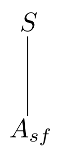
Solution to Homework 11

Task. Use the general algorithm to transform the pushdown automaton from Problem 6 into a context-free grammar. Show, step-by-step, how the resulting grammar will generate the word CLC.

Solution. Let us recall how the word CLC is accepted by this automaton:

read			C		L	C				
state	s	w	a_3	w	w	a_3	w	f	f	f
stack		\$	\$	C \$	\$	\$	C \$	C \$	\$	

We start with the state s, we end up in the final state f. Thus, the first rule we apply if the rule $S \to A_{sf}$;



The first symbol we push is the dollar sign, this dollar sign is popped at the end. Thus, we have the following combination of pop-push rules:

In general, we have the two transitions

What do we need to plug in instead of p, q, etc. in the general 2-rule picture to come up with this particular picture:

- instead of p, we place s;

- instead of q, we place w;
- instead of s and t, we place f;
- instead of x and y, we place ε.

If we make these substitutions in the general rule:

$$A_{ps} \to x A_{qr} y,$$

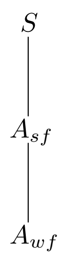
we get the rule

$$A_{sf} \to \varepsilon A_{wf} \varepsilon.$$

Since concatenation with the empty string does not change anything, this means

$$A_{sf} \to A_{wf}.$$

Thus, the derivation so far takes the following form:



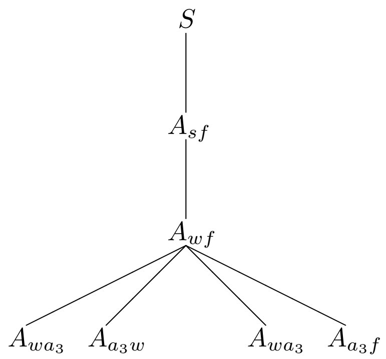
We covered how we push the dollar sign and how we pop it. Let us underline what we have covered:

read			C		L	C				
state	s	w	a_3	w	w	a_3	w	f	f	f
stack		$\underline{\$}$	$\underline{\$}$	C $\underline{\$}$	$\underline{\$}$	$\underline{\$}$	C $\underline{\$}$	C $\underline{\$}$	\$	

If we ignore the dollar signs – since we already took care of them – then we see that we have three intermediate states with the empty stack. So, we need to use the transitivity rule, which in this case takes the form

$$A_{wf} \to A_{wa_3} A_{a_3w} A_{wa_3} A_{a_3f}.$$

Thus, the derivation tree takes the following form:



Transition A_{wa_3}**.** In the first transition from w to a_1 we have only one rule: $C, \$ \to \$$. In this rule, we pop \$ and then push it, so, in effect, nothing changes in the stack. Thus, in this case, the use of this rule is equivalent to $C, \varepsilon \to \varepsilon$. Since there is only one rule, according to the general algorithm, we need to add a fictitious rule to form a pair. We ignore the dollar signs that do not change:

This combination leads to the rule $A_{wa_3} \to \mathrm{C}A_{a_3a_3}\varepsilon$, i.e., $A_{wa_3} \to \mathrm{C}A_{a_3a_3}$. Here, the remaining transition between a_3 and a_3 does not include any additional steps, so we can use the rule $A_{a_3a_3} \to \varepsilon$. So, we get the rule $A_{wa_3} \to \mathrm{C}$.

Transition A_{a_3w}**.** For the transition between a_3 and w, we push C and then pop C:

So, we have a transition $A_{a_3w} \to \varepsilon A_{ww}\mathrm{L}$, i.e., $A_{a_3w} \to A_{ww}\mathrm{L}$. Here, the remaining transition between w and w does not include any additional steps, so we can use the rule $A_{ww} \to \varepsilon$. So, we get the rule $A_{a_3w} \to \mathrm{L}$.

Transition A_{wa_3}**.** The next transition A_{wa_3} is the same as the first transition from w to a_3, so it corresponds to the rule $A_{wa_3} \to \mathrm{C}$.

Transition A_{a_3f}**.** In the last transition A_{a_3f}, we first push C, then pop C:

So, we get the transition $A_{a_3f} \to \varepsilon A_{wf}\varepsilon$, i.e., $A_{a_3f} \to A_{wf}$. This takes care of the second C, so we get:

read			C		L	C				
state	s	w	a_3	w	w	a_3	w	f	f	f
stack		$	$	C $	$	$	C $	C $	$	

For the remaining transition from w to f, there is only one rule – in which the stack does not change. Since there is only one rule, we pair it with a fictitious trivial rule:

So, we get $A_{wf} \to \varepsilon A_{ff} \varepsilon$, i.e., $A_{wf} \to A_{ff}$. There are no rules for the transition A_{ff}, so we get $A_{ff} \to \varepsilon$ and thus, $A_{wf} \to \varepsilon$. So, the generation of CLC takes the following form: