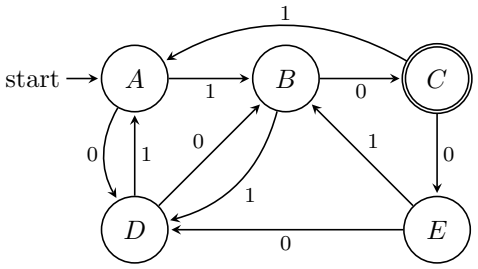
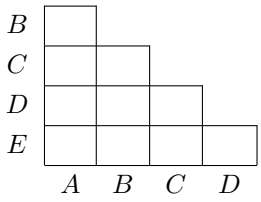
Nathan Bickel, Jackson Ginn, Jack Hyatt, Chance Storey
CSCE 355: Section 001
Professor: Dr. Fenner
March 14, 2024

CSCE 355 Homework 4

Problem 1 Consider the DFA N (below left) over the alphabet $\{0, 1\}$:





(a) Fill in the distiguishability table to the right with X in each entry corresponding to a pair of distinguishable states.

(b) Draw the minimal DFA equivalent to N.

(a) We fill in the distinguishability table for the reasons listed:

B	X			
C	X	X		
D	X	X	X	
E		X	X	X
	A	B	C	D

1. C is distinguishable from all other states since it is rejecting.

2. A and B are distinguishable because $\delta(A, 0) = D$ and $\delta(B, 0) = C$ are distinguishable (from **1**).

3. A and D are distinguishable because $\delta(A, 1) = B$ and $\delta(D, 1) = A$ are distinguishable (from **2**).

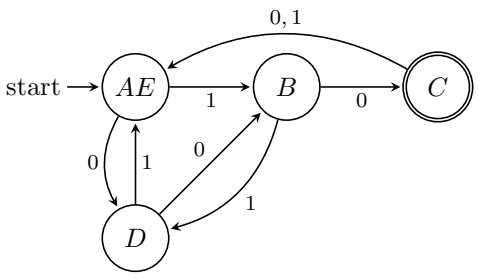
4. B and D are distinguishable because $\delta(B, 0) = C$ and $\delta(D, 0) = B$ are distinguishable (from **1**).

5. B and E are distinguishable because $\delta(B, 0) = C$ and $\delta(E, 0) = D$ are distinguishable (from **1**).

6. D and E are distinguishable because $\delta(D, 0) = B$ and $\delta(E, 0) = D$ are distinguishable (from **4**).

However, A and E are not distinguishable because $\delta(A, 0) = D$ and $\delta(E, 0) = D$ are not distinguishable and $\delta(A, 1) = B$ and $\delta(E, 1) = B$ are not distinguishable.

(b) We draw the equivalent minimal DFA:

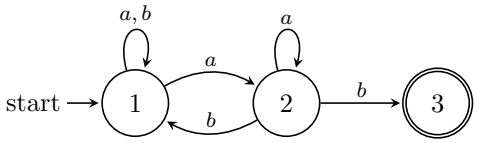


Problem 2 Using the sets-of-states method described in class or in the book, convert the following NFA N (no ϵ-moves) to an equivalent DFA D:

	a	b
$\to 1$	$\{1,2\}$	$\{1\}$
2	$\{2\}$	$\{1,3\}$
$*3$	$\emptyset$	$\emptyset$

Only give states of D that are reachable from its start state, and label each state of D with the states of N that it contains. Include all dead states (if there are any), and do not merge indistinguishable states.

To visualize it easier, here is the NFA diagram



So then the equivalent DFA is:

	a	b
$\to 1$	$\{12\}$	$\{1\}$
12	$\{12\}$	$\{13\}$
$*13$	$\{12\}$	$\{1\}$

Problem 3 Consider the regex $r := (a+b)^*(b+c)^*$ over the alphabet $\Sigma := \{a, b, c\}$. Find a regex r' such that $L(r') = \overline{L(r)}$, the complement of $L(r)$ in Σ^*. Do this as follows:

(a) Convert r to an equivalent ϵ-NFA N. (You may contract ϵ-transitions provided it is sound to do so.)

(b) Remove ϵ-transitions from N to get an equivalent NFA N' using the method described in class and the course notes (Method 2).

(c) Using the sets-of-states construction described in class, convert N' into an equivalent DFA D. (Only include states of D reachable from its start state.)

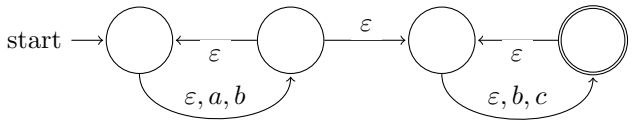
(d) (Optional) Minimize D by merging indistinguishable states, if any.

