



Geometric analogue of holographic reduced representation

Diederik Aerts^{a,b}, Marek Czachor^{a,b,c,*}, Bart De Moor^d

^a Centrum Leo Apostel (CLEA), Vrije Universiteit Brussel, 1050 Brussels, Belgium

^b Foundations of the Exact Sciences (FUND), Vrije Universiteit Brussel, 1050 Brussels, Belgium

^c Katedra Fizyki Teoretycznej i Informatyki Kwantowej, Politechnika Gdańska, 80-952 Gdańsk, Poland

^d ESAT-SCD, Katholieke Universiteit Leuven, 3001 Leuven, Belgium

ARTICLE INFO

Article history:

Received 1 February 2008

Received in revised form

10 January 2009

Available online 28 March 2009

Keywords:

Distributed representations

Quantum algorithms

Binding problem

Clifford algebras

Holographic models

Matrix memories

ABSTRACT

Holographic reduced representations (HRRs) are distributed representations of cognitive structures based on superpositions of convolution-bound n -tuples. Restricting HRRs to n -tuples consisting of ± 1 , one reinterprets the variable binding as a representation of the additive group of binary n -tuples with addition modulo 2. Since convolutions are not defined for vectors, the HRRs cannot be directly associated with geometric structures. Geometric analogues of HRRs are obtained if one considers a projective representation of the same group in the space of blades (geometric products of basis vectors) associated with an arbitrary n -dimensional Euclidean (or pseudo-Euclidean) space. Switching to matrix representations of Clifford algebras, one can always turn a geometric analogue of an HRR into a form of matrix distributed representation. In typical applications the resulting matrices are sparse, so that the matrix representation is less efficient than the representation directly employing the rules of geometric algebra. A yet more efficient procedure is based on 'projected products', a hierarchy of geometrically meaningful n -tuple multiplication rules obtained by combining geometric products with projections on relevant multivector subspaces. In terms of dimensionality the geometric analogues of HRRs are in between holographic and tensor-product representations.

© 2009 Elsevier Inc. All rights reserved.

1. Introduction

Reduced representations of cognitive structures are based essentially on two operations (binding and superposing) whose algebraic realizations vary from model to model. In Hinton (1990), where matrices representing roles act on vectors representing fillers, binding corresponds to matrix multiplication and superposition to vector addition. In tensor representations (Smolensky, 1990), roles and fillers are represented by vectors which are bound by means of tensor products. The resulting simple tensors are superposed by addition. In holography-inspired memory models (Borsellino & Poggio, 1973; Gabor, 1968; Liepa, 1977; Metcalfe, 1991; Metcalfe-Eich, 1982; Murdock, 1982; Slack, 1984; Willshaw, 1989), binding is represented by convolution. Replacing tensor products by circular convolutions, one arrives at holographic reduced representations (HRRs) (Plate, 1995, 2003). Restricting frequency-domain HRRs to a subspace and switching to appropriately defined 'logarithmic' variables, one obtains binary spatter codes (BSC) (Kanerva, 1996, 1997, 1998), with binary

strings of length n bound by n -dimensional sums mod 2 and superpositions modeled by majority-rule sums. Finally, in quantum computation (QC) (Nielsen & Chuang, 2000), bits are bound into n -bit numbers by means of tensor products of two-dimensional complex vectors called qubits. QC is mathematically similar to tensor-product reduced representations, but differences occur at interpretational levels (Aerts & Czachor, 2004).

Convolutions may be regarded as basis-dependent lossy compressions of the tensor product. The degree of compression can be estimated by means of dimensional analysis. In particular, circular convolutions occurring in HRRs map pairs of n -tuples into n -tuples. In contrast, the ordinary convolution of two n -tuples is a $(2n - 1)$ -tuple, and an analogous tensor product would produce a n^2 -tuple. These facts explain the efficiency and usefulness of HRRs in applications (Plate, 2003).

The simplest way of understanding properties of the circular convolution \otimes is to use its Fourier-space ('frequency domain') form, where

$$(x_1, \dots, x_n) \otimes (y_1, \dots, y_n) = (x_1 y_1, \dots, x_n y_n) \quad (1)$$

(later, whenever we employ (1) we will tend to write the n -tuple entries with hats, $\hat{x}_1, \dots, \hat{y}_n$, in order to remind us that (1) is the definition traditionally associated with the frequency domain).

* Corresponding author at: Katedra Fizyki Teoretycznej i Informatyki Kwantowej, Politechnika Gdańska, 80-952 Gdańsk, Poland.

E-mail address: mzczachor@pg.gda.pl (M. Czachor).

All the essential features of HRRs are encoded in this simple formula. For example, comparing \otimes with the tensor product \otimes ,

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \otimes \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 y_1 \\ \vdots \\ x_1 y_n \\ \vdots \\ x_n y_1 \\ \vdots \\ x_n y_n \end{pmatrix}, \quad (2)$$

we can see that \otimes is equivalent to the ‘diagonal part’ of \otimes , a fact explaining why \otimes can be regarded as a (very lossy) compression of \otimes . Definition (1) is simultaneously an example of how to bind two n -tuples x and y (a role x with a filler y , say); a sum of several such expressions would be an example of a reduced representation à la HRR. Similarly, (2) would lead to the Smolensky (1990) tensor-product binding, while a sum of several such terms would be a tensor-product reduced representation.

