Physica Scripta

PAPER

Algebraic language for the efficient representation and optimization of quantum circuits

Daniel Escanez-Exposito[1,*], **Pino Caballero-Gil**[1], **Marcos Rodriguez-Vega**[1], **Francisco Costa-Cano**[2] **and Eduardo Sáenz-de-Cabezón**[3]

[1] University of La Laguna La Laguna Tenerife, Spain
[2] Qubitalent S L U Murcia, Spain
[3] University of La Rioja Logroño La Rioja, Spain
[*] Author to whom any correspondence should be addressed.

E-mail: jescanez@ull.edu.es, pcaballe@ull.edu.es, mrodrive@ull.edu.es, francisco.cc@qubitalent.com and eduardo.saenz-de-cabezon@unirioja.es

OPEN ACCESS

RECEIVED
20 September 2024

REVISED
5 December 2024

ACCEPTED FOR PUBLICATION
16 December 2024

PUBLISHED
13 January 2025

Keywords: quantum computing, quantum circuit representation, quantum gate identities, quantum simulation, quantum circuit optimization

Abstract

The use of graphical language in quantum computing for the representation of algorithms, although intuitive, is not very useful for different tasks such as the description of quantum circuits in text environments, the calculation of quantum states or the optimization of quantum circuits. While classical circuits can be represented either by circuit graphs or by Boolean expressions, quantum circuits have until now predominantly been illustrated as circuit graphs because no formal language for quantum circuits that allows algebraic manipulations has so far been accepted. This work proposes a means to represent quantum circuits in a convenient and concise manner, similar to the way Boolean expressions are used in classical circuits. The proposed notation allows the consistent and parameterized description of quantum algorithms, as well as the easy handling of the elements that compose it to achieve powerful optimizations in the number of gates of the circuits. To visualize it, a software implementation of an algebraic quantum circuit framework has been made, which allows describing quantum circuits, as well as their respective state vectors, using the proposed algebraic language.

CCS CONCEPTS

• Theory of computation → Quantum computation theory; • Computing methodologies → Representation of mathematical objects; Quantum mechanic simulation; • Software and its engineering → Formal language definitions.

1. Introduction

Quantum circuits, pioneered by Deutsch in [1], are widely acknowledged as a practical and commonly used approach for illustrating the operations of quantum gates on qubits, ultimately describing the unitary matrix of a quantum computer through graphical representations. Consequently, quantum algorithms and protocols are typically depicted in the format of quantum circuits. This leads to a challenge when dealing with complex quantum algorithms or protocols, as their circuit graphs can quickly surpass manageable sizes, making drawing and manipulation impractical.

In classical computing, circuits can be not only graphically represented but also expressed as Boolean expressions using Boolean gates that implement logic based on truth tables. These Boolean expressions are well-suited for algebraic manipulations. Conversely, the derivation of an algebraic expression for a quantum circuit has been challenging because quantum computing lacks a language analogous to Boolean expressions in classical

computing. The present paper aims to introduce a formal semantics for a novel algebraic language designed to streamline the expression of quantum circuits in a concise manner so that fundamental algebraic laws for quantum circuits can be easily proven, and complex quantum algorithms can be simplified when expressed in this language.

A bibliographic review shows the great interest that the problem arouses, although no solution has been fully accepted to date.

In the work [2], a Mathematica package is presented for the simulation of quantum computing based on the circuit model, providing an interface to specify and draw a quantum circuit and to build the corresponding unitary matrix for quantum computing defined by the circuit. Another method for representing the unitary matrix of a quantum circuit as an algebraic equation is presented in the paper [3]. Note that the present work does not require the use of Mathematica or matrices.

Another close paper is [4], which presents the design of an algebraic language for formally specifying quantum circuits in distributed quantum computing so that using that language, quantum circuits can be represented in a compact way. However, the language proposed in the present work is simpler and closer to simulation programming, which facilitates its implementation. Indeed, this issue is demonstrated through the presentation of a first version of a software implementation of the proposal. In particular, the present notation proposal is more intuitive as it sets the quantum gates in order of execution (from left to right).

The equational theory of quantum circuits is discussed in the paper [5], which presents an axiomatization of the relationship between measurement, qubit initialization and a set of unitary gates. Another recent work [6] proposes the unification of quantum and classical computing in open quantum systems. Also recently, the paper [7] describes the flow of quantum compilation with several NP-hard tasks (problems that are at least as complex as the toughest NP problems, even if they are not in NP or are even undecidable) and proposes algorithms based on Boolean satisfiability to address those computationally complex problems. Note that the three aforementioned recent papers address related but different topics than the one addressed in the present paper.

In the paper [8], the existence of quantum logic gate identities is highlighted to address the connectivity problems of quantum hardware equipment in the Noisy Intermediate-Scale Quantum (NISQ) era. This stage of quantum computing is characterized by quantum processors with a limited number of qubits (currently around a thousand qubits), which are not yet advanced enough to guarantee fault tolerance. Optimisation schemes are proposed in this sense, in order to respect the coupling map of the qubits involved. However, this approach focuses on equivalences between the products of matrices, and not on identities emanating from pre-established rules.

Finally, one of the latest related works is [9] on the equivalence of dynamic quantum circuits. That proposal looks promising, but has only been evaluated on toy examples, with no available software packages for checking yet.

This paper is structured as follows. Section 2 provides a brief introduction to the topic, including the first fundamental notations for states and gates. Section 3 contains several general basic properties, while section 4 focuses on specific properties of the controlled Pauli-X gate. Section 5 introduces a beta software implementation that makes use of the proposed language. Section 6 shows the usefulness of the proposed language by demonstrating a use case that illustrates an effective circuit simplification based on the introduced notation. Section 7 completes this paper with some conclusions and future work. Appendix A includes the longest proofs of the theoretical results. Finally, appendix B further emphasises the operation and importance of the method of applying the rules.

2. Preliminaries

The fundamental information unit within a quantum computing system is the qubit, which is denoted by a unitary vector within a Hilbert space of dimension 2. Multiple qubits can be taken together so that an element of the Hilbert space of n qubits is considered. This case can be seen as the tensor product of the n Hilbert spaces corresponding to each qubit in the system, so that the resulting space dimension is 2^n.

Let $\mathbb{N}$ and $\mathbb{C}$ denote the sets of natural and complex numbers, respectively. The notation $\mathbb{N}_n$ is used to denote the subset of $\mathbb{N}$ that contains the natural numbers smaller or equal than n. In set theory, for any finite set S, its number of elements $|S|$ is called the cardinality of S.

2.1. Kets

As introduced in [10], qubits are normally written in the so-called bra-ket notation.

On the one hand, each qubit is represented as a ket, of the form $|a\rangle$, which denotes a vector a in a vector space V. For instance, the computational basis states are written as $|0\rangle$ and $|1\rangle$, and called ket 0 and ket 1, respectively (see [11]). Thus, any qubit can be described by a linear combination of these two basis states as $|a\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where $\alpha_0,\ \alpha_1 \in \mathbb{C}$ such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

On the other hand, a bra is of the form $\langle f|$ and denotes a linear form f that represents a linear map that maps each vector in V to a number in $\mathbb{C}$.

In this way, when the function $\langle f|$ acts on a vector $|a\rangle$, the result is written as $\langle f|a\rangle$.

Notation 2.1.1. For any qubit $|a_k\rangle = \alpha_{k0}|0\rangle + \alpha_{k1}|1\rangle$, the following notation is used: $\alpha_{k0} = \langle a_k|0\rangle$ and $\alpha_{k1} = \langle a_k|1\rangle$.

