Research Internship (PRe)

Field of Study : Quantum Computing
Scholar Year : 2022-2023

Hamiltonian representation of a neutral atom quantum computer using the ZW†-Calculus graphical language

Confidentiality Notice
Non-confidential and publishable report

Author : Joaquim MINARELLI GASPAR **Promotion : 2024**

ENSTA Paris Tutor : Omar Hammami
Host Organism Tutor : Renaud Vilmart and Benoît Valiron

Internship from 15/06/2023 to 18/08/2023

Inria
Address : LMF 4,Avenue des Sciences, 91190, Gif-Sur-Yvette

This page was intentionally left blank.

Confidentialy Notice

This present document is not confidential. It can be communicated outside in paper format or distributed in electronic format.

Abstract

An extension of the ZW-calculus for the quantum has recently been introduced to represent and manipulate Hamiltonians. During this internship, we used this language and its equational theory to explore the possibility of transforming any Hamiltonian, to satisfy constraints imposed by the architecture of a neutral atom quantum computer while altering it as little as possible.

Keywords— Graphical language, ZW-Calculus, Quantum computing, Neutral atom quantum computer

Résumé

Une extension du ZW-calcul pour le quantique a récemment été introduite pour représenter et manipuler des Hamiltoniens. Durant ce stage, nous avons utilisé ce langage et sa théorie équationnelle pour explorer la possibilité de transformer un Hamiltonien quelconque, pour satisfaire des contraintes imposées par l'architecture d'un ordinateur quantique à atomes neutres tout en l'altérant au minimum.

Mots clés— Langage graphique, ZW-Calculus, Informatique quantique, Ordinateur quantique à atome neutre

Acknowledgments

I would like to express my gratitude to my supervisors at Inria, Renaud Vilmart and Benoît Valiron, for all their help and for introducing me to such a fascinating field of study. I thank my colleagues from the QuaCs/LMF group for all the enriching discussions I had during the internship period. To my parents and brother, for all their support.

Contents

List of Figures

1 Introduction

Quantum computing is an area of physics and computer science that studies processing and information tasks that can be performed with quantum systems (NIELSEN; CHUANG, 2001). This field of study aims to use intrinsic properties of quantum mechanics, such as quantum entanglement and superposition of quantum states, to simulate quantum systems and perform algorithms more efficiently.

Several quantum computer architectures are currently under development, focusing on efficiently executing quantum algorithms with minimal error rates. One of these architectures, called neutral atom quantum computer, developed by companies like *QuEra* and the French start-up *Pasqal*, has gained attention thanks to recent developments in optical-tweezer technology. This technology enables efficient simulation of quantum many-body systems and facilitates the resolution of challenging combinatorial optimization problems, such as the Maximum Independent Set problem.

Like every quantum system, the neutral atom quantum computer has a Hamiltonian $\mathcal{H}$ that governs the dynamics of the system (evolution of the qubits states in time). However, given a quantum algorithm represented by a certain Hamiltonian $\hat{\mathcal{H}}$, simulating it in an equivalent way using the architecture of a neutral atom quantum computer is not trivial. This happens because the Hamiltonian structure of the neutral atom quantum computer changes according to the way the atoms are initially positioned and the frequency at which they are excited, and finding such conditions so that the sctructure of $\mathcal{H}$ approximates $\hat{\mathcal{H}}$ can be very complex and even require the use of auxiliary qubits.

One of the possible alternatives to find an equivalence between $\mathcal{H}$ and $\hat{\mathcal{H}}$ is to use a diagram representation in some quantum graphical language, using the simplification rules of that language to graphically approach both hamiltonians, finding a direct equivalence between them.

The first graphical language for quantum computing (ZX-Calculus) was conceived in 2008 (COECKE; DUNCAN, 2008). The first to be proven complete and universal was ZW-Calculus, conceived in 2010 (COECKE; KISSINGER, 2010). In this work, we will the ZW-Calculus and one of its extensions, named ZW†-Calculus, to manipulate quantum circuits (VILMART, 2023). The choice of using the ZW†-Calculus will be justified later on, but it is related to the ease of representing the sum of diagrams, which will facilitate the Hamiltonian representation of the neutral atom quantum computer.

1.1 Objectives and division of the report

The objective of this work is to use the ZW†-Calculus graphical language to manipulate and simplify the Hamiltonian of a neutral atom quantum computer, specifically the analog version produced by the French company Pasqal, in order to use this architecture to approximate any Hamiltonian that represents a quantum algorithm.

Therefore, this report is divided as follows: a background section will describe all the quantum computing concepts needed to understand the architecture of the neutral atom quantum computer. Furthermore, a detailed section on ZW†-Calculus will cover all the concepts needed to describe and simplify any Hamiltonian.

In the results section, the diagram representing the Hamiltonian of the neutral atom quantum computer will be written for a system of 1, 2 and n qubits. Also, a fragment of ZW†-Calculus will be defined, pW†-Calculus, which can be used to efficiently rewrite diagrams. Finally, appendix A brings a different approach to solve the proposed problem, translating the problem into an optimization model, and showing the difficulties in using the method, justifying the approach by graphical language simplification.

2 Background

This section will present the basic concepts of quantum computing, the architecture of a neutral atom quantum computer and the graphical language that was used to manipulate the Hamiltonian of this computer, named ZW†-Calculus.

2.1 Quantum computing principles

Unlike a classical computer, a quantum computer operates using quantum bits, also called *qubits*. Each qubit possesses two distinct states, which are commonly represented using Dirac notation $|0\rangle$ and $|1\rangle$ (analogous to "spin up" and "spin down" in the context of qubits as spins).

The distinction between bits and qubits lies in the fact that while a bit can only be in states 0 or 1, a qubit can form linear combinations of states, commonly known as superpositions:

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

In the equation 1, α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. Thus, $|\alpha|^2$ represents the probability of the qubit collapses into the value 0 after the measurement, and $|\beta|^2$ the probability that it collapses to the value 1. Analogously, a qubit can be considered as a vector in a complex vector space of dimension 2, such that the vectors $|0\rangle$ and $|1\rangle$ form an orthonormal basis of this space (NIELSEN; CHUANG, 2001).

Normally, the usual representation of the vectors $|0\rangle$ and $|1\rangle$ are $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$, respectively. However, exceptionally when working with a neutral atom quantum computer, there is a convention of using the notation $|1\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|0\rangle = \begin{bmatrix}0\\1\end{bmatrix}$, which helps to simplify the Hamiltonian of the system as a sum of Pauli matrices (set of complex 2x2 hermitian and unitary matrices).

Since $|\alpha|^2 + |\beta|^2 = 1$, is there a way to rewrite the equation 1 using spherical coordinates:

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\phi}sin\frac{\theta}{2}|1\rangle \tag{2}$$

This allows for a graphical representation of the qubit as a point on a three-dimensional sphere, as illustrated in figure 1 (NIELSEN; CHUANG, 2001).

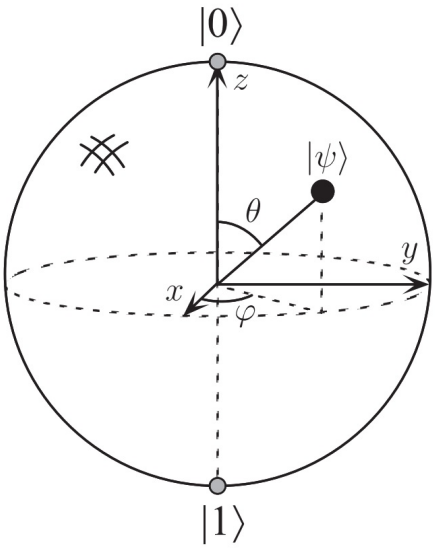


Figure 1 – Bloch-sphere

Naturally, there can be systems consisting of more than one qubit. For the case of 2 qubits, the state of the system would be $|\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$, with $|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1$. In general, given a system with n qubits, the system can assume 2^n quantum states. In this case, the state of the system would have the following mathematical representation:

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} a_x |x\rangle \tag{3}$$

where $\{0,1\}^n$ denotes the set of all possible strings of 0 and 1 of lenght n, and $\sum_{x\in\{0,1\}^n} |a_x|^2 = 1$. In addition, the notation $|x\rangle = |b_1...b_n\rangle$, where n is the number of qubits, means the Kronecker product $|b_1\rangle \otimes ... \otimes |b_n\rangle$. The Kronecker product between 2 matrices A and B can be formalized as:

$$A \otimes B = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

where a_{ij} are the terms of the matrix A.

2.1.1 Quantum gates and quantum circuits

We call a quantum gate a unitary operator $U : \psi_1 \to \psi_2$ that maps the state ψ_1 into ψ_2, and is described as a unitary matrix relative to the base of ψ_1 and ψ_2. Quantum gates are usually represented by boxes connected to lines, which represents qubits.

The simplest example of a quantum gate is the quantum equivalent of the NOT gate $X : \begin{bmatrix}\alpha \\ \beta\end{bmatrix} \to \begin{bmatrix}\beta \\ \alpha\end{bmatrix}$, which performs the action of swapping the amplitude α and β. Its matrix representation is the same as the Pauli-X matrix:

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

The usual graphical representation of this operation can be seen in figure 2 (NIELSEN; CHUANG, 2001), and should be read from left to right.

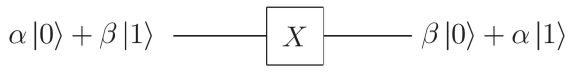


Figure 2 – NOT gate

Quantum circuits are a set of one or more quantum gates applied to qubits, which map an initial state $|\psi_{initial}\rangle$ into a final state $|\psi_{final}\rangle$. The single wires represent qubits, and the double wires represent classic bits, which were mapped from the probabilistic operator M, which converts the qubit into bit 0 with probability $|\alpha|^2$ and into bit 1 with probability $|\beta|^2$. Such a representation can be seen in figure 3 (NIELSEN; CHUANG, 2001).

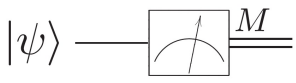


Figure 3 – Measurement process

2.2 Neutral atom quantum computer

Neutral atoms quantum computers have two computing modes: digital and analog. In the analog mode, which will be explored in this work, the dynamics of the system (state of the qubits) is governed by a Hamiltonian that depends on the interactions of atoms in the Rydberg state, which are the key to understanding this type of quantum computer architecture.

