Bachelor Thesis

Integrating Quantum Computing concepts in Petri Nets

Georgios Papavarnavas

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2021

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Integrating Quantum computing concepts in Petri nets

Georgios Papavarnavas

Supervisor

Dr. Anna Philippou

Thesis submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Computer Science at the University of Cyprus

May 2021

Acknowledgements

The development and completion of this thesis would not have been possible without the continuous support of Dr. Anna Philippou, to whom I would like to express my utmost gratitude and appreciation. Her guidance was crucial and extremely vital for the correct development of functional and useful work on the actively researched topic of Quantum Computation. I am hopeful that I have contributed sufficiently under her supervision in her research, and that this thesis will be used as a foundation for new breakthroughs in the field.

It would be an omission not to thank the PhD student Eleftheria Kouppari for her help covering many gray areas, as well as provision of clarifications and useful material. She was a great person to work with.

Additionally, I would like to thank all my colleagues, and especially my closer friends. They provided a lot of mental support and relief during times of stress and tension, that the workload and the pandemic caused.

Last, but not least, I want to express my deepest gratitude to all my family members, for their love, mental and physical support. My parents, Chrysostomos and Panagiota, my sister Eleni, my brothers Varnavas and Antonis, my brother-in-law Fivos and our pet Roza. Without them, this thesis would have been impossible to complete.

This last “thank you” goes to all the people that entered my life during my work on my thesis. Their impact, small or big helped me grow as a person, and work further for this thesis, and myself. This includes you, the reader, I hope you learn something from my work, and you are more than welcome to use the contained knowledge to build great things!

Abstract

This Bachelor thesis aims at the creation of a framework that is based on Petri nets and captures quantum computing elements. The proposed formalism will be used to represent distributed quantum algorithms.

Petri nets are a mathematical modeling framework that supports the analysis and representation of distributed and concurrent systems. Petri nets can be used to represent the flow and transformation of data throughout the execution of a system. An extension of Petri nets includes the notion of reversibility. This describes the ability of the system to be found in a previously executed state, in a number of ways. Traditionally, they are used to represent natural and biological processes. Reversibility is also useful in a variety of computing applications.

Quantum computing is a new and highly researched area of computer science and physics. It revolutionizes computation as it is conventionally defined, with the aid of Quantum mechanics. The basic difference of a quantum computer from a classical computer, is that it operates on qubits, which can encode the values of traditional bits (0 and 1), but all the values in the complex space between them as well. Quantum computers can also harness several phenomena that are derived from quantum mechanics, such as Entanglement. This advantage of quantum computers enables us to execute a new class of algorithms that are more efficient and powerful.

The creation of a model that combines the aforementioned fields of quantum computation, Petri nets and reversibility is beneficial, as nothing like it exists and denotes a gap that needs to be filled. First, the ability of graphically representing reversible, distributed/parallel quantum systems will be realized. Additionally, more features that are included with Petri nets will be inherited by this model.

This thesis consists of an attempt of filling that gap in the field of quantum theoretical computing. The models of Quantum Petri nets and their extension, Reversible Quantum Petri nets are formally defined. Additionally, some practical examples with highly used quantum algorithms are presented and encoded in the mentioned models.

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

Quantum computation revolves around the harnessing of the quantum phenomena of superposition and entanglement to perform computation. The area of quantum computation is a pole of attraction for researchers as it is believed that quantum computers (machines that can perform quantum computation) can solve problems that are very time consuming to be solved on a classical computer and are regarded as insolvable, this is defined as quantum supremacy. The most famous example of the topic of quantum supremacy is Shor's algorithm [31] for factoring integers, which is exponentially faster than the most efficient classical algorithm, this is a threat for the RSA encryption scheme, as it is built around the complexity of integer factoring by conventional computing. This, however, is an opportunity for creating new, more secure encryption methods in the post-quantum era. Still, there is a limit to the potential of quantum computers, which is a topic that is being researched [23].

Quantum computers and quantum information theory are relatively new topics are actively being researched. Many models, that reflect their conventional computing counterparts have been created, such as Quantum Finite Automata [32] and Quantum Turing Machines [33]. Their behavior and appearance are similar to their classical counterparts but with an expanded logic, so it can cover Quantum Computing aspects, such as the notions of qubits, superposition, entanglement, and destructive measurements. This difference at the core of the information representation has many implications in the

development of such models. However, theoretical models of quantum computation are of great significance as an important motivation is the ability to create quantum computing models and verify the correctness of quantum algorithms and systems. As such, they form the very foundation of physical quantum computers.

Petri nets [1] are a theoretical model of parallel and distributed systems. They are very flexible and scalable, thus the existence of many variations of Petri nets is justified, as they attempt to capture different aspects of certain systems. They are a great tool for visualizing the operation and purpose of a distributed system, aid in the analysis of its behavior and expansion of its logic.

Reversibility, as a concept, tries to capture the operation of rolling back to a previous state of a system. Theoretically, this form of computation has the same cost of reverting to a previous state, as moving forward to the next state in forward execution. Reversibility is an important concept as it appears in many Natural Sciences and is a desirable property for a variety of computing systems. It is important to mention that quantum computation is naturally reversible, to an extent.

When it comes to the topic of reversibility in Petri nets, Reversible Petri nets [2] have been developed as an attempt to capture various ways of reversible computation in Petri nets. Reversible Petri nets were created to aid in the encoding of some biological models etc.

The need for a model that is capable of containing the distributed aspects of Petri nets and Quantum Computation is high. As of today, no attempt has been made to capture quantum computation within Petri nets. Such a model would provide a much-needed level of flexibility in Quantum Computation models, as the concept of parallel and distributed quantum computation is still a work in progress and would be benefitted from a graphical model for the analysis, error detection and fault management of such systems. Additionally, a tool for typical analysis of quantum systems and critical processes is important to be created as it will aid in proving their correctness.

1.2 Work Purpose

The main goal of this thesis is the creation of a complete and functional model that is able to capture the main concepts of quantum computation within a Petri net framework.

Additionally, the proposed framework should be able to support the modeling of implementations of quantum algorithms and biological systems.

1.3 Work Methodology

The first step for accomplishing our goal, was conducting an in-depth theoretical research about Quantum Mechanics, Petri nets, Reversible Petri nets, Linear algebra, Quantum algorithms and Quantum Computing concepts. This study was crucial and continuous throughout all the steps of this thesis, as the knowledge of the concepts required to understand and work on the topics mentioned previously, is vast and highly interlinked. The materials used, include books, wikis, research papers and abstracts, academic lectures, and various related websites. They are all listed in the References section.

When a strong knowledge foundation was established, the second step was to investigate Quantum Computing models and to try to figure out their approach towards reflecting their conventional counterparts' functionality, including Quantum Information Theory.

The third and most important step was the establishment of the concept of Quantum Petri nets, as the proposed model. This was made possible by taking the teleportation protocol [6] and trying to fit it into the framework of simple Petri nets by gradually changing each of their components into ones that would be able to grasp the concepts of Quantum Computation. This part was more time consuming than expected as some issues came up, regarding special cases of tokens and entanglement. Moreover, some approaches for representing the state of the system at any given time, needed to be remade from scratch, more than once. Some limitations and parts of the model were also found and are listed in the final chapter of the thesis, for future work.

Finally, the teleportation protocol was clearly implemented using the finalized model. An overview of the encoding of Quantum Fourier Transformation [12] into Quantum Petri nets was also made. The teleportation protocol is a relatively simple algorithm for "teleporting" the state of a qubit that is found at one machine, to another. Despite the simplicity, all the aspects of Quantum Computing are shown, which is ideal for testing the framework we are trying to create. In the topic of Quantum Fourier Transformation,

it has many applications in many quantum algorithms, such as Shor's and quantum phase estimation algorithms, and the approach for encoding it into our proposed model is described.

1.4 Thesis Structure

This thesis is composed of five chapters.

In Chapter 2, a brief overview of the required knowledge for the understanding of the proposed model is presented. The chapter is split into the subchapters of Petri nets, Complex Number theory, Quantum phenomena and Quantum Gates. Each subchapter introduces the topic and explains the notions that are going to be used in later chapters, as simply as possible, with examples where possible.

In Chapter 3, the model of Quantum Petri nets is presented. First, a formal definition is made, and the components of them are explained in detail. Then, the normal mode of execution (forward execution) is explained. In addition, an attempt to incorporate a basic mode of reversibility, backtracking, is made. Some properties of the model, as well as assumptions that were made are also explained.

In Chapter 4, two quantum algorithms, the Teleportation protocol, and the Quantum Fourier Transformation, are explained, and encoded into Quantum Petri nets. Regarding the Teleportation protocol, a related theorem, the "No cloning theorem" is explained briefly.

In Chapter 5, a conclusion for everything contained in this thesis, is drawn. The problems and challenges faced and the ways that they were addressed are noted. Additionally, all the limitations of the model are listed. Finally, an overview of ideas for the expansion and improvement of the framework, is presented as a proposal for further work in the future.

In Appendix A, a way of representing entangled states is analyzed. Eventually, it was not used in the Quantum Petri nets model due to its complexity. However, it is a great way for working with entanglement, and it would be an omission to leave it out.

In Appendix B, a sample execution of the quantum teleportation protocol (analyzed in Section 4.1) is presented. The model used is a Reversible Quantum Petri net.

Chapter 2

Related work

Contents

2.1 Petri nets

First defined by Carl Adam Petri in 1962, as his PhD dissertation topic, Petri nets are a graphical and mathematical modelling tool, that is mainly used for the analysis/specification of parallel and distributed systems. Petri nets were initially created for the description of chemical processes but were expanded over time to many areas in Information Technology and Mathematics.

Formally, a Marked Petri net is defined as follows:

Definition 1. A (*marked) Petri net* (MPN) is a tuple (P, T, F, M_0) where:

1. P is a finite set of *places.*
2. T is a finite set of *transitions.*
3. $F : (P \times T \cup T \times P) \rightarrow \mathbb{N}$ defines a set of directed *arcs.*
4. M_0 is the initial *marking* of the Petri net.

2.1.1 Structure

A simple Petri net is composed of *tokens*, *places*, *transitions*, and *arcs*. Places may be connected to transitions through directed arcs, and vice versa. However, no place can be connected to another place, or similarly, a transition to another transition. Places contain tokens, while transitions modify the tokens contained in their respective input and output places (as defined by the directed arcs interconnecting them), by removing and modifying tokens from the input places and placing them in the output places.

A distribution of tokens over the places of a Petri net is called a marking. While a Petri net executes, it may iterate though several markings, or even repeat some.

A transition may be fired if it is enabled. This means that only if the number of the tokens contained in the input places is sufficient, the transition may fire. Generally, transitions fire in a non-deterministic manner. An arc may be labeled with the number of tokens it requires to fire, if it is incoming, or respectively the number of tokens it puts in its destination place after the transition fires, if it is an outgoing one. This labeling may be omitted if the number is 1.

Arcs can be characterized as *incoming* or *outgoing*, based on their direction from the closest transition. Put simply, an arc is outgoing if it originates from a transition or incoming if it points to a transition.

Tokens can be uniquely named, in this way they can easily be distinguished, and this provides a sturdy base for the notion of Reversing Petri nets.

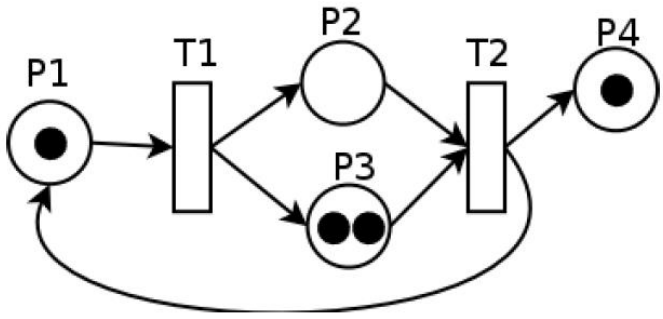


Figure 2.1: A simple Petri Net

In Figure 2.1 a simple PN can be observed. It is composed of four places {P1, P2, P3, P4}, and two transitions {T1, T2}. Place P1 is filled with a token, as well as P4, while P3 is filled with two tokens, this also captures the notion of M_0 (initial marking). There are seven arcs, of which three are incoming, and four are outgoing. The lack of labels on the arcs implies the default number 1. Additionally, T1 is an enabled transition, as all its input places are filled with tokens, while transition T2 is not enabled for firing, since place P2, that is one of its input places, is not filled with tokens.

2.1.2 Forward Execution

The standard mode of execution of most Petri net variations, as well as the classic Petri nets, is *Forward Execution.* In this mode, arcs are only followed in the direction they are pointing, all actions are irreversible for any execution. All transitions are fired, and when all transitions are disabled, then the final marking M_n can be obtained.

It is important to note that some models may loop infinitely, rendering the notion of final marking non applicable.

Some detailed examples follow, to clarify the mode of Forward execution.

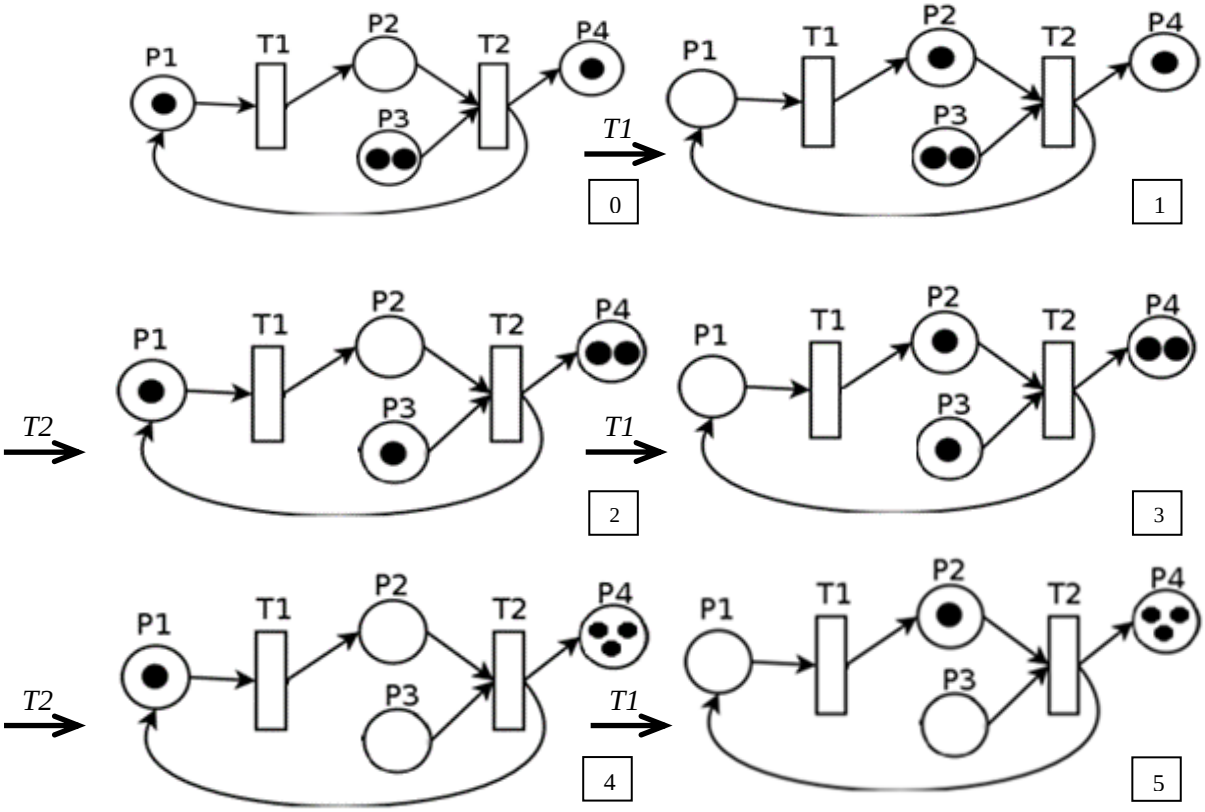


Figure 2.2: Forward Execution in Standard Petri Nets

In Figure 2.2 a Petri net is shown to progress over time from its initial state 1 (and marking) to a final state 5. The Petri net consists of 4 places (P1-P4) and 2 transitions (T1-T2). The initial Marking (M_0) specifies 1 token at P1, 2 tokens at P3 and 1 token at P4. A possible execution of this Petri net is shown. Specifically:

In state 0, the only enabled transition is T1, as the incoming arc and linked place have met the requirements (1 token needed in the arc, 1 token in P1). Thus, transition T1 is fired and place P2 which is the place pointed by the outgoing arc is filled with 1 token.

In state 1, the Marking (M_1) has changed, P2 contains 1 token, P3 contains 2 tokens and P4 contains 1 token. The only enabled transition is T2. It is fired and 1 token is removed from P2 and P3, while 1 token is placed in places P1 and P4, respectively.

In state 2, the Marking (M_2) has changed, P1 contains 1 token, P3 contains 1 token and P4 contains 2 tokens. Similarly, to State 0, the only enabled transition is T1. It is fired and 1 token is removed from P2 and P3, while 1 token is placed in places P1 and P4, respectively. T1 is fired and place P2 which is the place pointed by the outgoing arc is filled with 1 token.

In state 3, the Marking (M_3) has changed once again, P2 contains 1 token, P3 contains 1 token and P4 contains 2 tokens. The enabled transition is T2. It is fired and 1 token is removed from P2 and P3, while 1 token is placed in places P1 and P4, respectively.

In state 4, the Marking (M_4) has changed, P1 contains 1 token, and P4 contains 3 tokens. Like State 0, the only enabled transition is T1. It is fired and 1 token is removed from P1, while 1 token is placed in place P2.

In the final state (5), the final Marking (M_5) is the following: P2 contains 1 token, and P4 contains 3 tokens. This is the final state as all transitions are disabled, due to insufficient tokens in the places specified by incoming arcs. Thus, the execution ends here.

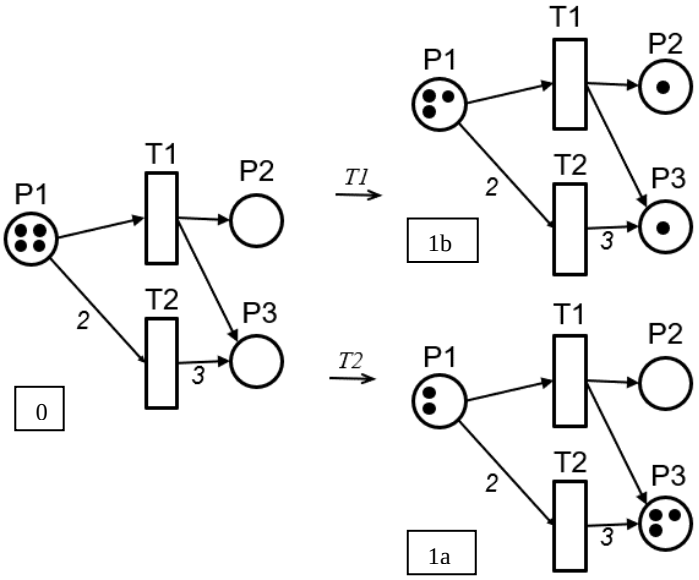


Figure 2.3: Forward Execution in Standard Petri Nets

In Figure 2.3 another Petri net is shown, with 2 possible evolution states. The Petri net has 3 places (P1-P3) and 2 transitions (T1-T2). The initial marking (M_0) specifies 4 tokens in P1. Notice that in this scenario, some arcs specify a different number of tokens. Additionally, the transitions may produce a different number of tokens as an output, than the amount they require as input.

The immediate possible next states are 1a and 1b. This non-determinism is due to the fact that both transitions T1 and T2 are enabled. In 1a, T2 is executed, and 2 tokens are removed from P1 and 3 are placed in P3. In 1b, T1 is executed, and 1 token is removed from P1 and 1 is placed in P3 and 1 in P2. Both resulting states may be analyzed further, with more executions of the transitions.

2.1.3 Reversibility

An alternate mode of execution of some Petri net variations, is *Reversible Execution.* Reversible Execution completes Forward execution in the notion of undoing a transition. In this mode, arcs are followed in the opposite direction that they are pointing. Informally, this can be decoded into a forward execution mode by reversing arc directions.

There are many modes of reversible execution, however, only backtracking will be covered and used in this thesis.

Backtracking restricts reversible execution of transitions in the exact same order that they were fired. For instance, if transitions T1, T2, T3, T4 were fired in that order for forward execution, then backtracking defines reversible execution in order T4, T3, T2, T1, etc. Backtracking can be visualized as a stack, where every transition fired in forward execution mode is pushed in the stack. In order to backtrack the transitions are reversibly executed in the order that they are popped from the stack.

Reversible Petri Nets (RPNs) [2] support reversible execution. Additionally, RPNs define uniquely named tokens, and bonds as connections of two or more tokens after they pass together from a transition.

An example follows.

