Lecture 3
Circuit models of computation

Circuit models are simplistic (but powerful) models of computation based around composition of a (typically) finite set of basic operations called gates. Wires are used to compose gates by connecting inputs & outputs.

(Classical circuits)

In the classical circuit model,

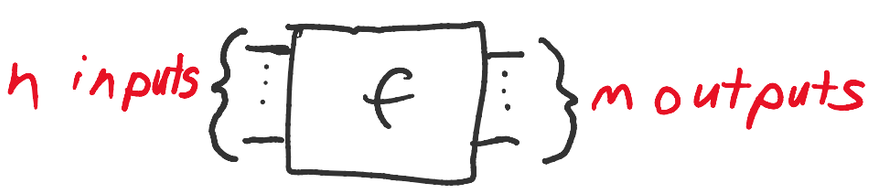
- The state of a bit is 0 or 1
- The state of n bits is a bitstring

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

- Computations are functions

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

We describe functions graphically via circuit diagrams



A **gate** computes a fixed function with a fixed number of inputs and outputs

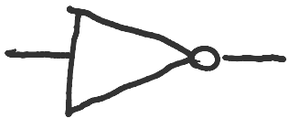
Ex.

The **NOT** gate computes the 1-bit function

$$f(a) = \neg a = 1 \oplus a$$

↑ addition mod 2: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$

We draw the NOT gate as



Other gates are

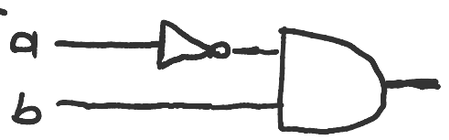
 (AND gate, $f(a,b) = ab$)

 (XOR gate, $f(a,b) = a \oplus b$)

(Note: AND and XOR are multiplication and addition resp. in the Boolean field $\mathbb{Z}_2$ or $\mathbb{F}_2$)

Circuits are created by connecting the input and output wires

Ex.



$(1 \oplus a)b = b \oplus ab$ ← distributivity of mult.

In the classical circuit model, we typically assume we can **copy** a bit, called **FANOUT**

(FANOUT)

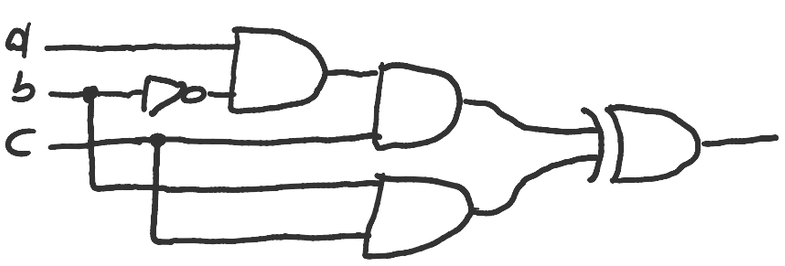
Universality

A finite set G of gates is **universal** if any **Boolean function** $f: \{0,1\}^n \to \{0,1\}$ can be written as a circuit using only gates taken from G

Ex.

Let $f(a,b,c) = a(1 \oplus b)c \oplus bc$

A circuit for f over $\{AND, NOT, XOR, FANOUT\}$ is



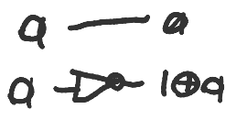
Thm.

$\{AND, XOR, NOT, FANOUT\}$ is universal

Proof

By induction on the number of inputs n

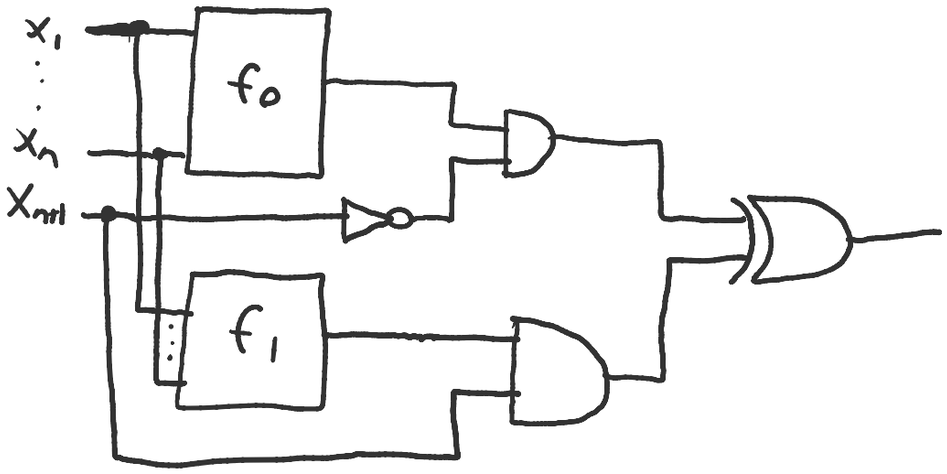
For $n=1$, there are 2 unique functions:



