1 Recall

In classical case the state probabilities were given by a vector composed of state probabilities in corresponding rows and the values were summed up to 1, e.g. \[
\begin{pmatrix}
\frac{1}{2} \\
0 \\
\frac{1}{2}
\end{pmatrix}.
\]
In the quantum case the values in the vector are interchanged with the complex probability amplitudes where the squares of the values give the probabilities as in the case of \[
\begin{pmatrix}
\frac{1}{\sqrt{2}} \\
0 \\
-\frac{1}{\sqrt{2}}
\end{pmatrix}.
\]

2 Real Time Quantum Finite Automata (QFA)

Real time QFA is a 5-tuple \(\{Q, \Sigma, q_1, \{E_\sigma\}_{\sigma \in \Sigma}, F\}\) where

\(Q :=\) set of possible states
\(\Sigma :=\) input alphabet
\(q_1 :=\) initial state
\(\{E_\sigma\}_{\sigma \in \Sigma} :=\) defining matrices
\(F :=\) set of accept states.

Similar to the case of Deterministic Finite Automata (DFA) and Probabilistic Finite Automata (PFA), we have a different transition matrix \(E_\sigma\) for every different symbol \(\sigma\) in the alphabet. Each \(E_\sigma\) will be a collection of \(m \times |Q|\) dimensional matrices which is called operation elements for some \(m > 0\); \(m\) being the maximum number of probabilistic branching the machine can do. In the quantum case we will have \(m\) many square matrices, not a single matrix (as in the case of DFA and PFA).
The QFA can be thought to be composed of two systems with two distinct probability types. There is the 'classical probability' just like being one of the branches with corresponding Heads/Tails throw (or for more than two branches and corresponding probabilities) and quantum intrinsic probability of states which are characterized by the complex probability amplitudes given by the components of the state vector.

$$\begin{bmatrix} 1/\sqrt{3} \\ 0 \\ \sqrt{2}/\sqrt{3} \end{bmatrix}$$ with prob = 0.5(Heads) \hspace{1cm} \begin{bmatrix} 0 \\ i/\sqrt{2} \\ -i/\sqrt{2} \end{bmatrix}$$ with prob = 0.5(Tails)

The vectors can be though as the 'quantum not-knowing' and the probabilities assigned to these branchings (Heads/Tails) as 'classical not-knowing'. The classical and quantum branchings need to be considered seperately in QFA.

3 Simple QFA Example

Let's say we have a state set \(\{q_1, q_2, q_3\}\), the alphabet consisting of only one letter \(a (\Sigma = \{a\})\) and an accept state of \(F = \{q_2\}\). The number of classical branching will be \(m = 2\).

Since the number of branchings is 2 (let’s call them Heads/Tails), we will have two transition matrices \(E_{a,1}\) and \(E_{a,2}\); they are defined as follows:

\[
E_{a,1} = \begin{bmatrix} 1/\sqrt{2} & 0 & 0 \\ 1/\sqrt{2} & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 \end{bmatrix} \hspace{1cm} E_{a,2} = \begin{bmatrix} 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\] (1)

Let’s say we started with \(q_1\), i.e the initial state is \(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\). The letter ’a’ is read, the first case (‘Heads’) the state evolves as

\[
\begin{bmatrix} 1/\sqrt{2} & 0 & 0 \\ 1/\sqrt{2} & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}\] (2)

The second case (‘Tails’) evolves as

\[
\begin{bmatrix} 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\] (3)
Then we read the second letter ‘a’ and similarly there will be two branchings (Heads/Tails): The first case (‘Heads’) the state evolves as

\[
\begin{bmatrix}
\frac{1}{\sqrt{2}} & 0 & 0 \\
\frac{1}{\sqrt{2}} & 0 & 0 \\
0 & \frac{1}{\sqrt{2}} & 0 \\
\end{bmatrix}
\begin{bmatrix}
\frac{1}{\sqrt{2}} \\
0 \\
0 \\
\end{bmatrix}
= 
\begin{bmatrix}
\frac{1}{2} \\
\frac{1}{2} \\
\frac{1}{2} \\
\end{bmatrix}
\tag{4}
\]

The second case (‘Tails’) evolves as

\[
\begin{bmatrix}
0 & \frac{1}{\sqrt{2}} & 0 \\
0 & 0 & \frac{1}{\sqrt{2}} \\
0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
\end{bmatrix}
= 
\begin{bmatrix}
\frac{1}{2} \\
0 \\
0 \\
\end{bmatrix}
\tag{5}
\]

So, for an input string 'aa', the first ‘a’ gives two brancings (2) and (3), namely

\[
\begin{bmatrix}
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
0 \\
\end{bmatrix}
\text{ and }
\begin{bmatrix}
0 \\
0 \\
0 \\
\end{bmatrix}
\].

