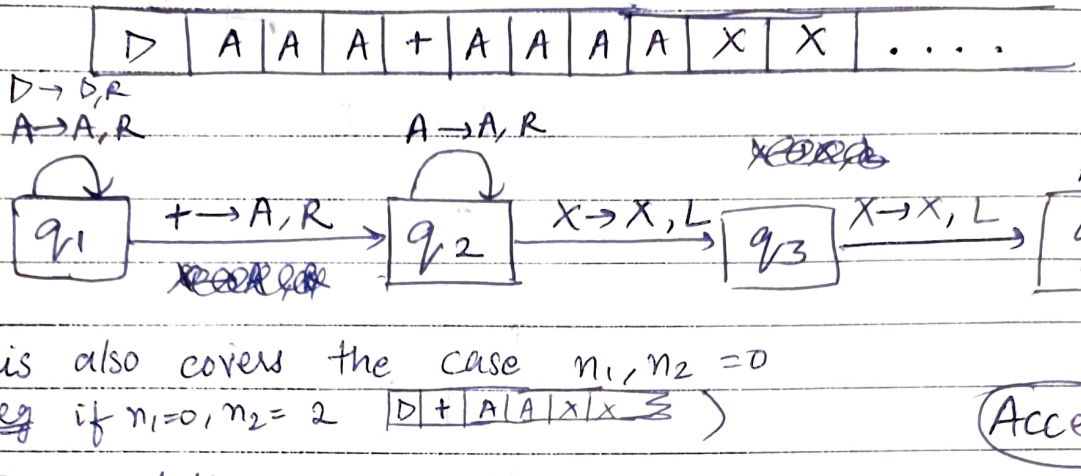
Prob 1.1 AAA + AAAAXXX

(1)



* This also covers the case $n_1, n_2 = 0$
(eg if $n_1 = 0, n_2 = 2$ ▷ + A A X X)

Description of the Turing Machine :

i	i_el	f	r_el	dir^n
q_1	▷	q_1	▷	R
q_1	A	q_1	A	R
q_1	+	q_2	A	R
q_2	A	q_2	A	R
q_2	X	q_3	X	L
q_3	A	q_4	X	S

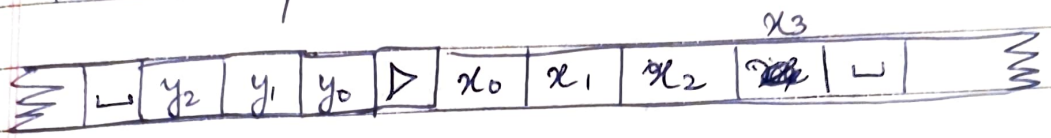
i ⇒ initial state
f ⇒ final state
i_el ⇒ element in the box being read.
r_el ⇒ replaced element
dir ⇒ direction

(2) As the execution steps include reading, writing and scanning through the n elements of list once or twice, the $T(n)$ will be $O(n)$

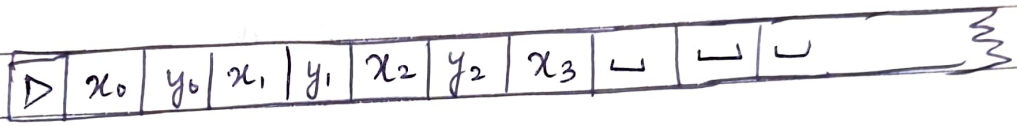
Prob 1.2

(1) A bi-infinite TM can be simulated on a standard TM by shifting the left part to the right (or folding) such that the elements of the left and right side appear alternatively.

For example, this :



can be written as :



- The internal states can be q_{x_0}, q_x, and q_{y_0}, q_{y_1}, etc depending on the element which is being scanned.

- To access the elements of left side of the strip (or right) only, one can shift the counter by 2 rights (RR) instead of the normal 1R

- The Turing Machine description would then include :

$q_0, \triangleright$	$q_{x_0}, \triangleright, R$
q_{x_0}, x_0 or q_{x_0}, x_0	q_{x_0}, x_0', RR q_{y_0}, x_0, R

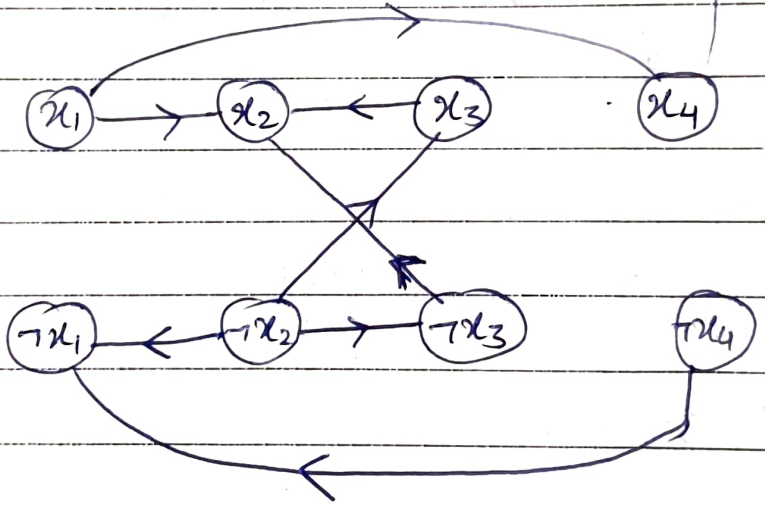
(2) – From the tape structure and transition function, we can see that the execution steps would not increase by an appreciable amount.

– There might ~~~~ be additional steps ~~depending on~~ because of transitioning from left-type of state to right-type of state but that would only scale the correspondance by a scalar and would not affect $O(p(n))$.

Prob 1.3

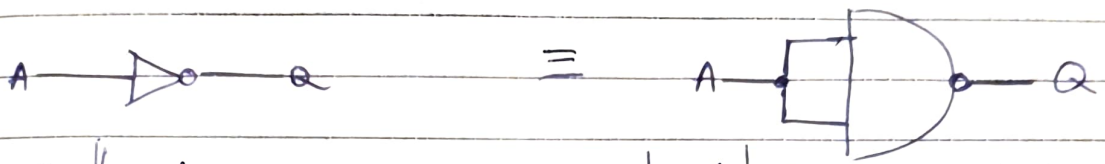
Clause	$\neg\alpha \vee \beta$	$\beta \vee \neg\alpha$
$\neg x_1 \vee x_2$	$x_1 \rightarrow x_2$	$\neg x_2 \rightarrow \neg x_1$
$x_2 \vee \neg x_3$	$\neg x_2 \rightarrow \neg x_3$	$x_3 \rightarrow x_2$
$x_4 \vee \neg x_1$	$\neg x_4 \rightarrow \neg x_1$	$x_1 \rightarrow x_4$
$x_2 \vee x_3$	$\neg x_2 \rightarrow x_3$	$\neg x_3 \rightarrow x_2$

$G(\phi)$ graph :



1.5 We will prove that NAND Gate is a universal gate by showing that all the basic classical logic gates can be designed by just using NAND gates. (and FANOUT and CROSSOVER)

1) NOT Gate



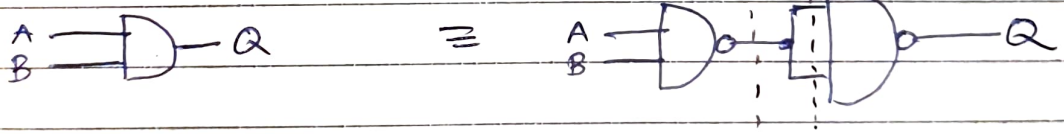
in	out
0	1
1	0

inp	~~in~~ 1	2	output
0	0	0	→ 1
	0	0	
1	1	1	→ 0
	1	1	

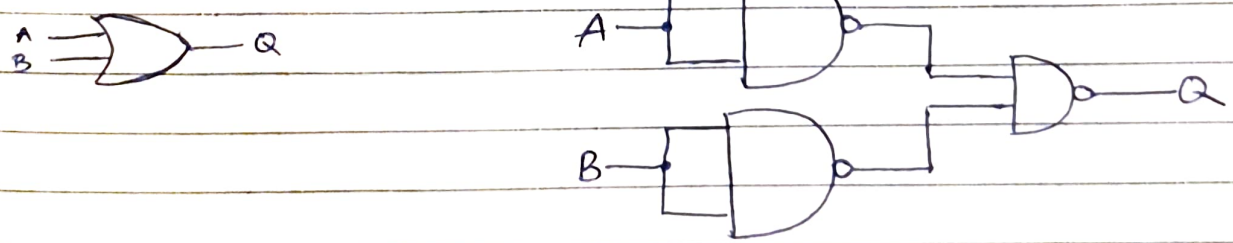
 ⇒ fanout (copies state A)

