Software-Modellierung und Verifikation
Informatik 2
Prof. J.-P. Katoen
RWTH Aachen

Priv.-Doz. T. Noll noll@cs.rwth-aachen.de
Ch. Jansen christina.jansen@cs.rwth-aachen.de

# 3. Exercise sheet *Compiler Construction 2010*

Due Wed., 10 November 2010, *before* the exercise course begins.

**Exercise 3.1:** **(3 points)**

Prove the correctness of the top down analysis automaton $NTA(G)$ for a grammar $G = \langle N, \Sigma, P, S\rangle$, i.e. show that for all $w \in \Sigma^*$ and all $z \in \{1, \ldots, |P|\}^*$:

$$(w, S, \varepsilon) \vdash^* (\varepsilon, \varepsilon, z) \quad \text{implies} \quad S \underset{l}{\overset{z}{\Rightarrow}} w$$

**Exercise 3.2:** **(3+2+1+2+2 points)**

Consider the propositional logic given by:

$$tt \,|\, ff \,|\, \text{ words over the alphabet } \Sigma = \{a, \ldots, z\}$$

Formulae are then inductively defined by: If $\Phi, \Psi$ propositional formulae, then so are

$$(\neg\Phi), (\Phi \wedge \Psi), (\Phi \vee \Psi), (\Phi \rightarrow \Psi)$$

(a) Write a `(f)lex`-program, that scans such propositional formulae.

The `main`-function should output pairs of tokens and attributes. The attribute should hold the label of the proposition or $-$ in all other cases. Any whitespace in the input should be ignored.

Please use the following token definition:

```
enum token OPEN=1, CLOSE, ATOM, NOT, AND, OR, IFTHEN
```

Your program should compile to an executable by means of the command:

```
flex progName.l && cc lex.yy.c -lfl
```

(b) Provide four example runs of your scanner.

(c) Modify your `(f)lex`-program in a way such that it additionally counts (and outputs) the number of opening and closing brackets.

(d) Have a look into the generated file `lex.yy.c`. Which lines correspond to the DFA `(f)lex` generated according to your specification? Which lines (or program labels) correspond to the different modes of the backtracking DFA introduced in the lecture?

(e) `(f)lex` offers the possibility to output the generated DFA (and NFA) by setting the option `-T`. Try to picture the generated DFA, then compare it to the one you generated manually for a slightly modified propositional logic in Ex. 1.

Please send your (compilable) source code and example runs to `christina.jansen@cs.rwth-aachen.de`. Include your matriculation numbers in the subject!