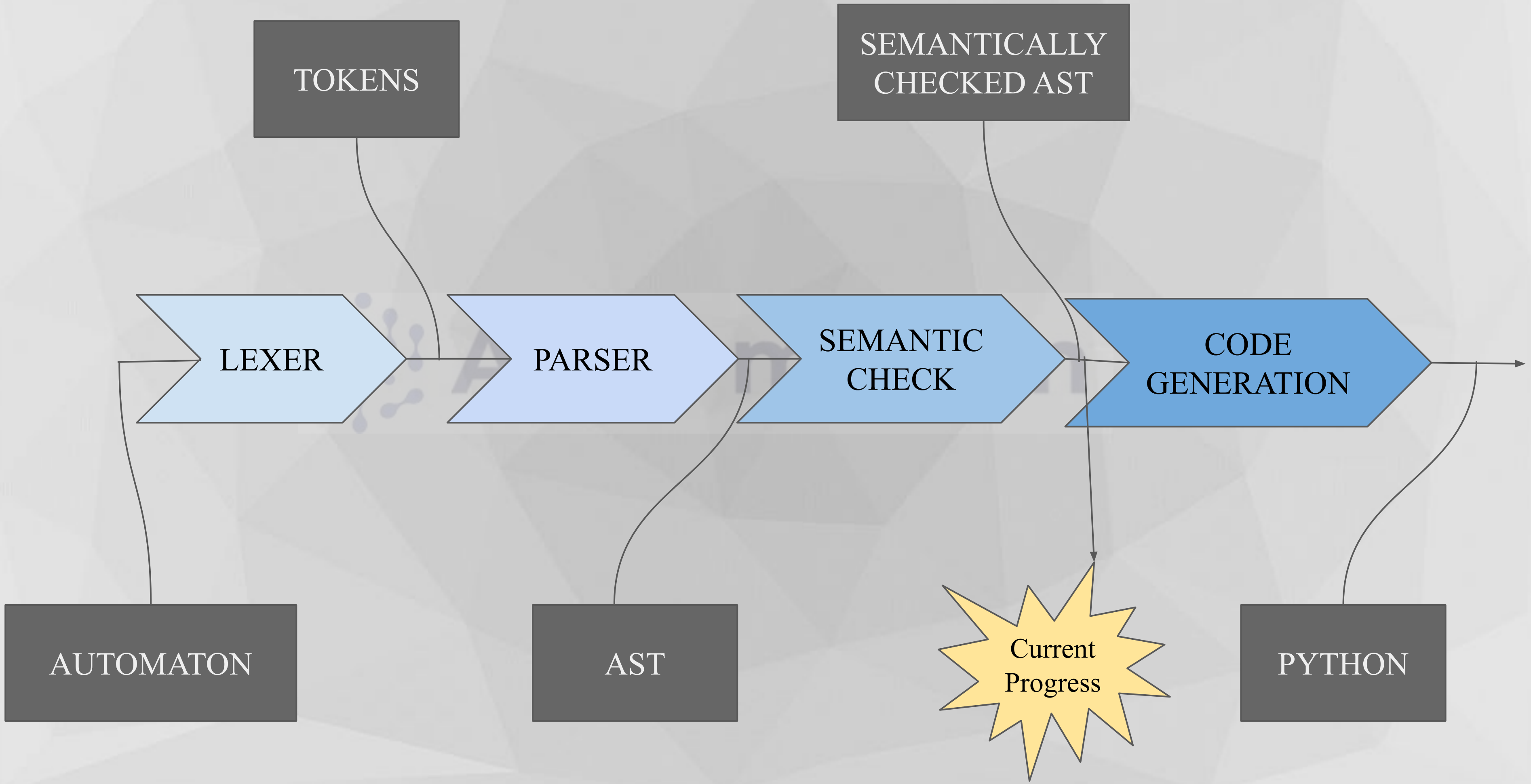
AUTOMATON



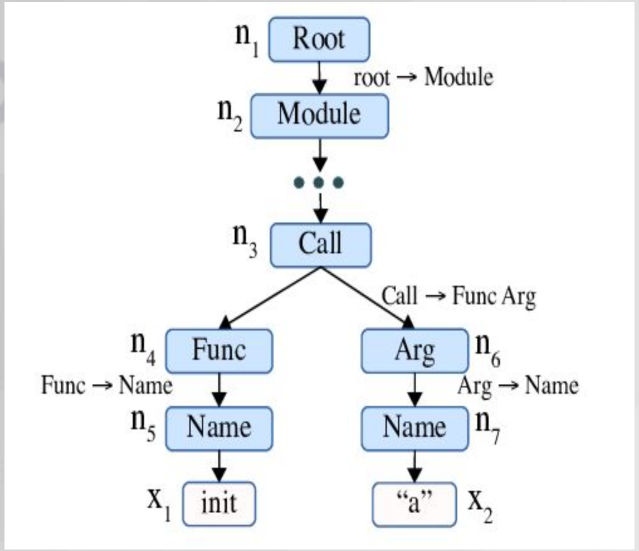
GROUP 6

V. Sathwik - CS19BTECH11022
D. Sarat Chandra Sai - CS19BTECH11003
Sushma Nalari - CS19BTECH11006
Pochetti Rashmitha - CS19BTECH11019
L. Harsha Vardhan Reddy - CS19BTECH11023
K. Lakshmi Sravya - CS19BTECH11017
K. Sri Teja - CS19BTECH11002


TOKENS
SEMANTICALLY CHECKED AST
LEXER
PARSER
SEMANTIC CHECK
CODE GENERATION
AUTOMATON
AST
Current Progress
PYTHON

ABSTRACT SYNTAX TREE

- The AST (Abstract Syntax Tree) is the tree representation of the source code and its construct.
- The elements of our programming statements are broken down into the parts and represented as a hierarchical tree.
- The compiler generates symbol tables on the AST during semantic analysis.
- During the semantic analysis, the tree is completely traversed from the root node(top) to all the nodes till the bottom.
- For example, a tree for a conditional statement has the rules for variables hanging down from the required operator.



SEMANTIC ANALYSIS

- Semantic Analysis is the process of drawing meaning from a text.
- It is the third phase of compilation process which interprets sentences so that computers can understand them.
- Ensuring the declarations and statements of a program is done in this process.
- Functions of Semantic Analysis are
  - ➢ Type Checking: Makes sure that each operator has matching operands or in other words ensures that data types are used in a way consistent with their definition.
  - ➢ Label Checking: Every program must contain labels references.
  - ➢ Flow Control Check: Keeps a track of whether the control structures are used in proper manner or not.(ex: no break statement outside a loop)
- It occurs during compile time(static semantics) and run time(dynamic semantics).

SEMANTIC ANALYZER

- Semantic Analysis helps us in maintaining the semantic correctness of the program.
- Syntax tree of the parser phase and the symbol table are used to check the consistency of the program in accordance with the language definition.
- It stores the gathered information in either syntax tree or symbol table which would be used by compiler during the intermediate-code generation.
- Errors recognized by semantic analyzer are
  - ➢ Type mismatch
  - ➢ Undeclared Variables
  - ➢ Reserved identifier misuse.
- The output of semantic phase is the annotated tree syntax.

FUNCTION OF .ml FILE

- An AST is passed to the semantic check, which performs a semantic check on it.

- It looks for a number of unusual conditions, including:
  - The existence of a DFA named main with a void return type.
  - Every DFA that isn't void actually returns something.
  - There is a start node in every DFA.
  - In a DFA, every node contains a transition statement or returns.
  - If a node lacks a return, it must include an unconditional transition statement at the very least.
  - It also verifies typical requirements for assignments, such as the existence of a variable inside a defined scope and type consistency.

- The final product is an AST that has been semantically checked.

