Solutions to Test 2 for CS 3350 Automata
Spring 2024

1–3. Let us consider a finite automaton that checks whether a test is challenging or too easy. Let us consider an alphabet consisting of two symbols: a (for "*a*dd a problem"), and d (for "*d*elete a problem"). This automaton has two states:

- the starting state c (for "*c*hallenging"), which is also final, and
- the state t (for "*t*oo easy").

Transitions are as follows:

- from the state c, d leads to t, while a lead back to c;
- from the state t, a leads to c, while d leads back to t.

This automaton accepts the word daa.

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 daa 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 daa 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 C and T, and the following rules:

$$C \to dT, \quad C \to aC, \quad T \to aC, \quad T \to dT, \quad C \to \varepsilon$$

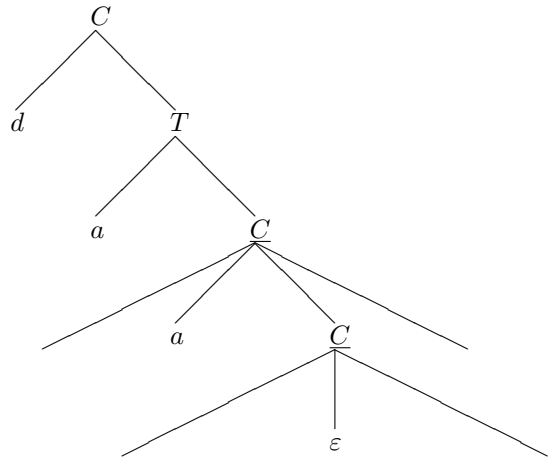
The word daa is accepted by the finite automaton as follows:

$$\begin{array}{ccccccc} | & d & | & a & | & a & | \\ c & & t & & c & & c \end{array}$$

Thus, its derivation takes the following form:

$$\underline{C} \to d\underline{T} \to da\underline{C} \to daa\underline{C} \to daa.$$

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



Here, $u = da$, $v = a$, $x = y = z = \varepsilon$.

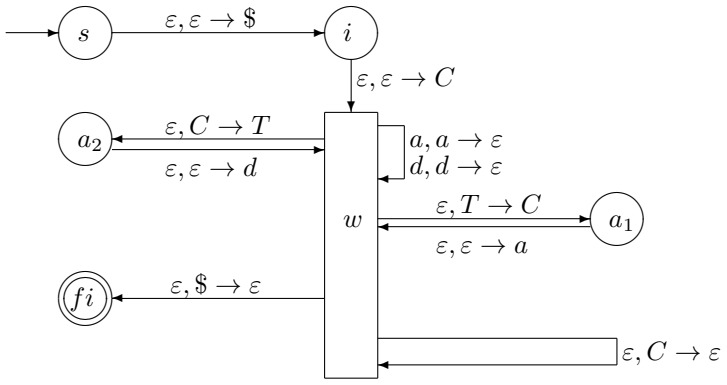
4-6. Let us consider the grammar with the starting variable C and the rules $C \to dT$, $C \to \varepsilon$, and $T \to aC$.

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 da will be accepted by this automaton.

6. Transform this grammar into Chomsky normal form.

Solution to Problem 4.



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

$$\underline{C} \to d\underline{T} \to da\underline{C} \to da.$$

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

					d			a		
s	i	w	a_2	w	w	a_1	w	w	w	fi
	\$	C	T	d	T	C	a	C	\$	
		\$	\$	T	\$	\$	C	\$		
				\$			\$			

Solution to Problem 6.

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

$$C \to dT, \quad C \to \varepsilon, \quad T \to aC, \quad S_0 \to C.$$

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

$$C \to dT, \quad T \to aC, \quad S_0 \to C, \quad T \to a, \quad S_0 \to \varepsilon.$$

Step 1. We eliminate the rule $S_0 \to C$ by adding the rule $S_0 \to dT$:

$$C \to dT, \quad T \to aC, \quad T \to a, \quad S_0 \to \varepsilon, \quad S_0 \to dT.$$

Step 2. We add new variables V_a and V_d, add new rules $V_a \to a$ and $V_d \to d$, and replace a and d in rules in which the right-hand side has length 2 or more with, correspondingly, V_a and V_d:

$$C \to V_dT, \quad T \to V_aC, \quad T \to a, \quad S_0 \to \varepsilon, \quad S_0 \to V_dT, \quad V_a \to a, \quad V_d \to d.$$

Step 3. We do not have any rules with right-hand side of length 3 or more, so we already have the Chomsky normal form.

7-8. Show, step by step:

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

8. how a second stack-based algorithm will compute the value of this expression (use integer operations in Java).

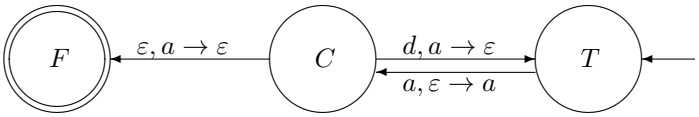
Solution. Let us show, step by step, how the above expression is transformed into the postfix form:

3	/	(	2	/	1	)	
3			2		1	/	/
	/	(		/		)	
		/		(		(	
				/		/	

Let us now show how this expression will be computed:

3	2	1	/	/
3	2	1	2	1
	3	2	3	
		3		

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



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

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

	a	d	a	
T	C	T	C	F
	a		a	

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

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

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

So, we form the rule $A_{TT} \to aA_{CC}d$. 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 a and then pop a, but now we have a different pair of rules:

This leads to the rule $A_{TC} \to aA_{CC}\Lambda$, i.e., $A_{TC} \to aA_{CC}$. 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:

$$\underline{S_0} \to \underline{A_{TF}} \to \underline{A_{TT}}A_{TF} \to a\underline{A_{CC}}dA_{TF} \to ad\underline{A_{TF}} \to ada\underline{A_{CC}} \to ada.$$