2. Summary of theory of classical computation

(a) Bits, gates, circuits, universality, complexity
(In class, NC 3.1.2, KLM 1.3)

(b) The linear algebra formalism (classical)
(In class, KLM 1.4)

(c) Models of computation
(Turing Machines, randomized computation)
(Optional reading, NC 3.1, KLM 1.2)

(d) Reversible computation
(KLM 1.5, NC 3.2.5)

Unit of classical information:

1 bit: use 0 or 1 to label one of two possible configurations.

Physically, the system has 2 distinguishable states that can be prepared and manipulated.

Unit of classical information:

1 bit: use 0 or 1 to label one of two possible configurations.

Physically, the system has 2 distinguishable states that can be prepared and manipulated.

With n such systems, there are 2 n possible configurations:

e.g. n=3

000, 001, 010, 011, 100, 101, 110, 111

Computation:

the task to evaluate a function on any given input

e.g., we can compute the parity of a 3-bit string
e.g., we can compute the sum of a pair of inputs

Computation:

the task to evaluate a function on any given input

e.g., we can compute the parity of a 3-bit string
e.g., we can compute the sum of a pair of inputs

Questions:

Do we need to rebuild our computing machine every time we change our computation?

Is there a collection of simple steps versatile enough so that they can be composed to compute anything?

Gates

Definition: A gate with r input bits and s output bits is a function from $\{0,1\}^r$ to $\{0,1\}^s$.

Gates

Definition: A gate with r input bits and s output bits is a function from $\{0,1\}^r$ to $\{0,1\}^s$.

e.g.: NOT gate, with r=s=1, NOT(0)=1
NOT(1)=0

Gates

Definition: A gate with r input bits and s output bits is a function from $\{0,1\}^r$ to $\{0,1\}^s$.

e.g.: NOT gate, with r=s=1,

NOT(0)=1
NOT(1)=0

e.g.: AND gate, with r=2, s=1,

AND(00)=0
AND(01)=0
AND(10)=0
AND(11)=1

Gates

Definition: A gate with r input bits and s output bits is a function from $\{0,1\}^r$ to $\{0,1\}^s$.

e.g.: NOT gate, with r=s=1,
NOT(0)=1
NOT(1)=0

e.g.: AND gate, with r=2, s=1,
AND(00)=0
AND(01)=0
AND(10)=0
AND(11)=1

e.g.: FANOUT, with r=1, s=2,
FANOUT(0)=00
FANOUT(1)=11

Computation by composing gates

e.g., can we compute the OR gate
(r=2, s=1)
if you are allowed to use the
AND gate and the NOT gate?

OR(00)=0
OR(01)=1
OR(10)=1
OR(11)=1

Computation by composing gates

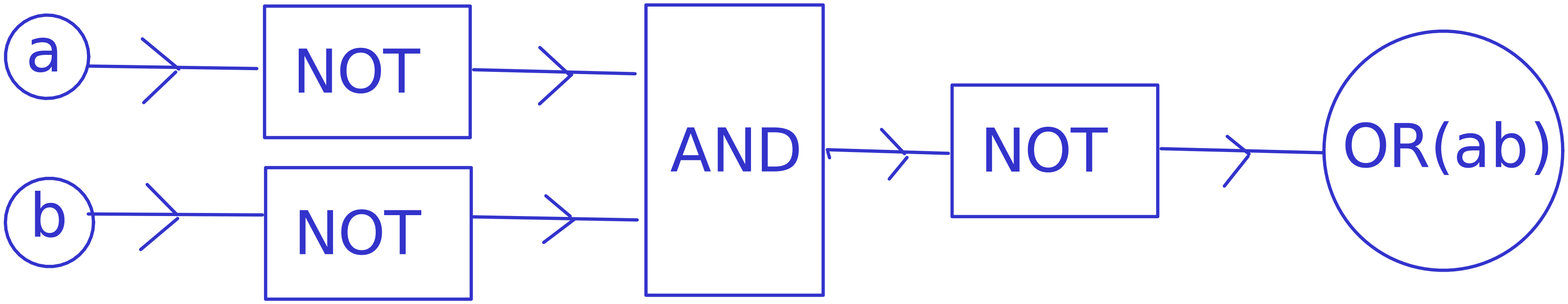
e.g., can we compute the OR gate
(r=2, s=1)
if you are allowed to use the
AND gate and the NOT gate?

OR(00)=0
OR(01)=1
OR(10)=1
OR(11)=1

Answer: for any two bits a and b, the following holds

NOT(AND(NOT(a), NOT(b))) = OR(ab)

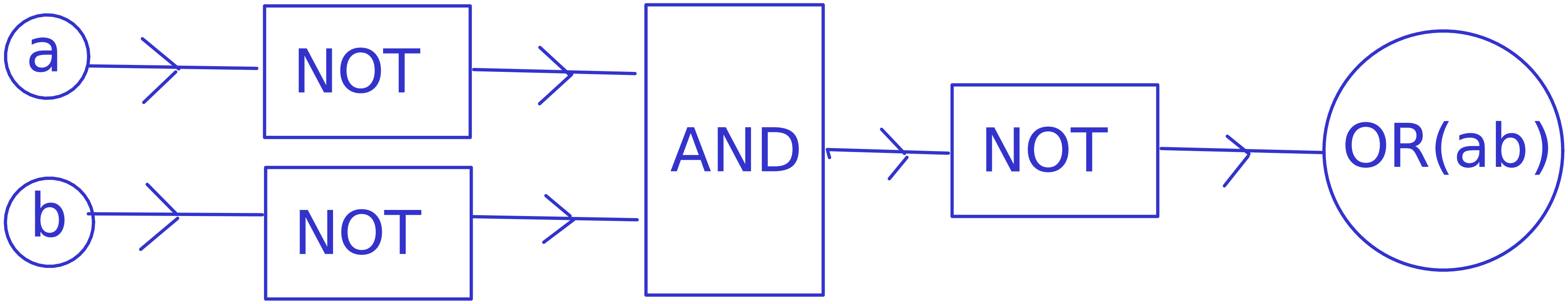
Circuit representation:



General properties of a circuit:

1. A circuit is an acyclic directed graph
2. Gates are vertices.
3. The direction of an edge gives the direction of time.
4. Gates are partially time-ordered.

e.g.,

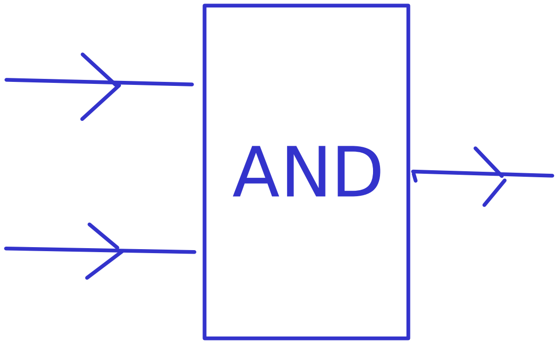


General properties of a circuit:

1. A circuit is an acyclic directed graph
2. Gates are vertices.
3. The direction of an edge gives the direction of time.
4. Gates are partially time-ordered.

5. Input/output bits of a gate are given by incoming/ outgoing edges.

e.g.

