A Brief Introduction to Quantum Computation

Tom Krüger

February 9, 2023

Mention that transition functions / matrices need to be effectively computable by a polynomial algorithm. Otherwise, the hard computations can be outsourced to the preparation of the circuit. Consider a decision problem. The decision could be computed in advance for each input and the transition matrix just writes a designated bit: 1 for `ACCEPT` or 1 for REJECT

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:

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

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:

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

The states (outcomes) in $\mathbf{E}$ are called basis states (outcomes).

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)**.** A state in superposition of basis states $\mathbf{E} = \{E_1, E_2, \ldots, E_n\}$ can be evaluated by collapsing it on one of its basis states. This is done by a measuring operator

$$M_{\mathbf{E}}\left(\sum_{i=1}^{n} p_i E_i\right) \coloneqq E_i \quad \text{with probability } p_i \tag{5}$$

Remark 1. The basis states are not unique. To see this, consider the experiment of rolling a dice. If the observable is *the number of eyes* we have the basis states $\mathbf{E}_{\text{eye}} = \{\mathbf{i}\}_{i=1}^{6}$. On the other hand, if the measurement is only supposed to distinguish between *even or odd* numbers of eyes we have $\mathbf{E}_{\text{parity}} = \{\text{even}, \text{odd}\}$. The corresponding measuring operators are $M_{\mathbf{E}_{\text{eye}}}$ and $M_{\mathbf{E}_{\text{parity}}}$.

2.3 Bit Registers in Superposition

Extending the probabilistic one-bit model from section 2.1 to bit registers is almost trivial given the definitions from section 2.2. A n-bit register can be in $N = 2^n$ possible states, giving rise to a superposition of N basis states for probabilistic register states.

Definition 3. The state of a n-bit register in a probabilistic computation is defined by a superposition of all possible basis states $\mathbf{B} = \{\mathbf{0}, \mathbf{1}\}^n = \{\mathbf{0}, \mathbf{1}, \ldots, \mathbf{N-1}\}$.

$$\mathbf{b} \coloneqq \sum_{i=0}^{N-1} p_i \cdot \mathbf{i} \quad \text{with } P(\mathbf{b} = \mathbf{i}) = p_i \tag{6}$$

Remark 2. It should be noted that the number representation $\{\mathbf{i}\}_{i=0}^{N-1}$ is defined as the bit string $\{\mathbf{0}, \mathbf{1}\}^n$ in a base of 10. So it is just a shorter label for the state of a n-bit register and NOT a scalar value.

Similar to section 2.1 the transition function δ can be defined on its effect on basis states. For each transition the probabilities of transitioning from basis state $\mathbf{i}$ to basis state $\mathbf{j}$ must be defined. The mapping between states in superposition will then be defined linearly.

Definition 4. Let $\mathbf{b} = \sum_{i=0}^{N-1} p_i \mathbf{i}$ be a n-bit register as defined in definition 3 and let $p_{\mathbf{ij}}$ be the probability of transitioning form basis state $\mathbf{i}$ to basis state $\mathbf{j}$, then the transition function is defined by:

$$\delta(\mathbf{b}) \coloneqq \sum_{i=0}^{N-1} p_i \delta(\mathbf{i}) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_i p_{ij} \mathbf{j} \tag{7}$$

Theorem 1. *A transition function as defined by definition 4 maps superposition to valid superpositions.*

Proof. Let δ be a probabilistic transition function and let $\mathbf{b}$ a register state in superposition. By definition 4 we get $\delta(\mathbf{b}) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_i p_{ij} \mathbf{j}$ a simple reordering leads to

$$\delta(\mathbf{b}) = \sum_{j=0}^{N-1} \left(\sum_{i=0}^{N-1} p_i p_{ij} \right) \mathbf{j}$$

Obviously, $p_i p_{ij} = P(\mathbf{b} = \mathbf{i}) P(\delta(\mathbf{b}) = \mathbf{j} \mid \mathbf{b} = \mathbf{i})$. It follows directly from the law of total probability that $\sum_{j=0}^{N-1} \sum_{i=0}^{N-1} p_i p_{ij} = \sum_{j=0}^{N-1} P(\delta(\mathbf{b}) = \mathbf{j}) = 1$ □

