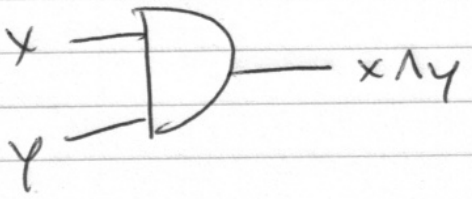
Lecture 10

- circuit model of computation

- a circuit is a directed, acyclic graph

Examples of basic logic gates

AND



truth table

x	y	x∧y
0	0	0
0	1	0
1	0	0
1	1	1

Notice that the gate is irreversible

i.e., from outputs we cannot figure out inputs (0 @ output has 00, 01, & 10 as input)

Thus information is erased when applying this logic gate

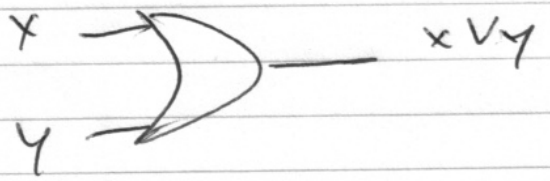
Aside: Landauer established a fundamental connection between physics & information w/ his erasure principle:

erasing 1 bit of information leads to $\geq$ (at least) $kT \ln 2$ units of heat to be dissipated to the environment

$\Rightarrow$ q. computing, we are interested in reversible gates in order to correspond w/ unitary evolution & to maintain superpositions

Other basic gates

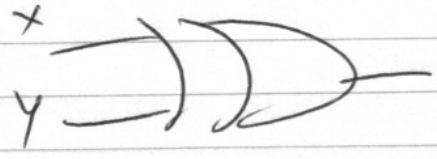
OR



x	y	x∨y
0	0	0
0	1	1
1	0	1
1	1	1

irreversible

XOR

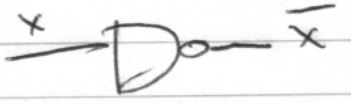


x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

irreversible

binary addition
can be made reversible by
copying input x to output
& then we get CNOT gate.

NOT



x	x̄
0	1
1	0

reversible

Fundamental Universality Theorem for computation w/ circuits:

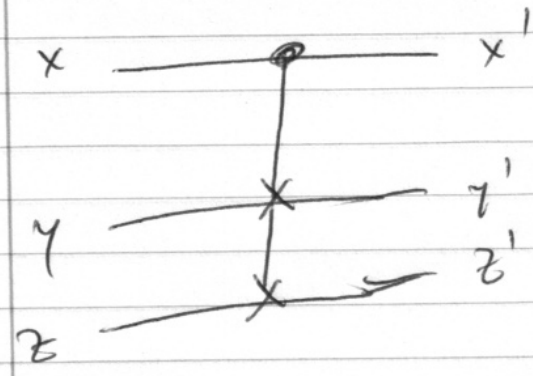
can compute any Boolean function $f: \{0,1\}^n \to \{0,1\}^m$

using a circuit composed of the aforementioned gates, in addition to copy & swap.

Important caveat: this theorem does not imply that every function is efficiently computable.

So we need to separate those functions that are efficiently computable from those that are not.

Let us now discuss reversible computation. Basic logic gate here is the controlled SWAP (c-SWAP)



if 1st bit is zero, do nothing.
otherwise swap 2nd & 3rd bits.

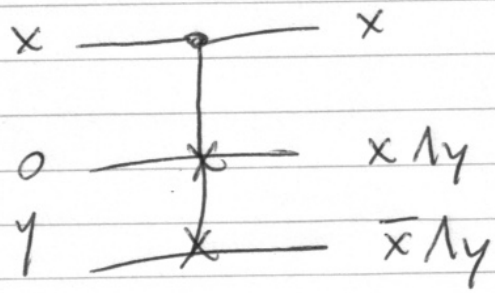
truth table

x y z	x' y' z'
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	1 1 0
1 1 0	1 0 1
1 1 1	1 1 1

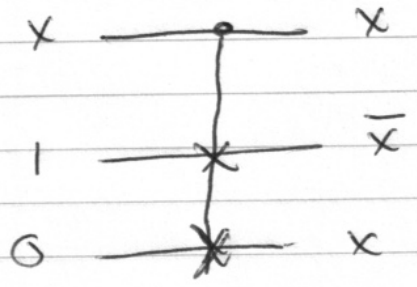
reversible gate & in fact we already used the quantum version

thus c-SWAP gate is self-inverse

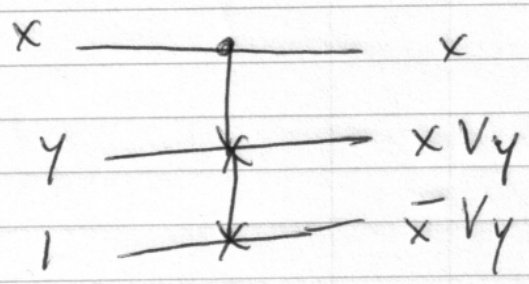
it can simulate AND as follows:



can simulate NOT



simulate OR



so AND & NOT can be simulated w/ c-SWAP & vice versa

we can always thus promote a general circuit to a reversible one, by replacing every gate w/ c-SWAP

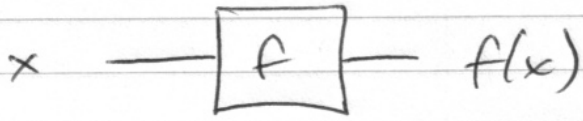
- this is a general principle that we can also use when promoting classical Boolean circuits to quantum circuits, which can be used in basic q. algorithms like Deutsch-Jozsa & Grover
- idea is to 1st make circuit reversible by replacing w/ c-SWAPs & then by understanding c-SWAPs as q. c-SWAPs, we have a q. circuit.

