Solution to Homework 9

**Background.** In Problem 7, we considered a grammar with rules

\[ S \rightarrow \varepsilon \text{ and } S \rightarrow +S - \cdot \]

**Tasks:**

1. Use a general algorithm to construct a (non-deterministic) pushdown automaton that corresponds to context-free grammar described in Problem 7.

2. Show, step by step, how the word \(+ + --\) will be accepted by this automaton.

**Solution to Task 1.** By using the general algorithm, we get the following pushdown automaton:

![Pushdown Automaton Diagram]

**Solution to Task 2.** Let us show how this is done on the example of the word \(+ + --\) generated by the above automaton:

\[ S \rightarrow +S- \rightarrow + + S - - \rightarrow + + -- \cdot \]

To make this derivation clearer, let us mark the variable \( S \) corresponding to different transitions by subscripts:
• we start with the first occurrence \( S_1 \) of the variable \( S \);

• we then use the rule \( S_1 \rightarrow +S_2- \) whose right-hand side contains the second occurrence \( S_2 \) of the variable \( S \);

• this occurrence, in its turn, gets transformed into \( S_2 \rightarrow +S_3- \) for yet another occurrence \( S_3 \) of the same variable \( S \); so far, the derivation takes the form

\[
S_1 \rightarrow +S_2- \rightarrow + + S_3 - -;
\]

• finally, to the occurrence \( S_3 \), we apply the rule \( S_3 \rightarrow \varepsilon \), resulting in the desired derivation:

\[
S_1 \rightarrow +S_2- \rightarrow + + S_3 - - \rightarrow + + - - .
\]

Let us now trace what our pushdown automaton will do. We start in the state \( s \) with an empty stack:

![Diagram](attachment:diagram.png)

The only thing we can do when in the state \( s \) is push the dollar sign into the stack and get to the intermediate state \( i \):
The contents of the stack is as follows:

\[ \varepsilon, S \rightarrow \$ \]

When we are in the state \( i \), the only thing we can do is push the starting variable \( S \) (which corresponds to the first occurrence \( S_1 \) of this variable) into the stack and go into the working state \( w \):

\[ \varepsilon, S \rightarrow \$ \]

Now, the stack contains the starting variable on top of the dollar sign:

\[ S, \$ \]

Now that we are in the working state, we can start following the rules that were used to derive the word \(+ + --\). The first rule was \( S \rightarrow +S- \), or, to be precise, \( S_1 \rightarrow +S_2- \). As we have mentioned, this rule is implemented in three steps:
• first, we pop $S$ (that corresponds to the first occurrence $S_1$) and push the last symbol of the right-hand side – in this cases, the symbol $-$ – into the stack, getting into the auxiliary state $a_1$;

• then, we push $S$ (that corresponds to the second occurrence $S_2$) into the stack, getting into the auxiliary state $a_2$;

• finally, we push $+$ into the stack, and go back to the working state $w$.

Let us illustrate this step by step.

First, we pop $S$, push $-$, and go into the state $a_1$:

The stack will now have $-$ instead of $S$:

Then, we push $S$ (corresponding to $S_2$) into the stack and go into the state $a_2$:
The stack will now have $S$ on top of its previous contents:

\[
\begin{array}{c}
S \\
-
\end{array}
\]

Finally, we push $+$ into the stack, and go back to the working state:

The stack will now have symbol $+$ at the top:

\[
\begin{array}{c}
+S \\
-
\end{array}
\]

Now, the symbol $+$ is top of the stack. The only thing we can do if a terminal symbol is on top of the stack is use one of the rules of the type $x, x \rightarrow \varepsilon$ where $x$ stands for the corresponding terminal symbol.