A direct consequence of theorem 1 is that the space of probabilistic transition functions is also closed under composition. In accordance to eq. (3) the state of a register $\mathbf{b}$ in a probabilistic computation at time t can be described by:

$$\begin{aligned} \mathbf{b}_t &= \begin{cases} \delta_t(\mathbf{b}_{t-1}) & \text{if } t > 0 \\ \mathbf{b}_0 \in \{\mathbf{0}, \mathbf{1}\}^N & \text{otherwise} \end{cases} \\ &= (\delta_t \circ \delta_{t-1} \circ \cdots \circ \delta_1)(\mathbf{b}_0) \end{aligned} \tag{8}$$

3 Introducing: Linear Algebra

The definitions of section 2 fully describe a probabilistic computational model. Unfortunately, working with them can be quite cumbersome. This section will introduce an algebraic apparatus based on the definitions from above, with many helpful tools to describe computations and state evolutions. As some terminology and especially the linear properties of definition 4 already suggest the mathematical framework of choice will be linear algebra. Let's start by assessing the components of the model described above. We have:

- States (in superposition)
- State transitions
- Measurements (collapse of superposition)

As it turns out, all three components and their interactions can be expressed in the language of linear algebra. Readers familiar with that field of mathematics probably already noticed that δ is a linear function and the space of states in superposition looks a lot like a vector space.

3.1 The State Space

The defining property of a superposition is the probability distribution of its basis states. Given an enumeration all basis states the superposition is fully defined by the list of probabilities $(p_0, p_1, \ldots, p_{N-1})$.

Definition 5 (State Spaces of Probabilistic Computations)**.** Given a state basis $\mathbf{B}$ of a n-bit register, the state space of probabilistic computations on this register is defined as:

$$\mathbf{B}^n \coloneqq \left\{ \mathbf{b} = \sum_{i=0}^{N-1} p_i \mathbf{i} \;\middle|\; p_i \in \mathbb{R}_+ \,,\; \sum_{i=0}^{N-1} p_i = 1 \right\}$$

Definition 6. The coordinate map is a linear function $\Psi_{\mathbf{B}} : \mathbf{B}^n \to \mathbb{R}^N$ mapping the state space to $\mathbb{R}^N$:

$$\forall \mathbf{b} \in \mathbf{B}^n : \quad \Psi_{\mathbf{B}}(\mathbf{b}) = (p_0, p_1, \ldots, p_{N-1})^T = \sum_{i=1}^{N} p_i \mathbf{e}_i$$

Often $\Psi_{\mathbf{B}}(\mathbf{b})$ is called the coordinate vector of $\mathbf{b}$ with respect to the basis $\mathbf{B}$.

Lemma 1. *The state space of probabilistic computations is isomorphic to the surface of the unit sphere in the first quadrant of* $\mathbb{R}^N$*.*

$$\mathbf{B}^n \cong \left\{ \mathbf{v} \in \mathbb{R}^N_+ \;\middle|\; \|\mathbf{v}\| = 1 \right\}$$

Proof. For an arbitrary state $\mathbf{b}$ the coordinate vector $\Psi_{\mathbf{B}}(b) = \mathbf{v}$ is the direction of a ray in the first quadrant of $\mathbb{R}^N$ starting from the origin. Rescaling $\mathbf{v}$ results in the point where this ray intersects the unit sphere $\varphi(\mathbf{v}) = \frac{\mathbf{v}}{\|\mathbf{v}\|} = \mathbf{v}'$ which can be inverted by $\varphi^{-1}(\mathbf{v}') = \frac{\mathbf{v}'}{\|\mathbf{v}'\|_1} = \mathbf{v}$. Thus, $\varphi \circ \varphi^{-1} = \varphi^{-1} \circ \varphi = \mathrm{id}$ and

$$\Psi_{\mathbf{B}}(\mathbf{B}^n) = \{\mathbf{v} \in \mathbb{R}^N_+ \mid \|\mathbf{v}\|_1 = 1\} \cong \{\mathbf{v} \in \mathbb{R}^N_+ \mid \|\mathbf{v}\| = 1\}$$

□

3.2 Transition Matrices

It follows directly from eq. (1) that $\delta : \mathrm{span}(\mathbf{B}) \to \mathrm{span}(\mathbf{B})$ is a linear transformation on the space spanned by state basis $\mathbf{B}$ and theorem 1 even states that $\delta : \mathbf{B}^n \to \mathbf{B}^n$ and $\mathbf{B}^n$ is closed under δ. It is well known, that the space of all linear maps $\hom_{\mathbb{R}}(V, W)$ between two finite-dimensional real vector spaces V and W is isomorphic to $\mathbb{R}^{(\dim(W),\dim(V))}$. So, there must exist an isomorphism between transition functions δ and $\mathbb{R}^{(N,N)}$.

