Cartoon Computation: Quantum-like computing without quantum mechanics

Diederik Aerts [1] and Marek Czachor [1,2]

[1] *Centrum Leo Apostel (CLEA) and Foundations of the Exact Sciences (FUND)*

Vrije Universiteit Brussel, 1050 Brussels, Belgium

[2] *Katedra Fizyki Teoretycznej i Informatyki Kwantowej*

Politechnika Gdańska, 80-952 Gdańsk, Poland

Abstract

We present a computational framework based on geometric structures. No quantum mechanics is involved, and yet the algorithms perform tasks analogous to quantum computation. Tensor products and entangled states are not needed — they are replaced by sets of basic shapes. To test the formalism we solve in geometric terms the Deutsch-Jozsa problem, historically the first example that demonstrated the potential power of quantum computation. Each step of the algorithm has a clear geometric interpetation and allows for a cartoon representation.

1

I. INTRODUCTION

Thinking of quantum computation one typically has in mind a quantum computer — a device based on and limited by the laws of the microworld. But the same laws that allow for quantum computation state that the noise occuring in actual systems may make the computation more or less unrealistic. The two recent US and UE strategic reports [1] show how the level of practical difficulties varies from implementation to implementation. The goal of our paper is to show that perhaps one should also look for non-microworld implementations of quantum computation. More precisely, we present a framework for quantum-like algorithms that does not refer to quantum mechanics, and involves only geometric structures algebraized by means of geometric algebras (GA). To prove that the new framework indeed works we solve in a GA way the celebrated Deutsch–Jozsa (DJ) problem [2].

There are some trivial ways of including GA in quantum computation [3], but this is not what we want to do. The GA algorithm we present below is not just a simple translation of the quantum one. As opposed to quantum computation the basic operation is not the tensor product but the geometric (Clifford) product (the map that forms an oriented square from two vectors, a cube from a vector and a square, a square from a vector and a cube, and so on, as shown on the figures). In quantum computation we are bound to use unitary operations (quantum dynamics is unitary) and projectors (measurements are represented by projections). Binary numbers are built by tensoring qubits with one another. All these operations are higly counterintuitive.

In GA computation the operations are different and there is nothing counterintuitive about them. After a necessary amount of exercise the operations can be visualized without great difficulty. Binary numbers are coded directly in terms of basic geometric shapes, with no tensor product or quantum entanglement. Parallel processing is performed on 'bags of shapes'. In GA computation one can really *see* the solution.

II. ELEMENTS OF GA

Let us now recall those basic facts about GA that are important for our purposes [4, 5, 6, 7, 8, 9]. One begins with an n-dimensional Euklidean space V_n whose orthonormal basis is $\{e_1, \ldots, e_n\}$. The associative geometric product ab of two vectors $a = \sum_{k=1}^{n} a_k e_k$,

2

$b = \sum_{k=1}^{n} b_k e_k$, is defined by linearity from the Clifford algebra

$$e_k e_l + e_l e_k = 2\delta_{kl}\mathbf{1} \tag{1}$$

of the basis. Here $\mathbf{1}$ is the neutral element of the GA: $a\mathbf{1} = \mathbf{1}a = a$. Directed subspaces are then associated with the set of *blades* defined as geometric products of different basis vectors supplemented by the identity $\mathbf{1}$, corresponding to the basic oriented scalar (analogous to a charged point). The blades include vectors (oriented line segments), bivectors (oriented plane segments), trivectors (oriented volume segments), and so on. A general element A of GA, called a multivector, is a linear combination of the blades

$$A = A_0 \mathbf{1} + \sum_k A_k e_k + \sum_{k<l} A_{kl} e_k e_l + \cdots + A_{1...n} e_1 \ldots e_n,$$

where the coefficients are real.