2.2.1 Rydberg Atoms

Rydberg atom is a generic name for any atom that has an electron in a higher quantum state. The goal of quantum computing with neutral atoms is to build the qubit concept based on the states of these atoms: the qubit $|0\rangle$ is considered to be the *ground state* (unexcited atom) and the qubit $|1\rangle$ is considered to be the *rydberg state* (excited atom). The notations used to refer to both states are $|g\rangle$ and $|r\rangle$, respectively.

The mechanism that allows quantum computing from Rydberg atoms, and that generates entanglement between qubits, is an effect called the Rydberg Blockade: if two atoms A and B are close enough, which means that $d(A, B) << R_b$, where R_b is called *blockade radius*, both will have a very small chance of being simultaneously excited to the Rydberg state, which is proportional to the sixth power of the distance between the two atoms.

Thus, if there is a system with two atoms that are both at a distance smaller than R_b, and both are excited with a laser with a certain Rabi frequency Ω, the final state of the system will be the entangled state $\frac{|rg\rangle+|gr\rangle}{\sqrt{2}}$. In general, if there is a system with N atoms and they are all within a certain radius R_b, the state of the system will be:

$$|\phi\rangle = \frac{|rgg...g\rangle + |grg...g\rangle + |ggg...r\rangle}{\sqrt{N}}$$

and each atom will have a $\frac{1}{N}$ chance of being excited.

2.2.2 Graph theory and independet sets

As the final state of the qubits of the neutral atom quantum computer depends on the initial disposition of the atoms, there is a natural connection between neutral atoms quantum computers and graph theory. Let's consider the undirected graph G = (V, E) such that every node $v \in V$ represents an atom, and if two nodes u and v are connected by an edge $(u, v) \in E$, it means that they cannot be simultaneously excited to the Rydberg state.

Figure 4 indicates an initial arrangement of 4 atoms, together with the possible states of the system. In the left figure, only atoms that are directly adjacent cannot be excited (called "Nearest-Neighbor" case). In the right figure, the blockade radius is bigger which means that only 1 atom can be excited simultaneously ("Next Nearest-Neighbor" case).

Figure 4 – Possible states

In the general case, given an undirected graph which represents the initial geometry of atoms, the possible states that the system can assume is equivalent to the sets of possible independent

sets of that graph. An independent set of a graph G is a set S of vertices of G such that there are no two adjacent vertices contained in S.

Hence, given a system formed by n atoms (qubits), instead of working with a Hilbert space $\mathbb{H}$ of dimension 2^n, we will have a number of possible states equivalent to the cardinality of the set of the independent sets of the initial graph.

2.2.3 Hamiltonian

The Hamiltonian that describes the system given a configuration of n atoms driven by a coherent laser with a Rabi frequency Ω and detuning frequency δ is given by the equation 4

$$\frac{\mathcal{H}(t)}{\hbar} = \sum_i \frac{\Omega_i(t)}{2}\sigma_x^i - \sum_i \delta_i(t)\hat{n}_i + \sum_{i<j} \frac{C_6}{|x_j - x_i|^6}\hat{n}_j\hat{n}_i \tag{4}$$

where:

— $\hat{n}_i \equiv |r_i\rangle\langle r_i|$: Rydberg state occupancy operator for atom i. It is related with σ_z by $\hat{n}_i = \frac{\mathbb{I}+\sigma_z^i}{2}$;

— x_i denotes the position of the atom i;

— C_6 is a coupling constant that describes the interaction strength between the atoms.

In addition, it is possible to find the Rydberg blockade radius from the Rabi frequency Ω and detuning frequency δ according to the following relation:

$$R_b = \left(\frac{C_6}{\sqrt{\Omega^2 + \delta^2}}\right)^{\frac{1}{6}}$$

Note that, from the equation 4 and the fact that the number operator can be written as a function of the identity and the Pauli-Z, the Hamiltonian describing the behavior of a neutral atom quantum computer can all be described in terms sum of Pauli matrices, which will be useful later for its representation in terms of ZW†-Calculus diagrams.

2.3 ZW-Calculus and ZW†-Calculus

In this section, all the necessary concepts for the representation and simplification of diagrams in ZW-Calculus and ZW†-Calculus will be presented, including the representation of Hamiltonians with controlled diagrams and the definition of a ZW-Calculus planar fragment, the pW-Calculus, which will be extended to ZW†.

2.3.1 Introduction and axiomatization

In quantum computing, every quantum circuit that maps n qubits into m qubits can be represented by a $\mathbb{C}^{2^n} \to \mathbb{C}^{2^m}$ map. Since the dimension of the operator that represents this map grows exponentially as the number of inputs and outputs increases, it is natural to use a graphical representation of this type of relationship such that the number of terms (or diagrams) grows polynomially as the number of qubits increases.

In ZW-Calculus, we define our quantum processes as boxes (which represent an operator, generically represented by D) with n wires of inputs and m outputs, representing the n and m qubits of input and output, respectively. We commonly use the $D : n \to m$ notation to refer to a diagram with n inputs and m outputs, with n and m non-negative integers, and its graphical representation is:

Diagrams can be composed either in parallel (physically representing a tensor product between two operators) or sequentially (representing a composition):

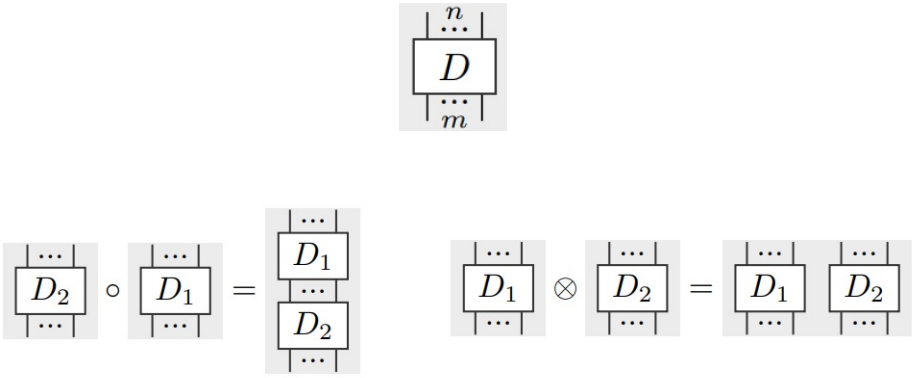


Figure 5 – Composition of diagrams

The vanilla ZW-Calculus is composed of the following set of generators, illustrated in figure 6. The two generators on the left are commonly called white dot (or white spider) and black dot (or black spider), respectively. Furthermore, a white dot with no value inside is used to represent the case $r = 1$. Finally, the notation $|xy|$ in the black spider generator summation represents the Hamming weight (VILMART, 2023).

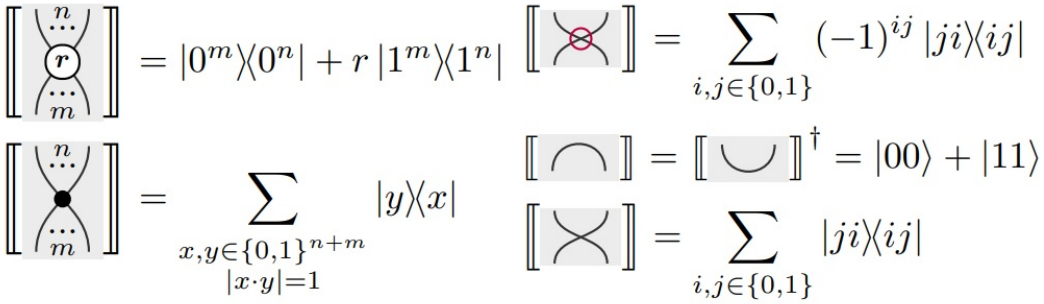


Figure 6 – ZW-Calculus generators

Furthermore, the swap with a red circle around it is called "fermionic swap", it is just the synthetic representation of the diagram illustrated in figure 7, and therefore presents a planar representation.

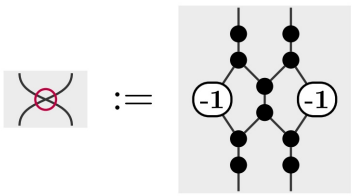


Figure 7 – Fermionic swap diagram

In ZW-Calculus, we also have the following definitions:

Definition (State): A *state* in ZW-Calculus is a diagram $D : 0 \to m$, for m natural greater than zero (called *outputs*). Given a diagram $D : n \to m$ in ZW-Calculus, there is always the isomorphism between $D : n \to m$ and $D' : 0 \to n + m$ given by figure 8. Thus, every diagram D can be represented as a state.

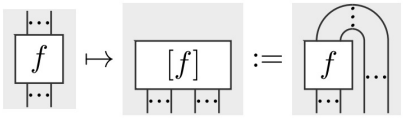
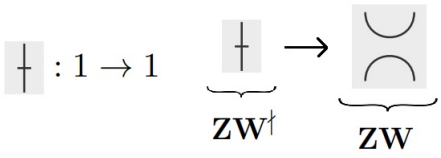


Figure 8 – Isomorphism between $D : n \to m$ and $D' : 0 \to n + m$

Definition (Degree): For every black and white node with n inputs and m outputs, we define the degree of this node as $n + m$.

In ZW-Calculus,there is also a diagram for the identity, represented simply by a wire. Analogously, when placing n wires in parallel, we obtain the identity applied in n qubits.

For ZW†-Calculus, we introduce an additional generator, named "tick", which has the following interpretation. There is a simple way to map every diagram in ZW†-Calculus to ZW-Calculus, which will be explained later.



It is important to note that two consecutive "ticks" in a wire result in identity, which means that a wire can have a maximum of 1 "tick".

The following figure illustrates a set of axioms in ZW-Calculus, with $r, s \in \mathbb{R}$, which are sufficient to prove the universality and completeness of the language for operations with qubits (CARETTE et al., 2023a).

