Project Report for Compilers @ Fudan

Author

Linghao Zhang (13307130225@fudan.edu.cn)

Overview

Functionalities

- Grammar file: `src/MiniJava.g4` √
- Output AST √
- Error Handling √
  - Error Reporting √
    - Lexcial Errors √
    - Syntactic Errors √
    - Semantic Errors √
  - Error Recovery X
    - For lexical & syntactic errors, MiniJavaCF uses ANTLR's default recovery strategy, which includes:
      - Single token deletion
      - Single token insertion
      - Sync-and-return
      - ...
    - For semantic errors, MiniJavaCF simply report the error and exits.
- Highlights
  - Support parsing of operator precedence
  - Human-friendly error reporting
  - Comprehensive semantic analysis with type deduction

Source Codes

Tools

Fundamentals of Syntax Analysis

Workflow

1. Lexical & Syntactic Analysis

2. First pass of Semantic Analysis
    - Build the scope tree
    -
3. Second pass of Semantic Analysis
    - Check for existences of variable type & method return type
    - Check for cyclic inheritence
4. Third pass of Semantic Analysis
    - Check for symbol reference
    - Check for type compatibility

Note that encountring unrecoverable errors in each of these steps will cause MiniJavaCF to exit early.

Subtlties

Grammar Expansion

Scope Design

First Pass: ScopeBuilder

Second Pass: SymbolChecker

Third Pass: TypeChecker & TypeEvaluator

Screenshots

Discussions

Limitations

- For now, MiniJavaCF cannot underline errors involving multiple offending tokens.

Future Works

- Void method
- Multiple declaration
- Inline initialization
- Initialization check

Subtlties

- Rationality of Expression Grammer Expansion: Split from Expression
- Motivation: Precedence
- Expression ==> Primitive Types (Not every expr in original grammar will include nonAtom)
- RightValue ==> Assignment (+: Array / nonAtom)
- Error Reporting: Underlining errors, Polymorphism, Suppressing cascading errors
- Scope Design: Class & Method: extends Class Symbol & implements Interface Scope
- Validity

Todolist

- Underline errors involving multiple tokens

Workflow for Semantic Check

- 1st Pass [x] Build Scope [x] Check Duplicate Declaration
- 2st Pass [x] For Class: Parent Class not found [x] For Class: Cyclic Inheritance [x] For Method & Variable: Check for symbol type lookup [ ] For Method & Variable: Lookup -> Moved to 3rd pass
- 3rd Pass [x] Identifier(Variable): Use Before Declaration + Type Deduction - Assignment (rightValue) - atom - nonAtom - array - Function Call: params & return type - If / While / Print Statement

Error Recovery [Notes]

sync-and-return

Resynchronization set <- call stack

Recovering from Mismatched Tokens

1. Delete a token
2. Conjure up a token
3. Throw an exception

Recovering from Errors in Subrules

...