Figures 1–2 explicitly illustrate the geometry behind multivectors and their geometric products in a plane. Geometrically the basic blades and their negatives in 2D are: $\mathbf{1} = \circ$, $e_1 = \rightarrow$, $e_2 = \uparrow$, $e_{12} = \square$, $-\mathbf{1} = \bullet$, $-e_1 = \leftarrow$, $e_2 = \downarrow$, $-e_{12} = \blacksquare$. Here $\circ$ and $\bullet$ denote the two oppositely 'charged' points.

Note that the dimension of the space of shapes associated with the plane is four. Multivectors are 'bags of shapes' and the high dimensionality is similar to the one known from configuration spaces in mechanics. As one does not have problems with imagining a $3N$-dimensional space representing configurations of N particles, there is no difficulty with visualizing the 4-dimensional space representing the 'bags' in Figure 2.

The first element that seems new and is beyond the standard presentation of GA is the binary parametrization of blades and the role it plays for the geometric product. Denote: $\mathbf{1} = e_{0...0}$, $e_1 = e_{10...0}$, $e_2 = e_{010...0}$, ..., $e_{125} = e_{110010...0}$, ..., $e_{12...n-1,n} = e_{11...11}$. The notation shows that there is a one-to-one relation between an n-bit number and an element of GA based on V_n. Geometric product in the binary parametrization reads [10, 11]

$$e_{A_1...A_n} e_{B_1...B_n} = (-1)^{\sum_{k<l} B_k A_l} e_{(A_1...A_n)\oplus(B_1...B_n)}, \tag{2}$$

where $(A_1 \ldots A_n) \oplus (B_1 \ldots B_n)$ means componentwise addition mod 2, i.e. the n-dimensional XOR. The geometric product may be thus regarded as a projective representation of XOR. This observation is the deperture point for our GA computational framework.

III. DJ PROBLEM IN GA FRAMEWORK

Assume $f : \{0,1\}^n \to \{0,1\}$ is a constant or balanced function. Consider an $(n+1)$-dimensional Euclidean space V_{n+1} with orthonormal basis $\{e_1, \ldots e_{n+1}\}$ and its associated GA. Let E_{n+1} denote the sum of all the blades,

$$E_{n+1} = \sum_{A_1 \ldots A_{n+1}} e_{A_1 \ldots A_{n+1}}. \tag{3}$$

Employing (2) we find, for $e_n = e_{0 \ldots 010}$,

$$E_{n+1} e_n = \sum_{A_1 \ldots A_{n+1}} (-1)^{A_{n+1}} e_{A_1 \ldots A_{n+1}}. \tag{4}$$

This step is analogous to the first step of the DJ quantum algorithm [2]. Indeed, let U_{n+1} be the tensor product of $n+1$ Hadamard gates. Then one begins with

$$U_{n+1} |0 \ldots 01\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{A_1 \ldots A_{n+1}} (-1)^{A_{n+1}} |A_1 \ldots A_{n+1}\rangle.$$

Note the difference in location of 1 in $e_{0 \ldots 010}$ and $|0 \ldots 01\rangle$. Now assume there exists an oracle E_f that performs

$$E_f e_{A_1 \ldots A_n A_{n+1}} = e_{A_1 \ldots A_n A_{n+1}} e_{0 \ldots 0 f(A_1 \ldots A_n)} \tag{5}$$

The action of the oracle reduces either to the multiplication by the $(n+1)$th basis vector $e_{n+1} = e_{0 \ldots 01}$, if $f(A_1 \ldots A_n) = 1$, or to the trivial multiplication by $\mathbf{1} = e_{0 \ldots 0}$ in the opposite case. Accordingly

$$E_f E_{n+1} e_n = \sum_{A_1 \ldots A_n} (-1)^{f(A_1 \ldots A_n)} (e_{A_1 \ldots A_n 0} - e_{A_1 \ldots A_n 1}).$$

This step is analogous to the oracle action in the quantum algorithm, where

$$U_f U_{n+1} |0 \ldots 01\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{A_1 \ldots A_n} (-1)^{f(A_1 \ldots A_n)} \Big(|A_1 \ldots A_n 0\rangle - |A_1 \ldots A_n 1\rangle \Big)$$

