Final Exam Questions 16.838

Answers to be returned by date and time specified by Prof. Leveson in class.

From your textbook, please answer the following questions: 1. p. 475 (10)

Answer is 5/663:

Number of total hands combin(52,5)=2598960

The number of hands with the two of diamonds and the three of spades is 50 choose 3 or 19600

19600/2598960=5/663

2. p. 491 (2)

Assuming that the question just means that rolling a 3 is twice as likely as the individual probabilities of the other face values of the die, the answer is 2/7 for rolling a 3 and 1/7 for each of the other outcomes.

P(1)+P(2)+P(3)+P(4)+P(5)+P(6)=1

Assigning x to each probability except for 3, which has a probability of 2x:

X+x+2x+x+x+x=1…7x=1

X=1/7, P(3)=2/7, all other are 1/7

3. p. 501 (2)

Answer is 5/9:

P(E given F)=P(F|E)P(E)/(P(F|E)P(E)+P(F|notE)P(notE))

=P(F|E)P(E)/P(F)

=(5/8\*2/3)/3/4

=5/9

4. p. 518 (4)

Answer is 6 Heads:

For Bernoulli trials, E(X)= n\*P(X)

=10\*.6

=6

5. p. 518 (48)

On page 518 there is no number 48… assuming you meant page 520 number 48, the answer is m/n:

Let the Random Variable X be the sum from 1 to m of X\_i denote the number of balls that fall into the first bin, where X\_i=1 if the ith ball goes to the first bin and 0 otherwise. E(X\_i)=1/n because there are n bins. Going back to that sum I first described, I can sum across expected values for each of the X\_i instances of the random variable. Therefore the expected value of X is the sum of the expected values of X\_i; since I summed them 1 to m, I have m\*1/n or m/n.

6. p. 608 (3)

a. not Reflexive, not Symmetric, not antisymmetric, and transitive

b. Reflexive, Symmetric, not antisymmetric, and transitive

c. not Reflexive, Symmetric, not antisymmetric, and not transitive

d. not Reflexive, not Symmetric, antisymmetric, and not transitive

e. Reflexive, Symmetric, antisymmetric, and transitive

f. Not Reflexive, Not Symmetric, Not antisymmetric, and Not transitive

7. p. 682 (4)

There is no number 4 on page 682… assuming you meant number 4 on page 683:

This is a multigraph. It’s undirected, the aren’t any loops, and for some nodes there are multiple edges.

8. p. 682 (8)

There is no number 8 on page 682. Assuming you meant 8 on page 683:

This is a directed multigraph. It’s directed, it has some loops, and it has multiple edges. Using Table 1, it’s a multigraph.

9. p. 699 (2)

Number of vertices=5

Number of edges=13

Deg(a)=6

Deg(b)=6

Deg(c)=6

Deg(d)=5

Deg(e)=3

There are no isolated or pendant vertices in this graph

10. p. 699 (8)

There are 4 vertices

There are 8 edges

In degrees:

A: 2

B:3

C:2

D:1

Out degrees:

A:2

B:4

C:1

D:1

11. p. 791 (2)

A Tree

B Tree

C Not Tree

D Tree

E Not Tree  
F Tree

12. p. 791 (4)

A- a

B- a, b, d, e, g, h, i, o

C- c, f, j, k, l, m, n, p, q, r, s

D- none

E- d  
F- p  
G- a, b, g

H- e, f, g, j k, l, m

13. p. 805 (4)

A- 3

B- 3

C- 3

D- 3

Additionally, please answer the questions below:

1. What is the language generated by the grammar with productions 𝑆 → 𝑆𝐴, 𝑆 → 0, 𝐴 → 1𝐴, and 𝐴 → 1, where S is the start symbol? Note that 𝑉 = {𝐴, 0, 1, 𝑆}, terminal elements are {0, 1}, and the start symbol is given by 𝑆.

The language generated will have the following strings: L={01, 011, 0111, 01111…}

This is the set of all strings that start with 0 and then have some consecutive finite number of ones.

1. Find a grammar *G* for the set {02𝑛1𝑛, 𝑛 ≥ 0}.

G= V, T, S, P.

