Clifford Algebras Meet Tree Decompositions*

Michał Włodarczyk

University of Warsaw, Faculty of Mathematics, Informatics, and Mechanics, Warsaw, Poland
m.wlodarczyk@mimuw.edu.pl

Abstract

We introduce the Non-commutative Subset Convolution – a convolution of functions useful when working with determinant-based algorithms. In order to compute it efficiently, we take advantage of Clifford algebras, a generalization of quaternions used mainly in the quantum field theory.

We apply this tool to speed up algorithms counting subgraphs parameterized by the treewidth of a graph. We present an $O^*((2^\omega + 1)^{tw})$-time algorithm for counting Steiner trees and an $O^*((2^\omega + 2)^{tw})$-time algorithm for counting Hamiltonian cycles, both of which improve the previously known upper bounds. The result for Steiner Tree also translates into a deterministic algorithm for Feedback Vertex Set. All of these constitute the best known running times of deterministic algorithms for decision versions of these problems and they match the best obtained running times for pathwidth parameterization under assumption $\omega = 2$.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Computations on Discrete Structures, G.2.2 [Graph Theory] Graph Algorithms

Keywords and phrases fixed-parameter tractability, treewidth, Clifford algebra, algebra isomorphism

Digital Object Identifier 10.4230/LIPIcs.IPEC.2016.29

1 Introduction

The concept of *treewidth* has been introduced by Robertson and Seymour in their work on graph minors [13]. The treewidth of a graph is the smallest possible width of its *tree decomposition*, i.e. a tree-like representation of the graph. Its importance follows from the fact that many NP-hard graph problems become solvable on trees with a simple dynamical programming. A similar idea of *pathwidth* captures the width of a graph in case we would like to have a *path decomposition*. Formal definitions can be found in Section 2.2.

A bound on the graph's treewidth allows to design efficient algorithms using *fixed-parameter tractability*. An algorithm is called fixed-parameter tractable (FPT) if it works in time complexity $f(k)n^{O(1)}$ where k is a parameter describing hardness of the instance and f is a computable function. We also use notation $O^*(f(k))$ that suppresses polynomial factors with respect to the input size. Problems studied in this work are parameterized by the graph's pathwidth or treewidth. To distinguish these cases we denote the parameter respectively pw or tw.

It is natural to look for a function f that is growing relatively slow. For problems with a local structure, like Vertex Cover or Dominating Set, there are simple FPT algorithms with single exponential running time. They usually store c^{tw} states for each node of the

* This work is partially supported by Foundation for Polish Science grant HOMING PLUS/2012-6/2 and a project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651).

decomposition and take advantage of the Fast Subset Convolution [2] to perform the *join* operation in time $O^*(c^{tw})$. As a result, time complexities for pathwidth and treewidth parameterizations remain the same. The Fast Subset Convolution turned out helpful in many other problems, e.g. Chromatic Number, and enriched the basic toolbox used for exponential and parameterized algorithms.

Problems with connectivity conditions, like Steiner Tree or Hamiltonian Cycle, were conjectured to require time $2^{\Omega(tw \log tw)}$ until the breakthrough work of Cygan et al. [8]. They introduced the randomized technique Cut & Count working in single exponential time. The obtained running times were respectively $O^*(3^{tw})$ and $O^*(4^{tw})$. Afterwards, a faster randomized algorithm for Hamiltonian Cycle parameterized by the pathwidth was presented with running time $O^*((2+\sqrt{2})^{pw})$ [7]. This upper bound as well as $O^*(3^{pw})$ for Steiner Tree are tight modulo subexponential factors under the assumption of *Strong Exponential Time Hypothesis* [7, 8].

The question about the existence of single exponential deterministic methods was answered positively by Bodlaender et al. [4]. What is more, presented algorithms count the number of Steiner trees or Hamiltonian cycles in a graph. However, in contrast to the Cut & Count technique, a large gap emerged between the running times for pathwidth and treewidth parameterizations – the running times were respectively $O^*(5^{pw})$, $O^*(10^{tw})$ for Steiner Tree and $O^*(6^{pw})$, $O^*(15^{tw})$ for Hamiltonian Cycle. This could be explained by a lack of efficient algorithms to perform the *join* operation, necessary only for tree decompositions. Some efforts have been made to reduce this gap and the deterministic running time for Steiner Tree has been improved to $O^*((2^{\omega-1}\cdot 3+1)^{tw})$ [9].

1.1 Our contribution

The main contribution of this work is creating a link between Clifford algebras, objects not being used in algorithmics to the best of our knowledge, and fixed-parameter tractability. As the natural dynamic programming approach on tree decompositions uses the Fast Subset Convolution (FSC) to perform efficiently the *join* operation, there was no such a tool for algorithms based on the determinant approach.

Our first observation is that the FSC technique can be regarded as an isomorphism theorem for some associative algebras. To put it briefly, a Fourier-like transform is being performed in the FSC to bring computations to a simpler algebra. Interestingly, this kind of transform is just a special case of the Artin-Wedderburn theorem [1], which seemingly is not widely reported in computer science articles. The theorem provides a classification of a large class of associative algebras, not necessarily commutative (more in Appendix A). We use this theory to introduce the Non-commutative Subset Convolution (NSC) and speed up multiplication operations in an algebra induced by the *join* operation in determinant-based dynamic programming on tree decomposition. An important building block is a fast Fourier-like transform for a closely related algebra [11]. We hope our work will encourage researchers to investigate further algorithmic applications of the Artin-Wedderburn theorem.

1.2 Our results

We apply our algebraic technique to determinant approach introduced by Bodlaender et al. [4]. For path decomposition, they gave an $O^*(5^{pw})$-time algorithm for counting Steiner trees and an $O^*(6^{pw})$-time algorithm for counting Hamiltonian cycles. The running times for tree decomposition were respectively $O^*(10^{tw})$ and $O^*(15^{tw})$. These gaps can be explained by

the appearance of the *join* operation in tree decompositions which could not be handled efficiently so far.

By performing NSC in time complexity $O^*(2^{\frac{\omega n}{2}})$ we partially solve an open problem about the *different convolution* from [6]. Our further results may be considered similar to those closing the gap between time complexities for pathwidth and treewidth parameterizations for Dominating Set by switching between representations of states in dynamic programming [14]. We improve the running times to $O^*((2^\omega+1)^{tw})$ for counting Steiner trees and $O^*((2^\omega+2)^{tw})$ for counting Hamiltonian cycles, where ω denotes the matrix multiplication exponent (currently it is established that $\omega < 2.373$ [15]). These are not only the fastest known algorithms for counting these objects, but also the fastest known deterministic algorithms for the decision versions of these problems. The deterministic algorithm for Steiner Tree can be translated into a deterministic algorithm for Feedback Vertex Set [4] so our technique provides an improvement also in this case.

Observe that the running times for pathwidth and treewidth parameterizations match under the assumption $\omega = 2$. Though we do not hope for settling the actual value of ω, this indicates there is no further space for significant improvement unless pure combinatorial algorithms (i.e. not based on matrix multiplication) are invented or the running time for pathwidth parameterization is improved.

1.3 Organization of the paper

Section 3 provides a brief introduction to Clifford algebras. The bigger picture of the employed algebraic theory can be found in Appendix A. In Section 4 we define the NSC and design efficient algorithms for variants of the NSC employing the algebraic tools. Sections 5 and 6 present how to apply the NSC in counting algorithms for Steiner Tree and Hamiltonian Cycle. They contain main ideas improving the running times, however in order to understand the algorithms completely one should start from Section 4 (*Determinant approach*) in [4]. The algorithm for Hamiltonian Cycle is definitely more complicated and its details, formulated as two isomorphism theorems, are placed in Appendix C.

