

Homework 3

Q1 [10 pts] P.131 Ex.4.1.1: b), e)

b)

Let p be the pumping-lemma constant. Pick $w = (')^p$. String w contains p 's, which are followed by p 's. Then when we write $w = xyz$, we know that $|xy| \leq p$, and therefore y consists of only 's. Thus, $xyyz$, which must be in L if L is regular, consists of more than p 's, followed by exactly p 's. That string is not in L , so we contradict the assumption that L is regular.

f)

Let p be the pumping-lemma constant. Pick $w = 0^p 1^{2p}$. Then when we write $w = xyz$, we know that $|xy| \leq p$, and therefore y consists of only 0's. Thus, $xyyz$, which must be in L if L is regular, consists of more than p 0's, followed by exactly $2p$ 1's. That string is not in L , so we contradict the assumption that L is regular.

Q2 [10 pts] P.132 Ex.4.1.2: c)

c)

Let p be the pumping-lemma constant. Pick a string 0^{2^p} . Then when we write it as xyz , we know that $|xy| \leq p$, and therefore y consists of only 0's. Let's assume that $|y| = m$, thus, $xy^k z$, which must be in L if L is regular, consists of $2^p + m * (k-1)$ 0's. Clearly, for all $k \geq 0$, the total number of 0's cannot always be the power of 2. This string is not in L , so we contradict the assumption that L is regular.

Q3 [15 pts] P.147 Ex.4.2.4: b), c)

b) **wrong**

If $L = \{ a, aab, baa \}$, then $a \setminus L = \{ \epsilon, ab \}$, then left side = $a(a \setminus L) = \{ a, aab \} \neq L$

c) **true**

By doing the concatenation of L and a , we get a new language L' , which is the set of strings wa such that w is in L . Then we go ahead to get the quotient of L' and a , which by definition is the set of strings w such that wa is in L' . Obviously this leads us back to L .

Q4 [10 pts] P.155 Ex.4.3.3

Given a regular language L , we can construct a corresponding DFA for it, say it is A . By reversing the non-accepting states and the accepting states of A , we get a DFA A' which describe the complement of language L , which takes $O(n)$ time if A has at most $O(n)$ states and transitions. It's clear to see the problem whether L contains all strings over its alphabet is equivalent to the problem whether the complement of L is empty. Section 4.3.2 has given us an algorithm to test emptiness of regular languages, so basically we've done.

Q5 [15 pts] P.165 Ex.4.4.2

a) The table of distinguishabilities

B	X							
C	X	X						
D		X	X					
E	X		X	X				
F	X	X		X	X			
G		X	X		X	X		
H	X		X	X		X	X	
I	X	X		X	X		X	X
	A	B	C	D	E	F	G	H

b) the minimum-state equivalent DFA

	0	1
->ADG	BEH	BEH
BEH	CFI	CFI
*CFI	ADG	BEH

