lab session 1: compiler construction

November 18, 2015

The first lab session of the course *Compiler Construction* focuses on constructing a DFA (Deterministic Finite state Automaton) from an NFA (Non-Deterministic Finite state Automaton). This construction is a crucial part of any scanner generator (including the one that we will build in future lab sessions).

The conversion process (NFA to DFA) has been discussed in the lectures, and can be found in the lecture slides of week 1. Moreover, you are referred to the following wikipedia page `https://en.wikipedia.org/wiki/Powerset_construction` or youtube movie `https://www.youtube.com/watch?v=taClnxU-nao`.

On the internet you can find codes that perform this conversion process. It is perfectly acceptable to have a look at those and use them as an inspiration for your own implementation. However, it is definitely not allowed to plainly copy (parts of) these codes. That would be plagiarism. If the teaching assistants detect plagiarism, then the report is sent directly to the exam committee.

Some code has been prepared for you already. There are three files available on Nestor (a.k.a. the student portal). The files `intset.h` and `intset.c` implement an ADT (Abstract Data Type) `intSet` which is used to represent sets of unsigned integers. In the NFA to DFA conversion process, this data type is used for representing sets of states, where the states are simply integer numbers. Have a look at the file `intset.h` which operations are available. You are not allowed to change the implementation of the data type `intSet`.

In the third file, called `nfa.c`, a simple data structure `nfa` has been implemented. The data type is used to store an NFA. The file contains a simple program that only reads an NFA from a file, and copies it into the file `out.nfa`. An example input file specifying an NFA can be found in the file `example.nfa`. Of course, copying input to output is not a very useful functionality. The objective of this lab is to output an equivalent DFA, so you should change the program such that it still accepts an NFA as its input but outputs a DFA (for example in the file `out.dfa`). Before you start coding, you are advised to study the details of the current implementation. Make sure you understand the structure of the program.

Deadline: Tuesday November 24th, beginning of lab session 2.
Submit your code to Justitia (at `http://justitia.housing.rug.nl`). Note that justitia will always accept your code, even if it is incorrect! In this course, we use Justitia just as a dropbox for your solutions which can be tested manually by the teaching assistants. You do not have to write a report for this lab session.

Grading: The grading of your implementation will be based on:

- Correctness: 50%
- Programming quality, efficiency, etc.: 30%
- Tests performed by the teaching assistants: 20%

- Penalty for missing deadlines: -1 grade point per day.