2 Preliminaries

We will start with notation conventions.

1. $A \uplus B = C$ stands for $(A \cup B = C) \wedge (A \cap B = \emptyset)$.
2. $A \triangle B = (A \setminus B) \cup (B \setminus A)$.
3. $[\alpha]$ equals 1 if condition α holds and 0 otherwise.
4. For permutation f of a linearly ordered set U
$$\operatorname{sgn}(f) = (-1)^{|\{(a,b) \in U \times U \wedge a<b \wedge f(a)>f(b)\}|}.$$
5. For A, B being subsets of a linearly ordered set
$$I_{A,B} = (-1)^{|\{(a,b) \in A \times B \wedge a>b\}|}. \tag{1}$$

Let us note two simple properties of I.

▶ **Claim 1.** *For disjoint A, B*
$$I_{A,B} I_{B,A} = (-1)^{|A||B|}.$$

▶ **Claim 2.** *For $A \cap B = \emptyset$ and $C \cap D = \emptyset$*
$$I_{A \cup B, C \cup D} = I_{A,C} I_{A,D} I_{B,C} I_{B,D}.$$

2.1 Fast Subset Convolution

Let us consider a universe U of size n and functions $f, g : 2^U \longrightarrow \mathbb{Z}$.

▶ **Definition 3.** The Möbius transform of f is function $\hat{f}$ defined as

$$\hat{f}(X) = \sum_{A \subseteq X} f(A).$$

▶ **Definition 4.** Let $f * g$ denote a *subset convolution* of f, g defined as

$$(f * g)(X) = \sum_{A \uplus B = X} f(A)g(B).$$

▶ **Theorem 5** (Björklund et al. [2])**.** *The Möbius transform, its inverse, and the subset convolution can be computed in time* $O^*(2^n)$.

2.2 Pathwidth and treewidth

▶ **Definition 6.** A *tree (path) decomposition* of a graph G is a tree $\mathbb{T}$ (path $\mathbb{P}$) in which each node x is assigned a bag $B_x \subseteq V(G)$ such that

1. for every edge $uv \in E(G)$ there is a bag B_x containing u and v,
2. for every vertex v the set $\{x \mid v \in B_x\}$ forms a non-empty subtree (subpath) in the decomposition.

The width of the decomposition is defined as $\max_x |B_x| - 1$ and the treewidth (pathwidth) of G is a minimum width over all possible tree (path) decompositions.

If a graph admits a tree decomposition of width t then it can be found in time $n \cdot 2^{O(t^3)}$ [3] and a decomposition of width at most $4t + 1$ can be constructed in time $poly(n) \cdot 2^{O(t)}$ [10]. We will assume that a decomposition of the appropriate type and width is given as a part of the input.

▶ **Definition 7.** A *nice tree (path) decomposition* is a decomposition with one special node r called the root and in which each bag is one of the following types:

1. *Leaf bag:* a leaf x with $B_x = \emptyset$,
2. *Introduce vertex v bag:* a node x having one child y for which $B_x = B_y \uplus \{v\}$,
3. *Forget vertex v bag:* a node x having one child y for which $B_y = B_x \uplus \{v\}$,
4. *Introduce edge uv bag:* a node x having one child y for which $u, v \in B_x = B_y$,
5. *Join bag:* **(only in tree decomposition)** a node x having two children y, z with condition $B_x = B_y = B_z$.

We require that every edge from $E(G)$ is introduced exactly once and B_r is an empty bag. For each x we define V_x and E_x to be sets of respectively vertices and edges introduced in the subtree of the decomposition rooted at x.

Given a tree (path) decomposition we can find a nice decomposition in time $n \cdot tw^{O(1)}$ [8, 10] and we will work only on these. When analyzing running time of algorithms working on tree decompositions we will estimate the bag sizes from the above assuming $|B_x| = tw$.

2.3 Problems definitions

Steiner Tree
Input: graph G, set of terminals $K \subseteq V(G)$, integer k
Decide: whether there is a subtree of G with at most k edges connecting all vertices from K

Hamiltonian Cycle
Input: graph G
Decide: whether there is a cycle going through every vertex of G exactly once

Feedback Vertex Set
Input: graph G, integer k
Decide: whether there is a set $Y \subseteq V$ of size at most k such that every cycle in G contains a vertex from Y

In the counting variants of problems we ask for a number of structures satisfying the given conditions. This setting is at least as hard as the decision variant.

3 Clifford algebras

Some terms used in this section originate from advanced algebra. For better understanding we suggest reading Appendix A.

▶ **Definition 8.** The Clifford algebra $Cl_{p,q}(R)$ is a 2^{p+q}-dimensional associative algebra over a ring R. It is generated by $x_1, x_2 \ldots, x_{p+q}$.

These are rules of multiplication of generators:

1. e is a neutral element of multiplication,
2. $x_i^2 = e$ for $i = 1, 2, \ldots, p$,
3. $x_i^2 = -e$ for $i = p+1, p+2, \ldots, p+q$,
4. $x_i x_j = -x_j x_i$ if $i \neq j$.

All 2^{p+q} products of ordered sets of generators form a basis of $Cl_{p,q}(R)$ (e is treated as a product of an empty set). We provide a standard addition and we extend multiplication for all elements in an associative way.

We will be mainly interested only in $Cl_{n,0}(\mathbb{Z})$ [1] and its natural embedding into $Cl_{n,0}(\mathbb{R})$. As $q = 0$, we can neglect condition 3 when analyzing these algebras.

For $A = \{a_1, a_2, \ldots, a_k\} \subseteq [1 \ldots n]$ where $a_1 < a_2 < \cdots < a_k$ let $x_A = x_{a_1} x_{a_2} \cdots x_{a_k}$. Each element of $Cl_{n,0}(\mathbb{R})$ can be represented as $\sum_{A \subseteq [1 \ldots n]} a_A x_A$, where a_A are real coefficients. Using condition 4 we can deduce a general formula for multiplication in $Cl_{n,0}(\mathbb{R})$:

$$\left(\sum_{A \subseteq [1 \ldots n]} a_A x_A \right) \left(\sum_{B \subseteq [1 \ldots n]} b_B x_B \right) = \sum_{C \subseteq [1 \ldots n]} \left(\sum_{A \triangle B = C} a_A b_B I_{A,B} \right) x_C \tag{2}$$

where the meaning of $I_{A,B}$ is explained in (1).

As a Clifford algebra over $\mathbb{R}$ is semisimple, it is isomorphic to a product of matrix algebras by the Artin-Wedderburn theorem (see Theorem 31). However, it is more convenient to first embed $Cl_{n,0}(\mathbb{R})$ in a different Clifford algebra that is isomorphic to a single matrix algebra. As a result, we obtain a monomorphism $\phi : Cl_{n,0}(\mathbb{R}) \longrightarrow \mathbb{M}_{2^m}(\mathbb{R})$ (see Definition 28) where $m = \frac{n}{2} + O(1)$ and the following diagram commutes ($*$ stands for multiplication).

$$\begin{array}{ccc} Cl_{n,0}(\mathbb{R}) & \xrightarrow{\phi} & \mathbb{M}_{2^m}(\mathbb{R}) \\ \downarrow * & & \downarrow * \\ Cl_{n,0}(\mathbb{R}) & \xrightarrow{\phi} & \mathbb{M}_{2^m}(\mathbb{R}) \end{array} \tag{3}$$

[1] Clifford algebras with $q = 0$ appear also in geometric literature as *exterior algebras*.