(e) Form the complementary DFA $D' := \neg D$.

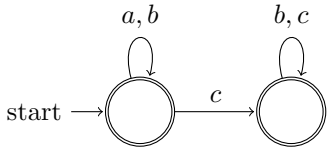
(f) Starting with a clean ϵ-NFA equivalent to D', find the equivalent regex r' by the state elimination method described in class.

As far as anyone knows, there is no general procedure for negating a regex that is significantly faster than going through the steps above. The same holds for finding a regex for the intersection of two languages given by regexes, which would involve the product construction on two DFAs.

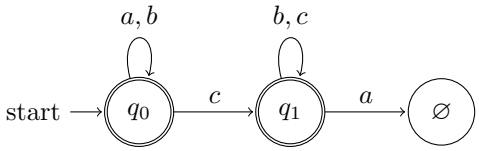
(a)



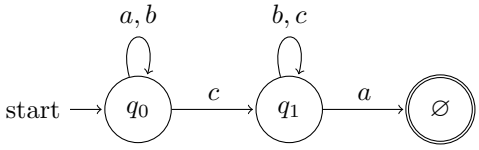
(b)



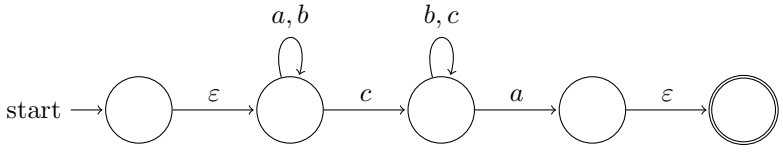
(c)

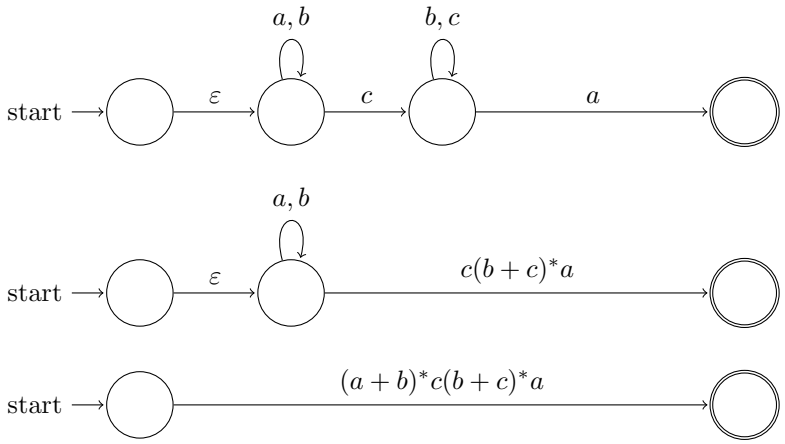


(e)



(f)





This yields the final result as $r' = (a+b)^*c(b+c)^*a$.

Problem 4 For any string $w \neq \epsilon$, the *principal suffix* of w is the string resulting by removing the first symbol from w. We will denote this string by $ps(w)$. For any language L, define $ps(L) := \{ps(w) : w \in L \wedge w \neq \epsilon\}$. Show that if L is regular, then $ps(L)$ is regular. (The underlying alphabet is arbitrary.)

First we will prove that the language of prefixes (all but the last letter) of a regular language is regular, denoted by $pre(L)$.
This can be shown quite easily by taking any automaton recognizing L, and making any state accepting if it has any move to an already accepting state, which pretty clearly recognizes the prefixes of words of L.
Also, note that we proved in class that the reversal of a regular language is regular.
Now since $ps(L) = pre(L^R)^R$, we have that $ps(L)$ must be regular by the closure properties. □

Problem 8 (Exercise 4.1.1 (selected items)): Prove that the following are not regular languages. For each, show that the given language is not pumpable. [You may use the template given above.]

(a) The set of strings of balanced parentheses. These are the strings of characters "(" and ")" that can appear in a well-formed arithmetic expression.

(b) $\{0^n 1 0^n \mid n \geq 1\}$.

(c) $\{0^n 1^m 2^n \mid n \text{ and } m \text{ are arbitrary integers}\}$.

(d) $\{0^n 1^{2n} \mid n \geq 1\}$.

(a) Given any $p > 0$,
Let $s := ({}^p)^p$. $|s| = 2p \geq p, s \in L$ Now for any x, y, z with $xyz = s$ and $|xy| \geq p$ and $|y| > 0$,
let $i = 0$.
Then $xy^iz = xy^0z = xz \notin L$, which can be seen as follows: Since $|xy| \leq p$, it must be that x and y consist entirely of the character (, and so $y = ({}^m$ for some m, and we further have $m \geq 1$ because $|y| > 0$. But then $xz = ({}^{p-m})^p$, and so because $p - m \neq p$ the string xz does not have the same amount of (and), and so cannot be balanced, and thus $xz \notin L$.

