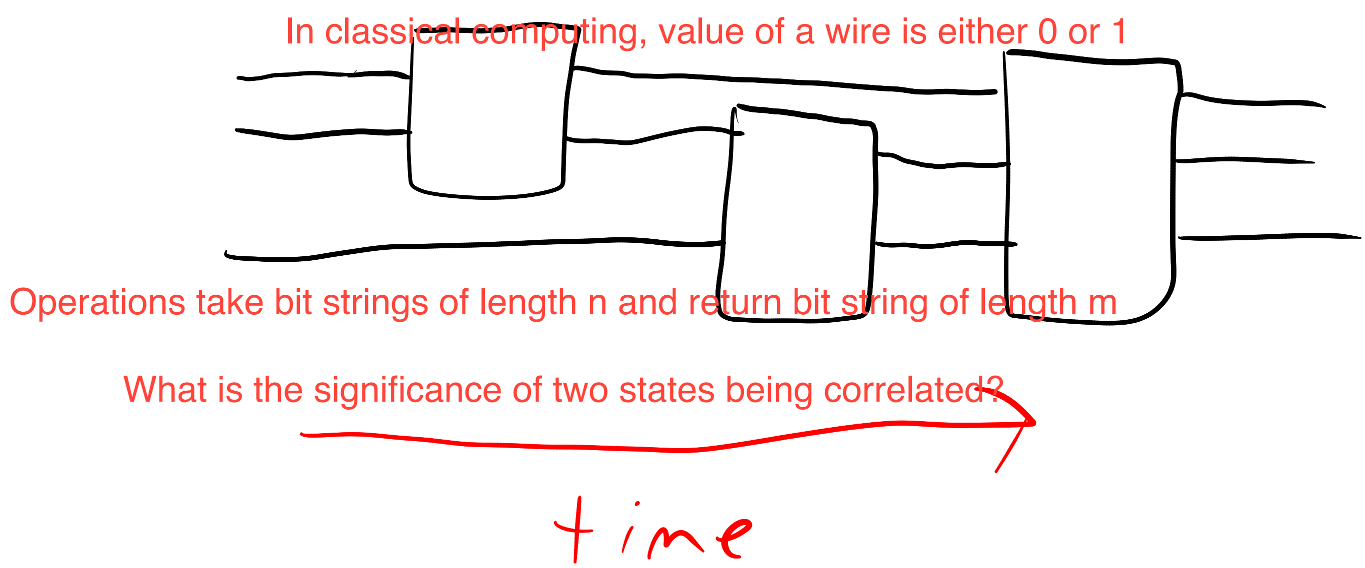
CMPT 476 Lecture 2

... The circuit model ...

Last class we learned that quantum computation is linear (i.e. matrices & vectors). Today we'll build a linear algebraic model of classical computing which will extend nicely to probabilistic and then quantum computing.

(The circuit model)

Circuit models are simple (but powerful) models of computation based around composition of a set of basic operations called gates. Wires are used to connect inputs & outputs of gates. We draw circuits graphically as below



(Classical circuits)

In the classical circuit model...

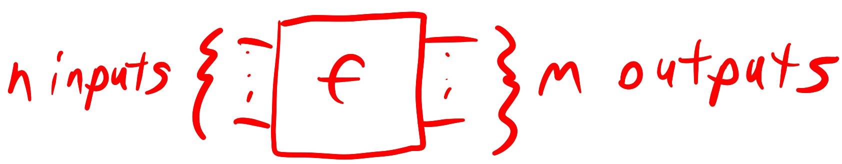
- The state of a bit/wire is 0 or 1
- The state of n bits is a bitstring

$$x \in \{0,1\}^n$$

- Computations are functions

$$f : \{0,1\}^n \longrightarrow \{0,1\}^m$$

As a gate,



Ex.

The NOT gate is a 1-bit function which computes the Boolean not ($\neg$): $\mathrm{NOT}(x) = \neg x$.