Thus, we can perform multiplication in the structure that is more convenient for us. For $a, b \in Cl_{n,0}(\mathbb{Z})$ we can treat them as elements of $Cl_{n,0}(\mathbb{R})$, find matrices $\phi(a)$ and $\phi(b)$, multiply them efficiently, and then revert the ϕ transform. The result always exists and belongs to $Cl_{n,0}(\mathbb{Z})$ because $Cl_{n,0}(\mathbb{Z})$ is closed under multiplication. The monomorphism $\phi : Cl_{n,0}(\mathbb{R}) \longrightarrow \mathbb{M}_{2^m}(\mathbb{R})$ can be performed and reverted (within the image) in $O^*(2^n)$ time [11]. However, the construction in [11] is analyzed in the infinite precision model. For the sake of completeness, we revisit this construction and prove the following theorem in Appendix B.

▶ **Theorem 9.** *The multiplication in $Cl_{n,0}(\mathbb{Z})$, with coefficients having poly(n) number of bits, can be performed in time $O^*(2^{\frac{\omega n}{2}})$.*

In order to unify the notation we will represent each element of $Cl_{n,0}(\mathbb{Z})$, that is $\sum_{A\subseteq[1\ldots n]} a_A x_A$, as a function $f : 2^{[1\ldots n]} \longrightarrow \mathbb{Z}$, $f(A) = a_A$. We introduce $\diamond_S$ convolution as an equivalence of multiplication in $Cl_{n,0}(\mathbb{Z})$. The equation (2) can be now rewritten in a more compact form

$$(f \diamond_S g)(X) = \sum_{A \triangle B = X} f(A)g(B)I_{A,B}. \tag{4}$$

4 Non-commutative Subset Convolution

We consider a linearly ordered universe U of size n and functions $f, g : 2^U \longrightarrow \mathbb{Z}$.

▶ **Definition 10.** Let $f \diamond g$ denote *Non-commutative Subset Convolution* (NSC) of functions f, g defined as

$$(f \diamond g)(X) = \sum_{A \uplus B = X} f(A)g(B)I_{A,B}.$$

▶ **Theorem 11.** *NSC on an n-element universe can be performed in time $O^*(2^{\frac{\omega n}{2}})$.*

Proof. Observe that condition $A \uplus B = X$ is equivalent to $A \triangle B = X \wedge |A| + |B| = |X|$ so

$$(f \diamond g)(X) = \sum_{\substack{i+j=|X| \\ i,j\geq 0}} \sum_{A\triangle B = X} f(A)\Big[|A| = i\Big]g(B)\Big[|B| = j\Big]I_{A,B}.$$

Alternatively, we can write

$$(f \diamond g)(X) = \sum_{\substack{i+j=|X| \\ i,j\geq 0}} (f_i \diamond_S g_j)(X),$$

where $f_i(X) = f(X)\Big[|X| = i\Big]$ and likewise for g. The $\diamond_S$ convolution, introduced in (4), is equivalent to multiplication in $Cl_{n,0}(\mathbb{R})$. This means we reduced NSC to $O(n^2)$ multiplications in $Cl_{n,0}(\mathbb{R})$ which could be performed in time $O(2^{\frac{\omega n}{2}})$ according to Theorem 9. ◀

▶ **Observation 12.** *The technique of paying polynomial factor for grouping the sizes of sets will turn useful in further proofs. We will call it size-grouping.*

In our applications we will need to compute a slightly more complex convolution.

▶ **Definition 13.** When f, g are of type $2^U \times 2^U \longrightarrow \mathbb{Z}$ we can define $f \diamond_2 g$ (NSC2) as follows

$$(f \diamond_2 g)(X,Y) = \sum_{\substack{X_1 \uplus X_2 = X \\ Y_1 \uplus Y_2 = Y}} f(X_1,Y_1) g(X_2,Y_2) I_{X_1,X_2} I_{Y_1,Y_2}.$$

▶ **Theorem 14.** *NSC2 on an n-element universe can be performed in time $O^*(2^{\omega n})$.*

Proof. Let us introduce a new universe $U' = U_X \cup U_Y$ of size $2n$ consisting of two copies of U with an order so each element of U_Y is greater than any element of U_X. To underline that $X \subseteq U_X, Y \subseteq U_X$ we will use $\uplus$ notation when summing subsets of U_X and U_Y. In order to reduce NSC2 to NSC on the universe U' we need to replace factor $I_{X_1,X_2} I_{Y_1,Y_2}$ with $I_{X_1 \uplus Y_1, X_2 \uplus Y_2}$. The latter term can be expressed as $I_{X_1,X_2} I_{Y_1,Y_2} I_{X_1,Y_2} I_{Y_1,X_2}$ due to Claim 2. As all elements from $X_i \subseteq U_X$ compare less to elements from $Y_i \subseteq U_Y$ then $I_{X_1,Y_2} = 1$ and I_{Y_1,X_2} depends only on the sizes of Y_1 and X_2. To summarize,

$$I_{X_1,X_2} I_{Y_1,Y_2} = I_{X_1 \uplus Y_1, X_2 \uplus Y_2} (-1)^{|Y_1||X_2|}.$$

To deal with factor $(-1)^{|Y_1||X_2|}$ we have to split the convolution into 4 parts for different parities of $|Y_1|$ and $|X_2|$. We define functions $f', f'_0, f'_1, g', g'_0, g'_1 : 2^{U'} \longrightarrow \mathbb{Z}$ as

$$\begin{aligned}
f'(X \uplus Y) &= f(X,Y), \\
f'_0(X \uplus Y) &= f(X,Y)\Big[\,|Y| \equiv 0 \bmod 2\Big], \\
f'_1(X \uplus Y) &= f(X,Y)\Big[\,|Y| \equiv 1 \bmod 2\Big], \\
g'(X \uplus Y) &= g(X,Y), \\
g'_0(X \uplus Y) &= g(X,Y)\Big[\,|X| \equiv 0 \bmod 2\Big], \\
g'_1(X \uplus Y) &= g(X,Y)\Big[\,|X| \equiv 1 \bmod 2\Big].
\end{aligned}$$

Now we can reduce NSC2 to 4 simpler convolutions.

$$(f \diamond_2 g)(X,Y) = \sum_{\substack{X_1 \uplus X_2 = X \\ Y_1 \uplus Y_2 = Y}} f'(X_1 \uplus Y_1) g'(X_2 \uplus Y_2) I_{X_1 \uplus Y_1, Y_2 \uplus X_2} (-1)^{|Y_1||X_2|} =$$
$$= (f'_0 \diamond g'_0)(X \uplus Y) + (f'_0 \diamond g'_1)(X \uplus Y) + (f'_1 \diamond g'_0)(X \uplus Y) - (f'_1 \diamond g'_1)(X \uplus Y)$$

We have shown that computing NSC2 is as easy as NSC on a universe two times larger. Using Theorem 11 directly gives us the desired complexity. ◀

5 Counting Steiner trees

We will revisit the theorem stated in the aforementioned work.