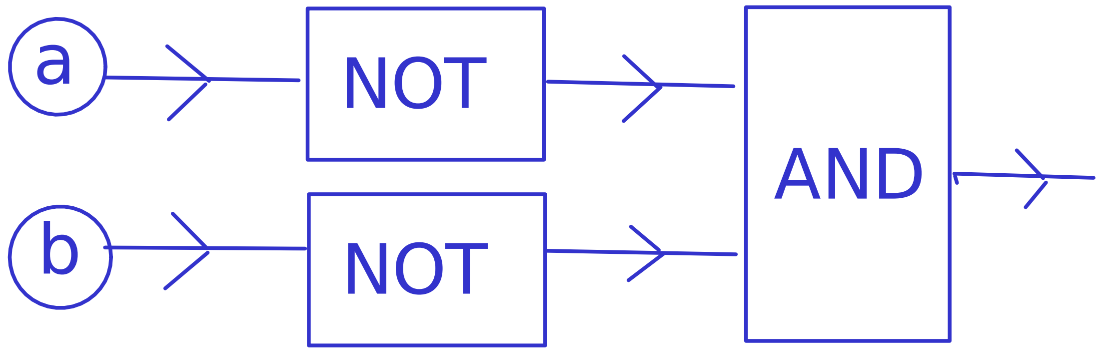


General properties of a circuit:

1. A circuit is an acyclic directed graph
2. Gates are vertices.
3. The direction of an edge gives the direction of time.
4. Gates are partially time-ordered.

5. Input/output bits of a gate are given by incoming/ outgoing edges.
6. An edge takes the output bit of a gate to the input bit of a subsequent gate.

e.g.,



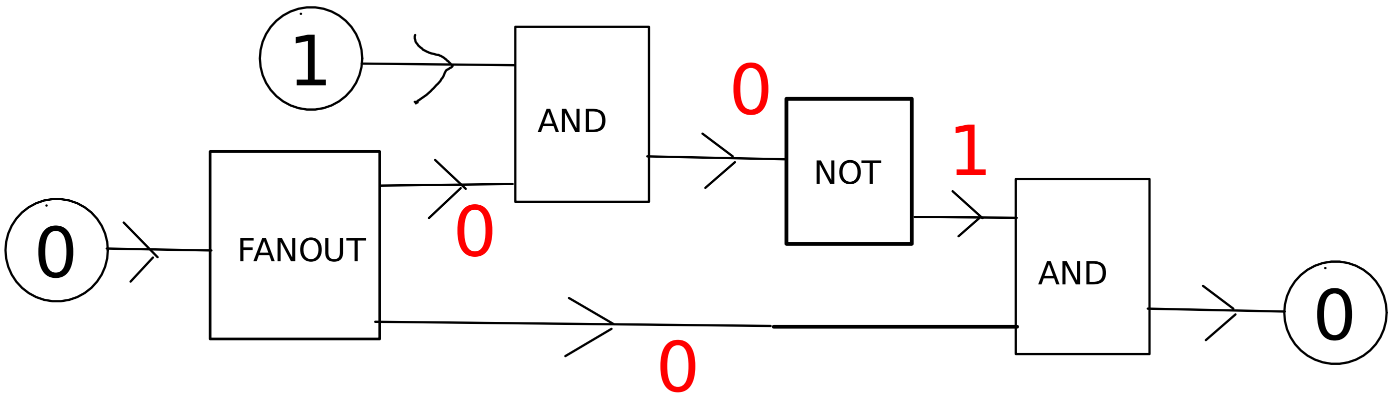
General properties of a circuit:

1. A circuit is an acyclic directed graph
2. Gates are vertices.
3. The direction of an edge gives the direction of time.
4. Gates are partially time-ordered.

5. Input/output bits of a gate are given by incoming/outgoing edges.
6. An edge takes the output bit of a gate to the input bit of a subsequent gate.
7. Input/output bits of the ciruit are source/sink vertices.

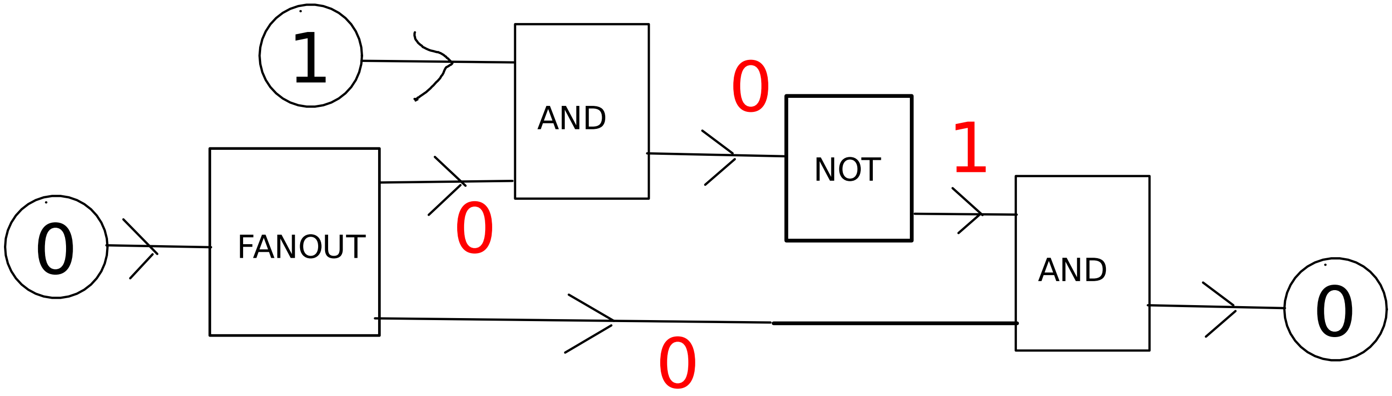
8. A circuit shows which bits are transformed by gates, and how simple functions given by the gates are composed to obtain any computation.

Circuit example:



time runs from left to right, arrows often omitted

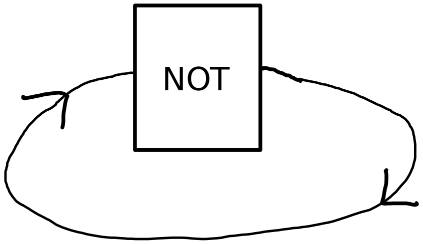
Circuit example:



time runs from left to right, arrows often omitted

Not a circuit example:

Not acyclic:



Questions:

Do we need to rebuild our computing machine every time we change our computation?

Is there a collection of gates versatile enough so that they can be composed to compute anything?

Definition: universal set of gates

A set of gates G is universal if :

for any positive integers n,m
and
for any function f : $\{0,1\}^n \to \{0,1\}^m$
there is a circuit to compute f using the gates in G.

Example:

1. If the number of output bits m=1,
   then f is given by a truth table,
   and can be computed by a circuit with
   AND and NOT gates.

   (e.g., OR gates)

Example:

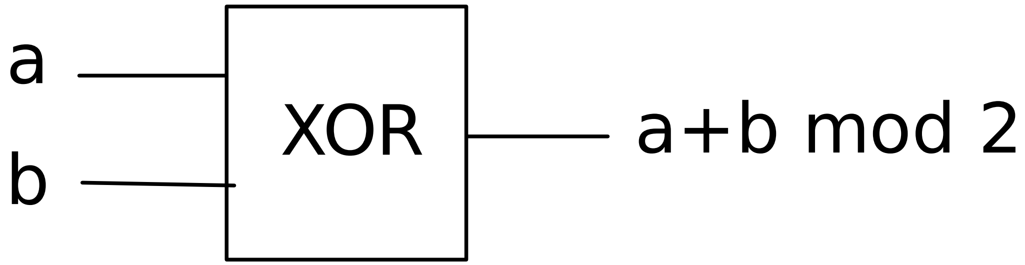
1. If the number of output bits m=1, then f is given by a truth table, and can be computed by a circuit with AND and NOT gates.

2. For general f (larger m), given FANOUT, we can computer each output bit using {AND,NOT}.

Theorem: {AND,NOT,FANOUT} is universal.