The tensor-product representation is mathematically very close to the diadic-product representations used in linear matrix memories (Anderson, 1972; Kohonen, 1972),

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \star \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} (y_1 \dots y_n) = \begin{pmatrix} x_1 y_1 & \dots & x_1 y_n \\ \vdots & \ddots & \vdots \\ x_n y_1 & \dots & x_n y_n \end{pmatrix}. \quad (3)$$

A linear combination of terms such as (3) is the reduced representation known in artificial neural network theory as the weight matrix. Circular convolution thus can be regarded as a lossy compression of \star obtained by removing the off-diagonal elements,

$$\begin{pmatrix} x_1 y_1 & \dots & x_1 y_n \\ \vdots & \ddots & \vdots \\ x_n y_1 & \dots & x_n y_n \end{pmatrix} \rightarrow \begin{pmatrix} x_1 y_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & x_n y_n \end{pmatrix}. \quad (4)$$

The diadic product is defined only for pairs of n -tuples x and y . As opposed to \otimes and \otimes it does not naturally generalize to problems where binding of more than two terms is needed.

In spite of what one can often read in the literature, a convolution of two vectors is not well defined. Thus having two vectors, that is, geometric objects, we cannot unambiguously identify a geometric object corresponding to their circular convolution: the compression (4) is basis dependent.

This seems to be a drawback, at least at the conceptual level. Geometry of some sort is at the roots of visualization, and visualization seems important for mathematical understanding and proving (Gärdenfors, 2003; Penrose, 1994; Widdows, 2004; Widdows & Higgins, 2004). Hence the question: Is it possible to replace circular convolution by something similar but geometrically meaningful? If so, is there a relation to HRRs? Moreover, can we make it into a practically implementable alternative to HRRs or similar known constructions?

We will argue that the most natural choice is to replace tensor products by geometric products (Dorst, Fontijne & Mann, 2007), and not by circular convolutions. Geometric products, similarly to circular convolutions, preserve dimensionality at the level of multivectors. Multivectors are superpositions of blades, geometric-product analogues of simple tensors. Geometric products are also ‘exponents’, in a sense that will be made precise later, of n -dimensional sums mod 2. In consequence, geometric-product binding is in a unique relation to the binding employed in BSC, and the latter is related to the standard HRR by a kind of exponential map.

Coding based directly on geometric products was recently applied to QC (Aerts & Czachor, 2007, 2008; Czachor, 2007) (a somewhat less direct way of linking geometric products with QC was used earlier in Somaroo, Cory and Havel (1998)). As it turned out, all quantum algorithms of the standard formalism have geometric analogues. However, as opposed to standard QC that requires quantum mechanical implementations, the formalism based on geometric algebra (GA) requires geometry and not quantum mechanics. In principle, any system involving some geometry (Euclidean or not) is a candidate for implementation of a quantum algorithm. Systems where holographic or tensor-product representations are applicable might, therefore, at least in principle, perform quantum algorithms.

The concept of GA is not new – it appeared in the 19th century works of Grassmann (1877) and Clifford (1878) – but geometric insights behind GA were forgotten for almost a century. At the end of 1960s the subject was revived with the works of Hestenes (1966). Today the Hestenes system (Hestenes & Sobczyk, 1984; Hestenes, 1986, 2003) has found applications to topics as diverse as black holes, cosmology, quantum mechanics, quantum field theory, supersymmetry, beam dynamics, computer vision, robotics, protein folding, computer aided design, and recently, quantum computation (cf. Baylis (1996), Sommer (2001), Dorst, Doran and Lasenby (2002), Pavšič (2001), Doran and Lasenby (2003), Lasenby, Lasenby, Doran, and Fitzgerald (1996) and Bayro-Corrochano, Danilidis and Sommer (2000)). In the context of cognitive science one should mention the GA formulation of neural computing (Bayro-Corrochano, 2001) and the Hestenes treatment of invariant body kinematics with applications to neurogeometry (Hestenes 1994a; 1994b). The present paper is a step towards a direct geometric formulation of distributed representations of cognitive structures.

The paper is organized as follows. Section 2 introduces GA and its basic constructions (multivectors, invertibility, coding based on blades) and explains why GA naturally formalizes relations between geometric objects. In Section 3 we compare the geometric product with circular convolution and explain why the latter cannot be interpreted in geometric terms. In Section 4 we discuss the Fourier-space convolution algebra and show that it is in a one-to-one relation with the algebra of commuting matrices. In Section 5 we explain why variable binding in a Fourier-space HRR is a representation, in group-theoretic sense, of the group $Z_2 \times \dots \times Z_2$. Then, in Section 6 we show that blades form a projective representation the same group and thus convolution binding is naturally represented in GA. We illustrate the new reduced representation in Section 7 by reformulating the example Kanerva gave as an illustration of his BSC. Section 8 introduces a projected product, a useful operation that allows us to generate only those parts of multivectors we are interested in. In Section 9 we concentrate on the issue of dimensionality of n -tuples in HRRs, tensor representations, and geometric analogues of HRRs, especially with decoding that involves the projected products. Finally, in Section 10 we reformulate the Kanerva example by means of a matrix representation of GA.

Comparison of the proposed reduced representation with HRRs and BSC will be done in a future work, but preliminary results are already available in Patyk (in print).