▶ **Theorem 15** (Bodlaender et al. [4])**.** *There exist algorithms that given a graph G count the number of Steiner trees of size i for each $1 \leq i \leq n-1$ in $O^*(5^{pw})$ time if a path decomposition of width pw is given, and in $O^*(10^{tw})$ time if a tree decomposition of width tw is given.*

Both algorithms use dynamic programming over tree or path decompositions. We introduce some decomposition-based order on V and fix vertex v_1. Let $A = (a_{v,e})_{v \in V, e \in E}$ be an incidence matrix, i.e. for $e = uv$, $u < v$ we have $a_{u,e} = 1$, $a_{v,e} = -1$ and $a_{w,e} = 0$ for

any other vertex w. For each node x of the decomposition we define a function A_x with arguments $0 \leq i \leq n-1$, $s_Y, s_1, s_2 \in \{0,1\}^{B_x}$. The idea is to express the number of Steiner trees with exactly i edges as $A_r(i+1, \emptyset, \emptyset, \emptyset)$.

$$A_x(i, s_Y, s_1, s_2) =$$
$$= \sum_{\substack{Y \subseteq V_x \\ |Y|=i \\ (K \cap V_x) \subseteq Y \\ Y \cap B_x = s_Y^{-1}(1)}} \sum_{X \subseteq E(Y,Y) \cap E_x} \sum_{\substack{f_1 : X \overset{1-1}{\to} Y \setminus \{v_1\} \setminus s_1^{-1}(0) \\ f_2 : X \overset{1-1}{\to} Y \setminus \{v_1\} \setminus s_2^{-1}(0)}} \operatorname{sgn}(f_1)\operatorname{sgn}(f_2) \prod_{e \in X} a_{f_1(e),e} a_{f_2(e),e} \tag{5}$$

As observed in [4] condition $s_Y(v) = 0$ implies that either $s_1(v) = s_2(v) = 0$ or $A_x(i, s_Y, s_1, s_2) = 0$. This means there are at most $n5^{tw}$ triples for which A_x returns a nonzero value.

If a node x has a child y and is of type *introduce vertex*, *introduce edge*, or *forget vertex*, then the function A_x can be computed from A_y in linear time with respect to the number of non-trivial states. Saying this is just a reformulation of Theorem 15 for path decompositions. The only thing that is more difficult for tree decompositions is that they include also *join* nodes having two children each. Here is the recursive formula[2] for A_x for a *join* node x having children y, z.

$$A_x(i, s_Y, s_1, s_2) = \sum_{\substack{i_y + i_z = i + |s_Y^{-1}(1)| \\ s_{1,y} + s_{1,z} = s_1 \\ s_{2,y} + s_{2,z} = s_2}} \begin{matrix} A_y(i_y, s_Y, s_{1,y}, s_{2,y}) A_z(i_z, s_Y, s_{1,z}, s_{2,z}) \\ I_{s_{1,y}^{-1}(1), s_{1,z}^{-1}(1)} I_{s_{2,y}^{-1}(1), s_{2,z}^{-1}(1)} \end{matrix} \tag{6}$$

The next lemma, however not stated explicitly in the discussed work, follows from the proof of Theorem 15 (Theorem 4.4 in [4]).

▶ **Lemma 16.** *Assume there is an algorithm computing all nonzero values of A_x given by* (6) *with running time $f(tw)$. Then the number of Steiner trees of size i in a graph G can be counted in $O^*(\max(f(tw), 5^{tw}))$ time if a tree decomposition of width tw is given.*

We will change notation for our convenience. Each function s_i will be matched with a set $s_i^{-1}(1)$. Let us replace functions A_x, A_y, A_z with h_i, f_i, g_i having first argument fixed and operating on triples of sets. In this setting, the convolution can we written as

$$h_i(A, B, C) = \sum_{\substack{i_y + i_z = i + |A| \\ B_y \uplus B_z = B \\ C_y \uplus C_z = C}} f_{i_y}(A, B_y, C_y) g_{i_z}(A, B_z, C_z) I_{B_y, B_z} I_{C_y, C_z}. \tag{7}$$

Observe that size-grouping allows us to sacrifice a polynomial factor and neglect the restrictions for i, i_y, i_z. Hence, we can work with a simpler formula

$$h(A, B, C) = \sum_{\substack{B_y \uplus B_z = B \\ C_y \uplus C_z = C}} f(A, B_y, C_y) g(A, B_z, C_z) I_{B_y, B_z} I_{C_y, C_z}. \tag{8}$$

The only triples $(s_Y(v), s_1(v), s_2(v))$ allowed for each vertex v are $(0,0,0)$, $(1,0,0)$, $(1,0,1)$, $(1,1,0)$, $(1,1,1)$. In terms of set notation we can say that if $f(A,B,C) \neq 0$ then $B \cup C \subseteq A$. Let $f_A : 2^A \times 2^A \longrightarrow \mathbb{Z}$ be f with the first set fixed, i.e. $f_A(B,C) = f(A,B,C)$.

[2] As confirmed by the authors [5], the formula in [4] for the join node is missing the first argument to the A_x function tracking the number of vertices of a Steiner tree, hence we present a corrected version of this formula.

▶ **Lemma 17.** *For fixed A all values $h(A,B,C)$ can be computed in time $O^*(2^{\omega|A|})$.*

Proof. We want to compute

$$h_A(B,C) = \sum_{\substack{B_y \uplus B_z = B \\ C_y \uplus C_z = C}} f_A(B_y,C_y) g_A(B_z,C_z) I_{B_y,B_z} I_{C_y,C_z} = (f_A \diamond_2 g_A)(B,C),$$

what can be done in time $O^*(2^{\omega|A|})$ according to Theorem 14. ◀

▶ **Lemma 18.** *The convolution (7) can be performed in time $O^*((2^\omega+1)^{tw})$.*

Proof. We use size-grouping to reduce the problem to computing (8). Then we iterate through all possible sets A and take advantage of Lemma 17. The total number of operations (modulo polynomial factor) is bounded by

$$\sum_{A \subseteq U} 2^{\omega|A|} = \sum_{k=0}^{tw} \binom{tw}{k} 2^{\omega k} = (2^\omega+1)^{tw}.$$ ◀

Keeping in mind that (6) and (7) are equivalent and combining Lemmas 16, 18, we obtain the following result.

▶ **Theorem 19.** *The number of Steiner trees of size i in a graph G can be computed in $O^*((2^\omega+1)^{tw})$ time if a tree decomposition of width tw is given.*

▶ Remark. The space complexity of the algorithm is $O^*(5^{tw})$.

Solving the decision version of Feedback Vertex Set can be reduced to the Maximum Induced Forest problem [4]. As observed in [4] the *join* operation for Maximum Induced Forest is analogous to (6).

▶ **Corollary 20.** *The existence of a feedback vertex set of size at most i in a graph G can be determined in $O^*((2^\omega+1)^{tw})$ time if a tree decomposition of width tw is given.*

6 Counting Hamiltonian cycles

Likewise in the previous section, we will start with a previously known theorem.

▶ **Theorem 21** (Bodlaender et al. [4])**.** *There exist algorithms that given a graph G count the number of Hamiltonian cycles in $O^*(6^{pw})$ time if a path decomposition of width pw is given, and in $O^*(15^{tw})$ time if a tree decomposition of width tw is given.*

For each node x of the decomposition a function A_x is defined with arguments $s_1, s_2 \in \{0,1\}^{B_x}$ and $s_{deg} \in \{0,1,2\}^{B_x}$. The idea and notation is analogous to (5). The number of Hamiltonian cycles can be expressed as $A_r(\emptyset,\emptyset,\emptyset)/n$.

$$A_x(s_{deg}, s_1, s_2) =$$
$$= \sum_{\substack{X \subseteq E_x \\ \forall_{v \in (V_x \setminus B_x)} deg_X(v)=2 \\ \forall_{v \in B_x} deg_X(v)=s_{deg}(v)}} \sum_{S \subseteq X} \sum_{\substack{f_1 : S^{1-1}_{\to} V_x \setminus \{v_1\} \setminus s_1^{-1}(0) \\ f_2 : S^{1-1}_{\to} V_x \setminus \{v_1\} \setminus s_2^{-1}(0)}} \operatorname{sgn}(f_1)\operatorname{sgn}(f_2) \prod_{e \in S} a_{f_1(e),e} a_{f_2(e),e} \qquad (9)$$

As observed in [4] we can restrict ourselves only to some subspace of states. When $s_Y(v) = 0$ then all non-zero summands in the (9) satisfy $s_1(v) = s_2(v) = 0$. When $s_Y(v) = 2$ then we can neglect all summands except for those satisfying $s_1(v) = s_2(v) = 1$.