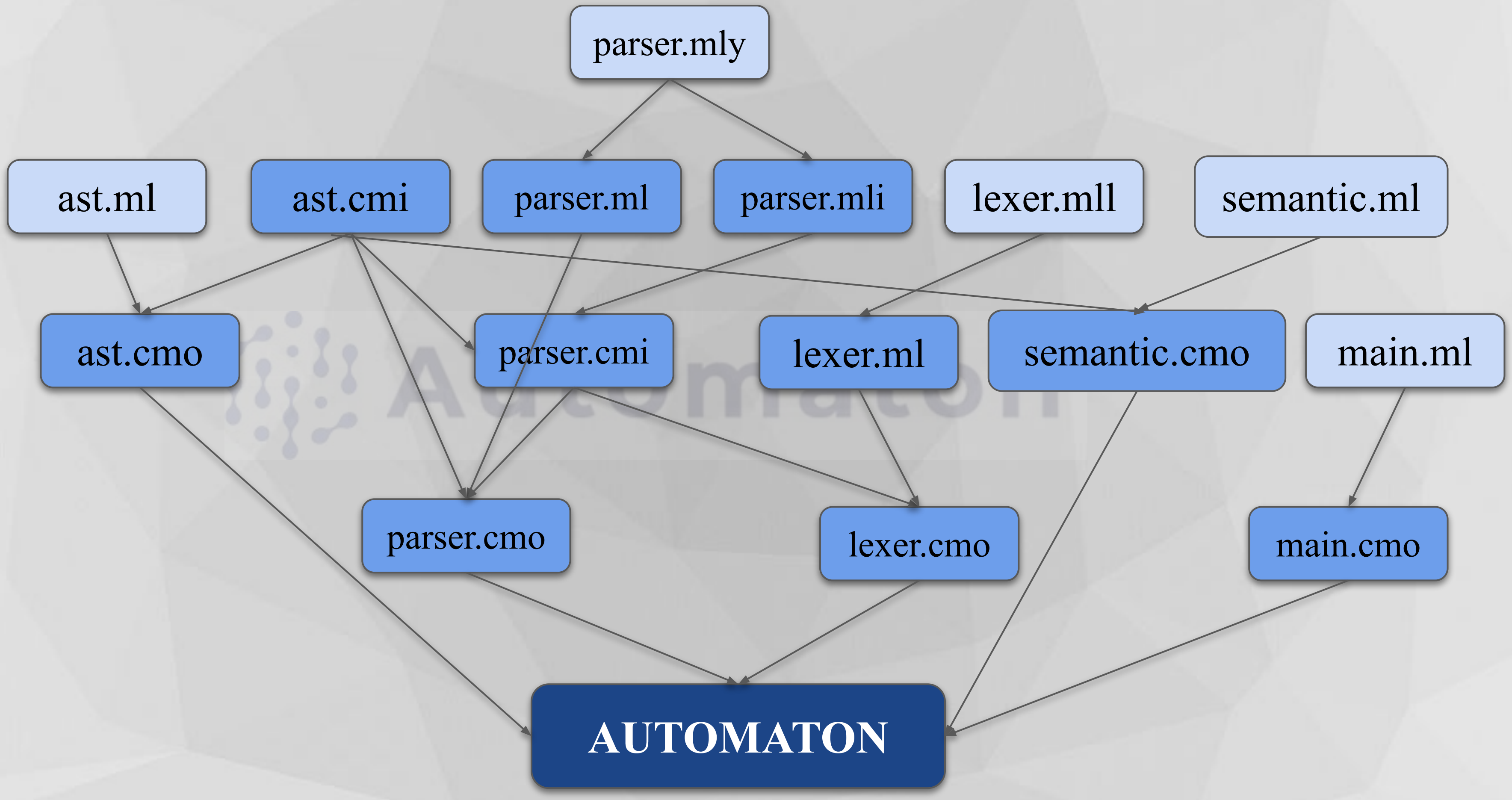
IMPLEMENTATION

- The tokens of our source code are pre-defined in “lexer.mll”. Using Ocamllex, “lexer.ml” is generated from the above file.
- Context-free grammar specification is defined in “parser.mly”. Using Ocamlyacc, “parser.mli” and “parser.ml” are generated.
- “ast.ml” file is the result of the syntax analysis phase of the compiler. The structure of our program code is represented in this file.
- Using Ocamlc command, “semantic.cmo” file is generated from “semantic.ml” and “ast.mi” files.
- All of the above which are combination of lexer,parser, ast, semantic generate object file named “semantic”, which on executing gives out the names of tokens to stdout, taking input from stdin or “input.am”.