The final step is performed by means of

$$F_{n+1} = \sum_{A_1 \ldots A_n} e^\dagger_{A_1 \ldots A_n 0}, \tag{6}$$

essentially the reverse [12] of E_n:

$$F_{n+1} E_f E_{n+1} e_n = \sum_{A_1 \ldots A_n = 0}^{1} (-1)^{f(A_1 \ldots A_n)} e_{0 \ldots 0} + \ldots,$$

4

where the dots stand for the combination of all the blades different from $\mathbf{1} = e_{0\ldots0}$. If Π projects on $e_{0\ldots0}$, then

$$\Pi F_{n+1} E_f E_{n+1} e_n = \begin{cases} (-1)^{f(0\ldots0)} 2^n \mathbf{1} & \text{for constant } f \\ 0 & \text{for balanced } f \end{cases}$$

Looking at the $e_{0\ldots0}$ component we conclude that f is constant if the component is nonzero, and balanced if the component is zero. We have achieved the same goal as the quantum algorithm.

IV. CARTOON ALGORITHMS

Cartoon versions of the 2-bit GA algorithm are shown on Figures 3–5. All the three figures involve the same first step, but the oracles are different. The projection Π means that we select from the final bag the dots. If the dots are black the function is constant with $f(0) = 1$; white dots mean constant function, $f(0) = 0$ (the oracle is then trivial), and no dots means zero, i.e. a balanced function. Figure 6 shows the algorithm for the 3-bit problem and constant $f(00) = 0$. We do not show the oracle since in this case, analogously to Figure 4, the oracle acts trivially. For three bits there are 8 blades: 1 scalar, 3 edges e_k, 3 walls e_{kl}, and 1 cube e_{123}. The walls are white on one side, and black on the other. e_{123} is white, and its negative is black. We recommend the readers to translate the cartoon into a GA expression.

V. ADVANTAGES AND LIMITATIONS

The basic factor that limits practical applicability of cartoon computation is how to physically implement E_f, E_{n+1}, F_{n+1}. The same problem occurs in standard quantum computation but one hopes that any finte unitary transformation can be physically realized by means of a quantum system, at least in principle. However, assuming that black boxes that perform E_f, E_{n+1}, F_{n+1} in single runs exist, we obtain the same complexity of the algorithm as the quantum one. The main advantage is that we no longer have to look for such implementations in the microscopic domain. We only need geometry, and not necessarily that of an Euclidean space. Another advantage is that the notions of superposition and entanglement have here a clear geometric meaning (bags of blades) and no tensor products

are needed. The coefficients occuring in our 'entangled states' do not have a probabilistic meaning, but can be both positive and negative, and thus lead to interference effects. The latter property, in addition to parallelism, is the main feature making our algorithms similar to the quantum ones.

VI. DISCUSSION

The algorithms could also be represented in a matrix way with n-bit numbers given by Cartan's representation of an appropriate Clifford algebra [13]. Examples of such calculations can be found in [10, 11]. The exercise is instructive but can be conceptually misleading. As often stressed in the GA literature, the matrix representations introduce redundant elements that obscure the actual geometric meaning of GA operations. In particular, Cartan's representation is constructed by means of tensor products of Pauli matrices, a fact that may make the impression we are using quantum algorithms in notational disguise, perhaps extended by unphysical operations, which is not the case.

A lot of inspiration for our own work came from certain 'quantum-like' constructions known in semantic analysis and artificial intelligence (AI) [14]. For example, the idea of replacing tensor product representations [15] by their 'compressions' based on alternative multiplications occurs in convolution based distributed representations of cognitive structures [16]. 'Bags of shapes' are analogous to 'bags of words' from latent semantic analysis (LSA) [17]. The links to AI and, more generally to the studies of human intelligence become especially intriguing if one thinks of geometric product as a way of *decoding* relations between geometric objects. Indeed, consider the following problem: If $\rightarrow$ ■ = □ $\rightarrow$=$\leftarrow$ □ =↓ then ■ $\leftarrow$=? The GA solution is ■ $\leftarrow$=↑$\rightarrow$$\leftarrow$=↑ • = − ↑=↓=$\rightarrow$ ■. The computation is based on the observation that ■ should be identified with ↑$\rightarrow$, □ with $\rightarrow$↑, ■ with −□, and $\rightarrow$$\leftarrow$ with −. Similarity to certain IQ tests is striking. Of particular relevance to our approach are the binary spatter codes (BSC) [10, 18], a method of coding and processing distributed information, based on appropriately defined superpositions of XORed binary strings.

