On Geometric Algebra representation of Binary Spatter Codes

Diederik Aerts [1], Marek Czachor [1,2], and Bart De Moor [3]
[1] *Centrum Leo Apostel (CLEA) and Foundations of the Exact Sciences (FUND)*
Brussels Free University, 1050 Brussels, Belgium
[2] *Katedra Fizyki Teoretycznej i Informatyki Kwantowej*
Politechnika Gdańska, 80-952 Gdańsk, Poland
[3] *ESAD-SCD, K. U. Leuven, 3001 Leuwen, Belgium*

Kanerva's Binary Spatter Codes are reformulated in terms of geometric algebra. The key ingredient of the construction is the representation of XOR binding in terms of geometric product.

I. INTRODUCTION

Distributed representation is a way of representing information in a pattern of activation over a set of neurons, in which each concept is represented by activation over multiple neurons, and each neuron participates in the representation of multiple concepts [1]. Examples of distributed representations include Recursive Auto-Associative Memory (RAAM) [2], Tensor Product Representations [3], Holographic Reduced Representations (HRRs) [4, 5], and Binary Spatter Codes (BSC) [6, 7, 8].

BSC is a powerful and simple method of representing hierarchical structures in connectionist systems and may be regarded as a binary version of HRRs. Yet, BSC has some drawbacks associated with the representation of chunking. This is why different versions of BSC can be found in the literature. In [6, 7] chunking is given by a majority-rule thresholded addition of binary strings, an operation that often discards a lot of important information. In [8] the ordinary addition is employed, and bits are parametrized differently.

The main message we want to convey in this paper is that there exists a very natural representation of BSC at the level of Clifford algebras. Binding of vectors is here performed by means of the Clifford product and chunking is just ordinary addition. Since Cliford algebras possess a geometric interpretation in terms of Geometric Algebra (GA) [17, 18, 19], the cognitive structures processed in BSC or HHRs obtain a geometric content. This is philosophically consistent with many other approaches where cognition is interpreted in geometric terms [14, 22]. Of particular relevance may be the links to neural computation whose GA and HRR versions were formulated by different authors (cf. [5, 29, 30].

The present paper can be also seen in a wider context of a "quantum structures" approach to cognitive problems we have outlined elsewhere [9, 10, 11, 12, 13]. Cartan's representation of GA in terms of tensor products of Pauli matrices introduces formal links to quantum computation (cf. [23, 24, 25, 26]). The philosophy we advocate here is also not that far from the approach of Widdows, where both geometric an "quantum" aspects play an important role [14, 15, 16].

It should be stressed that the GA calculus has already proved to be a powerful tool in applied branches of computer science (computer vision [20], robotics [21]). GA is a comprehensive language that simplified and integrated many branches of classical and quantum physics [31]. One may hope that it will play a similar role in cognitive science.

II. BINARY SPATTER CODES

In BSC information is encoded into long unstructured strings of bits that form a *holistic record*. The record is composed in two steps called binding and chunking.

Binding of a role $\mathbf{x}$ with a filler $\mathbf{y}$ is performed by means of XOR $\oplus$ (componentwise addition of binary strings mod 2); the role-filler object is $\mathbf{x} \oplus \mathbf{y}$. Chunking means adding the bound structures in a suitable way.

In order to illustrate the original BSC and its algebraic modification let us take the example from [7]. The encoded record is

$$\mathbf{PSmith} \;=\; \mathbf{name} \oplus \mathbf{Pat} + \mathbf{sex} \oplus \mathbf{male} + \mathbf{age} \oplus \mathbf{66}. \tag{1}$$

Decoding of the "name" looks as follows

$$\begin{aligned} \mathbf{Pat}' \;&=\; \mathbf{name} \oplus \mathbf{PSmith} \\ &=\; \mathbf{name} \oplus \big[\mathbf{name} \oplus \mathbf{Pat} + \mathbf{sex} \oplus \mathbf{male} + \mathbf{age} \oplus \mathbf{66}\big] \\ &=\; \mathbf{Pat} + \mathbf{name} \oplus \mathbf{sex} \oplus \mathbf{male} + \mathbf{name} \oplus \mathbf{age} \oplus \mathbf{66} \\ &=\; \mathbf{Pat} + \text{noise} \to \mathbf{Pat}. \end{aligned} \tag{2}$$

We have used here the involutive nature of XOR and the fact that the "noise" can be eliminated by clean-up memory. The latter means that we compare **Pat**$'$ with records stored in some memory and check, by means of the Hamming distance, which of the stored elements is closest to **Pat**$'$. A similar trick could be done be means of circular convolution in HRRs, but then we would have used an approximate inverse **name***, and an appropriate measure of distance. Again, the last step is comparison of the noisy object with "pure" objects stored in clean-up memory.