This time there are at most 6^{tw} triples for which A_x returns a nonzero value. We again argue that *introduce vertex*, *introduce edge*, and *forget vertex* nodes can be handled the same way as for the path decomposition and the only bottleneck is formed by *join* nodes. We present a formula for A_x if x is a *join* node with children y, z.

$$A_x(s_{deg}, s_1, s_2) = \sum_{\substack{s_{deg,y}+s_{deg,z}=s_{deg} \\ s_{1,y}+s_{1,z}=s_1 \\ s_{2,y}+s_{2,z}=s_2}} A_y(s_{deg,y}, s_{1,y}, s_{2,y}) A_z(s_{deg,z}, s_{1,z}, s_{2,z}) \, I_{s_{1,y}^{-1}(1), s_{1,z}^{-1}(1)} I_{s_{2,y}^{-1}(1), s_{2,z}^{-1}(1)} \tag{10}$$

Analogously to the algorithm for Steiner Tree, we formulate our claim as a lemma following from the proof of Theorem 21 (Theorem 4.3 in [4]).

▶ **Lemma 22.** *Assume there is an algorithm computing all nonzero values of A_x given by* (10) *with running time $f(tw)$. Then the number of Hamiltonian cycles in a graph G can be counted in $O^*(\max(f(tw), 6^{tw}))$ time if a tree decomposition of width tw is given.*

The only allowed triples of $(s_{deg}(v), s_1(v), s_2(v))$ for each vertex v are $(0,0,0)$, $(1,0,0)$, $(1,0,1)$, $(1,1,0)$, $(1,1,1)$, $(2,1,1)$.

▶ **Lemma 23.** *Assume the equation* (10) *holds. Then it remains true after the following translation of the set of allowed triples $(s_{deg}(v), s_1(v), s_2(v))$.*

$0,0,0 \longrightarrow 0,0,0$
$1,0,0 \longrightarrow 1,0,0$
$1,0,1 \longrightarrow 1,0,1$
$1,1,0 \longrightarrow 0,1,0$
$1,1,1 \longrightarrow 0,1,1$
$2,1,1 \longrightarrow 1,1,1$

Proof. The $I_{\cdot,\cdot}$ factors do not change as we do not modify the coordinates given by functions s_1, s_2. Triples that match in (10) translate into matching triples as the transformation keeps their additive structure. This fact can be seen on the tables below.

	000	100	101	110	111	211
000	000	100	101	110	111	211
100	100	X	X	X	211	X
101	101	X	X	211	X	X
110	110	X	211	X	X	X
111	111	211	X	X	X	X
211	211	X	X	X	X	X

	000	100	101	010	011	111
000	000	100	101	010	011	111
100	100	X	X	X	111	X
101	101	X	X	111	X	X
010	010	X	111	X	X	X
011	011	111	X	X	X	X
111	111	X	X	X	X	X

◀

Therefore we can treat s_{deg} functions as binary ones. We start with unifying the notation binding functions s_i with sets $s_i^{-1}(1)$. Let us replace functions A_x, A_y, A_z with their equivalences h, f, g operating on triples of sets. In this setting, the convolution looks as follows.

$$h(A,B,C) = \sum_{\substack{A_1 \uplus A_2 = A \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} f(A_1, B_1, C_1) g(A_2, B_2, C_2) I_{B_1,B_2} I_{C_1,C_2} \tag{11}$$

Performing convolution (11) within the space of allowed triples is noticeably more complicated than computations in Section 5. Therefore the proof of the following lemma is placed in Appendix C.

▶ **Lemma 24.** *The convolution* (11) *can be computed in time* $O^*((2^\omega + 2)^{tw})$.

This result, together with Lemmas 22 and 23, leads to the main theorem of this section.

▶ **Theorem 25.** *The number of Hamiltonian cycles in a graph* G *can be computed in* $O^*((2^\omega + 2)^{tw})$ *time if a tree decomposition of width* tw *is given.*

▶ Remark. The space complexity of the algorithm is $O^*(6^{tw})$.

7 Conclusions

We have presented the Non-commutative Subset Convolution, a new algebraic tool in algorithmics based on the theory of Clifford algebras. This allowed us to construct faster deterministic algorithms for Steiner Tree, Feedback Vertex Set, and Hamiltonian Cycle, parameterized by the treewidth. As the determinant-based approach applies to all problems solvable by the Cut & Count technique [4, 8], the NSC can improve running times for a larger class of problems.

The first open question is whether the gap between time complexities for the decision and counting versions of these problems could be closed. Or maybe one can prove this gap inevitable under a well-established assumption, e.g. SETH?

The second question asked is if it is possible to prove a generic theorem so the lemmas like 18 or 24 would follow from it easily. It might be possible to characterize convolution algebras that are semisimple and algorithmically construct isomorphisms with their canonical forms described by the Artin-Wedderburn theorem.

The last question is what other applications of Clifford algebras and Artin-Wedderburn theorem can be found in algorithmics.

Acknowledgements. I would like to thank Marek Cygan for pointing out the bottleneck of the previously known algorithms and for the support during writing this paper. I would also like to thank Paul Leopardi for helping me understand the fast Fourier-like transform for Clifford algebras.

References

1 John A. Beachy. *Introductory lectures on rings and modules*, volume 47. Cambridge University Press, 1999.

2 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: Fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, STOC'07, pages 67–74, New York, NY, USA, 2007. ACM. doi:10.1145/1250790.1250801.

3 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

4 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243(C):86–111, August 2015. doi:10.1016/j.ic.2014.12.008.

5 Marek Cygan. Private communication, 2016.

6 Marek Cygan, Fedor Fomin, Bart MP Jansen, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Open problems for fpt school 2014. URL: `http://fptschool.mimuw.edu.pl/opl.pdf`.

7 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 301–310. ACM, 2013.

8 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 150–159. IEEE, 2011.

9 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative sets of product families. In *Algorithms – ESA 2014*, pages 443–454. Springer, 2014.

10 Ton Kloks. *Treewidth: computations and approximations*, volume 842. Springer Science & Business Media, 1994.

11 Paul Leopardi. A generalized FFT for Clifford algebras. *Bulletin of the Belgian Mathematical Society*, 11(5):663–688, 03 2005.

12 David K. Maslen and Daniel N. Rockmore. Generalized ffts – a survey of some recent results. In *Groups and Computation II*, volume 28, pages 183–287. American Mathematical Soc., 1997.

13 Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

14 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. *Dynamic Programming on Tree Decompositions Using Generalised Fast Subset Convolution*, pages 566–577. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. `doi:10.1007/978-3-642-04128-0_51`.

15 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 887–898. ACM, 2012.

A Associative algebras

This section is not crucial to understanding the paper but it provides a bigger picture of the applied theory. We assume that readers are familiar with basic algebraic structures like rings or fields. More detailed introduction can be found, e.g. in [1].

▶ **Definition 26.** A linear space A over a field K (or, more generally, a module over a ring K) is called an *associative algebra* if it admits a multiplication operator $A \times A \to A$ satisfying the following conditions:

1. $\forall_{a,b,c \in A}\ a(bc) = (ab)c$,
2. $\forall_{a,b,c \in A}\ a(b+c) = ab + ac,\ (b+c)a = ba + ca$,
3. $\forall_{a,b \in A, k \in K}\ k(ab) = (ka)b = a(kb)$.

