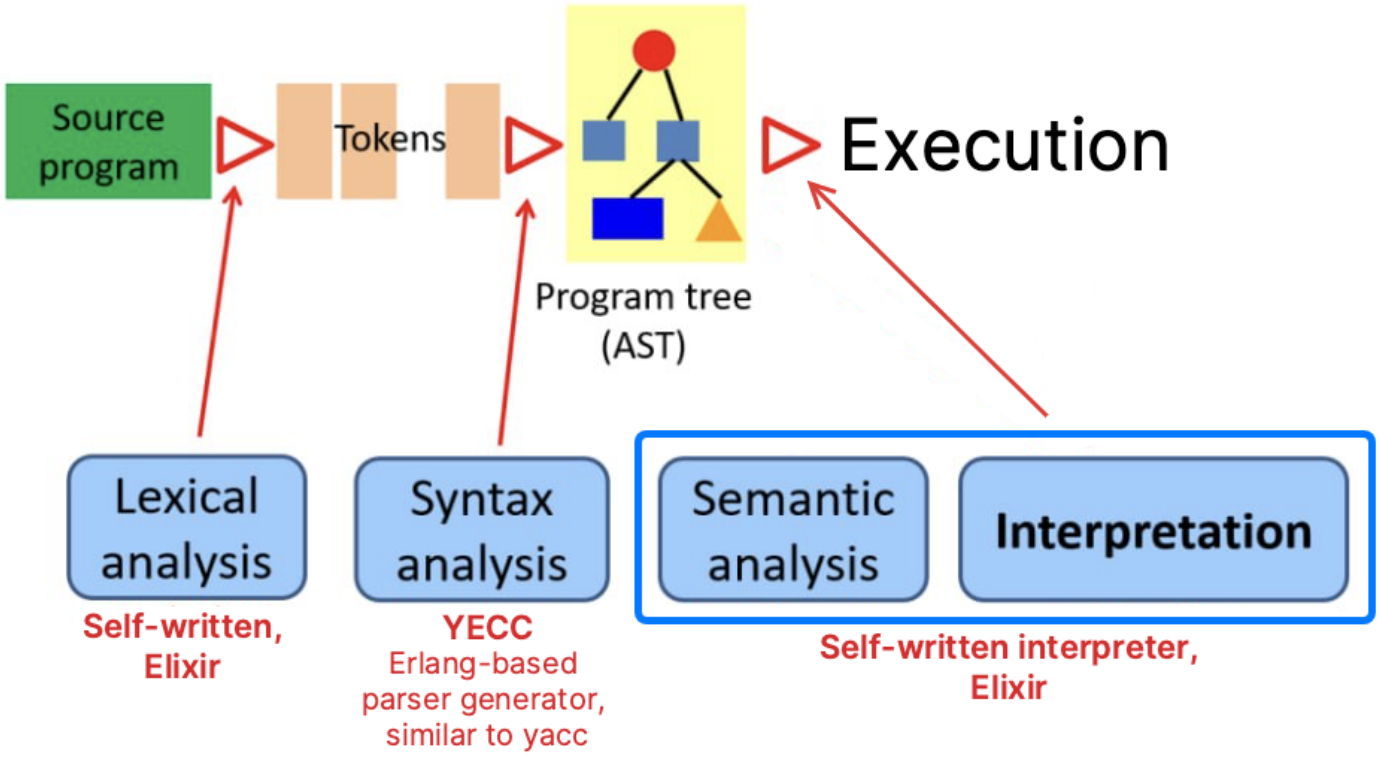
Pechenen team

Project delivery 3 & 4

Project People


Source program
Tokens
Execution
Program tree (AST)
Lexical analysis
Syntax analysis
Semantic analysis
Interpretation
Self-written, Elixir
YECC
Erlang-based parser generator, similar to yacc
Self-written interpreter, Elixir

Semantic Analyzer

1: All called/used functions and variables are previously declared/initialized.