Theorem 2. *Let* $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N$ *be a n-bit state basis and* $\mathcal{B} = \{\mathbf{v}_j\}_{j=1}^N$ *a basis of* $\mathbb{R}^N$*, then there exists a matrix* $A = (a_{ij}) \in \mathbb{R}^{(N,N)}$ *such that*

- $\forall \mathbf{b}_i \in \mathbf{B} : \quad \delta(\mathbf{b}_i) = \sum_{j=1}^N a_{ji} \mathbf{v}_j$

- $\delta\left(\sum_{i=1}^{N} x_i \mathbf{b}_i\right) = \sum_{j=1}^{N} y_j \mathbf{v}_j \iff A(x_1, x_2, \ldots, x_N)^T = (y_1, y_2, \ldots, y_N)^T$

Remark 3. Usually it is custom to choose the standard basis $\{\mathbf{e}_i\}_{i=1}^N$ for $\mathbb{R}^N$, then theorem 2 describes how A can be used to describe how δ affects basis states in coordinate space. The j-th column vector $A\mathbf{e}_j = \mathbf{a}^j = (a_{1j}, a_{2j}, \ldots, a_{Nj})^T$ represents the probability distribution of $\delta(\mathbf{b_j})$. It follows that $a_{ij} = p_{ji}$, with p_{ji} being the probability of transitioning from $\mathbf{b}_j$ to $\mathbf{b}_i$. Consequently, $A = P^T$ with $P = (p_{ij})$.

3.3 Measurements

The final component that sill needs to be expressed in the framework of linear algebra are measurements. Let's go back and think about what measuring actually means in our case. The computational model described in section 2 provides a macroscopic view of randomized computations. The result of such a randomized computation will be a random state. Usually it is only of interest if a computation outputs a desired state given a specific input, which entails the correctness of said computation. For randomized computations, such an analysis requires the final random state distribution. Superposition states are exactly that. Asking "How likely is it to end up in state $\mathbf{k}$?" corresponds to measuring the quotient of $\mathbf{k}$ in the final superposition $\mathbf{b}$. In the framework of linear algebra this means calculating the scalar product $(\mathbf{k} \,.\, \mathbf{b})$.

Definition 7. Let $\mathbf{b} \coloneqq \sum_{i=1}^{N} p_i \mathbf{b}_i \in \mathbf{B}^n$ with basis states $\{\mathbf{b}_i\}_{i=1}^N$, then there exist N operators $\hat{M}_k : \mathbf{B}^n \to [0, 1]$

- in state space: $\hat{M}_k(\mathbf{b}) = (\mathbf{b}_k \,.\, \mathbf{b}) = p_k$
- in coordinate space: $M_k = \mathbf{b}_k^t \in \mathbb{R}^{(1,N)}, \quad M_k\mathbf{b} = \mathbf{b}_k^t\mathbf{b}$ With $\mathbf{b}^t$ being the transposed vector of $\mathbf{b} \in \mathbb{R}^N$

3.4 Tensor Product

motivate, combining (state) vector spaces, stochastic matrices closed under kronecker product

4 Making it Quantum

Section 3 formulates mathematical tools to algebraically describe an abstract model of probabilistic computations defined in section 2. This section takes a reverse approach. The tools developed in section 3 are based on stochastic matrices, which is an obvious choice to model probabilistic state transitions. Unfortunately this model has some shortcomings. This section first highlights these inconveniences and then fixes them. By doing so the model will gain in computational power, demonstrated by the implementation of Deutsch's algorithm. Finally, it will be shown that this extended model is indeed physically realizable.

