Solutions to Test 2 for CS 3350 Automata
Fall 2022

1–3. Let us consider a finite automaton that checks whether a dog is hungry. Let us consider an alphabet consisting of two symbols: f (for "*f*ood"), and n (for "*n*o food"). This automaton has two states:

- the start state h (for "*h*ungry") and
- the final state s (for "*s*atisfied").

Transitions are as follows:

- from the state h, f leads to s, while n lead back to h;
- from the state s, every symbol leads back to s.

This automaton accepts the word nfn.

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 nfn 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 nfn 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 S, and the following rules:

$$H \to fS, \quad H \to nH, \quad S \to fS, \quad S \to nS, \quad S \to \varepsilon$$

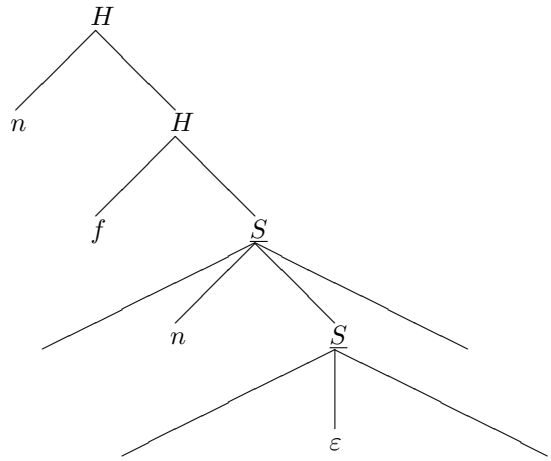
The word nfn is accepted by the finite automaton as follows:

$$\begin{array}{ccccccc} | & n & | & f & | & n & | \\ h & & h & & s & & s \end{array}$$

Thus, its derivation takes the following form:

$$\underline{H} \to n\underline{H} \to nf\underline{S} \to nfn\underline{S} \to nfn.$$

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



Here, $u = nf$, $v = n$, $x = y = z = \varepsilon$.

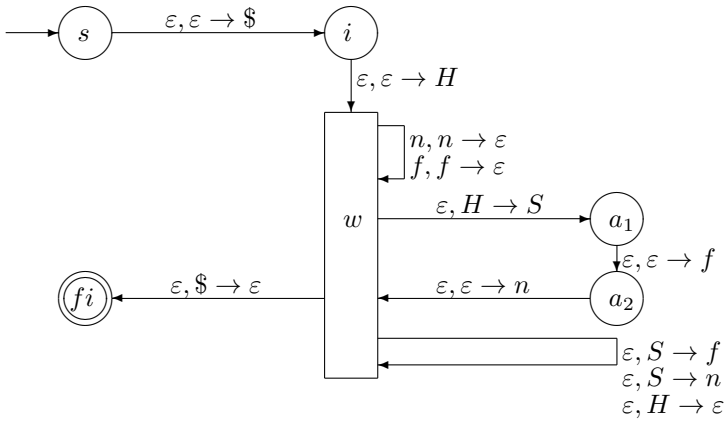
4-6. Let us consider the grammar with the starting variable H and the rules $H \to nfS$, $S \to n$, $S \to f$, and $H \to \varepsilon$.

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 nfn will be accepted by this automaton.

6. Transform this grammar into Chomsky normal form.

Solution to Problem 4.



Solution to Problem 5. The word nfn is derived as follows:

$$\underline{H} \to nf\underline{S} \to nfn;$$

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

						n	f		n	
s	i	w	a_1	a_2	w	w	w	w	w	fi
	\$	H	S	f	n	f	S	n	\$	
		\$	\$	S	f	S	\$	\$		
				\$	S	\$				
					\$					

Solution to Problem 6.

