Metafox

Context: **CS-540 "Advanced Topics in Programming Languages Development“**

University Of Crete, Computer Science Department

Kostas Drakonakis kostasDrk@csd.uoc.gr

Anna Kokolaki kokolaki@csd.uoc.gr

Giorgos Nikitakis nikitak@csd.uoc.gr

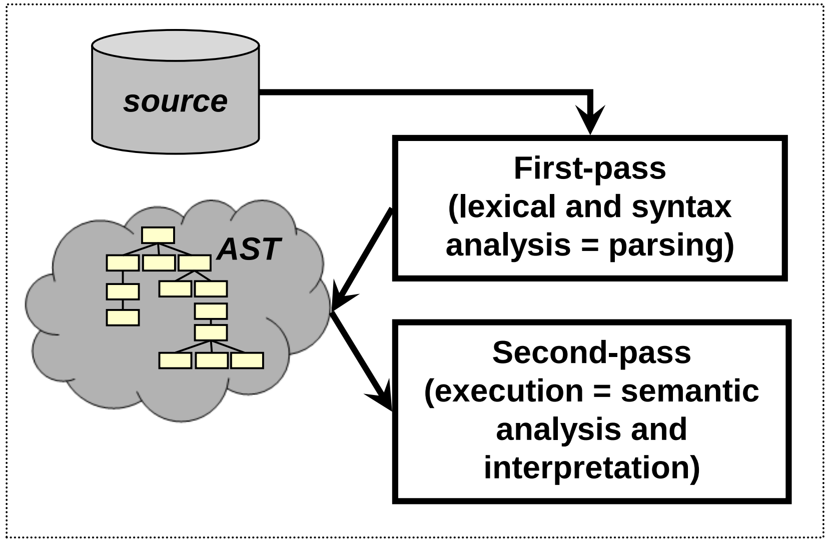
Metafox

Metafox is:

- Interpreted
- Untyped
- Statement based
- A meta language

Basic Flow

- First pass
  - Lexical Analysis
  - Syntax Analysis
  - AST is produced
- Second pass
  - ExecutionASTVisitor invoked
  - AST nodes are visited recursively (interpretation)
  - Further internal passes may occur


source
First-pass (lexical and syntax analysis = parsing)
AST
Second-pass (execution = semantic analysis and interpretation)

ASTNodes and Visitors

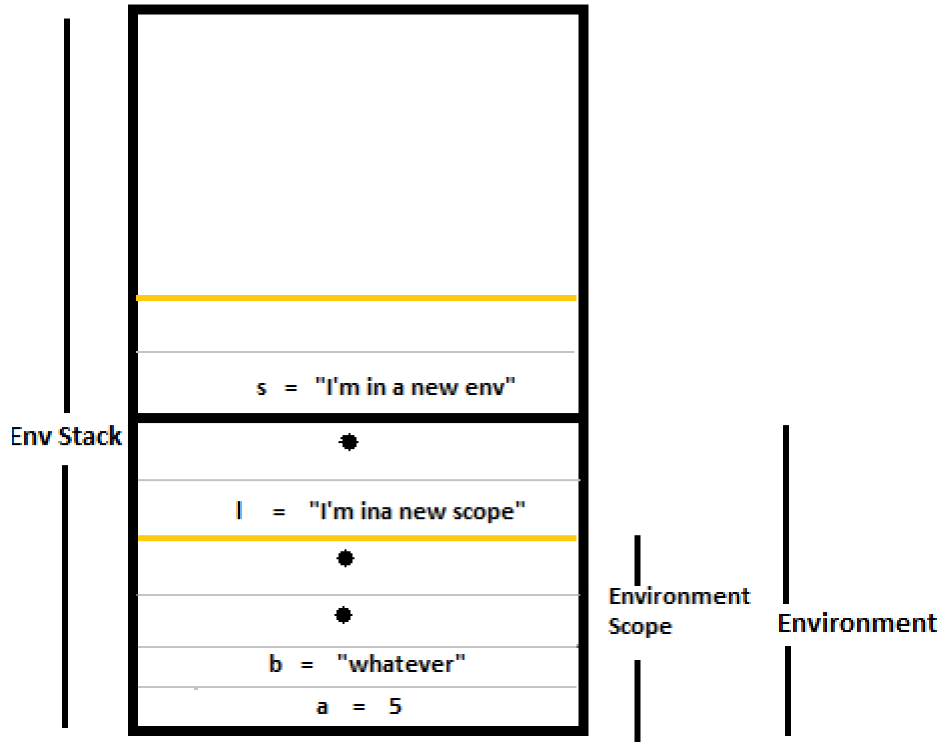
- Three types of ASTVisitors:
  - ExecutionASTVisitor
  - ToStringASTVisitor
  - IteratorASTVisitor
- AST classes include their own *accept* method, which allows for an ASTVisitor to visit them

Example ASTNodes

- IdentifierExpression
- FunctionDef
- BinaryExpression
- ObjectDefinition
- Etc…

Environment Stack

- Stack of environments, which internally store scope stacks
- *IdentifierExpressions* hold links towards their stack entries and stack entries hold links to their *IdentifierExpression* references



Built-in library functions

- General purpose library functions:
  - println
  - isNull, isInteger etc. (type checking)
  - len
  - keys
  - copy
  - push
  - Etc..

- Meta-related library functions for AST manipulation
  - iterator
  - isIfStatement, isBinaryExpression etc.
  - getExpression
  - setExpression
  - setIdentifier, setIdentifierNew
  - And more!

Meta features

- Metafox supports a set of meta operators, for additional AST manipulation

.<[expr|stmt|stmtlist]>.	MetaSyntax: Get Expression/Statement AST
.!expr	MetaExecute: Execute AST stored in expression
.~expr	MetaEscape: Assume expression carries an AST
.@”string”	MetaRun: Lift string to valid AST
.eval(expr)	MetaEval: Equivalent to .!.@”<validcode>”
.#expr	MetaToText: Transform AST to equivalent string

Demonstration

- General fox script (Queens problem from CS-340)
- Diagnostic checks and aspectual transformations
- Object factory
- Static function analysis (runtime warnings)
  - Exit paths
  - Simple dead code elimination
  - Assignment in condition
- Static function style checker
  - Function size in statements
  - Expression tokenization and complexity

Thank you for your attention.

Questions!