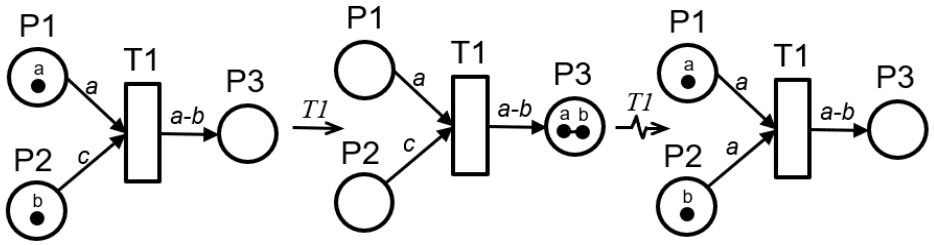


Figure 2.4: Backtracking in Reversible Petri Nets

In Figure 2.4 another Petri net is shown. This Petri net has three places (P1-P3) and one transition (T1). The initial marking (M_0) specifies one token (a) in P1 and one token (b) in P2.

First, transition T1 is executed, as a result, the tokens a and b are removed from places P1 and P2, respectively and put in P3 as a bond. T1 is also pushed as the first executed transition in the history stack of the transition, for possible backtracking.

In the final step, backtracking is observed, as the lastly executed state is undone. The transition saved on top of the history stack (T1) is popped and executed reversibly. This is done be switching arc directions; thus, the bond a-b is removed from place P3 and one token (a) is placed in P1 and one token (b) is placed in P2.

2.2 Complex numbers, vectors, and matrices

Complex numbers are critical in quantum computing as they characterize many of the features of a qubit [21], the basic computing unit of quantum computers.

Imaginary numbers were firstly conceived in the 1st century by Hero of Alexandria [24] and were regarded as fictitious or useless until the 18th century. However, by works of other renowned mathematicians, like Leonhard Euler, Augustin-Louis Cauchy, and Carl Friedrich Gauss they grew in popularity and importance. They currently are an important part of modern mathematics and computer science, as they are used in algorithms such as Fourier transformation.

2.2.1 Structure and operations

A complex number [19] is defined as any number that can be written in the form $a + bi$, where i denotes a special symbol, the *imaginary unit*. The imaginary unit is the solution the equation $i = \sqrt{-1}$, and it is called imaginary since no real number satisfies this equation. In any arbitrary complex number, $a + bi$, a is called the *real part* and b is called the *imaginary part*. Note that $a, b \in \mathbb{R}$. Examples of complex numbers are: $5 + 8i$, $-3{,}5 - i$, $\pi + \sqrt{2}i$.

An important observation is that all real numbers are complex numbers that have their imaginary part equal to 0. Thus, we can derive that real numbers are a subset of complex numbers ($\mathbb{R} \subset \mathbb{C}$).

Since complex numbers redefine numbers at their core, it is not that complex to expand the notion of real number vectors and matrices to include complex numbers.

Simply put, a *Vector* is defined as an ordered tuple of elements. In this thesis, vectors are going to be composed of complex numbers.

$$\begin{bmatrix} 1+\sqrt{3}\,i \\ 2-8i \\ -3i \\ 4 \end{bmatrix} \qquad \begin{bmatrix} i \\ -i \end{bmatrix} \qquad \begin{bmatrix} 1-3.3i \\ 8+8i \\ \sqrt{2}-2i \end{bmatrix}$$

Figure 2.5: Sample Complex Number Vectors

Similarly, a *Matrix* is an ordered tuple of vectors, or a rectangular arrangement of numbers into rows and columns.

2.2.2 Operations and Linear Algebra

Just like in real numbers, all basic arithmetic operations can be applied to complex numbers. First, an overview of binary operations will be presented, then more complex matrix operations will follow.

Addition and Subtraction

$$\begin{bmatrix} 1+\sqrt{3}\,i & 3-2i & \pi+\ i \\ 2-8i & 8.5 & 825-\ i \\ -3i & -3i & 2+\ 36i \end{bmatrix} \qquad \begin{bmatrix} 1-3.3i & 6+2i \\ 8+8i & -3+2i \\ \sqrt{2}-2i & 39 \end{bmatrix}$$

Figure 2.6: Sample Complex Number Matrices

In + and – operators, the real and imaginary parts are considered as independent and the operation is applied to the imaginary parts and real parts individually, as if they were separate; $(a + bi) + (c + di) = (a + c) + (b + d)i$. For example:

$(5 + 8i) + (9 - 3i) = 14 + 5i$.

Multiplication

In multiplication, every part of one complex number is multiplied with every part of the other complex number, following the same logic as polynomial multiplication. Remember $i^2 = -1$.

$(a + bi)(c + di) = ac + adi + bci + bdi^2 = ac + (ad + bc)\, i - bd = (ac - bd) + (ad + bc)i$

For example, $(5+8i)(9-3i) = 69+57i$

Division

In division, we multiply the numerator and denominator by the complex *conjugate*, expand and simplify. The conjugate of a complex number has the same real part, but the opposite imaginary part (i.e., the conjugate of $5+8i$ is $5-8i$).

$$\frac{a+bi}{c+di} \cdot \frac{c-di}{c-di} = \frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$$

And an example:

$$\begin{aligned}\frac{3+2i}{4-5i} &= \frac{3+2i}{4-5i} \cdot \frac{4+5i}{4+5i} \\ &= \frac{12+15i+8i+10i^2}{16+25} \\ &= \frac{2+23i}{41} \\ &= \frac{2}{41} + \frac{23}{41}i\end{aligned}$$

Matrix addition and subtraction

Matrices and vectors are very simple to handle in addition and subtraction. Two matrices are added by adding each number of the first matrix with the respective number, in the same position of the second matrix. We can define the proces formally, consider A, B, C matrices, $A, B, C \in \mathbb{C}^{m \times n}$. $A + B = C$, where $c_{ij} = a_{ij} + b_{ij}, \forall i \in [1, m]$ and $\forall j \in [1, n]$ It should be noted that matrices and vectors need to have the same dimensions for these operations to work.

For example, $\begin{bmatrix} 1+\sqrt{3}\,i & 3-2i \\ 2-8i & 8.5 \end{bmatrix} + \begin{bmatrix} i & 8+2i \\ 2-8i & -6+i \end{bmatrix} = \begin{bmatrix} 1+(1+\sqrt{3})i & 11 \\ 0 & 2.5+i \end{bmatrix}$

Matrix multiplication and dot product

In general, matrix multiplication falls into two categories, multiplication of a matrix with a scalar or another matrix. Multiplying a matrix with a scalar (an arbitrary number) is simple; every entry of the matrix is multiplied with the scalar. Multiplying with another matrix is more complex You can multiply two matrices if the number of columns in the first matrix equals the number of rows in the second matrix. The resulting matrix, (which has dimensions (rows of first matrix) × (columns of the second matrix)) is computed as follows: Each element in each row of the first matrix is multiplied by the respective element of each column of the second matrix, then the sum of the products makes each element in the resulting matrix. For instance, the sum of first row of the first matrix multiplied by the first column or the second matrix makes the first value, multiplied by the second column makes the next value. Changing rows, changes rows on the resulting matrix as well.

Let us define the process described above more formally, a matrix A with dimensions $m \times n$ and a matrix B with dimensions $n \times p$ can be multiplied to form matrix C with dimensions $m \times p$.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}, \qquad B = \begin{pmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{pmatrix}, \qquad C = \begin{pmatrix} c_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & a_{mp} \end{pmatrix}$$

The matrix product $C = AB$ is defined in such a way that $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$. Thus, C can be rewritten as:

$$C = \begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

Additionally, the product AB is defined iff the number of columns in A equals the number of rows in B [25].

Let us see an example for clarification.

$$\begin{bmatrix} 1 & 3 & 2 \\ 2 & 0 & 6 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 25 \\ 10 & 26 \end{bmatrix}$$

Notice that:

- The element at place (1,1) is equal to the first row multiplied by the first column ($4 = 1 * 2 + 3 * 0 + 2 * 1$)
- The element at place (1,2) is equal to the first row multiplied by the second column ($25 = 1 * 4 + 3 * 5 + 2 * 3$)
- The element at place (2,1) is equal to the second row multiplied by the second column ($10 = 2 * 2 + 0 * 0 + 6 * 1$)
- The element at place (2,2) is equal to the second row multiplied by the second column ($26 = 2 * 4 + 0 * 5 + 6 * 3$)

Inner Product

First let us define the *bra-ket* or *Dirac* notation [26]. A *ket* is written as: $|\psi\rangle$, and denotes a vector ψ in an arbitrary complex vector space $\mathbb{C}^m$, $\psi \in \mathbb{C}^m$. It represents a state of a m m-dimensional quantum system. A *bra* on the other hand is denoted by $\langle f|$,

and represents the conjugative transpose of $|\psi\rangle$. Column vectors represent kets and row vectors represent bras, for $\mathbb{C}^m$.

The inner product of two vectors is a fairly simple process that reflects the application of the mapping $\langle f|\psi\rangle$. Every element of the first vector is multiplied by the element that is found in its respective place in the second vector. The products are then added to yield a number. This operation is the same as the matrix product, described above. A ket can be converted to a bra, and vice versa by computing the conjunctive transpose of the vector that represents the element we want to transform.

Consider: $|\psi\rangle = \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix}, \langle\psi| = |\psi\rangle^{\dagger}[1] = [\overline{c_0} \quad \cdots \quad \overline{c_n}]$. Overlining, denotes conjugation.

$\langle\psi|\psi\rangle = \begin{bmatrix} 3+2i \\ -1 \\ -i \end{bmatrix} [3-2i \quad -1 \quad i]. = \sum_{i=0}^{n} c_i \cdot \overline{c_i}$

Tensor Product $\otimes$

The tensor product is similar to the notion of the cartesian product in sets. The tensor product of two metrices creates a new matrix with dimensions proportional to the dimensions of the multiplied matrices as follows: $A \otimes B = C$, where $A \in \mathbb{C}^{k\times l}$, $B \in \mathbb{C}^{m\times n}$, $C \in \mathbb{C}^{(km)\times(ln)}$. When computing the tensor product [3][27] of two matrices, each element of the first matrix acts as a scalar and is multiplied by the second matrix as is, like the scalar multiplication described above.

An example follows:

$$\begin{bmatrix} 1 & 3 & 2 \\ 2 & 0 & 6 \end{bmatrix} \otimes \begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 1\begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} & 3\begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} & 2\begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} \\ 2\begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} & 0\begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} & 6\begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 2 & 4 \\ 0 & 5 \\ 1 & 3 \end{bmatrix} & \begin{bmatrix} 6 & 12 \\ 0 & 15 \\ 3 & 9 \end{bmatrix} & \begin{bmatrix} 4 & 8 \\ 0 & 10 \\ 2 & 6 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 0 & 10 \\ 2 & 6 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 12 & 24 \\ 0 & 30 \\ 6 & 18 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 & 12 & 4 & 8 \\ 0 & 5 & 0 & 15 & 0 & 10 \\ 1 & 3 & 3 & 9 & 2 & 6 \\ 4 & 8 & 0 & 0 & 12 & 24 \\ 0 & 10 & 0 & 0 & 0 & 30 \\ 2 & 6 & 0 & 0 & 6 & 18 \end{bmatrix}$$

[1] The dagger notation denotes the transpose and inverse of a matrix or vector it references.

All matrix operations described above can be applied to vectors as they can be considered as matrices with just one column.

2.3 Quantum Phenomena

In the scale of atoms and subatomic particles the laws of conventional physics break down, and the environment at this level is dictated by *Quantum Mechanics*. Quantum computers harness various phenomena that occur at this subatomic scale, such as *Superposition, Spin* and *Entanglement* in order to perform computations.

Quantum Mechanics is a relatively new branch of Physics as it was developed in the early 20th century, in an attempt to explain certain phenomena that troubled the scientific community. The duality in the nature of light as a wave-particle mix was an important discovery in the field. Some renowned scientists who contributed to Quantum mechanics are: Max Planck, Albert Einstein, Niels Bohr, Werner Heisenberg, Wolfgang Pauli, Erwin Schrödinger, John von Neumann, and Richard Feynman.

2.3.1 Basic quantum theory

An experiment done by Thomas Young in 1807 (τhe double slit experiment) [3] was the foundation for extensive research over the next century, which proposed that photons and similarly sized particles exist in such a way that express both the properties of a wave and a particle.

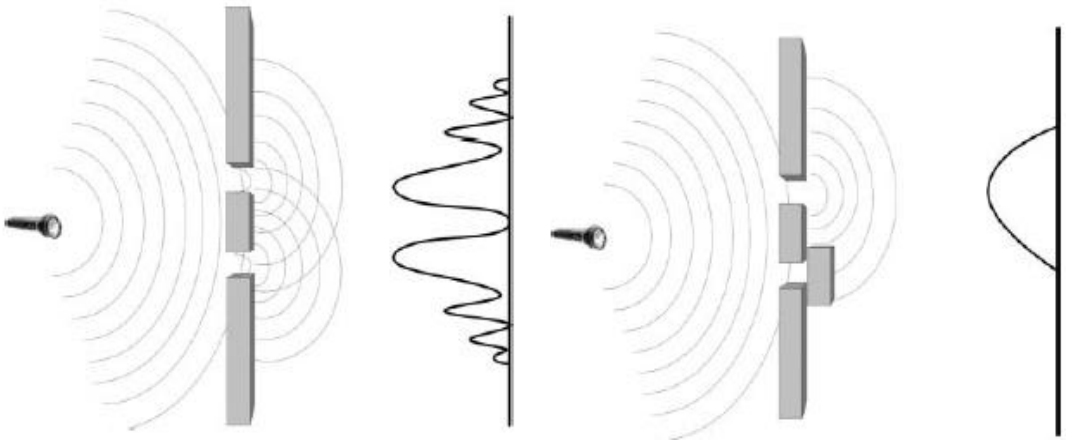


Figure 2.7: Young's double slit experiment (left), and with one slit closed (right).

The experiment consists of a light source, a thin separating layer, and a projection surface. The separating layer has two openings of about 0.5mm which correspond to the wavelength of the light emitted by the source. When the light source is turned on, a pattern forms on the projection surface. The pattern corresponds to a bump (bright center with decreasing brightness as we move away from it) when one slit is open, and to. On the other hand, when both slits are open, the projection surface consists of alternating dark and bright strips. This behavior can only be observer by a wave. This experiment implies that light is a wave, which contradicted the norm of the time that accepted light as a beam of particles.

Takeaways

The double slit experiment projects concentrations of photons in a wave like pattern on a surface [3]. We can derive from this, that all the photons emitted by the light source have a special probability signature encoded on them, that corresponds to the place they are going to land on the projection surface.

Consider the flat projection surface, we may mark it with n points, each one equally spaced with δ increment. ($x_0, x_1 \ldots x_{n-1}$). Then we denote a particle being at point x_i with $|x_I\rangle$. Where $|x_I\rangle = [0, 0, \ldots\ 1, \ldots, 0]^T$, 1 is the i^{th} position in the vector.

The signature carried by all the photons before landing in the projection surface is called the *state* of the particles. The state (denoted as $|\psi\rangle$) is composed of a linear combination of all the $|x_I\rangle$ column vectors. $|\psi\rangle = \sum_{i=0}^{n-1} c_i |x_I\rangle$ where c_i is a complex number and $\sum_{i=0}^{n-1} c_i^2 = 1$. The particles shot through the slit(s) are in *superposition* of these column vectors. More on this topic will be discussed in chapter 2.3.2

2.3.2 Superposition

This fundamental principle of Quantum mechanics states that any subatomic particle may be expressed as the addition of two or more valid quantum states. A valid subatomic particle (such as a photon or an electron) can be at an “excited” or a “non-excited” state, which directly translates the energy level it contains. Quantum mechanics allows a particle to be in any of these two states, or a combination of them. For instance, under the right conditions, an electron may be at 50% “excited” and 50% “non-excited” states, or at 100% “non-excited” state etc. These states and their combinations may be visualized

with the help of a Bloch sphere. Quantum computers use a particle as the basics information unit, each particle is able to represent a two-state quantum system, as previously explained, they are referred to as *qubits.*

In Quantum mechanics, these two states are referred to *basis states,* and are represented by the notations $|0\rangle$ and $|1\rangle$. Any possible quantum state is written as a combination of these two basis states, $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$, where c_0 and c_1 are two complex numbers that satisfy the following equation: $c_0^2 + c_1^2 = 1$.

The state of a qubit is usually written as a column vector with the coefficients for each basis state in the respective position. For instance, the state: $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$ is written as $|\psi\rangle = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$. Similarly, quantum systems with more qubits may be written with bigger vectors, reflecting their size, such as:

$|\psi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$, is written as $|\psi\rangle = \begin{bmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{bmatrix}$, etc.

Bloch sphere

The Bloch Sphere [18], named after Felix Bloch, is a geometrical representation of the pure state space of a quantum particle (or a two-state quantum system, regarding the polarities).

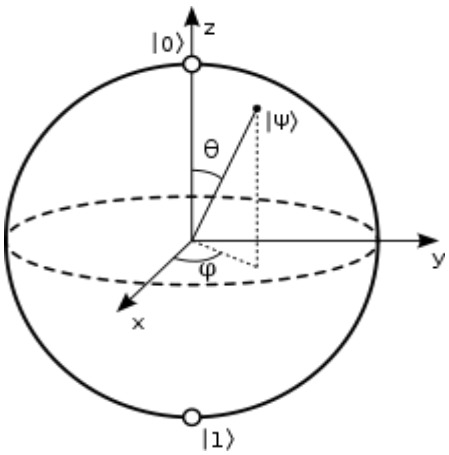


Figure 2.8: Bloch Sphere

The Bloch sphere is derived from the following: any pure state of a two-level quantum system can be written as a superposition of the two basis states $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$. Thus, four real numbers are needed to specify the coefficients, in order to define a state, since only the relative phase[2] between the two basis states is meaningful, the coefficient of $|0\rangle$ may be considered as real and non-negative. Now only three real numbers are needed to define a pure state, that denote the

[2] **Phase** is the part denoted by the imaginary part of the coefficients of a quantum state. The phase does not affect the outcome of the measurement, it is however an important property of quantum systems. There are gates in quantum computers that act on phase alone.

three dimensions of the Bloch sphere. Pure states are limited on the surface of the sphere, whereas mixed states are composed of all the points within the Bloch Sphere.

Since $|\psi\rangle^2 = 1$, the arbitrary pure state $|\psi\rangle$ can be written as: $|\psi\rangle = \cos\theta|0\rangle + e^{i\varphi}|1\rangle$, where $0 \leq \varphi < 2\pi$ and $0 \leq \theta < \frac{\pi}{2}$

The antipodal points (north and south poles) represent the two basis states that the particle may be found in, and their superposition is controlled with the θ variable. The φ variable on the other hand, controls the phase of the quantum system.

The Bloch sphere is often found in representations of qubits and their states, more about qubits in section 2.4.

2.3.3 Entanglement

Referred to as "A spooky action at a distance" by Einstein, this quantum property is crucial for quantum information technology. In the previous section, we examined the properties of a single particle, or a two-state quantum system, in many areas of quantum mechanics however, many particles are used together in order to perform more complex operations.

Consider a four-state quantum system, since a particle is able to only be superposed to a two-state quantum system, we now need two particles to be able to generate it. Suppose that the first particle p1 is found in state $|\psi_1\rangle = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$ and particle p2 is found in state $|\psi_2\rangle = \begin{bmatrix} c_2 \\ c_3 \end{bmatrix}$. In order to compute the state they produce together in a four-state quantum system, we use the notion of the tensor product (⊗), analyzed in a previous section.

$$|\psi_{shared}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \otimes \begin{bmatrix} c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \cdot \begin{bmatrix} c_2 \\ c_3 \end{bmatrix} \\ c_1 \cdot \begin{bmatrix} c_2 \\ c_3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} c_0 c_2 \\ c_0 c_3 \\ c_1 c_2 \\ c_1 c_3 \end{bmatrix}$$

This is a perfectly valid state, as the notion of $|\psi\rangle^2 = 1$ is maintained and is dependent on the two other composing state vectors.

Now, if we reverse the process, we can obtain the two state vectors of the particles composing the four-state quantum system, that are independent from each other. However, this is not always possible, as some shared states cannot be broken into tensor

products of two-state quantum systems. States that fall under this category are called *entangled* and we say that *entanglement* exists between the involved particles, or any two-state quantum systems that can be used to produce them.

An example of an entangled state is the first *Bell state*[3] $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ which is inseparable.

It is important to note that measurements on particles of separable states have no impact on the other particles, as they are independent, as mentioned above. This is not the case for entangled states, as measuring one entangled particle, directly affects the state of the other(s).