An interesting exercise is to try to imagine the three steps of the algorithm from Figure 6 as three different levels of geometric relations between the three 'bags of shapes' representing F_3, E_3 and e_2. After some training one indeed starts to *understand* and *see* why the
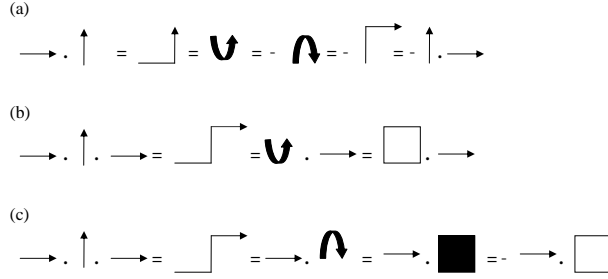
6

FIG. 1: Geometric product is noncommutative. (a) Geometry behind $e_1 e_2 = e_{12} = -e_2 e_1$. (b) Associativity implies $e_1 e_2 e_1 = e_{12} e_1$. (c) The same as (b) but performed in a different order. Self-consistency implies that $e_1 e_2 e_1 = e_1(-e_{12}) = -e_1 e_{12}$.

multiplication $F_3 E_3 e_2$ looks like the rightmost bag. In this way we have approached the intriguing problem of understanding via visualization, and the role played in this context by quantum geometry [19].

To conclude, we seem to be dealing with a new research field on the borderland between 'quantum structures' and cognitive science. The counterintuitive elements typical of quantum computation are here absent. The basic structure is geometric and thus very general. How to build a 'parallel geometric processor' based on a physical process is a separate issue, but one is no longer confined to quantum systems. In particular, realizations based on classical physics cannot be excluded. Finally, implications for 'the missing science of consciousness', artificial intelligence, and various representations of cognitive structures can be far reaching and should be further studied.

Acknowledgments

[1] *Quantum Computation Roadmap*, http://qist.lanl.gov. (2004); *ERA Pilot Roadmap — Quantum Information Sciences and Technologies*, http://qist.ect.it/Reports/reports.htm (2006).

[2] D. Deutsch and R. Jozsa, Proc. Roy. Soc. London A **439**, 553 (1992).

(a)

$= 3 \times \mathbf{1} + 2 \times e_1 - 0.5 \times e_2 + 4 \times e_{12}$

(b)

$= (3 \times \mathbf{1} + 2 \times e_1 + 0.5 \times e_2 - 4 \times e_{12})e_{12} = 3 \times e_{12} + 2 \times e_2 - 0.5 \times e_1 + 4 \times \mathbf{1}$
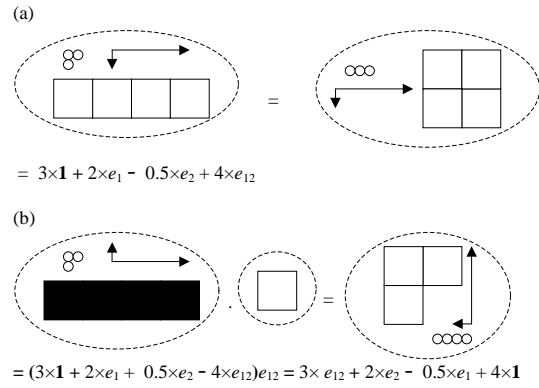
FIG. 2: (a) Multivectors are 'bags of blades'. The two different bags are equivalent. (b) Geometric product of a multivector and a blade.