For example, for two qubits $|a_1\rangle$ and $|a_2\rangle$, the following can be written:

$$\begin{aligned} |a_1a_2\rangle &= \alpha_{10}|0a_2\rangle + \alpha_{11}|1a_2\rangle \\ &= \alpha_{20}|a_10\rangle + \alpha_{21}|a_11\rangle \\ &= \alpha_{10}\alpha_{20}|00\rangle + \alpha_{10}\alpha_{21}|01\rangle + \alpha_{11}\alpha_{20}|10\rangle + \alpha_{11}\alpha_{21}|11\rangle. \end{aligned} \tag{1}$$

In general, for n qubits $|a_1\rangle\ldots|a_n\rangle$, the composite ket $|a\rangle = |a_1 \cdots a_n\rangle$ can be written as shown below:

$$|a\rangle = \alpha_{k0}|a_1 \cdots a_{k-1}0a_{k+1} \cdots a_n\rangle + \alpha_{k1}|a_1 \cdots a_{k-1}1a_{k+1} \cdots a_n\rangle. \tag{2}$$

Notation 2.1.2. For any ket $|a\rangle = |a_1\ldots a_n\rangle$ of n qubits, the following notation is used: $|a\rangle_u^k = |a_1 \cdots a_{k-1}ua_{k+1} \cdots a_n\rangle$, where $k \in \mathbb{N}_n$ and $u \in \{0, 1\}$.

Consequently, the expression (2) can be written as follows:

$$|a\rangle = \alpha_{k0}|a\rangle_0^k + \alpha_{k1}|a\rangle_1^k. \tag{3}$$

For instance, if $|a_1\rangle$, $|a_2\rangle$ and $|a_3\rangle$ are three qubits, then:

- $|a_1\rangle_0^1 = |0\rangle$ and $|a_1\rangle_1^1 = |1\rangle$.
- $|a_1a_2\rangle_0^2 = |a_10\rangle$.
- $|a_1a_2a_3\rangle = \alpha_{20}|a_10a_3\rangle + \alpha_{21}|a_11a_3\rangle = \alpha_{20}|a\rangle_0^2 + \alpha_{21}|a\rangle_1^2$.
- $|00a_3\rangle_1^3 = |001\rangle$.

2.2. Quantum gates

A quantum gate is a basic quantum circuit operating on a small number of qubits.

For example, a single-qubit gate A can be applied over the first qubit of a ket $|a_1a_2\rangle$ and the result is $A|a_1\rangle \otimes |a_2\rangle$. It can also be applied over the last of both qubits and then the result is $|a_1\rangle \otimes A|a_2\rangle$.

Notation 2.2.1. If A is a single-qubit gate and $|a\rangle = |a_1 \cdots a_n\rangle$ is a ket with n qubits, the following notation is used: $|a_1 \cdots a_hAa_{h+1} \cdots a_n\rangle = |a_1 \cdots a_{h-1}\rangle \otimes A|a_h\rangle \otimes |a_{h+1} \cdots a_n\rangle$, where $1 < h < n$.

For example, if $|a\rangle = |a_1a_2\rangle$ then $|a_1Aa_2\rangle = A|a_1\rangle \otimes |a_2\rangle$ and $|a_1a_2A\rangle = |a_1\rangle \otimes A|a_2\rangle$.

Notation 2.2.2. The effect over the k-th qubit of the ket $|a\rangle = |a_1 \cdots a_n\rangle$ of the quantum gate A can be denoted by: $|a\rangle A_k = |a\rangle_A^k$.

Notation 2.2.3. For any set $\Gamma = \{l_1,\ldots,l_h\} \subseteq \mathbb{N}_n$ and any single-qubit gate A, the following notation is used:

$$\begin{aligned} A_\Gamma &= A_{l_1,\ldots,l_h} \\ &= \textstyle\prod_{k\in\Gamma} A_k \end{aligned} \tag{4}$$

Note that, according to the previous notation, for the specific case of the empty set $\Gamma = \varnothing$, the resulting quantum gate is the identity gate $A_\Gamma = I$.

In conclusion, for example, the action of a single-qubit gate A over the ket $|a\rangle = |a_1a_2\rangle$ can be expressed using four distinct forms:

$$A|a_1\rangle \otimes |a_2\rangle = |a_1Aa_2\rangle = |a_1a_2\rangle A_1 = |a_1a_2\rangle_A^1$$

2.3. Pauli gates

Pauli gates are physical and mathematical operators in quantum computing represented by the well-known Pauli matrices, which are a set of three 2×2 complex matrices that are traceless, Hermitian, involutory and unitary. They are represented by the following matrix expressions: $\mathrm{X}=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$, $\mathrm{Y}=\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$, $\mathrm{Z}=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$. In particular, the identity and Pauli gates I, X, Y, Z can be expressed according to the notation introduced above as follows:

$$|a\rangle I_k = a_{k0}|a\rangle_0^k + a_{k1}|a\rangle_1^k \tag{5}$$

$$|a\rangle X_k = a_{k0}|a\rangle_1^k + a_{k1}|a\rangle_0^k \tag{6}$$

$$|a\rangle Y_k = ia_{k0}|a\rangle_1^k - ia_{k1}|a\rangle_0^k \tag{7}$$

$$|a\rangle Z_k = a_{k0}|a\rangle_0^k - a_{k1}|a\rangle_1^k \tag{8}$$

2.4. Hadamard gate

The Hadamard gate is a well-known single-qubit gate that maps the basis states $|0\rangle$ and $|1\rangle$ to $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$, respectively.

Thus, the Hadamard gate can be expressed using the introduced notation as follows:

$$\begin{aligned}|a\rangle H_k &= a_{k0}|a\rangle_+^k + a_{k1}|a\rangle_-^k \\ &= \tfrac{a_{k0}+a_{k1}}{\sqrt{2}}|a\rangle_0^k + \tfrac{a_{k0}-a_{k1}}{\sqrt{2}}|a\rangle_1^k\end{aligned} \tag{9}$$

2.5. Controlled gates

Controlled gates are applied to kets formed by multiple qubits so that one or more qubits serve as controllers for a specific operation on one or more qubits of the ket.

The simplest controlled gate A over a ket $|a\rangle = |a_1 \ldots a_n\rangle$ has only one control qubit $|a_h\rangle$ and only one target qubit $|a_k\rangle$.

Notation 2.5.1. The effect over the ket $|a\rangle = |a_1 \cdots a_n\rangle$ of the controlled gate A with control qubit $|a_h\rangle$ and target qubit $|a_k\rangle$ can be denoted by:

$$\begin{aligned}|a\rangle A_k^h &= a_{h0}|a\rangle_0^h + a_{h1}|a\rangle_1^h A_k \\ &= a_{h0}|a\rangle_0^h + a_{h1}|a\rangle_{1A}^{hk}\end{aligned} \tag{10}$$

Notation 2.5.2. The effect over the ket $|a\rangle = |a_1 \cdots a_n\rangle$ of a controlled gate A with various control qubits $\Gamma = \{h_1,\ldots,h_l\} \subseteq \mathbb{N}_n$ and various target qubits $\Lambda = \{k_1,\ldots,k_m\} \subseteq \mathbb{N}_n$ can be denoted by: $A_\Lambda^\Gamma = A_{k_1,\ldots,k_m}^{h_1,\ldots,h_l}$.

Note that, according to the previous notation, for the specific case of the empty set $\Gamma = \varnothing$, the resulting quantum gate is $A_\Gamma^\Lambda = A_\Lambda$.

3. General basic properties

Some generic properties of gates in a quantum circuit are proven below.