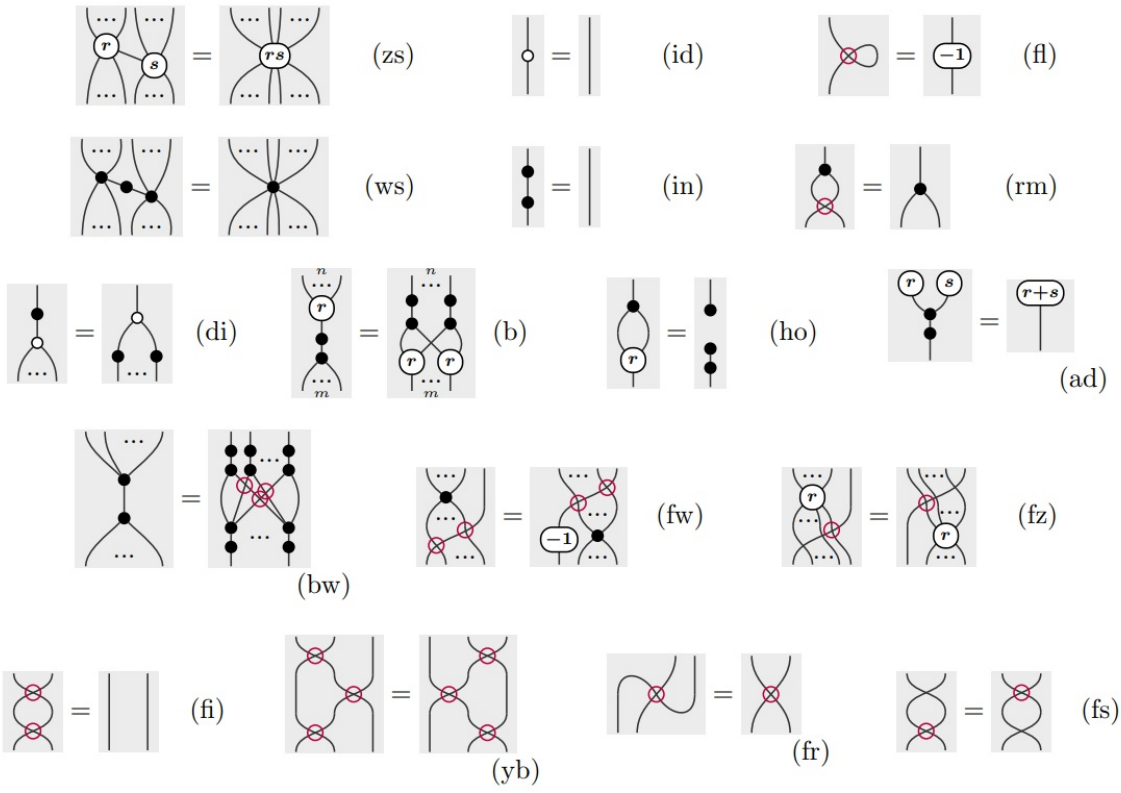


Figure 9 – ZW-Calculus axioms

For ZW†-Calculus, some additional axioms are needed to prove the completeness of the language, which are described in figure 10 (VILMART, 2023).

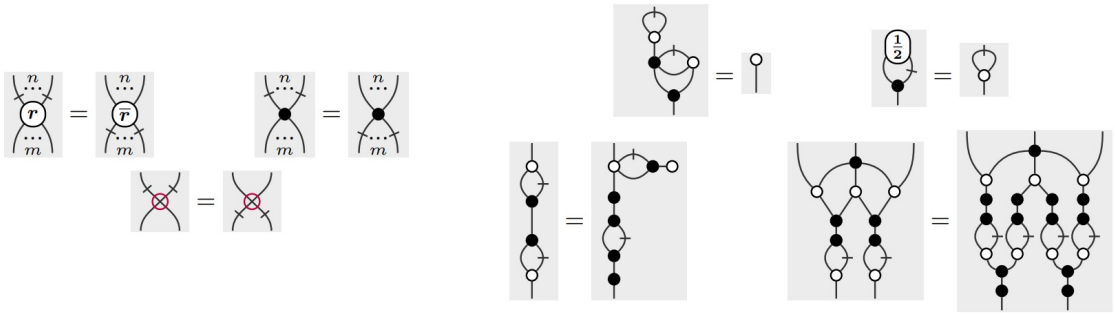


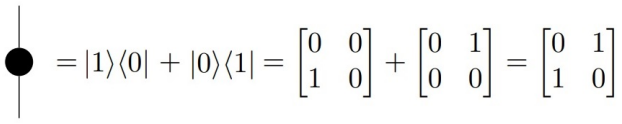
Figure 10 – ZW†-Calculus axioms

Finally, it is always possible to multiply a diagram by a real number. This has the interpretation of multiplying the diagram equivalent operator by this real number.

From now on, we will consider a diagram D that has at least one ticked wire to be written in ZW†-Calculus. In case it doesn't have any ticks, we will consider the diagram is written in ZW-Calculus.

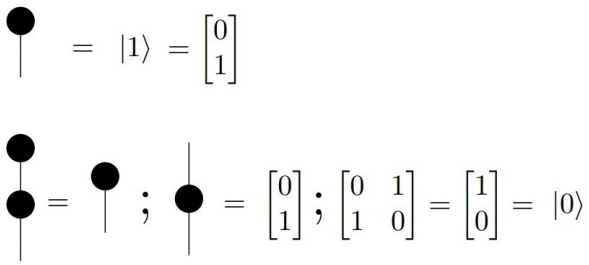
2.3.2 Interpretation of ZW and ZW†-Calculus diagrams

As mentioned in the previous section, a diagram D in ZW-Calculus represents an operator acting on n qubits in the input, and mapping them to m qubits in the output. For example, a diagram represented by a black dot with one input and one output ($n = 1$ and $m = 1$) represents the Pauli-X operator acting on 1 qubit:



$$= |1\rangle\langle 0| + |0\rangle\langle 1| = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Since there is no restriction that n and m must be strictly positive, we call the set of diagrams D such that $n = 0$ *states*. Thus, the pure states $|0\rangle$ and $|1\rangle$ can be represented as shown in figure 11, where the symbol ";" represents the composition made from left to right.



$$= |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \; ; \; = \begin{bmatrix} 0 \\ 1 \end{bmatrix} ; \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

Figure 11 – Pure ZW-Calculus states

In ZW†-Calculus, due to the action of the tick generator, a diagram with n inputs and m outputs represents a super-operator which maps one operator onto another, and is represented mathematically by $\mathcal{P} : \mathbb{M}_{2^n}(\mathbb{C}) \to \mathbb{M}_{2^m}(\mathbb{C})$. Thus, a state in ZW†-Calculus (diagram in which $n = 0$), mathematically represented by $\rho : \mathbb{C} \to \mathbb{M}_{2^m}(\mathbb{C})$, is equivalent to an operator in ZW-Calculus.

In summary, we have the following equivalence, with some examples, between ZW-Calculus and ZW†-Calculus, illustrated in figure 12.

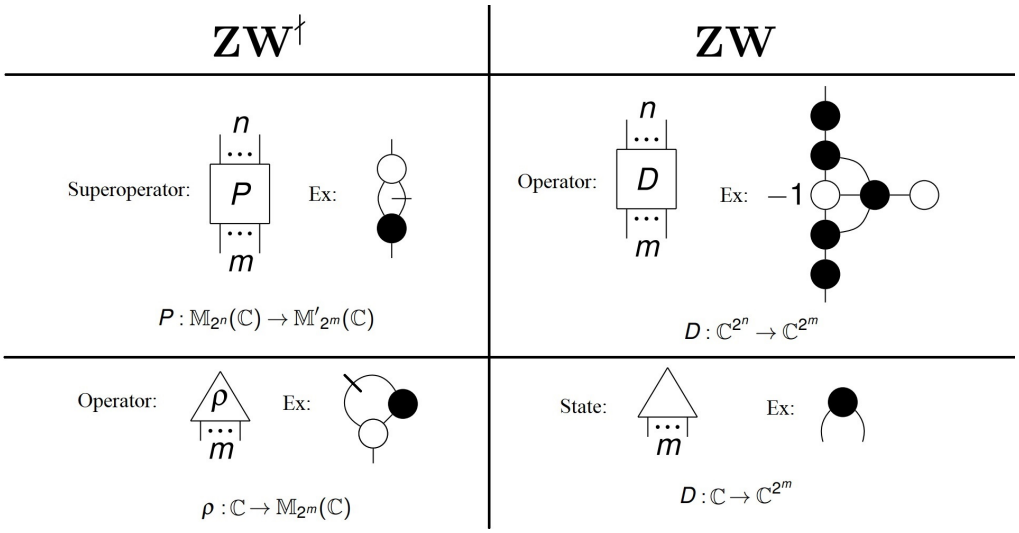


Figure 12 – Summary of ZW-Calculus and ZW†-Calculus

Every diagram in ZW†-Calculus has an equivalent diagram in ZW-Calculus. To find this equivalence, it is necessary to define two other maps: HP and ι, such that, for every diagram $D^\dagger$ in ZW†-Calculus, $D = \iota(HP(D^\dagger))$ represents its respective operator in ZW-Calculus (VILMART, 2023).

Thus, we define the map $HP : ZW^\dagger \to ZW$ as follows:

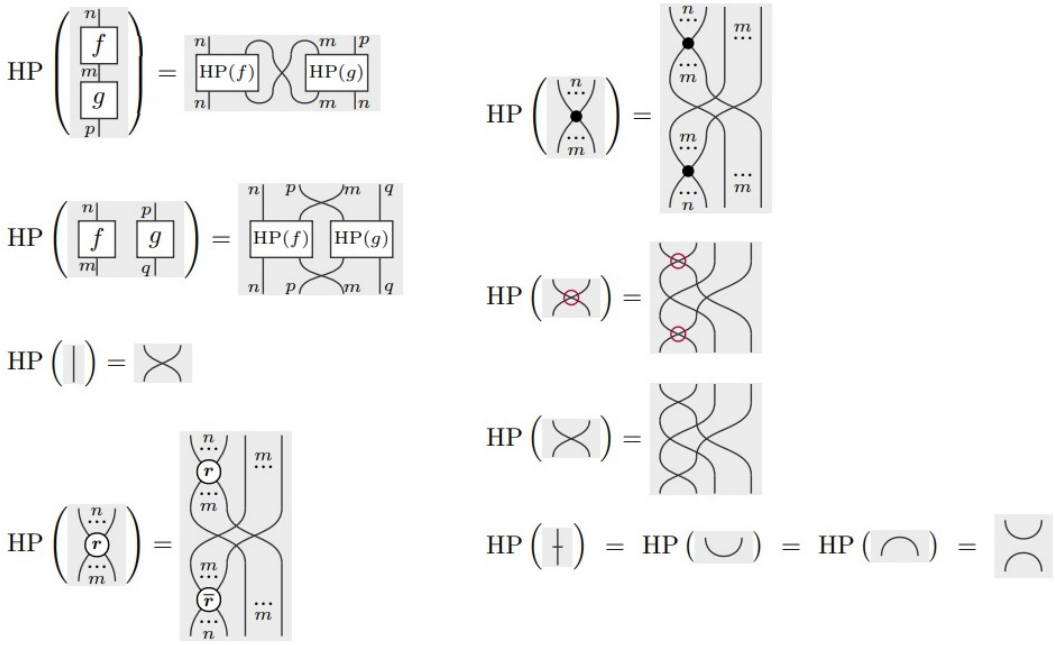


Figure 13 – HP map rules

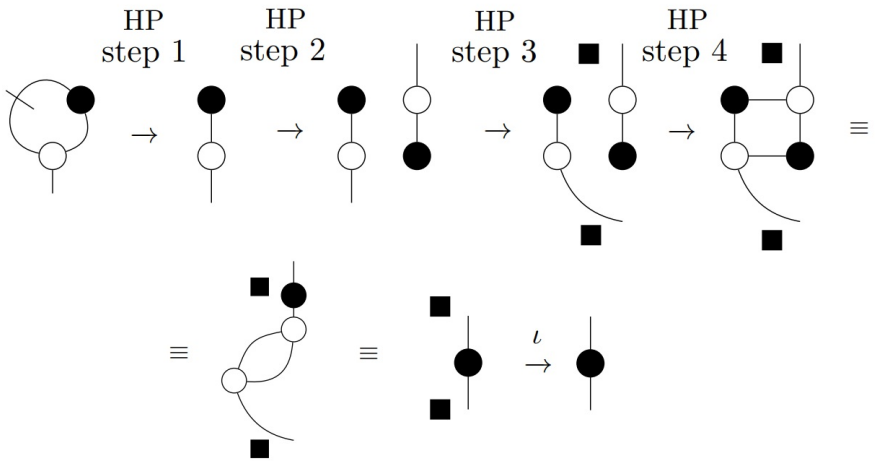
Note that, given a diagram with n inputs and m outputs in ZW†-Calculus, when applying the HP map, the equivalent diagram in ZW-Calculus remains with $2n$ inputs and $2m$ outputs. Also, note that in the representation of the HP(D) map, the diagram inputs are always on the left and the outputs on the right.

There is a more intuitive algorithm to apply the HP map in a ZW†-Calculus diagram $D^\dagger$, which is described as follows:

1. Take the diagram $D^\dagger$ and remove all wires that have a tick;
2. Copy the remaining diagram (which now has no ticks) upside down and with the conjugated values (in case there are any white dots with complex r). Place both diagrams next to each other. Thus, there will be two diagrams that are not connected, having the nodes of the original diagram (which we will denote by u) and the nodes of the inverted and conjugated diagram (denoted by $\bar{u}$);
3. Swap the outputs of both diagrams;
4. From the original D diagram in ZW†-Calculus, whenever there was a tick between nodes u and v, draw a wire between nodes u and $\bar{v}$ and v and $\bar{u}$ in the resulting diagram.

Finally, from the ZW-Calculus diagram obtained through the HP map, to find the equivalent operator, it is necessary to apply the ι operator. It is defined only considering that the diagram inputs are the wires that are above the nodes and the outputs are the ones below the nodes. In other words, ι is a superoperator that maps an operator with $2n$ inputs and $2m$ outputs into an operator with $n+m$ inputs and $n+m$ outputs.

To illustrate the application of both maps, let's demonstrate that the following state in ZW†-Calculus is equivalent to Pauli-X in ZW-Calculus. Note that, from step 3 of the algorithm to find the equivalent diagram after applying the HP map, a symbol of a black square was added. This symbol is not an operator, but a notation to indicate that what is on the left of the axis of the two squares are inputs, and what is on the right of the axis are outputs.



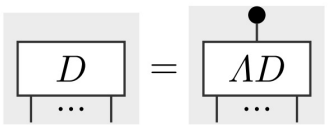
Note that, from step 3 of the algorithm to find the equivalent diagram after applying the HP map, a symbol of a black square was added. This symbol is not an operator, but a notation to indicate that what is on the left of the axis of the two squares are inputs, and what is on the right of the axis are outputs.

2.3.3 Controlled diagrams and Hamiltonian composition

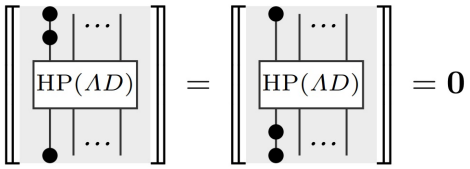
Our main goal is to be able to simply represent the Hamiltonian that describes the behavior of the neutral atom quantum computer using ZW†-Calculus. As seen earlier, we have the ability to easily perform the tensor product and diagram composition. However, to perform diagram summation intuitively, we need to define the concept of *Controlled Diagram*.

Definition (Controlled Diagram): Let $D : 0 \to n$ be a state in ZW†-Calculus (an operator in ZW-Calculus). Now, let $\Lambda D : 1 \to n$ be a superoperator in ZW†-Calculus such that:

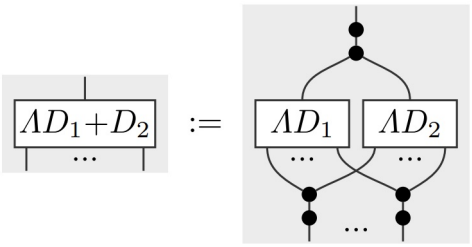
— if the qubit $|0\rangle$ is applied to the input of ΛD, the output will be the state $|0...0\rangle\langle 0...0|$ in ZW†-Calculus;
— if the qubit $|1\rangle$ is applied to the input of ΛD, the obtained diagram can be rewritten into D. This way, if we want to recover the D diagram from ΛD, we just need to apply qubit $|1\rangle$ in the input:



— if the qubits $|0\rangle$ and $|1\rangle$ are both applied in the $HP(\Lambda D)$ operator (one on each input), the result should be the null map $\overrightarrow{0}$:



The advantage of using controlled diagrams is that there is a simple way to perform the sum of two diagrams ΛD_1 and ΛD_2 (VILMART, 2023), expressed below:



Furthermore, the tensor product of two controlled diagrams can be easily represented through the following construction:

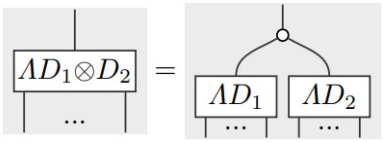


Figure 14 shows some controlled diagrams commonly used for the composition of Hamiltonians (VILMART, 2023).

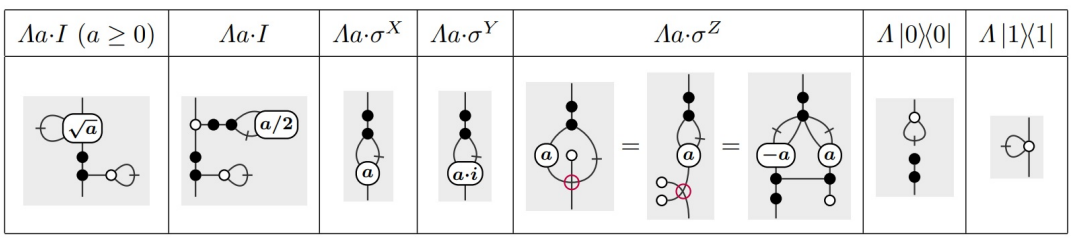


Figure 14 – List of Controlled Diagrams

Finally, given a control diagram ΛD, it is possible to find the diagram $\Lambda a \cdot D$, for $a \in \mathbb{R}$, plugging the following diagram into the control input:

Figure 15 – Product of a controlled diagram by a real number

Based on this framework, every Hamiltonian that follows the form of equation 5, even time dependent, can be represented in the form of sums and tensor products of control diagrams, and can be simplified based on the axioms of both ZW and from ZW†.

$$\mathcal{H}(t) = \sum_i a_i(t) D_i, a_i \in \mathbb{R} \tag{5}$$

2.3.4 pW-fragment and Graph-State

When working with graphical language, we can define a subset of diagrams (commonly called *fragments*) such that a diagram belongs to that fragment if and only if it respects a certain set of properties. We can define the *planar* W-Calculus fragment of ZW-Calculus, or pW-Calculus, the set of diagrams that respect the following properties::

— They don't have the usual swap (which characterizes the name of the fragment):

— Every white node must have degree equal to 2.

Thus, the set of axioms that defines the pW fragment can be seen in figure 16. This set of axioms also guarantees the universality and completeness of the fragment (CARETTE et al., 2023b).

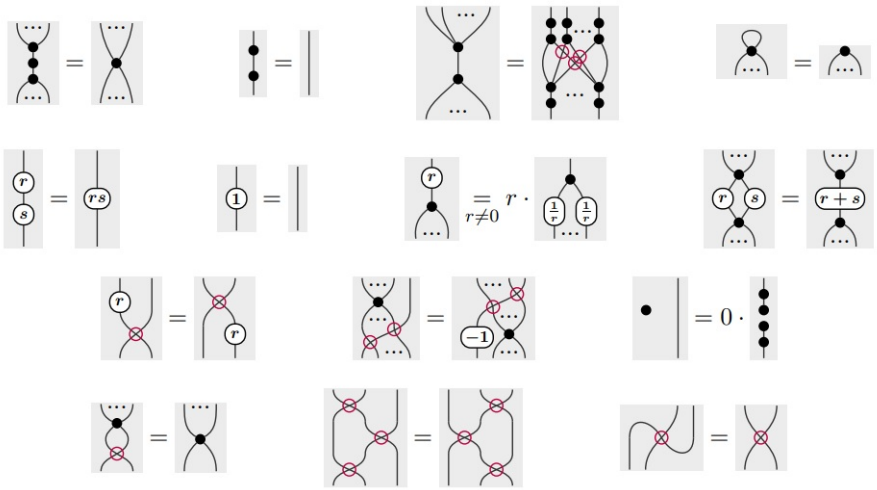


Figure 16 – Axioms for the pW fragment

Note that the fermionic swap is inside the fragment as it is just a synthetic representation of a planar diagram.

We can define a normal form for diagrams belonging to the pW fragment, called Graph-state, such that every diagram D belonging to the fragment can be rewritten using the axioms

of the fragment into a diagram in the Graph-state form. To define this normal form, we will first need the formal definition of boundary nodes and internal nodes (CARETTE et al., 2023b).

Definition (Boundary Nodes and Internal Nodes): A black node is a *type 1 boundary node* if it is connected directly to an output. A black node is a *type 0 boundary node* if it is connected directly to a binary type 1 boundary node. A black node is called an *internal node* if it is not a boundary node.