4.1 Cleaning Up

The straight forward and rather simplistic choice of using probability coefficients in definitions 3 and 5 results in quite unwieldy state objects especially in the linear algebra representation. Of course, the probability mass of a complete sample space must always sum up to 1, demanding the normalization of state vectors by the $\|.\|_1$ norm. The state space $\mathbf{B}^n$ defined in this way is an affine combination of its basis vectors. For an 1-bit system this corresponds to the line segment from $\mathbf{0}$ to $\mathbf{1}$ (see fig. 1a). As lemma 1 already suggest, randomized computations could be viewed as rotating a ray around the origin. If computations essentially are rotations, then angles between state vectors seem somewhat important. Of course with $\mathbf{a}, \mathbf{b} \in \mathbf{B}^n$ it would be possible to calculate the angle between both sates by rescaling their dot product by their lengths $\mathbf{0}^t\mathbf{1}(|\mathbf{a}||\mathbf{b}|)^{-1}$. State vectors with unit length would greatly simplify angle calculations. Then, the dot product would suffice. Fortunately, lemma 1 states that $\mathbf{B}^n$ is isomorphic to a subset of the surface of the unit sphere. Therefore, it should also be possible to represent the state space as vectors with unit length. To distinguish between both representation we will write state vectors with coordinates on the unit sphere as $|b\rangle$. This notation is the standard notation of quantum states. For now, $\langle b|$ will be the transposed vector of $|b\rangle$ and $\langle b_1|b_2\rangle = \langle b_1|\ |b_2\rangle$ is the dot product of $|b_1\rangle$ and $|b_2\rangle$. By definition the length of $|b\rangle = \sum_{i=1}^{N} \alpha_i |b_i\rangle$ is 1. The linear coefficients α_i are not probabilities but so-called probability amplitudes and the Pythagorean theorem states that $1 = \sum_{i=1}^{N} \alpha_i^2$. This means squaring the amplitudes or taking the square root of probabilities maps between affine combinations of basis vectors and points on the unit sphere in the state space. As it turns out negative amplitudes must be allowed, thus this mapping is ambiguous and NOT an isomorphism. Each point on the unit sphere describes exactly one possible state of a probabilistic computation. This means moving on the 2^n-dimensional unit sphere can be viewed as a kind of computation on a state space $\mathcal{B}_{\mathbb{R}}^n$

Definition 8 (Real State Space)**.** Given a mapping $\pi : \mathcal{B} \to (0,1)^n$ of each configuration of a n-bit register to an orthonormal basis $\mathcal{B}$ of $\mathbb{R}^N$ with $N = 2^n$, points on the N-dimensional unit sphere form a computational state space:

$$\mathcal{B}_{\mathbb{R}}^n := \left\{ |b\rangle = \sum_{i=1}^{N} a_i \mathbf{b}_i \in \mathbb{R}^N \;\middle|\; \||b\rangle\| = 1,\ \mathbf{b}_i \in \mathcal{B} \right\}$$

with

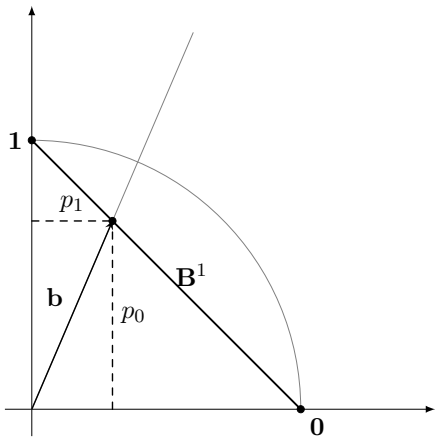
$$a_i^2 = P(\pi(\mathbf{b}_i))$$

4.2 Orthogonal Operators

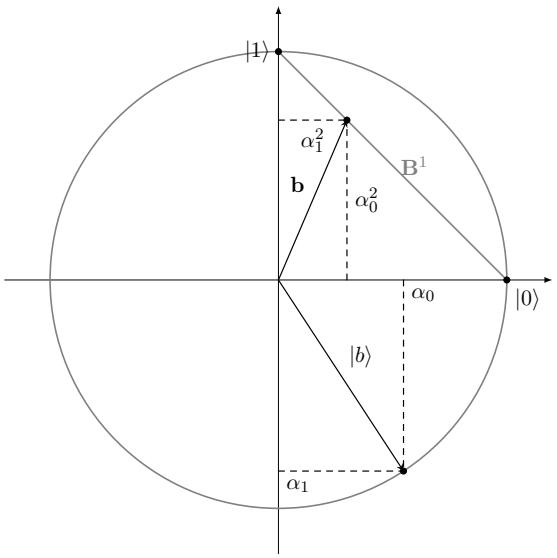
The set of operations mapping $\mathcal{B}_{\mathbb{R}}^n$ to $\mathcal{B}_{\mathbb{R}}^n$ is exactly the set of all rotations and rotoreflections, which together form the orthogonal group. Every element of the orthogonal group can be represented by an orthogonal matrix.