2. Algebraic representation of geometric relations

Consider the following set of relations \approx involving two-dimensional basic shapes:

$$\begin{aligned} - - &\approx || \approx \square \square \approx 1 \\ - | &\approx | - \approx \square \\ - \square &\approx \square - \approx | \\ | \square &\approx \square | \approx - \end{aligned}$$

One can think of them in at least two categories. One is simply a category of *understanding* relations between one- and two-dimensional objects: A square is formed from two orthogonal segments, a segment and a square imply which segment is missing, 1 means identity, ... Another way of looking at these relations, more in the spirit of Grassmann and Clifford, would be in terms of an *algebra* of geometric objects if associativity is assumed and 1 is treated as a neutral element. Associativity then implies, for example, $- | - \approx \square - \approx - \square \approx |$.

The next step is to ask for higher-dimensional generalization and inclusion of *orientation*: Oriented line segments are vectors, plane segments have 'sides', the relations between them will include a positive or negative sign, and we can also add cubes (having 'insides' and 'outsides'), their walls, and even higher-dimensional structures.

Orientation makes the algebra noncommutative

$$\begin{aligned} \rightarrow \uparrow &\approx -\square \\ \uparrow \rightarrow &\approx +\square \\ \rightarrow \rightarrow &\approx \uparrow \uparrow \approx 1 \end{aligned} \tag{5}$$

if we assume that 'first up then right' generates the right-handed orientation, opposite to the left-handed 'first right then up'. Adding both relations we find $\rightarrow \uparrow + \uparrow \rightarrow \approx 0$. Taking three arrows and using associativity we obtain another, similar rule: $\square \rightarrow + \rightarrow \square \approx 0$. The proof goes as follows:

$$\square \rightarrow \approx \rightarrow \uparrow \rightarrow \approx \rightarrow (-\square) \approx - \rightarrow \square \approx \leftarrow \square$$

However, we cannot simultaneously assume $\rightarrow \rightarrow \approx \uparrow \uparrow \approx 1$ and $\square \square \approx 1$. Indeed, anticommutativity of $\rightarrow \uparrow \approx - \uparrow \rightarrow$ implies

$$\square \square \approx \rightarrow \uparrow \rightarrow \uparrow \approx - \rightarrow \rightarrow \uparrow \uparrow \approx -11 \approx -1 \tag{6}$$

so we have to decide at which level to define the algebra. One can also think of this example as a hierarchy of geometric structures with increasing dimensions and different metric signatures. The first level is the pair $\{\rightarrow, \uparrow\}$ and the Euclidean-space signature is $(+, +)$. The second level is $\{1, \rightarrow, \uparrow, \square\}$ and the signature is $(+, +, +, -)$, known from space-time pseudo-Euclidean geometry (the three pluses for the three spatial directions and the minus corresponding to the time direction). So, the relations (5) and (6) can be understood to mean identities (' \rightarrow is the same as \rightarrow ', ' \square is the same as \square ') but at different hierarchical levels.

The type of construction we have just outlined led Grassmann and Clifford to the algebra based on the concise formula

$$b_k b_l + b_l b_k = 2\delta_{kl} \mathbf{1}. \tag{7}$$

Here the b_s denote orthonormal basis vectors in some n -dimensional real Euclidean space, $\mathbf{1}$ is the neutral element of the algebra, and δ_{kl} is the Kronecker delta. The two-dimensional (2D) plane example is reconstructed from (7) if $b_1 = \rightarrow, b_2 = \uparrow, b_1 b_2 = \square = b_{12}$.

The algebra (7) is known as the Clifford algebra and may be regarded as the grammar of GA. It refers to a concrete basis, but can be reformulated in a basis-free way. Indeed, consider two vectors $x = \sum_{k=1}^n x_k b_k$ and $y = \sum_{k=1}^n y_k b_k$. Their geometric product reads

$$xy = \underbrace{\sum_{k=1}^n x_k y_k \mathbf{1}}_{x \cdot y} + \underbrace{\sum_{k < l} (x_k y_l - y_k x_l) b_k b_l}_{x \wedge y}$$

The geometric product xy is a sum of two terms. The *scalar* $x \cdot y = y \cdot x$ is known as the *inner product*. The *bivector* $x \wedge y = -y \wedge x$ is the *outer product*. In three dimensions the length of $x \wedge y$ represents the area of the parallelogram spanned by x and y .

In arbitrary dimension the bivector $x \wedge y$ represents an oriented plane segment. Grassmann and Clifford introduced the geometric

product by means of the basis-independent formula involving the *multivector*

$$xy = x \cdot y + x \wedge y \tag{8}$$

which implies (7) when restricted to an orthonormal basis. The inner and outer products can be defined directly from xy :

$$x \cdot y = \frac{1}{2}(xy + yx),$$

$$x \wedge y = \frac{1}{2}(xy - yx).$$

The most ingenious element of (8) is that it adds two apparently different objects: a scalar and a plane element. This seems 'wrong' but is precisely what happens when we speak of complex numbers or extend space and time to space-time. Apparently, the person to be blamed for the fact that multivectors may nowadays seem weird is Gibbs (1906), who was more famous at the time than Grassmann or Clifford, and spoiled their work by separating the geometric product into two separate operations – losing associativity and invertibility, as we shall see shortly.