Proposition 3.1. *Let* A *and* B *be two single-qubit gates. Then, for any two distinct indexes* $h,\ k \in \mathbb{N}_n$

$$A_k B_h = B_h A_k \tag{11}$$

Proof. Let $|a\rangle$ be a ket. For any two distinct indexes $h,\ k \in \mathbb{N}_n$

$$|a\rangle A_k B_h = |a\rangle_A^k B_h = |a\rangle_{AB}^{kh} = |a\rangle_B^h A_k = |a\rangle B_h A_k.$$

□

Proposition 3.2. *Let* A *be a single-qubit gate. Then, for any four indexes* $h,\ k,\ h',\ k' \in \mathbb{N}_n$ *such that* $\mathrm{h} \neq \mathrm{k}$ *and* $h' \neq k'$

$$A_k^h A_{k'}^{h'} = A_{k'}^{h'} A_k^h \tag{12}$$

Proof. Let $|a\rangle$ be a ket. For any four $h,\ k,\ h',\ k' \in \mathbb{N}_n$ such that $h \neq k$ and $h' \neq k'$:

$$\begin{aligned}
|a\rangle A_k^h A_{k'}^{h'} &= a_{h0}|a\rangle_0^h A_{k'}^{h'} + a_{h1}|a\rangle_1^h A_k A_{k'}^{h'} \\
&= a_{h0}a_{h'0}|a\rangle_{00}^{hh'} + a_{h0}a_{h'1}|a\rangle_{01}^{hh'} A_{k'} \\
&\quad + a_{h_1}a_{h'0}|a\rangle_{10}^{hh'} A_k + a_{h_1}a_{h'1}|a\rangle_{11}^{hh'} A_k A_{k'} \\
&= a_{h0}a_{h'0}|a\rangle_{00}^{hh'} + a_{h_1}a_{h'0}|a\rangle_{10}^{hh'} A_k \\
&\quad + a_{h0}a_{h'1}|a\rangle_{01}^{hh'} A_{k'} + a_{h_1}a_{h'1}|a\rangle_{11}^{hh'} A_{k'} A_k \\
&= a_{h'0}|a\rangle_0^{h'} A_k^h + a_{h'1}|a\rangle_1^{h'} A_{k'} A_k^h \\
&= |a\rangle A_{k'}^{h'} A_k^h
\end{aligned}$$

□

More generally, it be can extend this result to sets

Proposition 3.3. *Let* A *and* B *be two single-qubit gates. Then for all* $\Gamma,\ \Lambda,\ \Gamma',\ \Lambda' \subset \mathbb{N}_n$ *sets such that* $\Gamma \cap \Lambda = \varnothing$ *and* $\Gamma' \cap \Lambda' = \varnothing$

$$A_\Gamma B_\Lambda = B_\Lambda A_\Gamma \tag{13}$$

$$A_\Gamma^{\Gamma'} B_\Lambda^{\Lambda'} = B_\Lambda^{\Lambda'} A_\Gamma^{\Gamma'} \tag{14}$$

Proof. Let $\Gamma = \{h_1, \ldots, h_l\}$ and $\Lambda = \{k_1, \ldots, k_m\}$ be two subsets of $\mathbb{N}_n$. Then

$$A_\Gamma B_\Lambda = A_\Gamma B_{k_1} \cdots B_{k_m} = A_{h_1} \cdots A_{h_l} B_{k_1} \cdots B_{k_m}.$$

Since $\Gamma \cap \Lambda = \varnothing$, then $h_l \neq k_1$. Thus, according to (11), the following is true $A_{h_l} B_{k_1} = B_{k_1} A_{h_l}$. The same reasoning is valid for all elements of Γ, so $A_\Gamma B_{k_1} = B_{k_1} A_\Gamma$. Consequently,

$$\begin{aligned}
A_\Gamma B_\Lambda &= A_\Gamma B_{k_1} \cdots B_{k_m} = B_{k_1} A_\Gamma B_{k_2} \cdots B_{k_m} \\
&= \cdots = B_{k_1} \cdots B_{k_m} A_\Gamma = B_\Lambda A_\Gamma
\end{aligned}$$

For the same reason, using the result (12) the equality (14) is obtained. □

Proposition 3.4. *Let* A *be a single-qubit gate such that* AA = I. *Then* $\forall\ h,\ k \in \mathbb{N}_n$

$$A_k A_k = I \tag{15}$$

$$A_k^h A_k^h = I. \tag{16}$$

Proof. Let $|a\rangle$ be a ket and two indexes $h \neq k \in \mathbb{N}_n$. To prove expression (16):

$$|a\rangle A_k A_k = |a\rangle_A^k A_k = |a\rangle_{AA}^k = |a\rangle_I^k = |a\rangle.$$

Analogously, to prove expression (17), according to (10):

$$\begin{aligned}
|a\rangle A_k^h A_k^h &= a_{h0}|a\rangle_0^h A_k^h + a_{h1}|a\rangle_{1A}^{hk} A_k^h \\
&= a_{h0}|a\rangle_0^h + a_{h1}|a\rangle_{1AA}^{hk} \\
&= a_{h0}|a\rangle_0^h + a_{h1}|a\rangle_1^h = |a\rangle.
\end{aligned}$$

□

Proposition 3.5. *Let* $|\mathrm{a}\rangle$ *be a ket and let* A *be a single-qubit gate such that* AA = I. *Then for all* $\Gamma,\ \Lambda \subset \mathbb{N}_n$ *sets such that* $\Gamma \cap \Lambda = \varnothing$

$$A_\Gamma A_\Gamma = I \tag{17}$$

$$A_\Gamma^\Lambda A_\Gamma^\Lambda = I. \tag{18}$$

Proof. Let $\Gamma = \{h_1, \ldots, h_l\}$ and $\Lambda = \{k_1, \ldots, k_m\}$ be two subsets of $\mathbb{N}_n$. According to (11), $A_h A_{h_1} = A_{h_1} A_h \quad \forall\ h \neq h_1$. Consequently,

$$\begin{aligned} A_\Gamma A_\Gamma &= A_\Gamma A_{h_1} \cdots A_{h_l} = A_{h_1} \cdots A_{h_l} A_{h_1} \cdots A_{h_l} \\ &= A_{h_1} A_{h_1} \cdots A_{h_l} A_{h_2} \cdots A_{h_l} = I A_{h_2} \cdots A_{h_l} A_{h_2} \cdots A_{h_l} \\ &= \cdots = A_{h_l} A_{h_l} = I. \end{aligned}$$

□

Proposition 3.6. *Let* A *be a single-qubit gate. Then, for any two sets* $\Gamma,\ \Lambda \subset \mathbb{N}_n$

$$A_\Gamma A_\Lambda = A_{\Gamma \bigcup \Lambda} A_{\Gamma \bigcap \Lambda} \tag{19}$$

Proof. From basic set theory, $\Gamma \cup \Lambda$ can be expressed as $(\Gamma - (\Gamma \cap \Lambda)) \cup (\Lambda - (\Gamma \cap \Lambda)) \cup (\Gamma \cap \Lambda)$. Then, the following is true

$$\begin{aligned} A_\Gamma A_\Lambda &= \prod_{h \in \Gamma} A_h \prod_{k \in \Lambda} A_k \\ &= \prod_{h \in \Gamma - (\Gamma \cap \Lambda)} A_h \prod_{h' \in \Gamma \cap \Lambda} A_{h'} \prod_{k \in \Lambda - (\Gamma \cap \Lambda)} A_k \prod_{k' \in \Gamma \cap \Lambda} A_{k'} \\ &= \prod_{h \in (\Gamma - (\Gamma \cap \Lambda)) \bigcup (\Lambda - (\Gamma \cap \Lambda)) \bigcup (\Gamma \cap \Lambda)} A_h \prod_{k' \in \Gamma \cap \Lambda} A_{k'} \\ &= \prod_{h \in \Gamma \bigcup \Lambda} A_h \prod_{k' \in \Gamma \cap \Lambda} A_{k'} = A_{\Gamma \cup \Lambda} A_{\Gamma \cap \Lambda} \end{aligned}$$

□

Below a theoretical result shows that if a single-qubit gate is applied over a subset of qubits that are eigenstates of the gate with eigenvalue -1, then the result coincides with that of the result with another subset with the same cardinality parity (see [11]). On the other hand, if this latter subset has a different cardinality parity, then the result coincides with the product of the first result with the eigenvalue.