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

$$H \to nfS, \quad S \to n, \quad S \to f, \quad H \to \varepsilon, \quad S_0 \to H.$$

Step 0. The only inappropriate rules with right-hand side of length 0 is the rule $H \to \varepsilon$. So, we add the rule $S_0 \to \varepsilon$:

$$H \to nfS, \quad S \to n, \quad S \to f, \quad S_0 \to H, \quad S_0 \to \varepsilon.$$

Step 1. We eliminate the rule $S_0 \to H$ by adding the rule $S_0 \to nfS$:

$$H \to nfS, \quad S \to n, \quad S \to f, \quad S_0 \to \varepsilon, \quad S_0 \to nfS.$$

Step 2. We add new variables V_n and V_f, add new rules $V_n \to n$ and $V_f \to f$, and replace n and f in rules in which the right-hand side has length 2 or more with, correspondingly, V_n and V_f:

$$H \to V_nV_fS, \quad S \to n, \quad S \to f, \quad S_0 \to \varepsilon, \quad S_0 \to V_nV_fS, \quad V_n \to n, \quad V_f \to f.$$

Step 3. We replace the rule $H \to V_nV_fS$ with $H \to V_{nf}S$ and $V_{nf} \to V_nV_f$; similarly $S_0 \to V_nV_fS$ is replaced with $S_0 \to V_{nf}S$:

$$H \to V_{nf}S, \quad V_{nf} \to V_nV_f, \quad S \to n, \quad S \to f, \quad S_0 \to \varepsilon, \quad S_0 \to V_{nf}S,$$

$$V_n \to n, \quad V_f \to f.$$

This is already Chomsky normal form.

7-8. Show, step by step:

7. how the stack-based algorithm will transform the expression $a/b - c \cdot d$ into a postfix expression, and then

8. how a second stack-based algorithm will transform this postfix expression into quadruples.

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

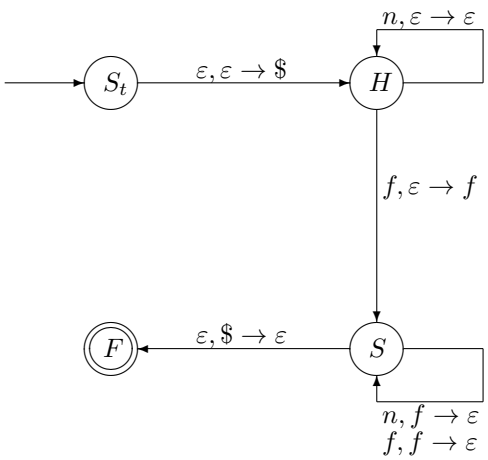
a	$/$	b	$-$	c	$*$	d		
a		b	$/$	c		d	$*$	$-$
	$/$		$-$		$*$			
					$-$			

Let us now show how this expression will be transformed into quadruples:

a	b	$/$	c	d	$*$	$-$
a	b	r_1	c	d	r_2	y
	a		r_1	c	r_1	
				r_1		

$$
\begin{array}{cccc}
/ & a & b & r_1 \\
* & c & d & r_2 \\
- & r_1 & r_2 & y.
\end{array}
$$

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



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

Solution. Acceptance of the word nfn by this pushdown automaton has the form:

		n	f	n	
S_t	H	H	S	S	F
	\$	\$	f	\$	
			\$		

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

Then, the first symbol we push is \$, it is popped at the end, so we have the two rules:

So, we form the rule $A_{S_tF} \to \Lambda A_{HS} \Lambda$, i.e., the rule $A_{S_tF} \to A_{HS}$. This takes care of the dollar sign. So now, we can ignore the dollar sign and only consider what happens on top.

We see that in the transition from H to S, there is an intermediate state with an empty stack, so we need to use the transitivity rule $A_{HS} \to A_{HH} A_{HS}$.

The first variable A_{HH} describes the transition right after pushing the dollar sign. In this transition, we read n, and we remain in the state W without

pushing anything or popping anything. Since we are not pushing anything, we do not have a natural transition to pair with this transition. So, according to the general algorithm, we pair it with the trivial transition, when we do not see anything, do not push anything, do not pop anything, and stay in the same state:

This leads to the rule $A_{HH} \to nA_{HH}\Lambda$, i.e., the rule $A_{HH} \to nA_{HH}$. After that, we use the rule $A_{HH} \to \varepsilon$ that covers a trivial transition.

In the transition corresponding to the second part A_{HS}, the next symbol we push (and later pop) is f:

So, we use the rule $A_{HS} \to fA_{SS}n$. Finally, we apply the rule $A_{SS} \to \varepsilon$.

So, the derivation of the string nfn takes the following form:

$$\underline{S_t} \to \underline{A_{S_tF}} \to \underline{A_{HS}} \to \underline{A_{HH}}A_{HS} \to n\underline{A_{HH}}A_{HS} \to n\underline{A_{HS}} \to nf\underline{A_{SS}}n \to nfn.$$