Geometric interpretation and visualization of multivectors can be formulated in various ways, and various interpretations can be found in the literature. Perhaps the easiest way of getting used to thinking in GA terms is to browse through the websites devoted to GA.¹ The approach we found useful for multi-bit problems of QC was a representation in terms of directed colored polylines (Aerts & Czachor, 2008).

Yet another interpretation of multivectors was recently proposed in Aerts, Czachor and Orłowski (2009). Directed magnitudes were interpreted as *colors*, while blades were shown to correspond to 'parts' of different dimensionality in cubes of appropriate dimension. For example, the multivector $x_0 \mathbf{1} + x_1 b_1$ in one dimension is a unit segment and the colors x_0, x_1 correspond to, respectively, its endpoints and interior (or, better, only one endpoint and the interior); in two dimensions, multivectors are unit squares whose blades are corners, sides, and interiors (i.e. one corner, two sides, and the interior, all of possibly different colors); in three dimensions we have one corner, three edges, three walls, and the interior of a cube. The idea was applied in Aerts, Czachor and Orłowski (2009) to give a 3D interpretation of a 3-bit quantum teleportation algorithm (Bennett et al., 1993).

Geometric product for vectors x, y , and z can be axiomatically defined by the following rules:

$$(xy)z = x(yz),$$

$$x(y + z) = xy + xz,$$

$$(x + y)z = xz + yz,$$

$$x^2 = |x|^2,$$

where $|x|$ is a positive scalar called the magnitude of x . The rules imply that $x \cdot y$ must be a scalar since

$$xy + yx = |x + y|^2 - |x|^2 - |y|^2.$$

GA allows us to speak of inverses of vectors: $x^{-1} = x/|x|^2$. x is invertible (i.e. possesses an inverse) if its magnitude is nonzero. The geometric product of an arbitrary number of invertible vectors is also invertible. The possibility of inverting all nonzero-magnitude vectors is perhaps the most important difference between GA and tensor or convolution algebras.

Geometric products of *different* basis vectors

$$b_{k_1 \dots k_j} = b_{k_1} \dots b_{k_j},$$

$k_1 < \dots < k_j$, are called basis blades (or just blades). In n -dimensional (pseudo-)Euclidean space there are 2^n different blades. This can be seen as follows. Let $\{x_1, \dots, x_n\}$ be a sequence

¹ One can start, say, with <http://www.lomont.org/Math/GeometricAlgebra/Papers.php>, and browse through the links.

of bits. Blades in an n -dimensional space can be written as

$$c_{x_1 \dots x_n} = b_1^{x_1} \dots b_n^{x_n}$$

where $b_0^0 = \mathbf{1}$, which shows that blades are in a one-to-one relation with n -bit numbers. This observation is at the root of the GA reformulation of QC introduced in Aerts and Czachor (2007). A general multivector is a linear combination of blades,

$$\psi = \sum_{x_1 \dots x_n=0}^1 \psi_{x_1 \dots x_n} c_{x_1 \dots x_n}, \tag{9}$$

with real ² coefficients $\psi_{x_1 \dots x_n}$.

An inverse of a multivector is a well defined notion but not all multivectors are invertible. To find an inverse of a multivector is not an entirely trivial task in general. It resembles an analogous problem of inverting matrices. But all blades and geometric products of invertible vectors are invertible.

3. Circular convolution vs. geometric product

We have mentioned in Section 1 that convolutions are not defined on vectors but only on n -tuples. Let us explain this statement in more detail on the example of circular convolution. Circular convolution $x \otimes y$ of the n -tuples $x = (x_0, \dots, x_{n-1})$, $y = (y_0, \dots, y_{n-1})$ is defined as

$$(x \otimes y)_j = \sum_{k=0}^{n-1} x_k y_{j-k \bmod n}. \tag{10}$$

For pairs, formula (10) reads explicitly

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 y_0 + x_1 y_1 \\ x_0 y_1 + x_1 y_0 \end{pmatrix}. \tag{11}$$

Let us note that if we tried to interpret (11) in terms of vectors we would have to implicitly assume that the pairs on both sides of (11) correspond to the same basis (otherwise the formula would be completely ambiguous). So let us take two different bases $\{\mathbf{b}_0, \mathbf{b}_1\}$ and $\{\mathbf{b}'_0, \mathbf{b}'_1\}$, and two vectors \mathbf{x}, \mathbf{y} . Each of these vectors can be written in both bases:

$$\mathbf{x} = x_0 \mathbf{b}_0 + x_1 \mathbf{b}_1 = x'_0 \mathbf{b}'_0 + x'_1 \mathbf{b}'_1,$$

$$\mathbf{y} = y_0 \mathbf{b}_0 + y_1 \mathbf{b}_1 = y'_0 \mathbf{b}'_0 + y'_1 \mathbf{b}'_1.$$