For $n+1$ inputs, $n \geq 1$, we have the following equality

$$f(x_1, \ldots, x_n, x_{n+1}) = (1 \oplus x_{n+1}) f_0(x_1, \ldots, x_n) \oplus x_n f_1(x_1, \ldots, x_n)$$

We can thus implement f as follows



By the hypothesis, f_0 & f_1 can each be written over the gate set, so we're done □

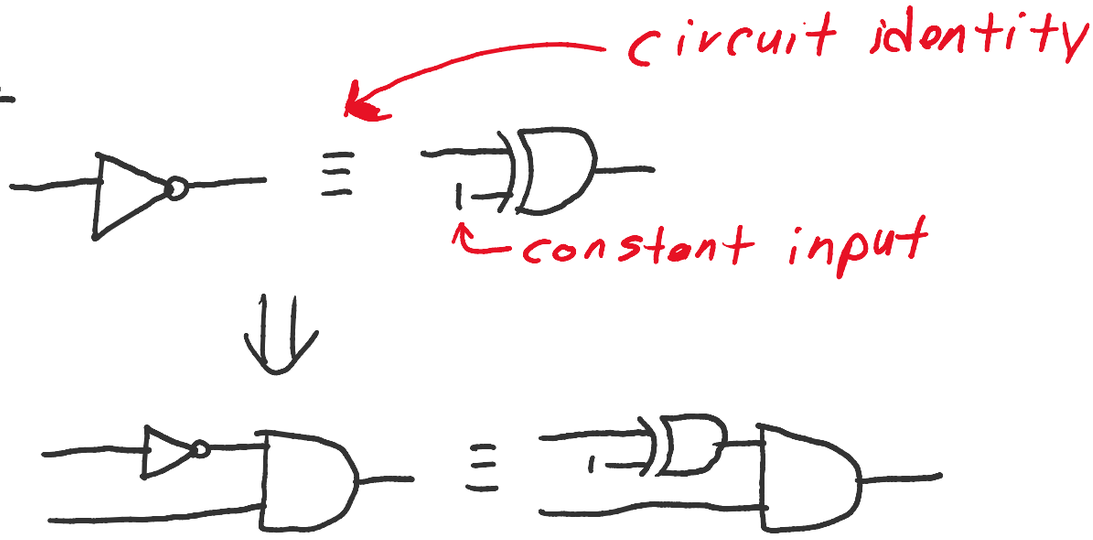
Note:

f_0 & f_1 are called **cofactors** of f, and the expression for f via its cofactors is called the **Shannon expansion** of f

(Translating between gate sets)

We can translate circuits written in one gate set to another gate set by replacing gates with equivalent circuits

Ex.

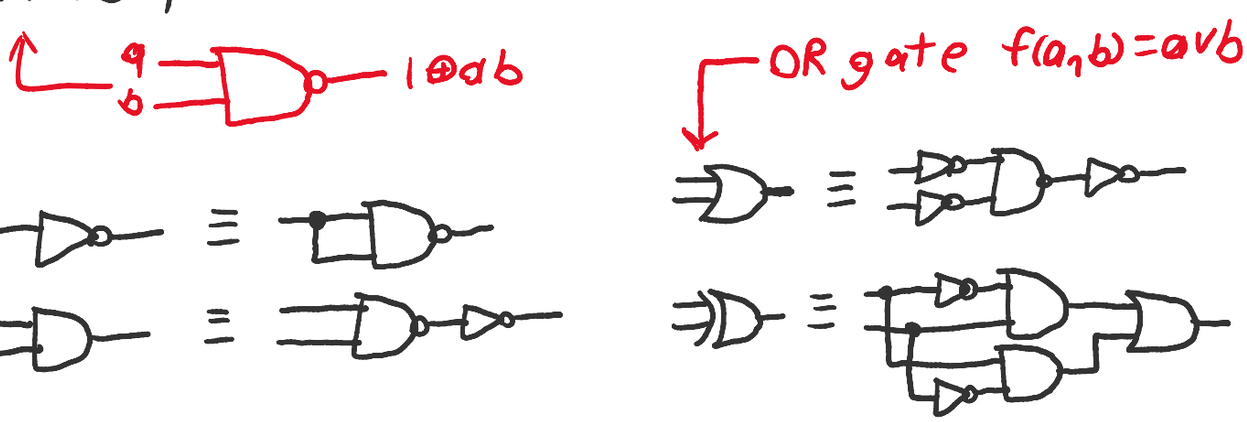


Given two gate sets G_1 & G_2, if G_1 is universal and every gate of G_1 can be written as a circuit over G_2, then G_2 is also universal

Ex.

