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

Priv.-Doz. T. Noll noll@cs.rwth-aachen.de
D. Klink klink@cs.rwth-aachen.de

4. Exercise sheet *Compiler Construction 2008*

Due to Wed., 4 June 2008, *before* the exercise course begins.
Hand in your solutions in groups of three!

Exercise 4.1:

Consider the grammar G given by:

$$\begin{array}{lcl} S & \to & N * N \mid N * * N \\ N & \to & N0 \mid N1 \mid M \\ M & \to & 1 \end{array}$$

a) Show that $G \notin LL(1)$ by analysing the lookahead sets.

b) Construct $G' \in LL(1)$ with $\mathcal{L}(G) = \mathcal{L}(G')$.

c) Show that $G' \in LL(1)$, again by analysing the lookahead sets.

Exercise 4.2:

Consider $G_n = \langle N, \Sigma, P_n, S \rangle$ (a variant of the grammar from Exercise 3.2) where instead of linebreaks there are commas and the matrix contains `a`'s and `b`'s.

$$\begin{array}{lllll} P_n: & S & \to & 1\texttt{,}S_1 \mid \ldots \mid n\texttt{,}S_n & \\ & S_i & \to & R_i \stackrel{i\times}{\ldots} R_i & \text{for all } i \in \{1,\ldots,n\} \\ & R_i & \to & N \stackrel{i\times}{\ldots} N\texttt{,} & \text{for all } i \in \{1,\ldots,n\} \\ & N & \to & \texttt{a} \mid \texttt{b} & \end{array}$$

a) Write a Java class `Lexer` that implements `java.util.Iterator`. Choose singleton symbol classes for $1,\ldots,n$, `a`, `b`, the comma and document how tokens are represented.

b) Write a Java recursive descent parser `Parser` for the grammar using the `Lexer` from a). The first commandline argument is to specify n, the second argument should be the input word. Calling, say, `java g.Parse 2 2,ab,ba,` should print something like

```
dimensions 2x2
parsing 2 rows
parsing 2 chars
parsing char: a
parsing char: b
parsing 2 chars
parsing char: b
parsing char: a
```

to the console (instead of just giving a sequence of numbers as in the example from the lecture).

Avoid cryptic and/or messy code and send it (the java files) to `klink@cs.rwth-aachen.de`. Use `Exercise 4.2` as subject and add your student ID numbers (Mat.nr.).

Exercise 4.3:

Consider the grammar G given by:

$$\begin{array}{lcl} S' & \rightarrow & S \\ S & \rightarrow & MR \\ M & \rightarrow & LM \mid m \\ L & \rightarrow & LL' \mid l \\ L' & \rightarrow & LL' \mid l' \\ R & \rightarrow & RR' \mid r \\ R' & \rightarrow & RR' \mid r' \end{array}$$

a) Show that $G \notin LR(0)$.

b) Give a grammar $G' \in LR(0)$ (and show that $G' \in LR(0)$) such that for $w \in \Sigma^*$, $a \in \Sigma$:

$$wa \in \mathcal{L}(G) \Leftrightarrow wm \in \mathcal{L}(G')$$

c) Compute an accepting run of $NBA(G')$ for input $lll'mrm$

Exercise 4.4: (optional)

Use ANTLR to write a calculator capable of (at least) addition and multiplication of binary encoded numbers (including parentheses), i.e. for an input `(100+001)*011*010` the parser should print `101010` to the console (the example code shown below should contain all necessary ANTLR constructs).

a) Write an appropriate ANTLR grammar.

b) Add functionality.

Avoid cryptic and/or messy code and send it (the ANTLR file) to `klink@cs.rwth-aachen.de`. Use `Exercise 4.4` as subject and add your student ID numbers (Mat.nr.).

```
grammar CalcEx;

start  : e=multExpr {
             System.out.println("="+$e.value);
         };

multExpr returns [int value]
       : e=NUMBER {
             int n = Integer.parseInt($e.text);
             $value = n;
             System.out.print(n);
         }
         ( '*' {
               System.out.print("*");
           }
           e=NUMBER {
               n = Integer.parseInt($e.text);
               $value *= n;
               System.out.print(n);
           }
         )*;

NUMBER : ('0'..'9')+;
```