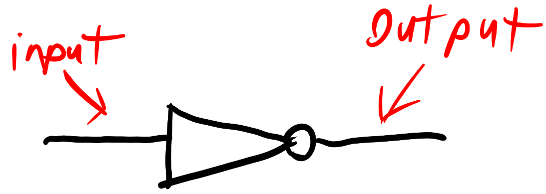
We can write the function explicitly via a truth table.

input →

x	$\mathrm{NOT}(x)$
0	1
1	0

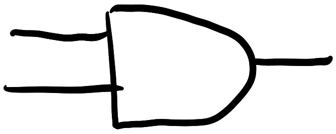
← output

We draw a NOT gate as

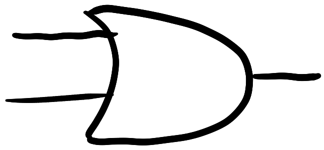


Ex.

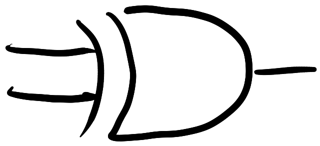
Other common gates are:



$AND(x,y) = x \wedge y$



$OR(x,y) = x \vee y$



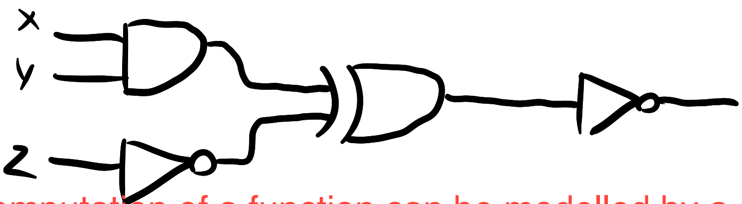
$XOR(x,y) = x \oplus y$

Their truth tables are

x	y	AND(x,y)	OR(x,y)	XOR(x,y)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Ex.

What function does this circuit implement?



Computation of a function can be modelled by a series of gates

We list out each intermediate value in the truth table:

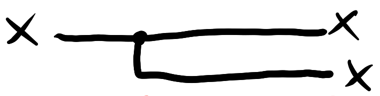
x	y	z	a = AND(x,y)	b = NOT(z)	c = XOR(a,b)	NOT(c)
0	0	0	0	1	1	0
0	0	1	0	0	0	1
0	1	0	0	1	1	0
0	1	1	0	0	0	1
1	0	0	0	1	1	0
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	1	1	1	0	1	0

(Universality)

A set of gates Γ is **universal for classical computation** if for any $n, m > 0$ and $f : \{0,1\}^n \to \{0,1\}^m$, a circuit computing f can be constructed using only gates in Γ.

(FANOUT)

In classical computing we often assume we can use a bit any number of times in a computation. Formally, this is achieved through the **FANOUT** or **copy** gate



ents, do we assume that using fanout on a bit automatically renders that bit null?

Thm.

How can you make a copy without measuring the state?

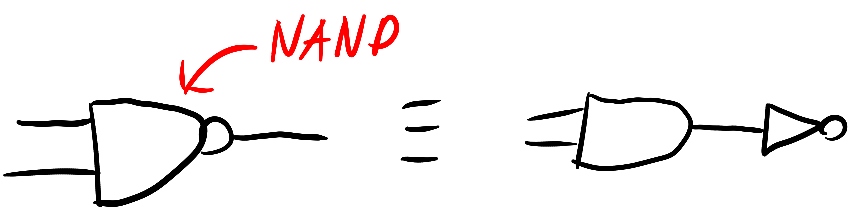
The set $\{$AND, XOR, NOT, FANOUT$\}$ is universal.

XOR gate is preferred over OR gate.
why?

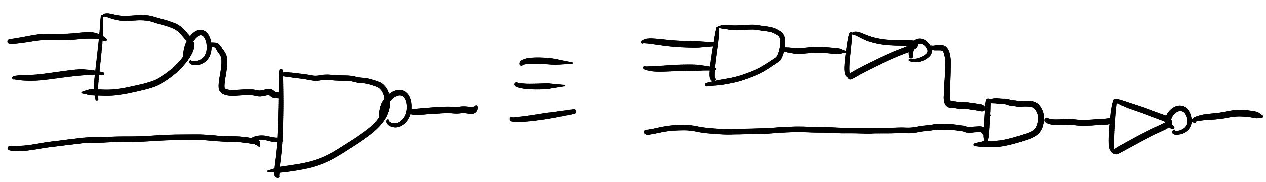
(Translating between gate sets)

We can **translate** circuits written in one gate set to another by replacing each gate with an equivalent **circuit.**

E.x.



ate NAND gates to a composition of AND and NOT gates

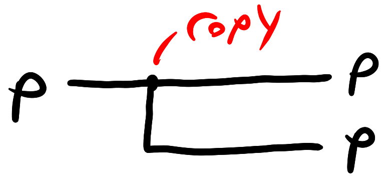


(Probabilistic circuits)

(hence quantum)

What if we wanted to model probabilistic computation? We could say a bit with probability $p \in [0,1]$ of being "1" has state p. Does this work?

