Saarland University
Faculty of Mathematics and Computer Science
Department of Mathematics and Computer Science

Bachelor's thesis

Towards a Concrete Model for the Quantum Permutation Group

submitted by
Nicolas Faroß

submitted
October 16, 2020

Reviewers
Prof. Dr. Moritz Weber
Prof. Dr. Roland Speicher

Erklärung
Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement
I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung
Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent
I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, October 16, 2020

Abstract

In this thesis we consider a magic unitary M_3 and the C^*-algebra B_3 which is generated by the entries of M_3. We give evidence that B_3 is $*$-isomorphic to $C(S_4^+)$ which would imply that (B_3, M_3) is a somewhat simple and concrete model for the quantum permutation group S_4^+.

In particular, we show that each non-zero polynomial p in the generators of $C(S_4^+)$ up to degree 50 does not vanish in the entries of M_3. To prove this result, we designed an efficient algorithm using Gröbner bases and finite automata.

The model (B_3, M_3) was introduced by Jung-Weber while they presented series of increasing models for the quantum permutation groups S_n^+. The fact that no vanishing polynomial up to degree 50 exists indicates that no such polynomial exists for an arbitrary degree. This gives evidence that the C^*-algebra B_3 from Jung-Weber is already $*$-isomorphic to $C(S_4^+)$.

Contents

1 Introduction **1**

2 Preliminaries **3**
2.1 C^*-algebras 4
2.2 Compact matrix quantum groups 6
2.3 Gröbner bases 8
2.4 Finite automata 9

3 Models by Jung-Weber **11**
3.1 The ⏀-operator 11
3.2 Inverse system 12
3.3 Limit object 14
3.4 A concrete example 15

4 A concrete magic unitary **16**
4.1 The magic unitary ideal and separating polynomials 17
4.2 Main result 18
4.3 Dimension result 20

5 Polynomial and quotient bases **21**
5.1 Polynomial basis 22
5.2 Quotient basis 23
5.3 Dimension 25

6 Algorithms and time complexity **27**
6.1 Automaton construction 27
6.2 Algorithm 1: Matrix construction 30
6.3 Algorithm 2: Kernel of Ψ_m 32
6.4 Time complexity 33

7 Proofs of the main results **35**

A Finite automaton **37**

B Program output **39**

C Code **40**

1 Introduction

In this thesis we consider a magic unitary M_3 and the C^*-algebra B_3 which is generated by the entries of M_3. We give evidence that B_3 is $*$-isomorphic to $C(S_4^+)$ which would imply that (B_3, M_3) is a concrete model for the quantum permutation group S_4^+. In particular, we partially answer open questions about models of S_n^+ from Jung-Weber in [JW20].

The quantum permutation groups S_n^+ were first defined in [Wan98] by Wang. They are examples of compact matrix quantum groups which were introduced by Woronowicz in [Wor87]. Compact matrix quantum groups can be seen as a generalization of classical matrix groups $G \subseteq \mathrm{GL}_n(\mathbb{C})$ and are an active research area. However, in this thesis we focus only on the C^*-algebra structure of such quantum groups.

The C^*-algebra $C(S_n^+)$ can be defined as the universal unital C^*-algebra which is generated by a magic unitary:

$$C(S_n^+) := C^*(u_{ij},\, 1 \leq i,j \leq n \mid u \text{ is a magic unitary}).$$

Here a magic unitary is a matrix over a unital C^*-algebra where the entries are projections and the rows and columns sum up to one. This means the entries u_{ij} satisfy the relations

$$u_{ij}^2 = u_{ij}^* = u_{ij}, \qquad \sum_{k=1}^{n} u_{ik} = 1, \qquad \sum_{k=1}^{n} u_{kj} = 1 \qquad (1 \leq i,j \leq n).$$

The quantum permutation groups S_n^+ were used by Wang to describe so called quantum automorphism groups of finite spaces in the context of non-commutative geometry. However, they have further applications and can for example be used to define quantum automorphism groups of graphs. Since classical automorphism groups of finite sets and graphs are given by permutations, their quantum version can be defined with the help of S_n^+. See for example [Con94], [Wan98], [Bic99] and [Ban05].

In the article [JW20] Jung-Weber constructed sequences of models (B_k, M_k) and corresponding $*$-homomorphisms $\varphi_k \colon C(S_n^+) \to B_k$ given an initial pair (B_1, M_1). By constructing additional $*$-homomorphisms $\pi_{k+1,k} \colon B_{k+1} \to B_k$ they obtained inverse systems with inverse limits $G := (B_\infty, M_\infty)$.

The models (B_k, M_k) are not required to have a comultiplication Δ and hence are not necessarily compact matrix quantum groups. However, it turns out that the limit objects (B_∞, M_∞) are compact matrix quantum groups which are isomorphic to S_n^+. It remained open if the sequences of models (B_k, M_k) are strictly increasing or if the limit objects are obtained in a finite number of steps. In particular, the following question [JW20, Q. 4.9] was asked for such a sequence.

Question 3.3. Are there polynomials p_k in the generators $u_{ij} \in C(S_n^+)$ such that $\varphi_k(p_k) = 0$ for $\varphi_k \colon C(S_n^+) \to B_k$, but $\varphi_{k+1}(p_k) \neq 0$?

The existence of such polynomials would imply that each φ_k is not injective in contrast to the limit case. There the corresponding mapping $\varphi_\infty \colon C(S_n^+) \to B_\infty$ is

injective. In the following we focus on the case $n = 4$. In this case Jung-Weber considered the matrix

$$R := \begin{pmatrix} p & 0 & 1-p & 0 \\ 1-p & 0 & p & 0 \\ 0 & q & 0 & 1-q \\ 0 & 1-q & 0 & q \end{pmatrix} \in M_4(A_{pq}).$$

Here A_{pq} is the universal unital C^*-algebra which is generated by two projections p and q:

$$A_{pq} := C^*(1, p, q \mid 1,\ p,\ q \text{ are projections}, 1p = p1 = p, 1q = q1 = 1)$$

With the so called $\textcircled{+}$-product it is possible to define a sequence of magic unitaries $M_k := R^{\textcircled{+}k} \in M_4(A_{pq}^{\otimes k})$. Let $B_k \subseteq A_{pq}^{\otimes k}$ be generated by the entries of M_k. Then the sequence (B_k, M_k) allows the construction of an inverse limit as described above.

To answer Question 3.3 for this concrete sequence and $k \leq 2$, Jung-Weber gave examples of non-zero polynomials p_1 and p_2 in the generators of $C(S_4^+)$. These polynomials satisfy $p_1(M_1) = 0$, $p_1(M_2) \neq 0$ and $p_2(M_2) = 0$, $p_2(M_3) \neq 0$. However, the case $k = 3$ and the existence of a corresponding polynomial p_3 remained open.

In this thesis we consider the matrix $M_3 := R^{\textcircled{+}3}$ and the C^*-algebra $B_3 \subseteq A_{pq}^{\otimes 3}$ which is generated by the entries of M_3. The matrix $R^{\textcircled{+}3} \in M_4(A_{pq}^{\otimes 3})$ is given by

$$R^{\textcircled{+}3} = \begin{pmatrix} \begin{matrix} p\otimes p\otimes p \\ +(1-p)\otimes q\otimes(1-p)\end{matrix} & \begin{matrix} p\otimes(1-p)\otimes q \\ +(1-p)\otimes(1-q)\otimes(1-q)\end{matrix} & \begin{matrix} p\otimes p\otimes(1-p) \\ +(1-p)\otimes q\otimes p\end{matrix} & \begin{matrix} p\otimes(1-p)\otimes(1-q) \\ +(1-p)\otimes(1-q)\otimes q\end{matrix} \\ \begin{matrix} (1-p)\otimes p\otimes p \\ +p\otimes q\otimes(1-p)\end{matrix} & \begin{matrix} (1-p)\otimes(1-p)\otimes q \\ +p\otimes(1-q)\otimes(1-q)\end{matrix} & \begin{matrix} (1-p)\otimes p\otimes(1-p) \\ +p\otimes q\otimes p\end{matrix} & \begin{matrix} (1-p)\otimes(1-p)\otimes(1-q) \\ +p\otimes(1-q)\otimes q\end{matrix} \\ \begin{matrix} q\otimes(1-p)\otimes p \\ +(1-q)\otimes(1-q)\otimes(1-p)\end{matrix} & \begin{matrix} q\otimes p\otimes q \\ +(1-q)\otimes q\otimes(1-q)\end{matrix} & \begin{matrix} q\otimes(1-p)\otimes(1-p) \\ +(1-q)\otimes(1-q)\otimes p\end{matrix} & \begin{matrix} q\otimes p\otimes(1-q) \\ +(1-q)\otimes q\otimes q\end{matrix} \\ \begin{matrix} (1-q)\otimes(1-p)\otimes p \\ +q\otimes(1-q)\otimes(1-p)\end{matrix} & \begin{matrix} (1-q)\otimes p\otimes q \\ +q\otimes q\otimes(1-q)\end{matrix} & \begin{matrix} (1-q)\otimes(1-p)\otimes(1-p) \\ +q\otimes(1-q)\otimes p\end{matrix} & \begin{matrix} (1-q)\otimes p\otimes(1-q) \\ +q\otimes q\otimes q\end{matrix} \end{pmatrix}.$$

We then obtain the following result.

Theorem 4.8. *Let $p \in \mathbb{C}\langle X_4\rangle$ with $\deg p \leq 50$. If $p(M_3) = 0$ then $p \in I_4$.*

Here X_4 are the entries of a general 4×4 matrix and $I_4 \subseteq \mathbb{C}\langle X_4\rangle$ is the ideal which is generated by relations of a general 4×4 magic unitary. These relations include the ones defining a magic unitary and pairwise products of the entries. In other words, Theorem 4.8 shows that polynomials $p \neq 0$ with $\deg p \leq 50$ vanish in M_3 exactly because M_3 is a magic unitary. Hence, the entries of M_3 satisfy no further polynomial relations up to degree 50. This result can then be reformulated as follows.

Corollary 4.10. *Let $\varphi_3 \colon C(S_4^+) \to B_3$ the $*$-homomorphism which maps u_{ij} to $(M_3)_{ij}$ and let $p \in \mathbb{C}\langle X_4\rangle$ with $\deg p \leq 50$. If $p(u) \neq 0$ then $\varphi_3(p(u)) \neq 0$.*

This implies that no non-zero vanishing polynomial p up to degree 50 exists, which satisfies $p(M_3) = 0$ and $p(M_4) \neq 0$. As mentioned before, Jung-Weber constructed polynomials p_1 with $\deg p_1 = 1$ and p_2 with $\deg p_2 = 2$ which satisfy

$p_1(M_1) = 0$, $p_1(M_2) \neq 0$ and $p_2(M_2) = 0$, $p_2(M_3) \neq 0$. If a similar polynomial p_3 would exist for M_3 it would have to satisfy $\deg p_3 > 50$. However, such a large jump in the degree seems unreasonable. This indicates that no vanishing polynomial p_3 exists for M_3. This is a strong hint, but no proof, that φ_3 is already injective. Hence, the limit object (B_∞, M_∞) is already reached at $k = 3$. Then $C(S_4^+)$ is $*$-isomorphic to B_3 and (B_3, M_3) would be a somewhat simple concrete model of S_4^+.

Other models of $C(S_4^+)$ can for example be found in [BB07] where Banica and Bichon showed that $C(S_4^+) \simeq C(SO_{-1}(3))$.

The proof of Theorem 4.8 is done with the help of a computer. The key step is to construct a matrix Ψ_m such that $\ker \Psi_m \setminus \{0\}$ contains all polynomials p with $\deg p \leq m$ and $p(M_3) = 0$ but $p \notin I_4$. It is then verified that $\ker \Psi_m = \{0\}$ for $m \leq 50$ which proves Theorem 4.8. A difficulty is that the dimension of non-commutative polynomials in a 4×4-matrix up to degree m grows exponentially in m. Hence, to reduce the space of possible polynomials and make the computations feasible, we make use of Gröbner bases and finite automata. We refer to Remark 4.9 for more details on the proof of Theorem 4.8.

In addition, we use finite automata to obtain the following combinatorial result.

Theorem 4.17. *Let $V_m := \{p \in P_m \mid p(M_3) = 0\} \subseteq P_m$ be the vector space of polynomials up to degree m which vanish in M_3. Then $\operatorname{codim} V_m \leq \binom{2m+3}{3}$ for all $m \in \mathbb{N}$. Furthermore, equality implies that Corollary 4.10 holds for an arbitrary degree.*

This can be seen as a step towards showing that Theorem 4.8 holds for an arbitrary degree.

Let us now give a short overview on how this thesis is structured. At first we give some basic definitions in Section 2. In particular, we define the quantum permutation group S_4^+ which is the main subject of this work. We also introduce Gröbner bases and finite automata which will be used later.

In Section 3 we recall some results from [JW20]. In particular, we give an overview how the inverse system and the corresponding limit were constructed. Furthermore, we focus on the case $n = 4$ were the concrete magic unitary R is defined. In addition we recall the open questions which our work partially answers.

In Section 4 we first introduce some definitions and notations before we formulate and discuss our main results. The proof is then shifted to Section 5 and Section 6. In Section 5 we present some mathematical results in order to prove the main theorems. In particular, we construct bases of vector spaces and show how finite automata can be used to compute the dimension of them. Section 6 then contains the computational part. We give a construction of a finite automaton and present our algorithm for building and solving a resulting linear system. In addition, we analyse the complexity of these algorithms at the end of this section.

Finally, we present our computational results in Section 7 and complete the proofs of our main theorems.

2 Preliminaries

Throughout the rest of this thesis let $n, m \in \mathbb{N}$ if not stated otherwise.

2.1 C^*-algebras

We first recall some definitions and facts about C^*-algebras which are then used in Section 2.2 to define compact matrix quantum groups and in particular S_n^+. More details about C^*-algebras can for example be found in [Mur90] and [Bla06].

Definition 2.1 (C^*-algebras)**.** A *C^*-algebra* A is a $\mathbb{C}$-algebra equiped with a norm $\|\cdot\|$ and a mapping $^*\colon A \to A$ which statisfies the following properties.

1. The algebra is complete with respect to the norm and it holds $\|xy\| \leq \|x\| \cdot \|y\|$ for all $x, y \in A$.
2. The mapping * satisfies $(\lambda x)^* = \overline{\lambda} x^*$, $(x+y)^* = x^* + y^*$, $(x^*)^* = x$ and $(xy)^* = y^* x^*$ for all $\lambda \in \mathbb{C}$, $x, y \in A$.
3. The C^*-identity $\|x\|^2 = \|x^* x\|$ holds for all $x \in A$.

A C^*-algebra is called *unital* if it contains a unit which we denote by 1. An algebra homomorphism $\varphi\colon A \to B$ between two C^*-algebras A and B is called *$*$-homomorphism* if it holds $\varphi(x^*) = \varphi(x)^*$ for all $x \in A$.

An example of a C^*-algebra is the algebra $C(X)$ of continuous functions on a compact Hausdorff space X with $\|\cdot\|_\infty$ and * given as pointwise complex conjugation. Another example of a C^*-algebra is the algebra $B(\mathcal{H})$ of bounded linear operators on a Hilbert space $\mathcal{H}$, where * denotes the adjoint operator. A concrete example of the latter is the algebra of all complex $n \times n$-matrices $M_n(\mathbb{C})$.

The Gelfand-Naimark theorem shows that every C^*-algebra is isometrically $*$-isomorphic to a subalgebra of $B(\mathcal{H})$ for some Hilbert space $\mathcal{H}$. However, it is also possible to construct C^*-algebras in an abstract way. See for example Chapter II.8 in [Bla06] for more information on the construction of C^*-algebras and on universal C^*-algebras.

Definition 2.2 (Universal C^*-algebra)**.** Let $X = \{x_1, \ldots, x_n\}$ be a set of generators and put $\widehat{X} = \{x_1, \ldots, x_n, x_1^*, \ldots, x_n^*\}$. Let R be a set of polynomials in $\widehat{X}$ and define A to be the quotient of the non-commutative polynomials in $\widehat{X}$ by the two-sided ideal generated by R. Consider the seminorm

$$\|x\| := \sup\{p(x) \mid p \text{ is a } C^*\text{-seminorm on A}\} \qquad \forall x \in A.$$

Here a C^*-seminorm is a seminorm which satisfies $\|xy\| \leq \|x\| \cdot \|y\|$ and the C^*-identity $\|x\|^2 = \|x^* x\|$ for all $x, y \in A$. If $\|x\| < \infty$ for all $x \in A$ we let $N = \{x \in A \mid \|x\| = 0\}$. Then we define

$$C^*(X|R) := \overline{A/N}^{\|\cdot\|}$$

as the *universal C^*-algebra* with generators X and relations R. We write universal *unital* C^*-algebra if $C^*(X|R)$ contains in addition a unit 1. Note that in general we can get $\|x\| = \infty$ for some $x \in A$. However, if $\|x\| < \infty$ for all generators $x \in X$ it follows that $\|x\| < \infty$ for all $x \in A$. In this case we say that the universal C^*-algebra exists.

Example 2.3. Consider the universal C^*-algebra

$$A_{pq} := C^*(1, p, q \mid p = p^* = p^2,\ q = q^* = q^2,\ 1p = p1 = p,\ 1q = q1 = q).$$

Then A_{pq} is a unital C^*-algebra which is generated by two projections p and q. Since p and q are projections, it follows $\|p\|, \|q\| \leq 1$. Hence, every element has a finite norm and the universal C^*-algebra A_{pq} exists as in Definition 2.2.

An important tool for working with universal C^*-algebras is their following universal property.

Proposition 2.4. *Let A be a C^*-algebra, $X = \{x_1, \ldots, x_n\}$ be a set of generators and R be corresponding relations. If A has elements $Y = \{y_1, \ldots, y_n\}$ which satisfy the relations R, then there exists a $*$-homomorphism $\varphi\colon C(X|R) \to A$ with $\varphi(x_i) = y_i$ for all $i = 1, \ldots, n$.*

This universal property can for example be used to show the following result about the algebra A_{pq} from Example 2.3.

Lemma 2.5. *Let A_{pq} be the universal unital C^*-algebra which is generated by two projections p and q. Then A_{pq} is non-commutative.*

Proof. Consider the matrices

$$P := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad Q := \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}. \quad \text{Then} \quad PQ = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \end{pmatrix}, \quad QP = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \end{pmatrix},$$

$P^2 = P^* = P$, $Q^2 = Q^* = Q$ and $PQ \neq QP$. By the universal property in Proposition 2.4 there exists a $*$-homomorphism $A_{pq} \to M_2(2)$ with $p \mapsto P$ and $q \mapsto Q$. But this implies $pq \neq qp$. □

It is also possible to define a tensor product for C^*-algebras.

Definition 2.6 (Minimal tensor product)**.** Let A and B be two C^*-algebras and consider their vector space tensor product $A \otimes B$. It is possible to turn $A \otimes B$ into a $*$-algebra by defining

$$(a \otimes b) \cdot (a' \otimes b') := (aa') \otimes (bb'),$$
$$(a \otimes b)^* := a^* \otimes b^*$$

for all $a, a' \in A$, $b, b' \in B$. If one equipes this algebra $A \otimes B$ with a suitable C^*-norm and considers the completion with respect to this norm, one obtains a C^*-algebra. In general, there are several choices for a C^*-norm on $A \otimes B$ which have different completions.

In the following we will use $\otimes$ as the minimal tensor product for C^*-algebras, which can be constructed this way. More details about this construction and tensor products of C^*-algebra can be found in Chapter 3 of [BO08].

In addition to the previous constructions we will use basic results about positive elements and about the spectrum of elements throughout this thesis. These facts can for example be found in Chapter 1 and Chapter 2 of [Mur90].

2.2 Compact matrix quantum groups

After giving some facts about C^*-algebras we now continue with compact matrix quantum groups. These were first introduced by Woronowicz in [Wor87] and can be seen as a generalization of classical groups. Since our main results are in the context of quantum groups, we give the definition of compact matrix quantum groups and S_n^+ in this section. However, the main part of this thesis will then focus mostly on the algebraic properties of $C(S_n^+)$.

Definition 2.7 (Compact matrix quantum group)**.** Let A be a unital C^*-algebra and $u \in M_n(A)$ a matrix such that A is generated by the entries u_{ij}, $1 \leq i, j \leq n$. The pair $G := (A, u)$ is called *compact matrix quantum group* of size n if

1. u and $\bar{u} = (u_{ji}^*)_{i,j=1}^n$ are invertible.
2. the mapping $\Delta\colon A \to A \otimes A$ with

$$\Delta(u_{ij}) = \sum_{k=1}^{n} u_{ik} \otimes u_{kj}$$

is a $*$-homomorphism.

The C^*-algebra A is then denoted by $C(G)$.

In [Wan98] Wang introduced quantum permutation groups S_n^+ which are an example of compact matrix quantum groups. They can be seen as a generalization of the classical symmetric group S_n in the sense of compact matrix quantum groups. We first introduce magic unitaries which will be used to define S_n^+.

The name magic unitary was first used by Banica in [BN06] and is similiar to magic squares. In such a square the entries in each row and each column sum up to the same number.

Definition 2.8 (Magic unitary)**.** Let A be a unital C^*-algebra and $u \in M_n(A)$. The matrix u is called magic unitary if its entries are projections and each row and column sums up to 1. This means the entries satisfy

$$u_{ij}^2 = u_{ij}^* = u_{ij}, \qquad \sum_{k=1}^{n} u_{ik} = 1, \qquad \sum_{k=1}^{n} u_{kj} = 1 \qquad (1 \leq i, j \leq n).$$

Magic unitaries can be seen as generalizations of classical permutation matrices where each row and column contains exactly one 1 and all other entries are zeros. Now we can define S_n^+ as the universal C^*-algebra which is generated by a magic unitary.

Definition 2.9 (Quantum permutation group)**.** Let $u = (u_{ij})_{i,j=1}^n$ be a matrix of n^2 generators. Define the universal unital C^*-algebra

$$A := C^*(u_{ij},\ 1 \leq i, j \leq n \mid u \text{ is a magic unitary}).$$

Then the compact matrix quantum group $S_n^+ := (A, u)$ is called the quantum permutation group. Since all u_{ij} are projections, we get $\|u_{ij}\| \leq 1$ and hence $C(S_n^+)$ exists as universal C^*-algebra as in Definition 2.2.

Remark 2.10. Consider magic unitaries in $M_n(\mathbb{C})$, then one obtains exactly permutation matrices. Hence, these magic unitaries correspond to the classical permutation groups S_n. Consequently, one can view a general magic unitary as a permutation matrix with operator-valued entries. This justifies the name quantum permutation group for S_n^+ as a quantum version of S_n.

The next lemma shows an important property of magic unitaries, in particular of S_n^+, which does not follow algebraically from the definition.

Lemma 2.11. *Let A be a C^*-algebra and $M \in M_n(A)$ a magic unitary. Then the product of two different elements in the same row or column equals zero. This means*

$$M_{ik} \cdot M_{jk} = 0, \qquad M_{ki} \cdot M_{kj} = 0 \qquad (1 \leq i, j, k \leq n,\ i \neq j).$$

Proof. We consider the case $M_{ik}M_{jk} = 0$ for $1 \leq i, j, k \leq n$ and $i \neq j$. The case $M_{ki}M_{kj} = 0$ follows then by the same argument with swapped indicies. Consider the element $M_{jk}M_{ik}M_{jk}$, then we have $M_{jk}M_{ik}M_{jk} \geq 0$ since it is of the form

$$M_{jk}M_{ik}M_{jk} = (M_{ik}M_{jk})^*(M_{ik}M_{jk}).$$

On the other hand, we obtain

$$M_{jk}M_{ik}M_{jk} = M_{jk}\left(1 - \sum_{l=1}^{n} M_{lk}\right)M_{jk} = -\sum_{l \neq i} M_{jk}M_{lk}M_{jk}.$$

Since each $M_{jk}M_{lk}M_{jk} \geq 0$ as before, we get $M_{jk}M_{ik}M_{jk} \leq 0$ as negative sum of positive elements. From this follows $M_{jk}M_{ik}M_{jk} = 0$. We conclude with the C^*-identity that

$$\|M_{ik}M_{jk}\|^2 = \|(M_{ik}M_{jk})^*(M_{ik}M_{jk})\| = \|M_{jk}M_{ik}M_{jk}\| = 0,$$

hence $M_{ik}M_{jk} = 0$. □

During this work we will focus on S_4^+. One can show that $C(S_4^+)$ is a non-commutative C^*-algebra whereas $C(S_n^+)$ is commutative for $n \leq 3$. For the non-commutativity consider again the universal unital C^*-algebra A_{pq} from Example 2.3, which is generated by two projections p, q. Define the matrix

$$M := \begin{pmatrix} p & 1-p & 0 & 0 \\ 1-p & p & 0 & 0 \\ 0 & 0 & q & 1-q \\ 0 & 0 & 1-q & q \end{pmatrix}.$$

Then M is a magic unitary and there exists a surjective $*$-homomorphism $C(S_4^+) \to A_{pq}$ by the universal property of $C(S_4^+)$. Since A_{pq} is non-commutative by Lemma 2.5, it follows that $C(S_4^+)$ is also non-commutative.

2.3 Gröbner bases

Gröbner bases were first introduced by Buchberger in [Buc65]. They are constructed given a monomial ordering and allow computations with ideals in multivariate polynomial rings, for example to solve the ideal membership problem. In this section we introduce basic notations and definitions which will then be used in Section 5. A more detailed introduction can be found in [Mor94].

Let X be a set of generators. We denote by $\mathbb{C}\langle X\rangle$ the free $\mathbb{C}$-algebra generated by X. It can be regarded as the algebra of non-commutative polynomials in the variables X. In the following we denote the set of all monomials in $\mathbb{C}\langle X\rangle$ (including 1) with $X^\star$.

Given a well-ordering $\leq$ on X, we can define an ordering on $X^\star$. First we compare the degree of two monomials and if they have the same degree we compare them lexicographicaly from the left to the right. This ordering on $X^\star$ is then also a well-ordering.

A non-zero polynomial $p \in \mathbb{C}\langle X\rangle$ has a unique representation as $p = \sum_{i=1}^{n} \alpha_i m_i$ with $a_1, \ldots, \alpha_n \in \mathbb{C} \setminus \{0\}$ and $m_1, \ldots, m_n \in X^\star$. Let i_0 be the index of the largest monomial in this representation. Then we define the leading term $\mathrm{LT}(p) = m_{i_0}$ and the leading coefficient $\mathrm{LC}(p) = \alpha_{i_0}$. In this case the degree $\deg p$ is given the total number of factors of $\mathrm{LT}(p)$. Note that $\mathrm{LT}(p) = 1$ and $\deg p = 0$ for all non-zero constant polynomials $p \in \mathbb{C}\langle X\rangle$.

Example 2.12. Consider $X = \{x, y, z\}$ with $x < y < z$. Then we obtain the following ordering on $X^\star$:

$$1 < x < y < z < xx < xy < xz < yx < yy < yz < zz < zy < zz < \ldots$$

Let for example $p = 2x^2 + 3xy - z \in \mathbb{C}\langle X\rangle$ with $\deg p = 2$. Then the leading term and leading coefficient are given by $\mathrm{LT}(p) = xy$ and $\mathrm{LC}(p) = 3$.

Next we present a lemma which follows directly from the previous definitions and will be useful later in Section 5.

Lemma 2.13. *Let $x, y \in X^\star$ such that $x \leq y$. Then $axb \leq ayb$ for all $a, b \in X^\star$. Furthermore, it holds that $\mathrm{LT}(apb) = a\,\mathrm{LT}(p)b$ for all $p \in \mathbb{C}\langle X\rangle$ and $a, b \in X^\star$.*

Proof. Let $a, b, x, y \in X^\star$ such that $x \leq y$. If $\deg x \leq \deg y$, then $\deg axb \leq \deg ayb$ and $axb \leq ayb$. Otherwise we get $axb \leq ayb$ by lexicographical comparison from the left to right. Now consider the polynomial

$$p = \sum_{i=1}^{n} \alpha_i x_i \in \mathbb{C}\langle X\rangle$$

and assume without loss of generality $x_1 \leq \ldots \leq x_n$. The previous inequality yields $ax_1b \leq \ldots \leq ax_nb$. Hence

$$\mathrm{LT}(apb) = ax_nb = a\,\mathrm{LT}(p)b.$$

□

In the following we will require an ideal $I \subseteq \mathbb{C}\langle X\rangle$ to be two-sided since $\mathbb{C}\langle X\rangle$ is non-commutative. With these definitions we can now introduce Gröbner bases.

Definition 2.14 (Gröbner basis)**.** Let $I \subseteq \mathbb{C}\langle X\rangle$ an ideal. A finite set $G \subseteq I$ is called Gröbner basis for I if:

1. The ideal I is generated by G.
2. For every $p \in I$ exists a $g \in G$ and $a, b \in X^\star$ such that $\mathrm{LT}(p) = a\,\mathrm{LT}(g)b$.

Remark 2.15. Note that a Gröbner basis depends on the monomial ordering defined on $\mathbb{C}\langle X\rangle$. It is possible that a set is a Gröbner basis with respect to a given monomial ordering but is not a Gröbner basis with respect to other orderings. This has also practical implications since changing the monomial odering can affect the difficulty of computing a Gröbner basis on a computer.

A main application of Gröbner bases is to determine wether a polynomial is contained in a given ideal and to solve the ideal membership problem. Next we give some background and show how Gröber bases can be used in this context. Howerver, in this thesis we will use Gröbner bases to describe a basis for the quotient $\mathbb{C}\langle X\rangle/I$. These results will then be proven diffently in Section 5.

Example 2.16 (Ideal membership problem)**.** Let $I \subseteq \mathbb{C}\langle X\rangle$ be an ideal and $p_0 \in \mathbb{C}\langle X\rangle$, then the *ideal membership problem* asks, if $p_0 \in I$. This problem can be solved as follows. Let G be a Gröbner basis for I and assume there are $a_0, b_0 \in X^\star$ and $g_0 \in G$ such that $\mathrm{LT}(p_0) = a_0\,\mathrm{LT}(g_0)b_0$. Then it is possible to perform a reduction step and eliminate $\mathrm{LT}(p_0)$ by defining

$$p_1 := p_0 - \frac{\mathrm{LC}(p_0)}{\mathrm{LC}(g_0)} \cdot a_0 g_0 b_0.$$

As long as those conditions are satisfied we can repeat this reduction step and obtain a sequence $p_0, p_1, \ldots$. Since $X^\star$ is well-ordered and $\mathrm{LT}(p_i)$ is decreasing, we can perform at most finitely many of such steps. Hence, we obtain finitely many polynomials $p_0, \ldots, p_n$. If $p_n = 0$ then we can write p_0 as

$$p_0 = \sum_{i=0}^{n-1} \frac{\mathrm{LC}(p_i)}{\mathrm{LC}(g_i)} \cdot a_i g_i b_i$$

such that $p_0 \in I$. Otherwise, $p_0 \notin I$ by the definition of Gröbner basis since no suitable $g_n \in G$ exists anymore.

2.4 Finite automata

Let Σ be a finite set of symbols which is called *alphabet*. A *word* over Σ is a finite sequences $w_1 \cdots w_n$ of symbols $w_1, \ldots, w_n \in \Sigma$.

Finite automata were first used in [MP43] and became a fundamental tool in theoretical computer science. They can be used to describe sets of words over an alphabet and allow the design of linear time algorithms for searching a given subword in words. A detailed introduction to languages and automata can be found in [HMU06].

We will use them in Section 5 in combination with a Gröbner basis to describe a vector space basis of a quotient algebra. In Section 6 we show how they can be constructed and used for efficient computations.

Before we define finite automata we introduce some more notation. In the following let Σ be an alphabet. With $\Sigma^\star$ we denote the Kleene closure of Σ which is the set of all words over Σ including the empty word ε. A set of words $L \subseteq \Sigma^\star$ is called *language*.

Remark 2.17. Consider the non-commutative polynomials $\mathbb{C}\langle X\rangle$. Then the set of variables X can be considered as alphabet. In the previous notation the set $X^\star$ of all monomials coincides with the Kleene closure $X^\star$, if we identify the empty word ε with the unit 1.

A finite automaton can now be defined as a special labelled graph.

Definition 2.18 (Finite automaton)**.** A *finite automaton* over an alphabet Σ is a directed and labelled graph $\Gamma = (V, E, \ell, s_0, F)$ where

1. V denotes the set of vertices and E the set of directed edges.
2. $\ell\colon E \to \Sigma$ assigns to each edge in E a symbol from Σ.
3. $s_0 \in V$ is the *initial state*.
4. $F \subseteq E$ is a set of *final states*.

Note that multi-edges are allowed and the set F can be empty. The vertices are also called *states* and the edges *transitions*.

Example 2.19. Figure 1 shows a simple finite automaton Γ over the alphabet $\Sigma = \{a, b, c\}$. It has the vertices $V = \{1, 2, 3, 4\}$, the initial state $s_0 = 1$ and the final states $F = \{3, 4\}$. The initial state is marked with a small arrow and the final states are circled twice. Multi-edges are grouped together and labels are separated by commas in this case.

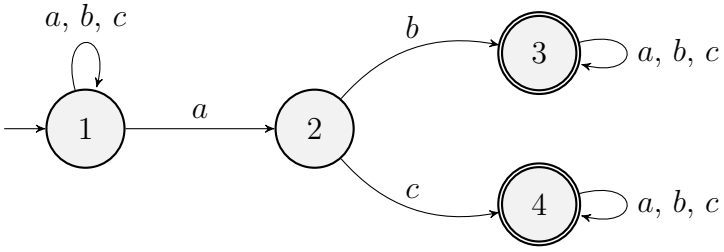


Figure 1: A simple finite automaton.

Let Γ be a finite automaton and $e_1, \ldots, e_n \in E$ be a directed path in Γ. Then we can assign the word $w = \ell(e_1) \cdots \ell(e_n) \in \Sigma^\star$ to this path. In this way each directed path in Γ corresponds to a word $w \in \Sigma^\star$. We say Γ that *accepts* a word $w \in \Sigma^\star$ if there exists a corresponding path in Γ which starts at s_0 and ends in F. Denote by $L(\Gamma) \subseteq \Sigma^\star$ the language which consists of all words which are accepted by Γ. Such a language, which can be described by a finite automaton, is called *regular language*.

Note that there might be multiple paths which result in the same accepted word. We call a finite automaton *non-deterministic* if there exists a node with two outgoing edges and same labels. Otherwise we call it *deterministic*. From this definition follows directly that in a deterministic finite automaton each accepted word has as a unique associated path.

Example 2.20. Consider again the finite automaton Γ from Figure 1. The accepting language $L(\Gamma)$ consists of all words $w \in \Sigma^\star$ which contain ab or ac as a subword. In addition, Γ is non-deterministic because node 1 has two outgoing edges with label a.

The next two propositions can be used to transform a given automaton and are useful when constructing new automata frome existing ones. See for example [HMU06] for proofs of these results. The second proposition is usually proven using the so called powerset construction, which can be found in Section 2.3.5 of [HMU06].

Proposition 2.21 (Complement)**.** *Let Γ be a deterministic finite automaton over the alphabet Σ. Then there exists a deterministic finite automaton $\overline{\Gamma}$ with accepting language*

$$L(\overline{\Gamma}) = \Sigma^\star \setminus L(\Gamma).$$

The automaton $\overline{\Gamma}$ is called the complement of Γ.

Proposition 2.22 (Powerset construction)**.** *Let Γ be a (possibly non-deterministic) finite automaton. Then there exists a deterministic finite automaton* $\det(\Gamma)$ *with*

$$L(\det(\Gamma)) = L(\Gamma).$$

This last proposition might be surprising since it shows that non-deterministic finite automata are not more powerfull than deterministic ones. It is especially useful since it might be easier to find a non-deterministic finite automaton for a given regular language. This can be seen in Section 6.1, where we first construct a non-deterministic finite automaton which then gets transformed into a deterministic one.

3 Models by Jung-Weber

In this section we recall some definitions from [JW20], which will be used in the following sections. In addition, we summarize the main steps in the construction of the inverse system and the existence of the inverse limit as compact matrix quantum group. At the end, we consider a concrete matrix R in the case $n = 4$ and state the open question which we attempt to answer in this thesis.

3.1 The ⏀-operator

We begin with the definition of the ⏀-operator similar to the one [Wor87].

Definition 3.1. Let A and B be two C^*-algebras. Then $\obar: M_n(A) \times M_n(B) \to M_n(A \otimes B)$ is defined by

$$(M \obar N)_{ij} = \sum_{k=1}^{n} M_{ik} \otimes N_{kj} \qquad (1 \leq i, j \leq n)$$

for all $M \in M_n(A)$, $N \in M_n(B)$.

In this context an important property is that ⏀ preserves magic unitaries.

Lemma 3.2. *Let A and B be two C^*-algebras, $M \in M_n(A)$ and $N \in M_n(B)$ two magic unitaries. Then $M \owedge N \in M_n(A \otimes B)$ is a magic unitary.*

Proof. Recall the tensor product $A \otimes B$ from Definition 2.6. Using the linearity of $\otimes$ and * it follows that

$$(M \owedge N)^*_{ij} = \left(\sum_{k=1}^n M_{ik} \otimes N_{kj}\right)^* = \sum_{k=1}^n M^*_{ik} \otimes N^*_{kj} = \sum_{k=1}^n M_{ik} \otimes N_{kj} = (M \owedge N)_{ij}.$$

Hence, all entries are self-adjoint. Using Lemma 2.11 we obtain

$$M_{ik}M_{il} = \begin{cases} M^2_{ik} = M_{ik}, & l = k \\ 0, & l \neq k \end{cases}, \qquad N_{kj}N_{lj} = \begin{cases} N^2_{kj} = N_{kj}, & l = k \\ 0, & l \neq k \end{cases}$$

for $1 \leq i, j, l, k \leq n$. This implies

$$\begin{aligned}(M \owedge N)^2_{ij} &= \left(\sum_{k=1}^n M_{ik} \otimes N_{kj}\right) \cdot \left(\sum_{l=1}^n M_{il} \otimes N_{lj}\right) \\ &= \sum_{k=1}^n \sum_{l=1}^n (M_{ik}M_{il}) \otimes (N_{kj}N_{lj}) = \sum_{k=1}^n M_{ik} \otimes N_{kj} = (M \owedge N)_{ij}\end{aligned}$$

such that all entries of $M \owedge N$ are projections. Now consider the sum of the entries in a column. Then

$$\begin{aligned}\sum_{i=1}^n (M \owedge N)_{ij} &= \sum_{i=1}^n \sum_{k=1}^n M_{ik} \otimes N_{kj} = \sum_{k=1}^n \left(\sum_{i=1}^n M_{ik}\right) \otimes N_{kj} \\ &= \sum_{k=1}^n 1_M \otimes N_{kj} = 1_M \otimes \left(\sum_{k=1}^n N_{kj}\right) = 1_M \otimes 1_N.\end{aligned}$$

Similar we obtain for a row

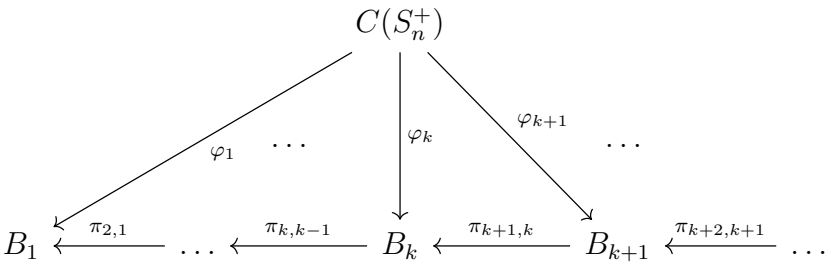
$$\sum_{j=1}^n (M \owedge N)_{ij} = 1_M \otimes 1_N.$$

Hence, $M \owedge N$ is a magic unitary. □

3.2 Inverse system

Consider an initial pair $(B_1,\ M_1)$ of a C^*-algebra B_1 and a magic unitary $M_1 \in M_n(B_1)$ such that B_1 is generated by the entries of M_1. Now one can define the matrices $M_k := (M_1)^{\owedge k}$ and $B_k \subseteq B_1^{\otimes k}$ as the C^*-algebras which are generated by the entries of M_k respectively. Since each M_k is a magic unitary, there exists a $*$-homomorphism $\varphi_k \colon C(S_n^+) \to B_k$ by the universal property of $C(S_n^+)$.

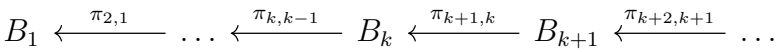
In Definition 3.5 of the article [JW20] Jung-Weber constructed general initial pairs (B_1, M_1). These pairs allow $*$-homomorphisms $\pi_{k+1,k} \colon B_{k+1} \to B_k$, mapping generators to generators, such that the following diagram commutes:



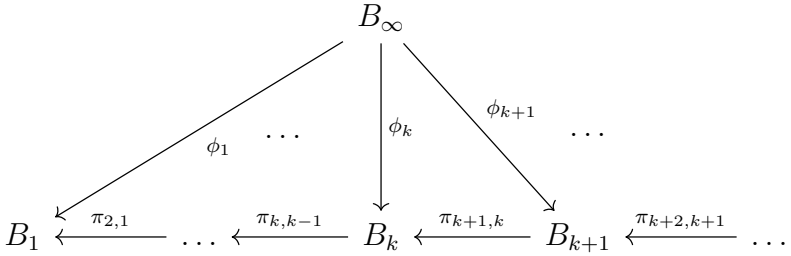
Note that the initial pairs (B_1, M_1) are only required for constructing the mappings $\pi_{k+1,k}$. See Definition 3.5 in [JW20] for more details on the general construction of such initial pairs.

Next we give a short sketch of how the mappings $\pi_{k+1,k}$ are constructed. The main idea is to use the corresponding initial pair (B_1, M_1) to first construct a $*$-homomorphism $\phi\colon B_1 \to C(S_n)$ mapping generators to generators. Here $C(S_n)$ is the universal unital C^*-algebra obtained by adding commutativity relations to the generators of $C(S_n^+)$. Since the identity matrix $\mathrm{id} \in M_n(\mathbb{C})$ is a magic unitary with commuting entries, we can use the universal property to obtain a $*$-homomorphism $\nu'\colon C(S_n) \to \mathbb{C}$ mapping generators of $C(S_n)$ to entries of id. Define $\nu\colon B_1 \to \mathbb{C}$ by $\nu' \circ \phi$ then the restriction of $(\mathrm{id}_{B_1})^{\otimes n} \otimes \nu$ to B_k gives a $*$-homomorphism $\pi_{k+1,k}\colon B_{k+1} \to B_k$. For more details on how ϕ is constructed from an initial pair (B_1, M_1) see Lemma 3.6 in [JW20].

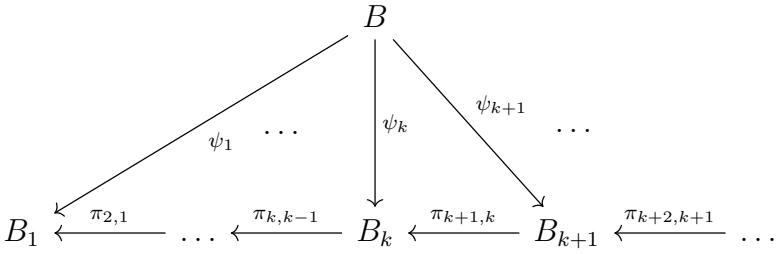
Consider again the previous sequence of pairs (B_k, M_k) and the diagram



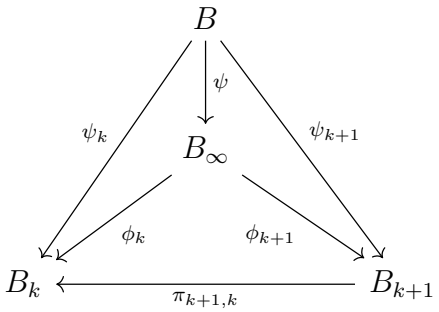
We call such a sequence an *inverse system*. More information on general inverse systems and inverse limits in the context of category theory can for example be found in [Mac71]. For inverse limits of C^*-algebras we refer to [Phi88]. In our case, a limit of this inverse system is the minimal pair (B_∞, M_∞) such that the following diagram commutes:



Here $\phi_k\colon B_\infty \to B_k$ are again $*$-homomorphisms mapping generators to generators. Minimality means that for every other pair (B, M) which allows a similar diagram



there exists a $\psi\colon B \to B_\infty$ such that each ϕ_k factors through ψ. In other words, the diagram



commutes for every $k \in \mathbb{N}$.

3.3 Limit object

In Lemma 4.1 in [JW20], Jung-Weber proved the existence of the limit object (B_∞, M_∞) if

$$\sup_{k\in\mathbb{N}} \|(M_k)_{ij}\| < \infty$$

for all entries $1 \leq i, j \leq n$. Hence, (B_∞, M_∞) exists in our case since the entries of magic unitaries are projections with $\|(M_k)_{ij}\| \leq 1$. Furthermore, M_∞ is a magic unitary because there exists an arrow $\psi\colon C(S_n^+) \to B_\infty$ by the minimality of (B_∞, M_∞).

To prove that (B_∞, M_∞) is a compact matrix quantum group the only thing remaining to show is the existence of a $*$-homomorphism $\Delta\colon B_\infty \to B_\infty \otimes B_\infty$ with

$$\Delta(M_{ij}) = \sum_{k=1}^{n} M_{ik} \otimes M_{kj}.$$

In the following we summarize the main ideas of this proof. The full proof and more details can be found in Section 4.2 of [JW20].

The first step is to show that the norms on B_∞ and $B_\infty \otimes B_\infty$ are given by

$$\begin{aligned} \|\cdot\|_{B_\infty} &= \lim_{n\to\infty} \|\phi_n(\cdot)\|_{B_n}, \\ \|\cdot\|_{B_\infty\otimes B_\infty} &= \lim_{n\to\infty} \|(\phi_n \otimes \phi_n)(\cdot)\|_{B_n\otimes B_n}. \end{aligned}$$

Let p be a polynomial in the entries of a matrix, then we denote by $p(M)$ the element which is obtained by substituting the entries of M into p. The next step is to verify that

$$\phi_{2n}(p(M_\infty)) = (\phi_n \otimes \phi_n)(p(M_\infty \leftplus M_\infty))$$

holds for all polynomials p as an equation in $B_1^{\otimes 2n} = B_1^{\otimes n} \otimes B_1^{\otimes n}$. Then it follows

$$\begin{aligned} \|p(M_\infty)\|_{B_\infty} &= \lim_{n\to\infty} \|\phi_n(p(M_\infty))\|_{B_n} \\ &= \lim_{n\to\infty} \|\phi_{2n}(p(M_\infty))\|_{B_{2n}} \\ &= \lim_{n\to\infty} \|(\phi_n \otimes \phi_n)(p(M_\infty \leftplus M_\infty))\|_{B_n\otimes B_n} \\ &= \|p(M_\infty \leftplus M_\infty)\|_{B_\infty\otimes B_\infty} \end{aligned}$$

for all polynomials p. Now define Δ for all polynomials in M_∞ by

$$\Delta(M_{ij}) = \sum_{k=1}^{n} M_{ik} \otimes M_{kj}$$

and note that $\Delta(p(M)) = p(M_\infty \oplus M_\infty)$ by definition. The previous equality shows that Δ is an isometry on polynomials, which is in particular bounded. Since these polynomials are dense in B_∞, we can extend Δ to a $*$-homomorphism $\Delta\colon B_\infty \to B_\infty \otimes B_\infty$ by continuity.

Hence, (B_∞, M_∞) is indeed a compact matrix quantum group. Furthermore, it turns out that we have $(B_\infty, M_\infty) = S_n^+$. This was shown by Chirvasitu and Józiak and can be found in the appendix of [JW20].

However, it remained unclear if such inverse systems will become stationary at some point or if one obtains infinitely many different pairs (B_k, M_k). In particular, the following question [JW20, Q. 4.9] was asked for a sequence of pairs (B_k, M_k) constructed from an initial pair (B_1, M_1).

Question 3.3. Are there polynomials p_k in the generators $u_{ij} \in C(S_n^+)$ such that $\varphi_k(p_k) = 0$ for $\varphi_k\colon C(S_n^+) \to B_k$, but $\varphi_{k+1}(p_k) \neq 0$?

It was expected that such polynomials exist such that all (B_k, M_k) would be distinct. However, we consider a concrete sequence of pairs (B_k, M_k) in this thesis and show that for $k = 3$ no polynomial from Question 3.3 up to degree 50 exists.

3.4 A concrete example

In the following we consider the case $n = 4$ and a specific initial pair (B_1, M_1) which our main result is based on.

Example 3.4. Let

$$A_{pq} := C^*(1, p, q \mid 1,\ p,\ q \text{ are projections}, 1p = p1 = p, 1q = q1 = 1)$$

be the universal unital C^*-algebra with is generated by two projections p and q (see Example 2.3). Consider the matrix

$$R := \begin{pmatrix} p & 0 & 1-p & 0 \\ 1-p & 0 & p & 0 \\ 0 & q & 0 & 1-q \\ 0 & 1-q & 0 & q \end{pmatrix} \in M_4(A_{pq})$$

and set $M_1 := R$. Then one obtains a sequence of pairs (B_k, M_k) where M_k is given by $R^{\oplus k} \in M_n(A_{pq}^{\otimes k})$ and $B_k \subseteq A_{pq}^{\otimes k}$ is the C^*-algebra which is generated by the entries of M_k.

It is then possible to construct $*$-homomorphisms $\pi_{k+1,k}\colon B_{k+1} \to B_k$ such that one obtains an inverse system with inverse limit (B_∞, M_∞) as described in the previous sections. In this case the following concrete question [JW20, Q. 4.10] was asked.

Question 3.5. Are there polynomials $(p_k)_{k\in\mathbb{N}}$ such that $p_k(M_k) = 0$ and $p_k(M_{k+1}) \neq 0$?

Such polynomials would answer Question 3.3 for the case $n = 4$ and show that no φ_k is injective. Consider the following matrices $M_2 := R^{\textcircled{2}}$ and $M_3 := R^{\textcircled{3}}$ with

$$R^{\textcircled{2}} = \begin{pmatrix} p \otimes p & (1-p) \otimes q & p \otimes (1-p) & (1-p) \otimes (1-p) \\ (1-p) \otimes p & p \otimes q & (1-p) \otimes (1-p) & p \otimes (1-p) \\ q \otimes (1-p) & (1-q) \otimes (1-q) & q \otimes p & (1-q) \otimes q \\ (1-q) \otimes (1-p) & q \otimes (1-q) & (1-q) \otimes p & q \otimes q \end{pmatrix},$$

$$R^{\textcircled{3}} = \begin{pmatrix} p \otimes p \otimes p \\ +(1-p) \otimes q \otimes (1-p) & p \otimes (1-p) \otimes q \\ +(1-p) \otimes (1-q) \otimes (1-q) & p \otimes p \otimes (1-p) \\ +(1-p) \otimes q \otimes p & p \otimes (1-p) \otimes (1-q) \\ +(1-p) \otimes (1-q) \otimes q \\ \\ (1-p) \otimes p \otimes p \\ +p \otimes q \otimes (1-p) & (1-p) \otimes (1-p) \otimes q \\ +p \otimes (1-q) \otimes (1-q) & (1-p) \otimes p \otimes (1-p) \\ +p \otimes q \otimes p & (1-p) \otimes (1-p) \otimes (1-q) \\ +p \otimes (1-q) \otimes q \\ \\ q \otimes (1-p) \otimes p \\ +(1-q) \otimes (1-q) \otimes (1-p) & q \otimes p \otimes q \\ +(1-q) \otimes q \otimes (1-q) & q \otimes (1-p) \otimes (1-p) \\ +(1-q) \otimes (1-q) \otimes p & q \otimes p \otimes (1-q) \\ +(1-q) \otimes q \otimes q \\ \\ (1-q) \otimes (1-p) \otimes p \\ +q \otimes (1-q) \otimes (1-p) & (1-q) \otimes p \otimes q \\ +q \otimes q \otimes (1-q) & (1-q) \otimes (1-p) \otimes (1-p) \\ +q \otimes (1-q) \otimes p & (1-q) \otimes p \otimes (1-q) \\ +q \otimes q \otimes q \end{pmatrix}.$$

For these matrices Jung-Weber constructed the polynomials $p_1 = u_{12}$ and $p_2 = u_{12}u_{24}$. These polynomials satisfy

$$\begin{aligned} p_1(M_1) &= 0, & p_1(M_2) &\neq 0, \\ p_2(M_2) &= 0, & p_2(M_3) &\neq 0 \end{aligned}$$

and answer the case $k = 1, 2$ in Question 3.5.

Our main result in Section 4 then partially answers the existence of such polynomials in the case $k = 3$. It implies that for each non-zero polynomial p with $\deg p \leq 50$ holds that if $p(M_3) = 0$ then $p(M_4) = 0$. This gives evidence that non such vanishing polynomial from Question 3.5 exists. See Remark 4.11 for a more detailed discussion of our main result.

4 A concrete magic unitary

In this section we present our main result. We consider a concrete C^*-algebra B_3 and a magic unitary $M_3 := R^{\textcircled{3}} \in M_4(B_3)$ from the previous section. Let $\varphi_3 \colon C(S_n^+) \to B_3$ be the $*$-homomorphism which maps generators to generators and let p be a polynomial in the generators of $C(S_4^+)$ with $\deg p \leq 50$. We show that $p \neq 0$ implies $\varphi_3(p) \neq 0$.

This gives evidence that no non-zero polynomial p with $\varphi_3(p) \neq 0$ exists. As a consequence, the mapping φ_3 might be injective. In this case, (B_3, M_3) is a concrete model for the quantum permutation group S_4^+ in the sense that $C(S_4^+)$ is $*$-isomorphic to B_3. For a further discussion of this result we refer to Remark 4.11 and Remark 4.12.

The main idea is to show that each non-zero polynomial $p \in \mathbb{C}\langle X_4 \rangle$ up to degree 50, which vanishes in M_3, lies in the ideal $I_4 \subseteq \mathbb{C}\langle X_4 \rangle$ generated by magic unitary relations. Hence, it also vanishes in the generators of $C(S_4^+)$. Here, $\mathbb{C}\langle X_4 \rangle$ denotes

algebra of non-commutative polynomials in the entries $X = \{x_{11}, x_{12}, \dots, x_{44}\}$ of a general 4×4-matrix.

Furthermore, consider the subspace $V_m \subseteq P_m$ of polynomials up to degree m which vanish in M_3. We show that $\operatorname{codim} V_m \leq \binom{2m+3}{3}$ as a subspace of all polynomials up to degree m. Equality would imply that the previous result holds for an arbitrary degree.

4.1 The magic unitary ideal and separating polynomials

Before we come to our main theorem, we start with some definitions and introduce the setting in which we prove our main results. The idea is to go from a C^*-algebra to the algebra of non-commutative polynomials. It is then possible to add relations to this algebra and answer the questions about vanishing polynomials in this context.

Definition 4.1. Let $X_n = \{x_{11}, x_{12}, \dots, x_{nn}\}$. Then $\mathbb{C}\langle X_n \rangle$ is the algebra of non-commutative polynomials in the entries of a general $n \times n$-matrix $(x_{ij})_{i,j=1}^n$. Let A be a C^*-algebra and $M \in M_n(A)$. Denote by $\varphi_M \colon \mathbb{C}\langle X_n \rangle \to A$ the replacement homomorphism which maps x_{ij} to M_{ij}. Furthermore, define $p(M) := \varphi_M(p)$ for all $p \in \mathbb{C}\langle X_n \rangle$ which replaces each x_{ij} in the polynomial p with M_{ij}.

Remark 4.2. We are interested in polynomials in the entries of magic unitaries. Since the entries of a magic unitary are self-adjoint, it is sufficient to consider only the variables x_{ij} and ignore x_{ij}^* in the definition of X_n.

In the following let A be a C^*-algebra and $M \in M_n(A)$ a magic unitary. Recall that by the definition of a magic unitary each M_{ij} is a projection and the rows and columns sum up to one. Let $I \subseteq \mathbb{C}\langle X_n \rangle$ be the ideal which is generated by the corresponding polynomials

$$x_{ij}^2 - x_{ij}, \qquad \sum_{k=1}^n x_{ik} - 1, \qquad \sum_{k=1}^n x_{kj} - 1 \qquad (1 \leq i, j \leq n).$$

Then it follows directly that $p(M) = 0$ for all $p \in I$. However, Lemma 2.11 shows that a magic unitary satisfies further relations which do not follow algebraically from the previous ones. This leads to the following definition.

Definition 4.3 (Magic unitary ideal)**.** Let $I_n \subseteq \mathbb{C}\langle X_n \rangle$ be the ideal which is generated by the following polynomials

$$\begin{aligned} &x_{ij}^2 - x_{ij}, \qquad \sum_{k=1}^n x_{ik} - 1, \qquad \sum_{k=1}^n x_{kj} - 1 && (1 \leq i, j \leq n), \\ &x_{ij} \cdot x_{ik}, \qquad x_{ji} \cdot x_{ki} && (1 \leq i, j, k \leq n,\ j \neq k). \end{aligned}$$

We call I_n the *magic unitary ideal*.

Remark 4.4. For a magic unitary M follows from the definition that $p(M) = 0$ for all $p \in I_n$.

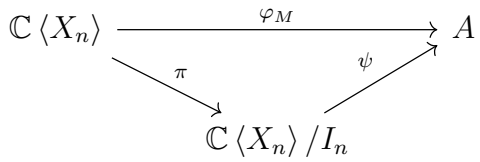
Now we can introduce separating polynomials which play an important role in this thesis.

Definition 4.5. Let A be a C^*-algebra and $M \in M_n(A)$ be a magic unitary. A non-zero polynomial $p \in \mathbb{C}\langle X_n \rangle$ is called *separating* if $\varphi_M(p) = 0$ but $p \notin I_n$.

To prove our main result, we are interested in the existence of separating polynomials for a given magic unitary. Hence, we consider again the replacement homomorphism φ_M. Since $I_n \subseteq \ker \varphi_M$, we can write the replacement homomorphism as

$$\varphi_M = \psi \circ \pi.$$

Here $\pi \colon \mathbb{C}\langle X_n \rangle \to \mathbb{C}\langle X_n \rangle / I_n$ is the canonical projection and $\psi \colon \mathbb{C}\langle X_n \rangle / I_n \to A$ is uniquely determined such that the following diagram commutes:



Consequently, all residue classes of polynomials, which vanish in M and do not lie in the magic unitary ideal, are contained in $\ker \psi \setminus \{0\}$. In other words, $\ker \psi \setminus \{0\}$ contains all separating polynomials of the matrix M.

Now consider polynomials $p \in \mathbb{C}\langle X_n \rangle$ with degree $\deg p$ bounded by $m \in \mathbb{N}$. Then we can define the following mapping.

Definition 4.6. Let A be a C^*-algebra and $M \in M_n(A)$ be a magic unitary such that the replacement homomorphism $\varphi_M \colon \mathbb{C}\langle X_n \rangle \to A$ can be factored as $\varphi_M = \psi \circ \pi$. Furthermore, define

$$P_m := \{p \in \mathbb{C}\langle X_n \rangle \mid \deg p \leq m\} \subseteq \mathbb{C}\langle X_n \rangle$$

as the vector space of all polynomials up to degree m. Then the mapping

$$\psi_m := \psi|_{\pi(P_m)}$$

is given by the restriction of ψ to the subspace $\pi(P_m) \subseteq \mathbb{C}\langle X_n \rangle / I_n$.

Assume there exists a separating polynomial p with $\deg p \leq m$. In other words, $p \in P_m$ with $\varphi_M(p) = 0$ but $p \notin I_n$. Then a corresponding residue class $[p] \neq 0$ can be found in $\ker \psi_m \setminus \{0\}$. Hence, all separating polynomials up to degree m are contained in $\ker \psi_m \setminus \{0\}$.

4.2 Main result

In the following we consider $n = 4$ and the quantum permutation group $S_4^+ = (C(S_4^+), u)$. Note that $u = (u_{ij}) \in M_4(C(S_4^+))$ is the magic unitary which contains the generators of $C(S_4^+)$.

Recall the $\textcircled{1}$-product from Definition 3.1 and the matrix R from Example 3.4 which is defined as

$$R := \begin{pmatrix} p & 0 & 1-p & 0 \\ 1-p & 0 & p & 0 \\ 0 & q & 0 & 1-q \\ 0 & 1-q & 0 & q \end{pmatrix} \in M_4(A_{pq}).$$

Here A_{pq} denotes the universal C^*-algebra which is generated by the two projections p and q (see Example 2.3).

Definition 4.7 (The concrete magic unitary)**.** Let $M_3 := R^{\textcircled{\scriptsize 1}3} \in M_4(A_{pq}^{\otimes 3})$ where

$$R^{\textcircled{\scriptsize 1}3} = \begin{pmatrix} \begin{matrix} p\otimes p\otimes p \\ +(1-p)\otimes q\otimes(1-p)\end{matrix} & \begin{matrix} p\otimes(1-p)\otimes q \\ +(1-p)\otimes(1-q)\otimes(1-q)\end{matrix} & \begin{matrix} p\otimes p\otimes(1-p) \\ +(1-p)\otimes q\otimes p\end{matrix} & \begin{matrix} p\otimes(1-p)\otimes(1-q) \\ +(1-p)\otimes(1-q)\otimes q\end{matrix} \\ \begin{matrix} (1-p)\otimes p\otimes p \\ +p\otimes q\otimes(1-p)\end{matrix} & \begin{matrix} (1-p)\otimes(1-p)\otimes q \\ +p\otimes(1-q)\otimes(1-q)\end{matrix} & \begin{matrix} (1-p)\otimes p\otimes(1-p) \\ +p\otimes q\otimes p\end{matrix} & \begin{matrix} (1-p)\otimes(1-p)\otimes(1-q) \\ +p\otimes(1-q)\otimes q\end{matrix} \\ \begin{matrix} q\otimes(1-p)\otimes p \\ +(1-q)\otimes(1-q)\otimes(1-p)\end{matrix} & \begin{matrix} q\otimes p\otimes q \\ +(1-q)\otimes q\otimes(1-q)\end{matrix} & \begin{matrix} q\otimes(1-p)\otimes(1-p) \\ +(1-q)\otimes(1-q)\otimes p\end{matrix} & \begin{matrix} q\otimes p\otimes(1-q) \\ +(1-q)\otimes q\otimes q\end{matrix} \\ \begin{matrix} (1-q)\otimes(1-p)\otimes p \\ +q\otimes(1-q)\otimes(1-p)\end{matrix} & \begin{matrix} (1-q)\otimes p\otimes q \\ +q\otimes q\otimes(1-q)\end{matrix} & \begin{matrix} (1-q)\otimes(1-p)\otimes(1-p) \\ +q\otimes(1-q)\otimes p\end{matrix} & \begin{matrix} (1-q)\otimes p\otimes(1-q) \\ +q\otimes q\otimes q\end{matrix} \end{pmatrix}$$

and define

$$B_3 := C^*\left((M_3)_{ij} \mid 1 \leq i,j \leq 4\right) \subseteq A_{pq}^{\otimes 3}$$

as the C^*-algebra which is generated by the entries of M_3.

By Lemma 3.2 the $\textcircled{\scriptsize 1}$-product preserves magic unitaries such that M_3 is a magic unitary. The next lemma shows that each polynomial p up to degree 50, which vanishes in M_3, lies already in the magic unitary ideal I_4. In other words, p vanishes because M_3 is a magic unitary rather than due to some additional relations in B_3.

Theorem 4.8. *Let $p \in \mathbb{C}\langle X_4\rangle$ with $\deg p \leq 50$. If $p(M_3) = 0$ then $p \in I_4$.*

Remark 4.9. The previous theorem will be proven with the help of a computer in Section 7 after some preparations in Section 5 and Section 6. Here we give a short overview of the main steps.

In the notation of Definition 4.5 we have to show that no separating polynomial up to degree 50 exists for M_3. This can be done by showing that $\ker \psi_{50} = \{0\}$ for the mapping ψ_{50} from Definition 4.6. Since ψ_{50} is a linear transformation, we can instead consider the transformation matrix Ψ_{50} of ψ_{50}. After constructing the matrix Ψ_{50} we can verify that $\ker \Psi_{50} = \{0\}$ with the help of a computer. In order to construct the matrix Ψ_{50}, we have to choose a basis for the domain $\pi(P_{50})$ and a basis for the image of ψ_{50}. In Section 5.1 we describe a basis $\mathcal{A}^{\otimes 3}$ for $\operatorname{Im}\psi_{50}$. The basis $\mathcal{B}_{50}$ for the domain $\pi(P_{50})$ is then constructed from a Gröbner basis for I_4 in Section 5.2.

In Section 6.2 we then present Algorithm 1 for constructing the transformation matrix Ψ_{50}. However, this algorithm requires the basis of $\pi(P_{50})$ to be given by a finite automaton. Therefore, the construction of such an automaton will be explained before in Section 6.1. After constructing the matrix Ψ_{50}, we use Algorithm 2 in Section 6.3 to verify that $\ker \Psi_{50} = \{0\}$. The final proof combining the previous steps is then presented in Section 7.

However, with Theorem 4.8 it is now possible to prove our main result.

Corollary 4.10. *Let $\varphi_3 \colon C(S_4^+) \to B_3$ the $*$-homomorphism which maps u_{ij} to $(M_3)_{ij}$ and let $p \in \mathbb{C}\langle X_4\rangle$ with $\deg p \leq 50$. If $p(u) \neq 0$ then $\varphi_3(p(u)) \neq 0$.*

Proof. We prove this statement by contraposition. Let $p \in \mathbb{C}\langle X_4\rangle$ with $\deg p \leq 50$ and $\varphi_3(p(u)) = 0$. Since $p(u)$ is a polynomial expression and φ_3 a homomorphism, we get $\varphi_3(p(u)) = p(M_3)$. Hence, we obtain $p(M_3) = \varphi_3(p(u)) = 0$ and $p \in I_4$ by Theorem 4.8. It follows from Remark 4.4 that $p(u) = 0$ because u is a magic unitary. □

Remark 4.11. Corollary 4.10 shows that no polynomial $p \in \mathbb{C}\langle X_4 \rangle$ with $\deg \leq 50$ and $p(u) \neq 0$ but $\varphi_3(p(u)) = 0$ exists. Based on this corollary, we believe that no such polynomial p with $\deg p > 50$ exists. Since Jung-Weber found polynomials with degree 1 and 2 for $k \leq 2$, it seems unlikely that the case $k = 3$ would require a polynomial with degree larger than 50. This would be a negative answer for Question 3.5 because it would imply the following. If $p(R^{\circledcirc 3}) = \varphi_3(p(u)) = 0$ then $p(u) = 0$ such that $p(R^{\circledcirc 4}) = \varphi_4(p(u)) = 0$.

Remark 4.12. To additionally prove that φ_3 is injective, one would have to show that no such polynomial p with $p(u) \neq 0$ but $\varphi_3(p(u)) = 0$ exists. Further, one has to show that φ_3 is isometric on polynomials. In this case φ_3 would be isometric on $C(S_4^+)$ and injective since polynomials are dense in $C(S_4^+)$. Note that being isometric is necessary since $C(S_4^+)$ is a C^*-algebra. If φ_3 is indeed injective then (B_3, M_3) is a concrete model for the quantum permutation group S_4^+ in the sense that $C(S_4^+)$ is $*$-isomorphic to B_3.

Remark 4.13. The maximum degree $m = 50$ could be increased if one provides more memory and more computation time. In Section 6.4 we analyse the complexity of the algorithm used to prove Theorem 4.8 and show that the memory and time required are asymptotically proportional to m^6. This is still polynomial but grows fast such that checking $m = 50$ required approximately 904 GB of memory. The actual running time and memory usage can be found in Appendix B. However, $m = 50$ gives already strong evidence that no separating polynomial (see Definition 4.5) for M_3 exists.

Remark 4.14. A $*$-algebra A is said to be *residually finite dimensional* if there exists an injective $*$-homomorphism

$$\pi \colon A \to \prod_{i \in I} M_{n_i}(\mathbb{C})$$

into a product of matrix algebras. In [BCF20], Brannan, Chirvasitu and Freslon showed that the $*$-algebra A corresponding to S_n^+ is residually finite dimensional. Hence, for each $*$-polynomial $p \neq 0$ in the generators of $C(S_n^+)$ there exists a $*$-homomorphism $\pi_i \colon A \to M_{n_i}(\mathbb{C})$ such that $\pi_i(p) \neq 0$. Here π_i is obtained by projecting onto the i-th component of π for some $i \in I$ depending on p. Compare this to our result from Corollary 4.10, where the algebra B_3 is not a matrix algebra but might satisfy $\varphi(p) \neq 0$ for all $*$-polynomials $p \neq 0$. Hence, being residually finite dimensional gives further evidence that quiet simple representations of quantum permutation groups S_n^+ exist.

4.3 Dimension result

Furthermore, we obtain the following combinatorial results about the dimension of the subspace $\pi(P_m)$.

Lemma 4.15. *Let $P_m \subseteq \mathbb{C}\langle X_4 \rangle$ be the vector space of polynomials up to degree m and $\pi \colon \mathbb{C}\langle X_4 \rangle \to \mathbb{C}\langle X_4 \rangle / I_4$ the canonical projection. Then* $\dim \pi(P_m) = \binom{2m+3}{3}$.

Remark 4.16. The previous lemma will be proven with the help of a computer in Section 7 after some preparation in Section 5 and Section 6. Here we give a short overview of the main steps.

In Section 5.2 we construct a basis $\mathcal{B}_m$ for $\pi(P_m)$ from a Gröbner basis for I_4. This basis $\mathcal{B}_m$ can alternatively be described by a finite automaton Γ. In Section 5.3 we show how $\dim \pi(P_m) = |\mathcal{B}_m|$ can be computed by counting paths in the finite automaton Γ. The construction of Γ is then presented in Section 6.1 and the proof is completed in Section 7.

The previous lemma allows us now to prove the following theorem, which can be seen as a step towards showing that Corollary 4.10 holds for an arbitrary degree. This would then imply that Theorem 4.8 holds for an arbitrary degree.

Theorem 4.17. *Let $V_m := \{p \in P_m \mid p(M_3) = 0\} \subseteq P_m$ be the vector space of polynomials up to degree m which vanish in M_3. Then $\operatorname{codim} V_m \leq \binom{2m+3}{3}$ for all $m \in \mathbb{N}$. Furthermore, equality implies that Corollary 4.10 holds for an arbitrary degree.*

Proof. Consider the canonical projection $\pi\colon \mathbb{C}\langle X_4\rangle \to \mathbb{C}\langle X_4\rangle / I_4$. Then we obtain

$$\dim P_m = \dim \ker \pi|_{P_m} + \dim \operatorname{Im} \pi|_{P_m}.$$

Since $\ker \pi|_{P_m} = I_4 \cap P_m$, we obtain further

$$\dim P_m = \dim(I_4 \cap P_m) + \dim \pi(P_m).$$

Because there exist only finitely many monomials up to degree m, it holds that $\dim P_m < \infty$. From Lemma 4.15 follows

$$\operatorname{codim}(I_4 \cap P_m) = \dim P_m - \dim(I_4 \cap P_m) = \dim \pi(P_m) = \binom{2m+3}{3}.$$

Since $I_4 \cap P_m \subseteq V_m$, we get

$$\operatorname{codim} V_m \leq \operatorname{codim}(I_4 \cap P_m) = \binom{2m+3}{3}.$$

Equality would imply that $I_4 \cap P_m = V_m$. In particular, it would follow that $p \in I_4$ for each $p \in \mathbb{C}\langle X_4\rangle$ with $p(M_3) = 0$. □

5 Polynomial and quotient bases

In this section we present some general results which will be used in Section 6 and Section 7 to prove our main theorems. In particular, we present a basis for polynomials in tensor products of 1, p and q as subspace of $A_{pq}^{\otimes n}$. This basis can then be used to represent $\operatorname{Im} \psi_m$ of the mapping ψ_m from Definition 4.6.

Furthermore, we show how Gröbner bases can be used to construct a basis for $\mathbb{C}\langle X\rangle / I$ and how finite automata can be used to compute the dimension of $\pi(P_m) \subseteq \mathbb{C}\langle X\rangle / I$ in the notation of Section 4.1. Then these results get applied to the magic unitary I_4 in Section 6 and Section 7 in order to prove our main results.

5.1 Polynomial basis

We begin with the universal unital C^*-algebra A_{pq} which is generated by the two projections p and q. The next lemma shows that alternating products of p's and q's are linearly independent.

Lemma 5.1. *The set $\mathcal{A} := \{1, p, q, pq, qp, pqp, \ldots\} \subseteq A_{pq}$ is linearly independent.*

Proof. Let $v \in \mathcal{A}$, then we denote by $|v|$ the number of factors of v. Assume $\mathcal{A}$ is not linearly independent, then there exist $v_0, \ldots, v_n \in \mathcal{A}$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{C}$ such that

$$v_0 = \sum_{i=1}^{n} \alpha_i v_i.$$

Without loss of generality we can assume $|v_0|$ is maximal of $|v_0|, \ldots, |v_n|$. Then we can substitute v_0 into all $w \in \mathcal{A}$ with $|w| > |v_0|$. Denote by $\langle \mathcal{A} \rangle$ the linear span of $\mathcal{A}$ then

$$\langle \mathcal{A} \rangle = \langle \{v \in \mathcal{A} \mid |v| \leq |v_0|\} \rangle$$

such that $A_0 := \langle \mathcal{A} \rangle \subseteq A_{pq}$ is a finite dimensional subspace. We show that the unit ball $B := \{v \in A_0 \mid \|v\| \leq 1\}$ is not sequencially-compact which is a contradiction.

Since $\|p\| = \|q\| = 1$ and the norm is submultiplicative, we get $\|v\| \leq 1$ for all $v \in \mathcal{A}$. Consider the following matrices

$$P := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad Q_\theta := \begin{pmatrix} \cos^2\theta & \cos\theta\sin\theta \\ \cos\theta\sin\theta & \sin^2\theta \end{pmatrix}.$$

These matrices satisfy $P^2 = P^* = P$, $Q_\theta^* = Q_\theta$ and

$$Q_\theta^2 = \begin{pmatrix} \cos^4\theta + \cos^2\theta\sin^2\theta & \cos^3\theta\sin\theta + \cos\theta\sin^3\theta \\ \cos^3\theta\sin\theta + \cos\theta\sin^3\theta & \cos^2\theta\sin^2\theta + \sin^4\theta \end{pmatrix} = Q_\theta$$

by using $\cos^2\theta + \sin^2\theta = 1$. According to the universal property of A_{pq} (Proposition 2.4) there exists a $*$-homomorphism $\varphi_\theta \colon A_{pq} \to M_2(\mathbb{C})$ with $p \mapsto P$ and $q \mapsto Q_\theta$. Since

$$Q_\theta P = \begin{pmatrix} \cos^2\theta & 0 \\ \cos\theta\sin\theta & 0 \end{pmatrix},$$

we obtain that $\cos^2\theta$ is an eigenvalue of $Q_\theta P$. Further, we obtain that $\cos^2\theta \in \sigma(qp)$ by the existence of φ_θ. By ranging θ between 0 and $\frac{\pi}{2}$ it follows that $[0,1] \subseteq \sigma(qp)$. The spectral mapping theorem then implies

$$\sigma((qp)^n) = \{\lambda^n \mid \lambda \in \sigma(qp)\} \supseteq \{\lambda^n \mid \lambda \in [0,1]\} = [0,1].$$

Consider the sequence $(p(qp)^n)_{n\in\mathbb{N}} \subseteq B$. It holds that $p(qp)^n$ is self-adjoint because p and q are self-adjoint. Since $\sigma(xy) \cup \{0\} = \sigma(yx) \cup \{0\}$ for all $x, y \in A_{pq}$, it follows that

$$\begin{aligned} \sigma\left(p(qp)^n - p(qp)^{2n}\right) \cup \{0\} &= \sigma\left(p\left[(qp)^n - (qp)^{2n}\right]\right) \cup \{0\} \\ &= \sigma\left(\left[(qp)^n - (qp)^{2n}\right]p\right) \cup \{0\} \\ &= \sigma\left((qp)^n - (qp)^{2n}\right) \cup \{0\}. \end{aligned}$$

This implies

$$\left\|p(qp)^n - p(qp)^{2n}\right\| = r\left(p(qp)^n - p(qp)^{2n}\right) = r\left((qp)^n - (qp)^{2n}\right)$$

because $p(qp)^n - p(qp)^{2n}$ is self-adjoint. With the spectral mapping theorem follows further

$$\left\|p(qp)^n - p(qp)^{2n}\right\| = \sup_{\lambda\in\sigma((qp)^n)} \left|\lambda - \lambda^2\right| \geq \sup_{\lambda\in[0,1]} \left|\lambda - \lambda^2\right| \geq \frac{1}{2} - \left(\frac{1}{2}\right)^2 = \frac{1}{4}.$$

Hence, $(p(qp)^n)_{n\in\mathbb{N}}$ cannot contain any Cauchy subsequence and consequently no convergent subsequence. This shows B is not sequencially-compact. □

The next two lemmas show that a similar result holds for tensor products of A_{pq} and that $\mathcal{A}$ can be used to construct a basis for polynomials in tensor products of 1, p and q.

Lemma 5.2. *The set $\mathcal{A}^{\otimes n} := \{v_1 \otimes \cdots \otimes v_n \mid v_1, \ldots, v_n \in \mathcal{A}\} \subseteq A_{pq}^{\otimes n}$ is linearly independent.*

Proof. Follows from Lemma 5.1 and since $\otimes$ is constructed from the vector space tensor product (Definition 2.6). □

Lemma 5.3. *Denote by $A_0 \subseteq A_{pq}^{\otimes n}$ the algebra of polynomials in $\mathcal{A}^{\otimes n}$. Then $A_0 = \langle \mathcal{A}^{\otimes n}\rangle$. In particular, $\mathcal{A}^{\otimes n}$ is a basis of A_0.*

Proof. The inclusion $\langle \mathcal{A}^{\otimes n}\rangle \subseteq A_0$ follows from the definition of A_0. On the other hand, $\mathcal{A}^{\otimes n}$ is closed under multiplication since repeated p's and q's cancel:

$$(\ldots p)\cdot(p\ldots) = \ldots p\ldots, \qquad (\ldots p)\cdot(q\ldots) = \ldots pq\ldots,$$
$$(\ldots q)\cdot(p\ldots) = \ldots qp\ldots, \qquad (\ldots q)\cdot(q\ldots) = \ldots q\ldots$$

Hence, $A_0 \subseteq \langle \mathcal{A}^{\otimes n}\rangle$. Furthermore, $\mathcal{A}^{\otimes n}$ is a basis because $A_0 = \langle \mathcal{A}^{\otimes n}\rangle$ and $\mathcal{A}^{\otimes n}$ is linearly independent by Lemma 5.2 □

This basis can then be applied to the replacement homomorphism φ_{M_3} of our concrete magic unitary M_3.

Lemma 5.4. *Let $\varphi_{M_3}\colon \mathbb{C}\langle X_4\rangle \to A_{pq}^{\otimes 3}$ be the replacement homomorphism of the concrete magic unitary M_3. Then $\operatorname{Im}\varphi_{M_3} \subseteq \langle \mathcal{A}^{\otimes 3}\rangle$.*

Proof. Since the entries of M_3 are polynomials in tensor products of 1, p and q, we obtain that $\operatorname{Im}\varphi_{M_3}$ consists also of polynomials in tensor products of 1, p and q. Hence, $\operatorname{Im}\varphi_{M_3} \subseteq \langle \mathcal{A}^{\otimes 3}\rangle$ by Lemma 5.3. □

5.2 Quotient basis

Consider an ideal $I \subseteq \mathbb{C}\langle X\rangle$ for some arbitrary finite set X and recall the concept of Gröbner bases from Section 2.3. Let $G \subseteq I$ be a Gröbner basis for I. In the following we describe how one can obtain a basis for the quotient $\mathbb{C}\langle X\rangle / I$ from G.

This result gets applied later to the magic unitary ideal I_4 in order to obtain a basis for $\mathbb{C}\langle X_4\rangle / I_4$. Such a basis allows us to construct the transformation matrix of ψ_m from Definition 4.6. This matrix plays an important role in proving our main result.

Now consider again the general case. The following lemma is not new but we give an own proof of it.

Lemma 5.5. *Let $I \subseteq \mathbb{C}\langle X\rangle$ be an ideal and G a Gröbner basis for I. Then the residue classes of*

$$\mathcal{B} := X^\star \setminus \{a\,\mathrm{LT}(g)b \mid g \in G, a, b \in X^\star\}$$

are a basis of the quotient $\mathbb{C}\langle X\rangle / I$. In particular, the residue classes of $\mathcal{B}$ are pairwise distinct.

Proof. We show separately that $\mathcal{B}$ is linearly independent and spanning. Denote by $[p] \in \mathbb{C}\langle X\rangle / I$ the residue class of $p \in \mathbb{C}\langle X\rangle$.

Independence: Assume $\{[x] \mid x \in \mathcal{B}\}$ is not linearly independent, then there exist $x_1, \ldots, x_n \in \mathcal{B}$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{C} \setminus \{0\}$ such that

$$\sum_{i=1}^{n} \alpha_i [x_i] = 0.$$

Without loss of generality assume $x_1 < \ldots < x_n$ and define $p := \sum_{i=1}^{n} \alpha_i x_i$. Then $\mathrm{LT}(p) = x_n$ and $[p] = 0$ since $[\cdot]$ is linear. But this means $p \in I$ and by definition of G there exist $g \in G$, $a, b \in X^\star$ such that

$$x_n = \mathrm{LT}(p) = a\,\mathrm{LT}(g)b.$$

This contradicts $x_n \in \mathcal{B}$ by definition. Hence, the residue classes are linear independent and in particular pairwise disjoint.

Spanning set: Let $S := \langle\{[x] \mid x \in \mathcal{B}\}\rangle$ be the spanning set. It is sufficient to show $[x] \in S$ for all monomials $x \in X^\star$. Assume there exists $x \in X^\star$ such that $[x] \notin S$. Since $X^\star$ is well-ordered, let x_0 be the minimal monomial with this property. From the definition of $\mathcal{B}$ and since $x_0 \notin \mathcal{B}$, it follows that x_0 has the form $x_0 = a\,\mathrm{LT}(g)b$ for some $g \in G$, $a, b \in X^\star$. Define $p := agb$ then $p \in I$ and by Lemma 2.13 we obtain

$$\mathrm{LT}(p) = \mathrm{LT}(agb) = a\,\mathrm{LT}(g)b = x_0.$$

Hence, p can be written as

$$p = \alpha_0 x_0 + \sum_{i=1}^{n} \alpha_i x_i$$

for some $\alpha_0, \ldots, \alpha_n \in \mathbb{C} \setminus \{0\}$ and $x_1, \ldots, x_n \in X^\star$. Since $\mathrm{LT}(p) = x_0$ we have $x_i \leq x_0$ for $1 \leq i \leq n$. Because x_0 is minimal with $[x_0] \notin S$, we obtain $[x_i] \in S$ for $1 \leq i \leq n$. Since $p \in I$, it follows that $[p] = 0$ and

$$[x_0] = -\frac{1}{\alpha_0} \sum_{i=1}^{n} \alpha_i [x_i] \in S.$$

This contradicts our assumption such that S contains the residue classes of all monomials and $S = \mathbb{C}\langle X\rangle / I$. □

Remark 5.6. The set $\mathcal{B}$ from the previous proposition can alternatively be described as

$$\mathcal{B} = \{x \in X^\star \mid x \text{ contains no leading term from } G\}.$$

Remark 5.7. Let $P_m \subseteq \mathbb{C}\langle X\rangle$ be the subspace of polynomials up to degree m and $\pi\colon \mathbb{C}\langle X\rangle \to \mathbb{C}\langle X\rangle / I$ the canonical projection. Since $x_i < x_0$ for $1 \leq i \leq n$ in the proof of the spanning property in Lemma 5.5, we obtain that the residue classes of

$$\mathcal{B}_m := \{x \in \mathcal{B} \mid \deg x \leq m\}$$

are a basis for $\pi(P_m)$.

Remark 5.8. Recall finite automata from Section 2.4. In the following we use X as an alphabet and we identify monomials with words over X as described in Remark 2.17. If we consider again the description of $\mathcal{B}$ in Remark 5.6, one can recognize that $\mathcal{B}$ is a regular language over the alphabet X. This means there exists a deterministic finite automaton Γ such that $\mathcal{B} = L(\Gamma)$. Hence, there is a bijection between elements $w \in \mathcal{B}$ and accepting paths in Γ. In particular, there exists a bijection between elements in $\mathcal{B}_m$ and accepting paths up to length m. We give an explicit construction of such a finite automaton in Lemma 6.7 in Section 6.1. The first one who used graphs to describe bases of quotient algebras was Ufnarovskiĭ in [Ufn91]. With this approach it is possible to efficiently enumerate all basis elements and count them in order to obtain a formula for the dimension.

5.3 Dimension

As in the previous section let $I \subseteq \mathbb{C}\langle X\rangle$ be an ideal and $G \subseteq I$ a Gröbner basis for I. Furthermore, let Γ be a deterministic finite automaton with accepting language $L(\Gamma) = \mathcal{B}$ as mentioned in Remark 5.8 and constructed in Lemma 6.7. Here

$$\mathcal{B} = \{x \in X^\star \mid x \text{ contains no leading term from } G\}$$

is the basis for $\mathbb{C}\langle X\rangle / I$ from Remark 5.6. In this section we show how to compute $|\mathcal{B}_m|$ where

$$\mathcal{B}_m = \{x \in \mathcal{B} \mid \deg x \leq m\}.$$

This corresponds to computing $\dim \pi(P_m)$ in the notation of Remark 5.7 from the previous section. In particular, this can be used to prove Lemma 4.15 by applying the results from this section to the magic unitary ideal I_4. We refer to Section 7 for the corresponding proof.

According to Remark 5.8 accepting paths in Γ correspond to words in $\mathcal{B}$. In particular, accepting paths up to length m correspond to words in $\mathcal{B}_m$. Hence, to compute $|B_m|$ we have to count all accepting paths in Γ up to length m. For this task, we first introduce adjacency matrices.

Definition 5.9 (Adjacency matrix)**.** Let Γ be a finite automaton with n states $s_1, \ldots, s_n$. Then $M \in M_n(\mathbb{R})$ with

$$M_{ij} = |\{e \in E \mid e \text{ goes from } s_i \text{ to } s_j\}| \qquad (1 \leq i, j \leq n)$$

is the *adjacency matrix* of Γ.

The next proposition shows that the adjacency matrix M can be used to count paths in Γ.

Proposition 5.10. *Let Γ be a finite automaton with states $s_1, \ldots, s_n$ and adjacency matrix M. Then the number of paths from s_i to s_j of length m is given by $(M^m)_{ij}$.*

Proof. Denote by $N_{i,j,m}$ the number of paths from state s_i to state s_j with length m. We want to show that $N_{i,j,m} = (M^m)_{ij}$. We prove this statement by induction on m. The case $m = 1$ follows directly from Definition 5.9 since paths of length 1 are given by edges. Now consider $m > 1$ and two states s_i and s_j. Paths from s_i to s_j of length m can be decomposed into a path of length $m-1$ from s_i to an intermediate state s_k and an edge from s_k to s_j. Hence, we obtain all paths of length m by summing over all intermediate states s_k and combining the number $N_{i,k,m-1}$ of paths to s_k with the number $N_{k,j,1}$ of outgoing edges to s_j. Since $(M^{m-1})_{ij} = N_{i,k,m-1}$ holds by induction, we get

$$N_{i,j,m} = \sum_{k=1}^{n} N_{i,k,m-1} \cdot N_{k,j,1} = \sum_{k=1}^{n} (M^{m-1})_{ik} \cdot M_{kj} = (M^m)_{ij}.$$

□

Example 5.11. Consider the finite automaton from Figure 2. Then we obtain the adjacency matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 1 & 0 & 1 \end{pmatrix} \qquad \text{and} \qquad M^4 = \begin{pmatrix} 3 & 2 & 4 \\ 6 & 3 & 8 \\ 4 & 2 & 5 \end{pmatrix}.$$

For example there are $(M^4)_{21} = 6$ paths from state 2 to state 1 by using Proposition 5.10. These are given by

$$2 \xrightarrow{c} 3 \xrightarrow{f} 3 \xrightarrow{f} 3 \xrightarrow{e} 1, \qquad 2 \xrightarrow{d} 3 \xrightarrow{f} 3 \xrightarrow{f} 3 \xrightarrow{e} 1,$$
$$2 \xrightarrow{c} 3 \xrightarrow{e} 1 \xrightarrow{a} 2 \xrightarrow{b} 1, \qquad 2 \xrightarrow{d} 3 \xrightarrow{e} 1 \xrightarrow{a} 2 \xrightarrow{b} 1,$$
$$2 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{c} 3 \xrightarrow{e} 1, \qquad 2 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{d} 3 \xrightarrow{e} 1.$$

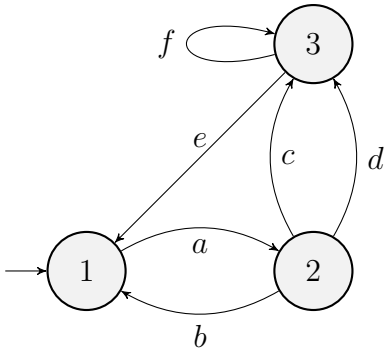


Figure 2: Finite automaton with many transitions.

Remark 5.12. Using Proposition 5.10, we obtain a formula for all accepting paths up to length m by summing over all paths of length $k \leq m$ ending at a final states $s_j \in F$. Let M be the adjacency matrix of Γ and assume s_1 is the initial state. Then we obtain

$$|\mathcal{B}_m| = \sum_{k=0}^{m} \sum_{s_j \in F} (M^k)_{1j}.$$

If M is for example diagonalizable or admits another simple form, one can possibly derive an explicit expression for $|\mathcal{B}_m|$.

6 Algorithms and time complexity

In this section we present a construction and two algorithms. At first we construct a finite automaton from a Gröbner basis in order to describe a quotient basis as in Remark 5.8. Such a finite automaton is required to construct the transformation matrix Ψ_m of ψ_m from Definition 4.6. Further, it allows us to compute $\dim \pi(P_m)$ in order prove Lemma 4.15. We refer again to [HMU06] for an introduction to languages and automaton theory.

Next we present two algorithms to prove Theorem 4.8 in Section 7. In particular, Algorithm 1 constructs the transformation matrix Ψ_m and Algorithm 2 shows that $\ker \psi_m = \{0\}$. This implies that no separating polynomial up to degree 50 exists for the magic unitary M_3.

At the end of this section we analyse the complexity of the last algorithm and show that checking until degree m requires time polynomial in m. We refer to [CLRS09] for an introduction to the design and analysis of algorithms. In particular, we will use basic data structures, like stacks, queues and hash maps, which can be found in Chapter 10 and Chapter 11 of [CLRS09].

6.1 Automaton construction

Let $I \subseteq \mathbb{C}\langle X \rangle$ be an ideal for some arbitrary finite set X. In this section, we give a general construction of a deterministic finite automaton which describes a basis for $\mathbb{C}\langle X \rangle / I$ as mentioned in Remark 5.8.

In particular, this construction gets applied later to the magic unitary ideal I_4 in order to obtain a basis $\mathcal{B}_m$ for $\pi(P_m) \subseteq \mathbb{C}\langle X_4 \rangle / I_4$ in the notation of Section 4.1. This basis gets used in Section 6.2 to construct the transformation matrix Ψ_m of the mapping ψ_m from Definition 4.6. Further, this finite automaton will be used to compute $\dim \pi(P_m)$ as described in Section 5.3. We refer to Remark 4.9 and Remark 4.16 for a overview of the proofs of our main results and the role of the finite automaton in this context.

Now consider again the general case. At first, we need a Gröbner basis for the ideal I. To compute a Gröbner basis we used the computer algebra system GAP and the package GBNP. For further details see [GAP20] and [CK16]. The corresponding script can be found in Listing 1 in Appendix C.

Remark 6.1. We are in the non-commutative setting which makes the computation of a Gröbner basis challenging. It is not guaranteed that an ideal I has a finite Gröbner basis, even if I is generated by finitely many polynomials. Hence, an algorithm computing a non-commutative Gröbner basis might not terminate. This is linked to the undecidability of the word problem in semigroups which is discussed in Section 1.3 of [Mor94].

Now given a Gröbner basis $G \subseteq I$, according to Remark 5.6 a basis for $\mathbb{C}\langle X \rangle / I$ is given by the residue classes of

$$\mathcal{B} = \{x \in X^\star \mid x \text{ contains no leading term from } G\}.$$

Hence, we want to construct a deterministic finite automaton which accepts all words $x \in X^\star$ which do not contain any word from $\{\mathrm{LT}(g) \mid g \in G\}$ as a subword. In the following we start with the construction of a non-deterministic finite

automaton which accepts all words over an alphabet which contain some subwords. This automaton gets then transformed into a suitable automaton for $\mathcal{B}$.

Proposition 6.2. *Let Σ be an alphabet and $S \subseteq \Sigma^\star$ a finite set of words. Then there exists a finite automaton Γ_S with accepting language*

$$L(\Gamma_S) = \{w \in \Sigma^\star \mid w \text{ contains a word from } S \text{ as subword}\}.$$

Proof. Define the set of vertices

$$V := \{p \in \Sigma^\star \mid \exists s \in S\colon p \text{ is a prefix of } s\}.$$

Then V is finite and contains all prefixes of all words in S. Further, define the initial state $s_0 := \varepsilon$ as the the empty word and the final states $F := S$. For each pair $v, w \in V$ add a directed edge from v to w with label $\alpha \in \Sigma$ if $w = v\alpha$. In addition, add for each $\alpha \in \Sigma$ a self-loop to s_0 and to all $s \in F$. Then the resulting graph looks like a tree where each word $w = w_1 w_2 \cdots w_n \in S$ has a corresponding path

$$\varepsilon \xrightarrow{w_1} w_1 \xrightarrow{w_2} w_1 w_2 \xrightarrow{w_3} \ldots \xrightarrow{w_n} w_1 w_2 \cdots w_n.$$

Now consider a word $w \in \Sigma^\star$ which contains a word $s \in S$ as subword. Then w has the form $w = asb$ for some $a, b \in \Sigma^\star$. The self-loops at the initial state ε allow to process the word a. Then the word s can be processed along the path to the final state s. Finally, the word b can be processed using the self-loops at the final state s. Hence, the word w gets accepted by the finite automaton Γ_S.

Let $w \in \Sigma^\star$ be a word which gets accepted by Γ_S. Then there exists a final state, corresponding to a word $s \in S$, where the word w ends. But then the word w has to contain the word s as subword in order to reach the final state s.

Hence, we obtain

$$L(\Gamma_S) = \{w \in \Sigma^\star \mid w \text{ contains a word from } S \text{ as subword}\}.$$

□

The last proposition is illustrated in the following example.

Example 6.3. Consider the alphabet $\Sigma = \{a, b, c\}$ and the words $S = \{ac, aba, abc\}$. Then Figure 3 shows the resulting finite automaton Γ_S from Proposition 6.2, which accepts all words $w \in \Sigma^\star$ containing *ac*, *aba* or *abc* as subword. For example, the word *baca* gets accepted because of the path

$$\varepsilon \xrightarrow{b} \varepsilon \xrightarrow{a} a \xrightarrow{c} ac \xrightarrow{a} ac.$$

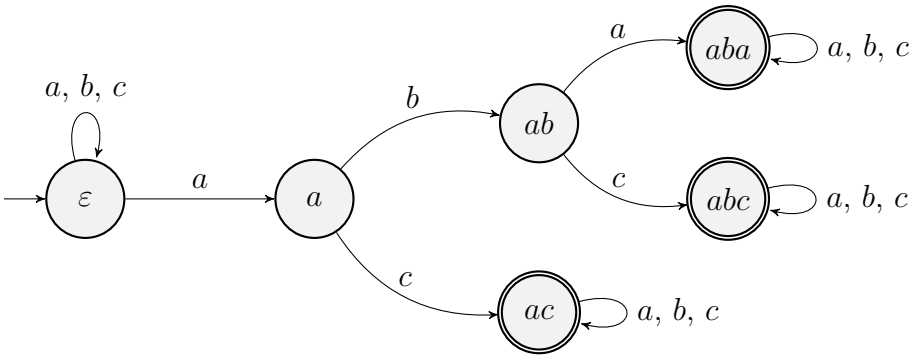


Figure 3: A finite automaton recognizing subwords.

Using Proposition 6.2 we can now construct a deterministic finite automaton accepting all words which do not contain any given subwords.

Lemma 6.4. *Let Σ be an alphabet and $S \subseteq \Sigma^\star$ a finite set of words. Then there exists a deterministic finite automaton Γ with accepting language*

$$L(\Gamma) = \{w \in \Sigma^\star \mid w \text{ contains no word from } S \text{ as subword}\}.$$

Proof. Consider the non-deterministic finite automaton Γ_S from Proposition 6.2. Using Proposition 2.22 we can transform Γ_S into a deterministic finite automaton $\det(\Gamma_S)$ which accepts the same language. Next we complement the automaton $\det(\Gamma_S)$ using Proposition 2.21 to obtain the final automaton $\Gamma := \overline{\det(\Gamma_S)}$. Then Γ accepts all words in $\Sigma^\star$ which do not contain any subword from S. □

Remark 6.5. Another possibility than using Proposition 2.22 to transform a non-deterministic finite automaton into a deterministic one is to use the Brzozowski algorithm [Brz62], which additionally minimizes the number of states. It is for example implemented in the computer algebra system SageMath [The20] which we used.

Note that in the worst case the Brzozowski algorithm requires exponential time depending on the size of the input finite automaton. However, it was fast enough in our case and the computation of the Gröbner basis was the more expensive part. Otherwise, one could for example use the more complex Aho-Corasick algorithm [AC75] to directly construct the deterministic automaton.

The deterministic finite automaton can then be complemented by basically swapping the accepting and non-accepting states. This algorithm is also implement in the computer algebra system SageMath.

Example 6.6. Consider again the alphabet $\Sigma = \{a, b, c\}$ and the words $S = \{ac, aba, abc\}$. Figure 4 shows the transformed version of the automaton in Example 6.3. It accepts all words in $\Sigma^\star$ which do not contain ac, aba or abc as a subword. In addition, it is minimized such that it has only 3 states.

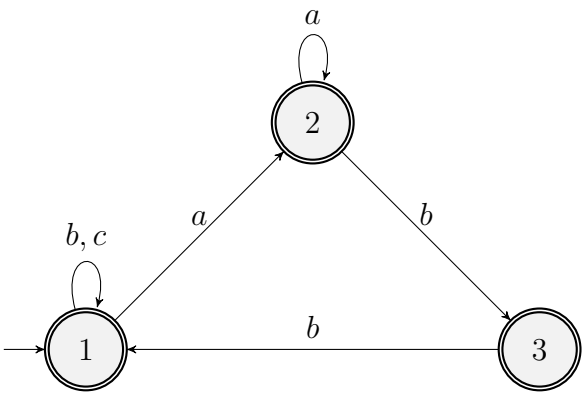


Figure 4: The transformed finite automaton.

The construction in Lemma 6.4 can now be used to construct a deterministic finite automaton which describes a basis for $\mathbb{C}\langle X\rangle / I$.

Lemma 6.7. *Let $I \subseteq \mathbb{C}\langle X\rangle$ be an ideal and G a Gröbner basis for I. Then there exists a finite automaton Γ over the alphabet X such that the accepting language $L(\Gamma)$ is a basis for $\mathbb{C}\langle X\rangle/I$. In particular, all accepted words up to length m are a basis for $\pi(P_m) \subseteq \mathbb{C}\langle X\rangle/I$. Here $P_m \subseteq \mathbb{C}\langle X\rangle$ is the vector space of polynomials up to degree m and $\pi\colon \mathbb{C}\langle X\rangle \to \mathbb{C}\langle X\rangle/I$ the canonical projection.*

Proof. According to Lemma 5.5 and Remark 5.6 a basis for $\mathbb{C}\langle X\rangle/I$ is given by

$$\mathcal{B} = \{x \in X^\star \mid x \text{ contains no leading term from } G\}.$$

Define $S := \{\mathrm{LT}(g) \mid g \in G\}$ and let Γ be the deterministic finite automaton from Lemma 6.4. Then

$$L(\Gamma) = \{x \in X^\star \mid x \text{ contains no word from } S \text{ as subword}\} = \mathcal{B}.$$

Further, Remark 5.7 shows that

$$\mathcal{B}_m := \{x \in \mathcal{B} \mid \deg x \leq m\}$$

is a basis for $\pi(P_m) \subseteq \mathbb{C}\langle X\rangle/I$. Since $\deg p$ corresponds to the length of the word p, we conclude that $\mathcal{B}_m$ consists of all accepted words up to length m. □

Remark 6.8. The corresponding finite automaton for the magic unitary ideal I_4 can be found in Appendix A and will be discussed in Section 7. The code for computing this automaton can be found in Appendix C. Listing 1 contains the GAP script which computes a Gröbner basis for the magic unitary ideal I_4. The output is then used in Listing 2 to compute the finite automaton.

6.2 Algorithm 1: Matrix construction

Consider again the replacement homomorphism $\varphi_{M_3}\colon \mathbb{C}\langle X_4\rangle \to A_{pq}^{\otimes 3}$ in the notation of Section 4. It can be factored as

$$\begin{array}{ccc} \mathbb{C}\langle X_4\rangle & \xrightarrow{\quad\varphi_{M_3}\quad} & A_{pq}^{\otimes 3} \\ & {\scriptstyle\pi}\searrow \quad \nearrow{\scriptstyle\psi} & \\ & \mathbb{C}\langle X_4\rangle/I_4 & \end{array}$$

where I_4 is the magic unitary ideal. As in Definition 4.6 denote by ψ_m the restriction of ψ on the subspace $\pi(P_m) \subseteq \mathbb{C}\langle X_4\rangle/I_4$, where $P_m \subseteq \mathbb{C}\langle X_4\rangle$ is the vector space of polynomials up to degree m.

In this section we describe how to construct the transformation matrix Ψ_m of the mapping ψ_m. The matrix Ψ_m gets used in the next section to show that $\ker \psi_m = \{0\}$ in order to proof our main result in Section 7.

Using Lemma 5.4 we obtain $\operatorname{Im}\psi_m \subseteq \langle \mathcal{A}^{\otimes 3}\rangle$. Here

$$\mathcal{A} := \{1, p, q, pq, qp, pqp, \ldots\} \subseteq A_{pq}$$

is defined as in Lemma 5.1 and

$$\mathcal{A}^{\otimes 3} := \{v_1 \otimes v_2 \otimes v_3 \mid v_1, v_2, v_3 \in \mathcal{A}\} \subseteq A_{pq}^{\otimes 3}.$$

Hence, we can choose $\mathcal{A}^{\otimes 3}$ as a basis for $\operatorname{Im}\psi_m$.

Remark 6.9. It is straightforward to implement the arithmetics in $\langle \mathcal{A}^{\otimes 3} \rangle$. One possibility is to store an alternating product of p's and q's by its first factor and its length. It is possible to directly combine such products by looking at the parity and canceling two repeated factors if necessary. The corresponding code can be found in Listing 3 in Appendix C.

Now let $G \subseteq I_4$ be a Gröbner basis for I_4. Then Remark 5.7 shows that the residue classes of

$$\mathcal{B}_m = \{x \in X^\star \mid \deg x \leq m,\ x \text{ contains no leading term from } G\}$$

are a basis for the domain $\pi(P_m)$.

In Lemma 6.7 in the previous section we describe how to construct a deterministic finite automaton Γ, whose accepting path corresponds to $\mathcal{B}$. In particular, to construct $\mathcal{B}_m$ we have to consider all accepting paths up to length m in Γ. We refer to Section 5.2 for more information on the sets $\mathcal{B}$ and $\mathcal{B}_m$.

Remark 6.10. Note that in the finite automaton for I_4 every state is final, hence we can consider all paths up to length m. See Appendix A for the automaton Γ corresponding to I_4.

Since we have chosen the basis $\mathcal{B}_m$ for the domain and $\mathcal{A}^{\otimes 3}$ for the image of ψ_m, we can now construct the transformation matrix Ψ_m with respect to these bases.

Lemma 6.11. *Let Γ be a deterministic finite automaton describing the basis $\mathcal{B}_m$ of $\pi(P_m) \subseteq \mathbb{C}\langle X_4 \rangle / I_4$. Then Algorithm 1 constructs the transformation matrix Ψ_m of the mapping ψ_m with respect to the bases $\mathcal{B}_m$ and $\mathcal{A}^{\otimes 3}$.*

Proof. To construct the transformation matrix we have to evaluate

$$\psi_m([x]) = \varphi_{M_3}(x) = \prod_{k=1}^{n} \varphi_{M_3}(x_{i_k j_k}) = \prod_{k=1}^{n} M_{i_k j_k}$$

for each monomial $x = x_{i_1 j_1} x_{i_1 j_1} \cdots x_{i_n j_n} \in \mathcal{B}_m$. Each product gets then represented with respect to the basis $\mathcal{A}^{\otimes 3}$ and results in a column of the transformation matrix Ψ_m.

To efficiently enumerate the basis $\mathcal{B}_m$, we can use the deterministic finite automaton Γ. According to Lemma 6.7 and Remark 6.10 the elements of $\mathcal{B}_m$ correspond to all paths up to length m in Γ. These paths can be obtained by a breadth-first search. It is then possible to directly multiply the corresponding entries $(M_3)_{ij}$ along a path and obtain all $\varphi_{M_3}(x)$ for $x \in \mathcal{B}_m$ in the same step. More information on breadth-first searchs and other graph algorithms can for example be found in Chapter 22 of [CLRS09].

Algorithm 1 shows the pseudocode of the construction algorithm which implements a breadth-first search using a queue. We represent the state of the search by a triple (s, p, k). Here s is a node in Γ, $p \in \langle \mathcal{A}^{\otimes 3} \rangle$ is the monomial evaluated along the path and k is the length of the path. The idea is to start with $(s_0, 1 \otimes 1 \otimes 1, 0)$ and to remove the next item (s, p, k) in each step from the queue. Then consider all outgoing transitions from s to s' with label x_{ij} and add a new item $(s', p', k+1)$ to the queue. This new item extends the current path along this transition and contains the corresponding polynomial $p' = p \cdot (M_3)_{ij}$.

□

Remark 6.12. The main advantage of the approach using a finite automaton is that the running time only depends on the size of the basis $|\mathcal{B}_m|$ and not on the total number of monomials up to length m. In Section 6.4 we give a more detailed analysis and show that we obtain a running time which is polynomial in m.

A more naive approach would be to generate all monomials up to degree m, check if they lie in $\mathcal{B}_m$ and then evaluate them. However, this would require exponential time in m since there are 16^m of these monomials. Hence, this approach would be infeasible even for small m.

Algorithm 1 Matrix construction

Input:
matrix M_3, finite automaton Γ, maximimum degree m

Output:
matrix Ψ_m

```
1: initialize empty matrix Ψ_m
2: initialize queue Q with (s_0, 1 ⊗ 1 ⊗ 1, 0)
3: while Q is not empty do
4:     remove (s, p, k) from Q
5:     insert column corresponding to p into Ψ_m
6:     if k < m then
7:         for all transition s → s' with label x_ij do
8:             p' ← p · (M_3)_ij
9:             insert (s', p', k + 1) into Q
```

6.3 Algorithm 2: Kernel of Ψ_m

Next we present an efficient algorithm to show that $\ker \Psi_m = \{0\}$ where Ψ_m is the matrix constructed in the last section. Since Ψ_m is the transformation matrix of ψ_m, we then obtain $\ker \psi_m = \{0\}$. Consequently, no separating polynomials up to degree m exist for the magic unitary M_3 in this case. This result for $m = 50$ will then be used to prove Theorem 4.8 which implies our main result.

Before we come to the algorithm we describe how we store the matrix Ψ_m as a sparse matrix.

Definition 6.13 (Sparse matrix format)**.** The matrix Ψ_m is stored as a pair (*Rows*, *Columns*) where *Row* and *Columns* are maps such that

1. $\mathit{Rows}(i) = \{j_1, \ldots, j_{n_i}\}$ is the set of non-zero columns $j_1, \ldots, j_{n_i}$ in row i,
2. $\mathit{Columns}(j) = \{i_1, \ldots, i_{m_j}\}$ is the set of non-zero rows $i_1, \ldots, i_{m_j}$ in column j.

This structure is chosen for efficiency reasons. The two maps might seem redundant but they allow direct access to every information without searching through the matrix. Furthermore, only non-zero entries are stored. This saves memory as it turned out that Ψ_m is sparse. Algorithm 2 contains the pseudocode of our algorithm for showing $\ker \Psi_m = \{0\}$. It implements a special form of Gaussian elimination and uses a stack and the previously described maps to efficiently update the matrix and process rows. In particular, Algorithm 2 computes the following.

Lemma 6.14. *Algorithm 2 computes a lower bound on the rank of the matrix Ψ_m. If the lower bound equals the number of columns of Ψ_m, then ψ_m is injective.*

Proof. The main idea of the algorithm is to use a special form of Gaussian elimination and transform Ψ_m using elementary row operations, which preserve the rank of Ψ_m. We are looking for a row i which contains only one non-zero entry in some column j. Next we eliminate all entries in column j which occur in other rows $k \neq i$. If such a row then contains only one non-zero remaining entry, we can push k to a stack. This stack contains the rows with one remaining entry which we consider next. Each row with one non-zero entry is linearly independent to all other rows. Hence, the total number of such rows is a lower bound on the rank of the matrix Ψ_m. Note that a row which was pushed to the stack could have been eliminated completely before it gets processed. In this case the row can be written as a linear combination of other rows and does not contribute to the rank of the matrix.

Assume we obtain a lower bound equal to the number of columns of Ψ_m. Since the number of columns is a upper bound on the rank of a matrix, the rank of Ψ_m has to equal the number of its columns. Hence $\dim \ker \Psi_m = 0$ and Ψ_m has to be injective. Since Ψ_m is the transformation matrix of ψ_m, we obtain that ψ_m is injective. □

Algorithm 2 Matrix elimination

Input:

sparse $n_r \times n_c$ matrix $\Psi_m = (\mathit{Rows}, \mathit{Columns})$

Output:

lower bound *rank* for the rank of Ψ_m

```
1:  rank ← 0
2:  initialize empty stack S
3:  for i = 1, ..., n_r do
4:      if |Rows(i)| = 1 then
5:          push i to S
6:  while S is not empty do
7:      pop i from S
8:      if |Rows(i)| = 1 then
9:          {j} ← Rows(i)
10:         for k ∈ Columns(j) do
11:             if k ≠ i then
12:                 delete j from Rows(k)
13:                 if |Rows(k)| = 1 then
14:                     push k to S
15:         Columns(j) ← {i}
16:         rank ← rank + 1
```

6.4 Time complexity

Now we analyse the time complexity of Algorithm 1 and Algorithm 2. See again [CLRS09] for more information on the time complexity of data structures, like hash maps or linked-lists, and on the analysis of algorithms in general. At first we introduce some notation.

Definition 6.15. Let $f, g\colon \mathbb{N} \to \mathbb{N}$, then we say $f \in \mathcal{O}(g)$ if

$$\limsup_{x\to\infty} \frac{f(x)}{g(x)} < \infty.$$

This means f does not grow faster than g ignoring constant factors.

When analysing algorithms we try to estimate the number of basic operations depending on an input size n. Such operations are for example arithmetic operations, comparisons or memory operations like reading or assigning a variable. We say an algorithm has complexity $\mathcal{O}(f(n))$ if the number of such basic operations, depending on n, is in $\mathcal{O}(f(n))$. In the following we fix the matrix dimension $n = 4$ and let the maximimum degree m be our input size.

At first consider the algebra $A_0 := \langle \mathcal{A}^{\otimes 3} \rangle$ from Lemma 5.3. Denote by $|a|$ the number of non-zero coefficients of $a \in A_0$ if a is represented with respect to $\mathcal{A}^{\otimes 3}$. Then we obtain the following complexity for multiplication in A_0.

Lemma 6.16. *Let $a, b \in A_0$. Then $a \cdot b$ can be computed in $\mathcal{O}(|a| \cdot |b|)$ with respect to the basis $\mathcal{A}^{\otimes 3}$.*

Proof. This can be done with naive multiplication by using the linearity of $\otimes$ and combining each basis vector of a with each basis vector of b. If one represents an alternating product of p's and q's for example as a pair of its first factor and its length, one can combine two basis vectors in constant time. □

Next we consider the complexity of Algorithm 1 and Algorithm 2.

Lemma 6.17. *Algorithm 1 for constructing the matrix Ψ_m has a complexity of $\mathcal{O}(N)$ where N is the number of non-zero entries in Ψ_m.*

Proof. With Lemma 6.16 follows that

$$p' \leftarrow p \cdot (M_3)_{ij}$$

takes time $\mathcal{O}(|p|)$ since we fix M_3 such that $|(M_3)_{ij}|$ is bounded. Hence, computing a triple (s, p, k) can be done in $\mathcal{O}(|p|)$. Using hash maps for our sparse matrix $\Psi_m = (\mathit{Rows}, \mathit{Columns})$, we can insert the coefficients of p into Ψ_m in $\mathcal{O}(|p|)$ on average. Furthermore, a queue allows all operations in $\mathcal{O}(1)$ if implemented for example as linked-list. Hence, we obtain an overall complexity of $\mathcal{O}(N)$ for Algorithm 1 where

$$N := \sum_p |p| = \sum_{x \in \mathcal{B}_m} |\psi_m(x)|$$

is the number of non-zero entries in the matrix Ψ_m. □

Lemma 6.18. *Algorithm 2 for checking $\ker \Psi_m = \{0\}$ has a complexity of $\mathcal{O}(N)$ where N is the number of non-zero entries in Ψ_m.*

Proof. Since we store the matrix Ψ_m in the special form $\Psi_m = (\mathit{Rows}, \mathit{Columns})$, we can process each non-zero entry of Ψ in $\mathcal{O}(1)$. This is because each access to *Rows* or *Columns* takes time $\mathcal{O}(1)$ on average and a stack can be accessed and modified in $\mathcal{O}(1)$ if implemented for example as linked-list. Similar, sets allow access and modifications in $\mathcal{O}(1)$ on average if they are implemented hash-based. Hence, we obtain an overall complexity of $\mathcal{O}(N)$ for eliminating the matrix where N is the total number of non-zero entries in Ψ_m. □

Remark 6.19. Since every algorithm computing Ψ_m needs at least $\mathcal{O}(N)$ time to compute the N non-zero entries, both algorithms are optimal in this sense.

Next we estimate the number of non-zero entries N such that we obtain an overall complexity which only depends on m.

Lemma 6.20. *The matrix Ψ_m has $\mathcal{O}(m^6)$ non-zero entries.*

Proof. We estimate the dimension of the matrix Ψ_m to obtain an upper bound on the total number of entries in Ψ_m. This number is also an upper bound on the number N of non-zero entries in Ψ_m.

The number of columns of Ψ_m is given by $|\mathcal{B}_m| = \dim \pi(P_m)$ which equals $\binom{2m+3}{3}$ by Lemma 4.15. On the other hand, consider the set

$$\mathcal{A}_m := \{1, p, q, \ldots, \underbrace{(pq\ldots)}_{m}, \underbrace{(qp\ldots)}_{m}\} \subseteq \mathcal{A}.$$

Since monomials in $\mathcal{B}_m$ consists of at most m factors, we get that $\operatorname{Im} \Psi_m$ is contained in the linear span of

$$\mathcal{A}_m^{\otimes 3} := \{v_1 \otimes v_2 \otimes v_3 \mid v_1, v_2, v_3 \in \mathcal{A}_m\}.$$

Because $|\mathcal{A}_m| = 2m + 1$, we obtain that $|\mathcal{A}_m^{\otimes 3}| = (2m+1)^3$. Hence

$$\dim \operatorname{Im} \Psi_m \leq \left|\mathcal{A}_m^{\otimes 3}\right| = (2m+1)^3$$

such that Ψ_m has at most $(2m+1)^3$ rows. This implies that

$$N \leq \binom{2m+3}{3} \cdot (2m+1)^3 = \frac{32}{3}m^6 + 48m^5 + \ldots + 1.$$

where N is the number of non-zero entries in Ψ_m. Consequently, Ψ_m has $\mathcal{O}(m^6)$ non-zero entries. □

Lemma 6.21. *Algorithm 1 and Algorithm 2 for constructing the matrix Ψ_m and checking $\ker \Psi_m = \{0\}$ have a complexity of $\mathcal{O}(m^6)$*

Proof. Lemma 6.17 and Lemma 6.18 show that Algorithm 1 and Algorithm 2 have a complexity of $\mathcal{O}(N)$ where N is the number of non-zero entries in Ψ_m. Lemma 6.20 then shows that $N \in \mathcal{O}(m^6)$. □

7 Proofs of the main results

In this section we present the results from running the algorithms in the previous section and prove Theorem 4.8 and Lemma 4.15. The corresponding listings containing the code can be found in Appendix C. In the end we discuss the generalizability of our approach to $n > 4$.

We begin with Gröbner bases and the construction of the finite automaton. The GAP script for computing a Gröbner basis for the magic unitary ideal I_4 can be found in Listing 1. Its output can be used to construct the corresponding finite automaton using the script in Listing 2 which implements the approach described in Section 6.1.

The resulting finite automaton and its adjacency matrix can then be found in Appendix A. It has 17 states where state 0 is the initial state and every state is accepting.

Proof of the dimension formula. Using this automaton it is now possible to prove Lemma 4.15 from Section 4.2.

Proof of Lemma 4.15. Recall that we want to show the following statement:

Let $P_m \subseteq \mathbb{C}\langle X_4\rangle$ be the vector space of polynomials up to degree m and $\pi\colon \mathbb{C}\langle X_4\rangle \to \mathbb{C}\langle X_4\rangle / I_4$ the canonical projection. Then $\dim \pi(P_m) = \binom{2m+3}{3}$.

According to Remark 5.7 a basis for $\pi(P_m)$ is given by

$$\mathcal{B}_m = \{x \in \mathcal{B} \mid \deg x \leq m\}$$

where the set $\mathcal{B}$ is defined as in Lemma 5.5. In Section 6.1 we showed that the paths up to length m in the previously computed finite automaton correspond to monomials in $\mathcal{B}_m$.

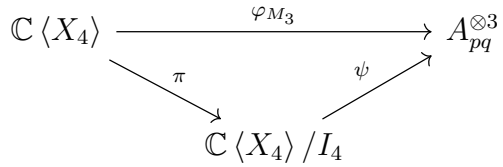
By using Proposition 5.10 and computing the Jordan normal form of the adjacency matrix in Listing 2, we obtain that there are $(2k+1)^2$ accepting paths of length k. Hence

$$\dim \pi(P_m) = |\mathcal{B}_m| = \sum_{k=0}^{m} (2k+1)^2 \overset{(*)}{=} \frac{1}{6}(2m+1)(2m+2)(2m+3) = \binom{2m+3}{3}.$$

Here $(*)$ can be proven using induction. □

Remark 7.1. We were also able to compute Gröbner bases for the magic unitary ideals I_5 and I_6 and obtained finite automata with 26 and 37 states respectively. These contain $\mathcal{O}(6.86^k)$ and $\mathcal{O}(13.93^k)$ paths of length k. Furthermore, $|\mathcal{B}_m|$ grows indeed exponentially in these cases such that it would be infeasible to construct the matrix Ψ_m even for smaller m.

Proof of the main theorem. Recall the notation from Section 4 and consider the replacement homomorphism $\varphi_{M_3}\colon \mathbb{C}\langle X_4\rangle \to A_{pq}^{\otimes 3}$ which can be factored as



As in Definition 4.6 denote by ψ_m the restriction of ψ on the subspace $\pi(P_m) \subseteq \mathbb{C}\langle X_4\rangle / I_4$, where $P_m \subseteq \mathbb{C}\langle X_4\rangle$ is the vector space of polynomials up to degree m.

Both Algorithm 1 and Algorithm 2 from Section 6 are implemented in Listing 4. They construct the transformation matrix Ψ_m of ψ_m and check that $\ker \psi_m = \{0\}$ for a given m. We ran these algorithms on a computer until $m = 50$ and found that $\ker \psi_m = \{0\}$ in these cases. This can be used to prove Theorem 4.8 from Section 4.2.

Proof of Theorem 4.8. Recall the statement of the theorem:

Let $p \in \mathbb{C}\langle X_4\rangle$ with $\deg p \leq 50$. If $p(M_3) = 0$ then $p \in I_4$.

Now let $p \in \mathbb{C}\langle X_4\rangle$ with $\deg p \leq 50$ and $p(M_3) = 0$. Then $\varphi_{M_3}(p) = 0$ and $\pi(p) \in \ker \psi_m$ since $\varphi_{M_3} = \psi \circ \pi$. Using Algorithm 1 and Algorithm 2 from Section 6 we obtain $\ker \psi_m = \{0\}$ for $m \leq 50$. Hence, $\pi(p) = 0$ which implies $p \in \ker \pi = I_4$. □

Remark 7.2. The following table contains the number of rows, columns and non-zero entries of Ψ_m for some $m \leq 50$. Furthermore, we include the memory usage and runtime of our script. The complete table can be found in Appendix B.

m	rows	columns	non-zero entries	memory in MB	runtime in s
0	1	1	1	64.10	0.01
1	25	10	56	64.93	0.02
2	123	35	533	71.19	0.10
3	337	84	2,329	72.56	0.32
4	715	165	7,661	77.31	0.81
5	1,325	286	2,1475	87.71	1.65
10	9,259	1,771	766,875	291.82	10.26
15	29,785	5,456	7,428,961	1,459.33	47.27
20	68,907	12,341	38,549,908	6,617.87	223.58
25	132,645	23,426	141,363,373	21,644.80	811.13
30	226,979	39,711	409,560,591	58,731.43	2404.19
40	531,427	91,881	2,226,617,868	287,786.20	15656.99
50	1,030,299	176,851	8,332,940,867	904,425.89	75200.15

Generalizability. Consider again Question 3.3 which asks for vanishing polynomials in the general setting where n can be greater than 4. It turns out that our approach does not work for $n = 5, 6$ and any sequence of pairs (B_k, M_k) constructed in [JW20].

Consider such an arbitrary sequence of pairs (B_k, M_k) and fix a $k \in \mathbb{N}$. Then $B_k \subseteq A_{pq}^{\otimes N_k}$ for some $N_k \in \mathbb{N}$. As in the case $n = 4$ we can consider the magic unitary ideal I_n and factor the replacement homomorphism $\varphi_{M_k} \colon \mathbb{C}\langle X_n \rangle \to A_{pq}^{\otimes N_k}$ as $\varphi_{M_k} = \psi \circ \pi$. Next we restrict ψ to $\pi(P_m)$ and consider the mappings ψ_m from Definition 4.6. A similar argument as in Lemma 6.20 shows that $\dim \operatorname{Im} \psi_m \leq (2m+1)^{N_k}$. Hence, the dimension grows only polynomial in m. On the other hand, Remark 7.1 shows that $\dim \pi(P_m)$ grows exponentially in m for $n = 5$ and $n = 6$. Consequently, there exists a m_0 such that $\dim \operatorname{Im} \psi_{m_0} < \dim \pi(P_{m_0})$. In this case $\ker \psi_{m_0} \neq \{0\}$ and ψ_{m_0} is not injective anymore.

Consider a non-zero $[p] \in \ker \psi_{m_0}$ then p does not lie in the magic unitary ideal I_n but vanishes in B_k. Hence, p is a separating polynomial for M_k such that Theorem 4.8 no longer holds for M_k and a degree of at least m_0.

The polynomial p would be a candiate to answer Question 3.3 positively, however it is still possible that p would also vanish in $C(S_n^+)$. Consequently, we are not able to answer Question 3.3 and both outcomes are still possible.

The previous dimension argument further shows that the case $n = 4$ and the quantum permutation group S_4^+ are special and that S_4^+ has other properties than S_n^+ for $n \geq 5$. A different property is for example that S_4^+ is amenable whereas S_n^+ is not amenable anymore for $n \geq 5$ (see [Ban98]). This gives further evidence that somewhat simple models of S_4^+ exist.

A Finite automaton

Figure 5 shows the finite automaton which was constructed from a Gröbner basis for the magic unitary ideal I_4. See again Section 6.1 for details on the construction.

The automaton was computed using Listing 1 and Listing 2 in Appendix C. Paths up to length m correspond to monomials which form a basis of $\pi(P_m) \subseteq \mathbb{C}\langle X_4 \rangle / I_4$. The grey edges are only coloured differently for a better visualization. The finite automaton has 17 states where state 0 is the initial state and all states are final.

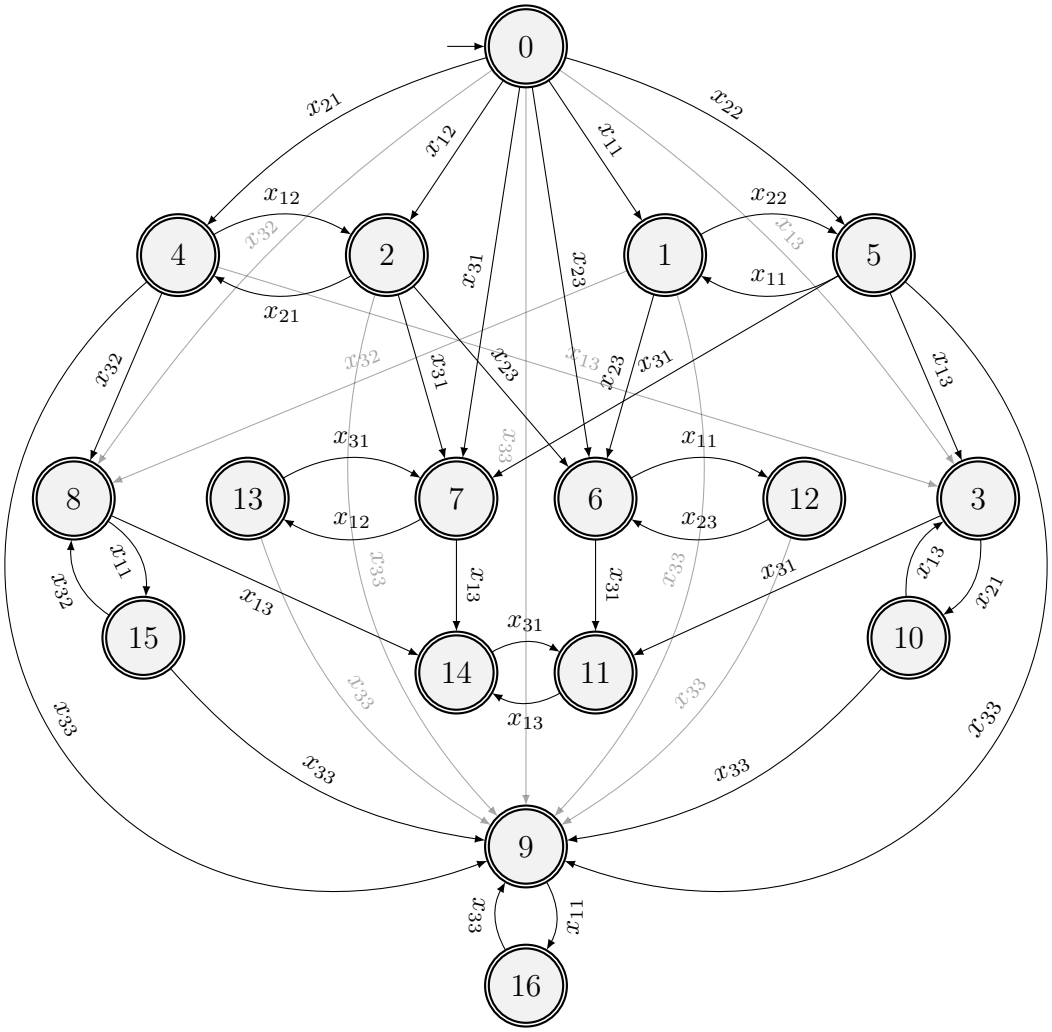


Figure 5: Finite automaton for the magic unitary ideal I_4.

The following matrix is the adjacency matrix of the finite automaton in Figure 5. The rows and columns of this matrix correspond to the states $0, \ldots, 16$. Note that zeros are represented by dots.

$$\begin{pmatrix}
\cdot & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{pmatrix}$$

B Program output

Table 1 shows the output of the main script in Listing 4 up to $m = 50$. It contains information on the matrix Ψ_{50}, on the runtime and memory usage of the program. Note that the memory usage and runtime were measured after all paths up to length k were processed and triples for paths of length $k + 1$ were already stored in the queue. The last row was recorded after the whole matrix Ψ_{50} was constructed. The complete runtime was 83292.65s $\approx$ 23h where Algorithm 1 took 75200.15s and Algorithm 2 took 8092.49s.

m	rows	columns	non-zero entries	memory in MB	runtime in s
0	1	1	1	64.10	0.01
1	25	10	56	64.93	0.02
2	123	35	533	71.19	0.10
3	337	84	2,329	72.56	0.32
4	715	165	7,661	77.31	0.81
5	1,325	286	2,1475	87.71	1.65
6	2,195	455	5,2188	101.01	2.77
7	3,369	680	11,5576	124.14	4.17
8	4,899	969	230,836	158.97	5.76
9	6,853	1,330	435,710	207.65	7.71
10	9,259	1,771	766,875	291.82	10.26
11	12,161	2,300	130,1782	377.08	13.73
12	15,611	2,925	2,102,394	567.04	18.50
13	19,677	3,654	3,309,355	751.73	25.14
14	24,387	4,495	5,001,791	1,114.01	34.46
15	29,785	5,456	7,428,961	1,459.33	47.27
16	35,923	6,545	10,690,275	2,117.95	65.10
17	42,869	7,770	15,183,027	2,728.92	88.88
18	50,651	9,139	20,994,111	3,814.02	124.38
19	59,313	10,660	28,764,159	4,832.58	167.88
20	68,907	12,341	38,549,908	6,617.87	223.58
21	79,501	14,190	51,306,330	8,156.81	293.74
22	91,123	16,215	66,978904	11,075.49	388.22

23	103,817	18,424	86,984702	13,512.27	504.34
24	117,635	20,825	111,100,537	17,452.11	638.06
25	132,645	23,426	141,363,373	21,644.80	811.13
26	148,875	26,235	177,214,474	26,158.60	1014.31
27	166,369	29,260	221,537,068	31,445.63	1263.80
28	185,179	32,509	273,349,116	41,159.18	1582.60
29	205,373	35,990	336,588,947	48,915.85	1959.78
30	226,979	39,711	409,560,591	58,731.43	2404.19
31	250,041	43,680	497,647,792	70,061.36	2949.53
32	274,611	47,905	598,230,517	83,046.25	3593.48
33	300,757	52,394	718,460,728	98,681.49	4377.11
34	328,507	57,155	854,404,297	116,366.46	5284.42
35	357,905	62,196	1,015,513,951	138,693.17	6423.08
36	389,003	67,525	1,196,211,892	163,767.06	7718.23
37	421,869	73,150	1,408,706,390	193,272.38	9273.69
38	456,531	79,079	1,645,147,738	227,551.41	11162.03
39	493,033	85,320	1,921,306,990	264,278.20	13321.08
40	531,427	91,881	2,226,617,868	287,786.20	15656.99
41	571,781	98,770	2,580,967,612	338,526.96	19012.22
42	614,123	105,995	2,970,195,853	393,371.16	22895.79
43	658,497	113,564	3,419,412,752	457,684.81	27845.98
44	704,955	121,485	3,910,194,852	492,858.52	32279.49
45	753,565	129,766	4,473,713,835	577,502.78	39637.46
46	804,355	138,415	5,086,153,397	636,655.29	46904.09
47	857,369	147,440	5,786,063,119	685,628.80	54060.19
48	912,659	156,849	6,543,289,250	739,143.00	62507.53
49	970,293	166,650	7,404,918,969	796,545.04	71628.69
50	1,030,299	176,851	8,332,940,867	904,425.89	75200.15

Table 1: Output of Listing 4 up to $m = 50$.

C Code

The following section contains the code which was used to obtain our results. Listing 1 is used to compute a Gröbner basis for a magic unitary ideal I_n. We ran it with GAP 4.11.0 [GAP20] and it requires the package GBNP 1.0.3 [CK16]. The output of Listing 1 is then used in Listing 2 to construct a finite automaton corresponding to a basis for $\mathbb{C}\langle X_n\rangle / I_n$. This script was executed with Python 3.8.5 and requires the computer algebra system SageMath 9.1 [The20]. It outputs the finite automaton, its adjacency matrix and a formula for the number of paths. This formula is obtained by SageMath as described in Section 5.3 by computing the Jordan normal form of the adjacency matrix.

Listing 3 then contains the arithmetics in $A_{pq}^{\otimes 3}$ whereas Listing 4 contains the main program which implements Algorithm 1 and Algorithm 2 from Section 6. It was run with PyPy 7.0.0 for a speedup but it is compatible with Python 2.7.13 and Python 3.8.5. The maximum degree m is passed as first command-line argument.

```
LoadPackage("GBNP");

N := 4;  # size of the magic unitary

A := FreeAssociativeAlgebraWithOne(Rationals, N * N, "R");
Generators := GeneratorsOfAlgebra(A);
A_One := Generators[1];  # unit in A

# construct a matrix M containing the generators
M := [];
for i in [1..N] do
    Row := [];
    for j in [1..N] do
        Add(Row, Generators[N * (i - 1) + j + 1]);
    od;
    Add(M, Row);
od;

# define the magic unitary relations
Relations := [];

for i in [1..N] do
    # sum of elements in the same column
    Rel := -A_One;
    for j in [1..N] do
        Rel := Rel + M[i][j];
    od;
    AddSet(Relations, GP2NP(Rel));
    # sum of elements in the same row
    Rel := -A_One;
    for j in [1..N] do
        Rel := Rel + M[j][i];
    od;
    AddSet(Relations, GP2NP(Rel));
od;

for i1 in [1..N] do for j1 in [1..N] do
    for i2 in [1..N] do for j2 in [1..N] do
        if i1 = i2 and j1 = j2 then
            # square of an element
            Rel := M[i1][j1] * M[i1][j1] - M[i1][j1];
            AddSet(Relations, GP2NP(Rel));
        elif i1 = i2 or j1 = j2 then
            # product in the same row / column
            Rel := M[i1][j1] * M[i2][j2];
            AddSet(Relations, GP2NP(Rel));
        fi;
    od; od;
od; od;

# compute and display the Groebner basis
GB := SGrobner(Relations);
Display(GB);
```

Listing 1: `code/quotient.g`

```python
import ast
import sage.all as sage

# read the output of 'quotient.g' from the file 'groebner_basis.txt'
with open("groebner_basis.txt") as file:
    data = file.read()

basis = ast.literal_eval(data)

n = 4  # size of the magic unitary
variables = range(1, n * n + 1)
leading_terms = [tuple(monomials[0]) for monomials, _ in basis]

# construction of the non-deterministic finite automaton
initial_state = tuple()
transitions = []

# add self-loops to the initial state
for variable in variables:
    transitions.append((initial_state, initial_state, variable))

for word in leading_terms:
    # add a path corresponding to the leading term
    for i in range(len(word)):
        transitions.append((word[:i], word[:i+1], word[:i+1][-1]))

    # add self-loops to the final state
    for variable in variables:
        transitions.append((word, word, variable))

automaton = sage.Automaton(
    transitions,
    initial_states=[initial_state],
    final_states=leading_terms,
    input_alphabet=variables
)

# transform the automaton
automaton = automaton.minimization().complement()

# remove the 'dead state'
for state in automaton.states():
    is_dead_end = all(trans.to_state == state for trans in state.transitions)
    if not state.is_final and is_dead_end:
        break

automaton.delete_state(state)

# print the final automaton, adjacency matrix and number of paths
automaton = automaton.relabeled()
M = automaton.adjacency_matrix(entry=lambda x: 1)

print(automaton)
print(M)
print(automaton.number_of_words())
```

Listing 2: `code/automaton.py`

```python
def proj_mult(x, y):
    """
    Multiplies 'x' and 'y' as alternating products of p's and q's.
    Such products are represented as pairs (start, length).
    """
    start1, length1 = x
    start2, length2 = y

    # nothing to check if a factor is '1'
    if start1 == START_ONE:
        return y
    if start2 == START_ONE:
        return x

    # combine the products and possibly remove a factor
    if start1 == START_P:
        if ((start2 == START_P and length1 % 2 == 0) or
            (start2 == START_Q and length1 % 2 == 1)):
            return (START_P, length1 + length2)
        else:
            return (START_P, length1 + length2 - 1)

    if start1 == START_Q:
        if ((start2 == START_Q and length1 % 2 == 0) or
            (start2 == START_P and length1 % 2 == 1)):
            return (START_Q, length1 + length2)
        else:
            return (START_Q, length1 + length2 - 1)


def tensor_mult(x, y):
    """
    Multiplies 'x' and 'y' as tensor product of alternating p's and q's.
    Such a tensor products are represented as triple (alt1, alt2, alt3).
    """
    return (
        proj_mult(x[0], y[0]), proj_mult(x[1], y[1]), proj_mult(x[2], y[2]),
    )

def mult(x, y):
    """
    Multiplies 'x' and 'y' as linear combinations of tensor products.
    Such linear combinations are represented as dictionary
    {tensor1: coeff1, tensor2: coeff2, ...} storing only non-zero coefficients.
    """
    result = {}
    for tensor1, coeff1 in x.items():
        for tensor2, coeff2 in y.items():
            new_tensor = tensor_mult(tensor1, tensor2)
            new_coeff  = result.get(new_tensor, 0) + coeff1 * coeff2
            if new_coeff != 0:
                result[new_tensor] = new_coeff
            elif new_tensor in result:
                del result[new_tensor]
    return result

START_ONE, START_P, START_Q = 0, 1, 2

```

```
I = (START_ONE, 0)  # the unit '1'
P = (START_P,   1)  # a single 'p'
Q = (START_Q,   1)  # a single 'q'

# the matrix M_3 := R^{\operp 3}
M = [
    [
        {(P, P, P): 1,
         (I, Q, I): 1, (I, Q, P): -1, (P, Q, I): -1, (P, Q, P): 1},
        {(P, I, Q): 2, (P, P, Q): -1,
         (I, I, I): 1, (I, I, Q): -1, (I, Q, I): -1, (I, Q, Q): 1,
         (P, I, I): -1, (P, Q, I): 1, (P, Q, Q): -1},
        {(P, P, I): 1, (P, P, P): -1, (I, Q, P): 1, (P, Q, P): -1},
        {(P, I, I): 1, (P, I, Q): -2, (P, P, I): -1, (P, P, Q): 1,
         (I, I, Q): 1, (I, Q, Q): -1, (P, Q, Q): 1}
    ],
    [
        {(I, P, P): 1, (P, P, P): -1, (P, Q, I): 1, (P, Q, P): -1},
        {(I, I, Q): 1, (I, P, Q): -1, (P, I, Q): -2, (P, P, Q): 1,
         (P, I, I): 1, (P, Q, I): -1, (P, Q, Q): 1},
        {(I, P, I): 1, (I, P, P): -1, (P, P, I): -1, (P, P, P): 1,
         (P, Q, P): 1},
        {(I, I, I): 1, (I, I, Q): -1, (I, P, I): -1, (I, P, Q): 1,
         (P, I, I): -1, (P, I, Q): 2, (P, P, I): 1, (P, P, Q): -1,
         (P, Q, Q): -1}
    ],
    [
        {(Q, P, P): -1,
         (I, I, I): 1, (I, I, P): -1, (I, Q, I): -1, (I, Q, P): 1,
         (Q, I, I): -1, (Q, I, P): 2, (Q, Q, I): 1, (Q, Q, P): -1},
        {(Q, P, Q): 1,
         (I, Q, I): 1, (I, Q, Q): -1, (Q, Q, I): -1, (Q, Q, Q): 1},
        {(Q, I, I): 1, (Q, I, P): -2, (Q, P, I): -1, (Q, P, P): 1,
         (I, I, P): 1, (I, Q, P): -1, (Q, Q, P): 1},
        {(Q, P, I): 1, (Q, P, Q): -1,
         (I, Q, Q): 1, (Q, Q, Q): -1}
    ],
    [
        {(I, I, P): 1, (I, P, P): -1, (Q, I, P): -2, (Q, P, P): 1,
         (Q, I, I): 1, (Q, Q, I): -1, (Q, Q, P): 1,},
        {(I, P, Q): 1, (Q, P, Q): -1, (Q, Q, I): 1, (Q, Q, Q): -1},
        {(I, I, I): 1, (I, I, P): -1, (I, P, I): -1, (I, P, P): 1,
         (Q, I, I): -1, (Q, I, P): 2, (Q, P, I): 1, (Q, P, P): -1,
         (Q, Q, P): -1},
        {(I, P, I): 1, (I, P, Q): -1, (Q, P, I): -1, (Q, P, Q): 1,
         (Q, Q, Q): 1}
    ]
]
```

Listing 3: `code/algebra.py`

```
try:
    import queue
except ImportError:
    import Queue as queue
import resource
import sys

```

```
8  from algebra import I, mult, M
9
10 # description of the finite automaton generating the quotient basis of I_4
11 # it has the form {state: [(next_state, evaluated_label), ...], ...}
12 transitions = {
13     0: [(1, M[0][0]), (2, M[0][1]), (3, M[0][2]), (4, M[1][0]), (5, M[1][1]),
14         (6, M[1][2]), (7, M[2][0]), (8, M[2][1]), (9, M[2][2])],
15
16     1: [(9, M[2][2]), (8, M[2][1]), (6, M[1][2]), (5, M[1][1])],
17     2: [(4, M[1][0]), (6, M[1][2]), (7, M[2][0]), (9, M[2][2])],
18     4: [(2, M[0][1]), (3, M[0][2]), (8, M[2][1]), (9, M[2][2])],
19     5: [(1, M[0][0]), (3, M[0][2]), (7, M[2][0]), (9, M[2][2])],
20
21     8: [(15, M[0][0]), (14, M[0][2])],
22     7: [(13, M[0][1]), (14, M[0][2])],
23     6: [(12, M[0][0]), (11, M[2][0])],
24     3: [(10, M[1][0]), (11, M[2][0])],
25
26     15: [(8, M[2][1]), (9, M[2][2])],
27     12: [(6, M[1][2]), (9, M[2][2])],
28     13: [(7, M[2][0]), (9, M[2][2])],
29     10: [(3, M[0][2]), (9, M[2][2])],
30
31     14: [(11, M[2][0])],
32     11: [(14, M[0][2])],
33
34     9: [(16, M[0][0])],
35     16: [(9, M[2][2])],
36 }
37
38 def log(msg):
39     """
40     Prints the current matrix size and resource usage. Uses the global
41     variables 'rows', 'columns' and 'non_zero_entries'.
42     """
43     res = resource.getrusage(resource.RUSAGE_SELF)
44     form = "{} - rows: {} - columns: {} - entries: {} - memory: {} - time: {}"
45     print(form.format(
46         msg, len(rows), len(columns), non_zero_entries,
47         res.ru_maxrss, res.ru_utime
48     ))
49
50 max_degree = int(sys.argv[1]) # maximum degree 'm' to check
51 last_degree = 0               # used for logging
52
53 # sparse matrix \Psi_m
54 rows = {}            # maps a row index to non-zero column indices
55 columns = {}         # maps a column index to non-zero row indices
56 row_id = {}          # maps a tensor to a row index
57 non_zero_entries = 0 # used for logging
58
59 # Algorithm 1: breadth-first search
60 queue = queue.Queue()
61 queue.put((0, {(I, I, I): 1}, 0))
62
63 while not queue.empty():
64     state, monomial, degree = queue.get()
65
```

```
66      # log intermediate results when starting to process the next degree
67      if degree > last_degree:
68          log("degree %d" % last_degree)
69      last_degree = degree
70
71      column = len(columns)    # current column index
72      columns[column] = set()
73
74      for tensor, coeff in monomial.items():
75          # add a new row if the tensor has not be seen before
76          if tensor not in row_id:
77              row_id[tensor] = len(rows)
78              rows[len(rows)] = set()
79
80          row = row_id[tensor]  # current row index
81
82          # insert (row, column) into the sparse matrix
83          columns[column].add(row)
84          rows[row].add(column)
85          non_zero_entries += 1
86
87      if degree < max_degree:
88          for next_state, label in transitions[state]:
89              queue.put((next_state, mult(monomial, label), degree + 1))
90
91  log("degree %d" % max_degree)
92
93
94  # Algorithm 2: matrix elimination
95  stack = []
96  rank = 0    # lower bound on the rank of \Psi_m
97
98  for i in rows:
99      if len(rows[i]) == 1:
100         stack.append(i)
101
102 while stack:
103     i = stack.pop()
104
105     # row already completely eliminated
106     if len(rows[i]) == 0:
107         continue
108
109     j, = rows[i]
110
111     for k in columns[j]:
112         if k != i:
113             rows[k].remove(j)
114             if len(rows[k]) == 1:
115                 stack.append(k)
116
117     columns[j] = {i}
118     rank += 1
119
120 log("finished")
121
122 # print result
123 print("The rank is at least %d of %d." % (rank, len(columns)))
```

```
124 if rank == len(columns):
125     print("No such polynomial up to degree %d exists." % max_degree)
126 else:
127     print("Such a polynomial might exist.")
```

Listing 4: `code/main.py`

References

[AC75] A. Aho and M. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18:333–340, June 1975.

[Ban98] T. Banica. Symmetries of a generic coaction. *Mathematische Annalen*, 314, December 1998.

[Ban05] T. Banica. Quantum automorphism groups of homogeneous graphs. *Journal of Functional Analysis*, 224:243–280, July 2005.

[BB07] T. Banica and J. Bichon. Quantum groups acting on 4 points. *Journal für die Reine und Angewandte Mathematik*, 2009, April 2007.

[BCF20] M. Brannan, A. Chirvasitu, and A. Freslon. Topological generation and matrix models for quantum reflection groups. *Advances in Mathematics*, 363:106982, March 2020.

[Bic99] J. Bichon. Quantum automorphism groups of finite graphs. *Proceedings of the American Mathematical Society*, 131, March 1999.

[Bla06] B. Blackadar. *Operator algebras. Theory of C*-algebras and von Neumann algebras*, volume 122. Springer, January 2006.

[BN06] T. Banica and R. Nicoara. Quantum groups and hadamard matrices. *arXiv: Operator Algebras*, November 2006.

[BO08] N. P. Brown and N. Ozawa. *C*-algebras and Finite-dimensional Approximations*. Graduate studies in mathematics. American Mathematical Society, 2008.

[Brz62] J. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, 12, January 1962.

[Buc65] B. Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Ph.D. dissertation, University of Innsbruck, 1965.

[CK16] A. Cohen and J. Knopper. GBNP, computing gröbner bases of noncommutative polynomials, Version 1.0.3. `https://www.gap-system.org/Packages/gbnp.html`, March 2016.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

[Con94] A. Connes. *Noncommutative geometry*. Academic Press, January 1994.

[GAP20] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.11.0*, 2020. `https://www.gap-system.org`.

[HMU06] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., 2006.

[JW20] S. Jung and M. Weber. Models of quantum permutations. *Journal of Functional Analysis*, 279:108516, February 2020.

[Mac71] Saunders MacLane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 1971.

[Mor94] T. Mora. An introduction to commutative and noncommutative gröbner bases. *Theoretical Computer Science*, 134:131–173, November 1994.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, December 1943.

[Mur90] G. J. Murphy. *C*-Algebras and Operator Theory*. Elsevier Science, 1990.

[Phi88] N. C. Phillips. Inverse limits of c * -algebras. *Journal of Operator Theory*, 19, January 1988.

[The20] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.1)*, 2020. `https://www.sagemath.org`.

[Ufn91] V. Ufnarovskiĭ. On the use of graphs for computing a basis, growth and hilbert series of associative algebras. *Mathematics of The Ussr-sbornik*, 68:417–428, February 1991.

[Wan98] S. Wang. Quantum symmetry groups of finite spaces. *Communications in Mathematical Physics*, 195, July 1998.

[Wor87] S. Woronowicz. Compact matrix pseudogroups. *Communications in Mathematical Physics*, 111:613–665, January 1987.