Definition 9. A matrix $A \in \mathbb{R}^n$ is orthogonal iff

$$A^{-1} = A^t \quad \Leftrightarrow \quad AA^t = A^tA = \mathbb{1}$$



(a) todo



(b) todo

Remark 4. It is important that orthogonal matrices from a group, because this means the composition of two orthogonal operators is orthogonal again. So, orthogonal computation can be composed or decomposed by or into other orthogonal computations. This is extremely useful for describing and developing algorithms in this model.

Definition 10 (Orthogonal Computation)**.** A computation on the state space $\mathcal{B}_{\mathbb{R}}^n$ is defined by an orthogonal matrix $A \in \mathbb{R}^N$ with $N = 2^n$.

What does it mean if a matrix is orthogonal? Let $A = (a_{ij}) = (\mathbf{a}_i, \ldots, \mathbf{a}_n) \in \mathbb{R}^{(n,n)}$ be orthogonal with. Then, it follows directly from $AA^t = (b_{ij}) = \mathbb{1}$ that $b_{ij} = \mathbf{a}_i^t \mathbf{a}_j = \delta_{ij}$. Hence, the columns (and rows) of A form an orthonormal basis of $\mathbb{R}^n$. It is also easy to check that A preserves the dot product making it angle and length preserving. Another direct consequence of $AA^t = \mathbb{1}$ is the reversibility of orthogonal computations.

4.3 Measurements

The measurement operators of section 3.3 obviously also need to be adjusted to the new state space and also suffered from some shortcomings. The dot product operators of definition 7 completely leave the state space when applied to a state vector. But the most important reason for why to redesign measurements are the newly used probably amplitudes. Just extracting an amplitude and squaring it would of course be a possible, but alien solution to the linear framework developed so far. This is because squaring is not linear. The desired goal is to design a linear operator nicely fitting in the framework at hand.

Definition 11 ((Real) Measurement Operators)**.** Let Ω be the set of possible outcomes of an experiment. Then the corresponding measurement is described by a set of linear operators $\{M_m\}_{m \in \Omega} \subset \mathbb{R}^{(N,N)}$ with:

- $P(m) = \langle b|M_m^t M_m|b\rangle$
- $\sum_m P(m) = \sum_m \langle b|M_m^t M_m|b\rangle = 1 \quad \Leftrightarrow \quad \sum_m M_m^t M_m = \mathbb{1}$

and $|b\rangle \in \mathcal{B}_{\mathbb{R}}^n$

4.3.1 (Real) Projective Measurements

One of the most important special cases of measurement operators are projective measurements. As the name already suggests, projective measurements are linear projections onto subspaces of $\mathcal{B}_{\mathbb{R}}^n$.

4.4 Interference - Computational Power

So far, moving computations on affine combinations to points on the unit sphere had merely subjective and rather esoteric reasons to *clean up* an abstract description of a computational model. In short: It's mathematically nicer to move around on the unit sphere. This section shows, that utilizing the power of probably amplitudes one actually gains computational power compared to the previous model.

4.4.1 Reversing a Coin Flip

In the classical probabilistic model a coin flip destroys any information stored in a bit, even in superposition states. It is easy to verify that $P_{1/2} = \frac{1}{2}\left(\begin{smallmatrix}1 & 1\\ 1 & 1\end{smallmatrix}\right)$ indeed implements a 1-bit coin flip and satisfies the conditions of theorem 2. Two consecutive coin flips are independent, which is illustrated by $P_{1/2}P_{1/2} = P_{1/2}$. The coin flip applied to an arbitrary superposition $\mathbf{b} = p_0\mathbf{b}_0 + p_1\mathbf{b}_1$ yields:

$$P_{1/2}\mathbf{b} = p_0\frac{1}{2}(\mathbf{b}_0 + \mathbf{b}_1) + p_1\frac{1}{2}(\mathbf{b}_0 + \mathbf{b}_1) = (p_0 + p_1)\frac{1}{2}(\mathbf{b}_0 + \mathbf{b}_1) = \frac{1}{2}(\mathbf{b}_0 + \mathbf{b}_1)$$

Given any input state $P_{1/2}$ reaches a fixed point after one iteration. With $P_{1/2}$ not being orthogonal a different operator is needed for the orthogonal model, which is the *Hadamard* operator.

