<u>Meeting 6.1.:</u> Classical warm ups to quantum computing

I. Probabilistic classical computing: BPP and MA

II. Reversible classical computing

Next time: quantum circuits, BQP and QMA

Axioms of quantum mechanics lead to important problems if we want to use quantum mechanical systems to build a computer, even in the ideal case of a noise-less system:

1. The classical information extracted via measurement is a probability distribution. How can we formalize complexity theory around this?
2. If we want to use a Hilbert space of some system as a "quantum memory register," then (quantum) transformations must be unitary, hence, in particular, invertible/reversible. Is reversible computation feasible?

<u>Goal today</u>: answer these questions in classical warm-up cases.

The classical analogs won't address all of the issues in the quantum case. E.g. quantum states are not "just" classical probability distributions, and unitary group $U(n)$ is uncountably infinite.

Also important later: non-idealized quantum computing. Need a theory of quantum error correction and fault tolerance.

I. Probabilistic classical computing

Informally: a classical probabilistic algorithm is any algorithm that is allowed access to coin flips, or, equivalently, random bit strings.

Two equivalent ways to make this more formal:

1. Extend the definition of Turing machine so the transition function can, in addition to using the machine's internal state and read of the memory, toss a fair coin.
2. "Resolve" a non-deterministic Turing machine by flipping a coin to decide how to branch.

Remarks:

1. It doesn't matter so much if the coin is fair, but if $p(\text{heads}) \neq 1/2$, it should at least be a reasonable number....
2. Coin tosses are always independent. So our algorithm could do all of them at the beginning. Equivalent to choosing a (uniformly) random bit string, and using the bits one by one as needed.
3. Access to coin tosses does not change "computable." It might change "efficiently computable", thus violating extended Church-Turing thesis.
4. Flipping a coin counts as one time step.

A probabilistic algorithm / Turing machine for a counting problem induces, for any input x, a probability distribution on $\{0,1\}^*$.

For a decision problem, each input x yields a probability distribution on $\{0,1\} = \{\text{Yes}, \text{No}\}$.

Informally: A decision problem should be considered efficiently probabilistically solvable if there's a poly. time Turing machine that get's the correct answer with high probability.

Fix a constant $0 < \varepsilon < 1/2$. A decision problem L is in BPP_ε ("bounded-error probabilistic polynomial time") if there exists a PTM T and a polynomial $p(|x|)$ such that when input x, T terminates in at most $p(|x|)$ steps, and:

(i) if $L(x) =$ Yes, then T answers Yes w/ prob $\geq 1-\varepsilon > 1/2$.

(ii) if $L(x) =$ No, then T answers No w/ prob $\geq 1-\varepsilon > 1/2$.

So, ε is probability of a wrong answer.

Fact: For any $0 < \varepsilon < \varepsilon' < 1/2$,

$$BPP_\varepsilon = BPP_{\varepsilon'}.$$

Why? "Amplification of probability:"

$BPP_\varepsilon \subseteq BPP_{\varepsilon'}$ obvious from definition

$BPP_\varepsilon \supseteq BPP_{\varepsilon'}$: repeat (enough times) and use majority rule.

Take-away: Define $BPP = BPP_{1/3}$.

Equivalent Formulation: BPP is all decision problems L decidable by an NP TM such that at most $1/3$ of the branches report the wrong answer.

Variats: RP, PP

RP: Same as BPP_r except if the answer is Yes, the PTM always reports the correct answer.

PP: what we get if we set $\varepsilon = 1/2$.

$BPP_0 = P$. (But beware of ZPP.)