Our terminal elements are {0, 1}, the start symbol is given by *S*, and our vocabulary V is { 0, 1, 𝑆}. Our production rules are as follows:

𝑆 → 00*S* 1

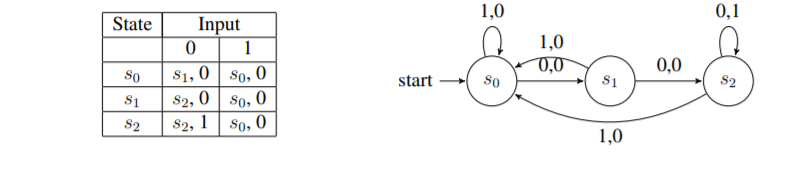
𝑆 → 𝜀 (…. Because n can be 0!)

Our grammar G is made up of these four elements.

1. Let G be the phrase-structure grammar with vocabulary 𝑉 = {𝐴, 𝐵, 0, 1, 𝑆}, terminal elements {0, 1}, and start symbol 𝑆. For each of the set of productions below, determine the type of grammar. That is, determine whether G is a regular grammar (type 3), context free grammar (type 2), context sensitive grammar (type 1), or phrase structured grammar (type 0). Justify your reasoning and show your work.
   * 𝑃 = {𝑆 → 0𝐵, 𝐵 → 1𝐴, 𝐴 → 0𝐵, 𝐵 → 0}
   * This is a type 3 grammar. Because we have a production rule in the form of X→aY, where X,Y∈N and a∈T. Here A and B are not terminal, but 0 and 1 are. By definition this is a type 3 grammar.
   * 𝑃 = {𝑆 → 10𝑆𝐴𝐵, 𝑆 → 01𝑆𝐵𝐴, 𝐴 → 0, 𝐵 → 1, 𝑆 → 𝜀}

* This is a type 1 grammar. First off, we have 𝑆 → 𝜀, so from my perspective as a student that’s a hint that it’s type 1. Then, looking at the first two production rules, we can see that the language is context-sensitive. For that reason, it must be type 1.

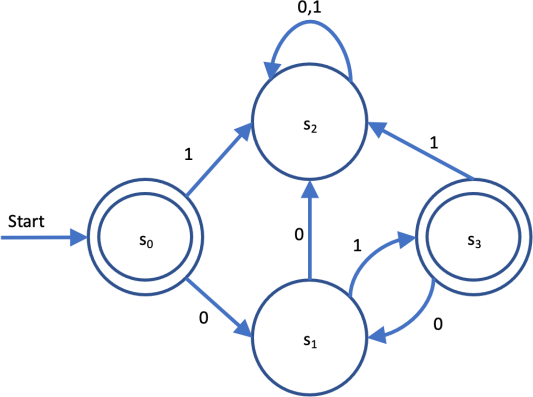
1. Construct a finite-state machine with output that produces a 1 if and only if the last 3 input bits read are 0's (i.e., input contained 3 consecutive zeros).



1. Find the language (i.e., all strings) recognized by the following deterministic finite automaton (DFA):

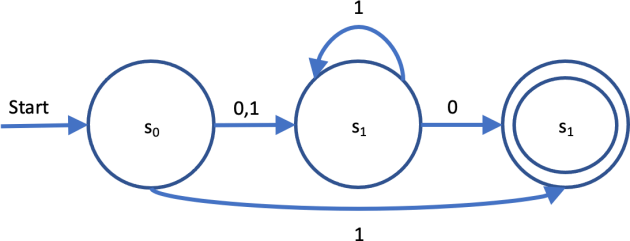
The language recognized by this DFA is as follows: L(M)= { 01,0101,010101,…}. This can be better expressed as (01)\*.

