Compiler Construction

Dr. Michael Petter, Raphaela Palenta SS 2019

Exercise Sheet 5

Assignment 5.1 Parser for string-represented RegEx

To complete our own regular expression toolkit, we attach a parser for string-represented regular expressions. In the last week we constructed a contex-free-grammar for regular expressions that is in LL(1). Implement a recursive descent parser that parses a string according to this grammar and returns the corresponding `RegexTree`. Use this parser in front of your implementation from the previous exercises for constructing a DFA and NFA for a `RegexTree`.

Assignment 5.2 Shift-reduce parser

Consider the following grammar G with start symbol A:

$$\begin{array}{lcl} A & \to & Ab \mid aB \\ B & \to & Bd \mid d \end{array}$$

1. Is this grammar LL(1)? Justify your answer!
2. Construct a (generalized) pushdown automaton M_G^R for parsing G by specifying the states (stack alphabet), start state, end state, and the transition table. Partition the transitions into shifts and reduces!

Assignment 5.3 LR(0)-parser

Consider the following grammar G with start symbol A:

$$\begin{array}{lcl} A & \to & ABa \mid c \\ B & \to & b \mid \epsilon \end{array}$$

1. Draw the characteristic automaton $c(G)$.
2. Draw the canonical LR(0)-automaton $LR(G)$.
3. Construct the $LR(0)$ parser for G by specifying the states (stack alphabet), start state, end state, and the transition table. Partition the transitions into shifts and reduces!
4. Is the automaton deterministic? If not, then list all conflicts.