A Brief Introduction to Quantum Computation

Tom Krüger

January 22, 2023

1 A Simple Computational Model

What are Qubits? That's usually the first question getting addressed in any introduction to quantum computing, for a good reason. If we want to construct a new computational model, we first need to define the most basic building block: a single *bit* of information. In classical computer science, the decision on how to define this smallest building block of information seems quite straight forward. We just take the most basic logical fact: either something is *true* or *false*, either 1 or 0. We have a name for an object holding this information: a **Bit**. Let's envision a computational model based on logical gates. Such a gate has one or more inputs and an output, with each either being *true* or *false*. Now consider a bit b and a gate $f : \{0,1\} \to \{0,1\}$. We have a *bit* of information b and can get another *bit* of information $b' \coloneqq f(b)$. In a final third step, we introduce a timescale, which means that now our *bit* of information is time dependent. It can have different values at different times. To make it easier, we choose a discrete timescale. Our Bit b has a distinct value on each point on the timescale. A value of a bit can only be changed in between time steps, by applying a logical gate to it:

$$\begin{array}{cccccccccc} \text{Bit} & b & \xrightarrow{f_1} & b & \xrightarrow{f_2} & \cdots & \to & b & \xrightarrow{f_k} & b \\ \text{time} & t_0 & \to & t_1 & \to & \cdots & \to & t_{k-1} & \to & t_k \end{array}$$

Of course, we need more than one bit of information, if we want to be able to perform meaningful computations. For this, we simply look at a list, vector or register of bits $\mathbf{b} \in \{0,1\}^n$ and modify our gates to be functions $f : \{0,1\}^n \to \{0,1\}^n$ mapping from bit vectors to bit vectors.

Let's recap: We've now designed a computational model with just three components.

- A notion of Information: bits and registers.
- A way of reasoning: logical gates.
- A dimension to do the reasoning in: the timescale

Notice how the system described above is fully deterministic. The state $\mathbf{b}_l$ of our system at time t_l recursively defined by:

$$\mathbf{b}_l = \begin{cases} f_l(\mathbf{b}_{l-1}) & \text{if} \quad l > 0 \\ \mathbf{b}_0 & \text{otherwise} \end{cases}$$

Or by the composition of all gate applications up to this point: $(f_l \circ f_{l-1} \circ \cdots \circ f_1)(\mathbf{b}_0)$. Actually, a composition of gates is also just another logical gate $F \coloneqq (f_l \circ f_{l-1} \circ \cdots \circ f_1) : \{0,1\}^n \to \{0,1\}^n$. If we are not interested in intermediate states, we can thus define our computation in the form of $\mathbf{b}_{\text{out}} \coloneqq F(\mathbf{b}_{\text{in}})$‘, with ‘$F : \{0,1\}^n \to \{0,1\}^n$.

2 A Bit of Randomness

2.1 Single Bits in Superposition

Many real world problems are believed to not be efficiently solvable on fully deterministic computers like the model described above (if $\mathbf{P} \neq \mathbf{NP}$). Fortunately, it turns out that if we allow for some randomness in our algorithms, we're often able to efficiently find solutions for such hard problems with sufficiently large success probabilities. Often times, the error probabilities can even be made exponentially small. For this reason, we also want to introduce randomness into our model. Algorithms or computational models harnessing the power of randomness are usually called *probabilistic*.

Again, we start with simple one bit systems. Later, we'll see how to expand the following methods to full bit vectors/registers. In the deterministic single bit model above, the state transition of a bit b in step t is defined by $f_t(b) \in \{0,1\}$. Now, the transition function (or gate) is simply allowed to flip an unfair coin and either output 0 or 1 for heads or tails respectively. Of course, the state of b prior to the transition should have an effect on the computation. That is, why we allow different (unfair) coins for either $b = 0$ or $b = 1$. To distinguish between deterministic and probabilistic transition functions, we will denote the latter by $\delta(b) \in \{0,1\}$. Or to reformulate this idea: Depending on the value of b, the output of $\delta(b)$ follows one of two Bernoulli trials. There are 4 possible transitions with probabilities p_{00}, p_{01}, p_{10} and p_{11}, where p_{ij} is the probability of b transitioning form i to j. Obviously, $\sum_j p_{ij} = 1$ always needs to be satisfied.