The final states are states 0 and 3. State 0 to itself is given only by ε. The strings that take 0 to 3 are any number of “01” pairs. Hence why the language recognized by this DFA is the union of these two…

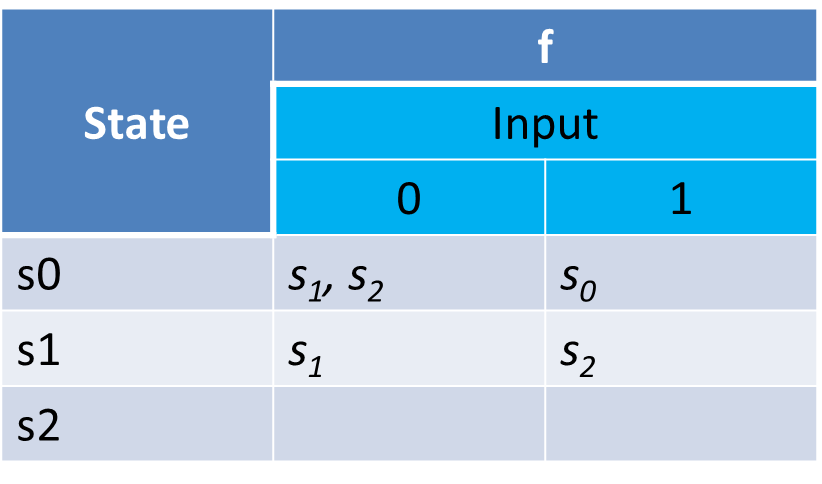
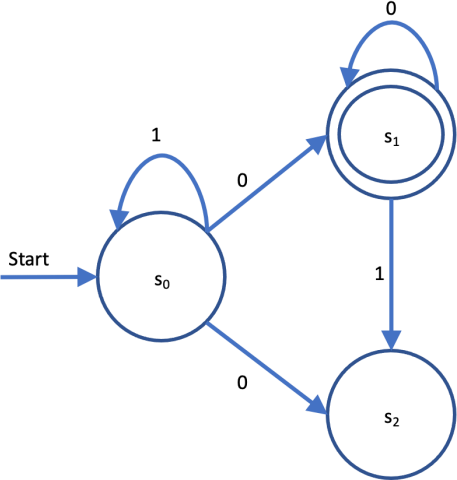


1. Find the language (i.e., all strings) recognized by the following non-deterministic finite automaton (NFA):

The language recognized by this NFA is as follows: L(M)= {1, 01\*0, 11\*0} where n is greater than or equal to 0.

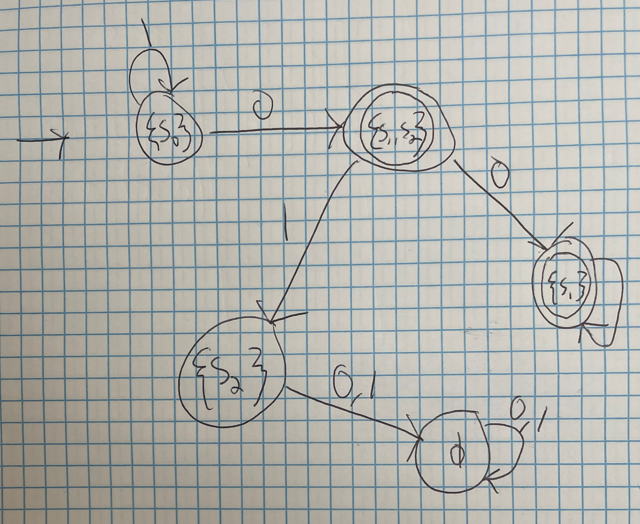


1. Find a deterministic finite state automaton (DFA) equivalent to the following nondeterministic finite state automaton:



Here is the state diagram. I is 0 and 1, theta is s0, F is s1.

My DFA is below. The language recognized by this DFA is the same as the NFA above. Because this is true, we know that they are equivalent.



For the following Turing machine *T*, given the tape below, determine whether *T* halts. If so, what does the final tape read? Is this string in the language of *T*? Note that T begins in the initial position (i.e., the first nonblank entry from the left) in the initial state s0 and proceeds to read the tape below.

