Universidad de Oviedo

Programa de Doctorado en
Matemáticas y Estadística

Tesis Doctoral:

Algoritmos Cuánticos para Estructuras Algebraicas

Jefferson Miguel Hernández Cáceres

Directores:
Ignacio Fernández Rúa, Elías Fernández-Combarro Álvarez

Oviedo
2024

Universidad de Oviedo

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: Algoritmos cuánticos para estructuras algebraicas	Inglés: Quantum Algorithms for Algebraic Structures

2.- Autor	
Nombre: Jefferson Miguel Hernández Cáceres	
Programa de Doctorado: Matemáticas y Estadística	
Órgano responsable: : Centro Internacional de Postgrado	

RESUMEN (en español)

La computación cuántica es un paradigma de computación bien asentado desde el punto de vista teórico [NC11] y prometedor desde el punto de vista práctico [Aea19].

Por otra parte, el estudio de estructuras algebraicas a través de técnicas computacionales es bien conocido y constituye una línea de investigación habitual en Algebra [vzGG99, CCS99]. En concreto, entre unas de las estructuras algebraicas que han sido estudiadas usando esta metodología se encuentran los semicuerpos finitos [Knu65]. Para ellos, se han desarrollado algunas técnicas computacionales para su estudio y potenciales aplicaciones. En el primer aspecto, se incluye la clasificación computacional de semicuerpos finitos [RCR09, RCR12]; en el segundo, la construcción de S-cajas criptográficas a partir de ellos [RC18]. En todos los casos mencionados anteriormente, los algoritmos considerados son clásicos.

Para el modelo de computación cuántica, se han propuesto algunos algoritmos para el estudio de estructuras algebraicas. Por ejemplo, consideremos el caso de determinar si una estructura (algebra, anillo, grupo) es conmutativa o no. Este es un problema estudiado en diferentes contextos desde un punto de vista estrictamente teórico [Jac45, MW05, Pso84]. Desde el punto de vista efectivo, esto es, computacional, se ha estudiado especialmente en el caso de grupos, para los que se han propuesto algoritmos (no solamente clásicos aleatorizados sino también cuánticos) [MN05, Pak12].

En el caso de las algebras finito dimensionales, la necesidad de un procedimiento efectivo para determinar la conmutatividad de la estructura es un problema natural en el contexto del estudio computacional de semicuerpos finitos. Para este problema se han propuesto diferentes algoritmos cuánticos, basados en el algoritmo de Grover [CRR19a], en métodos adiabáticos [CRR19c] y en caminos cuánticos [CRR19b]. En todos estos casos la detección de pares de constantes de estructura que no coinciden es el elemento fundamental sobre el que se articulan los algoritmos cuánticos considerados. Como consecuencia de estos estudios se ha desarrollado una abstracción de diversas técnicas cuánticas de detección bajo el formalismo de los Quantum Abstract Detecting Systems (QADS) [CRR20].

En esta tesis presentamos dos familias de QADS, llamadas QADS Combinatorios y QADS Rotacionales que, respectivamente, generalizan los sistemas de detección basados en puertas controladas por un solo qubit y en el algoritmo de Grover. Además, estudiamos sus propiedades, entre estas, el cierre algorítmico de cada familia, y demostramos que algunos de estos QADS son equivalentes (en el sentido de tener la misma tasa de detección) a otros construidos a partir del producto tensorial de operadores controlados y sus raíces cuadradas. también aplicamos la construcción de QADS combinatorios a un problema de decisión de valores propios, problema de estimación de fase y, adicionalmente, al problema de determinar la conmutatividad de algebras finito-dimensionales. La segunda familia, QADS rotacionales, incluye un caso particular, los QADS de la búsqueda de Grover [HCCR22].

En relación con los métodos cuánticos que resuelven el problema del subgrupo oculto (por ejemplo, el algoritmo de Shor), como segunda parte de esta tesis, presentamos algoritmos cuánticos que encuentran subestructuras de manera eficiente, al formularlo como una instancia del problema del subgrupo oculto. Estos algoritmos cuánticos tienen orden de complejidad polinomial (cuántica) en la dimensión del algebra [JER23].

Por último, desarrollamos métodos cuánticos para la clasificación computacional de semicuerpos finitos mediante técnicas computacionales cuánticas (basadas en el algoritmo de Grover) y su implementación, incluida la clasificación efectiva mediante simuladores cuánticos de semicuerpos finitos de tamaños pequeños [HCR23].

RESUMEN (en Inglés)

On one hand, quantum computing is a well-established computing paradigm, from a theoretical point of view [NC11], and promising, from a practical point of view [Aea19].

On the other hand, the study of algebraic structures using computational techniques is well known and constitutes a common line of research in Algebra [vzGG99, CCS99]. Specifically, in the study and applications of finite Semifields [Knu65]. For these structures, some computational techniques have been developed for their classification. For instance, [RCR09, RCR12]; and also, for the construction of cryptographic S-boxes from them [RC18]. In all these cases, the algorithms considered are classical, understood as those that can be executed on an ordinary computer, that is, not quantum.

For the quantum computing model, some algorithms have been proposed for the study of algebraic structures. For example, consider the case of determining whether a structure (algebra, ring, group) is commutative or not. This is a problem studied in different contexts from a strictly theoretical point of view [Jac45, MW05, Pso84]. From the effective point of view, that is, computational, it has been studied especially in the case of groups, for which algorithms have been proposed (not only randomized classical, but also quantum) [MN05, Pak12].

In the case of finite-dimensional algebras, the need for an effective procedure to determine the commutativity of the structure is a natural problem in the context of the computational study of finite semifields. For this problem, different quantum algorithms have been proposed, based on Grover's algorithm [CRR19a], adiabatic methods [BF28] and quantum walks [CRR19b]. In all these cases, the detection of pairs of structure constants that do not coincide is the fundamental element on which the considered quantum algorithms are articulated. As a consequence of these studies, an abstraction of various quantum detection techniques has been developed under the formalism of Quantum Abstract Detecting Systems (QADS) [CRR20].

So, in this thesis we introduce two family of QADS, namely Combinatorial QADS and Rotational QADS which, respectively, generalize detecting systems based on single qubit controlled gates and on Grover's algorithm. Additionally, we study their properties, namely, the algorithmic closure of each family, and prove that some of these QADS are equivalent (in the sense of having the same detection rate) to others constructed from tensor product of controlled operators and their square roots. We also apply the combinatorial QADS construction to a problem of eigenvalue decision, and to a problem of phase estimation. Also to the problem of determining the commutativity of finite dimensional algebras. The second family, Rotational QADS includes as a particular case the QADS from Grover's search [HCCR22].

Next, in connection with the quantum methods that solve the hidden subgroup problem (such as Shor's algorithm), the study of problems related to the calculation of substructures is proposed. In fact, we introduce quantum algorithms that find substructures efficiently, by formulating it as an instance of the Hidden Subgroup Problem. These quantum algorithms has a polynomial (quantum) complexity order in the dimension of the algebra [JER23].

Universidad de Oviedo

Finally, we develop quantum methods for the computational classification of finite semifields by quantum computational techniques (based on Grover's algorithm) and their implementation, including the effective classification by quantum simulators of finite semifields of small sizes [HCR23].

SR. PRESIDENTE DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO EN MATEMÁTICAS Y ESTADÍSTICA

Contents

Resumen **v**
Publicaciones vii

Abstract **vii**
Publications xi

1 Introduction **1**
Overview of the Contents 5
Notation and Conventions 6
Acknowledgment 7

2 Algebraic Foundations **9**
2.1 Groups 9
2.2 Rings and Modules 11
2.3 K-algebras 13
2.4 Finite Semifields 16
2.5 Character Theory of Finite Abelian Groups 19
2.6 Probability of Generating a Group 21

3 Quantum preliminaries **23**
3.1 Quantum Circuit Model 23
3.1.1 Braket Notation 23
3.1.2 Qubits 24
3.1.3 Measurement 27
3.1.4 Quantum gates 28
3.2 Quantum Circuits 31
3.2.1 Oracles 37
3.3 Quantum Fourier Transform over Abelian Groups 40
3.3.1 QFT over $(\mathbb{Z}/p\mathbb{Z})^n$ 43
3.4 Quantum Optimization 45

4 Some Quantum Algorithms **49**
4.1 Simon's Algorithm 49
4.2 The Hidden Subgroup Problem 52

4.3 Quantum Phase Estimation . . . 58
4.4 Grover's Search Algorithm . . . 60
4.5 Technique For Listing All Elements Marked By An Oracle . . . 65
4.6 Quantum Abstract Detecting Systems (QADS) . . . 67
4.6.1 Algorithmic closure of QADS . . . 69
4.6.2 Properties of QADS . . . 71
4.6.3 Detection with a QADS . . . 71

5 Combinatorial and Rotational QADS 73
5.1 *m*-Combinatorial QADS . . . 73
5.2 Rotational QADS . . . 82
5.3 Application: Decision on Eigenvalues . . . 89
5.4 Application: Phase estimation . . . 91
5.4.1 Generalized Hadamard Test . . . 91
5.4.2 Dichotomy search . . . 95
5.4.3 Hybrid methodology . . . 95
5.5 Application: Commutativity of Finite Algebras with Combinatorial QADS. . . 98

6 Efficient Quantum Algorithms To Find Substructures On Finite Algebras 103
6.1 Substructures . . . 104
6.2 The classical approach . . . 107
6.2.1 Hiding functions . . . 107
6.2.2 Classical solution . . . 108
6.3 The quantum approach . . . 110
6.3.1 Oracle of the hiding function . . . 110
6.3.2 Quantum Algorithm To Find Substructures . . . 112
6.3.3 Classical post processing . . . 115
6.3.4 Examples . . . 116

7 An approach to the Classification of Finite Semifields by Quantum Computing 125
7.1 Quantum Computational Search of Finite Semifields with Grover's algorithm . 125
7.1.1 Semifield of Order 8 . . . 127
7.1.2 Description of Semifields of Order 16 . . . 133
7.1.3 Estimation of costs for the general case, in terms of Quantum Gates . . 152
7.2 Quantum Computational Search of Finite Semifields with Quantum Optimization 155

8 Conclusions 159

Conclusiones 160

Appendices 163
A Codes for Chapter 5 . . . 163
B Codes for Chapter 6 . . . 173
C Codes for Chapter 7 . . . 183

Bibliography **205**

Index **213**

Resumen

La computación cuántica es un paradigma de computación bien asentado desde el punto de vista teórico [NC11] y prometedor desde el punto de vista práctico [Aea19].

Por otra parte, el estudio de estructuras algebraicas a través de técnicas computacionales es bien conocido y constituye una línea de investigación habitual en Álgebra [vzGG99, CCS99]. En concreto, entre unas de las estructuras algebraicas que han sido estudiadas usando esta metodología se encuentran los semicuerpos finitos [Knu65]. Para ellos, se han desarrollado algunas técnicas computacionales para su estudio y potenciales aplicaciones. En el primer aspecto, se incluye la clasificación computacional de semicuerpos finitos [RCR09, RCR12]; en el segundo, la construcción de S-cajas criptográficas a partir de ellos [RC18]. En todos los casos mencionados anteriormente, los algoritmos considerados son clásicos.

Para el modelo de computación cuántica, se han propuesto algunos algoritmos para el estudio de estructuras algebraicas. Por ejemplo, consideremos el caso de determinar si una estructura (álgebra, anillo, grupo) es conmutativa o no. Este es un problema estudiado en diferentes contextos desde un punto de vista estrictamente teórico [Jac45, MW05, Pso84]. Desde el punto de vista efectivo, esto es, computacional, se ha estudiado especialmente en el caso de grupos, para los que se han propuesto algoritmos (no solamente clásicos aleatorizados sino también cuánticos) [MN05, Pak12].

En el caso de las álgebras finito dimensionales, la necesidad de un procedimiento efectivo para determinar la conmutatividad de la estructura es un problema natural en el contexto del estudio computacional de semicuerpos finitos. Para este problema se han propuesto diferentes algoritmos cuánticos, basados en el algoritmo de Grover [CRR19a], en métodos adiabáticos [CRR19c] y en caminos cuánticos [CRR19b]. En todos estos casos la detección de pares de constantes de estructura que no coinciden es el elemento fundamental sobre el que se articulan los algoritmos cuánticos considerados. Como consecuencia de estos estudios se ha desarrollado una abstracción de diversas técnicas cuánticas de detección bajo el formalismo de los Quantum Abstract Detecting Systems (QADS) [CRR20].

En esta tesis presentamos dos familias de QADS, llamadas QADS Combinatorios y QADS Rotacionales que, respectivamente, generalizan los sistemas de detección basados en puertas controladas por un solo qubit y en el algoritmo de Grover. Además, estudiamos sus propiedades, entre estas, el cierre algorítmico de cada familia, y demostramos que algunos de estos QADS son equivalentes (en el sentido de tener la misma tasa de detección) a otros construidos a partir del producto tensorial de operadores controlados y sus raíces cuadradas. También aplicamos la construcción de QADS combinatorios a un problema de decisión de valores propios, problema de estimación de fase y, adicionalmente, al problema de determinar la conmutatividad de álge-

bras finito-dimensionales. La segunda familia, QADS rotacionales, incluye un caso particular, los QADS de la búsqueda de Grover [HCCR22].

En relación con los métodos cuánticos que resuelven el problema del subgrupo oculto (por ejemplo, el algoritmo de Shor), como segunda parte de esta tesis, presentamos algoritmos cuánticos que encuentran subestructuras de manera eficiente, al formularlo como una instancia del problema del subgrupo oculto. Estos algoritmos cuánticos tienen orden de complejidad polinomial (cuántica) en la dimensión del álgebra [JER23].

Por último, desarrollamos métodos cuánticos para la clasificación computacional de semicuerpos finitos mediante técnicas computacionales cuánticas (basadas en el algoritmo de Grover) y su implementación, incluida la clasificación efectiva mediante simuladores cuánticos de semicuerpos finitos de tamaños pequeños [HCR23].

Publicaciones

Publicaciones en las que está basada esta tesis:

- Hernández Cáceres, J.M., Combarro, E.F. Rúa, I.F. Combinatorial and rotational quantum abstract detecting systems. *Quantum Inf Process* **21**, Paper No. 66, 27, (2022). [HCCR22]
- Hernández Cáceres, J.M., Rúa, I.F. An approach to the Classification of Finite Semifields by Quantum Computing. *Springer Proceedings in Mathematics & Statistics (PROMS, volume 427)* , 245-260, (2023) [HCR23]
- Hernández Cáceres, J.M., Rúa, I.F., Elías F. Combarro. Efficient Quantum Algorithms To Find Substructures On Finite Algebras. *Quantum Information & Computation* Vol.23 No.15&16 (2023) [JER23]

Abstract

On one hand, quantum computing is a well-established computing paradigm, from a theoretical point of view [NC11], and promising, from a practical point of view [Aea19].

On the other hand, the study of algebraic structures using computational techniques is well known and constitutes a common line of research in Algebra [vzGG99, CCS99]. Specifically, in the study and applications of finite Semifields [Knu65]. For these structures, some computational techniques have been developed for their classification. For instance, [RCR09, RCR12]; and also, for the construction of cryptographic S-boxes from them [RC18]. In all these cases, the algorithms considered are classical, understood as those that can be executed on an ordinary computer, that is, not quantum.

For the quantum computing model, some algorithms have been proposed for the study of algebraic structures. For example, consider the case of determining whether a structure (algebra, ring, group) is commutative or not. This is a problem studied in different contexts from a strictly theoretical point of view [Jac45, MW05, Pso84]. From the effective point of view, that is, computational, it has been studied especially in the case of groups, for which algorithms have been proposed (not only randomized classical, but also quantum) [MN05, Pak12].

In the case of finite-dimensional algebras, the need for an effective procedure to determine the commutativity of the structure is a natural problem in the context of the computational study of finite semifields. For this problem, different quantum algorithms have been proposed, based on Grover's algorithm [CRR19a], adiabatic methods [BF28] and quantum walks [CRR19b]. In all these cases, the detection of pairs of structure constants that do not coincide is the fundamental element on which the considered quantum algorithms are articulated. As a consequence of these studies, an abstraction of various quantum detection techniques has been developed under the formalism of Quantum Abstract Detecting Systems (QADS) [CRR20].

So, in this thesis we introduce two family of QADS, namely Combinatorial QADS and Rotational QADS which, respectively, generalize detecting systems based on single qubit controlled gates and on Grover's algorithm. Additionally, we study their properties, namely, the algorithmic closure of each family, and prove that some of these QADS are equivalent (in the sense of having the same detection rate) to others constructed from tensor product of controlled operators and their square roots. We also apply the combinatorial QADS construction to a problem of eigenvalue decision, and to a problem of phase estimation. Also to the problem of determining the commutativity of finite dimensional algebras. The second family, Rotational QADS includes as a particular case the QADS from Grover's search [HCCR22].

Next, in connection with the quantum methods that solve the hidden subgroup problem (such as Shor's algorithm), the study of problems related to the calculation of substructures

is proposed. In fact, we introduce quantum algorithms that find substructures efficiently, by formulating it as an instance of the Hidden Subgroup Problem. These quantum algorithms has a polynomial (quantum) complexity order in the dimension of the algebra [JER23].

Finally, we develop quantum methods for the computational classification of finite semifields by quantum computational techniques (based on Grover's algorithm) and their implementation, including the effective classification by quantum simulators of finite semifields of small sizes [HCR23].

Publications

Publications this thesis is based on:

- Hernández Cáceres, J.M., Combarro, E.F. Rúa, I.F. Combinatorial and rotational quantum abstract detecting systems. *Quantum Inf Process* **21**, Paper No. 66, 27, (2022). [HCCR22]
- Hernández Cáceres, J.M., Rúa, I.F. An approach to the Classification of Finite Semifields by Quantum Computing *Springer Proceedings in Mathematics & Statistics (PROMS, volume 427)* , 245-260, (2023). [HCR23]
- Hernández Cáceres, J.M., Rúa, I.F., Elías F. Combarro. Efficient Quantum Algorithms To Find Substructures On Finite Algebras. *Quantum Information & Computation.* Vol.23 No.15&16 [JER23]

To Maira, Alejandro and Oriana.

Chapter 1

Introduction

Ideas for Quantum Computing date back to the pioneering work of Feymann, Manin, Benioff, and others [Fey82], [Man80], [Ben80]. In fact, Quantum Computing emerges from the fields of quantum mechanics and computer science, when back in 1980, Paul Benioff showed that a computer could operate under the laws of quantum mechanics by introducing a Schrödinger equation description of Turing machines.

But what is Quantum Computing? Quantum Computing is a computational model based on exploiting quantum phenomena such as superposition (which gives the possibility that a qubit (quantum memory units) can be in the states $|0\rangle$ and $|1\rangle$ simultaneously, so n qubits can handle 2^n states in a moment), interference (which is used to affect probability amplitudes, in other words, every possible outcome has some probability of occurring), and entanglement (which refers to the fact that multiple qubits can be linked in such a way that their states are correlated, even when they are very far apart).

It has been one of the most intense and promising areas of research in complexity theoretical computer science in recent years, with some brilliant and remarkable results of quantum algorithms that outperform their classical counterparts.

For instance, in 1996, Lov Kumar Grover presents a quantum algorithm based on the concepts of superposition, and quantum parallelism (which is the ability to perform many calculations simultaneously, for example, the possibility that a function can be evaluated at many values at once), for searching databases, that is quadratically faster than any possible classical algorithm for the same purpose. More explicitly, a marked element in a list of N unordered ones (unstructured search), can be found by Grover's algorithm [Gro96] in expected time $O(\sqrt{N})$ (with probability $> 1/2$), and it has been shown [BBBV97] that Grover's algorithm is optimal in the sense that no quantum Turing machine can do this in the less than $O(\sqrt{N})$ operations in the black box model. Classicaly, this problem has complexity $\Omega(N)$.

In the computational (or effective) study of finite algebraic structures, Grover's algorithm has been successfully applied for testing the commutativity of a finite dimensional algebra [CRR19a],

achieving a quadratic speedup over the classical case [CRR19a]. Besides [CRR19a], there are also some quantum procedures for solving this task, that outperform their classical counterpart, which are: [CRR19b], based on quantum walks, and [CRR19c] based on adiabatic methods.

In all these cases, the authors were only interested in detecting if a witness of noncommutativity exist, i.e., the detection of certain pairs of structure constants that do not coincide. As a consequence of these studies, an abstraction of various quantum detection techniques has been developed under the formalism of, Quantum Abstract Detecting Systems (QADS) [CRR20]. Indeed, QADS were introduced as a common framework for the study and design of detecting algorithms in a quantum computing setting. Given a black-box oracle for a boolean function f, the QADS construct an initial state and an operator that can be used to detect if the function is identically zero or not. For instance, if U_f denotes a quantum oracle evaluating f, then the QADS related to Grover's algorithm [Gro96] constructs a uniformly superposed *initial state* $|\varphi_0\rangle$, and a quantum operator $G = U_s U_f$, product of the quantum oracle and the diffusion operator U_s. Such an operator can be used to evolve the quantum system from the initial state, so that measurement of the resulting state, $G^k |\varphi_0\rangle$, gives always $|\varphi_0\rangle$ when f is zero, where as when f is not zero, it gives the initial state with non-zero probability. These facts can be used to determine whether f is zero or not, i.e., to *detect* the existence of an element x such that $f(x) = 1$.

There are two main advantages to the introduction of the QADS methodology. The first one is that it helps to systematically analyse the effectiveness of the detection procedures under study. Namely, the actual usefulness of a particular QADS can be analized in terms of a trade-off between the precomputation cost of the QADS (efficient constructibility), and the number of iterations required to achieve a bounded success probability. The second advantage is that the methodology allows to construct new QADS from given ones, which might yield better detecting probabilities. These transformed QADS are members of the *algorithmic closure* of QADS. Most of these closure procedures are quite natural, such as extending the number of qubits used, inverting the detecting operator, multiplication of detecting operators with the same initial state, conjugation by a unitary operator, or control of a detecting operator with a qubit.

The possibility of constructing new QADS from existing ones leads to new families of QADS, such as combinatorial and rotational QADS. In Chapter 5, we introduce these new families of QADS. The combinatorial QADS generalise the well-known controlled operators. The rotational QADS, includes as a particular case the QADS from Grover's search. For them, we study the expression of the state after application of the detecting operator on the initial state, and their algorithmic closure. Interestingly, we derive some nice equivalences for these QADS in terms of tensor products and products of square roots of the original QADS. We also apply the combinatorial QADS construction to a problem of eigenvalue decision, to a problem of phase estimation, and to the problem of determining the commutativity of a finite dimensional algebra (which can be used to see whether a finite semifield is commutative or not).

So, as a second part of this thesis, we focus our attention on the classification of finite semifields. In fact, finite semifields are finite nonassociative rings with an identity element such that

the set of nonzero elements is a loop under the product. Their number of elements is a prime power, known as order. They were considered first by Dickson [Dic06], and studied by Albert [Alb60] and Knuth [Knu65]. Finite Semifields of order 16 have been classified by Kleinfeld in [Kle60], and of order 32 by Knuth in [Knu65] and by Walker in [Wal62]. The case of order 81 was solved by Dempwolff [Dem08].

The classification of finite semifields of order 64 was achieved by Rúa, Combarro, Ranilla in [RCR09], and of order 243 by Rúa, Combarro, Ranilla in [RCR12], based on the fact that classification of finite semifields can be rephrased as a problem of finding certain sets of matrices which can be solved by computer search. In this computer assited classifications, determining the commutativity of millions of algebras is important. Additionally, in the computational classification of finite semifields, an important task is the determination of substructures such as the right, middle, and left nuclei, the nucleus, and the center. Finding these structures may become computationally expensive when there is no additional information about the algebra properties, and there are millions of structures to classify.

On the other hand, if we carry on with quantum algorithms that outperform their classical counterparts, we have Simon's algorithm [Sim94]. Daniel Simon, in 1994, presented one of the first quantum algorithms to show an exponential speed-up versus the best classical algorithm in solving a specific problem.

Simon's algorithm uses $O(n)$ queries to solve Simon's problem. In his paper, he shows that the best classical algorithm requires $\Omega\left(2^{\frac{n}{4}}\right)$ queries (bounded error setting). Moreover, Simon's algorithm is significant because it paved the way for Shor's algorithm [Sho97]. In 1994, and based on the quantum Fourier transform (perhaps one of the most important unitary transformation in quantum computing), Peter Shor gave a quantum algorithm for factoring integers in polynomial time, while no classical algorithm with such a complexity is known. This is a breakthrough since, for instance, the RSA public-key cryptosystem is absolutely vulnerable to attackers that use this algorithm. These two algorithms (Simon's and Shor's) can be regarded under the framework of the hidden subgroup problem (HSP), which states the following:

Hidden Sugroup Problem. Let G be a finite group, and let $H \leq G$ be one of its subgroups. Let S be a set, and let $g : G \to S$ be a function that distinguishes cosets of H, i.e., for all $g_1, g_2 \in G, f(g_1) = f(g_2) \Leftrightarrow g_1H = g_2H$. The hidden subgroup problem (HSP) is to determine a generating set for the subgroup H given access to a black box that evaluates f on arbitrary elements.

Which can be rephrased as:

Given: The ability to evaluate a hiding function f, for a subgroup H of a finite group G, (i.e., a function f that is constant on a subgroup H of G, and is distinct on different cosets of H), on arbitrary elements of G.

Problem: Finding $s_1, s_2, \ldots, s_l$, a generating set for H.

For specific groups, abelian for instance, efficient quantum algorithms solving the HSP are known. As a second part of this thesis, we want to use this fact to efficiently compute substructures such as the right, middle, and left nuclei, the nucleus, and the center of finite algebras, and in particular of finite semifields. Therefore, we turn our attention on the following problem: Given the multiplication table of a $\mathbb{F}_p$-algebra A, with $\mathbb{F}_p$ a finite field of order p, consider the additive group $G = (A, +)$. We want to find algebraic substructures of A, such as the right, middle, and left nuclei, the nucleus and the center, using efficient quantum algorithms. These sets, which can be written in terms of an $\mathbb{F}_p$-basis β and the multiplication table of the algebra, provide information about the algebra. For instance, when A is a finite semifield, i.e., a finite division ring, these sets are related to properties of the corresponding coordinates projective planes [Alb60].

So, our problem is, then, stated as follows:

Given: Multiplication table of a finite dimensional $\mathbb{F}_p$-algebra A ($\mathbb{F}_p$ finite field of order p).

Problem: Finding $N_r(A), N_m(A), N_l(A), N(A)$, and $Z(A)$.

Here $N_r(A), N_m(A), N_l(A), N(A)$, and $Z(A)$, denote the right, middle, and left nuclei, the nucleus, and the center of the $\mathbb{F}_P$-algebra A, respectively. Now, in order to solve it with quantum techniques, we will transform each problem of finding $N_r(A), N_m(A), N_l(A), N(A)$ and $Z(A)$ into an instance of the HSP.

Thus, in Chapter 6, we explicitly and efficiently construct quantum circuits that, from the multiplication table of the n-dimensional algebra over a finite field $\mathbb{F}_p$, implement hiding functions f that can be used to determine these sets using only a polynomial number of quantum gates. Namely, of order $O(n^5 r^3)$, with $O(nr)$ queries to the oracle to find those sets, where $r = \lceil \log_2(p) \rceil$ (i.e., with an asymptotically linear number of evaluations of the function f for fixed p). This is achieved by suitable choices of functions f in the previous problem.

Finally, in this thesis we address the effective classification of finite binary semifields by quantum techniques. It was pointed out on [CRR19a] and [RCR12], that the classification of all finite semifields of size 128 is completely out of reach with current classical computing technology. Since Grover's Algorithm has a quadratic speedup for finding marked items in long lists, application of quantum computing to classify finite semifields seems promising.

So, we introduce a quantum procedure for classifying finite semifields with 8 and 16 elements, based on Grover's quantum search algorithm. We also discuss the scalability of the method to higher orders in Chapter 7. Thus, in order to classify a finite semifield of order 2^d, we show that with this method at least $d(d-1)^2$ qubits are required, together with an estimate on the number of quantum gates needed to build up the quantum circuit, showing that this approach is not as cheap as it would be desired. Indeed, we show, that the cost in terms of quantum gates would be at least:

Number	Gate
$6a+b$	$CNOTS$
$2a$	H
$3a$	$T^\dagger$
$4a$	T

where

$$a = (d+1)\sum_{k=1}^{d-1}\binom{d-1}{k}k^d(d-1)!(d-2)$$
$$b = \sum_{k=1}^{d-1}\binom{d-1}{k}k^d\left((d-1)!-1\right)(1+d)+2^{d-1}-1,$$

and d is the dimension of the binary finite semifield, over $\mathbb{F}_2$.

Overview of the Contents

More in detail, this thesis is organized in seven chapters. After this introductory chapter, Chapter 2 is meant to sum up all the algebraic foundations required for this thesis. For instance, we present notions on finite semifields and the way classification of finite semifields can be rephrased as a problem of finding certain sets of matrices.

After the algebraic preliminaries are given, we also give a background on Quantum Computing, that can be found in Chapter 3. There, we present notions on the quantum circuit model, i.e., we collect some basic notions on qubits, measurements, quantum gates and oracles that would be useful for the next chapters. Also, we recall the Quantum Fourier Transform over Abelian Groups, which would be use in Chapter 6.

In Chapter 4, we explain some well known quantum algorithms, such as Simon's Algorithm, the solution to the Hidden Subgroup Problem for the abelian case, and Grover's Algorithm. They play an important role on this thesis, as we can see, from Chapter 5 through Chapter 7. Lastly, in section 4.6, we explain what is a quantum abstract detecting system, that we use in Chapter 5.

In Chapter 5, we introduce combinatorial QADS and study their algorithmic closure. Rotational QADS are introduced and studied in Section 5.2, including their algorithmic closure. An application of the combinatorial QADS construction to a concrete eigenvalue decision problem is given in Section 5.3.

In Chapter 6, we model the problem of finding substructures in a finite-dimensional algebra as an instance of the HSP, and we show that, in some cases, it can not be classically solved with a polynomial number of function accesses to the hiding function f. In Section 6.3, we construct an efficient quantum oracle for the function f and we build an efficient circuit for the solution of the corresponding HSP.

Notation	Description
$\|\varphi\rangle$	Known as ket, represent a vector in $\mathbb{C}^n$.
$\langle\varphi\|$	Known as bra, is the transposed conjugate of $\|\varphi\rangle$.
$\|\varphi\rangle \otimes \|\psi\rangle$	Tensor product of $\|\varphi\rangle$ and $\|\psi\rangle$.
$\|\psi\rangle\|\varphi\rangle$	Abbreviated notation for tensor product of $\|\phi\rangle$ and $\|\psi\rangle$.
$\langle\psi\|U\|\varphi\rangle$	Inner product between $\|\psi\rangle$ and $U\|\varphi\rangle$ (U is an $n\times n$ complex matrix).
A^t	Transpose of the matrix A.
$\overline{A}$	Complex conjugate of the matrix A.
$A^\dagger$	$A^\dagger = \overline{A^t}$.
I_n	Denotes the identity matrix of size $n\times n$.
arg z	Denotes the argument of a complex number z.
Re z	Denotes the real part of a complex number z.

Table 1.2: Table of notations used in this dissertation.

In Chapter 7, using a simulator for quantum circuits, we find the multiplication tables for the finite semifield $\mathbb{F}_8$ (which is the only finite semifield of order 8), and for finite commutative semifields of order 16, based on Grover's quantum search algorithm. This chapter is meant to be a compendium of possible (unsatisfactory) approaches to the problem of classifying binary semifields using quantum computing techniques.

Finally, in Chapter 8, we give some conclusions of our study and print out some future work.

Additionally, a more detailed description of the contents of each chapter, and sections are given at its beginning, together with the sources used, in order to guide the reader through the text.

Notation and Conventions

In what follows, $\mathbb{C},\mathbb{R},\mathbb{Q},\mathbb{Z}$, and $\mathbb{N}$ stand for the field of complex numbers, the field of real numbers, the field of rational numbers, the ring of integers, and the set of natural numbers respectively. $\mathbb{Z}^+$ will denote the positive integers, and $\mathbb{C}^* = \mathbb{C} - \{0\}$, denotes the complex numbers without the zero. $\mathrm{Mat}_{n\times n}(K)$, with K a field, stands for the set of matrices of size $n\times n$ with entries in K. $\mathbb{Z}/p\mathbb{Z} = \{0,1,\ldots,p-1\}$ stands for the additive group of integers modulo p. For a real number x, by $\lfloor x \rceil$ we mean the nearest integer. We use the standard notation of quantum mechanics for linear algebraic concepts. Namely, Dirac notation, as presented in Table 1.2.

Acknowledgment

First and foremost, I would like to express my sincerest gratitude to my supervisors, Prof. Dr. Iñaki, and Prof. Dr. Elías, for your kindness, patient, guidance, encouragement, and constant support in every possible way, during all these years, virtually and in person. For the many explanations given, they help me grow academically, specially with your book Prof Dr. Elías. Thanks to both of you, for introducing me to such interesting path of research.

Prof. Dr. Iñaki thank you for support in many ways, not only academically, but also with bureaucracy issues.

I am grateful to Prof. Dr. Santos, for your selfless support, for letting me join the Cybercamp with the workshops and conferences. Also, I want to thank Dr. Luis Ovejero for the experience and personal grow in Satec HUB.

Lastly, I am really grateful with my family, for their endless support, love, and motivation, specially in tough times. Maira, Alejandro and Oriana you mean all to me.

Chapter 2

Algebraic Foundations

Nothing in this chapter is new, we only give overview of known facts. See, for instance, [DF04].

2.1 Groups

Definition 2.1. A group is a nonempty set G together with a binary operation $*$ on G such that $*$ is associative (that is, for any $a,b,c \in G,\ a*(b*c) = (a*b)*c$), there is an identity (or unity) element e in G such that for all $a \in G,\ a*e = e*a = a$, and for each $a \in G$, there exists an inverse element $a^{-1} \in G$ such that $a*a^{-1} = a^{-1}*a = e$. The order of a finite group is the number of its elements, and it is denoted by $|G|$.

If the group also satisfies that for all $a,b \in G,\ a*b = b*a$, then the group is called abelian. Usually, multiplicative notation is used, i.e., $*$ is written as $\cdot$, or simply as juxtaposition.

Example 2.2. The set $U(n) = \{A \in \mathrm{Mat}_{n\times n}(\mathbb{C}) : A^\dagger A = AA^\dagger = I_n\}$ is a group under multiplication of matrices, called unitary group.

Example 2.3. Let $G_1,\ldots,G_n$ be n groups, we consider $G = G_1 \times \cdots \times G_n$ to be the set of n-tuples with $x_i \in G_i$, for $i = 1,\ldots,n$, and componentwise multiplication. Then, G is a group called the direct product of groups, whose unit element is $(e_1,\ldots,e_n)$ (where e_i is the unit element of G_i).

An additive group is a group in which the group operation is to be thought of as addition, and $*$ is usually written as $+$.

Example 2.4. The additive group of integers modulo n is the group with domain $\{0,1,2,\ldots,n-1\}$, and with the operation of addition mod n. It is denoted as $\mathbb{Z}/n\mathbb{Z}$.

Definition 2.5. Let $H \subseteq G$. We say that H is a subgroup of G if H is nonempty, for every $a,b \in H, a*b \in H$, and for every $a \in H$, the inverse $a^{-1} \in H$.

Example 2.6. The set $SO(n) = \{A \in O(n) : \det(A) = 1\}$ is a subgroup of the orthogonal group $O(n) = \{A \in \mathrm{Mat}_{n\times n}(\mathbb{C}) : A^tA = AA^t = I_n\}$, called the special orthogonal subgroup.

Definition 2.7. Let G be a group and let S be a subset of G. We say that S generates G, if every element of G can be expressed as a product $x_1 \cdots x_n$ where each x_i or x_i^{-1} is in S. A cyclic group is a group which has one generator, i.e., if there exists $g \in G$ such that any $x \in G$ can be written as g^n, for some integer number n.

Definition 2.8. The order of an element $g \in G$ is the smallest positive natural number n such that $g^n = e$. If $g^r = e$ does not hold for all positive r, we say that the order is infinity.

In an additive group, the order of an element is the smallest positive integer n such that $x+^{(n}\cdots+x = 0$.

Definition 2.9. Let G be a group, and let $H \subseteq G$ be a subgroup or G. A left coset of H in G is a subset of the form $aH = \{ah : h \in H\}$, for some $a \in G$. The element a is a representative of the coset aH. The collection of left cosets is denoted G/H. Likewise, a right coset is a subset of the form $Ha = \{ha : h \in H\}$, for some $a \in G$.

If the group operation is written additively, as is often the case when the group is abelian, the notation changes to $a + H$. Two elements of $x, y \in G$, are called equivalent with respect to the subgroup H, if $xH = yH$, or, equivalently, if $x^{-1}y \in H$. This defines an equivalence relation in G, of which G/H is the quotient set. The number of left cosets of H is called the index of H in G and is denoted by $[G : H]$. The well-known Lagrange's Theorem states that the order of a subgroup must divide the order of a finite group.

Theorem 2.10 (Lagrange's Theorem)**.** *For any finite group G, if H is a subgroup of G, then $|G| = [G : H]|H|$.*

Definition 2.11. A subgroup N of a group G is called normal, if for every element of $g \in G$, $gNg^{-1} = N$. Here $gNg^{-1} = \{gng^{-1} : n \in N\}$.

Note that every subgroup of an abelian group is a normal subgroup.

Definition 2.12. A quotient group is defined in G/N, for any normal subgroup $N \subseteq G$. It is equipped with the operation $(gN) \circ (hN) = (gh)N$.

Definition 2.13. Let $(G, *)$ and $(H, \cdot)$ be two groups. A group homomorphism from $(G, *)$ to $(H, \cdot)$ is a function $\phi : G \to H$ such that, for all $g_1, g_2 \in G$, it holds that $\phi(g_1 * g_2) = \phi(g_1) \cdot \phi(g_2)$. A group homomorphism that is bijective, i.e., injective and surjective, is an isomorphism.

The following theorem says that a particular family of groups has a particular structure or form.

Theorem 2.14 (Fundamental Theorem of Finite Abelian Groups)**.** *Every finite abelian group G is isomorphic to a direct product of cyclic groups*

$$G \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}, \tag{2.1}$$

where $\mathbb{Z}/n_i\mathbb{Z} = \{0, 1, \ldots, n_i - 1\}$ is the additive group of integers modulo n_i.

2.2 Rings and Modules

Definition 2.15. A non-associative ring A is a set with two binary operations (addition and multiplication) such that A is an abelian group with respect to addition (so that A has a zero element, denoted by 0, and every $x \in A$ has an additive inverse, $-x$), and multiplication is distributive over addition, i.e. $x(y+z) = xy+xz, (y+z)x = yx+zx$, for all $x,y,z \in A$.

Definition 2.16. If a non-associative ring A satisfies the multiplicative associative law ($x(yz) = (xy)z$, for $x,y,z \in A$), then we shall say that it is an associative ring, or simply a ring. If it satisfies the commutative law ($xy = yx$, for all $x,y \in A$,) we will call it commutative.

We say that a non-associative ring A has identity or one, if there exists an element $1 \neq 0 \in A$, such that $1x = x1 = x$, for all $x \in A$.

Definition 2.17. A non-associative ring with identity is called a division ring if, for all $x \in A$, there exists $y,z \in A$ such that $xy = zx = 1$. When A is finite, then it is called a finite semifield. When it is associative and commutative, then it is called a field.

Next, we shall assume that rings are (associative) commutative, and with identity.

Definition 2.18. A subset S of a ring A is called a *subring*, if S is a ring with the restrictions of multiplication and addition to S. An *ideal* $\mathfrak{a}$ of a ring A is a subset of A which is an additive subgroup and is such that $A\mathfrak{a} = \{ab \,:\, a \in A, b \in \mathfrak{a}\} \subseteq \mathfrak{a}$, and $\mathfrak{a}A \subseteq \mathfrak{a}$. For any $a \in A$, the set Aa is an ideal of A, known as ideal generated by a (or a *principal ideal*). It will be denoted as $(\mathfrak{a})$.

Definition 2.19. A *zero-divisor* in a ring A is a non-zero element x which divides 0, i.e., for which there exists $y \neq 0$ in A such that $xy = 0$. A ring with no zero-divisors is called an *integral domain*.

A *unit* in A is an element x which divides 1, i.e., an element x such that $xy = 1$ for some $y \in A$. The element y is then uniquely determined by x, and is written as x^{-1}. The units in A form an abelian group.

A *principal ideal domain* (PID) is an integral domain in which every ideal is principal.

Example 2.20. For example, let K be a field, let x be an indeterminate over K, then $K[x]$, called the polynomial ring in x, with coefficients in K, is a principal ideal domain. It is well-known that this makes $K[x]$ a unique factorization domain, i.e., any non-constant polynomial can be uniquely (up to the order of the factors) written as the product of irreducible polynomials (those that can be only divided by constants and themselves, up to a multiplicative constant).

Definition 2.21. If $\mathfrak{a}$ is an ideal of a ring A, then the quotient group $A/\mathfrak{a}$ (whose elements are the cosets of $\mathfrak{a}$ in A) inherits a uniquely defined multiplication from A which makes it into a ring, called the quotient ring $A/\mathfrak{a}$.

Definition 2.22. A (unitary) A-module is an abelian group M, written additively, together with a map $A \times M \to M$, written by am for all $a \in A$, and all $m \in M$, which satisfies that $(a+b)x = ax+bx$, $a(x+y) = ax+by$, $(ab)x = a(bx)$, for all $a,b \in A$, and for all $x,y \in M$; and also, $1m = m$, for all $m \in M$.

Example 2.23. Note that when $A = K$ is a field, then the A-module M is a K-vector space.

Example 2.24. A commutative ring with identity A is an A-module, where the multiplicative map is the ring product.

Definition 2.25. Let M,N be A-modules. A mapping $f : M \to N$ is an *A-module homorphism* if $f(x+y) = f(x) + f(y)$, and $f(ax) = a \cdot f(x)$, for all $a \in A$ and all $x,y \in M$ (also note that when A is a field, an A-module homomorphism is a linear transformation of vector spaces). A module homomorphism is called a *module isomorphism* if it admits an inverse homomorphism; in particular, it is a bijection. It will be denoted as $M \cong N$.

Definition 2.26. A module M is *cyclic* if there exists an element $x \in M$ such that M is generated by one element: $M = Ax = \{ax \ : \ a \in A\}$.

Example 2.27. If A is a commutative ring with identity, then any principal ideal $\mathfrak{a} \subset A$ is a cyclic A-module.

Definition 2.28. If M,N are A-modules, their *direct* sum $M \oplus N$ is the set of all pairs (x,y) with $x \in M, y \in N$. This is an A-module with componentwise addition an scalar multiplication $a(x,y) = (ax,ay)$.

Example 2.29. Let $A = K[x]$ be the polynomial ring in the indeterminate x, with coefficients in a field K. Let V be a finite dimensional vector space over K of dimension n. Let $T : V \to V$ be a linear map, and let $p(x) \in K[x]$. Consider the action of the ring element $p(x) = \sum_{k=1}^{n} a_k x^k$ on $v \in V$ as

$$p(x)v = \sum_{k=0}^{n} a_k T^k(v),$$

where $T^k = T \circ T \circ \cdots \circ T$, and $\circ$ denotes function composition (with $T^0 = I : V \to V$ the identity map). With this map $K[x] \times V \to V$, V can be seen as a $K[x]$-module.

Recall the following notions from linear algebra. Let V be a K-vector space of dimension n, and let $T : V \to V$ be a linear map.

Definition 2.30. Let λ be an indeterminate over a field K, and $A \in \mathrm{Mat}_{n\times n}(K)$. We call $p_A(\lambda) = \det(\lambda I_n - A)$ the characteristic polynomial of A. If A is the coordinate matrix of T with respect to one of its bases, then the polynomial $p_T(\lambda) = \det(\lambda I - A)$ is called the characteristic polynomial of T. It will be denoted as p_T. It is independent of the choice of basis.

Definition 2.31. Let $m(x) \in K[x]$ be the unique monic polynomial generating the annihilator ideal of V in $K[x]$, i.e., $\mathrm{Ann}(V) = \{p(x) \in K[x] \ : \ p(T) = 0\}$. Equivalently, $m(x)$ is the unique monic polynomial of minimal degree annihilating V, i.e., such that $m(T) = 0$ (here $0 : V \to V$ is the null linear transformation) (And so, if $f(x) \in K[x]$ is any polynomial annihilating V, then $m(x)|f(x)$). It is called the minimal polynomial of T, and it will be denoted as $m_T(x)$. The unique monic polynomial of least degree which annihilates the matrix $A \in \mathrm{Mat}_{n\times n}(K)$, is called the minimal polynomial of A (i.e., such that $p(A) = 0$), and it will be denoted as $m_A(x)$.

The following theorem states the structure of the $K[x]$-module V.

Theorem 2.32. *(Fundamental Theorem of endomorphisms of a finite dimensional vector space) Let V a finite dimensional K-vector space, and let $T : V \to V$ be a linear map. Let x be an indeterminate over K. Then as $K[x]$-modules,*

1. $$V \cong K[x]/(a_1(x)) \oplus K[x]/(a_2(x)) \oplus \cdots \oplus K[x]/(a_m(x))$$
   where $a_1(x), a_2(x), \ldots, a_m(x)$ are monic polynomials (called invariant factors) in $K[x]$ of degree at least one with the divisibility conditions
   $$a_1(x)|a_2(x)|\cdots|a_m(x).$$
2. *$Ann(V) = (a_m(x))$. And so the minimal polynomial $m_T(x)$ is the largest invariant factor of V. All the invariant factors of V divide $m_T(x)$.*
3. *The characteristic polynomial T is the product of all invariant factors of T.*
4. *(Cayley-Hamilton) The minimal polynomial of T divides the characteristic polynomial of T.*
5. *The characteristic polynomial of T divides some power of the minimal polynomial of T. In particular, they have the same irreducible factors.*

As a consequence, we get:

Proposition 2.33. Let $A \in \mathrm{Mat}_{n\times n}(K)$. Then

1. (The Cayley-Hamilton Theorem) The minimal polynomial of A divides the characteristic polynomial of A.
2. The characteristic polynomial of A divides some power of the minimal polynomial of A. In particular, these polynomials have the same irreducible factors, not counting multiplicities

2.3 K-algebras

Definition 2.34. A finite dimensional algebra A over a field K (or more simply a K-algebra) is a non-associative ring A that has the structure of a K-vector space of dimension d, such that for all $b, c \in A$ and $\alpha \in K, \alpha(b \cdot c) = (\alpha b) \cdot c = b(\alpha c)$.

Note that we not necessarily have a commutative nor associative product, or identity. Throughout this thesis, we will use the term non-associative algebra to emphasize the fact that the multiplication may not be associative, and non-commutative that might not be commutative. An algebra is called a division algebra, if it is a division ring. Now, let S be a non-associative

non-commutative finite dimensional algebra over a field K. Let us fix a K-basis $\beta = \{x_1, \ldots, x_d\}$ of S, so there exists a unique set of constants $\{M_{ijk}\}_{i,j,k=1}^{d} \subseteq K$, such that,

$$x_i \cdot x_j = \sum_{k=1}^{d} M_{ijk} x_k, \text{ for all } i, j \in \{1, \ldots, d\}.$$

That set is known as the multiplication table (or structure constants) of the algebra (with respect to the basis β). For each $x \in S$, consider the maps $L_x : S \to S$, and $R_x : S \to S$, given by $L_x(a) = x \cdot a$, and $R_x(a) = a \cdot x$, respectively. Both are K-linear homomorphisms. Additionally, $L_{x_i}(x_j) = x_i \cdot x_j = \sum_{k=1}^{d} M_{ijk} x_k$. Let $x = \sum_{j=1}^{d} \alpha_i x_i$, with $\alpha_i \in K$. Then,

$$L_x(a) = x \cdot a = \sum_{j=1}^{d} \alpha_i x_i \cdot a = \alpha_1 (x_1 \cdot a) + \cdots + \alpha_d (x_d \cdot a)$$

$$= \alpha_1 L_{x_1}(a) + \cdots + \alpha_d L_{x_d}(a) = \sum_{i=1}^{d} \alpha_i L_{x_i}(a).$$

So, the map L_x can be described by the maps $L_{x_1}, \ldots, L_{x_d}$. Denote the column coordinate matrix of L_{x_i} with respect to β by:

$$A_i = \begin{bmatrix} M_{i11} & M_{i21} & M_{i31} & \ldots & M_{id1} \\ M_{i12} & M_{i22} & M_{i32} & \ldots & M_{id2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{i1d} & M_{i2d} & M_{i3d} & \ldots & M_{idd} \end{bmatrix}.$$

Consider the set $\mathbb{M} := \{A_1, \ldots, A_d\}$. Then, if $x = \sum_{i=1}^{d} \alpha_i x_i$, with $\alpha_i \in K$, then $L_x = \sum_{i=1}^{d} \alpha_i L_{x_i}$, and so we have that the column coordinate matrix of L_x with respect to β is

$$A_x = \sum_{i=1}^{d} \alpha_i A_i.$$

Which justifies that $\mathbb{M}$ is a multiplication table for S (all the products can be described by d^3 constants in K). Furthermore, we can see some properties of the algebra in terms of $\mathbb{M}$. For instance, note that $x \neq 0$ is not a left zero divisor (i.e., there does not exist a $y \neq 0$ such that $xy = 0$), if and only if L_x is a K-linear isomorphism if and only if A_x is invertible. The same holds for $x \neq 0$ not being a right zero divisor. Since for any x there exist $\alpha_1, \ldots, \alpha_d \in K$ such that $x = \alpha_1 x_1 + \cdots \alpha_n x_n$, then S is a division algebra if and only if any non zero linear combination of $\mathbb{M}$ is invertible.

Definition 2.35. Let A be a non-associative non-commutative finite dimension algebra. If $a, b \in A$, we define the commutator as $[a,b] = ab - ba$. We define the associator as the multilinear map $[\cdot,\cdot,\cdot] : A \times A \times A \to A$, given by $[x,y,z] = (xy)z - x(yz)$.

The commutator measures the non-commutativity of A. For instance, A is a commutative algebra if and only if $[x_i, x_j] = 0$, for all $i, j = 1, \ldots, d$, i.e.,

$$0 = [x_i, x_j] = x_i x_j - x_j x_i = \sum_{k=1}^{d} M_{ijk} x_k - \sum_{k=1}^{n} M_{jik} x_k = \sum_{k=1}^{d} M_{ijk} x_k - M_{jik} x_k = \sum_{k=1}^{d} \left(M_{ijk} - M_{jik}\right) x_k,$$

if and only if $M_{ijk} = M_{jik}$, for all $i, j, k = 1, \ldots, n$.

The associator measures the non-associativity of A, so A is associative if and only if, the associator is identically zero, i.e., for every $i, j, k = 1, \ldots, d$,

$$\begin{aligned}
0 = [x_i, x_j, x_k] &= \left(x_i \cdot x_j\right) x_k - x_i \left(x_j \cdot x_k\right) \\
&= \left(\sum_{m=1}^{d} M_{ijm} x_m \right) x_k - x_i \left(\sum_{m=1}^{d} M_{jkm} x_m \right) \\
&= \sum_{m=1}^{d} M_{ijm} x_m \cdot x_k - \sum_{m=1}^{d} M_{jkm} x_i \cdot x_m \\
&= \sum_{m=1}^{d} \sum_{l=1}^{d} M_{ijm} M_{mkl} x_l - \sum_{m=1}^{d} \sum_{l=1}^{d} M_{jkm} M_{iml} x_l \\
&= \sum_{m=1}^{d} \sum_{l=1}^{d} M_{ijm} M_{mkl} x_l - M_{jkm} M_{iml} x_l \\
&= \sum_{m=1}^{d} \sum_{l=1}^{d} \left(M_{ijm} M_{mkl} - M_{jkm} M_{iml}\right) x_l.
\end{aligned}$$

So, A is associative if and only if for every $i, j, k, l = 1, \ldots, d$, we have that ,

$$\sum_{m=1}^{d} \left(M_{ijm} M_{mkl} x_l - M_{jkm} M_{iml}\right) = 0.$$

Definition 2.36. Consider the following sets, known as the right, middle, and left nuclei, the nucleus and the center, of an algebra over K:

$$\begin{aligned}
N_r(A) &= \{a \in A : [x, y, a] = 0, \text{ for all } x, y \in A\} \\
N_m(A) &= \{a \in A : [x, a, y] = 0, \text{ for all } x, y \in A\} \\
N_l(A) &= \{a \in A : [a, x, y] = 0, \text{ for all } x, y \in A\} \\
N(A) &= N_r(A) \cap N_m(A) \cap N_l(A) \\
Z(A) &= N(A) \cap \{a \in A : [a, x] = 0, \text{ for all } x \in A\}
\end{aligned}$$

These sets are subalgebras of A, i.e., K-vector subspaces closed under multiplication.

2.4 Finite Semifields

In this section, we collect definitions and facts on finite semifields, which is the class of finite division algebras. Proofs can be found, for instance, in [Knu65], [HR07]. The term finite semifield was introduced in 1965 by Knuth. However, in 1906, and 1960, the concept of finite semifield had been previously studied by Dickson [Dic06] and Albert [Alb60], respectively.

Definition 2.37. A finite non-associative ring D is called presemifield, if the set of nonzero elements D^* is closed under the product. If D has an identity element, then it is called (finite) semifield (i.e., if it is a division ring).

Example 2.38. Any finite field $\mathbb{F}_q$ of q elements is a finite semifield.

If D is a finite semifield, then D^* is a multiplicative loop. That is, there exists an element $e \in D^*$ (the identity of D) such that $1x = x1 = x$, for all $x \in D$, and for all $a, b \in D^*$, the equation $ax = b$ (respectively $xa = b$) has a unique solution.

Definition 2.39. Let S be a finite semifield. Its cardinality is called order and denoted as $|S|$. The additive order of the identity of S is called the characteristic of S.

Proposition 2.40. Let D be a finite semifield. The characteristic of D is a prime number p. Its center is a finite field $\mathbb{F}_q$ of $q = p^c$ elements, where $c \in \mathbb{N}$. Moreover, D is a finite-dimensional algebra over $Z(D)$ of dimension d, where $|D| = q^d$. Also, the nuclei $N_r(D), N_m(D), N_l(D)$, and the nucleus $N(D)$ of D are finite fields.

It is well known that for every prime p and every positive integer n there exists a finite field with p^n elements [LN96]. Any associative finite semifield is necessarily commutative and hence a finite field (by Wedderburn's Theorem [MW05]). Nevertheless, not every finite semifield is necessarily associative. If a finite semifield is not associative, it is called proper.

Theorem 2.41 ([Knu65])**.** *A proper semifield has order p^n, where $n \geq 3$, and $p^n \geq 16$.*

Knuth proved that a proper semifield has at least has 16 elements by showing that the only finite semifield of order 2^3 is the finite field $\mathbb{F}_8$. As an example of a proper semifield we have:

Example 2.42. Consider the field $F = \mathbb{F}_4$, with the elements $0, 1, \omega$ and $\omega^2 = 1 + \omega$. Let $V = \{u + \lambda v : u, v \in F\}$, and define

$$\begin{array}{rccc} + : & V \times V & \to & V \\ & (u_1 + \lambda v_1, u_2 + \lambda v_2) & \mapsto & (u_1 + v_1) + \lambda\,(u_2 + v_2) \end{array}$$

$$\begin{array}{rccc} \cdot : & V \times V & \to & V \\ & (u_1 + \lambda v_1, u_2 + \lambda v_2) & \mapsto & \left(u_1 u_2 + v_1^2 v_2\right) + \lambda\left(v_1 u_2 + u_1^2 v_2 + v_1^2 v_2^2\right) \end{array}$$

Then, V is a finite semifield with identity $1 + \lambda 0$. Note that V is not commutative, and hence it is a proper semifield.

Another example is Knuth's binary semifield:

Example 2.43. Let n be odd, and $mn > 3$. Consider $\mathbb{F}_{2^{mn}}$ as an $\mathbb{F}_{2^m}$-vector space, let $f : \mathbb{F}_{2^{mn}} \to \mathbb{F}_{2^m}$ be the unique linear functional, i.e., $f(\alpha a + \beta b) = \alpha f(a) + \beta f(b)$, for all $a, b \in \mathbb{F}_{2^{mn}}$, and all $\alpha, \beta \in \mathbb{F}_{2^m}$, such that $f(1) = 1$, and $f(x) = f(x^{2^m})$. Define a multiplication in $\mathbb{F}_{2^{mn}}$ as follows:

$$\begin{array}{rccc} \circ: & \mathbb{F}_{2^{mn}} \times \mathbb{F}_{2^{mn}} & \to & \mathbb{F}_{2^{mn}} \\ & (a,b) & \mapsto & ab + (f(a)b + f(b)a)^2 \end{array}$$

Now, define a product $*$ in the vector vector space as $(1 \circ a) * (1 \circ b) = a \circ b$. It can be shown that $(\mathbb{F}_{2^{mn}}, +, *)$ is a proper commutative semifield, known as Knuth's binary semifield.

It is well-known that, for a finite field $\mathbb{F}_q$, the set $\mathbb{F}^* = \mathbb{F}_q - \{0\}$ is a cyclic group. We shall call primitive element to any generator of $\mathbb{F}_q^*$. It is also well-known that, for any $n \in \mathbb{N}$ the Galois group $G\left(\mathbb{F}_q^n | \mathbb{F}_q\right)$ (i.e., the set of all $\mathbb{F}_q$-automorphism of $\mathbb{F}_{q^n}$) is a cylic group of order n.

Example 2.44. Albert's generalized *twisted fields* were introduced in [Alb61] as a generalization of other families of semifields that Albert himself had previously discovered. Its construction is based on the deformation of the product of a finite field, hence the name of *"twisted fields"*.

Let $F = \mathbb{F}_q$ with $q = p^r > 2$, and $K = \mathbb{F}_q^m$ with $m \geq 3$. Let $\sigma, \tau : K \to K$ be elements in the Galois group $G(K|F)$ such that

$$< \sigma >^{\perp} \cap < \tau >^{\perp} = F \cap F = F,$$

where $< \sigma >^{\perp} = \{a \in K : \sigma(a) = a\}$. Consider the product operation defined as:

$$a \bullet b = ab - g\tau(a)\sigma(b),$$

for all $a, b \in K$, which makes the group $(K, +)$ a finite presemifield. Thus, the applications $L_1^{\bullet}, R_1^{\bullet} : K \to K$ are bijections, and the group $(K, +)$ with the new operation:

$$a * b = (R_1^{\bullet})^{-1}(a) \bullet (L_1^{\bullet})^{-1}(b),$$

for all $a, b \in K$, is a finite semifield, called generalized twisted field. It is a proper finite semifield if and only if $\sigma \neq \tau$.

Note that its identity is $f = 1 - g$, and its nucleus and center are $N = Z = Ff$, its right nucleus $N_r = < \sigma >^{\perp} f$, its left nucleus $N_l = < \tau >^{\perp} f$, and its middle nucleus is $N_m = R_1^{\bullet}(M) = L_1^{\bullet}(M)$, where $M = \{a \in K \ : \ \sigma(a) = \tau(a)\}$.

Definition 2.45. Let D be a finite semifield, and $a \in D$. We inductively define the left principal powers of a as

$$a^{(0} = 1, \text{ for all } i \in \mathbb{N} : a^{(i+1} = aa^{(i},$$

and the right principal powers of a as

$$a^{0)} = 1, \text{ for all } i \in \mathbb{N} : a^{i+1)} = aa^{i)}.$$

Definition 2.46. A finite semifield D is called left primitive semifield, if it possesses an element ω such that D^* is the set of all left principal powers of ω. The element ω is called a left primitive element. D is called right primitive semifield, if it possesses an element ω such that D^* is the set of all right principal powers of ω. The element ω is called a right primitive element. The semifield is called primitive, if its both left and right primitive.

Example 2.47. Any finite field is a primitive semifield.

Example 2.48. Knuth's binary semifield of order 32 [Knu65] is neither left nor right primitive. This was shown by [Rú04].

Definition 2.49. A polynomial $f \in \mathbb{F}_q[x]$ of degree $m \geq 1$ is called primitive over $\mathbb{F}_q$ if f is monic, $f(0) \neq 0$, and $\text{ord}(f) = q^m - 1$. Here, the order of f (denoted as $\text{ord}(f)$) means the least positive integer l for which $f(x)$ divides $x^l - 1$. Any primitive polynomial is irreducible [LN83].

Proposition 2.50 ([HR07])**.** If D is a finite semifield of dimension d over its center $Z(D) = \mathbb{F}_q$, then $\omega \in D$ is a left primitive element of D if and only if the characteristic polynomial of the linear map $L_\omega : D \to D$, is a primitive polynomial of degree d over $Z(D)$.

Corollary 2.51 ([HR07])**.** *If ω is a left primitive element of a finite semifield D, then $\{1, \omega, \omega^{(2}, \ldots, \omega^{(d-1}\}$ is a $Z(D)$-basis of the algebra D.*

An effective description of finite semifields can be given in terms of matrices, as the following proposition shows.

Proposition 2.52 ([RCR09])**.** Any finite semifield D of order q^d and center containing $\mathbb{F}_q$ can be described by a set of d matrices $\{A_1, \ldots, A_d\}$, known as *standard basis*, such that

1. A_1 is the identity matrix.
2. $\sum_{i=1}^{d} \alpha_i A_i$ is invertible for all nonzero tuples $(\alpha_1, \ldots, \alpha_d) \in \mathbb{F}_q^d$.
3. The first column of the matrix A_i is the column vector with a 1 in the i-th position, and 0 everywhere else.

The semifield D can be identified with the algebra $(\mathbb{F}_q^d, +, \cdot)$, where the multiplication is given by $x \cdot y = \sum_{i=1}^{d} x_i A_i y$. As a consequence of this proposition, we have:

Corollary 2.53.
1. *[Alb60] If $\mathbb{F}_q$ is the center of D, then any non-scalar linear combination of a standard basis (i.e., not of the form λA_1) has a characteristic polynomial without linear factors.*
2. *In the conditions of Proposition 2.52: [HR07] D is not left primitive if and only if for any non-scalar linear combination of a standard basis, its characteristic polynomial is not a primitive polynomial.*

2.5 Character Theory of Finite Abelian Groups

In this section, let G be a finite abelian group. By theorem 2.14, G is isomorphic to a product of cyclic groups

$$G \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}, \tag{2.2}$$

where $\mathbb{Z}/n_i\mathbb{Z} = \{0,1,\ldots,n_i-1\}$ is the additive group of integers modulo n_i. Let $\varphi : G \to \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$ be a group isomorphism. For $g \in G$, denote $\varphi(g)$ as k-tuples: $\varphi(g) = (g_1,\ldots,g_k)$, with $g_i \in \mathbb{Z}/n_i\mathbb{Z}$. Write $-\varphi(g)$ for the (additive) inverse of $g \in G$ in $\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$. Denote the identity of G as 1, so $\varphi(1) = (0,0,\ldots,0)$. Let $\varphi(e_1) = \beta_1 = (1,0,0,\ldots,0), \varphi(e_2) = \beta_2 = (0,1,0,\ldots,0), \ldots, \varphi(e_k) = \beta_k = (0,0,0\ldots,1) \in G$. Then,

$$\varphi(g) = \sum_{j=1}^{k} g_j\beta_j.$$

We shall use this additive representation of G.

Definition 2.54. A character of a group G is a group homomorphism χ from G to the multiplicative group of nonzero complex numbers $\mathbb{C}^* = \mathbb{C} - \{0\}, \chi : G \to \mathbb{C}^*$.

Let $g = (g_1,\ldots,g_k) \in G$, so

$$\chi(g) = \chi\left(\sum_{j=1}^{k} g_j\beta_j\right) = \prod_{j=1}^{k} \chi\left(\beta_j\right)^{g_j}.$$

Hence, χ is completely determined by $\beta_1,\ldots,\beta_k$. Since the order of β_j is n_j, $\chi\left(\beta_j\right)$ has order dividing n_j, so $\chi\left(\beta_j\right) = \omega_{n_j}^{h_j}$, for some $h_j \in \mathbb{Z}/n_j\mathbb{Z}$, and $\omega_{n_j} = \exp\left(\frac{2\pi i}{n_j}\right)$ (a complex primitive n_j-th root of unity). Now, any given character $\chi : G \to \mathbb{C}^*$ is determined by $(h_1,\ldots,h_k)$, with $h_j \in \{0,1,\ldots,n_j\}$. Therefore, each character χ can be labelled by an element of G. Reciprocally, any of such labellings determines a character χ.

Definition 2.55. For each $g \in G$, we define the character,

$$\begin{array}{rccl} \chi_g : & G & \to & \mathbb{C}^* \\ & h & \mapsto & \chi_g(h) = \prod_{j=1}^{k} \omega_{n_j}^{g_j h_j}. \end{array}$$

Proposition 2.56. Let $\chi(G)$ denote the set $\{\chi_g : g \in G\}$ of all such maps. Then, $\chi(G)$ is a group under $\chi_g \cdot \chi_h = \chi_{g+h}$, and $\chi(G)$ is isomorphic to G.

Definition 2.57. Given a subgroup H of a finite abelian group G, its orthogonal subgroup $H^\perp$ is defined as the set of all elements in G orthogonal to H, i.e.,

$$H^\perp = \{g \in G : \chi_g(h) = 1, \text{ for all } h \in H\}.$$

It follows that $H^\perp$ is a subgroup of G, since the identity $1 \in G$ is in $H^\perp$ ($\chi_1(g) = 1$, for all $g \in G$), and if $a, b \in H^\perp$ then, for any $h \in H$, we have $\chi_h(a-b) = \chi_h(a)/\chi_h(b) = 1$, thus $a - b \in H^\perp$.

Proposition 2.58. $H^\perp \cong G/H$, and $\left(H^\perp\right)^\perp = H$.

In particular, if $G \cong (\mathbb{Z}/2\mathbb{Z})^n$, $(\mathbb{Z}/2\mathbb{Z})^n$ naturally comes with a group structure given by the (component-wise) XOR (addition mod 2) between bit vectors: let $x, y \in (\mathbb{Z}/2\mathbb{Z})^n$, with $x = (x_1, \ldots, x_n)$, and $y = (y_1, \ldots, y_n)$. So, $(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) = (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$. By $x \cdot y$, let us denote the inner product modulo 2 of x and y, i.e.,

$$x \cdot y = \left(\sum_{i=1}^{n} x_i y_i \right) \mod 2.$$

Therefore, if H is a subgroup of G, and since $\chi_x(y) = \prod_{j=1}^{n} (-1)^{x_j y_j} = 1$ if and only if $\sum_{j=1}^{n} x_j y_j \equiv 0 \mod 2$, we have

$$H^\perp = \{x \in G : \chi_x(y) = 1, \text{ for all } y \in H\} = \{x \in G : x \cdot y = 0, \text{ for all } y \in H\}.$$

This means that, if x, y are binary vectors in an n-dimensional $\mathbb{F}_2$-vector space, then H is a subspace, and $H^\perp$ is the orthogonal complement to H (with respect to the inner product $\cdot$).

In general, if $G \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$, let m be the least common multiple of all n_i's. Any root of unity ω_{n_i} can be written as a power of ω_m, where $m = \operatorname{lcm}(n_1, \ldots, n_k)$. Indeed, $\omega_{n_i} = \omega_m^{\frac{m}{n_i}}$. Let $x = (g_1, g_2, \ldots, g_n), y = (h_1, h_2, \ldots, h_n)$ be elements of $\mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$. Then,

$$\chi_g(h) = \prod_{j=1}^{k} \omega_{n_j}^{g_j h_j} = \prod_{j=1}^{k} \left(\omega_m^{\frac{m}{n_j}} \right)^{g_j h_j} = \prod_{j=1}^{k} \left(\omega_m^{\frac{m g_j h_j}{n_j}} \right) = \omega_m^{\frac{m g_1 h_1}{n_1} + \cdots + \frac{m g_k h_k}{n_k}},$$

which means that

$$H^\perp = \{x \in G : \chi_x(y) = 1, \text{ for all } y \in H\} = \left\{ x \in G : \frac{m g_1 h_1}{n_1} + \cdots + \frac{m g_k h_k}{n_k} = 0, \text{ for all } y \in H \right\}.$$

In particular, if $G \cong (\mathbb{Z}/p\mathbb{Z})^n$, then,

$$\begin{aligned} H^\perp &= \{x \in G : \chi_x(y) = 1, \text{ for all } y \in H\} \\ &= \{x \in G : g_1 h_1 + \cdots + g_k h_k = 0, \text{ for all } y \in H\}. \end{aligned}$$

Finally, let us recall this well-known fact:

Proposition 2.59. Let ω be a complex n-th root of unity (i.e., $\omega^n = 1$, and $\omega^k \neq 1$, if $1 < k < n$), and suppose that $\omega \neq 1$. Then, $\sum_{k=0}^{n-1} \omega^k = 0$.

2.6 Probability of Generating a Group

In this subsection, we follow [Pak00]. We are interested in the probability that a certain amount of elements chosen uniformly at random from a finite group will generate the whole group.

Let $\varphi_k(G)$ denote the probability that k random elements of G generate the entire group, i.e.,

$$\varphi_k(G) = Pr(\langle g_1, g_2, \ldots, g_k \rangle = G),$$

where g_i are elements of G, chosen independently and uniformly at random from G. For a finite cyclic group $G = \langle g \rangle$, we know that $G = \langle g^k \rangle$ if and only if $\gcd(n,k) = 1$. So, for instance, if $|G| = n$, then G has $\varphi(n)$ generators, where φ denotes the Euler Phi function. Hence, the probability of any randomly chosen element to be a generator of the group is

$$\frac{\varphi(n)}{n} = \frac{n \prod_{p|n} \left(1 - \frac{1}{p}\right)}{n} = \prod_{p|n} \left(1 - \frac{1}{p}\right).$$

In general, we have:

Lemma 2.60. *Let G be any finite group or order $|G| \leq 2^r, r \geq 1$. Then, for all $t \geq 1, \varphi_t(G) \geq \varphi_t(\mathbb{Z}_2^r)$.*

As a consequence of

Lemma 2.61. *$\varphi_{r+t}(\mathbb{Z}_2^r) \geq 1 - \frac{1}{2^t}$, for $t \geq 0$.*

We get,

Theorem 2.62. *Let G be a finite group. For an integer $t \geq 0$, the probability that $k = t + \log_2 |G|$ elements chosen uniformly at random from G will generate G is bounded by*

$$\varphi_k(G) \geq 1 - \frac{1}{2^t},$$

for $t \geq 0$.

For completeness, in this section, we present a result on the probability of coprimality of integers uniformly sampled from a fixed range.

Lemma 2.63 ([Lom04])**.** *Suppose that we have $k \geq 2$ uniformly random samples $t_1, \ldots, t_k$ from the integers $\{0, \ldots, d-1\}$, with $d \geq 2$. Then*

$$Pr(gcd(t_1, \ldots, t_k) = 1) \geq 1 - \left(\frac{1}{2}\right)^{\frac{k}{2}}.$$

Chapter 3

Quantum preliminaries

The main purpose of this chapter is to give an explicit explanation of the quantum circuit model, one of the most popular quantum computing paradigms, in order to understand quantum algorithms. Additionally, we give the conventions and choices of notation used throughout this thesis. As a last section, we briefly give an explanation of another quantum computing model, namely Adiabatic Quantum Computing.

3.1 Quantum Circuit Model

The quantum circuit model is a model for quantum computing. In it, qubits store data, operations are performed with quantum gates, and results are obtained via measurements. All of these are ruled by the laws of quantum mechanics. We start by explaining some basic notions on qubits and quantum gates that would be useful for the next chapters. Details can be found for instance in [NC11], [CGC23], and [YM08].

3.1.1 Braket Notation

Let us begin with some notation. Let K be a field (usually $\mathbb{C}$), and let V be a K-vector space. A *ket* is an expression of the form $|v\rangle$. Mathematically, it denotes a vector. Hence, $|\cdot\rangle$ notation is used to indicate that the object is a vector and it is the standard quantum-mechanical notation for a vector in a vector space. A bra is an expression of the form $\langle .|$. Mathematically, it denotes a linear map $f : V \to K$. Letting the linear functional $\langle f|$ act on a vector $|v\rangle$ is written as $\langle f|v\rangle \in K$.

Example 3.1. In the $\mathbb{C}$-vector space $\mathbb{C}^n$, the space of all n-tuples of complex numbers, kets can be seen as column vectors and bras are the Hermitian conjugates of kets. That is, for a given ket, the corresponding bra is a row vector (the transpose of a ket), where the elements have been complex conjugated.

