Advanced Compiler Design

A quick introduction to the Javali compiler framework

Luca Della Toffola
luca.dellatoffola@inf.ethz.ch
Administrative Issues

• 3 Assignments (50%)
  – A0: Control Flow Graph
  – A1: SSA Form construction/destruction
  – A2: SSA Optimizations

• Team Project (50%)
  – Details will follow
Administrative Issues

• Office hours
  – Fridays, 16:00 – 17:00
  – If possible please write an e-mail

• Mailing-list
  – acd@lists.inf.ethz.ch
  – Discuss about assignments and questions
  – Working when all groups are formed
Javali

• **Simple** OO programming language
  – Subset of Java

• **Javali specification** in the course web-site

• Used in the **assignments** and for the **project**
Javali

• We provide a framework that already implements
  – Parsing
  – Intermediate Representation
  – Semantic Checking
  – Code Generation

• You can extend the framework

• You can use your own compiler from CD
Example: counting # of zero/non-zero elements

class Main {
    void main() {
        int count, i, input, zeros, notZeros;

        count = read();
        i = 0; input = 0;
        zeros = 0;
        notZeros = 0;

        while (i < count) {
            input = read();
            if (input == 0) {
                zeros++;
            } else {
                notZeros++;
            }
            i++;
        }

        write(zeros);
        write(notZeros);
    }
}
class Main {
    void main() {
        int count, i, input, zeros, notZeros;

        count = read();
        i = 0; input = 0;
        zeros = 0;
        notZeros = 0;

        while (i < count) {
            input = read();
            if (input == 0) {
                zeros++;
            } else {
                notZeros++;
            }
            i++;
        }

        write(zeros);
        write(notZeros);
    }
}
class Main {
    void main() {
        int count, i, input, zeros, notZeros;
        count = read();
        i = 0; input = 0;
        zeros = 0;
        notZeros = 0;
        while (i < count) {
            input = read();
            if (input == 0) {
                zeros++;
            } else {
                notZeros++;
            }
            i++;
        }
        write(zeros);
        write(notZeros);
    }
}
Implementing CFG Construction

- Add the functionality to `CFGBuilder` class
- `AstVisitor` class to traverse the AST nodes
- Check `BasicBlock` and `ControlFlowGraph` classes

General strategy

for each `method` in program {
    `walk` through the method’s list of `AST nodes`
    add each `AST node` to the current `basic block`
}
Implementing CFG Construction

**IfElse, WhileLoop, and ReturnStmt** AST nodes

– **Create** new basic block if necessary

– **Connect** new basic blocks to existing ones so that connections reflect the method’s control flow

– **IfElse** and **WhileLoop** nodes are **not allowed** in the CFG

– Take additional care of **ReturnStmt** nodes
Debug

• In debug mode the compiler outputs the **CFG** into a *cfg.dot* file
  – Set the field `Main.debug` to a proper `java.io.Writer`

• Use the **Graphviz** package to visualize the CFG

Testing

• Example programs included in **javali_tests**

• Reference solution output generated by our server
  – Run **TestSamplePrograms** as JUnit test
  – To delete reference files use ANT target **clean-tests-all**
Get fragment from:
https://svn.inf.ethz.ch/svn/trg/cd_students/2015ss/teams/<your_team>/CD2_A0/
Demo