Classical Models of Computation

- Circuit
- Reversible → Quantum Computation
- Randomized

Circuit - way of describing functions on bits

\[ \text{bit: } x \in \{0,1\}^n \quad \text{time} \rightarrow \quad \text{gate ex:} \]

\[ \begin{align*}
& x_1 \quad \land x_2 \\
& x_3 \quad \Rightarrow \quad \neg x_4 \\
\end{align*} \]

\[ x \quad \lor \quad x_5 \quad \lor \quad x_6 \]

\[ x \quad \rightarrow \quad \text{fanout} \]

Gate set is universal, can compute any \( f : \{0,1\}^n \rightarrow \{0,1\}^n \)

Generalized Boolean function

Circuits provide way of determining how difficult it is to compute a function.

Measures of circuit complexity:

- Gate count = # gates used
- Depth = # time steps (depends on parallelization)
- Width/Size = # input wires

Ex: Find circuit that implements function \( f : \{0,1\}^3 \rightarrow \{0,1\} ; \ f(x) = \begin{cases} 1 & x_1 = x_2 = x_3 = 0 \\ 0 & \text{otherwise} \end{cases} \)
Reversible

\[ x_1 \rightarrow y = 0 \quad \Rightarrow \quad x_1 x_2 = \{00, 10, 11\} \]

Not reversible

(only 1 output bit for 2 input bits – information must have been destroyed)

Toffoli Gate

\[ \begin{align*}
X_1 & \quad \text{Tof} \quad X_1 \\
X_2 & \quad \text{Tof} \quad X_2 \\
X_3 & \quad \text{Tof} \quad (X_1 \Lambda X_2) \\
\end{align*} \]

\[ \theta = \text{addition mod } 2 \]

Claim: Toffoli Gate is universal

ex: AND using Toffoli

\[ \begin{align*}
X_1 & \quad \text{Tof} \quad X_1 \\
X_2 & \quad \text{Tof} \quad X_2 \\
0 & \quad \text{Tof} \quad X_1 \Lambda X_2 \\
\end{align*} \]

*cost: extra input bit

All use Toffoli to create NOT, FANOUT, OR

With reversible gates, can do all computations reversibly: (at cost of extra input bits)

If can efficiently compute \( x \rightarrow f(x) \)
Can efficiently & reversibly compute \( (x, y, 0) \rightarrow (x, y, f(x), 0) \)

Q ops that preserve quantumness are reversible!
Randomized Computation

Deterministic: \( x_i = 0 \) or \( x_i = 1 \)

Randomized: \( p(0) = 0.75 \rightarrow (0.75) \)
\( p(1) = 0.25 \rightarrow (0.25) \)

Probability vectors:

1-bit:
\[
\begin{pmatrix}
(p(0)) \\
(p(1))
\end{pmatrix}
\]

2-bit:
\[
\begin{pmatrix}
(p(00)) \\
(p(01)) \\
(p(10)) \\
(p(11))
\end{pmatrix} = \begin{pmatrix}
0.1 \\
0.7 \\
0.2 \\
0
\end{pmatrix}
\]

\( n \)-bit:

- in \( \mathbb{R}^{2^n} \):
\[
\begin{pmatrix}
p(00...0) \\
p(00...1) \\
p(10...0) \\
p(10...1) \\
\vdots \\
p(11...1)
\end{pmatrix}
\]

- Vector elements \( p(s) \geq 0 \) \( \forall s \in \{0,1\}^n \)

- \( \sum_{s \in \{0,1\}^n} p(s) = 1 \)

Gate: \( 2^n \times 2^n \) left stochastic matrices: (cols sum to 1 & non-negative entries)

\[
\text{ex.} \begin{pmatrix}
0.5 & 0 \\
0.5 & 1
\end{pmatrix} \quad \text{ex.} \begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
\]

AL: Write the 2-bit gate that does the following:

- If 1st bit is 0 \( \rightarrow \) probability of these strings stays same
- If 1st bit is 1 \( \rightarrow \) probability of these strings are averaged

Connection: vectors rep. q states; matrices rep q ops; q is probabilistic

Independent & Correlated Variables

Variables \( x, y \) indep if \( p(x=x_i, y=y_j) = p(x=x_i) p(y=y_j) \) \( \forall x_i, y_j \) val

\( \text{Ind vars:} \)
\[
\begin{pmatrix}
p(00) \\
p(01) \\
p(10) \\
p(11)
\end{pmatrix} \otimes \begin{pmatrix}
p(00) \\
p(01) \\
p(10) \\
p(11)
\end{pmatrix} = \begin{pmatrix}
p(00) \\
p(01) \\
p(10) \\
p(11)
\end{pmatrix} \otimes \begin{pmatrix}
p(00) \\
p(01) \\
p(10) \\
p(11)
\end{pmatrix}
\]

\( \text{Ind ops:} \)
\[
\begin{pmatrix}
a_0 \ a_1 \\
a_2 \ a_3
\end{pmatrix} \otimes \begin{pmatrix}
b_0 \ b_1 \\
b_2 \ b_3
\end{pmatrix}
\]

Ex: \( \frac{1}{2} \otimes \frac{1}{2} = \frac{0}{1} \otimes \frac{1}{2} = \frac{0}{2} \)
\( \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix} \times \begin{pmatrix}
0 \\
\frac{1}{2} \\
\frac{1}{2} \\
0
\end{pmatrix} = \begin{pmatrix}
0 \\
\frac{1}{2} \\
\frac{1}{2} \\
0
\end{pmatrix}
\]

\( \text{CNOT:} \)
\( \text{Q Connection: tensor product for nd. states \& ops} \)
Other Models: cellular automata, Turing Machines, etc.

Does model matter?

Church-Turing Thesis: Any function that can be computed by a physically reasonable model of computation can be computed by a Turing Machine.

- def: A computation is efficient if number of operations is polynomially related to the number of input bits.

Strong Church-Turing Thesis: efficiently

Quantum computers likely violate!

Summarize:

- with power, discuss 3 types of computation important for Q.C.
- what does Q.C. have in common w/ these models?