Example 3.2. If $\mathbb{C}^n$ has the standard inner product, then for vectors $|u\rangle$ and $|v\rangle$ in $\mathbb{C}^n$, i.e.,

$$|u\rangle = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}, |v\rangle = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix},$$

with $u_i, v_i \in \mathbb{C}$, for all $i = 1, \ldots, n$, the bra of $|u\rangle$ is the conjugate transpose of the vector $|u\rangle$, which is

$$(|u\rangle)^\dagger = \overline{\left(\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \right)^t} = \begin{bmatrix} \overline{u_1} & \overline{u_2} & \ldots & \overline{u_n} \end{bmatrix},$$

where the t superindex denotes the transpose of a matrix. Thus, the standard inner product in $\mathbb{C}^n$ can be written as

$$\langle u|v\rangle := (|u\rangle)^\dagger |v\rangle = \sum_{i=1}^{n} \overline{u_i} v_i,$$

and the norm of a vector $|v\rangle$ is defined by

$$|| \, |v\rangle ||^2 = \langle v|v\rangle = \sum_{i=1}^{n} \overline{v_i} v_i = \sum_{i=1}^{n} |v_i|^2.$$

Note that the product between $|v\rangle$ and $\langle v|$, called the outer product, yields a matrix of size $n \times n$:

$$|v\rangle \langle v| = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \begin{bmatrix} \overline{v_1} & \overline{v_2} & \ldots & \overline{v_n} \end{bmatrix} = \begin{bmatrix} v_1\overline{v_1} & v_1\overline{v_2} & \ldots & v_1\overline{v_n} \\ v_2\overline{v_1} & v_2\overline{v_2} & \ldots & v_2\overline{v_n} \\ \vdots & \vdots & \ddots & \vdots \\ v_n\overline{v_1} & v_n\overline{v_2} & \ldots & v_n\overline{v_n} \end{bmatrix}.$$

Note that in the finite dimensional complex vector spaces that come up in quantum computation and quantum information, a Hilbert space is exactly the same thing as an inner product space (since every finite dimensional normed vector space is complete).

The simplest quantum mechanical system, and the system which we will be the most concerned with, is the qubit.

3.1.2 Qubits

Consider the set $\{|0\rangle, |1\rangle\}$, where $|0\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix}^t$ and $|1\rangle = \begin{bmatrix} 0 & 1 \end{bmatrix}^t$. It is an orthonormal basis of $\mathbb{C}^2$. Indeed, $\langle 0|1\rangle = \langle 1|0\rangle = 0$ and $\langle 0|0\rangle = \langle 1|1\rangle = 1$, since

$$\langle 0|1\rangle = (|0\rangle)^\dagger |1\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0.$$

It is called the computational basis of $\mathbb{C}^2$. Now, let us talk about qubits. A quantum-bit or qubit for short, is the minimal information unit in quantum computing. In contrast to normal bits, which can be in state 0 or in state 1, it can be in state $|0\rangle$ or $|1\rangle$, or it could be in a superposition $\alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$, and $|\alpha|^2 + |\beta|^2 = 1$.

The scalars α, β are called the amplitudes of the state. The quantity $\sqrt{|a|^2 + |b|^2}$ is called the norm of the state, and, when its is 1, the state is normalised.

Example 3.3. For example,

- $|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
- $|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
- $|i+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$
- $|i-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$

are qubits states.

Definition 3.4. *Superposition* refers to the fact that any linear combination of two quantum states, once normalized, will also be a valid quantum state.

A quantum computer contains several qubits. Thus, it is necessary to know how to construct the combined state of a system of qubits given the states of the individual qubits. The joint state of a system of qubits is described using an operation known as the tensor product $\otimes$. The tensor product of two vectors $|u\rangle$ and $|v\rangle$ of $\mathbb{C}^n$, denoted as $|u\rangle \otimes |v\rangle$ is defined by

$$|u\rangle \otimes |v\rangle = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \otimes \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} u_1 v_1 \\ u_1 v_2 \\ \vdots \\ u_2 v_m \\ u_2 v_2 \\ \vdots \\ u_2 v_m \\ u_3 v_1 \\ \vdots \\ u_n v_m \end{bmatrix},$$

with $u_i, v_i \in \mathbb{C}$, for all $i = 1, \ldots, n$. Now, let us define the computational basis for $\mathbb{C}^4$. Denote $|i\rangle \otimes |j\rangle$ as $|ij\rangle$ for $i, j \in \{|0\rangle, |1\rangle\}$, so

$$\begin{aligned} |00\rangle &= |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^t \\ |01\rangle &= |0\rangle \otimes |1\rangle = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^t \\ |10\rangle &= |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^t \\ |11\rangle &= |1\rangle \otimes |1\rangle = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^t. \end{aligned}$$

The computational basis would be $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. For a more convenient notation, denote $|00\rangle = |0\rangle$, $|01\rangle = |1\rangle, |10\rangle = |2\rangle, |11\rangle = |3\rangle$, so the computational basis for $\mathbb{C}^4$ would be $\{|0\rangle, |1\rangle, |2\rangle, |3\rangle\}$. A two-qubit state is of the form

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

where $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11} \in \mathbb{C}$ and $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

Example 3.5. The vectors

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}, \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \frac{|10\rangle + |01\rangle}{\sqrt{2}}, \frac{|01\rangle - |10\rangle}{\sqrt{2}},$$

are two-qubit states. These four states are known as the Bell states, or the Bell base, or the EPR pairs.

In general, one can see that the dimension of the state space grows exponentially with the number of qubits n and the number of basis vectors is 2^n. Indeed, the computational basis for $\mathbb{C}^{2^n}$ would be

$$\begin{aligned}&\{|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle, |0\rangle \otimes |0\rangle \otimes \cdots \otimes |1\rangle, \ldots, |1\rangle \otimes |1\rangle \cdots \otimes |1\rangle\}\\ =&\{|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle\},\end{aligned}$$

so, a generic state of a multi-qubit system is $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ where $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ and $\alpha_i \in \mathbb{C}$ for $0 \leq i \leq 2^n - 1$.

Definition 3.6. Let $|\psi\rangle$ be a state. We say that $|\psi\rangle$ is a product state if there are two states $|\varphi_1\rangle$ and $|\varphi_2\rangle$ such that $|\psi\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle$, that is, $|\psi\rangle$ can be written as the tensor product between two states $|\varphi_1\rangle$ and $|\varphi_2\rangle$. If $|\psi\rangle$ is not a state product, then we say it is entangled.

Example 3.7. For example, the state $\frac{|01\rangle - |10\rangle}{\sqrt{2}}$ is an entangled state. Indeed, let us suppose not. Then, for some $\alpha_1, \beta_1, \alpha_2, \beta_2$ we would have

$$\begin{aligned}\frac{|00\rangle - |10\rangle}{\sqrt{2}} &= (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle)\\ &= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle\end{aligned}$$

So, $\alpha_1\alpha_2 = 0, \alpha_1\beta_2 = \frac{1}{\sqrt{2}}, \beta_1\alpha_2 = -\frac{1}{\sqrt{2}}, \beta_1\beta_2 = 0$. Then, either $\alpha_1 = 0$ or $\alpha_2 = 0$. If $\alpha_2 = 0$, then $\beta_1\alpha_2 = 0$, which is not valid. If $\alpha_1 = 0$, then $\alpha_1\beta_2 = 0$ which is not valid. Hence, $\frac{|01\rangle - |10\rangle}{\sqrt{2}}$ is an entangled state.

Lastly, let us define what is a quantum register,

Definition 3.8. A quantum register is a system comprising multiple qubits.

3.1.3 Measurement

An important ingredient of quantum circuits is the measurement of a qubit in the computational basis. It allows us to obtain classical values from quantum states, with two outcomes. In fact, consider the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. If we measure $|\psi\rangle$, then the probability of obtaining measurement outcome 0 is $|\alpha|^2$. And the probability of obtaining measurement outcome 1 is $|\beta|^2$. Naturally, these two probabilities must add up to one (i.e., the probability of finding the system in any of them is 1), hence the need for the normalisation condition $|\alpha|^2 + |\beta|^2 = 1$. The state after measurement in the two cases is $|0\rangle$ and $|1\rangle$, respectively.

Now, let $|\psi\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle$, with $a_i \in \mathbb{C}$ such that $\sum_{i=0}^{2^n-1} |a_i|^2 = 1$, be a generic state of a system of n qubits. If we measure all the qubits of the system $|\psi\rangle$, we obtain i with probability $|a_i|^2$, and the state would collapse to $|i\rangle$. If we measure the i-th qubit, then we will obtain 0 with probability $\sum_{j\in J} |a_j|^2$ where J is the set of numbers whose j-th bit is 0, and the state of the system after measuring 0 will be

$$\frac{\sum\limits_{j\in J} a_j |j\rangle}{\sqrt{\sum\limits_{j\in J} |a_j|^2}}.$$

The case in which we obtain 1 is analogous.

In general, if we have a state

$$\sum_{x\in\{0,1\}^n, y\in\{0,1\}^m} a_{xy} |x\rangle |y\rangle,$$

and we measure the first $n-$qubit register, we will obtain $|x_0\rangle$ with probability $\sum_{y\in\{0,1\}^m} |a_{x_0y}|^2$, which is equivalent to

$$\left|\left| \sum_{y\in\{0,1\}^m} a_{x_0y} |y\rangle \right|\right|^2,$$

and the quantum state then collapses to

$$\frac{\sum\limits_{y\in\{0,1\}^m} a_{x_0y} |x_0\rangle |y\rangle}{\sqrt{\sum\limits_{y\in\{0,1\}^m} |a_{x_0y}|^2}}.$$

The case where the second register is measured is analogous as measuring the first register. When we have more than two registers, the situation is also analogous.

Example 3.9. Suppose that we have the superposition

$$|\psi\rangle = \frac{1}{2}(|000\rangle + |100\rangle + |101\rangle - |111\rangle),$$

and we measure the third qubit. The probability that the measurement outcome is 0 is

$$\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = \frac{1}{2},$$

and in this case the resulting state is

$$\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |100\rangle).$$

The probability that the measurement outcome is 1 is also $\frac{1}{2}$, and the resulting superposition is

$$\frac{1}{\sqrt{2}}(|10\rangle - |11\rangle)|1\rangle = \frac{1}{\sqrt{2}}(|101\rangle - |111\rangle).$$

Note 1. There is a well-known principle called, **The Principle of Deferred Measurement**, which states that measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit; if the measurement results are used at any stage of the circuit then the classically controlled operations can be replaced by conditional quantum operations. [[NC11], Section 4.4]. This principle will be used in Section 6.3.2.

3.1.4 Quantum gates

In order to perform computations, we need to manipulate the state of qubits. In the quantum circuit model, operations are a special type of linear transformation applied to the vectors representing the states of the qubits. These linear transformations are unitary matrices and are called quantum gates. Recall that a matrix U is unitary if $U^\dagger U = UU^\dagger = I_n$ where I_n is the identity matrix of size $n \times n$, and $\dagger$ is the conjugate transpose (see Table 1.2). In other words, U is unitary if U is invertible and $U^{-1} = U^\dagger$. Now, let us see some general properties that the unitary matrices satisfy, and then we will give some examples of those matrices.

Proposition 3.10. Let T be a unitary matrix and λ an eigenvalue of T. Then $|\lambda| = 1$.

Proof. Suppose $|v\rangle$ is not the zero vector, and $T|v\rangle = \lambda|v\rangle$. Then

$$|\lambda|^2\langle v|v\rangle = \langle \lambda v|\lambda v\rangle = \langle Tv|Tv\rangle = (T|v\rangle)^\dagger T|v\rangle = \langle v|T^\dagger T|v\rangle = \langle v|v\rangle.$$

Since $\langle v|v\rangle = ||v||^2 \neq 0$, then $|\lambda|^2 = 1$ and hence $|\lambda| = 1$. □

Proposition 3.11. Unitary matrices preserve inner products between vectors.

Proof. Let $|v\rangle$ and $|w\rangle$ be any two vectors, and U a unitary operator. Then,

$$\langle Uv|Uw\rangle = \langle v|U^\dagger U|w\rangle = \langle v|I|w\rangle = \langle v|w\rangle.$$

□

Proposition 3.11 tell us that unitary matrices preserve the inner product, which means that no matter how you transform a vector with a sequence of unitary matrices, the normalization condition still holds. Indeed, if $\langle\psi|\psi\rangle = 1$ and U is unitary then let $U|\psi\rangle = |\psi'\rangle$. It holds that

$$\langle\psi'|\psi'\rangle = |\psi'\rangle^\dagger |\psi'\rangle = U|\psi\rangle^\dagger U|\psi\rangle = \langle\psi|U^\dagger U|\psi\rangle = \langle\psi|I|\psi\rangle = \langle\psi|\psi\rangle = 1.$$

Proposition 3.12. The tensor product between two unitary matrices is a unitary matrix.

Proof. Let U_1, U_2 be two unitary matrices of size $n_1 \times n_1$ and $n_2 \times n_2$ respectively. On one hand

$$(U_1 \otimes U_2)^\dagger (U_1 \otimes U_2) = \left(U_1^\dagger \otimes U_2^\dagger\right)(U_1 \otimes U_2) = U_1^\dagger U_1 \otimes U_2^\dagger U_2 = I_{n1} \otimes I_{n2} = I_n,$$

with $n = n_1 + n_2$, and on the other hand,

$$(U_1 \otimes U_2)(U_1 \otimes U_2)^\dagger = (U_1 \otimes U_2)\left(U_1^\dagger \otimes U_2^\dagger\right) = U_1 U_1^\dagger \otimes U_2 U_2^\dagger = I_{n_1} \otimes I_{n_2} = I_n.$$

Hence, $U_1 \otimes U_2$ is unitary. □

Now, let us see some examples of important one-qubit quantum gates:

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}, H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The X gate is the quantum version of the NOT gate and H is the Hadamard gate, one of the most widely used single-qubit gates. The reason behind this is that the Hadamard gate can be used to create superposition of states, as shown by its action on the computational basis:

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$
$$|1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Other examples of quantum gates are the parametrized gates

$$U_1(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}, U_2(\phi, \lambda) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\phi+\lambda)} \end{bmatrix}.$$

Note that $U_1(\pi) = Z, U_1\left(\frac{\pi}{2}\right) = S, U_1\left(\frac{\pi}{4}\right) = T$, and $U_2(0, \pi) = H$. The $U_1(\lambda)$ gates are also known as phase gates.

So far we have only seen one-qubit gates. Let us start with two-qubit gates. Since the tensor product between two unitary operators is unitary, two one-qubit gates acting on each qubit independently would form a two-qubit gate whose action is given by

$$U|\psi_1\rangle \otimes |\psi_2\rangle = (U_1 \otimes U_2)(|\psi_1\rangle \otimes |\psi_2\rangle) = U_1|\psi_1\rangle \otimes U_2|\psi_2\rangle,$$

where $|\psi_1\rangle, |\psi_2\rangle$ are two states of qubits and U_1 and U_2 are one-qubit gate acting respectively on states $|\psi_1\rangle$ and $|\psi_2\rangle$.

However, there are unitary matrices that cannot be written as the tensor product of unitary matrices. Indeed, an example of this is one of the most important two-qubit gate, the CNOT or controlled-NOT given by

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

Note that

$$\begin{aligned} CNOT|00\rangle &= |00\rangle \quad CNOT|10\rangle = |1\rangle NOT|0\rangle = |11\rangle \\ CNOT|01\rangle &= |01\rangle \quad CNOT|11\rangle = |1\rangle NOT|1\rangle = |10\rangle. \end{aligned}$$

In general, for any quantum single-qubit gate U, its controlled version CU is given by

$$\begin{aligned} CU|00\rangle &= |00\rangle \quad CU|10\rangle = |1\rangle U|0\rangle \\ CU|01\rangle &= |01\rangle \quad CU|11\rangle = |1\rangle U|1\rangle. \end{aligned}$$

So, if U is given by

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

then

$$CU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}.$$

It can be seen that if U is a unitary, CU is also unitary, and the application or not of U is controlled by the first qubit

$$C_1U = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U,$$

but if the first qubit is the target and the second qubit is the controlled one then

$$C_2U = I \otimes |0\rangle\langle 0| + U \otimes |1\rangle\langle 1|.$$

Now, an example of a three-qubit gate is the doubly-controlled NOT gate (CCNOT) or Toffoli gate. Its matrix is given by

$$CCNOT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

This gate behaves analogously to the CNOT gate, with the difference that both the first and second qubits must be 1 in order for the third qubit to be negated.

Now, let us see an example of a n-qubit gate. If we take the gate H and make the tensor product of itself n-times we would have an n-qubit gate $H^{\otimes^n} = H \otimes H \otimes \cdots \otimes H$.

Proposition 3.13. Let $|x\rangle = |x_1 \cdots x_n\rangle$ denote a computational basis state on n qubits. That is, $x \in \{0,1\}^n$ and each $x_i \in 0,1$ denotes a particular bit value. Then

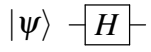
$$H^{\otimes^n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{n} (-1)^{x \cdot j} |j\rangle .$$

where $x \cdot j = x_1 j_1 \oplus x_2 j_2 \oplus \cdots \oplus x_n j_n$, and $\oplus$ stands for XOR.

Proof. See, for instance, [Section 6.1, [YM08]]. □

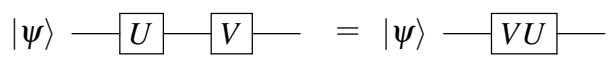
3.2 Quantum Circuits

Let us now describe *quantum circuits*. In the circuit representation, the horizontal lines are called wires and they represent the qubits that we are working with, and the circuit is read from left to right. Gates are then drawn on the qubits they act on. This is done in sequence from left to right. The initial state of the qubit is denoted at the beginning of each of the qubit lines. So for instance, the circuit



represents a Hadamard transform applied to a single qubit, with state $|\psi\rangle$. If the input is $|\psi\rangle$ the output is $H|\psi\rangle$.

Now, let us assume that we have two gates U and V, that both act on a qubit $|\psi\rangle$. When V is put after U in a circuit, then the effect of the two gates can be described as a single gate VU which is the product between the two matrices. It can be seen in the following quantum circuit



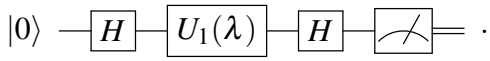
Example 3.14. It can be proved that

since $X = HZH$.

The measurement operator allows us to obtain classical values from quantum states. In quantum circuits, it is denoted by the gauge symbol shown in Figure 3.1.

Figure 3.1: Quantum circuit symbol for measurement.

Example 3.15. Consider the following quantum circuit:



Let us see what it produces.

1. First, we apply the Hadamard gate on the first qubit to obtain

$$|0\rangle \xrightarrow{H} H|0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right).$$

2. Then, we apply the phase gate $U_1(\lambda)$ to obtain

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \xrightarrow{U_1(\lambda)} U_1(\lambda)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \left(\frac{|0\rangle + e^{i\lambda}|1\rangle}{\sqrt{2}}\right).$$

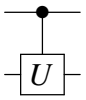
3. Then, we apply the Hadamard gate to get

$$\begin{aligned}\left(\frac{|0\rangle + e^{i\lambda}|1\rangle}{\sqrt{2}}\right) \xrightarrow{H} H\left(\frac{|0\rangle + e^{i\lambda}|1\rangle}{\sqrt{2}}\right) \\ &= \frac{1}{2}\left(|0\rangle + |1\rangle + e^{i\lambda}(|0\rangle - |1\rangle)\right) \\ &= \cos\frac{\lambda}{2}|0\rangle - i\sin\frac{\lambda}{2}|1\rangle.\end{aligned}$$

4. Finally, we measure. The probability of obtaining 0 is $\cos^2\frac{\lambda}{2}$, and the probability of obtaining 1 is $\sin^2\frac{\lambda}{2}$.

Figure 3.2: CNOT Representation.

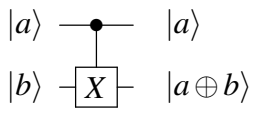
The circuit representation of a controlled one qubit-gate U is



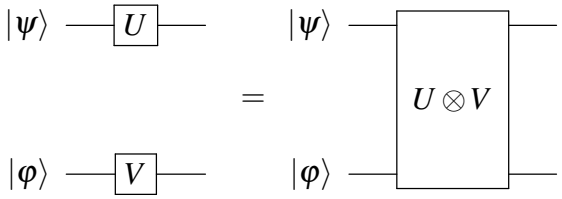
.

This leads to an important example.

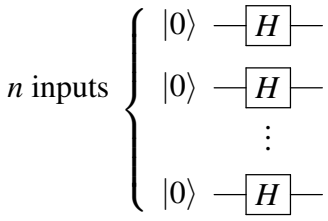
Example 3.16. The representation of the CNOT gate in a quantum circuit can be seen in Figure 3.2.
As an example, the operation $a \oplus b$, with a, b Boolean variables, can be implemented with 1 CNOT gate as



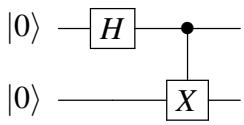
Now, if we have two quantum gates U and V in parallel, with U acting on a qubit $|\psi\rangle$ and V acting on a qubit $|\varphi\rangle$, this is equivalent to the gate $U \otimes V$ acting on $|\psi\rangle \otimes |\varphi\rangle$, as it is shown in the following circuit:



Example 3.17. In general, the gate $H^{\otimes^n}$ acting on $|0\rangle^{\otimes^n}$ in the quantum circuit representation can be seen as



Example 3.18. Consider the following quantum circuit:



Let us see what it produces.

1. First, we apply the Hadamard gate on the first qubit and, at the same time, we apply the identity on the second qubit, and as we know this is in fact the tensor product between the Hadamard gate and the identity. This give us

$$|0\rangle|0\rangle \xrightarrow{H\otimes I} (H\otimes I)|0\rangle|0\rangle = H|0\rangle \otimes I|0\rangle = \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\otimes|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle+|10\rangle).$$

2. Then, we apply the CNOT gate to get

$$CNOT\left(\frac{1}{\sqrt{2}}(|00\rangle+|10\rangle)\right) = \frac{1}{\sqrt{2}}(CNOT|00\rangle + CNOT|10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle+|11\rangle).$$

Thus, this quantum circuit prepares a Bell state.

Example 3.19 ([CEMM98] Hadamard Test)**.** The Hadamard Test is a routine to estimate the real part of the value $\langle\psi|U|\psi\rangle$ where $|\psi\rangle$ is a quantum state and U is a unitary gate acting on the space of $|\psi\rangle$. Its quantum circuit is given in Figure 3.3.

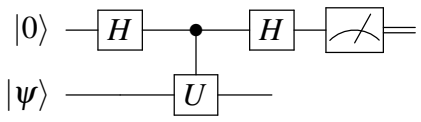


Figure 3.3: Circuit of the Hadamard Test.

The action of this circuit can be described with the following steps:

1. First, apply $H\otimes I$ to $|0\rangle\otimes|\psi\rangle$ to get

$$\begin{aligned}|0\rangle\otimes|\psi\rangle &\xrightarrow{H\otimes I} \frac{(|0\rangle+|1\rangle)}{\sqrt{2}}\otimes|\psi\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle\otimes|\psi\rangle+|1\rangle\otimes|\psi\rangle).\end{aligned}$$

2. Then, we apply CU to obtain

$$\frac{1}{\sqrt{2}}(|0\rangle\otimes|\psi\rangle+|1\rangle\otimes|\psi\rangle) \xrightarrow{CU} \frac{1}{\sqrt{2}}(|0\rangle\otimes I|\psi\rangle+|1\rangle\otimes U|\psi\rangle).$$

3. Then, we apply $H\otimes I$ to obtain

$$\begin{aligned}\frac{1}{\sqrt{2}}(|0\rangle\otimes I|\psi\rangle+|1\rangle\otimes U|\psi\rangle) &\xrightarrow{H\otimes I} \frac{1}{\sqrt{2}}(H|0\rangle\otimes|\psi\rangle+H|1\rangle\otimes U|\psi\rangle) \\ &= \frac{1}{\sqrt{2}}\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\otimes|\psi\rangle+\frac{|0\rangle-|1\rangle}{\sqrt{2}}\otimes U|\psi\rangle\right) \\ &= \frac{1}{2}(|0\rangle\otimes(I+U))|\psi\rangle+\frac{1}{2}(|1\rangle\otimes(I-U)|\psi\rangle).\end{aligned}$$

Let us write $|\psi\rangle$ as $\alpha|0\rangle+\beta|1\rangle$ with $\alpha,\beta\in\mathbb{C}$ and $|\alpha|^2+|\beta|^2=1$. There exist $\delta,\varepsilon\in\mathbb{C}$, such that $U|\psi\rangle=\delta|0\rangle+\varepsilon|1\rangle$. So,

$$\begin{aligned}&\frac{1}{2}(|0\rangle\otimes(I+U))|\psi\rangle+\frac{1}{2}(|1\rangle\otimes(I-U)|\psi\rangle)\\ =&\frac{1}{2}(|0\rangle\otimes((\alpha+\delta)|0\rangle+(\beta+\varepsilon)|1\rangle))+\frac{1}{2}(|1\rangle\otimes((\alpha-\delta)|0\rangle+(\beta-\varepsilon)|1\rangle)).\end{aligned}$$

4. Finally we measure the first qubit. The result is 0 with probability

$$\frac{1}{4}\left(|\alpha+\delta|^2+|\beta+\varepsilon|^2\right).$$

And, the result would be 1 with probability

$$\frac{1}{4}\left(|\alpha-\delta|^2+|\beta-\varepsilon|^2\right).$$

Now, we can obtain the desired result with a simple computation:

$$\begin{aligned}&\frac{1}{4}\left(|\alpha+\delta|^2+|\beta+\varepsilon|^2\right)(1)+\frac{1}{4}\left(|\alpha-\delta|^2+|\beta-\varepsilon|^2\right)(-1)\\ =&\frac{1}{4}\left(|\alpha+\delta|^2-|\alpha-\delta|^2+|\beta+\varepsilon|^2-|\beta-\varepsilon|^2\right)\\ =&\frac{1}{4}((4\,\mathrm{Re}(\alpha)\,4\,\mathrm{Re}(\delta)+4\,\mathrm{Im}(\alpha)\,4\,\mathrm{Im}(\delta))+(4\,\mathrm{Re}(\beta)\,4\,\mathrm{Re}(\varepsilon)+4\,\mathrm{Im}(\beta)\,4\,\mathrm{Im}(\varepsilon)))\\ =&\,\mathrm{Re}(\overline{\alpha})\,\mathrm{Re}(\delta)-\mathrm{Im}(\overline{\alpha})\,\mathrm{Im}(\delta)+\mathrm{Re}\left(\overline{\beta}\right)\mathrm{Re}(\varepsilon)-\mathrm{Im}\left(\overline{\beta}\right)\mathrm{Im}(\varepsilon)\\ =&\,\mathrm{Re}\left(\overline{\alpha}\delta+\overline{\beta}\varepsilon\right)\\ =&\,\mathrm{Re}\,\langle\psi|U|\psi\rangle.\end{aligned}$$

Now, let us consider a particular case where $|\psi\rangle$ is an eigenstate of the unitary operator U, so $U|\psi\rangle=e^{i\varphi}|\psi\rangle$, for $\varphi\in[0,2\pi]$. When we apply CU we obtain

$$\begin{aligned}\frac{1}{\sqrt{2}}(|0\rangle\otimes|\psi\rangle+|1\rangle\otimes|\psi\rangle)\xrightarrow{CU}&\frac{1}{\sqrt{2}}(|0\rangle\otimes I|\psi\rangle+|1\rangle\otimes U|\psi\rangle)\\ =&\frac{1}{\sqrt{2}}\left(|0\rangle\otimes|\psi\rangle+|1\rangle\otimes\left(e^{i\varphi}|\psi\rangle\right)\right)\\ =&\frac{1}{\sqrt{2}}\left(|0\rangle\otimes|\psi\rangle+e^{i\varphi}|1\rangle\otimes|\psi\rangle\right)\\ =&\frac{1}{\sqrt{2}}\left(\left(|0\rangle+e^{i\varphi}|1\rangle\right)\otimes|\psi\rangle\right).\end{aligned}$$

Then, applying $H\otimes I$ we get

$$\frac{1}{\sqrt{2}}\left(\left(|0\rangle+e^{i\varphi}|1\rangle\right)\otimes|\psi\rangle\right)\xrightarrow{H\otimes I}\frac{1}{2}\left(\left(1+e^{i\varphi}\right)|0\rangle+\left(1-e^{i\varphi}\right)|1\rangle\otimes|\psi\rangle\right).$$

Now, if we measure the first qubit, we get 0 with probability

$$\begin{aligned}\frac{1}{4}\left|\left(1+e^{i\varphi}\right)\right|^2 &= \frac{1}{4}\left((1+\cos\varphi)^2+(\sin\varphi)^2\right)\\ &= \frac{1}{4}\left(2+2\cos\varphi\right)\\ &= \frac{1}{4}\left(2\left(2\cos^2\left(\frac{\varphi}{2}\right)\right)\right)\\ &= \cos^2\left(\frac{\varphi}{2}\right).\end{aligned}$$

And we get 1 with probability,

$$\begin{aligned}\frac{1}{4}\left|\left(1-e^{i\varphi}\right)\right|^2 &= \frac{1}{4}\left((1-\cos\varphi)^2+(-\sin\varphi)^2\right)\\ &= \frac{1}{4}\left(2-2\cos\varphi\right)\\ &= \frac{1}{4}\left(4\sin^2\left(\frac{\varphi}{2}\right)\right).\\ &= \sin^2\left(\frac{\varphi}{2}\right).\end{aligned}$$

As a last example, let us give the quantum circuit representation of the Toffoli gate. It is known that the Toffoli gate is a universal reversible logic gate. In fact, it can be used to implement the AND gate. For instance, $a \wedge b$ (where $\wedge$ is AND, a, b are Boolean variables) can be constructed as in Figure 3.4.

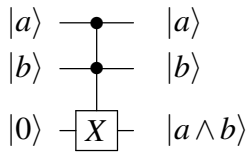


Figure 3.4: Circuit for $a \wedge b$ constructed with a Toffoli gate.

It is worth mentioning that the CNOT, Hadamard and T half-phase gates are a universal set of gates [NC11] [Section 4.5], which means that we can approximate to any desired accuracy any quantum transformation on an arbitrary number of qubits using only these gates. For example, the circuit in Figure 3.5, shows a possible decomposition (or transpilation, which is the process of rewriting a given input circuit to map the topology of a specific quantum device) of a Toffoli gate, in the base $[H, T, T^\dagger, CNOT]$ [[NC11] Section 4.2].

Lastly, let us define what is the size of a quantum circuit, and what we mean by *efficient* quantum circuits, since most of this thesis is concerned with those concepts.

Definition 3.20. The size of a quantum circuit in a fixed basis of universal gates is the number of gates it contains.

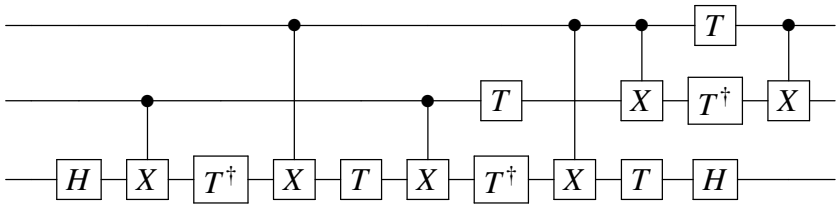


Figure 3.5: Descomposition of a Toffoli gate.

By efficient, we mean that as the size of the input grows, the size of the necessary quantum gates such as NOT, Hadamard, T gates and CNOT should grow at most polynomially with the size of the input.

3.2.1 Oracles

Many quantum algorithms are based on the analysis of some function f that is given as an oracle. An oracle (also called black box) is a special kind of unitary transformation that is defined by its action on the computational basis, and that is given as a gate that can be used in a quantum circuit but whose inner workings cannot be inspected.

One of the main forms that oracles take is that of Boolean oracles. For a given Boolean function $f : \{0,1\}^n \to \{0,1\}^m$, such a unitary operator, denoted U_f, acts on quantum states in $(\mathbb{C}^2)^{\otimes(n+m)}$, where $a \in \{0,1\}^n$ and $h \in \{0,1\}^m$, and is defined by

$$U_f |a\rangle |h\rangle = |a\rangle |h \oplus f(a)\rangle .$$

Observe that, when the second $m-$qubit register is $|0\rangle$, application of U_f yields an evaluation of the content of the first $n-$qubit register.

We can represent such an oracle with the circuit shown in Figure 3.6.

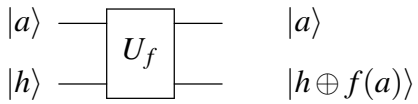


Figure 3.6: Representation of the oracle U_f.

Example 3.21. Let $f : \{0,1\} \to \{0,1\}$ defined by $f(0) = 1$ and $f(1) = 0$. A unitary operator U_f that implements $U_f |a\rangle |h\rangle = |a\rangle |h \oplus f(a)\rangle$ is

Another form of oracle is the phase oracle, defined as $P_f |x\rangle = (-1)^{f(x)} |x\rangle$, where $f(x) \in \{0,1\}$. Note that, in fact, it can be obtained from a Boolean oracle. Indeed, consider the Boolean

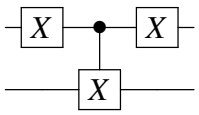


Figure 3.7: Representation of the oracle U_f of example 3.21.

oracle U_f that correspond to the same function. Then

$$\begin{aligned} U_f(|x\rangle \otimes |-\rangle) &= |x\rangle \otimes \left(\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle)\right) \\ &= \frac{1}{\sqrt{2}}(-1)^{f(x)}|x\rangle \otimes (|0\rangle - |1\rangle) \\ &= (-1)^{f(x)}|x\rangle \otimes |-\rangle \\ &= P_f|x\rangle \otimes |-\rangle \end{aligned}$$

Here, we have used the fact that $|0 \oplus a\rangle - |1 \oplus a\rangle = (-1)^a(|0\rangle - |1\rangle)$ for $a \in \{0,1\}$. Also note that $|-\rangle = H|1\rangle$. Therefore, we can get a phase oracle from a Boolean oracle.

These oracles typically requires a number of ancillary qubits that are initially in a well-defined state, usually $|0\rangle$, to store partial results, and must be returned to the same state at the end of the computation. This technique is used so that the ancillary qubits can be re-used. In many cases, uncomputing a quantum register can be achieved by the inverse of the steps required for the computation [NC11][Chapter 4].

Example 3.22. Using CNOT gates and Toffoli gates, we can represent the oracle for the Boolean function f given by

$$\begin{aligned} f : \{0,1\}^4 &\longrightarrow \{0,1\} \\ (a,b,c,d) &\longmapsto (a \wedge b) \oplus (c \wedge d), \end{aligned}$$

with a,b,c,d Boolean variables. Indeed, we can achieve it with 4 Toffoli gates and 1 CNOT gate, as we can see in the quantum circuit shown in Figure 3.8.

Example 3.23. Consider the Boolean function:

$$\begin{aligned} f : \{0,1\}^3 &\longrightarrow \{0,1\} \\ (a_1,a_2,a_3) &\longmapsto (a1 \oplus a2) \wedge (\sim a2 \oplus a3) \wedge (a1 \oplus a3). \end{aligned}$$

Here, $\sim$ stands for NOT. Now, let us build its corresponding oracle (see Figure 3.9).
As it can be seen, it has a quantum register of 4 qubits, and a ancillary register of 5 qubits, 13 CNOT, 4 *NOT* and 4 Toffoli gates.

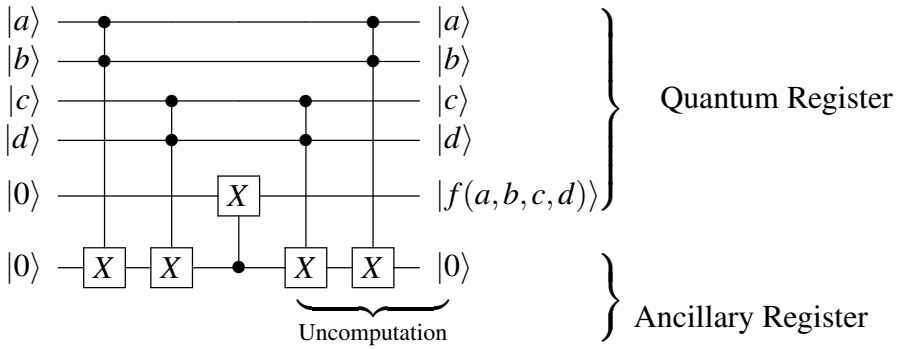


Figure 3.8: Representation of the oracle U_f of example 3.22

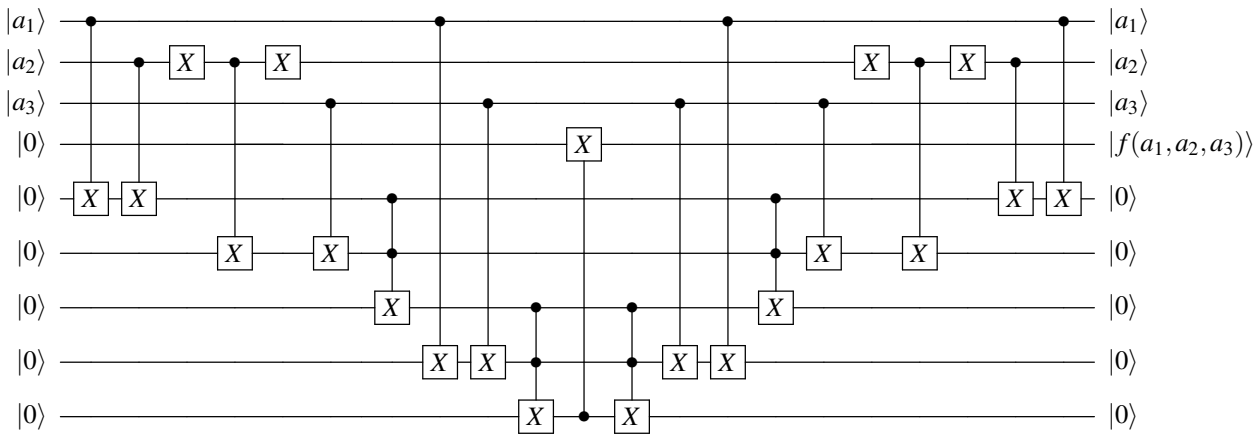


Figure 3.9: Representation of the oracle U_f of example 3.23.

3.3 Quantum Fourier Transform over Abelian Groups

Now, let us talk about one of the most important unitary transformations in quantum computing, which is the **Quantum Fourier Transform** (QFT) . The Quantum Fourier Transform is the quantum implementation of the discrete Fourier transform over the amplitudes of a wave function. And as a matter of fact, the QFT lies at the heart of quantum algorithms such as Shor's factoring algorithm, the quantum phase estimation algorithm and the algorithm used for the Abelian HSP.

So, in this section we define the Quantum Fourier Transform over finite Abelian groups, and for that, we require the basic elements of representation theory, given in Section 2.5. Then we will see some particular examples we are interested in, together with their implementation.

Definition 3.24. Let G be a finite group. We can form a vector space over $\mathbb{C}$ which has the group elements of G as a basis, denoted by $\{|g\rangle \ : \ g \in G\}$, such that the addition is given by

$$\sum_{g\in G} \alpha_g |g\rangle + \sum_{g\in G} \beta_g |g\rangle := \sum_{g\in G} (\alpha_g + \beta_g) |g\rangle$$

where $\alpha_g, \beta_g \in \mathbb{C}$ for all $g \in G$. Scalar multiplication is given by

$$\lambda \sum_{g\in G} \alpha_g |g\rangle = \sum_{g\in G} \lambda \alpha_g |g\rangle .$$

Definition 3.25. Let $G = \mathbb{Z}/N\mathbb{Z}$. Consider $\mathbb{C}^{n'}$ such that $2^{n'} \geq N = |G|$, so that to each group element g we can associate a basis element $|g\rangle$. Then, the QFT over G is the transformation with the following action on the basis states:

$$F_G |g\rangle = \frac{1}{\sqrt{N}} \sum_{h\in G} \chi_g(h) |h\rangle ,$$

where $\chi_g(h) = e^{2\pi i g h/N}$ for $g, h \in G$. For basis elements that do not represent group elements F_G is the identity. For this reason, we usually omit the part of the QFT that acts as the identity.

Note 2. Equivalently, the Quantum Fourier Transform can be viewed as a unitary matrix

$$F_G = \frac{1}{\sqrt{N}} \sum_{g,h\in G} \chi_g(h) |h\rangle \langle g| .$$

In fact, the equality holds, because, for a basis state $|j\rangle$, we have that

$$F_G |j\rangle = \left(\frac{1}{\sqrt{N}} \sum_{g,h\in G} \chi_g(h) |h\rangle \langle g| \right) |j\rangle = \frac{1}{\sqrt{N}} \sum_{g,h\in G} \chi_g(h) |h\rangle \langle g|j\rangle = \frac{1}{\sqrt{N}} \sum_{h\in G} \chi_j(h) |h\rangle .$$

Now, let us see that it is unitary, i.e., $F_G F_G^\dagger = F_G^\dagger F_G = I_N$. It holds that

$$\begin{aligned}
F_G F_G^\dagger &= \left(\frac{1}{\sqrt{N}} \sum_{g,h \in G} \chi_g(h) |h\rangle \langle g|\right) \left(\frac{1}{\sqrt{N}} \sum_{g,h \in G} \chi_g(h) |h\rangle \langle g|\right)^\dagger \\
&= \frac{1}{\sqrt{N}} \frac{1}{\sqrt{N}} \sum_{g,h \in G} \chi_g(h) \overline{\chi_g(h)} |h\rangle \langle g| (|h\rangle \langle g|)^\dagger \\
&= \frac{1}{N} \sum_{g,h \in G} \chi_g(h) \overline{\chi_g(h)} |h\rangle \langle g|g\rangle \langle h| \\
&= \frac{1}{N} \sum_{g,h \in G} |\chi_g(h)|^2 |h\rangle \langle h| \\
&= \frac{1}{N} \sum_{g,h \in G} 1 |h\rangle \langle h| \\
&= \frac{1}{N} \sum_{h \in G} I_N \\
&= \frac{1}{N} N I_N \\
&= I_N
\end{aligned}$$

The same holds for $F_G^\dagger F_G$. Furthermore, from the unitary property it follows that the inverse of the Quantum Fourier Transform is $F_G^\dagger$.

Now, let us see how we can implement the QFT. For that, let us first consider the case when $G = \mathbb{Z}/N\mathbb{Z}$, with $N = 2^n$. So far, we know that F_G maps a basis element $|x\rangle$ to $\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N} |y\rangle$.

However, if we use the fractional binary notation on the integer y, $y/N = \sum_{k=1}^{n} x_k 2^{-k}$ or as $0.y_1 y_2 \ldots y_n$, the action of the quantum Fourier transform can be expressed as

$$\begin{aligned}
F_G|x\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i \left(\sum_{k=1}^{n} y_k 2^{-k}\right) x} |y_1 \ldots y_n\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=0}^{n} e^{2\pi i x y_k 2^{-k}} |y_1 \ldots y_n\rangle \\
&= \frac{1}{\sqrt{N}} \bigotimes_{k=0}^{n} \left(|0\rangle + e^{2\pi i x 2^{-k}} |1\rangle\right) \\
&= \frac{1}{\sqrt{N}} \left(|0\rangle + e^{2\pi i [0.x_n]} |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i [0.x_1 . x_2 \ldots x_{n-1} . x_n]} |1\rangle\right).
\end{aligned}$$

Then, the representation as a quantum circuit can be seen in Figure 3.10. The quantum gates

used in the circuit of n qubits are the Hadamard gate H and the phase gate

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.$$

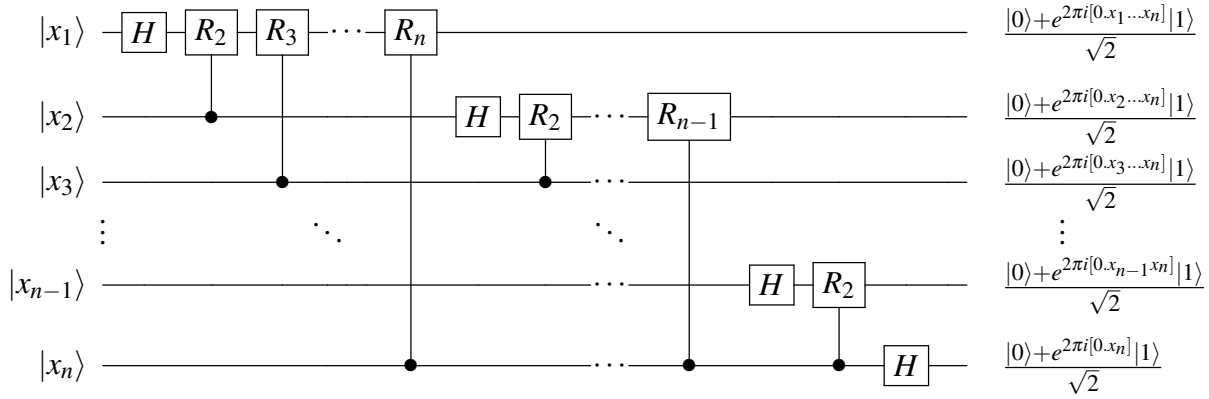


Figure 3.10: Circuit of the Quantum Fourier Transform Case $N = 2^n$.

The operations on each individual qubit can be implemented efficiently using one Hadamard gate and a linear number of controlled phase gates. The first term requires one Hadamard gate and $(n-1)$ controlled phase gates, the next term requires one Hadamard gate an $(n-2)$ controlled phase gates, and the last step requires only one Hadamard gate. Hence, $n+(n+1)+\ldots+1 = n(n+1)/2 = O(n^2)$ quantum gates (which is quadratic in the number of qubits) are needed. This means that it can be efficiently implemented.

One of the main reasons why we introduce the QFT, is that it lies at the heart of many applications of quantum computing and simulation that demonstrate exponential speed-ups compared to the best-known classical counterparts. In particular, as part of the quantum phase estimation sub-routine. In this procedure it is used the inverse of the Quantum Fourier Transform, so let us see its quantum circuit.

To invert the QFT, we must run the circuit shown in Figure 3.10 in reverse, with the inverse of each gate in place to achieve the transform

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i jk/N} \to |j\rangle.$$

Since the Hadamard gate is self-inverse, and the inverse of the rotations gate R_k is given by:

$$R_k^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & e^{-2\pi i/2^k} \end{pmatrix},$$

the inverse QFT circuit is:

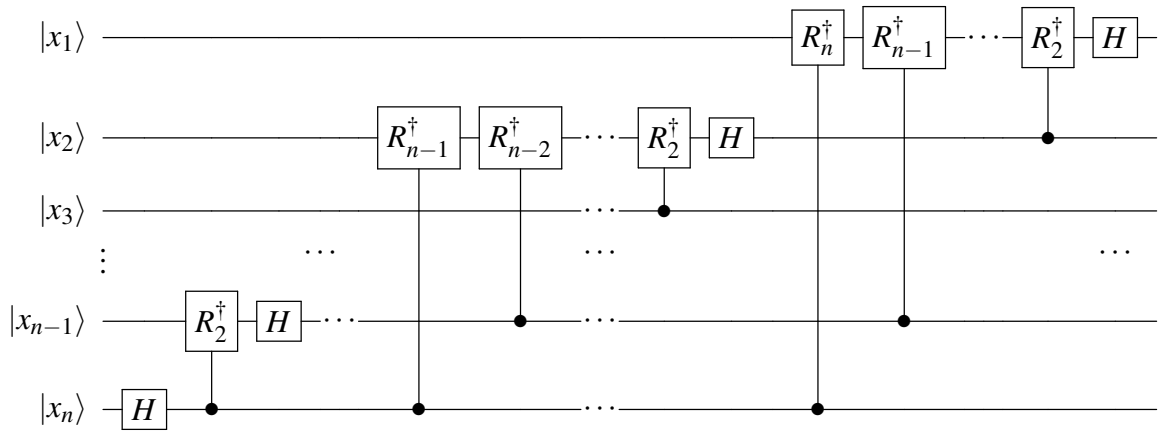


Figure 3.11: Circuit of the Inverse Quantum Fourier Transform Case $N = 2^n$.

Note that the quantum circuits shown in Figures 3.10 and 3.11, are only a valid implementation of the QFT and the inverse of the QFT, when N is a power of 2. However, efficient quantum circuits for the Fourier transform over $\mathbb{Z}/N\mathbb{Z}$ with N odd are well studied. For instance, see [Lom04] [Algorithm A.2.3] which use an implementation based on the Quantum Fourier Transforms of powers of 2 to get an approximation of the odd one, within a desired error measured by the total variation distance defined as follows:

Definition 3.26. Given two probability distributions D and D' over $\{0,1,\ldots,M-1\}$, let $|D - D'|_{TV} = \sum_{k=0}^{M-1} |D(k) - D(k')|$.

Theorem 3.27. *[Lom04] Given an odd integer $N \geq 13$, and any $\sqrt{2} \geq \varepsilon > 0$. Then for F_G with $G = \mathbb{Z}/N\mathbb{Z}$,*

1. *There is a unit vector $|\psi\rangle$ such that the output $|v\rangle$ of the algorithm A.2.3 in [Lom04] satisfies $\| |v\rangle - F_G |u\rangle |\psi\rangle \| \leq \varepsilon$, with $|u\rangle$ an element of $G = \{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$.*

2. *F_G can be computed using at most $\lceil 12.53 + 3\log_2 \frac{\sqrt{N}}{\varepsilon} \rceil$ qubits, and the algorithm has operation complexity*

$$O\left(\log_2 \frac{\sqrt{N}}{\varepsilon}\left(\log_2 \log_2 \frac{\sqrt{N}}{\varepsilon} + \log_2 \frac{1}{\varepsilon}\right)\right).$$

3. *The induced probability distributions D_v from the output of the algorithm, and D from $F_G |u\rangle \otimes |\psi\rangle$ satisfy that $|D_v - D|_{TV} \leq 2\varepsilon + \varepsilon^2$.*

Proof. The proof is fundamentally technical and it can be seen in [[Lom04] Appendix A]. □

3.3.1 QFT over $(\mathbb{Z}/p\mathbb{Z})^n$.

Now we know how to implement the QFT for $\mathbb{Z}/N\mathbb{Z}$ when $N = 2^n$ and when N is odd. Consider the Abelian group $G = (\mathbb{Z}/p\mathbb{Z})^n$ with p a prime number. The vector space over $\mathbb{C}$ which has the

group elements of G as a basis is isomorphic to the tensor product of each of the vector spaces over $\mathbb{C}$ of $\mathbb{Z}/p\mathbb{Z}$, because we can map one basis to another by

$$|(x_1, x_2, \ldots, x_n)\rangle \mapsto |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$$

for $x_i \in \mathbb{Z}/p\mathbb{Z}$. The QFT over a direct product of the cyclic groups $\mathbb{Z}/p\mathbb{Z}$ is the tensor product of the QFT over each group, i.e.,

$$F_G = \bigotimes_{i=1}^{n} F_{\mathbb{Z}/p\mathbb{Z}},$$

which acts on the entire $\mathbb{C}$-vector space of G. Indeed, note that it is unitary, since the tensor product of unitary operators is unitary. Furthermore, since for any $g, h \in (\mathbb{Z}/p\mathbb{Z})^n$ there exist $g_i, h_i \in \mathbb{Z}/p\mathbb{Z}$ for $i = 1, \ldots n$, such that $g = (g_1, \ldots, g_n)$ and $h = (h_1, \ldots, h_n)$, it holds that

$$\begin{aligned}
&\bigotimes_{i=1}^{n} F_{\mathbb{Z}/p\mathbb{Z}} \\
=&\frac{1}{\sqrt{p}} \left(\sum_{g_1, h_1 \in \mathbb{Z}/p\mathbb{Z}} \chi_{g_1}(h_1) \, |h_1\rangle \langle g_1| \right) \otimes \frac{1}{\sqrt{p}} \left(\sum_{g_2, h_2 \in \mathbb{Z}/p\mathbb{Z}} \chi_{g_2}(h_2) \, |h_2\rangle \langle g_2| \right) \otimes \cdots \otimes \\
&\frac{1}{\sqrt{p}} \left(\sum_{g_n, h_n \in \mathbb{Z}/p\mathbb{Z}} \chi_{g_n}(h_n) \, |h_n\rangle \langle g_n| \right) \\
=&\frac{1}{\sqrt{p^n}} \sum_{g_1, h_1 \in \mathbb{Z}/p\mathbb{Z}} \cdots \sum_{g_n, h_n \in \mathbb{Z}/p\mathbb{Z}} \chi_{g_1}(h_1) \cdots \chi_{g_n}(h_n) \, |h_1\rangle \langle g_1| \otimes \cdots \otimes |h_n\rangle \langle g_n| \\
=&\frac{1}{\sqrt{p^n}} \sum_{g_1, h_1 \in \mathbb{Z}/p\mathbb{Z}} \cdots \sum_{g_n, h_n \in \mathbb{Z}/p\mathbb{Z}} e^{\frac{2\pi i g_1 h_1}{p} + \cdots + \frac{2\pi i g_n h_n}{p}} \, |h_1\rangle \langle g_1| \otimes \cdots \otimes |h_n\rangle \langle g_n| \\
=&\frac{1}{\sqrt{p^n}} \sum_{g, h \in (\mathbb{Z}/p\mathbb{Z})^n} \chi_g(h) \, |h\rangle \langle g| \, . \\
=&F_G. \qquad (3.1)
\end{aligned}$$

Then, we can implement the corresponding $QFTs$ over $\mathbb{Z}/p\mathbb{Z}$. Now, each quantum circuit is independent from each other and by Theorem 3.27, we have that for any $\sqrt{2} \geq \varepsilon' > 0$, the $F_{\mathbb{Z}/p\mathbb{Z}}$ can be computed using at most $\lceil 12.53 + 3\log_2 \frac{\sqrt{p}}{\varepsilon'} \rceil$ qubits and with complexity of order,

$$O\left(\log_2 \frac{\sqrt{p}}{\varepsilon'} \left(\log_2 \log_2 \frac{\sqrt{p}}{\varepsilon'} + \log_2 \frac{1}{\varepsilon'} \right) \right). \qquad (3.2)$$

Now, let $D_{v_1}, \ldots, D_{v_n}$ be the induced probability distributions from the output, and $D_1, \ldots, D_n$ from each $F_{\mathbb{Z}/p\mathbb{Z}} |u\rangle \otimes |0\rangle$. Define D_v and D as $\prod_{i=1}^{n} D_{v_i}$ and $D = \prod_{i=1}^{n} D_i$, respectively. Then by [LP17] [Section 4.7], the total variation distance between D and D' is,

$$|D_v - D|_{TV} \leq \sum_{i=1}^{n} |D_{v_i} - D_i|_{TV}. \qquad (3.3)$$

So, for $\sqrt{2} \geq \varepsilon' = \frac{\varepsilon}{n} > 0$, each $|D_{v_i} - D_i|_{TV} \leq 2\frac{\varepsilon'}{n} + \left(\frac{\varepsilon'}{n}\right)^2$. Hence,

$$|D_v - D|_{TV} \leq \sum_{i=1}^{n} |D_{v_i} - D_i|_{TV} \leq n\left(2\frac{\varepsilon}{n} + \left(\frac{\varepsilon}{n}\right)^2\right) = 2\varepsilon + \frac{(\varepsilon)^2}{n}. \tag{3.4}$$

Thus, $F_{(\mathbb{Z}/p\mathbb{Z})^n}$ can be computed using at most $n\lceil 12.53 + 3\log_2 n\frac{\sqrt{p}}{\varepsilon}\rceil$, qubits, and the complexity would be of,

$$O\left(\log_2 n\frac{\sqrt{p}}{\varepsilon}\left(\log_2\log_2 n\frac{\sqrt{p}}{\varepsilon} + \log_2\frac{n}{\varepsilon}\right)\right). \tag{3.5}$$

Example 3.28. Let $G = (\mathbb{Z}/2\mathbb{Z})^n$. Then,

$$F_G|g\rangle = \frac{1}{\sqrt{N}}\sum_{h\in G}\left(\prod_{i=1}^{n}(-1)^{g_i h_i}\right)|h\rangle = \frac{1}{\sqrt{N}}\sum_{h\in G}(-1)^{g\cdot h}|h\rangle$$

Thus, the **Quantum Fourier Transform** (QFT) over $(\mathbb{Z}/2\mathbb{Z})^n$ is equal to $H^{\otimes^n}$.

3.4 Quantum Optimization

In Chapter 7, we will be interested on finding the multiplication table of a non primitive binary semifield of order 32 using quantum techniques. For that, first we will find a Boolean formula, and then, in order to determine whether there are assignments that makes the formula true, we will rephrase it as a Higher Order Binary Optimization problems, HOBO for short, so that we can apply for instance, Quantum Approximate Optimization Algorithm, QAOA for short, or Quantum Adabiatic Computing, for finding those assignments. Thus, in this section we will give a brief explanation of Quadratic Unconstrained Binary Optimization problems, QUBO for short, and HOBO. We follow [CGC23][Chapter 4-5].

Let us begin by defining what are QUBO problems. They are minimization problems in which the cost function is a quadratic polynomial over binary variables with no restrictions. More explicitly, they are problems of the form

$$\begin{aligned} &\text{Minimize} \quad q(x_0,\ldots,x_m) \\ &\text{subject to} \quad x_j \in \{0,1\}, \quad j = 0,\ldots,m, \end{aligned}$$

where $q(x_0,\ldots,x_m)$ is a quadratic polynomial on the x_j variables. Now, optimization problems in which we are asked to minimize a binary polynomial of any degree, with no additional restrictions, are called Higher Order Binary Optimization problems, HOBO for short, or Polynomial Unconstrained Binary Optimization problems, PUBO for short.

Example 3.29. As an example, consider the following Boolean formula on binary variables, given in conjuctive normal form (CNF),

$$(x_0 \vee \sim x_1 \vee \sim x_2) \wedge (\sim x_0 \vee x_1 \vee \sim x_2) \wedge (x_0 \vee x_1 \vee x_2).$$

We want to determine assignments of values that make the formula true, if they exist. For that, let us rewrite the problem as an instance of a HOBO problem by mapping the operations $x \vee y, x \wedge y$ and x to $xy, x+y$ and $1-x$, respectively, so it can be represented as the polynomial,

$$\begin{aligned} q(x_0,x_1,x_2) &= (1-x_0)\,x_1\,(1-x_2) + x_0\,(1-x_1)\,x_2 + (1-x_0)\,(1-x_1)\,(1-x_2) \\ &= -x_0x_1x_2 + 2x_0x_2 - x_0 - x_2 + 1. \end{aligned}$$

Hence our problem is,

$$\begin{aligned} &\text{Minimize} && q\,(x_0,x_1,x_1) \\ &\text{subject to} && x_0,x_1,x_2 \in \{0,1\}. \end{aligned}$$

Thus, if the minimum of the polynomial is 0, then the formula will be satisfiable. Otherwise, the formula will be unsatisfiable.

Now, one way of solving this HOBO problem is by transforming it into a QUBO problem by introducing auxilary variables, so that the objective function is a binary quadratic polynomial. For more detailes see, for instance, [CGC23][Section 5.1.5]. Then we can apply Quantum Annealing to find those assignments.

In fact, Quantum Annealing is a form of computation that efficiently samples the low-energy configurations of a quantum system. However, it is not a universal quantum computing model, but it is closely related to Adiabatic Quantum Computing, which is indeed universal. It was introduced by Farhi et al. [FGGS00]. In contrast with quantum circuits, that rely in the application of discrete operations, adiabatic quantum computing which is polynomially equivalent to the quantum circuit model [AvDK+04], relies on the use of continuous transformations, namely a time-dependent Hamiltonian $H(t)$ of the form

$$H(t) = A(t)H_0 + B(t)H_1,$$

with H_0 an initial Hamiltonian and H_1 a final Hamiltonian whose ground state encodes the solution to the problem of interest, and $H(t)$ gradually changes the acting Hamiltonian from the initial to the final one by using real-valued functions A, B that accept inputs over the interval $[0,T]$ for some time T and such that $A(0) = B(T) = 1$ and $A(T) = B(0) = 0$.

Now, in order to apply Quantum Annealing, the final Hamiltonian H_1 has to be selected from a certain, restricted class, for instance, an Ising Hamiltonian, which is of the form

$$-\sum_{j,k} J_{jk} Z_j Z_k - \sum_j h_j Z_j,$$

where each Z_i is a Z gate acting on qubit i and the coefficients J_{jk}, h_j are real numbers. In our case, after transforming from HOBO to QUBO, the QUBO can be seen as Ising Hamiltonian by a suitable sustitution, $x_j = (1-Z)/2$, which will be the Hamiltonian that encodes the assignemts that satisfies the Boolean formula.

Another alternative for solving HOBO problems is using QAOA. This is a hybrid method in which both a classical and a quantum computer are used. It was initially proposed by [FGG14] as a discretization of Adiabatic Quantum Computing, so that it can approximate optimal solutions to combinatorial optimization problems. It is a gate-based algorithm that can be understood to be the counterpart to quantum annealing in the quantum circuit model. In contrast with quantum annealing, the binary polynomial can be of any degree, and we can transform it directly into a Hamiltonian with the substitution $x_i = \frac{1-Z_i}{2}$. It can be seen that then we will have a Hamiltonian that is a sum of tensor products of the matrix Z [CGC23][Section 5.1.5]. Thus, in Chapter 7 we will study ways of formulating the problem of finding satisfying assignments of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ as QUBO and HOBO problems that can be solved with QAOA and Quantum Annealing.

Chapter 4

Some Quantum Algorithms

We have now introduced all the concepts that we need to study quantum algorithms. A quantum algorithm consists in an initial state which is transformed by a series of quantum gates and eventually measured. In this section, we explicitly give some highly influential examples of quantum algorithms, such as Grover's algorithm, Simon's algorithm and the Quantum Phase Estimation algorithm (which is a quantum algorithm that estimates the phase corresponding to an eigenvalue of a given unitary operator), that we will use in the next chapters.

4.1 Simon's Algorithm

In 1994, at the IEEE Symposium on the Foundation of Computer Science [Sim94], Daniel R. Simon, introduced a problem that a quantum algorithm can solve exponentially faster than any classical algorithm. In fact, it was one of the first quantum algorithms to show an exponential speed-up versus the classical algorithm for solving a specific problem, since any classical deterministic algorithm that solves Simon's problem requires $\Omega(\sqrt{2^n})$ queries [CQ18]. The definition of Simon's problem is as follows:

Given: A function $f : \{0,1\}^n \to \{0,1\}^n$.

Promise: There exists a secret string $s \in \{0,1\}^n$ with $s \neq 0$, such that for all $x_1, x_2 \in \{0,1\}^n$, $f(x_1) = f(x_2) \Leftrightarrow x_1 = x_2$ or $x_2 = x_1 \oplus s$.

Problem: Find s, the secret string.

The algorithm that gives solution to this problem uses a quantum subroutine and a classical post-processing procedure. We first show the quantum part, which is the following:

Algorithm 4.1 (Simon's Algorithm-Quantum Procedure).
Input: An oracle that performs the operation $U_f |a\rangle |0\rangle = |a\rangle |f(a)\rangle$ for $a \in \{0,1\}^n$.

Quantum Procedure:

1. Initial state: $|0\rangle^{\otimes n} |0\rangle^{\otimes n}$.
2. Create superposition by applying $H^{\otimes n}$ on the first register.
3. Apply the black box U_f.
4. Apply the $H^{\otimes n}$ on the first register.
5. Measure the registers.

Output: $z \in \{0,1\}^n$ which is an element in $H^{\perp} = \{x \in \{0,1\}^n : x \cdot s \equiv 0 \mod 2\}$.

The quantum procedure is implemented in the circuit of Figure 4.1.

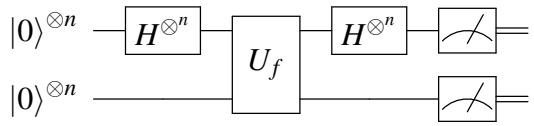


Figure 4.1: Circuit of the Quantum Procedure to Solve Simon's Problem.

Now, let us understand the outcome of the above circuit with a litte more detail.

Step 1 First, the algorithm starts with two registers, initialized to $|0\rangle^{\otimes n} |0\rangle^{\otimes n}$.

Step 2 Then, we apply the Hadamard transform to the first register, which gives the state

$$|0\rangle^{\otimes n} |0\rangle^{\otimes n} \xrightarrow{H^{\otimes n} \otimes I} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle |0\rangle^{\otimes n}.$$

Step 3 We apply the oracle U_f to obtain

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle |0\rangle^{\otimes n} \xrightarrow{U_f} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle |f(k)\rangle.$$

Step 4 We apply another Hadamard transform to the first register. This will produce the state

$$\begin{aligned} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle |f(k)\rangle \xrightarrow{H^{\otimes n} \otimes I} & \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left(\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{jk} |j\rangle \right) |f(k)\rangle \\ & = \sum_{j=0}^{2^n-1} |j\rangle \left(\frac{1}{2^n} \sum_{k=0}^{2^n-1} (-1)^{jk} |f(k)\rangle \right). \end{aligned}$$

Step 5 We measure the first register. The probability of finding the state $|j\rangle$ is

$$\left|\left|\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{jk}|f(k)\rangle\right|\right|^2.$$

But since there exist exactly two elements x_1, x_2, such that $f(x_1) = f(x_2) = z$ for each $z \in \text{Range}(f)$, then

$$\begin{aligned}\left|\left|\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{jk}|f(k)\rangle\right|\right|^2 &= \left|\left|\frac{1}{2^n}\sum_{z\in\text{Range}(f)}\left((-1)^{jx_1}+(-1)^{jx_2}\right)|z\rangle\right|\right|^2 \\ &= \left|\left|\frac{1}{2^n}\sum_{z\in\text{Range}(f)}\left((-1)^{jx_1}+(-1)^{j(x_1\oplus s)}\right)|z\rangle\right|\right|^2 \\ &= \left|\left|\frac{1}{2^n}\sum_{z\in\text{Range}(f)}(-1)^{jx_1}\left(1+(-1)^{js}\right)|z\rangle\right|\right|^2.\end{aligned}$$

Now, when js is odd then the expression will vanish, but when js is even then the expression is $2^{-(n-1)}$. Thus, the output $z \in \{0,1\}^n$ is a uniformly random element of $H^\perp = \{x \in \{0,1\}^n : x \cdot s \equiv 0 \mod 2\}$. Now, we can run the quantum subroutine $O(n)$ times to get a list of linearly independent bitstrings $z_1, \ldots, z_n$. By Lemma 2.61, we can ensure that $\text{span}\{z_1, \ldots, z_n\} = H^\perp$ with high probability.

The second step of Simon's algorithm is a purely classical post-processing step given by:

Algorithm 4.2 (Simon's Algorithm-Classical Procedure)**.**
Input: $z_1, \ldots, z_n \in \{0,1\}^n$

Classical Post processing: Apply Gaussian elimination to

$$\begin{array}{ccccccccc} z_1^1 x_1 & + & z_1^2 x_2 & + & \cdots & + & z_1^n x_n & \equiv & 0 \mod 2 \\ \vdots & & \vdots & & \vdots & & \vdots & \vdots & \\ z_n^1 x_1 & + & z_n^2 x_2 & + & \cdots & + & z_n^n x_n & \equiv & 0 \mod 2 \end{array}$$

Output: $s \in \{0,1\}^n$.

Simon's problem is a special case of the (Abelian) Hidden Subgroup Problem. For that, let us introduce the Hidden Subgroup Problem, and see why Simon's problem is a special case of it.

4.2 The Hidden Subgroup Problem

In this section, we start by stating the Hidden Subgroup Problem (HSP). Then, we see that Simon's problem is an instance of it. After that, we give a quantum procedure for finding subgroups of the form $\langle d \rangle$ in a cyclic group $(\mathbb{Z}/N\mathbb{Z}, +)$ ($d \in \mathbb{Z}/N\mathbb{Z}$). Lastly, we draw our attention to a quantum algorithm that find subgroups H of an Abelain group, $(\mathbb{Z}/p\mathbb{Z})^n$, with p a prime number. For that, we follow [Lom04].

Definition 4.3. Let G be a group and $H \leq G$ one of its subgroups. Let S be any set, and $f : G \to S$ be a function that distinguishes cosets of H, i.e., for all $g_1, g_2 \in G, f(g_1) = f(g_2) \Leftrightarrow g_1 H = g_2 H$. The hidden subgroup problem (HSP) is to determine a generating set for the subgroup H given the ability to evaluate f on elements of G.

It is worth mentioning that classical query complexity of the HSP is known. In fact, suppose that G has a set S of N subgroups such that $H_1 \cap H_2 \cap \ldots \cap H_S = \{e\}$ (e the identity of G). Then, a classical computer must make $\Omega(\sqrt{N})$ queries to solve the HSP [HK18]. For example, on Simon's problem, we worked over the group $(\mathbb{Z}/2\mathbb{Z})^n$ with the operation bitwise addition modulo 2, and the subgroup H is $\{0^n, s\}$ (so the goal of finding s is equivalent to learning the subgroup H). Then by [HK18], a classical computer must make $\Omega(\sqrt{2^n})$ queries to solve it. However, as we saw in the previous section, there is a quantum algorithm that makes $O(n)$ queries to solve it.

Now, let us see a quantum procedure for finding subgroups of a cyclic group of order N.

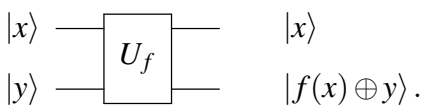
Example 4.4. Let N be an integer greater than 1, let X be a finite set, and consider the additive group of integers mod N denoted by $G = (\mathbb{Z}/N\mathbb{Z}, +)$. Let us state the problem, and see a solution.

Given: Suppose we are given a function $f : G \to X$ that separates cosets of the subgroup $H = \langle d \rangle$ of G.
Problem: Determine a generating set for H.

First, note that we can represent G and H as $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$, $H = \{|0\rangle, |d\rangle, \ldots, |(M-1)d\rangle\}$, where $M = |H|$. Second, assume we have U_f that performs the unitary transform,

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle,$$

for $x, y \in G$. In the quantum circuit model it would be represented by



Now, let us determine a generating set for H, trying to reduce the number of queries to the oracle U_f. The algorithm is as follows:

Algorithm 4.5 (Abelian Hidden Subgroup Problem for Cyclic Groups)**.**
Input: A black box that performs the operation $U_f |a\rangle |0\rangle = |a\rangle |f(a)\rangle$ for $a \in \{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$.

Quantum Procedure:

1. Initial state: $|0\rangle |0\rangle$.
2. Create superposition over all elements of G by applying the Quantum Fourier Transform 3.25 to the first register.
3. Apply the black box U_f.
4. Measure the second register.
5. Apply the quantum Fourier transform to the first register.
6. Measure the first register.

Output: $z \in \{0,1\}^{r_1}$, which is a bitstring that represents a multiple of M.

Let us see in more detail the steps of the algorithm:

Step 1 Initial state $|0\rangle |0\rangle$.

Step 2 Apply the quantum Fourier transform to the first register of the zero state, with this, we obtain a superposition over all elements of G:

$$|0\rangle |0\rangle \xrightarrow{QFT \otimes I} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |0\rangle .$$

Step 3 Now, apply U_f, which is actually the coset separating function f, to obtain

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |0\rangle \xrightarrow{U_f} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |f(j)\rangle .$$

Step 4 Now, since we know that f is a function that distinguishes cosets of H, we measure the second register collapsing the second register to some state $|f(j)\rangle$ and leaving the first register with those values where they agree on f, namely the coset $j+H$

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle |f(j)\rangle \xrightarrow{Measure} \frac{1}{\sqrt{|H|}} \sum_{h \in H} |j+h\rangle |f(j)\rangle = \frac{1}{\sqrt{M}} \sum_{s=0}^{M-1} |j+sd\rangle |f(j)\rangle .$$

Step 5 Then, apply the quantum Fourier transform F_N on the first register to get

$$\frac{1}{\sqrt{M}}\sum_{s=0}^{M-1}|j+sd\rangle \xrightarrow{QFT} \frac{1}{\sqrt{M}}\sum_{s=0}^{M-1}\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{2\pi i(j+sd)k/N}|k\rangle$$
$$= \frac{1}{\sqrt{MN}}\sum_{k=0}^{N-1}e^{2\pi ijk/N}|k\rangle\sum_{s=0}^{M-1}e^{2\pi isdk/N}.$$

Using that $Md = N$, we can evaluate the geometric series

$$\sum_{s=0}^{M-1}\left(e^{2\pi ik/M}\right)^s$$

in which if $M|k$ then there exist $c \in \mathbb{Z}^+$ such that $k = Mc$, so

$$\sum_{s=0}^{M-1}\left(e^{2\pi ik/M}\right)^s = \sum_{s=0}^{M-1}\left(e^{2\pi iMc/M}\right)^s = \sum_{s=0}^{M-1}\left(e^{2\pi ic}\right)^s = \sum_{s=0}^{M-1}1^s = M.$$

But when $M \nmid k$ then

$$\sum_{s=0}^{M-1}\left(e^{2\pi ik/M}\right)^s = \frac{1-\left(e^{2\pi ikM/M}\right)}{1-e^{2\pi ik/M}} = \frac{1-\left(e^{2\pi ik}\right)}{1-e^{2\pi ik/M}} = \frac{0}{1-e^{2\pi ik/M}} = 0.$$

Therefore

$$\frac{1}{\sqrt{MN}}\sum_{k=0}^{N-1}e^{2\pi ijk/N}|k\rangle\sum_{s=0}^{M-1}e^{2\pi isdk/N} = \frac{1}{\sqrt{MN}}\left(\sum_{\substack{k=0\\M|k}}^{N-1}e^{2\pi ijk/N}|k\rangle\sum_{s=0}^{M-1}e^{2\pi isdk/N}\right)$$
$$+\frac{1}{\sqrt{MN}}\left(\sum_{\substack{k=0\\M\nmid k}}^{N-1}e^{2\pi ijk/N}|k\rangle\sum_{s=0}^{M-1}e^{2\pi isdk/N}\right)$$
$$= \frac{1}{\sqrt{MN}}\left(\sum_{\substack{k=0\\M|k}}^{N-1}e^{2\pi ijk/N}|k\rangle M\right)$$
$$= \frac{M}{\sqrt{MN}}\left(\sum_{t=0}^{d-1}e^{2\pi ijtM/N}|tM\rangle\right)$$
$$= \frac{1}{\sqrt{d}}\left(\sum_{t=0}^{d-1}e^{2\pi ijtM/N}|tM\rangle\right).$$

Step 6 Finally, measuring the first register, gives as a result a multiple of M in $\{0, M, \ldots, (d-1)M\}$ with uniform probability. We repeat this quantum procedure l times for some positive

integer l, and obtain l multiples of M, let us call them $t_1M,\ldots,t_lM$. Then, we compute the greater common divisor of all of them $\gcd(t_1,\ldots,t_l)$ with the Euclidean algorithm. Now, by Lemma 2.63 we know that $\Pr(\gcd(t_1,\ldots,t_l)=1)\geq 1-\left(\frac{1}{2}\right)^{\frac{l}{2}}$, hence we determine M, and therefore H (because $d=\frac{N}{M}$).