Proposition 3.7. *Let* A *be a single-qubit gate. Let* $|\mathrm{a}\rangle$ *be a ket and* $\Gamma,\ \Lambda \subseteq \mathbb{N}_n$ *such that* $\forall \mathrm{k} \in \Gamma \cup \Lambda$, $|\mathrm{a}\rangle \mathrm{A_k} = -|\mathrm{a}\rangle$. *Then*

$$|a\rangle A_\Gamma = (-1)^{|\Lambda| - |\Gamma|} |a\rangle A_\Lambda. \tag{20}$$

Proof. If $\Gamma = \{k_1, \ldots, k_l\}$ where $l = |\Gamma|$, then

$$\begin{aligned} |a\rangle A_\Gamma &= |a\rangle \prod_{k \in \Gamma} A_k = |a\rangle A_{k_1} A_{k_2} \ldots A_{k_l} \\ &= (-1)|a\rangle A_{k_2} \ldots A_{k_l} = \cdots = (-1)^{|\Gamma|} |a\rangle. \end{aligned}$$

Analogously, $|a\rangle A_\Lambda = (-1)^{|\Lambda|} |a\rangle$. Then

$$\begin{aligned} |a\rangle A_\Gamma &= (-1)^{|\Gamma|} |a\rangle = (-1)^{|\Gamma| - |\Lambda|} (-1)^{|\Lambda|} |a\rangle \\ &= (-1)^{|\Gamma| - |\Lambda|} |a\rangle A_\Lambda. \end{aligned}$$

□

4. Specific properties of the controlled pauli-*X* gate

In this section, some specific properties involving the controlled Pauli-*X* gate are demonstrated.

Proposition 4.1. *Let* $h,\ k \in \mathbb{N}_n$ *be two indexes. Then*

$$X_k^h Z_k X_k^h = Z_{h,k} \tag{21}$$

Proof. If $|a\rangle$ is a ket and $h \neq k \in \mathbb{N}_n$ are two indexes, then

$$\begin{aligned} |a\rangle X_k^h Z_k X_k^h &= a_{h0} |a\rangle_0^h Z_k X_k^h + a_{h1} |a\rangle_{1X}^{hk} Z_k X_k^h \\ &= a_{h0} |a\rangle_{0Z}^{hk} X_k^h + a_{h1} |a\rangle_{1ZX}^{hk} X_k^h \\ &= a_{h0} |a\rangle_{0Z}^{hk} + a_{h1} |a\rangle_{1XZX}^{hk} \\ &= a_{h0} |a\rangle_{0Z}^{hk} - a_{h1} |a\rangle_{1Z}^{hk} \\ &= a_{h0} |a\rangle_0^h Z_k - a_{h1} |a\rangle_1^h Z_k \\ &= |a\rangle Z_{h,k} \end{aligned}$$

□

For the following result, it is necessary to introduce the SWAP gate, which acts over two qubits, exchanging the state of both positions:

$$|a\rangle (SWAP)_{h,k} = a_{h0}a_{k0}|a\rangle_{00}^{hk} + a_{h0}a_{k1}|a\rangle_{10}^{hk} + a_{h1}a_{k0}|a\rangle_{01}^{hk} + a_{h1}a_{k1}|a\rangle_{11}^{hk} \tag{22}$$

Proposition 4.2. *Let $h,\ k \in \mathbb{N}_n$ be two indexes. Then*

$$X_k^h X_h^k X_k^h = (SWAP)_{h,k} \tag{23}$$

Proof. If $|a\rangle$ is a ket and $h \neq k \in \mathbb{N}_n$ are two indexes, then

$$\begin{aligned} |a\rangle X_k^h X_h^k X_k^h &= a_{h0}|a\rangle_0^h X_k^h X_h^k + a_{h1}|a\rangle_{1X}^{hk} X_k^h X_h^k \\ &= a_{h0}|a\rangle_0^h X_h^k X_k^h + a_{h1}a_{k0}|a\rangle_{11}^{hk} X_h^k X_k^h \\ &\quad + a_{h1}a_{k1}|a\rangle_{10}^{hk} X_h^k X_k^h = a_{h0}a_{k0}|a\rangle_{00}^{hk} X_k^h \\ &\quad + a_{h0}a_{k1}|a\rangle_{11}^{hk} X_k^h + a_{h1}a_{k0}|a\rangle_{01}^{hk} X_k^h \\ &\quad + a_{h1}a_{k1}|a\rangle_{10}^{hk} X_k^h = a_{h0}a_{k0}|a\rangle_{00}^{hk} \\ &\quad + a_{h0}a_{k1}|a\rangle_{10}^{hk} + a_{h1}a_{k0}|a\rangle_{01}^{hk} + a_{h1}a_{k1}|a\rangle_{11}^{hk} \\ &= (SWAP)_{h,k} \end{aligned}$$

□

Finally, the last three propositions are stated. The proofs of these, due to its extensive length, are given in the Appendix A.

Proposition 4.3. *Let $h,\ k \in \mathbb{N}_n$ be two indexes. Then*

$$H_{hk} X_h^k H_{hk} = X_k^h \tag{24}$$

Proposition 4.4. *Let $j,\ h,\ l \in \mathbb{N}_n$ be three indexes. Then*

$$X_l^{jh} X_h X_l^{jh} = X_l^j X_h \tag{25}$$

Proposition 4.5. *Let $j,\ h,\ k,\ l \in \mathbb{N}_n$ be four indexes. Then*

$$X_l^{jhk} X_h X_l^{jhk} = X_l^{jk} X_h \tag{26}$$

5. Implementation

An independent Python module has been developed for the simulation of quantum circuits [12], adapted to the previously described notation. From the source code in LaTeX for the description of any circuit, following the instructions of the proposal, an automatic calculation is obtained that outputs a corresponding state vector. This tool also optimises an initial expression of a circuit, using a set of rules defined from the propositions of the previous section. In addition, it has different functionalities for the representation of the circuit and its associated state vector.

5.1. Formal grammar

In order to achieve a consistent notation, close to how a programming language could be defined, aspects close to the underlying compiler theory have been defined, resulting in a formal analysis of the allowed expressions. The context-free grammar generated by the proposed language is as follows:

$$
\begin{aligned}
\text{S} &\rightarrow \text{TAG ARGS S} \mid \varepsilon \\
\text{TAG} &\rightarrow \text{LETTER TAG} \mid \text{LETTER} \\
\text{LETTER} &\rightarrow \text{A} \mid \text{B} \mid \ldots \mid \text{Z} \\
\text{ARGS} &\rightarrow \text{TARGET CONTROL} \mid \text{CONTROL TARGET} \mid \text{TARGET} \\
\text{TARGET} &\rightarrow \text{\_DIGIT} \mid \text{\_\{INDEXES\}} \\
\text{CONTROL} &\rightarrow \text{\^{}DIGIT} \mid \text{\^{}\{INDEXES\}} \\
\text{NUMBER} &\rightarrow \text{DIGIT NUMBER} \mid \text{DIGIT} \\
\text{INDEXES} &\rightarrow \text{RANGE} \mid \text{LIST} \\
\text{RANGE} &\rightarrow \text{NUMBER : LIST} \\
\text{LIST} &\rightarrow \text{NUMBER , LIST} \mid \text{NUMBER , RANGE} \mid \text{NUMBER} \\
\text{DIGIT} &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

The non-terminal TAG symbol defines the name of the gate in capital letters (when derived in LETTER sequences). Each gate is always accompanied by a number of arguments (ARGS), which are the targets and controls of the gate. The order of the arguments is not relevant, and the control argument is optional. In both cases, if the argument is a unique DIGIT, the brackets can be omitted. Otherwise, it may be an index range, which is continuous or discontinuous thanks to the INDEXES production rule (RANGE | LIST), and the use of the corresponding separator.

