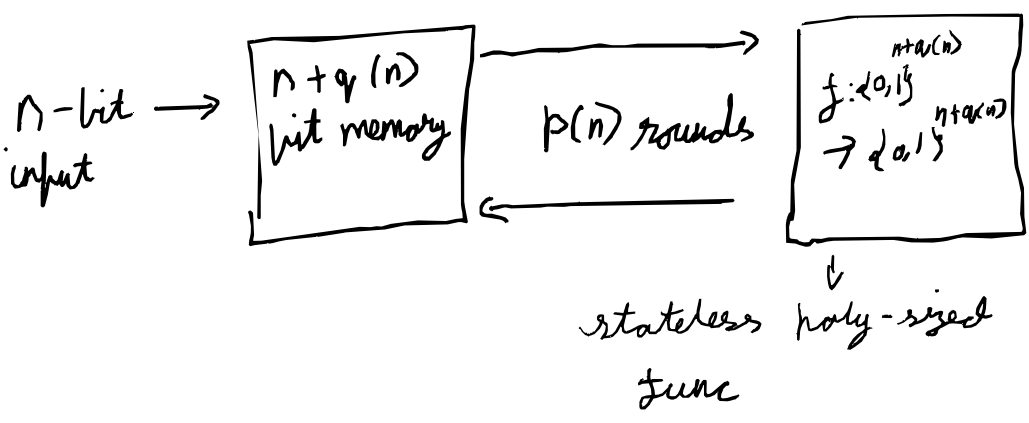
Complexity in Practice

The following are intended to be pointers, not exhaustive notes

Outline:

- Cook Levin theorem that proves SAT is NP-complete can be easily proved if programs take the following structure



p(n) & q(n) are polynomials
The algorithm uses q(n) extra bits

- Claim: Any boolean formula can be converted to Conjunctive Normal Form (CNF) aka AND-of-ORs with polynomial blow up. Converting to Disjunctive Normal Form (DNF) aka OR-of-ANDs can have exponential blow-up
  - only if you allow auxiliary variables
- Converting from CNF to ILP is easy
- ILP is harder than LP because one point can have many neighbors This is related to MIMO decoding and lattice based cryptography
- Boolean formulas are unusual in allowing conversion to CNF with auxiliary variables. This works for deciding $\exists x f(x)$ formulas, but not if you want to compute $f(x)$
  Going from depth $d \Rightarrow \log d$ can have exponential blow-up
- The same holds for formulas involving $(+, *)$ & $(\wedge, \vee)$
  This happens because + & * don't commute with each other, so reducing depth is not possible
  This does not happen with just + or * or $\vee$ or $\wedge$ or $\oplus$ or $\wedge$ or $\vee$
  However, these are not commutative or associative
  Eg. NAND, a universal gate is not
- This has general applications:
  a) Boolean circuits for e.g. adders & multipliers have a complex tradeoff
  b) Deep neural networks are better than shallow ones because shallow ones need to be exponentially larger
  For example, consider a ReLU network trying to determine if its input from $\mathbb{R}^d$ is inside the unit sphere. Consider the d-dimensional hyper-cube around the hyper-sphere It has 2^d vertices and you need to "cut" every vertex at-least once
  In a 2 layer ReLU network, this needs 2^d neurons!
- The fact that d-dimensional cubes have 2^d vertices is also related to why ILP is hard & why lattice problems are hard
  MIMO decoding in wireless networks is related to lattices