Step 1:

$= E_2 e_1 \quad = \quad \quad = \mathbf{1} + e_1 - e_2 - e_{12}$

Step 2: Oracle E_f

$E_f E_2 e_1 = E_f$

Step 3:

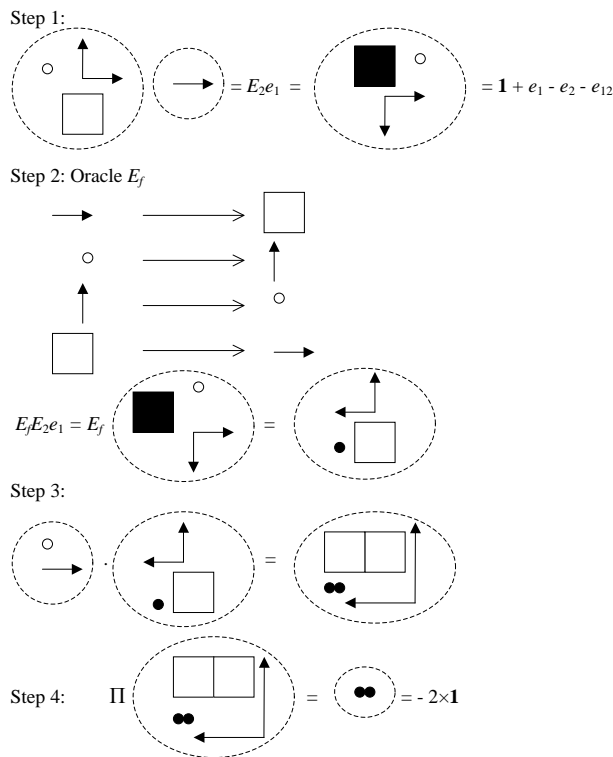Step 4: Π $\quad = \quad \quad = -2 \times \mathbf{1}$

FIG. 3: DJ problem solved exclusively by means of geometric operations: a 2-bit problem with $f(0) = f(1) = 1$. One can *see* the result: The two black dots mean $-2 = (-1)^{f(0)} 2^1$.

[3] It is enough to say that any qubit can be regarded as a 2×2 matrix with empty second column, and all 2×2 matrices are combinations of Pauli matrices, which are a representation of some GA. Also all 2×2 unitary transformations can be parmetrized by elements of GA. This type of formulation is behind the approaches discussed in [20, 21, 22]. It seems that such an inclusion
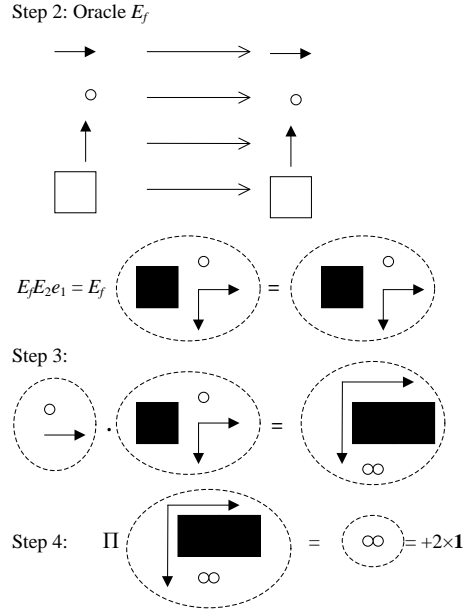
Step 2: Oracle E_f