From now on, we will consider all diagrams as states, since there is always an isomorphism between a diagram and a state, given by the transformation illustrated in figure 8. Thus, we have the following definition for a graph-state in pW-Calculus.

Definition (Graph-State): A graph-state in pW-Calculus is a state in pW-Calculus such that:

— There are no internal black nodes;
— No output wire crosses another wire;
— All white nodes with $r \neq 1$ must be connected to exactly 2 boundary nodes;
— All wire crossings are fermionic swaps, and there can be fermionic swaps between 2 boundary nodes or between a white node with $r \neq 1$ and a boundary node.

In figure 17 we can see an example of a diagram in graph-state form on the left, and on the right a diagram that is not in graph-state (CARETTE et al., 2023b).

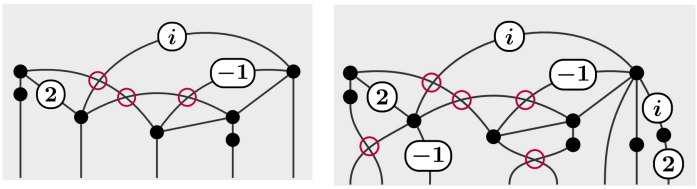


Figure 17 – pW-Calculus example diagrams

The following figure illustrates a strategy for rewriting any pW-Calculus diagram D in its graph-state form (CARETTE et al., 2023b).

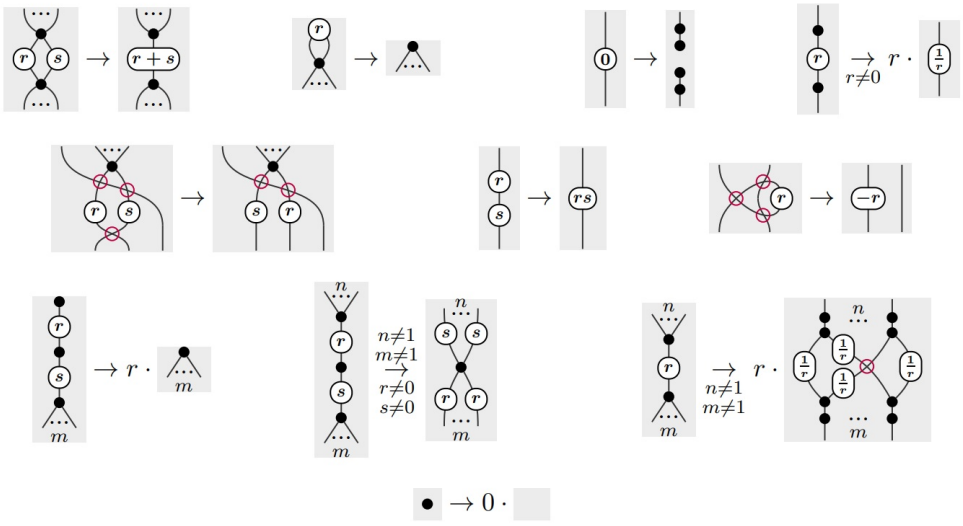


Figure 18 – Rewrite strategy for pW-Calculus diagrams

That way, we can characterize pW-fragment as a subset of ZW-Calculus diagrams for which there are efficient rewriting and simplification rules.

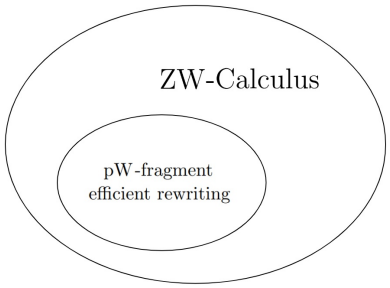


Figure 19 – ZW-Calculus Venn diagram

3 Results

The main problem to be solved can be formalized as follows:

Given a quantum algorithm that maps n qubits into n qubits, and is represented by a Hamiltonian $\hat{H}$, find an atom geometry, a Rabi frequency and a detuning frequency for each atom such that the Hamiltonian H of this neutral atom quantum computer gets close enough to $\hat{H}$.

In terms of graphical language, given that every Hamiltonian can be represented by a diagram, and knowing that the Hamiltonian of the neutral atom quantum computer has a structure given by the equation 4, the problem can be reformulated as:

Given a diagram of a Hamiltonian $\hat{H}$ in ZW†-Calculus, and given the diagram H representing the Hamiltonian of the neutral atom quantum computer, find the parameters of H so that both diagrams are equivalent.

The way in which this problem can be solved is summarized in Figure 20.

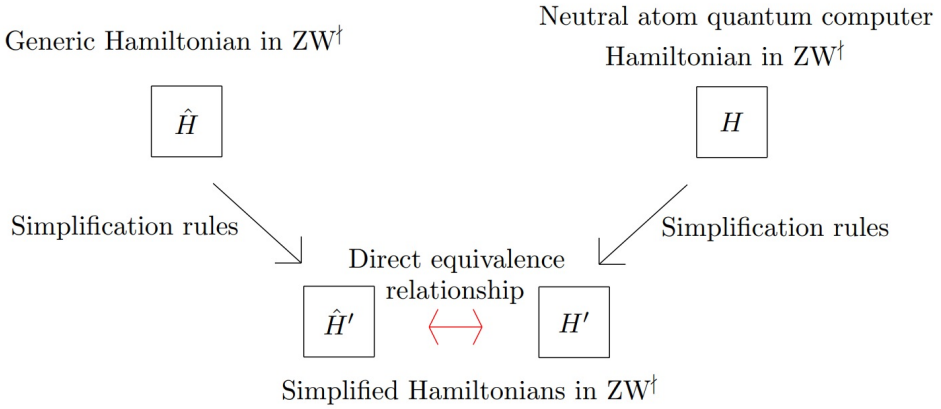


Figure 20 – Resolution path

This section will present the results obtained regarding the representation of the Hamiltonian of the neutral atom quantum computer using ZW†-Calculus, in addition to the simplification of the diagram for the cases of 1, 2 and n qubits. Finally, a fragment of ZW†-Calculus will be defined, pW†-Calculus, which can be used to efficiently rewrite diagrams.

3.1 ZW† diagram for a one-qubit system

In the case of a neutral atom quantum computer with 1 qubit, there will be no interactions between 2 or more atoms. Hence, the equation 4 can be reduced to the equation 6.

$$\frac{\mathcal{H}(t)}{\hbar} = \frac{\Omega(t)}{2}\sigma_x - \delta(t)\hat{n} \tag{6}$$

In section 2.1 it was said that when working with a neutral atom quantum computer, it is normal to use the convention that $|1\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|0\rangle = \begin{bmatrix}0\\1\end{bmatrix}$. This happens because, in addition to allowing us to write that $\hat{n}_i = \frac{\mathbb{I}+\sigma_z^i}{2}$, we also have $\hat{n} = |0\rangle\langle 0|$, which allows us to use the controlled diagram of the operator $|0\rangle\langle 0|$, illustrated in Figure 14, to represent the Hamiltonian of our system, which is simpler than using the controlled diagram of $|1\rangle\langle 1|$.

In this way, the equation 6 can be rewritten as:

$$\frac{\mathcal{H}(t)}{\hbar} = \frac{\Omega(t)}{2}\sigma_x - \delta(t)|0\rangle\langle 0| \tag{7}$$

Using the equational theory shown in section 2.3.3 and the controlled diagrams illustrated in figure 14, we can represent the Hamiltonian of equation 7 as:

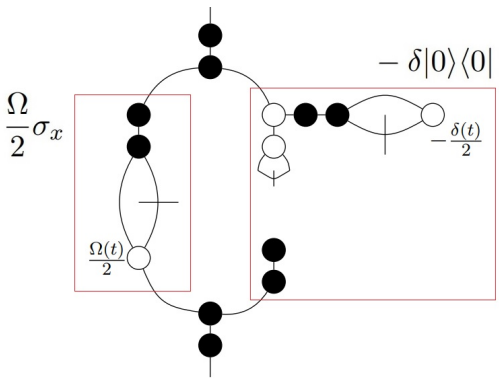


Figure 21 – One qubit diagram

As we want the diagram D representing the Hamiltonian, and not the controlled diagram ΛD, we can apply the qubit $|1\rangle$ to the input of the diagram. The sequence of simplifications of this diagram can be seen below:

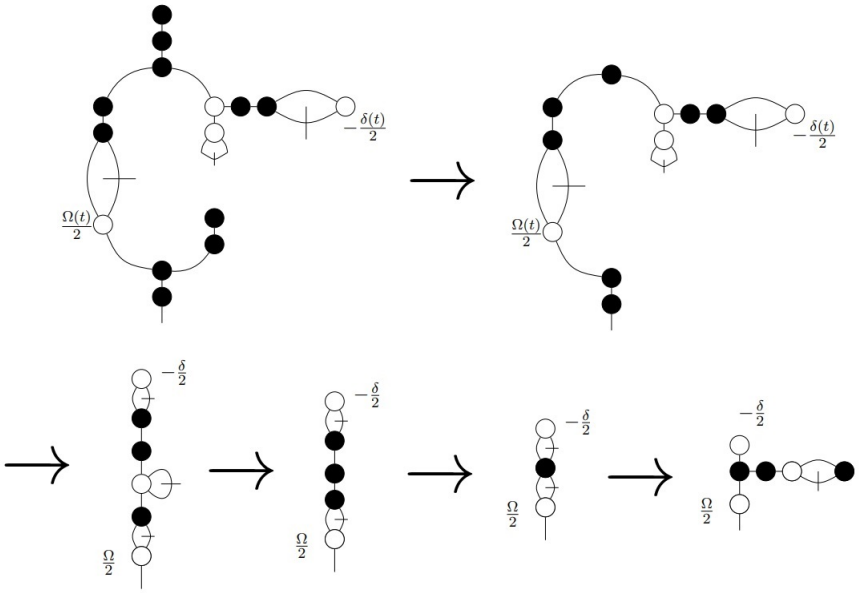


Figure 22 – One qubit diagram simplification

Suppose we wanted to simulate a NOT gate using the neutral atom quantum computer. It is known that the diagram in ZW-Calculus that represents the NOT gate is:

Thus, comparing the Not gate diagram and the final diagram illustrated in figure 22, we know directly that $\delta = 0$ and that $\Omega = 2$, without having to perform the term-to-term comparison of the Hamiltonians, which would have a complexity of $2^2 = 4$ (2^n for n qubits).