$$\begin{aligned} p_{00} &\coloneqq P(\delta(b) = 0 \mid b = 0) \\ p_{01} &\coloneqq P(\delta(b) = 1 \mid b = 0) \\ p_{10} &\coloneqq P(\delta(b) = 0 \mid b = 1) \\ p_{11} &\coloneqq P(\delta(b) = 1 \mid b = 1) \end{aligned}$$

Note that we regain our deterministic transition function f from δ, if we restrict the probabilities: $p_{00}, p_{10} \in \{0,1\}$. At this point, we can randomize our computation from above as follows:

$$\begin{array}{cccccccccc} \text{Bit} & b & \overset{\delta_1}{\to} & b & \overset{\delta_2}{\to} & \cdots & \to & b & \overset{\delta_k}{\to} & b \\ \text{time} & t_0 & \to & t_1 & \to & \cdots & \to & t_{k-1} & \to & t_k \end{array}$$

Let's have a look at the state of b after the first transition. In the deterministic model, we know with certainty that at this point in time, b will have the value $f_1(b)$. In a probabilistic model, we can not predict the value of b at time t_1 with 100% certainty. In the terminology of probability theory, a probabilistic state transition or even the whole computation would be an *experiment* and the value of bit b at time t would be described by a *random variable* X_t. Random variables are defined to take a value out of a set of predefined value options $\Omega = \{\omega_1, \ldots, \omega_n\}$ with certain probabilities $p_1, \ldots, p_n$ for each value. Only after we perform the experiment and *observe* its outcome, we get a specific value x_t of the random variable X_t. We say that x_t is a *random sample* or realization of X_t. If we don't want to or can't sample (perform) the experiment, we still could compute the *expected value* $E(X_t) = \sum_i p_i \omega_i$ (if Ω mathematically allows for such operations).

Let's return to our example: Just as in the deterministic case we would like to predict the state of b after the transition δ_t. For this we want to calculate the expected state of b at time t. Let p^t_{ij} be the transition probabilities of δ_t, furthermore $p^t_{b=x}$ denotes the probability of b being in state x at time t. Now we have:

$$E(\delta_t(b)) = p^t_{b=0} \cdot \mathbf{0} + p^t_{p=1} \cdot \mathbf{1} \tag{1}$$

$$p^t_{b=x} = \begin{cases} p^t_{0x} \cdot p^{t-1}_{b=0} + p^t_{1x} \cdot p^{t-1}_{b=1} & , t > 0 \\ 0, 1 & \text{otherwise} \end{cases} \tag{2}$$

It is important to note, that $\mathbf{0}$ and $\mathbf{1}$ in eq. (1) are not the scalar values of b. They define abstract objects denoting the fact that b is in state 0 or 1, so they are just arbitrary labels. For instance, same states could also be labeled $\{\mathbf{T}, \mathbf{F}\}$ or $\{\top, \bot\}$. But if $\mathbf{0}$ and $\mathbf{1}$ are some kind of abstract object and not scalar value, how can eq. (1) be evaluated? As of now it can't. Later we will define representations of these abstract stats, which are closed under addition and scalar multiplication, making eq. (1) also (a representation of) an abstract state.

From eq. (1), we will now derive a standard form of our random bit b. We don't view b as being either in state $\mathbf{0}$ OR $\mathbf{1}$ anymore. From now on, we think of b as being in $\mathbf{0}$ AND $\mathbf{1}$ simultaneously with certain probabilities $p_{b=0}$ and $p_{b=1}$, The one bit system b is in a *superposition* of two *basis states* $\mathbf{0}$ and $\mathbf{1}$:

$$b = p_0\mathbf{0} + p_1\mathbf{1} \quad , p_0 + p_1 = 1$$

Until now, we have not given an explicit definition of the transition function δ, apart from describing its effect. This is partly the case because we were lacking a formalism to describe uncertain states, so there was no direct way to describe the output of $\delta(b)$. The other big problem would have been the question of how to handle an uncertain input

