apl. Prof. Dr. Thomas Noll

Friedrich Gretz, Souymodip Chakraborty

Exercise 1 (Prefix property): (2 Points)

A language $L \in \Sigma^*$ is called prefix-free, if $L \cap L\Sigma^+ = \emptyset$, i.e. if no proper prefix of a word in L is in L, too.

Show that the following holds for all non prefix-free languages L: $L \notin \mathfrak{L}(LR(0))$.

Exercise 2 (LR items): (3 Points)

The grammar G is defined as:

$$\begin{array}{lcl} S' & \to & S\ c \\ S & \to & S\ A \mid A \\ A & \to & a\ S\ b \mid a\ b \end{array}$$

Consider a right most derivation: $S' \Rightarrow^*_{rm} \alpha A w \Rightarrow_{rm} \alpha\beta w$. $\alpha\beta w$ is called a *right sentential form*, say γ. THE (unique) *handle* of γ is β. A *viable prefix* of G is a prefix of any right sentential form γ ending no farther right than the right end of the handle of γ.

a) Build an NFA $A = (Q, V \cup T, \delta, q_0, Q)$ recognising the viable prefixes of G using the following definition. Q is the set of *items*, V and T are the sets of non-terminals and terminals of of G. q_0 is the initial state.

- $(q_0, \epsilon, S' \to .\ \alpha) \in \delta$ iff $S' \to \alpha$ is a production of G,
- $(A \to \alpha\ .\ B\ \beta, \epsilon, B \to .\ \eta) \in \delta$ iff $B \to\ \eta$ is a production of G,
- $(A \to \alpha\ .\ X\ \beta, X, A \to \alpha\ X\ .\ \beta) \in \delta$.

b) Determinise A to A'.

c) What is the relationship between the states of A' and the LR(0) sets defined in the lecture?

d) Is G an LR(0) grammar?

Exercise 3 (Implementation): (5 Points)

After building a lexer in the previous exercises we now start building a parser for our *WHILE* language. In this exercise we take the first steps towards a parser.
Assume the following grammar for the *WHILE* language. The terminal alphabet is the set of tokens, non-terminals and starting symbol are obvious. The production rules are given below:

start	→	program EOF
program	→	statement program | statement
statement	→	declaration SEM | assignment SEM | branch | loop | out SEM
declaration	→	INT ID
assignment	→	ID ASSIGN expr | ID ASSIGN READ LBRAC RBRAC
out	→	WRITE LBRAC expr RBRAC | WRITE LBRAC STRING RBRAC
branch	→	IF LBRAC guard RBRAC LCBRAC program RCBRAC |
		IF LBRAC guard RBRAC LCBRAC program RCBRAC ELSE LCBRAC program RCBRAC
loop	→	WHILE LBRAC guard RBRAC LCBRAC program RCBRAC
expr	→	NUM | ID | subexpr | LBRAC subexpr RBRAC
subexpr	→	expr PLUS expr | expr MINUS expr | expr TIMES expr | expr DIV expr
guard	→	relation | subguard | LBRAC subguard RBRAC | NOT LBRAC guard RBRAC
subguard	→	guard AND guard | guard OR guard
relation	→	expr LT expr | expr LEQ expr | expr EQ expr | expr NEQ expr | expr GEQ expr | expr GT expr

Task:

You will find this grammar hard coded in the file parser/WhileGrammar.java. Your task is to

- compute the LR(0) sets for this grammar,
- detect conflicts and indicate where they occur,
- provide the total number of sets and conflicts.

As with the previous exercises, please download the framework from the course website and fill the gaps in parser/GotoDFA.java and parser/LR0Set.java. **Please submit your code via the L2P system. Do not send it by email any more. Make sure you pack the whole project (not just single files) in one zip file. Name this file with names and matriculation numbers of your group members!**