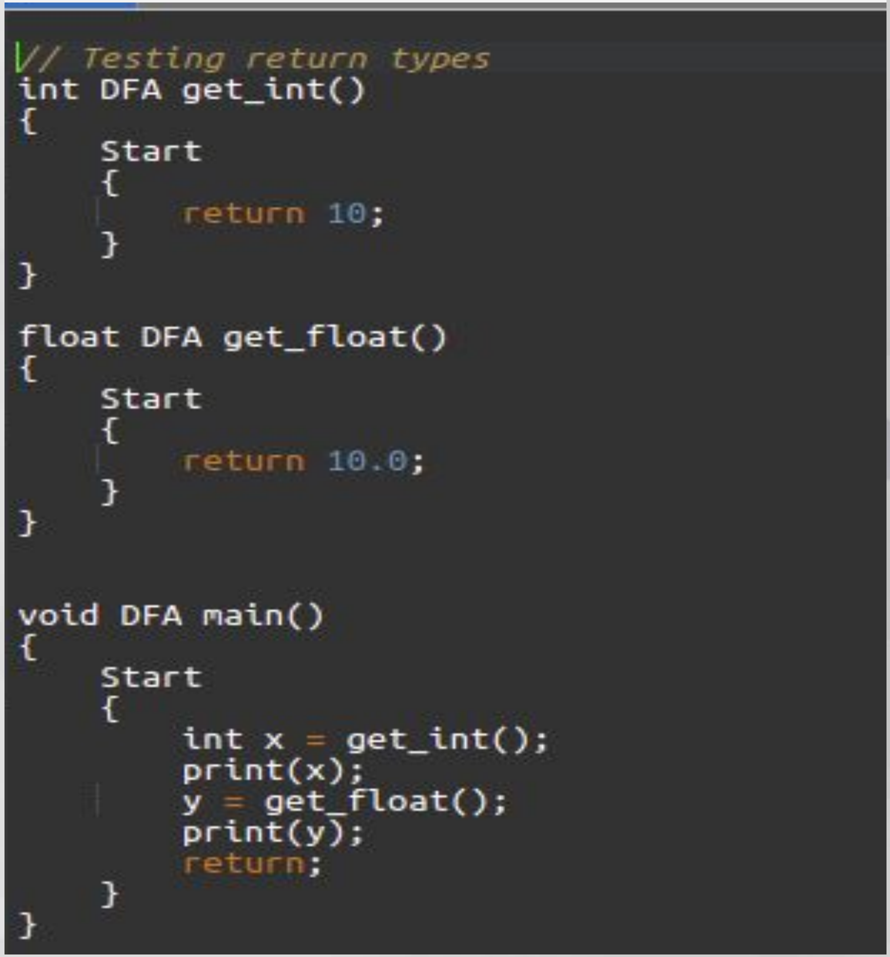
HOW TO COMPILE AND RUN

- The folder “Semantic” consists of the following files:
  - lexer.mll
  - parser.mll
  - ast.ml
  - semantic.ml
  - main.ml
  - input.am
  - makefile
- “make” command is used to run these files and generate the required output.

  (Refer to the next slide for better understanding)


parser.mly
ast.ml
ast.cmi
parser.ml
parser.mli
lexer.mll
semantic.ml
ast.cmo
parser.cmi
lexer.ml
semantic.cmo
main.ml
parser.cmo
lexer.cmo
main.cmo
AUTOMATON

Testing return types


```
// Testing return types
int DFA get_int()
{
    Start
    {
        return 10;
    }
}

float DFA get_float()
{
    Start
    {
        return 10.0;
    }
}


void DFA main()
{
    Start
    {
        int x = get_int();
        print(x);
        y = get_float();
        print(y);
        return;
    }
}
```

OUTPUT :

```
single line comments start
single line comment end
found int
found digit
found return statement
found transition
found float
found float
found return statement
found transition
found void
found int
int assigned to identifier
variable declaration
found ID
assignnig value
found ID
found return
found transition
Uncaught exception:
Semantic.Error("Uninitialized variable")
```

THANK YOU!!!