A set $W \subseteq A$ is called a *generating set* if every element of A can be obtained from W by addition and multiplication. The elements of W are called *generators*. It is easy to see that multiplication defined on a generating set extends in an unambiguous way to the whole algebra. We will often abbreviate the term *associative* as we will study only such algebras.

▶ **Definition 27.** The product of algebras $A_1, A_2, \ldots, A_m$ is an algebra $A_1 \otimes A_2 \otimes \cdots \otimes A_m$ with multiplication performed independently on each coordinate.

▶ **Definition 28.** For algebras A, B over a ring K, function $\phi : A \to B$ is called a *homomorphism of algebras* if it satisfy the following conditions:

1. $\forall_{a,b\in A}\ \phi(a+b)=\phi(a)+\phi(b)$,
2. $\forall_{a,b\in A}\ \phi(ab)=\phi(a)\phi(b)$,
3. $\forall_{a\in A,k\in K}\ \phi(ka)=k\phi(a)$.

If ϕ is reversible within its image then we call it a *monomorphism* and if additionally $\phi(A)=B$ then we call ϕ an *isomorphism*

Monomorphisms of algebras turn out extremely useful when multiplication in algebra B is simpler than multiplication in A, because we can compute ab as $\phi^{-1}\big(\phi(a)\phi(b)\big)$. This observation is used in Theorem 9 and Lemmas 24, 32. For a better intuition, we depict the various ways of performing multiplication on diagrams (3), (14).

▶ **Definition 29.** A subset M of algebra A is called a *simple left module* if

1. $\forall_{a\in A,b\in M}\ ab\in M$,
2. $\forall_{b,c\in M}\ b+c\in M$,

and the only proper subset of M with these properties is $\{0\}$.

The next definition is necessary to exclude some cases of obscure algebras.

▶ **Definition 30.** An algebra A is called *semisimple* if there is no non-zero element a so for every simple left module $M\subseteq A$ the set $aM=\{ab\,|\,b\in M\}$ is $\{0\}$.

The theorem below was proven in full generality for algebras over arbitrary rings but we will formulate its simpler version for fields.

▶ **Theorem 31** (Artin-Wedderburn [1])**.** *Every finite-dimensional associative semisimple algebra A over a field K is isomorphic to a product of matrix algebras*

$$A\cong M_{n_1}(K_1)\otimes M_{n_2}(K_2)\otimes\cdots\otimes M_{n_m}(K_m),$$

where K_i are fields containing K.

The related isomorphism is called a generalized Fourier transform (GFT) for A. If we are able to perform GFT efficiently then we can reduce computations in A to matrix multiplication. For some classes of algebras, e.g. abelian group algebras [12], there are known algorithms for GFT with running time $O(n\log n)$ where $n=\dim A$.

If the field K is algebraically closed (e.g. $\mathbb{C}$) then all $K_i=K$ and $\sum_{i=1}^m n_i^2$ equals the dimension of A. If the algebra A is commutative then all $n_i=1$ and A is isomorphic to a product of fields. This is actually the case in the Fast Subset Convolution [2] where the isomorphism is given by the Möbius transform.

B Proof of Theorem 9

Proof. The transformation ϕ can be computed and reverted (within the image) in time $O^*(2^n)$ assuming infinite precision and $O(1)$ time for any arithmetic operation [11]. In order to compute ϕ accurately, we need to look inside the paper [11].

Transformation ϕ can be represented as $\phi=\gamma\circ\upsilon$ where υ is a monomorphic embedding into another Clifford algebra and γ is an isomorphism with the matrix algebra. We modify isomorphism diagram (3) to show these mappings in more detail.

$$\begin{array}{ccccccc}
Cl_{n,0}(\mathbb{Z}) & \hookrightarrow & Cl_{n,0}(\mathbb{R}) & \xrightarrow{\upsilon} & Cl_{m,m}(\mathbb{R}) & \xrightarrow{\gamma} & \mathbb{M}_{2^m}(\mathbb{R}) \\
\downarrow * & & \downarrow * & & \downarrow * & & \downarrow * \\
Cl_{n,0}(\mathbb{Z}) & \hookrightarrow & Cl_{n,0}(\mathbb{R}) & \xrightarrow{\upsilon} & Cl_{m,m}(\mathbb{R}) & \xrightarrow{\gamma} & \mathbb{M}_{2^m}(\mathbb{R})
\end{array}$$

We begin with embedding $\upsilon : Cl_{n,0}(\mathbb{R}) \longrightarrow Cl_{m,m}(\mathbb{R})$ where $m = \frac{n}{2} + O(1)$ (see Definition 4.4 in [11]). Transformation υ is just a translation of basis so no arithmetic operations are required.

For the sake of disambiguation, we indicate the domain of the function γ with a lower index: $\gamma_k : Cl_{k,k}(\mathbb{R}) \longrightarrow \mathbb{M}_{2^k}(\mathbb{R})$. In the k-th step, we construct a matrix representation of $y \in Cl_{k,k}(\mathbb{R})$. Let y^+, y^- denote the projections of y onto subspaces spanned by products of respectively even and odd number of generators. Of course, $y = y^+ + y^-$ and $\gamma_k(y) = \gamma_k(y^+) + \gamma_k(y^-)$. Such an element y can be represented as $y = a + b\mathbf{x}_- + c\mathbf{x}_+ + d\mathbf{x}_-\mathbf{x}_+$ for $\mathbf{x}_+, \mathbf{x}_-$ being the first and the last generator ($\mathbf{x}_+^2 = e$, $\mathbf{x}_-^2 = -e$) and $a, b, c, d \in Cl_{k-1,k-1}(\mathbb{R})$. Now we can apply the recursive formula from Theorem 5.2 in [11]:

$$\gamma_k(y^+) = \gamma_{k-1}\left(\begin{bmatrix} a^+ - d^+ & -b^- - c^- \\ -b^- - c^- & a^+ + d^+ \end{bmatrix}\right), \quad \gamma_k(y^-) = \gamma_{k-1}\left(\begin{bmatrix} a^- - d^- & -b^+ + c^+ \\ b^+ + c^+ & -a^- - d^- \end{bmatrix}\right),$$

where $\gamma_{k-1}(M)$ stands for a block matrix with γ_{k-1} applied to each element of M.

We see that computing $\big(\gamma_k(y^+), \gamma_k(y^-)\big)$ can be reduced to computing 4 analogous pairs for $k-1$ and combining them using addition and subtraction. Hence, the coefficients of the obtained matrix will also be integers with $poly(n)$ number of bits and the total number of arithmetic operations is $O(m4^m) = O(n2^n)$.

The inverse transform γ^{-1} is also computed in m steps and we continue using lower index to indicate the domain alike for the forward transform. Let $Y \in \mathbb{M}_{2^k}(\mathbb{Z})$ and

$$Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}, \quad y_{ij} = \gamma_{k-1}^{-1}(Y_{ij}).$$

Then from Theorem 7.1 in [11] we know that

$$\gamma_k^{-1}(Y) = \frac{1}{2}\big((\hat{y_{22}} + y_{11}) + (\hat{y_{21}} - y_{12})\mathbf{x}_- + (\hat{y_{21}} + y_{12})\mathbf{x}_+ + (\hat{y_{22}} - y_{11})\mathbf{x}_-\mathbf{x}_+\big),$$

where $\hat{y} = y^+ - y^-$ and the rest of notation is as above. We can reduce computing γ_k^{-1} to 4 queries from $(k-1)$-th step so the total number of arithmetic operations is $O(m4^m) = O(n2^n)$.

This time the coefficients at each step are given as sums of elements from the previous step divided by 2. We do not need to prove that they remain integer at all steps because we can postpone the division until the last step. As long as $\gamma^{-1}(Y)$ is a product of two elements from $Cl_{m,m}(\mathbb{Z})$, it is guaranteed that the numbers in the last step would be divisible by 2^m. What is more, if we know that $\gamma^{-1}(Y) \in \upsilon(Cl_{n,0}(\mathbb{Z}))$ then we can revert the υ transform and obtain $\phi^{-1}(Y)$.

We have proven that we can switch representation between $Cl_{n,0}(\mathbb{Z})$ and $\mathbb{M}_{2^m}(\mathbb{Z})$ in time $O^*(2^n)$. The multiplication in $\mathbb{M}_{2^m}(\mathbb{Z})$ for inputs of $poly(n)$ size can be performed in time complexity $O^*(2^{\omega m}) = O^*(2^{\frac{\omega n}{2}})$ and the resulting matrix also contains only $poly(n)$-bits integers. This proves that the multiplication in $Cl_{n,0}(\mathbb{Z})$ admits an algorithm with running time $O^*(2^{\frac{\omega n}{2}})$. ◀

C Proof of Lemma 24

This section reduces the complicated algorithm for Hamiltonian Cycle to two isomorphism theorems and we suggest reading Appendix A first. Our goal is to compute values of h for the allowed triples assuming that non-zero values of f, g also occur only for the allowed triples.

$$h(A,B,C) = \sum_{\substack{A_1 \uplus A_2 = A \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} f(A_1,B_1,C_1)g(A_2,B_2,C_2)I_{B_1,B_2}I_{C_1,C_2} \tag{12}$$

Taking advantage of the size-grouping technique (see Observation 12) we can replace condition $A_1 \uplus A_2 = A$ with $A_1 \cup A_2 = A$ and focus on the following convolution.

$$(f \odot g)(A,B,C) = \sum_{\substack{A_1 \cup A_2 = A \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} f(A_1,B_1,C_1) g(A_2,B_2,C_2) I_{B_1,B_2} I_{C_1,C_2} \tag{13}$$

Let Ham be a subspace of $2^U \times 2^U \times 2^U \longrightarrow \mathbb{Z}$ given by functions admitting only the allowed triples (see Lemma 23), i.e. $f \in Ham \wedge f(A,B,C) \neq 0$ implies $A \cap (B \triangle C) = C \backslash B$. Observe that Ham is closed under the $\odot$ operation so it can be regarded as a 6^{tw}-dimensional algebra. Let H_D be an algebra over space $2^{U \backslash D} \times 2^D \times 2^D \longrightarrow \mathbb{Z}$ with multiplication given by the $\oslash$ operator defined as

$$(f \oslash g)(E,B,C) = \sum_{\substack{E_1 \uplus E_2 = E \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} f(E_1,B_1,C_1) g(E_2,B_2,C_2) I_{B_1,B_2} I_{C_1,C_2} (-1)^{|E_1|(|B_2|+|C_2|)}.$$

We want to show that Ham is isomorphic (see Definition 28) with a product of all H_D for $D \subseteq U$ (see Definition 27). In particular, diagram (14) commutes.

$$\begin{array}{ccc} Ham & \xrightarrow{\quad\tau\quad} & \bigotimes_{D \subseteq U} H_D \\ \downarrow \odot & & \downarrow \oslash \\ Ham & \xrightarrow{\quad\tau\quad} & \bigotimes_{D \subseteq U} H_D \end{array} \tag{14}$$

where $\tau_D : Ham \longrightarrow H_D$ is given as

$$(\tau_D f)(E,B,C) = I_{B,E} I_{C,E} \sum_{A \subseteq D} f(A, B \cup E, C \cup E).$$

▶ **Lemma 32.** *Transform τ and its inverse can be performed in time $O^*(6^{tw})$.*

▶ **Corollary 33.** *Transformation τ is reversible.*

▶ **Lemma 34.** *Given $f, g \in H_D$ we can compute $f \oslash g$ in time $O^*(2^{\omega|D|} 2^{|U \backslash D|})$.*

▶ **Lemma 35.** *Diagram (14) commutes, i.e. τ is a homomorphism of algebras.*

▶ **Corollary 36.** *Transformation τ is an isomorphism of algebras.*

As for the Clifford algebras, we can switch the representation of the algebra to perform multiplication in the simpler one, and then revert the isomorphism to get the result. The most time consuming part of the algorithm is performing the $\oslash$ convolutions. Total number of operations modulo polynomial factor can be bounded with Lemma 34 by

$$\sum_{D \subseteq U} 2^{\omega|D|} 2^{|U \backslash D|} = \sum_{k=0}^{tw} \binom{tw}{k} 2^{\omega k} 2^{tw-k} = (2^\omega + 2)^{tw}. \tag{15}$$

The rest of the appendix is devoted to proving Lemmas 32, 34, 35.

Proof of Lemma 32. For fixed sets B, C let $H = B \cap C$, $F = B \triangle C$, $B_1 = B \backslash C$, $C_1 = C \backslash B$. Observe that every allowed triple (A,B,C) must satisfy $A \cap F = C_1$. Therefore we can represent Ham as a union of sets

$$T_{B_1,C_1,H} = \Big\{ (A_1 \cup C_1, B_1 \cup H, C_1 \cup H) \;\Big|\; A_1 \subseteq U \backslash (B_1 \cup C_1) \Big\}.$$

for all pairwise disjoint triples $B_1, C_1, H \subseteq U$. Functions over $T_{B_1,C_1,H}$ can be parameterized with only the A_1 argument. Consider following transformation over function space on $T_{B_1,C_1,H}$.

$$(\gamma_{B_1,C_1,H} f)(A_1) = \sum_{A_0 \subseteq A_1} f(A_0 \cup C_1, B_1 \cup H, C_1 \cup H)$$

Transform $\gamma_{B_1,C_1,H}$ is just the Möbius transform, therefore it can be performed and reverted in time $O^*(2^{|U\setminus(B_1\cup C_1)|})$ (see Theorem 5). Values of γf correspond directly to values of τf.

$$\begin{aligned}(\tau_D f)(E,B,C) &= I_{B,E} I_{C,E} \sum_{A \subseteq D} f(A, B \cup E, C \cup E) = \\ &= I_{B,E} I_{C,E} \sum_{A \subseteq D} f(A, B_1 \cup H \cup E, C_1 \cup H \cup E) = \\ &= I_{B,E} I_{C,E} \sum_{A_0 \subseteq D\setminus F} f(A_0 \cup C_1, B_1 \cup H \cup E, C_1 \cup H \cup E) = \\ &= I_{B,E} I_{C,E} (\gamma_{B_1,C_1,H\cup E} f)(D\setminus F)\end{aligned}$$

$$\begin{aligned}(\gamma_{B_1,C_1,H} f)(A_1) &= \sum_{A_0 \subseteq A_1} f(A_0 \cup C_1, B_1 \cup H, C_1 \cup H) = \\ &= \sum_{A_0 \subseteq A_1 \cup C_1} f(A_0, B_1 \cup H, C_1 \cup H) = \\ &= \sum_{A_0 \subseteq A_1 \cup C_1} f\big(A_0, B_2 \cup (H\setminus A_1), C_2 \cup (H\setminus A_1)\big) = \\ &= (\tau_{A_1\cup C_1} f)(E, B_2, C_2) I_{B_2,E} I_{C_2,E}\end{aligned}$$

where $E = H\setminus A_1$, $B_2 = B_1 \cup (H \cap A_1)$, $C_2 = C_1 \cup (H \cap A_1)$ are valid arguments of $\tau_{A_1\cup C_1}$.

To estimate the total number of operations consider all choices of F. The partition into $F = B_1 \uplus C_1$ can be done in $2^{|F|}$ ways, the set H can be chosen in $2^{|U\setminus F|}$ ways, and for such triple we have to perform the $\gamma_{B_1,C_1,H}$ transform (or its inverse) what involves $O^*(2^{|U\setminus F|})$ operations. Hence, the total running time (modulo polynomial factors) is

$$\sum_{F \subseteq U} 2^{|F|} 4^{|U\setminus F|} = \sum_{k=0}^{tw} \binom{tw}{k} 2^k 4^{tw-k} = 6^{tw}. \qquad \blacktriangleleft$$

Proof of Lemma 34. Applying the size-grouping (see Observation 12) allows us to neglect the $(-1)^{|E_1|(|B_2|+|C_2|)}$ factor and replace condition $E_1 \uplus E_2 = E$ with $E_1 \cup E_2 = E$. Therefore it suffices to perform the $\odot$ convolution on H_D (the same as in (13)).

$$(f \odot g)(E,B,C) = \sum_{\substack{E_1\cup E_2 = E \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} f(E_1, B_1, C_1) g(E_2, B_2, C_2) I_{B_1,B_2} I_{C_1,C_2}.$$

Let us denote

$$(\mu_E f)(B,C) = \sum_{F \subseteq E} f(F,B,C).$$

Transform μ and its inverse can be computed using Möbius transform (see Theorem 5) in time $O^*(2^{|U\setminus D|})$ for all E and a fixed pair of sets B, C. We perform it for all $4^{|D|}$ such pairs.

It turns out that μ is an isomorphism between $(H_D, \odot)$ and a product of all algebras given by images of μ_E for $E \subseteq U \setminus D$ (see Definitions 27, 28) with multiplication given by NSC2, i.e. $(\mu_E f) \diamond_2 (\mu_E g) = \mu_E(f \odot g)$. We can again switch the representation of the algebra, multiply the elements, and then revert the isomorphism. The computations below show that μ is a homomorphism of algebras and we know already that μ is reversible.

$$\begin{aligned}
\big((\mu_E f) \diamond_2 (\mu_E g)\big)(B, C) = \\
&= \sum_{\substack{B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} (\mu_E f)(B_1, C_1)(\mu_E g)(B_2, C_2) I_{B_1,B_2} I_{C_1,C_2} = \\
&= \sum_{\substack{E_1, E_2 \subseteq E \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} f(E_1, B_1, C_1) g(E_2, B_2, C_2) I_{B_1,B_2} I_{C_1,C_2} = \\
&= \sum_{F \subseteq E} \sum_{\substack{E_1 \cup E_2 = F \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} f(E_1, B_1, C_1) g(E_2, B_2, C_2) I_{B_1,B_2} I_{C_1,C_2} = \\
&= \big(\mu_E(f \odot g)\big)(B, C)
\end{aligned}$$

To perform multiplication of $\mu(a)$ and $\mu(b)$, where $a, b \in H_D$, we have to perform NSC2 $\big(O^*(2^{\omega|D|})$ time complexity, see Theorem 14$\big)$ for each $E \subseteq U \setminus D$, what results in desired running time. ◀

Proof of Lemma 35. We need to show that for each $B, C \subseteq D, D \cap E = \emptyset$ it is $(\tau_D(f \odot g))(E, B, C) = ((\tau_D f) \oslash (\tau_D g))(E, B, C)$. Let us start with unrolling the formula for $\tau_D(f \odot g)$. Keeping in mind that $B \cap E = C \cap E = \emptyset$ we can see that

$$\begin{aligned}
&(\tau_D(f \odot g))(E, B, C) = \\
&= \sum_{A \subseteq D} (f \odot g)(A, B \cup E, C \cup E) I_{B,E} I_{C,E} = \\
&= \sum_{\substack{A_1, A_2 \subseteq D \\ B_1 \uplus B_2 = B \\ E_1 \uplus E_2 = E \\ C_1 \uplus C_2 = C \\ F_1 \uplus F_2 = E}} \begin{aligned}[t] & f(A_1, B_1 \cup E_1, C_1 \cup F_1) g(A_2, B_2 \cup E_2, C_2 \cup F_1) \\ & I_{B_1 \cup E_1, B_2 \cup E_2} I_{C_1 \cup F_1, C_2 \cup F_2} I_{B,E} I_{C,E}. \end{aligned}
\end{aligned} \tag{16}$$

On the other hand, we have

$$\begin{aligned}
&((\tau_D f) \oslash (\tau_D g))(E, B, C) = \\
&= \sum_{\substack{E_1 \uplus E_2 = E \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} (\tau_D f)(E_1, B_1, C_1)(\tau_D g)(E_2, B_2, C_2) I_{B_1,B_2} I_{C_1,C_2} (-1)^{|E_1|(|B_2|+|C_2|)} = \\
&= \sum_{\substack{A_1, A_2 \subseteq D \\ E_1 \uplus E_2 = E \\ B_1 \uplus B_2 = B \\ C_1 \uplus C_2 = C}} \begin{aligned}[t] & f(A_1, B_1 \cup E_1, C_1 \cup E_1) g(A_2, B_2 \cup E_2, C_2 \cup E_2) \\ & I_{B_1,B_2} I_{C_1,C_2} I_{B_1,E_1} I_{C_1,E_1} I_{B_2,E_2} I_{C_2,E_2} (-1)^{|E_1|(|B_2|+|C_2|)}. \end{aligned}
\end{aligned} \tag{17}$$

We want to argue that all non-zero summands of (16) satisfy $E_1 = F_1, E_2 = F_2$. Indeed, let us assume $v \in F_1 \setminus E_1$. As $v \in E$ so $v \notin D \supseteq A, B, C$ and $\big([v \in A_1], [v \in B_1 \cup E_1], [v \in C_1 \cup F_1]\big) = (0, 0, 1)$ which is not a valid triple what implies $f(A_1, B_1 \cup E_1, C_1 \cup F_1) = 0$.

Assumption $v \in E_1 \backslash F_1$ leads to $\big([v \in A_1], [v \in B_1 \cup E_1], [v \in C_1 \cup F_1]\big) = (0,1,0)$ but $v \in E = E_1 \uplus E_2 = F_1 \uplus F_2$ so $\big([v \in A_2], [v \in B_2 \cup E_2], [v \in C_2 \cup F_2]\big) = (0,0,1)$ and $g(A_2, B_2 \cup E_2, C_2 \cup F_1) = 0$. The same arguments can be used if $v \in E_2 \triangle F_2$.

Now we just need to prove that for $E_1 = F_1, E_2 = F_2$ the I factors in (16) and (17) are equivalent. We apply Claim 2 to $I_{B_1 \cup E_1, B_2 \cup E_2} I_{C_1 \cup E_1, C_2 \cup E_2}$. We can omit factor $I^2_{E_1,E_2} = 1$ as well as $I_{B_1,B_2} I_{C_1,C_2}$ appearing also in (17). What is left to prove is that

$$I_{B_1,E_2} I_{E_1,B_2} I_{B,E} = I_{B_1,E_1} I_{B_2,E_2} (-1)^{|E_1||B_2|},$$
$$I_{C_1,E_2} I_{E_1,C_2} I_{C,E} = I_{C_1,E_1} I_{C_2,E_2} (-1)^{|E_1||C_2|}.$$

According to Claim 1 we can replace $I_{E_1,B_2}(-1)^{|E_1||B_2|}$ with I_{B_2,E_1} what reduces the formula in the first row to Claim 2 for $B = B_1 \uplus B_2$, $E = E_1 \uplus E_2$. Applying analogous observation to the second row finishes the proof. ◀