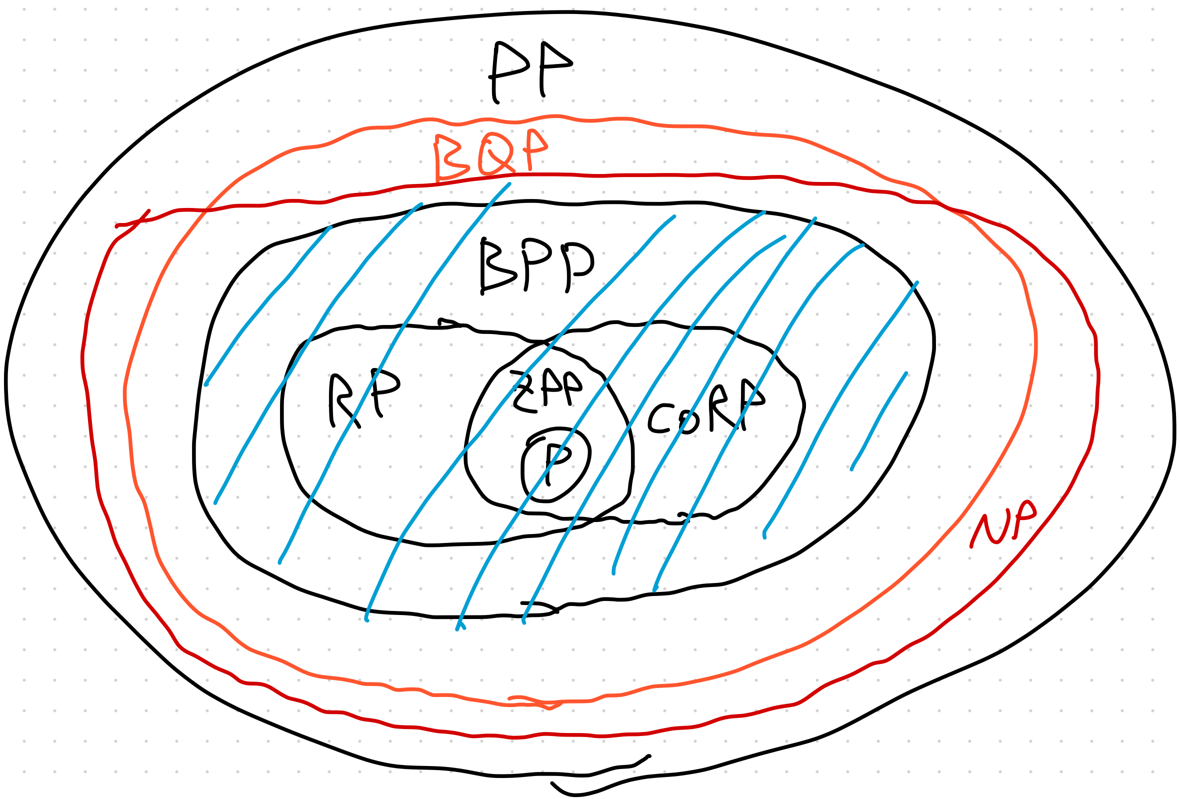
<u>Example</u>: Primality testing is in BPP via Miller-Rabin test.

Input: natural number N (in binary)

Question: Is N prime?

In Fact, it was shown to be in P.

<u>Derandomization:</u> It's expected good random number generators exist, hence BPP = P.



Merlin-Arthur: probabilistic analog of NP.

Has same definition as before, except we use a BPP Turing machine to decide when a witness is believable.

Name "Merlin-Arthur" is supposed to invoke a "game."

Multi-round (but constant) games generalize NP (or MA) to polynomial hierarchy.

II. Reversible classical computing

Classically, interested in computing Boolean functions

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

Of course, these are not all bijections. Is there a way to encode f "inside" of a bijection?

Even better, can we do this "locally" and "uniformly"?

First: CSAT.

