§9.3(a) #2, 11, 20, p750-

(b) #25a, 34, 68, p750-

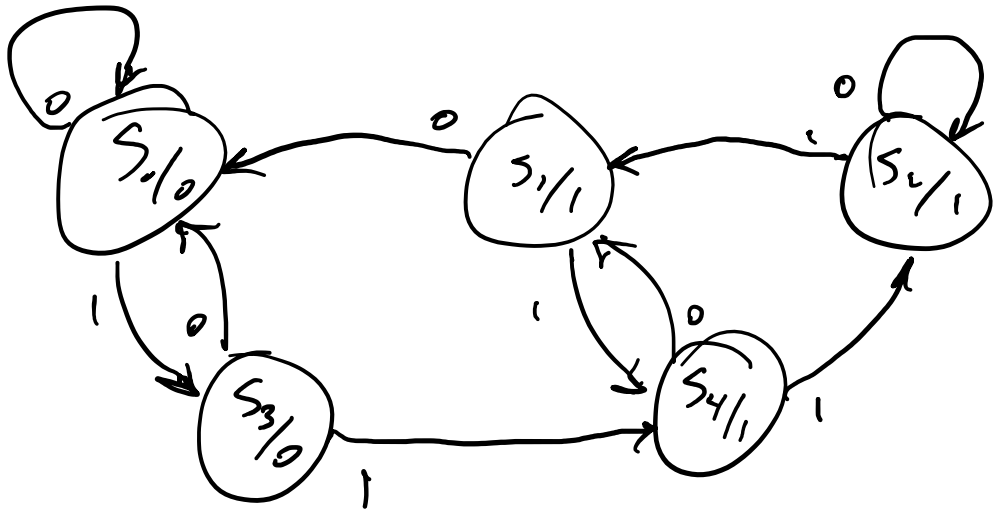
a) #2 a) Find all input sequences yielding an output sequence of 011110.
(This is called an "inverse problem".)

0 is the output of S_0, assuming that we're in S_0 so we've got the 1st 0 no matter what. So we need to tell what six digit string will generate 011110 starting from S_0.

We have to get to a state that produces a 1 on a path of length 2 from S_0, & there's only one way to do it: 11. Now we're sitting on S_4 & need to produce 110 - so we have to land on a state writing 0 in a path of length 3, writing 1s as we go. There's only one way to do it: $S_4 \xrightarrow{1} S_2 \xrightarrow{1} S_1 \xrightarrow{0} S_0$

So there's only one way to create that output: 11110 (if we assume we get the 0 for free).

input	1	1	1	1	0
state	S_0	S_3	S_4	S_2	S_1	S_0
output	0	1	1	1	1	0



If we don't count the 1st output of S

then: ① We need to get to 01 in a path of length 2, so 11. Where S_4

01111 0

② Now we need to get to 0, via 1110, in a path of length 6.

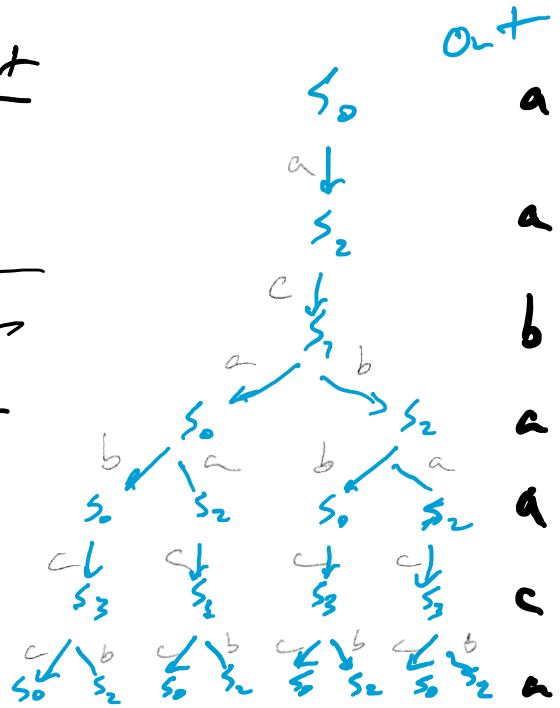
Two options: $S_1 S_4 S_1 S_0$: 0 1 0

$S_2 S_2 S_1 S_0$: 1 0 1

∴ **1 1 0 1 0 0 or 1 1 1 0 1 0**

26.

	a	b	c	Out
S_0	S_2	S_0	S_3	a
S_1	S_0	S_2	S_3	b
S_2	S_2	S_0	S_1	a
S_3	S_1	S_2	S_0	c



Produce abaaaca

So a tree provides a good format for solving this problem.

$$a\,c\,(a \vee b)(a \vee b)\,c\,(c \vee b)$$

2c. The machine's states produce 0s for all states but S_3; so it's just a question of how we can end up in S_3.