The notion of decoherence may now be introduced. Qubits are isolated particles in a quantum computer, with operations acting upon them, defined by *quantum gates*. Since particles, thus qubits as well, tend to get entangled with each other, there is the possibility of a qubit interacting and becoming entangled with another particle that is not being monitored. As quantum computers can only isolate the qubits at a certain degree. The implications of this, is that our monitored qubit will behave unexpectedly, and the operations will be wrong.

2.3.4 Observables and Measurement

Consider the double slit experiment again, but remove the projection surface, now is there a way that we could extract the state of the particles while they are travelling in space? In quantum mechanics, there are some operations that may be done on the particles that help us obtain some aspects of their state.

These operations are matrix multiplications with some predefined Hermitian operators, called *observables*.

Simply put, observables[22] allow us to pose questions to the quantum system and receive the answers by modifying the state of the system, this is called a *collapse* to a basis state. An example would be asking the system of the double slit experiment on its position. This would be extracted by multiplying the state of the system with the position

[3] Bell states, derived from John Stewart Bell's work and synonymous theorem, are four four-state quantum system states that represent maximal entanglement between the involved two sub systems.

observable's matrix. The result would be the collapse to a position x_i and the probabilities of collapsing in each position. The act of using an observable to obtain this information is called *measurement.*

Let us introduce the concepts of *eigenvectors* and *eigenvalues.* Consider a matrix M in $\mathbb{C}^{m\times m}$, a vector V in $\mathbb{C}^{m}$ and a real number λ. If MV=λV, then the vector V is called eigenvector of M and λ is called an eigenvalue for the respective eigenvector. A matrix may have zero to many eigenvectors and eigenvalue pairs.

All observables have unique eigenvalue and eigenvector pairs. Each eigenvector of an observable marks one of the possible basis states the measured system may collapse into.

A measurement results in the following:

a. The output of the eigenvalue of the measurement of the respective eigenvector that the qubit(s) will collapse into.
b. The measured qubit(s) probabilistically collapses in one of the basis states.

A measurement [30] encodes a *Hermitian* operator (matrix M where $M = M^{\dagger}$) of an observable that acts on a quantum state. An observable has two eigenvectors and their respective eigenvalues. For each eigenvector $|e_i\rangle$ a matrix P_i is calculated with $P_i = |e_i\rangle\langle e_i|$. Now by calculating $p(i) = \langle\psi|P_i|\psi\rangle$ we obtain the probability of the measured state ψ being in state $|e_i\rangle$.

The measurement obtains an eigenvalue λ_j where j denotes the j^{th} eigenvector that the qubit will collapse in.

Moreover, for z-axis measurements ($|0\rangle\ and\ |1\rangle$) the probability of collapsing into any eigenvector $|e_i\rangle$ is obtained by computing the squared of the absolute value of the inner product between the original state $|\psi\rangle$ and the eigenvector $|e_i\rangle$, e.g., $P_1 = |\langle\psi|e_1\rangle|^2$ for a two-state system. This probability is also computed by squaring the absolute of the coefficient c_i of the eigenvector $|e_i\rangle$.

Let us proceed to a complete measurement example.

Consider the state of a four-state quantum system, composed of two qubits, namely $\alpha\begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}$ and $b\begin{bmatrix}1\\0\end{bmatrix}$.

Using the observable $\Omega \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ for measuring the state of α

The probability of collapsing into state $|0\rangle$ is $|c_0|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = 0.5 = 50\%$.

The probability of collapsing into state $|1\rangle$ is $|c_1|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = 0.5 = 50\%$

The probabilities were computed from the coefficients as the observable is for the z-axis. Using the normal extensive process for obtaining the probabilities, we have the following:

$$|e_0\rangle = |0\rangle \; and \; |e_1\rangle = |1\rangle, \qquad \lambda_0 = 1 \; and \; \lambda_1 = -1$$

For $|e_0\rangle$:

$$P_0 = |e_0\rangle\langle e_0| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} [1 \quad 0] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$p(0) = \langle a|P_i|a\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{2}$$

The same process for $|e_1\rangle$ results in $p(1) = \frac{1}{2}$

The application of the measurement obtains an eigenvalue that relates to an eigenvector $|e_i\rangle$. It is at this point that the qubit probabilistically collapses into an basis state, denoted by the eigenvector.

Suppose then that we obtain the eigenvalue λ_0 then the state that the qubit collapses into is $|0\rangle$, thus, $|a\rangle = |0\rangle$.

The other case is obtaining the eigenvalue λ_1. Then the state that the system would collapse into is $|1\rangle$, then $|a\rangle = |1\rangle$.

Additionally, let us consider a four-state quantum system state: $|\psi\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$

Notice that this state is entangled. Suppose then that we measure the first qubit, and it collapses into state $|0\rangle$. Then it is impossible for the second qubit to be in state $|1\rangle$. This can be seen from the shared state of the qubits before the measurement, since each place in the state vector denotes the portion of the system's state being in the respective basis state (such as $|00\rangle$ $|10\rangle$ etc). Since the state only had values in the places of states $|00\rangle$ and $|11\rangle$ then by measuring the first qubit being in state $|0\rangle$ then the only possible outcome for the second qubit is $|0\rangle$ as well.

The new superposition is obtained by crossing out all those terms of $|\psi\rangle$ that are inconsistent with the outcome of the measurement (i.e., those whose first bit is 0). Of course, the sum of the squared amplitudes is no longer 1, so we must renormalize to obtain a unit vector:

$$|\psi_{new}\rangle = \frac{c_{10}\,|10\rangle + c_{11}\,|11\rangle}{\sqrt{|c_{10}\,|^2 + |c_{11}\,|^2}}$$

2.4 Quantum Gates

Modern computers are composed of a central processing unit that carries out all operations between the basic value units, aka bits. The CPU takes groups of bits, often organized in bytes, and applies certain modifications upon then by passing them through special gates. For instance, the byte 00110011 passed through the NOT gate becomes 11001100. The AND and OR gates are examples of gates that take more than one byte input.

Quantum computers revolutionize computation by introducing their own concept of *quantum gates* and basic value units. The basic units of information in quantum computers are called *qubits*, and what makes them different from classical bits, is the fact that they can be superposed between the basis states $|0\rangle$ and $|1\rangle$, whereas a conventional bit can only take one of the basis states.

2.4.1 Representation

As expected, quantum gates [20] are operators that act upon qubits or groups of qubits and modify them. All quantum gates are composed of a matrix that when multiplied by a state of a quantum system, change it accordingly. Most important quantum gates act on a single qubit or two. Quantum logic gates are represented by unitary matrices (a matrix M is unitary if $MM^{\dagger} = \mathrm{I}$ I is the identity matrix). A gate which acts on n qubits is represented by a $2^n \times 2^n$ unitary matrix.

Here are the matrices of some quantum gates that will be used in thesis, and their effects on the basis states $|0\rangle$ and $|1\rangle$:

Pauli matrices:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Converts $|0\rangle$ to $|1\rangle$ and vice versa

Converts $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$

Converts $|1\rangle$ to $-|1\rangle$

CNOT gate:

The CNOT gate inverts the value of the input if the control qubit is in state $|1\rangle$.

(control qubit first) $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, (control qubit second) $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

Converts the state of the controlled qubit from $|0\rangle$ to $|1\rangle$ and vice versa, if the control qubit is in state $|1\rangle$.

Some examples for clarification follow:

Consider two qubits q0 and q1 with states $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ respectively. Using the CNOT with the first qubit as the control qubit, we can change the state of q1 if the state of q0 is $|1\rangle$, which it is thus the state of q1 is converted from $|0\rangle$ to $|1\rangle$.

If q0 was in state $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ the state of q1 would not be changed at all

Hadamard gate:

Changes the basis of the qubit into Hadamard basis states, from $|0\rangle$ and $|1\rangle$, to $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

Identity matrix:

Does not affect the state of the system.

$$I = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

2.4.2 Forward execution

Since the states of qubits are represented by vectors, and the gates are represented by matrices, the first thing that comes to mind in order to combine them is by using the dot product, and that is exactly the way a quantum system evolves [3]. The simplest scenario is a single qubit with one gate, the matrix has a matrix $\begin{bmatrix}a & b\\ c & d\end{bmatrix}$ and the qubit is initially in state $\begin{bmatrix}s_0\\ s_1\end{bmatrix}$. The state of the qubit after passing through the gate is found by: $\begin{bmatrix}a & b\\ c & d\end{bmatrix} \cdot \begin{bmatrix}s_0\\ s_1\end{bmatrix} = \begin{bmatrix}as_0 + bs_1\\ cs_0 + ds_1\end{bmatrix}$. This process is repeated for all gates in the system.

Now, two (or more) qubit gates, contain a 4 by 4 (or bigger) matrix. The questing that may arise here is "How combine the states of the qubits in order to carry out the dot product with the gate's matrix?". The answer comes from the tensor product, two qubit gates require a vector in $\mathbb{C}^4$ that is able to be constructed from both input qubits. Such a vector is found by computing the tensor product of two qubits, which is in turn multiplied with the matrix of the transition. The qubits' states after the execution of the gate are found if the resulting combined state vector is able to be written as the tensor product of two composing qubit states, which are then assigned to each qubit, in the order they were used to compute the combined state vector before the execution of the gate. In the case that we are unable to write the resulting combined state vector as a tensor product of two composing qubit states, then the qubits are entangled. Some examples follow:

Example 1:

Qubits: q1$\begin{bmatrix}0\\1\end{bmatrix}$, q2$\begin{bmatrix}1\\0\end{bmatrix}$, gate: $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$.

The combined state is found by: $q1 \otimes q2 = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$

Now we can multiply it with the gate matrix $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \cdot \begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

The resulting combined state is: $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$, which can be written as $\begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix}$. That are the new states of the qubits q1 and q2, respectively.

Example 2:

Qubits: q1$\begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}$, q2$\begin{bmatrix}1\\0\end{bmatrix}$, gate: $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$.

The combined state is found by: $q1 \otimes q2 = \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\\frac{1}{\sqrt{2}}\\0\end{bmatrix}$

Now we can multiply it with the gate matrix $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \cdot \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\\frac{1}{\sqrt{2}}\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix}$

The resulting combined state is: $\begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix}$, which cannot be written as a tensor product of two qubit states. Qubits q1 and q2 are now entangled.

Another question that may arise at this point is " How do we compute the next state of two qubits after passing through a two-qubit gate if one of them is entangled with a third qubit, hence it lacks an individual state?". The answer again is tensor product. In the case of an qubit passing through a gate and it being entangled with one or more qubits that are left out of the gate, we "expand" the matrix of the gate so it can include the left out qubits, and as a result the whole entangled shared state. The tensor product of the gate's matrix with the identity matrix (size relative to the number of entangled qubits left out) is computed and the resulting matrix is used as above, to calculate the next state of the system. If the entangled qubit is located at the start of a multi-qubit gate, then the tensor product is: $I \otimes Gate$, if it is at the end, it is in the opposite order: $Gate \otimes I$. In single qubit gates, the order is insignificant. For instance:

Qubits: q1,q3$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$, q2$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, gate: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$.

The combined state is found by: $q1q3 \otimes q2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$,

Now, the transition matrix needs to be expanded, the entangled qubit (q1) is placed first in the transition: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Now we can multiply the resulting matrix with the combined state, which results in state

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

This state cannot be written as a tensor product of qubits. Thus, an entanglement of those three qubits is created.

Any quantum system contains one or more qubits that are in specific states. The evolution of the system is dictated by the gates that are encountered in order. The principles applied for any quantum system are the ones that were analyzed above.

Now let us briefly introduce the representation of quantum circuits. Graphically, every qubit is identified by its unique name on the left and its state is shown in ket notation. A continuous horizontal line starts after the declaration of the state of each qubit, which will then enable gates to be sequentially placed on these lines and show the order of operation. Quantum gates are usually denoted by rectangles with a symbol or letter in them or just by a symbol, referring to the respective gate they represent. For instance, H denotes the Hadamard gate, X denotes the X Pauli or NOT gate (also symbolized with ⊕), etc. The Controlled NOT (CNOT) gate is denoted by connecting the control and controlled qubits with a vertical line. On the one end of the qubit connecting line there is a filled dot, which denotes the control qubit, and on the other end the NOT gate representation (⊕), which is going to have its state changed.

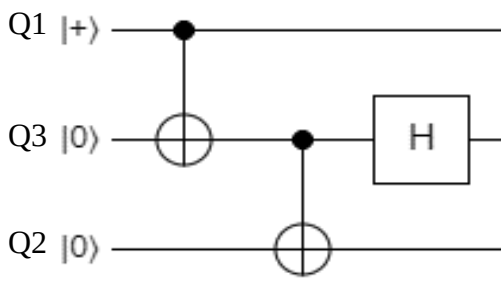


Figure 2.9: A quantum circuit

A usual representation of Quantum Gate systems is the one shown in Figure 2.9. The qubits are presented in *ket* ($|x\rangle$) notation and the gates are symbolized with interconnecting lines or squares with a character. The system shown in the figure is actually the one described in the previous example, with two consecutive CNOT gates and a Hadamard gate following the qubit q3. The first CNOT exists to create the entanglement between q3 and q1, whereas the second simulates the example above. Notice that there is no difference in the outcome if the control qubit for the second CNOT is q1 or q3, as the entangled state will be taken into consideration in any case.

Summarizing, we can derive that there are four main postulates in quantum computing [28].

1. A quantum system can be written as a complex vector.
2. Observables are represented by Hermitian matrices.
3. A measurement returns one of the eigenvalues of the matrix and results in the measured system collapsing into the corresponding eigenvector.
4. The evolution of the quantum system is computed by unitary transformation matrices with the state of the system.

This concludes the chapter of the background knowledge; it is strongly advised that the reader is able to understand how quantum systems evolve and operate in general, as well as an understanding of the model of Petri nets.

In the next chapter, the concept of Quantum Petri nets will be introduced and operated, as a means of expanding distributed and parallel models of conventional

Chapter 3

Quantum Petri nets

Contents

In this chapter, we are now at a point where we can formally define a model that satisfies the criteria and requirements that were set in Chapter 1.

3.1 Model Structure

Definition 2. A *Quantum Petri Net* (QPN) is a tuple (*A, P, T, MT, F, M, Val*) where:

1. *A* is a finite set of tokens ranged over by a, b ... Each one represents an arbitrary qubit.
2. *P* is a finite set of places.
3. *T* is a finite set of transitions. A transition is a tuple (Matrix, Variables, Conditional).
4. *MT* is a finite set of Measurement transitions. A MT is a tuple (Observable's matrix, Variables, Conditional).
5. $F : (P \times T \cup T \times P) \rightarrow 2^V$ defines a set of directed arcs. Arcs are labeled to define which variables are carried through them.
6. $M : P \rightarrow 2^A$. M_0 is the initial Marking of the QPN. The marking is defined as a set of the distribution of tokens over the places.
7. $Val : partition(A) \rightarrow \mathrm{V},\ \mathrm{V} \in \mathbb{C}^m, m \in \mathbb{N}$ where $Val(x) \cap Val(y) = \emptyset$ for all x, y. Also, if $|x| = n$ then $Val(x) \in \mathbb{C}^{2^n}$ and if $Val(x) = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$ then $\sum_{i=1}^{n} v_i^2 = 1$. Val_0 is the initial Value of the QPN. The Value component is composed of the states of each qubit, and the common states of entangled

qubit groups. In the case of two or more tokens being expressed by a single state vector in this component, then those states are Entangled.

A Quantum Petri net is built on the basis of a set of *tokens*. Each token represents a qubit[4] in a quantum system and is uniquely named. Each token also has a value characterized by a state vector in $\mathbb{C}^2$. We consider tokens to be indestructible and impossible to be generated out of nothing throughout the execution of a QPN. In this way their persistence can be guaranteed, and their history inferred from the structure of a Petri net through the introduction of the *History* element in section 3.3. They occur as stand-alone elements, but they may be entangled with each other.

Places have the standard meaning, which is, simply put, a reservoir for the tokens to exist in. They denote that the tokens that are enclosed into them share a property (such as progress in the algorithm defined by the QPT).

Directed *arcs* connect places to transitions and vice versa are labeled by a subset of V as defined in the transition they are incoming to or outgoing from. For a label $\ell = F(x, t)$ or $\ell = F(t, x), t \in T, x \in P$ we assume that each variable can appear in ℓ at most once. If $v \in F(x, t)$ then a token b which can be encoded into v is required for the transition t to fire.

The initial *marking* M_0, represents the distribution of tokens across the places of the QPN, before any transition takes place. The marking evolves over time with every transition executed, resulting in M_1, M_2… . For example

M_0 = {P1: {a,b}, P2:{}, P3:{c}}

The initial value Val_0, represents the states of the tokens of the QPN, before any transition takes place. In the case of entangled qubits, the shared state is contained in the value component for all the entangled states of each entangled group . The value evolves over time with every transition executed, resulting in Val_1, Val_2… The Value is directly linked to the Marking component as they complement each other and evolve together over time. For example:

[4] Note that the terns of tokens and qubits will be used interchangeably for this chapter.

$Val_0 = \{a: \begin{bmatrix}0\\1\end{bmatrix}, bc \begin{bmatrix}0\\0\\1\\0\end{bmatrix}\}\}$

Transitions represent quantum operations and essentially manipulate the state of the input qubit(s), such as CNOT and Hadamard gates. The matrix component of each transition is described by its square transition matrix, which has dimensions analogous to the number of input tokens, $2^n \times 2^n$, where *n* is the number of input tokens.

The Variable set *V* of the transition tuple is an ordered set that defines which variables are going to be used by the transition, and in which order. Each transition defines its own set variable set $V = \{v_0 \dots v_n\}$. Each variable represents a quantum state. Moreover, we define a function size, which assigns a vector size (2^m, $m \in \mathbb{N}$) to each variable. This size essentially reflects the variables' vector size ($\mathbb{C}^{2^m}$). In this way, a variable may take a size of the state of an input token, or its shared state if it is entangled.

For a specific distribution of tokens into the variables, dist is the respective ordered token set. We refer to the i^{th} element of an order set M as M_i. Consider A^* as a valid partition of A that and $Val(A_p^*) = V_p$.

The size of each variable is decided dynamically through the execution of the QPN, right before the transition takes place, as follows:

$$size(V_i, dist_i, Val) = |2^n| \; where \; n = |A_p| \; and \; dist_i \in A_p$$

The conditional part of transitions contains an expression, which is a condition COND that is built using a simple propositional language and an assignment of input tokens to variables. The expressions may be evaluated using the current state of the QPN, which results in a Boolean value; "TRUE" or "FALSE". If the expression is true, then the transition is enabled, thus it is ready to fire. The conditional expression and evaluation languages are a modification of the ones proposed in [29].

The propositional language that is used to build the conditional expression consists of names of variables, which correspond to tokens in the QPN, which in turn correspond to quantum states, and Boolean symbols which are coupled with the variables to produce a Boolean result.

We can construct the following grammar:

$$\phi := \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid v = s$$

Where ϕ denotes an expression, v denotes a variable and s is a quantum state, $s \in \mathbb{C}^m$.

Additionally, we define the distribution function: $Dist : V \to A$. V is the ordered Variable set containing all the variables used in the conditional part of a transition as well as in the arcs, as defined above. This function dynamically creates a possible matching at the time of execution of the QPN, or it is defined beforehand for restriction purposes.

Based on the notions mentioned above, we can now define an evaluation function that maps the conditional expression of a transition into a Boolean value.

For a Transition T:

$$Eval(\phi, M, Val, Dist) = \begin{cases} \neg Eval(\phi', M, Val, Dist), & if\ \phi = \neg\phi' \\ Eval(\phi_1, M, Val, Dist) \vee Eval(\phi_2, M, Val, Dist) & if \phi = \phi_1 \vee \phi_2 \\ Eval(\phi_1, M, Val, Dist) \wedge Eval(\phi_2, M, Val, Dist) & if \phi = \phi_1 \wedge \phi_2 \\ True, \quad if\ \phi := v = s, Dist(v) = a, Val(a) = s, a \in M(p)\ and\ p \in \bullet T & \\ False, & Otherwise \end{cases}$$

The function $Eval(\phi, M, Val, Dist) \to BOOL$ evaluates the conditional expression Φ into a Boolean value.

Additionally, the order of the variables in the ordered set V used in the expression of a transition, denote the order that the transition will perform its computations. More on this will be presented in the following section.

Measurement Transitions correspond to the measurement operation in quantum systems and extend the standard notion of transitions. The conditional and variable set parts are defined in the same way as in standard transitions. Each qubit entering a Measurement Transition, usually has a probabilistic transformation of its State Vector. The measured qubit (token) with state $\begin{bmatrix} a \\ b \end{bmatrix}$, either collapses at state $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ with probability a^2 or at state $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ with probability b^2 for the observable with these two eigenvectors. In case of entangled qubits, the common state vector is either transformed into separate qubit

[5] The •t symbol denotes all the places that are before transition t, or the places that incoming arcs to the transition connect to. Similarly, t• symbolizes the places that come after the transition t, the places that outgoing arcs from the transition connect to

vectors, thus eliminating the entanglement, or is narrowed down to $\mathbb{C}^{(x-1)^2}$, where x is the number of entangled qubits before the measurement. An eigenvalue is obtained after the measurement, and the qubit (plus any effects of the any previously existing entanglement) collapses in any basis state, based on the calculated probability, using the coefficients of the eigenvalues.