3.2 ZW† diagram for a two-qubits system

For two qubits there is the addition of the interaction term between them, and the equation 4 can be written as 8. Note that the subscripts indicate which is the respective qubit in which the Pauli matrix is being applied.

$$\frac{\mathcal{H}(t)}{\hbar} = \frac{\Omega_1(t)}{2}\sigma_1^x + \frac{\Omega_2(t)}{2}\sigma_2^x - \delta_1(t)|0\rangle\langle 0|_1 - \delta_2(t)|0\rangle\langle 0|_2 + \frac{C_6}{|x_1 - x_2|^6}\left(|0\rangle\langle 0|_1 \otimes |0\rangle\langle 0|_2\right) \tag{8}$$

Using the same approach as in section 3.1, we can represent the Hamiltonian of equation 8 as:

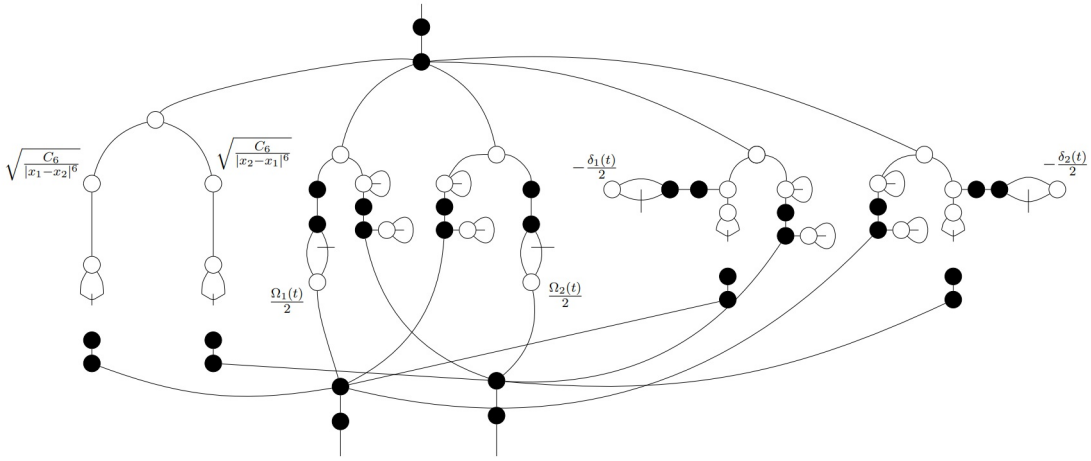


Figure 23 – Two qubits diagram

Applying the qubit $|1\rangle$ to the input of the diagram and simplifying the diagram, and simplifying it with the rules shown in section 2.3.2, the final diagram can be seen in Figure 24. The complete proof of the simplification can be seen in the appendix B.

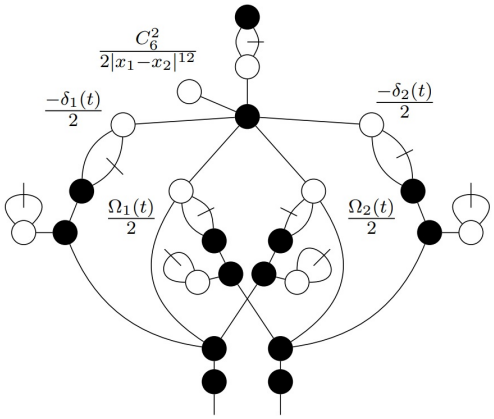


Figure 24 – Two qubit simplified diagram

Note that, from the diagram illustrated in figure 24, it is not trivial which simplification rule should be applied to make the diagram simpler. More, there is no way to say that it is possible

to continue simplifying this diagram. So, one way to prove whether this diagram can still be simplified (or not) is to extend the notion of the planar fragment from ZW to the ZW†, which will be presented further on.

3.3 ZW† diagram for a n-qubit system

For the case of n qubits, the diagram that represents the Hamiltonian of the system can be seen in figure 25, in which the controlled diagrams of the identity were simplified to focus only on the elements shown in the equation 4.

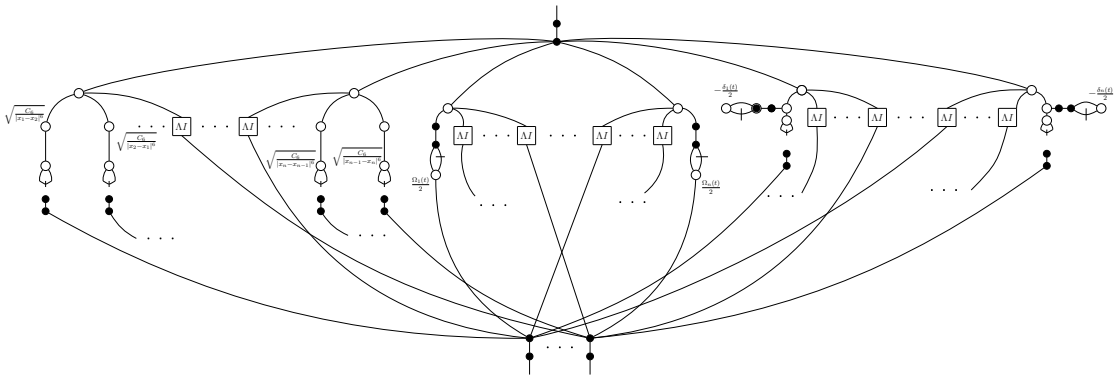


Figure 25 – n-qubit diagram

It is interesting to note that this diagram is composed of a maximum of $2n^2 + \frac{n^2(n-1)}{2}$ controlled diagrams, which may be smaller if it is considered that the Rydberg blockade effect excludes some states from the system. In this case, the term $\frac{n^2(n-1)}{2}$ must be replaced by the cardinality of the set of independent sets of the graph that represents the initial geometry of the atoms.

The advantage of using this representation is that as the number of qubits n increases, the number of controlled diagrams used increases polynomially, and not exponentially like the matrix representation of the Hamiltonian, which grows with the rate 2^n. This becomes a significant advantage from $n = 9$.

3.4 pW†-fragment

It is known that there are efficient rules for rewriting and simplifying diagrams in the planar fragment of ZW-Calculus (CARETTE et al., 2023b). Given the difficulty of simplifying diagrams in ZW†-Calculus, the natural approach is to define the planar fragment for diagrams with † and define a set of rewriting rules that is efficient and ensures that is always possible to write the diagrams in normal form.

Similarly to the definition of the pW-fragment, we define the planar W†-Calculus fragment of ZW†-Calculus, or pW†-Calculus, to be the subset of all diagrams in ZW†-Calculus that respect the following properties:

— They don't have the usual swap;
— Every white node must have degree equal to 2.

Hence, the set of diagrams that belong to pW†-fragment is equivalent to the set of diagrams that can be written from the following 4 generators:

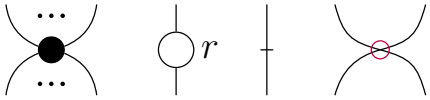


Figure 26 – pW†-Calculus generators

The set of axioms that define pW†-fragment is given by the axioms of pW-fragment, shown in figure 19, with the addition of the following axioms:

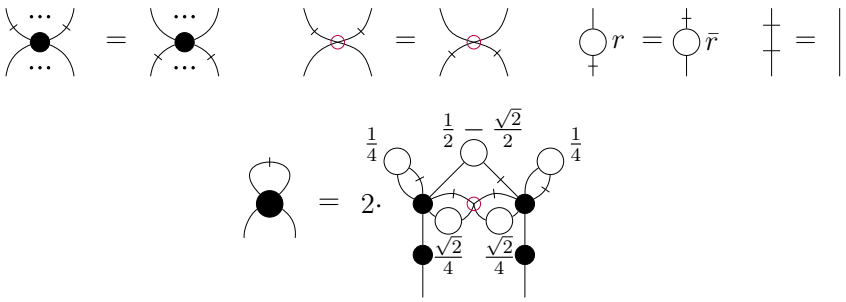


Figure 27 – pW†-Calculus additional axioms

The last axiom shown in figure 27 is not trivial, and can be deduced from the normal form of pW†-fragment, which will be introduced later on.

We also consider as an axiom the following natural property, which says that it is always possible to shift fermionic swaps between diagrams that are composed of combinations of fermionic swaps without "tick" and with "tick" on opposite edges:

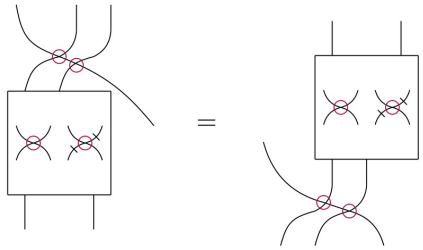


Figure 28 – Natural properties of fermionic swap

To prove that this set of axioms allows efficient rewriting of diagrams in pW†, it is necessary to prove that:

— Given an exponential number of fermionic swaps (with and without "tick"), it is always possible to reduce it to a constant number of fermionic swaps;
— It is always possible to shift a fermionic swap (with or without a "tick") across a white, a black node and another fermionic swap.

