Solutions to Test 2 for CS 3350 Automata
Fall 2023

1–3. Let us consider a finite automaton that checks whether a soup is hot or cold. Let us consider an alphabet consisting of two symbols: w (for "$w$arm up"), and ℓ (for "ℓeave on the table"). This automaton has two states:

- the final state h (for "$h$ot"), and
- the starting state c (for "$c$old").

Transitions are as follows:

- from the state h, w leads back to h, while ℓ lead to c;
- from the state c, w leads to h, while ℓ leads back to c.

This automaton accepts the word $w\ell w$.

1. Show how the general algorithm will produce a context-free grammar that generates all the words accepted by this automaton – and only words generated by this automaton.
2. On the example of the word $w\ell w$ accepted by this automaton, show how the tracing of acceptance of this word by the finite automaton can be translated into a generation of this same word by your context-free grammar.
3. Show how the word $w\ell w$ can be represented as $uvxyz$ according to the Pumping Lemma for context-free grammars.

Solution. According to the general algorithm, the corresponding grammar should have two variables H and C, and the following rules:

$$H \to wH, \quad H \to \ell C, \quad C \to wH, \quad C \to \ell C, \quad H \to \varepsilon$$

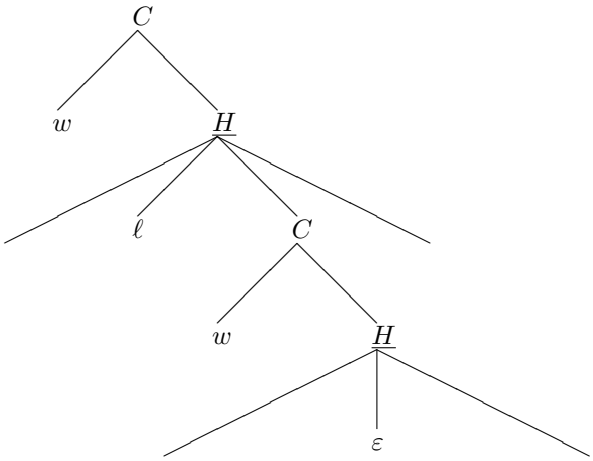
The word $w\ell w$ is accepted by the finite automaton as follows:

$$\begin{array}{ccccccccc} | & & w & & | & & \ell & & | & & w & & | \\ c & & & & h & & & & c & & & & h \end{array}$$

Thus, its derivation takes the following form:

$$\underline{C} \to w\underline{H} \to w\ell\underline{C} \to w\ell w\underline{H} \to w\ell w.$$

In terms of a tree, this can be represented as follows:



Here, $u = w$, $v = \ell w$, $x = y = z = \varepsilon$.

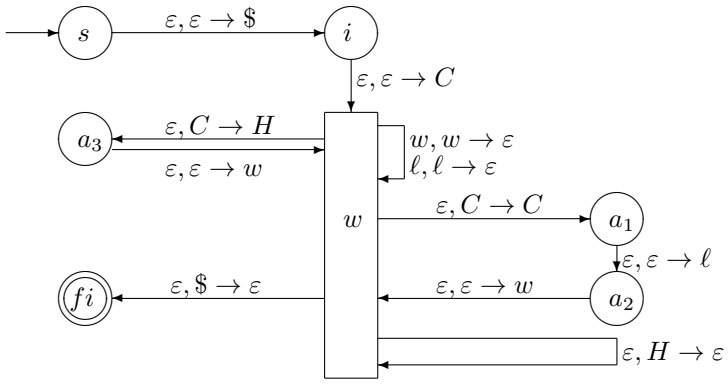
4-6. Let us consider the grammar with the starting variable C and the rules $C \to w\ell C$, $H \to \varepsilon$, and $C \to wH$.

4. Use a general algorithm to construct a (non-deterministic) pushdown automaton that corresponds to this grammar.

5. Show, step by step, how the word $w\ell w$ will be accepted by this automaton.

6. Transform this grammar into Chomsky normal form.

Solution to Problem 4.



Solution to Problem 5. The word $w\ell w$ is derived as follows:

$$\underline{C} \to w\ell\underline{C} \to w\ell w\underline{H} \to w\ell w.$$

So, we have the following acceptance by the pushdown automaton:

						w	ℓ			w		
s	i	w	a_1	a_2	w	w	w	a_3	w	w	w	fi
	\$	C	C	ℓ	w	ℓ	C	H	w	H	\$	
		\$	\$	C	ℓ	C	\$	\$	H	\$		
				\$	C	\$			\$			
					\$							

Solution to Problem 6.