$\{AND, XOR, FANOUT\}$ is universal by the above

$\{NAND, FANOUT\}$ is universal since



Aside: interpretations

Previously we **interpreted** circuits using the **Boolean field** $(\mathbb{F}_2, \cdot, \oplus)$ (note that $\{0,1\}^n = \mathbb{F}_2^n$).

Eg. $[\![\text{AND}]\!] = \cdot$ ← "meaning" of a gate in the interpretation

$[\![\text{XOR}]\!] = \oplus$

We could instead interpret circuits using linear algebra by identifying the state of n bits with a unit vector $|x_1 \cdots x_n\rangle \in \mathbb{F}_2^{2^n}$ and gates as linear operators from $\mathbb{F}_2^{2^n} \to \mathbb{F}_2^{2^m}$

Ex. in the linear interpretation,

$$[\![\text{NOT}]\!] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ since } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} |a\rangle = |1 \oplus a\rangle$$

$$[\![\text{AND}]\!] = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that $\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} |01\rangle = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} |11\rangle = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Reversible computation

- Recall that physical processes are reversible
- The AND gate is not reversible, in the sense that we can't retrieve a & b from $a \cdot b$

  e.g. if $a \cdot b = 0$, then what are a and b?

  $\Rightarrow$ information is lost!

(Landauer's principle)

Erasing a single bit of energy dissipates at least $k_B T \ln 2$ of energy into the environment

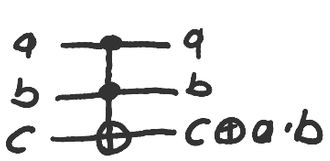
(A reversible AND gate?)

The AND function can be computed "reversibly" by saving the inputs:

If we think of a physical process operating on a finite memory, this is not reversible because it may involve overwriting some other bit to store $a \cdot b$

(The Toffoli gate)

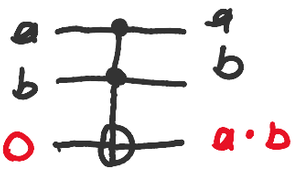
Tommaso Toffoli (1980) proposed the Toffoli gate



as a matrix on $\mathbb{F}_2^3$

as a reversible AND gate completely conserving information (and hence reversible)

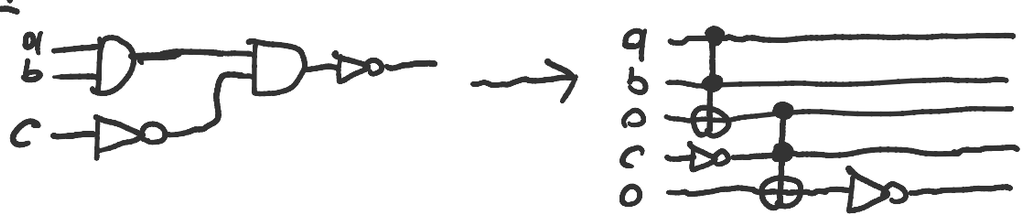
Using the Toffoli gate, the AND gate can be implemented with the aid of an ancilla (a bit in a known state)



(From irreversible to reversible)

we can make a classical circuit reversible by replacing each gate with a reversible analogue, given enough ancillas

Ex.

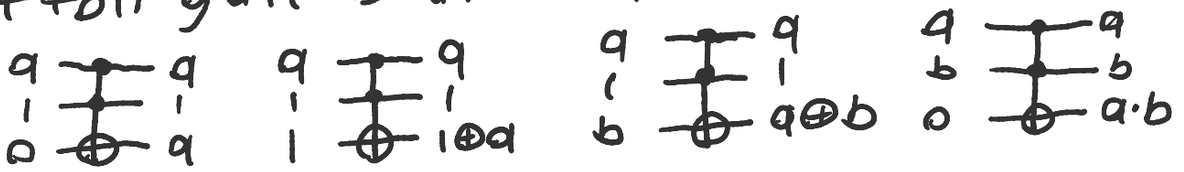


Thm.

A classical circuit running in space S and time T can be simulated by a reversible circuit running in space $O(S+T)$ and time $O(T)$

Pf.

Toffoli gate is universal:

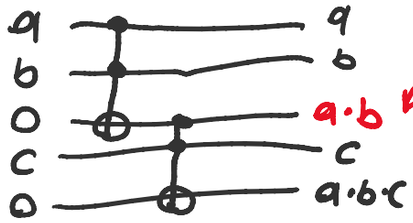


Replacing each gate over a finite gate set takes constant time & space overhead □

Uncomputation

- Reversible computation keeps allocating more and more space to store temporary values

E.g.