Exercise:

Define XOR(a,b) = a+b mod 2. Is {XOR} universal?



a

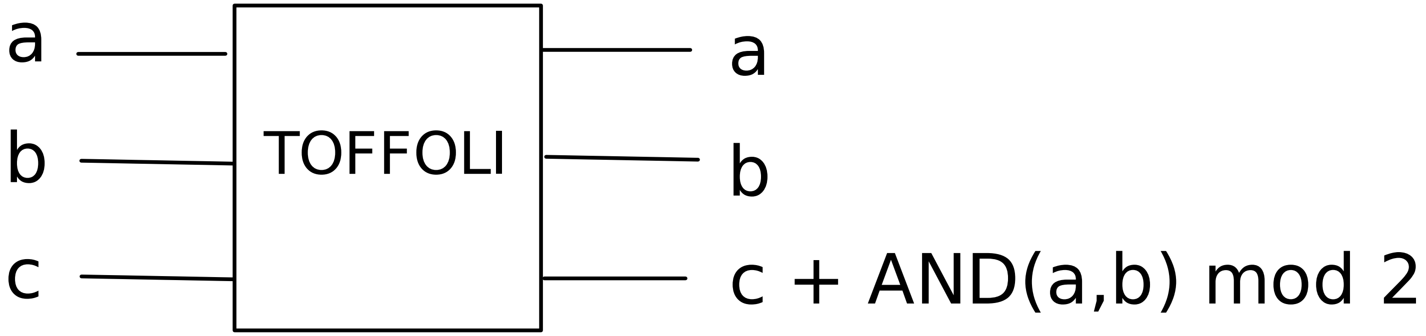
b

XOR

a+b mod 2

Exercise:

Define TOFFOLI(a,b,c) = (a, b, c + AND(a,b) mod 2).



a

b

c

TOFFOLI

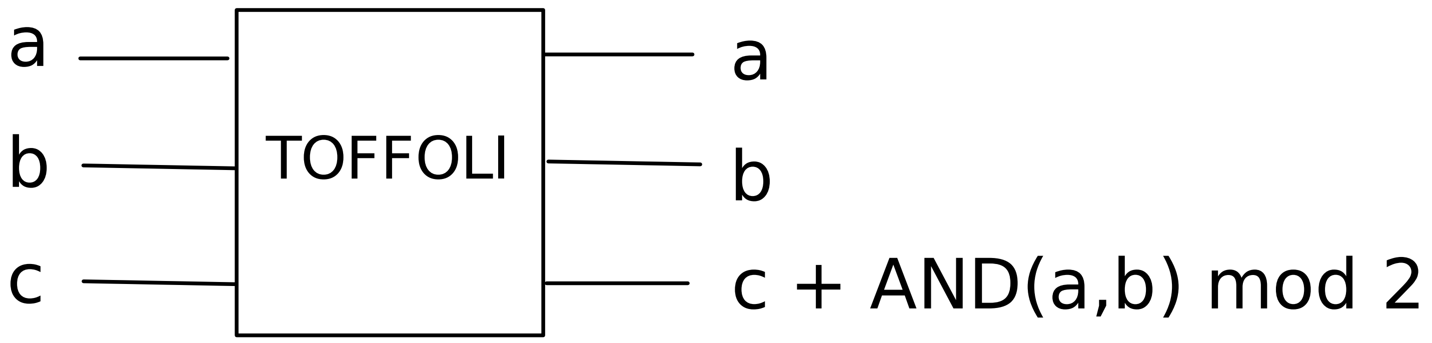
a

b

c + AND(a,b) mod 2

Exercise:

Define TOFFOLI(a,b,c) = (a, b, c + AND(a,b) mod 2).

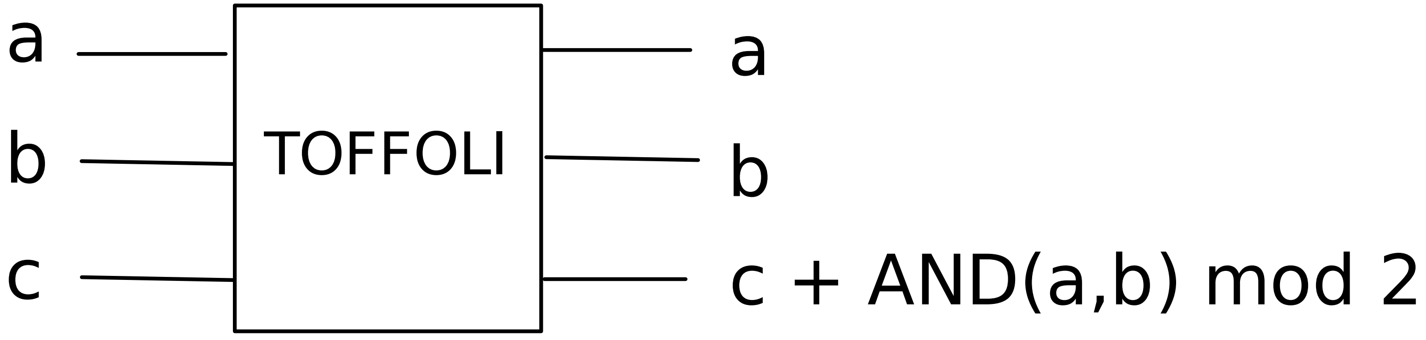


Theorem: assuming the ability to prepare 0 and 1 as inputs, {TOFFOLI} is universal.

Proof: exercise. Hint: how to compute each of AND, NOT, FANOUT using TOFFOLI ? Qn: do we need 0/1?

Exercise:

Define TOFFOLI(a,b,c) = (a, b, c + AND(a,b) mod 2)



Theorem: assuming the ability to prepare 0 and 1 as inputs, {TOFFOLI} is universal.

Proof: exercise. Hint: how to compute each of AND, NOT, FANOUT using TOFFOLI ? Qn: do we need 0/1?

Corollary: TOFFOLI self-inverse, gives reversible comp & quantum computation of classical circuits !

Why the circuit model?

1. Facilitates analysis

2. Pathway for implementation

Examples: complexity, reversible computation.

Circuit complexity:

Fix a universal set of gates G, and a computation.

(e.g., factoring positive integers n)

Generate a family of circuits for each input size.

Hardness (complexity) measures:

Width w : number of wires (space) in the circuits
Depth d : number of (non-parallelizable) time steps in the circuits
Size : wd

(We care about how w,d grow with the input size.)

Exercise:

How does the specific choice of a finite universal set of gates affect the depth and width?

2. Summary of theory of classical computation

✓ (a) Bits, gates, circuits, universality, complexity
(In class, NC 3.1.2, KLM 1.3)

→ (b) The linear algebra formalism (classical)
(In class, KLM 1.4)

(c) Models of computation
(Turing Machines, randomized computation)
(Optional reading, NC 3.1, KLM 1.2)

(d) Reversible computation
(KLM 1.5, NC 3.2.5)

(b) The linear algebra formalism (classical)

Goal: represent bit configurations as vectors, and
represent the action of the gates as matrices.

Why?
- analysis via powerful tools from linear/Lie algebra
- simple composition rules
- warm up for quantum!

(b) The linear algebra formalism (classical)

Goal: represent bit configurations as vectors, and
represent the action of the gates as matrices.

We identify:

0 as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 1 as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

e.g.1 What matrix corresponds to the NOT gate?

Linear algebra:

$$M = \left[\begin{array}{c|c|c} c_1 & c_2 & \cdots \end{array}\right] = \begin{bmatrix} r_1 \\ \hline r_2 \\ \hline \vdots \\ \vdots \end{bmatrix}, \quad e_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \leftarrow \text{i-th position}$$

What is Me_i?

Linear algebra:

$$M = \left[\begin{array}{c|c|c} c_1 & c_2 & \cdots \end{array}\right] = \left[\begin{array}{c} r_1 \\ \hline r_2 \\ \hline \vdots \\ \vdots \end{array}\right], \quad e_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \leftarrow$$

i-th position

What is Me_i? c_i

If $M e_i = v_i$ for all i, what is M ?