A Measurement (and as a result a Measurement Transition) results in the following:

a. The eigenvalue of the measurement of the respective eigenvector that the qubit will collapse into is obtained.

b. The measured qubit probabilistically collapses in one of the basis states.

The process described above and more specifically in section 2.3.4 is encoded into MTs, for the tokens passing through.

In the case that the token being measured is entangled with another, then the measurement acts on the common state vector, as if both qubits were being measured. This causes the entanglement to "break" and the state of the entangled qubit collapses into a basis state too. The process for the breakup of bipartite states into tensor products of simpler quantum systems is described in the following section, where the operation of transitions will be explained in detail.

Graphically, the tokens are denoted by black dots with their name close by, transitions as rectangles that have their respective symbol (e.g., cnot) if applicable, the conditional part is shown near a transition. Places are symbolized as circles with tokens or not within. Arcs are depicted as directed arrows connecting places and transitions, and vice versa, with their label. Entangled tokens are connected with a dashed line.

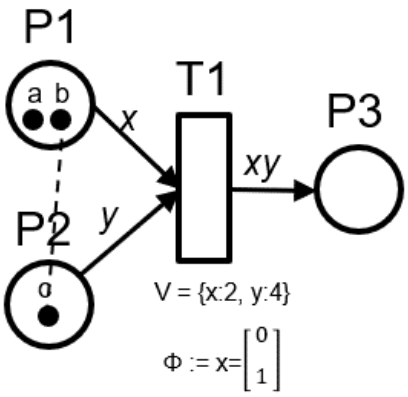


Figure 3.1: A Simple Quantum Petri net

We are now ready to proceed to an example, as all the components have been examined!

Let us examine a sample QPN for clarification. For instance, in the drawn QPN there are 3 places {P1, P2, P3}, one transition T1, with properties:

$$T1 = \{\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \{ V = \{x, y\}, \Phi := x=\begin{bmatrix}1\\0\end{bmatrix}\}.$$

and the marking is M_0 ={P1: {a,b}, P2:{c}, P3:{}}, Val_0 ={a: $\begin{bmatrix}0\\1\end{bmatrix}$, bc $\begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix}$}}.

There are no Measurement Transitions.

The order followed by the conditional part of the Transition, in order to carry out the correct operation (e.g., CNOT) is x, then y, then z. The corresponding tokens' distribution based on the Dist function is {a, c}.We can also notice that the conditional part of the transition allows token a to be in a specific state, whereas b and c can be in any state, for the transition to be enabled.

The evaluation of the expression of T1 is mapped to true as: x=a=$\begin{bmatrix}0\\1\end{bmatrix}$. The transition is enabled in this state.

The tokens b and c are entangled.

Note that we define the global state of a QPN as $\langle M, Val\rangle$. The combination of the Marking and Value components is enough to uniquely distinguish the current state of the QPN from any other.

3.2 Forward Execution

The main mode of execution of a Quantum Petri net, is *Forward Execution*, just like with most Petri net variations. Let us first see a pseudocode that defines this mode, then we will define it more formally.

First, let us define a transition t as *enabled* in a QPN's state $\langle M, Val \rangle$ if:

1. There exists an assignment of tokens Dist to F(x,t) for all $x \in \bullet t$, such that $Dist(v) = a \Rightarrow a \in M(x)$.
2. $Eval(\phi, M, Val, Dist) \rightarrow true$, for the conditional expression Φ assigned to t, and a valid assignment of tokens Dist, as mentioned in 1.

The QPN evolves over time as follows:

1. An enabled transition is selected.
2. The tokens are removed from the input places, inserted into the transition in a specified order and outputted in a different state in the places specified by the outgoing arcs.
    a. A transition is a matrix corresponding to a quantum gate. For single qubits, the effect of the transition is the dot product of the transition matrix and the state of the input qubit. For multiqubit gates, the logic is similar, with the difference that since the transition matrix is at least 4x4, the shared state vector, which represents all the input qubits must be computed for the dot product to work. This is done by computing the tensor product of the input qubits (with the correct input order, defined in the conditional part of transitions), resulting in the required shared state vector. The dot product may now be computed, the resulting state vector may be split if possible. If not, then an entangled state has occurred, and the qubits involved are now entangled. This appears in the new Marking and Value components.
    b. Measurement transitions convert the input token's state to either $|0\rangle$ or $|1\rangle$ and severing any entanglements in place. The "released qubits" from the entanglement have a defined state based on the outcome of the measured qubit. The marking is updated accordingly with the new states. MTs are irreversible.
3. The marking and value components are updated with the new states and new token distribution.

You may notice that Forward execution is almost the same as with standard Petri nets, the only differentiation is at the execution of transitions, and the effects that come along with it.

Now, let us formally define the algorithm described above.

Formal definition:

If Transition T is enabled in an arbitrary QPN state $\langle M, Val\rangle$, then, the next state $\langle M', Val'\rangle$ is computed with the execution of a transition T; $\langle M, Val\rangle \xrightarrow{T} \langle M', Val'\rangle$. A transition is composed of matrix t, conditional expression Φ,a distribution of tokens Dist and variable set V.

If $Eval(\phi, M, Val, Dist) \rightarrow true$ then:

For all places x and tokens a:

$$M'(x) = \begin{cases} M(x) - Dist(v), & \text{if x} \in \bullet \text{T, v} \in \text{F(x, T)} \\ M(x) \cup Dist(v), & \text{if x} \in \text{T} \bullet \text{, v} \in \text{F(T, x)} \\ M(x), & \text{otherwise} \end{cases}$$

and

$$Val'(A_{s1}), \dots Val'(A_{sm}) = \begin{cases} S_1, \dots, S_m \;\; where\; S_1 \otimes \dots \otimes S_m = \left(Val(A_1') \otimes Val(A_2') \otimes \dots \otimes Val(A_n')\right) \cdot T, \\ i \in [1, n], m = max, a_1 \dots a_k = \cup A_i, S_i \in \mathbb{C}^{2^{l_i}}, A_{si} = a_{\sum_{p=1}^{i} l_p - l_i} \dots a_{\sum_{p=1}^{i} l_p}, \;\; \text{if Dist(v)} \rightarrow A_i' \\ Val(A_{s1}), \qquad \text{m} = 1, \text{otherwise} \end{cases}$$

The above implies that the tokens are kept as simple and independent as possible. All tokens are written with the smallest possible state vector, with only inseparable token groups being described by a shared state vector, thus being entangled.

In case of entangled qubits, the entangled qubit may appear first or last in the conditional part of a transition. The operation is carried out in the same way.

We will now graphically denote the evolution of a QPN with the following arrow:

$\longrightarrow_{T_i}$ Denotes that the QPN at the pointing end, is the QPN on the opposite end, but with the transition T_i, forward executed.

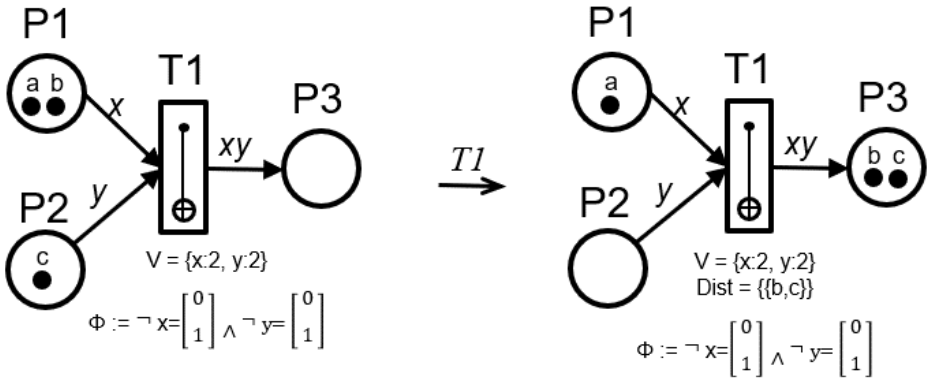


Figure 3.2: Forward execution in a QPN

Returning on a similar example. In the drawn QPN there are 3 places {P1,P2,P3}, one transition T1(CNOT) and the marking is M_0 ={1: {a,b}, 2:{c}, 3:{}}, Val_0 = {a: $\begin{bmatrix}0\\1\end{bmatrix}$,b $\begin{bmatrix}1\\0\end{bmatrix}$,c $\begin{bmatrix}1\\0\end{bmatrix}$}. The transition is enabled as both b and c that are required, are provided by the incoming arcs and are not in the prohibited states. The order of the qubits entering the transition is: b, c, which are assigned to variables x, y accordingly. The transition matrix for cnot in this orientation is $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$. The tensor product of $x \otimes y$ is computed so the dot product with the matrix reveals the next computed state. $x \otimes y = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \cdot \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, and is translated back to x=b $\begin{bmatrix}1\\0\end{bmatrix}$, y=c $\begin{bmatrix}1\\0\end{bmatrix}$.

Thus M_1 =-{{1: {a}, 2:{}, 3:{b, c}} Val_1 =-{{a: $\begin{bmatrix}0\\1\end{bmatrix}$,b $\begin{bmatrix}1\\0\end{bmatrix}$,c $\begin{bmatrix}1\\0\end{bmatrix}$}}.

It should be noted that we suppose that all quantum gates and transition matrices can be expressed as two and one qubit matrices and gates. This is a valid assumption as a subset of two and one qubit gates can be universal.

Let us now proceed to measurement transitions.

Formally, the next state of a QPN, after a measurement is probabilistically calculated based on the pre-measured state of the measured qubit and the observable used.

If Measurement Transition MT is enabled in $\langle M, Val\rangle$, then, the next state $\langle M', Val'\rangle$ is computed with the execution of a transition MT; $\langle M, Val\rangle \overset{MT,p(i)}{\rightarrow} \langle M', Val'^{i}\rangle$. A transition is composed of matrix t, conditional expression Φ,a distribution of tokens Dist and variable set V. The token used for the execution of the transition is referred to as d.

If $Eval(\phi, M, Val, Dist) \rightarrow true$ then:

For all places x and subsets of A, A*:

$$M'(x) = \begin{cases} M(x) - Dist(v), & \text{if x} \in \bullet \text{T, v} \in \text{F(x, T)} \\ M(x) \cup Dist(v), & \text{if x} \in \text{T} \bullet \text{, v} \in \text{F(T, x)} \\ M(x), & \text{otherwise} \end{cases}$$

and

$$\begin{matrix} Val'^{0}(A^*) = |e_0\rangle \\ Val'^{1}(A^*) = |e_1\rangle \end{matrix}, \qquad if\ d = A^*$$

$$Val'^{i}(A^*) = Val(A^*), \qquad otherwise, for\ i = 0{,}1$$

Thus,

$$\langle M, Val\rangle \overset{MT,p(0)}{\rightarrow} \langle M', Val'^{0}\rangle$$

$$\langle M, Val\rangle \overset{MT,p(1)}{\rightarrow} \langle M', Val'^{1}\rangle$$

The probabilities p(0) and p(1) are computed from the equation $\langle\psi|P_i|\psi\rangle$ as mentioned in section 2.3.4. Additionally, from the same section we can define the next Val component in case token d is entangled with another token.

For a state s=$\begin{bmatrix} c00 \\ c01 \\ c10 \\ c11 \end{bmatrix}$. The probability p(i) for measuring a qubit in a specific state i, equals the square of the coefficients where that qubit is found in state i. (eg for the first qubit being in state $|0\rangle$ p(0)=$|c00^2| + |c01^2|$)

The new superposition is obtained by crossing out all those terms of $|\psi\rangle$ that are inconsistent with the outcome of the measurement (i.e., those whose first bit is 0). Of

course, the sum of the squared amplitudes is no longer 1, so we must renormalize to obtain a unit vector:

$$|\psi_{new}\rangle = \frac{\sum_{i=0}^{1} \sum_{j=0}^{1} |Cij^2|}{\sqrt{\sum_{i=0}^{1} \sum_{j=0}^{1} |Cij^2|}}$$

The resulting vector is split in its simplest form as a tensor product of simpler systems and assigned to the respective tokens' values as shown above with normal transitions.

Where c_i denotes the coefficient in the decomposition of qubit d's state, $|e_1\rangle$ is the state (eigenvector) that the qubit will collapse into.

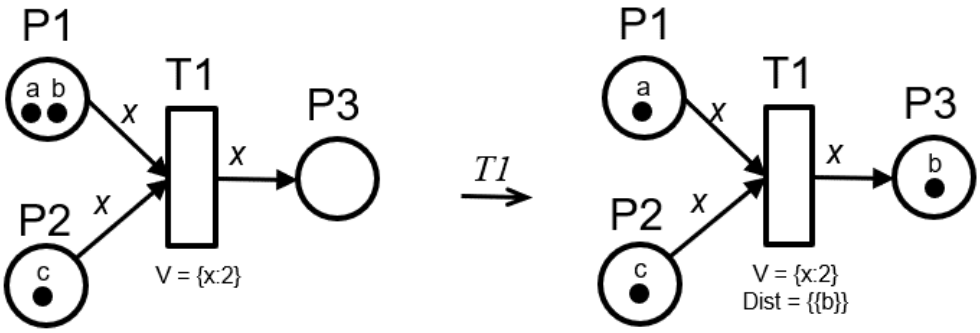


Figure 3.3: Forward execution in a QPN

Returning on a similar example. In the drawn QPN there are 3 places {P1,P2,P3}, one measurement transition T1 and the marking is M_0 ={1: {a,b}, 2:{c}, 3:{}}, Val_0 = {a: $\begin{bmatrix}0\\1\end{bmatrix}$,b $\begin{bmatrix}1\\0\end{bmatrix}$,c $\begin{bmatrix}1\\0\end{bmatrix}$}. The transition is enabled any token that can be placed in the x variable, is provided by the incoming arcs and satisfy the conditional part of the transition (none). The qubit entering the transition is either b or c, which is assigned to variable x. The observable's matrix is $\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$. And is split into eigenvectors $|e_0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$, $|e_1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$. With $\lambda_0 = 1\ and\ \lambda_1 = -1$. Using the process described before,

suppose we pick token c. The probability of collapsing into state $|0\rangle$ is 100%. When measured, the token collapses in $|0\rangle$ while we obtain the eigenvalue $1 = \lambda_0$. Thus M_1 ={{1: {a,b}, 2:{}, 3:{c}} Val_1 =-{{a: $\begin{bmatrix}0\\1\end{bmatrix}$,b $\begin{bmatrix}1\\0\end{bmatrix}$,c $\begin{bmatrix}1\\0\end{bmatrix}$}}.

3.3 Backtracking

As mentioned in Section 2.1.3, the most basic way of reversible execution, is *backtracking*. In this mode, the order of execution of transitions is stored, and the way of reverting back to a previous state, is by undoing each transition, from the most recently executed to the most formerly executed. In this thesis we will only explore backtracking as it is the most basic form of reversibility and should be the first one to be tested before introducing new methods of reversible execution.

Inspired by the insertion of the concept of reversibility in Petri nets by a recent publication [2], we are in position to adapt the mentioned techniques for QPNs.

Definition 3. A *Reversible Quantum Petri Net* (RQPN) is a tuple (*Q, H, E*) where:

1. *Q* is a *Quantum Petri Net.*
2. $H : T \rightarrow 2^{\mathbb{N}}$, where N is the number of executed transitions.
3. *Entanglement History*, $E: T \rightarrow 2^{A}$

You might have noticed that RQPNs do not differentiate the basic concepts of standard QPNs. That is almost the case! However, RQPNs employ the notion of history, this essentially assigns a memory to each transition of a RQPN. The initial history [2] of all states, $H(t) = \emptyset$, and this shows the transition t has not yet been executed. A history of $H(t) = \{k_0, k_1, \dots, k_m,\}$, signifies that the transition t has been executed as the k_0^{th}, k_1^{st}…, k_m^{th}, transition in the execution of the RQPN, and has not been reversed so far, $0 \leq k_i \leq n$, where *n* is the total amount of transitions executed. Graphically, the notion of history is depicted under each transition as an array. The history component is needed for the correct reversal to a previous state. Let us demonstrate why; consider a triple entanglement between tokens a, b, c, with a marking P1{a}P2{b, c}, P3{}, and two transitions T1(takes token a from P1 and outputs to P3), T2(takes tokens b, c from P2 and outputs to P3). Both transitions have no conditional expression. First, we fire T1 and the shared state of the three tokens is modified

accordingly. Then we fire T2, however T2 breaks the entanglement into tokens a and b, while c has its own state vector. In order to reverse to the original state, we need to reversibly execute transitions T2 and then T1. However, if we lack the H component then we might fire backwards the transitions in any order. If we execute T1 we arrive at a state that never occurred before (as only the shared state of a and b changes), which is problematic in larger systems.

The *Entanglement History* component is an array of token sets, which denote the tokens and if there were any extra entangled tokens with any of the input tokens that entered a transition. This is complementary to the History component of a transition, as for each entry in the history component, the respective tokens (plus any entangled ones) that entered the transition are stored. $E(t) = \{A_0, A_1, \dots, A_n\}, A_i \subseteq A$. The initial entanglement history of all transitions, is $E(t) = \emptyset$. Graphically, the notion of entanglement history is depicted under the history array. Do not let the name fool you, as this component not only serves the memorization of previously existing entanglements, and the decomposition of the resulting tensor product into the correct states. This concept is essential for the operation of RQPNs as we need to know exactly which tokens to pick from the output places (in case more than one matches for tokens exist in the distribution function of the transition) of a forward executed transition, in order to correctly backtrack to a previous state.

An RQPN operates in Forward execution mode just like a QPN, but the history and entanglement history components are added. Suppose that at any given time k a transition T is executed. Time is measured with the transitions executed. Initially k is 0 and increases by one with each transition executed. A_u is a set to token sets and reflects the tokens and entangled token groups that enter the transition.

For a transition T_e:

$$H'(T_e)\begin{cases} H(T_e) \cup \{k\}, & \text{if } T_e = T \\ H(T_e), & \text{otherwise} \end{cases}$$

$$E'(T_e)\begin{cases} E(T_e) \cup Dist(v), & \text{if } t = T, and\ v \in V \\ E(T_e), & \text{otherwise} \end{cases}$$

Contrary to QPNs, the global state of a RQPN is denoted by as $\langle M, Val, H, E \rangle$. As order of execution of transitions and the executed in contribute to the factor of

uniqueness and distinguishability of global states of an RQPN. Something that regular QPNs did not capture.

A transition is said to be *bt-enabled* (backtracking-enabled), with the condition that it was the last forward-executed transition. Thus, if a transition contains the highest value as a finishing element in its history, then it is bt-enabled. We will build upon this notion soon.

A transition executed in reverse, should be able to

1. Take the input (outgoing arcs), compute the tensor product of the tokens if applicable,
2. Apply the *reverse matrix* that undoes the transition matrix that was executed in forward-execution mode.
3. Place the tokens in the places denoted by the incoming arcs, and update the value, history and marking components.

The first step is quite simple to define, as the notion of computing the tensor product of the arcs supplying the input (outgoing arcs in this scenario), as well as covering the notion of any entangled tokens is covered in subchapters 3.1-3.2.

The second step requires us to compute a reverse matrix, suppose *R* that can undo the effects of the original transition matrix, suppose *M*, of a transition *T*. Also consider the shared state vector before the forward-execution of T, S and after *S'*. Simply put, we want:

$$S \cdot M = S' and\ S' \cdot R = S$$

if we substitute, we need: $S \cdot M \cdot R = S$

The only matrix that can substitute $M \cdot R$ in the equation above is the identity Matrix I!

$$M \cdot R = I$$

$$S \cdot I = S$$

The question that we now need to answer is, how can we compute R from the equation $M \cdot R = I$? The answer is that R is the inverse of M.

$$R = M^{-1}$$

All the matrices contained in quantum gates are reversible, thus Transitions in RQPNs are reversible! This is derived from the fact that since the transition matrices of quantum gates are unitary, then their inverse is their conjugative transpose.

$$M^{-1} = M^{\dagger}$$

The only catch is that Measurements or spontaneous decoherence, are irreversible actions.

The third step is again simple to define as this operation was described in the previous subchapter. After computing the shared state of the input qubits and calculating the dot product with the inverse of the transition matrix, we now need to separate their state as much as possible into tensor products of states of smaller quantum systems. The last element of the Transition's history is also removed, as since the transition was reversed, the action of the transition being the most recently forward executed one is not necessarily valid anymore.