5.2. Greedy optimization

To take advantage of and demonstrate in a practical way the applicability of the proposed notation in the optimization of quantum circuits, a first approximation of a greedy algorithm to tackle this task is suggested in algorithm 1.

Algorithm 1. Quantum Circuit Synthesis

Data: Initial quantum circuit QC, set of rules R
$QC^* \leftarrow QC$
repeat
  for ***each*** *rule* ***in*** R **do**
    $QC' \leftarrow$ Apply *rule* to QC until no further changes occur
    if $|QC'| \leqslant |QC^*|$ **then**
    $QC^* \leftarrow QC'$
  end
  if QC^* has changed **then**
  $QC \leftarrow QC^*$
until QC^* *has not changed*
return QC^*

The *Quantum Circuit Synthesis* algorithm is designed to improve an initial quantum circuit QC using a set of transformation rules R derived from specific theorems presented in the research (e.g. Equation (21), 24, 25). The primary objective is to iteratively apply these rules to simplify the circuit, focusing on minimizing certain aspects such as circuit depth, the total number of gates or the number of two-qubit gates.

The input of this first approach of the optimization algorithm is:

- *Initial quantum circuit* QC: It represents the starting circuit to be optimised, using the presented notation.
- *Set of rules* R: The rules are derived from theoretical foundations and are expressed as equalities. Each rule has a *left* part, which identifies a pattern in the circuit, and a *right* part, which, in this context, conveniently specifies the transformation that minimizes the expression in the circuit ($|left| \geqslant |right|$).

The algorithm outputs the optimised quantum circuit QC^*, which represents the best configuration found through the synthesis process. This procedure is particularly efficient in scenarios where the transformation rules can effectively reduce the total number of gates of the quantum circuits. However, its performance is highly dependent on the quality and comprehensiveness of the rule set R.

Note that, as a greedy algorithm, the choice at each step is locally optimal, with the expectation that those local choices will lead to a globally optimal solution. In particular, the idea of the algorithm is that the generated circuit QC^* is locally optimal with respect to the applied rules, but, since the rules can be applied in different ways, a global optimum cannot be guaranteed (see appendix B). In this sense, certain rule sets prevent the described greedy optimization from achieving the global optimum. For example, it might be necessary to apply

the rules to deliberately enlarge the expression, using the right-hand side for pattern recognition and the left-hand side for the corresponding substitution. However, for those rules that can be applied recursively without limit, this may pose a problem. It is worth studying the possibilities of defining upper bounds for the number of successive applications of this type of rule sets, and of generating equivalent rule sets that are confluent and terminating. In any case, the more complete the set of defined rules, the better the results of this greedy algorithm, which for different executions always returns the same local optimum value in a deterministic way, as it is the best solution found in the defined space of solutions. Therefore, further optimization techniques or more sophisticated rule sets may be necessary for more complex quantum circuits. Both issues are currently being investigated using a metaheuristic techniques approach.

5.3. Additional available functionalities

Once the grammar has been defined and the greedy optimization algorithm has been implemented, a Parser has been built capable of recognising and understanding instances generated by means of these syntactic structures, transforming them into functional programming objects. In addition to this, a Writer has been implemented capable of generating the file with LaTeX code that develops the expression introduced step by step, calculating the final state vector. Initially, the main program allows setting a number of qubits in the system, and the desired circuit encoding using this language. This allows the generated state vector to be obtained, as well as its visualisation using Circle Notation (see [13]). It also performs the graphical visualisation of the entered circuit. All these functionalities have been developed in an original way, without using external frameworks or libraries for the simulation of quantum algorithms such as Qiskit [14], Pennylane [15], Q# [16], Amazon Braket [17], Cirq [18], etc. A Parser, a Writer, an execution engine, and a circuit and state vectors visualisers have been implemented to achieve the proposed functionalities.

5.4. Examples

Some of the best-known quantum circuits in the scientific literature [11] are presented below, showing the equivalent notation that serves as input for the implementation. For each circuit, the output of the main program is the mathematical workout of the expression until the final state vector is obtained, its representation in Circle Notation and the circuit graphical representation. As these small circuits show no improvement in the optimiser, it is in the next section that their effectiveness will be empirically demonstrated.

(1) Bell state: $H_1X_2^1$. This circuit generates a Bell entangled pair, driving the $|00\rangle$ state to the entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

(a) Generated mathematical workout:

$$\begin{aligned} &|00\rangle H_1X_2^1 \\ &\quad = [0.5\sqrt{2}\,|00\rangle + 0.5\sqrt{2}\,|10\rangle]X_2^1 \\ &\quad = [0.5\sqrt{2}\,|00\rangle + 0.5\sqrt{2}\,|11\rangle] \end{aligned}$$

(b) Circle Notation (figure 1).

(c) Graphical representation (figure 2).

(2) Superdense coding (of $|01\rangle$ state): $H_1X_2^1X_1X_2^1H_1$. It is a protocol that transmits two classical bits using a single communication qubit.

(a) Generated mathematical workout:

$$\begin{aligned} &|00\rangle H_1X_2^1X_1X_2^1H_1 \\ &\quad = [0.5\sqrt{2}\,|00\rangle + 0.5\sqrt{2}\,|10\rangle]X_2^1X_1X_2^1H_1 \\ &\quad = [0.5\sqrt{2}\,|00\rangle + 0.5\sqrt{2}\,|11\rangle]X_1X_2^1H_1 \\ &\quad = [0.5\sqrt{2}\,|01\rangle + 0.5\sqrt{2}\,|10\rangle]X_2^1H_1 \\ &\quad = [0.5\sqrt{2}\,|01\rangle + 0.5\sqrt{2}\,|11\rangle]H_1 \\ &\quad = [|01\rangle] \end{aligned}$$

(b) Circle Notation (figure 3).

(c) Graphical representation (figure 4).

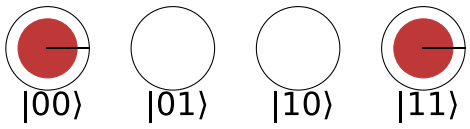

Figure 1. Circle Notation of Bell State.

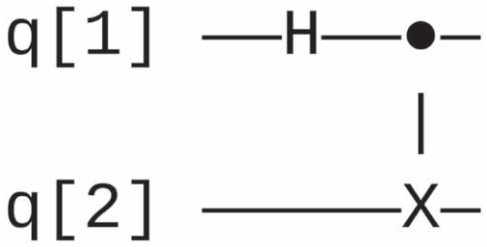

Figure 2. Graphical Representation of Bell State Circuit.

(3) Quantum Teleportation: $H_2X_3^2X_2^1H_1$. It is a protocol that transmits a qubit using two classical communication bits.

(a) Generated mathematical workout:

$$\begin{aligned}
&|000\rangle H_2X_3^2X_2^1H_1\\
&= [0.5\sqrt{2}\,|000\rangle + 0.5\sqrt{2}\,|010\rangle]X_3^2X_2^1H_1\\
&= [0.5\sqrt{2}\,|000\rangle + 0.5\sqrt{2}\,|011\rangle]X_2^1H_1\\
&= [0.5\sqrt{2}\,|000\rangle + 0.5\sqrt{2}\,|011\rangle]H_1\\
&= [0.5|000\rangle + 0.5|011\rangle + 0.5|100\rangle + 0.5|111\rangle]
\end{aligned}$$

(b) Circle Notation (figure 5).

(c) Graphical representation (figure 6).