From the natural properties of fermionic swap and knowing that a sequence of 2 fermionic swaps without "ticks" results in identity (as shown in figure 9), we can show that, given any combination of 3 or more fermionic swaps , we can reduce them to a maximum of 2 fermionic swaps with "tick":

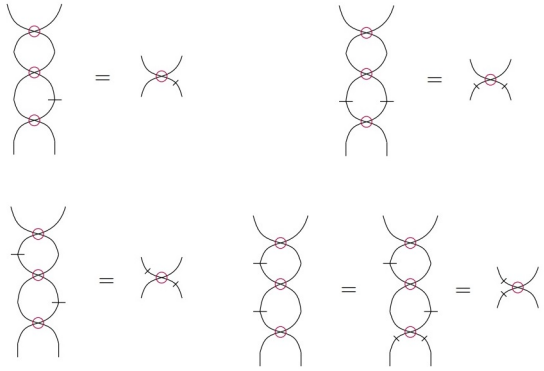


Figure 29 – Reduction in the number of fermionic swaps

To prove that we can always shift fermionic swaps between diagrams, we start by proving that it is always possible to shift fermionic swaps between fermionic swaps. All possible cases are illustrated in the following figure:

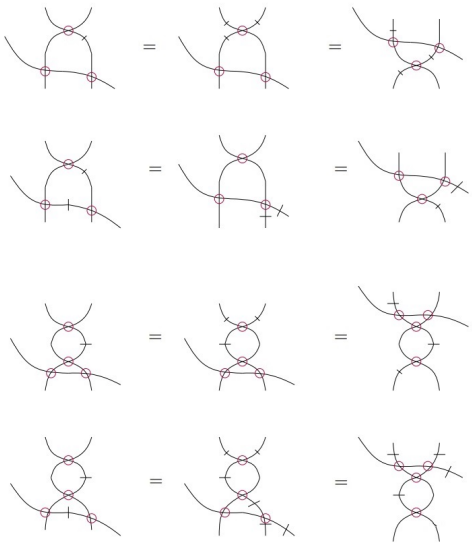


Figure 30 – Fermionic swap shift between other fermionic swaps

Next, we prove that it is always possible to traverse a fermionic swap on a black and white node:

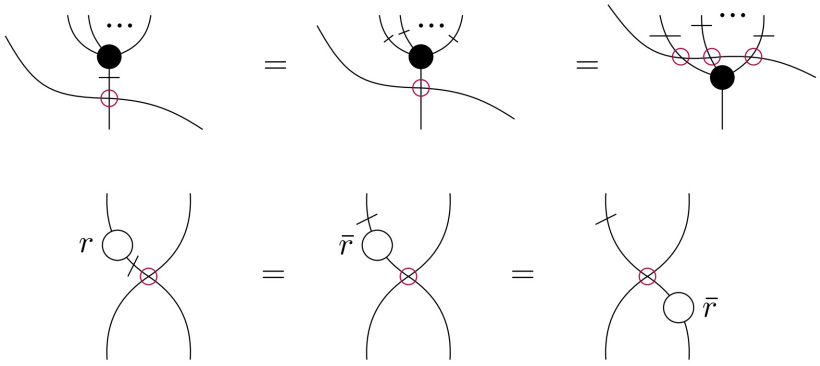


Figure 31 – Fermionic swap shift between black and white nodes

Similarly to pW-fragment, we can define our Graph-state in pW†-Calculus from the following definition:

Definition (Graph-state in pW†-Calculus): A graph-state in pW†-Calculus is a state in pW†-Calculus such that:

— There are no internal nodes;
— No output wires crosses another wire;
— All white nodes with $r \neq 1$ must be connected to exactly 2 boundary nodes with simple wires or wires with "tick";
— All wire crossings are fermionic swaps, and there can be fermionic swaps between 2 boundary nodes or between a white node with $r \neq 1$ and a boundary node.

From (CARETTE et al., 2023b), we know that to prove that every diagram of a fragment can be written in the form of a graph state, it is necessary to prove that it is always possible to remove the internal nodes. We can separate these internal nodes into 3 types: a binary black node, a black node with degree other than 2 and a black node with self-loop.

For the binary black node, as we can always shift the "tick", the fermionic swap and the white nodes through it, it is always possible to perform the following simplification:

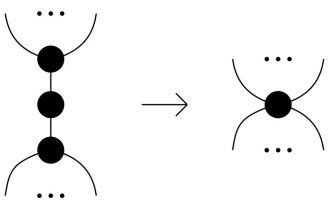


Figure 32 – 3-black node simplification

For the case of a black node with degree different from 2, but without self-loop, it is always possible to apply the bi-algebra rule (axiom *bw* shown in figure 9) to remove the internal nodes. The following figure illustrates this procedure for a diagram with all possible combinations of white nodes, wires with "tick" and fermionic swaps:

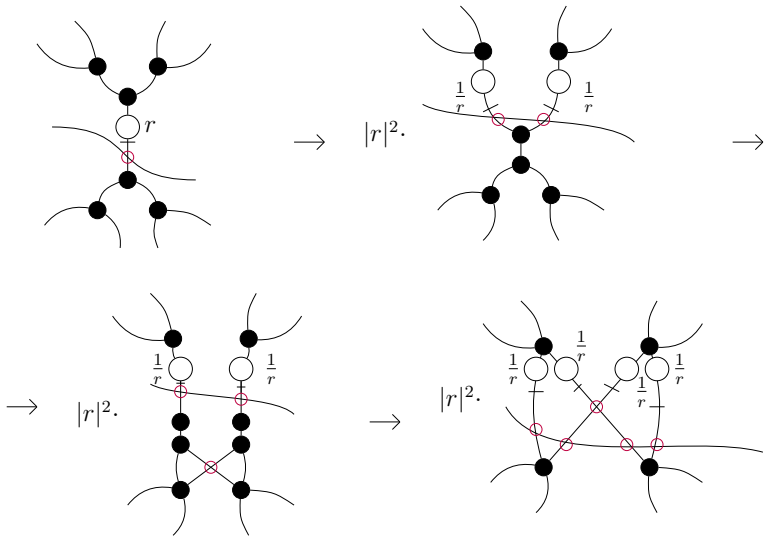


Figure 33 – Simplification example in pW†-Calculus

The only case that is not directly covered by the relationships shown in figure 31 or by the

pW-fragment diagram rewriting strategy (figure 18) is precisely the case of the black node with an self-loop + †, which makes it necessary to add the last axiom of figure 27.

In pW-fragment, if there is a black node with a self-loop, it can be rewritten as just the black node, without that specific edge (as shown in figure 16), which does not happen if there is a self-loop with "tick":

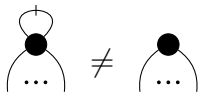


Figure 34 – pW†-fragment self-loop + †

It is not trivial to prove that this type of node can be removed from a diagram if it is an internal node. Let's prove that it is possible to remove the black node with self-loop + † illustrated in figure 34 for the case $m = 2$ (two outputs):

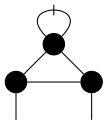


Figure 35 – Self-loop + † internal node

Consider a generic 2-qubit diagram of the pW†-fragment that is in graph-state form. Thus, it must be in the format shown in figure 36. We can consider this as the only case, without loss of generality, since the addition of type 0 boundary nodes only changes the position of the terms of the equivalent operator of the diagram.

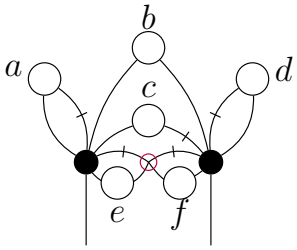


Figure 36 – Two-qubit graph state representation in pW†-Calculus

Also, note that the graph-state for 2 qubits has only one fermionic swap, not two in sequence. This is because the maximum number of fermionic swaps with † in which each pair of end edges are connected to the same black node is equal to 1. This can be proved as shown in figure 37 (all other cases where the "ticks" are positioned differently are equivalent, and this proof is also valid if there are white nodes between the fermionic swaps).

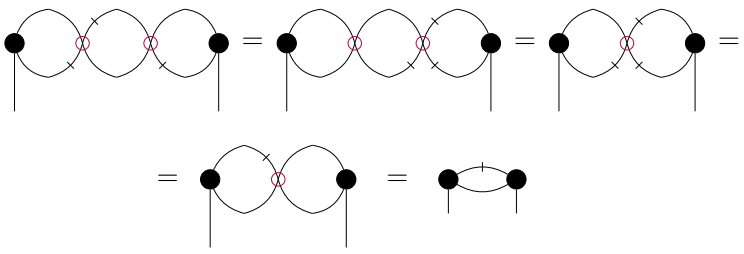


Figure 37 – Two fermionic swaps simplification

Considering $a, b, c, d, e, f \in \mathbb{C}$, and applying the maps HP and ι to the diagram illustrated in figure 36 to find the equivalent diagrams in ZW-Calculus, we obtain the following operator that represent the diagrams:

$$\begin{bmatrix} 2b^2+2c^2+8ad+4ce+2e^2+4(c-e)f+2f^2 & 0 & 0 & 2b \\ 0 & 4d & 2c+2e+2f & 0 \\ 0 & 2c+2e+2f & 4d & 0 \\ 2b & 0 & 0 & 2 \end{bmatrix}$$

Doing the same thing in the diagram shown in figure 35, we obtain the following operator:

$$\begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

By equating both matrices and solving the system of non-linear equations, we obtain the values shown in the last axiom of figure 27. Note that, since the equivalence between the diagrams has not been proven from the rewriting rules of the pW†-fragment, and neither has the completeness of this fragment been proven, it is necessary to add such a relation as an axiom of the fragment.

3.5 Neutral atom quantum computer diagram approximation in pW†-Calculus

It is important to note that pW†-Calculus is limited to diagrams whose white nodes have degree equal to 2, which does not occur in the case of diagrams representing the Hamiltonian of the neutral atom quantum computer, which also have connected conventional swaps to the outputs. Thus, the following approach should be used to take advantage of the efficient pW†-fragment rewriting rules for simplifying generic ZW†-Calculus diagrams:

— Use the following ZW†-Calculus rules to simplify white nodes with self-loop + $\dagger$. If it is not possible to continue simplifying the diagram after this rewrite, extend the definition of graph-state to consider this white node as a "virtual output";

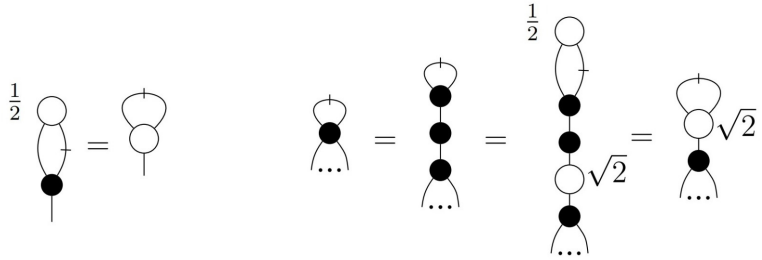


Figure 38 – ZW†-Calculus white node with self-loop + $\dagger$ simplification

— Consider a white node with $r \neq 1$ of degree 1 as a "virtual output";
— If possible, simplify white nodes of degree greater than 3 using general ZW†-Calculus rules;
— Extend the graph state concept to allow usual swaps under specific conditions.

The main mechanisms that must be discovered to make this framework efficient will be discussed in the future work section.

4 Conclusion

Finding the values for the parameters of a neutral atom quantum computer such that the resulting Hamiltonian of the system approximates a desired algorithm is a computationally very costly task, and has a minimum complexity of 2^n, for a computer with n qubits .

During this internship, we focused on solving this problem from an approach using graphical languages, which are capable of representing operators with a polynomial number of diagrams.

We develop a part of the theory needed to efficiently represent the superoperators that approximate the Hamiltonian of a neutral atom quantum computer, representing such Hamiltonians in the form of diagrams and expanding the pW-Calculus concept to diagrams that represent superoperators.

