

Exercise Sheet 4

Assignment 4.1 Top-Down-Parsing

Consider the grammar $G = (N, T, \delta, S)$ with $T = \{s, a\}$, $N = \{S, A\}$, start state S and the following production rules

$$\begin{aligned} \delta: \quad S &\rightarrow A s \\ A &\rightarrow a A^0 \mid \epsilon^1 \end{aligned}$$

1. Construct the Item Pushdown Automaton M_G^L for G following the algorithm introduced in the lecture. Split your set of transitions in the three disjoint sets expansions, shifts, and reductions.
2. Can the automaton M_G^L be used to construct a deterministic $LL(0)$ -Parser? Justify your answer!
3. Construct the lookahead sets First_1 for S and A !
4. Construct the Follow_1 sets for S and A !
5. Construct the lookahead table for M_G^L with lookahead 1!

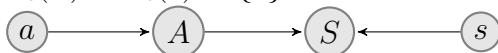
Suggested Solution 4.1

1. start state $[S \rightarrow \bullet As]$, end state $[S \rightarrow As \bullet]$, transition relations:

1	e	$[S \rightarrow \bullet As]$	ϵ	$[S \rightarrow \bullet As][A \rightarrow \bullet aA]$
2	e	$[S \rightarrow \bullet As]$	ϵ	$[S \rightarrow \bullet As][A \rightarrow \bullet]$
3	s	$[A \rightarrow \bullet aA]$	a	$[A \rightarrow a \bullet A]$
4	r	$[S \rightarrow \bullet As][A \rightarrow \bullet]$	ϵ	$[S \rightarrow A \bullet s]$
5	s	$[S \rightarrow A \bullet s]$	s	$[S \rightarrow As \bullet]$
6	r	$[A \rightarrow a \bullet A][A \rightarrow \bullet]$	ϵ	$[A \rightarrow aA \bullet]$
7	e	$[A \rightarrow a \bullet A]$	ϵ	$[A \rightarrow a \bullet A][A \rightarrow \bullet]$
8	e	$[A \rightarrow a \bullet A]$	ϵ	$[A \rightarrow a \bullet A][A \rightarrow \bullet aA]$
9	r	$[A \rightarrow a \bullet A][A \rightarrow aA \bullet]$	ϵ	$[A \rightarrow aA \bullet]$
10	r	$[S \rightarrow \bullet As][A \rightarrow aA \bullet]$	ϵ	$[S \rightarrow A \bullet s]$

Expansions are marked by 'e', shifts by 's' and reduces by 'r'.

2. No! Conflicts arise between transitions (1,2) and (7,8), respectively.
3. $\text{empty}(S) = \text{false}$
 $\text{empty}(A) = \text{true}$
 $F_\epsilon(S) \supseteq F_\epsilon(A) \supseteq F_\epsilon(a) \supseteq \{a\}$
 $F_\epsilon(S) \supseteq F_\epsilon(s) \supseteq \{s\}$



Each node is in a separate strongly connected component.

$$\begin{aligned}
F_\epsilon(A) &= \{a\} \\
F_\epsilon(S) &= \{a, s\} \\
\text{First}_1(A) &= F_\epsilon(A) \cup \{\epsilon\} = \{a, \epsilon\} \\
\text{First}_1(S) &= F_\epsilon(S) = \{a, s\}
\end{aligned}$$

4. $\text{empty}(S) = \text{false}$
 $\text{empty}(A) = \text{true}$
 $\text{Follow}_1(S) \supseteq \{\epsilon\}$
 $\text{Follow}_1(A) \supseteq F_\epsilon(s) \supseteq \{s\}$
 $\text{Follow}_1(A) \supseteq \text{Follow}_1(A)$



$$\begin{aligned}
\text{Follow}_1(S) &= \{\epsilon\} \\
\text{Follow}_1(A) &= \{s\}
\end{aligned}$$

5.

$$\begin{aligned}
\text{A-rule 0: } \text{First}_1(aA) \odot_1 \text{Follow}_1(A) &= \{a\} \odot_1 \{s\} = \{a\} \\
\text{A-rule 1: } \text{First}_1(\epsilon) \odot_1 \text{Follow}_1(A) &= \{\epsilon\} \odot_1 \{s\} = \{s\}
\end{aligned}$$

LL(1)-lookahead table:

	a	s
A	0	1

Assignment 4.2 Recursive descent parser

Complete the implementation (`Parser.java`) of the recursive descent parser for the grammar of Assignment 4.1. If the input is accepted a success message should be printed out. Otherwise the reason why the parsing failed should be printed out.

Assignment 4.3 Grammar for regular expressions

We want to prepare a implementation of a parser for string-reperesented regular expressions. (The implementation is part of the next exercise sheet.)

1. Give a grammar for regular expressions that is LL(1). We consider regular expressions as defined in the lecture, i.e., we have operators \cdot , $|$, $*$.
2. Prove that the grammar is LL(1).

Hint: You may start with any grammar for regular expressions and then transform this grammar if it is not already LL(1).

Suggested Solution 4.3

1. Functionally, we would like to use something like the Arithmetic Expression Gram-

mar, just with regular expression operators:

$$\begin{array}{lcl}
\delta_1 : < regex > & \rightarrow & < concat > \quad '|' \quad < regex > \\
& & | & < concat > \\
& & | & \epsilon \\
< concat > & \rightarrow & < rep > \quad < concat > \\
& & | & < rep > \\
< rep > & \rightarrow & < atom > \quad '*' \\
& & | & < atom > \\
< atom > & \rightarrow & '(' \quad < regex > \quad ')' \\
& & | & '[' a' -' z']
\end{array}$$

However, since production of the form $A \rightarrow B\beta|B$ immediately introduce alternatives with intersecting first sets, we transform the grammar a little:

$$\begin{array}{lcl}
\delta_2 : < regex > & \rightarrow & < concat > \quad A_1 \\
& A_1 & \rightarrow & '|' \quad < regex > \\
& & | & \epsilon \\
< concat > & \rightarrow & < rep > \quad A_2 \\
& A_2 & \rightarrow & < concat > \\
& & | & \epsilon \\
< rep > & \rightarrow & < atom > \quad A_3 \\
& A_3 & \rightarrow & '*' \\
& & | & \epsilon \\
< atom > & \rightarrow & '(' \quad < regex > \quad ')' \\
& & | & '[' a' \dots ' z', ' \epsilon ']
\end{array}$$

2. We need to concentrate on differentiating the alternatives from each other by means of determining the $First_1$ Sets of their right hand sides 1-concatenated with their $Follow_1$ sets. This means, we need to determine F_ϵ sets, thus we get

$$\begin{aligned}
F_\epsilon(< regex >) &\supseteq F_\epsilon(< concat >) \supseteq F_\epsilon(< rep >) \supseteq F_\epsilon(< atom >) \\
F_\epsilon(A_1) &\supseteq \{'|\} \quad F_\epsilon(A_2) \supseteq F_\epsilon(< concat >) \quad F_\epsilon(A_3) \supseteq \{'*\} \\
F_\epsilon(< atom >) &\supseteq \{'(' , ' a' -' z', ' \epsilon '\}
\end{aligned}$$

... while the relevant first set is:

$$First_1(< concat >) = \{'(' , ' a' \dots ' z', ' \epsilon '\}$$

Let's continue with $Follow_1$ Sets:

$$\begin{aligned}
Follow_1(A_1) &\supseteq Follow_1(< regex >) \\
Follow_1(A_2) &\supseteq Follow_1(< concat >) \\
Follow_1(A_3) &\supseteq Follow_1(< rep >) \\
Follow_1(< regex >) &\supseteq Follow_1(A_1) \cup \{' '\} \cup \{\epsilon\} \\
Follow_1(< concat >) &\supseteq Follow_1(A_2) \cup F_\epsilon(A_1) \cup Follow_1(< regex >) \\
Follow_1(< rep >) &\supseteq F_\epsilon(A_2) \cup Follow_1(< concat >)
\end{aligned}$$

Now, we can compare the $First_1$ of the alternatives for A_1, A_2 and A_3 :

$$\begin{aligned}
First_1(A_1, 1) &: \quad '| \\
First_1(A_1, 2) &: \quad ')', \epsilon \\
First_1(A_2, 1) &: \quad '(', 'a' \dots 'z', 'e' \\
First_1(A_2, 2) &: \quad '|', ')', \epsilon \\
First_1(A_3, 1) &: \quad '* \\
First_1(A_3, 2) &: \quad '|', ')', \epsilon
\end{aligned}$$

and we see, that their respective intersection is empty