6. Case of use

A clear example of an efficient use case of quantum circuit optimization is explained using the Deutsch-Jozsa algorithm implementation described by the Qiskit IBM documentation [19]. It explains how to build a quantum circuit for the oracle of a function from its truth table and shows the final result (see figure 7).

The circuit above can be intuitively improved through the Qiskit transpile function [20], to obtain a simpler version (see figure 8).

This circuit can be expressed like algebraic circuit as:

$$X_3X_4^{1,2,3}X_2X_4^{1,2,3}X_{1,2}X_4^{1,2,3}X_2X_4^{1,2,3}X_{1,2,3} \tag{27}$$

The developed implementation automatically performs the following optimization steps. Applying the result (26) over the red box, the result (25) over the blue box and the result (15) over the green box, it is possible to reduce the circuit, it is obtained:

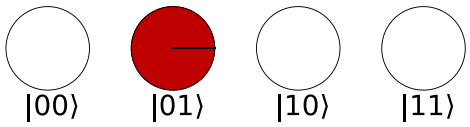

Figure 3. Circle Notation of Superdense Coding (of $|01\rangle$ State).

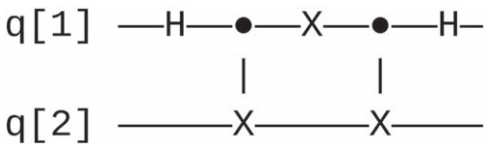

Figure 4. Graphical Representation of Superdense Coding Circuit.

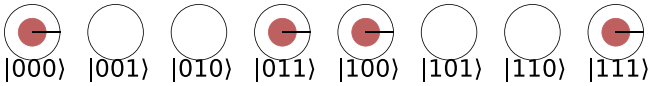

Figure 5. Circle Notation of Quantum Teleportation.

$$
\begin{aligned}
& X_3 \boxed{X_4^{1,2,3} X_2 X_4^{1,2,3}} X_{1,2} \boxed{X_4^{1,2,3} X_2 X_4^{1,2,3}} X_{1,2,3} \\
&= X_3 X_2 X_4^{1,3} \boxed{X_{1,2} X_2} X_4^{1,3} X_{1,2,3} \\
&= X_3 X_2 \boxed{X_4^{1,3} X_1 X_4^{1,3}} X_{1,2,3} \\
&= X_3 X_2 X_1 X_4^3 X_{1,2,3} = X_3 X_4^3 X_2 \boxed{X_1 X_{1,2,3}} \\
&= X_3 X_4^3 \boxed{X_2 X_{23}} = X_3 X_4^3 X_3
\end{aligned}
$$

With the quantum circuit representation, the improvement can be appreciated in figure 9.

The additional results generated from the realised implementation on both circuits (unoptimised and optimised) are shown below. Thanks to the tool developed, the difference between the number of operations needed to compute one circuit or another is noticeable. Furthermore, the available representations of the results are shown, showing the graphical difference between circuits, and the common state generated using Circle Notation.

1. Generated mathematical workout of the unoptimised circuit:

$$
\begin{aligned}
& |0000\rangle X_3 X_4^{1:3} X_2 X_4^{1:3} X_{1,2} X_4^{1:3} X_2 X_4^{1:3} X_{1:3} \\
&= [|0010\rangle] X_4^{1:3} X_2 X_4^{1:3} X_{1,2} X_4^{1:3} X_2 X_4^{1:3} X_{1:3} \\
&= [|0010\rangle] X_2 X_4^{1:3} X_{1,2} X_4^{1:3} X_2 X_4^{1:3} X_{1:3} \\
&= [|0110\rangle] X_4^{1:3} X_{1,2} X_4^{1:3} X_2 X_4^{1:3} X_{1:3} \\
&= [|0110\rangle] X_{1,2} X_4^{1:3} X_2 X_4^{1:3} X_{1:3} \\
&= [|1010\rangle] X_4^{1:3} X_2 X_4^{1:3} X_{1:3} \\
&= [|1010\rangle] X_2 X_4^{1:3} X_{1:3} \\
&= [|1110\rangle] X_4^{1:3} X_{1:3} \\
&= [|1111\rangle] X_{1:3} \\
&= [|0001\rangle]
\end{aligned}
$$

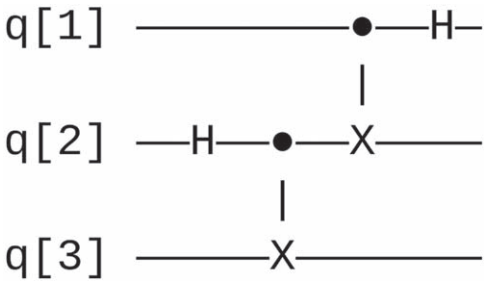

Figure 6. Graphical Representation of Quantum Teleportation Circuit.

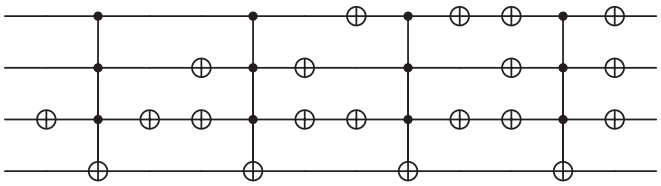

Figure 7. Deutsch-Jozsa oracle described by Qiskit IBM documentation.

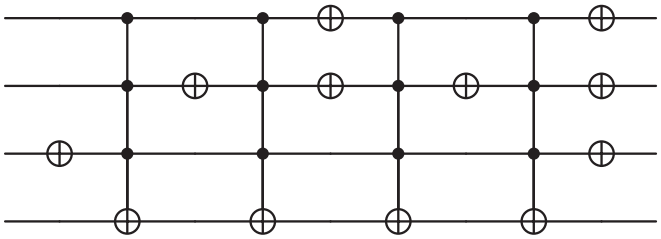

Figure 8. Deutsch-Jozsa Oracle Optimised by Qiskit IBM Documentation.

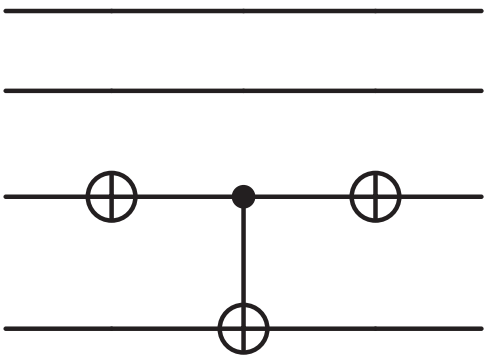

Figure 9. Deutsch-Jozsa Oracle Optimised Using the Proposed Notation.

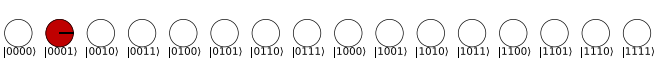

Figure 10. Circle Notation of Deutsch-Jozsa state.

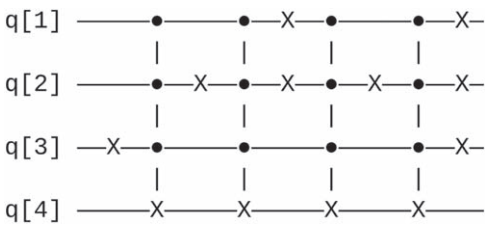

Figure 11. Graphical Representation of Unoptimised Deutsch-Jozsa Circuit.