$S_3\,S_3^+$ 11111 11110 10111

$S_1\,S_3\,S_3^+$ 01221 11100 10110

$S_1\,S_2\,S_3\,S_3^+$ 00111 11000 10100

$S_2\,S_1\,S_2^+\,S_3^+$ 00011 10000 01011

00001 01010

00000

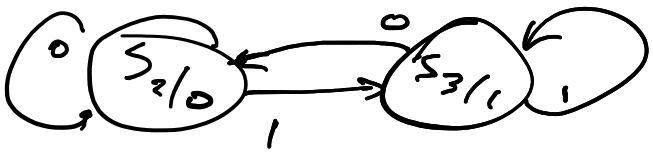
Looks like "what can't you get?" Would be a better question... :)

You can get them all, $2^5 = 32$ outputs. But more specifically,

input	a_1 a_2 a_3 a_4 a_5
state	S_0
output	0 a_1 a_2 a_3 a_4 a_5

Each state writes its input!

R is a slow delay FSM! And it could be minimized:



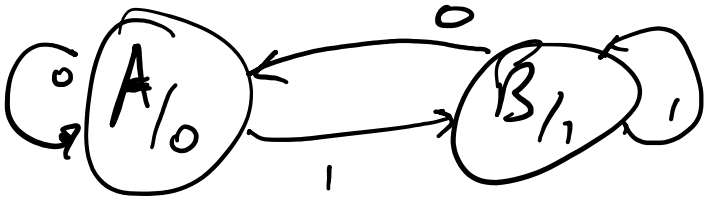
Let's see:

0-equivalent: $\{0,1,2\}, \{3\}$

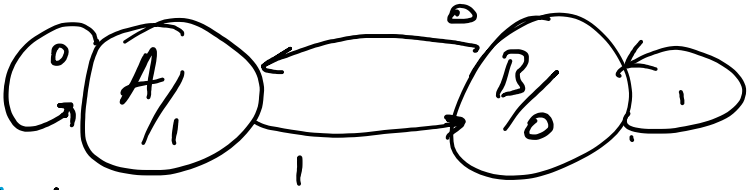
1-equivalent: $\{0,1,2\}, \{3\}$ Done!

It's not easy....