Linear algebra:

$$M = \left[\begin{array}{c|c|c} c_1 & c_2 & \cdots \end{array}\right] = \left[\begin{array}{c} r_1 \\ \hline r_2 \\ \hline \vdots \end{array}\right], \quad e_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

i-th position

What is Me_i? c_i

If $M e_i = v_i$ for all i, what is M ? $M = \left[\begin{array}{c|c|c} v_1 & v_2 & \cdots \end{array}\right]$

Recipe to reconstruct the matrix!

(b) The linear algebra formalism (classical)

Goal: represent bit configurations as vectors, and
represent the action of the gates as matrices.

We identify: 0 as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 1 as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

e.g.1 What matrix M corresponds to the NOT gate?

(b) The linear algebra formalism (classical)

Goal: represent bit configurations as vectors, and
represent the action of the gates as matrices.

We identify: 0 as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 1 as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

e.g.1 What matrix M corresponds to the NOT gate?

NOT(0) = 1, so, M takes $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ to $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ which is the 1st column of M

(b) The linear algebra formalism (classical)

Goal: represent bit configurations as vectors, and represent the action of the gates as matrices.

We identify: 0 as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 1 as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

e.g.1 What matrix M corresponds to the NOT gate?

NOT(0) = 1, so, M takes $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ to $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ which is the 1st column of M

NOT(1) = 0, so, M takes $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ to $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ which is the 2nd column of M

$\therefore$ $M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

We identify: 0 as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 1 as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

00 as $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ 01 as $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ 10 as $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ 11 as $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$.

e.g.2 FANOUT(0) = 00, FANOUT(1) = 11

If F is the matrix that corresponds to FANOUT,

(a) how many columns does F has? 2? 4?
(b) how many rows?

We identify: 0 as $\begin{pmatrix}1\\0\end{pmatrix}$ 1 as $\begin{pmatrix}0\\1\end{pmatrix}$

00 as $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ 01 as $\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$ 10 as $\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$ 11 as $\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$.

e.g.2 FANOUT(0) = 00, FANOUT(1) = 11

If F is the matrix that corresponds to FANOUT,
1st col of F is:

We identify: 0 as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 1 as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

00 as $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ 01 as $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ 10 as $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ 11 as $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$.

e.g.2 FANOUT(0) = 00, FANOUT(1) = 11

If F is the matrix that corresponds to FANOUT,
1st col of F is: 2nd col is:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

We identify: 0 as $\begin{pmatrix}1\\0\end{pmatrix}$ 1 as $\begin{pmatrix}0\\1\end{pmatrix}$

00 as $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ 01 as $\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$ 10 as $\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$ 11 as $\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$.

e.g.2 FANOUT(0) = 00, FANOUT(1) = 11

If F is the matrix that corresponds to FANOUT,
1st col of F is: 2nd col is:

$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ $\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$

We identify: 0 as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 1 as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

00 as $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ 01 as $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ 10 as $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ 11 as $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$.

e.g.2 FANOUT(0) = 00, FANOUT(1) = 11

If F is the matrix that corresponds to FANOUT,
1st col of F is: 2nd col is:

$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ $\therefore$ $F = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$.

Exercise: which of the following corresponds to AND

(a) $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$

(b) $\begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}$

(c) $\begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$

(d) $\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

(e) $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$

Exercise:

Let CNOT(a,b) = (a, a $\oplus$ b mod 2).

What matrix corresponds to CNOT?

Summary:

The matrix representation for a gate

$f: \{0,1\}^n \text{ -> } \{0,1\}^m$

is a matrix with 2^n columns,
the i-th column is the vector (of length 2^m)
representing f(b(i)) where b(i) is the i-th bitstring
in the ordered list :

00...00, 00...01, 00...10, 00...11, ... , 11...11.

We've learnt how to derive the matrix representation for a gate.

What about a circuit of gates?

Goal: for a circuit, we want to derive the transformation on the input string by composing the actions of individual gates.

Prescription:

1. Describe the input data as a tensor product of vectors.

2a. Evolve the register(s) acted on by a gate G, leaving other registers unchanged.

2b. Derive the matrix representation of the joint system due to the gate G.

3. Compose the evolutions by the gates in the circuit (multiply the matrix representations).

Definition: tensor product for vectors

Let $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$

The tensor product of a, b,
denoted as $a \otimes b$,
has nm entries given by:

$$a \otimes b = \begin{pmatrix} a_1 (b) \\ a_2 (b) \\ \vdots \\ a_n (b) \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_1 b_2 \\ \vdots \\ a_1 b_m \\ a_2 b_1 \\ a_2 b_2 \\ \vdots \\ a_n b_m \end{pmatrix}$$

Properties of tensor product:

1. In general, $a \otimes b \neq b \otimes a$

2. For all a, b, c, $(a \otimes b) \otimes c = a \otimes (b \otimes c)$

Proof left as exercise.

Goal: for a circuit, we want to derive the transformation on the input string by composing the actions of individual gates.

Prescription:

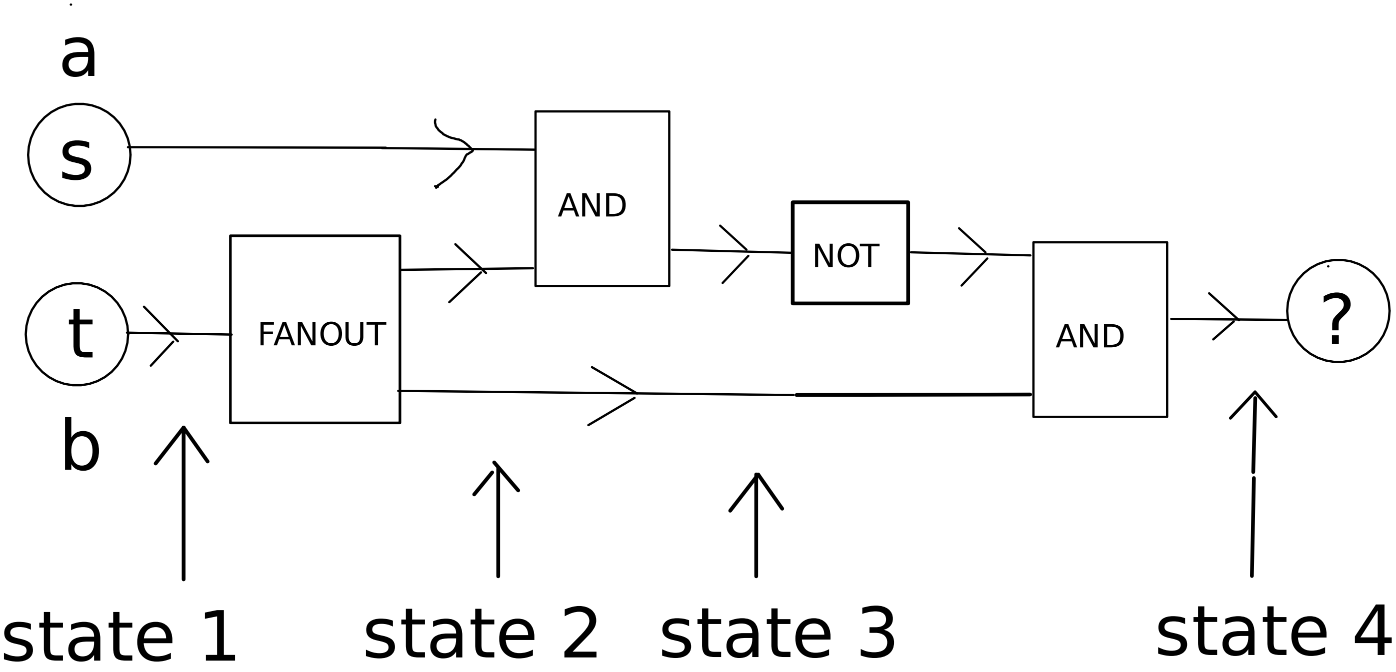
✓ 1. Describe the input data as a tensor product of vectors.