III. GEOMETRIC-ALGEBRA REPRESENTATION OF BINARY SPATTER CODES

Euclidean-space GA is constructed as follows. One takes an n-dimensional linear space with orthonormal basis $\{e_1, \ldots, e_n\}$. Directed subspaces are then associated with the set

$$\{1, e_1, \ldots, e_n, e_{12}, e_{13} \ldots, e_{n-1,n}, \ldots, e_{12\ldots n}\}. \tag{3}$$

Here 1 corresponds to scalars, i.e. a 0-dimensional space. Then we have vectors (oriented segments), bivectors (oriented parallelograms), and so on. There exists a natural parametrization: $1 = e_{0\ldots0}$, $e_1 = e_{10\ldots0}$, $e_2 = e_{010\ldots0}$, $\ldots$, $e_{125} = e_{110010\ldots0}$, $\ldots$, $e_{12\ldots n-1,n} = e_{11\ldots1}$, which shows that there is a one-to-one relation between an n-bit number and an element of GA. An element with k 1s and $n-k$ 0s is called a k-blade.

A *geometric product* of k 1-blades is a k-blade. For example, $e_{1248} = e_1e_2e_4e_8$. Moreover, $e_ne_m = -e_me_n$, if $m \neq n$, and $e_ne_n = 1$, for any n. GA is a Clifford algebra [28] enriched by certain geometric interpretations and operations.

Particularly interesting is the form of the geometric product that occurs in the binary parametrization. Let us work out a few examples:

$$\begin{aligned}
e_1e_1 &= e_{10\ldots0}e_{10\ldots0} = 1 = e_{0\ldots0} = e_{(10\ldots0)\oplus(10\ldots0)} && (4)\\
e_1e_{12} &= e_{10\ldots0}e_{110\ldots0} = e_1e_1e_2 = e_2 = e_{010\ldots0} = e_{(10\ldots0)\oplus(110\ldots0)} && (5)\\
e_{12}e_1 &= e_{110\ldots0}e_{10\ldots0} = e_1e_2e_1 = -e_2e_1e_1 = -e_2 = -e_{010\ldots0} = -e_{(110\ldots0)\oplus(10\ldots0)} && (6)\\
e_{1257}e_{26} &= e_{11001010\ldots0}e_{0100010\ldots0} = e_1e_2e_5e_7e_2e_6 = (-1)^2e_1e_2e_2e_5e_7e_6 = (-1)^2(-1)^1e_1e_2e_2e_5e_6e_7\\
&= (-1)^3e_1e_5e_6e_7 = (-1)^3e_{10001110\ldots0} = (-1)^De_{(11001010\ldots0)\oplus(0100010\ldots0)}. && (7)
\end{aligned}$$

The number D is the number of times a 1 from the right string had to "jump" over a 1 from the left one during the process of shifting the right string to the left. Symbolically the operation can be represented as

$$\begin{bmatrix} \longleftarrow\ 01000100\ldots0 \\ 11001010\ldots0 \end{bmatrix} \mapsto (-1)^D \begin{bmatrix} 01000100\ldots0 \\ 11001010\ldots0 \end{bmatrix} \mapsto (-1)^D \begin{bmatrix} 01000100\ldots0 \\ \oplus \\ 11001010\ldots0 \end{bmatrix} = (-1)^D \begin{bmatrix} 10001110\ldots0 \end{bmatrix}$$

The above observations, generalized to arbitrary strings of bits, yield

$$e_{A_1\ldots A_n}e_{B_1\ldots B_n} = (-1)^{\sum_{k<l} B_kA_l}e_{(A_1\ldots A_n)\oplus(B_1\ldots B_n)}. \tag{8}$$

Indeed, for two arbitrary strings of bits we have

$$\begin{bmatrix} \longleftarrow\ B_1B_2\ldots B_n \\ A_1A_2\ldots A_n \end{bmatrix} \mapsto (-1)^D \begin{bmatrix} B_1B_2\ldots B_n \\ A_1A_2\ldots A_n \end{bmatrix} \tag{9}$$

where

$$D = B_1(A_2 + \cdots + A_n) + B_2(A_3 + \cdots + A_n) + \cdots + B_{n-1}A_n = \sum_{k<l} B_kA_l. \tag{10}$$

We conclude that the map

$$(A_1\ldots A_n) \times (B_1\ldots B_n) \mapsto (A_1\ldots A_n) \oplus (B_1\ldots B_n) \tag{11}$$

has GA projective (i.e. up to a sign) representation by means of (8). Accordingly, the geometric product is a representation of Kanerva's binding at the level of GA.