When we inspect the first branching and calculate the probabilities, we see that the probability of \( p(q_1) = (1/\sqrt{2})^2 = 1/2 \), \( p(q_2) = (1/\sqrt{2})^2 = 1/2 \) and \( p(q_3) = 0 \). The second branching gives the all-zero vector, thus we can say that there is no valid branching to that. Thus we can safely say that it transitions to the first branch with probability 1.

The second letter ‘a’ gives two branches (4) and (5), namely

\[
\begin{bmatrix}
\frac{1}{2} \\
0 \\
0 \\
\end{bmatrix}
\text{ and }
\begin{bmatrix}
\frac{1}{2} \\
\frac{1}{2} \\
0 \\
\end{bmatrix}
\].

The probabilities for each states are \( p(q_1) = p(q_2) = p(q_3) = 1/4 \) and the total probability is 3/4. The other branch gives \( p(q_1) = 1/4 \) and 0 for other states. The total probabilities 3/4 and 1/4 are considered as ‘classical probabilities’ whereas the probabilities computed for each state in each branchings from the vector components are considered as ‘quantum probabilities’. But notice that at the end of reading the second letter, the states we get (4) and (5) are not normalized. In order to consider them as probability distributions we normalize each of them by dividing them by their \( l_2 \) norm (i.e square root of sum of the components squared) and we get

\[
\begin{bmatrix}
\frac{1}{2} \\
\frac{1}{2} \\
\frac{1}{2} \\
\end{bmatrix}
\text{ divide by } \sqrt{\frac{3}{4}} 
\rightarrow 
\begin{bmatrix}
\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} \\
\end{bmatrix}
\text{ and }
\begin{bmatrix}
\frac{1}{2} \\
0 \\
0 \\
\end{bmatrix}
\text{ divide by } \sqrt{\frac{1}{4}} 
\rightarrow 
\begin{bmatrix}
\frac{1}{4} \\
\frac{1}{4} \\
0 \\
\end{bmatrix}
\tag{6}
\]

From these probabilities, we can say that the machine will say YES (i.e in state \( q_2 \) with probability \( \frac{3}{4} \cdot (\frac{1}{\sqrt{3}})^2 = \frac{1}{4} \). The remaining states’ probabilities are \( p(q_1) = 1/2 \) and \( p(q_3) = 1/4 \).

If we want to represent the transition of the system with one matrix, we need to put \( E_{a,1} \) and \( E_{a,2} \) on top of each other and create a bigger transition matrix:
When we look to the matrix (7), we see now that the squares of the numbers in each column add up to 1 and the columns are orthogonal to each other.

4 Density Matrix

There is a more convenient way to represent the transitions of QFA which is called the density matrix representation. First we need to represent the bra-ket notation which is commonly used in quantum physics literature. So a ket-vector is represented as a column vector and written as:

\[ |\psi\rangle \rightarrow \begin{bmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix} \]

and corresponding bra-vector is the row vector:

\[ \langle \psi | = (1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}) \]

They are related by \( \langle \psi | = (|\psi\rangle)^* \) where * denotes complex conjugation.

The states of the machine is represented by a ket-vector and its complex conjugate-transpose is a bra-vector.

The density matrix representation \( \rho \) of a set of probabilities and corresponding quantum states \( \{(p_i, |\psi_i\rangle) : i \leq i \leq n\} \) is given by:

\[ \rho = \sum_i p_i |\psi_i\rangle \langle \psi_i | \tag{8} \]

There is a density matrix for each step, reading a letter from the input string. For instance the density matrix for the step after reading ‘a’ can be calculated using (8):

\[
\begin{bmatrix}
3/4 & 1/\sqrt{3} & 1/\sqrt{3} \\
1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\
1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3}
\end{bmatrix}
\begin{bmatrix}
1/\sqrt{3} \\
1/\sqrt{3} \\
1/\sqrt{3}
\end{bmatrix}
\begin{bmatrix}
1/4 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
1/2 & 1/4 & 1/4 \\
1/4 & 1/4 & 1/4 \\
1/4 & 1/4 & 1/4
\end{bmatrix}
\]

\[
\begin{bmatrix}
1/2 & 1/4 & 1/4 \\
1/4 & 1/4 & 1/4 \\
1/4 & 1/4 & 1/4
\end{bmatrix}
\]
The numbers in the diagonal (bold-faced) of the density matrix are the corresponding probability values for each state ($i^{th}$ diagonal corresponding to probability of state $q_i$).