You talk about the complexity of a circuit with respect to a gate set.



The final state above has probabilities

$$00 \rightarrow (1-p)(1-p)$$

$$01 \rightarrow (1-p)\,p$$

e destructive interference that gets rid of the 10 and 01 states?

$$10 \rightarrow p(1-p)$$

Since fanout only makes a copy of a bit, it should not give us states with differing bits

$$11 \rightarrow p^2$$

To represent probababilistic computations, we need to change the way that we think

BUT the states 01 and 10 are impossible!

The problem here is we can't express joint prob distributions. For this we need more degrees of freedom in our state description!

(Linear algebraic circuits)

In the linear algebraic view, we can represent a bit with probability p in the 1 state as

$$\begin{bmatrix} 1-p \\ p \end{bmatrix}$$

If $p=0$, then we have state $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, or 0, and if $p=1$, then we have state $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ or 1.

Equivalently, we can describe the state as

$$(1-p)\begin{bmatrix} 1 \\ 0 \end{bmatrix} + p\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

(Linear algebraic gates)

Suppose we apply NOT to the probabilistic state $\begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$. Then we have probability p_1 of being 1 and p_2 of being 0, or $\begin{bmatrix} p_2 \\ p_1 \end{bmatrix}$. This transformation can be described as a transition matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} p_2 \\ p_1 \end{bmatrix}$$

Note also that

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

So $\text{NOT} \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ sends 0 to 1 and vice versa

(Multiple bits)

Given two bits $\begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$ and $\begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$ we can write their joint probability distribution as

$$\begin{array}{l} \begin{bmatrix} p_1 q_1 \\ p_1 q_2 \\ p_2 q_1 \\ p_2 q_2 \end{bmatrix} \end{array} \begin{array}{l} \longleftarrow 00 \\ \longleftarrow 01 \\ \longleftarrow 10 \\ \longleftarrow 11 \end{array}$$

This is called the tensor product

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \otimes \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} p_1 \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \\ p_2 \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} p_1 q_1 \\ p_1 q_2 \\ p_2 q_1 \\ p_2 q_2 \end{bmatrix}$$

(Correlated distributions)

A distribution $\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ is correlated if it can't be written as a tensor product $\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \otimes \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$. Otherwise we say it is separable.

Ex.

The CNOT or controlled-NOT gate takes 2 bits and applies NOT to the second if and only if the first is 1. As a matrix,

$$\begin{array}{c} \text{input} \rightarrow \\ \\ \end{array} \begin{array}{cc} \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} & \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} & \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} \leftarrow \text{output} \end{array}$$

0 state

Applying CNOT to the state $\begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ gives

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.25 \\ 0 \\ 0.75 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0 \\ 0 \\ 0.75 \end{bmatrix} \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array}$$

Now suppose

$$\begin{bmatrix} 0.25 \\ 0 \\ 0 \\ 0.75 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$$

Then $ac = 0.25$, $ad = 0 \Rightarrow d = 0$, but then $bd = 0$, a contradiction, so the distribution is correlated.

(A note on vectors & matrices)

We used the term distribution informally. Formally, a vector (i.e. state) $p \in \mathbb{R}^n$ (real vector space of dim n) is a distribution on $\{0, \ldots, n-1\}$ or simply a probability vector if

1. $p_i \geq 0$ for all i
2. $\sum_i p_i = 1$ ← Note: this is the 1-norm

If states are probability vectors, then gates should map distributions to distributions. These are exactly the stochastic matrices A, which have as columns A_i probability vectors.

Ex.

$$A = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

is a stochastic matrix modeling a coin flip. Calling $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ heads and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ tails, we have

$$A\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} = \tfrac{1}{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \tfrac{1}{2}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$A\begin{bmatrix} p \\ q \end{bmatrix} = \frac{p+q}{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{p+q}{2}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

the input state is irrelevant!

In quantum computing, our allowable operations will take on a very similar restriction :)

hat is universality?
t is the circuit model?