Circular convolutions of vectors would be meaningful if in any two bases we would find

$$\begin{aligned} \mathbf{x} \otimes \mathbf{y} &= (x_0 y_0 + x_1 y_1) \mathbf{b}_0 + (x_0 y_1 + x_1 y_0) \mathbf{b}_1 \\ &= (x'_0 y'_0 + x'_1 y'_1) \mathbf{b}'_0 + (x'_0 y'_1 + x'_1 y'_0) \mathbf{b}'_1 \end{aligned}$$

which is not the case. Indeed, let us take the basis rotated by $\pi/2$ ($\mathbf{b}'_0 = \mathbf{b}_1, \mathbf{b}'_1 = -\mathbf{b}_0, x'_0 = x_1, x'_1 = -x_0, y'_0 = y_1, y'_1 = -y_0$)

$$\mathbf{x} = x'_0 \mathbf{b}'_0 + x'_1 \mathbf{b}'_1 = x_1 \mathbf{b}_1 + (-x_0)(-\mathbf{b}_0),$$

$$\mathbf{y} = y'_0 \mathbf{b}'_0 + y'_1 \mathbf{b}'_1 = y_1 \mathbf{b}_1 + (-y_0)(-\mathbf{b}_0),$$

implying

$$\begin{aligned} (x'_0 y'_0 + x'_1 y'_1) \mathbf{b}'_0 + (x'_0 y'_1 + x'_1 y'_0) \mathbf{b}'_1 &= (x_1 y_0 + x_0 y_1) \mathbf{b}_0 \\ &+ (x_0 y_0 + x_1 y_1) \mathbf{b}_1 \\ &\neq (x_0 y_0 + x_1 y_1) \mathbf{b}_0 + (x_0 y_1 + x_1 y_0) \mathbf{b}_1. \end{aligned}$$

² A complex structure, if needed, may be introduced by means of an additional bit without any need of complex numbers. This is how the ‘imaginary unit’ i was used in Czachor (2007) in the context of quantum gates. It must be stressed, however, that in the GA literature it is a tradition to replace i by a blade. For example, we have seen that in two dimensions $(b_1 b_2)^2 = -\mathbf{1}$, which explains why $b_1 b_2$ has properties analogous to i . Nevertheless, this construction does not properly work here (cf. the discussion in Aerts and Czachor (2008)) and we have to use a different representation of i , equivalent to a $\pi/2$ rotation in two dimensions. These subtleties are not important for the discussion given in the present paper.

In contrast, for the geometric product we find

$$\begin{aligned} \mathbf{x} \mathbf{y} &= (x_0 y_0 + x_1 y_1) \mathbf{1} + (x_0 y_1 - x_1 y_0) \mathbf{b}_0 \mathbf{b}_1 \\ &= (x'_0 y'_0 + x'_1 y'_1) \mathbf{1} + (x'_0 y'_1 - x'_1 y'_0) \mathbf{b}'_0 \mathbf{b}'_1, \end{aligned}$$

which does not depend on the choice of basis. Let us note that the difference between the geometric product and circular convolution boils down in this example to a single change of sign and reshuffling of components. To understand the latter property we have to bear in mind that the GA of a 2D plane is 2^2 -dimensional. A general multivector is here of the form $\psi = \alpha \mathbf{1} + \beta \mathbf{b}_0 + \gamma \mathbf{b}_1 + \delta \mathbf{b}_0 \mathbf{b}_1$. Our calculation, in terms of the 4-tuples $(\alpha, \beta, \gamma, \delta)$, reads

$$\begin{pmatrix} 0 \\ x_0 \\ x_1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ y_0 \\ y_1 \\ 0 \end{pmatrix} = \begin{pmatrix} x_0 y_0 + x_1 y_1 \\ 0 \\ 0 \\ x_0 y_1 - x_1 y_0 \end{pmatrix}.$$

There do exist matrix representations of GA (cf. Section 10). It is known, however, that matrix representations of GA are not the most efficient implementations of GA-based algorithms. The algorithms that work efficiently in practice are based on direct calculations performed in terms of the GA rules.³

4. Circular convolution in the Fourier space vs. matrix algebra

A general multivector (9) can be represented by the 2^n -tuple

$$(\psi_{0_1 \dots 0_n}, \dots, \psi_{1_1 \dots 1_n}).$$

The neutral element $\mathbf{1}$ corresponds in this notation to $(1, 0, \dots, 0)$.

Similarly, the neutral element of the \otimes -algebra of n -tuples is the n -tuple $I = (1, 0, \dots, 0)$. By definition, the \otimes -inverse x^{-1} of the n -tuple x satisfies $x^{-1} \otimes x = I$. An important map is the ‘involution’

$$(x^*)_j = x_{-j \bmod n}.$$

Let us now rewrite \otimes and $*$ in the Fourier space. The Fourier transform of (x_0, \dots, x_{n-1}) ,

$$\hat{x}_k = \sum_{l=0}^{n-1} x_l e^{-2\pi i k l / n},$$

satisfies

$$(\widehat{x \otimes y})_k = \hat{x}_k \hat{y}_k,$$

$$(\widehat{x^*})_k = \overline{\hat{x}_k},$$

$$(\widehat{x^* \otimes y})_k = \overline{\hat{x}_k} \hat{y}_k,$$

where $\overline{\hat{x}_k}$ denotes complex conjugation. The Fourier transform of $I = (1, 0, \dots, 0)$ is $\hat{I} = (1, 1, \dots, 1)$. Thus

$$(\widehat{x^{-1}})_k = \frac{1}{\hat{x}_k}.$$

x is not \otimes -invertible if any component of its Fourier transform is 0. This is why exact inverses are not used in standard HRR. Plate explains in Plate (2003) why x^* may be regarded as an approximate inverse of x , and why in the presence of noise – a generic situation in HRR – the application of exact inverses would lead to unstable algorithms. Only in the case of unitary x (i.e. such that $x^* = x^{-1}$,

³ See, for example, GABLE: Geometric Algebra Learning Environment, <http://staff.science.uva.nl/~leo/clifford/gable.htm>; GAIGEN: Geometric Algebra Implementation Generator, <http://sourceforge.net/projects/geigen>; GAViewer: Interactive Geometric Algebra with OpenGL Visualization, <http://www.science.uva.nl/ga/viewer/>; CLUCalc, <http://www.perwass.de/CLU/index.html>; GA Package for Maple, <http://www.mrao.cam.ac.uk/~maja1/software/GA/>; CLIFFORD – A Maple Package for Clifford Algebra Computations with Bigebra, <http://math.ntech.edu/rafal/>.