Quantum algorithm for the HSP over finite Abelian groups of the form $(\mathbb{Z}/p\mathbb{Z})^n$.

In Chapter 6 we will be interested on finding certain types of substructures of a finite dimensional Algebra over $\mathbb{F}_p$, with p a prime number. For that, we state the problem as an instance of the (Abelian) HSP. Here, we present a quantum procedure to find a subgroup H of the Abelian group $(\mathbb{Z}/p\mathbb{Z})^n$, with n a positive integer, based on [Lom04].

Let $G\cong(\mathbb{Z}/p\mathbb{Z})^n$. Suppose we are given a function $f:G\to X$ to a finite set X such that there is a subgroup $H<G$ with the property that f separates cosets of H. We want to find a generating set for H. Now, the $p^n=N$ elements of G which has the form of n-tuples $x=(x_1,\ldots,x_n)$ can be represented as $|x_1\rangle|x_2\rangle\cdots|x_n\rangle$. Let us see a quantum algorithm for the HSP over this kind of finite Abelian groups. For now, we ignore the size of the quantum circuit. However in Chapter 6 we will give explicit details on the size of it. The algorithm is as follows:

Algorithm 4.6 (Finding the orthogonal of a subgroup H of a group of the form $(\mathbb{Z}/p\mathbb{Z})^n$)**.**
Input: A black box which performs the operation $U_f|x\rangle|0\rangle=|x\rangle|f(x)\rangle$ for x in the vector space over $\mathbb{C}$ which contains the group elements of G.

Quantum Procedure:

1. Initial state: $|0\rangle^{\otimes r_1}|0\rangle^{\otimes r_2}$, for $r_1=n\lceil\log_2 p\rceil$ and $r_2\in\mathbb{N}$ (ancillary qubits for the oracle U_f).
2. Create superposition over all elements of G by applying the Quantum Fourier Transform (Equation 3.1) to the first register.
3. Apply the black box U_f.
4. Measure the second register.
5. Apply the Quantum Fourier Transform (Equation 3.1) on the first register.
6. Measure the first register.

Output: $z\in\{0,1\}^{r_1}$ a bitstring of an element in $H^\perp$ (here $H^\perp$, is as Definition 2.57).

Let us analyze the procedure step by step.

Step 1 Initial state $|0\rangle|0\rangle$.

Step 2 Apply the Quantum Fourier Transform (Equation 3.1 from Subsection 3.3.1) to the first register in order to obtain a superposition over all elements of G :

$$|0\rangle|0\rangle \xrightarrow{QFT\otimes I} \frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|0\rangle.$$

Step 3 Apply U_f, to obtain

$$\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|0\rangle \xrightarrow{U_f} \frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|f(g)\rangle.$$

Step 4 Measure the second register. So, since we know that f is a function that distinguishes cosets of H, i.e., f returns the same value if and only if g_1 and g_2 belong to the same coset of H, measuring the second register will collapse to some state $|f(g_z)\rangle$ and will leave the first register with those values where they agree on f, namely the coset g_z+H.

So, if $\{x_1,x_2,\ldots,x_l\}$ is a $\mathbb{Z}/p\mathbb{Z}$-basis of the subgroup H of G, then for all $z\in \text{Range}(f)$, there exists $g_z\in G$ such that $z=f(g_z+\sum_{i=1}^{l}\lambda_i x_i)$, for all $0\leq\lambda_1,\ldots,\lambda_l\leq p-1$. So, the probability of the outcome $f(g_z)$ is

$$\left\|\frac{1}{\sqrt{|G|}}\sum_{\substack{g\in G\\ \text{such that } f(g)=z}}|g\rangle\right\|^2=\frac{1}{|G|}|H|,$$

and the state after the measurement is,

$$\frac{1}{\sqrt{\frac{|H|}{|G|}}}\frac{1}{\sqrt{|G|}}\sum_{\substack{g\in G\\ \text{such that } f(g)=z}}|g\rangle|z\rangle=\frac{1}{\sqrt{|H|}}\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\left|g_z+\sum_{i=1}^{l}\lambda_i x_i\right\rangle|z\rangle.$$

Step 5 Apply the Fourier transform F_G from Subsection 3.3.1, Equation 3.1, to the first register to obtain

$$\begin{aligned}\frac{1}{\sqrt{|H|}}\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\left|g_z+\sum_{i=1}^{l}\lambda_i x_i\right\rangle|z\rangle &\xrightarrow{QFT}\frac{1}{\sqrt{|G|}}\frac{1}{\sqrt{|H|}}\sum_{g,h\in G}\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\chi_g(h)|h\rangle\left\langle g\left|g_z+\sum_{i=1}^{l}\lambda_i x_i\right.\right\rangle|z\rangle\\ &=\frac{1}{\sqrt{|G|}}\frac{1}{\sqrt{|H|}}\sum_{h\in G}\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\chi_{g_z+\sum_{i=1}^{l}\lambda_i x_i}(h)|h\rangle|z\rangle.\end{aligned}$$

From Section 2.5, for $g,h\in G$ with $g=(g_1,\ldots,g_n),h=(h_1,\ldots,h_n)$ and $g_i,h_i\in\mathbb{Z}/p\mathbb{Z}$, for all $i=1,\ldots,n$ and $\omega_p=\exp\left(\frac{2\pi i}{p}\right)$ we have that,

$$\chi_g(h)=\omega_p^{gh}=\omega_p^{g_1h_1+\cdots+g_kh_k}.$$

Thus,

$$\frac{1}{\sqrt{|G|}}\frac{1}{\sqrt{|H|}}\sum_{h\in G}\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\chi_{g_z+\sum_{i=1}^{l}\lambda_i x_i}(h)\,|h\rangle\,|z\rangle=\frac{1}{\sqrt{|G|}}\frac{1}{\sqrt{|H|}}\sum_{h\in G}\left(\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\omega_p^{h(g_z+\sum_{i=1}^{l}\lambda_i x_i)}\right)|h\rangle\,|z\rangle\,.$$

Step 6 Measuring the first register, we obtain a random element uniformly distributed over $H^{\perp}$. Indeed, if we measure the first register, the probability of measuring a state $|h\rangle$ is

$$\left\|\frac{1}{\sqrt{|G||H|}}\sum_{g\in G}\sum_{h\in G}\left(\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\omega_p^{h(g_z+\sum_{i=1}^{l}\lambda_i x_i)}\right)|h\rangle\right\|^2.$$

Then extracting the common factor ω^{hg_z}, the probability becomes

$$\left\|\frac{1}{\sqrt{|G||H|}}\left(\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\omega_p^{h(g_z+\sum_{i=1}^{l}\lambda_i x_i)}\right)|z\rangle\right\|^2=\left\|\frac{1}{\sqrt{|G||H|}}\omega_p^{hg_z}\prod_{i=1}^{l}\left(\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\omega_p^{h\lambda_i x_i}\right)|z\rangle\right\|^2.$$

Now, when $hx_i\neq 0 \bmod p$ for some $i=1,\ldots,l$, then by Proposition 2.59 we have that

$$\sum_{\lambda_i=0}^{p-1}\omega_p^{\lambda hx_i}=0.$$

Otherwise, $h\in H^{\perp}$, and the probability of finding the state $|h\rangle$ is,

$$\frac{1}{\sqrt{|G||H|}^2}\left|\omega_p^{hg_z}\prod_{i=1}^{l}\left(\sum_{\lambda_i=0}^{p-1}\omega_p^{0}\right)\right|^2=\frac{1}{p^n p^l}(1\cdot p^l)^2=p^{l-n}.$$

Thus, we obtain an element uniformly distributed over $H^{\perp}$.

Run the algorithm $t_1+\lceil\log_2 G\rceil$ times to obtain $g_1,\ldots,g_{t_1+\lceil\log_2 G\rceil}$ elements of $H^{\perp}$. Now, recall that from Section 2.6 that $\varphi_{t_1+\lceil\log_2 G\rceil}(H^{\perp})$ denotes the probability that $t_1+\lceil\log_2 G\rceil$ random elements of $H^{\perp}$ generate the entire subgroup. Thus, by Theorem 2.62 we know that

$$\varphi_{t_1+\lceil\log_2 G\rceil}(H^{\perp})=Pr\left(\langle\tilde{g}_1,\tilde{g}_2,\ldots,\tilde{g}_{t_1+\lceil\log_2 G\rceil}\rangle=H^{\perp}\right)\geq 1-\frac{1}{2^{t_1}}.$$

Note that $h\in H$ if and only if $\chi_h(\tilde{g}_k)=1$ for $\tilde{g}_k\in H^{\perp}$ with $k=1,2,\ldots,t_1+\lceil\log_2 G\rceil$, which is equivalent to $h\cdot\tilde{g}_k=0$ for all $k=1,2,\ldots,t_1+\lceil\log_2 G\rceil$ (see Section 2.5). Now, let us consider the following system of $t_1+\lceil\log_2 G\rceil$ linear equations:

$$\begin{array}{ccccccccc}
\tilde{g}_1^1x_1 & + & \tilde{g}_1^2x_2 & + & \cdots & + & \tilde{g}_1^nx_n & \equiv & 0 \mod p\\
\vdots & & \vdots & & \vdots & & \vdots & \vdots & \\
\tilde{g}_{t_1+\lceil\log_2 G\rceil}^1x_1 & + & \tilde{g}_{t_1+\lceil\log_2 G\rceil}^2x_2 & + & \cdots & + & \tilde{g}_{t_1+\lceil\log_2 G\rceil}^nx_n & \equiv & 0 \mod p
\end{array}$$

The second step of the algorithm is a purely classical post-processing step in which, for instance, an algorithm like Gaussian elimination can be used to compute the solution of the system over $\mathbb{F}_p$. In fact, taking $t_1=s$ for s a positive integer and applying Gaussian elimination the solutions will generate H with probability at least $1-2^{-s}$.

4.3 Quantum Phase Estimation

As another application of the QFT, we have the Quantum Phase Estimation algorithm. It is used to estimate the eigenvalue of an eigenvector of a unitary operator. The algorithm was initially introduced by Alexei Kitaev in 1995 [Kit95]. The problem is formally stated as follows:

Given: U a unitary operator that operates on m qubits and an eigenvector $|\psi\rangle$ of U such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, $0 \leq \theta < 1$.

Problem: Find the eigenvalue $e^{2\pi i\theta}$ of $|\psi\rangle$, which is equivalent to estimating the value θ.

The phase estimation algorithm is:

Algorithm 4.7 (Quantum Phase Estimation)**.**
Input: A black box which performs controlled U^j operations, for any integer j, an eigenstate $|\psi\rangle$ of U with eigenvalue $e^{2\pi i\theta}$, and n qubits initialized to $|0\rangle$.

Quantum Procedure:

1. Apply $H^{\otimes n}$ to the first n qubits.
2. Apply all the n controlled operations CU^{2^j} for $0 \leq j \leq n-1$, on the second register.
3. Apply the inverse Quantum Fourier Transform on the first n qubits.
4. Measure the first n qubits.

Output: $\lfloor 2^n\theta \rfloor$ with probability greater or equal than $\frac{4}{\pi^2}$, from which we can estimate θ.

The procedure is implemented in Figure 4.2.

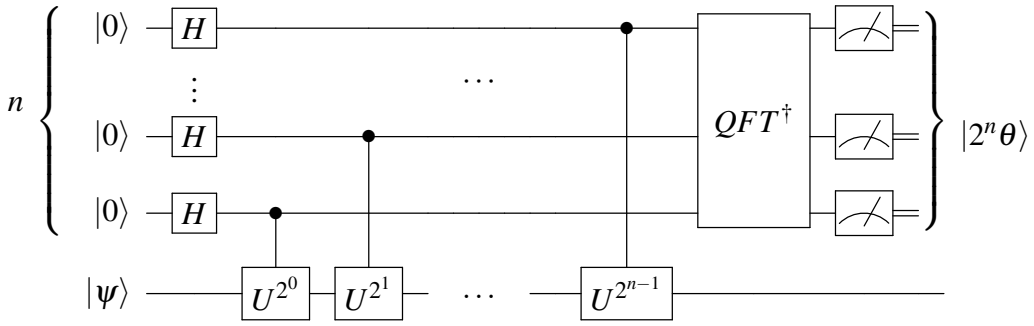


Figure 4.2: Circuit of the Quantum Phase Estimation

Let us analyze the steps of the procedure:

Step 1 Create superposition by applying $H^{\otimes n}$ on the first register. We get,

$$|0\rangle^{\otimes n}|\psi\rangle \xrightarrow{H^{\otimes n}\otimes I} = \frac{1}{\sqrt{2^n}}\left(\sum_{j=0}^{2^n-1}|j\rangle\right)|\psi\rangle.$$

Step 2 Apply all the n controlled operations CU^{2^j} for $1 \leq j \leq n-1$. Note that, $U^{2^j}|\psi\rangle = Ue^{2\pi i 2^j\theta}$, and the transformation implemented by the controlled gates applying $U^{2^0}, U^{2^1}, \ldots, U^{2^{n-1}}$ is $|k\rangle|\psi\rangle \mapsto |k\rangle U^k|\psi\rangle$. Indeed, let $\sum_{j=0}^{n-1} 2^{n-1-j}k_{n-1-j}$ with $k_j \in \{0,1\}$ be the binary representation of k, so

$$U^{\sum_{j=0}^{n-1} 2^{n-1-j}k_{n-1-j}} = \prod_{j=0}^{n-1} U^{2^{n-1-j}k_{n-1-j}}.$$

Recall that the unitary CU operation applies the unitary operator U on the target register only if its corresponding control qubit is $|1\rangle$. So, $U^{2^j k_j}$ will only be apply if the qubit k_j is 1. Therefore,

$$\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}|j\rangle|\psi\rangle \xrightarrow{CU^{2^0},CU^{2^1},\ldots,CU^{2^{2^n-1}}} \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi ik\theta}|k\rangle|\psi\rangle.$$

Step 3 Apply the inverse Quantum Fourier Transform to obtain

$$\begin{aligned}\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i\theta k}|k\rangle\otimes|\psi\rangle \xrightarrow{QFT^{-1}} &\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi ik\theta}\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{-2\pi ikx/2^n}|x\rangle\otimes|\psi\rangle \\ &= \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1} e^{(-2\pi ik)(x-2^n\theta)/2^n}|x\rangle\otimes|\psi\rangle.\end{aligned}$$

Step 4 Lastly, we perform a measurement of the first register. First, note that $2^n\theta$ can be written as $a+2^n\delta$ where $a = \lfloor 2^n\theta \rfloor$ and $0 \leq |\delta| \leq \frac{1}{2^{n+1}}$, so

$$e^{-2\pi ik(x-2^n\theta)/2^n} = e^{-2\pi ik(x-a-2^n\delta)/2^n} = e^{-\frac{2\pi ik}{2^n}(x-a)}e^{2\pi ik\delta}.$$

Then,

$$\frac{1}{2^n}\sum_{x=0}^{2^{n-1}}\sum_{k=0}^{2^{n-1}} e^{(-2\pi ik)(x-2^n\theta)/2^n}|x\rangle|\psi\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^{n-1}}\sum_{k=0}^{2^{n-1}} e^{-\frac{2\pi ik}{2^n}(x-a)}e^{2\pi ik\delta}|x\rangle|\psi\rangle.$$

Thus, performing a measurement in the computational basis on the first register yields the result $|a\rangle$ with probability

$$\left|\frac{1}{2^n}\sum_{k=0}^{2^{n-1}} e^{-\frac{2\pi ik}{2^n}(x-a)}e^{2\pi ik\delta}\right|^2 = \frac{1}{2^{2n}}\left|\sum_{k=0}^{2^{n-1}} e^{2\pi ik\delta}\right|^2,$$

which is 1 if $\delta = 0$. However if $\delta \neq 0$ then using the formula of the geometric series

$$\sum_{k=0}^{n-1} x^k = \frac{1-x^n}{1-x},$$

we have that

$$\frac{1}{2^{2n}}\left|\sum_{k=0}^{2^{n-1}} e^{2\pi i k\delta}\right|^2 = \frac{1}{2^{2n}}\left|\frac{1-\left(e^{2\pi i\delta}\right)^{2n}}{1-e^{2\pi i\delta}}\right|^2 = \frac{1}{2^{2n}}\left|\frac{1-e^{2\pi i\delta 2^n}}{1-e^{2\pi i\delta}}\right|^2.$$

Since $|\delta| \leq \frac{1}{2^{n+1}}$, it follows that $2\pi\delta 2^n \leq \pi$, and hence $\pi\delta 2^n \leq \frac{\pi}{2}$. Now,

$$\left|1-e^{2\pi 2^n\delta i}\right| = \sqrt{1-2\cos(2\pi\delta 2^n)+1} = 2\left|\sin(\pi\delta 2^n)\right|,$$

and

$$\left|1-e^{2\pi\delta i}\right| = \sqrt{1-2\cos(2\pi\delta)+1} = 2\left|\sin(\pi\delta)\right|.$$

Since, $|\sin(\pi\delta)| \leq |\pi\delta|$ and $|\sin(\pi\delta 2^n)| \geq \frac{2\pi\delta 2^n}{\frac{\pi}{2}} = 4\delta 2^n$,

$$\frac{1}{2^{2n}}\left|\frac{1-e^{2\pi i\delta 2^n}}{1-e^{2\pi i\delta}}\right|^2 \geq \left(\frac{1}{2^n}\left(\frac{4\delta 2^n}{2\pi\delta}\right)\right)^2 = \frac{4}{\pi^2}.$$

4.4 Grover's Search Algorithm

As we mentioned in the Introduction, another advantage that a quantum computer has over a classical computer is in searching databases. Indeed, Lov Kumar Grover introduced a quantum algorithm, based on the concepts of superposition and quantum parallelism, for searching databases that is quadratically faster than any possible classical algorithm for the same purpose. More explicitly, consider a list of N items in which we want to locate an element ω that we say is marked. To find the marked element using classical computation, in the worst scenario, we would have to check all of them. On a quantum computer, however, with Grover's amplitude amplification technique, the item ω can be found in $O(\sqrt{N})$ queries with probability at least $\frac{1}{2}$. Now, let us present formally the problem and the quantum procedure of Grover's algorithm more explicitly:

Given: The search problem can conveniently be represented by a function $f : \{0,1,\ldots,N-1\} \to \{0,1\}$, with $N = 2^n$ and $n \in \mathbb{N}$ such that

$$f(x) = \begin{cases} 0 & \text{if } x \text{ is not a marked element} \\ 1 & \text{if } x \text{ is a marked element} \end{cases}$$

Problem: Find x such that $f(x) = 1$ (that is, a marked element).

Let us assume that we can access f in the form of a standard quantum oracle. This standard oracle, denoted as U_f, uses an ancillary qubit to compute

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle .$$

It holds that,

$$U_f |x\rangle |y\rangle = \begin{cases} |x\rangle |y \oplus 1\rangle & \text{for} \quad x = \omega, \quad \text{(if and only if } f(x) = 1) \\ |x\rangle |y\rangle & \text{for} \quad x \neq \omega, \quad \text{(if and only if } f(x) = 0) . \end{cases}$$

Another vital ingredient is the reflection $U_s = 2|s\rangle\langle s| - I$, where $|s\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$, called the diffusion operator. Its implementation can be seen in the circuit from Figure 4.3.

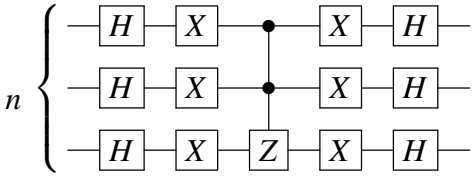


Figure 4.3: Circuit of the diffusion operator.

Now, based on [J06] and [BBHT98] we present Grover's algorithm, also known as the quantum search algorithm, which is as follows:

Algorithm 4.8 (Grover's quantum search)**.**
Input: A black box U_f which performs the transformation $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$, where $f(x) = 0$ for all $0 \leq x < N = 2^n$ except ω for which $f(\omega) = 1$.

Quantum Procedure:

1. Initial state: $|0\rangle^{\otimes n} |0\rangle$.
2. Create superposition. Apply $H^{\otimes n}$ to the first n qubits, and XH to the last qubit.
3. Apply $G = U_s U_f$, $\lfloor \frac{\pi\sqrt{N}}{4} \rfloor$ times.
4. Measure the first n qubits, and apply H on the last qubit.

Output: ω, with probability greater or equal than $\frac{1}{2}$.

The quantum procedure is implemented in the circuit shown in Figure 4.4.
Let us analyze the algorithm step by step.

Step 1 The algorithm uses a register with $n+1$ qubits. It starts out with $|0\rangle^n |0\rangle$ as the initial state.

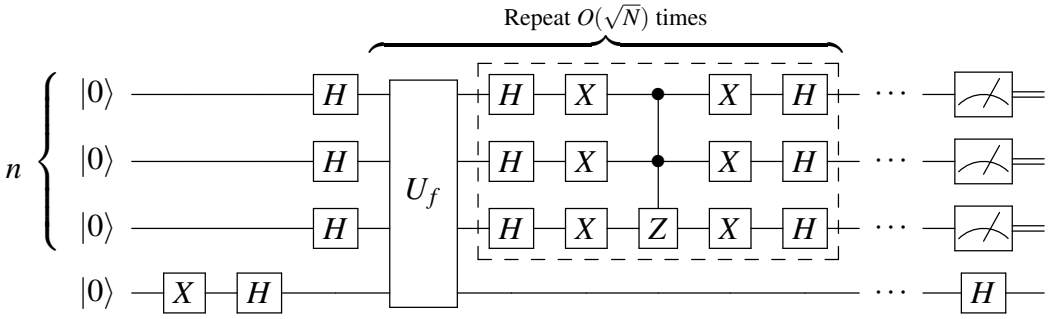


Figure 4.4: Circuit of Grover's Quantum Search.

Step 2 We apply the NOT gate, and after that, we apply the Hadamard transform to the last qubit,

$$\begin{aligned} |0\rangle^{\otimes n}|0\rangle &\xrightarrow{I^{\otimes n}\otimes X} |0\rangle^{\otimes n}(|1\rangle) \\ &\xrightarrow{I^{\otimes n}\otimes H} |0\rangle^{\otimes n}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \\ &\xrightarrow{I^{\otimes n}\otimes H} |0\rangle^{\otimes n}|-\rangle . \end{aligned}$$

Now, we apply the Hadamard transform to the first n qubits, which gives the uniform superposition $|s\rangle$ given by

$$|0\rangle^{\otimes n}|-\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|-\rangle = |s\rangle|-\rangle$$

Step 3 We apply the oracle U_f to the state $|s\rangle$. Now, after applying U_f, we apply $U_s = 2|s\rangle\langle s| - I_n$, known as the diffusion operator. This transformation is, thus, U_sU_f, which correspond to a rotation. Actually, it rotates the initial state $|s\rangle$ closer towards $|\omega\rangle$. Let us see why. Consider the following sets $A = \{x \in \{0,1\}^n : \ f(x) = 1\}$ and $B = \{x \in \{0,1\}^n : \ f(x) = 0\}$. Denote $a = |A|$, and $b = |B|$. Let us assume that $a \leq \frac{N}{2}$. Additionally define

$$|A\rangle = \frac{1}{\sqrt{a}}\sum_{x\in A}|x\rangle \text{ and } |B\rangle = \frac{1}{\sqrt{b}}\sum_{x\in B}|x\rangle ,$$

which are orthogonal unit vectors. Moreover, $|s\rangle$ is a vector in the subspace spanned by $\{|A\rangle, |B\rangle\}$. Indeed

$$|s\rangle = \sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle .$$

Let $G = U_sU_f$, and let us see how G acts on the states of $|A\rangle$ and $|B\rangle$. Since they are orthogonal

vectors and $\langle v| = (|v\rangle)^\dagger$, it holds that

$$\begin{aligned}
G|A\rangle|-\rangle &= U_s U_f |A\rangle|-\rangle \\
&= (2|s\rangle\langle s| - I_n)(-|A\rangle)|-\rangle \\
&= ((-2|s\rangle\langle s|)|A\rangle + |A\rangle)|-\rangle \\
&= (-2\langle s|A\rangle|s\rangle + |A\rangle)|-\rangle \\
&= \left(-2\sqrt{\frac{a}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) + |A\rangle\right)|-\rangle \\
&= \left(\left(1-\frac{2a}{N}\right)|A\rangle - \frac{2\sqrt{ab}}{N}|B\rangle\right)|-\rangle,
\end{aligned}$$

analogously for $G|B\rangle$ we have

$$\begin{aligned}
G|B\rangle|-\rangle &= U_s U_f |B\rangle|-\rangle \\
&= (2|s\rangle\langle s| - I_n)|B\rangle|-\rangle \\
&= ((2|s\rangle\langle s|)|B\rangle - |B\rangle)|-\rangle \\
&= (2\langle s|B\rangle|s\rangle - |B\rangle)|-\rangle \\
&= \left(2\sqrt{\frac{b}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) - |B\rangle\right)|-\rangle \\
&= \left(\frac{2\sqrt{ab}}{N}|A\rangle + \left(\frac{2b}{N}-1\right)|B\rangle\right)|-\rangle. \\
&= \left(\frac{2\sqrt{ab}}{N}|A\rangle - \left(1-\frac{2b}{N}\right)|B\rangle\right)|-\rangle.
\end{aligned}$$

Therefore, G maps the subspace spanned by $\{|A\rangle, |B\rangle\}$ to the subspace spanned by $\{|A\rangle, |B\rangle\}$. Furthermore, the action of G on the space spanned by $\{|A\rangle, |B\rangle\}$ can be seen as the following matrix where the first row and column correspond to $|B\rangle$ and the second row and column to $|A\rangle$:

$$\begin{aligned}
\begin{bmatrix} -\left(1-\frac{2b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(1-\frac{2a}{N}\right) \end{bmatrix} &= \begin{bmatrix} -\left(\frac{N-2b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(\frac{N-2a}{N}\right) \end{bmatrix} \\
&= \begin{bmatrix} -\left(\frac{a+b-2b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(\frac{a+b-2a}{N}\right) \end{bmatrix} \\
&= \begin{bmatrix} -\left(\frac{a-b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(\frac{b-a}{N}\right) \end{bmatrix}.
\end{aligned}$$

Note that

$$\begin{bmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{a}{N}} & \sqrt{\frac{b}{N}} \end{bmatrix}^2 = \begin{bmatrix} -\left(\frac{a-b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(\frac{b-a}{N}\right) \end{bmatrix}.$$

Now, take $\theta \in \left(0, \frac{\pi}{2}\right)$ such that $\sin\theta = \sqrt{\frac{a}{N}}$ and $\cos\theta = \sqrt{\frac{b}{N}}$. Recall that, in two dimensions, the standard rotation matrix has the form

$$\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

for $\phi \in [0, 2\pi]$. So, using a rotation of angle 2θ we get

$$\begin{aligned} \begin{bmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{bmatrix} &= \begin{bmatrix} \cos^2\theta - \sin^2\theta & -\cos\theta\sin\theta - \sin\theta\cos\theta \\ \sin\theta\cos\theta + \cos\theta\sin\theta & -\sin^2\theta + \cos^2\theta \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^2 \\ &= \begin{bmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{a}{N}} & \sqrt{\frac{b}{N}} \end{bmatrix}^2 \\ &= \begin{bmatrix} -\left(\frac{a-b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(\frac{b-a}{N}\right) \end{bmatrix}. \end{aligned}$$

Therefore, $G = U_s U_f$ is a rotation of angle 2θ in the two-dimensional space spanned by $\{|A\rangle, |B\rangle\}$ rotating the space by 2θ radians per application of G, with $\theta = \arcsin\sqrt{\frac{a}{N}}$. Also

$$|s\rangle = \sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle = \sin\theta\,|A\rangle + \cos\theta\,|B\rangle.$$

Applying G k times, we get

$$G^k|s\rangle|-\rangle = (\cos((2k+1)\theta)|B\rangle + \sin((2k+1)\theta)|A\rangle)|-\rangle.$$

We would like the state of $|s\rangle$ to be as close to $|A\rangle$ as possible, i.e., $\sin((2k+1)\theta) \approx 1$ or $(2k+1)\theta \approx \frac{\pi}{2}$ which gives us $k \approx \frac{\pi}{4\theta} - \frac{1}{2}$.

If $a = 1$, which is the case that we first consider, then $\theta = \arcsin\sqrt{\frac{1}{N}} \approx \frac{1}{\sqrt{N}}$, hence

$$k \approx \frac{\pi}{4\theta} - \frac{1}{2} = \frac{\pi}{4\frac{1}{\sqrt{N}}} - \frac{1}{2} = \frac{\pi\sqrt{N}}{4} - \frac{1}{2},$$

and since k must be an integer, then we take

$$k = \left\lfloor \frac{\pi\sqrt{N}}{4} \right\rfloor.$$

If $a > 1$, as before we want the state of $|s\rangle$ to be as close to $|A\rangle$ as possible, or equivalently that $\cos((2k+1)\theta)$ is close to 0 as possible. So, we would have that $\cos((2k+1)\theta) = 0$, when $k = (\pi - 2\theta)/4\theta$ if that were an integer. Consider, $\tilde{k} = \lfloor \pi/4\theta \rfloor$. Note that $|\tilde{k} - k| \leq \frac{1}{2}$, hence

$$\left|\left(2\tilde{k}+1\right)\theta - \frac{\pi}{2}\right| = \left|\left(2\tilde{k}+1\right)\theta - (2k+1)\theta\right| = \left|2\theta\left(k-\tilde{k}\right)\right| \leq \theta.$$

Therefore, $|\cos\big((2\tilde{k}+1)\theta\big)| \leq |\sin\theta|$, which will be used as a bound for computing the probability of not finding a marked element, after applying k times the G operator.

Step 4 Measurement. When we measure the first n qubits, after $\tilde{k}$ iterations, and since

$$|\cos\big((2\tilde{k}+1)\theta\big)| \leq |\sin\theta|,$$

the probability of not finding a marked element is,

$$\cos^2((2\tilde{k}+1)\theta) \leq \sin^2\theta = \frac{a}{N} \leq \frac{1}{2}.$$

One must be careful in using this algorithm because the probability of success does not necessarily increase with the number of iterations. For instance, if we have a list of 20 elements and only one is marked, and if we apply $\lfloor \frac{\pi}{2}\sqrt{20} \rfloor$ iterations of G the probability of finding the marked element is 0.06. But if we apply $\lfloor \frac{\pi}{4}\sqrt{20} \rfloor$ iterations, the probability is 0.94. Hence, the estimation of the necessary number of iterations is one of the most important parts in the algorithm. For this reason, in the next section, we are going to briefly explain how to apply Grover's algorithm in the case where we do not know how many marked elements there are, following [CRO+24].

As a particular case, on Chapter 7, the methodology developed in [CRO+24] is going to be use for finding satisfying assignments of a Boolean function $f : \{0,1\}^n \to \{0,1\}$.

4.5 Technique For Listing All Elements Marked By An Oracle

In this section, we are going to introduce a technique for finding all marked elements from a list of ν elements, with a quantum algorithm inspired by Grover's quantum search, with a desired probability. For that, we follow [CRO+24]. This technique, in particular, is of interest to us, because it can be applied for finding satisfying assignments of a certain type of Boolean functions f that we are going to deal with in Chapter 7.

From the previous section,we know that Grover's algorithm only returns a marked element with a certain probability. However, when the number of marked elements is unknown, we do not know the number of oracle queries that we need to use. So, for that, let us suppose we have an oracle U_f, given by $U_f(|x\rangle|0\rangle) = |x\rangle|1\rangle$ if x is marked, otherwise $U_f(|x\rangle|0\rangle) =$

$|x\rangle|0\rangle$, such that mark exactly μ elements from a set of ν elements. Then, apply j times the Grover's operator, with j chosen randomly from $\{0,\ldots,\lfloor\sqrt{\nu}\rfloor-1\}$. After that, let us measure, and see if the result is correct, in case it is, we apply again Grover's algorithm, but with a new oracle, U'_f, such that the marked element found x_0 is excluded from being marked by it (U'_f), so $U'_f(|x\rangle|0\rangle) = |x\rangle|1\rangle$ if x is marked and $x \neq x_0$, otherwise $U'_f(|x\rangle|0\rangle) = |x\rangle|0\rangle$. However if the result x_0 is not correct, the oracle is not modify, and we apply again Grover's algorithm with j times the Grover's operator, and j chosen randomly from $\{0,\ldots,\lfloor\sqrt{\nu}\rfloor-1\}$. This procedure is applied iteratively until all elements are found. More explicitly the procedure is as follows:

Algorithm 4.9 (Algorithm 2 [CRO+24])**.**
Data: An oracle U_f marking an *unknown* number of μ elements (upper bounded by a *known or estimated* B) in a database of ν elements ($0 \leq \mu \leq B \leq \frac{3\nu}{4}$). A desired error bound $0 < w < 1$.
Result: A set of r marked database elements $L = \{x_1,\ldots,x_r\}$. With probability at least $1-w$, we will have $r = \mu$.
Procedure:

$L \leftarrow \emptyset$
$R \leftarrow \left\lceil \frac{\log\left(1-(1-w)^{\frac{1}{B}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil$
$found \leftarrow false$
$done \leftarrow false$
while $done = false$ **do**
  $l \leftarrow 1$;
  while $found = false$ and $l \leq R$ **do**
    Choose j uniformly at random from the set $\{0,\ldots,\lfloor\sqrt{\nu}\rfloor-1\}$;
    Run Grover's algorithm with j applications of the oracle plus diffusion operator;
    Measure to obtain x;
    if x is a marked element **then** $found \leftarrow true$;
    else $l \leftarrow l+1$;
    end if
  end while
  if $found = true$ **then** ;
    $L \leftarrow L \cup \{x\}$; ▷ Add found element and search for another
    $found \leftarrow false$;
    Modify the oracle so that it does not mark x
  else $done \leftarrow true$; ▷ Tried R times without finding anything
  end if
end while
return L

Note that, in the case where the number of elements marked by the oracle is greater than

0, by Lemma 2 and the proof of Theorem 3 in [BBHT98], the probability of finding a marked element on the nested while, is $\delta(\nu) \geq \frac{1}{4}, 1$ with $O(\sqrt{\nu})$ oracle calls. So, the overall probability of finding a marked element is

$$1-(1-\delta(\nu))^{R(\nu)} \geq 1-\left(\frac{3}{4}\right)^{R(\nu)}.$$

Since the loop of the first while must be independently repeated $\mu(\nu)+1$ times for the algorithm to succeed (the last iteration is the one forcing the output), the probability $P'(\nu)$ of not finding all marked elements is

$$P'(\nu) := 1-\left(1-(1-\delta(\nu))^{R(\nu)}\right)^{\mu(\nu)} \leq 1-\left(1-\left(\frac{3}{4}\right)^{R(\nu)}\right)^{\mu(\nu)}$$

which, in order to obtain a bounded algorithm, is required to be less than some $w<1$, for all ν. This yields

$$R(\nu) \geq \frac{\log\left(1-(1-w)^{\frac{1}{\mu(\nu)}}\right)}{\log\left(\frac{3}{4}\right)}.$$

Taking $R(\nu)$ to be

$$\left\lceil \frac{\log\left(1-(1-w)^{\frac{1}{\mu(\nu)}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil,$$

we have that $R(\nu) = O(\log(\mu(\nu)))$, and the procedure requires an overall number of

$$O\left(\sqrt{\nu}\mu(\nu)\log(\mu(\nu))\right)$$

oracle calls. Taking, $B = \frac{3\nu}{4}$

$$R(\nu) = \left\lceil \frac{\log\left(1-(1-w)^{\frac{1}{B(\nu)}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil = O(\log(B)),$$

and the overall asymptotic complexity is $O\left(\sqrt{\nu}\mu(\nu)\log(B(\nu))\right)$. For more details, see [CRO+24] Appendix A].

4.6 Quantum Abstract Detecting Systems (QADS)

In this section, we give the definition of Quantum Abstract Detecting Systems (QADS), together with some of their properties, given in [CRR20].

Quantum abstract detecting systems (QADS) were introduced in [CRR20] as a unified framework for the study and design of detection algorithms in a quantum computing setting. Namely, given a black-box oracle for a Boolean function f, the QADS constructs an initial state and an operator that can be used to detect if the function is identically zero or not.

Definition 4.10. A quantum abstract detecting system (QADS) is any (classical deterministic) algorithm that takes, from a set of inputs M, a Boolean function (given by a circuit) $f : \{0,1\}^k \to \{0,1\}$ and outputs a unitary transformation $U = U_f$ on a Hilbert space H whose dimension only depends on k, together with a state $|\varphi_0\rangle \in H$ (that only depends on k too) such that

$$\{x \in \{0,1\}^k \mid f(x) = 1\} = \emptyset \Longrightarrow U|\varphi_0\rangle = |\varphi_0\rangle$$

The transformation U is called *detecting operator* and $|\varphi_0\rangle$ is known as the *initial state*. The input set M usually contains all Boolean functions. However there are some situations in which restrictions on M may apply. The only conditions required of the set M are that it is infinite (i.e., there is no $K \in \mathbb{N}$ such that all Boolean functions f belonging to M have domain $\{0,1\}^k$ with $k \leq K$), and that if $f : \{0,1\}^k \to \{0,1\} \in M$, then the constant zero function with domain $\{0,1\}^k$ also belongs to M. These conditions guarantee that the addressed detecting problem is not trivial.

In [CRR20], it is proved that Grover's algorithm falls under this formalism. Indeed, if O denotes a quantum oracle evaluating $f : \{0,1\}^k \to \{0,1\}$, then the QADS related to Grover's algorithm [Gro96] is the following: Grover's algorithm requires a state space $\left(\mathbb{C}^2\right)^{\otimes k}$ to look for marked elements, i.e., those in $W = \{x \in \{0,1\}^k \mid f(x) = 1\}$, and the initial state $|s\rangle$ which is the superposition of all the elements of the computational basis $\frac{1}{\sqrt{2^k}} \sum_{x=0}^{2^k-1} |x\rangle$. The search iterates two operators that can be effectively constructed, namely:

- Oracle: $U_f(|x\rangle) = (-1)^{f(x)}|x\rangle$, i.e., $U_f = I - 2\sum_{x \in W} |x\rangle\langle x|$.
- Diffusion operator: $U_S = 2|s\rangle\langle s| - I$

The algorithm which constructs $G := U_s U_f$ from f is a QADS because if $W = \emptyset$ then $U_f|s\rangle = |s\rangle$ and

$$G|s\rangle = (2|s\rangle\langle s| - I)|s\rangle = 2|s\rangle\langle s|s\rangle - I|s\rangle = 2|s\rangle - |s\rangle = |s\rangle$$

QADS related to other well-known quantum computing search methodologies, such as quantum walks [Sze04, Por13, San16, Won17] or the quantum abstract search [AKR05], and even other non-search techniques (like Deutsch-Jozsa algorithm [DJ92]) have been considered in [CRR20].

4.6.1 Algorithmic closure of QADS

We can derive new QADS from other existing ones. For instance, the extension, inversion, powers, roots, conjugation, and controlled detecting operator of a QADS is a QADS. Their description as quantum circuits and operators is given in Table 4.1.

Proposition 4.11. Consider a QADS that generates a pair $(|\varphi_0\rangle\ ,\ U) \in H \times \mathscr{U}(H)$ for any given Boolean input f from a set of inputs M, where $\mathscr{U}(H)$ is the group of unitary operators on the Hilbert space H.

1. Algorithms generating the following pairs of initial state/unitary transformation, are also QADS.

   (a) Extension: $(|\varphi_0\rangle |0\rangle^{\otimes l}\ ,\ U \otimes I) \in H' \times \mathscr{U}(H')$, where $H' = H \otimes (\mathbb{C}^2)^l$.

   (b) Inversion: $(|\varphi_0\rangle\ ,\ U^\dagger) \in H \times \mathscr{U}(H)$.

   (c) Powers: $(|\varphi_0\rangle\ ,\ U^{n_f}) \in H \times \mathscr{U}(H)$, for all $n_f \in \mathbb{N}$.

   (d) Roots: $(|\varphi_0\rangle\ ,\ U^{1/n_f}) \in H \times \mathscr{U}(H)$, for all $n_f \in \mathbb{N}$.

   (e) Conjugation: $(T|\varphi_0\rangle\ ,\ TUT^\dagger) \in H \times \mathscr{U}(H)$, for all $T \in \mathscr{U}(H)$. Moreover, conjugation induces an equivalence relation on the set of possible outputs of a QADS for a given input $f \in M$.

   (f) Controlled detecting operator: $(|+\rangle |\varphi_0\rangle\ ,\ CU) \in \mathbb{C}^2 \otimes H \times \mathscr{U}(\mathbb{C}^2 \otimes H)$. where $CU |i\rangle |x\rangle = |i\rangle U^i |x\rangle$.

2. If a second QADS generates pairs $(\left|\varphi_0'\right\rangle\ ,\ U') \in H' \times \mathscr{U}(H')$ for Boolean functions from the same set of inputs M, then:

   (a) A QADS tensor product of QADS can be realized: $(|\varphi_0\rangle \left|\varphi_0'\right\rangle\ ,\ U \otimes U') \in H \otimes H' \times \mathscr{U}(H \otimes H')$.

   (b) If $H' = H$ and $\left|\varphi_0'\right\rangle = |\varphi_0\rangle$, then a product of detecting operators can be considered as a QADS: $(|\varphi_0\rangle\ ,\ U'U) \in H \times \mathscr{U}(H)$.

   (c) The pair of QADS can be doubly controlled according to the following scheme: $(|+\rangle |\varphi_0\rangle \left|\varphi_0'\right\rangle\ ,\ (U \otimes U')_{dc}) \in \mathbb{C}^2 \otimes H \otimes H' \times \mathscr{U}(\mathbb{C}^2 \otimes H \otimes H')$, where

$$(U \otimes U')_{dc} |i\rangle |x\rangle \left|x'\right\rangle = |i\rangle U^i |x\rangle U'^{1-i} \left|x'\right\rangle.$$

Proof. See [CRR20], Appendix B] □

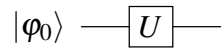
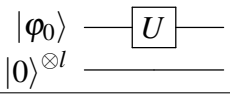
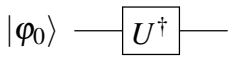
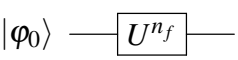
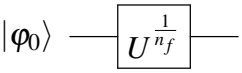
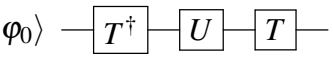
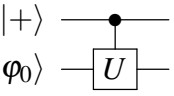
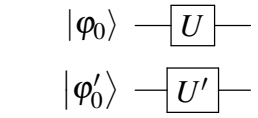
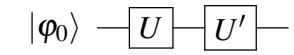
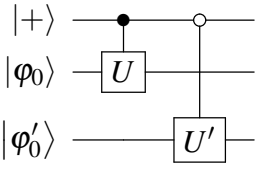
Name	Initial state	Detecting operator	Circuit
QADS	$\|\varphi_0\rangle$	U	
Extension	$\|\varphi_0\rangle \|0\rangle^{\otimes l}$	$U \otimes I$	
Inversion	$\|\varphi_0\rangle$	$U^\dagger$	
Powers	$\|\varphi_0\rangle$	U^{n_f}	
Roots	$\|\varphi_0\rangle$	U^{1/n_f}	
Conjugation	$T\|\varphi_0\rangle$	$TUT^\dagger$	
Controlled	$\|+\rangle \|\varphi_0\rangle$	$CU \|i\rangle \|x\rangle = \|i\rangle U^i \|x\rangle$	
Tensor product	$\|\varphi_0\rangle \|\varphi_0{}'\rangle$	$U \otimes U'$	
Product	$\|\varphi_0\rangle (= \|\varphi_0{}'\rangle)$	$U'U$	
Doubly controlled	$\|+\rangle \|\varphi_0\rangle \|\varphi_0{}'\rangle$	$U_{dc} \|i\rangle \|x\rangle \|x'\rangle$ $= \|i\rangle U^i \|x\rangle U'^{1-i} \|x'\rangle$	

Table 4.1: Transformations in the algorithmic closure of a QADS

4.6.2 Properties of QADS

Here, we define when a QADS is efficiently constructible, which is when both unitary operator and the initial state should be computed in polynomial time in the input size n. As an example, Grover's QADS is efficiently constructible. Also, we define when a QADS has an efficient detection, that is, it should have a detection rate asymptotically independent of the input size.

Definition 4.12. A QADS is called *efficiently constructible* if for any input circuit $f \in M$ of size n, the output pair initial state/unitary transformation can be computed in $O(\text{poly}(n))$ time and, as a consequence, their circuits are of $O(\text{poly}(n))$ width, depth and number of gates.

Definition 4.13. Let $(|\varphi_0\rangle, U = U(f))$ be the output of a QADS on input $f \in M$. Then, for a given $0 < \delta \leq 1$, a function $T : \mathbb{N} \to \mathbb{N}$ is a *δ-quantum detecting time for the QADS* if for all nonzero $f \in M$ of input size k it holds that

$$\frac{\sum_{t=0}^{T(k)} |\langle\varphi_0| U^t |\varphi_0\rangle|^2}{T(k)+1} \leq 1-\delta.$$

For instance, the QADS of Grover search provides efficient constructibility and a $\frac{\sqrt{2}-1}{4\sqrt{2}}-$ detection time of order $O(\sqrt{2^k})$, [Example 13, [CRR20]].

4.6.3 Detection with a QADS

In this subsection, we present an algorithm that is used in a decision procedure to detect the existence of marked elements when we have a QADS i.e., existence of x such that $f(x) = 1$.

Algorithm 4.14 (Detection scheme)**.**
Input: A QADS Q, a Boolean function $f : \{0,1\}^k \to \{0,1\}$ from the set of inputs M of the QADS, and a natural number T.
Procedure:

- Precomputation of the initial state $|\varphi_0\rangle$ and the detecting operator U with Q on input f.
- Computation:
  - Choose t uniformly in the set $\{0,1,\ldots,T\}$
  - Compute $|\varphi_t\rangle = U^t |\varphi_0\rangle$.
- Measurement of $|\varphi_t\rangle$ on an orthonormal basis containing $|\varphi_0\rangle$

Output:

- NO: If the measurement is the initial state $|\varphi_0\rangle$.
- YES: Otherwise.

The detection scheme is readily described by the circuit in Figure 4.5.

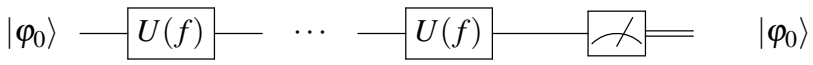


Figure 4.5: Circuit of the detection scheme.

In general, the following result can be proved:

Theorem 4.15. *[CRR20][Main Theorem] The detection scheme of Algorithm 4.14 always provides a correct output on input zero (i.e., when no marked elements exist), and so the probability of error is fully attributed to nonzero inputs. That probability is equal to*

$$\frac{\sum_{t=0}^{T} |\langle\varphi_0| U^t |\varphi_0\rangle|^2}{T+1}$$

Therefore, if a QADS is both efficiently constructible and has a $\delta-$detecting time, then the detection scheme can be run in $O(poly(n))$ precomputation time, and the detection problem can be solved by a one-side error quantum algorithm with error at most $1-\delta$.

Proof. See [CRR20], Appendix E] □

Chapter 5

Combinatorial and Rotational QADS

In this chapter, we introduce new families of QADS, known as combinatorial and rotational, which respectively generalize detecting systems based on single qubit controlled gates and on Grover's algorithm. We study the algorithmic closure of each family, and prove that some of these QADS are equivalent (in the sense of having the same detection rate) to others constructed from tensor product of controlled operators and their square roots. The aim is to improve the detecting techniques used in the effective determination of the commutativity of a finite-dimensional algebra. This is accomplished at the end of the chapter. Incidentally, we also apply the new QADS to the problem of the phase estimation.

First, let us introduce the definition of combinatorial QADS.

5.1 m-Combinatorial QADS

Definition 5.1. If U_f is the detecting operator of a QADS $Q, |\varphi_0\rangle$ is its inital state, and m is a non-negative integer, we define the $m-$combinatorial QADS obtained from Q as the QADS whose initial state is $|0\rangle^{\otimes m} |\varphi_0\rangle$, and whose detecting operator is given by

$$C(m,U_f) := \left(H^{\otimes m} \otimes I\right) c_1 U_f \cdots c_m U_f \left(H^{\otimes m} \otimes I\right),$$

where $c_i U_f$ is the unitary operator that applies U_f to the second register if the i-th qubit of the first register is $|1\rangle$, and applies the identity if that qubit is $|0\rangle$ (i.e. it is the operator U_f controlled by the i-th qubit of the first register).

Observe that, when $m = 1$, we recover the *controlled QADS* of [CRR20] ($7-$th entry in Table 4.1). The following result guarantees that the $m-$combinatorial QADS is indeed a QADS, and that it is efficiently constructible provided the original QADS is.

Proposition 5.2. If we have a QADS Q providing an output $\left(U_f, |\varphi_0\rangle\right)$ on input f, then for all $m \geq 1$, the algorithm that returns the operator depicted in circuit of Figure 5.1, and the state $|0\rangle^{\otimes m} |\varphi_0\rangle$ is also a QADS. What is more, if the original QADS is efficiently constructible, so is the new QADS, for fixed m.

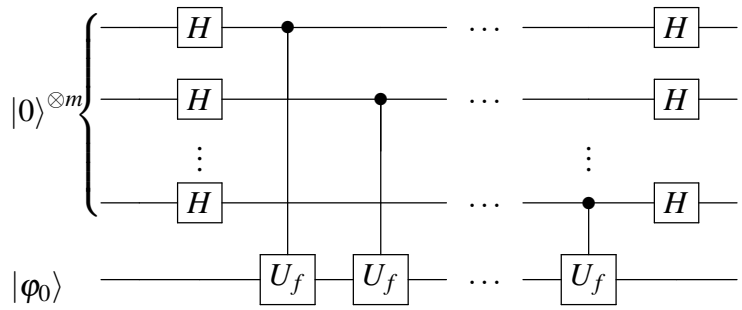


Figure 5.1: Circuit of a m-Combinatorial QADS

Proof of Proposition 5.2. The new algorithm is a QADS because if $f = 0$, then $U_f |\varphi_0\rangle = |\varphi_0\rangle$. Therefore, $c_i U_f |\psi\rangle |\varphi_0\rangle = |\psi\rangle |\varphi_0\rangle$, for all $|\psi\rangle$, and for all i. Consequently,

$$\begin{aligned} C(m,U_f)\,|0\rangle^{\otimes m}\,|\varphi_0\rangle &= \left[\left(H^{\otimes m}\otimes I\right) c_1 U_f \cdots c_m U_f \left(H^{\otimes m}\otimes I\right)\right] |0\rangle^{\otimes m}\,|\varphi_0\rangle \\ &= |0\rangle^{\otimes m}\,|\varphi_0\rangle\,, \end{aligned}$$

which shows that the $m-$combinatorial QADS is actually a QADS.

When the QADS is efficiently contructible, $|\varphi_0\rangle$ can be constructed in polynomial time (on n, the size of f), and the same holds for the initial state $|0\rangle^{\otimes m}|\varphi_0\rangle$, for fixed m. On the other hand, because of [NC11][Section 4.3], any controlled operator $c_i U_f$ can also be constructed in polynomial time because of the constructibility of the QADS. Therefore, the $m-$combinatorial QADS is efficiently constructible, with a cost of order $O(m \cdot \mathrm{poly}(n))$.

□

The reason that justifies the name "combinatorial" for this type of QADS is given in the following result, where the amplitude of the state

$$C(m,U_f)\,|0\rangle^{\otimes m}\,|\varphi_0\rangle\,,$$

related to the state $|0\rangle^{\otimes m}|\varphi_0\rangle$, is given.

Proposition 5.3. The amplitude of the state $C(m,U_f)\,|0\rangle^{\otimes m}\,|\varphi_0\rangle$ related to the basis state $|0\rangle^{\otimes m}|\varphi_0\rangle$, is

$$\frac{1}{2^m}\sum_{k=0}^{m}\binom{m}{k}\langle\varphi_0|\,U_f^k\,|\varphi_0\rangle\,. \tag{5.1}$$

A concrete and complete description of the state

$$C(m,U_f)\,|0\rangle^{\otimes m}\,|\varphi_0\rangle$$

is given in Proposition 5.5.

Proof of Proposition 5.3. Applying $H^{\otimes m} \otimes I$ to the state $|0\rangle^{\otimes m} |\varphi_0\rangle$, we get

$$\frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle |\varphi_0\rangle .$$

Using the controlled versions of the U_f operator, we get

$$|\psi\rangle = \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle U_f^{|x|} |\varphi_0\rangle ,$$

where $|x|$ is the Hamming weight of x, i.e., if x is described by exactly $|x|$ ones and $m-|x|$ zeroes, then the controlled operators $c_i U_f$ will contribute with exactly $|x|$ hits of U_f. Therefore,

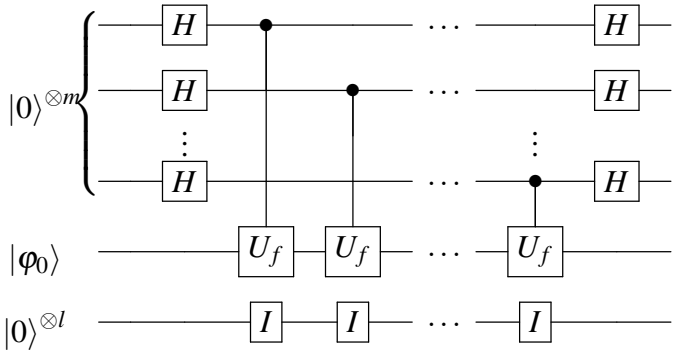
$$\begin{aligned}
\left\langle |0\rangle^{\otimes m} |\varphi_0\rangle \middle| (H^{\otimes m} \otimes I) |\psi\rangle \right\rangle &= \left\langle (H^{\otimes m} \otimes I) |0\rangle^{\otimes m} \middle| |\psi\rangle \right\rangle \\
&= \left\langle \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} |y\rangle |\varphi_0\rangle \middle| \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle U_f^{|x|} |\varphi_0\rangle \right\rangle \\
&= \frac{1}{2^m} \left(\sum_{y=0}^{2^m-1} \langle y| \langle \varphi_0| \right) \left(\sum_{x=0}^{2^m-1} |x\rangle U_f^{|x|} |\varphi_0\rangle \right) \\
&= \frac{1}{2^m} \sum_{k=0}^{m} \binom{m}{k} \langle \varphi_0| U_f^k |\varphi_0\rangle ,
\end{aligned}$$

as desired. □

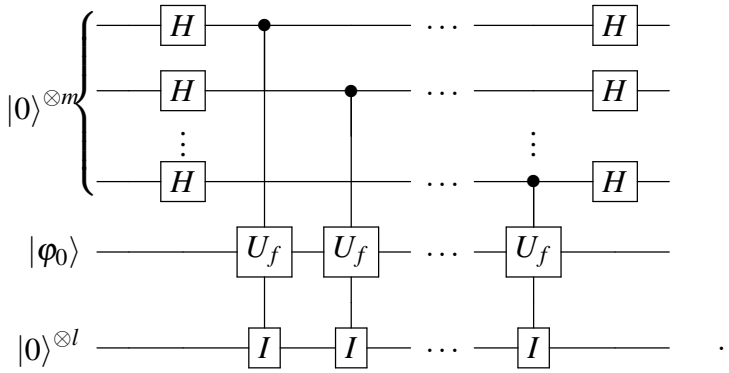
Such an expression can be useful, for instance, for providing algebraic proofs of some of the results related to the algorithmic closure of combinatorial QADS, that we introduce next. The proofs below are first sketched by circuit depiction of the QADS operators. In these results, we determine some procedures which leave the subclass of combinatorial QADS algorithmically closed.

Proposition 5.4. The extension, powers, and roots of an $m-$combinatorial QADS, are also $m-$combinatorial QADS.

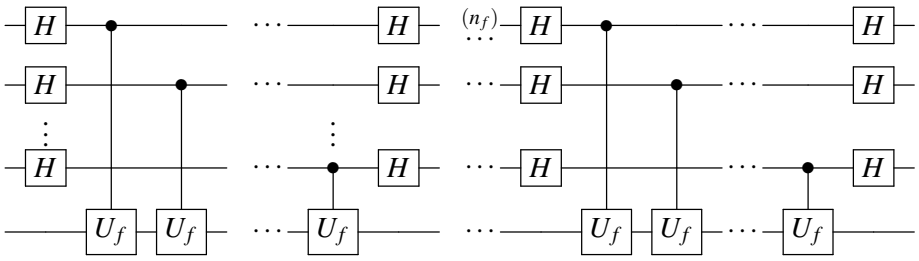
Graphical sketch of proof. 1. Extension: It is straightforward to see that the following circuit



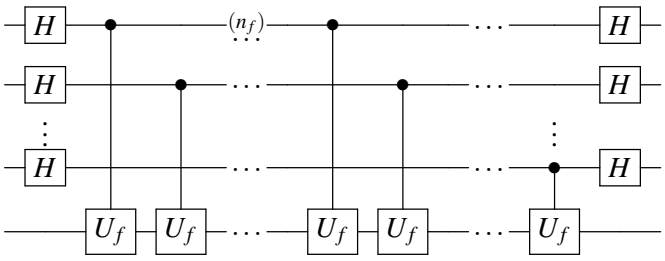
is equivalent to



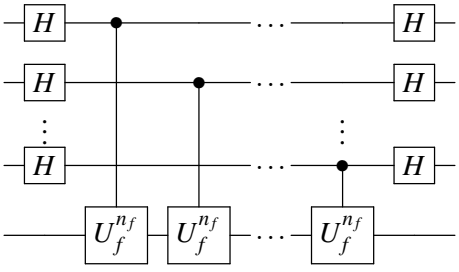
2. Powers: Since $H^2 = I$, and U_f commutes with itself, we have the following equivalencies for n_f copies of the $m-$combinatorial detecting operator:



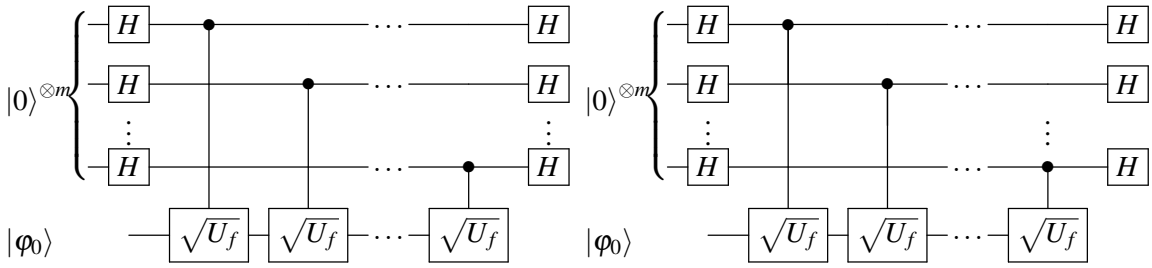
$\equiv$



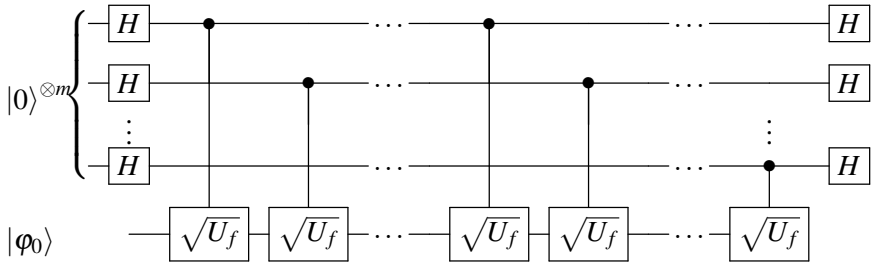
$\equiv$



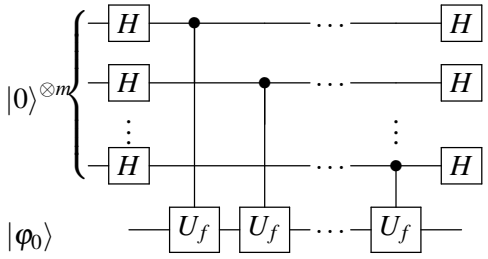
3. Roots: Since $C\left(m,\sqrt{U_f}\right)^2 = C\left(m,\sqrt{U_f}\right)$, then



$\equiv$



$\equiv$



□

Proof of Proposition 5.4. Consider a QADS that, on input f, provides an output $\left(|\varphi_0\rangle\ ,\ U_f\right)$, and let $\left(|0\rangle^{\otimes m}|\varphi_0\rangle\ ,\ C(m,U_f)\right)$ be the output of the corresponding $m-$combinatorial QADS.

1. Extension: Observe that

$$(c_i U_f \otimes I)|x\rangle|\psi\rangle|\xi\rangle = (\alpha|0\rangle|\psi\rangle + \beta|1\rangle U_f|\psi\rangle)|\xi\rangle$$

$$= (\alpha|0\rangle + \beta|1\rangle(U_f \otimes I))|\psi\rangle|\xi\rangle = c_i(U_f \otimes I)|x\rangle|\psi\rangle|\xi\rangle$$

where $|x\rangle = \alpha|0\rangle + \beta|1\rangle$. So, $C(m,U_f)\otimes I = C(m,U_f \otimes I)$.

2. Powers: For any unitary operators U and V

$$cVcU\left|x\right\rangle\left|\varphi\right\rangle = \alpha\left|0\right\rangle\left|\varphi\right\rangle + \beta\left|1\right\rangle VU\left|\varphi\right\rangle = c(VU)\left|x\right\rangle\left|\varphi\right\rangle,$$

where $\left|x\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$. Also, notice that when U and V commute,

$$\begin{aligned} c_2Vc_1U\left|x\right\rangle\left|y\right\rangle\left|\psi\right\rangle &= c_2V(\alpha\left|0\right\rangle\left|y\right\rangle\left|\psi\right\rangle + \beta\left|1\right\rangle\left|y\right\rangle U\left|\psi\right\rangle) \\ &= \alpha\gamma|0\rangle\left|0\right\rangle\left|\psi\right\rangle + \alpha\delta\left|0\right\rangle\left|1\right\rangle V\left|\psi\right\rangle + \beta\gamma|1\rangle\left|0\right\rangle U\left|\psi\right\rangle + \beta\delta\left|1\right\rangle\left|1\right\rangle VU\left|\psi\right\rangle \\ &= \gamma\alpha\left|0\right\rangle\left|0\right\rangle\left|\psi\right\rangle + \gamma\beta\left|1\right\rangle\left|0\right\rangle U\left|\psi\right\rangle + \delta\alpha\left|0\right\rangle\left|1\right\rangle V\left|\psi\right\rangle + \delta\beta\left|1\right\rangle\left|1\right\rangle UV\left|\psi\right\rangle \\ &= c_1U(\gamma|x\rangle\left|0\right\rangle\left|\psi\right\rangle + \delta\left|x\right\rangle\left|1\right\rangle V\left|\psi\right\rangle) \\ &= c_1Uc_2V\left|x\right\rangle\left|y\right\rangle\left|\varphi\right\rangle, \end{aligned}$$

where $\left|x\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle, \left|y\right\rangle = \gamma|0\rangle + \delta\left|1\right\rangle$. Because $H^2 = I$, we have $C(m,U_f)^{n_f} = C(m,U_f^{n_f})$.

3. Roots: Taking in the previous equation $V = U_f^{n_f}$, we have that $U_f = V^{1/n_f}$, and that $C(m,V^{1/n_f})^{n_f} = C(m,V)$, i.e., $C(m,V^{1/n_f}) = C(m,V)^{1/n_f}$. This shows that the n_f-th root of an $m-$combinatorial QADS is also an $m-$combinatorial QADS.

□

Some other operations in the algorithmic closure of QADS might not leave the subclass of combinatorial QADS closed. This is, for instance, the case of the product of two combinatorial QADS when the corresponding detecting operators do not commute.

Proposition 5.5. Given a QADS with output $(\left|\varphi_0\right\rangle, U_f)$, and a natural number m, the state of the corresponding $m-$combinatorial QADS, after one hit of the detecting operator on the initial state, is:

$$\frac{1}{2^m}\sum_{y=0}^{2^m-1}\left|y\right\rangle\left(\sum_{k=0}^{m}\left(\sum_{s=0}^{\lfloor\frac{k}{2}\rfloor}\binom{|y|}{2s}\binom{m-|y|}{k-2s} - \sum_{s=0}^{\lfloor\frac{k-1}{2}\rfloor}\binom{|y|}{2s+1}\binom{m-|y|}{k-2s-1}\right)U_f^k\right)\left|\varphi_0\right\rangle.$$

Proof of Proposition 5.5. From the proof of Proposition 5.3, we get the state

$$\frac{1}{\sqrt{2^m}}\sum_{x=0}^{2^m-1}\left|x\right\rangle U_f^{|x|}\left|\varphi_0\right\rangle$$

after applying $H^{\otimes m}\otimes I$ to the initial state $\left|0\right\rangle^{\otimes m}\left|\varphi_0\right\rangle$, and then the controlled gates on the operators U_f. Next, we apply $H^{\otimes m}\otimes I$ to get

$$\frac{1}{\sqrt{2^m}}\frac{1}{\sqrt{2^m}}\sum_{x=0}^{2^m-1}\sum_{y=0}^{2^m-1}(-1)^{x\cdot y}\left|y\right\rangle U_f^{|x|}\left|\varphi_0\right\rangle = \frac{1}{2^m}\sum_{y=0}^{2^m-1}\left|y\right\rangle\left(\sum_{x=0}^{2^m-1}(-1)^{x\cdot y}U_f^{|x|}\right)\left|\varphi_0\right\rangle.$$

Now, we fix a binary array y with $|y|$ ones and $m-|y|$ zeroes. Assume that another array x of the same length m, contains t ones colliding in positions of the array y with a one, and $|x|-t$ ones colliding in zero entries of the array y (its remaining entries are all zero). Without loss of generality, this can be depicted as:

$$y = (1\ldots\overset{|y|}{\ldots}\ldots 1 \mid 0\ldots\ldots\overset{m-|y|}{\ldots\ldots}\ldots\ldots 0)$$
$$x = (1\ \overset{t}{\ldots}\ 1 \mid 0\ \overset{|y|-t}{\ldots}\ 0 \mid 1\ \overset{|x|-t}{\ldots}\ 1 \mid 0\ \overset{m-|y|-|x|+t}{\ldots}\ 0).$$

The number of arrays x in this situation is

$$\binom{|y|}{t}\binom{m-|y|}{|x|-t}.$$

Now, $x\cdot y = 0$ if and only if t is even. In this case, the possible values of t are of the form $t = 2s$, where $x = 0,1,\ldots,\left\lfloor\frac{|x|}{2}\right\rfloor$. On the other hand, $x\cdot y = 1$ if and only if t is odd, and the possible values of t are of the form $t = 2s+1$, where $x = 0,1,\ldots,\left\lfloor\frac{|x|-1}{2}\right\rfloor$. Summarizing, the number of possible x such that $x\cdot y = 0$ is

$$\sum_{s=0}^{\lfloor\frac{k}{2}\rfloor}\binom{|y|}{2s}\binom{m-|y|}{k-2s},$$

whereas the number of possible x such that $x\cdot y = 1$ is

$$\sum_{s=0}^{\lfloor\frac{k-1}{2}\rfloor}\binom{|y|}{2s+1}\binom{m-|y|}{k-2s-1}.$$

This gives us the desired expression for the final state of the $m-$combinatorial QADS. □

Next, we provide a result relating the detecting times of the combinatorial and the original QADS. Two technical lemmas are introduced first.

Lemma 5.6 ([Pet33]). *Let α be a real number, and let $0<\theta<\frac{\pi}{2}$. If $z_1,\ldots,z_n$ are complex numbers such that*

$$\alpha-\theta \leq \arg z_j \leq \alpha+\theta, \text{ for all } j = 1,\ldots n,$$

then

$$\left|\sum_{j=1}^{n} z_j\right| \geq \cos\theta \sum_{j=1}^{n}\left|z_j\right|.$$

Proof of Lemma 5.6. Since $|z| \geq |\mathrm{Re}(z)| \geq \mathrm{Re}(z)$, for all $z \in \mathbb{C}$, then

$$\begin{aligned}
\left|\sum_{j=1}^{n} z_j\right| &= \left|e^{-i\alpha}\sum_{j=1}^{n} z_j\right| \\
&\geq \left|\mathrm{Re}\left(e^{-i\alpha}\sum_{j=1}^{n} z_j\right)\right| \\
&\geq \mathrm{Re}\left(\sum_{j=1}^{n} |z_j|\left(\cos\left(-\alpha+\arg z_j\right)+i\sin\left(-\alpha+\arg z_j\right)\right)\right) \\
&= \sum_{j=1}^{n} |z_j|\cos\left(-\alpha+\arg z_j\right) \\
&\geq \cos\theta\sum_{j=1}^{n} |z_j|.
\end{aligned}$$

□

Lemma 5.7. *Let m be a positive integer. Then,*

$$\sum_{k=0}^{m}\binom{m}{k}^2 = \binom{2m}{m}.$$

Proof of Lemma 5.7. Let us recall a well-known identity for binomial coefficients, namely, the *Vandermonde's identity*. It states that, for any nonnegative integers r, m, n,

$$\binom{m+n}{r} = \sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k}.$$

Indeed, note that, by the binomial theorem,

$$\begin{aligned}
\sum_{r=0}^{m+n}\binom{m+n}{r}x^r = (1+x)^{m+n} = (1+x)^m(1+x)^n &= \sum_{i=0}^{m}\binom{m}{i}x^i\sum_{r=0}^{n}\binom{n}{j}x^j \\
&= \sum_{i=0}^{m+m}\sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k}x^r.
\end{aligned}$$

In particular, when $n = m$ and $r = m$,

$$\binom{2m}{m} = \binom{m+m}{m} = \sum_{k=0}^{m}\binom{m}{k}\binom{m}{m-k} = \sum_{k=0}^{m}\binom{m}{k}\binom{m}{k} = \sum_{k=0}^{m}\binom{m}{k}^2.$$

□

Proposition 5.8. Let Q be a QADS, and let $\tilde{Q}$ be the corresponding m-combinatorial QADS. Suppose $S:\mathbb{N}\to\mathbb{N}$ is a $\tilde{\delta}$-detecting time for $\tilde{Q}$, and let $z_l := \langle\varphi_0|U_f^l|\varphi_0\rangle$ for any $l\in\mathbb{N}$. Assume that, for all $w\in\mathbb{N}$, there exist $a_w\in\mathbb{R}$, with $\alpha_w\in\left(0,\frac{\pi}{2}\right)$, such that $\frac{(1-\tilde{\delta})2^{2m}}{\cos^2\alpha_w\binom{2m}{m}}\leq 1-\delta$, with $\delta>0$, and such that, for all $l=0,\ldots,m\cdot S(k)$, $\arg(z_l)\in[a_w-\alpha_w,a_w+\alpha_w]$. Then, $T:\mathbb{N}\to\mathbb{N}$ given by $T(w)=m\cdot S(w)$, for all $w\in\mathbb{N}$, is $\frac{\delta}{2m}$−detecting time for Q.