(b) Given any $p > 0$,
Let $s := 0^p 1 0^p$. $|s| = 2p + 1 \geq p, s \in L$ Now for any x, y, z with $xyz = s$ and $|xy| \geq p$ and $|y| > 0$,
let $i = 0$.
Then $xy^i z = xy^0 z = xz \notin L$, which can be seen as follows: Since $|xy| \leq p$, it must be that x and y consist entirely of 0's, and so $y = 0^m$ for some m, and we further have $m \geq 1$ because $|y| > 0$. But then $xz = 0^{p-m} 1 0^p$, and so because $p - m \neq p$ the string xz does not have the same amount of 0's on either side of the 1,and thus $xz \notin L$.

(c) Given any $p > 0$,
Let $s := 0^p 1^1 2^p$. $|s| = 2p + 1 \geq p, s \in L$ Now for any x, y, z with $xyz = s$ and $|xy| \geq p$ and $|y| > 0$,
let $i = 0$.
Then $xy^i z = xy^0 z = xz \notin L$, which can be seen as follows: Since $|xy| \leq p$, it must be that x and y consist entirely of 0's, and so $y = 0^m$ for some m, and we further have $m \geq 1$ because $|y| > 0$. But then $xz = 0^{p-m} 1^1 2^p$, and so because $p - m \neq p$ the string xz does not have the same amount of 0's and 2's, and thus $xz \notin L$.

(d) Given any $p > 0$,
Let $s := 0^p 1^{2p}$. $|s| = 3p \geq p, s \in L$ Now for any x, y, z with $xyz = s$ and $|xy| \geq p$ and $|y| > 0$,
let $i = 0$.
Then $xy^i z = xy^0 z = xz \notin L$, which can be seen as follows: Since $|xy| \leq p$, it must be that x and y consist entirely of 0's, and so $y = 0^m$ for some m, and we further have $m \geq 1$ because $|y| > 0$. But then $xz = 0^{p-m} 1^{2p}$, and so because $p - m \neq p$ the string xz does not have twice as many 1's as 0's, and thus $xz \notin L$.

Problem 9 Consider the following grammar generating the language of strings of well-balanced parentheses:

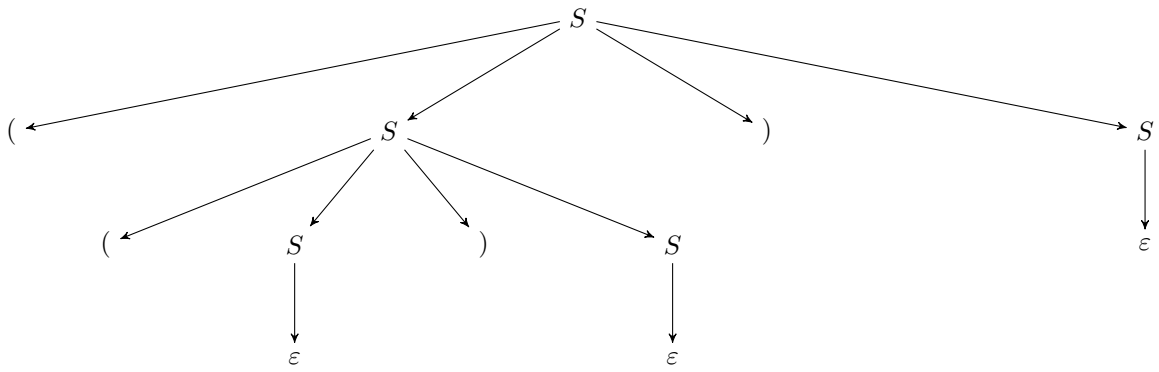
$$S \to (S)S \mid \epsilon$$

Give a leftmost derivation of the string (()) and a rightmost derivation of the string ()(()()). Also, give a parse tree yielding each string (two parse trees in all).

