15-459: Undergraduate Quantum Computation September 13, 2022

Lecture 5 : Reversible Deterministic Computation

Lecturer: Ryan O'Donnell
Scribe: Rajeev Godse

1 Reversibility

In the scratch demos, we observed that all our operations had their inverse and their reverse being identical[1]:

- "Hat" (Hadamard) (H)
- Deterministic operations which are permutations on the state space. In Scratch, all operations were of the form "Add $F(\mathbf{x})$ to Answer"
  - For $F() = 0$, we get "Add 0 to Ans". The real name is I, the **identity**.
  - For $F() = 1$, we get "Add 1 to Ans". The real name is **NOT** or X. The diagrammatic representation is $\boxed{NOT}$ or $\oplus$.
  - For $F(x_1) = x_1$, we get "Add x_1 to Ans". The real name is Controlled-NOT or $CNOT$. In a diagram, we put a dot on the control (x_1) and $\oplus$ on the negated bit (Ans), connected by a line.
  - For $F(x_1) = \neg x_1$, get "Add (NOT x_1 to Ans)", which is not very common. The real name is C_0NOT. The diagram is the same as above but with a hollow dot on the control.
  - For $F(x_1, x_2) = x_1$ AND x_2, we get "Add (x_1 AND x_2) to Ans", which is very common. The real name is Controlled-CNOT, $CCNOT$, or Toffoli gate. The diagram uses two black dots through the controls and $\oplus$ on Ans, connected by a line.
  - For $F(x_1, x_2) = x_1$ OR x_2, we get "Add (x_1 or x_2)", which is never really used in quantum computing, so it doesn't have a real name.
  - *Important note*: There are other reversible deterministic instructions (LeftShift, RightShift) whose inverses are not themselves.

2 Computation

We are focused on "compute-a-function tasks": $F : \{0,1\}^n \to \{0,1\}^m$[2].

Examples:

- Count : $\{0,1\}^3 \to \{0,1\}$. Count(x_1, x_2, x_3) is the binary representation of the number of 1's in the input

Questions:

1. Can **every** truth table be computed by an AND/OR/NOT circuit?
2. How efficiently?

Answers

[1] In fact, all the operations we observed were their own inverses, i.e. they are involutions
[2] These functions can also be represented as truth tables

1. Yes. Observe that we can use that the number of output bits is $m = 1$: if m is greater, we can compute each bit separately. Observe further that it suffices to solve functions of the form $F(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} = \mathbf{x}_0 \\ 0 & \text{otherwise} \end{cases}$, and take the disjunction of all of the finitely many accepted inputs.

   To compute the recognition circuit for an arbitrary $\mathbf{x}_0$, just negate the bits we want set to 0 and AND together all the resulting wires.

   See that making temporary variables for each of our gates and using the "Add $F(\mathbf{x})$ to Ans" versions of each instruction, we get a reversible program computing the function (assuming that our temporary variables, corresponding to our gates are initially set to 0).

2. If you can compute a function in a classical circuit using G gates, then you can convert to reversible "Add $F(\mathbf{x})$ to Ans" code using $n + m + G$ bits and $G + m$ total instructions with the precondition that the extra variables are initialized to 0 and left in a trash state.

   Moreover, $G \leq n \cdot m \cdot 2^n$ always via the method we described of computing each bit via the truth table. Claude Shannon improved this upper bound to $m \cdot \frac{2^n}{n}$.

   Remark: If a TM can compute a function $F : \{0,1\}^n \to \{0,1\}^m$ in T steps, it can be compiled into AND/OR/NOT circuits with $G \leq \tilde{O}(T^2)$, but usually $\leq \tilde{O}(T)$.

3 Taking out the Trash

Why?

1. Waste of space! It would be great to **restore** the trash bits to $00\ldots0$, and we can then reuse those temporary variables several times.

2. "Magic" in quantum computing is that the amplitudes can cancel out to 0.

   Consider the quantum program:

   (a) Make A.

   (b) Had A. (This transforms the quantum state to Ampl[0] $= \sqrt{\frac{1}{2}}$ and Ampl[1] $= \sqrt{\frac{1}{2}}$)

   (c) Had A. (This transforms the quantum state to Ampl[0] $= 1$ and Ampl[1] $= 1$)

   (d) PrintAll (We have a 100% chance of printing $A = 0$!)

   Suppose we replaced line 2 with many instructions that all clean up their temporary variables. Then note that we get the same behavior as above where we can cleanly extract the value of A with amplitude 1.

   On the other hand, if we didn't clean up the trash, then the amplitudes that we want to merge together and cancel amplitudes to 0 cannot be added because they are no longer the same state: the differing values of garbage differentiate the two states.

How?

Put our thing down, flip it, and reverse it.

For all the instructions except the one that writes the answer, we can reverse them in reverse order: we can in particular exploit the fact that all of our operations have their reverse as their inverse.

See that given deterministic reversible code with G gates, we can always make it garbage free with $\leq 2G$ gates.