$|\hat{x}_k| = 1$) the approximate inverse is exact. In GA, in contrast to HRR, exact inverses of nonzero vectors always exist, do not lead to instabilities, and are easy to calculate since $x^{-1} = x/|x|^2$.

Circular convolution in the Fourier space is thus equivalent to the multiplication of diagonal matrices. Accordingly, the \otimes -inverse is the matrix inverse, and involution means Hermitian conjugation. The notion of \otimes -unitarity coincides with matrix unitarity. Fourier-space HRR involves binding represented by the matrix product of diagonal matrices, and superposition is performed by matrix addition. HRR is implicitly an operator procedure but involving only commuting operators. These operators are in general neither Hermitian nor unitary but have the property of being *normal*, i.e. they commute with their Hermitian conjugates.

Plate mentions in Plate (2003, Section 3.6.7) that commutativity can cause ambiguities, so certain noncommutative variants of \otimes may in principle be considered. For example, the combination of \otimes with permutations of components introduces noncommutativity for the price of associativity. Still another alternative mentioned in Plate (2003) is to work with vectors that can be written as matrices (i.e. with $n = m^2$, for some m) and use matrix multiplication. Apparently, this kind of reduced representation has not been studied in the literature so far.

Our claim is that the GA formalism is a natural noncommutative alternative to HRR, but to appreciate it we have to go deeper into the structure of the Fourier-space HRR.

5. From Fourier-space HRR to BSC

Consider a general HRR-type Fourier-space superposition of N diagonal matrices U_j : $\psi = \sum_{j=1}^N U_j$. Now let us restrict U_j to matrices of the form $U_j = e^{i\pi P_{x_j}} = (-1)^{P_{x_j}}$, where P_{x_j} are diagonal matrices whose only nonzero elements are equal to 1. In other words, the diagonal of P_{x_j} is a sequence of bits: $P_{x_j} = \text{diag}(x_{j,1}, \dots, x_{j,n})$, $x_{j,k} = 0, 1$. Such a P_{x_j} is a projector: $P_{x_j}^2 = P_{x_j}$. Under these restrictions the diagonal unitary matrix U_j has the diagonal consisting of $(-1)^{x_{j,k}} = \pm 1$. The next step is to consider the new diagonal matrix $\Psi = \text{sign}(\psi)$ defined via the spectral theorem from

$$\text{sign}(x) = \begin{cases} +1 & \text{for } x \geq 0 \\ -1 & \text{for } x < 0 \end{cases}$$

Ψ is again a unitary diagonal matrix whose only nonzero elements are equal to ± 1 , and hence can be written as $\Psi = e^{i\pi P_x} = (-1)^{P_x}$. $P_x = \text{diag}(x_1, \dots, x_n)$ is a new projector, i.e. a diagonal matrix with bits on the diagonal. In effect, we have produced a new binary sequence (x_1, \dots, x_n) from a collection of N binary sequences $(x_{j,1}, \dots, x_{j,n})$. The relevant formula (majority-rule summation) reads

$$x_k = \boxplus_{j=1}^N x_{j,k} = \Theta \left(\frac{1}{N} \sum_{j=1}^N x_{j,k} - \frac{1}{2} \right)$$

where

$$\Theta(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

is the Heaviside step function.

The final step leading to BSC is to treat each U_j as a product of two unitary diagonal matrices R_j and F_j , also consisting of pluses and minuses on diagonals,

$$\begin{aligned} \Psi &= \text{sign}(\psi) = (-1)^{P_x} = \text{sign} \left(\sum_{j=1}^N R_j F_j \right) \\ &= \text{sign} \left(\sum_{j=1}^N (-1)^{P_{x_j}} (-1)^{P_{y_j}} \right) = \text{sign} \left(\sum_{j=1}^N (-1)^{P_{x_j \oplus y_j}} \right). \end{aligned}$$

R_j, P_{x_j} represent roles, and F_j, P_{y_j} are fillers.

Since P_{x_j} and P_{y_j} are diagonal matrices with 0s and 1s on the diagonals, say, $P_{x_j} = \text{diag}(x_{j,1}, \dots, x_{j,n})$, $P_{y_j} = \text{diag}(y_{j,1}, \dots, y_{j,n})$, the map

$$(x_1, \dots, x_n) = \boxplus_{j=1}^N (x_{j,1}, \dots, x_{j,n}) \oplus (y_{j,1}, \dots, y_{j,n}) \quad (12)$$

is the thresholded majority-rule componentwise addition of n -dimensional sums mod 2 (n -dimensional exclusive alternatives, XORs).