This is the tape string I used, corrected from your email:

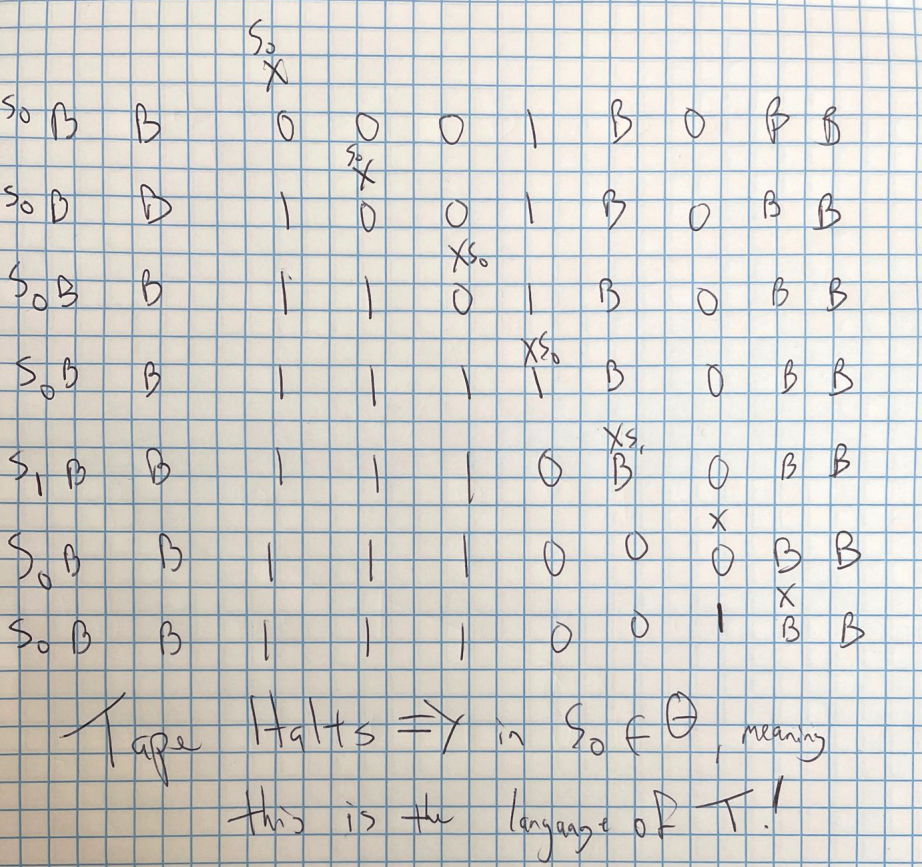
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B | B | 0 | 0 | 0 | 1 | B | 0 | B | B |

*T* = *(S, I, f,* 𝜃*)* is given by:

* 𝑆 = {𝑠0, 𝑠1, 𝑠2, 𝑠3}
* 𝐼 = {0, 1, 𝐵}
* 𝜃 = {𝑠0}, and
* 𝑓: 𝑆 × 𝐼 → 𝑆 × 𝐼 × {𝐿, 𝑅} is defined by
  + (*s*0, 0, *s*0, 1, *R*),
  + (*s*0, 1, *s*1, 0, *R*),
  + (*s*1, 1, *s*2, 1, *R*), and
  + (*s*1, *B*, *s*0, 0, *R*)

The tape does indeed halt, and it halts in S0, which is an element of theta our accepting or final states. Because it halts and is in a final state, this means that this string is recognized by *T*. It is of the language o

Here is my work by hand (extra points for stellar penmanship?) of how *T* will operate, including the intermediate steps.

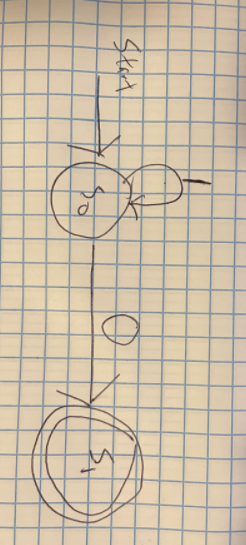


Which of the following languages are regular? Show your work (i.e., construct a DFA or NFA that accepts that language, or give a proof why no DFA accepts that language).

* + 𝐿 = {0𝑛|𝑛 = 1, 2, 3, , …}

Drawing a really fast DFA, we know that this language is regular because a DFA accepts it.