Proof of Proposition 5.8. Let $w\in\mathbb{N}$ be fixed, and denote $T=T(w)$, $S=S(w)=m\cdot T(w)$, $a=a_w$, and $\alpha=\alpha_w$. By Proposition 5.4, we have that, for all $s\in\mathbb{N}$, $C(m,U_f)^s=C(m,U_f^s)$, and by Proposition 5.3, we know the amplitude of the state $C(m,U_f)|0\rangle^{\otimes m}|\varphi_0\rangle$, so

$$\begin{aligned}\frac{\sum_{s=0}^{S}\left|\langle|0^{\otimes m}\rangle|\varphi_0\rangle|C(m,U_f)^s||0^{\otimes m}\rangle|\varphi_0\rangle\rangle\right|^2}{S+1} &= \frac{\sum_{s=0}^{S}\left|\sum_{k=0}^{m}\binom{m}{k}\langle\varphi_0|U_f^{ks}|\varphi_0\rangle\right|^2}{2^{2m}(S+1)}\\ &= \frac{\sum_{s=0}^{S}\left|\sum_{k=0}^{m}\binom{m}{k}z_{ks}\right|^2}{2^{2m}(S+1)}\\ &\leq 1-\tilde{\delta}.\end{aligned}$$

Here, $z_{ks}:=\langle\varphi_0|U_f^{ks}|\varphi_0\rangle$, for all $k=0,\ldots,m$, and all $s=0,\ldots,S$. Let us define the following sets:

$$\begin{aligned}A_0 &:= \{0\}\\ \overline{A_0} &:= \{1,\ldots,T\}=\{0,\ldots,T\}\setminus\{A_0\}\\ A_k &:= \{k,2k,\ldots,Sk\}\text{, for all } 1\leq k\leq m\\ \overline{A_k} &:= \{0,\ldots,T\}\setminus\{A_k\}.\end{aligned}$$

Now, by Lemma 5.7,

$$\frac{\sum_{t=0}^{T}\left|\langle\varphi_0|U_f^t|\varphi_0\rangle\right|^2}{T+1}=\frac{\sum_{t=0}^{T}|z_t|^2}{T+1}=\frac{\sum_{k=0}^{m}\binom{m}{k}^2}{\binom{2m}{m}}\cdot\frac{\sum_{t=0}^{T}|z_t|^2}{T+1}=\frac{\sum_{k=0}^{m}\binom{m}{k}^2\left(\sum_{t\in A_k}|z_t|^2+\sum_{t\in\overline{A_k}}|z_t|^2\right)}{\binom{2m}{m}(T+1)}.$$

But, since $|z_t| \leq 1, |A_k| = T - S$, for $k > 0$, and $|z_0|^2 = 1$, we find

$$\begin{aligned}
\frac{\sum_{k=0}^{m} \binom{m}{k}^2 \left(\sum_{t \in A_k} |z_t|^2 + \sum_{t \in \overline{A_k}} |z_t|^2 \right)}{\binom{2m}{m}(T+1)} &\leq \frac{\left(|z_0|^2 + T\right) + \sum_{k=1}^{m} \binom{m}{k}^2 \left(\sum_{s=0}^{S} |z_{ks}|^2 + (T-S) \right)}{\binom{2m}{m}(T+1)} \\
&= \frac{\sum_{k=0}^{m} \binom{m}{k}^2 \left(\sum_{s=0}^{S} |z_{ks}|^2 + (T-S) \right)}{\binom{2m}{m}(T+1)} \\
&= \frac{\sum_{s=0}^{S} \sum_{k=0}^{m} \left(\binom{m}{k} |z_{ks}| \right)^2}{\binom{2m}{m}(T+1)} + \frac{\sum_{k=0}^{m} \binom{m}{k}^2}{\binom{2m}{m}} \cdot \frac{T-S}{T+1} \\
&\leq \frac{\sum_{s=0}^{S} \left(\sum_{k=0}^{m} \left| \binom{m}{k} z_{ks} \right| \right)^2}{\binom{2m}{m}(T+1)} + \frac{T-S}{T+1},
\end{aligned}$$

because $\binom{m}{k}|z_{ks}| = \left|\binom{m}{k} z_{ks}\right| \geq 0$. Therefore, by Lemma 5.6, we have

$$\begin{aligned}
\frac{\sum_{s=0}^{S} \left(\sum_{k=0}^{m} \left| \binom{m}{k} z_{ks} \right| \right)^2}{\binom{2m}{m}(T+1)} + \frac{T-S}{T+1} &\leq \frac{\sum_{s=0}^{S} \left| \sum_{k=0}^{m} \binom{m}{k} z_{ks} \right|^2}{\cos^2 \alpha \binom{2m}{m}(T+1)} \cdot \frac{2^{2m}(S+1)}{2^{2m}(S+1)} + \frac{T-S}{T+1} \\
&\leq (1-\tilde{\delta}) \cdot \frac{2^{2m}(S+1)}{\cos^2 \alpha \binom{2m}{m}(T+1)} + \frac{T-S}{T+1} \\
&\leq (1-\delta) \cdot \frac{S+1}{T+1} + \frac{T-S}{T+1} \leq 1 - \frac{\delta}{2m},
\end{aligned}$$

since $\frac{S+1}{T+1} \geq \frac{1}{2m}$.

□

The conditions on the previous result are satisfied, for instance, for some families of QADS known as rotational, that we introduce in the next section (see Corollary 5.16).

5.2 Rotational QADS

In some well-studied searching procedures, the iterating operator acts only on a small dimensional invariant subspace, leaving the remaining directions unchanged. This is the case, for instance, of the operator of Szegedy's quantum walk with queries on the complete graph [San16], which acts on an invariant three-dimensional space when only one vertex is marked, and on an invariant four-dimensional space when multiple marked vertices are considered. Of course, this is also the case of the operator of Grover's search, which acts as a rotation in a two-dimensional invariant subspace, and leaves the orthogonal directions unaltered [Gro96]. In this section, we

consider QADS in which the detecting operator U_f behaves in this way, acting as a rotation in a two-dimensional invariant subspace, with an operator described by a matrix in $\mathrm{SO}(2)$. As in the case of the combinatorial QADS, we study their properties, such as an explicit expression of the final amplitude, and their algorithmic closure. We also consider combinatorial QADS derived from rotational QADS, concluding some interesting equivalences.

The definition of a rotational QADS is as follows.

Definition 5.9. If U_f is the detecting operator of a QADS Q with initial state $|\varphi_0\rangle$, we shall say that it is a rotational QADS if there exist $\alpha \in [0, 2\pi)$, orthonormal states $|\varphi_1\rangle, |\varphi_2\rangle$, and $\beta_1, \beta_2 \in \mathbb{R}$, such that

1. $|\varphi_0\rangle = \beta_1 |\varphi_1\rangle + \beta_2 |\varphi_2\rangle$
2. $U_f |\varphi_1\rangle = \cos\alpha \ |\varphi_1\rangle + \sin\alpha \ |\varphi_2\rangle$
3. $U_f |\varphi_2\rangle = -\sin\alpha \ |\varphi_1\rangle + \cos\alpha \ |\varphi_2\rangle$.

As said before, the QADS associated to Grover's search is a rotational QADS. The detecting operator U_f of a rotational QADS can be straightforwardly described by a matrix

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \in SO(2),$$

since the coordinate matrix of U_f with respect to an orthonormal basis whose first two elements are $|\varphi_1\rangle, |\varphi_2\rangle$ is

$$\left(\begin{array}{cc|c} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ \hline 0 & 0 & I_{n-2} \end{array}\right).$$

In the following result, we obtain the amplitude of the state $U_f |\varphi_0\rangle$, and a generalised version for QADS that can be described by an arbitrary matrix in the orthogonal group $O(n)$ (Proposition 5.11).

Proposition 5.10. Given a rotational QADS with output $(|\varphi_0\rangle, U_f)$, the state after k hits of the detecting operator on the initial state is

$$U_f^k |\varphi_0\rangle = (\beta_1 \cos k\alpha - \beta_2 \sin k\alpha) |\varphi_1\rangle + (\beta_1 \sin k\alpha + \beta_2 \cos k\alpha) |\varphi_2\rangle.$$

In particular, the amplitude of such a final state, related to the initial state $|\varphi_0\rangle$, is $\cos k\alpha$.

Proof of Proposition 5.10. In terms of $|\varphi_1\rangle, |\varphi_2\rangle$, for all $k \geq 0$, the action of U_f^k on $|\varphi_0\rangle$ is given by the matrix

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}^k = \begin{pmatrix} \cos k\alpha & -\sin k\alpha \\ \sin k\alpha & \cos k\alpha \end{pmatrix},$$

(the $n-2$ invariant directions have been omitted). The final state has coordinates related to $|\varphi_1\rangle, |\varphi_2\rangle$ given by the array

$$\begin{pmatrix} \cos k\alpha & -\sin k\alpha \\ \sin k\alpha & \cos k\alpha \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \beta_1 \cos k\alpha - \beta_2 \sin k\alpha \\ \beta_1 \sin k\alpha + \beta_2 \cos k\alpha, \end{pmatrix}$$

as desired.

On the other hand,

$$\begin{aligned} \langle \varphi_0 | U_f^k | \varphi_0 \rangle &= \left(\overline{\beta_1}, \overline{\beta_2} \right) \begin{pmatrix} \beta_1 \cos k\alpha - \beta_2 \sin k\alpha \\ \beta_1 \sin k\alpha + \beta_2 \cos k\alpha \end{pmatrix} \\ &= |\beta_1|^2 \cos k\alpha + \left(\overline{\beta_2}\beta_1 - \overline{\beta_1}\beta_2 \right) \sin k\alpha + |\beta_2|^2 \cos k\alpha \\ &= \cos k\alpha. \end{aligned}$$

□

Proposition 5.11. Let Q be a QADS such that any input function U_f can be described by an orthogonal matrix with respect to a suitable orthonormal basis. Then, there exist an orthonormal basis $\{|\varphi_1^1\rangle, |\varphi_2^1\rangle \ldots, |\varphi_1^l\rangle, |\varphi_2^l\rangle, |\varphi_{2l+1}\rangle, \ldots, |\varphi_n\rangle\}$, angles $\theta_1, \ldots, \theta_l \in [0, 2\pi)$, and real scalars $\beta_1^2, \beta_1^2, \ldots, \beta_1^l, \beta_2^l, \beta_{2l+1}, \ldots, \beta_n$, such that the final state $U_f |\varphi_0\rangle$ is

$$\begin{aligned} U_f^k |\varphi_0\rangle = &\sum_{i=1}^{l} \left((\beta_1^i \cos k\theta_i - \beta_2^i \sin k\theta_i) |\varphi_1^i\rangle + (\beta_1^i \sin k\theta_i + \beta_2^i \cos k\theta_i) |\varphi_2^i\rangle \right) \\ &+ \sum_{i=2l+1}^{n-1} \beta_i |\varphi_i\rangle + (\pm 1)^k \beta_n |\varphi_n\rangle . \end{aligned}$$

As a consequence, the amplitude of such a final state, related to the initial state $|\varphi_0\rangle$, is

$$\sum_{i=1}^{l} ((|\beta_1^i|^2 + |\beta_2^i|^2) \cos k\theta_i) + \sum_{i=2l+1}^{n-1} |\beta_i|^2 + (\pm 1)^k |\beta_n|.$$

Proof of Proposition 5.11. Since U_f admits a coordinate matrix in the orthogonal group $O(n)$, there must exist an orthonormal basis $\{|\varphi_1^1\rangle, |\varphi_2^1\rangle \ldots, |\varphi_1^l\rangle, |\varphi_2^l\rangle, |\varphi_{2l+1}\rangle, \ldots, |\varphi_n\rangle\}$ such that the coordinate matrix of U_f with respect to such a basis is the block diagonal matrix

$$\begin{pmatrix} R_{\theta_1} & & & & & & \\ & R_{\theta_2} & & & & & \\ & & \ddots & & & & \\ & & & R_{\theta_k} & & & \\ & & & & I_{n-2l-1} & & \\ & & & & & \pm 1 \end{pmatrix},$$

for some angles $\theta_1,\dots,\theta_l \in [0,2\pi)$, where $R_{\theta_i} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix}$, I_{n-2l-1} is the identity matrix, and the sign of the last diagonal entry depends on whether U_f is a rotation or a reflection. The initial state can be written as a linear combination of the basis, in the following way:

$$|\varphi_0\rangle = \sum_{i=1}^{l}(\beta_1^i|\varphi_1^i\rangle + \beta_2^i|\varphi_2^i\rangle) + \sum_{i=2l+1}^{n}\beta_i|\varphi_i\rangle,$$

for real complex coordinates $\beta_1^2,\beta_1^2,\dots,\beta_1^l,\beta_2^l,\beta_{2l+1},\dots,\beta_n$. Straightforward application of the same ideas of the previous proof yields the desired expression for $U_f^k|\varphi_0\rangle$, and $\langle\varphi_0|U_f^k|\varphi_0\rangle$. □

Analogously as in the case of combinatorial QADS, we consider different procedures that allow to derive new rotational QADS from others.

Proposition 5.12. The powers, roots, and inversion of a rotational QADS are also rotational QADS. Also, if two rotational QADS share the same initial state, then their product is also a rotational QADS.

Proof of Proposition 5.12.

1. Powers: The action of $U_f^{n_f}$ on $|\varphi_0\rangle$ is given by the matrix

$$\begin{pmatrix} \cos n_f\alpha & -\sin n_f\alpha \\ \sin n_f\alpha & \cos n_f\alpha \end{pmatrix}$$

(again, the $n-2$ invariant directions have been omitted).

2. Roots: The action of $U_f^{\frac{1}{n_f}}$ on $|\varphi_0\rangle$ is given by the matrix

$$\begin{pmatrix} \cos\left(\frac{\alpha}{n_f}\right) & -\sin\left(\frac{\alpha}{n_f}\right) \\ \sin\left(\frac{\alpha}{n_f}\right) & \cos\left(\frac{\alpha}{n_f}\right) \end{pmatrix}.$$

3. Inversion: The action of $U_f^\dagger$ on $|\varphi_0\rangle$ is given by the matrix

$$\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} = \begin{pmatrix} \cos -\alpha & -\sin -\alpha \\ \sin -\alpha & \cos -\alpha \end{pmatrix}.$$

Note that, in this case $U^\dagger = U^{-1}$.

4. Product: The action of $U_f U_f'$ on $|\varphi_0\rangle$ is given by the matrix

$$\begin{pmatrix} \cos(\alpha+\alpha') & -\sin(\alpha+\alpha') \\ \sin(\alpha+\alpha') & \cos(\alpha+\alpha') \end{pmatrix}.$$

Note that, in general, rotation matrices do not commute under multiplication. However, if both rotations are taken with respect to the same initial state, then they do commute.

□

Like in the case of combinatorial QADS, some other operations in the algorithmic closure of QADS might not leave the subclass of rotational QADS closed. This is for instance the case of the extension of a rotational QADS, or the product of two rotational QADS when they do not share the same initial state.

Proposition 5.13. Let m be a natural number, then

$$\sum_{k=0}^{m} \binom{m}{k} \cos k\alpha = 2^m \left(\cos\left(\frac{\alpha}{2}\right)\right)^m \cos\left(\frac{\alpha}{2}m\right). \tag{5.2}$$

Proof of Lemma 5.13. We know that $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$. So,

$$\begin{aligned}\sum_{k=0}^{m} \binom{m}{k} \cos k\alpha &= \sum_{k=0}^{m} \binom{m}{k} \frac{e^{ik\alpha}+e^{-ik\alpha}}{2} \\ &= \frac{1}{2}\left[\sum_{k}^{m} \binom{m}{k} \left(e^{i\alpha}\right)^k + \sum_{k}^{m} \binom{m}{k} \left(e^{-i\alpha}\right)^k\right].\end{aligned}$$

Now, by the Binomial Theorem, we have that

$$\sum_{k=0}^{m} \binom{m}{k} \left(e^{i\alpha}\right)^k + \sum_{k}^{m} \binom{m}{k} \left(e^{-i\alpha}\right)^k = \left[\left(1+e^{i\alpha}\right)^m + \left(1+e^{-i\alpha}\right)^m\right].$$

Therefore,

$$\begin{aligned}\sum_{k=0}^{m} \binom{m}{k} \cos k\alpha &= \frac{1}{2}\left[\left(1+e^{i\alpha}\right)^m + \left(1+e^{-i\alpha}\right)^m\right] \\ &= \frac{1}{2}\left[e^{i\alpha m}\left(1+e^{-i\alpha}\right)^m + \left(1+e^{-i\alpha}\right)^m\right] \\ &= \frac{1}{2}\left[e^{i\frac{\alpha}{2}m}\left(1+e^{-i\alpha}\right)^m \left(1+e^{i\alpha m}\right) e^{-i\frac{\alpha}{2}m}\right] \\ &= \frac{1}{2}\left[\left(e^{\frac{-i\alpha}{2}}+e^{\frac{i\alpha}{2}}\right)^m \left(e^{\frac{i\alpha}{2}m}+e^{\frac{-i\alpha}{2}m}\right)\right] \\ &= 2^m \left(\cos\left(\frac{\alpha}{2}\right)\right)^m \cos\left(\frac{\alpha}{2}m\right).\end{aligned}$$

□

Corollary 5.14. *Under the hypothesis of Proposition 5.11, the amplitude of the final state of the corresponding $m-$combinatorial QADS, related to the initial state $|0\rangle^m|\varphi_0\rangle$, is*

$$\sum_{i=1}^{l}\left((|\beta_1^i|^2+|\beta_2^i|^2)\cos\left(\frac{\theta_i}{2}\right)^m \cos\left(\frac{\theta_i m}{2}\right)\right) + \sum_{i=2l+1}^{n-1} |\beta_i|^2 + \delta_{+*}|\beta_n|,$$

where $* \in \{+,-\}$, depending on whether U_f is a rotation or a reflection.

Next, we want to consider the $m-$combinatorial QADS of a rotational QADS. In particular, we study the amplitude of the final state, related to the initial state, which is connected to the detection rate when a single hit of the detecting operator is used. As a consequence of Proposition 5.13, we conclude some interesting equivalences of detecting operators from different QADS in the algorithmic closure, related to the square root QADS.

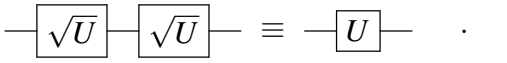
Theorem 5.15. *If Q is a rotation QADS, $m \in \mathbb{Z}^+$, and we consider the corresponding $m-$ combinatorial QADS, then the amplitude of the initial state after one hit of the detecting operator, i.e., of $C(m,U_f)\,|0\rangle^m\,|\varphi_0\rangle$, related to the initial state $|0\rangle^m\,|\varphi_0\rangle$, is*

$$\frac{1}{2^m}\sum_{k=0}^{m}\binom{m}{k}\cos k\alpha = \left(\cos\left(\frac{\alpha}{2}\right)\right)^m \cos\left(\frac{\alpha}{2}m\right). \tag{5.3}$$

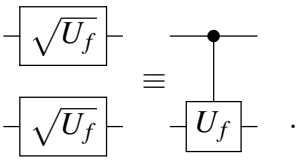
Proof of Theorem 5.15. Equation (5.3) is a direct consequence of Propositions 5.3, 5.10, and 5.13. □

On the other hand, note that:

- Since the QADS is rotational, we have that $\langle\varphi_0|\sqrt{U_f}\,|\varphi_0\rangle = \cos\left(\frac{\alpha}{2}\right)$ (applying the same idea of Proposition 5.10).
- For a tensor QADS, we directly have $\langle\varphi_0|\langle\psi_0|U_f\otimes V_f\,|\varphi_0\rangle\,|\psi_0\rangle = \langle\varphi_0|U_f\,|\varphi_0\rangle\langle\psi_0|V_f\,|\psi_0\rangle$.
- For a product QADS, in terms of equality of the detecting operators, we have



- Since the QADS is rotational, in terms of one hit detection rate, we have



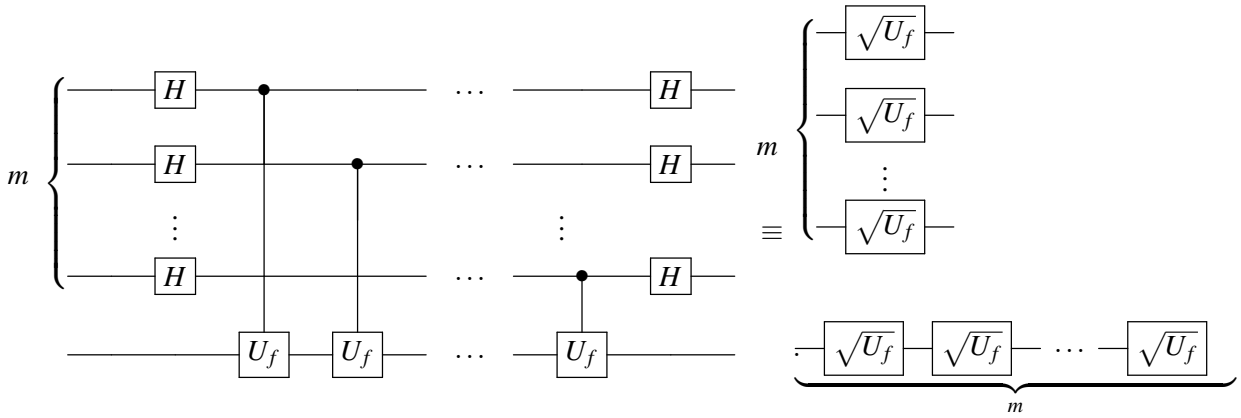
This is because

$$\begin{aligned}\langle\psi_0|\langle\psi_0|\sqrt{U_f}\otimes\sqrt{U_f}\,|\psi_0\rangle\,|\psi_0\rangle &= \left(\langle\psi_0|\sqrt{U_f}\,|\psi_0\rangle\right)^2\\ &= \cos\left(\frac{\alpha}{2}\right)^2\\ &= \frac{1+\cos\alpha}{2}\\ &= \langle\varphi_{0_c}|cU_f\,|\varphi_{0_c}\rangle\end{aligned}$$

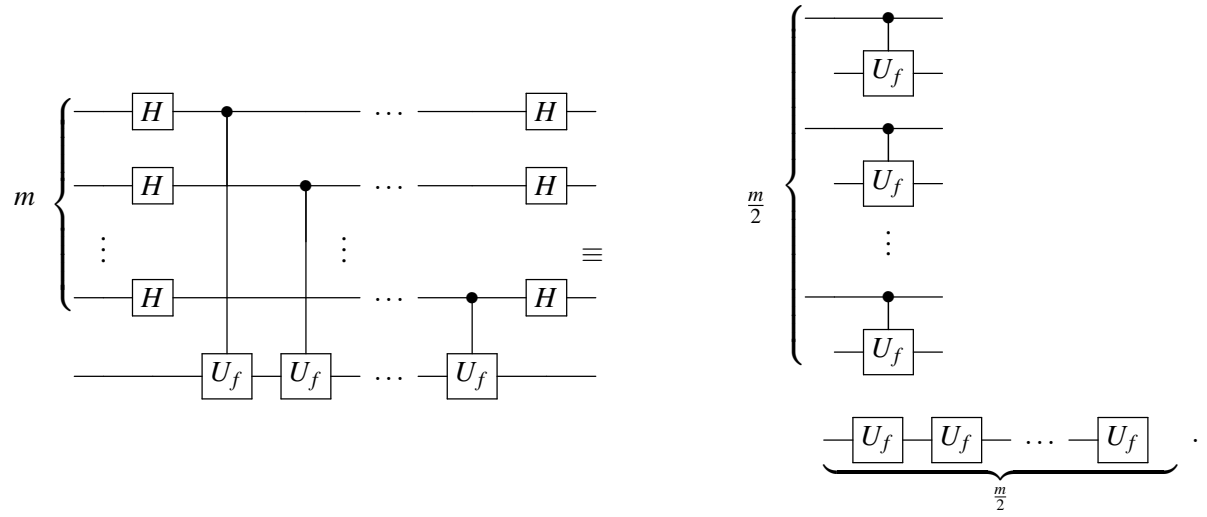
(proof of [CRR20][Proposition 2, item 3]).

Remark. Note that the last equivalence of the proof holds because the QADS is rotational. In general, such an equivalence is not true. For instance, taking U_f as the NOT gate.

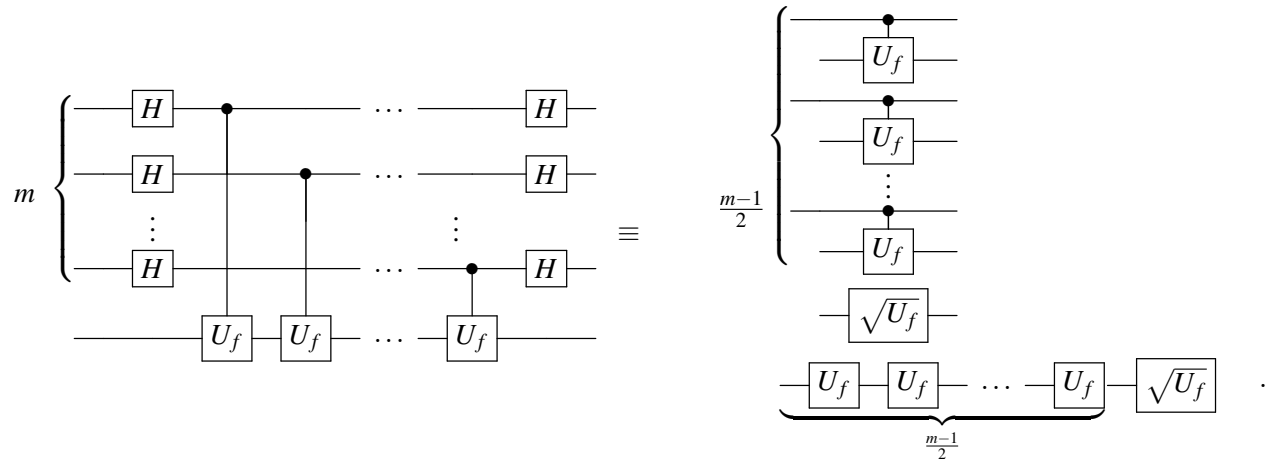
As a consequence, the $m-$combinatorial QADS of a rotational QADS is equivalent, in terms of detection rate, when one single hit of the detecting operator is taken, to the tensor product of m copies of its square root QADS, tensored with the $m-th$ power of its square root QADS:



In particular, when m is even, it is equivalent to the tensor product of $\frac{m}{2}$ copies of its controlled QADS, tensored with its $\frac{m}{2}-$th power:



On the other hand, when m is odd, it is equivalent to the tensor product of $\frac{m-1}{2}$ copies of its controlled QADS plus one copy of its square roots, tensored with a product of exactly the same operators:



Let us finish this section with the previously mentioned result on the detecting time of a family of combinatorial from rotational QADS.

Corollary 5.16. *Let Q be a rotational QADS with a rotation angle smaller than θ_w, on entries f of size w, and let $\tilde{Q}$ be the corresponding m-combinatorial QADS. Suppose $S:\mathbb{N}\to\mathbb{N}$ is a $\tilde{\delta}$-detecting time for $\tilde{Q}$, such that $m < min\left\{\frac{1}{4(1-\tilde{\delta})^2}, \frac{\pi}{2\Delta}\right\}$, where $\Delta \geq \theta_w S(w)$, for all $w\in\mathbb{N}$. Then, $T:\mathbb{N}\to\mathbb{N}$ given by $T(w) = m\cdot S(w)$, for all $w\in\mathbb{N}$, is $\frac{\delta}{2m}-$detecting time for Q, where $\delta = \frac{1-2\sqrt{m}(1-\tilde{\delta})}{2} > 0$.*

Proof of Corollary 5.16. For all $w\in\mathbb{N}$, let us consider any possible input f of size w. For all $0\leq l\leq T(w) = m\cdot S(w)$, we have that $z_l = \cos(l\theta_w) > 0$, because $0\leq l\theta_w \leq m\cdot S(w)\theta_w \leq m\Delta < \frac{\pi}{2}$. Consequently, for all $0\leq l\leq T(w)$, $\arg(z_l)\in[0-\alpha_w, 0+\alpha_w]$, with α_w as close to zero as desired. In particular, we can take α_w such that $\frac{(1-\tilde{\delta})2^{2m}}{\cos^2\alpha_w\binom{2m}{m}} \leq 1-\delta$, because

$$1-\delta > 1-2\delta = 2\sqrt{m}(1-\tilde{\delta}) \geq \frac{2^{2m}}{\binom{2m}{m}}\cdot(1-\tilde{\delta}),$$

and $\cos^2\alpha_w$ can be made as close as needed to 1. The result now follows from Proposition 5.8. □

5.3 Application: Decision on Eigenvalues

Although the QADS methodology was initially introduced as a common framework to deal with the detection problem, it can also be adapted to other problems. Consider, for instance, the situation in which we are given a quantum state $|\varphi_0\rangle$, and a unitary operator U, under the promise that $|\varphi_0\rangle$ is one of its eigenvectors, and we want to check whether the associated eigenvalue is $e^{i\alpha}$ or not. Namely,

Input: A real value α, a quantum state $|\varphi_0\rangle$, $U_\beta\in\{U_\gamma\}_{\gamma\in[0,2\pi)}$, such that $U_\beta|\varphi_0\rangle = e^{i\beta}|\varphi_0\rangle$.
Problem: Decide whether $\beta=\alpha$ or not (up to a certain prefixed accuracy, i.e., if $|\beta-\alpha|<\varepsilon$).

If α and $|\varphi_0\rangle$ are efficiently computable, then this problem can be approached with an efficiently constructible $m-$combinatorial QADS. In fact, let us consider the unitary transformation $V = e^{-i\alpha}U$. Then

$$V|\varphi_0\rangle = e^{-i\alpha}U_\beta|\varphi_0\rangle = e^{i(\beta-\alpha)}|\varphi_0\rangle = |\varphi_0\rangle,$$

where the last equality holds if, and only if, $\alpha=\beta$.

Now, from the results of Section 5.1, we now that the projection of the final state $C(m,V)\left|0\right\rangle^m\left|\varphi_0\right\rangle$ on the $m-$combinatorial QADS initial state $\left|0\right\rangle^m\left|\varphi_0\right\rangle$ is

$$\frac{1}{2^m}\sum_{k=0}^{m}\binom{m}{k}\left\langle\varphi_0\right|V^k\left|\varphi_0\right\rangle=\frac{1}{2^m}\sum_{k=0}^{m}\binom{m}{k}\left(e^{i(\beta-\alpha)}\right)^k=\left(\frac{1+e^{i(\beta-\alpha)}}{2}\right)^m.$$

Thus, the probability of measuring $\left|0\right\rangle^m\left|\varphi_0\right\rangle$ is

$$\begin{aligned}\left|\left(\frac{1+e^{i(\beta-\alpha)}}{2}\right)^m\right|^2&=\left|\left(\frac{1+e^{i(\beta-\alpha)}}{2}\right)^2\right|^m\\&=\left(\frac{\left|1+\cos(\beta-\alpha)+i\sin(\beta-\alpha)\right|^2}{4}\right)^m\\&=\left(\frac{1+2\cos(\beta-\alpha)+\cos^2(\beta-\alpha)+\sin^2(\beta-\alpha)}{4}\right)^m\\&=\left(\frac{2+2\cos(\beta-\alpha)}{4}\right)^m\\&=\left(\frac{1+\cos(\beta-\alpha)}{2}\right)^m\\&=\left(\cos\left(\frac{\beta-\alpha}{2}\right)\right)^{2m}.\end{aligned}$$

Therefore, we can think of the following procedure to decide whether $\alpha=\beta$.

Algorithm 5.17 (For the eigenvalue decision problem)**.**
Input: A real value α, a quantum state $\left|\varphi_0\right\rangle$, $U_\beta\in\{U_\gamma\}_{\gamma\in[0,2\pi)}$, such that $U_\beta\left|\varphi_0\right\rangle=e^{i\beta}\left|\varphi_0\right\rangle$.
Procedure:

- Precomputation of the initial state $\left|\varphi_0\right\rangle$, the unitary operator $V=e^{-i\alpha}U$, and the output of the corresponding $m-$combinatorial QADS $(C(m,V),\left|0\right\rangle^m\left|\varphi_0\right\rangle)$, for a chosen m.
- Computation:
  - Compute $\left|\varphi\right\rangle=C(m,V)\left|0\right\rangle^m\left|\varphi_0\right\rangle$.
- Measurement of $\left|\varphi\right\rangle$ on an orthonormal basis containing $\left|\varphi_0\right\rangle$.

Output:

- YES: If the measurement is the initial state $\left|\varphi_0\right\rangle$.
- NO: Otherwise.

The observations above prove the following result.

Theorem 5.18. *Algorithm 5.17 always provides a correct output when $\beta \neq \alpha$, and so the probability of error is fully attributed to the case $\beta = \alpha$. Namely, such a probability is equal to*

$$\theta = \left(\cos\left(\frac{\beta - \alpha}{2}\right)\right)^{2m}.$$

Therefore, if the QADS is efficiently constructible, then the eigenvalue decision problem can be solved in $O(poly(n))$ precomputation time of a one-side error quantum algorithm with error at most θ, which decreases exponentially with m. The probability of success of the algorithm is $1 - \theta$.

5.4 Application: Phase estimation

5.4.1 Generalized Hadamard Test

Another application of the QADS methodology is phase estimation. Consider again that we are given a quantum state $|\varphi_0\rangle$, and a unitary operator U, under the promise that $|\varphi_0\rangle$ is one of its eigenvectors with associated eigenvalue is $e^{i\alpha}$. The aim is to estimate α.

Input: A quantum state $|\varphi_0\rangle$, $U_\beta \in \{U_\gamma\}_{\gamma\in[0,\pi)}$, such that $U_\beta |\varphi_0\rangle = e^{i\beta} |\varphi_0\rangle$, and a natural number SHOTS.
Problem: PROBLEM: An approximation α of β, using at most SHOTS executions of prefixed quantum circuit.

Of course, this problem can be solved with the well-known quantum phase estimation (QPE) Algorithm 4.7. However, as pointed out in [OTT19] "the size and shallowness of the QPE circuit is important since, in the absence of error correction or error mitigation, one expects entropy build-up during computation." In fact, it has been shown in [MOS+19] that, when implemented on current quantum hardware, the accuracy of the QPE algorithm is "severely constrained by NISQ's physical characteristics such as coherence time and error rates." For these reasons, some authors have proposed replacing the QPE algorithm with less demanding methods, that make implementing quantum algorithms that rely on it easier in practice (see, for instance, [AR20, DJCS21, Nak20, Ral21, SUR+20, Wie19]).

A simpler algorithm, that sometimes is used for the phase estimation problem instead of QPE, is the Hadamard test (Example 3.19). Which, in fact, consists in the quantum circuit of the combinatorial QADS with $m = 1$, with a final measurement of the controlling qubit [ADZ93].

From Example 3.19, we get that the probability of measuring the quantum state $|0\rangle|\varphi_0\rangle$ is

$$\begin{aligned}\left|\left|\frac{1}{2}\left(1+e^{i\beta}\right)|0\rangle+|1\rangle\left(1-e^{i\beta}\right)\right|\right|^2 &= \left|\left|\frac{1}{2}\left((1+\cos\beta)^2+\sin^2\beta\right)\right|\right| \\ &= \frac{1}{4}\left(2\left(1+\cos\beta\right)\right) \\ &= \frac{1}{4}\left(2\left(2\cos^2\left(\frac{\beta}{2}\right)\right)\right) \\ &= \cos^2\left(\frac{\beta}{2}\right).\end{aligned}$$

Running the test SHOTS times provides an approximation P of such a probability, from which β can be estimated. Namely,

$$\alpha = 2\arccos\left(\sqrt{P}-1\right).$$

If we follow a similar procedure with $m>1$, i.e., with another combinatorial QADS, we obtain a generalization of the Hadamard test. In this case, the probability of measuring the quantum state $|0\rangle^{\otimes n}|\varphi_0\rangle$ is

$$\left|\left|\left(\frac{1+e^{i\beta}}{2}\right)^m\right|\right|^2 = \cos\left(\frac{\beta}{2}\right)^{2m}, \tag{5.4}$$

and running the test SHOTS times provides an approximation P of such a probability, from which β can be estimated as

$$\alpha = \arccos\left(2\sqrt[m]{P}-1\right).$$

We have tested this 'm-Hadamard test' with different values of m, and equispaced angles in $[0,\pi)$, with a number SHOTS equal to 10^4. We run the experiment, whose code can be found in Appendix A.1, 10^3 times to get an estimation of the phase, measuring the mean absolute error of such an estimation. The results are collected in Fig. 5.2. For convenience, the interval $[0,\pi)$ has been splitted in two subintervals $[0,\frac{\pi}{2})$ and $[\frac{\pi}{2},\pi)$. Observe the different scale of the two figures. When the phase is "small" (namely in the first subinterval), m bigger yields a smaller mean absolute error, and the opposite occurs for bigger phases. This can be easily explained by the effect of the $m-$th root, since in the first case the cosines are closer to one, whereas in the second one, cosines are closer to zero.

Another way of visualizing this fact is with the average error, whose code can be found in Appendix A.2, for the different phases in the first and second interval, (in both cases for $m=1,2,3,4,5$), as depicted in Fig. 5.3. We can see that increasing m is better for the estimation of angles in the first subinterval, but it is worse for those in the second subinterval. Therefore, we can conclude that, unless the phase is promised to be in the first half of the interval $[0,\pi)$, the $m-$Hadamard test with $m>1$ would be better avoided.

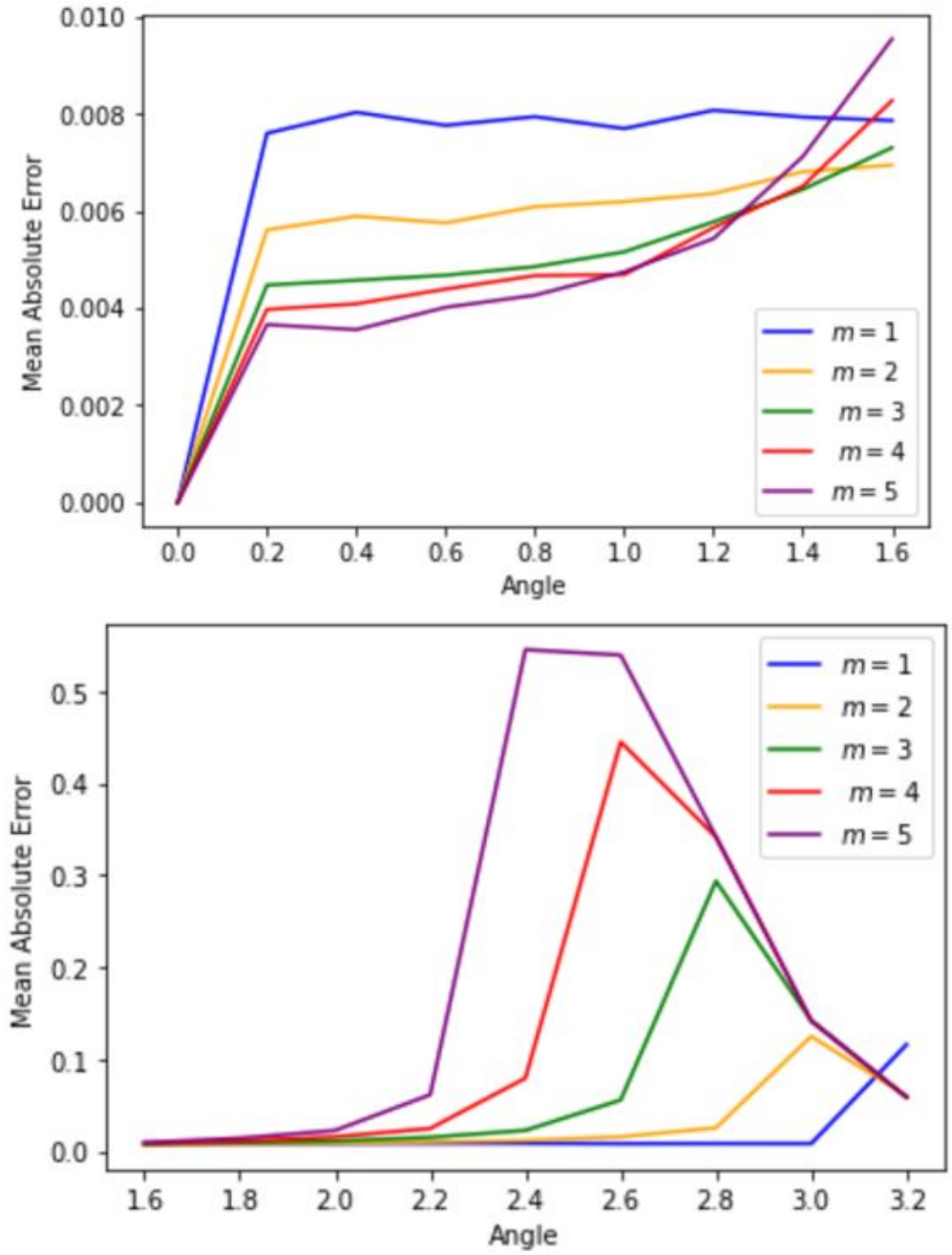


Figure 5.2: Mean absolute error of the estimated phase, with 10^3 experiments, of the m-Hadmard test (with $m = 1, \ldots, 5$), with 10^4 SHOTS in each experiment, for different equispaced phase values.

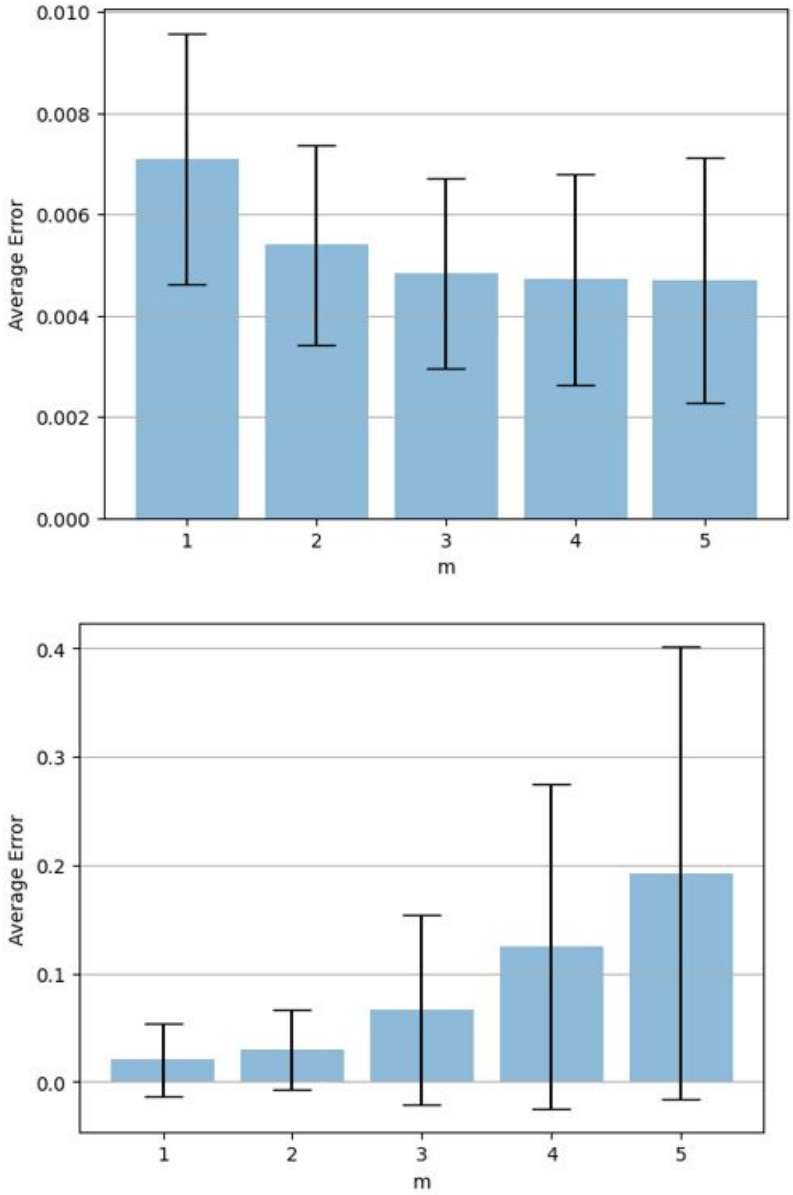


Figure 5.3: Average error of the estimated phase with 10^3 experiments of the $m-$Hadamard test (with $m = 1,2,3,4,5$), with 10^4 SHOTS, for the equispaced phase values in the first and second halves of the interval $[0,\pi)$.

5.4.2 Dichotomy search

An alternative for phase estimation is a dichotomy search based on the decision of eigenvalues procedure of the previous subsection. The idea is, as in the original dichotomy search, to iteratively split the interval $[0,\pi)$ in halves, deciding in each iteration to which half the phase belongs to. The decision is based on comparing the phase against the angles that define each subinterval. So, in the first iteration, the phase is compared against 0 and π, in the second one, against 0 and $\frac{\pi}{2}$, or against $\frac{\pi}{2}$ and π, and so on. For this decision, we also use Theorem 5.18, choosing the "left" or "right" subinterval, depending on which extreme angle provides a bigger probability θ (α takes the value of one or another extreme angle.)

We have tested this dichotomy test with different values of m, and 10 equispaced angles in $[0,\pi)$, with 10 iterations, and a number of SHOTS equal to 10^3 in each iteration. We run the experiment 10^3 times to get an estimation of the phase, measuring the mean absolute error of such an estimation, and the overall average error for different values of m. Its code can be found in Appendix A.3. The results are collected in Fig. 5.4. It can be noticed that this method provides uniformly better results when m increases. However, the error is still bigger than the error provided by the standard Hadamard test.

5.4.3 Hybrid methodology

As a consequence, we propose a hybrid approach which takes the advantages of each of the methods presented above. First, we use the dichotomy search to "locate" the phase, and then, we get an actual estimation by using the m-Hadamard test. We have experimented with this hybrid methodology with different values of m, and 10 equispaced angles in $[0,\pi)$, with 2 iterations of the dichotomy search, with a number of 10^3 SHOTS in each iteration. Another 8000 SHOTS are used in the m-Hadamard test. In order to apply the m-Hadamard test, we hit the operator with a rotation of angle e^{iL} , where L is the lower extreme of the interval in which the phase is located. In the end, we add the m-Hadamard estimation to L. The results are collected in Fig. 5.5. The code can by found in Appendix A.4. As in the case of the dichotomy search, this methodology provides uniformly better results when m increases. Moreover, the overall errors, when $m > 1$, beat those of the standard m-Hadamard test.

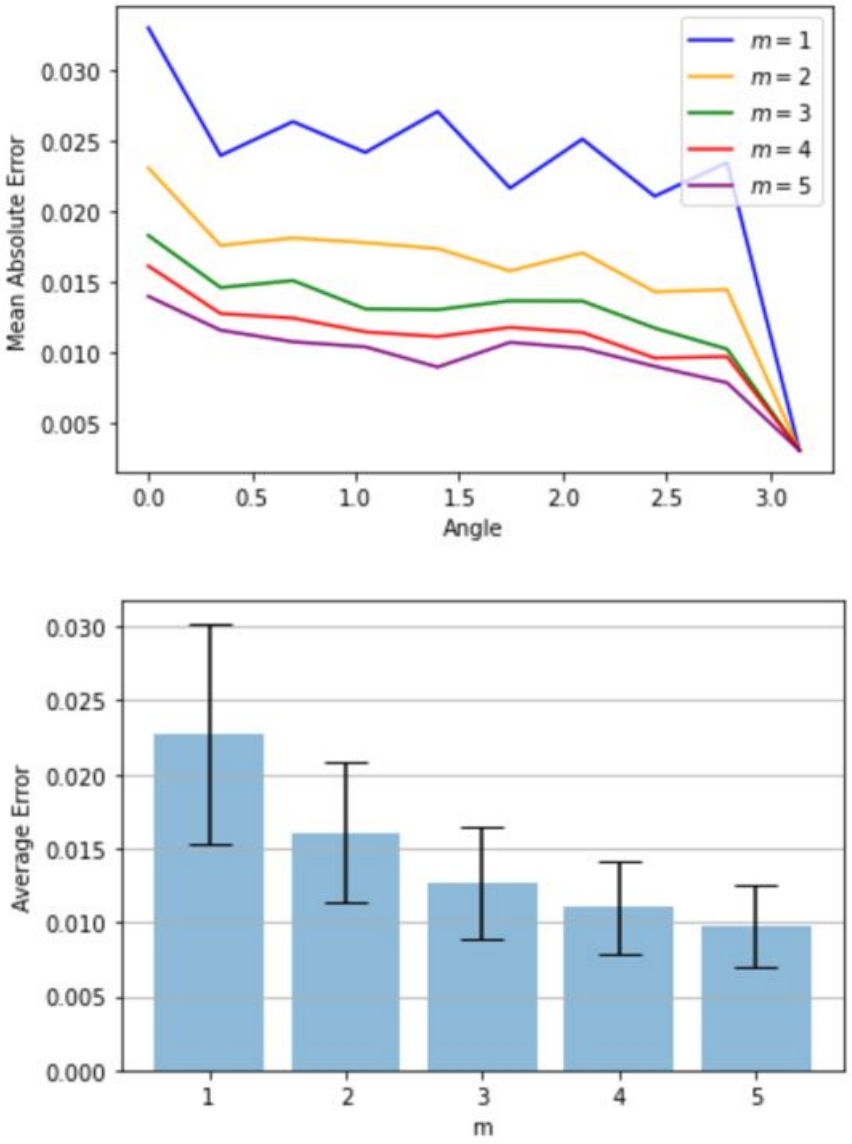


Figure 5.4: Mean absolute and average error of the estimated phase with 10^3 experiments of the Dichotomy test (with $m = 1,2,3,4,5$), with 10 iterations and 10^4 overall SHOTS, for ten equispaced phase values in the first and second halves of the interval $[0,\pi)$.

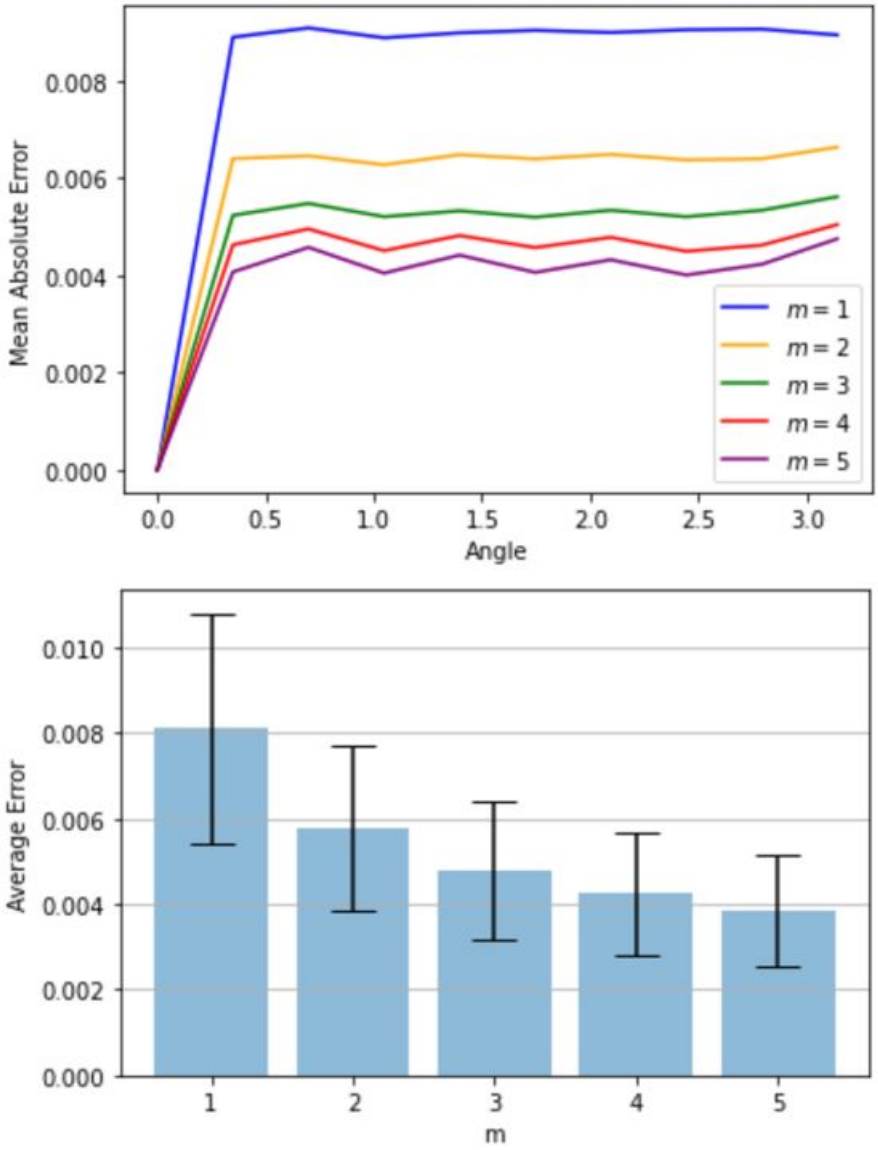


Figure 5.5: Mean absolute and average errors of the estimated phase with 10^3 experiments of the hybrid methodology, with 2 steps of the dichotomy search, and 10^3 SHOTS in each iteration, plus 8000 SHOTS of the m-Hadamard test, for different equispaced phase values.

5.5 Application: Commutativity of Finite Algebras with Combinatorial QADS.

As we mentioned on Chapter 1, the need for an effective procedure to determine the commutativity of the structure is a natural problem in the context of the computational study of finite dimensional algebras [RCR09], [CRR11], [RC12]. As a solution for that, with quantum techniques, an effective and faster algorithmic method for determining the commutativity of finite dimensional algebras, based on Grover's algorithm was proposed in [CRR19a]. As we know, an abstraction of Grover's Algorithm is Grover's QADS. Hence, in this subsection, we perform experiments applying QADS to the problem of the commutativity of algebras of small dimension, comparing the Combinatorial QADS with other QADS.

Lemma 5.19. *For all $s,m \in \mathbb{N}$, and for any $\theta \in [0,2\pi]$,*

$$\begin{aligned}&\sum_{k=0}^{m}\binom{m}{k}\left(\sin\theta\sin((2ks+1)\theta)+\cos\theta\cos((2ks+1)\theta)\right)\\ =&\frac{1}{2}e^{-i\theta}\left(1+e^{-2i\theta s}\right)^m\left((\cos\theta+i\sin\theta)+e^{2i\theta(1+sm)}(\cos\theta-i\sin\theta)\right).\end{aligned}$$

Proof. On one hand,

$$\begin{aligned}\sum_{k=0}^{m}\binom{m}{k}\sin\theta\sin((2ks+1)\theta) &= \sin\theta\sum_{k=0}^{m}\binom{m}{k}\frac{ie^{-i(2ks+1)\theta}-ie^{i(2ks+1)\theta}}{2}\\ &= \sin\theta\frac{i}{2}\left(\sum_{k=0}^{m}\binom{m}{k}\left(e^{-i2s\theta}\right)^k e^{-i\theta}-\sum_{k=0}^{m}\binom{m}{k}\left(e^{i2s\theta}\right)^k e^{i\theta}\right)\\ &= \sin\theta\frac{i}{2}\left(e^{-i\theta}\sum_{k=0}^{m}\binom{m}{k}\left(e^{-i2s\theta}\right)^k-e^{i\theta}\sum_{k=0}^{m}\binom{m}{k}\left(e^{i2s\theta}\right)^k\right)\\ &= \sin\theta\frac{i}{2}\left(e^{-i\theta}\left(1+e^{-i2s\theta}\right)^m-e^{i\theta}\left(1+e^{i2s\theta}\right)^m\right).\end{aligned}$$

On the other hand,

$$\begin{aligned}\sum_{k=0}^{m}\binom{m}{k}\cos\theta\cos((2ks+1)\theta) &= \cos\theta\sum_{k=0}^{m}\binom{m}{k}\frac{e^{i(2ks+1)\theta)}+e^{-i(2ks+1)\theta}}{2}\\ &= \frac{\cos\theta}{2}\left(\sum_{k=0}^{m}\binom{m}{k}\left(e^{i2s\theta}\right)^k e^{i\theta}+\sum_{k=0}^{m}\binom{m}{k}\left(e^{-i2s\theta}\right)^k e^{-i\theta}\right)\\ &= \frac{\cos\theta}{2}\left(e^{i\theta}\sum_{k=0}^{m}\binom{m}{k}\left(e^{i2s\theta}\right)^k+e^{-i\theta}\sum_{k=0}^{m}\binom{m}{k}\left(e^{-i2s\theta}\right)^k\right)\\ &= \frac{\cos\theta}{2}\left(e^{i\theta}\left(1+e^{i2s\theta}\right)^m+e^{-i\theta}\left(1+e^{-i2s\theta}\right)^m\right).\end{aligned}$$

Hence,

$$\begin{aligned}
&\sum_{k=0}^{m}\binom{m}{k}\left(\sin\theta\sin((2ks+1)\theta)+\cos\theta\cos((2ks+1)\theta)\right)\\
=&\sin\theta\frac{i}{2}\left(e^{-i\theta}\left(1+e^{-i2s\theta}\right)^m-e^{i\theta}\left(1+e^{i2s\theta}\right)^m\right)\\
&+\frac{\cos\theta}{2}\left(e^{i\theta}\left(1+e^{i2s\theta}\right)^m+e^{-i\theta}\left(1+e^{-i2s\theta}\right)^m\right)\\
=&\frac{1}{2}e^{-i\theta}\left(i\sin\theta\left(\left(1+e^{-2i\theta s}\right)^m-e^{2i\theta}\left(1+e^{2i\theta s}\right)^m\right)\right)\\
&+\frac{1}{2}e^{-i\theta}\left(\cos\theta\left(\left(1+e^{-2i\theta s}\right)^m+e^{2i\theta}\left(1+e^{2i\theta s}\right)^m\right)\right)\\
=&\frac{1}{2}e^{-i\theta}\left(\left(1+e^{-2i\theta s}\right)^m(\cos\theta+i\sin\theta)+e^{2i\theta}\left(1+e^{2i\theta s}\right)^m(\cos\theta-i\sin\theta)\right)\\
=&\frac{1}{2}e^{-i\theta}\left(1+e^{-2i\theta s}\right)^m\left((\cos\theta+i\sin\theta)+e^{2i\theta}e^{2i\theta sm}(\cos\theta-i\sin\theta)\right)\\
=&\frac{1}{2}e^{-i\theta}\left(1+e^{-2i\theta s}\right)^m\left((\cos\theta+i\sin\theta)+e^{2i\theta(1+sm)}(\cos\theta-i\sin\theta)\right),
\end{aligned}$$

as desired. □

Proposition 5.20. Let $Q=\left(U_f,|\varphi_0\rangle=\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}\right)$, with $N=2^n$, be the output of Grover's QADS, (Section 4.6). And denote by $\tilde{Q}$ its corresponding combinatorial QADS. Then, the probability of error of the detection scheme (Algorithm 4.14) for $\tilde{Q}$ is

$$\frac{\sum_{s=0}^{S}\cos^{2m}(\theta s)\cos^2(\theta sm)}{(S+1)}.$$

Proof. By Proposition 5.4, we have that, for all $s\in\mathbb{N}$, $C(m,U_f)^s=C(m,U_f^s)$. By Proposition 5.3, we know the amplitude of the state $C(m,U_f)|0\rangle^{\otimes m}|\varphi_0\rangle$. Also, by section 4.4, we know that $U_f^k|\varphi_0\rangle=\cos((2k+1)\theta)|B\rangle+\sin((2k+1)\theta)|A\rangle$. Thus,

$$\langle\varphi_0|U_f^k|\varphi_0\rangle=\cos((2k+1)\theta)\cos\theta+\sin((2k+1)\theta)\sin\theta.$$

And, by Lemma 5.19, we have that

$$
\begin{aligned}
&\frac{\sum_{s=0}^{S}\left|\langle|0^{\otimes m}\rangle|\varphi_0\rangle|C(m,U_f)^s||0^{\otimes m}\rangle|\varphi_0\rangle\rangle\right|^2}{S+1}\\
=&\frac{\sum_{s=0}^{S}\left|\sum_{k=0}^{m}\binom{m}{k}\langle\varphi_0|U_f^{ks}|\varphi_0\rangle\right|^2}{2^{2m}(S+1)}\\
=&\frac{\sum_{s=0}^{S}\left|\sum_{k=0}^{m}\binom{m}{k}\left(\sin\theta\sin((2ks+1)\theta)+\cos\theta\cos((2ks+1)\theta)\right)\right|^2}{2^{2m}(S+1)}\\
=&\frac{\sum_{s=0}^{S}\left|\frac{1}{2}e^{-i\theta}\left(1+e^{-2i\theta s}\right)^m\left(\left(\cos\theta+i\sin\theta\right)+e^{2i\theta(1+sm)}\left(\cos\theta-i\sin\theta\right)\right)\right|^2}{2^{2m}(S+1)}\\
=&\frac{\sum_{s=0}^{S}\left|\frac{1}{2}e^{-i\theta}\left(1+e^{-2i\theta s}\right)^m\right|^2\left|\left(e^{i\theta}+e^{2i\theta(1+sm)}e^{-i\theta}\right)\right|^2}{2^{2m}(S+1)}\\
=&\frac{\sum_{s=0}^{S}\frac{1}{2^2}2^{2m}\cos^{2m}(\theta s)\left|\left(e^{i\theta}\left(1+e^{2i\theta sm}\right)\right)\right|^2}{2^{2m}(S+1)}\\
=&\frac{\sum_{s=0}^{S}\frac{1}{4}\cos^{2m}(\theta s)4\cos^2(\theta sm)}{(S+1)}\\
=&\frac{\sum_{s=0}^{S}\cos^{2m}(\theta s)\cos^2(\theta sm)}{(S+1)},
\end{aligned}
$$

as desired. □

Now, we apply the m-combinatorial QADS to the detection problem considered in [Section 5 [CRR20]]. Namely, detection among 32 elements, where exactly one is marked. We want to find the probability of success with the detection scheme, Algorithm 4.14, for 50 iterations. By applying the previous result on that situation, we found the results given in Figure 5.6.

Recall that, for $m = 1$, our QADS is Controlled-Grover, and for $m = 2$, our QADS is Controlled Grover $\otimes$ Controlled Grover. So, we recover the probabilities of [CRR20]. For $m > 2$, the m- combinatorial QADS yields higher success probabilities.

Now, let us consider the case where we have a non-commutative algebra A of dimension 3, with multiplication table $\{M_{ijk}\}_{i,j,k=1}^{3}$ such that $M_{ijk} \neq M_{jik}$, for exactly one (i,j,k), i.e., it is

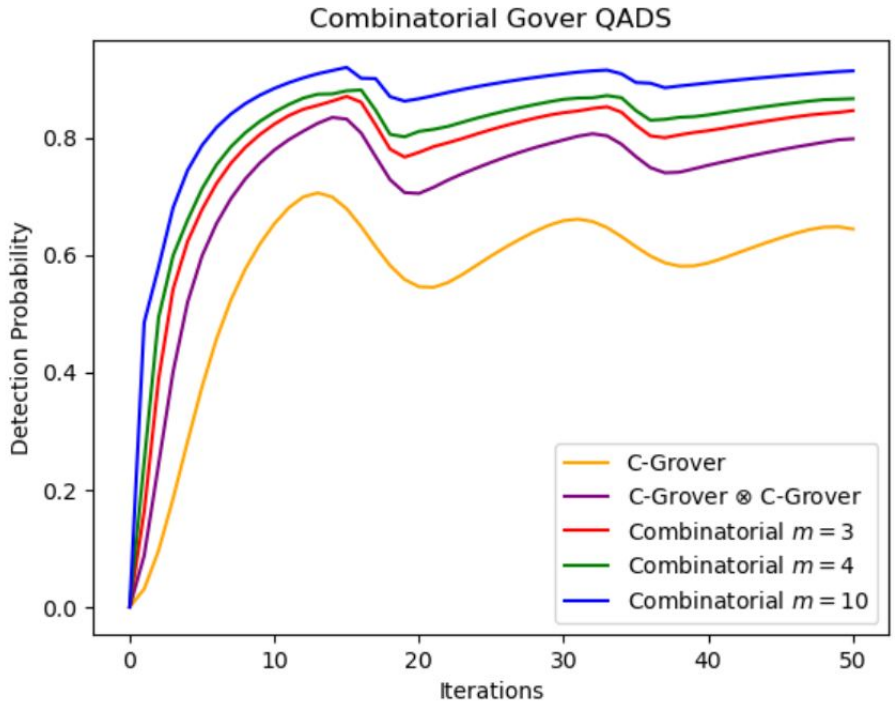


Figure 5.6: Probability of success of Algorithm 4.14, with $m = 1,2,3,4,10$, for detecting a unique marked elements among 32 elements.

not commutative, but only one pair of constants $\left(M_{ijk}, M_{jik}\right)$ is different (see Section 2.3). From the same section, we know that the multiplication table of A would have the following form,

$$A_1 = \begin{pmatrix} M_{111} & M_{121} & M_{131} \\ M_{112} & M_{122} & M_{132} \\ M_{113} & M_{123} & M_{133} \end{pmatrix} A_2 = \begin{pmatrix} M_{211} & M_{221} & M_{231} \\ M_{212} & M_{222} & M_{232} \\ M_{213} & M_{223} & M_{233} \end{pmatrix} A_3 = \begin{pmatrix} M_{311} & M_{321} & M_{331} \\ M_{312} & M_{322} & M_{332} \\ M_{313} & M_{323} & M_{333} \end{pmatrix}.$$

So, without loss of generality, let us assume that only $M_{123} \neq M_{213}$. By [Lemma 1, [CRR19a]], we can embed our algebra A of dimension 3 into an algebra $\tilde{A}$ of dimension $2^2 = 4$, so we have 64 constants in the multiplication table of $\tilde{A}$ such that two of them are pairwise different. Thus, the multiplication table would become as:

$$\tilde{A_1} = \begin{pmatrix} M_{111} & M_{121} & M_{131} & 0 \\ M_{112} & M_{122} & M_{132} & 0 \\ M_{113} & M_{123} & M_{133} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \tilde{A_2} = \begin{pmatrix} M_{211} & M_{221} & M_{231} & 0 \\ M_{212} & M_{222} & M_{232} & 0 \\ M_{213} & M_{223} & M_{233} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\tilde{A_3} = \begin{pmatrix} M_{311} & M_{321} & M_{331} & 0 \\ M_{312} & M_{322} & M_{332} & 0 \\ M_{313} & M_{323} & M_{333} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \tilde{A_4} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

So, applying Algorithm 4.14, with $m = 1,2,3,4,10$, we can see that, for an m-combinatorial QADS, the probability of success of finding the pair of different constants improves (see Figure 5.7, whose respective code can be found in Appendix A.6).

Consequently, the m-combinatorial QADS might be used in the computational study of finite semifields providing better detection probabilities of non-commutativity that the methods considered in [CRR20].

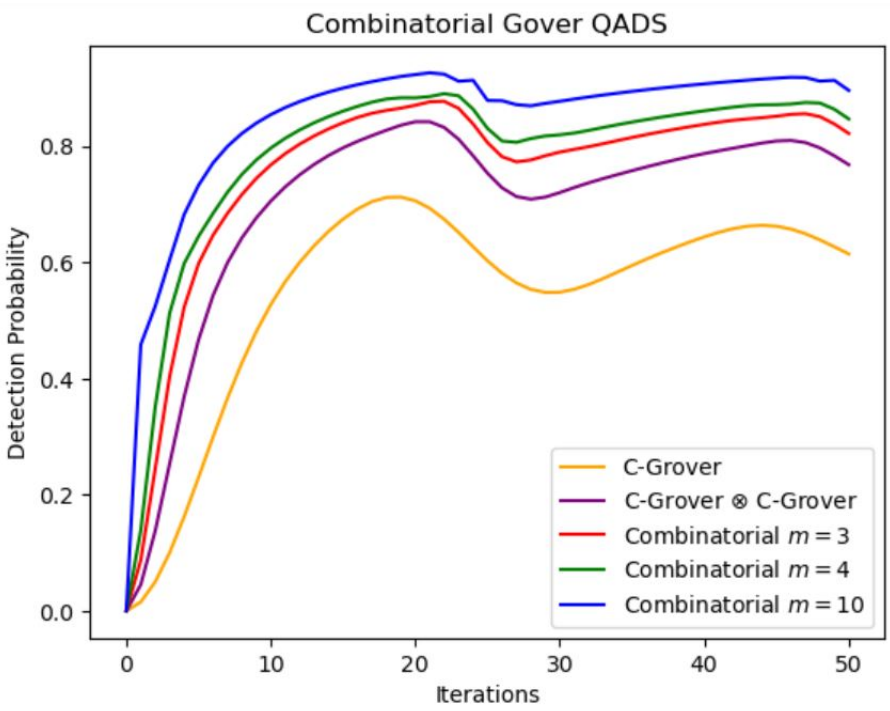


Figure 5.7: Probability of detecting with Algorithm 4.14, and an m-combinatorial QADS with $m = 1,2,3,4,10$, the non-commutativity of a 3-dimensional algebra where only one pair of constants $\left(M_{ijk}, M_{jik}\right)$ is different.

Chapter 6

Efficient Quantum Algorithms To Find Substructures On Finite Algebras

In this chapter, we introduce quantum algorithms that find substructures of a given finite dimensional algebra over a finite field $\mathbb{F}_p$ (such as the right, middle, and left nuclei, the nucleus, and the center) from the multiplication table . We solve this task efficiently, by formulating it as an instance of the Hidden Subgroup Problem (HSP) (Section 4.2). We give detailed constructions of the quantum circuits involved in the process, and prove that the overall (quantum) complexity of our algorithm is polynomial in the dimension of the algebra.

Let A be a non-associative and non-commutative K-algebra, with $K = \mathbb{F}_p$ a finite field of prime cardinality p, and with a fixed basis $\beta = \{e_1, \ldots, e_n\}$. Let $\{M_{ijk}\}_{i,j,k=1}^{n} \subseteq K$ be the multiplication table of the algebra with respect to β. Consider the additive group $G = (A, +)$, i.e., the elements of the algebra with the addition operation. So, G is a finite abelian group, namely $G \cong (\mathbb{Z}/p\mathbb{Z})^n$. Recall the following sets, known as the right, middle, and left nuclei, the nucleus and the center:

$$
\begin{aligned}
N_r(A) &= \{a \in A : [x, y, a] = 0 \text{ for all } x, y \in A\} \\
N_m(A) &= \{a \in A : [x, a, y] = 0 \text{ for all } x, y \in A\} \\
N_l(A) &= \{a \in A : [a, x, y] = 0 \text{ for all } x, y \in A\} \\
N(A) &= N_r(A) \cap N_m(A) \cap N_l(A) \\
Z(A) &= N(A) \cap \{a \in A : [a, x] = 0 \text{ for all } x \in A\}.
\end{aligned}
$$

These sets, which can be written in terms of the K-basis β and the multiplication table, provide information about the algebra. For instance, when A is a finite semifield, i.e., a finite division ring, these sets are related to properties of the corresponding coordinates projective planes [Alb60]. Finding those sets can be stated in terms of the HSP, and it is important in the context of effective classification of finite semifields, see for instance [RCR09], and [HCR23]. So, the problem addressed in this chapter is as follows.

Given: Multiplication table of a finite dimensional K-algebra A (K a finite field $\mathbb{F}_p$).

Problem: To find $N_r(A), N_m(A), N_l(A), N(A)$, and $Z(A)$.

In order to solve it with quantum techniques, we will transform each problem of finding $N_r(A), N_m(A), N_l(A), N(A)$ and $Z(A)$ into an instance of the HSP, which in general can be stated as follows:

Given: The ability to evaluate a hiding function f for a subgroup H of a group G (i.e., a function f that is constant on a subgroup H of G and is distinct on different cosets of H) on arbitrary elements of G.

Problem: To Find $s_1, s_2, \ldots, s_l$, a generating set for H.

In order to solve it, in Section 6.1, we show first that those substructures can be written in terms of the given multiplication of the finite dimensional algebra. In Section 6.2.1, we model the problem of finding substructures in a finite-dimensional algebra as an instance of the HSP, and we show that, without the knowledge of extra information on the hiding function, this kind of problem can not be classically solved with a polynomial number of function accesses to the hiding function f (Section 6.2.2). In Section 6.3, we construct an efficient quantum oracle for the function f, and we build an efficient circuit for the solution of the corresponding HSP.

6.1 Substructures

In this section, we show that the right nucleus, middle nucleus, left nucleus, nucleus and center of a non-associative and non-commutative K-algebra A can be written in terms of the K-basis and the structure constants.

Firstly, in terms of the K-basis: let $x, y \in A$, so there exist $\alpha_i, \beta_j \in K$, for all $i, j = 1, \ldots, n$, such that $x = \sum_{i=1}^{n} \alpha_i e_i, y = \sum_{j=1}^{n} \beta_j e_j$. Let $a \in A$, then,

$$\begin{aligned}
[x,y,a] &= (xy)a - x(ya) \\
&= \left(\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j e_i e_j\right) a - \left(\sum_{i=1}^{n} \alpha_i e_i\right)\left(\sum_{j=1}^{n} \beta_j e_j a\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j \left(e_i e_j\right) a - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j e_i \left(e_j a\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j \left(\left(e_i e_j\right) a - e_i \left(e_j a\right)\right)
\end{aligned}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j[e_i,e_j,a].$$

Also,

$$\begin{aligned}
[x,a,y] &= (xa)y - x(ay) \\
&= \left(\sum_{i=1}^{n} \alpha_i (e_i a)\right)\left(\sum_{j=1}^{n} \beta_j e_j\right) - \left(\sum_{i=1}^{n} \alpha_i e_i\right)\left(\sum_{j=1}^{n} \beta_j a e_j\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j (e_i a) e_j - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j e_i (a e_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j \left((e_i a) e_j - e_i (a e_j)\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j [e_i,a,e_j],
\end{aligned}$$

and

$$\begin{aligned}
[a,x,y] &= (ax)y - a(xy) \\
&= \left(\sum_{i=1}^{n} \alpha_i (a e_i)\right)\left(\sum_{j=1}^{n} \beta_j e_j\right) - a\left(\sum_{i=1}^{n} \alpha_i e_i\right)\left(\sum_{j=1}^{n} \beta_j e_j\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j (a e_i) e_j - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j a (e_i e_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j \left((a e_i) e_j - a (e_i e_j)\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j [a,e_i,e_j].
\end{aligned}$$

Therefore,

$$\begin{aligned}
N_r(A) &= \{a \in A : [e_i,e_j,a] = 0 \text{ for } i,j = 1,\ldots,n\} \\
N_m(A) &= \{a \in A : [e_i,a,e_j] = 0 \text{ for } i,j = 1,\ldots,n\} \\
N_l(A) &= \{a \in A : [a,e_i,e_j] = 0 \text{ for } i,j = 1,\ldots,n\}.
\end{aligned}$$

And, since

$$[a,x] = ax - xa = \sum_{i=1}^{n} \alpha_i a e_i - \sum_{i=1}^{n} \alpha_i e_i a = \sum_{i=1}^{n} \alpha_i (a e_i - e_i a) = \sum_{i=1}^{n} \alpha_i [a,e_i],$$

the center is

$$Z(A) = N(A) \cap \{a \in A : [a,e_i] = 0 \text{ for } i = 1,\ldots,n\}.$$

When the K-algebra is associative, then $N_r(A) = N_m(A) = N_l(A) = N(A) = A$, and $Z(A) = \{a \in A : [a, e_i] = 0 \text{ for } i = 1, \ldots, n\}$. And, when the K-algebra is both associative and commutative, then $Z(A) = A$.