2a. Evolve the register(s) acted on by a gate G, leaving other registers unchanged.

2b. Derive the matrix representation of the joint system due to the gate G.

3. Compose the evolutions by the gates in the circuit (multiply the matrix representations).

Example:

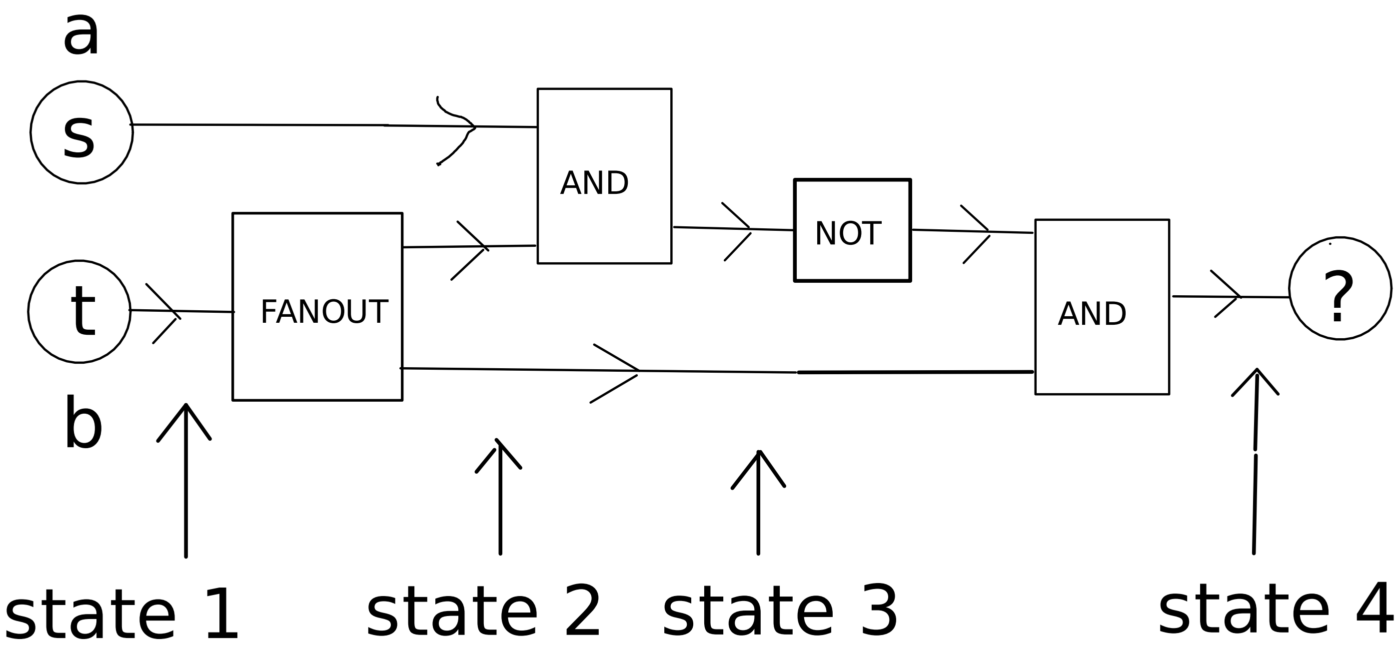


If s = 0, a = $\begin{pmatrix}1\\0\end{pmatrix}$, if s = 1, a = $\begin{pmatrix}0\\1\end{pmatrix}$.

Similarly for t and b.

State 1 = $a \otimes b$. What is state 2?
What matrix transforms
state 1 to state 2?

Example:



What is state 2?

In bit form: stt.

In vector form: $a \otimes (Fb)$ where $F = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$

What matrix takes

$a \otimes b$ to $a \otimes (Fb)$?

What matrix takes $a \otimes b$ to $a \otimes (Fb)$? $F = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$

1st column =

image of $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right]$$

What matrix takes $a \otimes b$ to $a \otimes (Fb)$?

$$F = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

1st col =

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes F \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right)$$

2nd col =

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes F \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$= \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right)$$

What matrix takes $a \otimes b$ to $a \otimes (Fb)$? $F = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$

1st col =

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes F \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

2nd col =

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes F \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3rd col =

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes F \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \hline 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

What matrix takes $a \otimes b$ to $a \otimes (Fb)$? $F = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$

1st col =

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes F \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right)$$

2nd col =

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes F \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$= \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right)$$

3rd col =

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes F \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ \hline 1 \\ 0 \\ 0 \\ 0 \end{array}\right)$$

4th col

$$= \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 1 \end{array}\right)$$

What matrix takes $a \otimes b$ to $a \otimes (Fb)$? $F = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$

Answer $= \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right)$

$$= \left(\begin{array}{c|c} 1\,F & 0\,F \\ \hline 0\,F & 1\,F \end{array}\right) := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes F = I_2 \otimes F$$

Definition: tensor product for matrices

Let $A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$ $B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix}$

Then, $A \otimes B =$

$$\begin{bmatrix} a_{11}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} & a_{12}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} & \cdots & a_{1n}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} \\ a_{21}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} & a_{22}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} & \cdots & a_{2n}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} \\ \vdots & & \ddots & \vdots \\ a_{m1}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} & a_{m2}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} & \cdots & a_{mn}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & & b_{2r} \\ \vdots & & & \\ b_{s1} & b_{s2} & \cdots & b_{sr} \end{pmatrix} \end{bmatrix}$$

Properties of tensor product of matrices:

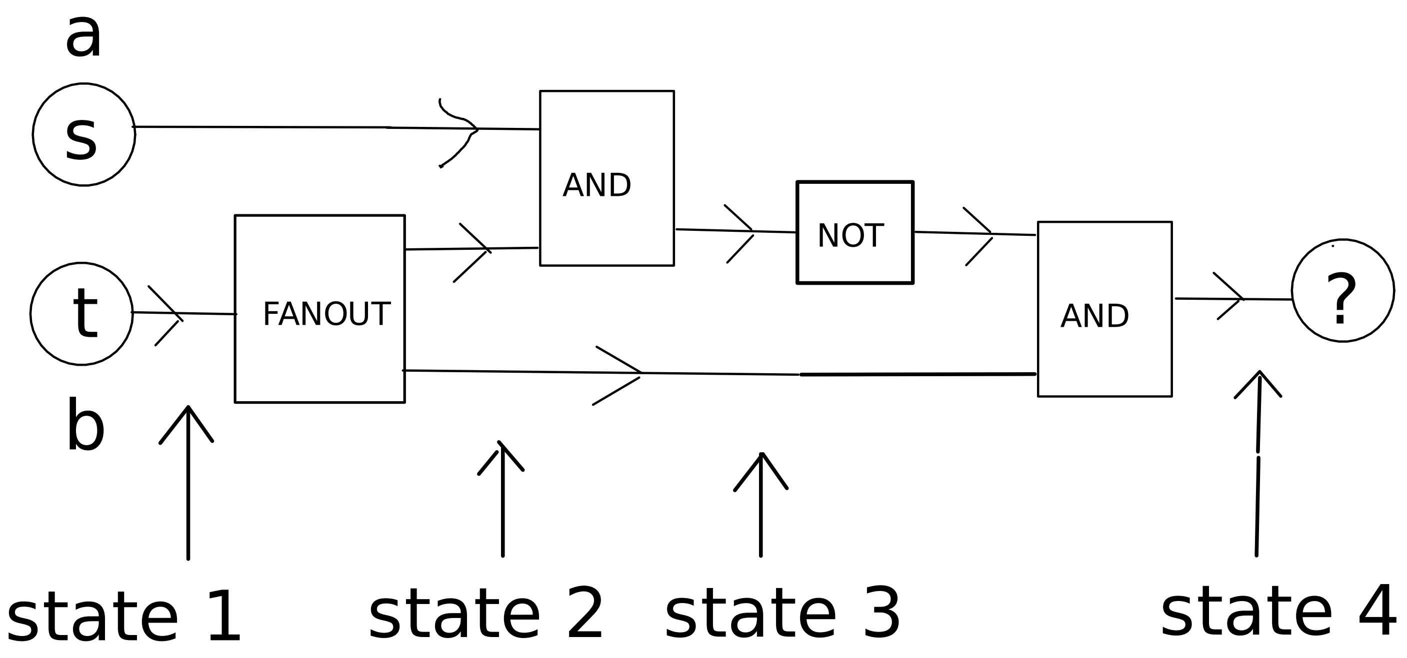
1. Consistent with tensor product of vectors as a special case.

2. $(A \otimes B) (a \otimes b) = (Aa) \otimes (Bb)$

3. $(A \otimes B) (C \otimes D) = (AC) \otimes (BD)$

Proof: exercise

Example:



What matrix corresponds to the transformation by the entire circuit?

state 1 = $a \otimes b$

state 2 = $I \otimes F \; (a \otimes b)$

Goal: for a circuit, we want to derive the transformation on the input string by composing the actions of individual gates.

Prescription:

✓ 1. Describe the input data as a tensor product of vectors.

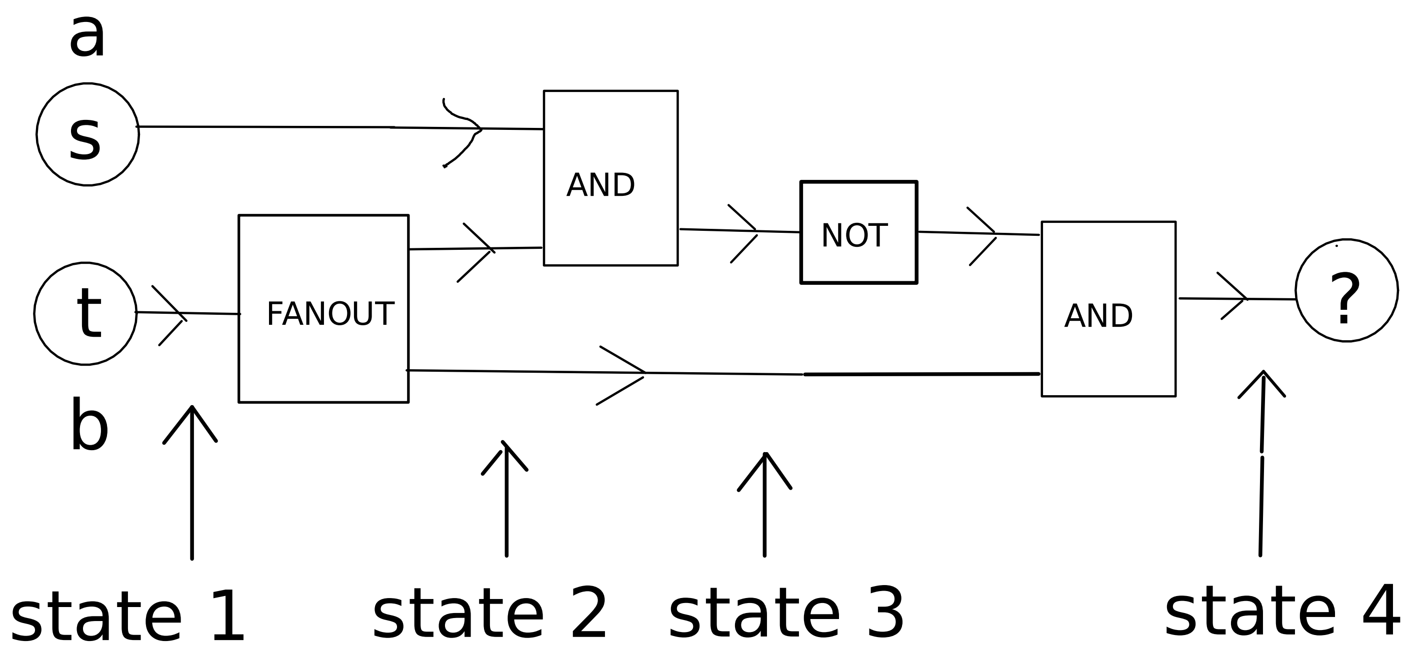
2a. Evolve the register(s) acted on by a gate G, leaving other registers unchanged.

2b. Derive the matrix representation of the joint system due to the gate G.

If M is the matrix rep of G, then $I \otimes M$ is the matrix rep on the joint system.

3. Compose the evolutions by the gates in the circuit (multiply the matrix representations).

Example:



What matrix corresponds to the transformation by the entire circuit?

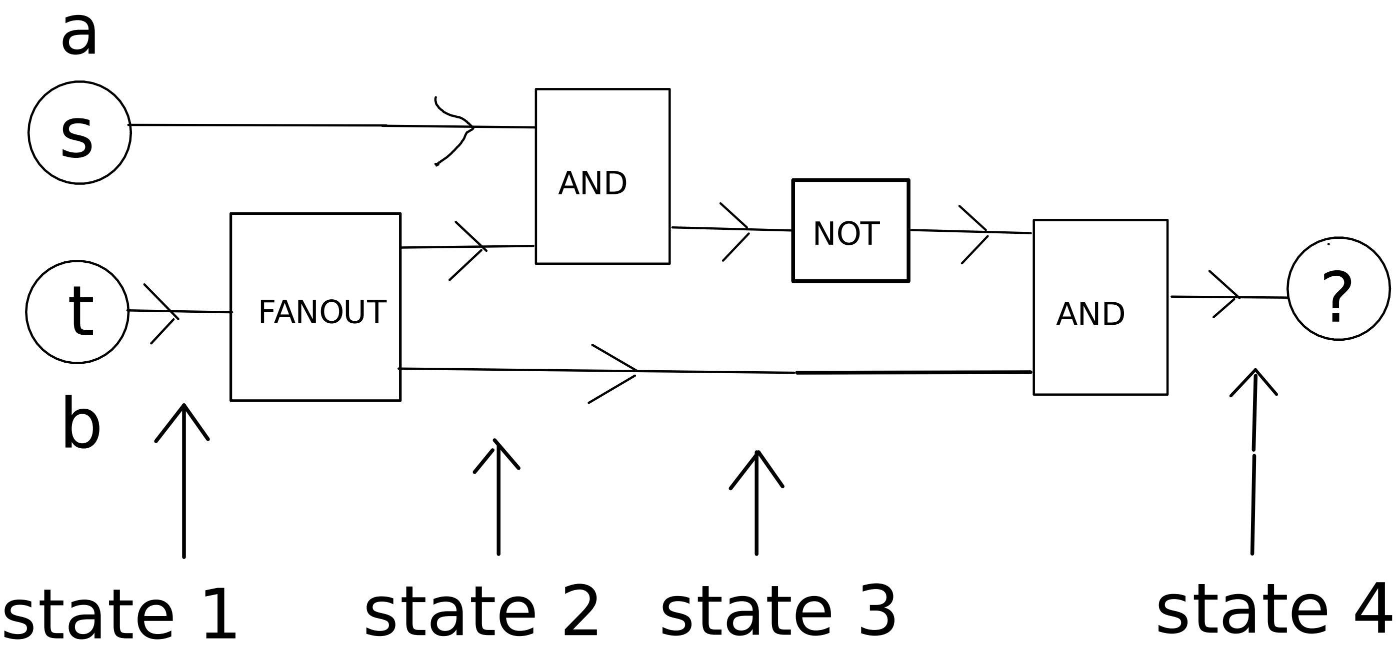
Let $A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

state 1 = $a \otimes b$

state 2 = $I \otimes F \; (a \otimes b)$

state 3 = $(A \otimes I) \; (I \otimes F) \; (a \otimes b)$

Example:



What matrix corresponds to the transformation by the entire circuit? Let $A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

state 1 = $a \otimes b$

state 2 = $I \otimes F \; (a \otimes b)$

state 3 = $(A \otimes I) \; (I \otimes F) \; (a \otimes b)$

state 4 = $A \; (X \otimes I) \; (A \otimes I) \; (I \otimes F) \; (a \otimes b)$

For all a and b, the circuit takes

$$a \otimes b \quad \text{to} \quad A\ (X \otimes I)\ (A \otimes I)\ (I \otimes F)\ (a \otimes b)$$

The transformation is thus given by the matrix

$$A\ (X \otimes I)\ (A \otimes I)\ (I \otimes F)$$

$$= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right) \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\left(\begin{array}{c|c} 0 \cdot I & 1 \cdot I \\ \hline 1 \cdot I & 0 \cdot I \end{array}\right)$$

Summary:

matrix corresponding to the i-th gate

1. From a circuit, a gate on the i-th bit transforms the joint vector by $M_i = I \otimes A \otimes I$

on bits 1, 2, ..., i-1

on bits i+1, i+2, ...

on the i-th bit

Summary:

1. From a circuit, a gate on the i-th bit transforms the joint vector by

$$M_i = \underbrace{I}_{\text{on bits 1, 2, ..., i-1}} \otimes \overbrace{\underbrace{A}_{\text{on the i-th bit}}}^{\text{matrix corresponding to the i-th gate}} \otimes \underbrace{I}_{\text{on bits i+1, i+2, ...}}$$

2. We multiple the matrices corresponding to the gates in the circuit to get the transformation by the circuit:

$$\underbrace{M_r}_{\text{last gate}} \dots \underbrace{M_2}_{\text{second gate}} \underbrace{M_1}_{\text{first gate}}$$

2. Summary of theory of classical computation

✓ (a) Bits, gates, circuits, universality, complexity
(In class, NC 3.1.2, KLM 1.3)

✓ (b) The linear algebra formalism (classical)
(In class, KLM 1.4)

(c) Models of computation
(Turing Machines, randomized computation)
(Optional reading, NC 3.1, KLM 1.2)

(d) Reversible computation
(KLM 1.5, NC 3.2.5)

Probabilistic computation:

The input bits are now given according to a distribution.   For example, for 1 bit, the input vector is now : $\begin{pmatrix} p_0 \\ p_1 \end{pmatrix}$

and similarly for more input bits.

Probabilistic computation:

The input bits are now given according to a distribution. For example, for 1 bit, the input vector is now :

$$\begin{pmatrix} p_0 \\ p_1 \end{pmatrix}$$

and similarly for more input bits.

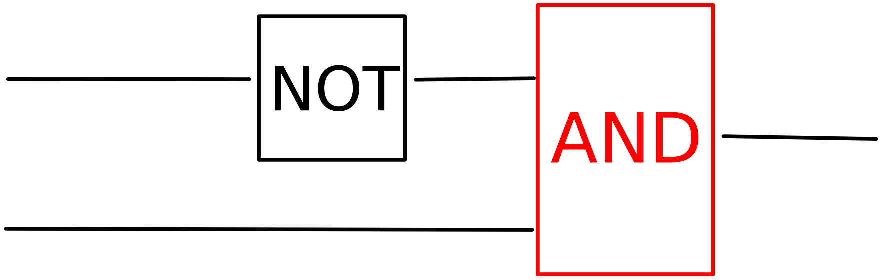
Theorem: the matrix representation for the circuit transforms the input distribution (the input vector) to the output distribution (the output vector).

Reason: the matrix representation of the circuit acts LINEARLY on the vector representation of the input.

Proof: exercise.

Example:

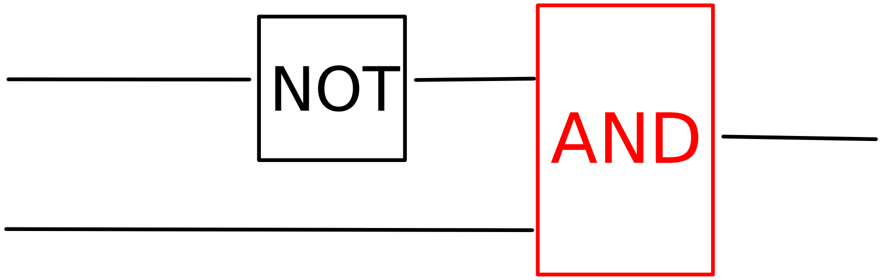
Probabilities of 00, 01, 10, 11 are 1/2, 1/3, 0, 1/6 resp.



The output distribution is:

Example:

Probabilities of 00, 01, 10, 11 are 1/2, 1/3, 0, 1/6 resp.



output = 0
with pr 2/3
output = 1
with pr 1/3

The output distribution is:

$$\underbrace{\begin{pmatrix}1&1&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{pmatrix}}_{\text{matrix rep for the circuit}}\underbrace{\begin{pmatrix}1/2\\1/3\\0\\1/6\end{pmatrix}}_{\text{input distribution}} = \begin{pmatrix}1&0&1&1\\0&1&0&0\end{pmatrix}\begin{pmatrix}1/2\\1/3\\0\\1/6\end{pmatrix} = \begin{pmatrix}2/3\\1/3\end{pmatrix}$$

Reversible computation:

We will learn that quantum mechanical evolution is reversible. But many classical gates are not reversible.

Question: can a quantum computer perform any classical computation?

Reversible computation:

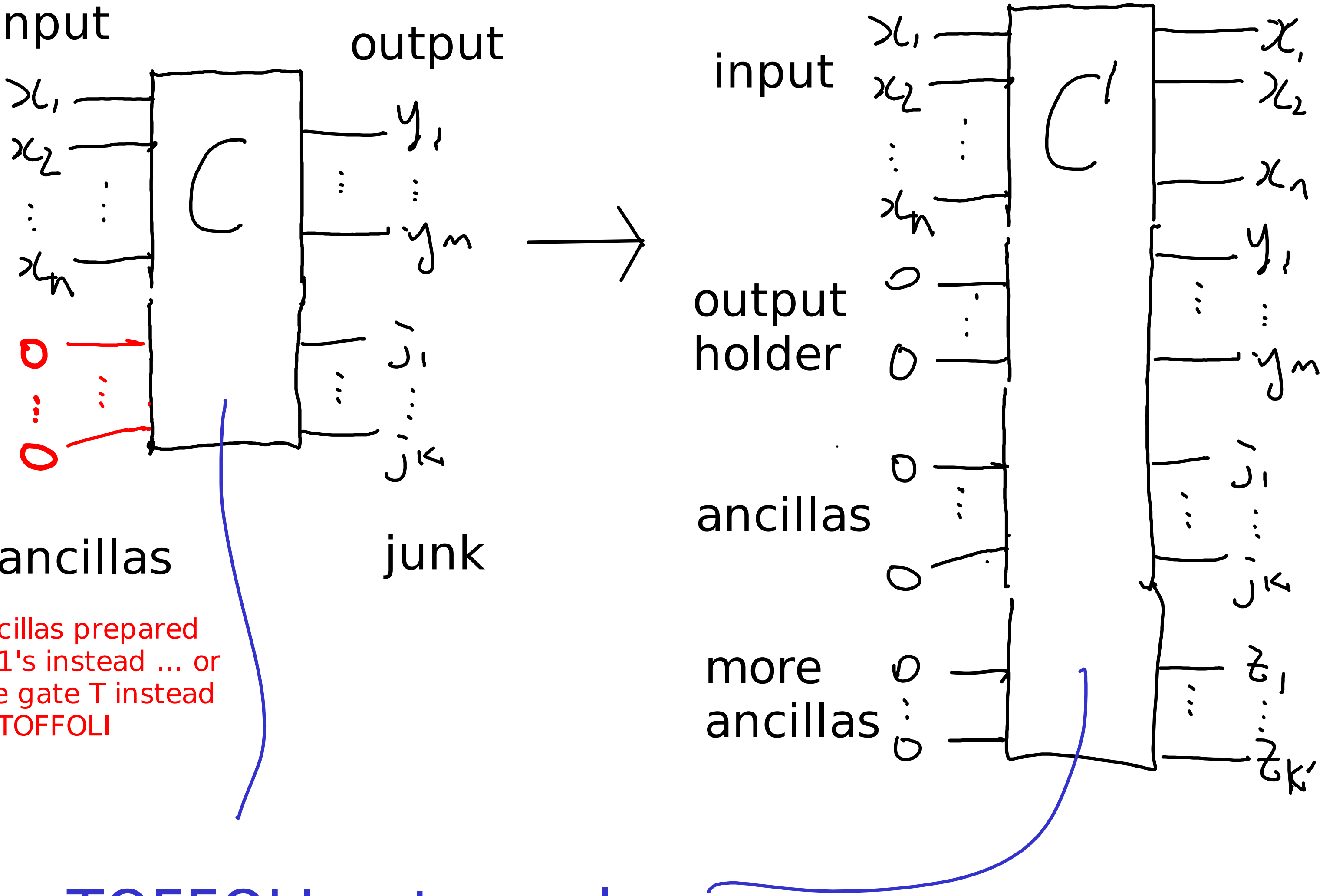
We will learn that quantum mechanical evolution is reversible. But many classical gates are not reversible.

Question: can a quantum computer perform any classical computation?

Answer: yes, luckily!

Idea: use the universal gate set {TOFFOLI} where the gate TOFFOLI is self-inverse.

Reversible computation:



input

output

ancillas

junk

ancillas prepared
in 1's instead ... or
the gate T instead
of TOFFOLI

input

output
holder

ancillas

more
ancillas

input
copied

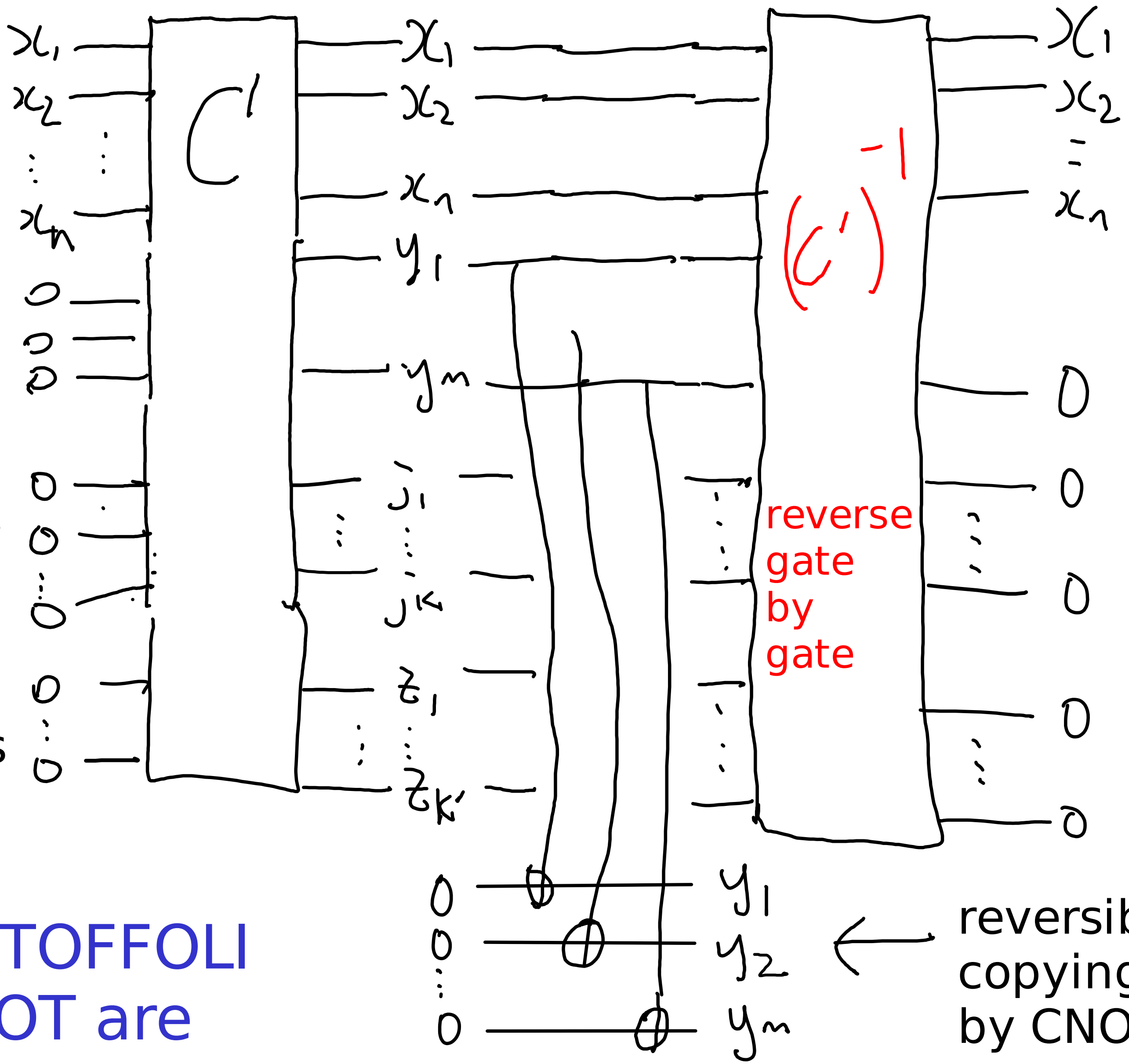
output

junk

more
junk

TOFFOLI gates only

Reversible computation (canonical):



input

output holder

ancillas

more ancillas

reverse gate by gate

input copied

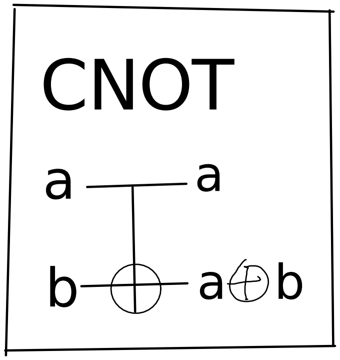
no more junk

← reversible copying by CNOTS

output

Both TOFFOLI & CNOT are reversible !

CNOT



Overhead:

An irreversible circuit C with width w and depth d can be turned into a reversible circuit C" with width O(wd) and depth d, copying the output and cleaning roughly preserves the width and doubles the depth.

Bennett 73: much more efficient reversible versions

depth $O(d^{1+\epsilon})$ and width O(w log(d))

or

depth O(d) and width $O(wd^{\epsilon})$.

NB. depth = time, width = space in KLM.

Summary for topic 2:

- Bits, gates, circuits, width, depth, size, universality
  - corollary: reversible computation

- linear algebraic representation of bit strings & gates
  - corollary: probabilistic computation

These results are readily extended to the quantum setting.