Chunking can be represented in GA similarly to what is done in HRRs, that is, by ordinary addition. To see this consider

$$\begin{aligned}
X &= e_{A_1\ldots A_n}e_{B_1\ldots B_n} + e_{C_1\ldots C_n}e_{D_1\ldots D_n}, && (12)\\
Y &= (-1)^{\sum_{k<l} A_kA_l}e_{A_1\ldots A_n}X = e_{B_1\ldots B_n} + (-1)^{\sum_{k<l} A_kA_l}e_{A_1\ldots A_n}e_{C_1\ldots C_n}e_{D_1\ldots D_n} && (13)\\
&= e_{B_1\ldots B_n} \pm e_{(A_1\ldots A_n)\oplus(C_1\ldots C_n)\oplus(D_1\ldots D_n)} && (14)\\
&= e_{B_1\ldots B_n} + \text{noise}. && (15)
\end{aligned}$$

Until now the procedure is similar to what is done in BSC and HRRs.

An analogue of clean-up memory can be constructed in various ways. One possibility is to make sure that fillers, $e_{B_1\ldots B_n}$ etc. are orthogonal to the noise term. For example, let us take the fillers of the form $e_{B_1\ldots B_k 0\ldots 0}$, where the first $k \ll n$ bits are selected at random, but the remaining $n-k$ bits are all 0. Let the roles be taken, as in Kanerva's BSC, with *all* the bits generated at random. The term $e_{(A_1\ldots A_n)\oplus(C_1\ldots C_n)\oplus(D_1\ldots D_n)}$ will with high probability contain at least one $B_j = 1$, $k < j \leq n$, and thus will be orthogonal to the fillers. The clean-up memory will consist of vectors with $B_j = 0$, $k < j \leq n$, i.e of the filler form.

Tle final step is performed again in analogy to HRRs. We compute a scalar product between Y and the elements of clean-up memory. Depending on our needs we can play with different scalar products, or with the so-called contractions [27]. The richness of GA opens here several possibilities.

IV. CARTAN REPRESENTATION

In this section we give an explicit matrix representation of GA. We begin with Pauli's matrices

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{16}$$

GA of a plane is represented as follows: $1 = 2 \times 2$ unit matrix, $e_1 = \sigma_1$, $e_2 = \sigma_2$, $e_{12} = \sigma_1\sigma_2 = i\sigma_3$. Alternatively, we can write $e_{00} = 1$, $e_{10} = \sigma_1$, $e_{01} = \sigma_2$, $e_{11} = i\sigma_3$, and

$$\alpha_{00}e_{00} + \alpha_{10}e_{10} + \alpha_{01}e_{01} + \alpha_{11}e_{11} = \begin{pmatrix} \alpha_{00} + i\alpha_{11} & \alpha_{10} - i\alpha_{01} \\ \alpha_{10} + i\alpha_{01} & \alpha_{00} - i\alpha_{11} \end{pmatrix}. \tag{17}$$

This is equivalent to encoding $2^2 = 4$ real numbers into two complex numbers.

In 3-dimensional space we have $1 = 2 \times 2$ unit matrix, $e_1 = \sigma_1$, $e_2 = \sigma_2$, $e_3 = \sigma_3$, $e_{12} = \sigma_1\sigma_2 = i\sigma_3$, $e_{13} = \sigma_1\sigma_3 = -i\sigma_2$, $e_{23} = \sigma_2\sigma_3 = i\sigma_1$, $e_{123} = \sigma_1\sigma_2\sigma_3 = i$.

Now the representation of

$$\sum_{ABC=0,1} \alpha_{ABC}e_{ABC} = \begin{pmatrix} \alpha_{000} + i\alpha_{111} + \alpha_{001} + i\alpha_{110}, & \alpha_{100} + i\alpha_{011} - i\alpha_{010} - \alpha_{101} \\ \alpha_{100} + i\alpha_{011} + i\alpha_{010} + \alpha_{101}, & \alpha_{000} + i\alpha_{111} - \alpha_{001} - i\alpha_{110} \end{pmatrix} \tag{18}$$

is equivalent to encoding $2^3 = 8$ real numbers into 4 complex numbers.