In an arbitrary state $\langle M, Val, H, E\rangle$, a transition is *bt-enabled* if:

a. The transition was the last forward-executed transition.
b. The transition belongs to T.

These conditions are enough, as all the gates are included, and measurements are excluded. Thus, if a transition contains the highest value as a finishing element in its history, and it is not a MT then it is bt-enabled.

Let us now formally define the backtracking algorithm:

If Transition T bt-enabled in $\langle M', Val', H', E'\rangle$, then, the next state $\langle M, Val, H, E\rangle$, is computed with the backtracking the transition T;$\langle M', Val', H', E'\rangle \overset{T}{\rightsquigarrow} \langle M, Val, H, E\rangle$. A transition is composed of matrix t, conditional expression Φ,a distribution of tokens Dist and variable set V.

For all places x and tokens a:

If T is bt-enabled in $\langle M', Val', H', E'\rangle$, then, for all places x and tokens a:

$$M(x) = \begin{cases} M'(x) \cup Dist(v), & \text{if } x \in \bullet T, v \in F(x, T) \\ M'(x) - Dist(v), & \text{if } x \in T \bullet, v \in F(T, x) \\ M'(x), & \text{otherwise} \end{cases}$$

and

$$Val(A_{s1}), \dots Val(A_{sm})$$
$$= \begin{cases} S_1, \dots, S_m \ \ where\ S_1 \otimes \dots \otimes S_m = \left(Val(A_1') \otimes Val(A_2') \otimes \dots \otimes Val(A_n')\right) \cdot T^{\dagger}, \\ i \in [1,n], m = max, a_1 \dots a_k = \cup A_i, S_i \in \mathbb{C}^{2^{l_i}}, A_{si} = a_{\sum_{p=1}^{i} l_p - l_i} \dots a_{\sum_{p=1}^{i} l_p}, \quad \text{if Dist(v)} \to A_i' \\ Val(A_{s1}), \qquad \text{m} = 1, \text{otherwise} \end{cases}$$

and

$$H(t') \begin{cases} H'(t') - \{k\}, & \text{if } t' = T, \text{k} = \max(\text{H(T)}) \\ H'(t'), & \text{otherwise} \end{cases}$$

and

$$E(t) \begin{cases} E'^{(t)} - Dist(v), & \text{if } t = T, and\ v \in V \\ E(t), & \text{otherwise} \end{cases}$$

Additionally, if Val(a_0, a_1, $a_2 \dots a_{n-1}$) $\otimes$ Val(a_n) = Val(a_0, a_1, $a_2 \dots a_n$). then Val = (Val - Val(a_0, a_1, $a_2 \dots a_n$)) $\cup$ Val(a_0, a_1, $a_2 \dots a_{n-1}$) $\cup$ Val(a_n)).

The above implies that the tokens are kept as simple and independent as possible. All tokens are written with the smallest possible state vector, with only inseparable token groups being described by a shared state vector, thus being entangled.

In case of entangled qubits, the entangled qubit may appear first or last in the conditional part of a transition. The definition is the same if all the entangled tokens (from the same entangled group) appear together at the beginning or the end of the conditional part of a transition. One or more entangled tokens may be left out of the transition, however, the new Val component is calculated as ((a1,a2…an-k) $\otimes$ b $\otimes$ c $\otimes$ ... $\otimes$ n) $\cdot$ ($I^k \otimes T^{\dagger}$) if the entangled token is found at the beginning, or (n $\otimes$ b $\otimes$ c $\otimes$ ... $\otimes$ ((a1, a2 ... an $-$ k)) $\cdot$ ($T^{\dagger} \otimes I^k$), if it is found at the end of the transition.

We will now graphically denote the evolution of a RQPN with the following arrows:

$\rightsquigarrow_{T_i}$ Denotes that the RQPN at the pointing end, is the RQPN on the opposite end, but with the transition T_i reversed.

$\rightarrow_{T_i}$ Denotes that the RQPN at the pointing end, is the RQPN on the opposite end, but with the transition T_i, forward executed.

Let us now proceed to an example:

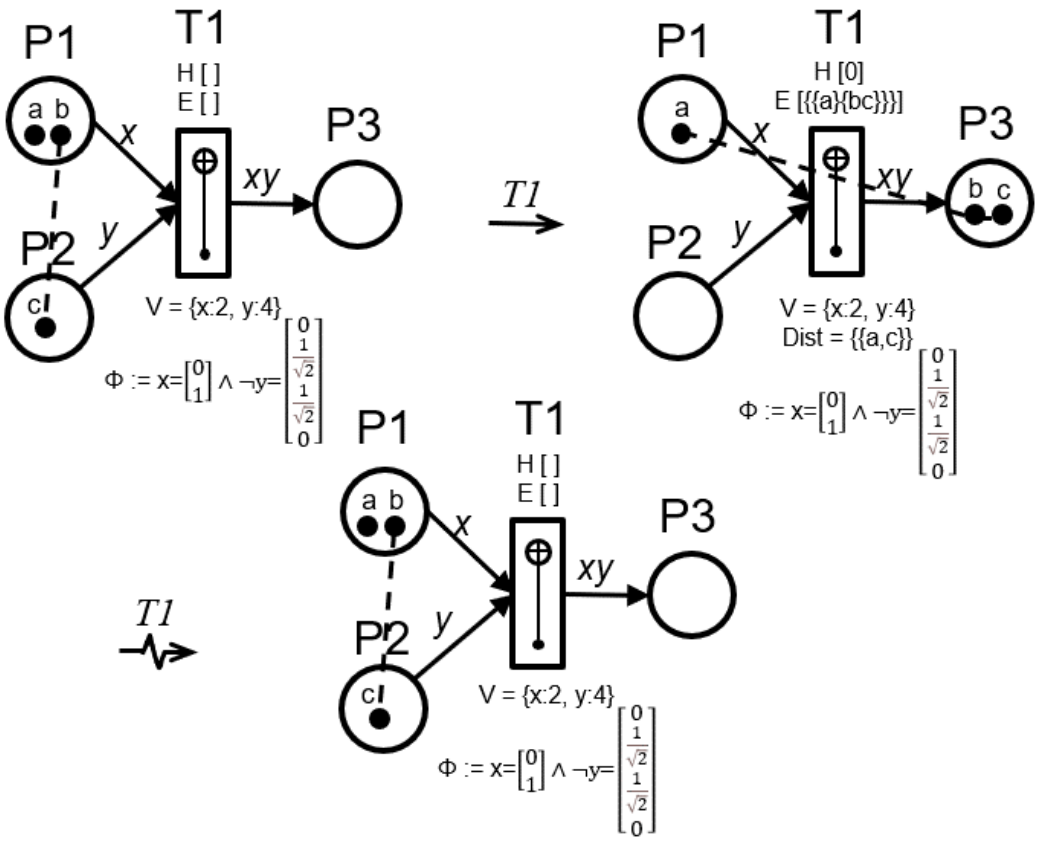


Figure 3.4: Forward execution and Backtracking in a RQPN.

In figure 3.4 we see a RPQT with 3 places, namely P1, P2, P3. One transition T1 with the CNOT matrix (control qubit second) $\{\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}$, the conditional part is shown on the figure. M_0 = {P1:{a,b},P2:{c},P3{}}. Val_0={a$\begin{bmatrix}1\\0\end{bmatrix}$,bc$\begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix}$}

The transition is enabled as the token x=a has the required state, and the token y=c does not have the restricted state (entangled with b).

First, the transition is fired in forward execution mode.

The shared state vector after the transition computed by:

$$(|a\rangle \otimes |cb\rangle) \cdot \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

The resulting state cannot be written as a tensor product of smaller quantum systems. Thus, the three tokens a, b, c are now entangled.

M_1 = {P1:{b},P2:{},P3{a,c}}. Val_1={abc $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$}, $H_1(T_1)$ = [0], $E_1(T_1)$ = [{{a},{b,c}}]

Next, we backtrack before the execution of the transition T1.

The shared state vector after the transition computed by:

$$(|acb\rangle) \cdot \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Notice that the mentioned matrix is the inverse of the CNOT gate, which is the same matrix!

The resulting shared state has two possible simplifications $\begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix}$ and $\begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix}$. We know that only the first one is valid. Since the previous entanglement is saved in the E component. The values of 0, {{a},{b,c}} are popped from the transition's history and entanglement history components. The final global state is the same as the initial!

Chapter 4

Applications of Quantum Petri nets

Contents

In this chapter, practical applications of the Quantum Petri net model variations that were presented in Chapter 3 are going to be presented and analyzed.

Specifically, in subchapter 4.1, a detailed analysis of a classical quantum algorithm, the Teleportation protocol is going to be encoded into QPNs. In the same chapter, an overview of the No cloning theorem is going to be presented.

In subchapter 4.2, the quantum version of the Discrete Fourier Transformation, Quantum Fourier Transformation is defined and then encoded into our proposed models.

4.1 Quantum Teleportation protocol

The Quantum Teleportation Protocol [3][5][6][7] (QTP) describes the process of transferring quantum information (qubit states) from a sender to a receiver that is located some distance away. Quantum teleportation has been already implemented and tested in real quantum computers. The longest successful teleportation attempt is at 1400 km [4].

The protocol states the following. There are two people, Alice (sender) and Bob (receiver). Alice wants to send the state of a qubit $|\psi\rangle = a|0\rangle + b|1\rangle$. The information about a and b are transferred via a physical connection and a pre-existing entanglement between the two hosts. The main takeaway is that the qubit that carries the original state at Alice's end, is collapsed to another state, while that state is "transferred", or better, teleported to another qubit at Bob's end. This is because of the No cloning theorem,

oversimplified, it states that it is impossible to copy the state of any qubit and create an exact duplicate. At the end of the description of the protocol, the protocol will be discussed and proved with more detail, in section 4.1.1.

Let us see the steps for the quantum teleportation protocol:

1. First, an entangled qubit pair must be created, specifically a pair that is in one of the four Bell states. In this thesis we are going to use the Bell state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_\alpha \otimes |0\rangle_b + 1\rangle_\alpha \otimes |1\rangle_b)$.

   For this purpose, we consider a third party, Telamon, who creates the entangled pair in the mentioned Bell state and gives one qubit to Alice and the other to Bob. The entanglement is created by passing the first qubit through a Hadamard transformation matrix. Then the CNOT is applied to the qubits, with control qubit the one that we just applied the Hadamard transformation to. After that, the two qubits are entangled in the state $|\Phi^+\rangle$. Suppose for clarification reasons, that the entangled qubits are called q_1 and q_2, q_1 is given to Alice and q_2 is given to Bob.
2. Now Alice has the qubit to be teleported q_0 in state $|\psi\rangle$ and the qubit q_1 that is entangled with q_2 at Bob's machine. Alice performs the following actions: CNOT to q_1 and q_0, with q_0 being the control qubit. Then a Hadamard transformation is applied to q_0.
3. Alice measures both q_0 and q_1 and stores the result in two conventional bits. These two bits, carrying the result of the measurement (collapsed basis state) for each qubit are sent to Bob through a normal communication channel (e.g fiber optics).
4. On the other end, Bob now has the following, the measurement results for q_0, q_1 and the intact qubit q_2. Now, q_2 was entangled with q_1, which was severed after Alice measured q_1, this means that q_2 is now in a non-entangled state $|\varphi\rangle = c|0\rangle + d|1\rangle$. Bob now applies the following matrices on q_2 based on the values received from Alice. First, if the measurement of q_1 is 1 then the X Pauli matrix is applied onto q_2. Then if the measurement of q_0 is 1 then the Z Pauli matrix is applied onto q_2. Now, q_2 contains the original state of q_0, $|\psi\rangle$ and Alice does not have it anymore.

The algorithm described above can be verified with the following operations:

Consider qubits a : $\begin{bmatrix}\alpha\\ \beta\end{bmatrix}$, b, c, a is the qubit that is on Alice's end and needs to be teleported, b and c are the entangled qubits in the Bell state $|\Phi^+\rangle$, b is on Alice's machine, whereas c is located on Bob's machine. (The order a, b, c will be used for the calculations below).

For simplicity reasons, the creation of the Bell state $|\Phi^+\rangle$ will not be presented.

S_i is the shared state of the system (qubits a, b, c) at any given time, i.

First, the CNOT is applied on b with a being the control qubit.

$$S_1 = (\mathrm{a} \otimes bc) \cdot (CNOT \otimes I^2) = \left(\begin{bmatrix}a\\ \beta\end{bmatrix} \otimes \begin{bmatrix}\frac{1}{\sqrt{2}}\\ 0\\ 0\\ \frac{1}{\sqrt{2}}\end{bmatrix} \right) \cdot \left(\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \right)$$

$$S_1 = \begin{bmatrix}\frac{1}{\sqrt{2}}\alpha\\ 0\\ 0\\ \frac{1}{\sqrt{2}}\alpha\\ 0\\ \frac{1}{\sqrt{2}}\beta\\ \frac{1}{\sqrt{2}}\beta\\ 0\end{bmatrix}$$

You may notice that this state may or may not be entangled, depending on the values of α and β, we will continue as if the state is entangled, as it also covers the latter.

Then the Hadamard matrix is applied on qubit a. Since a is entangled with b and c, we need to compute the tensor product of the Hadamard gate with the identity matrix as follows:

$$S_2 = (\mathrm{a}bc) \cdot (H \otimes I^4) = \begin{bmatrix} \frac{1}{\sqrt{2}}\alpha \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}}\alpha \\ 0 \\ \frac{1}{\sqrt{2}}\beta \\ \frac{1}{\sqrt{2}}\beta \\ 0 \end{bmatrix} \cdot \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

$$S_2 = \begin{bmatrix} \frac{1}{2}\alpha \\ \frac{1}{2}\beta \\ \frac{1}{2}\beta \\ \frac{1}{2}\alpha \\ \frac{1}{2}\alpha \\ -\frac{1}{2}\beta \\ -\frac{1}{2}\beta \\ \frac{1}{2}\alpha \end{bmatrix} \begin{matrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{matrix}$$

Now we have the shared state S_2, you can notice that the binary equivalent of the index's position is written next to each state. This is helpful as is shows which values correspond to the qubits a, b, c, once they collapse to a basis state.

In the next two steps, there comes the measurement of a and b and the application of Pauli matrices X and/or Y depending on their values:

There are four possible outcomes:

1. a = 0 and b = 0

   The shared state is now:

$$S_3 = \begin{bmatrix} \alpha \\ \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

   Since no matrices need to be applied, c has the value $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, which is the original value of qubit a!

2. a = 0 and b = 1

   The shared state is now:

$$S_3 = \begin{bmatrix} 0 \\ 0 \\ \beta \\ \alpha \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

   Qubit c now has the value $\begin{bmatrix} \beta \\ \alpha \end{bmatrix}$. The algorithm states that the Pauli matrix X is applied when b is 1. Thus, c is transformed:

   $c = c \cdot X = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, which is the original value of qubit a!

3. a = 1 and b = 0

   The shared state is now:

$$S_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \alpha \\ -\beta \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \alpha \\ -\beta \end{bmatrix}$$

   Qubit c now has the value $\begin{bmatrix} \alpha \\ -\beta \end{bmatrix}$ The algorithm states that the Pauli matrix Z is applied when a is 1. Thus, c is transformed:

   $c = c \cdot Z = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, which is the original value of qubit a!

4. a = 1 and b = 1

   The shared state is now:

$$S_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\beta \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -\beta \\ \alpha \end{bmatrix}$$

   Qubit c now has the value $\begin{bmatrix} -\beta \\ \alpha \end{bmatrix}$ The algorithm states that the Pauli matrix X is applied when b is 1, and Pauli matrix Z is applied when a is 1. Thus, c is transformed:

   $c = c \cdot X \cdot Z = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, which is the original value of qubit a!

We can now conclude that the protocol is valid and can be encoded in the following RQPN (step 1, creation of entanglement and distribution is omitted) :

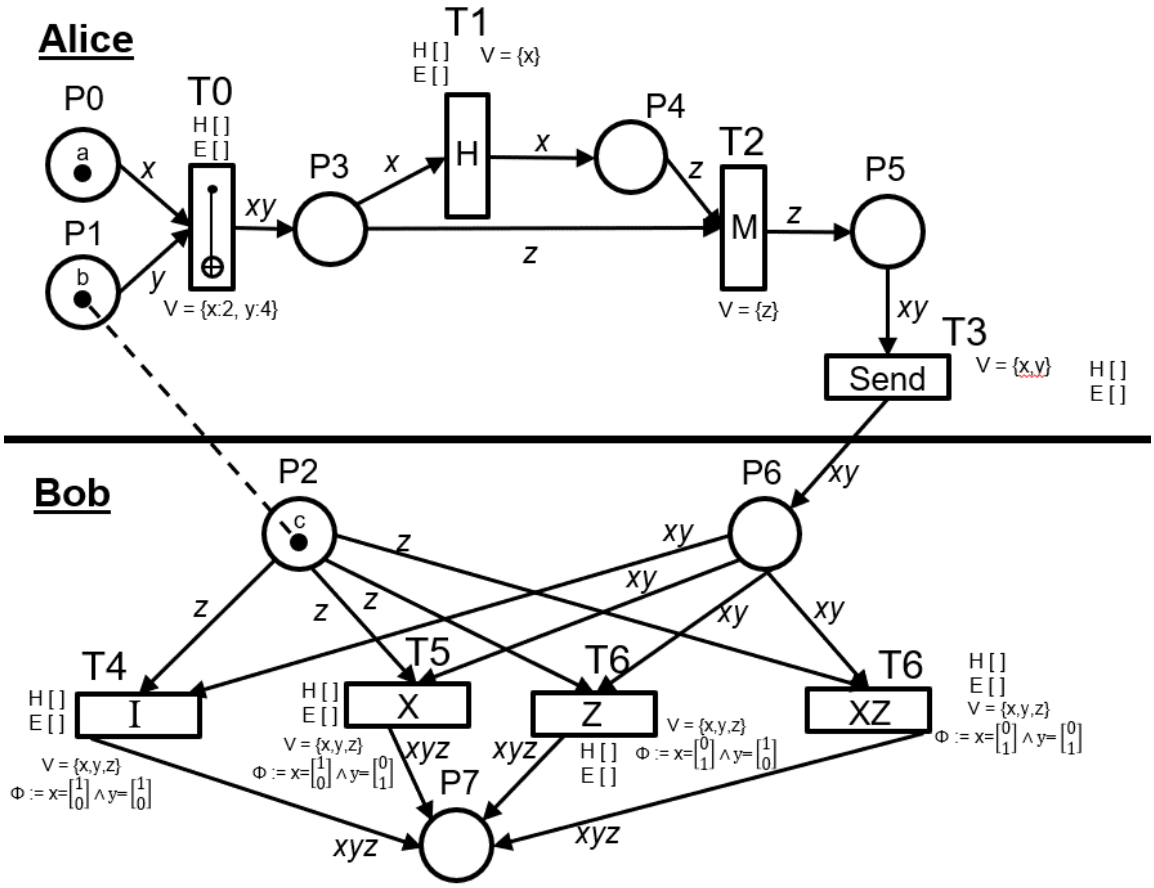


Figure 4.1: The RQPN for quantum teleportation of one qubit.

The model is composed of 8 places and 8 transitions. There is also a simple separation between Alice's and Bob's parts. Telamon's part was omitted, as it is not significant.

Each transition reflects a Hermitian operation on some of the tokens, as defined by the algorithm. The places are a reflection of our progress in the QTP.

Notice that only 1 transition can be enabled at any given time on any possible execution. Thus, the process of this protocol is deterministic.

We may reverse the transitions T1 and T0 as long as T2 has not been fired. After that we may reverse T3-T7.

The quantum circuit model can be seen in figure 4.2. The simulation of the QTP in quantum circuits can be found at Quirk[11].

A sample execution and some explanation can be found at Appendix B.