Secondly, in terms of the structure constants: let $a = \sum_{m=1}^{n} \alpha_m e_m$, so the conmutator is

$$\begin{aligned}
[a, e_i] &= ae_i - e_i a \\
&= \left(\sum_{m=1}^{n} \alpha_m e_m \right) e_i - e_i \left(\sum_{m=1}^{n} \alpha_m e_m \right) \\
&= \sum_{m=1}^{n} \alpha_m e_m e_i - \sum_{m=1}^{n} \alpha_m e_i e_m \\
&= \sum_{m=1}^{n} \sum_{k=1}^{n} \alpha_m M_{mik} e_k - \sum_{m=1}^{n} \sum_{k=1}^{n} \alpha_m M_{imk} e_k \\
&= \sum_{m=1}^{n} \sum_{k=1}^{n} \alpha_m \left(M_{mik} - M_{imk} \right) e_k,
\end{aligned}$$

and the associator $[e_i, e_j, a]$ is

$$\begin{aligned}
[e_i, e_j, a] &= \left(e_i e_j \right) a - e_i \left(e_j a \right) \\
&= \left(\sum_{k=1}^{n} M_{ijk} e_k \right) \left(\sum_{m=1}^{n} \alpha_m e_m \right) - e_i \left(e_j \left(\sum_{m=1}^{n} \alpha_m e_m \right) \right) \\
&= \sum_{k=1}^{n} M_{ijk} e_k \left(\sum_{m=1}^{n} \alpha_m e_m \right) - e_i \sum_{m=1}^{n} \alpha_m e_j e_m \\
&= \sum_{k=1}^{n} \sum_{m=1}^{n} \alpha_m M_{ijk} e_k e_m - \sum_{m=1}^{n} \sum_{k=1}^{n} \alpha_m e_i M_{jmk} e_k \\
&= \sum_{k=1}^{n} \sum_{m=1}^{n} \sum_{s=1}^{n} \alpha_m M_{ijk} M_{kmr} e_s - \sum_{m=1}^{n} \sum_{k=1}^{n} \sum_{s=1}^{n} \alpha_m M_{jms} M_{iks} e_s \\
&= \sum_{s=1}^{n} \left(\sum_{m=1}^{n} \left(\sum_{k=1}^{n} \left(M_{ijk} M_{kms} - M_{jmk} M_{iks} \right) \alpha_m \right) \right) e_s.
\end{aligned}$$

Analogously, for the associators $[e_i, a, e_j]$ and $[a, e_i, e_j]$, we have

$$[e_i, a, e_j] = \sum_{s=1}^{n} \left(\sum_{m=1}^{n} \left(\sum_{k=1}^{n} \left(M_{imk} M_{kjs} - M_{mjk} M_{iks} \right) \alpha_m \right) \right) e_s,$$

$$[a, e_i, e_j] = \sum_{s=1}^{n} \left(\sum_{m=1}^{n} \left(\sum_{k=1}^{n} \left(M_{mik} M_{kjs} - M_{ijk} M_{mks} \right) \alpha_m \right) \right) e_s.$$

Thus,

$$\begin{aligned} N_r(A) = &\{a \in A : \left(M_{ij1}M_{111} - M_{j11}M_{i11}\right)\alpha_1 + \cdots + \left(M_{ijn}M_{nn1} - M_{jnn}M_{in1}\right) \\ &\alpha_n = 0, \ldots, \left(M_{ij1}M_{11n} - M_{j11}M_{i1n}\right)\alpha_1 + \cdots + \left(M_{ijn}M_{nnn} - M_{jnn}\right. \\ &\left.M_{inn}\right)\alpha_n = 0, \text{ for all } i, j = 1, \ldots, n\}. \end{aligned}$$

In a similiar way, we can obtain expressions for $N_m(A), N_l(A), N(A)$, and $Z(A)$.

6.2 The classical approach

In this section, we model the problem of finding substructures in a finite-dimensional algebra as a HSP, and we show that without the knowledge of extra information on the hidding function, this kind of problem can not be classically solved in a polynomial number of function accesses to the hiding function f. In particular, we explicitly give the functions that hide the right, middle, and left nuclei, nucleus and center of a finite dimensional K-algebra A, in terms of its multiplication table.

6.2.1 Hiding functions

Namely, for N_r consider the following function:

$$\begin{array}{rccc} f_{N_r}: & A & \to & A^{n^2} \\ & a & \mapsto & f_{N_r}(a) = ([e_1, e_1, a], [e_1, e_2, a], \ldots, [e_n, e_n, a]). \end{array}$$

Note that $f_{N_r}(a_1) = f_{N_r}(a_2)$ if and only if $a_1 - a_2 \in N_r(A)$. Indeed,

$$\begin{aligned} f_{N_r}(a_1) = f_{N_r}(a_2) \Leftrightarrow & ([e_1, e_1, a], \ldots, [e_n, e_n, a]) = ([e_1, e_1, a], \ldots, [e_n, e_n, a]) \\ \Leftrightarrow & ([e_1, e_1, a_1] - [e_1, e_1, a_2], \ldots, [e_n, e_n, a_1] - [e_n, e_n, a_2]) = (0, \ldots, 0) \\ \Leftrightarrow & [e_i, e_j, a_1] - [e_i, e_j, a_2] = 0, \text{ for all } i, j = 1, \ldots, n \\ \Leftrightarrow & \left(e_i e_j\right) a_1 - e_i\left(e_j a_1\right) - \left(e_i e_j\right) a_2 + e_i\left(e_j a_2\right) = 0, \text{ for all } i, j = 1, \ldots, n \\ \Leftrightarrow & \left(e_i e_j\right)(a_1 - a_2) - e_i\left(e_j(a_1 - a_2)\right) = 0, \text{ for all } i, j = 1, \ldots, n \\ \Leftrightarrow & (a_1 - a_2) \in N_r(A). \end{aligned}$$

Hence, we can say that f hides the subgroup $N_r(A)$. For $N_m(A), N_l(A)$, we consider the functions

$$\begin{array}{rccc} f_{N_m}: & A & \to & A^{n^2} \\ & a & \mapsto & f_{N_m}(a) = ([e_1, a, e_1], [e_1, a, e_2], \ldots, [e_n, a, e_n]), \end{array}$$

and

$$\begin{array}{rccc} f_{N_l}: & A & \to & A^{n^2} \\ & a & \mapsto & f_{N_l}(a) = ([a, e_1, e_1], [e_1, a, e_2], \ldots, [e_n, a, e_n]). \end{array}$$

Analogously, as for f_{N_r}, it can be seen that $f_{N_m}(a_1) = f_{N_m}(a_2)$ if and only if $a_1 - a_2 \in N_m(A)$, and that $f_{N_l}(a_1) = f_{N_l}(a_2)$ if and only if $a_1 - a_2 \in N_l(A)$. Hence, we can say that f_{N_m} and f_{N_l} hide the subgroups $N_m(A)$ and $N_l(A)$, respectively. For $N(A)$, consider the function

$$\begin{array}{rccc} f_N: & A & \to & A^{3n^2} \\ & a & \mapsto & f_N(a) = (f_{N_r}(a), f_{N_m}(a), f_{N_l}(a)). \end{array}$$

So, $f_N(a_1) = f_N(a_2)$ if and only if $a_1 - a_2 \in N_r(A) \cap N_m(A) \cap N_l(A) = N(A)$. And, for $Z(A)$ consider the following function

$$\begin{array}{rccc} f_Z: & A & \to & A^{3n^2+n} \\ & a & \mapsto & f_{Z_1}(a) = (f_N(a), [a,e_1], \ldots, [a,e_n]). \end{array}$$

Hence, $f_Z(a_1) = f_Z(a_2)$ if and only if $a_1 - a_2 \in N(A) \cap \{a \in A : [a,e_i] = 0 \text{ for } i = 1,\ldots,n\} = Z(A)$.

As we can see, all hiding functions are given in terms of commutators and associators of the basis elements of the algebra and the argument of the hiding function. So, in order to show that the hiding functions can be written in terms of the multiplication table of the algebra, we only need to observe that, if $a = \sum_{m=1}^{n} \alpha_m e_m$, then we have the conmutator

$$[a,e_i] = \sum_{k=1}^{n} \left(\sum_{m=1}^{n} \alpha_m \left(M_{mik} - M_{imk} \right) \right) e_k,$$

and the associator

$$[e_i,e_j,a] = \sum_{s=1}^{n} \left(\sum_{m=1}^{n} \left(\sum_{k=1}^{n} \left(M_{ijk} M_{kms} - M_{jmk} M_{iks} \right) \alpha_m \right) \right) e_s.$$

Analogously, for the associators $[e_i,a,e_j]$, and $[a,e_i,e_j]$.

6.2.2 Classical solution

Next, we consider the classical (i.e., non-quantum) solution to the HSP via the hiding output of a hiding function. The idea is to show that, as long as there exist different subgroups hidden by the same function, extra evaluations of such a hiding function are needed in order to distinguish them. The following is a technical result.

Lemma 6.1. *Let G be a finite group having N subgroups with trivial pairwise intersection. Let $g_1,\ldots,g_t \in G$ be such that $m = N - \binom{t}{2} \geq 1$. Then, there exist m subgroups $H_1,\ldots,H_m$, out of the N, given by hiding functions $f_1,\ldots,f_m : G \to \mathbb{N}$, such that $f_i(g_k) = f_j(g_k)$, for all $1 \leq i,j \leq m$, and $1 \leq k \leq t$.*

Proof. The proof follows by induction over t. For $t = 2$, define $f_i(g_1) = 1$, for every one of the $i = 1,\ldots,N$ subgroups, and extend it to G in the following way: $f_i(g_k) = j$ if $g_k H_i = e_j H_i$, where $e_1 H_i, e_2 H_i, \ldots, e_{t_i} H_i$ are the different classes of G mod H_i.

Assume, for $t > 2$, that there exist $m = N - \binom{t-1}{2}$ subgroups $H_1, \ldots, H_m$, with hiding functions $f_1, \ldots, f_m$, such that $f_i(g_k) = f_j(g_k)$, for all $1 \leq i, j \leq m$, $1 \leq k \leq t-1$. Take $g_t \in G$, if there exists a $k < t$ such that $g_t = g_k$, the result follows directly. Otherwise, $g_t \notin \{g_1, \ldots, g_{t-1}\}$, and consider $h_k = g_t^{-1} g_k \neq 1$, for $k = 1, \ldots, t-1$. A fixed h_k belongs, at most, to one of the $H_1, \ldots, H_m$ subgroups (because $h_k \neq 1$). Take those H_l, out of the N given, such that $h_k \notin H_l$, for $k = 1, \ldots, m$. We can redefine $f_l(g_t h_l) = \max\{f_l(g_k)\} + 1$, for all $h_l \in H_l$. This function still can be seen to hide H_l, because $g_t^{-1} g_k \notin H_l$, for all $k = 1, \ldots, m$. At most, there are $t-1$ subgroups that are not taken in this step, so we are left with

$$m - (t-1) = N - \binom{t-1}{2} - (t-1) = N - \binom{t}{2}$$

subgroups, thus the result follows. □

Theorem 6.2. *Under the conditions of the previous lemma, t evaluations of a general hiding function f are not enough to solve the corresponding HSP via the hiding function in a classical computer. This holds, in particular, if $t \leq \lfloor \sqrt{N} \rfloor$, and $N \geq 2$.*

Proof. Clearly,

$$\binom{t}{2} < \frac{t^2}{2} \leq \frac{N}{2}.$$

Then,

$$N - \binom{t}{2} = \frac{N}{2} + \frac{N}{2} - \binom{t}{2} \geq \frac{N}{2} + 1 \geq 2.$$

Assume the existence of a HSP solver using t evaluations of the hiding function. Evaluations in the t elements $g_1, \ldots, g_t$ of the previous theorem provide the same information for the m subgroups $H_1, \ldots, H_m$. Consequently, they can not be distinguished by the HSP solver. □

Corollary 6.3. *Computing the right, middle, and left nuclei, center, and center of a finite dimensional algebra A over a finite field K of q elements, with a HSP solver, and no extra knowledge of the hiding function, on a classical computer, requires at least $\Omega(\sqrt{p^n})$ evaluations of the corresponding hiding function.*

Proof. In our case, if we consider the additive group of the algebra $G = (A, +) \cong K^n$ with $K = \mathbb{F}_p$, the number of $1-$dimensional subspaces is $1 + p + \cdots + p^{n-1}$, all of them with trivial pairwise intersection. Following the corollary, it is necessary at least $\lfloor \sqrt{1 + p + \cdots + p^{n-1}} \rfloor = \Omega(\sqrt{p^n})$ evaluations for a HSP solver to compute each of the mentioned substructures. □

As an aside note, observe that it is always possible to have a finite dimensional algebra A for which $N_l = N_m = N_r = N = Z$ is of dimension 1 over K. For instance, a Generalized Twisted Field, for particular choice of its defining parameters [Alb61]. In detail, let $F = \mathbb{F}_q$ with $q = p^r > 2$, and $K = \mathbb{F}_q^m$, with $m \geq 3$ and odd. Let $g \in K$ be a primitive element K. Then the polynomial $x^{q-1} - g$ has no roots on K. This is because, if $h \in K$ with $h^{q-1} = g$,

then $1 - h^{q^m-1} = h^{(q-1)\left(1+q+\cdots+q^{m-1}\right)}$, and so the order of g is less than $q^m - 1$, which is a contradiction (the order of g is $q^m - 1$).

Let $A = (K, +, *)$ with the operation defined as in Example 2.44, i.e., $a \cdot b = ab - g\tau(a)\sigma(b)$, with $\sigma : K \to K$ be such that $\sigma(\alpha) = \alpha^q$, $\tau = \sigma^{-1}$, and $a * b = (R_1^{\bullet})^{-1}(a) \bullet (L_1^{\bullet})^{-1}(b)$, for all $a, b \in K$.

Thus, A is a finite semifield, where $N_l = N_r = N = Z = F$. Now, note that $\sigma(a) = \tau(a)$ if and only if $a = \sigma^2(a)$. But, since m is odd, and $\text{ord}(\sigma^2) = \frac{\text{ord}(\sigma)}{gcd(m,2)}$, then $< \sigma^2 >= \text{Aut}(K|F)$. Then, $|M| = |\mathbb{F}_q|$, where $M = \{a \in K \ : \ \sigma(a) = \tau(a)\}$, hence $N_m \cong F$ (here, $\text{ord}(\sigma)$ denotes the order of σ as an element of the group $\text{Aut}(K|F)$).

6.3 The quantum approach

In this section, we solve the problem of finding substructures in a finite-dimensional algebra by quantum procedures. In particular, we construct an efficient quantum oracle for the hiding functions of the substructures, and we build an efficient circuit for the solution of the corresponding HSP. Let us first give an overall picture of the whole procedure.

Algorithm 6.4 (Solution to the HSP for substructures of a finite algebra)**.**
Input: Multiplication table of a finite dimensional K-algebra A with respect to a basis $\{e_1, \ldots, e_m\}$ (K is a finite field of p elements)

Construction of quantum oracle: From the multiplication table, construct a quantum oracle for the hiding function f of a specific subgroup H.

Quantum procedure: Construct a quantum circuit that, using the quantum oracle, that returns tuples $(\alpha_1, \ldots, \alpha_m)$ of elements in K, which provide cooordinates of elements in the orthogonal complement of H, i.e., $(\alpha_1, \ldots, \alpha_m) \cdot (\beta_1, \ldots, \beta_m) = 0$, for all $\sum_{i=1}^m \beta_i e_i \in H$.

Classical Post processing: Compute generators of H from Gaussian elimination on the tuples given by the quantum procedure.

6.3.1 Oracle of the hiding function

Our first task is to show that an efficient quantum oracle (in terms of the number of quantum gates) can be constructed from the multiplication table of the algebra, for each of the hiding functions introduced in the previous section.

We consider the construction for $N_r(A)$, as those for $N_m(A), N_l(A), N(A), Z(A)$, follow the same lines. From the previous section, $f_{N_r}(a)$ can be written in terms of the structure constants,

since if $a = \sum_{m=1}^{n} \alpha_m e_m$, then for all $i, j = 1, \ldots, n$,

$$[e_i, e_j, a] = \sum_{s=1}^{n} \left(\sum_{m=1}^{n} \left(\sum_{k=1}^{n} \left(M_{ijk} M_{kms} - M_{jmk} M_{iks} \right) \alpha_m \right) \right) e_s$$
$$= \sum_{s=1}^{n} \left(\sum_{m=1}^{n} \alpha_m \lambda_{ijms} \right) e_s,$$

where $\lambda_{ijms} = \sum_{k=1}^{n} \left(M_{ijk} M_{kms} - M_{jmk} M_{iks} \right)$ are n^4 constants in the finite field $K = \mathbb{F}_p$. If we consider the coordinates of such an expression, we can go further and expand it to get a coordinate version of f_{N_r} namely

$$\left(\sum_{m=1}^{n} \alpha_m \lambda_{11m1}, \ldots, \sum_{m=1}^{n} \alpha_m \lambda_{ijms}, \ldots, \sum_{m=1}^{n} \alpha_m \lambda_{nnmn} \right) \in K^{n^3}.$$

Let us write, for all λ_{ijms}, with $i, j, m, s = 1, \ldots, n$, their binary representations $\{\lambda_{ijms}^t\}$ for $t = 1, \ldots, r = \lceil \log_2(p) \rceil$ (i.e., $\lambda_{ijms} = \sum_{t=1}^{r} 2^{t-1} \lambda_{ijms}^t$). Using the controlled-multiplier modulo p of [VBE96], each $\lambda_{ijms} \alpha_m$ can be implemented by repeated modular additions (modulo p), requiring $3r+1$ ancillary qubits, and $O(r^2)$ gates (among NOT, CNOT and Toffoli gates). These products can be computed in sequence, reusing the ancillary qubits, while carrying out the modular addition $\sum_{m=1}^{n} \lambda_{ijms} \alpha_m$. Again, by [VBE96], such a modular addition requires only r extra ancillary qubits (to store intermediate values of $\lambda_{ijms} \alpha_m$), and $O((n-1)r)$ gates.

Overall, computing $\sum_{m=1}^{n} \alpha_m \lambda_{ijms}$ requires $O(nr^2)$ gates, and $(3r+1) + r = 4r+1$ ancillary qubits. Since there are n^3 sums of that form, and all ancillary qubits can be reused, the U_{N_R} oracle can be efficiently constructed with $4r+1$ ancillary qubits.

In summary, in Table 6.1, we give the number of qubits corresponding to input, output, ancillary, and the order of the number of gates required to build each oracle $U_{N_r}, U_{N_m}, U_{N_l}, U_N, U_Z$.

Oracle		$U_{N_r}, U_{N_m}, U_{N_l}$	U_N	U_Z
Number of qubits	Input	nr		
	Output	$n^3 r$	$3n^3 r$	$3n^3 r + n^2 r$
	Ancillary	$4r+1$		
Number of gates		$O(n^4 r^2)$		

Table 6.1: Cost in terms of number of qubits and gates of each oracle, for the hiding function of N_r, N_m, N_l, N, Z.

6.3.2 Quantum Algorithm To Find Substructures

Next, we present a quantum efficient algortihm to find the substructures of a finite-dimensional algebra over a finite field $\mathbb{F}_p$, based on the above constructed quantum oracle and the ideas of [Lom04, Section 3]. It uses an efficient computation of the quantum Fourier transform on $G = (A,+) \cong (\mathbb{Z}/p\mathbb{Z})^n$, and it outputs elements of the orthogonal subgroup H to be found:

$$H^{\perp} = \left\{ \alpha = \sum_{i=1}^{n} \alpha_i e_n \in G : \alpha_1\beta_1 + \cdots + \alpha_n\beta_n 0 \equiv \text{mod } p, \text{ for all } \beta = \sum_{i=1}^{n} \beta_i e_i \in H \right\}.$$

As before, we consider the construction for $N_r(A)$, as that of $N_m(A), N_l(A), N(A)$, or $Z(A)$, follow the same lines.

Algorithm 6.5 (Quantum procedure for finding the orthogonal of the Right Nucleus)**.**
Input: A black box which performs the operation $U_{N_r} |a\rangle |0\rangle = |a\rangle |f_{N_r}(a)\rangle$ for $a \in A$, and $\varepsilon \in \mathbb{R}$ such that $\sqrt{2} \geq \varepsilon > 0$, so that the QFT on $(\mathbb{Z}/p\mathbb{Z})^n$ can be computed with error bounded by ε.

Quantum Procedure:

1. Initial state: $|0\rangle^{\otimes r_1} |0\rangle^{\otimes r_2} |0\rangle^{\otimes r_3}$, with $r_1 = nr$ input qubits, $r_2 = n^3 r$ output qubits plus $r_3 = 4r + 1 + n\lceil 12.53 + 3\log_2 n\frac{\sqrt{p}}{\varepsilon}\rceil$, ancillary qubits.
2. Create superposition and remove elements which are $\geq p$.
3. Apply the black box U_{N_r}.
4. Apply the Quantum Fourier Transform from Equation (3.1) on the first register.
5. Measure the first register.

Output: The binary expansion of the coordinates of an element in $N_r(A)^{\perp}$.

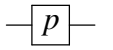
As in many quantum algorithms, superposition is achieved by applying the Hadamard transformation. However, we must notice that removal of elements greater or equal than p is needed, as we are only interested in values mod p. After that, as it is standard in quantum solutions to the HSP, an application of the quantum oracle U_{N_r} is followed by a QFT and a measurement.

Superposition and removal of elements greater or equal than p.

Let us explain with a litte more detail the second step of the quantum procedure. Each element in the algebra is represented by the binary expansion of its coordinates. So, we need nr qubits to deal with all the elements in the algebra in superposition. Let us divide then in n registers of r qubits, each one encoding a single coordinate, which is an integer mod p. Therefore, since

we are only interested in a superposition of p constants ($\frac{1}{\sqrt{p}}\sum_{x=0}^{p-1}|x\rangle$), after a standard superposition of the r qubits with Hadamard gates ($\frac{1}{\sqrt{2^r}}\sum_{x=0}^{2^r-1}|x\rangle$), we need to remove those sumands which are greater or equal than p.

This is accomplished by the use of an extra r qubit register, storing the binary representation of the integer p (by an application of at most r X gates). We represented it on the circuit of Figure 6.1 as



.

Now, for each of the n pairs of $2r$ qubit registers, we use the quantum bit string comparator (QBSC) from [OR07] to remove the undesirable summands (see Figure 6.1). It has a total cost of $O(r)$ CNOT and single qubit gates, and $3r-1$ ancillary qubits. The last two of them, Q_1, Q_2, provide after measurement a comparison with p. Namely, the integer is smaller than p if and only if $Q_1 = 1$ and $Q_2 = 0$. So, such a measurement yields the collapse of the first register to the desired superposition. An undesired measurement forces a repetition of the process.

The probability of failure of one single comparison is $\frac{2^r-p}{2^r} < \frac{1}{2}$. Therefore, if the process is repeated t times, the probability of failure is at most $\frac{1}{2^t}$. Since this technique is to be applied in parallel to each of the n different $2r-$qubit registers, the overall probability of failure is at most $\frac{n}{2^t}$. Choosing $t = \lfloor \log_2\left(\frac{n}{\delta}\right)\rfloor + 1 = O(\log(n))$, yields a bounded probability error $0 < \delta < 1$ of the whole comparison procedure, that can be made arbitrarily small.

For a single coordinate, the number of ancillary qubits required in a single use of a QBSC is $3(r-1)+2$, and the number of qubits measured are two. However, we can uncompute the $3(r-1)$ qubits that were not measured, and reset the measured ones. We apply this process t times sequentially for each $2r$ qubits from the pairs of the first register, giving an overall of $3(r-1)+2$ ancillary qubits. Since the r qubits from the state of p can be shared, we need an overall number of $3(r-1)+2+r = 4r-1$ ancillary qubits.

Steps 3 to 5 in the algorithm.

After the previous step, we achieve the quantum state

$$\frac{1}{\sqrt{p^n}}\sum_{a\in A}|a\rangle|0\rangle^{\otimes r_2}.$$

Here, summation is to be understood over coordinates of elements in A (ancillary qubits are omited), and application of the oracle U_{N_r} yields

$$\frac{1}{\sqrt{p^n}}\sum_{a\in A}|a\rangle|f_{N_r}(a)\rangle,$$

and application of the QFT from Subsection 3.3.1 (Equation 3.1) on the first register, gives

$$\sum_{b\in A}|b\rangle\left(\frac{1}{p^n}\sum_{a\in A}\omega_p^{ab}|f_{N_r}(a)\rangle\right),$$

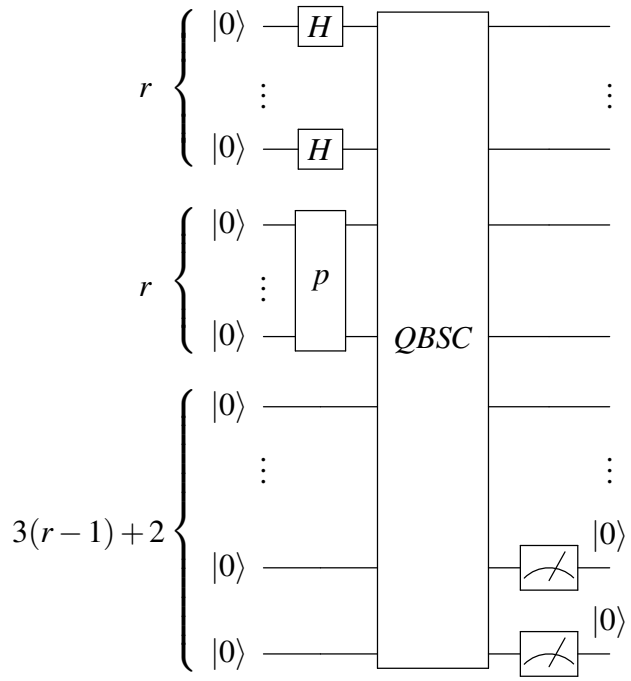


Figure 6.1: Quantum circuit of the step 2 of the quantum procedure for finding $N_r(A)$.

where $\omega_p = \exp\left(\frac{2\pi i}{p}\right)$. A measurement $|b\rangle$ on the first register occurs with probability

$$\left|\left|\frac{1}{p^n}\sum_{a\in A}\omega_p^{ab}\,|f_{N_r}(a)\rangle\right|\right|^2.$$

If $\{s_1, s_2, \ldots, s_l\}$ is a $\mathbb{Z}/p\mathbb{Z}-$basis of H, then, for all $z \in \text{Range}(f_{N_r})$, there exists $a_z \in A$ such that $z = f_{N_r}(a_z + \sum_{i=1}^{l}\lambda_i s_i)$, for all $0 \leq \lambda_1, \ldots, \lambda_l \leq p-1$. So, the probability becomes

$$\begin{aligned}&\left|\left|\frac{1}{p^n}\sum_{z\in\text{Range}(f_{N_r})}\left(\sum_{\lambda_1,\ldots,\lambda_l=0}^{p-1}\omega_p^{b(a_z+\sum_{i=1}^{l}\lambda_i s_i)}\right)|z\rangle\right|\right|^2\\ =&\left|\left|\frac{1}{p^n}\sum_{z\in\text{Range}(f_{N_r})}\omega_p^{ba_z}\prod_{i=1}^{l}\left(\sum_{\lambda_i=0}^{p-1}\omega_p^{b\lambda s_i}\right)|z\rangle\right|\right|^2.\end{aligned}$$

Now, when $bs_i \not\equiv 0 \bmod p$, for some $i = 1, \ldots, l$, then $\sum_{\lambda_i=0}^{p-1}\omega_p^{\lambda_i bs_i} = 0$ (because of Proposition 2.59), and the corresponding summand vanishes. Otherwise, $b \in H^{\perp}$, and the probability is

$$\frac{1}{p^{2n}}\sum_{z\in\text{Range}(f_{N_r})}\left|\omega_p^{ba_z}\prod_{i=1}^{l}\left(\sum_{\lambda_i=0}^{p-1}\omega_p^{0}\right)\right|^2 = \frac{1}{p^{2n}}p^{n-l}(1\cdot p^l)^2 = p^{l-n}$$

Thus, we obtain an element uniformly distributed in $N_r(A)^{\perp}$.

6.3.3 Classical post processing

In order to determine the number of times that the quantum procedure should be run in order to find a generator set of the subgroup $N_r(A)^{\perp}$, we shall use Theorem 2.62. So, running the quantum procedure $s+nr$ times, with $s \geq 0$, gives a generator set $\{g^1, \ldots, g^{s+nr}\}$ of $N_r(A)^{\perp}$, with probability at least $1 - \frac{1}{2^s}$ (because $N_r(A)^{\perp}$ is subgroup of $(A,+)$, which has order at most p^n). Once that we have such a generating set, Gaussian elimination on the following system of $s+nr$ linear equations

$$\begin{array}{ccccccccc} g_1^1 x_1 & + & g_2^1 x_2 & + & \cdots & + & g_n^1 x_n & \equiv & 0 \mod p \\ \vdots & & \vdots & & \vdots & & \vdots & \vdots & \\ g_1^{s+nr} x_1 & + & g_2^{s+nr} x_2 & + & \cdots & + & g_n^{s+nr} x_n & \equiv & 0 \mod p \end{array},$$

gives a generator set of $N_r(A)$. We have the following main result:

Theorem 6.6. *Given the multiplication table of a n-dimensional nonassociative noncommutative $\mathbb{F}_p$-algebra A, for each substructure*

$$H = N_r(A), N_m(A), N_l(A), N(A), Z(A)$$

of A, there exists a quantum algorithm that using the number of qubits and quantum gates of Table 6.2, and together with a classical post-processing algorithm of complexity $O(n^3)$, finds H with a bounded probability error.

Substructure	N_r, N_m, N_l	N	Z
Number of qubits	n^3r+nr	$3n^3r+nr$	$3n^3r+n^2r+nr$
Number of ancillary qubits	$4r+1+n\lceil 12.53+3\log_2 n\frac{\sqrt{p}}{\varepsilon}\rceil$		
Number of gates	$O\left(n^5r^3\right)$		
Number of oracle queries	$O(nr)$		

Table 6.2: Cost of Algorithm 6.5 in terms of number of qubits and gates, for the computation of each substructure $N_r(A), N_m(A), N_l(A), N(A), Z(A)$.

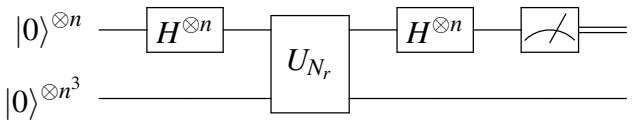
Remark. Observe that the number of ancillary qubits is upper-bound by the number of such qubits involved in the quantum oracle. In other steps of the algorithm, ancillary qubits can be reused, either by uncomputation, or by measuring and resetting them to zero.

Remark. Note that, when p is even, the number of ancillary qubits is $4r+1$, but when p is odd the number of ancillary qubits is $4r+1+n\lceil 12.53+3\log_2 n\frac{\sqrt{p}}{\varepsilon}\rceil$ (see Section 3.3.1).

6.3.4 Examples

As a proof of concept, we shall apply our algorithm to binary algebras. So, we will illustrate the behaviour of Algorithm 6.5 with two examples. In the first one, we compute the center of an associative 3-dimensional $\mathbb{F}_2$-algebra which is not commutative. In the second one, we obtain the right, left, middle nuclei, the nucleus, and the center of a non associative and non commutative 4-dimensional $\mathbb{F}_2$-algebra. Recall that, in our notation, n is the dimension of the $\mathbb{F}_2$-algebra, and $r=\lceil \log_2(p)\rceil$, which in the case $p=2$, gives $r=1$.

Note that, when $p=2$, we can skip the QBSC, and the picture of the quantum circuit from the quantum procedure would look like



Recall that, over $(\mathbb{Z}/2\mathbb{Z})^n$, the Quantum Fourier Transform is $H^{\otimes n}$, see Section 3.3 (Example 3.28).

Example 6.7. Consider the following set:

$$A=\left\{\begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \;:\; a,b,c\in\mathbb{F}_2\right\},$$

with ordinary matrix addition and multiplication. It is a 3-dimensional $\mathbb{F}_2$-algebra, associative but not commutative. Since it is associative, $N_r(A)=N_m(A)=N_l(A)=A$, so $N=A$. Let us find

$Z(A)$ with our algorithm. As a first step, we must find the multiplication table of the algebra. A $\mathbb{F}_2$-basis of A is

$$\beta = \left\{ e_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, e_2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, e_3 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right\}.$$

As we know, the multiplication table can be found by applying $L_{e_i}(e_j)$, for all $i, j = 1,2,3$. Thus,

$$L_{e_1}(e_1) = 1e_1 + 0e_2 + 0e_3, \quad L_{e_1}(e_2) = 0e_1 + 1e_2 + 0e_3, \quad L_{e_1}(e_3) = 0e_1 + 0e_2 + 0e_3$$

$$L_{e_2}(e_1) = 0e_1 + 0e_2 + 0e_3, \quad L_{e_2}(e_2) = 0e_1 + 0e_2 + 0e_3, \quad L_{e_2}(e_3) = 0e_1 + 1e_2 + 0e_3$$

$$L_{e_3}(e_1) = 0e_1 + 0e_2 + 0e_3, \quad L_{e_3}(e_2) = 0e_1 + 0e_2 + 0e_3, \quad L_{e_3}(e_3) = 0e_1 + 0e_2 + 1e_3,$$

Hence, the multiplication table of A is

$$\mathbb{M} = \left\{ A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, A_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, A_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right\}.$$

Now that we have the multiplication table, we would like to find the expansion of the function $f_{Z(A)}$. Because A is associative, f_N is identically zero, and we only need to consider the last 3 components of the f_Z function. Each of the commutators in f_Z has 3 coordinates in the basis β. Using the code B.7, its expression is

$$(0, x_2, 0, 0, x_1 + x_3, 0, 0, x_2, 0),$$

which can be shortened to $(x_2, x_1 + x_3)$, where $x_1, x_2, x_3 \in \mathbb{F}_2$, by eliminating the coordinates that are always 0. The quantum oracle that performs the unitary operation $U_Z |a\rangle |0\rangle = |a\rangle |f_Z(a)\rangle$, with $a = x_1e_1 + x_2e_2 + x_3e_3$, can be seen in Figure 6.2. And it can be simulated with the code

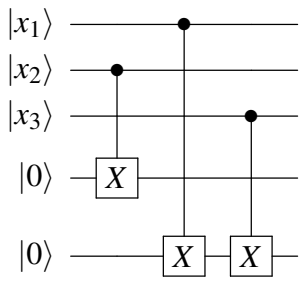


Figure 6.2: Oracle for U_Z, in Example 6.7.

B.8. Now that we have our oracle, we have the circuit for our quantum procedure, which can be seen with B.9, and we can simulated with B.10. Repeatedly using Algorithm 6.5, we find that the elements that belong to the orthogonal of $Z(A)$ are, with high probability,

$$\{010, 000, 101, 111\},$$

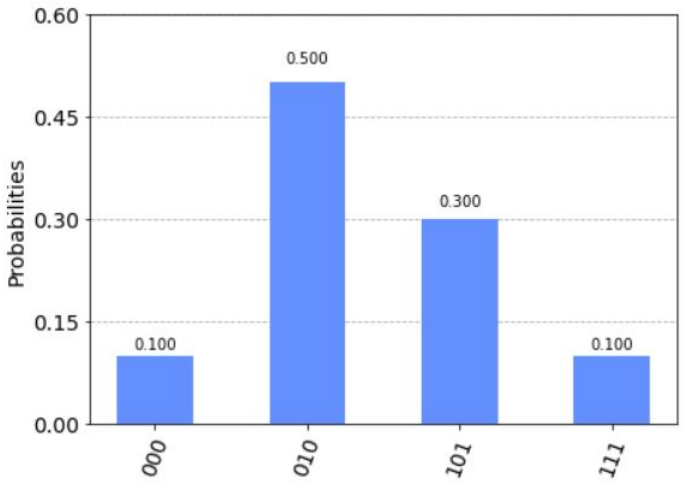


Figure 6.3: Elements in $Z(A)^{\perp}$, after 10 repetitions of Algorithm 6.5.

(see Figure 6.3) following Theorem 2.62 (which have been obtained with 10 repetitions, so the probability of not obtaining a complete set of generators for $Z(A)^{\perp}$ is below 1%). This leads to the system of equations(6.1).

$$\begin{cases} 0x_1 + 0x_2 + 0x_3 \equiv 0 \mod 2 \\ 0x_1 + 1x_2 + 0x_3 \equiv 0 \mod 2 \\ x_1 + 0x_2 + x_3 \equiv 0 \mod 2 \\ x_1 + x_2 + x_3 \equiv 0 \mod 2 \end{cases} \tag{6.1}$$

So, by applying Gauss-Jordan, we found two solutions $(x_1 = 1, x_2 = 0, x_3 = 1)$, and $(x_1 = 0, x_2 = 0, x_3 = 0)$.

Therefore $(x_1 = 1, x_2 = 0, x_3 = 1)$ generates $Z(A)$. This is, indeed, the correct solution, since the only non-zero element that is mapped to $(0,0)$ by the hiding function $(x_2, x_1 + x_3)$ is exactly $(1,0,1)$.

Example 6.8. Consider the following multiplication table for a 4-dimensional $\mathbb{F}_2$-algebra:

$$A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} A_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} A_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

This algebra is neither associative nor commutative. Let us use our algorithm to find its right, middle, and left nuclei, the nucleus, and the center.

Right Nucleus. First, let us find the coordinate expansion of the function $f_{N_r}(A)$, using the code B.11. After eliminating zeroes and repeated coordinates, we arrive at a hiding function $f_{N_r}(A)$, whose shortened coordinate expansion is given by

$$(x_1 + x_2, x_3 + x_4),$$

where $x_1, x_2, x_3, x_4 \in \mathbb{F}_2$. The quantum oracle that performs the unitary operation $U_{N_r} |a\rangle |0\rangle = |a\rangle |f_{N_r}(a)\rangle$, with $a = x_1e_1 + x_2e_2 + x_3e_3 + x_4e_4$, can be seen in Figure 6.4.

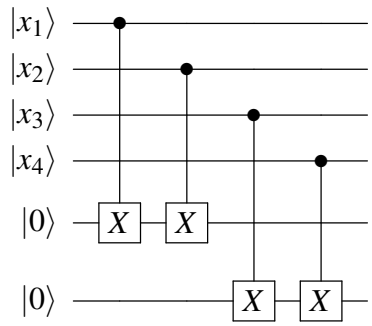


Figure 6.4: Oracle for U_{N_r}, in Example 6.8.

The oracle together with the quantum procedure are given by the code B.12. As in example 6.7, repeatedly using Algorithm 6.5, we can obtain (with high probability) that $N_r(A)^{\perp}$ is generated by $\{(1,1,0,0),(0,0,1,1)\}$. Simulations are in B.13, and the results can be seen in Figure 6.5.

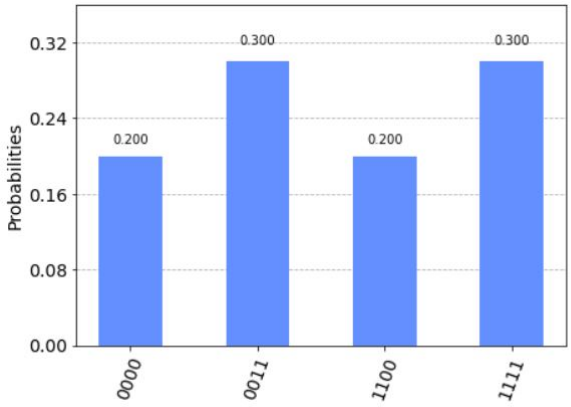


Figure 6.5: Elements in $N_r(A)^{\perp}$, after 10 repetitions of Algorithm 6.5.

Leading us to the system of equations(6.2).

$$\begin{cases} 0x_1 + 0x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ x_1 + x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ 0x_1 + 0x_2 + x_3 + x_4 \equiv 0 \mod 2 \\ x_1 + x_2 + x_3 + x_4 \equiv 0 \mod 2 \end{cases} \tag{6.2}$$

So, by applying Gauss-Jordan, we found the elements of $N_r(A)$, which are

$$\{(0,0,0,0),(1,1,0,0),(0,0,1,1),(1,1,1,1)\}.$$

Thus, $\{(1,1,0,0),(0,0,1,1)\}$ generates $N_r(A)$.

Middle Nucleus. First, we find the expansion of the function $f_{N_m}(A)$, using code B.14, which is actually the same as the right nucleus. Hence, in this particular case, $N_r(A) = N_m(A)$.

Left Nucleus. Let us find the coordinate expansion of the function $f_{N_l}(A)$, using the code B.15. After eliminating repeated coordinates, we arrive at a hiding function $f_{N_l}(A)$, whose shortened coordinate expansion is given by

$$(x_2 + x_3 + x_4, x_1),$$

where $x_1, x_2, x_3, x_4 \in \mathbb{F}_2$. The quantum oracle that performs the unitary operation $U_{N_r} |a\rangle |0\rangle = |a\rangle |f_{N_l}(a)\rangle$, with $a = x_1e_1 + x_2e_2 + x_3e_3 + x_4e_4$, can be seen in Figure 6.6.

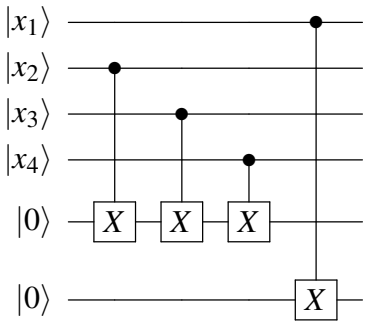


Figure 6.6: Oracle for U_{N_l}, in Example 6.8.

The oracle together with the quantum procedure are given by code B.16 and B.17, respectively. After 10 repetitions of Algorithm 6.5, the probability of not finding elements in the orthogonal $N_l^{\perp}$ is also below 1%. The found elements can be seen in Figure 6.7. Leading us to the system

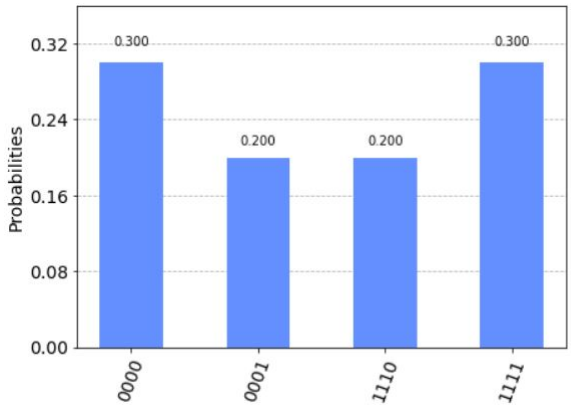


Figure 6.7: Elements in $N_l(A)^{\perp}$, after 10 repetitions of Algorithm 6.5.

of equations(6.3).

$$\begin{cases} 0x_1 + 0x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ x_1 + 0x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ 0x_1 + 1x_2 + x_3 + x_4 \equiv 0 \mod 2 \\ x_1 + x_2 + x_3 + x_4 \equiv 0 \mod 2 \end{cases} \tag{6.3}$$

So, by applying Gauss-Jordan, we found the elements of $N_l(A)$, which are

$$\{(0,0,0,0),(0,1,0,1),(0,1,1,0),(0,0,1,1)\},$$

and so $\{(0,1,0,1),(0,1,1,0)\}$ generates $N_l(A)$.

Nucleus. Knowing the coordinate version of the hiding functions of the right, middle and left nuclei, we can build the oracle of the coordinate version of the shortened hiding function $f_N(A)$, which is

$$(x_1+x_2, x_3+x_4, x_2+x_3+x_4, x_1).$$

The quantum oracle that performs the unitary operation $U_{N_r}|a\rangle|0\rangle = |a\rangle|f_{N_r}(a)\rangle$, with $a = x_1e_1+x_2e_2+x_3e_3+x_4e_4$, can be seen in Figure 6.8.

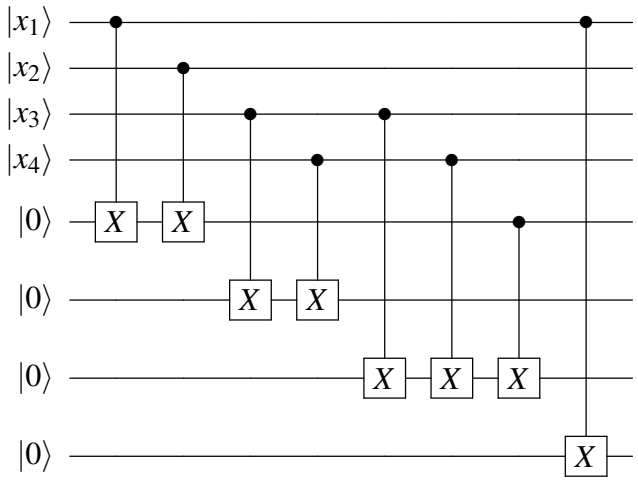


Figure 6.8: Oracle for U_N, in Example 6.8.

The quantum procedure is given by code B.18, and it is simulated with the code B.19. After 30 repetitions of Algorithm 6.5, we find the elements of $N(A)^{\perp}$, with high probability (see Figure 6.9). Leading us to the system of equations(6.4).

$$\begin{cases} 0x_1 + 0x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ x_1 + 0x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ 0x_1 + x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ x_1 + x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\ x_1 + 0x_2 + x_3 + x_4 \equiv 0 \mod 2 \\ 0x_1 + x_2 + x_3 + x_4 \equiv 0 \mod 2 \\ x_1 + x_2 + x_3 + x_4 \equiv 0 \mod 2. \end{cases} \tag{6.4}$$

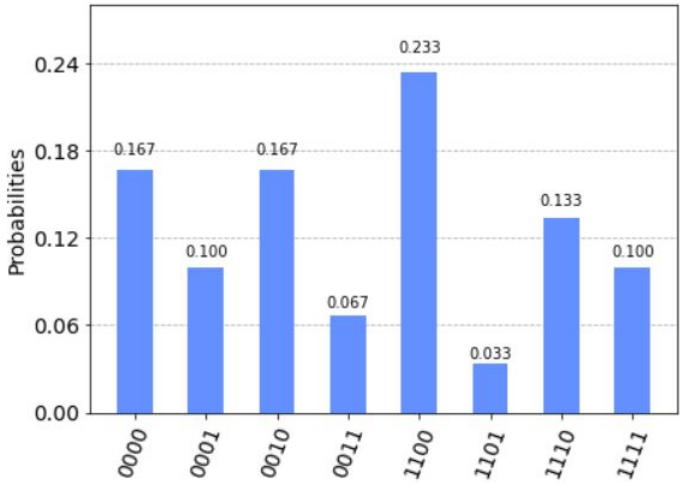


Figure 6.9: Elements in $N(A)^{\perp}$, after 30 repetitions of Algorithm 6.5.

So, by applying Gauss-Jordan, we found that $\{(0,0,1,1)\}$ generates $N(A)$. Which of course coincides with $N_r(A) \cap N_m(A) \cap N_l(A)$.

Center. Note that knowing the coordinate expansion of $f_N(A)$, we just need to find the coordinate expansion of $f_Z(A)$ from the commutators. So, with the code B.20, and after eliminating the coordinates that are always 0, we found its coordinate expansion:

$$(x_1+x_2, x_3+x_4, x_2+x_3+x_4, x_1, x_3, x_2).$$

The quantum oracle that performs the unitary operation $U_Z |a\rangle |0\rangle = |a\rangle |f_Z(a)\rangle$, with $a = x_1e_1 + x_2e_2 + x_3e_3 + x_4e_4$, can be seen in Figure 6.10.
The quantum procedure is given by the code B.21, and it can be simulated with code B.22. After 50 repetitions of Algorithm 6.5, we find the elements of $Z(A)^{\perp}$, with high probability (see Figure 6.11).

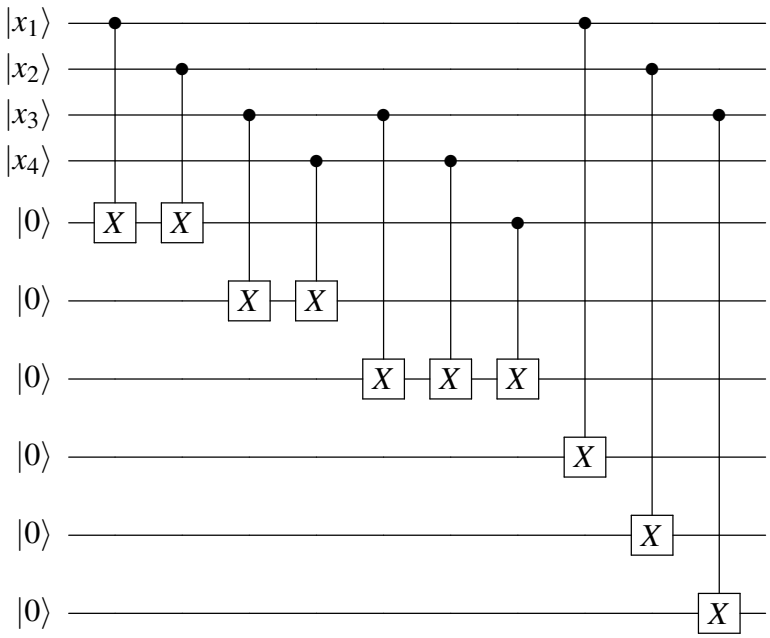


Figure 6.10: Oracle for U_Z, in Example 6.8.

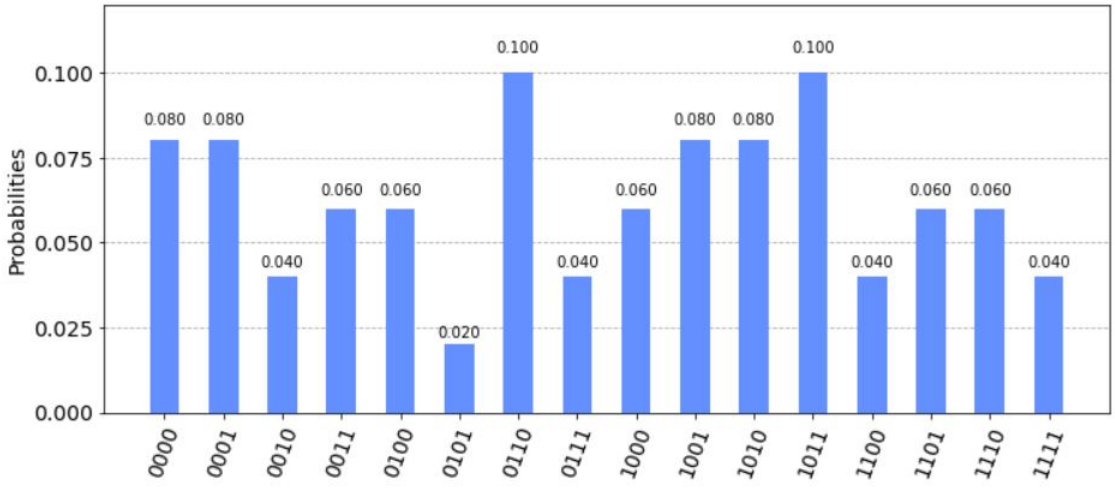


Figure 6.11: Elements on $Z(A)^{\perp}$, after 50 repetitions of Algorithm6.5.

Leading us to the system of equations(6.5).

$$\begin{cases}
0x_1 + 0x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\
x_1 + 0x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\
0x_1 + x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\
x_1 + x_2 + 0x_3 + 0x_4 \equiv 0 \mod 2 \\
0x_1 + 0x_2 + x_3 + 0x_4 \equiv 0 \mod 2 \\
x_1 + 0x_2 + x_3 + 0x_4 \equiv 0 \mod 2 \\
0x_1 + x_2 + x_3 + 0x_4 \equiv 0 \mod 2 \\
x_1 + x_2 + x_3 + 0x_4 \equiv 0 \mod 2 \\
0x_1 + 0x_2 + 0x_3 + x_4 \equiv 0 \mod 2 \\
x_1 + 0x_2 + 0x_3 + x_4 \equiv 0 \mod 2 \\
0x_1 + x_2 + 0x_3 + x_4 \equiv 0 \mod 2 \\
x_1 + x_2 + 0x_3 + x_4 \equiv 0 \mod 2 \\
0x_1 + 0x_2 + x_3 + x_4 \equiv 0 \mod 2 \\
x_1 + 0x_2 + x_3 + x_4 \equiv 0 \mod 2 \\
0x_1 + x_2 + x_3 + x_4 \equiv 0 \mod 2 \\
x_1 + x_2 + x_3 + x_4 \equiv 0 \mod 2.
\end{cases} \tag{6.5}$$

So, by applying Gauss-Jordan we find that, $\{(0,0,0,0)\} = Z(A)$.

Chapter 7

An approach to the Classification of Finite Semifields by Quantum Computing

In this chapter, we address the problem of classification of finite semifields, using quantum computational methods. Following [HR07], and [RCR09], in Proposition 2.52 of this dissertation, it was stated that, any finite semifield D of order q^d can be described by a set of d matrices, known as *standard basis*. So, the effective classification of finite semifields can be rephrased as a problem of finding certain sets of matrices. Hence, our main approach will be based on Grover's Algorithm.

In Section 7.1, we model the problem, and we present the results using a simulator for quantum circuits to find some of the multiplication tables for the finite semifield $\mathbb{F}_8$ (which is the only finite semifield of order 8), and for finite commutative semifields of order 16. Specifically, we build an oracle for finding standard bases of those semifields with Grover's quantum Algorithm. In Section 7.1.3, we give an estimation of the cost for the general case, in terms of quantum gates (for the sake of simplicity, we restrict to binary semifields). Additionally, we draw some conclusions from this approach to the general classification of finite semifields. In Section 7.2, we look for an alternative of quantum computing techniques to study finite semifields: Quantum Annealing, a form of computation that efficiently samples the low-energy configurations of a quantum system [KN98]. So, we give an approach towards finding the multiplication table of the binary primitive finite semifield of order 32.

7.1 Quantum Computational Search of Finite Semifields with Grover's algorithm

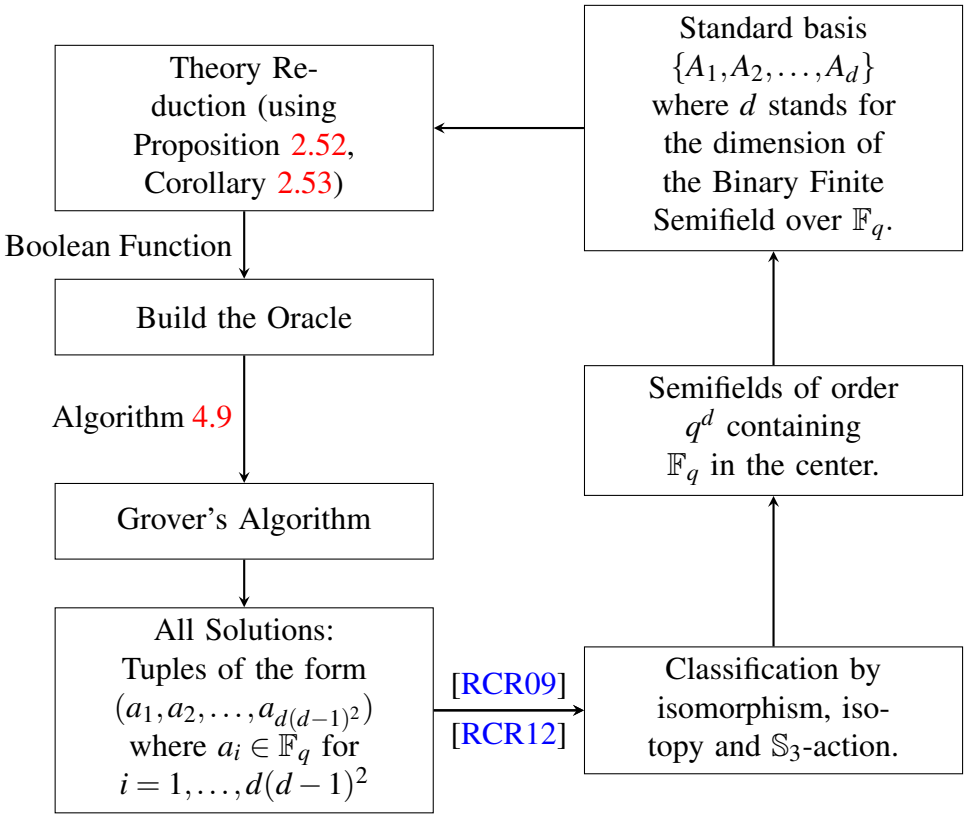
In this section, we introduce a procedure for the classification of finite semifiels using Grover's algorithm.

The multiplication table of a finite semifield D with q^d elements is related to a standard basis, a set of d matrices satisfying the properties of Proposition 2.52 (namely, take $\{x_1 = 1, x_2, \ldots, x_d\}$ a $\mathbb{F}_q-$basis of D, so that the coordinate matrices of the maps L_{x_i}, for $i = 1, \ldots, d$, with respect

to such a basis satisfy those conditions). So, we focus our attention on finding standard bases $\{A_1, A_2, \ldots, A_d\}$. In order to do that, we build up a Boolean function f, using the determinant of each linear combination of the matrices in the standard basis, in terms of the operations of XOR, AND, NOT. Such a Boolean function is used to construct an oracle with Toffoli and CNOT gates, to be used in an implementation of Grover's algorithm.

However, it is important to notice that there are two problems that need to be addressed. First, we must know how many iterations are needed for the algorithm to succeed. Second, after a series of runs of the algorithm, we need to make sure that we are not leaving out any solution (with high probability). To solve those problems, we use Algorithm 4.9. So, we apply Grover's algorithm, which gives tuples that are all the solutions for all the standard bases (with a probability as high as desired). Finally, following [RCR09] and [RCR12], semifields can be classically classified up to isomorphism, isotopy and $\mathbb{S}_3$-action.

In summary, the previous procedure can be seen in the following mind chart.



Note that, in order to apply Algorithm 4.9, we must show first that the number of satisfying assignments of f, is actually less than three quarters of the total of elements of the domain of f.

Proposition 7.1. Among the set of $\nu = q^{d(d-1)^2}$ potential standard bases of finite semifields of order q^d contaning $\mathbb{F}_q$ in the center with identity, there are at most $\mu \leq B := \nu \left(1 - \frac{1}{q}\right)^{d-1}$

standard bases actually corresponding to finite semifields. In particular, when $q = 2, d \geq 3$, there are at most $\frac{3\nu}{4}$ potential bases corresponding to binary semifields.

Proof. Consider the set $\{A_1, \ldots, A_d, \}$ of coordinate matrices of L_{x_i}, with $i = 1, \ldots, d$, where $\{x_1 = 1, x_2, \ldots, x_d\}$ is a $\mathbb{F}_q$ basis of a $d-$dimensional algebra. Each matrix A_i, for $i = 2, \ldots, d$, has a submatrix of size $d \times (d-1)$, in which each entry belongs to $\mathbb{F}_q$. Any choice of those $d(d-1)^2$ entries yields a potential standard bases of a finite semifield containing $\mathbb{F}_q$ in the center. Therefore, $\nu = q^{d(d-1)^2}$.

Now, let us give an upper bound for μ. So, recall that all A_i, for $i = 2, \ldots, d-1$, must be invertible. For each A_i, we will have

$$\prod_{i=1}^{d-1} \left(q^d - q^i\right)^{d-1}$$

choices. Now,

$$\begin{aligned}
\prod_{i=1}^{d-1} \left(q^d - q^i\right)^{d-1} &= \prod_{i=1}^{d-2} \left(q^d - q^i\right)^{d-1} \left(q^d - q^{d-1}\right)^{d-1} \\
&\leq q^{d(d-2)(d-1)} q^{d(d-1)} \left(1 - \frac{1}{q}\right)^{d-1} \\
&\leq q^{d(d-1)^2} \left(1 - \frac{1}{q}\right)^{d-1}.
\end{aligned}$$

□

Therefore, in the binary case, the conditions of Algorithm 4.9 are satisfied. Thus, we can apply it, in order to find the multiplication tables of binary semifields. Note that, by section 4.5, the worst case of number of oracle queries is $O\left(\sqrt{\nu} B \log\left(\frac{3}{4}\nu\right)\right)$.

7.1.1 Semifield of Order 8

In this subsection, we will apply Grover's algorithm to the problem of classification of finite semifields. We consider the small cases of orders 8 and 16 (commutative). For these cases, we explicitly construct a boolean function that would become the oracle to be used on Grover's algorithm to the problem of classification of finite semifields.

Consider the following standard basis:

$$\left\{ A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} 0 & a_1 & a_4 \\ 1 & a_2 & a_5 \\ 0 & a_3 & a_6 \end{pmatrix}, A_3 = \begin{pmatrix} 0 & a_7 & a_{10} \\ 0 & a_8 & a_{11} \\ 1 & a_9 & a_{12} \end{pmatrix} \right\}.$$

If we want to classify finite semifields of order 8 by Proposition 2.52, we need to find $a_1,\dots,a_{12} \in \mathbb{F}_2$ such that each linear combination of the basis, except the trivial one, yields a matrix with determinant different from zero, i.e.,

$$\det(\alpha_1 A_1 + \alpha_2 A_2 + \alpha_3 A_3) = 1,$$

for each nonzero tuples $(\alpha_1,\alpha_2,\alpha_3) \in \mathbb{F}_2^3$. Now, in one hand, the determinant of a matrix is based on multiplications and additions. On the other hand, the product mod 2 is an AND, and the addition or subtraction mod 2 is an exclusive OR (XOR). Hence, we rewrite each determinant in terms of AND, XOR and NOT, and create a boolean function from them. Namely,

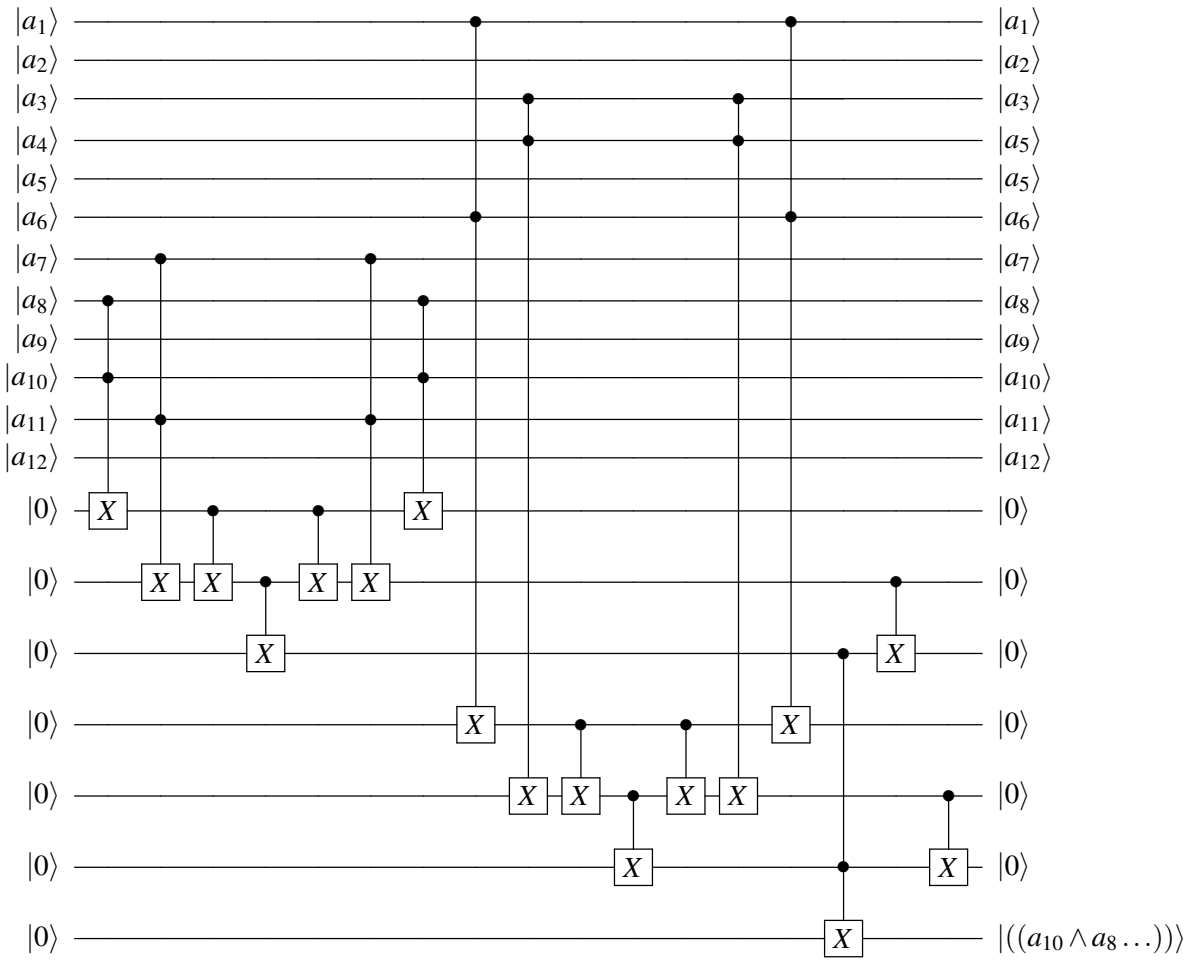
$$\begin{aligned} f : \{0,1\}^{12} &\longrightarrow \{0,1\} \\ (a_1,\dots,a_{12}) &\longmapsto f(a_1,\dots,a_{12}), \end{aligned}$$

such that $f(a_1,\dots,a_{12}) = 1$ if and only if the corresponding set of matrices satisfies the above-mentioned conditions. Specifically, using the python code in C.23, we find that

$$\begin{aligned} f(a_1,\dots,a_{12}) =& (a_{10} \wedge a_8 \oplus a_{11} \wedge a_7) \wedge (a_1 \wedge a_6 \oplus a_3 \wedge a_4) \wedge (a_1 \wedge a_{11} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_5 \\ & \oplus a_1 \wedge a_6 \oplus a_{10} \wedge a_2 \oplus a_{10} \wedge a_3 \oplus a_{10} \wedge a_8 \oplus a_{10} \wedge a_9 \oplus a_{11} \wedge a_7 \oplus a_{12} \wedge a_7 \\ & \oplus a_2 \wedge a_4 \oplus a_3 \wedge a_4 \oplus a_4 \wedge a_8 \oplus a_4 \wedge a_9 \oplus a_5 \wedge a_7 \oplus a_6 \wedge a_7) \wedge (\sim a_{10} \oplus a_{12} \\ & \oplus a_8 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_7 \oplus a_{11} \wedge a_9 \oplus a_{12} \wedge a_8) \wedge (\sim a_1 \oplus a_2 \oplus a_6 \oplus a_1 \\ & \wedge a_6 \oplus a_2 \wedge a_6 \oplus a_3 \wedge a_4 \oplus a_3 \wedge a_5) \wedge (\sim a_1 \oplus a_{10} \oplus a_{12} \oplus a_2 \oplus a_4 \oplus a_6 \\ & \oplus a_7 \oplus a_8 \oplus a_1 \wedge a_{11} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_5 \oplus a_1 \wedge a_6 \oplus a_{10} \wedge a_2 \oplus a_{10} \wedge a_3 \\ & \oplus a_{10} \wedge a_8 \oplus a_{10} \wedge a_9 \oplus a_{11} \wedge a_3 \oplus a_{11} \wedge a_7 \oplus a_{11} \wedge a_9 \oplus a_{12} \wedge a_2 \oplus a_{12} \\ & \wedge a_7 \oplus a_{12} \wedge a_8 \oplus a_2 \wedge a_4 \oplus a_2 \wedge a_6 \oplus a_3 \wedge a_4 \oplus a_3 \wedge a_5 \oplus a_4 \wedge a_8 \oplus a_4 \wedge a_9 \\ & \oplus a_5 \wedge a_7 \oplus a_5 \wedge a_9 \oplus a_6 \wedge a_7 \oplus a_6 \wedge a_8). \end{aligned}$$

where $\oplus$ stands for XOR, $\wedge$ for AND, $\sim$ for NOT. Note that this expression for f has 57 gates AND, 60 XOR gates, and 3 NOT gates. This function is to be used in an oracle in order to perform Grover's Algorithm, as stated in the previous section. The quantum circuit can be build in the same way as in Examples 3.8 and 3.9. Now, because of its length, we give the circuit for

$((a_{10} \wedge a_8) \oplus (a_{11} \wedge a_7)) \wedge ((a_1 \wedge a_6) \oplus (a_3 \wedge a_4))$ (the rest follows the same lines)



The whole circuit can be seen with code C.24 from appendix C. Notice that at least 61 ancillary qubits are required. An alternative is using qiskit tools. For instance, for the grover oracle function C.25, we could write down its quantum circuit, and use it as an oracle for Grover's algorithm. Now that we have already built the oracle, we apply Algorithm 4.9 from Section 4.5, (which can be found in C.26), and we find eight solutions.

In fact, for a probability of failure $w = 0.00001$, we have

$$R = \left\lceil \frac{\log\left(1-(1-0.01)^{\frac{1}{3072}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil = 68,$$

and we find fourteen solutions, as it can be seen in the following table:

Table 7.1: Results for the multiplication table of $\mathbb{F}_8$.

Iteration	**Valid Solution**	**Solution Found**	**Corresponding j**
1	no	000111110001	4
2	yes	111101101100	47
3	no	010100011100	31
4	no	000010101111	10
5	no	111011001100	45
6	no	111011010111	54
7	no	110000011000	24
8	no	110110100011	9
9	no	010101110100	5
10	no	100110001001	40
11	no	101101001011	16
12	no	101010011001	20
13	no	110010011101	25
14	no	011100100011	7
15	no	010100011111	20
16	no	100001010001	61
17	no	110011000011	58
18	no	001000011000	51
19	no	001101010110	5
20	no	011001001000	21
21	no	010000011000	30
22	no	001000001111	0
23	no	111000110001	39
24	no	100011101111	0
25	no	000100111000	4
26	no	111100010111	1
27	no	100011111110	8
28	no	100001100101	23
29	no	011011101010	6
30	no	110110010111	1
31	yes	101110101011	6
32	no	011000011110	61
33	yes	010011011111	51
34	yes	110011011100	48
35	yes	010101101110	27
36	yes	011110110111	60
37	no	111010010000	6
38	yes	011001001110	26
39	yes	111110110101	29
40	no	000100000001	51

41	no	010100011100	10
42	no	110010010011	9
43	no	010111101010	42
44	no	111100100110	32
45	no	110000000010	24
46	no	011100100000	48
47	no	011100000000	14
48	no	000001000011	16
49	no	000000000001	14
50	no	100000111000	61
51	no	000100011010	7
52	no	101001010010	1
53	no	110101011001	8
54	no	011111101001	32
55	no	100000100000	13
56	no	011100000000	17
57	no	010000110111	57
58	no	101001001101	18
59	no	101110001001	12
60	no	001011101000	30
61	no	110111110110	42
62	no	101000110111	15
63	no	010110011110	15
64	no	101010110000	40
65	no	010011011000	54
66	no	100001101101	39
67	no	010111000011	37
68	no	101001010101	56
69	no	111011111000	38
70	no	111011000001	27
71	no	011100001110	26
72	no	110110101011	32
73	no	110011111010	34
74	no	010000101101	52
75	no	111100100000	34
76	no	101011010111	57
77	no	010101010100	16
78	no	101110000000	34
79	no	101000111000	18
80	no	101011010110	1
81	no	011001001101	27

82	no	010001111000	40
83	no	111011110000	4
84	no	111111011000	9
85	no	001011011101	56
86	no	110010101001	58
87	no	001001000100	33
88	no	011100111011	3
89	no	010110011010	47
90	no	111111101111	37
91	no	011111011000	54
92	no	011111001111	33
93	no	110111110001	47
94	no	110011011001	27
95	no	000011001010	59
96	no	011110010000	25
97	no	011011011011	23
98	no	000101100010	49
99	no	011001101101	0
100	no	000111000110	3
101	no	010010111111	51
102	no	100011010100	17
103	no	101010111111	15
104	no	111110001010	1
105	no	110101100001	46
106	no	000011010011	26
107	no	000010101111	5
108	no	011000000001	0
109	no	000110001110	29
110	no	011000010011	3
111	no	110101000010	28
112	no	001011100010	7
113	no	011111101100	51
114	no	001001111101	26
115	no	111101110000	39
116	no	011010101000	2
117	no	010000101111	50
118	no	101110000110	9
119	no	000010111011	38
120	no	010101000101	20
121	no	100001000110	12
122	no	000100110110	61

123	no	010010111011	0
124	no	110100000011	23
125	no	110111001111	56
126	no	011100011110	26
127	no	010100101110	36
128	no	011001010011	37
129	no	011101011000	8
130	no	111100000101	16
131	no	001110110100	1
132	no	010001101001	5
133	no	101101100100	57
134	no	000101110000	61
135	no	001000110001	32
136	no	111111101001	15

The solutions, for instance, the first one, 110011011100, must be read from right to left, i.e.,

$$a_1 = 0, a_2 = 0, a_3 = 1, a_4 = 1, a_5 = 0, a_6 = 1, a_7 = 1, a_8 = 0, a_9 = 1, a_{10} = 1, a_{11} = 1, a_{12} = 1.$$

Which means that the multiplication table is

$$\left\{ A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}, A_3 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right\}.$$

The same applies to each result. Note that all the results satisfy the condition that, for all i, j the i-th column of A_j is the j-th column of A_i. This means that the corresponding finite semifield is commutative. Actually, all of them provide standard bases of the only finite semifield of order 8: the Galois field $\mathbb{F}_8$.