Instance: Boolean circuit C

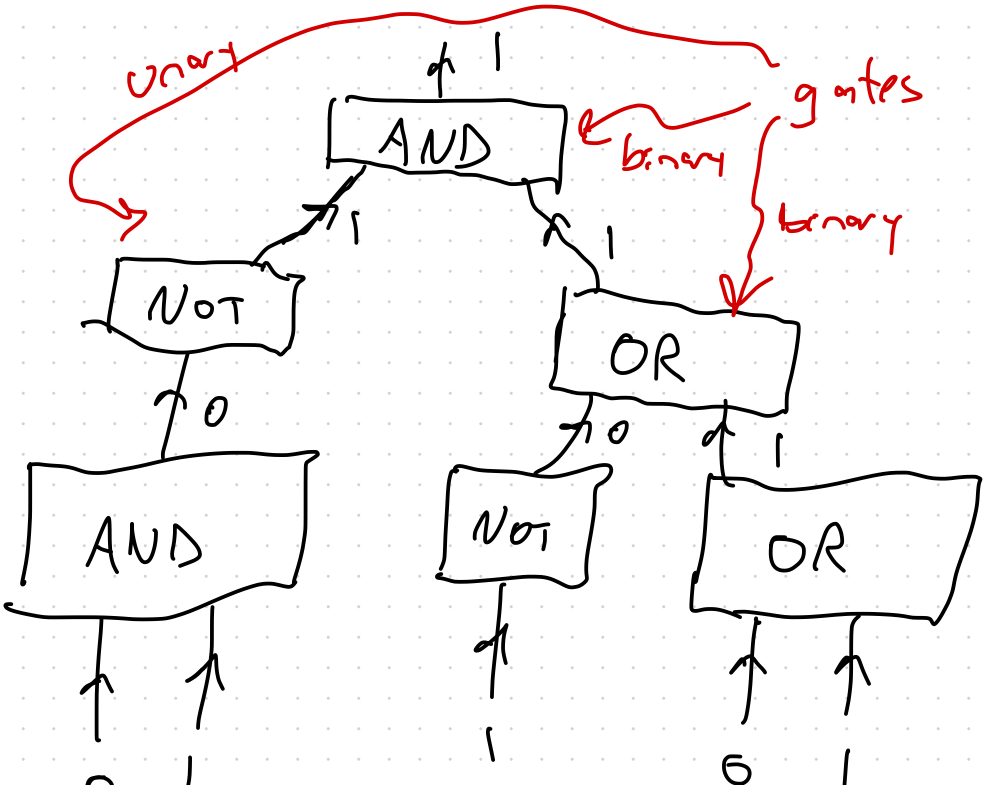
Question: Is C satisfiable?

Planar Boolean circuit is something like this:

Drawn in plane w/out crossings of wires

$\{0,1\} = \{Yes, No\}$

C :



$\{0,1\}^5$

$C(0,1,1,0,1) = 1$, so C is satisfiable.

If we have crossings, can get rid w/ a swap:

CSAT is NP-complete.

(Can reduce from SAT.)

Size of a circuit is $O(\#\text{ gates})$.

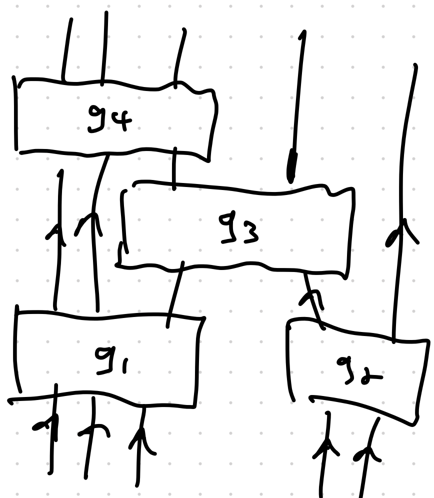
Is there some NP-complete analog of CSAT for "reversible circuits?"

Fix a <u>gate set</u> $\mathcal{G}$, which a set of bijections

$$g : \{0,1\}^n \to \{0,1\}^n$$

where n may vary with the gate.

We can wire gates from $\mathcal{G}$ to build planar reversible circuits.



$R:$

width 5

$g_i \in \mathcal{G}$.

$$R : \{0,1\}^5 \to \{0,1\}^5.$$

Q: Can we find an NP-complete problem for circuits with gate set $\mathcal{G}$? Call it RSAT($\mathcal{G}$)...

A: Depends on $\mathcal{G}$.

Why is this unclear?

Note, we can't fix $y \in \{0,1\}^n$ and ask if there exists $x \in \{0,1\}^n$ such that $R(x) = y$?

Why not? Because the answer is always Yes!

Let $\mathscr{G} = \mathrm{Sym}(\{0,1\}^3)$, and define $\mathrm{RSAT}(\mathscr{G})$ as follows:

Input: Reversible circuit R of width $2k$

Question: Does there exist $x, y \in \{0,1\}^k$ such that

$$R(x, \underbrace{0, \dots, 0}_{k}) = (y, \underbrace{0, \dots, 0}_{k})?$$

Claim: This is NP-complete.