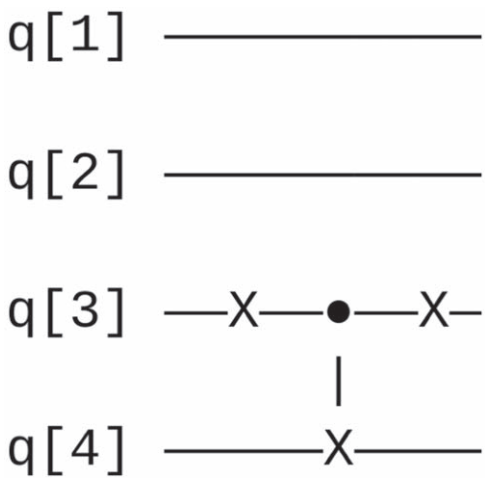

Figure 12. Graphical representation of Optimised Deutsch-Jozsa Circuit.

2. Generated mathematical workout of the optimised circuit:

$$
\begin{aligned}
&|0000\rangle X_3 X_4^3 X_3 \\
&\quad = [|0010\rangle] X_4^3 X_3 \\
&\quad = [|0011\rangle] X_3 \\
&\quad = [|0001\rangle]
\end{aligned}
$$

3. Circle Notation (figure 10).
4. Graphical representation of the unoptimised circuit (figure 11).
5. Graphical representation of the optimised circuit (figure 12).

7. Conclusions and future works

In this paper, a notation with great potential to represent quantum circuits from an algebraic point of view has been proposed. This new notation greatly facilitates the discovery and proof of new properties and relationships between quantum gates, as well as the handling of calculations and the efficiency of systems that simulate them classically. To demonstrate this, a computer tool has been developed that can be used to simulate classically the evolution of the state vector through a circuit expressed in algebraic notation. Several examples are shown that clarify the results and consolidate the idea of easier handling of circuits in algebraic notation compared to the traditional graphical representation.

Four future lines of research are the following: extending the notation to add higher level abstraction features, finding properties that simplify the most commonly used patterns in the development of quantum algorithms, introducing the concept of measurement (both in notation and implementation) and improving considerably the optimisation algorithm from a metaheuristic point of view. These lines can enhance the growth

of the theoretical basis and the proposed implementation, leading to the creation of a new, manageable language for the characterization and optimization of quantum circuits.

Acknowledgments

This research has been made possible thanks to the following projects and organizations: the 2023DIG28 project funded by CajaCanarias and Fundación la Caixa, the PID2022-138933OB-I00 project funded by MCIN/AEI/10.13039/501100011033/ FEDER EU, and the SCITALA C064/23 project and the Cybersecurity Chair ULL-INCIBE funded by the Recovery, Transformation and Resilience Plan financed by the European Union (Next Generation EU).

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Appendix A. Extensive proofs

The following proofs have been added as an appendix due to their excessive length. The proof of (24) is as follows.

Proof. If $|a\rangle$ is a ket and $h \neq k \in \mathbb{N}_n$ are two indexes, then

$$\begin{aligned}
|a\rangle H_{kh} X_h^k H_{kh} &= a_{h0}|a\rangle_+^h H_k X_h^k H_{kh} + a_{h1}\,|a\rangle_-^h H_k X_h^k H_{kh} \\
&= a_{h0}a_{k0}|a\rangle_{++}^{hk} X_h^k H_{kh} + a_{h0}\,a_{k1}|a\rangle_{+-}^{hk} X_h^k H_{kh} \\
&\quad + a_{h1}a_{k0}|a\rangle_{-+}^{hk} X_h^k H_{kh} + a_{h1}\,a_{k1}|a\rangle_{--}^{hk} X_h^k H_{kh} \\
&= a_{h0}a_{k0}|a\rangle_{++}^{hk} H_{hk} + a_{h0}a_{k1}|a\rangle_{+-}^{hk} H_{hk} \\
&\quad + a_{h1}a_{k0}|a\rangle_{--}^{hk} H_{hk} + a_{h1}a_{k1}|a\rangle_{-+}^{hk} H_{hk} \\
&= a_{h0}a_{k0}|a\rangle_{+0}^{hk} H_h + a_{h0}a_{k1}|a\rangle_{+1}^{hk} H_h \\
&\quad + a_{h1}a_{k0}|a\rangle_{-1}^{hk} H_h + a_{h1}a_{k1}|a\rangle_{-0}^{hk} H_h \\
&= a_{h0}|a\rangle_+^h H_h + a_{h1}|a\rangle_-^h H_h X_k \\
&= a_{h0}|a\rangle_0^h + a_{h1}|a\rangle_1^h X_k \\
&= |a\rangle X_k^h
\end{aligned}$$

□

Furthermore, the proof of (25) is shown below.

Proof. If $|a\rangle$ is a ket and $j,\ h,\ l \in \mathbb{N}_n$ are three distinct indexes, then the ket $|b\rangle = |a\rangle X_l^{jh} X_h X_l^{jh}$ is for construction, as shown below

$$\begin{aligned}
|b\rangle &= a_{j0}a_{h0}\,|a\rangle_{00}^{jh} X_h X_l^{jh} + a_{j0}a_{h1}\,|a\rangle_{01}^{jh} X_h X_l^{jh} \\
&\quad + a_{j1}a_{h0}\,|a\rangle_{10}^{jh} X_h X_l^{jh} + a_{j1}a_{h1}\,|a\rangle_{11X}^{jhl} X_h X_l^{jh} \\
&= a_{j0}a_{h0}\,|a\rangle_{01}^{jh} X_l^{jh} + a_{j0}a_{h1}\,|a\rangle_{00}^{jh} X_l^{jh} \\
&\quad + a_{j1}a_{h0}\,|a\rangle_{11}^{jh} X_l^{jh} + a_{j1}a_{h1}\,|a\rangle_{10X}^{jhl} X_l^{jh} \\
&= a_{j0}a_{h0}\,|a\rangle_{01}^{jh} + a_{j0}a_{h1}\,|a\rangle_{00}^{jh} + a_{j1}a_{h0}\,|a\rangle_{11X}^{jhl} + a_{j1}a_{h1}\,|a\rangle_{00X}^{jhl} \\
&= \boxed{a_{j0}a_{h0}\,|a\rangle_{00}^{jh} X_h} + \boxed{a_{j0}a_{h1}\,|a\rangle_{01}^{jh} X_h} \\
&\quad + \boxed{a_{j1}a_{h0}\,|a\rangle_{10}^{jh} X_l X_h} + \boxed{a_{j1}a_{h1}\,|a\rangle_{11}^{jh} X_l X_h} \\
&= a_{h0}\,|a\rangle_0^h X_l^j X_h + a_{h1}\,|a\rangle_1^h X_l^j X_h \\
&= |a\rangle X_l^j X_h
\end{aligned}$$

Finally, analogous to the previous proof, the expression (26) is proved. □

Proof. If $|a\rangle$ is a ket and $j,\ h,\ k,\ l \in \mathbb{N}_n$ are four distinct indexes, then the ket $|b\rangle = |a\rangle X_l^{jhk} X_h X_l^{jhk}$ is for construction, as shown below