The obtained results generalize the cases of diagrams written in pW†-Calculus to diagrams without black nodes with self-loop $+\dagger$ and also illustrate an efficient writing rule for the case of black nodes with self-loop $+\dagger$ and with two outputs.

4.1 Future work

Future work can be divided into two groups:

Programming work and development: Performing the following tasks would be of fundamental help in making diagram analysis in ZW-Calculus more efficient:

— Create a code (or a library) that allows the declaration of diagrams in ZW-Calculus efficiently, and that checks if two diagrams are equivalent in polynomial time;

— Creation of a mechanism that simplifies diagrams in ZW and ZW†-Calculus, showing the chosen simplification steps. This would be extremely important because applying some simplification rules in an incorrect order can lead to an exponential growth of the diagram components.

Graphical language work and development: The main developments required to be able to efficiently rewrite and compare diagrams in ZW†-Calculus involve:

— Prove that it is possible to rewrite any diagram in pW†-Calculus with black nodes of degree n with self-loop $+\dagger$ in the form of a graph-state. This is equivalent to finding a more general axiom for the last axiom shown in figure 27 by simplifying the following diagrams to an arbitrary number m of outputs:

Figure 39 – Black node of degree n with self-loop $+\dagger$ case

— Prove the completeness of pW†-fragment. If it is true, the last axiom of figure 27 could be removed and considered as a simplification rule;

— Prove that pW†-fragment + *degree 1 white nodes* is equivalent to ZW†-Calculus. If this is true, a general form of graph-state could be defined in a way that considers a white node of degree 1 as a "virtual output", allowing an efficient rewriting of diagrams in ZW†-Calculus. In this case, the new graph state would have the following format:

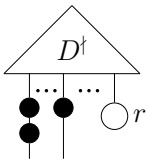


Figure 40 – Possible extended graph-state in ZW†-Calculus

5 References

CARETTE, T.; HOFFREUMON, T.; LARROQUE, É.; VILMART, R. Complete graphical language for hermiticity-preserving superoperators. *arXiv preprint arXiv:2302.04212*, 2023.

CARETTE, T.; MOUTOT, E.; PEREZ, T.; VILMART, R. Compositionality of planar perfect matchings. *arXiv preprint arXiv:2302.08767*, 2023.

COECKE, B.; DUNCAN, R. Interacting quantum observables. In: SPRINGER. *International Colloquium on Automata, Languages, and Programming*. [S.l.], 2008. p. 298–310.

COECKE, B.; KISSINGER, A. The compositional structure of multipartite quantum entanglement. In: SPRINGER. *International Colloquium on Automata, Languages, and Programming*. [S.l.], 2010. p. 297–308.

NIELSEN, M. A.; CHUANG, I. L. Quantum computation and quantum information. *Phys. Today*, v. 54, n. 2, p. 60, 2001.

VILMART, R. Technical report: Graphical language for the representation and manipulation of hamiltonians. In: . [S.l.: s.n.], 2023.

A Hamiltonian approximation using linear programming

To justify the problem solving approach via simplification of ZW†-Calculus diagrams, the problem of approximating a given Hamiltonian $\hat{H}$ through the Hamiltonian H of the neutral atom quantum computer was solved through modeling it as a linear optimization problem. This appendix will detail this approach and the difficulties in using this method.

This problem was solved using the CPLEX library for linear programming problems in Julia. The result was compared with some Hamiltonians generated through the BloQade library, from the company QuEra, which allows the simulation of a neutral atom quantum computer in Julia. All the detailed code, from the creation of the optimization function, to the function that generates the Hamiltonian of the neutral atom quantum computer, can be consulted through the following link: https://github.com/joaquim-gaspar/Hamiltonian_Simulation.git

As said in the Result section, the problem can be formalized as:

Given a quantum algorithm that maps n qubits into n qubits, and is represented by a Hamiltonian $\hat{H}$, find an atom geometry, a Rabi frequency, and a detuning frequency for each atom such that the Hamiltonian H of this neutral atom quantum computer get close enough to $\hat{H}$.

Mathematically, we have the following optimization problem:

$$\begin{array}{lll} min & dist(H, \hat{H}) & \\ \text{s.t.} & r(i,j) = r(j,i) & \forall i,j \in \mathcal{G} \\ & r(i,j) \geq 0 & \forall i,j \in \mathcal{G} \\ & r(i,i) = 0 & \forall i \in \mathcal{G} \\ & \delta_i \in \mathbb{R} & \forall i \in \mathcal{G} \\ & \Omega_i \in \mathbb{R}^+ & \forall i \in \mathcal{G} \end{array} \tag{9}$$

where $\mathcal{G}$ represents the set of all atoms, Ω_i and δ_i are the applied Rabi frequency and detunning frequency of the atom i, and $r(i,j)$ is the distance between atoms i and j. Finally, $\hat{H}$ is a Hamiltonian representing a certain quantum algorithm, and H is the Hamiltonian of the neutral atom quantum computer, described by the equation 4.

Note that no physical constraints regarding the physical limits of Rabi frequencies, detunning frequencies and the geometry of atoms have been added. However, the problem can be easily modified by adding such limits in the form of inequalities.

The simplest example of a Hamiltonian that can be simulated with a neutral atom quantum computer would be a NOT gate applied to 1 qubit, which would be equivalent to exciting an atom with a frequency of Rabi Ω and without detuning frequency, since the Hamiltonian of the system would be represented by equation 10.

$$\frac{\mathcal{H}(t)}{\hbar} = \frac{\Omega}{2}\sigma_x \tag{10}$$

A more interesting example would be to apply the same Rabi frequency Ω to two nearby atoms in order to create an entangled state. This is the example that is resolved in the code illustrated in the given github link. A Hamiltonian of a 2-qubit system that generates an entangled state was placed as input to the optimization problem and the output were the respective Rabi frequencies and detunning frequencies Ω_1, Ω_2, δ_1 and δ_2, beyond the distance between the 2 atoms.

By simulating in time the Hamiltonian found as an output of the optimization problem, we can observe the state evolution of each atom:

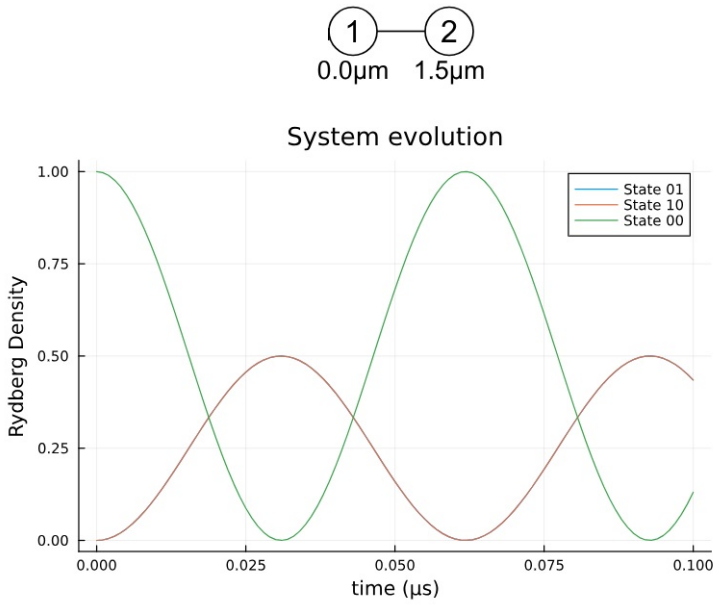


Figure 41 – State evolution

At the top of figure 41 we can obverse the graph equivalent to the geometry of our neutral atom quantum computer. Since atoms 1 and 2 are connected by an edge, they cannot be simultaneously excited. Furthermore, the evolution curve of the system presents a sinusoidal behavior, which is characteristic of the Pauli-X matrix present in the equation 4. Note that the state curve $|01\rangle$ and $|10\rangle$ overlaps in the graph, since the system is completely symmetric, and the probability of atom 1 being excited is identical to that of atom 2. After a period of time $\frac{1}{\Omega}$, we have the entangled state $\frac{|01\rangle+|10\rangle}{\sqrt{2}}$, since both atoms will have exactly a 50% chance of being excited.

However, the explained method cannot be used to simulate any quantum circuit. This is because the neutral atom quantum computer cannot simulate any quantum circuit with n qubits using only n qubits. In other words, it is not universal if it does not use auxiliary qubits (commonly called *ancilla qubit*). A simple example of this would be performing a controlled NOT gate (C-NOT). The matrix representing the C-NOT gate can be seen below (considering that the qubit 2 controls the qubit 1):

$$CNOT = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

Developing the equation 4 for 2 qubits, we would have:

$$\frac{\mathcal{H}(t)}{\hbar} = \frac{\Omega(t)}{2}\sigma_1^x + \frac{\Omega(t)}{2}\sigma_2^x - \delta(t)|1\rangle\langle 1|_1 - \delta(t)|1\rangle\langle 1|_2 + \\ + \frac{C_6}{|x_1 - x_2|^6}\left(|1\rangle\langle 1|_1 \otimes |1\rangle\langle 1|_2\right)$$

$$\frac{\mathcal{H}(t)}{\hbar} = \begin{bmatrix} 0 & \Omega_2 & \Omega_1 & 0 \\ \Omega_2 & -\delta_2 & 0 & \Omega_1 \\ \Omega_1 & 0 & -\delta_1 & \Omega_2 \\ 0 & \Omega_1 & \Omega_2 & \frac{C_6}{|x_1-x_2|^6} - \delta_1 - \delta_2 \end{bmatrix} \tag{12}$$

Comparing the structure of 11 and 12, we observe that it would only be possible to approximate this Hamiltonian if Ω_1 could assume 2 different values simultaneously, which is impossible. In that case, the optimization problem would never converge and the proposed method would not work to find the parameters of the neutral atom quantum computer such that the resulting Hamiltonian approximates the given Hamiltonian.

Another problem with this method is that, as the dimension of the Hamiltonian matrix grows exponentially according to the number of qubits, the number of constraints in the optimization problem would also grow exponentially, which would be unfeasible for an algorithm with a large amount of qubits. A solution to this exponential number of constraints would be to disregard the columns of H referring to states that cannot occur due to the Rydberg blockade effect. However, finding the set of independent sets of a graph is an NP-Hard problem, which would be difficult to accomplish in large dimensions.

B Two qubits diagram simplification

Here is the step-by-step simplification of the two qubits diagram, shown in section 3.2: