Fill out the fields above.

Make use only of an indelible pen in black or blue color.

Do not make use of “Tipp-Ex” or something similar.

To solve the exam you have 90 minutes.

Check that you received 4 pages.

The maximum number of points that you can obtain in this exam is 50. You need 20 points to pass.

premature hand in ................. rest room break from ................. until .................
Assignment 1 Berry-Sethi Approach [15 Points]
Use the Berry-Sethi Algorithm and transform the expression $r = ((a|b)^*d)(b|c)^*$ as follows:

1. Draw the regular expression as a tree.
2. Compute the empty attribute.
3. Compute the first attribute.
4. Compute the next attribute.
5. Compute the last attribute.
6. Construct and draw the automaton.
7. Is the resulting automaton deterministic or not?
8. Is the resulting automaton minimal or not?
Consider grammar $G = (N, T, \delta, S')$ with the terminals $T = \{a, b\}$, non-terminals $N = \{S', A, B\}$, start symbol $S'$ and productions $\delta$:

\[
\delta : \\
S' \rightarrow A \\
A \rightarrow B \mid BbA \mid BbBa \\
B \rightarrow a
\]

1. For the Item-Pushdown-Automaton for $G$, create the Expansion and Shift transitions.

2. Can you create an LL(1) Parser from this Automaton? Either create the Lookahead-Table for $G$, or elaborate on the situation, in which the LL(1) Parser would run into a conflict.

3. Claim 1: “The canonical LR(0) Automaton for $G$ is conflict-free.” Prove or disprove with facts deduced from the canonical LR(0)-Automaton.

Assignment 3 Attribute Grammars

The following grammar represents the fraction of a language that treats expression statements. The expressions support binary operators, value assignment as well as multidimensional array access. A few specific properties of this language are not treated with syntactical rules, instead they are addressed via the semantical analysis.

<table>
<thead>
<tr>
<th>rule</th>
<th>production</th>
<th>attribute system</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$S' ::= { \text{stmts} }$</td>
<td>$\text{valid}[0] := \text{valid}[2]$</td>
</tr>
<tr>
<td>2</td>
<td>$\text{stmts} ::= \text{expr} ; \text{stmts}$</td>
<td>$\text{valid}[0] := \text{valid}[1] \land \text{valid}[3]$</td>
</tr>
<tr>
<td>3</td>
<td>$\varepsilon$</td>
<td>$\text{valid}[0] := \text{true}$</td>
</tr>
<tr>
<td>4</td>
<td>$\text{expr} ::= \text{expr} \text{binop} \text{expr}$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$\text{expr} ::= \text{expr} \text{=} \text{expr}$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$\text{expr} ::= \text{expr} [ \text{elist} ]$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$\text{const}$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$\text{var}$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$\text{elist} ::= \text{expr} , \text{elist}$</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>$\text{expr}$</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Complete the attribute grammar, such that

1. the boolean attribute $\text{valid}$ evaluates to true if in its particular subtree the only expressions that appear on the left hand sides of assignments are either variables or array access expressions.

2. expressions with unary pre-/postfix incrementors (++) become possible

3. only pre-/postfix incrementor expressions of array accesses and variables contribute to $\text{valid}$ evaluating to $\text{true}$

4. pre-/postfix incrementor expressions that appear on left hand sides of an assignments contribute to $\text{valid}$ evaluating to $\text{false}$

Attention:

- You may introduce further attributes by need
- You may introduce further productions/variants by need