state. Building on the superposition formalism $\delta(b)$ can be defined as a linear function:

$$\begin{aligned}\delta(b) &= \delta(p_0\mathbf{0} + p_1\mathbf{1}) \\ &= p_0\delta(\mathbf{0}) + p_1\delta(\mathbf{1}) \\ &= p_0(p_{00}\mathbf{0} + p_{01}\mathbf{1}) + p_1(p_{10}\mathbf{0} + p_{11}\mathbf{1}) \\ &= \underbrace{(p_0p_{00} + p_1p_{10})}_{=:p'_0}\mathbf{0} + \underbrace{(p_0p_{01} + p_1p_{11})}_{=:p'_1}\mathbf{1}\end{aligned}$$

A simple calculation verifies that

$$\begin{aligned}p'_0 + p'_1 &= (p_0p_{00} + p_1p_{10}) + (p_0p_{01} + p_1p_{11}) \\ &= p_0\underbrace{(p_{00} + p_{01})}_{=1} + p_1\underbrace{(p_{10} + p_{11})}_{=1} = p_0 + p_1 = 1\end{aligned}$$

and thus δ preserves valid superpositions, which finally makes predictions of the full computation through all steps possible. In line with the fully deterministic model the state of b at time t can be described by:

$$\begin{aligned}b_t &= \begin{cases}\delta_t(b_{t-1}) & \text{if} \quad t > 0 \\ b_0 \in \{\mathbf{0}, \mathbf{1}\} & \text{otherwise}\end{cases} \\ &= (\delta_t \circ \delta_{t-1} \circ \cdots \circ \delta_1)(b_0)\end{aligned} \tag{3}$$

2.2 Collapsing Superpositions

Extending this formalism to bit registers is actually fairly straight forward. Systems can be in superposition of arbitrary many basis states. But first, it is time to talk a bit more about the concept of superposition.

Definition 1. *Superposition of Probabilities If* $\mathbf{E} \coloneqq \{E_1, E_2, \ldots, E_n\}$ *is the set of all possible outcomes of an experiment, then a superposition of probable outcomes is defined by:*

$$E \coloneqq \sum_{i=1}^{n} p_iE_i \quad \textit{with } p_i = P(E_i) \textit{ and } \sum_{i=1}^{n} p_i = 1 \tag{4}$$

As mentioned above, a superposition can not immediately be evaluated. It rather should be seen as a mathematical object holding incomplete knowledge about a certain property of some (stochastic) process, described by a random distribution $(p_i)_{i=1}^n$. Too actually evaluate a superposition, the missing information needs to be filled in by some kind of extra process e.g. performing an experiment, measuring an observable. After this extra information is filled in the property under consideration is fully known and the superposition *collapses* to one of the actually realizable outcomes in $\mathbf{E}$. In this model a system can be in an uncertain state which only can be made concrete by some external influence like measuring an observable. This sounds quite abstract and especially the

fact that a measurement could alter the state of a real physical system seems quite counterintuitive, but we will later see that this principle is actually grounded in reality.

Let's consider the experiment of rolling a dice. Of course, for the observable *number of eyes* the expected outcomes are $\mathbf{E} = \{1, 2, \ldots, 6\}$. While the dice is still in the cup and in the state of being shaken number of eyes can not be reasonably determined, even if a transparent cup is being used. The dice is in a superposition $E = \sum_{i=1}^{6} \frac{1}{6}\mathbf{i}$ of showing all numbers of eyes 1 to 6 with uniform probability $\frac{1}{6}$. In order to determine the number of eyes thrown, the dice needs to rest on a solid base, such that one side is evidently showing up. So by *throwing the dice* we interfere with the system by stopping to shake the cup and placing the dice on a solid base (table). With the dice now laying on the table it is clearly showing only one number of eyes. The superposition collapsed!

Definition 2. *Collapse of Superposition*

2.3 Bit Registers in Superposition

3 Introducing: Linear Algebra

4 Making it Quantum