An arbitrary n-bit record can be encoded into the matrix algebra known as Cartan's representation of Clifford algebras [28]:

$$e_{2k} = \underbrace{\sigma_1 \otimes \cdots \otimes \sigma_1}_{n-k} \otimes \sigma_2 \otimes \underbrace{1 \otimes \cdots \otimes 1}_{k-1}, \tag{19}$$

$$e_{2k-1} = \underbrace{\sigma_1 \otimes \cdots \otimes \sigma_1}_{n-k} \otimes \sigma_3 \otimes \underbrace{1 \otimes \cdots \otimes 1}_{k-1}. \tag{20}$$

In practical calculations it is convenient to work with the tensor product implemented by means of the "drag-and-drop" rule: For arbitrary matrices A and B (not necessarily square, and possibly of different dimensions)

$$A \otimes B = \begin{pmatrix} a_{11} & \ldots & a_{1j} \\ \vdots & \ddots & \vdots \\ a_{i1} & \ldots & a_{ij} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & \ldots & b_{1s} \\ \vdots & \ddots & \vdots \\ b_{r1} & \ldots & b_{rs} \end{pmatrix} = \begin{pmatrix} a_{11}B & \ldots & a_{1j}B \\ \vdots & \ddots & \vdots \\ a_{i1}B & \ldots & a_{ij}B \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & \ldots & a_{1j}b_{1s} \\ \vdots & \ddots & \vdots \\ a_{i1}b_{r1} & \ldots & a_{ij}b_{rs} \end{pmatrix}. \tag{21}$$

This representation of $\otimes$ can be used to show that all e_k given by Cartan's representation are matrices of zero trace.

V. PAT SMITH REVISITED

So let us return to the example from Sec. 2. For simplicity take $n = 4$ so that we can choose the representation

$$\left.\begin{aligned} e_{\text{Pat}} &= e_{1100}, \\ e_{\text{male}} &= e_{1000}, \\ e_{66} &= e_{0100}, \end{aligned}\right\} \text{fillers} \tag{22}$$

$$\left.\begin{array}{rcl} e_{\rm name} &=& e_{1010}, \\ e_{\rm sex} &=& e_{0111}, \\ e_{\rm age} &=& e_{1011}. \end{array}\right\} \text{roles} \tag{23}$$

The fillers have only the first two bits selected at random, the last two are 00. The roles are numbered by randomly selected strings of bits.

The explicit matrix representations are:

$$\begin{aligned} e_{\rm Pat} &= e_{1100} = e_1 e_2 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2) = 1 \otimes 1 \otimes 1 \otimes \sigma_3\sigma_2 = 1 \otimes 1 \otimes 1 \otimes (-i\sigma_1) \quad (24)\\ e_{\rm male} &= e_{1000} = e_1 = \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3 \quad (25)\\ e_{66} &= e_{0100} = e_2 = \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2 \quad (26)\\ e_{\rm name} &= e_{1010} = e_1 e_3 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1) \\ &= 1 \otimes 1 \otimes \sigma_1\sigma_3 \otimes \sigma_3 = 1 \otimes 1 \otimes (-i\sigma_2) \otimes \sigma_3 \\ e_{\rm sex} &= e_{0111} = e_2 e_3 e_4 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1)(\sigma_1 \otimes \sigma_1 \otimes \sigma_2 \otimes 1) \\ &= (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2)(1 \otimes 1 \otimes \sigma_3\sigma_2 \otimes 1) = \sigma_1 \otimes \sigma_1 \otimes \sigma_1\sigma_3\sigma_2 \otimes \sigma_2 = \sigma_1 \otimes \sigma_1 \otimes (-i1) \otimes \sigma_2, \quad (27)\\ e_{\rm age} &= e_{1011} = e_1 e_3 e_4 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1)(\sigma_1 \otimes \sigma_1 \otimes \sigma_2 \otimes 1) \\ &= \sigma_1 \otimes \sigma_1 \otimes (-i1) \otimes \sigma_3 \quad (28) \end{aligned}$$