A B

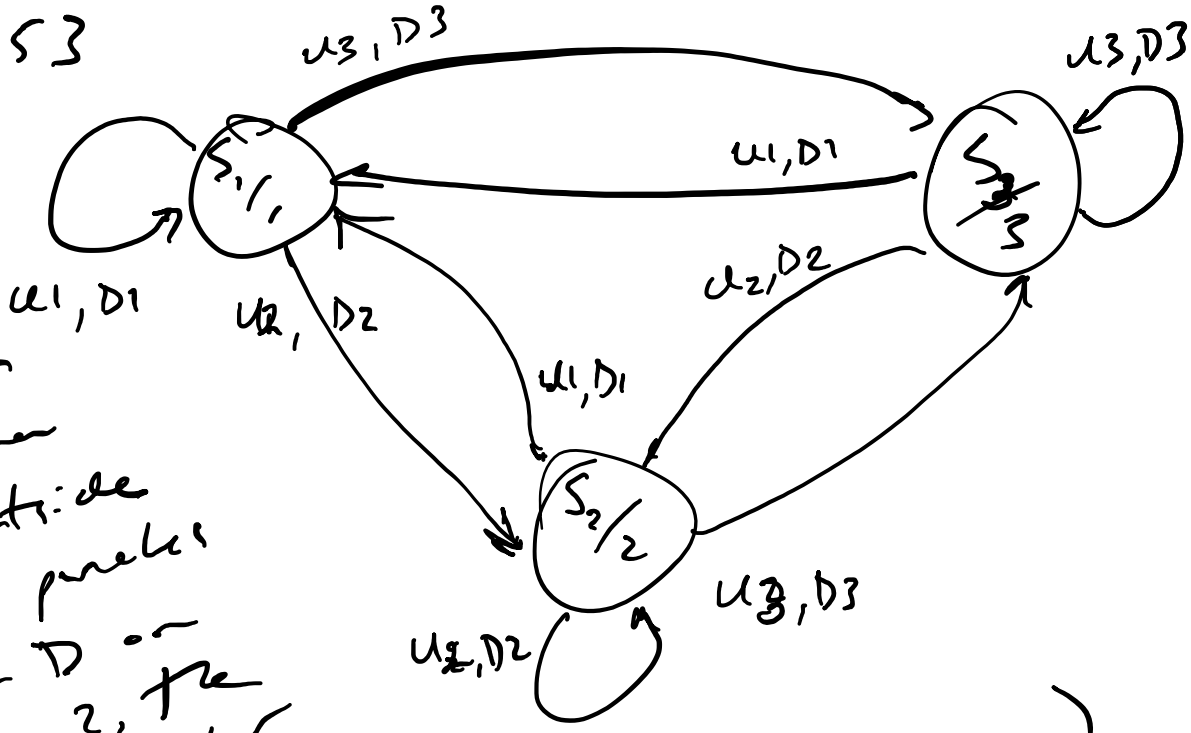


#11a. Let's repurpose the machine above — which writes the input string, to write the opposite:



In	0	1	0	1	1
	A	A	B	A	B	B
	1	1	0	1	0	0

#20 / 753

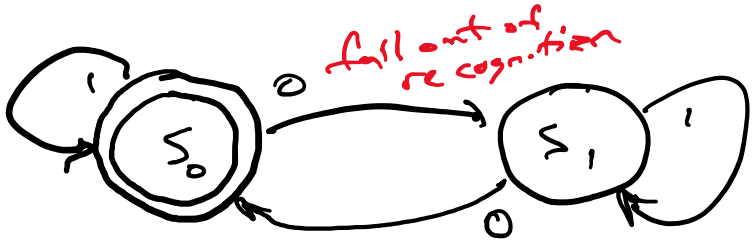


My interpretation was that when someone outside the elevator presses either U or D on floor 2, the elevator responds by going to that floor - or staying, if already there

---

Part b) #25a, 32, 68

#25a : Set of all strings containing an even # of 0s: (that includes 0 0s, so)



#34 There's a wastebasket state... if we fall into it, we're done.

00 is recognized, the 1st string.

Then, so long as we keep getting 0s, the state is one-of recognition.

1* 00 0*

We can pad w/ 1s at the outset, too.

#68 0-equivalent states:

{0, 1, 3, 6}, {2, 4, 5, 7}

1-equivalent:

{0, 3}, {1, 6}, {2, 4, 5, 7}

2-equivalent:

{0, 3}, {1, 6}, {2, 4}, {5, 7}

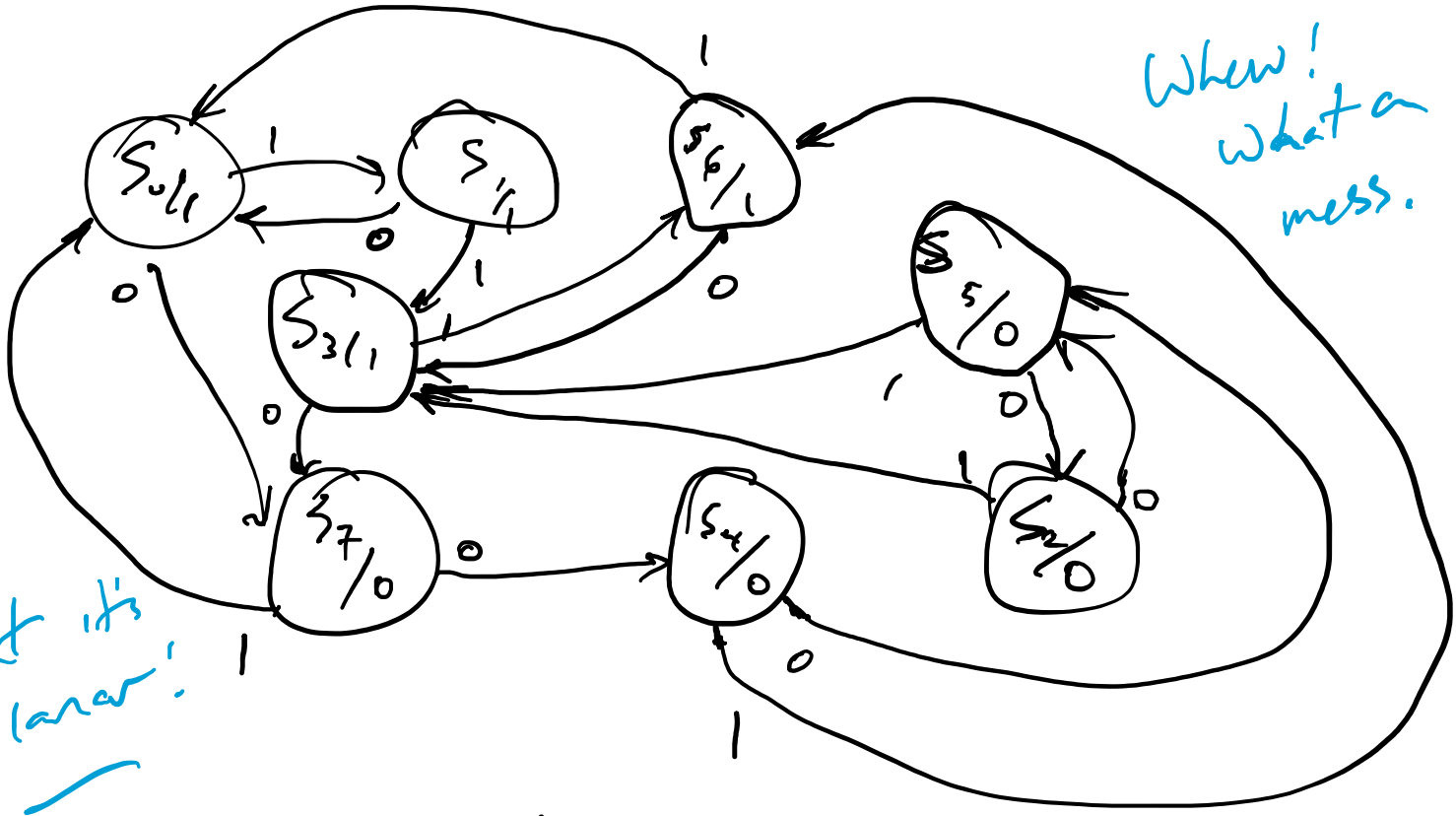
3-equivalent:

{0, 3}, {1, 6}, {2, 4}, {5, 7}

Done!

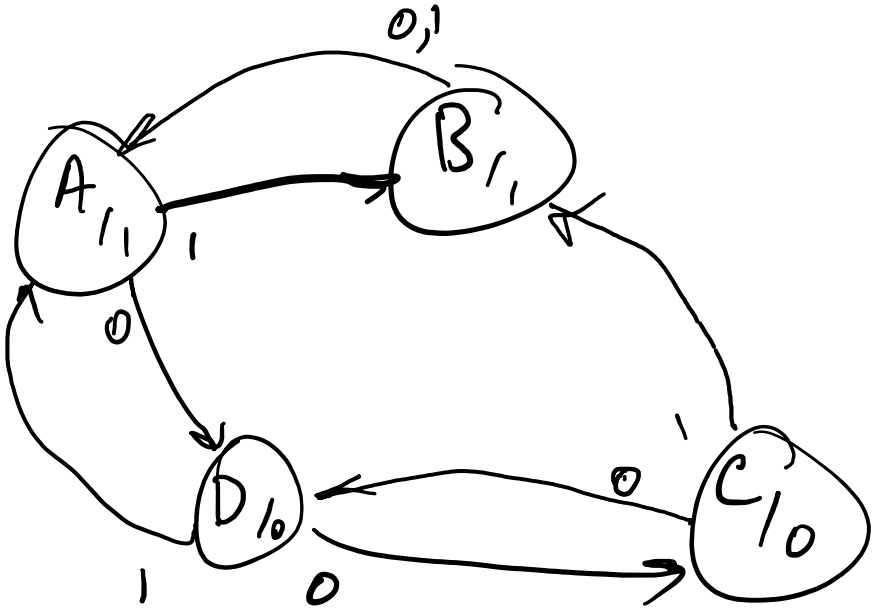
A B C D

No change.



Whew! What a mess.

But it's planar!



Simplified FSM, + we can't simplify further:

A + B can't be equivalent.

$A,0 \to D/0$ + $B,0 \to A/1$ + $D,1 \to A$ + $C,1 \to B$ } + these aren't equivalent.