Now consider a single bit X and a sequence of N bits (x_1, \dots, x_N) . Then the following distributivity of \oplus over \boxplus holds true:

$$X \oplus \left(\boxplus_{j=1}^N x_j \right) = \boxplus_{j=1}^N (X \oplus x_j). \quad (13)$$

(13) naturally generalizes to the n -dimensional variants of \boxplus and XOR. Eq. (12) is the BSC, a reduced representation where binary strings are bound by \oplus and superposed by thresholded addition. Decoding of information is based in BSC on (13).

Let us rephrase the main result as follows. Circular convolution of n -tuples whose entries consist of ± 1 may be regarded as a multiplicative representation of XOR of n -bit strings, and the appropriate map is

$$x \oplus y \mapsto (-1)^{P_{x \oplus y}} = (-1)^{P_x} (-1)^{P_y} = RF.$$

The link between HRR and BSC is given by the map $x \mapsto (-1)^{P_x}$, where x on its left-hand side is a binary string of length n while on the right-hand side x occurs at the diagonal of an $n \times n$ matrix P_x .

In the next section we will see that geometric product represents the *same* structure but in a *projective* way.

6. Geometric product as a projective representation of XOR

Let $x_1 \dots x_n$ and $y_1 \dots y_n$ be binary representations of two n -bit numbers x and y . Now let us consider two blades $c_x = c_{x_1 \dots x_n} = b_1^{x_1} \dots b_n^{x_n}$, $c_y = c_{y_1 \dots y_n} = b_1^{y_1} \dots b_n^{y_n}$. We will show that geometric product of c_x and c_y equals, up to a sign, $c_{x \oplus y}$. In this sense the map $x \mapsto c_x$ is an analogue of the exponential map $x \mapsto (-1)^{P_x}$.

Let us begin with examples:

$$\begin{aligned} b_1 b_1 &= c_{10\dots 0} c_{10\dots 0} = 1 = c_{0\dots 0} = c_{(10\dots 0) \oplus (10\dots 0)} \\ b_1 b_{12} &= c_{10\dots 0} c_{110\dots 0} = b_1 b_1 b_2 = b_2 = c_{010\dots 0} = c_{(10\dots 0) \oplus (110\dots 0)} \\ b_{12} b_1 &= c_{110\dots 0} c_{10\dots 0} = b_1 b_2 b_1 = -b_2 b_1 b_1 = -b_2 = -c_{010\dots 0} \\ &= -c_{(110\dots 0) \oplus (10\dots 0)} \\ b_{1257} b_{26} &= c_{11001010\dots 0} c_{0100010\dots 0} = b_1 b_2 b_5 b_7 b_2 b_6 \\ &= (-1)^3 b_1 b_5 b_6 b_7 = (-1)^3 c_{10001110\dots 0} \\ &= (-1)^D c_{(11001010\dots 0) \oplus (0100010\dots 0)}. \end{aligned}$$

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.

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

$$c_{x_1 \dots x_n} c_{y_1 \dots y_n} = (-1)^{\sum_{k<l} y_k x_l} c_{(x_1 \dots x_n) \oplus (y_1 \dots y_n)}. \quad (14)$$

Indeed, for two arbitrary strings of bits we have

$$\begin{aligned} D &= y_1(x_2 + \dots + x_n) + y_2(x_3 + \dots + x_n) + \dots + y_{n-1}x_n \\ &= \sum_{k<l} y_k x_l. \end{aligned}$$

We conclude that the map

$$\begin{aligned} (x_1, \dots, x_n) \times (y_1, \dots, y_n) &\mapsto (x_1, \dots, x_n) \oplus (y_1, \dots, y_n) \\ &= (x_1 \oplus y_1, \dots, x_n \oplus y_n) \end{aligned}$$

is projectively (i.e. up to a sign) represented in GA by means of (14).

Representations ‘up to a sign’ play an important role in physics and are at the roots of many phenomena such as half-integer spin

and fermionic statistics. To the best of our knowledge the link between projective representations of XOR and GA was noticed for the first time in the preliminary version (Aerts, Czachor & De Moor, 2006) of this paper.

7. Geometric analogue of HRR

Let us begin with illustrating the original BSC by means of the example taken from Kanerva (1997). The records are represented by unstructured randomly chosen strings of bits. The encoded record is

$$\mathbf{PSmith} = (\mathbf{name} \oplus \mathbf{Pat}) \boxplus (\mathbf{sex} \oplus \mathbf{male}) \boxplus (\mathbf{age} \oplus \mathbf{66}).$$

Decoding of the ‘name’ looks as follows:

$$\begin{aligned} \mathbf{Pat}' &= \mathbf{name} \oplus \mathbf{PSmith} \\ &= \mathbf{name} \oplus [(\mathbf{name} \oplus \mathbf{Pat}) \boxplus (\mathbf{sex} \oplus \mathbf{male}) \boxplus (\mathbf{age} \oplus \mathbf{66})] \\ &= \mathbf{Pat} \boxplus (\mathbf{name} \oplus \mathbf{sex} \oplus \mathbf{male}) \boxplus (\mathbf{name} \oplus \mathbf{age} \oplus \mathbf{66}) \\ &= \mathbf{Pat} \boxplus \text{noise} \rightarrow \mathbf{Pat}. \end{aligned}$$

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 by means of circular convolution in HRRs, but then we would have used the inverse \mathbf{name}^{-1} or the involution \mathbf{name}^* , and an appropriate measure of distance. Again, the last step is comparison of the noisy object with the ‘pure’ objects stored in clean-up memory. This is how standard BSC works.