Observe that the solutions are found in the very first iterations. The overall number of iterations is bigger to ensure that no solutions are missed with the desired probability.

7.1.2 Description of Semifields of Order 16

Now, let us move to the case of semifields of order 16, where the corresponding standard bases contain binary matrices of size 4×4. Consider the following set $\{A_1, A_2, A_3, A_4\}$, where

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0 & a_1 & a_5 & a_9 \\ 1 & a_2 & a_6 & a_{10} \\ 0 & a_3 & a_7 & a_{11} \\ 0 & a_4 & a_8 & a_{12} \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0 & a_{13} & a_{17} & a_{21} \\ 0 & a_{14} & a_{18} & a_{22} \\ 1 & a_{15} & a_{19} & a_{23} \\ 0 & a_{16} & a_{20} & a_{24} \end{pmatrix} \quad A_4 = \begin{pmatrix} 0 & a_{25} & a_{29} & a_{33} \\ 0 & a_{26} & a_{30} & a_{34} \\ 0 & a_{27} & a_{31} & a_{35} \\ 1 & a_{28} & a_{32} & a_{36} \end{pmatrix}.$$

If we carry on the same track as before, we end up with an expression for a Boolean function $f : \{0,1\}^{36} \to \{0,1\}$, with 4189 AND, 2314 XOR, and 7 NOT gates, which make it really expensive writing its circuit. As proof of concept, we restrict ourselves to the commutative case (which incidentally is interesting in general, because of what has been studied and mentioned in [LS23]). Thus, we consider that for all i, j the i-th column A_j is the j-th column of A_i.

In order to further reduce the number of variables, we will take A_2 among a prefixed set of matrices, according to the following theoretical reduction. From Corollary 2.53, for any fixed non-scalar (i.e., not 0 or 1) element b in the semifield, the matrix A_b of left multiplication by b has a characteristic polynomial without linear factors. So, there are 4 possibles characteristic polynomials for the matrix A_2:

$$x^4+x^3+x^2+x+1, x^4+x^3+1, x^4+x+1, x^4+x^2+1 = (x^2+x+1)^2.$$

Because $x^4+x^3+x^2+x+1 = (x+1)^4+(x+1)^3+1$, we can change the element b by $b+1$, and assume that there are 3 possibilities: $x^4+x^3+1, x^4+x+1, x^4+x^2+1 = (x^2+x+1)^2$.

If the elements in $\beta = \{1, b, b^{(2}, b^{(3}\}$ are linearly independent, we can change to basis β to get A_2 in the form of a companion matrix. If not, which can only happen with the third polynomial, then a basis $\beta = \{1, b, c, bc\}$ can be chosen so that A_2 has the form

$$\begin{pmatrix} C(x^2+x+1) & \\ & C(x^2+x+1) \end{pmatrix}.$$

Hence, we need to find binary values $a_1, a_2, \ldots, a_{12}$, such that each linear combination of the matrix in the standard basis, except for the trivial one, yields a matrix with determinant different from zero. The second matrix can be chosen among the following ones:

1. **Case 1:** Characterisitc Polynomial x^4+x+1, $A_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$.

   So, $\{A_1, A_2, A_3, A_4\}$ would become

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0 & 0 & a_1 & a_5 \\ 0 & 0 & a_2 & a_6 \\ 1 & 0 & a_3 & a_7 \\ 0 & 1 & a_4 & a_8 \end{pmatrix} \quad A_4 = \begin{pmatrix} 0 & 1 & a_5 & a_9 \\ 0 & 1 & a_6 & a_{10} \\ 0 & 0 & a_7 & a_{11} \\ 1 & 0 & a_8 & a_{12} \end{pmatrix}.$$

   And its Boolean function $f_1 : \{0,1\}^{12} \to \{0,1\}$ can be found with code C.27 (it is explicitly written after it). Such a Boolean Function has 141 AND, 156 XOR, and 5 NOT gates.

We found only one solution, after applying algorithm 4.9, with a probability of failure $w = 0.00001$, so

$$R = \left\lceil \frac{\log\left(1-(1-0.01)^{\frac{1}{3072}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil = 68.$$

See the following table:

Table 7.2: Results for the multiplication tables of commutative semifields of order 16, case 1.

Iteration	**Valid Solution**	**Solution Found**	**Corresponding j**
1	yes	110001100011	50
2	no	010111011010	19
3	no	001001010110	2
4	no	000010010010	28
5	no	010001001000	41
6	no	000101011011	10
7	no	100011110101	23
8	no	000001011101	26
9	no	010001111010	59
10	no	111100110101	60
11	no	000001011011	48
12	no	010000010111	18
13	no	100010011010	4
14	no	001001001100	21
15	no	001111011100	8
16	no	101000110111	27
17	no	110111110000	44
18	no	000011011011	58
19	no	100000001101	21
20	no	100001001111	17
21	no	010101110000	39
22	no	001000010110	52
23	no	010100001000	8
24	no	011010111010	34
25	no	001100111100	49
26	no	101010101011	1
27	no	011101111111	10
28	no	111100101110	52
29	no	001110100100	53
30	no	010010110011	53

31	no	001001100110	46
32	no	101011110111	42
33	no	100110111010	14
34	no	010101111001	23
35	no	111000001111	48
36	no	011000111000	12
37	no	000111010010	35
38	no	011101000110	54
39	no	110011111001	58
40	no	100000011101	38
41	no	101100001101	42
42	no	110111110000	31
43	no	001111001101	60
44	no	001011010110	46
45	no	110000111111	56
46	no	011000011001	21
47	no	101011001110	42
48	no	011101111001	62
49	no	101000001111	27
50	no	011110000111	18
51	no	011001000100	29
52	no	100011000100	23
53	no	100011001111	38
54	no	100011111001	54
55	no	111010111111	55
56	no	010101001110	10
57	no	010011011110	2
58	no	111100110000	37
59	no	000000011100	29
60	no	110111001101	5
61	no	100000011010	39
62	no	100010110001	11
63	no	100010100001	36
64	no	111010011001	9
65	no	001111101101	41
66	no	111011001110	58
67	no	111000010101	48
68	no	000100100010	49
69	no	101101110001	19
70	no	010110110111	23
71	no	100010010100	50

72	no	000010100101	20
73	no	100011100100	59
74	no	010101001100	7
75	no	000000101110	21
76	no	011010110011	52
77	no	101001000011	22
78	no	001000110100	6
79	no	100010110110	12
80	no	000010001110	0
81	no	000111011100	40
82	no	100111100010	60
83	no	110101011001	20
84	no	101001111111	56
85	no	110011110011	49
86	no	101101001100	13
87	no	001110010101	16
88	no	101110001100	33
89	no	010000111010	7
90	no	000001001001	18
91	no	011000010010	56
92	no	011000000011	55
93	no	100111000111	32
94	no	111001011101	30
95	no	000001001010	14
96	no	110100000110	51
97	no	011100111110	41
98	no	011001011111	42
99	no	100011110010	1
100	no	001010111011	14
101	no	100010100111	54
102	no	011101000111	22
103	no	101000010001	61
104	no	110110110010	12
105	no	100000011001	40
106	no	010110101010	21
107	no	101011001001	59
108	no	110011010001	34
109	no	000001001000	51
110	no	001100100101	44
111	no	001010001010	6
112	no	111001101111	22

113	no	011111010100	61
114	no	000101101000	34
115	no	110010111101	49
116	no	000000010011	50
117	no	100000110110	27
118	no	111101001101	24
119	no	100111111101	21
120	no	001001011001	50
121	no	001011111000	32
122	no	101011111001	6
123	no	010001000010	5
124	no	101001001010	20
125	no	110010000010	49
126	no	110010000010	50
127	no	010011111000	34
128	no	001100001100	52
129	no	011101011010	47
130	no	100111011101	31
131	no	010110100100	45
132	no	101110001100	12
133	no	011110000100	42
134	no	110010110001	21
135	no	011000110101	10
136	no	001111011001	61

Analogously as before, we should read the result from left to right. So, for this case,

$$a_1=1, a_2=1, a_3=0, a_4=0, a_5=0, a_6=1, a_7=1, a_8=0, a_9=0, a_{10}=0, a_{11}=1, a_{12}=1.$$

Thus, the multiplication table is

$$\left\{ A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, A_3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}, A_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \right\}.$$

2. **Case 2:** Characteristic polynomial x^4+x^3+1, $A_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$.

So, $\{A_1, A_2, A_3, A_4\}$ would become

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0 & 0 & a_1 & a_5 \\ 0 & 0 & a_2 & a_6 \\ 1 & 0 & a_3 & a_7 \\ 0 & 1 & a_4 & a_8 \end{pmatrix} \quad A_4 = \begin{pmatrix} 0 & 1 & a_5 & a_9 \\ 0 & 0 & a_6 & a_{10} \\ 0 & 0 & a_7 & a_{11} \\ 1 & 1 & a_8 & a_{12} \end{pmatrix}.$$

So, changing A_1, A_2, A_3, A_4 in C.27,

```
A1 = Matrix([[1, 0, 0, 0],[0, 1, 0 ,0],[0, 0, 1 , 0],[0,0,0,1] ])
A2 = Matrix([[0, 0, 0, 1],[1, 0, 0 ,0],[0, 1, 0 , 0],[0,0,1,1] ])
A3 = Matrix([[0, 0, a[1], a[5]],[0, 0, a[2] ,a[6]],[1, 0, a[3] ,
 ↪  a[7]],[0,1,a[4],a[8]]])
A4 = Matrix([[0, 1, a[5], a[9]],[0, 0, a[6] ,a[10]],[0, 0, a[7] ,
 ↪  a[11]],[1,1,a[8],a[12]] ])
```

we found its respective Boolean function f_2, which can be seen in appendix C. The complexity of the Boolean Function f_2 is 149 AND, 174 XOR, and 9 NOT gates. As above, we found only one solution, after applying algorithm 4.9 with a probability of failure $w = 0.00001$, so

$$R = \left\lceil \frac{\log\left(1-(1-0.01)^{\frac{1}{3072}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil = 68.$$

See the following table:

Table 7.3: Results for the multiplication tables of commutative semifields of order 16, case 2.

Iteration	Valid Solution	Solution Found	Corresponding j
1	yes	111110111001	28
2	no	011101101011	50
3	no	110001000000	41
4	no	011011100110	23
5	no	111011000000	57
6	no	100011010001	10
7	no	101001101000	10
8	no	010001100100	46
9	no	110100110101	14

11	no	001110101001	26
12	no	001001010100	59
13	no	010111101000	56
14	no	010111100110	37
15	no	100111001011	42
16	no	111100011000	38
17	no	111001000010	22
18	no	100010111100	57
19	no	000001100001	52
20	no	111011011011	0
21	no	100001000110	4
22	no	000001110111	34
23	no	001001111110	40
24	no	101111010101	46
25	no	101000110100	42
26	no	010001001110	50
27	no	110101111000	45
28	no	100100101111	37
29	no	111110001101	29
30	no	110100100010	7
31	no	010011110011	38
32	no	010111110110	30
33	no	111111100100	52
34	no	110110000110	17
35	no	011111110111	43
36	no	000100110110	53
37	no	111011101000	44
38	no	110110001011	30
39	no	110111111101	22
40	no	010000100001	13
41	no	011111100001	6
42	no	000010100110	44
43	no	011101111101	10
44	no	010111100111	23
45	no	110010101100	5
46	no	010111010110	12
47	no	010111001000	26
48	no	010100000001	56
49	no	101111011101	11
50	no	000110011111	47
51	no	011000110001	55

52	no	000100100011	52
53	no	101101000111	59
54	no	110100010000	45
55	no	111110001000	22
56	no	100000010110	30
57	no	011101001101	29
58	no	101001011000	58
59	no	111111100001	7
60	no	010111100110	14
61	no	011000110000	46
62	no	110010001000	61
63	no	100111101011	7
64	no	101001100000	28
65	no	011011110111	38
66	no	001011110000	7
67	no	001000000000	35
68	no	110001110101	43
69	no	110010000010	3
70	no	111010010111	5
71	no	111001100100	23
72	no	100001110001	35
73	no	101011000111	19
74	no	110010000000	47
75	no	000011001110	34
76	no	110011111110	58
77	no	100111000011	39
78	no	101110110001	26
79	no	011000101111	3
80	no	011110111101	34
81	no	010111000001	56
82	no	110111111000	59
83	no	101101000000	62
84	no	000101110010	54
85	no	100010000011	11
86	no	100011001100	22
87	no	111000100001	50
88	no	010001000110	41
89	no	111001110100	6
90	no	100000100111	18
91	no	110110110000	29
92	no	001010110101	17

93	no	100101010100	21
94	no	100011110100	35
95	no	011000111100	12
96	no	011000111001	14
97	no	111000000111	32
98	no	001110000111	44
99	no	111101001001	33
100	no	001101110001	14
101	no	011000010000	30
102	no	011110011101	1
103	no	101010010010	4
104	no	001011101001	51
105	no	100011000111	27
106	no	100110101000	18
107	no	100111011010	31
108	no	011001111000	15
109	no	100001011110	49
110	no	011000011011	61
111	no	100010001100	2
112	no	000100001011	0
113	no	000010111001	55
114	no	101010110000	48
115	no	001111110011	61
116	no	001011100001	54
117	no	001100011011	23
118	no	101001010000	24
119	no	101010110110	29
120	no	010100010100	22
121	no	010000111110	16
122	no	111011111111	37
123	no	000100011101	5
124	no	110100000010	38
125	no	101101111011	53
126	no	001011101110	9
127	no	101001100101	2
128	no	110111110000	49
130	no	010001011000	55
131	no	001101101101	55
132	no	101110101100	57
133	no	110101110001	8
134	no	111011101000	12

135	no	001011001110	21
136	no	000001111110	56
137	no	110010001001	7
138	no	110100110100	60

So,

$a_1=1, a_2=0, a_3=0, a_4=1, a_5=1, a_6=1, a_7=0, a_8=1, a_9=1, a_{10}=1, a_{11}=1, a_{12}=1.$

So, the multiplication table is

$$\left\{A_1=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}, A_2=\begin{pmatrix}0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&1\end{pmatrix}, A_3=\begin{pmatrix}0&0&1&1\\0&0&0&1\\1&0&0&0\\0&1&1&1\end{pmatrix}, A_4=\begin{pmatrix}0&1&1&1\\0&0&1&1\\0&0&0&1\\1&1&1&1\end{pmatrix}\right\}.$$

3. **Case 3:** Characteristic polynomial $(x^2+x+1)^2$ $A_2=\begin{pmatrix}0&1&0&0\\1&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}$.

So, $\{A_1, A_2, A_3, A_4\}$ would become,

$$A_1=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} \quad A_2=\begin{pmatrix}0&1&0&0\\1&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}$$

$$A_3=\begin{pmatrix}0&0&a_1&a_5\\0&0&a_2&a_6\\1&0&a_3&a_7\\0&1&a_4&a_8\end{pmatrix} \quad A_4=\begin{pmatrix}0&0&a_5&a_9\\0&0&a_6&a_{10}\\0&1&a_7&a_{11}\\1&1&a_8&a_{12}\end{pmatrix}.$$

The Boolean Function f_4 is found by changing A_1, A_2, A_3, A_4 by

```
C1 = Matrix([[1, 0, 0, 0],[0, 1, 0 ,0],[0, 0, 1 , 0],[0,0,0,1] ])
C2 = Matrix([[0, 1, 0, 0],[1, 1, 0 ,0],[0, 0, 0 , 1],[0,0,1,1] ])
C3 = Matrix([[0, 0, a[1], a[5]],[0,0, a[2] ,a[6]],[1, 0, a[3] ,
 ↪  a[7]],[0,1,a[4],a[8]] ])
C4 = Matrix([[0, 0, a[5], a[9]],[0, 0, a[6] ,a[10]],[0, 1, a[7],
 ↪  a[11]],[1,1,a[8],a[12]] ])
```

in code C.27 (it is given in appendix C). The Boolean function f_3 has 161 AND, 194 XOR, and 9 NOT gates. We found seven solutions, after applying algorithm 4.9 with a

probability of failure $w = 0.00001$, so

$$R = \left\lceil \frac{\log\left(1-(1-0.01)^{\frac{1}{3072}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil = 68.$$

See the following table:

Table 7.4: Results for the multiplication tables of commutative semifields of order 16, case 3.

Iteration	**Valid Solution**	**Solution Found**	**Corresponding j**
1	yes	101001011111	40
2	no	111110111101	44
3	yes	101101101101	53
4	no	001001011011	40
5	no	110111101011	2
6	no	111101101101	12
7	no	001111000011	27
8	no	010101101011	43
9	no	110111101011	5
10	no	100011011100	48
11	no	111001111100	24
12	no	101000111101	32
13	no	001100111000	55
14	no	111100110111	7
15	yes	110110110110	14
16	no	101110011011	51
17	no	101000111101	35
18	no	010111111011	49
19	no	111101010110	46
20	no	011001101010	8
21	no	010101101011	12
22	no	111000100011	49
23	no	101001011011	37
24	no	001001011011	61
25	yes	010111111010	33
26	no	010101101011	8
27	no	111110011010	5
28	no	010010000101	55
29	no	001110100100	53
30	no	110111101011	11

31	no	111101100001	38
32	no	111111111011	17
33	yes	011111101001	13
34	no	110101011111	6
35	no	111111010000	54
36	yes	111010010111	47
37	no	101000111101	2
38	no	101011010101	23
39	no	110111011001	3
40	no	111110100101	62
41	no	010111000111	3
42	no	001100011001	16
43	no	111001000000	58
44	no	110101101011	57
45	no	111100100000	30
46	no	001111011110	59
47	yes	010111110010	58
48	no	001000001011	21
49	no	100010100110	36
50	no	000010001001	35
51	no	111000001111	34
52	no	000100110001	13
53	no	110111001000	48
54	no	011110000000	13
55	no	001110100111	1
56	no	111111110110	29
57	no	001111111010	12
58	no	011101111001	15
59	no	100101001010	2
60	no	010011100001	11
61	no	101111100110	49
62	no	100010101110	40
63	no	001111110000	28
64	no	101110110011	52
65	no	001001011011	26
66	no	001001011011	44
67	no	000001010001	54
68	no	100000001000	54
69	no	110001000100	45
70	no	111010010101	47
71	no	000110001110	32

72	no	011100100101	21
73	no	100010101101	17
74	no	001001110100	47
75	no	000001100011	45
76	no	111000101001	52
77	no	111101110101	48
78	no	101010100101	45
79	no	000000111110	55
80	no	101101111101	26
81	no	000110100000	44
82	no	000110110110	29
83	no	001100010111	62
84	no	010101110111	14
85	no	111000111001	0
86	no	110111100001	25
87	no	110111101011	55
88	no	011101101001	46
89	no	101111000001	47
90	no	011110000011	60
91	no	011100010101	10
92	no	100010101110	18
93	no	111000111001	9
94	no	101011000100	41
95	no	110110110001	60
96	no	100010101000	45
97	no	100001110000	20
98	no	110101000011	27
99	no	011001110000	30
100	no	100000011111	34
101	no	000010001000	54
102	no	100110001111	57
103	no	101101101111	33
104	no	111111101010	35
105	no	100010000110	43
106	no	001011101110	42
107	no	110001100111	48
108	no	111110011010	20
109	no	010000110000	18
110	no	010100011100	57
111	no	011101100101	24
112	no	000001100111	12

113	no	100000111111	51
114	no	000001011111	45
115	no	001101111010	46
116	no	111001010001	11
117	no	100101000010	18
118	no	100010111110	33
119	no	110111111011	25
120	no	000000011010	38
121	no	111001001010	57
122	no	101000010111	31
123	no	011100111010	44
124	no	100100110001	23
125	no	100001101010	16
126	no	010001001011	11
127	no	000011010001	8
128	no	111000001111	23
129	no	000100111010	12
130	no	111110011011	18
131	no	110111101011	25
132	no	101001011011	53
133	no	010101101011	35
134	no	111000001001	56
135	no	000010110100	56
136	no	001100011010	42

Finally, for this case, the first solution is

$$a_1=1,a_2=1,a_3=1,a_4=1,a_5=1,a_6=0,a_7=1,a_8=0,a_9=0,a_{10}=1,a_{11}=0,a_{12}=1.$$

Thus, one of the multiplication tables is:

$$\left\{A_1=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},A_2=\begin{pmatrix}0&1&0&0\\1&1&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix},A_3=\begin{pmatrix}0&0&1&1\\0&0&1&0\\1&0&1&1\\0&1&1&0\end{pmatrix},A_4=\begin{pmatrix}0&0&1&0\\0&0&0&1\\0&1&1&0\\1&1&0&1\end{pmatrix}\right\}.$$

In all of the first three cases, the only semifield found is the finite field, as this is the only commutative finite semifield of order 16 that exists.

4. **Case 4:** Characteristic polynomial x^4+x^2+1, $A_2=\begin{pmatrix}0&0&0&1\\1&0&0&0\\0&1&0&1\\0&0&1&0\end{pmatrix}$.

So, $\{A_1, A_2, A_3, A_4\}$ would become,

$$A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0 & 0 & a_1 & a_5 \\ 0 & 0 & a_2 & a_6 \\ 1 & 0 & a_3 & a_7 \\ 0 & 1 & a_4 & a_8 \end{pmatrix} \quad A_4 = \begin{pmatrix} 0 & 1 & a_5 & a_9 \\ 0 & 0 & a_6 & a_{10} \\ 0 & 1 & a_7 & a_{11} \\ 1 & 0 & a_8 & a_{12} \end{pmatrix}.$$

The Boolean Function f_4 is found by changing A_1, A_2, A_3, A_4 by

```
A1 = Matrix([[1, 0, 0, 0],[0, 1, 0 ,0],[0, 0, 1 , 0],[0,0,0,1] ])
A2 = Matrix([[0, 0, 0, 1],[1, 0, 0 ,0],[0, 1, 0 , 1],[0,0,1,0] ])
A3 = Matrix([[0, 0, a[1], a[5]],[0, 0, a[2] ,a[6]],[1, 0, a[3] ,
 ↪  a[7]],[0,1,a[4],a[8]] ])
A4 = Matrix([[0, 1, a[5], a[9]],[0, 0, a[6] ,a[10]],[0, 1, a[7] ,
 ↪  a[11]],[1,0,a[8],a[12]] ])
```

in code C.27 (it is given in appendix C). The Boolean function f_4 has 137 AND, 162 XOR, and 9 NOT gates. After applying algorithm 4.9 with a probability of failure $w = 0.00001$, so

$$R = \left\lceil \frac{\log\left(1-(1-0.01)^{\frac{1}{3072}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil = 68,$$

we find no solutions. See the following table:

Table 7.5: Results for the multiplication tables of commutative semifields of order 16, case 4.

Iteration	Valid Solution	Solution Found	Corresponding j
1	no	001000100110	61
2	no	110111000100	56
3	no	101011110101	1
4	no	011111010110	45
5	no	101111100100	4
6	no	011010010101	34
7	no	110110111001	55
8	no	111011100001	17
9	no	001111111000	25
10	no	111110010101	50

11	no	001010010111	24
12	no	101111111100	58
13	no	111111110110	55
14	no	101100010001	40
15	no	010111010101	26
16	no	000010101010	7
17	no	110000011100	10
18	no	001110000000	35
19	no	010110110000	6
20	no	110110111010	21
21	no	100010001011	51
22	no	101101100001	36
23	no	100000011111	32
24	no	101011111101	56
25	no	010001101100	51
26	no	110010001010	43
27	no	000000101111	62
28	no	111011101001	60
29	no	110110101010	18
30	no	000100010111	25
31	no	111011010110	53
32	no	111001000110	18
33	no	010110111011	7
34	no	000110111100	50
35	no	110110010010	0
36	no	010110110001	62
37	no	111101010010	27
38	no	001101111100	34
39	no	100000011000	16
40	no	110001010101	53
41	no	010011010000	13
42	no	000101001001	57
43	no	011001100110	61
44	no	010001010111	33
45	no	100111001010	6
46	no	001000000110	16
47	no	000110110001	17
48	no	101111001001	11
49	no	110101010100	25
50	no	101100010010	33
51	no	101110111111	58

52	no	111000011110	47
53	no	001011010000	45
54	no	100001100110	17
55	no	011011001100	30
56	no	000110101100	42
57	no	101011000111	61
58	no	010110101110	25
59	no	001101011011	36
60	no	001010010011	22
61	no	100010110001	44
62	no	101110110011	8
63	no	100000010010	23
64	no	111110111001	57
65	no	110111001000	53
66	no	001101100000	1
67	no	011101010010	5
68	no	110110011011	41
69	no	101000000011	14
70	no	011001011001	19
71	no	000111101011	28
72	no	010100111000	28
73	no	110111110111	43
74	no	111100011101	3
75	no	001001110011	21
76	no	100000000101	26
77	no	011100101110	53
78	no	000101000011	12
79	no	011101010001	25
80	no	000110100010	55
81	no	100000100100	11
82	no	000001000001	54
83	no	100111110111	42
84	no	101111000101	0
85	no	011110001010	0
86	no	000010101001	7
87	no	101011111110	38
88	no	001111110011	40
89	no	011110100101	50
90	no	111000111110	46
91	no	100101100000	28
92	no	000011101000	41

93	no	010111100000	30
94	no	101000110010	18
95	no	010000111101	49
96	no	101111010000	34
97	no	011100001101	39
98	no	110100000011	5
99	no	010001111001	54
100	no	011111011010	43
101	no	011101100010	42
102	no	000011000111	24
103	no	001111101010	29
104	no	101010011111	62
105	no	100000000010	10
106	no	111011001101	14
107	no	101110000100	57
108	no	011111111101	18
109	no	110000011111	6
110	no	011110000110	31
111	no	100011000111	24
112	no	101000100001	30
113	no	011001011000	32
114	no	101000100100	51
115	no	111010101011	47
116	no	110010000101	38
117	no	111101100111	50
118	no	001101100011	34
119	no	010101100000	45
120	no	011100100111	9
121	no	000100111111	30
122	no	101010011111	15
123	no	010001111101	19
124	no	000110110110	30
125	no	001000010101	27
126	no	000000110100	56
127	no	011100101011	55
128	no	100101001100	49
129	no	011111111010	18
130	no	100100110101	17
131	no	110000000100	27
132	no	010010111100	39
133	no	010110011010	52

134	no	101001010110	10
135	no	111101101010	10
136	no	100101111001	62

Finally, in this case, as we have seen, no solutions have been found. This can be explained by the fact that the only possible solutions correspond to multiplication tables of the finite field $\mathbb{F}_{16}$. It is well-known that, for any element in a finite field extension (such as $\mathbb{F}_{16}|\mathbb{F}_2$), its minimal polynomial is always irreducible. Therefore, a solution in this case would yield the existence of an element $b \in \mathbb{F}_{16}$ with minimal polynomial $x^4+x^2+1 = (x^2+x+1)^2$, which is not irreducible. Hence, the non-existence of solution.

7.1.3 Estimation of costs for the general case, in terms of Quantum Gates

In this section, for the sake of simplicity, we restrict to binary semifields. The aim is to obtain an estimation of the complexity in terms of number of quantum gates, of the Boolean function for the procedure described in this section.

Definition 7.2. Let $A,B \in \mathrm{Mat}_{n\times n}(\mathbb{F}_2)$, then $\Gamma_n^i \det\left(A/B^i\right)$ is defined as a sum of the combination of determinants, in which subsets of i rows of A are substituted by the corresponding rows of B. For example, let

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix},$$

in $\mathrm{Mat}_{3\times 3}(\mathbb{F}_2)$, then

$$\Gamma_4^1 \det\left(A/B^1\right) = \det\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} + \det\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

$$+ \det\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} + \det\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix},$$

$$\Gamma_3^2 \det\left(A/B^2\right) = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} + \det\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

$$+\det\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} + \det\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}$$

$$+\det\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} + \det\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix},$$

$$\Gamma_4^3 \det\left(A/B^1\right) = \det\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} + \det\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}$$

$$+\det\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} + \det\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}.$$

The following lemma shows an useful formula for our estimation.

Lemma 7.3 ([XDS93] Lemma 2)**.** *Let $A, B \in Mat_{n\times n}(\mathbb{F}_2)$, then*

$$\det(A+B) = \det(A) + \det(B) + \sum_{i=1}^{n-1} \Gamma_n^i \det\left(A/B^i\right).$$

Estimation: Let D be a finite semifield of order 2^d, that we want to describe with a standard basis $\{A_1, A_2, \ldots, A_d\}$. Consider all possible linear combinations from $S = \{A_2, \ldots, A_d\}$, i.e., without involving the identity matrix. There are $\binom{d-1}{1}$ of them of the form A_i, which have the first column full of 0 except for one position, which is 1. Therefore, the determinant of A_i would have (after expansion) at most $(d-1)!(d-2)$ products, and $(d-1)!-1$, additions (since we are working mod 2, subtractions can be seen as additions). Now, let us consider all linear combinations of the form $A_i + A_j$. The first column has two ones and $d-2$ zeros, and on the remaining columns each position a_{ij} has one sum of 2 variables. So, the determinant of $A_i + A_j$ would be the sum of two determinants, each one of a matrix of size $(d-1)\times(d-1)$ in which each entrance is a sum of two different variables. So, the number of products would be at most $\binom{d}{2}2^d(d-1)!(d-2)$ products, and $\binom{d}{2}2^d((d-1)!-1)$ additions. Now, if we take

k matrices, and add them together, they would have the following form: on the first column, k ones and $d-k$ zeroes, and in the remain columns each a_{ij} position has $k-1$ sums of k variables. There are $\binom{d-1}{k}$ matrices of this form, and the number of products would be at most $\binom{d}{k}k^d(d-1)!(d-2)$ multiplications, and $\binom{d}{k}k^d((d-1)!-1)$ would be an upper bound for the number of additions.

Now, consider the matrices $B+A_1$, where $B=\sum_{i=2}^{k}A_i$, for $k=3,\ldots,d$. Then, by lemma 7.3,

$$\begin{aligned}\det(B+A_1) &= \det(A_1)+\det(B)+\sum_{i=1}^{d-1}\Gamma_d^i\det\left(B/A_1^i\right)\\ &= 1+\det(B)+\sum_{i=1}^{d-1}\Gamma_d^i\det\left(B/A_1^i\right).\end{aligned}$$

Now, note that the number of products and additions of $\Gamma_d^i\det\left(B/A_1^i\right)$ is less or equal than those of $\det(B)$. Hence, the number of products in $\det(B+A_1)$ would be at most d times the products of $\det(B)$, and for additions, d times the sums of $\det(B)$ plus one.

There are $2^{d-1}-1$ non trivial linear combinations on S, and so, the number of matrices of the form A_1+B is the same. So, joining all of them, we get 2^d-2 determinants (removing that of the identity matrix, and of the null matrix), the same ones that if we consider all of $A_1\cup S$. Therefore, the number of required products would be at most

$$\begin{aligned}&\sum_{k=1}^{d-1}\binom{d-1}{k}k^d(d-1)!(d-2)+\sum_{k=1}^{d-1}d\binom{d-1}{k}k^d(d-1)!(d-2)\\ =\;&(d+1)\sum_{k=1}^{d-1}\binom{d-1}{k}k^d(d-1)!(d-2),\end{aligned}$$

and the number of sums

$$\begin{aligned}&\sum_{k=1}^{d-1}\binom{d-1}{k}k^d\left((d-1)!-1\right)+\sum_{k=1}^{d-1}\binom{d-1}{k}\left(dk^d\left((d-1)!-1\right)+1\right)\\ =\;&\sum_{k=1}^{d-1}\binom{d-1}{k}k^d\left((d-1)!-1\right)(1+d)+\sum_{k=1}^{d-1}\binom{d-1}{k}\\ =\;&\sum_{k=1}^{d-1}\binom{d-1}{k}k^d\left((d-1)!-1\right)(1+d)+2^{d-1}-1.\end{aligned}$$

Now, in order to construct the Boolean function, we need $d(d-1)^2$ variables, so $f:\{0,1\}^{d(d-1)^2}\rightarrow\{0,1\}$. On Section 3, in Figure 3.5, we showed a possible decomposition of the Toffoli gate. We should mention that by the Solovay-Kitaev Theorem [NC11] [Appendix 3], the asymptotic growth is the same whatever decomposition we choose. As we saw, every Toffoli gate in a

Number	Gate
$6a+b$	$CNOT$
$2a$	H
$3a$	$T^{\dagger}$
$4a$	T

Table 7.6: Cost in terms of quantum gates.

quantum circuit may execute up to six CNOT gates. Therefore, in Table 7.6 we give an estimate of a lower bound, of the cost of our Boolean function, in terms of quantum gates. where

$$\begin{aligned} a &= (d+1)\sum_{k=1}^{d-1}\binom{d-1}{k}k^d(d-1)!(d-2) \\ b &= \sum_{k=1}^{d-1}\binom{d-1}{k}k^d\left((d-1)!-1\right)(1+d)+2^{d-1}-1. \end{aligned}$$

Which shows that this approach is computationally unaffordable (it is indeed exponentially).

7.2 Quantum Computational Search of Finite Semifields with Quantum Optimization

Apart from the method covered in the previous section, we also tried other approaches to obtain a quantum search procedure for semifields with given properties. As an example, the purpose of this section is to use quantum annealing to find the multiplication table of Knuth's binary finite semifield of order 32, which is neither left or right primitive. We shall explain our approach and the difficulties found that prevented us from success.

Let us sketch our strategy with the problem of finding the finite field of order 8, which is commutative. Thus, its standard basis is

$$\left\{A_1=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},A_2=\begin{pmatrix}0&a_7&a_4\\1&a_8&a_5\\0&a_9&a_6\end{pmatrix},A_3=\begin{pmatrix}0&a_4&a_1\\0&a_5&a_2\\1&a_6&a_3\end{pmatrix}\right\}.$$

Analogously as before, by Corollary 2.53, we have that the characteristic polynomial of each non-scalar linear combination of A_1,A_2,A_3 has no linear factors. But the polynomials of degree 3 in $\mathbb{F}_2[x]$ that have no linear factors are $p(x)=x^3+x+1$, and $p(x+1)=(x+1)^3+(x+1)+1=x^3+x^2+1$. And so, we can assume without loss of generality, that A_2 is the companion matrix $C(p(x))$ (by a suitable change of basis). So, the standard basis is

$$\left\{A_1=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},A_2=\begin{pmatrix}0&0&1\\1&0&1\\0&1&0\end{pmatrix},A_3=\begin{pmatrix}0&1&a_1\\0&1&a_2\\1&0&a_3\end{pmatrix}\right\}.$$

Now, in order to find the multiplication table, $(a \vee b) \wedge (c \vee d)$ must be true, where a, b, c, d are the clauses

a $p_{A_3}(x) = p(x)$ $\quad$ c $p_{A_2+A_3}(x) = p(x)$

b $p_{A_1+A_3}(x) = p(x)$ $\quad$ d $p_{A_1+A_2+A_3}(x) = p(x)$.

We know that, from the *Cayley–Hamilton theorem*, every square matrix satisfies its own characteristic equation. In other words, $P_A(A) = 0_n$, where 0_n is the matrix in which each entry is zero. However, it does not mean that, if $p(A) = 0$, then $p(x)$ is the characteristic polynomial of A. But, since $p(x)$ is irreducible (because neither 0 nor 1 are roots), then in this case $p_A(x) = m_A(x)$, its minimal polynomial. Hence,

$$(a \vee b) \wedge (c \vee d) \Leftrightarrow \left(\left(A_3^3 + A_3 + I_3 = 0_3\right) \vee \left(A_3^3 + A_3^2 + I_3 = 0_3\right)\right) \wedge \left(\left((A_2 + A_3)^3 + (A_2 + A_3) + I_3 = 0_3\right) \vee \left((A_2 + A_3)^3 + (A_2 + A_3)^2 + I_3 = 0\right)\right). \tag{7.1}$$

Now, by using the following code C.28, which is based on the expansion of the matrix expressions in 7.1, we find the Boolean formula that satisfies the conditions mention above, which is

$$(((a_1 \wedge a_3 \oplus a_2) \wedge (\sim a_1) \wedge (a_1 \wedge a_3 \oplus a_2 \wedge a_3 \oplus \sim a_2) \wedge (a_2 \wedge a_3 \oplus \sim a_2) \wedge (a_2) \wedge (\sim (a_1 \wedge a_2)) \wedge (a_1 \oplus a_3) \wedge (a_3) \wedge (a_2)) \vee ((a_1 \wedge a_3 \oplus a_1 \oplus a_2) \wedge (\sim a_1) \wedge (\sim a_1 \oplus (a_2 \wedge a_3) \wedge \sim (a_2 \wedge a_3)) \wedge a_2 \wedge (a_1 \oplus a_2 \oplus \sim (a_2 \wedge a_3)) \wedge (\sim a_1) \wedge (\sim a_3) \wedge (a_1 \oplus a_2))) \wedge (((a_1 \wedge a_3 \oplus a_1 \oplus a_2 \oplus \sim a_3) \wedge (a_1 \wedge a_3 \oplus a_2 \oplus \sim a_3) \wedge (a_1 \wedge a_2 \oplus a_1 \wedge a_3 \oplus a_2 \wedge a_3) \wedge (a_1 \oplus a_2 \wedge a_3 \oplus \sim a_3) \wedge (a_1 \oplus a_2 \wedge a_3 \oplus a_2 \oplus a_3) \wedge (a_1 \wedge a_2 \oplus a_1 \wedge a_3 \oplus a_2 \oplus \sim a_3) \wedge (a_1 \oplus a_2 \oplus \sim a_3) \wedge \sim a_1) \vee ((a_1 \wedge a_3 \oplus a_2 \oplus \sim a_3) \wedge (a_1 \wedge a_3 \oplus a_1 \oplus a_2 \oplus a_3) \wedge (a_1 \wedge a_2 \oplus a_1 \oplus a_2 \wedge a_3 \oplus a_2 \oplus a_3) \wedge (a_1 \oplus a_2 \wedge a_3 \oplus a_2 \oplus a_3) \wedge (a_1 \oplus a_2 \wedge a_3 \oplus a_3) \wedge (a_1 \wedge a_2 \oplus a_1 \wedge a_3 \oplus a_1 \oplus a_2 \wedge a_2 \oplus a_2) \wedge (a_1 \oplus a_2 \oplus a_3) \wedge (a_1 \oplus a_2) \wedge (\sim a_2))$$

In order to use quantum techniques, we could transform it into a conjuctive normal form (CNF) expression by using, for instance, the Tseitn transformation (a standard way to convert a Boolean expression to CNF) [Tse83]. It introduces additional variables, but keeps the number of clauses and variables relatively small, obtaining a formula whose size grows linearly with respect to the input formula[1]. Now, mapping the operations $x \vee y, x \wedge y$ and x to $xy, x + y$ and $1 - x$, respectively, we find a polynomial, with at least 267 auxiliary variables [2], which is rather large to be used in HOBO (see Section 3.4).

In that order of ideas, let us move to the case of the binary semifield of order 32 which is not primitive (Section 2.4). Consider one of is standard bases $\{A_1, A_2, A_2, A_4, A_5\}$. By Corollary 2.53 item 2 the characteristic polynomial of each non-scalar linear combination of them is

[1] The procedur of Tseitin transformation can be seen in the following repository of github JMiguel01/Chapter-7-An-approach-to-the-Classification-of-Finite-Semifields-by-Quantum-Computing/Tseitin Transformation, and its Polynomial

[2] The entire polynomial can be seen in, the following repository of github JMiguel01/Chapter-7-An-approach-to-the-Classification-of-Finite-Semifields-by-Quantum-Computing/Tseitin Transformation, and its Polynomial

not primitive. By Corollary 2.53 item 1, they can not have linear factors either. But, since all irreducible polynomials of degree n in $\mathbb{F}_2[x]$ are primitive if and only if 2^n-1 is a (Mersenne) prime number, the admissible polynomials of degree 5 in $\mathbb{F}_2[x]$ that are not primitive are the product of an irreducible factor of degree 2, and an irreducible factor of degree 3. Namely, $p(x)=x^5+x+1$, and $p(x+1)=(x+1)^5+(x+1)+1=x^5+x^4+1$, because they are reducible without linear factors. Hence, the matrix A_2 can be assumed to either be the companion matrix $C(p(x))$ or $C(p(x+1))$. Let us suppose that $A_2=C(p(x))$ (as a change of basis can be made otherwise), so the standard bases would become

$$A_1=\begin{pmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{pmatrix},\ A_2=\begin{pmatrix}0&0&0&0&1\\1&0&0&0&1\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\end{pmatrix},\ A_3=\begin{pmatrix}0&0&a_1&a_6&a_{11}\\1&0&a_2&a_7&a_{12}\\0&1&a_3&a_8&a_{13}\\0&0&a_4&a_9&a_{14}\\0&0&a_5&a_{10}&a_{15}\end{pmatrix}$$

$$A_4=\begin{pmatrix}0&0&a_6&a_{16}&a_{21}\\0&0&a_7&a_{17}&a_{22}\\0&0&a_8&a_{18}&a_{23}\\1&0&a_9&a_{19}&a_{24}\\0&1&a_{10}&a_{20}&a_{25}\end{pmatrix},\ A_5=\begin{pmatrix}0&1&a_{11}&a_{21}&a_{26}\\0&1&a_{12}&a_{22}&a_{27}\\0&0&a_{13}&a_{23}&a_{28}\\1&0&a_{14}&a_{24}&a_{29}\\0&0&a_{15}&a_{25}&a_{30}\end{pmatrix}.$$

In order to find the multiplication table of Knuth's binary semifield, it has to be true that $(a\vee b)\wedge(c\vee d)\wedge(e\vee f)\wedge(g\vee h)\wedge(i\vee j)\wedge(k\vee l)\wedge(m\vee n)\wedge(o\vee p)\wedge(q\vee r)\wedge(s\vee t)\wedge(u\vee v)\wedge(w\vee x)\wedge(y\vee z)$ must happen, where $a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z$ are the clauses

a $p_{A_3}(x)=p(x)$

b $p_{A_1+A_3}(x)=p(x)$

c $p_{A_4}(x)=p(x)$

d $p_{A_1+A_4}(x)=p(x)$

e $p_{A_5}(x)=p(x)$

f $p_{A_1+A_5}(x)=p(x)$

g $p_{A_4+A_5}(x)=p(x)$

h $p_{A_1+A_4+A_5}(x)=p(x)$

i $p_{A_3+A_5}(x)=p(x)$

j $p_{A_1+A_3+A_5}(x)=p(x)$

k $p_{A_3+A_4}(x)=p(x)$

l $p_{A_1+A_3+A_4}(x)=p(x)$

m $p_{A_3+A_4+A_5}(x)=p(x)$

n $p_{A_1+A_3+A_4+A_5}(x)=p(x)$

o $p_{A_2+A_5}(x)=p(x)$

p $p_{A_1+A_2+A_5}(x)=p(x)$

q $p_{A_2+A_4}(x)=p(x)$

r $p_{A_1+A_2+A_4}(x)=p(x)$

s $p_{A_2+A_3}(x)=p(x)$

t $p_{A_1+A_2+A_3}(x)=p(x)$

u $p_{A_2+A_3+A_5}(x)=p(x)$

v $p_{A_1+A_2+A_3+A_5}(x)=p(x)$

w $p_{A_2+A_3+A_4}(x) = p(x)$ y $p_{A_2+A_3+A_4+A_5}(x) = p(x)$

x $p_{A_1+A_2+A_3+A_4}(x) = p(x)$ z $p_{A_1+A_2+A_3+A_4+A_5}(x) = p(x)$

Now, as before, $p(A) = 0_5$, where 0_5 is the matrix in which each entry is zero, does not mean that $p(x)$ is the characteristic polynomial of A. In order to see that, in this case, it is indeed the characteristic polynomial, first note that $x^5+x+1 = (x^2+x+1)(x^3+x^2+1)$. So it is the product of irreducible polynomials of degree 2 and 3, respectively. Also, x^5+x^4+1 can be factored as the product of irreducible polynomials, in fact as $(x^2+x+1)(x^3+x+1)$.

Let us see the case for the polynomial $x^5+x+1 = (x^2+x+1)(x^3+x^2+1)$, since the other one follows the same lines. For that, note that we might have the case that $p(A) = 0_5$, because the matrix annihilates one of those irreducible factors. So, we must show that in fact the minimal polynomial is neither of them. Indeed, without loss of generality, let us suppose that the minimal polynomial is x^3+x+1, and consider the map $T : (\mathbb{Z}/2\mathbb{Z})^5 \to (\mathbb{Z}/2\mathbb{Z})^5$ given by $T(v) = Av$, for $v \in (\mathbb{Z}/2\mathbb{Z})^5$. So, from Theorem 2.32, we have that

$$(\mathbb{Z}/2\mathbb{Z})^5 \cong \left(\mathbb{F}_2[x]/(x^3+x+1)\right)^m$$

for some $m \geq 1$ as $\mathbb{F}_2[x]$-modules. But $\mathbb{F}_2[x]/(x^3+x+1)$ has 8 elements, so this is not possible. Therefore (x^3+x+1) can not be the minimal polynomial of A. Hence, both minimal and characteristic polynomials must agree.

The boolean expression for the case $(a \vee b)$ can be seen in C.29, which is very large. In this case, it would not involve all variables, as we have on case z above. Thus, the polynomial for the general expression would be too cumbersome. Therefore, we have been not able to apply this method for the many qubits required.

Chapter 8

Conclusions

In this last chapter, we want to briefly summarize the main conclusions of this thesis and to indicate possible lines for future work.

This thesis is based on the application of quantum computational techniques for the effective study of algebraic structures. It is articulated around three problems. Firstly, the problem of detecting pairs of different constants for determining the commutativity of a finite dimensional algebra. Thus, based on the results found in [HCCR22], in Chapter 5, we study two specific classes of QADS. The first are QADS of combinatorial type, which generalize the well-known controlled operators. We have determined its efficient constructibility, the expression of the state after the application of the detection operator and the closure of its algorithm as a subclass of QADS. As an application, we have considered the problem of deciding whether, for a given operator-eigenvector pair, the corresponding eigenvalue is a given one or not. The second family are QADS of the rotational type, which include as a particular case the QADS of Grover's search. We have studied the expression of the state after the application of the detection operator on the initial state, the algorithmic closure of this subclass of QADS, and we have also considered its combinatorial QADS. Interestingly, we have derived some nice equivalences for these QADS, in terms of tensor products and square root products of the original QADS. Furthermore, we have successfully applied them to the problem of the commutativity of algebras (Section 5.5), which was our original problem of interest.

As a future work, in relation to combinatorial QADS, we want to explore more applications of them, for instance, if there exist an approximation to the QFT for $\mathbb{Z}/N\mathbb{Z}$ with N odd and if the Swap test can be seen as a particular case of those QADS, together with a generalization of it. Furthermore, we want to study variations of combinatorial QADS for instance, a change on the initial or final Hadamard gates by rotations like the QFT for $\mathbb{Z}/N\mathbb{Z}$ with N a positive integer, and explore other families of QADS that include measurements.

The second problem, studied in [JER23], is included in Chapter 6. We have shown that for a given multiplication table, of a n-dimensional not necessarily associative and not necessarily commutative $\mathbb{F}_p$-algebra A, and for each substructure, right, middle and left nuclei, the nucleus and the center of A, we can build an efficient quantum algorithm which calculates each substructure. Our approach is based on the existence of a function that hides each substructure,

for which a quantum oracle can be built efficiently, both in terms of the number of qubits and the number of quantum gates. Our quantum algorithm uses such a black box, and the specific number of qubits and quantum gates required can be found in Theorem 6.6. In those situations, we have shown an exponential gain with our quantum algorithm.

As a future work, in relation to quantum methods that solve the hidden subgroup problems, of which the Shor-type algorithm is the greatest exponent, we propose the study of problems related to finding the order of an element, and the primitiveness of elements in finite semifields.

Finally, the problem studied in [HCR23] is collected in Section 7.1. We give some of the multiplication tables for $\mathbb{F}_8$ and $\mathbb{F}_{16}$, based on the quantum search of Grover's algorithm. Furthermore, to classify a finite semifield of order 2^d with this methodology, we show that at least $d(d-1)^2$ qubits are required. Additionally, we give an estimate of the number of quantum gates needed to build the quantum circuit, showing that this approach is not asymptotically efficient. As another alternative to quantum computing techniques for classifying finite semifields, we tried to use Quantum Annealing, a form of computing that efficiently samples the low-energy configurations of a quantum system [KN98], [FGGS00]. However, we end up with the problem of expanding a huge polynomial, which makes it impossible for us to apply these techniques. Thus, as a future work we want to find efficient quantum algorithms (or use classical and quantum algorithms together) that classify them efficiently.

Conclusiones

En este capítulo, queremos resumir brevemente las principales conclusiones de esta tesis e indicar posibles líneas de trabajo a futuro .

Esta tesis, centrada en la aplicación de tecnicas computacionales cuánticas para el estudio efectivo de estructuras algebraicas, se articula sobre tres problemas. En primer lugar, el problema de detección de pares de constantes diferentes para la determinación de la conmutatividad de un álgebra finito dimensional. Así, a partir de los resultados encontrados en [HCCR22], en el capítulo 5, estudiamos dos clases específicas de QADS. El primero son QADS de tipo combinatorio, que generalizan los conocidos operadores controlados. Hemos determinado su constructibilidad eficiente, la expresión del estado después de la aplicación del operador de detección y el cierre de su algoritmo como una subclase de QADS. Como aplicación, hemos considerado el problema de decidir si, para un par operador-vector propio dado, el valor propio correspondiente es uno dado o no. La segunda familia es la de QADS rotacionales, que incluyen como caso particular los QADS de la búsqueda de Grover. Hemos estudiado la expresión del estado después de la aplicación del operador de detección sobre el estado inicial, el cierre algorítmico de esta subclase de QADS, y también hemos considerado sus QADS combinatorios. Curiosamente, hemos derivado algunas equivalencias agradables para estos QADS, en términos de productos tensoriales y productos de raíces cuadradas de los QADS originales. Además, los hemos aplicado de forma satisfactoria al problema de la conmutatividad de álgebras (Sección 5.5), que era nuestro objetivo original.

Como trabajo futuro, en relación con los QADS combinatorios, queremos explorar más aplicaciones de ellos, por ejemplo si existe una aproximación de la transformada cuántica de Fourier para $\mathbb{Z}/N\mathbb{Z}$ para N impar, y si el test de Swap puede verse como un caso particular de esos QADS, junto con una generalización del mismo, tal como hicimos con el test de Hadamard. Además, queremos estudiar variaciones de los QADS combinatorios, por ejemplo, un cambio en las puertas de Hadamard inicial o final mediante rotaciones como la transformada cuántica de Fourier para $\mathbb{Z}/N\mathbb{Z}$ con N un entero positivo, y explorar otras familias de QADS que incluyan medidas.

El segundo problema, estudiado en [JER23], se encuentra recogido en el Capítulo 6. Hemos demostrado que para una tabla de multiplicación dada, de una $\mathbb{F}_p$-algebra n-dimensional no necesariamente asociativa y no necesariamente conmutativa A, y para cada subestructura, núcleo por derecha, medio e izquierda, el núcleo y el centro de A, podemos construir un algoritmo cuántico eficiente que calcula dicha subestructura. Nuestro enfoque se basa en la existencia de

una función que oculta la subestructura, para la cual se puede construir un oráculo cuántico de manera eficiente, tanto en términos de número de qubits como de número de puertas cuánticas. Nuestro algoritmo cuántico utiliza una caja negra de este tipo, y la cantidad concreta de qubits y puertas cuánticas requeridas se puede encontrar en Teorema 6.6. En esas situaciones, hemos mostrado una ganancia exponencial con nuestro algoritmo cuántico.

Por lo que como trabajo futuro en relación con los métodos cuánticos que resuelven los problemas de subgrupos ocultos, de los cuales el algoritmo de tipo Shor es el máximo exponente, proponemos el estudio de problemas relacionados con la búsqueda del orden de un elemento, y la primitividad de elementos en semicuerpos finitos.

Por último, el problema estudiado en [HCR23], es recogido en la Sección 7.1. Damos algunas de las tablas de multiplicar de $\mathbb{F}_8$ y $\mathbb{F}_{16}$, basadas en la búsqueda cuántica del algoritmo de Grover. Además, para clasificar un semicuerpo finito de orden 2^d con esta técnica, mostramos que se requieren al menos $d(d-1)^2$ qubits. Además, damos una estimación del número de puertas cuánticas necesarias para construir el circuito cuántico, lo que demuestra que este enfoque no es asintóticamente eficiente. Como otra alternativa de las técnicas de computación cuántica para clasificar semicuerpos finitos, intentamos usar Quantum Annealing, una forma de computación que muestra eficientemente las configuraciones de baja energía de un sistema cuántico [KN98], [FGGS00]. Sin embargo, terminamos con el problema de expandir un polinomio descomunal, lo que nos hace imposible aplicar estas técnicas.

Por lo que como trabajo futuro queremos encontrar algoritmos cuánticos (o usar algoritmos clásicos junto con algoritmos cuánticos) que sean eficientes para su clasificación.

Chapter

Appendices

A Codes for Chapter 5

Codes for Chapter 5 can also be found in: *github.com/JMiguel01/Combinatorial – And – Rotational – QADS*

Code A.1.

```
import math
import numpy as np

import matplotlib.pyplot as plt
xaxisfigure21 = np.arange(0,1.8,0.2)
xaxisfigure22 = np.arange(1.6, math.pi+0.2,0.2)

def probabilityq(angle,m):
    z = (math.cos(angle/2))**(2*m)
    return(z)
def bernoulli(n, q):
    n_success = 0
    for i in range(n):
        random_number = np.random.random()
        if random_number < q:
            n_success += 1
    return(n_success)
def anglep(a,m):
    p = a/(10**4)
    alpha = math.acos(2*(p**(1/m))-1)
    #alpha = 2*math.acos(p**(1/(2*m)))
    return(alpha)
def mae(beta,n,m):
    alphas = []
```

```
    for i in range(n):
        alphas.append(abs(anglep(bernoulli(10**4,
        ↪  (probabilityq(beta,m))),m)-beta))
    #print(alphas)
    Sum = sum(alphas)
    #print(Sum)
    result = Sum/n
    #result = abs((Sum)/n)
    return(result)

def yaxis(xaxis,m):
    yaxisfigure = []
    for i in xaxis:
        yaxisfigure.append(mae(i,10**3,m))
    return(yaxisfigure)

plt.plot(xaxisfigure21, yaxis(xaxisfigure21,1), color='blue', label =
↪  "$m=1$")
plt.plot(xaxisfigure21, yaxis(xaxisfigure21,2), color='orange', label =
↪  "$m=2$")
plt.plot(xaxisfigure21, yaxis(xaxisfigure21,3), color='green', label = "
↪  $m=3$")
plt.plot(xaxisfigure21, yaxis(xaxisfigure21,4) , color='red', label = "
↪  $m=4$")
plt.plot(xaxisfigure21, yaxis(xaxisfigure21,5) , color='purple', label =
↪  "$m=5$")

plt.xlabel("Angle")
plt.ylabel("Mean Absolute Error")

# Adding legend, which helps us recognize the curve according to it's color
plt.legend()

plt.show()

plt.plot(xaxisfigure22, yaxis(xaxisfigure22,1), color='blue', label =
↪  "$m=1$")
plt.plot(xaxisfigure22, yaxis(xaxisfigure22,2), color='orange', label =
↪  "$m=2$")
plt.plot(xaxisfigure22, yaxis(xaxisfigure22,3), color='green', label =
↪  "$m=3$")
plt.plot(xaxisfigure22, yaxis(xaxisfigure22,4) , color='red', label = "
↪  $m=4$")
plt.plot(xaxisfigure22, yaxis(xaxisfigure22,5) , color='purple', label =
↪  "$m=5$")
```

```

plt.xlabel("Angle")
plt.ylabel("Mean Absolute Error")

# Adding legend, which helps us recognize the curve according to it's color
plt.legend()

plt.show()
```

Code A.2.

```
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

xaxisfigure21 = np.arange(0,1.8,0.2)
xaxisfigure22 = np.arange(1.6, math.pi+0.2,0.2)
def probabilityq(angle,m):
    z = (math.cos(angle/2))**(2*m)
    return(z)
def bernoulli(n, q):
    n_success = 0
    for i in range(n):
        random_number = np.random.random()
        if random_number < q:
            n_success += 1
    return(n_success)
def anglep(a,m):
    p = a/(10**4)
    alpha = math.acos(2*(p**(1/m))-1)
    #alpha = 2*math.acos(p**(1/(2*m)))
    return(alpha)
def mae(beta,n,m):
    alphas = []
    for i in range(n):
        alphas.append(abs((anglep(bernoulli(10**4,
        ↪  (probabilityq(beta,m))),m))-beta))
    #print(alphas)
    Sum = sum(alphas)
    #print(Sum)
    result = Sum/n
    #result = abs((Sum)/n)
    return(result)
def yaxis(xaxis,m):
```

```
    yaxisfigure = []
    for i in xaxis:
        yaxisfigure.append(mae(i,10**3,m))
    return(yaxisfigure)
# Calculate the average
m1_mean = np.mean(yaxis(xaxisfigure21,1))
m2_mean = np.mean(yaxis(xaxisfigure21,2))
m3_mean = np.mean(yaxis(xaxisfigure21,3))
m4_mean = np.mean(yaxis(xaxisfigure21,4))
m5_mean = np.mean(yaxis(xaxisfigure21,5))
# Calculate the standard deviation
m1_std = np.std(yaxis(xaxisfigure21,1))
m2_std = np.std(yaxis(xaxisfigure21,2))
m3_std = np.std(yaxis(xaxisfigure21,3))
m4_std = np.std(yaxis(xaxisfigure21,4))
m5_std = np.std(yaxis(xaxisfigure21,5))
# Define labels, positions, bar heights and error bar heights
labels = ['1', '2', '3', '4', '5']
x_pos = np.arange(len(labels))
m = [m1_mean, m2_mean, m3_mean, m4_mean, m5_mean]
error = [m1_std, m2_std, m3_std, m4_std, m5_std]
# Build the plot
fig, ax = plt.subplots()
ax.bar(x_pos, m,
       yerr=error,
       align='center',
       alpha=0.5,
       ecolor='black',
       capsize=10)
ax.set_ylabel('Average Error')
ax.set_xlabel('m')
ax.set_xticks(x_pos)
ax.set_xticklabels(labels)
ax.yaxis.grid(True)

# Save the figure and show
plt.show()

# Calculate the average
m12_mean = np.mean(yaxis(xaxisfigure22,1))
m22_mean = np.mean(yaxis(xaxisfigure22,2))
m32_mean = np.mean(yaxis(xaxisfigure22,3))
m42_mean = np.mean(yaxis(xaxisfigure22,4))
m52_mean = np.mean(yaxis(xaxisfigure22,5))
# Calculate the standard deviation
```

```
m12_std = np.std(yaxis(xaxisfigure22,1))
m22_std = np.std(yaxis(xaxisfigure22,2))
m32_std = np.std(yaxis(xaxisfigure22,3))
m42_std = np.std(yaxis(xaxisfigure22,4))
m52_std = np.std(yaxis(xaxisfigure22,5))
# Define labels, positions, bar heights and error bar heights
labels = ['1', '2', '3', '4', '5']
x_pos = np.arange(len(labels))
m2 = [m12_mean, m22_mean, m32_mean, m42_mean, m52_mean]
error2 = [m12_std, m22_std, m32_std, m42_std, m52_std]
# Build the plot
fig2, ax2 = plt.subplots()
ax2.bar(x_pos, m2,
       yerr=error2,
       align='center',
       alpha=0.5,
       ecolor='black',
       capsize=10)
ax2.set_ylabel('Average Error')
ax2.set_xlabel('m')
ax2.set_xticklabels(labels)
ax2.yaxis.grid(True)

# Save the figure and show
plt.show()
```

Code A.3.

```
import math
import numpy as np
import matplotlib.pyplot as plt
xaxisfigure21 = np.linspace(0,np.pi,10)
def probabilitytheorem3(beta,alpha,m):
    z = (math.cos((beta-alpha)/2))**(2*m)
    return(z)
def bernoulli(n, q):
    n_success = 0
    for i in range(n):
        random_number = np.random.random()
        if random_number < q:
            n_success += 1
    return(n_success)
def anglep1(a):
    p1 = a/(10**4)
    return(p1)
```

```
def dichotomy_search_1(beta,n,m):
    k= 10
    low = 0
    high = np.pi
    for i in range(k):
        mid = (low + high) / 2.0
        pl = anglep1(bernoulli(n,probabilitytheorem3(beta,low,m)))
        ph = anglep1(bernoulli(n,probabilitytheorem3(beta,high,m)))
        if (pl > ph):
            high = mid
        else:
            low = mid
    return(mid)
def dichomae(beta,n,m):
    alphas = []
    for i in range(n):
        alphas.append(abs(dichotomy_search_1(beta,n,m)-beta))
    Sum = sum(alphas)
    result = Sum/n
    return(result)
def dichoyaxis(xaxisfigure21,n,m):
    yaxisfigure = []
    for i in xaxisfigure21:
        yaxisfigure.append(dichomae(i,n,m))
    return(yaxisfigure)
plt.plot(xaxisfigure21, dichoyaxis(xaxisfigure21,10**3,1), color='blue',
 ↪  label = "$m=1$")
plt.plot(xaxisfigure21, dichoyaxis(xaxisfigure21,10**3,2), color='orange',
 ↪  label = "$m=2$")
plt.plot(xaxisfigure21, dichoyaxis(xaxisfigure21,10**3,3), color='green',
 ↪  label = "$m=3$")
plt.plot(xaxisfigure21, dichoyaxis(xaxisfigure21,10**3,4), color='red',
 ↪  label = "$m=4$")
plt.plot(xaxisfigure21, dichoyaxis(xaxisfigure21,10**3,5), color='purple',
 ↪  label = "$m=5$")

plt.xlabel("Angle")
plt.ylabel("Mean Absolute Error")

# Adding legend, which helps us recognize the curve according to it's color
plt.legend()

plt.show()
# Calculate the average
m1_mean = np.mean(dichoyaxis(xaxisfigure21,10**3,1))
```

```
m2_mean = np.mean(dichoyaxis(xaxisfigure21,10**3,2))
m3_mean = np.mean(dichoyaxis(xaxisfigure21,10**3,3))
m4_mean = np.mean(dichoyaxis(xaxisfigure21,10**3,4))
m5_mean = np.mean(dichoyaxis(xaxisfigure21,10**3,5))
# Calculate the standard deviation
m1_std = np.std((dichoyaxis(xaxisfigure21,10**3,1)))
m2_std = np.std((dichoyaxis(xaxisfigure21,10**3,2)))
m3_std = np.std((dichoyaxis(xaxisfigure21,10**3,3)))
m4_std = np.std((dichoyaxis(xaxisfigure21,10**3,4)))
m5_std = np.std((dichoyaxis(xaxisfigure21,10**3,5)))
# Define labels, positions, bar heights and error bar heights
labels = ['1', '2', '3', '4', '5']
x_pos = np.arange(len(labels))
m = [m1_mean, m2_mean, m3_mean, m4_mean, m5_mean]
error = [m1_std, m2_std, m3_std, m4_std, m5_std]
# Build the plot
fig, ax = plt.subplots()
ax.bar(x_pos, m,
       yerr=error,
       align='center',
       alpha=0.5,
       ecolor='black',
       capsize=10)
ax.set_ylabel('Average Error')
ax.set_xlabel('m')
ax.set_xticks(x_pos)
ax.set_xticklabels(labels)
ax.yaxis.grid(True)

# Save the figure and show
plt.show()
```

Code A.4.

```
import math
import numpy as np
import matplotlib.pyplot as plt
xaxisfigure21 = np.linspace(0,np.pi,10)
def probabilitytheorem3(beta,alpha,m):
    z = (math.cos((beta-alpha)/2))**(2*m)
    return(z)
def probabilityq(angle,m):
    z = (math.cos(angle/2))**(2*m)
    return(z)
def bernoulli(n, q):
```

```
    n_success = 0
    for i in range(n):
        random_number = np.random.random()
        if random_number < q:
            n_success += 1
    return(n_success)
def anglep(a,m):
    p = a/(8000)
    alpha = math.acos(2*(p**(1/m))-1)
    #alpha = 2*math.acos(p**(1/(2*m)))
    return(alpha)
def anglep1(a):
    p1 = a/(10**3)
    return(p1)
def dichotomy_search_1(beta,n,m):
    k = 2
    low = 0
    high = np.pi
    for i in range(k):
        mid = (low + high) / 2.0
        pl = anglep1(bernoulli(n,probabilitytheorem3(beta,low,m)))
        ph = anglep1(bernoulli(n,probabilitytheorem3(beta,high,m)))
        if (pl > ph):
            high = mid
        else:
            low = mid
    return(low,high)
def mae(beta,n,m):
    alphas = []
    for i in range(n):
        alphas.append(abs(anglep(bernoulli(8000,
        (probabilityq(beta-dichotomy_search_1(beta,10**3,m)[0],m))),
        m)-(beta-dichotomy_search_1(beta,10**3,m)[0])))
    Sum = sum(alphas)
    result = Sum/n
    return(result)
def y_axis(m):
    y_axis_1 = []
    for i in xaxisfigure21:
        y_axis_1.append(mae(i,8000,m))
    return(y_axis_1)
plt.plot(xaxisfigure21, y_axis(1), color='blue', label = "$m=1$")
plt.plot(xaxisfigure21, y_axis(2), color='orange', label = "$m=2$")
plt.plot(xaxisfigure21, y_axis(3), color='green', label = "$m=3$")
plt.plot(xaxisfigure21, y_axis(4), color='red', label = "$m=4$")
```

```
plt.plot(xaxisfigure21, y_axis(5), color='purple', label = "$m=5$")

plt.xlabel("Angle")
plt.ylabel("Mean Absolute Error")

# Adding legend, which helps us recognize the curve according to it's color
plt.legend()
# Calculate the average
m1_mean = np.mean(y_axis(1))
m2_mean = np.mean(y_axis(2))
m3_mean = np.mean(y_axis(3))
m4_mean = np.mean(y_axis(4))
m5_mean = np.mean(y_axis(5))
# Calculate the standard deviation
m1_std = np.std(y_axis(1))
m2_std = np.std(y_axis(2))
m3_std = np.std(y_axis(3))
m4_std = np.std(y_axis(4))
m5_std = np.std(y_axis(5))
# Define labels, positions, bar heights and error bar heights
labels = ['1', '2', '3', '4', '5']
x_pos = np.arange(len(labels))
m = [m1_mean, m2_mean, m3_mean, m4_mean, m5_mean]
error = [m1_std, m2_std, m3_std, m4_std, m5_std]
# Build the plot
fig, ax = plt.subplots()
ax.bar(x_pos, m,
       yerr=error,
       align='center',
       alpha=0.5,
       ecolor='black',
       capsize=10)
ax.set_ylabel('Average Error')
ax.set_xlabel('m')
ax.set_xticks(x_pos)
ax.set_xticklabels(labels)
ax.yaxis.grid(True)

# Save the figure and show
plt.show()
```

Code A.5.

```
import math
import numpy as np
```

```
import matplotlib.pyplot as plt
def try1(x,s,m):
    z = ((math.cos(x*s))**(2*m))*((math.cos(x*s*m)**2))
    return(z)
def try2(x,S,m):
    First = []
    Second = []
    for i in range(S):
        First.append(try1(x,i,m))
        Sum = sum(First)
        result = 1-Sum /((i+1))
        Second.append(result)
    return(Second)
def plotcombinatorial(x,S,m):
    xaxisfigure1 = np.arange(0,51,1)
    yaxisfigure1 = try2(x,S,m)
    return(xaxisfigure1, yaxisfigure1)
plt.plot(xaxisfigure1, try2(0.177710,51,1), color='orange', label =
 ↪  "C-Grover")
plt.plot(xaxisfigure1, try2(0.177710,51,2), color='purple', label =
 ↪  "C-Grover $ \otimes $ C-Grover")
plt.plot(xaxisfigure1, try2(0.177710,51,3), color='red', label =
 ↪  "Combinatorial $m=3$")
plt.plot(xaxisfigure1, try2(0.177710,51,4) , color='green', label =
 ↪  "Combinatorial $m=4$")
plt.plot(xaxisfigure1, try2(0.177710,51,10) , color='blue', label =
 ↪  "Combinatorial $m=10$")
plt.title("Combinatorial Gover QADS")
plt.xlabel("Iterations")
plt.ylabel("Detection Probability")
plt.legend()
plt.plot()
plt.show()
```

Code A.6.

```
import math
import numpy as np
import matplotlib.pyplot as plt
def try1(x,s,m):
    z = ((math.cos(x*s))**(2*m))*((math.cos(x*s*m)**2))
    return(z)
def try2(x,S,m):
    First = []
    Second = []
```

```
    for i in range(S):
        First.append(try1(x,i,m))
        Sum = sum(First)
        result = 1-Sum /((i+1))
        Second.append(result)
    return(Second)
def plotcombinatorial(x,S,m):
    xaxisfigure1 = np.arange(0,51,1)
    yaxisfigure1 = try2(x,S,m)
    return(xaxisfigure1, yaxisfigure1)

plt.plot(xaxisfigure1, try2(0.1253278311,51,1), color='orange', label =
 ↪  "C-Grover")
plt.plot(xaxisfigure1, try2(0.1253278311,51,2), color='purple', label =
 ↪  "C-Grover $ \otimes $ C-Grover")
plt.plot(xaxisfigure1, try2(0.1253278311,51,3), color='red', label =
 ↪  "Combinatorial $m=3$")
plt.plot(xaxisfigure1, try2(0.1253278311,51,4) , color='green', label =
 ↪  "Combinatorial $m=4$")
plt.plot(xaxisfigure1, try2(0.1253278311,51,10) , color='blue', label =
 ↪  "Combinatorial $m=10$")
plt.title("Combinatorial Gover QADS")
plt.xlabel("Iterations")
plt.ylabel("Detection Probability")
plt.legend()
plt.plot()
plt.show()
```

B Codes for Chapter 6

Simulations for both examples in Chapter 5 can also be found in: *github.com/JMiguel01/Chapter−6−Efficient−Quantum−Algorithms−To−Find−Substructures−On−Finite−Algebras−Examples*.

Code B.7.

```
from sympy import symbols, Matrix, pprint, collect, factor
from itertools import accumulate
import itertools
x = []
name = "x"
for i in range(1,5):
    v = symbols(name+str(i))
    x.append(v)
```

```
A1 = Matrix([[1, 0, 0],[0, 1, 0],[0, 0, 0]])
A2 = Matrix([[0, 0, 0],[0, 0, 1],[0, 0, 0]])
A3 = Matrix([[0, 0, 0],[0, 0, 0],[0,0,1]])
def matrices(A1,A2,A3,A4):
    MatrixA1 = {}
    for i in range(3):
        for j in range(3):
            if (i == 0):
                MatrixA1[i,j] = A1[i+j]
            if (i == 1):
                 MatrixA1[i,j] = A1[2+i+j]
            if (i == 2):
                 MatrixA1[i,j] = A1[4+i+j]
    MatrixA2 = {}
    for i in range(3):
        for j in range(3):
            if (i == 0):
                MatrixA2[i,j] = A2[i+j]
            if (i == 1):
                 MatrixA2[i,j] = A2[2+i+j]
            if (i == 2):
                 MatrixA2[i,j] = A2[4+i+j]
    MatrixA3 = {}
    for i in range(3):
        for j in range(3):
            if (i == 0):
                MatrixA3[i,j] = A3[i+j]
            if (i == 1):
                 MatrixA3[i,j] = A3[2+i+j]
            if (i == 2):
                 MatrixA3[i,j] = A3[4+i+j]
    M = [MatrixA1, MatrixA2, MatrixA3]
    return(M)
def functionZ(A):
    s1 = []
    s2 = []
    for i in range(len(A)):
        for k in range(len(A)):
            for m in range(len(A)):
                s1.append((A[m][k,i]- A[i][k,m])*x[m])
    Input = s1
    split = len(s1) / len(A)
    length_to_split = []
    for i in range(int(split)):
          length_to_split.append(len(A))
```

```
    Output = [Input[x - y: x] for x, y in zip(accumulate(length_to_split),
     ↪  length_to_split)]
    s4 = []
    for i in range(len(Output)):
        s3 = sum(Output[i])
        s4.append(s3)
    return(s4)
functionZ(Matrices(A1,A2,A3))
```

Code B.8.

```
# importing Qiskit
from qiskit import IBMQ, Aer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister,
 ↪  transpile, assemble
# import basic plot tools
from qiskit.visualization import plot_histogram

qreg = QuantumRegister(3)
creg = ClassicalRegister(3)
ancillary = QuantumRegister(2)
qc = QuantumCircuit(qreg, ancillary, creg)
qc.cx(qreg[1],ancillary[0])
qc.cx(qreg[0],ancillary[1])
qc.cx(qreg[2],ancillary[1])
qc.barrier(qreg)
qc.h(range(3))
qc.draw()
```

Code B.9.

```
qreg = QuantumRegister(3)
creg = ClassicalRegister(3)
ancillary = QuantumRegister(2)
qc = QuantumCircuit(qreg, ancillary, creg)
qc.h(range(3))
qc.barrier(qreg)
qc.cx(qreg[1],ancillary[0])
qc.cx(qreg[0],ancillary[1])
qc.cx(qreg[2],ancillary[1])
qc.barrier(qreg)
qc.h(range(3))
qc.measure(qreg[0],creg[0])
qc.measure(qreg[1],creg[1])
qc.measure(qreg[2],creg[2])
```

```
qc.draw()
```

Code B.10.

```
aer_sim = Aer.get_backend('aer_simulator')
transpiled_qc = transpile(qc, aer_sim)
qobj = assemble(transpiled_qc)
results = aer_sim.run(qobj,shots =10).result()
counts = results.get_counts()
print(counts)
plot_histogram(counts)
```

Code B.11.

```
from sympy import symbols, Matrix, pprint, collect, factor
import itertools
x1, x2, x3, x4 = symbols('x1, x2, x3, x4')
x = []
name = "x"
for i in range(1,5):
    v = symbols(name+str(i))
    x.append(v)
MA1 = Matrix([[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]])
MA2 = Matrix([[0, 0, 0, 0],[0, 0, 1, 0],[0, 0, 0, 0],[0, 0, 0, 0]])
MA3 = Matrix([[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]])
MA4 = Matrix([[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]])
def matrices(A1,A2,A3,A4):
    MatrixAA1 = {}
    for i in range(4):
        for j in range(4):
            if (i == 0):
                MatrixAA1[i,j] = A1[i+j]
            if (i == 1):
                 MatrixAA1[i,j] = A1[3+i+j]
            if (i == 2):
                 MatrixAA1[i,j] = A1[6+i+j]
            if (i == 3):
                MatrixAA1[i,j] = A1[9+i+j]
    MatrixAA2 = {}
    for i in range(4):
        for j in range(4):
            if (i == 0):
                MatrixAA2[i,j] = A2[i+j]
            if (i == 1):
                 MatrixAA2[i,j] = A2[3+i+j]
            if (i == 2):
```

```
                MatrixAA2[i,j] = A2[6+i+j]
            if (i == 3):
                MatrixAA2[i,j] = A2[9+i+j]
    MatrixAA3 = {}
    for i in range(4):
        for j in range(4):
            if (i == 0):
                MatrixAA3[i,j] = A3[i+j]
            if (i == 1):
                MatrixAA3[i,j] = A3[3+i+j]
            if (i == 2):
                MatrixAA3[i,j] = A3[6+i+j]
            if (i == 3):
                MatrixAA3[i,j] = A3[9+i+j]
    MatrixAA4 = {}
    for i in range(4):
        for j in range(4):
            if (i == 0):
                MatrixAA4[i,j] = A4[i+j]
            if (i == 1):
                MatrixAA4[i,j] = A4[3+i+j]
            if (i == 2):
                MatrixAA4[i,j] = A4[6+i+j]
            if (i == 3):
                MatrixAA4[i,j] = A4[9+i+j]
    MMT = [MatrixAA1,MatrixAA2,MatrixAA3,MatrixAA4]
    return(MMT)
def Selecting(X):
    x1, x2, x3, x4 = symbols('x1, x2, x3, x4')
    X1 = []
    for i in range(len(X)):
        if (X[i] != 0):
            X1.append(X[i])
    return(X1)
def Equations(First):
    BB31= First
    Pl1 =[]
    for i in range(len(BB31)):
        Pl1.append(dict(BB31[i].as_coefficients_dict()))
    ##Symplify computations of coefficients module 2
    EQ1 =[]
    for l in range(len(Pl1)):
        for s in Pl1[l]:
            if (Pl1[l][s] % 2) == 0:
                Pl1[l][s] = 0
```

```
            else:
                Pl1[l][s] = 1
        EQ1.append(sum([key * val for key, val in Pl1[l].items()]))
    ##Selecting the equations we need
    EQ3=[]
    for l in range(len(EQ1)):
        if (EQ1[l] != 0) and (EQ1[l]!=1):
            EQ3.append(EQ1[l])
    return(EQ3)
    def functionNr1(A):
    s1 = []
    s2 = []
    s3 = []
    for i in range(len(A)):
        for j in range(len(A)):
            for s in range(len(A)):
                for m in range(len(A)):
                    for k in range(len(A)):
                        s1.append(A[i][k,j]*A[k][s,m]- A[j][k,m]*A[i][s,k])
                    l1 = sum(s1)
                    s2.append(l1*x[m])
                    l2 = sum(s2)
                s3.append(l2)
    return(s3)
RN = Equations(Selecting(functionNr1(matrices(MA1,MA2,MA3,MA4))))
print(RN)
```

Code B.12.

```
# importing Qiskit
from qiskit import IBMQ, Aer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister,
 ↪  transpile, assemble
# import basic plot tools
from qiskit.visualization import plot_histogram
aer_sim = Aer.get_backend('aer_simulator')
qregrn1 = QuantumRegister(4)
cregrn1 = ClassicalRegister(4)
ancillaryrn1 = QuantumRegister(2)
qcrn1 = QuantumCircuit(qregrn1, ancillaryrn1, cregrn1)
qcrn1.h(range(4))
qcrn1.barrier(qregrn1)
qcrn1.cx(qregrn1[0],ancillaryrn1[0])
qcrn1.cx(qregrn1[1],ancillaryrn1[0])
```

```
########################################
qcrn1.barrier(qregrn1)
########################################
qcrn1.cx(qregrn1[2],ancillaryrn1[1])
qcrn1.cx(qregrn1[3],ancillaryrn1[1])
########################################
qcrn1.barrier(qregrn1)
########################################
qcrn1.h(range(4))
for k in range(4):
    qcrn1.measure(qregrn1[k],cregrn1[k])
qcrn1.draw()
```

Code B.13.

```
aer_sim = Aer.get_backend('aer_simulator')
transpiled_qcrn1 = transpile(qcrn1, aer_sim)
qobj1 = assemble(transpiled_qcrn1)
results1 = aer_sim.run(qobj1,shots =10).result()
counts1 = results1.get_counts()
print(counts1)
plot_histogram(counts1)
```

Code B.14.

```
    def functionNm1(A):
    s1 = []
    s2 = []
    s3 = []
    for i in range(len(A)):
        for j in range(len(A)):
            for s in range(len(A)):
                for m in range(len(A)):
                    for k in range(len(A)):
                        s1.append(A[i][k,m]*A[k][s,j]- A[m][k,j]*A[i][s,k])
                    l1 = sum(s1)
                    s2.append(l1*x[m])
                    l2 = sum(s2)
                s3.append(l2)
    return(s3)
MN = quations(Selecting(functionNm1(matrices(MA1,MA2,MA3,MA4))))
print(MN)
```

Code B.15.

```
def functionNl1(A):
    s1 = []
    s2 = []
    s3 = []
    for i in range(len(A)):
        for j in range(len(A)):
            for s in range(len(A)):
                for m in range(len(A)):
                    for k in range(len(A)):
                        s1.append(A[m][k,i]*A[k][s,j]- A[i][k,j]*A[m][s,k])
                    l1 = sum(s1)
                    s2.append(l1*x[m])
                    l2 = sum(s2)
                s3.append(l2)
    return(s3)
LN = Equations(Selecting(functionNl1(matrices(MA1,MA2,MA3,MA4))))
print(LN)
```

Code B.16.

```
qregln = QuantumRegister(6) #------>33
cregln = ClassicalRegister(4)
qcln1 = QuantumCircuit(qregln,cregln)
qcln1.h(range(4))
qcln1.barrier(qregln)
qcln1.cx(qregln[1], qregln[4])
qcln1.cx(qregln[2], qregln[4])
qcln1.cx(qregln[3], qregln[4])
qcln1.barrier(qregln)
qcln1.cx(qregln[0], qregln[5])
qcln1.barrier(qregln)
qcln1.h(range(4))
qcln1.barrier(qregln)
for k in range(4):
    qcln1.measure(qregln[k],cregln[k])
qcln1.draw()
```

Code B.17.

```
transpiled_qcln1 = transpile(qcln1, aer_sim)
qobjln1 = assemble(transpiled_qcln1)
resultsln1 = aer_sim.run(qobjln1,shots =10).result()
countsln1 = resultsln1.get_counts()
print(countsln1)
plot_histogram(countsln1)
```

Code B.18.

```
qregn = QuantumRegister(8)
cregn = ClassicalRegister(4)

qcn = QuantumCircuit(qregn,cregn)

qcn.h(range(4))

##########################################
qcn.barrier(qregn)
##########################################

qcn.cx(qregn[0], qregn[4])
qcn.cx(qregn[1], qregn[4])
#########################################
qcn.barrier(qregn)
##########################################
qcn.cx(qregn[2], qregn[5])
qcn.cx(qregn[3], qregn[5])
##########################################
qcn.barrier(qregn)
##########################################
qcn.cx(qregn[1], qregn[6])
qcn.cx(qregn[2], qregn[6])
qcn.cx(qregn[3], qregn[6])
##########################################
qcn.barrier(qregn)
##########################################
qcn.cx(qregn[0], qregn[7])

qcn.h(range(4))

##########################################
qcn.barrier(qregn)
##########################################

for k in range(4):
    qcn.measure(qregn[k],cregn[k])

qcn.draw()
```

Code B.19.

```
transpiled_qcn = transpile(qcn, aer_sim)
qobjn = assemble(transpiled_qcn)
resultsn = aer_sim.run(qobjn,shots = 30).result()
countsn = resultsn.get_counts()
print(countsn)
plot_histogram(countsn)
```

Code B.20.

```
from itertools import accumulate
def functionZ(A):
    s1 = []
    s2 = []
    for i in range(len(A)):
        for k in range(len(A)):
            for m in range(len(A)):
                s1.append((A[m][k,i]- A[i][k,m])*x[m])
    Input = s1
    split = len(s1) / len(A)
    length_to_split = []
    for i in range(int(split)):
          length_to_split.append(len(A))
    Output = [Input[x - y: x] for x, y in zip(accumulate(length_to_split),
     ↪  length_to_split)]
    s4 = []
    for i in range(len(Output)):
        s3 = sum(Output[i])
        s4.append(s3)
    return(s4)
functionZ(matrices(MA1,MA2,MA3,MA4))
```

Code B.21.

```
qregc = QuantumRegister(4)
ancillaryc = QuantumRegister(6)
cregc = ClassicalRegister(4)

qcc = QuantumCircuit(qregc, ancillaryc, cregc)

qcc.h(range(4))

##########################################
qcc.barrier(qregc)
##########################################
qcc.cx(qregc[0],ancillaryc[0])
qcc.cx(qregc[1],ancillaryc[0])
```

```
######################################
qcc.barrier(qregc)
##########################################
qcc.cx(qregc[2],ancillaryc[1])
qcc.cx(qregc[3],ancillaryc[1])
######################################
qcc.barrier(qregc)

qcc.cx(qregc[1], ancillaryc[2])
qcc.cx(qregc[2], ancillaryc[2])
qcc.cx(qregc[3], ancillaryc[2])
qcc.barrier(qregc)
qcc.cx(qregc[0], ancillaryc[3])
qcc.barrier(qregc)
##########################################
qcc.barrier(qregc)
##########################################
qcc.cx(qregc[2], ancillaryc[4])
qcc.cx(qregc[1], ancillaryc[5])

##########################################
qcc.barrier(qregc)
##########################################
qcc.h(range(4))
##########################################
qcc.barrier(qregc)
##########################################
for k in range(4):
    qcc.measure(qregc[k],cregc[k])
qcc.draw()
```

Code B.22.

```
aer_sim = Aer.get_backend('aer_simulator')
transpiled_qcc = transpile(qcc, aer_sim)
qobjc1 = assemble(transpiled_qcc)
resultsc1 = aer_sim.run(qobjc1,shots = 50).result()
countsc1 = resultsc1.get_counts()
print(countsc1)
plot_histogram(countsc1, figsize=(12,5))
```

C Codes for Chapter 7

In this appendix we present the codes in python used in Chapter 7. Also they can be seen in *https://github.com/JMiguel01/Chapter-7-An-approach-to-the-Classification-*

of − Finite − Semifields − by − Quantum − Computing

Code C.23.

```
from sympy import symbols, Matrix, pprint, collect, factor
from sympy import *
import itertools
a = []
name = "a"
for i in range(0,13):
    v = symbols(name+str(i))
    a.append(v)
CN31 = Matrix([[1, 0, 0],[0, 1, 0],[0, 0, 1]])
CN32 = Matrix([[0, a[1], a[4]],[1, a[2], a[5]],[0, a[3], a[6]]])
CN33 = Matrix([[0, a[7], a[10]],[0, a[8], a[11]],[1, a[9], a[12]]])
aCN3=[]
cCN3=[]
for i in range(0,8):
    aCN3.append('{0:03b}'.format(i))
    M=int(aCN3[i][0])*CN31+int(aCN3[i][1])*CN32+int(aCN3[i][2])*CN33
    cCN3.append(M.det())
def Equations(First):
    BB31= First
    Pl1 =[]
    for i in range(len(BB31)):
        Pl1.append(dict(BB31[i].as_coefficients_dict()))
    ##Symplify computations of coefficients module 2
    EQ1 =[]
    for l in range(len(Pl1)):
        for s in Pl1[l]:
            if (Pl1[l][s] % 2) == 0:
                Pl1[l][s] = 0
            else:
                Pl1[l][s] = 1
        EQ1.append(sum([key * val for key, val in Pl1[l].items()]))
    ##Selecting the equations we need
    EQ3=[]
    for l in range(len(EQ1)):
        if (EQ1[l] != 0) and (EQ1[l]!=1):
            EQ3.append(EQ1[l])
    return(EQ3)
def Polynomial1(w):
    h_test1=[]
    w_test1=[]
    w_final_test1=[]
    for n in range(len(w)):
```

```
        if  (w[n] == 1) or (w[n] == 0):
            w_test1.append(w[n])
        else:
            h_test1.append(w[n].make_args(w[n]))
    g_test1 =[]
    for m in range(len(h_test1)):
        g_test1.append(list(h_test1[m]))
        for l in range(len(h_test1[m])):
            if(h_test1[m][l]==1):
                g_test1[m][1]=~(g_test1[m][1])
                del g_test1[m][l]
    w_final_test1 = '&'.join([str(item) for item in g_test1])
    w_final_test1 = w_final_test1.replace("*","&")
    w_final_test1 = w_final_test1.replace(","," ^")
    return(w_final_test1)
Polynomial1(Equations(cCN3))
```

Code C.24. *https://github.com/JMiguel01/Multiplication-table-GF-8*

Code C.25.

```
from qiskit.circuit import classical_function,  Int1
@classical_function
def grover_oracle(a1: Int1, a2: Int1, a3: Int1, a4: Int1, a5: Int1, a6:
 ↪  Int1, a7: Int1, a8: Int1, a9:Int1, a10: Int1, a11: Int1, a12:Int1) ->
 ↪  Int1:
    return ((a10&a8 ^ a11&a7)&(a1&a6 ^ a3&a4)&(a1&a11 ^ a1&a12 ^ a1&a5 ^
     ↪  a1&a6 ^ a10&a2 ^ a10&a3 ^ a10&a8 ^ a10&a9 ^ a11&a7 ^ a12&a7 ^ a2&a4
     ↪  ^ a3&a4 ^ a4&a8 ^ a4&a9 ^ a5&a7 ^ a6&a7)&((not a10) ^ a12 ^ a8 ^
     ↪  a10&a8 ^ a11&a7 ^ a11&a9 ^ a12&a8)&((not a1) ^ a2 ^ a6 ^ a1&a6 ^
     ↪  a2&a6 ^ a3&a4 ^ a3&a5)&((not a1) ^ a10 ^ a12 ^ a2 ^ a4 ^ a6 ^ a7 ^
     ↪  a8 ^ a1&a11 ^ a1&a12 ^ a1&a5 ^ a1&a6 ^ a10&a2 ^ a10&a3 ^ a10&a8 ^
     ↪  a10&a9 ^ a11&a3 ^ a11&a7 ^ a11&a9 ^ a12&a2 ^ a12&a7 ^ a12&a8 ^ a2&a4
     ↪  ^ a2&a6 ^ a3&a4 ^ a3&a5 ^ a4&a8 ^ a4&a9 ^ a5&a7 ^ a5&a9 ^ a6&a7 ^
     ↪  a6&a8))
quantum_circuitnc = grover_oracle.synth()
quantum_circuitnc.draw()

```

Code C.26. *https://github.com/JMiguel01/Chapter-7-An-approach-to-the-Classification-of-Finite-Semifields-by-Quantum-Computing*

Code C.27.

```
from sympy import symbols, Matrix, pprint, collect, factor
from sympy import *
```

```
import itertools

a = []
name = "a"
for i in range(0,13):
    v = symbols(name+str(i))
    a.append(v)

A1 = Matrix([[1, 0, 0, 0],[0, 1, 0 ,0],[0, 0, 1 , 0],[0,0,0,1] ])
A2 = Matrix([[0, 0, 0, 1],[1, 0, 0 ,1],[0, 1, 0 , 0],[0,0,1,0] ])
A3 = Matrix([[0, 0, a[1], a[5]],[0, 0, a[2] ,a[6]],[1, 0, a[3] ,
 ↪  a[7]],[0,1,a[4],a[8]] ])
A4 = Matrix([[0, 1, a[5], a[9]],[0, 1, a[6] ,a[10]],[0, 0, a[7] ,
 ↪  a[11]],[1,0,a[8],a[12]] ])
aC41=[]
A=[]
for i in range(0,16):
    aC41.append('{0:04b}'.format(i))
    M=int(aC41[i][0])*A1+int(aC41[i][1])*A2+int(aC41[i][2])*A3
     ↪  +int(aC41[i][3])*A4
    A.append(M.det())
def Equations(First):
    BB31= First
    Pl1 =[]
    for i in range(len(BB31)):
        Pl1.append(dict(BB31[i].as_coefficients_dict()))
    ##Symplify computations of coefficients module 2
    EQ1 =[]
    for l in range(len(Pl1)):
        for s in Pl1[l]:
            if (Pl1[l][s] % 2) == 0:
                Pl1[l][s] = 0
            else:
                Pl1[l][s] = 1
        EQ1.append(sum([key * val for key, val in Pl1[l].items()]))
    ##Selecting the equations we need
    EQ3=[]
    for l in range(len(EQ1)):
        if (EQ1[l] != 0) and (EQ1[l]!=1):
            EQ3.append(EQ1[l])
    return(EQ3)
def Polynomial1(w):
    h_test1=[]
    w_test1=[]
    w_final_test1=[]
```

```
    for n in range(len(w)):
        if  (w[n] == 1) or (w[n] == 0):
            w_test1.append(w[n])
        else:
            h_test1.append(w[n].make_args(w[n]))
    g_test1 =[]
    for m in range(len(h_test1)):
        g_test1.append(list(h_test1[m]))
        for l in range(len(h_test1[m])):
            if(h_test1[m][l]==1):
                g_test1[m][1]=~(g_test1[m][1])
                del g_test1[m][l]
    w_final_test1 = '&'.join([str(item) for item in g_test1])
    w_final_test1 = w_final_test1.replace("*","&")
    w_final_test1 = w_final_test1.replace(","," ^")
    return(w_final_test1)
Polynomial1(Equations(A))
```

$$
\begin{aligned}
f_1(a_1,\ldots,a_{12}) =& (a_{10}\wedge a_7 \oplus a_{11}\wedge a_5 \oplus a_{11}\wedge a_6 \oplus a_7\wedge a_9)\wedge(a_1\wedge a_6 \oplus a_2\wedge a_5)\\
&\wedge(a_1\wedge a_{10}\oplus a_1\wedge a_{11}\oplus a_1\wedge a_{12}\oplus a_1\wedge a_6\oplus a_1\wedge a_7\oplus a_1\wedge a_8\\
&\oplus a_{10}\wedge a_3\oplus a_{10}\wedge a_4\oplus a_{10}\wedge a_5\oplus a_{10}\wedge a_7\oplus a_{10}\wedge a_8\oplus a_{11}\wedge a_2\\
&\oplus a_{11}\wedge a_5\oplus a_{11}\wedge a_6\oplus a_{12}\wedge a_2\oplus a_{12}\wedge a_5\oplus a_{12}\wedge a_6\oplus a_2\wedge a_5\\
&\oplus a_2\wedge a_7\oplus a_2\wedge a_8\oplus a_2\wedge a_9\oplus a_3\wedge a_5\oplus a_3\wedge a_6\oplus a_3\wedge a_9\oplus a_4\\
&\wedge a_5\oplus a_4\wedge a_6\oplus a_4\wedge a_9\oplus a_6\wedge a_9\oplus a_7\wedge a_9\oplus a_8\wedge a_9)\wedge(\sim a_{11}\\
&\oplus a_5\oplus a_6\oplus a_8\oplus a_9\oplus a_{10}\wedge a_5\oplus a_{10}\wedge a_7\oplus a_{11}\wedge a_5\oplus a_{11}\wedge a_6\\
&\oplus a_{11}\wedge a_8\oplus a_{12}\wedge a_5\oplus a_{12}\wedge a_7\oplus a_6\wedge a_9\oplus a_7\wedge a_9\oplus a_8\wedge a_9)\\
&\wedge(\sim a_1\oplus a_2\oplus a_3\oplus a_4\oplus a_5\oplus a_1\wedge a_6\oplus a_1\wedge a_7\oplus a_1\wedge a_8\oplus a_2\\
&\wedge a_5\oplus a_3\wedge a_5\oplus a_4\wedge a_5)\wedge(\sim a_{10}\oplus a_{11}\oplus a_3\oplus a_4\oplus a_6\oplus a_8\\
&\oplus a_{10}\wedge a_3\oplus a_{10}\wedge a_4\oplus a_{10}\wedge a_7\oplus a_{10}\wedge a_8\oplus a_{11}\wedge a_2\oplus a_{11}\\
&\wedge a_4\oplus a_{11}\wedge a_6\oplus a_{11}\wedge a_8\oplus a_{12}\wedge a_2\oplus a_{12}\wedge a_3\oplus a_{12}\wedge a_6\\
&\oplus a_{12}\wedge a_7\oplus a_2\wedge a_7\oplus a_2\wedge a_8\oplus a_3\wedge a_6\oplus a_3\wedge a_8\oplus a_4\wedge a_6\\
&\oplus a_4\wedge a_7)\wedge(a_{10}\oplus a_{10}\wedge a_7\oplus a_{11}\wedge a_6)\wedge(\sim a_1\oplus a_3\oplus a_6\oplus a_8\\
&\oplus a_1\wedge a_6\oplus a_1\wedge a_8\oplus a_2\wedge a_5\oplus a_2\wedge a_7\oplus a_3\wedge a_6\oplus a_3\wedge a_8\oplus a_4\\
&\wedge a_5\oplus a_4\wedge a_7)\wedge(a_2\oplus a_6\oplus a_1\wedge a_{10}\oplus a_1\wedge a_6\oplus a_{10}\wedge a_4\oplus a_{10}\\
&\wedge a_5\oplus a_{10}\wedge a_8\oplus a_{12}\wedge a_2\oplus a_{12}\wedge a_6\oplus a_2\wedge a_5\oplus a_2\wedge a_8\oplus a_2\\
&\wedge a_9\oplus a_4\wedge a_6\oplus a_6\wedge a_9)\wedge(a_{11}\oplus a_{12}\oplus a_9\oplus a_{10}\wedge a_5\oplus a_{10}\\
&\wedge a_7\oplus a_{10}\wedge a_8\oplus a_{11}\wedge a_6\oplus a_{11}\wedge a_8\oplus a_{12}\wedge a_5\oplus a_{12}\wedge a_6\oplus a_{12}\\
&\wedge a_7\oplus a_6\wedge a_9\oplus a_8\wedge a_9)\wedge(a_1\oplus a_3\oplus a_4\oplus a_5\oplus a_7\oplus a_8\oplus a_1
\end{aligned}
$$

$$\wedge a_6 \oplus a_1 \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7) \wedge (\sim a_1 \oplus a_{11} \oplus a_{12} \oplus a_4 \oplus a_5 \oplus a_7 \oplus a_1 \wedge a_{11} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_7 \oplus a_1 \wedge a_8 \oplus a_{11} \wedge a_4 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_7 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_8 \oplus a_3 \wedge a_9 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_7 \oplus a_4 \wedge a_9 \oplus a_7 \wedge a_9 \oplus a_8 \wedge a_9)$$

Boolean function f_2

$$\begin{aligned} f_2(a_1,\ldots,a_{12}) =& (a_{10} \wedge a_7 \oplus a_{11} \wedge a_6) \wedge (a_1 \wedge a_6 \oplus a_2 \wedge a_5) \wedge (a_{10} \wedge a_3 \oplus a_{10} \wedge a_4 \oplus a_{10} \\ & \wedge a_7 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_6 \oplus a_{12} \wedge a_2 \oplus a_{12} \wedge a_6 \oplus a_2 \wedge a_7 \\ & \oplus a_2 \wedge a_8 \oplus a_3 \wedge a_6 \oplus a_4 \wedge a_6) \wedge (\sim a_{11} \oplus a_5 \oplus a_6 \oplus a_8 \oplus a_9 \oplus a_{10} \wedge a_5 \\ & \oplus a_{10} \wedge a_7 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_7 \oplus a_6 \\ & \wedge a_9 \oplus a_7 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_1 \wedge a_6 \oplus a_1 \\ & \wedge a_7 \oplus a_1 \wedge a_8 \oplus a_2 \wedge a_5 \oplus a_3 \wedge a_5 \oplus a_4 \wedge a_5) \wedge (\sim a_1 \oplus a_{10} \oplus a_{11} \oplus a_3 \\ & \oplus a_4 \oplus a_6 \oplus a_8 \oplus a_9 \oplus a_1 \wedge a_{10} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_8 \oplus a_{10} \wedge a_3 \\ & \oplus a_{10} \wedge a_4 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_4 \oplus a_{11} \\ & \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_2 \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \\ & \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_2 \wedge a_9 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_5 \\ & \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7 \oplus a_4 \wedge a_9 \oplus a_6 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_{12} \oplus a_7 \oplus a_9 \\ & \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_7 \oplus a_7 \wedge a_9) \wedge (\sim a_1 \oplus a_3 \oplus a_6 \oplus a_8 \\ & \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_8 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_5 \\ & \oplus a_4 \wedge a_7) \wedge (\sim a_1 \oplus a_{10} \oplus a_{12} \oplus a_2 \oplus a_3 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_1 \wedge a_{11} \\ & \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_7 \oplus a_1 \wedge a_8 \oplus a_{10} \wedge a_3 \oplus a_{10} \wedge a_4 \oplus a_{10} \wedge a_7 \oplus a_{10} \\ & \wedge a_8 \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_4 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_2 \\ & \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_3 \\ & \wedge a_5 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_3 \wedge a_9 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7 \oplus a_4 \\ & \wedge a_9 \oplus a_7 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_{10} \oplus a_6 \oplus a_8 \oplus a_9 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_8 \\ & \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_6 \oplus a_6 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_2 \oplus a_3 \oplus a_5 \oplus a_7 \oplus a_8 \\ & \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_6 \\ & \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7) \wedge (\sim a_{10} \oplus a_2 \oplus a_3 \oplus a_5 \oplus a_7 \oplus a_9 \oplus a_1 \\ & \wedge a_{10} \oplus a_1 \wedge a_{11} \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_{10} \wedge a_3 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \\ & \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_9 \oplus a_3 \wedge a_5 \\ & \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_9 \oplus a_6 \wedge a_9 \oplus a_7 \wedge a_9) \end{aligned}$$

Boolean function f_3

$$\begin{aligned} f_3(a_1,\ldots,a_{12}) =& (a_{10} \wedge a_5 \oplus a_6 \wedge a_9) \wedge (a_1 \wedge a_6 \oplus a_2 \wedge a_5) \wedge (a_1 \wedge a_{10} \oplus a_1 \wedge a_6 \oplus a_{10} \\ & \wedge a_5 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_9 \oplus a_6 \wedge a_9) \wedge (\sim a_{11} \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_9 \end{aligned}$$

$$\oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_7 \oplus a_6 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_1 \wedge a_8 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_8 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7) \wedge (\sim a_1 \oplus a_{10} \oplus a_{11} \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_8 \oplus a_1 \wedge a_{10} \oplus a_1 \wedge a_{11} \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_{10} \wedge a_3 \oplus a_{10} \wedge a_4 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_4 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_2 \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_2 \wedge a_9 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_3 \wedge a_9 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7 \oplus a_6 \wedge a_9 \oplus a_7 \wedge a_9) \wedge (\sim a_{10} \oplus a_{12} \oplus a_6 \oplus a_7 \oplus a_9 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \oplus a_6 \wedge a_9 \oplus a_7 \wedge a_9) \wedge (\sim a_1 \oplus a_3 \oplus a_6 \oplus a_8 \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_8 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_7) \wedge (\sim a_1 \oplus a_{12} \oplus a_2 \oplus a_3 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_1 \wedge a_{10} \oplus a_1 \wedge a_{11} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_1 \wedge a_8 \oplus a_{10} \wedge a_4 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_4 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_2 \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_8 \oplus a_2 \wedge a_9 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_8 \oplus a_3 \wedge a_9 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7 \oplus a_4 \wedge a_9 \oplus a_6 \wedge a_9 \oplus a_7 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_{10} \oplus a_{11} \oplus a_{12} \oplus a_5 \oplus a_8 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \oplus a_6 \wedge a_9 \oplus a_7 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_1 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_7 \oplus a_8 \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7) \wedge (\sim a_{10} \oplus a_{11} \oplus a_{12} \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_7 \oplus a_9 \oplus a_1 \wedge a_{10} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_8 \oplus a_{10} \wedge a_3 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_4 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_9 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_7 \oplus a_4 \wedge a_9 \oplus a_6 \wedge a_9 \oplus a_8 \wedge a_9)$$

$$f_4(a_1,\ldots,a_{12}) = (a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{11} \wedge a_6 \oplus a_6 \wedge a_9) \wedge (a_1 \wedge a_6 \oplus a_2 \wedge a_5) \wedge (a_{10} \wedge a_3 \oplus a_{10} \wedge a_4 \oplus a_{10} \wedge a_7 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_6 \oplus a_{12} \wedge a_2 \oplus a_{12} \wedge a_6 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_3 \wedge a_6 \oplus a_4 \wedge a_6) \wedge (\sim a_{11} \oplus a_6 \oplus a_8 \oplus a_{10} \wedge a_7 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_7) \wedge (\sim a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_1 \wedge a_8 \oplus a_2 \wedge a_5 \oplus a_3 \wedge a_5 \oplus a_4 \wedge a_5) \wedge (\sim a_1 \oplus a_{10} \oplus a_{11} \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_8 \oplus a_1 \wedge a_{10} \oplus a_1 \wedge a_{11} \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_{10} \wedge a_3 \oplus a_{10} \wedge a_4 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_2 \oplus a_{11} \wedge a_4 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_2 \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_2 \wedge a_9 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_3 \wedge a_9 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7 \oplus a_6 \wedge a_9 \oplus a_7 \wedge a_9) \wedge (\sim a_{10} \oplus a_{12} \oplus a_6 \oplus a_7 \oplus a_9 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_7 \oplus a_{10} \wedge a_8 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_{11} \wedge a_8 \oplus a_{12} \wedge a_6 \oplus a_{12} \wedge a_7 \oplus a_6 \wedge a_9 \oplus a_7 \wedge a_9) \wedge (\sim a_1 \oplus a_3 \oplus a_6 \oplus a_8 \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_8 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_7) \wedge (\sim a_1 \oplus a_{12} \oplus a_3 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_1 \wedge a_{11} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_7 \oplus a_1 \wedge a_8 \oplus a_{11} \wedge a_4 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge$$

$a_8 \oplus a_{12} \wedge a_3 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_7 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_8 \oplus a_3 \wedge a_9 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_7 \oplus a_4 \wedge a_9 \oplus a_7 \wedge a_9 \oplus a_8 \wedge a_9) \wedge (\sim a_{10} \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_9 \oplus a_{10} \wedge a_7 \oplus a_{11} \wedge a_5 \oplus a_{11} \wedge a_6 \oplus a_7 \wedge a_9) \wedge (\sim a_1 \oplus a_2 \oplus a_4 \oplus a_5 \oplus a_7 \oplus a_8 \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_7 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_7 \oplus a_2 \wedge a_8 \oplus a_3 \wedge a_5 \oplus a_3 \wedge a_6 \oplus a_3 \wedge a_8 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_7) \wedge (\sim a_1 \oplus a_{10} \oplus a_4 \oplus a_6 \oplus a_8 \oplus a_9 \oplus a_1 \wedge a_{10} \oplus a_1 \wedge a_{12} \oplus a_1 \wedge a_6 \oplus a_1 \wedge a_8 \oplus a_{10} \wedge a_4 \oplus a_{10} \wedge a_5 \oplus a_{10} \wedge a_8 \oplus a_{12} \wedge a_2 \oplus a_{12} \wedge a_5 \oplus a_{12} \wedge a_6 \oplus a_2 \wedge a_5 \oplus a_2 \wedge a_8 \oplus a_2 \wedge a_9 \oplus a_4 \wedge a_5 \oplus a_4 \wedge a_6 \oplus a_4 \wedge a_9 \oplus a_6 \wedge a_9 \oplus a_8 \wedge a_9)$

Code for boolean formula in section 7.2.

Code C.28.

```
import sympy
from sympy import symbols, Matrix, pprint, collect, factor
from sympy import *
import itertools
a1, a2, a3, a4, a5 = symbols('a1 a2 a3 a4 a5')
A31 = Matrix([[1, 0, 0],[0, 1, 0],[0, 0, 1]])
A32 = Matrix([[0, 0, 1],[1, 0, 1],[0, 1, 0]])
#A322 = Matrix([[0, 0, 1],[1, 0, 0],[0, 1, 1]])
A33 = Matrix([[0, 1, a1],[0, 1, a2],[1, 0, a3]])
#A332 = Matrix([[0, 1, a1],[0, 1, a2],[1, 0, a3]])
##checking for some matrices print((A33+A32)**3+(A33+A32)+A31)
 ↪  print((A32+A31)**3+(A32+A31)+A31) print((A33)**3+(A33)+A31

b31=[]
B31=[]
B32=[]
ONES=Matrix([[1, 1, 1],[1, 1, 1],[1, 1, 1]])
for i in range(0,4):
    b31.append('{0:03b}'.format(i))
    M31=int(b31[i][1])*A32+int(b31[i][2])*A33
    N31=M31**3+M31+A31+ONES
    N32=(M31+A31)**3+(M31+A31)+A31+ONES
    B31.append(N31)
    B32.append(N32)
#Now, we collect each polynomial from each entry of each matrix
def Equations(First,Second):
    BB31=[]
    BB32=[]
    for m in range(0,4):
        for n in range(0,9):
            BB31.append(expand(First[m][n]))
            BB32.append(expand(Second[m][n]))
            ##Reduce the exponent
    P11 =[]
    P12 = []
```

```
    for i in range(len(BB31)):
        for j in range(len(a)):
            for k in range(1,len(a)+1):
                BB31[i] = BB31[i].replace(a[j]**k,a[j])
                BB32[i] = BB32[i].replace(a[j]**k,a[j])

        Pl1.append(dict(BB31[i].as_coefficients_dict()))
        Pl2.append(dict(BB32[i].as_coefficients_dict()))
        #print(BB31)
    ##Symplify computations of coefficients module 2
    EQ1 =[]
    EQ2 =[]
    for l in range(len(Pl1)):
        for s in Pl1[l]:
            if (Pl1[l][s] % 2) == 0:
                Pl1[l][s] = 0
            else:
                Pl1[l][s] = 1
        EQ1.append(sum([key * val for key, val in Pl1[l].items()]))
    #print(EQ1)
    #print(len(EQ1))
    for i in range(len(Pl2)):
        for k in Pl2[i]:
            if (Pl2[i][k] % 2) == 0:
                Pl2[i][k] = 0
            else:
                Pl2[i][k] = 1
        EQ2.append(sum([key * val for key, val in Pl2[i].items()]))
    #print(EQ2)
    #print(len(EQ2))
    ##Selecting the equations we need
    EQ3=[]
    for l in range(len(EQ1)):
        if (EQ1[l] != 0) and (EQ1[l]!=1):
            EQ3.append(EQ1[l])
    #print(EQ3)
    #print(len(EQ3))
    EQ4=[]
    for l in range(len(EQ2)):
        if (EQ2[l] != 0) and (EQ2[l]!=1):
            EQ4.append(EQ2[l])
    #print(EQ4)
    #I MUST split the list print(EQ3) print(EQ3[8]) print(EQ3[9]) print(EQ4)
     ↪  print(EQ4[8]) print(EQ4[9])
    #Split the list of equations on the first half
```

```
    new1 = []
    new2 = []
    new3 = []
    for i in range(0, len(EQ3), 9):
        new1.append(EQ3[i : i+9])
        new2.append(EQ4[i : i+9])
    for j in range(len(new1)):
        new3.append(new1[j])
        new3.append(new2[j])
    return(new3)
print((Equations(B31,B32)))
def Polynomial1(w):
    h_test1=[]
    w_test1=[]
    w_final_test1=[]
    for n in range(len(w)):
        if  (w[n] == a1) or (w[n] == a2) or (w[n] == a3):
            w_test1.append(w[n])
        else:
            h_test1.append(w[n].make_args(w[n]))
    #print(h_test1)
    g_test1 =[]
    for m in range(len(h_test1)):
        g_test1.append(list(h_test1[m]))
        #print(g_test1)
#         g_test2 = []
        for l in range(len(h_test1[m])):
            #print(h_test1[m][l])
            if(h_test1[m][l]==1) and (h_test1[m][l+1]!=a1*a2):
                g_test1[m][1]=~(g_test1[m][1])
                del g_test1[m][l]
            if(h_test1[m][l]==1) and (h_test1[m][l+1]==a1*a2):
                g_test2 = (h_test1[m][l+1].make_args(h_test1[m][l+1]))
                g_test1[m][1]=(~(g_test2[0]) | ~(g_test2[1]))
                del g_test1[m][l]
                #print(g_test2[0])

                #g_test1[m][1]= ~(a[1])(*a[2])
                #del g_test1[m][l]
    w_final_test1 = '&'.join([str(item) for item in g_test1])
    w_final_test1 = w_final_test1.replace("*","&")
    w_final_test1 = w_final_test1.replace(","," ^")
    return(w_final_test1)
def booleanformula(w):
    boolean = []
```

```
    for i in range(len(w)):
        boolean.append(Polynomial1(w[i]))
    return(boolean)
booleanformula((Equations(B31,B32)))
```

Boolean Expression for $a \vee b$

Code C.29.

$[a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a13 \wedge a15 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a1 \wedge a2 \wedge a8 \oplus a1 \wedge a3 \oplus a1 \wedge a4 \wedge a8 \wedge a9 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a4 \wedge a6 \oplus a11 \wedge a13 \wedge a5 \oplus a11 \wedge a15 \wedge a3 \wedge a5 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a6 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \oplus a4 \wedge a6 \wedge a7 \oplus a4 \wedge a6 \wedge a8 \oplus a4 \wedge a6 \wedge (\sim a9)) \wedge (a1 \wedge a10 \wedge a11 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \oplus a1 \wedge a10 \wedge a13 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a9 \oplus a1 \wedge a10 \wedge a14 \wedge a8 \oplus a1 \wedge a13 \wedge a5 \wedge a8 \oplus a1 \wedge a3 \wedge a6 \oplus a1 \wedge a3 \wedge a8 \wedge a9 \oplus a1 \wedge a3 \wedge a8 \oplus a1 \wedge a4 \wedge a8 \oplus a1 \wedge a6 \wedge a9 \oplus a1 \wedge a7 \wedge a8 \oplus a1 \wedge a8 \wedge a9 \oplus a1 \wedge a8 \oplus a10 \wedge a11 \wedge a13 \wedge a5 \oplus a10 \wedge a11 \wedge a14 \oplus a10 \wedge a11 \wedge a15 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a7 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a12 \wedge a6 \oplus a10 \wedge a13 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a6 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \oplus a11 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a5 \wedge a6 \oplus a11 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a5 \wedge a6 \wedge a8 \oplus a2 \wedge a6 \wedge a8 \oplus a3 \wedge a4 \wedge a6 \wedge a8 \oplus a4 \wedge a6 \oplus a6 \wedge a9) \wedge (a1 \wedge a10 \wedge a13 \wedge a14 \wedge a5 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \wedge a4 \oplus a1 \wedge a10 \wedge a13 \wedge a2 \oplus a1 \wedge a10 \wedge a13 \wedge a4 \wedge a9 \oplus a1 \wedge a10 \wedge a14 \wedge a4 \wedge a8 \oplus a1 \wedge a12 \wedge a5 \wedge a8 \oplus a1 \wedge a13 \wedge a15 \wedge a5 \oplus a1 \wedge a13 \wedge a3 \wedge a5 \oplus a1 \wedge a13 \wedge a5 \oplus a1 \wedge a14 \wedge a15 \wedge a5 \wedge a8 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \wedge a9 \oplus a1 \wedge a2 \wedge a8 \wedge a9 \oplus a1 \wedge a3 \wedge a4 \wedge a8 \oplus a1 \wedge a4 \wedge a7 \wedge a8 \oplus a1 \wedge a4 \wedge a8 \wedge a9 \oplus a1 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a12 \wedge a5 \oplus a10 \wedge a11 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a11 \wedge a14 \wedge a4 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \wedge a2 \oplus a10 \wedge a11 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \wedge a3 \oplus a10 \wedge a11 \wedge a2 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \wedge a7 \oplus a10 \wedge a11 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a4 \wedge a6 \oplus a10 \wedge a13 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a2 \wedge a6 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a5 \wedge a6 \oplus a11 \wedge a14 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a4 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a2 \wedge a5 \wedge a8 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \wedge a9 \oplus a11 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \wedge a6 \oplus a12 \wedge a3 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a6 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \wedge a6 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \wedge a6 \oplus a13 \wedge a2 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a6 \wedge a7 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a6 \wedge a7 \oplus a2 \wedge a6 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \wedge a7 \oplus a3 \wedge a4 \wedge a6 \oplus a4 \wedge a6 \wedge a9 \oplus a4 \wedge a6) \wedge (a1 \wedge a10 \wedge a11 \wedge a15 \oplus a1 \wedge a10 \wedge a11 \wedge a3 \oplus a1 \wedge a10 \wedge a11 \wedge a9 \oplus a1 \wedge a10 \wedge a12 \wedge a8 \oplus a1 \wedge a10 \wedge a13 \wedge a14 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \oplus a1 \wedge a10 \wedge a13 \wedge a3 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a5 \oplus a1 \wedge a10 \wedge a13 \wedge a7 \oplus a1 \wedge a10 \wedge a13 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \oplus a1 \wedge a10 \wedge a14 \wedge a15 \wedge a8 \oplus a1 \wedge a10 \wedge a14 \wedge a3 \wedge a8 \oplus a1 \wedge a10 \wedge a14 \wedge a6 \oplus a1 \wedge a13 \wedge a15 \wedge a5 \wedge a8 \oplus a1 \wedge a13 \wedge a5 \wedge a6 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a1 \wedge a2 \wedge a8 \oplus a1 \wedge a3 \wedge a6 \wedge a9 \oplus a1 \wedge a3 \wedge a6 \oplus a1 \wedge a3 \wedge a7 \wedge a8 \oplus a1 \wedge a3 \wedge a8 \oplus a1 \wedge a4 \wedge a6 \wedge a8 \oplus a1 \wedge a6 \wedge a7 \oplus a1 \wedge a6 \wedge a9 \oplus a1 \wedge a6 \oplus a10 \wedge a11 \wedge a12 \oplus a10 \wedge a11 \wedge a13 \wedge a3 \wedge a5 \oplus a10 \wedge a11 \wedge a13 \wedge a4 \oplus a10 \wedge a11 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \wedge$

$$a4 \wedge a8 \oplus a10 \wedge a11 \wedge a15 \wedge a7 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a2 \wedge a8 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a5 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \wedge a6 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \wedge a6 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \wedge a6 \oplus a10 \wedge a13 \wedge a2 \wedge a6 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a6 \oplus a10 \wedge a14 \wedge a6 \wedge a9 \oplus a10 \wedge a14 \wedge a6 \oplus a11 \wedge a13 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \wedge a6 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \wedge a9 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \oplus a11 \wedge a3 \wedge a5 \wedge a6 \oplus a11 \wedge a3 \wedge a5 \wedge a8 \wedge a9 \oplus a11 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a11 \wedge a5 \wedge a6 \wedge a9 \oplus a11 \wedge a5 \wedge a7 \wedge a8 \oplus a11 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a5 \wedge a6 \wedge a8 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a2 \wedge a3 \wedge a6 \wedge a8 \oplus a2 \wedge a6 \oplus a3 \wedge a4 \wedge a6 \wedge a8 \oplus a3 \wedge a4 \wedge a6 \oplus a4 \wedge a6 \wedge a8 \wedge a9 \oplus a4 \wedge a6 \wedge a8 \oplus a6 \wedge a7 \wedge a9 \oplus a6 \wedge a7 \oplus a6 \wedge a9 \oplus a6) \wedge (a1 \wedge a10 \wedge a11 \wedge a14 \oplus a1 \wedge a10 \wedge a12 \wedge a13 \oplus a1 \wedge a10 \wedge a13 \wedge a14 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a14 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a10 \wedge a14 \wedge a8 \oplus a1 \wedge a11 \wedge a13 \wedge a5 \oplus a1 \wedge a11 \wedge a15 \wedge a3 \oplus a1 \wedge a11 \wedge a15 \oplus a1 \wedge a11 \wedge a3 \oplus a1 \wedge a11 \wedge a4 \wedge a8 \oplus a1 \wedge a11 \oplus a1 \wedge a12 \wedge a15 \wedge a8 \oplus a1 \wedge a12 \wedge a3 \wedge a8 \oplus a1 \wedge a12 \wedge a6 \oplus a1 \wedge a12 \wedge a8 \wedge a9 \oplus a1 \wedge a13 \wedge a15 \wedge a4 \wedge a8 \oplus a1 \wedge a13 \wedge a2 \wedge a8 \oplus a1 \wedge a13 \wedge a3 \oplus a1 \wedge a13 \wedge a4 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a15 \wedge a3 \wedge a8 \oplus a1 \wedge a14 \wedge a15 \wedge a6 \oplus a1 \wedge a14 \wedge a15 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a15 \wedge a8 \oplus a1 \wedge a14 \wedge a3 \wedge a6 \oplus a1 \wedge a14 \wedge a3 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a3 \wedge a8 \oplus a1 \wedge a14 \wedge a4 \wedge a8 \oplus a1 \wedge a14 \wedge a6 \wedge a9 \oplus a1 \wedge a14 \wedge a7 \wedge a8 \oplus a1 \wedge a14 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a8 \oplus a10 \wedge a11 \wedge a12 \wedge a9 \oplus a10 \wedge a11 \wedge a13 \wedge a2 \oplus a10 \wedge a11 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a11 \wedge a14 \wedge a15 \oplus a10 \wedge a11 \wedge a14 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a14 \wedge a7 \oplus a10 \wedge a11 \wedge a14 \wedge a9 \oplus a10 \wedge a11 \wedge a14 \oplus a10 \wedge a11 \wedge a4 \oplus a11 \wedge a12 \wedge a5 \wedge a8 \oplus a11 \wedge a13 \wedge a15 \wedge a5 \oplus a11 \wedge a13 \wedge a3 \wedge a5 \oplus a11 \wedge a13 \wedge a4 \wedge a5 \wedge a8 \oplus a11 \wedge a13 \wedge a5 \oplus a11 \wedge a14 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a14 \wedge a5 \wedge a8 \wedge a9 \oplus a11 \wedge a15 \wedge a4 \wedge a6 \oplus a11 \wedge a15 \oplus a11 \wedge a2 \wedge a6 \oplus a11 \wedge a3 \wedge a4 \wedge a6 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a6 \wedge a9 \oplus a11 \oplus a12 \wedge a13 \wedge a5 \wedge a6 \oplus a12 \wedge a15 \wedge a6 \wedge a9 \oplus a12 \wedge a15 \wedge a6 \oplus a12 \wedge a4 \wedge a6 \wedge a8 \oplus a12 \wedge a6 \wedge a7 \oplus a12 \wedge a6 \wedge a9 \oplus a13 \wedge a14 \wedge a3 \wedge a5 \wedge a6 \oplus a13 \wedge a14 \wedge a5 \wedge a6 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a6 \oplus a13 \wedge a15 \wedge a3 \wedge a4 \wedge a6 \oplus a13 \wedge a15 \wedge a4 \wedge a6 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a6 \oplus a13 \wedge a2 \wedge a3 \wedge a6 \oplus a13 \wedge a2 \wedge a6 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a6 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a6 \wedge a7 \oplus a13 \wedge a4 \wedge a6 \wedge a8 \oplus a13 \wedge a4 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a4 \wedge a6 \wedge a8 \oplus a14 \wedge a15 \wedge a6 \wedge a7 \oplus a14 \wedge a15 \wedge a6 \oplus a14 \wedge a2 \wedge a6 \wedge a8 \oplus a14 \wedge a3 \wedge a4 \wedge a6 \wedge a8 \oplus a14 \wedge a4 \wedge a6 \oplus a14 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a6 \wedge a9) \wedge (a1 \wedge a12 \wedge a5 \oplus a1 \wedge a4 \wedge a7 \oplus a10 \wedge a12 \wedge a14 \wedge a5 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \oplus a10 \wedge a12 \wedge a2 \oplus a10 \wedge a12 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a4 \wedge a9 \oplus a10 \wedge a13 \wedge a2 \wedge a4 \oplus a10 \wedge a14 \wedge a4 \wedge a7 \oplus a11 \wedge a2 \wedge a5 \oplus a12 \wedge a13 \wedge a5 \oplus a12 \wedge a15 \wedge a3 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \oplus a12 \wedge a3 \wedge a5 \oplus a12 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a7 \oplus a13 \wedge a15 \wedge a2 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a7 \oplus a14 \wedge a15 \wedge a5 \wedge a7 \oplus a14 \wedge a2 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a7 \oplus a14 \wedge a5 \wedge a7 \wedge a9 \oplus a2 \wedge a3 \wedge a7 \oplus a2 \wedge a3 \oplus a2 \wedge a4 \wedge a6 \oplus a2 \wedge a4 \wedge a8 \wedge a9 \oplus a2 \wedge a7 \wedge a9 \oplus a2 \wedge a8 \oplus a3 \wedge a4 \wedge a7 \wedge a9 \oplus a3 \wedge a4 \wedge a7 \oplus a4 \wedge a7 \wedge a8 \oplus a4 \wedge a7 \wedge a9 \oplus a4 \wedge a7) \wedge (a1 \wedge a2 \wedge a8 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a12 \wedge a13 \wedge a5 \oplus a10 \wedge a12 \wedge a14 \oplus a10 \wedge a12 \wedge a15 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a12 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a4 \wedge a7 \oplus a10 \wedge a14 \wedge a15 \wedge a7 \oplus a10 \wedge a14 \wedge a2 \wedge a8 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a2 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a3 \wedge a8 \wedge a9 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a4 \wedge a8 \oplus a2 \wedge a6 \wedge a9 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a7 \wedge a8 \oplus a4 \wedge a6 \wedge a7 \oplus a7 \wedge (\sim a9)) \wedge (a1 \wedge a10 \wedge a12 \wedge a4 \oplus a1 \wedge a12 \wedge a15 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a7 \oplus a1 \wedge a2 \wedge a3 \oplus a1 \wedge a2 \wedge a7 \oplus a1 \wedge a2 \oplus a1 \wedge a4 \wedge a7 \wedge a9 \oplus a10 \wedge a11 \wedge a2 \wedge a4 \oplus a10 \wedge a12 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a12 \wedge a14 \wedge a4 \oplus a10 \wedge a12 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \wedge a2 \oplus a10 \wedge a12 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \oplus$$

$$a10 \wedge a12 \wedge a2 \wedge a3 \oplus a10 \wedge a12 \wedge a2 \wedge a9 \oplus a10 \wedge a12 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a5 \oplus a10 \wedge a13 \wedge a14 \wedge a2 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \wedge a4 \wedge a9 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a4 \wedge a7 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \wedge a7 \oplus a10 \wedge a14 \wedge a2 \wedge a4 \wedge a8 \oplus a10 \wedge a14 \wedge a2 \wedge a7 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \wedge a7 \oplus a10 \wedge a14 \wedge a5 \wedge a7 \oplus a11 \wedge a12 \wedge a5 \oplus a11 \wedge a15 \wedge a2 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a7 \oplus a12 \wedge a14 \wedge a5 \wedge a8 \oplus a12 \wedge a15 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a15 \wedge a5 \wedge a7 \oplus a12 \wedge a15 \wedge a5 \oplus a12 \wedge a3 \wedge a5 \wedge a7 \oplus a12 \wedge a3 \wedge a5 \oplus a12 \wedge a4 \wedge a5 \wedge a6 \oplus a12 \wedge a4 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a5 \wedge a7 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \wedge a7 \oplus a13 \wedge a15 \wedge a2 \wedge a5 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \wedge a7 \oplus a13 \wedge a2 \wedge a3 \wedge a5 \oplus a13 \wedge a2 \wedge a5 \wedge a7 \oplus a13 \wedge a2 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a7 \wedge a9 \oplus a14 \wedge a15 \wedge a2 \wedge a5 \wedge a8 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \wedge a7 \oplus a14 \wedge a15 \wedge a5 \wedge a7 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a7 \oplus a14 \wedge a2 \wedge a5 \wedge a6 \oplus a14 \wedge a2 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \wedge a7 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \wedge a7 \oplus a14 \wedge a5 \wedge a7 \wedge a9 \oplus a14 \wedge a5 \wedge a7 \oplus a2 \wedge a3 \wedge a4 \wedge a8 \oplus a2 \wedge a3 \wedge a7 \wedge a9 \oplus a2 \wedge a3 \wedge a7 \oplus a2 \wedge a3 \oplus a2 \wedge a4 \wedge a6 \wedge a9 \oplus a2 \wedge a4 \wedge a7 \wedge a8 \oplus a2 \wedge a4 \wedge a8 \wedge a9 \oplus a2 \wedge a4 \wedge a8 \oplus a2 \wedge a6 \oplus a2 \wedge a7 \wedge a9 \oplus a2 \wedge a7 \oplus a2 \wedge a8 \wedge a9 \oplus a2 \oplus a4 \wedge a6 \wedge a7 \oplus a4 \wedge a7 \wedge a9) \wedge (a1 \wedge a10 \wedge a13 \wedge a2 \oplus a1 \wedge a12 \wedge a5 \wedge a8 \oplus a1 \wedge a2 \wedge a6 \oplus a1 \wedge a2 \wedge a8 \wedge a9 \oplus a1 \wedge a4 \wedge a7 \wedge a8 \oplus a10 \wedge a11 \wedge a12 \wedge a5 \oplus a10 \wedge a11 \wedge a15 \wedge a2 \oplus a10 \wedge a11 \wedge a2 \wedge a3 \oplus a10 \wedge a11 \wedge a2 \wedge a9 \oplus a10 \wedge a11 \wedge a4 \wedge a7 \oplus a10 \wedge a12 \wedge a13 \wedge a3 \wedge a5 \oplus a10 \wedge a12 \wedge a13 \wedge a4 \oplus a10 \wedge a12 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a12 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \wedge a4 \wedge a6 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a12 \oplus a10 \wedge a13 \wedge a14 \wedge a2 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \wedge a7 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \wedge a7 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \wedge a9 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a2 \wedge a5 \oplus a10 \wedge a13 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \wedge a7 \oplus a10 \wedge a14 \wedge a15 \wedge a2 \wedge a8 \oplus a10 \wedge a14 \wedge a15 \wedge a7 \oplus a10 \wedge a14 \wedge a2 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a2 \wedge a6 \oplus a10 \wedge a14 \wedge a7 \wedge a9 \oplus a10 \wedge a14 \wedge a7 \oplus a11 \wedge a2 \wedge a5 \wedge a8 \oplus a12 \wedge a13 \wedge a5 \wedge a8 \oplus a12 \wedge a15 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a15 \wedge a5 \wedge a6 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \oplus a12 \wedge a3 \wedge a5 \wedge a6 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \wedge a9 \oplus a12 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a5 \wedge a8 \oplus a13 \wedge a2 \wedge a5 \wedge a6 \oplus a13 \wedge a2 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a4 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a2 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \wedge a7 \oplus a2 \wedge a3 \wedge a6 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a6 \wedge a9 \oplus a2 \wedge a8 \wedge a9 \oplus a2 \wedge a8 \oplus a3 \wedge a4 \wedge a6 \wedge a7 \oplus a3 \wedge a4 \wedge a7 \wedge a8 \oplus a4 \wedge a7 \wedge a8 \wedge a9 \oplus a4 \wedge a7 \wedge a8) \wedge (a1 \wedge a11 \wedge a2 \oplus a1 \wedge a12 \wedge a13 \wedge a5 \oplus a1 \wedge a13 \wedge a15 \wedge a2 \oplus a1 \wedge a13 \wedge a4 \wedge a7 \oplus a1 \wedge a14 \wedge a2 \wedge a8 \oplus a10 \wedge a11 \wedge a12 \wedge a4 \oplus a10 \wedge a11 \wedge a14 \wedge a2 \oplus a10 \wedge a12 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a14 \wedge a15 \oplus a10 \wedge a12 \wedge a14 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \wedge a14 \wedge a7 \oplus a10 \wedge a12 \wedge a14 \wedge a9 \oplus a10 \wedge a12 \wedge a14 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a14 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a2 \wedge a4 \oplus a10 \wedge a14 \wedge a2 \wedge a8 \oplus a11 \wedge a12 \wedge a3 \wedge a5 \oplus a11 \wedge a14 \wedge a5 \wedge a7 \oplus a11 \wedge a15 \wedge a2 \wedge a3 \oplus a11 \wedge a15 \wedge a2 \oplus a11 \wedge a15 \wedge a4 \wedge a7 \oplus a11 \wedge a2 \wedge a3 \oplus a11 \wedge a2 \wedge a4 \wedge a8 \oplus a11 \wedge a2 \wedge a7 \oplus a11 \wedge a3 \wedge a4 \wedge a7 \oplus a11 \wedge a4 \wedge a7 \wedge a9 \oplus a12 \wedge a13 \wedge a15 \wedge a5 \oplus a12 \wedge a13 \wedge a3 \wedge a5 \oplus a12 \wedge a13 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a13 \wedge a5 \wedge a7 \oplus a12 \wedge a13 \wedge a5 \oplus a12 \wedge a14 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a14 \wedge a5 \wedge a6 \oplus a12 \wedge a14 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a15 \wedge a2 \wedge a8 \oplus a12 \wedge a15 \wedge a7 \wedge a9 \oplus a12 \wedge a15 \wedge a7 \oplus a12 \wedge a15 \oplus a12 \wedge a2 \wedge a3 \wedge a8 \oplus a12 \wedge a2 \wedge a6 \oplus a12 \wedge a2 \wedge a8 \wedge a9 \oplus a12 \wedge a4 \wedge a7 \wedge a8 \oplus a12 \wedge a5 \wedge a8 \oplus a12 \wedge a7 \wedge a9 \oplus a12 \wedge a7 \oplus a12 \oplus a13 \wedge a14 \wedge a3 \wedge a5 \wedge a7 \oplus a13 \wedge a14 \wedge a5 \wedge a7 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a4 \wedge a8 \oplus a13 \wedge a15 \wedge a2 \wedge a7 \oplus a13 \wedge a15 \wedge a2 \oplus a13 \wedge a15 \wedge a3 \wedge a4 \wedge a7 \oplus a13 \wedge a15 \wedge a4 \wedge a7 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a7 \oplus a13 \wedge a2 \wedge a3 \wedge a7 \oplus a13 \wedge a2 \wedge a3 \oplus a13 \wedge a2 \wedge a4 \wedge a6 \oplus a13 \wedge a2 \wedge a4 \wedge a8 \wedge a9 \oplus a13 \wedge a2 \wedge a7 \wedge a9 \oplus a13 \wedge a2 \wedge a8 \oplus a13 \wedge a3 \wedge a4 \wedge a7 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a7 \oplus a13 \wedge a4 \wedge a5 \wedge a7 \oplus$$

$$a13 \wedge a4 \wedge a7 \wedge a8 \oplus a13 \wedge a4 \wedge a7 \wedge a9 \oplus a13 \wedge a4 \wedge a7 \oplus a14 \wedge a15 \wedge a2 \wedge a3 \wedge a8 \oplus a14 \wedge a15 \wedge a2 \wedge a6 \oplus a14 \wedge a15 \wedge a2 \wedge a8 \wedge a9 \oplus a14 \wedge a15 \wedge a2 \wedge a8 \oplus a14 \wedge a15 \wedge a4 \wedge a7 \wedge a8 \oplus a14 \wedge a2 \wedge a3 \wedge a6 \oplus a14 \wedge a2 \wedge a3 \wedge a8 \wedge a9 \oplus a14 \wedge a2 \wedge a3 \wedge a8 \oplus a14 \wedge a2 \wedge a4 \wedge a8 \oplus a14 \wedge a2 \wedge a6 \wedge a9 \oplus a14 \wedge a2 \wedge a8 \wedge a9 \oplus a14 \wedge a3 \wedge a4 \wedge a7 \wedge a8 \oplus a14 \wedge a4 \wedge a6 \wedge a7 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a7 \wedge a9) \wedge (a1 \wedge a3 \oplus a1 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a4 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a5 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a2 \wedge a6 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a8 \oplus a3 \oplus a4 \wedge a6 \wedge a9 \oplus a4 \wedge a7 \wedge a8 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge (\sim a8)) \wedge (a1 \wedge a10 \wedge a13 \oplus a1 \wedge a6 \oplus a1 \wedge a8 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a3 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a12 \wedge a8 \oplus a10 \wedge a13 \wedge a14 \oplus a10 \wedge a13 \wedge a15 \wedge a3 \oplus a10 \wedge a13 \wedge a15 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \oplus a10 \wedge a13 \wedge a3 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \oplus a10 \wedge a13 \wedge a5 \oplus a10 \wedge a13 \wedge a7 \oplus a10 \wedge a13 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a8 \oplus a10 \wedge a14 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a6 \oplus a11 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \wedge a8 \oplus a13 \wedge a5 \wedge a6 \oplus a13 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a5 \wedge a8 \oplus a2 \wedge a8 \oplus a3 \wedge a6 \wedge a9 \oplus a3 \wedge a6 \oplus a3 \wedge a7 \wedge a8 \oplus a3 \wedge a8 \oplus a6 \wedge a7 \oplus a6 \wedge a9 \oplus a8 \wedge a9) \wedge (a1 \wedge a3 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a13 \wedge a5 \oplus a10 \wedge a12 \wedge a4 \wedge a8 \oplus a10 \wedge a13 \wedge a14 \wedge a4 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \wedge a7 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a14 \wedge a2 \wedge a8 \oplus a10 \wedge a14 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a3 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a3 \wedge a5 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a3 \wedge a5 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a6 \wedge a9 \oplus a2 \wedge a7 \wedge a8 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \oplus a3 \wedge a4 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a8 \oplus a4 \wedge a6 \wedge a7 \oplus a4 \wedge a6 \wedge a9 \oplus a4 \wedge a8 \wedge (\sim a9)) \wedge (a1 \wedge a10 \wedge a11 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \oplus a1 \wedge a10 \wedge a13 \wedge a9 \oplus a1 \wedge a10 \wedge a14 \wedge a8 \oplus a1 \wedge a3 \wedge a8 \oplus a1 \wedge a6 \wedge a9 \oplus a1 \wedge a7 \wedge a8 \oplus a1 \wedge a8 \wedge a9 \oplus a1 \wedge a8 \oplus a10 \wedge a11 \wedge a14 \oplus a10 \wedge a11 \wedge a15 \wedge a3 \oplus a10 \wedge a11 \wedge a15 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a3 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \oplus a10 \wedge a11 \wedge a7 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a12 \wedge a13 \oplus a10 \wedge a12 \wedge a15 \wedge a8 \oplus a10 \wedge a12 \wedge a3 \wedge a8 \oplus a10 \wedge a12 \wedge a6 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \wedge a8 \oplus a10 \wedge a13 \wedge a15 \wedge a3 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a7 \oplus a10 \wedge a13 \wedge a15 \oplus a10 \wedge a13 \wedge a3 \wedge a7 \oplus a10 \wedge a13 \wedge a3 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a15 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a8 \oplus a10 \wedge a14 \wedge a3 \wedge a6 \oplus a10 \wedge a14 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a8 \wedge a9 \oplus a10 \wedge a14 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \oplus a11 \wedge a5 \wedge a6 \oplus a11 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \wedge a6 \oplus a13 \wedge a15 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a5 \wedge a8 \oplus a13 \wedge a3 \wedge a5 \wedge a8 \oplus a13 \wedge a5 \wedge a6 \wedge a9 \oplus a13 \wedge a5 \wedge a7 \wedge a8 \oplus a13 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a5 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a3 \wedge a4 \wedge a8 \oplus a3 \wedge a6 \wedge a7 \oplus a3 \wedge a6 \oplus a3 \wedge a7 \wedge a8 \oplus a3 \wedge a8 \wedge a9 \oplus a3 \wedge a8 \oplus a4 \wedge a6 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8 \oplus a6 \wedge a9 \oplus a7 \wedge a8 \wedge a9 \oplus a7 \wedge a8 \oplus a8 \wedge a9 \oplus a8) \wedge (a1 \wedge a10 \wedge a13 \wedge a14 \oplus a1 \wedge a11 \wedge a15 \oplus a1 \wedge a12 \wedge a8 \oplus a1 \wedge a13 \wedge a15 \oplus a1 \wedge a13 \wedge a3 \oplus a1 \wedge a13 \oplus a1 \wedge a14 \wedge a15 \wedge a8 \oplus a1 \wedge a14 \wedge a6 \oplus a1 \wedge a14 \wedge a8 \wedge a9 \oplus a10 \wedge a11 \wedge a12 \oplus a10 \wedge a11 \wedge a14 \wedge a3 \oplus a10 \wedge a11 \wedge a14 \wedge a9 \oplus a10 \wedge a12 \wedge a13 \wedge a3 \oplus a10 \wedge a12 \wedge a13 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \wedge a15 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \oplus a10 \wedge a13 \wedge a14 \wedge a4 \wedge a8 \oplus a10 \wedge a13 \wedge a14 \wedge a7 \oplus a10 \wedge a13 \wedge a14 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a6 \oplus a11 \wedge a15 \wedge a4 \wedge a8 \oplus a11 \wedge a15 \oplus a11 \wedge a2 \wedge a8 \oplus a11 \wedge a3 \oplus a11 \wedge a4 \wedge a6 \oplus a11 \wedge a4 \wedge a8 \wedge a9 \oplus a11 \wedge a5 \oplus a12 \wedge a15 \wedge a3 \wedge a8 \oplus a12 \wedge a15 \wedge a6 \oplus a12 \wedge a15 \wedge a8 \wedge a9 \oplus a12 \wedge a15 \wedge a8 \oplus a12 \wedge$$

$$a3 \wedge a6 \oplus a12 \wedge a3 \wedge a8 \wedge a9 \oplus a12 \wedge a3 \wedge a8 \oplus a12 \wedge a4 \wedge a8 \oplus a12 \wedge a6 \wedge a9 \oplus a12 \wedge a7 \wedge a8 \oplus a12 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a8 \oplus a13 \wedge a15 \wedge a3 \oplus a13 \wedge a15 \wedge a4 \wedge a6 \oplus a13 \wedge a15 \wedge a4 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a15 \oplus a13 \wedge a2 \wedge a6 \oplus a13 \wedge a2 \wedge a8 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a8 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a3 \oplus a13 \wedge a4 \wedge a6 \wedge a9 \oplus a13 \wedge a4 \wedge a7 \wedge a8 \oplus a13 \wedge a4 \wedge a8 \wedge a9 \oplus a13 \wedge a4 \wedge a8 \oplus a13 \wedge a5 \oplus a13 \oplus a14 \wedge a15 \wedge a3 \wedge a6 \oplus a14 \wedge a15 \wedge a3 \wedge a8 \wedge a9 \oplus a14 \wedge a15 \wedge a4 \wedge a8 \oplus a14 \wedge a15 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a6 \oplus a14 \wedge a15 \wedge a7 \wedge a8 \oplus a14 \wedge a15 \wedge a8 \oplus a14 \wedge a2 \wedge a8 \oplus a14 \wedge a3 \wedge a6 \wedge a9 \oplus a14 \wedge a3 \wedge a6 \oplus a14 \wedge a3 \wedge a7 \wedge a8 \oplus a14 \wedge a3 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \oplus a14 \wedge a6 \wedge a7 \oplus a14 \wedge a6 \wedge a9 \oplus a14 \wedge a8 \wedge a9) \wedge (a1 \wedge a14 \wedge a5 \oplus a1 \wedge a2 \oplus a1 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \oplus a10 \wedge a14 \wedge a2 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \oplus a12 \wedge a3 \wedge a5 \oplus a12 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \oplus a13 \wedge a2 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a9 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \oplus a14 \wedge a15 \wedge a5 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \oplus a14 \wedge a3 \wedge a5 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \oplus a14 \wedge a5 \wedge a7 \oplus a14 \wedge a5 \wedge a9 \oplus a2 \wedge a3 \wedge a9 \oplus a2 \wedge a3 \oplus a2 \wedge a7 \oplus a2 \wedge a9 \oplus a3 \wedge a4 \wedge a7 \oplus a3 \wedge a4 \oplus a4 \wedge a6 \oplus a4 \wedge a9) \wedge (a1 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a4 \wedge a5 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a6 \oplus a3 \wedge a4 \wedge a6 \oplus a3 \wedge a4 \wedge a8 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8 \oplus a7 \wedge a9 \oplus a7 \oplus (\sim a9)) \wedge (a1 \wedge a10 \wedge a14 \wedge a4 \oplus a1 \wedge a12 \wedge a5 \oplus a1 \wedge a14 \wedge a15 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a9 \oplus a1 \wedge a2 \wedge a9 \oplus a1 \wedge a3 \wedge a4 \oplus a1 \wedge a4 \wedge a7 \oplus a1 \wedge a4 \wedge a9 \oplus a1 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \oplus a10 \wedge a12 \wedge a2 \oplus a10 \wedge a12 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a14 \wedge a4 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a2 \oplus a10 \wedge a14 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \oplus a10 \wedge a14 \wedge a2 \wedge a3 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a14 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a4 \oplus a11 \wedge a14 \wedge a5 \oplus a11 \wedge a15 \wedge a4 \wedge a5 \oplus a11 \wedge a2 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a9 \oplus a12 \wedge a13 \wedge a5 \oplus a12 \wedge a15 \wedge a3 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \wedge a9 \oplus a12 \wedge a15 \wedge a5 \oplus a12 \wedge a3 \wedge a5 \wedge a9 \oplus a12 \wedge a3 \wedge a5 \oplus a12 \wedge a5 \wedge a7 \oplus a12 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a5 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \oplus a13 \wedge a2 \wedge a5 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a7 \oplus a13 \wedge a4 \wedge a5 \wedge a9 \oplus a13 \wedge a4 \wedge a5 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a7 \oplus a14 \wedge a15 \wedge a5 \oplus a14 \wedge a3 \wedge a5 \wedge a7 \oplus a14 \wedge a3 \wedge a5 \oplus a14 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a9 \oplus a2 \wedge a3 \wedge a7 \oplus a2 \wedge a3 \oplus a2 \wedge a8 \oplus a2 \wedge a9 \oplus a3 \wedge a4 \wedge a7 \oplus a3 \wedge a4 \wedge a8 \oplus a3 \wedge a4 \wedge a9 \oplus a3 \wedge a4 \oplus a4 \wedge a7 \wedge a9 \oplus a4 \wedge a7 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8 \oplus a4 \wedge a9 \oplus a4) \wedge (a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a1 \wedge a2 \wedge a8 \oplus a1 \wedge a4 \wedge a6 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a13 \wedge a5 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \oplus a10 \wedge a13 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \wedge a5 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a13 \wedge a14 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \wedge a8 \oplus a13 \wedge a2 \wedge a5 \wedge a8 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \oplus a3 \wedge a4 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a8 \oplus a4 \wedge a6 \wedge a9 \oplus a4 \wedge a8 \wedge a9 \oplus a7 \wedge (\sim a9)) \wedge (a1 \wedge a11 \wedge a4 \oplus a1 \wedge a13 \wedge a14 \wedge a5 \oplus a1 \wedge a13 \wedge a15 \wedge a4 \oplus a1 \wedge a13 \wedge a2 \oplus a1 \wedge a13 \wedge a4 \wedge a9 \oplus a1 \wedge a14 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a11 \wedge a12 \wedge a5 \oplus a11 \wedge a14 \wedge a3 \wedge a5 \oplus a11 \wedge a14 \wedge a5 \wedge a9 \oplus a11 \wedge a15 \wedge a2 \oplus a11 \wedge a15 \wedge a3 \wedge a4 \oplus a11 \wedge a15 \wedge a4 \wedge a9 \oplus a11 \wedge a15 \wedge a4 \oplus a11 \wedge a2 \wedge a3 \oplus a11 \wedge a2 \wedge a9 \oplus a11 \wedge a3 \wedge$$