Preliminary step. We add a new starting variable S_0 and a rule $S_0 \to C$:

$$C \to w\ell C, \quad H \to \varepsilon, \quad C \to wH, \quad S_0 \to C.$$

Step 0. The only inappropriate rule with right-hand side of length 0 is the rule $H \to \varepsilon$. So, we add the rule $C \to w$:

$$C \to w\ell C, \quad C \to wH, \quad S_0 \to C, \quad C \to w.$$

Step 1. We eliminate the rule $S_0 \to C$ by adding the rules $S_0 \to w\ell C$, $S_0 \to wH$, and $S_0 \to w$:

$$C \to w\ell C, \quad C \to wH, \quad C \to w, \quad S_0 \to w\ell C, \quad S_0 \to wH, \quad S_0 \to w.$$

Step 2. We add new variables V_w and V_ℓ, add new rules $V_w \to w$ and $V_\ell \to \ell$, and replace w and ℓ in rules in which the right-hand side has length 2 or more with, correspondingly, V_w and V_ℓ:

$$C \to V_wV_\ell C, \quad C \to V_wH, \quad C \to w, \quad S_0 \to V_wV_\ell C, \quad S_0 \to V_wH, \quad S_0 \to w,$$

$$V_w \to w, \quad V_\ell \to \ell.$$

Step 3. We replace the rule $C \to V_wV_\ell C$ with $C \to V_{w\ell}C$ and $V_{w\ell} \to V_wV_\ell$; similarly $S_0 \to V_wV_\ell C$ is replaced with $S_0 \to V_{w\ell}C$:

$$C \to V_{w\ell}C, \quad V_{w\ell} \to V_wV_\ell, \quad C \to V_wH, \quad C \to w, \quad S_0 \to V_{w\ell}C,$$

$$S_0 \to V_wH, \quad S_0 \to w, \quad V_w \to w, \quad V_\ell \to \ell.$$

This is already Chomsky normal form.

7-8. Show, step by step:

7. how the stack-based algorithm will transform the expression $1 - (2 - 3)$ into a postfix expression, and then

8. how a second stack-based algorithm will compute the value of this expression.

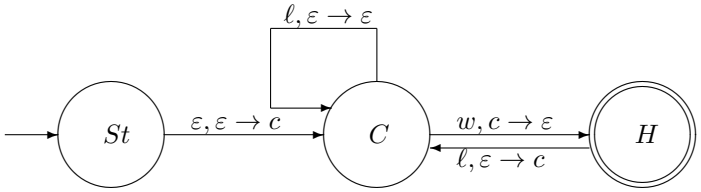
Solution. Let us show, step by step, how the above expression is transformed into the postfix form:

1	$-$	$($	2	$-$	3	$)$	
1			2		3	$-$	$-$
	$-$	$($		$-$		$)$	
		$-$		$($		$($	
				$-$		$-$	

Let us now show how this expression will be computed:

1	2	3	$-$	$-$
1	2	3	-1	2
	1	2	1	
		1		

9-10. Let us consider the following pushdown automaton:



This pushdown automaton accepts the word $w\ell w$. Use the general algorithm to show how this word will be generated in the corresponding context-free grammar.

Solution. Acceptance of the word $w\ell w$ by this pushdown automaton has the form:

		w	ℓ	w
St	C	H	C	H
	c		c	

We end up in the final state H, so first, we use the rule $S_0 \to A_{StH}$.

We have an intermediate stage with an empty stack, so we use the transitivity rule $A_{StH} \to A_{StH}A_{HH}$.

In the first transition, we push c, and then we pop c, so we have the two rules:

So, we form the rule $A_{StH} \to \Lambda A_{CC} w$, i.e., the rule $A_{StH} \to A_{CC} w$. The variable A_{CC} describes the transition to the same state without any changes, so we can use the rule $A_{CC} \to \varepsilon$.

In the second transition, we also push c and then pop c, but now we have a different pair of rules:

This leads to the rule $A_{HH} \to \ell A_{CC} w$. After that, we use the rule $A_{CC} \to \varepsilon$ that covers a trivial transition.

So, the derivation of the string $w\ell w$ takes the following form:

$$\underline{S_0} \to \underline{A_{StH}} \to \underline{A_{StH}} A_{HH} \to \underline{A_{CC}} w A_{HH} \to w \underline{A_{HH}} \to w\ell \underline{A_{CC}} w \to w\ell w.$$