FIG. 4: The 2-bit problem and the constant function $f(0) = f(1) = 0$. The oracle acts trivially. One again sees the result: The two white dots mean $+2 = (-1)^{f(0)}2^1$.
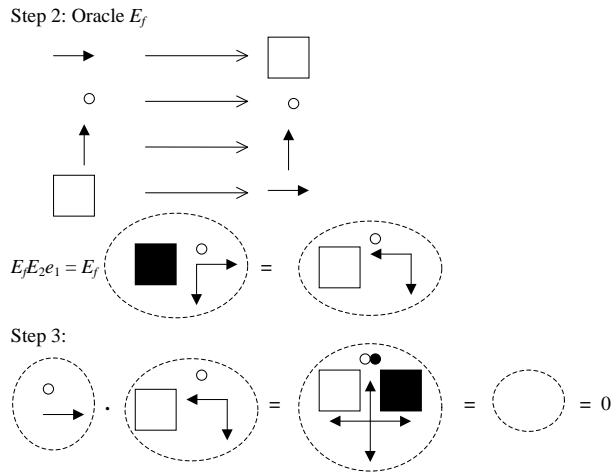
Step 2: Oracle E_f

FIG. 5: The 2-bit problem and the balanced function $f(0) = 0$, $f(1) = 1$. The bag is empty.

of GA in quantum computation is more like a translation into a different language, and does not, *per se*, bring new conceptual elements.

[4] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics* (Reidel, Dordrecht, 1984).

[5] G. Sommer (ed.), *Geometric Computing with Clifford Algebras* (Springer, Berlin 2001).

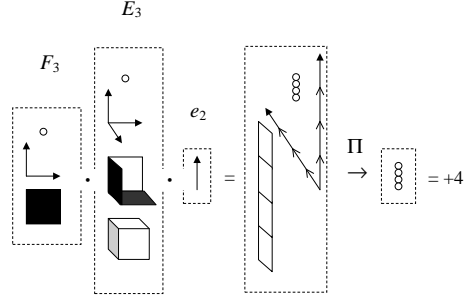[6] L.Dorst, C. J. L. Doran, J. Lasenby (eds.), *Applications of Geometric Algebra in Computer*

9

FIG. 6: 3-bit problem for the constant function $f(00) = 0$. The oracle is trivial. Four white dots mean $+4 = (-1)^{f(00)}2^2$.

Science and Engineering (Birkhauser, Boston, 2002).

[7] D. Hestenes, *Space-Time Algebra* (Gordon & Breach, New York, 1966).

[8] D. Hestenes, *New Foundations for Classical Mechanics* (Kluwer, Dordrecht, 1986).

[9] M. Pavsic, *The Landscape of Theoretical Physics: A Global View* (Kluwer, Boston, 2001).

[10] D. Aerts, M. Czachor, and B. De Moor, cs.AI/0610075 (2006).

[11] D. Aerts and M. Czachor, quant-ph/0610187 (2006).

[12] The reverse acts on blades as follows:

$$(e_{k_1}e_{k_2}\ldots e_{k_{j-1}}e_{k_j})^\dagger = e_{k_j}e_{k_{j-1}}\ldots e_{k_2}e_{k_1},$$

and is extended to multivectors by linearity.

[13] P. Budinich and A. Trautman, *The Spinorial Chessboard* (Springer, Berlin, 1988).

[14] D. Aerts and M. Czachor, J. Phys. A **37**, L123 (2004).

[15] P. Smolensky, Artif. Intelligence **46**, 159 (1990).

[16] T. Plate, *Holographic Reduced Representation: Distributed Representation for Cognitive Structures* (CSLI Publications, Stanford, 2003).

[17] T. K. Landauer and S. T. Dumais, Psychologigal Rev. **104**, 211 (1997).

[18] P. Kanerva, in *Artificial Neural Networks–ICANN Proceedings*, Lecture Notes in Computer Science vol. 1112, C. von der Malsburg *et al.* (Eds.) (Springer, Berlin, 1996) 869.

[19] R. Penrose, *Shadows of the Mind: A Search for the Missing Science of Consciousness* (Oxford University Press, Oxford, 1994). Of particular relevance is the discussion in Chapter 1.21.

[20] S. Somaroo, D. G. Cory, and T. F. Havel, Phys. Lett. A **240**, 1 (1998).

[21] T. F. Havel and C. J. L. Doran, quant-ph/0004031.

[22] M. Van den Nest, J. Dehaene, and B. De Moor, Phys. Rev. A **72**, 014317 (2005).