$$
\begin{aligned}
|b\rangle &= a_{j0}a_{h0}a_{k0}\,|a\rangle_{000}^{jhk}\,X_h X_l^{jhk} + a_{j0}a_{h0}a_{k1}\,|a\rangle_{001}^{jhk}\,X_h X_l^{jhk} \\
&+ a_{j0}a_{h1}a_{k0}\,|a\rangle_{010}^{jhk}\,X_h X_l^{jhk} + a_{j0}a_{h1}a_{k1}\,|a\rangle_{011}^{jhk}\,X_h X_l^{jhk} \\
&+ a_{j1}a_{h0}a_{k0}\,|a\rangle_{100}^{jhk}\,X_h X_l^{jhk} + a_{j1}a_{h0}a_{k1}\,|a\rangle_{101}^{jhk}\,X_h X_l^{jhk} \\
&+ a_{j1}a_{h1}a_{k0}\,|a\rangle_{110}^{jhk}\,X_h X_l^{jhk} + a_{j1}a_{h1}a_{k1}\,|a\rangle_{111X}^{jhkl}\,X_h X_l^{jhk} \\
&= a_{j0}a_{h0}a_{k0}\,|a\rangle_{010}^{jhk}\,X_l^{jhk} + a_{j0}a_{h0}a_{k1}\,|a\rangle_{011}^{jhk}\,X_l^{jhk} \\
&+ a_{j0}a_{h1}a_{k0}\,|a\rangle_{000}^{jhk}\,X_l^{jhk} + a_{j0}a_{h1}a_{k1}\,|a\rangle_{001}^{jhk}\,X_l^{jhk} \\
&+ a_{j1}a_{h0}a_{k0}\,|a\rangle_{110}^{jhk}\,X_l^{jhk} + a_{j1}a_{h0}a_{k1}\,|a\rangle_{111}^{jhk}\,X_l^{jhk} \\
&+ a_{j1}a_{h1}a_{k0}\,|a\rangle_{100}^{jhk}\,X_l^{jhk} + a_{j1}a_{h1}a_{k1}\,|a\rangle_{101X}^{jhkl}\,X_l^{jhk} \\
&= a_{j0}a_{h0}a_{k0}\,|a\rangle_{010}^{jhk} + a_{j0}a_{h0}a_{k1}\,|a\rangle_{011}^{jhk} \\
&+ a_{j0}a_{h1}a_{k0}\,|a\rangle_{000}^{jhk} + a_{j0}a_{h1}a_{k1}\,|a\rangle_{001}^{jhk} \\
&+ a_{j1}a_{h0}a_{k0}\,|a\rangle_{110}^{jhk} + a_{j1}a_{h0}a_{k1}\,|a\rangle_{111X}^{jhkl} \\
&+ a_{j1}a_{h1}a_{k0}\,|a\rangle_{100}^{jhk} + a_{j1}a_{h1}a_{k1}\,|a\rangle_{101X}^{jhkl} \\
&= \boxed{a_{j0}a_{h0}a_{k0}\,|a\rangle_{000}^{jhk}\,X_h} + \boxed{a_{j0}a_{h0}a_{k1}\,|a\rangle_{001}^{jhk}\,X_h} \\
&+ \boxed{a_{j0}a_{h1}a_{k0}\,|a\rangle_{010}^{jhk}\,X_h} + \boxed{a_{j0}a_{h1}a_{k1}\,|a\rangle_{011}^{jhk}\,X_h} \\
&+ \boxed{a_{j1}a_{h0}a_{k0}\,|a\rangle_{100}^{jhk}\,X_h} + \boxed{a_{j1}a_{h0}a_{k1}\,|a\rangle_{101X}^{jhkl}\,X_h} \\
&+ \boxed{a_{j1}a_{h1}a_{k0}\,|a\rangle_{110}^{jhk}\,X_h} + \boxed{a_{j1}a_{h1}a_{k1}\,|a\rangle_{111X}^{jhkl}\,X_h} \\
&= a_{j0}a_{k0}\,|a\rangle_{00}^{jk}\,X_h + a_{j0}a_{k1}\,|a\rangle_{01}^{jk}\,X_h \\
&+ a_{j1}a_{k0}\,|a\rangle_{10}^{jk}\,X_h + a_{j1}a_{k1}\,|a\rangle_{11X}^{jkl}\,X_h \\
&= |a\rangle\,X_l^{jk}X_h
\end{aligned}
$$

Appendix B. Rule application

The *Apply Rule* algorithm is a subroutine designed to apply a specific transformation rule r to a quantum circuit QC. The rule r consists of a *left* part and a *right* part, where the *left* part identifies a pattern in the circuit to be transformed, and the *right* part defines the transformation that minimizes the expression or configuration of the circuit. This process helps in optimizing the circuit by systematically reducing its complexity.

Algorithm 2. Apply Rule

Date: Quantum circuit QC, Rule r □
1 $QC' \leftarrow QC$
2 ▷ In rule r, designate *left* as the part to be reduced and *right* as the target part to minimize the expression
3 **while** *the left part matches in* QC' **do**
4 Find the first match in QC'
5 Transform QC' using the *right* part of r at the match position
6 **end**
7 **return** QC'

This procedure is crucial for incrementally transforming a quantum circuit using specific rules. Each rule targets particular patterns within the circuit, enabling systematic simplification or optimization. The effectiveness of this approach depends significantly on the selection of rules and the completeness of the rule set, as the final optimised form of the circuit QC' is limited to the transformations defined by these rules. This algorithm is typically used in conjunction with broader optimization strategies that iteratively apply multiple rules to achieve a more comprehensive circuit optimization.

ORCID iDs

Daniel Escanez-Exposito https://orcid.org/0000-0003-4215-0501
Pino Caballero-Gil https://orcid.org/0000-0002-0859-5876
Marcos Rodriguez-Vega https://orcid.org/0009-0004-2379-842X
Francisco Costa-Cano https://orcid.org/0000-0002-7669-8930
Eduardo Sáenz-de-Cabezón https://orcid.org/0000-0002-5615-4194

References

[1] Deutsch D 1989 Quantum computational networks *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* **425** 73–90
[2] Gerdt V P, Kragler R and Prokopenya A N 2009 A mathematica program for constructing quantum circuits and computing their unitary matrices *Phys. Part. Nucl. Lett.* **6** 526–9
[3] Hutsell S R and Greenwood G W 2009 Efficient algebraic representation of quantum circuits *Journal of Discrete Mathematical Sciences and Cryptography* **12** 429–49
[4] Ying M and Feng Y 2009 An algebraic language for distributed quantum computing *IEEE Trans. Comput.* **58** 728–43
[5] Staton S 2015 Algebraic effects, linearity, and quantum programming languages *ACM SIGPLAN Notices* **50** 395–406
[6] Wang Y 2019 Probabilistic process algebra to unifying quantum and classical computing in closed systems *Int. J. Theor. Phys.* **58** 436–3509
[7] Soeken M, Meuli G, Schmitt B, Mozafari F, Riener H and De Micheli G 2020 Boolean satisfiability in quantum compilation *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **378** 20190161
[8] Nash B, Gheorghiu V and Mosca M 2020 Quantum circuit optimizations for nisq architectures *Quantum Sci. Technol.* **5** 025010
[9] Wang Q, Li R and Ying M 2021 Equivalence checking of sequential quantum circuits *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **41** 3143–56
[10] Dirac P 1939 A new notation for quantum mechanics *Math. Proc. Cambridge Philos. Soc.* **35** 416–8
[11] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information* (Cambridge University Press)
[12] Escanez-Exposito D and Rodriguez-Vega M 2023 *QAlgebra* (https://github.com/jdanielescanez/qalgebra.https://github.com/jdanielescanez/qalgebra)
[13] Johnston E, Harrigan N and Gimeno-Segovia M 2019 *Programming Quantum Computers: Essential Algorithms and Code Samples* (OReilly Media, Inc)
[14] IBM Research, 2023. Accessed: 2023-11-13 (https://qiskit.org/)
[15] Xanadu , 2023. Accessed: 2023-11-13 (https://pennylane.ai/)
[16] Microsoft , 2023. Accessed: 2023-11-13 (https://learn.microsoft.com/enus/azure/quantum/user-guide/)
[17] Amazon, 2023. Accessed: 2023-11-13 (https://aws.amazon.com/braket/)
[18] Google Inc., 2023. Accessed: 2023-11-13 (https://quantumai.google/cirq)
[19] IBM Research 2023 *Deutsch-jozsa algorithm* (https://qiskit.org/documentation/stable/0.24/tutorials/algorithms/09_textbook_algorithms.html#Deutsch-Jozsa-algorithm)
[20] BM Research Nov 2023 *Transpiler (qiskit.transpiler)* (https://qiskit.org/documentation/apidoc/transpiler.html)