The whole record

$$\begin{aligned} \mathbf{PSmith} &= \alpha e_{\rm name} e_{\rm Pat} + \beta e_{\rm sex} e_{\rm male} + \gamma e_{\rm age} e_{66} \\ &= \alpha e_{1010} e_{1100} + \beta e_{0111} e_{1000} + \gamma e_{1011} e_{0100} \\ &= \alpha(-1)^2 e_{(1010)\oplus(1100)} + \beta(-1)^3 e_{(0111)\oplus(1000)} + \gamma(-1)^2 e_{(1011)\oplus(0100)} \\ &= \alpha e_{0110} - \beta e_{1111} + \gamma e_{1111} \end{aligned}$$

The fact that the last two terms are linearly dependent is a consequence of too small dimensionality of our binary strings (four bits, whereas in realistic cases Kanerva suggested 10^4 bit strings). This is the price we pay for simplicity of the example. Decoding the name involves two steps. First

$$\begin{aligned} e_{\rm name}\mathbf{PSmith} &= e_{1010}\mathbf{PSmith} = e_{1010}\big[\alpha e_{0110} - \beta e_{1111} + \gamma e_{1111}\big] \\ &= \alpha(-1)^1 e_{(1010)\oplus(0110)} - \beta(-1)^2 e_{(1010)\oplus(1111)} + \gamma(-1)^2 e_{(1010)\oplus(1111)} \\ &= -\alpha e_{1100} - \beta e_{0101} + \gamma e_{0101} \\ &= -\alpha e_{\rm Pat} \underbrace{-\beta e_{0101} + \gamma e_{0101}}_{\rm noise} = \mathbf{Pat}'. \end{aligned} \tag{29}$$

It remains to employ clean-up memory. But this is easy since the noise is perpendicular to $e_{\rm Pat}$. We only have to project on the set spanned by the fillers, and within this set check which element is closest to the cleaned up $\mathbf{Pat}'$.

Cartan's represenation allows to define scalar product in GA by means of the trace. We therefore compare scalar products between $\mathbf{Pat}'$ and elements of clean-up memory. The only nonzero scalar product is

$$\begin{aligned} \langle e_{\rm Pat}|\mathbf{Pat}'\rangle &= {\rm Tr}\left(e_{1100}\big[-\alpha e_{1100} - \beta e_{0101} + \gamma e_{0101}\big]\right) \\ &= {\rm Tr}\left(-\alpha e_1 e_2 e_1 e_2 - \beta e_1 e_2 e_2 e_4 + \gamma e_1 e_2 e_2 e_4\right) \\ &= {\rm Tr}\left(\alpha 1 - \beta e_1 e_4 + \gamma e_1 e_4\right) = 16\alpha. \end{aligned} \tag{30}$$

Indeed

$$\begin{aligned} e_1 e_4 &= e_{1000} e_{0001} = e_{1001} = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_2 \otimes 1) = 1 \otimes 1 \otimes (i\sigma_3) \otimes \sigma_3 \\ &= {\rm diag}\,(i,-i,-i,i,i,-i,-i,i,i,-i,-i,i,i,-i,-i,i) \end{aligned} \tag{31}$$

has zero trace.

VI. CONCLUSIONS

BSC represented at the level of GA maintain the essential element of the original construction, i.e. binding by means of XOR. However, instead of the straightforward map

$$(\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x} \oplus \mathbf{y} \tag{32}$$

we rather have the "exponential map" $\mathbf{x} \mapsto e_{\mathrm{x}}$ satisfying $e_{\mathrm{x}} e_{\mathrm{y}} = \pm e_{\mathrm{x} \oplus \mathrm{y}}$. Another difference is in mathematical implementation of chunking. Unbinding produces a noise term which, with high probability, is orthogonal to the original filler. In this respect the construction is analogous to error correcting linear codes. As opposed to tensor product representations, and similarly to BSC and HRRs, binding performed by means of geometric product does not increase dimensions.