In order to calculate the probabilities for each step, we need to calculate the density matrix for each iteration. Here is how to calculate the density matrix at the next step $\rho'$, if you are presently at a state with density matrix $\rho$ and read a symbol 'a':

$$\rho' = \sum_i E_{a,i} \rho E_{a,i}^\dagger$$

(9)

So starting with the initial state at $q_1$, i.e.

\[
\begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
\]

the initial density matrix becomes

\[
\rho = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

Calculating the next density matrix by using (9):

\[
\begin{bmatrix}
1\sqrt{2} & 0 & 0 \\
1\sqrt{2} & 0 & 0 \\
0 & 1\sqrt{2} & 0
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
1\sqrt{2} & 1\sqrt{2} & 0 \\
0 & 0 & 1\sqrt{2} \\
0 & 0 & 0
\end{bmatrix}
+ \begin{bmatrix}
0 & 1\sqrt{2} & 0 \\
0 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
= \begin{bmatrix}
1/2 & 1/2 & 0 \\
1/2 & 1/4 & 1/4 \\
1/4 & 1/4 & 1/4
\end{bmatrix}
\]

(11)

The probabilities after processing single 'a' is the values at the diagonal of the resulting matrix, namely $p(q_1) = p(q_2) = 1/2$ and $p(q_3) = 0$. Then to calculate the next step, we use the result in (11) and put it to the equation (9) again and we find:

$$\rho_{\text{final}} = \begin{bmatrix}
1/2 & 1/4 & 1/4 \\
1/4 & 1/4 & 1/4 \\
1/4 & 1/4 & 1/4
\end{bmatrix}$$

The diagonal entries again corresponds to the probabilities of each state, namely $p(q_1) = 1/2, p(q_2) = p(q_3) = 1/4$, the same result that we have obtained by the way of computing probabilities from the branchings in the beginning.
Question: Are we sure that can these machines simulate classical machines (DFA or PFA) Can we write QFA corresponding to DFA/PFA?

Remember PFA has stochastic transition matrices, where the column sums are all 1s. For instances for a 3-state machine with the initial state probabilities are given by the vector \( \begin{bmatrix} x^2 \\ y^2 \\ z^2 \end{bmatrix} \) where \( x \) is the corresponding probability amplitudes. The states after the machine reads an 'a' becomes let’s say:

\[
\rho = \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0 & 0.6 \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ z^2 \end{bmatrix} = \begin{bmatrix} 0.1x^2 + 0.7y^2 + 0.2z^2 \\ 0.5x^2 + 0.3y^2 + 0.2z^2 \\ 0.4x^2 + 0.6z^2 \end{bmatrix}
\] (12)

We want to construct a QFA which functions as the above PFA. We again need the same number of states, so the same size matrices for transition operation, so they will be \( 3 \times 3 \). If we construct them as below and act on the initial transition amplitude vector \( \begin{bmatrix} x \\ y \\ z \end{bmatrix} \) we get,

\[
E_{a,1} = \begin{bmatrix} \sqrt{0.1} & 0 & 0 \\ \sqrt{0.5} & 0 & 0 \\ \sqrt{0.4} & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \sqrt{0.1} & 0 & 0 \\ \sqrt{0.5} & 0 & 0 \\ \sqrt{0.4} & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sqrt{0.1}x \\ \sqrt{0.5}y \\ \sqrt{0.4}z \end{bmatrix}
\]

\[
E_{a,2} = \begin{bmatrix} \sqrt{0.7} & 0 & 0 \\ \sqrt{0.3} & 0 & 0 \\ \sqrt{0} & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & \sqrt{0.7} & 0 \\ 0 & \sqrt{0.3} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sqrt{0.7}x \\ \sqrt{0.3}y \\ 0 \end{bmatrix}
\]

\[
E_{a,3} = \begin{bmatrix} 0 & 0 & \sqrt{0.2} \\ 0 & 0 & \sqrt{0.2} \\ 0 & 0 & \sqrt{0.6} \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & \sqrt{0.2} \\ 0 & 0 & \sqrt{0.2} \\ 0 & 0 & \sqrt{0.6} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sqrt{0.2}x \\ \sqrt{0.2}y \\ \sqrt{0.6}z \end{bmatrix}
\] (13)

From these three results, by looking at the end probability amplitudes, we can calculate the probability of being at state \( q_1 \) as

\[
p(q_1) = 0.1x^2 + 0.7y^2 + 0.2z^2
\]

and vice versa, exactly the same results we get from the PFA (eq. (12)). Thus we conclude that we can simulate a PFA with a QFA with the same number of states. Since DFA is a special case of a PFA, QFA can also simulate DFA. Thus we can say that QFA can recognize all the regular languages which DFA and PFA recognizes.
The important question is, we have shown that the QFA is at least can generate PFA, but can it do more than PFA? Can it recognize also non-regular languages? That is the question of ‘quantum supremacy’ and it will be discussed in the next lectures.