- However, when doing so, we end up w/ unwanted "garbage" bits.
- In the q. case, we have to clean them up & the way to do so is called uncomputing

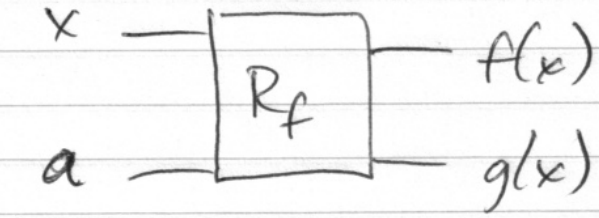
Observe in the examples above that we end up w/ extra bits besides inputs & results of computation. These are called garbage bits.

How to do this?

Model original circuit as



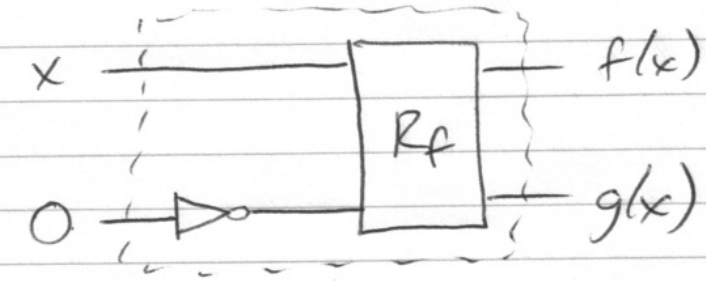
After being promoted to reversible circuit, it has the form



↑ auxiliary bits (required to be initialized to certain values in the simulations presented)

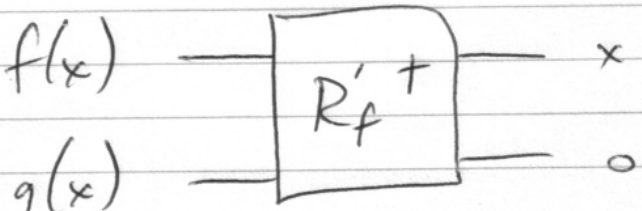
↑ garbage bits

can realize values of a via NOT gates acting on 0



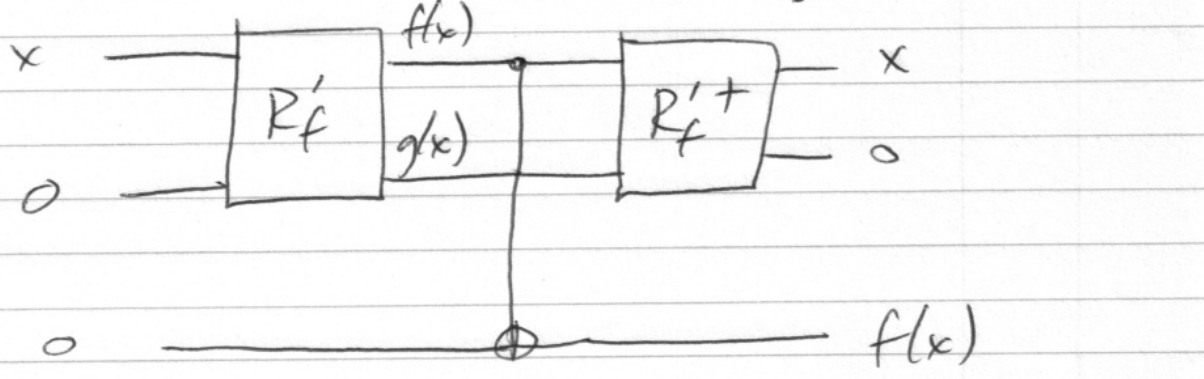
R_f'

to clean up, consider the following



↑ reverse of R_f'

then full circuit is



↑ sequence of CNOT gates to copy the answer

We can then consider this to be a q. circuit, call it U_f, w/ the following action on standard basis states

$$U_f |x\rangle|0\rangle|0\rangle = |x\rangle|0\rangle|f(x)\rangle$$

It acts on superpositions as follows

$$U_f |+\rangle^{\otimes n} |0\rangle|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} U_f |x\rangle|0\rangle|0\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_x |x\rangle|0\rangle|f(x)\rangle$$

the entanglement between 1st & 3rd registers is the right kind of entanglement for q. algorithms

If we had not done the cleanup, the resulting state from 1st R_f' + CNOTs would be

$$|+\rangle^{\otimes n} |0\rangle|0\rangle \rightarrow$$

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle |g(x)\rangle |f(x)\rangle$$

leaving undesired entanglement w/ 2nd register of garbage bits + would mess up q. algorithms that call functions like this.