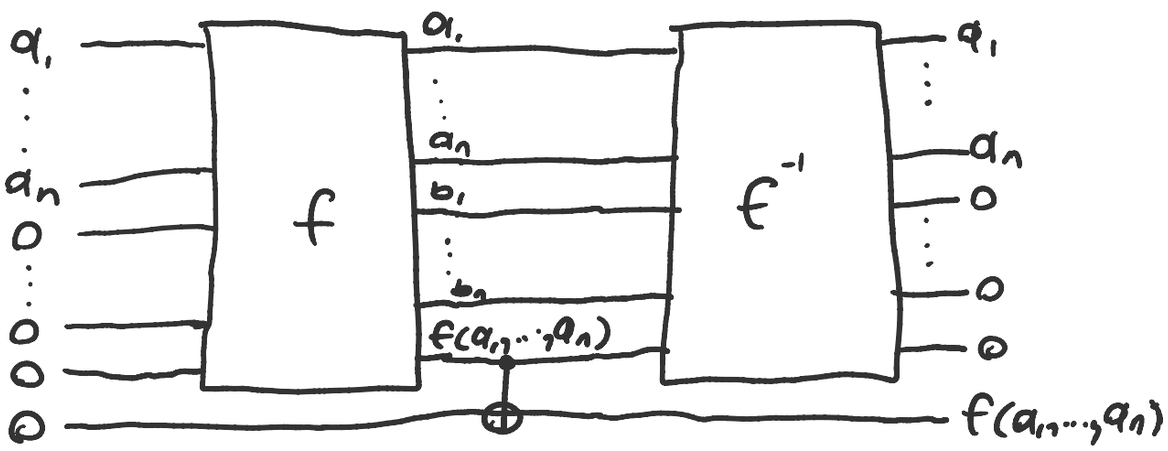


not needed after computing $a \cdot b \cdot c$

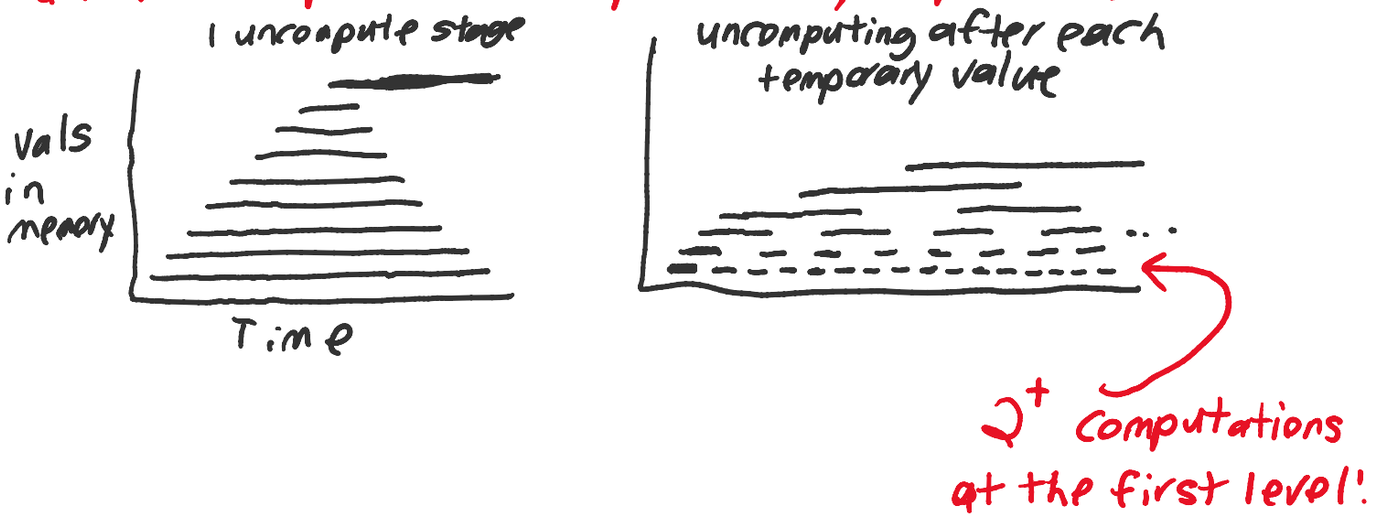
- Need some way to reclaim space without dissipating energy

(The Bennett trick)

After a computation is done, copy the output and reverse the computation to "uncompute" any temporary values.



By intermittently uncomputing, space can be reduced at the expense of (potentially exponential) extra time



Thm. (Bennett 1989)

There exist pebbling strategies to simulate an irreversible computation with time T and space S reversibly with

- Time $O(T^{1+\varepsilon})$ and space $O(S \log T)$
- Time $O(T)$ and space $O(ST^{\varepsilon})$

(Project idea: read Bennett's paper and implement or analyze his pebble games for a concrete class of circuits)