Again, extra points for artistry:



* + 𝐿 = {0𝑛|𝑛 = 2𝑘, 𝑘 = 1, 2, 3, …}

This is not regular.

Let’s use the pumping lemma and a proof by contradiction.

Assume L is regular, and that L has a pumping length of *p*.

Take a string such that *p*=4:

S=016… from (0k^2)=04^2

Let’s divide this S into 3 parts (using components of pumping lemma):

xyiz which is an element of L and i is greater than 0.

|xy|<=*p*

|x|<=1

x is 000, y is 0^1, and z is 0^12 for i=1.

When we move to i=2, S will be (0k^2+1) which is not contained in L.

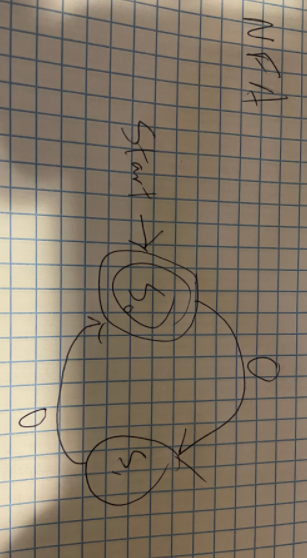
This is because 017 is not in L.

This invalidates our assumption that L is regular.   
Therefore, L is not regular.

* + 𝐿 = {0𝑛|𝑛 = 2𝑘, 𝑘 = 1, 2, 3, …}

In this series n will look something like 2, 4, 6, 8, 10… we can generate this language by the regular expression (00)\*. Because I have a regular expression that can generate it, I know the language is regular. For bonus points, I can also generate a state machine that recognizes this language, which means that it is regular.

The NFA looks like this:



Because we know we have a regular expression and an NFA that recognizes this language, we know that it is regular. Again, I’d like to request bonus points for the arts and crafts. The more of these things I draw by hand the more I’m impressed with my own handy work!

* + 𝐿 = {0𝑛1𝑚|𝑚, 𝑛 = 1, 2, 3 …}

This is a regular language. We know that it is a regular language because it can be generated by a regular grammar. There’s also a regular expression that can generate it: (0\*1\*).

Consider the Grammar G with alphabet V ={S,A,0,1}; Terminals {0,1}, and the productions S→0S, S→1A, S→1, A→1A, A→1, and S→*𝜆.* This grammar G is by all means regular.

Consider the rules for a regular grammar: *w*1 = *A* and *w*2 = *aB* or *w*2 = *a*, where *A* ∊ *N*, *B* ∊ *N*, and *a* ∊ *T*; or *w*1 = *S* and *w*2 =

This grammar follows type 3 rules for its productions. This grammar can also be used to generate the language *L* in question; any language generated by a regular grammar is called a regular language.

The productions are from example 6 on page 889, and the concept that this is regular is listed on page 890 in example 8.

* + 𝐿 = {0𝑛1𝑚|𝑚 ≠ 𝑛 ⋀ 𝑚, 𝑛 = 1, 2, 3 …} [Hint: Regular languages are closed under the set difference operator ‘\’. That is, if L1 is regular and L2 is regular, then L3 = L1\L2 must be regular.]

This language is not regular.

Let’s use 0𝑛1n as our L1. We know from class that this language is not regular. There are also many proofs in many forms both in the book and on the interwebs as to why it’s not regular.

Since we have proven that 0𝑛1m is regular in the last part of this problem, let’s call it L2.

Let’s consider a third language L1’… this is the language 0i1j where i and j are equal plus all strings not in the form 0i1j.

If we assume L1 is regular (which we know is not), and then we take L2 that we know to be regular and perform the following operation:

L3 = L2\L1’, where L3 is the language given in the problem.

Since regular languages are closed under this set difference operator, we must assume that because L1 and L2 are regular, L3 is also regular.

However, we have a contradiction! L1 is absolutely not regular (thanks extra office hours time!), therefore we know that L3, the language {0𝑛1𝑚|𝑚 ≠ 𝑛 ⋀ 𝑚, 𝑛 = 1, 2, 3 …}, is not a regular language.