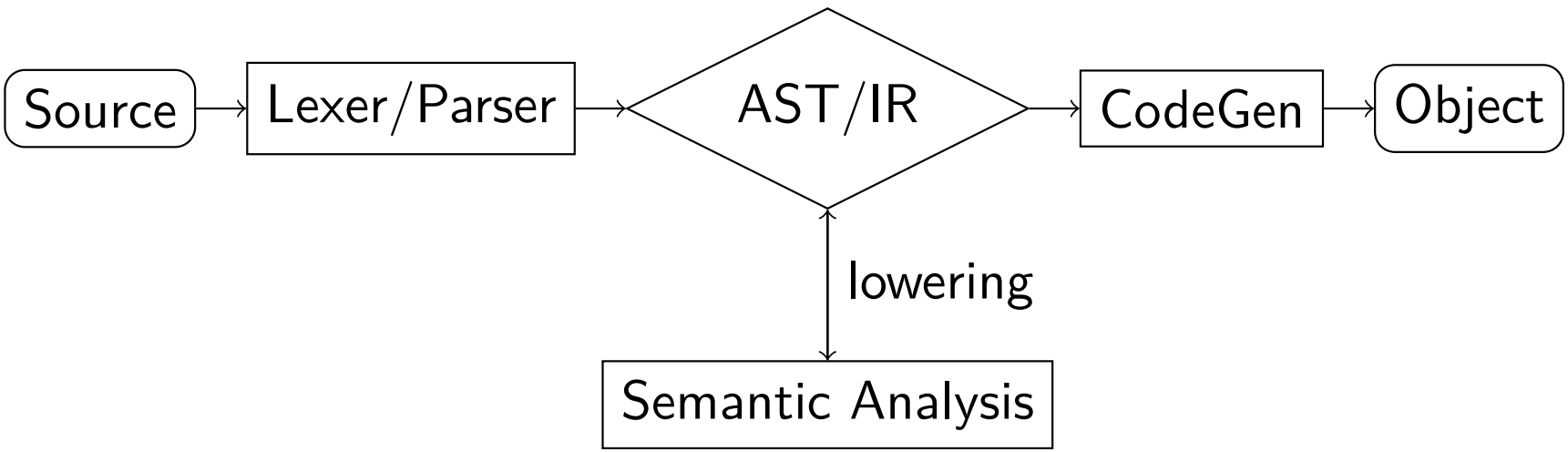
Deeplang AST Design

DIAO Zhongpu

Zhejiang University

Dec 20, 2020

Overview: Programming Language Front End



AST/IR Usage

- Syntax Analysis: diagnostic information
- Semantic Analysis: name resolution, type checking, de-sugaring
- Code Optimization/Generation: finer granularity

Typical Design: Rust

- Concrete Syntax Tree, Full Syntax Tree
- Abstract Syntax Tree
- High-level Intermediate Representation (HIR): result of semantic analysis and transformation
- Mid-level Intermediate Representation (MIR): based on CFG, flow-sensitive safety check, borrow checking
- Low-level Intermediate Representation (LIR): SSA, three-address code or sea of nodes

AST Implement

Node fields include:

- Lexer Context and Location: for diagnostic information
- Node Type: for pattern match and downcast to the right pointer
- Node-Specific Flags: literal value, type
- ... and Data: items within a block or an array

AST Implement

C++, use polymorphism for method dispatch
`class ASTNode` derives:

- `Declaration`: interface, function, class, or algebraic data, delegate
- `Statement`: to perform side efforts and control flow
- `Expression`: compute some value, possibly with side effects
- `Type`: algebraic data type, function signature

Design Patterns in AST Implement

Factory Method (Virtual Constructor)
An interface for creating objects in a superclass, allowing subclasses to alter the type of objects to be created.
`class AstNodeFactory`

Visitor Method
To add new operations without modifying existing structure.

Further Optimization

Memory Management to Speedup

Nanopass Framework

Little More: Occam's Razor

```
blockList :
    statement SEMICOLON_SYMBOL
    | statement SEMICOLON_SYMBOL blockList
    | exp SEMICOLON_SYMBOL
    | exp SEMICOLON_SYMBOL blockList
    | decl SEMICOLON_SYMBOL
    | decl SEMICOLON_SYMBOL blockList
;
⟹
blockList :
    (statement|exp|decl) (SEMICOLON_SYMBOL (statement|exp|decl))*
;
```

```
varDecl :
    LET_SYMBOL IDENTIFIER COLON_SYMBOL type
    | LET_SYMBOL IDENTIFIER COLON_SYMBOL type EQUAL_OPERATOR exp
    | LETMUT_SYMBOL IDENTIFIER COLON_SYMBOL type
    | LETMUT_SYMBOL IDENTIFIER COLON_SYMBOL type EQUAL_OPERATOR exp
;
⟹
varDecl :
    (LET_SYMBOL | LETMUT_SYMBOL) IDENTIFIER COLON_SYMBOL type (EQUAL_OPERATOR exp)?
;
```