In our case:

- since the terminal symbol on top of the stack is the symbol $+$,
- we need to use the rule $+, + \rightarrow \varepsilon$,

i.e., we read the symbol $+$ from the original word $+ + -$ and pop the top symbol $+$ from the stack:
After this popping, the variable $S$ (corresponding to $S_2$) will be on top of the stack:

\[
\begin{array}{c}
S \\
- \\
\$
\end{array}
\]

According to the original derivation of the word \texttt{++--}, to get rid of the second occurrence $S_2$ of the variable $S$, we also use the rule $S ightarrow +S-$, or, to be precise $S_2 ightarrow +S_3-$. So, similarly to what we have before when we used this rule, first, we pop $S$, push $-$, and go to the state $a_1$:

Now, we have $-$ instead of $S$ on top of the stack:

\[
\begin{array}{c}
- \\
- \\
$
\end{array}
\]
After that, we push $S$ (that corresponds to the third occurrence $S_3$) and go to state $a_2$:

The stack now has the form:

Finally, we push $+$ into the stack and go back to the working state $w$:

Now, the stack has the following form:
Now again, we have a terminal symbol $+$ on top of the stack, so the only thing we can do is use the rule $+,+ \rightarrow \varepsilon$: we read the second symbol $+$ of the word $++--$ (the first one we have already read, so the cursor points to the second one) and pop $+$ from the stack. As a result, we get the following state:

The stack has the following form:

On top of the stack is the variable $S$ corresponding to the third occurrence $S_3$. To get rid of this variable, in the original derivation of the sequence, we used the rule $S \rightarrow \varepsilon$ — or, to be more precise, $S_3 \rightarrow \varepsilon$. This rule of the grammar corresponds to the transition $\varepsilon, S \rightarrow \varepsilon$ of the pushdown automaton, i.e., we pop $S$ from the stack:
The stack now takes the following form:

\[
\begin{array}{c}
\text{−} \\
\text{−} \\
\text{§}
\end{array}
\]

There is a terminal symbol on top of the stack – in this case, the symbol −. We want an empty stack at the end. The only way to get rid of − is to use the rule −, − → ε, i.e., to read the next symbol − from the word + + − −, and to pop − from the stack:

\[
\begin{array}{c}
\text{−} \\
\text{−} \\
\text{§}
\end{array}
\]

Now, the stack has the following form:

\[
\begin{array}{c}
\text{−} \\
\text{§}
\end{array}
\]

Again, we have a terminal symbol − on top of the stack, so we again use the rule −, − → ε, i.e., we read the next symbol − of the word + + − −, and we pop − from the stack:

\[
\begin{array}{c}
\text{−} \\
\text{§}
\end{array}
\]
Now, the stack only contains the dollar sign:

\[ \$ \]

We have read all the letters of the original word, and all we have in the stack is the dollar sign. So now, we can use the rule \( \varepsilon, \$ \rightarrow \varepsilon \) to pop the dollar sign and to go to the final state:

Now, we are in the final state \( f \) with the empty stack. This means that the word \( + + -- \) is accepted by this pushdown automaton.

A graphical description of the transitions.

<table>
<thead>
<tr>
<th>read</th>
<th>state</th>
<th>stack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( s )</td>
<td>$ $ $ $ $ $ $ $ $ $</td>
</tr>
<tr>
<td></td>
<td>( i )</td>
<td>\varepsilon \varepsilon \varepsilon S \varepsilon + S \varepsilon S \varepsilon S \varepsilon S \varepsilon S</td>
</tr>
<tr>
<td></td>
<td>( w )</td>
<td>\varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon</td>
</tr>
<tr>
<td></td>
<td>( a_1 )</td>
<td>\varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon</td>
</tr>
<tr>
<td></td>
<td>( a_2 )</td>
<td>\varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon \varepsilon</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>read</th>
<th>state</th>
<th>stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>( + )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( - )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( - )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( - )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( - )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( - )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( - )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( - )</td>
<td>( w )</td>
<td>( S )</td>
</tr>
<tr>
<td>( S )</td>
<td>( $ )</td>
<td>( $ )</td>
</tr>
<tr>
<td>( S )</td>
<td>( $ )</td>
<td>( $ )</td>
</tr>
<tr>
<td>( S )</td>
<td>( $ )</td>
<td>( $ )</td>
</tr>
</tbody>
</table>

10