Compiler Construction
Winter 2020

Recitation 1:
Abstract Syntax of MiniJava

Yotam Feldman
Administrative

• Course website
  – Lectures: https://www.cs.tau.ac.il/~msagiv/courses/wcc20.html
  – Recitations and project: http://www.cs.tau.ac.il/research/yotam.feldman/courses/wcc20/wcc20.html
  – Forums in Moodle

• Grade: 50% project, 50% exam
  – 4 programming assignments
  – Assignments in groups of 3

• My reception hour: Sunday, 14-15
  – Email in advance, yotamfe1@mail.tau.ac.il
  – (Not @tauex)

• COVID19
  – I very much (!) appreciate you opening your webcams, but this is not required
Compilation Phases

Java code → Lexical analysis → Parsing → Abstract Syntax Tree (AST) → Semantic analysis → Intermediate code generation → LLVM code → Target code generation → x86 code

Ex. 1: Abstract Syntax Tree (AST) → Variable & method renaming → Abstract Syntax Tree (AST)

Ex. 2: Intermediate code generation

Ex. 3: Semantic analysis

Ex. 4: Abstract Syntax Tree (AST)
MiniJava

• A subset of Java
• Many restrictions
  – See project website
class Main {
    public static void main(String[] a) {
        System.out.println(new Simple().Start(1, 2));
    }
}

class Trivial {
    int f;
}

class Simple extends Trivial {
    public int Start(int a, int b) {
        int x;
        int y;
        x = a;
        y = b + 3;
        if (true) {
            f = 0;
        } else {
            f = 1;
        }
        return x + y + f;
    }
}
MiniJava Grammar

Goal ::= MainClass ( ClassDeclaration )* <EOF>
MainClass ::= "class" Identifier "{" "public" "static" "void" "main" "(" "String" "[" ""]" Identifier ")" "{" Statement "}" "}"
ClassDeclaration ::= "class" Identifier ( "extends" Identifier )? "{" ( VarDeclaration )* ( MethodDeclaration )* }" 
VarDeclaration ::= Type Identifier ;
MethodDeclaration ::= "public" Type Identifier "(" ( Type Identifier ( "," Type Identifier )? ")" ")" "{" ( VarDeclaration )* ( Statement )* "return" Expression "," }"
Type ::= "int" "[" "]"
 | "boolean"
 | "int"
 | Identifier
Statement ::= "{" ( Statement )* "}"
 | "if" ")" expression ")" Statement "else" Statement
 | "while" ")" expression ")" Statement
 | "System.out.println" ")" expression ")" ;
 | Identifier ")" expression ");"
 | Identifier ")" expression ")" ")" expression ");"
Expression ::= Expression ( "&" | "+" | "-" | "/" | "/" ) Expression
 | Expression ")" Expression ")" Expression
 | Expression ")" ")" expression ");"
 | Expression ")" expression ")" expression ");"
 | ")" expression ")" expression ");"
 | ")" expression ");" ")"
| <INTEGER_LITERAL>
```java
class Main {
    public static void main(String[] a) {
        System.out.println(new Simple().Start(1, 2));
    }
}
...
Writing MiniJava ASTs in XML
MiniJava AST

```
program
  type = program

  lineNumber 0..1
  type = int

  charPos 0..1
  type = int

mainclass 1..1
  type = mainClass

  classdecs 1..1
    type = complexType

  classdecl 0..∞
    type = classDecl
```
Class Declarations

![Diagram of class declarations]

- **classdecl**
  - type = classDecl
  - 0..∞

- **superName**
  - 0..1
  - type = string

- **fields**
  - 0..1
  - type = complexType

- **methoddecls**
  - 0..1
  - type = complexType

- **name**
  - 1..1

- **field**
  - 0..∞
  - type = varDecl

- **methoddecl**
  - 0..∞
Types
Statements

L-value

R-value

- block
  - type = blockStatement
- if
  - type = ifStatement
- while
  - type = whileStatement
- sysout
  - type = sysoutStatement
- assign
  - type = assignStatement
- assign-array
  - type = assignArrayStatement
- lv
  - type = string
- rv
  - type = exprWrapper
L-Values & R-Values

- \( x = 3; \)
- \( x = y; \)
- \( x = \text{new } B().\text{Start();} \)
- \( \text{new } B().\text{Start()} = x; \)
- \( x.f = y; \)
Expressions

- and
  - type = andExpr
- lt
  - type = ltExpr
- add
  - type = addExpr
- subtract
  - type = subtractExpr
- mult
  - type = multExpr
- not
  - type = notExpr

- array-access
  - type = arrayAccessExpr
- array-length
  - type = arrayLengthExpr
- call
  - type = methodCallExpr
- int-literal
  - type = integerLiteralExpr
- true
  - type = trueExpr
- false
  - type = falseExpr
- ref-id
  - type = identifierExpr
- this
  - type = thisExpr

- new-int-array
  - type = newIntArrayExpr
- new-object
  - type = newObjectExpr
Formal & Actual Parameters

• `public int Start(int a, int b)`
• `new Simple().Start(a, b)`
Writing AST XMLs

- Validate against the schema.
Summary

• MiniJava
• MiniJava AST
  – declarations
  – statements vs. expressions
  – formal vs. actual parameters
  – l-value vs. r-value
  – ...
• XML representation of ASTs