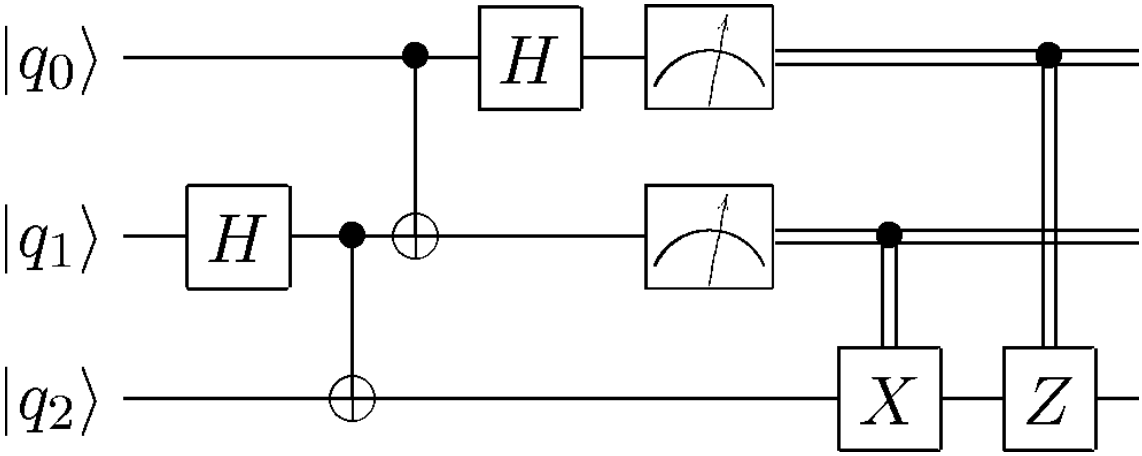


Figure 4.2: The quantum circuit model for quantum teleportation of one qubit.[10]

4.1.1 No cloning theorem

The quantum teleportation protocol is based on the fact that quantum states are impossible to duplicate perfectly [8]. A perfect transferal of a state from a qubit to another will result in the destruction (collapse) of the original qubit.

This can be proven [9]:

Assume the existence of a unitary operator U_{cl} that acts on two qubits, which clones the state of the first onto the second one.

Specifically:

$U_{cl}\,|\psi\rangle_A|e\rangle_B = |\psi\rangle_A|\psi\rangle_B = (a|0\rangle_A + b|1\rangle_A)(a|0\rangle_B + b|1\rangle_B) =$
$a^2|0\rangle_A|0\rangle_B + ab|0\rangle_A|1\rangle_B + ba|1\rangle_A|0\rangle_B + b^2|1\rangle_A|1\rangle_B$

This should be true for basis states as well:

$$U_{cl}\,|0\rangle_A|e\rangle_B = |0\rangle_A|0\rangle_B$$

$$U_{cl}\,|1\rangle_A|e\rangle_B = |1\rangle_A|1\rangle_B$$

Then the linearity of U_{cl} implies

$U_{cl}|\psi\rangle_A|e\rangle_B = U_{cl}(a|0\rangle_A + b|1\rangle_A)|e\rangle_B = a|0\rangle_A|0\rangle_B + b|1\rangle_A|1\rangle_B$
$\neq a^2|0\rangle_A|0\rangle_B + ab|0\rangle_A|1\rangle_B + ba|1\rangle_A|0\rangle_B + b^2|1\rangle_A|1\rangle_B\,|\}.$

Thus $U_{cl}|\psi\rangle_A|e\rangle_B$ is generally not equal to $|\psi\rangle_A|\psi\rangle_B$. Thus, U_{cl} cannot be used as a general copier.

The no cloning theorem wan initially considered to be the downfall of quantum computing in general, as this prohibited the use of classical error correction techniques on qubits. However, there are new techniques actively being researched for quantum error correction.

On the bright side, this theorem is an essential element in quantum cryptography, as eavesdroppers cannot create copies of any quantum cryptographic keys that are being transferred on a shared means of communication.

4.2 Quantum Fourier Transformation

The *Quantum Fourier Transformation* (QFT) [12] is the reflection of the *Discrete* Fourier *Transformation* (DFT) in classical computing. QFT has many applications in many quantum algorithms, such as Shor's factoring algorithm, as well as topics like differential equation analysis and solution of linear systems. As an extension of DFT it is applicable in many areas, such as: signal processing, machine learning, and spectral analysis [13][14].

Let us first take a general overview of DFT, before proceeding to QFT. The DFT is a process of converting equally spaced samples in the time domain to the frequency domain. For simplicity reasons, consider the frequency domain as a representation of data from a combination of harmonically related sinusoids, that refer to the frequency of occurrence of a feature rather than the time of its occurrence. This is particularly useful for signal and image processing, and data analysis in general [15].

The Quantum Fourier Transformation performs what we call *base transformation*. So far in this thesis we only talked about the basis states $|0\rangle$ and $|1\rangle$, which are the basis states of the computational basis, or on put in another way the two polarities of the Bloch sphere in Z-axis. If now we encode those two basis states to the polarities of the X-axis of the Bloch sphere, we now have $|+\rangle$ and $|-\rangle$ as our basis states. $|+\rangle$ and $|-\rangle$ are translated into $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ respectively.

A random single-qubit state $|\psi\rangle = a|0\rangle + b|1\rangle$ can be converted to the Fourier (or sometimes Hadamard) basis ($|+\rangle$ and $|-\rangle$) by replacing the basis state vectors. We will denote a state in Fourier basis with a tilde(~) on top.

$$|\psi\rangle = a|0\rangle + b|1\rangle$$

$$|\tilde{\psi}\rangle = a|+\rangle + b|-\rangle = a\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + b\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$|\tilde{\psi}\rangle = \frac{1}{\sqrt{2}}(a+b)|0\rangle + \frac{1}{\sqrt{2}}(a-b)|1\rangle$$

At this point, a certain gate that performs this transformation for a single qubit, might come to mind… This is no other than the Hadamard gate and respective matrix $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$. It is a valid thought to consider applying the Hadamard transformation to all the qubits that we want to apply QFT to, and it would be a very simple solution to the basis transformation problem. This is correct, but there is a catch! If we indeed have n qubits and apply H at each qubit, we will now have n qubits in Fourier basis. The QFT however treats the input qubits as a group and performs the basis transformation at their composite state.

So, we have the solution at converting the basis of a single qubit into Fourier basis with the Hadamard transformation, the problem now is "connecting" the involved qubits into a shared state. This is done by using the controlled phase shift gate and related matrix. CP(θ) = $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{i\theta}\end{bmatrix}$, or CP(k) = $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{\frac{2\pi i}{2^k}}\end{bmatrix}$. Which rotates the controlled qubit by θ degrees around the Z axis, only if the input state is $|11\rangle$.

Let us now take a break and see the Bloch spheres of various composite states in conventional and Fourier basis, in figure 4.3.

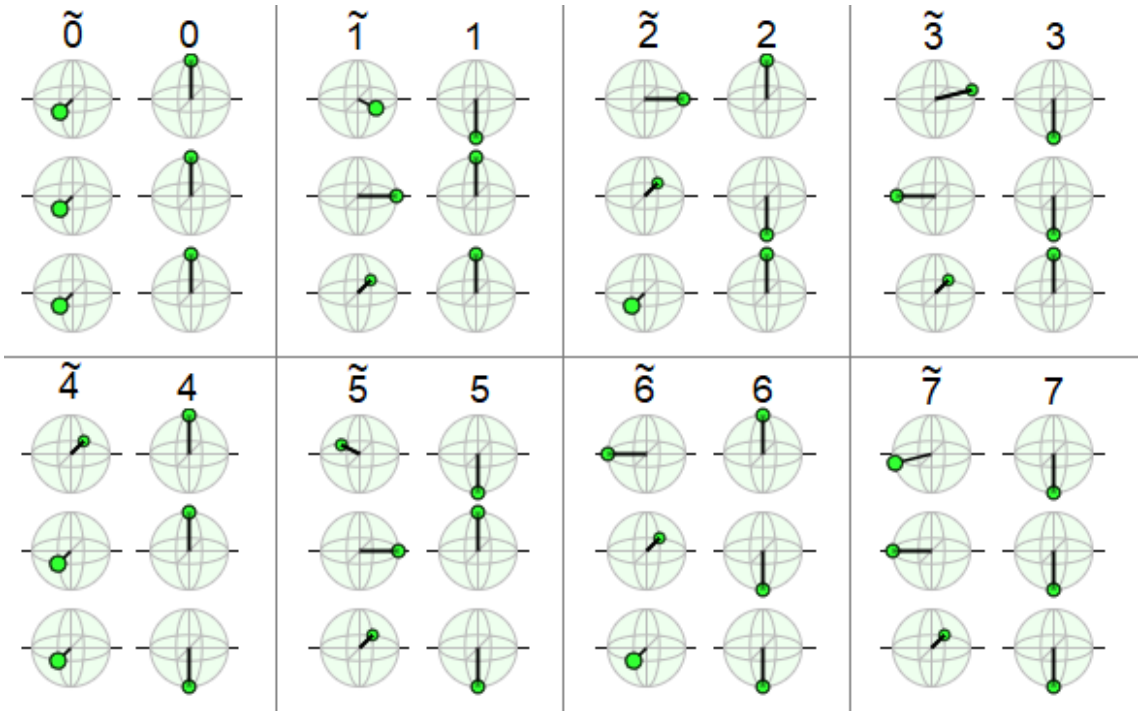


Figure 4.3: Bloch spheres of composite states 0-7, in conventional and Fourier basis. Qubits from top to bottom with increasing significance.

Notice that a pattern forms here. In the conventional basis, the least significant qubit changes basis with every (2^0) increment, the next one after two (2^1) increments and the most significant after four (2^2) increments. They remain on the same basis state otherwise.

In Fourier basis, the first thing is that the changes occur between the basis states $|+\rangle$ and $|-\rangle$, on the polarities of the Z axis. The most significant qubit iterates between these two basis states on every (2^0) iteration, or better, it changes phase of $\hat{\theta} = \pi$ with every increment. The next qubit switches basis states on every two (2^1) iterations, it has a phase change of $\hat{\theta} = \frac{\pi}{2}$ on every increment. Similarly, the least significant qubit switches basis states on every four (2^2) iterations, it has a phase change of $\hat{\theta} = \frac{\pi}{4}$ on every increment. This notion can be expanded to include more qubits.

The main differences between this and the conventual basis are:

1. The qubits iterate between the basis states through subdivisions of them via phase shifts.
2. The frequency with which each qubit changes between the basis states is reversed significance wise.

Now this can be encoded in a respective formula.

The Quantum Fourier Transformation converts a quantum state $|\psi\rangle = \sum_{i=0}^{n-1} \psi_i |i\rangle$, to a quantum state $|x\rangle = \sum_{i=0}^{n-1} x_i |i\rangle$. Where:

$$x_i = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \psi_j \omega_n^{ji}$$

$$i = 0, 1, 2, \dots, n-1, \qquad \omega_n^{jk} = e^{2\pi i \frac{jk}{n}}, \qquad and\ n = 2^N, N\ is\ the\ number\ of\ qubits$$

Similarly, this can be encoded in quantum circuit shown in figure 4.4[14].

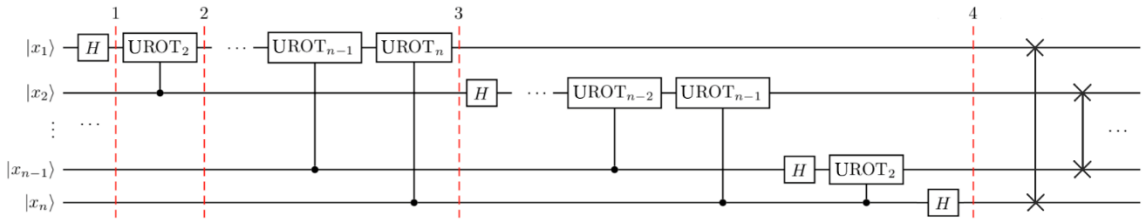


Figure 4.4: Quantum circuit for QFT.

Where the controlled UROT gate corresponds to the CP(k) matrix defined above. The connected X shaped gates at the end denote the swap gate and responding matrix $SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, which exchanges the states of two qubits.

This may be translated into an RQPN, for simplicity reasons the model will be for a three qubit QFT algorithm.

The RQPN has M_0= {P0:{a, b, c}, P1:{} P2:{}, P3:{}, P4:{}, P5:{}}

Val_0={a: $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, b: $\begin{bmatrix} \gamma \\ \delta \end{bmatrix}$, ψ: $\begin{bmatrix} \varepsilon \\ \zeta \end{bmatrix}$}, where α, β, γ, δ, ε, ζ are variables that correspond the respective quantum state coefficients. The Histories and Entanglement histories of the transitions are depicted.

The matrices of the transitions are the following:

$$\text{T0, T2, T5} = \text{H} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad \text{T3} = \text{CP}_3 = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{\frac{\pi i}{4}}\end{bmatrix}$$

$$\text{T1, T4} = \text{CP}_2 = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{\frac{\pi i}{2}}\end{bmatrix} \qquad \text{T6} = \text{SWAP} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

As you can see, the depicted RQPN directly translates the mentioned algorithm. Specifically, each transition is translated to the utilization of a quantum gate, encoding the respective transition matrix. Each place denotes the progress of each token throughout the execution of the algorithm. Arcs connect the two and ensure that each token progresses smoothly in the QRPT.

0Notice the lack of MTs. This firstly infers that there are no measurements on any qubit, and as a result no collapses to any conventional basis state. Secondly, we conclude that the whole model is reversible, as all the inverse matrices for the used transitions, exist.

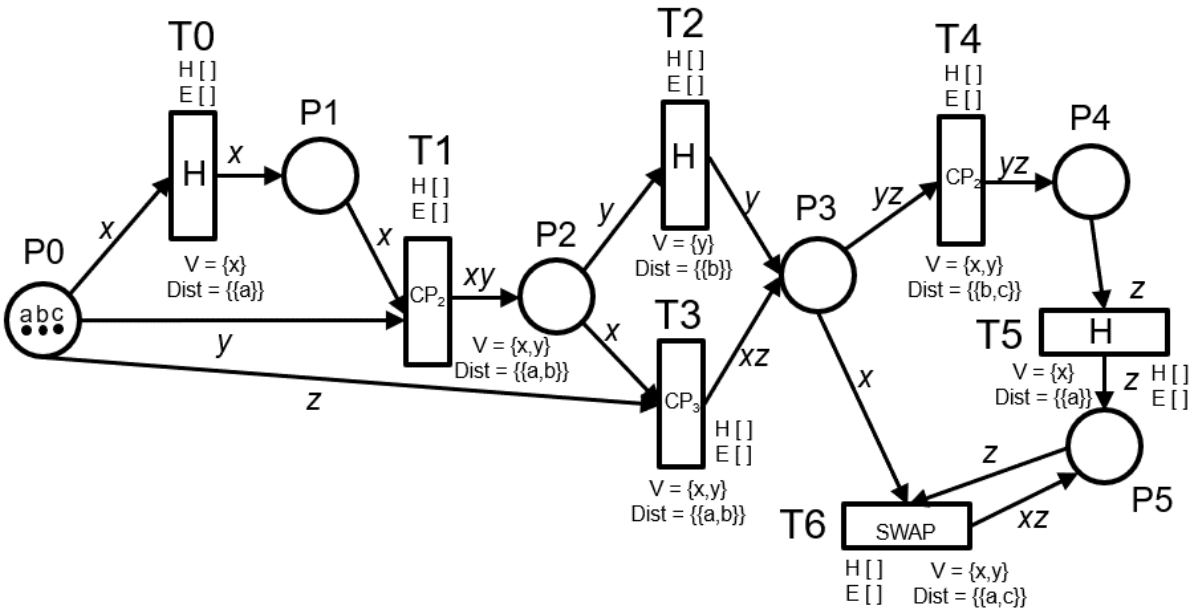


Figure 4.4: RQPN for 3 qubit QFT.

Chapter 5

Conclusion

Contents

5.1 Summary

This Bachelor thesis is focused on introducing aspects of quantum computing in Petri nets. After careful study of both the topics of Reversible Petri nets and quantum circuits, the model of *Quantum Petri Nets* (QPNs) and its extension *Reversible Quantum Petri Nets* (RQPNs) were introduced and defined.

Briefly, a QPN is a 7 tuple (*A, P, T, MT, F, M, Val)*. It is built around the concept of tokens (A), places (P), arcs (F), transitions (T) and capture the concept of marking (M), just like conventional Petri nets. Additionally, the value (Val) component was added, as tokens (which denote qubits) can be found in various states and we need to have a way of knowing the state each token has. Measurement Transitions(MTs) encode the operation of measuring qubit(s) and were separated from normal transitions due to their behavior and irreversibility. Similarly, a RQPN is a 9 tuple{*A, P, T, MT, F, M, Val, H, E*}. The history(H) and entanglement history (E) were added as complementary components for transitions so backtracking, as the main form of reversible execution would be made possible. The rest of the components are the same as in QPNs.

Quantum circuits and other quantum computing models are unable to capture the concepts of parallel, distributed, and reversible execution. The proposed models of QPNs and RQPNs however, capture and encode all these notions. While quantum computing and its foundation, quantum mechanics are areas that are still being actively researched, my contribution to them, with these models will unlock new possibilities of visualizing and representing quantum computing processes.

5.2 Evaluation

Just like most Petri net variations, the quantum models that we have defined in this chapter are excellent at representing distributed computation. While distributed and parallel quantum computing might not yet be physically possible to produce, QPNs and RQPNs are great at encoding and visualizing the behavior of such a system, encoding much of its functionality in their visual representation. They can be used as a preliminary model for these systems as well.

The proposed models are perfect at simulating quantum circuits. The advantages of using this graphical representation are numerous. First, cyclical structures may be built, simulating iterative behavior. Additionally, the entanglement of qubits is visible and easily distinguishable. The notion of reversibility and its requirements are clear and may be used for an algorithm operating with quantum circuits.

The use of a finite number of tokens, is perfect for the creation of algorithms for quantum computers with a small number of qubits. Thus, the presented models are also applicable at the time of writing of this thesis as well!

5.3 Challenges

Throughout the whole process of researching and writing this thesis, I encountered several challenges. Many of them arose while trying to create the proposed models, while others were encountered at other phases. Most of the difficulties have been overcome, while others are left as tasks to be revisited in the future.

The first phase of this thesis was an intense period of studying and researching all the topics that are mentioned in Chapter 2. Petri nets, Reversible Petri nets, linear algebra, quantum computing and mechanics, along with all other correlated topics were studied through books, research articles and useful wikis. I had little to no knowledge on the mentioned areas, thus this introductory step was crucial and at times hard to do, as these topics are complex and often difficult to understand. I needed to practice a lot of operations in linear algebra and then execute various quantum circuits step by step, calculating the states of the involved qubits though after each gate.

A time-consuming difficulty was the introduction of conditional components into the QPNs. Initially the conditions and ordering of the input tokens into the transitions.

This idea was later scrapped, as too much information would have to be written on transitions, and reordering would have been needed to take place at transition level too. After much thought, the conditions were implemented at the conditional part of the transitions, which also define the order of input of the qubits, which shows how the composite state is going to be calculated before being multiplied the transition's matrix.

Another challenge was how the (possibly entangled) states of the qubits would be stored. The value component was the initial idea and was indeed kept. However, its structure and the storage of entangled qubits was topic that changed often. At a point, I thought about introducing another component (namely entanglements) that would store token groups, which would later denote which tokens were entangled. This was not implemented this way as it would actually be inferred from the current structure of the value component.

Lastly, an issue that is inferred from the previous one, was the idea of entangled qubits being represented with their own independent states, which might sound impossible, but math proves it otherwise. This idea was scrapped as it added a lot of extra complexity into the model, however it was an important breakthrough on this thesis, and an overview of it is shown in Appendix A.

5.4 Future Work

The models of Quantum Petri nets and Reversible Petri nets, as they are introduced in this thesis are great for simulating many quantum computing processes, however, there is a number of possible extensions, or parts that would need a little more work.

Firstly, these models, consider that all possible quantum operations (thus transition matrices) can be broken down and performed by two qubit quantum gates. This assumption is valid and can be proven[17]. This was done so that the entanglement would be preserved correctly throughout the firing of transitions. Even though the models have no practical limitation in the quantum operations they can simulate, it would be a great addition to have a simple way of correctly emulating more than two qubit gates.

Secondly, only one basic form of reversible execution was analyzed. Specifically, backtracking is analyzed in section 3.3. As mentioned by this paper [2], there are many

forms of reversible execution and are applied to conventional Petri nets. Some forms of reversibility include Causal-order and out-of-causal-order. They would be important expansions of the proposed models. Research so far provides evidence for the operation of causal reversibility in quantum computation [31], additionally to backtracking.

Thirdly, since the action of decomposing tensor products is not an easily computed calculation, we could propose the existence of a state library that contains the tensor products of all possible states. A lookup on this library with the resulting shared product state vector after every calculation with a two-qubit gate, for the assignment of the new states over the input qubits.

Fourthly, even though the variables combined with the conditional expression on the transitions provide a lot of flexibility, there are some limitations. For instance, we cannot define a different conditional expression for a different set of tokens assigned to the variables, (a different distribution). A possible expansion of the models would solve this issue.

References

[1] Murata, Tadao. "Petri nets: Properties, analysis and applications." Proceedings of the IEEE 77.4 (1989): 541-580.

[2] Philippou, Anna, and Kyriaki Psara. "Reversible computation in Petri nets." International Conference on Reversible Computation. p.5-20 Springer, 2018.

[3] Yanofsky, N. S., & Mannucci, M. A. (2008). Quantum Computing for Computer Scientists. Cambridge University Press.

[4] China launches world's first quantum science satellite. (2016, August 16). Physics World. https://physicsworld.com/a/china-launches-worlds-first-quantum-science-satellite/

[5] Jupyter Book Community. Quantum Teleportation. Qiskit. https://qiskit.org/textbook/ch-algorithms/teleportation.html

[6] Teleportation Protocol Quantiki. (2015, October 26). Quantiki. https://quantiki.org/wiki/teleportation-protocol

[7] Bennett, C. H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., & Wootters, W. K. (1993). Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. Physical Review Letters, 70(13), 1895–1899. https://doi.org/10.1103/physrevlett.70.1895

[8] Berkeley university C/CS/Phys C191. (2005). No Cloning, Teleportation [Slides]. Berkeley.Edu. https://inst.eecs.berkeley.edu/~cs191/fa05/lectures/lecture6_fa05.pdf

[9] The no-cloning theorem | Quantiki. (2015, October 26). Quantiki. https://www.quantiki.org/wiki/no-cloning-theorem

[10] Building a matrix corresponding to the teleportation circuit. (2018, December 13). Quantum Computing Stack Exchange. https://quantumcomputing.stackexchange.com/questions/4917/building-a-matrix-corresponding-to-the-teleportation-circuit

[11] Quirk: Quantum Circuit Simulator for QTP. Quirk. https://bit.ly/3wGVWs7

[12] Coppersmith, D. (2002, January). An approximate Fourier transform useful in quantum factoring (RC 19642). Arxiv eprints. https://arxiv.org/abs/quant-ph/0201067

[13] Mittal, R. (2014). Lecture 5: Basic quantum algorithms [Slides]. IIT Kanpur. https://www.cse.iitk.ac.in/users/rmittal/prev_course/s19/reports/5_algo.pdf

[14] Team, T. Q. (2021, April 26). Quantum Fourier Transform. Qiskit. https://qiskit.org/textbook/ch-algorithms/quantum-fourier-transform.html

[15] Brigham, E. (1988). Fast Fourier Transform and Its Applications (1st ed.). Pearson.

[16] Quirk: Quantum Circuit Simulator. Quirk. https://bit.ly/3c5rwIa

[17] Brylinski, J. L., & Brylinski, R. (2002). Universal quantum gates. Mathematics of Quantum Computation, 79.

[18] Introduction to quantum computing: Bloch sphere. Akyrillidis.Github.Io. https://akyrillidis.github.io/notes/quant_post_7

[19] An Introduction to Complex Numbers. Nrich Maths. https://nrich.maths.org/1403/index

[20] Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A., & Weinfurter, H. (1995). Elementary gates for quantum computation. Physical Review A, 52(5), 3457–3467. https://doi.org/10.1103/physreva.52.3457

[21] Voorhoede, D. What is a qubit? Quantum Inspire. https://www.quantum-inspire.com/kbase/what-is-a-qubit/

[22] Quantum States and observables. (2016). [Slides]. Berkeley. https://inst.eecs.berkeley.edu/~cs191/fa14/lectures/lecture2.pdf

[23] Gay, S. J., & Nagarajan, R. (2013). Techniques for formal modelling and analysis of quantum systems. In Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky (pp. 264-276). Springer, Berlin, Heidelberg.

[24] Roy, S. C. (2007). Complex Numbers: Lattice Simulation and Zeta Function Applications. Woodhead Publishing.

[25] *Multiplying matrices and vectors*. (n.d.). Math Insight.

https://mathinsight.org/matrix_vector_multiplication

[26] Dirac, P. A. M. (1939). A new notation for quantum mechanics. Mathematical Proceedings of the Cambridge Philosophical Society, 35(3), 416–418. https://doi.org/10.1017/s0305004100021162

[27] Tensor product | Quantiki. Quantiki. Retrieved May 26, 2021, from https://www.quantiki.org/wiki/tensor-product

[28] Parekh, A. The Four Postulates of Quantum Computing - Avilay Parekh. Medium. Retrieved May 26, 2021, from https://medium.com/@avilayparekh/the-four-postulates-of-quantum-computing-a0e6e40d1964

[29] Psara, K. (2020, November). Reversible Computation in Petri Nets (PhD dissertation). University of Cyprus.

[30] Hui, J. (2019, September 3). QC — Observable - Jonathan Hui. Medium. https://jonathan-hui.medium.com/qc-observable-8a44d10c3f7a

[30] Pienaar, J. (2019). A time-reversible quantum causal model. arXiv preprint arXiv:1902.00129.

[31] Shor, P. W. (1994, November). Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th annual symposium on foundations of computer science (pp. 124-134). Ieee.

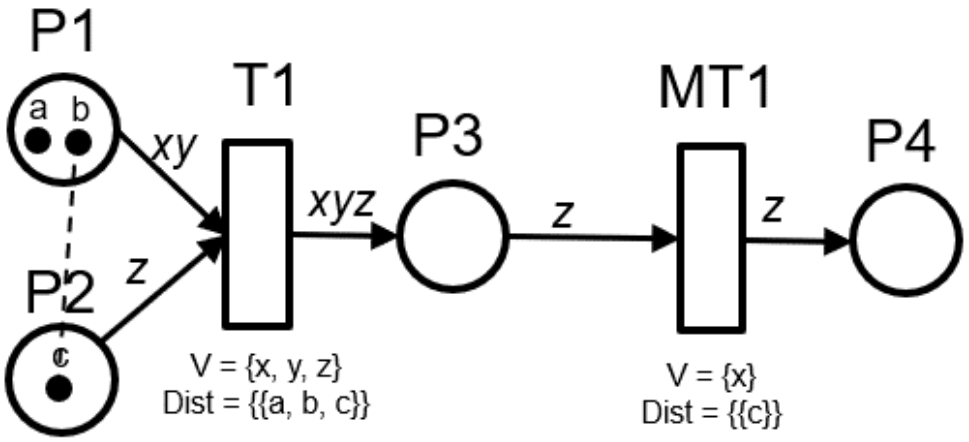
[32] Moore, C., & Crutchfield, J. P. (2000). Quantum automata and quantum grammars. Theoretical Computer Science, 237(1-2), 275-306.

[33] Molina, A., & Watrous, J. (2019). Revisiting the simulation of quantum Turing machines by quantum circuits. Proceedings of the Royal Society A, 475(2226), 20180767.

Appendix A

The variable-encoded entangled state

Consider the following QPN:



As we know so far, in the drawn QPN there are 4 places (P1,P2,P3), one transition T1 with properties: $T1 = \{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \{a, b, c\}$

And one Measurement transition MT1, the marking and value, are:

$M_0 = \{P1: \{a,b\}, P2:\{c\}, P3:\{\}, P4:\{\}\}$,

$Val_0 = \{a: \begin{bmatrix} 0 \\ 1 \end{bmatrix}, bc \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}\}$

Now, in the variable encoded entangled state, we attempt to "split" the entangled state bc, into two separate ones. But since the state of bc cannot be written as a result of tensor products of simpler states, the coefficients of b and c in their states are replaced by variables, which are inferred from the shared entangled state.

In the presented QPT the entangled state bc is split into $|b\rangle = w|0\rangle + x|1\rangle$ $|c\rangle = y|0\rangle + z|1\rangle$.

And together they satisfy the following equation:

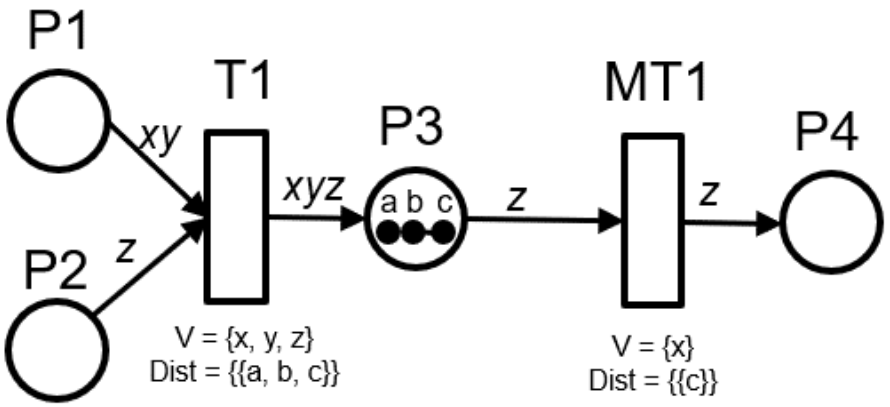
$$|b\rangle \otimes |c\rangle = \begin{bmatrix} w \\ x \end{bmatrix} \otimes \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} wy \\ wz \\ xy \\ xz \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

The value component is stored as Val_0 ={a: $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, b: $\begin{bmatrix} w \\ x \end{bmatrix}$, c: $\begin{bmatrix} y \\ z \end{bmatrix}$} additionally, an extra component is added into the definition of the QPT, *Entanglements* (S) which stores the original entangled state, the variables which encode every coefficient of the composing states.

Now, the QPT has the same graphical representation, but

Val_0 ={a: $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, b: $\begin{bmatrix} w \\ x \end{bmatrix}$, c: $\begin{bmatrix} y \\ z \end{bmatrix}$}, and S ={$|b\rangle \otimes |c\rangle = \begin{bmatrix} w \\ x \end{bmatrix} \otimes \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} wy \\ wz \\ xy \\ xz \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$}

After T1 fires, the QPT is the following:



M_0 ={P1: {}, P2:{}, P3:{a,b,c}, P4:{}}.

The val_1 is computed as normally, with the tensor products, however after the computation of the final state, the S and Val components are updated accordingly.

The calculations for the execution of the transition are:

$$(|a\rangle \otimes |b\rangle \otimes |c\rangle) \cdot \left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ wy \\ xy \\ wz \\ xz \end{bmatrix}$$

Notice that the resulting combinations of variables can be translated to values from the S component. If this is not possible, then the states are stored with the variables.

The resulting shared state can be split as $\begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} wy \\ xy \\ wz \\ xz \end{bmatrix}$. The bc tokens are still entangled, however the order has changed. Thus, the S component is updated:

$$S = \{|b\rangle \otimes |c\rangle = \begin{bmatrix} w_2 \\ x_2 \end{bmatrix} \otimes \begin{bmatrix} y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} w_2 y_2 \\ w_2 z_2 \\ x_2 y_2 \\ x_2 z_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}\},$$

$$\begin{bmatrix} w_2 y_2 \\ w_2 z_2 \\ x_2 y_2 \\ x_2 z_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} wy \\ xy \\ wz \\ xz \end{bmatrix} \text{ (computed from previous S)}$$

Val_1 ={a: $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, b: $\begin{bmatrix} w_2 \\ x_2 \end{bmatrix}$, c: $\begin{bmatrix} y_2 \\ z_2 \end{bmatrix}$}

The variables are updated with each transition that uses them and update the stored state in S by using the previous value pairs.

The execution of TM1 collapses c into a basis state, suppose $|1\rangle$. Now $\begin{bmatrix} y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

From S we can derive: $\begin{bmatrix} w_2y_2 \\ w_2z_2 \\ x_2y_2 \\ x_2z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ w_2 \\ 0 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ xy \\ 0 \\ xz \end{bmatrix} = \begin{bmatrix} wy \\ xy \\ wz \\ xz \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ (collapsed state).

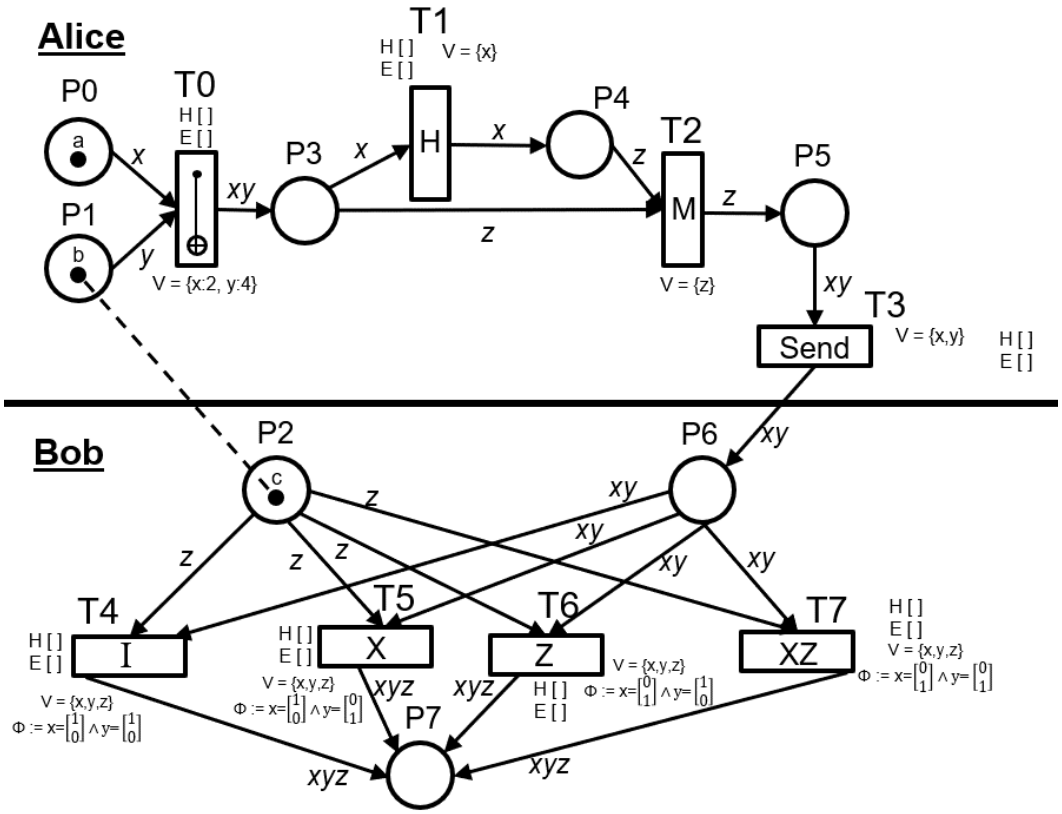
Thus $\begin{bmatrix} w_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ via solving the equations above.

Appendix B

Sample execution of quantum teleportation

In this appendix, a sample execution of the teleportation protocol is shown, step by step. The model shown is a RQPN, which covers the QPN version as well.

Consider the RQPN found in section 4.1, for the teleportation protocol.



Suppose that a is found in state $\begin{bmatrix} \frac{1}{\sqrt{2}}i \\ -\frac{1}{\sqrt{2}}i \end{bmatrix}$, and qubits b, c are found in state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_\alpha \otimes |0\rangle_b + 1\rangle_\alpha \otimes |1\rangle_b)$.

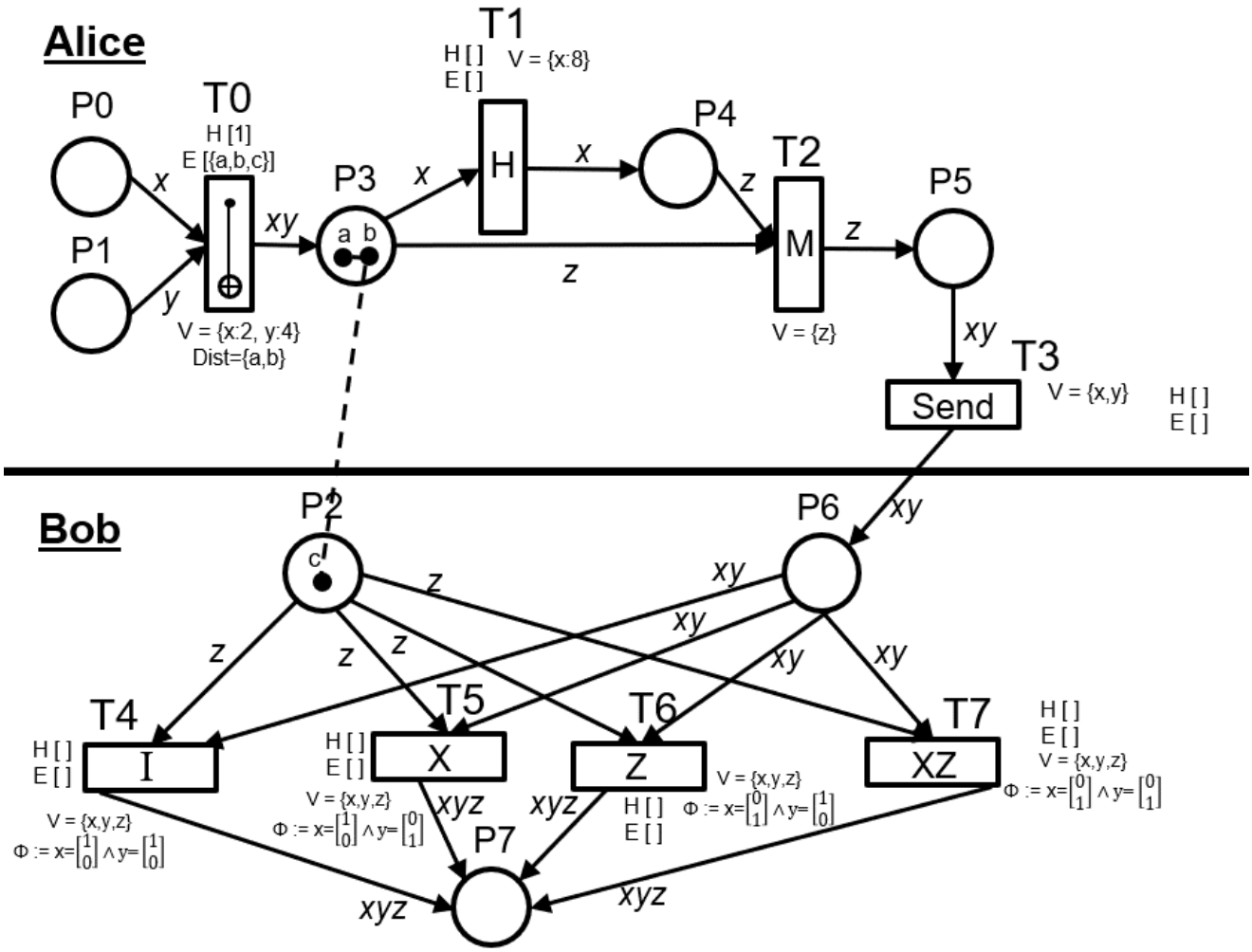
There are 8 places and 8 transitions.

M_0={P0:{a}, P1:{b}, P2:{c}, P3:{}, P4:{}, P5:{}, P6:{}, P7:{}}

Val_0={a: $\begin{bmatrix} \frac{1}{\sqrt{2}}i \\ -\frac{1}{\sqrt{2}}i \end{bmatrix}$, bc: $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$}

1. After T0 (CNOT on b with c being the control)

   The RQPN is found in the following state:



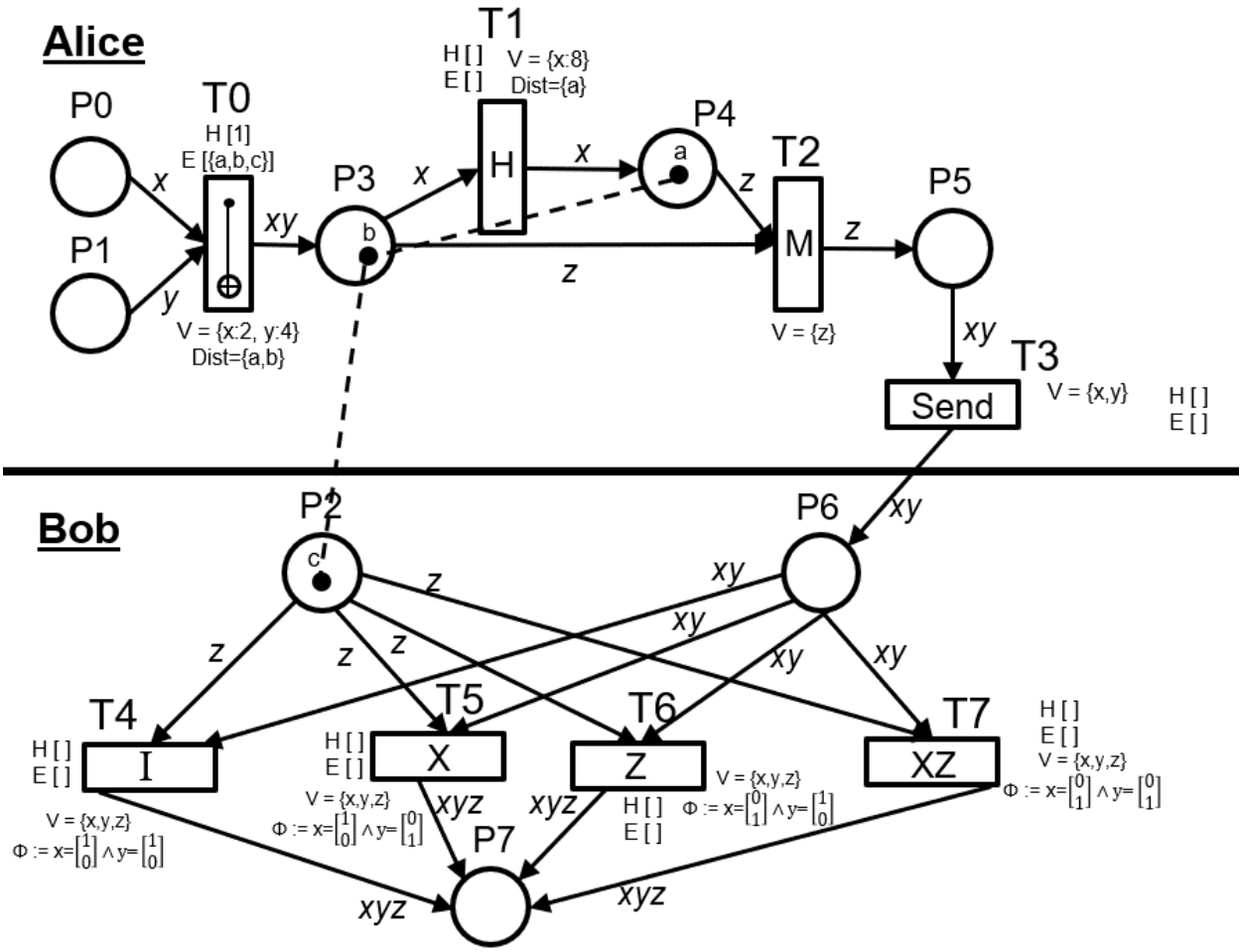
M_1 ={P0:{}, P1:{}, P2:{c}, P3:{a,b}, P4:{}, P5:{}, P6:{}, P7:{}}

$$Val_1 = \{abc: \begin{bmatrix} \frac{1}{2}i \\ 0 \\ 0 \\ \frac{1}{2}i \\ 0 \\ -\frac{1}{2}i \\ -\frac{1}{2}i \\ 0 \end{bmatrix}\}$$

The $H_1(t)$ and $E_1(t)$ components are shown on the diagram, next to each transition.