Similarly,

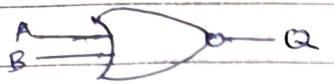
2) AND Gate

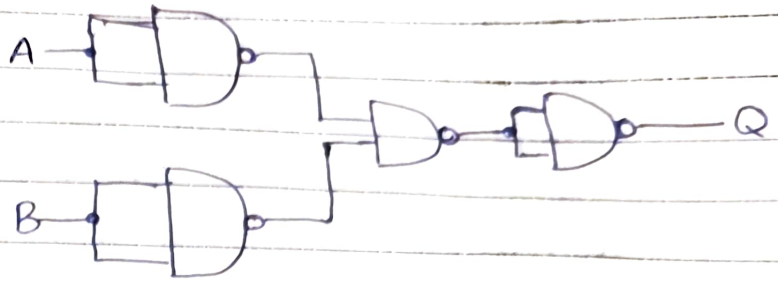


3) OR Gate

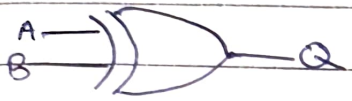


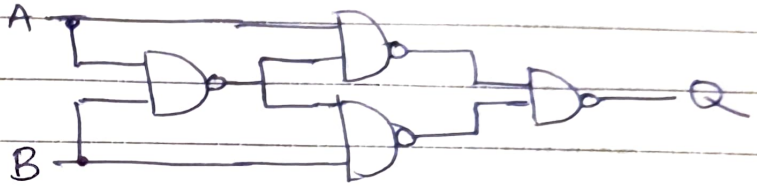
4) NOR





5) XOR

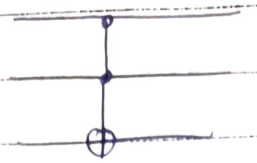




We can see how all the basic logic gates can be reconstructed using NAND gate and fanout. Hence, we can say that NAND gate is a universal gate.

B

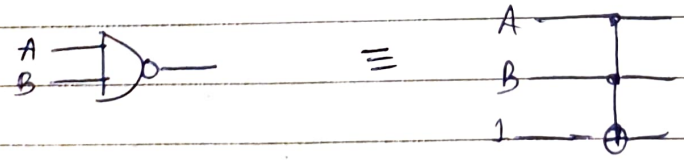
1.5.2 Toffoli



- A gate is reversible if it has same number of inputs and outputs and if the inputs can be retrieved even after the application of the gate.
- Toffoli gate has same number of inputs and outputs.
- On applying Toffoli gate to the circuit again, we can retrieve the inputs back.
- We can also see how all the outputs are unique and correspond to only one set of input. Hence, we can say that Toffoli gate is reversible.

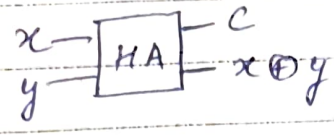
Proving Universality

NAND(a,b) ≡ Toffoli(a,b,1)

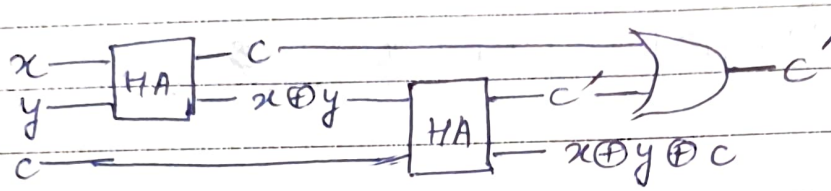


Toffoli gate can be written as a NAND gate. As NAND gate is universal, we can say that Toffoli gate is universal as well.

Prob 1.6 Half adder :



Full adder :



(1) ∴ $r \equiv x \oplus y \oplus c$ in the given Figure.

(2) $x \equiv x_2 x_1 x_0$

$y \equiv y_1 y_0$

s.t. $z = \sum_i x_i 2^i$

$\Rightarrow z = x_0 \cdot 1 + x_1 \cdot 2 + x_3 \cdot 4$

- x can be represented using 3 bits and y using 2 bits but their sum will require 4 bits.