$$a4 \wedge a9 \oplus a11 \wedge a3 \wedge a4 \oplus a11 \wedge a4 \wedge a7 \oplus a11 \wedge a4 \wedge a8 \oplus a11 \wedge a4 \wedge a9 \oplus a12 \wedge a13 \wedge a3 \wedge a5 \oplus a12 \wedge a13 \wedge a5 \wedge a9 \oplus a12 \wedge a15 \wedge a4 \wedge a8 \oplus a12 \wedge a15 \wedge a7 \oplus a12 \wedge a15 \oplus a12 \wedge a2 \wedge a8 \oplus a12 \wedge a3 \wedge a4 \wedge a8 \oplus a12 \wedge a4 \wedge a6 \oplus a12 \wedge a9 \oplus a13 \wedge a14 \wedge a15 \wedge a5 \oplus a13 \wedge a14 \wedge a3 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a3 \wedge a5 \oplus a13 \wedge a14 \wedge a4 \wedge a5 \wedge a8 \oplus a13 \wedge a14 \wedge a5 \wedge a7 \oplus a13 \wedge a14 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \oplus a13 \wedge a15 \wedge a2 \wedge a3 \oplus a13 \wedge a15 \wedge a2 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \oplus a13 \wedge a15 \wedge a3 \wedge a4 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a7 \oplus a13 \wedge a15 \wedge a4 \wedge a8 \oplus a13 \wedge a15 \wedge a4 \oplus a13 \wedge a2 \wedge a3 \wedge a9 \oplus a13 \wedge a2 \wedge a3 \oplus a13 \wedge a2 \wedge a5 \oplus a13 \wedge a2 \wedge a7 \oplus a13 \wedge a2 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a7 \oplus a13 \wedge a3 \wedge a4 \oplus a13 \wedge a4 \wedge a5 \wedge a9 \oplus a13 \wedge a4 \wedge a6 \oplus a13 \wedge a4 \wedge a9 \oplus a14 \wedge a15 \wedge a2 \wedge a8 \oplus a14 \wedge a15 \wedge a3 \wedge a4 \wedge a8 \oplus a14 \wedge a15 \wedge a4 \wedge a6 \oplus a14 \wedge a15 \wedge a4 \wedge a8 \oplus a14 \wedge a15 \wedge a7 \oplus a14 \wedge a15 \wedge a9 \oplus a14 \wedge a15 \oplus a14 \wedge a2 \wedge a3 \wedge a8 \oplus a14 \wedge a2 \wedge a6 \oplus a14 \wedge a3 \wedge a4 \wedge a6 \oplus a14 \wedge a3 \wedge a4 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a4 \wedge a8 \wedge a9 \oplus a14 \wedge a4 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a14 \wedge a7 \wedge a9 \oplus a14 \wedge a7 \oplus a14 \wedge a9 \oplus a14) \wedge (a1 \wedge a10 \wedge a4 \oplus a1 \wedge a15 \wedge a5 \oplus a10 \wedge a12 \wedge a5 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a14 \wedge a4 \oplus a10 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a15 \wedge a2 \oplus a10 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a15 \wedge a4 \oplus a10 \wedge a2 \wedge a3 \oplus a10 \wedge a2 \wedge a9 \oplus a10 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a3 \wedge a4 \oplus a10 \wedge a4 \wedge a7 \oplus a10 \wedge a4 \wedge a8 \oplus a10 \wedge a4 \wedge a9 \oplus a11 \wedge a5 \oplus a14 \wedge a5 \wedge a8 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \oplus a15 \wedge a5 \oplus a2 \wedge a5 \wedge a8 \oplus a3 \wedge a5 \oplus a4 \wedge a5 \wedge a6 \oplus a4 \wedge a5 \wedge a8 \wedge a9) \wedge (a1 \wedge a5 \wedge a8 \oplus a10 \wedge a11 \wedge a5 \oplus a10 \wedge a12 \oplus a10 \wedge a13 \wedge a3 \wedge a5 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a7 \oplus a10 \wedge a15 \oplus a10 \wedge a2 \wedge a8 \oplus a10 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a4 \wedge a6 \oplus a10 \wedge a9 \oplus a13 \wedge a5 \wedge a8 \oplus a15 \wedge a3 \wedge a5 \wedge a8 \oplus a15 \wedge a5 \wedge a6 \oplus a15 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a5 \wedge a8 \oplus a3 \wedge a5 \wedge a6 \oplus a3 \wedge a5 \wedge a8 \wedge a9 \oplus a3 \wedge a5 \wedge a8 \oplus a4 \wedge a5 \wedge a8 \oplus a5 \wedge a6 \wedge a9 \oplus a5 \wedge a7 \wedge a8 \oplus a5 \wedge a8 \wedge a9) \wedge (a1 \wedge a10 \wedge a14 \wedge a5 \oplus a1 \wedge a10 \wedge a15 \wedge a4 \oplus a1 \wedge a10 \wedge a2 \oplus a1 \wedge a10 \wedge a4 \wedge a9 \oplus a1 \wedge a15 \wedge a5 \oplus a1 \wedge a3 \wedge a5 \oplus a1 \wedge a5 \oplus a10 \wedge a12 \wedge a3 \wedge a5 \oplus a10 \wedge a12 \wedge a4 \oplus a10 \wedge a12 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a5 \oplus a10 \wedge a14 \wedge a2 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \wedge a9 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a14 \wedge a4 \wedge a5 \wedge a8 \oplus a10 \wedge a14 \wedge a5 \wedge a7 \oplus a10 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a14 \wedge a5 \oplus a10 \wedge a15 \wedge a2 \wedge a3 \oplus a10 \wedge a15 \wedge a2 \wedge a9 \oplus a10 \wedge a15 \wedge a2 \oplus a10 \wedge a15 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a15 \wedge a4 \wedge a7 \oplus a10 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a4 \oplus a10 \wedge a2 \wedge a3 \wedge a9 \oplus a10 \wedge a2 \wedge a3 \oplus a10 \wedge a2 \wedge a7 \oplus a10 \wedge a2 \wedge a9 \oplus a10 \wedge a3 \wedge a4 \wedge a7 \oplus a10 \wedge a3 \wedge a4 \oplus a10 \wedge a4 \wedge a6 \oplus a10 \wedge a4 \wedge a9 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a5 \oplus a14 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a2 \wedge a5 \wedge a8 \oplus a15 \wedge a3 \wedge a5 \oplus a15 \wedge a4 \wedge a5 \wedge a6 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \oplus a15 \wedge a5 \oplus a2 \wedge a5 \wedge a6 \oplus a2 \wedge a5 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a5 \wedge a8 \oplus a3 \wedge a5 \oplus a4 \wedge a5 \wedge a6 \wedge a9 \oplus a4 \wedge a5 \wedge a7 \wedge a8 \oplus a4 \wedge a5 \wedge a8 \wedge a9 \oplus a4 \wedge a5 \wedge a8 \oplus a5) \wedge (a1 \wedge a10 \wedge a13 \wedge a5 \oplus a1 \wedge a10 \wedge a4 \wedge a8 \oplus a1 \wedge a15 \wedge a5 \wedge a8 \oplus a1 \wedge a5 \wedge a6 \oplus a1 \wedge a5 \wedge a8 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \wedge a5 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a11 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a5 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a3 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a5 \oplus a10 \wedge a13 \wedge a4 \wedge a5 \wedge a8 \oplus a10 \wedge a13 \wedge a5 \wedge a7 \oplus a10 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a5 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a10 \wedge a15 \wedge a2 \wedge a8 \oplus a10 \wedge a15 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a4 \wedge a6 \oplus a10 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a7 \oplus a10 \wedge a15 \wedge a9 \oplus a10 \wedge a15 \oplus a10 \wedge a2 \wedge a3 \wedge a8 \oplus a10 \wedge a2 \wedge a6 \oplus a10 \wedge a3 \wedge a4 \wedge a6 \oplus a10 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a4 \wedge a8 \wedge a9 \oplus a10 \wedge a4 \wedge a8 \oplus a10 \wedge a7 \wedge a9 \oplus a10 \wedge a7 \oplus a10 \wedge a9 \oplus a10 \oplus a11 \wedge a5 \wedge a8 \oplus a13 \wedge a5 \wedge a6 \oplus a13 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a5 \wedge a8 \oplus a15 \wedge a3 \wedge a5 \wedge a6 \oplus a15 \wedge a3 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \oplus a15 \wedge a5 \wedge a6 \wedge a9 \oplus a15 \wedge a5 \wedge a6 \oplus a15 \wedge a5 \wedge a7 \wedge a8 \oplus a15 \wedge a5 \wedge a8 \oplus a2 \wedge a5 \wedge a8 \oplus a3 \wedge a5 \wedge a6 \wedge a9 \oplus a3 \wedge a5 \wedge a6 \oplus a3 \wedge a5 \wedge a7 \wedge a8 \oplus a3 \wedge a5 \wedge a8 \oplus a5 \wedge a6 \wedge a7 \oplus a5 \wedge a6 \wedge a9 \oplus a5 \wedge a8 \wedge a9) \wedge (a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a11 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a12 \wedge$$

$$a4 \wedge a8 \oplus a10 \wedge a12 \wedge a7 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \wedge a7 \oplus a10 \wedge a13 \wedge a4 \wedge a8 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a2 \wedge a8 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a14 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a9 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a3 \wedge a5 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a2 \wedge a5 \wedge a8 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a4 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \wedge (\sim a9)] \vee [a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a13 \wedge a15 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a1 \wedge a2 \wedge a8 \oplus a1 \wedge a3 \oplus a1 \wedge a4 \wedge a8 \wedge a9 \oplus a1 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a4 \wedge a6 \oplus a11 \wedge a13 \wedge a5 \oplus a11 \wedge a15 \wedge a3 \wedge a5 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a6 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \oplus a4 \wedge a6 \wedge a7 \oplus a4 \wedge a6 \wedge a8 \oplus a4 \wedge a6 \wedge a9 \oplus 1) \wedge (a1 \wedge a10 \wedge a11 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \oplus a1 \wedge a10 \wedge a13 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a9 \oplus a1 \wedge a10 \wedge a14 \wedge a8 \oplus a1 \wedge a13 \wedge a5 \wedge a8 \oplus a1 \wedge a3 \wedge a6 \oplus a1 \wedge a3 \wedge a8 \wedge a9 \oplus a1 \wedge a3 \wedge a8 \oplus a1 \wedge a4 \wedge a8 \oplus a1 \wedge a6 \wedge a9 \oplus a1 \wedge a7 \wedge a8 \oplus a1 \wedge a8 \wedge a9 \oplus a1 \wedge a8 \oplus a10 \wedge a11 \wedge a13 \wedge a5 \oplus a10 \wedge a11 \wedge a14 \oplus a10 \wedge a11 \wedge a15 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a7 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a12 \wedge a6 \oplus a10 \wedge a13 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a6 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \oplus a11 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a5 \wedge a6 \oplus a11 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a5 \wedge a6 \wedge a8 \oplus a2 \wedge a6 \wedge a8 \oplus a3 \wedge a4 \wedge a6 \wedge a8 \oplus a4 \wedge a6 \oplus a6 \wedge a9 \oplus a6) \wedge (a1 \wedge a10 \wedge a13 \wedge a14 \wedge a5 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \wedge a4 \oplus a1 \wedge a10 \wedge a13 \wedge a2 \oplus a1 \wedge a10 \wedge a13 \wedge a4 \wedge a9 \oplus a1 \wedge a10 \wedge a14 \wedge a4 \wedge a8 \oplus a1 \wedge a12 \wedge a5 \wedge a8 \oplus a1 \wedge a13 \wedge a15 \wedge a5 \oplus a1 \wedge a13 \wedge a3 \wedge a5 \oplus a1 \wedge a13 \wedge a5 \oplus a1 \wedge a14 \wedge a15 \wedge a5 \wedge a8 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \wedge a9 \oplus a1 \wedge a2 \wedge a8 \wedge a9 \oplus a1 \wedge a3 \wedge a4 \wedge a8 \oplus a1 \wedge a3 \oplus a1 \wedge a4 \wedge a7 \wedge a8 \oplus a1 \wedge a4 \wedge a8 \wedge a9 \oplus a1 \wedge a4 \wedge a8 \oplus a1 \oplus a10 \wedge a11 \wedge a12 \wedge a5 \oplus a10 \wedge a11 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a11 \wedge a14 \wedge a4 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \wedge a2 \oplus a10 \wedge a11 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \wedge a3 \oplus a10 \wedge a11 \wedge a2 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \wedge a7 \oplus a10 \wedge a11 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a4 \wedge a6 \oplus a10 \wedge a13 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a2 \wedge a6 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a5 \wedge a6 \oplus a11 \wedge a14 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a4 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a2 \wedge a5 \wedge a8 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a15 \wedge a5 \wedge a6 \oplus a12 \wedge a3 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a6 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \wedge a6 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \wedge a6 \oplus a13 \wedge a2 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a6 \wedge a7 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a6 \wedge a7 \oplus a2 \wedge a6 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \wedge a7 \oplus a3 \wedge a4 \wedge a6 \oplus a4 \wedge a6 \wedge a9) \wedge (a1 \wedge a10 \wedge a11 \wedge a15 \oplus a1 \wedge a10 \wedge a11 \wedge a3 \oplus a1 \wedge a10 \wedge a11 \wedge a9 \oplus a1 \wedge a10 \wedge a12 \wedge a8 \oplus a1 \wedge a10 \wedge a13 \wedge a14 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \oplus a1 \wedge a10 \wedge a13 \wedge a3 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a5 \oplus a1 \wedge a10 \wedge a13 \wedge a7 \oplus a1 \wedge a10 \wedge a13 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \oplus a1 \wedge a10 \wedge a14 \wedge a15 \wedge a8 \oplus a1 \wedge a10 \wedge a14 \wedge a3 \wedge a8 \oplus a1 \wedge a10 \wedge a14 \wedge a6 \oplus a1 \wedge a13 \wedge a15 \wedge a5 \wedge a8 \oplus a1 \wedge a13 \wedge a5 \wedge a6 \oplus a1 \wedge a13 \wedge a5 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a1 \wedge a2 \wedge a8 \oplus a1 \wedge a3 \wedge a6 \wedge a9 \oplus a1 \wedge a3 \wedge a6 \oplus a1 \wedge a3 \wedge a7 \wedge a8 \oplus a1 \wedge a3 \wedge a8 \oplus a1 \wedge a4 \wedge a6 \wedge a8 \oplus a1 \wedge a6 \wedge a7 \oplus a1 \wedge a6 \wedge a9 \oplus a1 \wedge a6 \oplus a1 \wedge a8 \oplus a10 \wedge a11 \wedge a12 \oplus a10 \wedge a11 \wedge a13 \wedge a3 \wedge a5 \oplus a10 \wedge a11 \wedge a13 \wedge a4 \oplus a10 \wedge a11 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge$$

$$a11 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a15 \wedge a7 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a2 \wedge a8 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a5 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a11 \oplus a10 \wedge a12 \wedge a15 \wedge a6 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \wedge a6 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \wedge a6 \oplus a10 \wedge a13 \wedge a2 \wedge a6 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a6 \oplus a10 \wedge a14 \wedge a6 \wedge a9 \oplus a10 \wedge a14 \wedge a6 \oplus a11 \wedge a13 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \wedge a6 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \wedge a9 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \oplus a11 \wedge a3 \wedge a5 \wedge a6 \oplus a11 \wedge a3 \wedge a5 \wedge a8 \wedge a9 \oplus a11 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a11 \wedge a5 \wedge a6 \wedge a9 \oplus a11 \wedge a5 \wedge a7 \wedge a8 \oplus a11 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a5 \wedge a6 \wedge a8 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a2 \wedge a3 \wedge a6 \wedge a8 \oplus a2 \wedge a6 \oplus a3 \wedge a4 \wedge a6 \wedge a8 \oplus a3 \wedge a4 \wedge a6 \oplus a4 \wedge a6 \wedge a8 \wedge a9 \oplus a4 \wedge a6 \wedge a8 \oplus a6 \wedge a7 \wedge a9 \oplus a6 \wedge a7) \wedge (a1 \wedge a10 \wedge a11 \wedge a14 \oplus a1 \wedge a10 \wedge a12 \wedge a13 \oplus a1 \wedge a10 \wedge a13 \wedge a14 \wedge a3 \oplus a1 \wedge a10 \wedge a13 \wedge a14 \wedge a9 \oplus a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a10 \wedge a14 \wedge a8 \oplus a1 \wedge a11 \wedge a13 \wedge a5 \oplus a1 \wedge a11 \wedge a15 \wedge a3 \oplus a1 \wedge a11 \wedge a15 \oplus a1 \wedge a11 \wedge a3 \oplus a1 \wedge a11 \wedge a4 \wedge a8 \oplus a1 \wedge a11 \oplus a1 \wedge a12 \wedge a15 \wedge a8 \oplus a1 \wedge a12 \wedge a3 \wedge a8 \oplus a1 \wedge a12 \wedge a6 \oplus a1 \wedge a12 \wedge a8 \wedge a9 \oplus a1 \wedge a13 \wedge a15 \wedge a4 \wedge a8 \oplus a1 \wedge a13 \wedge a2 \wedge a8 \oplus a1 \wedge a13 \wedge a3 \oplus a1 \wedge a13 \wedge a4 \wedge a8 \wedge a9 \oplus a1 \wedge a13 \oplus a1 \wedge a14 \wedge a15 \wedge a3 \wedge a8 \oplus a1 \wedge a14 \wedge a15 \wedge a6 \oplus a1 \wedge a14 \wedge a15 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a15 \wedge a8 \oplus a1 \wedge a14 \wedge a3 \wedge a6 \oplus a1 \wedge a14 \wedge a3 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a3 \wedge a8 \oplus a1 \wedge a14 \wedge a4 \wedge a8 \oplus a1 \wedge a14 \wedge a6 \wedge a9 \oplus a1 \wedge a14 \wedge a7 \wedge a8 \oplus a1 \wedge a14 \wedge a8 \wedge a9 \oplus a1 \wedge a14 \wedge a8 \oplus a10 \wedge a11 \wedge a12 \wedge a9 \oplus a10 \wedge a11 \wedge a13 \wedge a2 \oplus a10 \wedge a11 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a11 \wedge a14 \wedge a15 \oplus a10 \wedge a11 \wedge a14 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a14 \wedge a7 \oplus a10 \wedge a11 \wedge a14 \wedge a9 \oplus a10 \wedge a11 \wedge a14 \oplus a10 \wedge a11 \wedge a4 \oplus a11 \wedge a12 \wedge a5 \wedge a8 \oplus a11 \wedge a13 \wedge a15 \wedge a5 \oplus a11 \wedge a13 \wedge a3 \wedge a5 \oplus a11 \wedge a13 \wedge a4 \wedge a5 \wedge a8 \oplus a11 \wedge a13 \wedge a5 \oplus a11 \wedge a14 \wedge a3 \wedge a5 \wedge a8 \oplus a11 \wedge a14 \wedge a5 \wedge a8 \wedge a9 \oplus a11 \wedge a15 \wedge a4 \wedge a6 \oplus a11 \wedge a2 \wedge a6 \oplus a11 \wedge a3 \wedge a4 \wedge a6 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a6 \wedge a9 \oplus a12 \wedge a13 \wedge a5 \wedge a6 \oplus a12 \wedge a15 \wedge a6 \wedge a9 \oplus a12 \wedge a15 \wedge a6 \oplus a12 \wedge a4 \wedge a6 \wedge a8 \oplus a12 \wedge a6 \wedge a7 \oplus a12 \wedge a6 \wedge a9 \oplus a13 \wedge a14 \wedge a3 \wedge a5 \wedge a6 \oplus a13 \wedge a14 \wedge a5 \wedge a6 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a6 \oplus a13 \wedge a15 \wedge a3 \wedge a4 \wedge a6 \oplus a13 \wedge a15 \wedge a4 \wedge a6 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a6 \oplus a13 \wedge a2 \wedge a3 \wedge a6 \oplus a13 \wedge a2 \wedge a6 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a6 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a6 \wedge a7 \oplus a13 \wedge a4 \wedge a6 \wedge a8 \oplus a13 \wedge a4 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a4 \wedge a6 \wedge a8 \oplus a14 \wedge a15 \wedge a6 \wedge a7 \oplus a14 \wedge a15 \wedge a6 \oplus a14 \wedge a2 \wedge a6 \wedge a8 \oplus a14 \wedge a3 \wedge a4 \wedge a6 \wedge a8 \oplus a14 \wedge a4 \wedge a6 \oplus a14 \wedge a5 \wedge a6 \wedge a8 \oplus a14 \wedge a6 \wedge a9 \oplus a14 \wedge a6) \wedge (a1 \wedge a12 \wedge a5 \oplus a1 \wedge a4 \wedge a7 \oplus a10 \wedge a12 \wedge a14 \wedge a5 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \oplus a10 \wedge a12 \wedge a2 \oplus a10 \wedge a12 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a4 \wedge a9 \oplus a10 \wedge a13 \wedge a2 \wedge a4 \oplus a10 \wedge a14 \wedge a4 \wedge a7 \oplus a11 \wedge a2 \wedge a5 \oplus a12 \wedge a13 \wedge a5 \oplus a12 \wedge a15 \wedge a3 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \oplus a12 \wedge a3 \wedge a5 \oplus a12 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a7 \oplus a13 \wedge a15 \wedge a2 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a7 \oplus a14 \wedge a15 \wedge a5 \wedge a7 \oplus a14 \wedge a2 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a7 \oplus a14 \wedge a5 \wedge a7 \wedge a9 \oplus a2 \wedge a3 \wedge a7 \oplus a2 \wedge a3 \oplus a2 \wedge a4 \wedge a6 \oplus a2 \wedge a4 \wedge a8 \wedge a9 \oplus a2 \wedge a7 \wedge a9 \oplus a2 \wedge a8 \oplus a2 \oplus a3 \wedge a4 \wedge a7 \wedge a9 \oplus a3 \wedge a4 \wedge a7 \oplus a4 \wedge a7 \wedge a8 \oplus a4 \wedge a7 \wedge a9 \oplus a4 \wedge a7) \wedge (a1 \wedge a2 \wedge a8 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a12 \wedge a13 \wedge a5 \oplus a10 \wedge a12 \wedge a14 \oplus a10 \wedge a12 \wedge a15 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a12 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a4 \wedge a7 \oplus a10 \wedge a14 \wedge a15 \wedge a7 \oplus a10 \wedge a14 \wedge a2 \wedge a8 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a2 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a3 \wedge a8 \wedge a9 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a4 \wedge a8 \oplus a2 \wedge a6 \wedge a9 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a7 \wedge a8 \oplus a4 \wedge a6 \wedge a7 \oplus a7 \wedge a9 \oplus a7 \oplus 1) \wedge (a1 \wedge a10 \wedge a12 \wedge a4 \oplus a1 \wedge a12 \wedge a15 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a7 \oplus a1 \wedge a2 \wedge a3 \oplus a1 \wedge a2 \wedge a7 \oplus a1 \wedge a2 \oplus a1 \wedge a4 \wedge a7 \wedge a9 \oplus a10 \wedge a11 \wedge a2 \wedge a4 \oplus a10 \wedge a12 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a12 \wedge a14 \wedge a4 \oplus a10 \wedge a12 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \wedge a2 \oplus a10 \wedge a12 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \oplus a10 \wedge$$

$$a12\wedge a2\wedge a3\oplus a10\wedge a12\wedge a2\wedge a9\oplus a10\wedge a12\wedge a3\wedge a4\wedge a9\oplus a10\wedge a12\wedge a3\wedge a4\oplus a10\wedge a12\wedge a4\wedge a8\oplus a10\wedge a12\wedge a4\wedge a9\oplus a10\wedge a12\wedge a5\oplus a10\wedge a13\wedge a14\wedge a2\wedge a5\oplus a10\wedge a13\wedge a15\wedge a2\wedge a4\oplus a10\wedge a13\wedge a2\wedge a4\wedge a9\oplus a10\wedge a13\wedge a2\oplus a10\wedge a13\wedge a4\wedge a7\oplus a10\wedge a14\wedge a15\wedge a4\wedge a7\oplus a10\wedge a14\wedge a2\wedge a4\wedge a8\oplus a10\wedge a14\wedge a2\wedge a7\oplus a10\wedge a14\wedge a3\wedge a4\wedge a7\oplus a10\wedge a14\wedge a5\wedge a7\oplus a11\wedge a12\wedge a5\oplus a11\wedge a15\wedge a2\wedge a5\oplus a11\wedge a4\wedge a5\wedge a7\oplus a12\wedge a14\wedge a5\wedge a8\oplus a12\wedge a15\wedge a4\wedge a5\wedge a8\oplus a12\wedge a15\wedge a5\wedge a7\oplus a12\wedge a15\wedge a5\oplus a12\wedge a3\wedge a5\wedge a7\oplus a12\wedge a3\wedge a5\oplus a12\wedge a4\wedge a5\wedge a6\oplus a12\wedge a4\wedge a5\wedge a8\wedge a9\oplus a12\wedge a5\wedge a7\wedge a9\oplus a12\wedge a5\oplus a13\wedge a14\wedge a5\wedge a7\oplus a13\wedge a15\wedge a2\wedge a5\oplus a13\wedge a15\wedge a4\wedge a5\wedge a7\oplus a13\wedge a2\wedge a3\wedge a5\oplus a13\wedge a2\wedge a5\wedge a7\oplus a13\wedge a2\wedge a5\oplus a13\wedge a4\wedge a5\wedge a7\wedge a9\oplus a14\wedge a15\wedge a2\wedge a5\wedge a8\oplus a14\wedge a15\wedge a3\wedge a5\wedge a7\oplus a14\wedge a15\wedge a5\wedge a7\wedge a9\oplus a14\wedge a15\wedge a5\wedge a7\oplus a14\wedge a2\wedge a5\wedge a6\oplus a14\wedge a2\wedge a5\wedge a8\wedge a9\oplus a14\wedge a3\wedge a5\wedge a7\wedge a9\oplus a14\wedge a3\wedge a5\wedge a7\oplus a14\wedge a5\wedge a7\wedge a9\oplus a14\wedge a5\wedge a7\oplus a2\wedge a3\wedge a4\wedge a8\oplus a2\wedge a3\wedge a7\wedge a9\oplus a2\wedge a3\wedge a7\oplus a2\wedge a4\wedge a6\wedge a9\oplus a2\wedge a4\wedge a7\wedge a8\oplus a2\wedge a4\wedge a8\wedge a9\oplus a2\wedge a4\wedge a8\oplus a2\wedge a6\oplus a2\wedge a7\wedge a9\oplus a2\wedge a7\oplus a2\wedge a8\wedge a9\oplus a4\wedge a6\wedge a7\oplus a4\wedge a7\wedge a9\oplus a4\wedge a7)\wedge(a1\wedge a10\wedge a13\wedge a2\oplus a1\wedge a12\wedge a5\wedge a8\oplus a1\wedge a2\wedge a6\oplus a1\wedge a2\wedge a8\wedge a9\oplus a1\wedge a4\wedge a7\wedge a8\oplus a10\wedge a11\wedge a12\wedge a5\oplus a10\wedge a11\wedge a15\wedge a2\oplus a10\wedge a11\wedge a2\wedge a3\oplus a10\wedge a11\wedge a2\wedge a9\oplus a10\wedge a11\wedge a4\wedge a7\oplus a10\wedge a12\wedge a13\wedge a3\wedge a5\oplus a10\wedge a12\wedge a13\wedge a4\oplus a10\wedge a12\wedge a13\wedge a5\wedge a9\oplus a10\wedge a12\wedge a15\wedge a4\wedge a8\oplus a10\wedge a12\wedge a15\oplus a10\wedge a12\wedge a3\wedge a4\wedge a8\oplus a10\wedge a12\wedge a4\wedge a6\oplus a10\wedge a12\wedge a9\oplus a10\wedge a13\wedge a14\wedge a2\oplus a10\wedge a13\wedge a14\wedge a5\wedge a7\oplus a10\wedge a13\wedge a15\wedge a2\wedge a3\oplus a10\wedge a13\wedge a15\wedge a2\wedge a9\oplus a10\wedge a13\wedge a15\wedge a2\oplus a10\wedge a13\wedge a15\wedge a4\wedge a7\oplus a10\wedge a13\wedge a2\wedge a3\wedge a9\oplus a10\wedge a13\wedge a2\wedge a3\oplus a10\wedge a13\wedge a2\wedge a5\oplus a10\wedge a13\wedge a2\wedge a9\oplus a10\wedge a13\wedge a3\wedge a4\wedge a7\oplus a10\wedge a14\wedge a15\wedge a2\wedge a8\oplus a10\wedge a14\wedge a15\wedge a7\oplus a10\wedge a14\wedge a2\wedge a3\wedge a8\oplus a10\wedge a14\wedge a2\wedge a6\oplus a10\wedge a14\wedge a7\wedge a9\oplus a10\wedge a14\wedge a7\oplus a11\wedge a2\wedge a5\wedge a8\oplus a12\wedge a13\wedge a5\wedge a8\oplus a12\wedge a15\wedge a3\wedge a5\wedge a8\oplus a12\wedge a15\wedge a5\wedge a6\oplus a12\wedge a15\wedge a5\wedge a8\wedge a9\oplus a12\wedge a15\wedge a5\wedge a8\oplus a12\wedge a3\wedge a5\wedge a6\oplus a12\wedge a3\wedge a5\wedge a8\wedge a9\oplus a12\wedge a3\wedge a5\wedge a8\oplus a12\wedge a4\wedge a5\wedge a8\oplus a12\wedge a5\wedge a6\wedge a9\oplus a12\wedge a5\wedge a8\wedge a9\oplus a13\wedge a15\wedge a2\wedge a5\wedge a8\oplus a13\wedge a2\wedge a5\wedge a6\oplus a13\wedge a2\wedge a5\wedge a8\wedge a9\oplus a13\wedge a4\wedge a5\wedge a7\wedge a8\oplus a14\wedge a15\wedge a5\wedge a7\wedge a8\oplus a14\wedge a2\wedge a5\wedge a8\oplus a14\wedge a3\wedge a5\wedge a7\wedge a8\oplus a14\wedge a5\wedge a6\wedge a7\oplus a2\wedge a3\wedge a6\wedge a9\oplus a2\wedge a3\wedge a6\oplus a2\wedge a3\wedge a8\oplus a2\wedge a6\wedge a9\oplus a2\wedge a8\wedge a9\oplus a3\wedge a4\wedge a6\wedge a7\oplus a3\wedge a4\wedge a7\wedge a8\oplus a4\wedge a7\wedge a8\wedge a9\oplus a4\wedge a7\wedge a8\oplus a7\wedge a9\oplus a7)\wedge(a1\wedge a11\wedge a2\oplus a1\wedge a12\wedge a13\wedge a5\oplus a1\wedge a13\wedge a15\wedge a2\oplus a1\wedge a13\wedge a4\wedge a7\oplus a1\wedge a14\wedge a2\wedge a8\oplus a10\wedge a11\wedge a12\wedge a4\oplus a10\wedge a11\wedge a14\wedge a2\oplus a10\wedge a12\wedge a13\wedge a3\wedge a4\oplus a10\wedge a12\wedge a13\wedge a4\wedge a9\oplus a10\wedge a12\wedge a14\wedge a15\oplus a10\wedge a12\wedge a14\wedge a4\wedge a8\oplus a10\wedge a12\wedge a14\wedge a7\oplus a10\wedge a12\wedge a14\wedge a9\oplus a10\wedge a12\wedge a14\oplus a10\wedge a12\wedge a9\oplus a10\wedge a13\wedge a14\wedge a2\wedge a3\oplus a10\wedge a13\wedge a14\wedge a2\wedge a9\oplus a10\wedge a13\wedge a2\wedge a4\oplus a10\wedge a14\wedge a2\wedge a8\oplus a11\wedge a12\wedge a3\wedge a5\oplus a11\wedge a14\wedge a5\wedge a7\oplus a11\wedge a15\wedge a2\wedge a3\oplus a11\wedge a15\wedge a2\oplus a11\wedge a15\wedge a4\wedge a7\oplus a11\wedge a2\wedge a3\oplus a11\wedge a2\wedge a4\wedge a8\oplus a11\wedge a2\wedge a7\oplus a11\wedge a3\wedge a4\wedge a7\oplus a11\wedge a4\wedge a7\wedge a9\oplus a12\wedge a13\wedge a15\wedge a5\oplus a12\wedge a13\wedge a3\wedge a5\oplus a12\wedge a13\wedge a4\wedge a5\wedge a8\oplus a12\wedge a13\wedge a5\wedge a7\oplus a12\wedge a13\wedge a5\oplus a12\wedge a14\wedge a3\wedge a5\wedge a8\oplus a12\wedge a14\wedge a5\wedge a6\oplus a12\wedge a14\wedge a5\wedge a8\wedge a9\oplus a12\wedge a15\wedge a2\wedge a8\oplus a12\wedge a15\wedge a7\wedge a9\oplus a12\wedge a15\wedge a7\oplus a12\wedge a2\wedge a3\wedge a8\oplus a12\wedge a2\wedge a6\oplus a12\wedge a2\wedge a8\wedge a9\oplus a12\wedge a4\wedge a7\wedge a8\oplus a12\wedge a5\wedge a8\oplus a12\wedge a7\wedge a9\oplus a12\wedge a7\oplus a13\wedge a14\wedge a3\wedge a5\wedge a7\oplus a13\wedge a14\wedge a5\wedge a7\wedge a9\oplus a13\wedge a15\wedge a2\wedge a4\wedge a8\oplus a13\wedge a15\wedge a2\wedge a7\oplus a13\wedge a15\wedge a2\oplus a13\wedge a15\wedge a3\wedge a4\wedge a7\oplus a13\wedge a15\wedge a4\wedge a7\wedge a9\oplus a13\wedge a15\wedge a4\wedge a7\oplus a13\wedge a2\wedge a3\wedge a7\oplus a13\wedge a2\wedge a3\oplus a13\wedge a2\wedge a4\wedge a6\oplus a13\wedge a2\wedge a4\wedge a8\wedge a9\oplus a13\wedge a2\wedge a7\wedge a9\oplus a13\wedge a2\wedge a8\oplus a13\wedge a2\oplus a13\wedge a3\wedge a4\wedge a7\wedge a9\oplus a13\wedge a3\wedge a4\wedge a7\oplus a13\wedge a4\wedge a5\wedge a7\oplus a13\wedge a4\wedge a7\wedge a8\oplus a13\wedge a4\wedge a7\wedge a9\oplus$$

$$a13 \wedge a4 \wedge a7 \oplus a14 \wedge a15 \wedge a2 \wedge a3 \wedge a8 \oplus a14 \wedge a15 \wedge a2 \wedge a6 \oplus a14 \wedge a15 \wedge a2 \wedge a8 \wedge a9 \oplus a14 \wedge a15 \wedge a2 \wedge a8 \oplus a14 \wedge a15 \wedge a4 \wedge a7 \wedge a8 \oplus a14 \wedge a2 \wedge a3 \wedge a6 \oplus a14 \wedge a2 \wedge a3 \wedge a8 \wedge a9 \oplus a14 \wedge a2 \wedge a3 \wedge a8 \oplus a14 \wedge a2 \wedge a4 \wedge a8 \oplus a14 \wedge a2 \wedge a6 \wedge a9 \oplus a14 \wedge a2 \wedge a8 \wedge a9 \oplus a14 \wedge a3 \wedge a4 \wedge a7 \wedge a8 \oplus a14 \wedge a4 \wedge a6 \wedge a7 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a7 \wedge a9 \oplus a14 \wedge a7) \wedge (a1 \wedge a3 \oplus a1 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a4 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a5 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a2 \wedge a6 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a8 \oplus a4 \wedge a6 \wedge a9 \oplus a4 \wedge a7 \wedge a8 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8) \wedge (a1 \wedge a10 \wedge a13 \oplus a1 \wedge a6 \oplus a1 \wedge a8 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a3 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a12 \wedge a8 \oplus a10 \wedge a13 \wedge a14 \oplus a10 \wedge a13 \wedge a15 \wedge a3 \oplus a10 \wedge a13 \wedge a15 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \oplus a10 \wedge a13 \wedge a3 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \oplus a10 \wedge a13 \wedge a5 \oplus a10 \wedge a13 \wedge a7 \oplus a10 \wedge a13 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a8 \oplus a10 \wedge a14 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a6 \oplus a11 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \wedge a8 \oplus a13 \wedge a5 \wedge a6 \oplus a13 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a5 \wedge a8 \oplus a2 \wedge a8 \oplus a3 \wedge a6 \wedge a9 \oplus a3 \wedge a6 \oplus a3 \wedge a7 \wedge a8 \oplus a3 \wedge a8 \oplus a6 \wedge a7 \oplus a6 \wedge a9 \oplus a8 \wedge a9 \oplus a8) \wedge (a1 \wedge a3 \oplus a1 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a13 \wedge a5 \oplus a10 \wedge a12 \wedge a4 \wedge a8 \oplus a10 \wedge a13 \wedge a14 \wedge a4 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \wedge a7 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a14 \wedge a2 \wedge a8 \oplus a10 \wedge a14 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a5 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a3 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a3 \wedge a5 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a5 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a6 \wedge a9 \oplus a2 \wedge a7 \wedge a8 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \oplus a3 \wedge a4 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a8 \oplus a4 \wedge a6 \wedge a7 \oplus a4 \wedge a6 \wedge a9 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8 \oplus 1) \wedge (a1 \wedge a10 \wedge a11 \oplus a1 \wedge a10 \wedge a13 \wedge a15 \oplus a1 \wedge a10 \wedge a13 \wedge a9 \oplus a1 \wedge a10 \wedge a14 \wedge a8 \oplus a1 \wedge a3 \wedge a8 \oplus a1 \wedge a6 \wedge a9 \oplus a1 \wedge a7 \wedge a8 \oplus a1 \wedge a8 \wedge a9 \oplus a1 \wedge a8 \oplus a10 \wedge a11 \wedge a14 \oplus a10 \wedge a11 \wedge a15 \wedge a3 \oplus a10 \wedge a11 \wedge a15 \wedge a9 \oplus a10 \wedge a11 \wedge a15 \oplus a10 \wedge a11 \wedge a3 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \oplus a10 \wedge a11 \wedge a7 \oplus a10 \wedge a11 \wedge a9 \oplus a10 \wedge a12 \wedge a13 \oplus a10 \wedge a12 \wedge a15 \wedge a8 \oplus a10 \wedge a12 \wedge a3 \wedge a8 \oplus a10 \wedge a12 \wedge a6 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \wedge a8 \oplus a10 \wedge a13 \wedge a15 \wedge a3 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a7 \oplus a10 \wedge a13 \wedge a15 \oplus a10 \wedge a13 \wedge a3 \wedge a7 \oplus a10 \wedge a13 \wedge a3 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a9 \oplus a10 \wedge a13 \oplus a10 \wedge a14 \wedge a15 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a15 \wedge a6 \oplus a10 \wedge a14 \wedge a15 \wedge a8 \oplus a10 \wedge a14 \wedge a6 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a8 \oplus a11 \wedge a15 \wedge a5 \wedge a8 \oplus a11 \wedge a5 \wedge a6 \oplus a11 \wedge a5 \wedge a8 \wedge a9 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \wedge a6 \oplus a13 \wedge a15 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a5 \wedge a8 \oplus a13 \wedge a3 \wedge a5 \wedge a8 \oplus a13 \wedge a5 \wedge a6 \wedge a9 \oplus a13 \wedge a5 \wedge a7 \wedge a8 \oplus a13 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a5 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a3 \wedge a4 \wedge a8 \oplus a3 \wedge a6 \wedge a7 \oplus a3 \wedge a6 \oplus a3 \wedge a7 \wedge a8 \oplus a3 \wedge a8 \wedge a9 \oplus a4 \wedge a6 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8 \oplus a6 \wedge a9 \oplus a6 \oplus a7 \wedge a8 \wedge a9 \oplus a7 \wedge a8) \wedge (a1 \wedge a10 \wedge a13 \wedge a14 \oplus a1 \wedge a11 \wedge a15 \oplus a1 \wedge a12 \wedge a8 \oplus a1 \wedge a13 \wedge a15 \oplus a1 \wedge a13 \wedge a3 \oplus a1 \wedge a13 \oplus a1 \wedge a14 \wedge a15 \wedge a8 \oplus a1 \wedge a14 \wedge a6 \oplus a1 \wedge a14 \wedge a8 \wedge a9 \oplus a10 \wedge a11 \wedge a12 \oplus a10 \wedge a11 \wedge a14 \wedge a3 \oplus a10 \wedge a11 \wedge a14 \wedge a9 \oplus a10 \wedge a12 \wedge a13 \wedge a3 \oplus a10 \wedge a12 \wedge a13 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \wedge a15 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \oplus a10 \wedge a13 \wedge a14 \wedge a4 \wedge a8 \oplus a10 \wedge a13 \wedge a14 \wedge a7 \oplus a10 \wedge a13 \wedge a14 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a3 \wedge a8 \oplus a10 \wedge a14 \wedge a6 \oplus a11 \wedge a15 \wedge a4 \wedge a8 \oplus a11 \wedge a15 \oplus a11 \wedge a2 \wedge a8 \oplus a11 \wedge a3 \oplus a11 \wedge a4 \wedge a6 \oplus a11 \wedge a4 \wedge a8 \wedge a9 \oplus a11 \wedge a5 \oplus a11 \oplus a12 \wedge a15 \wedge a3 \wedge a8 \oplus a12 \wedge a15 \wedge a6 \oplus a12 \wedge a15 \wedge a8 \wedge a9 \oplus a12 \wedge a15 \wedge a8 \oplus a12 \wedge a3 \wedge a6 \oplus a12 \wedge a3 \wedge a8 \wedge$$

$$a9 \oplus a12 \wedge a3 \wedge a8 \oplus a12 \wedge a4 \wedge a8 \oplus a12 \wedge a6 \wedge a9 \oplus a12 \wedge a7 \wedge a8 \oplus a12 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a8 \oplus a13 \wedge a15 \wedge a3 \oplus a13 \wedge a15 \wedge a4 \wedge a6 \oplus a13 \wedge a15 \wedge a4 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a2 \wedge a6 \oplus a13 \wedge a2 \wedge a8 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a8 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a4 \wedge a6 \wedge a9 \oplus a13 \wedge a4 \wedge a7 \wedge a8 \oplus a13 \wedge a4 \wedge a8 \wedge a9 \oplus a13 \wedge a4 \wedge a8 \oplus a13 \wedge a5 \oplus a14 \wedge a15 \wedge a3 \wedge a6 \oplus a14 \wedge a15 \wedge a3 \wedge a8 \wedge a9 \oplus a14 \wedge a15 \wedge a4 \wedge a8 \oplus a14 \wedge a15 \wedge a6 \wedge a9 \oplus a14 \wedge a15 \wedge a6 \oplus a14 \wedge a15 \wedge a7 \wedge a8 \oplus a14 \wedge a15 \wedge a8 \oplus a14 \wedge a2 \wedge a8 \oplus a14 \wedge a3 \wedge a6 \wedge a9 \oplus a14 \wedge a3 \wedge a6 \oplus a14 \wedge a3 \wedge a7 \wedge a8 \oplus a14 \wedge a3 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \oplus a14 \wedge a6 \wedge a7 \oplus a14 \wedge a6 \wedge a9 \oplus a14 \wedge a8 \wedge a9 \oplus a14 \wedge a8) \wedge (a1 \wedge a14 \wedge a5 \oplus a1 \wedge a2 \oplus a1 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \oplus a10 \wedge a14 \wedge a2 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \oplus a12 \wedge a3 \wedge a5 \oplus a12 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \oplus a13 \wedge a2 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a9 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \oplus a14 \wedge a15 \wedge a5 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \oplus a14 \wedge a3 \wedge a5 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \oplus a14 \wedge a5 \wedge a7 \oplus a14 \wedge a5 \wedge a9 \oplus a2 \wedge a3 \wedge a9 \oplus a2 \wedge a3 \oplus a2 \wedge a7 \oplus a2 \wedge a9 \oplus a3 \wedge a4 \wedge a7 \oplus a3 \wedge a4 \oplus a4 \wedge a6 \oplus a4 \wedge a9 \oplus a4) \wedge (a1 \wedge a4 \wedge a8 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a13 \wedge a14 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a4 \wedge a5 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a6 \oplus a3 \wedge a4 \wedge a6 \oplus a3 \wedge a4 \wedge a8 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8 \oplus a7 \wedge a9 \oplus a7) \wedge (a1 \wedge a10 \wedge a14 \wedge a4 \oplus a1 \wedge a12 \wedge a5 \oplus a1 \wedge a14 \wedge a15 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a9 \oplus a1 \wedge a2 \wedge a9 \oplus a1 \wedge a3 \wedge a4 \oplus a1 \wedge a4 \wedge a7 \oplus a1 \wedge a4 \wedge a9 \oplus a1 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a12 \wedge a15 \wedge a4 \oplus a10 \wedge a12 \wedge a2 \oplus a10 \wedge a12 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a14 \wedge a4 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a2 \oplus a10 \wedge a14 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a4 \oplus a10 \wedge a14 \wedge a2 \wedge a3 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a14 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a4 \oplus a11 \wedge a14 \wedge a5 \oplus a11 \wedge a15 \wedge a4 \wedge a5 \oplus a11 \wedge a2 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a9 \oplus a12 \wedge a13 \wedge a5 \oplus a12 \wedge a15 \wedge a3 \wedge a5 \oplus a12 \wedge a15 \wedge a5 \wedge a9 \oplus a12 \wedge a15 \wedge a5 \oplus a12 \wedge a3 \wedge a5 \wedge a9 \oplus a12 \wedge a3 \wedge a5 \oplus a12 \wedge a5 \wedge a7 \oplus a12 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \wedge a5 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \oplus a13 \wedge a2 \wedge a5 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a7 \oplus a13 \wedge a4 \wedge a5 \wedge a9 \oplus a13 \wedge a4 \wedge a5 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \wedge a9 \oplus a14 \wedge a15 \wedge a5 \wedge a7 \oplus a14 \wedge a15 \wedge a5 \oplus a14 \wedge a3 \wedge a5 \wedge a7 \oplus a14 \wedge a3 \wedge a5 \oplus a14 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a9 \oplus a14 \wedge a5 \oplus a2 \wedge a3 \wedge a7 \oplus a2 \wedge a3 \oplus a2 \wedge a8 \oplus a2 \wedge a9 \oplus a2 \oplus a3 \wedge a4 \wedge a7 \oplus a3 \wedge a4 \wedge a8 \oplus a3 \wedge a4 \wedge a9 \oplus a4 \wedge a7 \wedge a9 \oplus a4 \wedge a7 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8) \wedge (a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a1 \wedge a2 \wedge a8 \oplus a1 \wedge a4 \wedge a6 \oplus a10 \wedge a11 \wedge a14 \wedge a5 \oplus a10 \wedge a11 \wedge a15 \wedge a4 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a12 \wedge a13 \wedge a5 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a13 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a13 \wedge a15 \wedge a2 \oplus a10 \wedge a13 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \wedge a5 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a15 \wedge a5 \wedge a8 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a13 \wedge a14 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a4 \wedge a5 \wedge a8 \oplus a13 \wedge a2 \wedge a5 \wedge a8 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a15 \wedge a5 \wedge a6 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a2 \wedge a3 \wedge a6 \oplus a2 \wedge a3 \wedge a8 \oplus a2 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a6 \oplus a3 \wedge a4 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a8 \oplus a4 \wedge a6 \wedge a9 \oplus a4 \wedge a8 \wedge a9 \oplus a4 \wedge a8 \oplus a7 \wedge a9 \oplus a7 \oplus 1) \wedge (a1 \wedge a11 \wedge a4 \oplus a1 \wedge a13 \wedge a14 \wedge a5 \oplus a1 \wedge a13 \wedge a15 \wedge a4 \oplus a1 \wedge a13 \wedge a2 \oplus a1 \wedge a13 \wedge a4 \wedge a9 \oplus a1 \wedge a14 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a11 \wedge a12 \wedge a5 \oplus a11 \wedge a14 \wedge a3 \wedge a5 \oplus a11 \wedge a14 \wedge a5 \wedge a9 \oplus a11 \wedge a15 \wedge a2 \oplus a11 \wedge a15 \wedge a3 \wedge a4 \oplus a11 \wedge a15 \wedge a4 \wedge a9 \oplus a11 \wedge a15 \wedge a4 \oplus a11 \wedge a2 \wedge a3 \oplus a11 \wedge a2 \wedge a9 \oplus a11 \wedge a3 \wedge a4 \wedge a9 \oplus a11 \wedge a3 \wedge a4 \oplus a11 \wedge a4 \wedge a7 \oplus a11 \wedge a4 \wedge a8 \oplus a11 \wedge a4 \wedge a9 \oplus a12 \wedge a13 \wedge$$

$$a3 \wedge a5 \oplus a12 \wedge a13 \wedge a5 \wedge a9 \oplus a12 \wedge a15 \wedge a4 \wedge a8 \oplus a12 \wedge a15 \wedge a7 \oplus a12 \wedge a15 \oplus a12 \wedge a2 \wedge a8 \oplus a12 \wedge a3 \wedge a4 \wedge a8 \oplus a12 \wedge a4 \wedge a6 \oplus a12 \wedge a9 \oplus a12 \oplus a13 \wedge a14 \wedge a15 \wedge a5 \oplus a13 \wedge a14 \wedge a3 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a3 \wedge a5 \oplus a13 \wedge a14 \wedge a4 \wedge a5 \wedge a8 \oplus a13 \wedge a14 \wedge a5 \wedge a7 \oplus a13 \wedge a14 \wedge a5 \wedge a9 \oplus a13 \wedge a14 \wedge a5 \oplus a13 \wedge a15 \wedge a2 \wedge a3 \oplus a13 \wedge a15 \wedge a2 \wedge a9 \oplus a13 \wedge a15 \wedge a2 \oplus a13 \wedge a15 \wedge a3 \wedge a4 \wedge a9 \oplus a13 \wedge a15 \wedge a4 \wedge a7 \oplus a13 \wedge a15 \wedge a4 \wedge a8 \oplus a13 \wedge a15 \wedge a4 \oplus a13 \wedge a2 \wedge a3 \wedge a9 \oplus a13 \wedge a2 \wedge a3 \oplus a13 \wedge a2 \wedge a5 \oplus a13 \wedge a2 \wedge a7 \oplus a13 \wedge a2 \wedge a9 \oplus a13 \wedge a3 \wedge a4 \wedge a7 \oplus a13 \wedge a3 \wedge a4 \oplus a13 \wedge a4 \wedge a5 \wedge a9 \oplus a13 \wedge a4 \wedge a6 \oplus a13 \wedge a4 \wedge a9 \oplus a13 \wedge a4 \oplus a14 \wedge a15 \wedge a2 \wedge a8 \oplus a14 \wedge a15 \wedge a3 \wedge a4 \wedge a8 \oplus a14 \wedge a15 \wedge a4 \wedge a6 \oplus a14 \wedge a15 \wedge a4 \wedge a8 \oplus a14 \wedge a15 \wedge a7 \oplus a14 \wedge a15 \wedge a9 \oplus a14 \wedge a2 \wedge a3 \wedge a8 \oplus a14 \wedge a2 \wedge a6 \oplus a14 \wedge a3 \wedge a4 \wedge a6 \oplus a14 \wedge a3 \wedge a4 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a4 \wedge a8 \wedge a9 \oplus a14 \wedge a4 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \oplus a14 \wedge a7 \wedge a9 \oplus a14 \wedge a7) \wedge (a1 \wedge a10 \wedge a4 \oplus a1 \wedge a15 \wedge a5 \oplus a10 \wedge a12 \wedge a5 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a14 \wedge a4 \oplus a10 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a15 \wedge a2 \oplus a10 \wedge a15 \wedge a3 \wedge a4 \oplus a10 \wedge a15 \wedge a4 \wedge a9 \oplus a10 \wedge a15 \wedge a4 \oplus a10 \wedge a2 \wedge a3 \oplus a10 \wedge a2 \wedge a9 \oplus a10 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a3 \wedge a4 \oplus a10 \wedge a4 \wedge a7 \oplus a10 \wedge a4 \wedge a8 \oplus a10 \wedge a4 \wedge a9 \oplus a11 \wedge a5 \oplus a14 \wedge a5 \wedge a8 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \oplus a15 \wedge a5 \oplus a2 \wedge a5 \wedge a8 \oplus a3 \wedge a5 \oplus a4 \wedge a5 \wedge a6 \oplus a4 \wedge a5 \wedge a8 \wedge a9 \oplus a5) \wedge (a1 \wedge a5 \wedge a8 \oplus a10 \wedge a11 \wedge a5 \oplus a10 \wedge a12 \oplus a10 \wedge a13 \wedge a3 \wedge a5 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a7 \oplus a10 \wedge a15 \oplus a10 \wedge a2 \wedge a8 \oplus a10 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a4 \wedge a6 \oplus a10 \wedge a9 \oplus a10 \oplus a13 \wedge a5 \wedge a8 \oplus a15 \wedge a3 \wedge a5 \wedge a8 \oplus a15 \wedge a5 \wedge a6 \oplus a15 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a5 \wedge a8 \oplus a3 \wedge a5 \wedge a6 \oplus a3 \wedge a5 \wedge a8 \wedge a9 \oplus a3 \wedge a5 \wedge a8 \oplus a4 \wedge a5 \wedge a8 \oplus a5 \wedge a6 \wedge a9 \oplus a5 \wedge a7 \wedge a8 \oplus a5 \wedge a8 \wedge a9) \wedge (a1 \wedge a10 \wedge a14 \wedge a5 \oplus a1 \wedge a10 \wedge a15 \wedge a4 \oplus a1 \wedge a10 \wedge a2 \oplus a1 \wedge a10 \wedge a4 \wedge a9 \oplus a1 \wedge a15 \wedge a5 \oplus a1 \wedge a3 \wedge a5 \oplus a1 \wedge a5 \oplus a10 \wedge a12 \wedge a3 \wedge a5 \oplus a10 \wedge a12 \wedge a4 \oplus a10 \wedge a12 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a4 \oplus a10 \wedge a14 \wedge a15 \wedge a5 \oplus a10 \wedge a14 \wedge a2 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \wedge a9 \oplus a10 \wedge a14 \wedge a3 \wedge a5 \oplus a10 \wedge a14 \wedge a4 \wedge a5 \wedge a8 \oplus a10 \wedge a14 \wedge a5 \wedge a7 \oplus a10 \wedge a14 \wedge a5 \wedge a9 \oplus a10 \wedge a14 \wedge a5 \oplus a10 \wedge a15 \wedge a2 \wedge a3 \oplus a10 \wedge a15 \wedge a2 \wedge a9 \oplus a10 \wedge a15 \wedge a2 \oplus a10 \wedge a15 \wedge a3 \wedge a4 \wedge a9 \oplus a10 \wedge a15 \wedge a4 \wedge a7 \oplus a10 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a4 \oplus a10 \wedge a2 \wedge a3 \wedge a9 \oplus a10 \wedge a2 \wedge a3 \oplus a10 \wedge a2 \wedge a7 \oplus a10 \wedge a2 \wedge a9 \oplus a10 \wedge a3 \wedge a4 \wedge a7 \oplus a10 \wedge a3 \wedge a4 \oplus a10 \wedge a4 \wedge a6 \oplus a10 \wedge a4 \wedge a9 \oplus a10 \wedge a4 \oplus a12 \wedge a5 \wedge a8 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a5 \oplus a14 \wedge a5 \wedge a6 \oplus a14 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a2 \wedge a5 \wedge a8 \oplus a15 \wedge a3 \wedge a5 \oplus a15 \wedge a4 \wedge a5 \wedge a6 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \oplus a2 \wedge a5 \wedge a6 \oplus a2 \wedge a5 \wedge a8 \wedge a9 \oplus a3 \wedge a4 \wedge a5 \wedge a8 \oplus a4 \wedge a5 \wedge a6 \wedge a9 \oplus a4 \wedge a5 \wedge a7 \wedge a8 \oplus a4 \wedge a5 \wedge a8 \wedge a9 \oplus a4 \wedge a5 \wedge a8) \wedge (a1 \wedge a10 \wedge a13 \wedge a5 \oplus a1 \wedge a10 \wedge a4 \wedge a8 \oplus a1 \wedge a15 \wedge a5 \wedge a8 \oplus a1 \wedge a5 \wedge a6 \oplus a1 \wedge a5 \wedge a8 \wedge a9 \oplus a10 \wedge a11 \wedge a3 \wedge a5 \oplus a10 \wedge a11 \wedge a4 \oplus a10 \wedge a11 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a5 \oplus a10 \wedge a13 \wedge a2 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a3 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a5 \oplus a10 \wedge a13 \wedge a4 \wedge a5 \wedge a8 \oplus a10 \wedge a13 \wedge a5 \wedge a7 \oplus a10 \wedge a13 \wedge a5 \wedge a9 \oplus a10 \wedge a13 \wedge a5 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a10 \wedge a15 \wedge a2 \wedge a8 \oplus a10 \wedge a15 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a4 \wedge a6 \oplus a10 \wedge a15 \wedge a4 \wedge a8 \oplus a10 \wedge a15 \wedge a7 \oplus a10 \wedge a15 \wedge a9 \oplus a10 \wedge a2 \wedge a3 \wedge a8 \oplus a10 \wedge a2 \wedge a6 \oplus a10 \wedge a3 \wedge a4 \wedge a6 \oplus a10 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a4 \wedge a8 \wedge a9 \oplus a10 \wedge a4 \wedge a8 \oplus a10 \wedge a7 \wedge a9 \oplus a10 \wedge a7 \oplus a11 \wedge a5 \wedge a8 \oplus a13 \wedge a5 \wedge a6 \oplus a13 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a5 \wedge a8 \oplus a15 \wedge a3 \wedge a5 \wedge a6 \oplus a15 \wedge a3 \wedge a5 \wedge a8 \wedge a9 \oplus a15 \wedge a4 \wedge a5 \wedge a8 \oplus a15 \wedge a5 \wedge a6 \wedge a9 \oplus a15 \wedge a5 \wedge a6 \oplus a15 \wedge a5 \wedge a7 \wedge a8 \oplus a15 \wedge a5 \wedge a8 \oplus a2 \wedge a5 \wedge a8 \oplus a3 \wedge a5 \wedge a6 \wedge a9 \oplus a3 \wedge a5 \wedge a6 \oplus a3 \wedge a5 \wedge a7 \wedge a8 \oplus a3 \wedge a5 \wedge a8 \oplus a5 \wedge a6 \wedge a7 \oplus a5 \wedge a6 \wedge a9 \oplus a5 \wedge a8 \wedge a9 \oplus a5 \wedge a8) \wedge (a1 \wedge a10 \wedge a13 \wedge a4 \oplus a1 \wedge a11 \wedge a5 \oplus a1 \wedge a14 \wedge a5 \wedge a8 \oplus a10 \wedge a11 \wedge a2 \oplus a10 \wedge a11 \wedge a3 \wedge a4 \oplus a10 \wedge a11 \wedge a4 \wedge a9 \oplus a10 \wedge a12 \wedge a15 \oplus a10 \wedge a12 \wedge a4 \wedge a8 \oplus a10 \wedge a12 \wedge a7 \oplus a10 \wedge a12 \wedge a9 \oplus a10 \wedge a13 \wedge a15 \wedge a4 \oplus a10 \wedge a13 \wedge a2 \wedge a3 \oplus a10 \wedge a13 \wedge a2 \wedge a9 \oplus a10 \wedge a13 \wedge$$

$a3 \wedge a4 \wedge a9 \oplus a10 \wedge a13 \wedge a3 \wedge a4 \oplus a10 \wedge a13 \wedge a4 \wedge a7 \oplus a10 \wedge a13 \wedge a4 \wedge a8 \oplus a10 \wedge a13 \wedge a4 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \wedge a9 \oplus a10 \wedge a14 \wedge a15 \oplus a10 \wedge a14 \wedge a2 \wedge a8 \oplus a10 \wedge a14 \wedge a3 \wedge a4 \wedge a8 \oplus a10 \wedge a14 \wedge a4 \wedge a6 \oplus a10 \wedge a14 \wedge a9 \oplus a10 \wedge a14 \oplus a11 \wedge a15 \wedge a5 \oplus a11 \wedge a3 \wedge a5 \oplus a11 \wedge a4 \wedge a5 \wedge a8 \oplus a12 \wedge a3 \wedge a5 \wedge a8 \oplus a12 \wedge a5 \wedge a6 \oplus a12 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a15 \wedge a3 \wedge a5 \oplus a13 \wedge a15 \wedge a5 \oplus a13 \wedge a2 \wedge a5 \wedge a8 \oplus a13 \wedge a3 \wedge a5 \oplus a13 \wedge a4 \wedge a5 \wedge a6 \oplus a13 \wedge a4 \wedge a5 \wedge a8 \wedge a9 \oplus a13 \wedge a5 \oplus a14 \wedge a15 \wedge a5 \wedge a8 \oplus a14 \wedge a3 \wedge a5 \wedge a6 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \wedge a9 \oplus a14 \wedge a3 \wedge a5 \wedge a8 \oplus a14 \wedge a4 \wedge a5 \wedge a8 \oplus a14 \wedge a5 \wedge a6 \wedge a9 \oplus a14 \wedge a5 \wedge a7 \wedge a8 \oplus a14 \wedge a5 \wedge a8 \wedge (\sim a9)]$

Bibliography

[ADZ93] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Phys. Rev. A*, 48:1687–1690, Aug 1993.

[Aea19] Frank Arute and et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.

[AKR05] Andris Ambainis, Julia Kempe, and Alexander Rivosh. Coins make quantum walks faster. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 1099–1108, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[Alb60] A. A. Albert. Finite division algebras and finite planes. *Proc. Symp. Appl. Math*, 10:53–70, 1960.

[Alb61] A. A. Albert. Generalized twisted fields. *Pac. J. of Math.*, 11:1–8, 1961.

[AR20] Scott Aaronson and Patrick Rall. Quantum approximate counting, simplified. *Algorithms*, 24:32, 2020.

[AvDK+04] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Loyd, and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. Proceedings of, 45:42–51, 2004.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[BBHT98] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschr. Phys*, 46 (4-5):493–505, 1998.

[Ben80] P. Benioff. The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. Journal of statistical physics, 22 (5):563–591, 1980.

[BF28] M. Born and V. Fock. Beweis des adiabatensatzes. Zeitschrift für Physik, 51 (3-4):165–180, 1928.

[CCS99] Arjeh M. Cohen, Hans Cuypers, and Hans Sterk, editors. *Some tapas of computer algebra*, volume 4 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 1999.

[CEMM98] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci.*, 454(1969):339–354, 1998. Quantum coherence and decoherence (Santa Barbara, CA, 1996).

[CGC23] E. F. Combarro and S. González-Castillo. *A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms.* Packt Publising, ISBN: 978-1804613832, 2023.

[CQ18] Guangya Cai and Daowen Qiu. Optimal separation in exact query complexities for Simon's problem. *J. Comput. System Sci.*, 97:83–93, 2018.

[CRO+24] E. F. Combarro, I. F. Rúa, F. Orts, G. Ortega, A. M. Puertas, and E. M. Garzón. Quantum algorithms to compute the neighbour list of N-body simulations. *Quantum Inf. Process.*, 23(2):Paper No. 61, 2024.

[CRR11] E. F. Combarro, I. F. Rúa, and J. Ranilla. New advances in the computational exploration of semifields. *International Journal of Computer Mathematics*, 88 (9):1990–2000, 2011.

[CRR19a] E. F. Combarro, J. Ranilla, and I.F. Rúa. A quantum algorithm for the commutativity of finite dimensional algebras. *IEEE Access*, 7:45554–45562, 2019.

[CRR19b] Elías F. Combarro, José Ranilla, and I. F. Rúa. Quantum walks for the determination of commutativity of finite dimensional algebras. *J. Comput. Appl. Math.*, 354:496–506, 2019.

[CRR19c] Elías F. Combarro, José Ranilla, and Ignacio F. Rúa. Experiments testing the commutativity of finite-dimensional algebras with a quantum adiabatic algorithm. volume 1, pages e1009,11, 2019.

[CRR20] Elías F. Combarro, José Ranilla, and Ignacio Fernández Rúa. Quantum abstract detecting systems. *Quantum Information Processing*, 19(8):258, 2020.

[Dem08] Ulrich Dempwolff. Semifield planes of order 81. *J. Geom.*, 89(1-2):1–16, 2008.

[DF04] David S. Dummit and Richard M. Foote. *Abstract algebra*. John Wiley & Sons, Inc., Hoboken, NJ, third edition, 2004.

[Dic06] Leonard Eugene Dickson. Linear algebras in which division is always uniquely possible. *Transactions of the American Mathematical Society*, 7(3):370–390, 1906.

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 439(1907):553–558, 1992.

[DJCS21] Grinko Dmitry, Gacon Julien, Zoufal Christa, and Woerner Stefan. Iterative quantum amplitude estimation. *npj Quantum Inf*, 7(1):1–6, 2021.

[Fey82] R. Feynman. Simulating physics with computers. International Journal of Theoretical Physics, 21 (6):467–488, 1982.

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[FGGS00] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv: Quantum Physics*, 2000.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[HCCR22] J. M. Hernández Cáceres, E. F. Combarro, and I. F. Rúa. Combinatorial and rotational quantum abstract detecting systems. *Quantum Inf. Process.*, 21(2):Paper No. 66, 27, 2022.

[HCR23] J. M. Hernández Cáceres and I. F. Rúa. An approach to the classification of finite semifields by quantum computing. In *Non-associative algebras and related topics*, volume 427 of *Springer Proc. Math. Stat.*, pages 245–260. Springer, Cham, 2023.

[HK18] Kelsey Horan and Delaram Kahrobaei. The hidden subgroup problem and post-quantum group-based cryptography. In *Mathematical software—ICMS 2018*, volume 10931 of *Lecture Notes in Comput. Sci.*, pages 218–226. Springer, Cham, 2018.

[HR07] I. R. Hentzel and I. F. Rúa. Primitivity of finite semifields with 64 and 81 elements. *Internat. J. Algebra Comput.*, 17(7):1411–1429, 2007.

[J06] Watrous J. Quantum computation lecture notes. *Waterloo Lecture notes*, 2006.

[Jac45] N. Jacobson. Structure theory for algebraic algebras of bounded degree. *Ann. of Math*, 46:695–707, 1945.

[JER23] Hernández Cáceres J.M., Combarro Elías, and I.F. Rúa. Efficient quantum algorithms to find substructures on finite algebras. *Quantum Information & Computation*, 23 No.15& 16, 2023.

[Kit95] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv:quant-ph/9511026*, 1995.

[Kle60] Erwin Kleinfeld. Techniques for enumerating veblen-wedderburn systems. *J. ACM*, 7:330–337, 1960.

[KN98] T. Kadowaki and H. Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58.5355:5355–5363, 1998.

[Knu65] D. E. Knuth. Finite semifields and projective planes. Journal of Algebra, 2:182–217, 1965.

[LN83] R. Lidl and H. Niederreiter. Finite fields. Encyclopedia of mathematics and its applications, 20, 1983.

[LN96] Rudolf Lidl and Harald Niederreiter. *Finite fields and their applications*, volume 1 of *Handb. Algebr.* Elsevier/North-Holland, Amsterdam, 1996.

[Lom04] Chris Lomont. The hidden subgroup problem - review and open problems. *arXiv:quant-ph/0411037*, 2004.

[LP17] David A. Levin and Yuval Peres. *Markov chains and mixing times*. American Mathematical Society, Providence, RI, second edition, 2017.

[LS23] Michel Lavrauw and John Sheekey. Symplectic 4-dimensional semifields of order 84 and 94. *Designs, Codes and Cryptography*, 91:1–15, 02 2023.

[Man80] Y. Manin. Vychislimoe i nevychislimoe. Sov. Radio, pages 13–15, 1980.

[MN05] F. Magniez and A. Nayak. *Quantum Complexity of Testing Group Commutativity*. Springer, Lecture Notes in Computer Science 3580, 2005.

[MOS+19] Hamed Mohammadbagherpoor, Young-Hyun Oh, Anand Singh, Xianqing Yu, and Andy J. Rindos. Experimental challenges of implementing quantum phase estimation algorithms on ibm quantum computer. *arXiv*, 1903.07605, 2019.

[MW05] J. H. Maclagan-Wedderburn. A theorem on finite algebras. *Trans. Amer. Math. Soc.*, 6(3):349–352, 1905.

[Nak20] Kouhei Nakaji. Faster amplitude estimation. *Quantum Inf. Comput.*, 20(13-14):1109–1123, 2020.

[NC11] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011.

[OR07] David Oliveira and Rubens Ramos. Quantum bit string comparator: Circuits and applications. *Quantum Computers and Computing*, 7, 01 2007.

[OTT19] Thomas E O'Brien, Brian Tarasinski, and Barbara M Terhal. Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments. *New Journal of Physics*, 21(2):023022, feb 2019.

[Pak00] Igor Pak. Probability of generating a finite group. pages 1–22, 2000.

[Pak12] I. Pak. Testing commutativity of a group and the power of randomization. LMS Journal of Computation and Mathematics, 15:38–43, 2012.

[Pet33] M. Petrovic. Théorème sur les intégrales curvilignes. *Math, de l'Univ. Beograd*, 2:45–59, 1933.

[Por13] R. Portugal. *Quantum Walks and Search Algorithms*. Springer New York, 2013.

[Pso84] E. Psomopoulos. Commutativity theorems for rings and groups with constraints on commutators. *Int. J. Math*, 7 (3):513–517, 1984.

[Ral21] Patrick Rall. Faster coherent quantum algorithms for phase, energy, and amplitude estimation. *Quantum*, 5:566, 2021.

[RC12] I. F. Rúa and E. F. Combarro. Commutative semifields of order 3^5. *Comm. Algebra*, 40(3):988–996, 2012.

[RC18] Ignacio F. Rúa and Elías F. Combarro. *Cryptographic uncertainness: some experiments on finite semifield based substitution boxes*, volume 142 of *Stud. Syst. Decis. Control*. Springer, Cham, 2018.

[RCR09] I. F. Rúa, E. F. Combarro, and J. Ranilla. Classification of semifields of order 64. *J. of Algebra*, 322 (11):941–961, 2009.

[RCR12] I. F. Rúa, E. F. Combarro, and J. Ranilla. Determination of division algebras with 243 elements. *Finite Fields and Their Applications*, 18:1148–1155, 2012.

[Rú04] Ignacio Rúa. Primitive and non primitive finite semifields. *Communications in Algebra*, 32:793–803, 03 2004.

[San16] Raqueline A. M. Santos. Szegedy's quantum walk with queries. *Quantum Information Processing*, 15(11):4461–4475, 2016.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Sim94] D. R. Simon. On the power of quantum computation. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 116–123, USA, 1994. IEEE Computer Society.

[SUR+20] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. Amplitude estimation without phase estimation. *Quantum Information Processing*, 19(2), jan 2020.

[Sze04] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 32–41, Washington, DC, USA, 2004. IEEE Computer Society.

[Tse83] G. S. Tseitin. On the complexity of derivation in propositional calculus. pages 466–483, 1983.

[VBE96] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A (3)*, 54(1):147–153, 1996.

[vzGG99] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, New York, 1999.

[Wal62] R.J Walker. Determination of division algebras with 32 elements. *Proc. Symp. Appl. Math. AMS*, 15:83–85, 1962.

[Wie19] C. R. Wie. Simpler quantum counting. *Quantum Information and Computation*, 19(11 & 12), sep 2019.

[Won17] T.G. Wong. Equivalence of Szegedy's and coined quantum walks. *Quantum Inf Process*, 16 (215), 2017.

[XDS93] S.J. Xu, M. Darouach, and J. Schaefers. Expansion of det(a+b) and robustness analysis of uncertain state space systems. *IEEE Transactions on Automatic Control*, 38(11):1671–1675, 1993.

[YM08] Noson S. Yanofsky and Mirco Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008.

Index

Bell States, 26

Character, 19

Division Algebra, 14

Field, 11
Finite Algebras, 13
Finite Semifield, 16

Group, 9
    Orthogonal, 10
    Special Orthogonal, 10

Hadamard Test, 34

Ideal, 11
    Principal, 11
Integral Domain, 11
    Principle, 11

Ket, 23

Measurement, 27
Multi-qubit, 26
Multiplication table, 14

Oracle
    Phase, 38
Oracles, 37
Orthogonal subgroup, 20

QADS, 67
    Combinatorial, 73
    Rotational, 82
Quantum Algorithm, 49
    Grover's Search Algorithm, 61
    Hidden Subgroup Problem, 52
    Quantum Phase Estimation, 58
    Simon's Algorithm, 49
Quantum Circuit
    Efficient, 37
    Size, 36
Quantum Circuits, 31
Quantum Fourier Transform, 40
Quantum Gate, 28
    $\pi/8$ half-phase gate, 29
    CNOT gate, 30
    Hadamard gate, 29
    Not gate, 29
    Toffoli, 31
Qubit, 25
    Ancilla, 38
Quotient
    Ring, 11

Register, 26
Ring, 11
    Division Ring, 11
    Non-associative, 11

State
    Entangled, 26
    Superposition, 25
Structure Constants, 14
Subgroup, 9

Theorem
    Fundamental Theorem of Abelian Groups, 10

Lagrange Theorem, 10

Uncomputation, 38
Unit, 11
Unitary Matrix, 28

Zero Divisor, 11