Definition 12 (Hadamard Operator)**.**

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{9}$$

The Hadamard operator is basically the same as $P_{1/2}$ but the global factor is adjusted for probability amplitudes and the matrix columns are made orthogonal by introducing a negative phase in the second column. Exactly this negative phase will have some surprisingly useful effects. It is easy to see that

$$P(H|b\rangle = |0\rangle) = P(H|b\rangle = |1\rangle) = \langle b| H^\dagger M_{|0\rangle} H|b\rangle = \langle b| H^\dagger M_{|1\rangle} H|b\rangle = \frac{1}{2}$$

for $|b\rangle \in \{|0\rangle, |1\rangle\}$, $M_{|0\rangle} = |0\rangle\langle 0|$ and $M_{|1\rangle} = |1\rangle\langle 1|$. So, H does indeed implement a fair coin flip. But contrary to $P_{1/2}$ the Hadamard operator does not destroy the information stored in a superposition.

$$\begin{aligned} H(a_0 |0\rangle + a_1 |1\rangle) &= \frac{a_0}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{a_1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \frac{1}{\sqrt{2}}((a_0 + a_1)|0\rangle + (a_0 - a_1)|1\rangle) \end{aligned} \tag{10}$$

Actually, applying H a second time completely reverses the computation, this is the result of $HH = \mathbb{1}$ and might seem strange at first but is in fact a trivial consequence of orthogonal operators representing rotations and rotoreflections, which both are obviously reversible. Thus, the first H can not have destroyed any information. It is interesting to look into how something like that happens. After the second H application the state is:

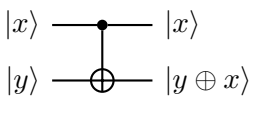
$$\begin{aligned} &\frac{1}{2}((a_0 + a_1)(|0\rangle + |1\rangle) + (a_0 - a_1)(|0\rangle - |1\rangle)) \\ &\qquad = \frac{1}{2}((a_0 + a_0 + a_1 - a_1)|0\rangle + (a_0 - a_0 + a_1 + a_1)|1\rangle) = a_0 |0\rangle + a_1 |1\rangle \end{aligned} \tag{11}$$

The key observation can already be made in eq. (10). Probability amplitudes can destructively interfere with each other. In eq. (10) this can be seen in the term $(a_0 - a_1)$ and in eq. (11) the amplitudes cancel each other out just perfectly to restore the original input state. It can't be mentioned enough: probability amplitudes are not probabilities. Destructive Interference is not possible with stochastic matrices from section 3 with all their entries being strictly positive. The next section shows how interference effects can be utilized effectively to outperform any probabilistic computation.

4.4.2 Deutsch's Algorithm

Given a function $f : \{0,1\} \to \{0,1\}$, the problem at hand is to determine whether $f(0) \stackrel{?}{=} f(1)$. Obviously, deterministic and even probabilistic computations need to evaluate f two times, once for each input, in order to answer this question. Surprisingly, orthogonal computations only need one call to f. But how is that possible?

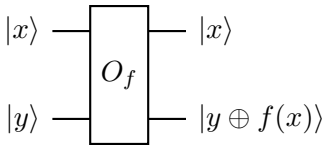
First, function evaluation needs to be addressed in the context of orthogonal computations. The requirement of orthogonality requires all computations to be reversible. But what if f is not injective e.g. $f(0) = f(1)$? Well, a simple trick solves this dilemma: Instead of evaluating f directly, f will be wrapped by an orthogonal operator O_f. Note that the XOR operator:



$$\hat{\text{XOR}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

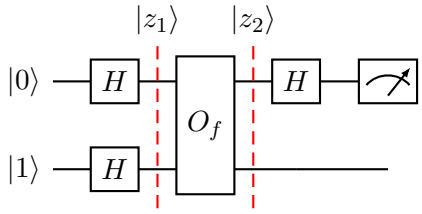
is reversible as $\text{XOR}\,|x, y \oplus x\rangle = |x, y \oplus x \oplus x\rangle = |x, y\rangle$. From the matrix form it apparent that O_f even is orthogonal. Similarly, it follows that $O_f\,|x, y\rangle \coloneqq |x, y \oplus f(x)\rangle$ is

reversible. A closer look reveals that O_f just permutes the basis states depending on f and again it is easily verifiable that $O_f O_f = \mathbb{1}$, making O_f orthogonal.