Program:

```
1  (prog (n)
2    (ConstructAsc (cons n '()))
3  )
4
```

Output:

```
** (RuntimeError) Error in Ln 2, Col 3: ConstructAsc/1 is not defined
    (pechenen_compiler 0.1.0) lib/interpreter.ex:348: Interpreter.interpret_function_call/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:194: Interpreter.interpret_node/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:22: Interpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:22: PechenenInterpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:10: PechenenInterpreter.main/1
    (elixir 1.14.4) lib/kernel/cli.ex:131: anonymous fn/3 in Kernel.CLI.exec_fun/2
```

Semantic Analyzer

2: Number of arguments in the declaration and the function call corresponds

Program:

```
1   (func ConstructAsc () (
2     '())
3   )
4
5   (prog (n)
6     (ConstructAsc (cons n '()))
7   )
8
```

Output:

```
** (RuntimeError) Error in Ln 6, Col 3: ConstructAsc/1 is not defined
    (pechenen_compiler 0.1.0) lib/interpreter.ex:348: Interpreter.interpret_function_call/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:194: Interpreter.interpret_node/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:22: Interpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:22: PechenenInterpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:10: PechenenInterpreter.main/1
    (elixir 1.14.4) lib/kernel/cli.ex:131: anonymous fn/3 in Kernel.CLI.exec_fun/2
```

Note: the error message is the same as if the function was undefined.
This is because the expression «(ConstructAsc arg)» lookups the function «ConstructAsc» with exactly 1 argument, and it was not defined.

Semantic Analyzer

3: Type-checking - validity of assignments, correspondence of types of function arguments, correspondence of nested user-defined types

Since the language is untyped, type errors may occur only on a predefined function call.

Program:

```
(prog ()
  (head 5)
)
```

Output:

```
** (RuntimeError) Error in Ln 8, Col 3: head argument should be list, got 5
    (pechenen_compiler 0.1.0) lib/interpreter.ex:71: anonymous fn/3 in Interpreter.list_operations/0
    (pechenen_compiler 0.1.0) lib/interpreter.ex:345: Interpreter.interpret_function_call/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:194: Interpreter.interpret_node/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:22: Interpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:22: PechenenInterpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:10: PechenenInterpreter.main/1
    (elixir 1.14.4) lib/kernel/cli.ex:131: anonymous fn/3 in Kernel.CLI.exec_fun/2
```

Semantic Analyzer

3: Type-checking - validity of assignments, correspondence of types of function arguments, correspondence of nested user-defined types

A slightly more complex example:

Program:

```
(func AddOne (n)
    (plus n 1)
)

(prog ()
    (AddOne '(1 2 3))
)

```

Output:

```
** (RuntimeError) Error in Ln 2, Col 3: Both plus arguments should be numbers, got [1, 2, 3] and 1
    (pechenen_compiler 0.1.0) lib/interpreter.ex:163: anonymous fn/3 in Interpreter.arithmetic_functions/0
    (pechenen_compiler 0.1.0) lib/interpreter.ex:345: Interpreter.interpret_function_call/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:194: Interpreter.interpret_node/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:345: Interpreter.interpret_function_call/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:194: Interpreter.interpret_node/3
    (pechenen_compiler 0.1.0) lib/interpreter.ex:22: Interpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:22: PechenenInterpreter.interpret/2
    (pechenen_compiler 0.1.0) lib/pechenen_interpreter.ex:10: PechenenInterpreter.main/1
```

Code interpretation

A program that accepts one integer N and generates ascending list [0, 1, …, N]
A program is recursive

Program:

```
(func ConstructAsc (l) (
  cond (less (head l) 1)
    l
    (ConstructAsc (cons (minus (head l) 1) l)))
)

(prog (n)
  (ConstructAsc (cons n '())))
)

```

Console:

```
Andrews-MacBook-Pro:pechenen-compiler andrewsandimirov$ ./pechenen_compiler list-construction.plisp 9
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Code interpretation

A program that generates N's fibonacci number
A program is recursive

Program:

```
(func Fibonacci (n) (
  cond (less n 2)
    n
    (plus (Fibonacci (minus n 1)) (Fibonacci (minus n 2))))
)

(prog (n)
  (Fibonacci n)
)
```

Console:

```
Andrews-MacBook-Pro:pechenen-compiler andrewsandimirov$ ./pechenen_compiler test/fixture/golden.plisp 7
13
```

Team Contribution

Nikita Pozdniakov

Code for Interpreter (base, loops)

Maxim Filonov

Code for Interpreter (arithmetics, std lib)

Pavel Bakharuev

Testing, code fixes, presentation

Andrey Sandimirov

Research & code for printing lists