TDDD55 - Compilers and Interpreters

Laboration Overview

Martin Sjölund martin.sjolund@liu.se

Department of Computer and Information Science
Linköping University

2018-11-07

Purpose of Lessons

- Practice theory
- Introduce the laboratory assignments
- Prepare for the final examination

Prepare by reading the laboratory instructions, the course book, and the lecture notes. All the laboratory instructions and material available in the course directory, ~**TDDD55/lab/** or on the course homepage.

Laboratory Assignments

- In the laboratory exercises you should get some practical experience in compiler construction.
- There are 4 separate assignments to complete in 4x2 laboratory hours. You will also (most likely) have to work during non-scheduled time.

Lesson Schedule

- Formal languages and automata theory
- Formal languages and automata theory, Flex
- Intermediate code generation, Bison
- Exam preparation

Handing in and deadline

- Demonstrate the working solutions during scheduled sessions.
- Then, hand in code and answers to any questions via e-mail. One e-mail from your LiU-email per group (subject: TDDD55: lab no.).
- Deadline for all the assignments is the study period. Check the homepage for dates.
- Sign up in the webreg!

Laboratory Assignments

- Lab 1 Attribute Grammars and Top-Down Parsing
- Lab 2 Scanner Specification
- Lab 3 Parser Generators
- Lab 4 Intermediate Code Generation

1. Attribute Grammars and Top-Down Parsing

- Some grammar rules are given
- Your task:
  - Rewrite the grammar (eliminate left recursion, etc.)
  - Add attributes and attribute rules to the grammar
  - Implement your attribute grammar in a C++ class named **Parser**. The method **Parser::Parse** should return the value of a single statement in the language.

2. Scanner Specification

- Finish a scanner specification given in Flex (scanner.l), by adding rules for comments, identifiers, integers, and reals.
- Details in lesson 2.

3. Parser Generators

- Finish a parser specification given in Bison (parser.y), by adding rules for expressions, conditions and function definitions, ...
- Augment the grammar with error productions.
- Details in lesson 3.

4. Intermediate Code Generation

- The purpose of this assignment to learn about how parse trees can be translated into intermediate code.
- Finish a generator for intermediate code by adding rules for some language statements.
- Details in lesson 3.

www.liu.se