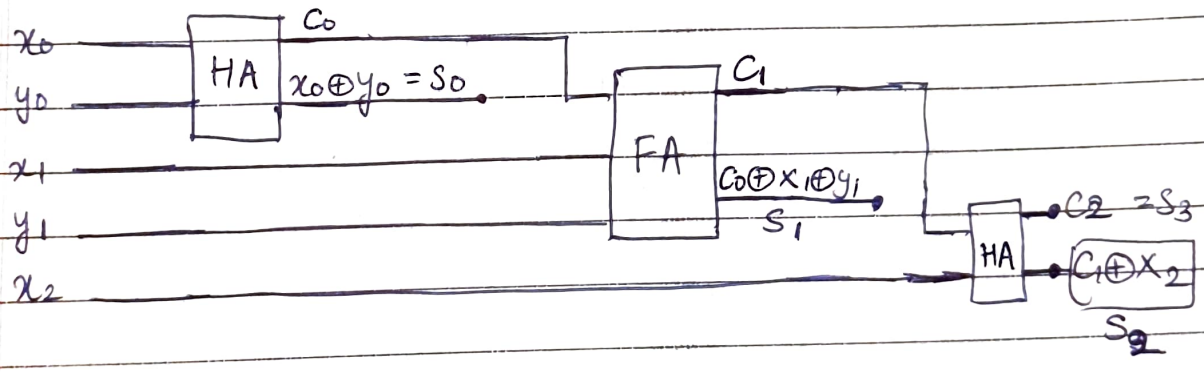
- Maximum 3 bit number is 111 and maximum 2 bit number is 11. Their sum is 1000 which needs 4 bits. Hence the sum of x and y will require atleast 4 bits.

(3) Using the property of ~~the~~ adders and the ideology behind summing numbers we can find the bits of sums.

$$S = S_3 S_2 S_1 S_0$$

$$\begin{array}{r} \quad\quad\quad\quad\quad\quad C \\ \quad\quad\quad\quad X\ y\ Z \\ + \quad\quad\quad\quad V\ W \\ \hline \quad\quad\quad\quad\quad Z \oplus W \end{array}$$

For summing two numbers, we do their XOR (or sum) bit-by-bit and save the carry-over for XOR with other bits



$\therefore \quad S = S_3 S_2 S_1 S_0$

where — $S_0 = x_0 \oplus y_0$

— $S_1 = C_0 \oplus x_1 \oplus y_1$

— $S_2 = C_1 \oplus x_2$

— $S_3 = C_2$

The output of the above circuit gives the bits of the required sum.

Prob 1.7

(1) OR gate

(in-1 V in-2) gives the output as shown in the table.

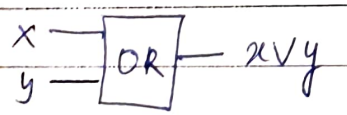
(2) Here, the same output '1' is received for multiple input sets ~~and hence~~. We cannot identify which output results from which input set, hence this function is not reversible.

(3) To build ~~the~~ a reversible version of any function $f : \{0,1\}^m \rightarrow \{0,1\}^n$ we can find an alternate function $\tilde{f}$: ~~[illegible]~~

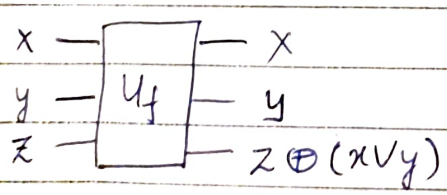
$$\tilde{f} : \{0,1\}^{m+n} \rightarrow \{0,1\}^{m+n}$$

such that $\tilde{f}(x,y) = (x, y \oplus f(x))$

for OR gate



we can construct U_f s.t :



This will result in the following truth table :

input			output		
x	y	z	x	y	$z \oplus (x \vee y)$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	0	1	1	1
0	0	1	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	1	1	1	0

We can see that every output is unique and can be traced back to its input. By looking at the output of third register we can also get the value of "OR" on inputs.

Now, looking at the input and output state, we will construct an 8 x 8 matrix that satisfies the transition of all these states correctly.

$$\begin{pmatrix} 1000 & & \\ 0100 & & O_4 \\ O_2 \; 0\,1 & 1\,0 & \\ & & 0\,1 \\ O_4 & 1\,0 & O_2 \\ & & 0\,1 \\ & O_2 & 1\,0 \end{pmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

For this matrix, the inverse turns out to be equal to its conjugate transpose

$$U_f = U_f^{-1} = U_f^{\dagger}$$

$\therefore U_f$ is unitary.