---

[1] T. Plate, Distributed representations, in *Encyclopedia of Cognitive Science* (Nature Publishing Group, 2003).
[2] J. B. Pollack, Recursive distributed representations, Artificial Intelligence **46**, 77-105 (1990).
[3] P. Smolensky, Tensor product variable binding and the representation of symbolic structures in connectionist systems, Artificial Intelligence **46**, 159-216 (1990).
[4] T. Plate, Holographic reduced representations, IEEE Transactions on Neural Networks **6**, 623-641 (1995).
[5] T. Plate, *Holographic Reduced Representation: Distributed Representation for Cognitive Structures* (CSLI Publications, Stanford, 2003).
[6] P. Kanerva, Binary spatter codes of ordered k-tuples, *Artificial Neural Networks–ICANN Proceedings*, Lecture Notes in Computer Science vol. 1112, pp. 869-873, C. von der Malsburg *et al.* (Eds.) (Springer, Berlin, 1996).
[7] P. Kanerva, Fully distributed representation, *Proc. 1997 Real World Computing Symposium (RWC'97, Tokyo)*, pp. 358-365 (Real World Computing Partnership, Tsukuba-City, Japan, 1997).
[8] P. Kanerva, Large patterns make great symbols: An example of learning from example, *Hybrid Neural Systems*, pp. 194-203 (1998).
[9] L. Gabora and D. Aerts, Contextualizing concepts using a mathematical generalization of the quantum formalism, Journal of Experimental and Theoretical Artificial Intelligence **14**, 327 (2002).
[10] D. Aerts and M. Czachor, Quantum aspects of semantic analysis and symbolic artificial intelligence, Journal of Physics A **37**, L123-L132 (2004).
[11] D. Aerts and L. Gabora, A theory of concepts and their combinations (I): The structure of sets of contexts and properties, Kybernetes **34**, 167-191 (2005).
[12] D. Aerts and L. Gabora, A theory of concepts and their combinations (II): A Hilbert space representation, Kybernetes **34**, 192-221 (2005).
[13] D. Aerts, M. Czachor, and B. D'Hooghe, Towards a quantum evolutionary scheme: Violating Bell's inequalities in language, *Evolutionary Epistemology, Language, and Culture: A non-adaptionist, systems theoretical approach*, Theory and Decision Library A, vol. 39, N. Gontier, J. P. Bendegem, and D. Aerts (Eds.) (Springer, Berlin, 2006).
[14] D. Widdows, *Geometry and Meaning* (CSLI Publications, Stanford, 2004).
[15] D. Widdows and S. Peters, Word vectors and quantum logic: Experiments with negation and disjunction, *Proc. Mathematics and Language (Bloomington, June 2003)*, R. T. Oehrle and J. Rogers (Eds.), pp. 141-154 (2003).
[16] D. Widdows and M. Higgins, Geometric ordering of concepts, logical disjunction, and learning by induction, *Compositional Connectionism in Cognitive Science*, AAAI Fall Symposium Series, Washington, DC, October 22-24, 2004.
[17] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics* (Reidel, Dordrecht, 1984).
[18] G. Sommer (ed.), *Geometric Computing with Clifford Algebras* (Springer, Berlin 2001).
[19] L. Dorst, C. J. L. Doran, J. Lasenby (eds.), *Applications of Geometric Algebra in Computer Science and Engineering* (Birkhauser, Boston, 2002).
[20] J. Lasenby, A. N. Lasenby, C. J. L. Doran, and W. J. Fitzgerald, New geometric methods for computer vision, International Journal of Computer Vision **36**, 191-213 (1998).
[21] E. Bayro-Corrochano, K. Danilidis, and G. Sommer, Motor algebra for 3D kinematics: The case of the hand-eye calibration, Journal of Mathematical Imaging and Vision **13**, 79-100 (2000).
[22] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought* (Oxford University Press, Oxford, 2003).
[23] S. Somaroo, D. G. Cory, and T. F. Havel, Expressing the operations of quantum computing in multiparticle geometric algebra, Physics Letters A **240**, 1-7 (1998).
[24] M. Van den Nest, J. Dehaene, and B. De Moor, The invariants of the local Clifford group, Physical Review A **71**, 022310 (2005).
[25] M. Van den Nest, J. Dehaene, and B. De Moor, On local unitary versus local Clifford equivalence of stabilizer states, Physical Review A **71**, 062323 (2005).
[26] M. Van den Nest, J. Dehaene, and B. De Moor, Finite set of invariants to characterize local Clifford equivalence of stabilizer states, Physical Review A **72**, 014317 (2005).
[27] L. Dorst, The inner products of geometric algebra, in [19], pp. 37-48.
[28] P. Budinich and A. Trautman, *The Spinorial Chessboard* (Springer, Berlin, 1988).
[29] E. J. Bayro-Corrochano, Geometric neural computing, IEEE Transactions on Neural Networks **12**, 968-986 (2001).
[30] J. Neumann, Learning the systematic transformation of holographic reduced representations, Cognitive Systems Research **3**, 227-353 (2002).
[31] D. Hestenes, Reforming the mathematical language of physics, American Journal of Physics **71**, 104-121 (2003).