$$O_f = \begin{pmatrix} 1-f(0) & f(0) & 0 & 0 \\ f(0) & 1-f(0) & 0 & 0 \\ 0 & 0 & 1-f(1) & f(1) \\ 0 & 0 & f(1) & 1-f(1) \end{pmatrix}$$

So it is indeed possible to evaluate a function $f : \{0,1\} \to \{0,1\}$ in the orthogonal computational model. If the second qubit $|y\rangle$ is initialized as $|0\rangle$, measuring that qubit in the computational basis after the application of O_f returns the value of $f(x)$. The trick that makes it possible to answer $f(0) \stackrel{?}{=} f(1)$ with one call to O_f is to initialize both qubits in a perfect superposition of $|0\rangle$ and $|1\rangle$. Then, both values of $f(0)$ and $f(1)$ will interfere $|y\rangle$.



So, now it is time to examen this circuit in detail to understand it's inner workings. After the first Hadamard gates the system is in state

$$\begin{aligned} |z_1\rangle &= \frac{1}{2}((|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle)) \\ &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \end{aligned}$$

applying O_f changes the system state to

$$\begin{aligned} O_f |z_1\rangle &= \tfrac{1}{2} \begin{pmatrix} (1-f(0)-f(0))\,|00\rangle + \\ (f(0)-1+f(0))\,|01\rangle + \\ (1-f(1)-f(1))\,|10\rangle + \\ (f(1)-1+f(1))\,|11\rangle \end{pmatrix} = \tfrac{1}{2} \begin{pmatrix} (-1)^{f(0)}\,|00\rangle - \\ (-1)^{f(0)}\,|10\rangle + \\ (-1)^{f(1)}\,|01\rangle - \\ (-1)^{f(1)}\,|11\rangle \end{pmatrix} \\ &= \tfrac{1}{2}\Big(\Big((-1)^{f(0)}\,|0\rangle + (-1)^{f(1)}\,|1\rangle\Big) \otimes (|0\rangle - |1\rangle)\Big) \\ &= |z_2\rangle \end{aligned}$$

That is a lot to digest, but what happened is: Both qubits interfered with each other and although f got applied to the second qubit, the effect of this operation moved through the amplitudes to the first qubit. From now on the second qubit is not important anymore, so only the first one will be considered from now on. The first bit viewed as an isolated

system is in state $1/\sqrt{2}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle)$. Applying a Hadamard gate results in

$$
\begin{aligned}
&H\frac{1}{\sqrt{2}}\Big((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big) = \\
&\frac{1}{2}\Big((-1)^{f(0)}(|0\rangle + |1\rangle) + (-1)^{f(1)}(|0\rangle - |1\rangle)\Big) = \\
&\frac{1}{2}\Big((-1)^{f(0)}|0\rangle + (-1)^{f(0)}|1\rangle + (-1)^{f(1)}|0\rangle - (-1)^{f(1)}|1\rangle\Big) = \\
&\frac{1}{2}\Big(\Big((-1)^{f(0)} + (-1)^{f(1)}\Big)|0\rangle + \Big((-1)^{f(0)} - (-1)^{f(1)}\Big)|1\rangle\Big) = \\
&\frac{1}{2}(-1)^{f(0)}\Big(\Big(1 + (-1)^{f(0)\oplus f(1)}\Big)|0\rangle + \Big(1 - (-1)^{f(0)\oplus f(1)}\Big)|1\rangle\Big)
\end{aligned}
$$

Now measuring the first bit returns $|0\rangle$ iff $f(0) = f(1)$ and otherwise $|1\rangle$.