2. After T1 (Hadamard on a)

   The RQPN is found in the following state:

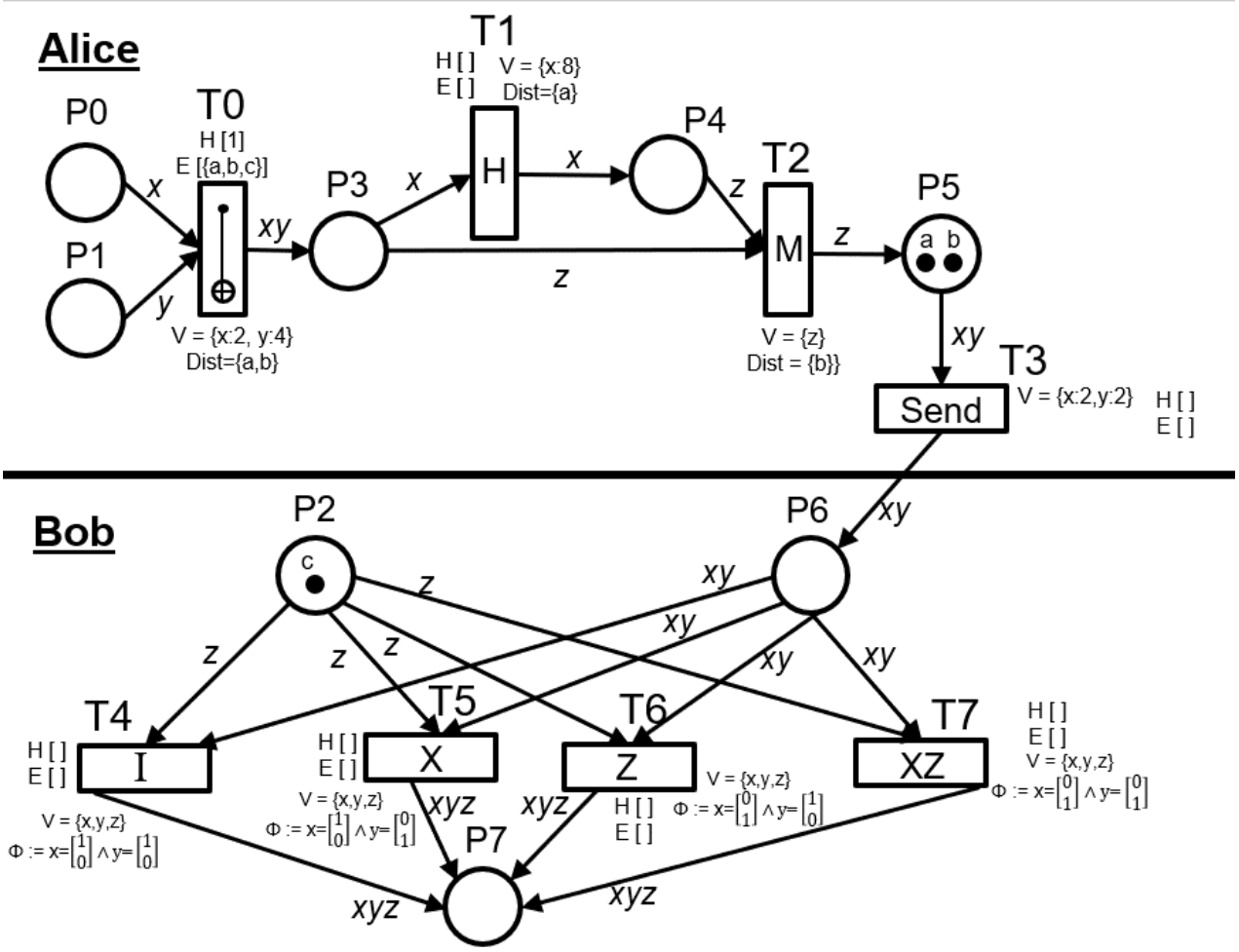


M_2 ={P0:{}, P1:{}, P2:{c}, P3:{b}, P4:{a}, P5:{}, P6:{}, P7:{}}

$$Val_2 = \{abc: \begin{bmatrix} \frac{1}{2\sqrt{2}}i \\ -\frac{1}{2\sqrt{2}}i \\ -\frac{1}{2\sqrt{2}}i \\ \frac{1}{2\sqrt{2}}i \\ \frac{1}{2\sqrt{2}}i \\ \frac{1}{2\sqrt{2}}i \\ \frac{1}{2\sqrt{2}}i \\ \frac{1}{2\sqrt{2}}i \end{bmatrix}\}$$

The $H_2(t)$ and $E_2(t)$ components are shown on the diagram, next to each transition.

3. After T2 – measurement of qubits a ,b (Suppose both qubits collapsed in state $|1\rangle$)
   The RQPN is found in the following state:



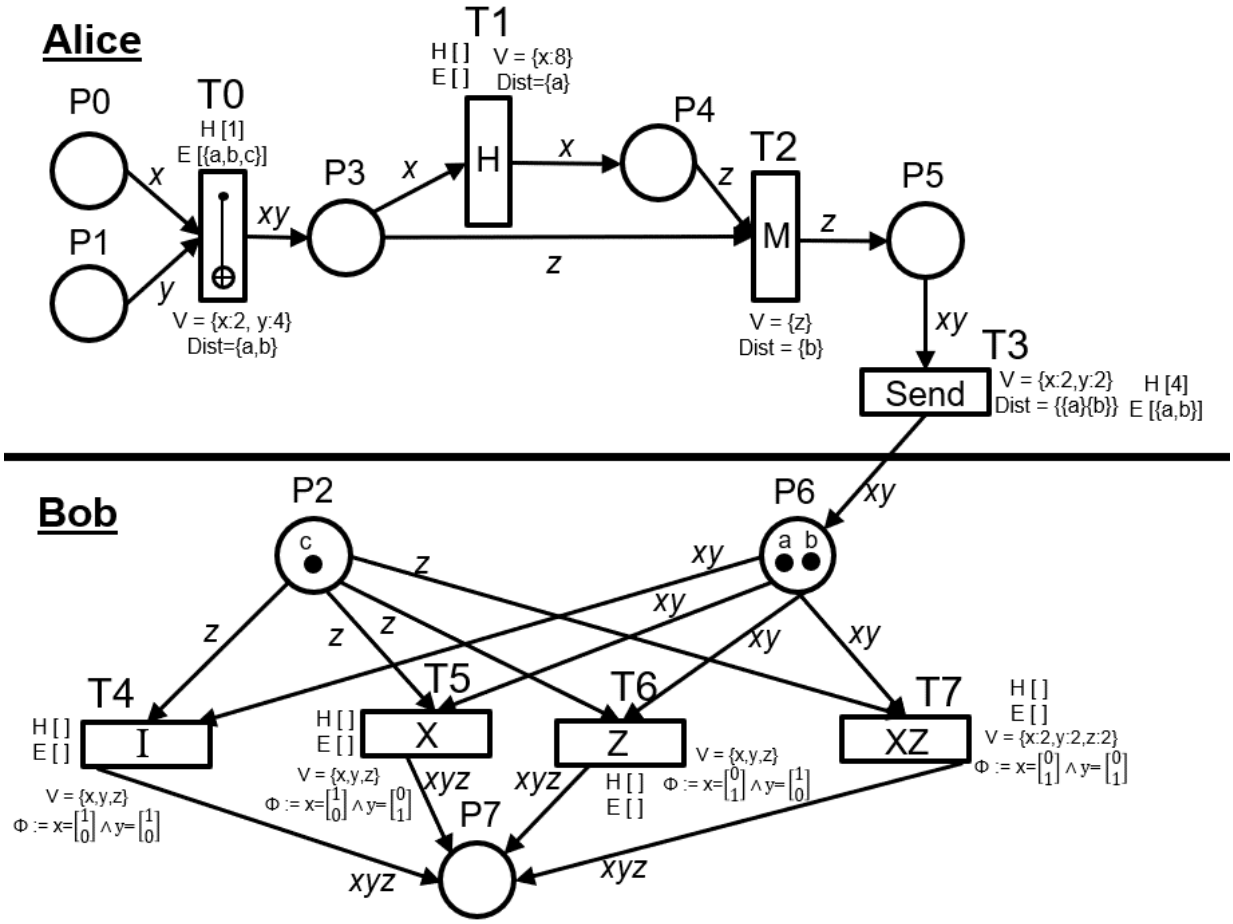
M_3 ={P0:{}, P1:{}, P2:{}, P3:{}, P4:{a}, P5:{}, P6:{}, P7:{}}

Val_3={ a: $\begin{bmatrix}0\\1\end{bmatrix}$, b: $\begin{bmatrix}0\\1\end{bmatrix}$, c: $\begin{bmatrix}\frac{1}{\sqrt{2}}i\\\frac{1}{\sqrt{2}}i\end{bmatrix}$}

The $H_3(t)$ and $E_3(t)$ components are shown on the diagram, next to each transition.

4. After T3 – sending measurement results to Bob.

   The RQPN is found in the following state:

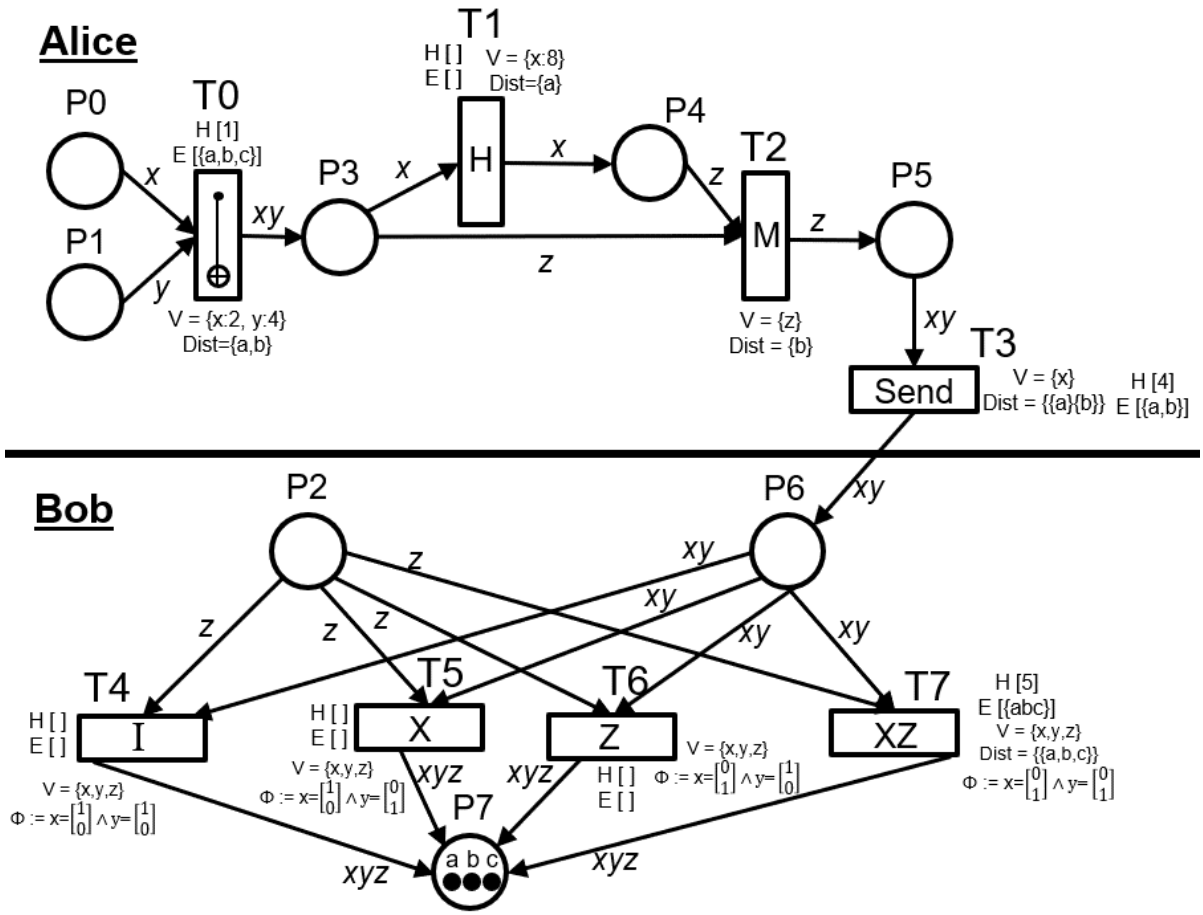


M_4 ={P0:{}, P1:{}, P2:{c}, P3:{}, P4:{}, P5:{}, P6:{a, b}, P7:{}}

Val_4={ a: $\begin{bmatrix}0\\1\end{bmatrix}$, b: $\begin{bmatrix}0\\1\end{bmatrix}$, c: $\begin{bmatrix}\frac{1}{\sqrt{2}}i\\\frac{1}{\sqrt{2}}i\end{bmatrix}$}

The $H_4(t)$ and $E_4(t)$ components are shown on the diagram, next to each transition.

5. After T7 – applying the respective transformation on c, based on the received bits. The RQPN is found in the following state:



M_4 ={P0:{}, P1:{}, P2:{c}, P3:{}, P4:{}, P5:{}, P6:{a, b}, P7:{}}

Val_4={ a: $\begin{bmatrix}0\\1\end{bmatrix}$, b: $\begin{bmatrix}0\\1\end{bmatrix}$, c: $\begin{bmatrix}\frac{1}{\sqrt{2}}i\\-\frac{1}{\sqrt{2}}i\end{bmatrix}$}

The $H_4(t)$ and $E_4(t)$ components are shown on the diagram, next to each transition.

The state of qubit c is now the original state of qubit a $\begin{bmatrix}\frac{1}{\sqrt{2}}i\\-\frac{1}{\sqrt{2}}i\end{bmatrix}$!

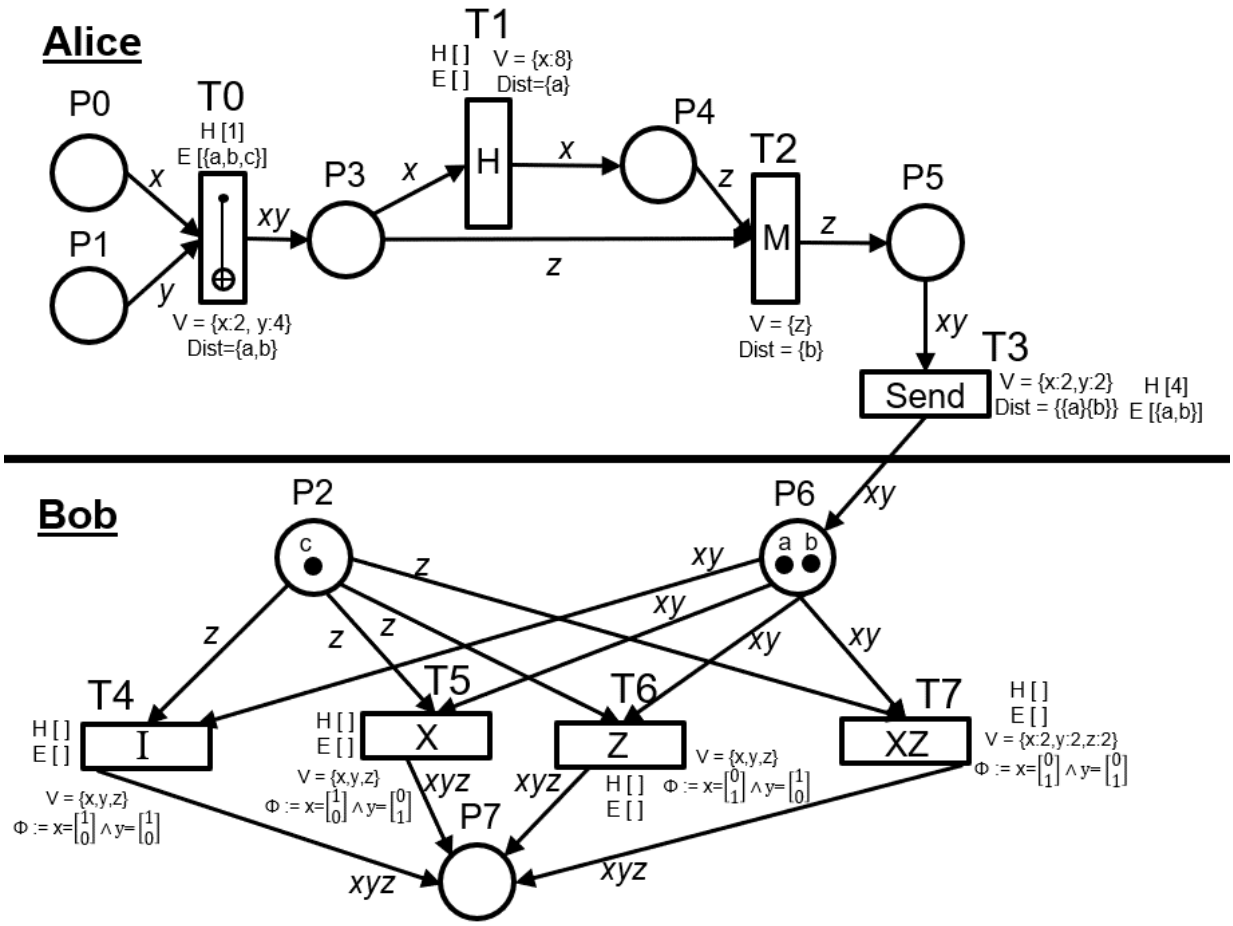
Let us clarify:

- In any step, the only possible execution was displayed, as only one transition was enabled at any time.
- After step 3 (measurement), there are four possible executions, depending on the values of the collapsed states of qubits a and b. This will later enable only 1 transition (T4-T7) in Bob's machine, that will transform the state of c accordingly and result in the original state of a.
- The execution with a QPN would be simpler, as the History and Entanglement History elements would be removed from the graphical representation, as well as the global state of the model. The rest would be the same.

As a bonus, let us examine the backtracking of step 5.

We only bt-enabled transition is T7, as it was the last forward-executed transition and it is not a measurement, thus its inverse can be calculated.

After applying the according computations and popping the according elements from E(T7) and H(T7). The RQPN is found in the following state:



M_4 ={P0:{}, P1:{}, P2:{c}, P3:{}, P4:{}, P5:{}, P6:{a, b}, P7:{}}

Val_4={ a: $\begin{bmatrix}0\\1\end{bmatrix}$, b: $\begin{bmatrix}0\\1\end{bmatrix}$, c: $\begin{bmatrix}\frac{1}{\sqrt{2}}i\\\frac{1}{\sqrt{2}}i\end{bmatrix}$}

The $H_4(t)$ and $E_4(t)$ components are shown on the diagram, next to each transition.

Upon taking a closer look, we can see that this is identical to the global state at step 4, which means that the Backtracking was successful.