The leftmost derivation of (()) is:

$$\underline{S} \to (\underline{S})S \to ((\underline{S})S)S \to (()\underline{S})S \to (())\underline{S} \to (())$$

Shown is the parse tree:

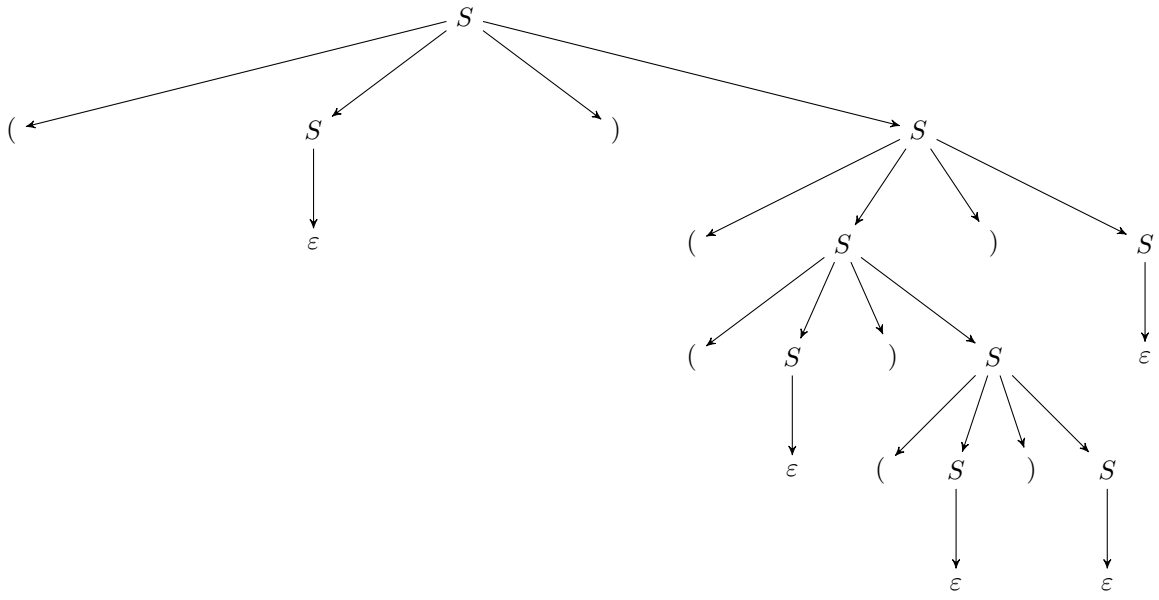


The rightmost derivation of ()(()()) is:

$$\underline{S} \to (S)\underline{S} \to (S)(S)\underline{S} \to (S)(\underline{S}) \to (S)((S)\underline{S}) \to (S)((S)(S)\underline{S})$$

$$\to (S)((S)(\underline{S})) \to (S)((\underline{S})()) \to (\underline{S})(()()) \to ()(()())$$

Shown is the parse tree:



Problem 10 Describe briefly in words the language $L(G)$, where $G = (\{A, B\}, \{a, b, c\}, A, P)$ is a context-free grammar and the productions in P are

$$A \to aAc \mid B$$
$$B \to \epsilon \mid Bc$$

This is the language of strings with characters a, c where there is some number of a's followed by an equal or greater amount of c's.

Problem 11 Give a context-free grammar for the language $\{a^\ell b^m c^n \mid \ell \leq m \text{ or } m \leq n\}$. (Note that the connective is "or," not "and.")

Since the language is clearly the union of two other languages, we just need to make two grammars and then add a production to lead to one of the grammars.
The first language is $L_1 = \{a^\ell b^m c^n \mid \ell \leq m\}$ and the second language is $L_2 = \{a^\ell b^m c^n \mid m \leq n\}$.
The grammar for the first language is

$$\begin{aligned} T &\to Tc \mid T' && \text{add as many c's as you want} \\ T' &\to aT'b \mid T'b \mid \epsilon && \text{add b's and a's, or just b's, then stop} \end{aligned}$$

The grammar for the second language is

$$\begin{aligned} U &\to aU \mid U' && \text{add as many a's as you want} \\ U' &\to bU'c \mid U'c \mid \epsilon && \text{add b's and c's, or just c's, then stop} \end{aligned}$$

So the full grammar for the original language is

$$\begin{aligned} S &\to T \mid U \\ T &\to Tc \mid T' \\ T' &\to aT'b \mid T'b \mid \epsilon \\ U &\to aU \mid U' \\ U' &\to bU'c \mid U'c \mid \epsilon \end{aligned}$$

Problem 12 Consider the grammar of Exercise 5.1.8:

$$S \to aSbS \mid bSaS \mid \epsilon$$

Show that *abba* is generated by the grammar but *aba* is *not* generated by the grammar. (This is a special case of the full exercise.)

We first show that *abba* is generated by the grammar with the following left-most derivation:

$$S \Rightarrow aSbS \Rightarrow abS \Rightarrow abbSaS \Rightarrow abbaS \Rightarrow abba,$$

using Productions 1, 3, 2, 3, and 3, respectively.

Next, we observe that each production adds either 0 or 2 non-terminals, both of which are even. Thus, every string generated by the grammar has even length, since the sum of even numbers is even. So *aba* is not generated by the grammar since it has odd length. □