Apparently interference of amplitudes is truly a remarkable feature. There is a key difference between probabilistic and orthogonal computations. In the probabilistic model the concept of computation superposition merely models the lack of knowledge about the true state of the system. If a system is in a probabilistic superposition of two states the true state is just unknown but the system is only in one of the two states at any time. The interference effects in orthogonal computations paint a different picture. In order to interfere with each other the two sates in a orthogonal superposition must somehow be present at the same time. The system has to be in both states, resulting a kind of inherent parallelism built into the model! In Deutsch's algorithm, the application of O_f to a superposition does in fact not equal one call to f but evaluating f in parallel for both inputs.

rename qubits as the term is not yet defined

4.5 Entering the Real World: Quantum Computing

The best theoretical model is of no value if it relies on some kind of magic that not be realized in the real world. Luckily if the real probability amplitudes of the orthogonal model are replaced by complex numbers, one ends up with a computational model which can perfectly described by quantum mechanics, thus making it a real world physical system, fittingly called *quantum computing*. Of course, real probability amplitudes are a subset of complex probably amplitudes and likewise orthogonal computations are a subset of quantum computations. Only minor adjustments need to be made to the mathematical framework to handle complex amplitudes. The fundamental unit of information is a *qubit* this special terminology is intended to highlight the special properties of quantum superpositions. Also, oftentimes greek leters are used in the quantum case.

Definition 13 (Qubit)**.** A qubit is a quantum superposition with unit length of two orthonormal basis vectors $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{12}$$

With the transposed complex conjugate of $|\psi\rangle$ denoted as:

$$(|\psi\rangle^*)^t = |\psi\rangle^\dagger = \langle\psi| = \langle 0|\,\alpha^* + \langle 1|\,\beta^* \tag{13}$$

Furthermore, every qubit satisfies the normalization criterion:

$$\langle\psi|\psi\rangle = \alpha^*\alpha + \beta^*\beta = |\alpha|^2 + |\beta|^2 = 1 \tag{14}$$

Remark 5. In accordance with the standard in quantum computing literature the computational basis states $|0\rangle$ and $|1\rangle$ where chosen for in definition 13. However, similarly to orthogonal matrices, unitary matrices (their complex extension) also map any complex orthonormal basis to another complex orthonormal basis of the same dimension. Also, every bijective mapping of two complex orthonormal bases is a unitary operation. So, any qubit can be transformed to another basis by simply applying a unitary transformation to it, making the basis choice of a qubit irrelevant.

Definition 14 (Computational Basis)**.** The standard bais of quantum computing is the computational basis. The basis 2^n values of a n-bit register are mapped to an orthonormal basis $\mathcal{B} = \{|i\rangle\}_{i=0}^{2^n-1}$, with i being the decimal value of the n-bit register. In coordinate space $|i\rangle$ is represented by $e_i \in \mathbb{C}^{2^n}$:

$$|i\rangle \cong e_i = (\underbrace{0,\ldots,0}_{i-1},1,0,\ldots,0)^t \tag{15}$$

Definition 15 (Quantum State Space)**.** Spanning a complex vector space over an arbitrary set of 2^n many orthonormal base vectors we get the state space of a n-qubit system. The standard base of choice is the computational basis (definition 14), giving the standard n-qubit state space $\mathcal{B}$

$$\mathcal{B}^n = \left\{ |\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle \;\middle|\; \alpha_i \in \mathbb{C},\ \langle\psi|\psi\rangle = 1 \right\}$$

Definition 16 (Unitary Operator)**.** Unitary operators and matrices are the complex extensions to orthogonal operators and matrices. An operator Φ is unitary if its matrix representation U_Φ is a unitary matrix, which is the case iff

$$U_\Phi^{-1} = U_\Phi^\dagger \quad \Leftrightarrow \quad U_\Phi U_\Phi^\dagger = U_\Phi^\dagger U_\Phi = \mathbb{1}$$

with $U_\Phi^\dagger$ being the transposed complex conjugate of U_Φ, defined as

$$\forall n \in \mathbb{N},\ A = (a_{ij}) \in \mathbb{C}^n\ :\quad A^\dagger = (a_{ji}^*)$$

Definition 17 (Unitary Computation)**.** In accordance with definition 10, a computation on the state space $\mathcal{B}^n$ is defined by a unitary matrix $U \in \mathbb{C}^{2^n}$.

Definition 18 (Measurement Operators)**.** The definition of measurements on quantum state spaces follow the definition of measurements on orthogonal state spaces (definition 11), but with states $|\psi\rangle \in \mathcal{B}^n$ and complex operators $\{M_m\}_{m\in\Omega} \subset \mathbb{C}^{(N,N)}$.

This wraps up the framework of quantum computing. Fortunately, definitions 15, 16 and 18 correspond to the fundermental postulates of quantum mechanics, meaning that quantum computing with all its seemingly strange properties is in fact a physically realizable computational model!