We can now use the exponential map $x \mapsto (-1)^{P_x}$ to turn BSC into HRR. Let us, however, proceed in the geometric way and employ $x \mapsto c_x$. The roles and fillers are represented by randomly chosen blades:

$$\mathbf{PSmith} = \mathbf{name Pat} + \mathbf{sex male} + \mathbf{age 66}.$$

The space denotes the geometric product and \boxplus is replaced, similarly to HRR, by ordinary addition. At the level of explicit blades the record corresponds to the multivector **PSmith**

$$\mathbf{PSmith} = c_{a_1 \dots a_n} c_{x_1 \dots x_n} + c_{b_1 \dots b_n} c_{y_1 \dots y_n} + c_{c_1 \dots c_n} c_{z_1 \dots z_n}.$$

The blades indexed by the beginning of the alphabet represent roles (name, sex, age) while the remaining ones correspond to the fillers (Pat, male, 66). The decoding looks as follows:

$$\begin{aligned} \mathbf{name}^{-1} \mathbf{PSmith} &= c_{a_1 \dots a_n}^{-1} \\ &\quad \times [c_{a_1 \dots a_n} c_{x_1 \dots x_n} + c_{b_1 \dots b_n} c_{y_1 \dots y_n} + c_{c_1 \dots c_n} c_{z_1 \dots z_n}] \\ &= c_x \pm c_{a \oplus b \oplus y} \pm c_{a \oplus c \oplus z} \\ &= \mathbf{Pat} + \text{noise}. \end{aligned}$$

The signs have to be computed by means of (14).

An analogue of clean-up memory can be constructed in various ways. One possibility is to make sure that the fillers, c_x , etc., are orthogonal to the noise term. For example, let us take the fillers of the form $c_{x_1 \dots x_k 0 \dots 0}$, 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 $c_{(a_1 \dots a_n) \oplus (b_1 \dots b_n) \oplus (y_1 \dots y_n)}$ will with high probability contain at least one $y_j = 1$, $k < j \leq n$, and thus will be orthogonal to the fillers. The clean-up memory will consist of vectors with $y_j = 0$, $k < j \leq n$, i.e. of the filler form.

The procedure will work in an ideal case. However, in applications it may turn out very impractical to require that the noise term be exactly orthogonal to the fillers, as the procedure can easily become unstable. In the next section we show how to decode in a way that does not lead to instabilities.

8. More efficient way of decoding: Projected products

The operation $\mathbf{name}^{-1} \mathbf{PSmith} = \mathbf{Pat} + \text{noise}$

is not as efficient as it could be since a part of computation is entirely devoted to generation of the noise term. The idea of a projected geometric product is to define a hierarchy of decoding operations that generate only those parts of multivectors that belong to given subspaces of interest. Let us describe the construction on examples.

Assume we deal with n -bit strings. The corresponding Euclidean space is spanned by an orthonormal basis $\{b_1, \dots, b_n\}$. Now assume that each of the roles and fillers is given by a vector. For example, $\mathbf{name} = \sum_{j=1}^n N_j b_j = N$, $\mathbf{sex} = \sum_{j=1}^n S_j b_j = S$, $\mathbf{age} = \sum_{j=1}^n A_j b_j = A$, $\mathbf{Pat} = \sum_{j=1}^n P_j b_j = P$, $\mathbf{male} = \sum_{j=1}^n M_j b_j = M$, $\mathbf{66} = \sum_{j=1}^n 66_j b_j = 66$. The encoded record $\mathbf{PSmith} = NP + SM + A66$ has the following structure:

$$\begin{aligned} \mathbf{PSmith} &= N \cdot P + N \wedge P \\ &\quad + S \cdot M + S \wedge M \\ &\quad + A \cdot 66 + A \wedge 66. \end{aligned}$$

PSmith is thus a multivector consisting of scalars and bivectors only. So, this is not a general linear combination of all the possible 2^n blades, but a multivector from an $\binom{n}{0} + \binom{n}{2} = 1 + n(n-1)/2$ -dimensional subspace. The decoding operation

$$\begin{aligned} \mathbf{name}^{-1} \mathbf{PSmith} &= N(N \cdot P) + N(N \wedge P) \\ &\quad + N(S \cdot M) + N(S \wedge M) \\ &\quad + N(A \cdot 66) + N(A \wedge 66), \end{aligned}$$

produces a multivector consisting of vectors and trivectors, with all the trivectors belonging to the noise part. What this means is that since the original $\mathbf{Pat} = P$ was a vector, it certainly did not contain a trivector component. It thus makes no sense to generate the entire multivector $\mathbf{name}^{-1} \mathbf{PSmith}$ and compare it with items stored in clean-up memory, since all these items have the same (vanishing) trivector components. It would be also inefficient to first generate $\mathbf{name}^{-1} \mathbf{PSmith}$ and then project it on the vector subspace. What we want to do is to directly generate only the vector part of $\mathbf{name}^{-1} \mathbf{PSmith}$.

Denote by $\langle \cdot \rangle_k$ the projection on the component spanned by k -blades, and let

$$X = \langle X \rangle_0 + \langle X \rangle_2 = x_0 + \sum_{l < m} x_{lm} b_l b_m,$$

where $x_{lm} = -x_{ml}$. If A is a vector, then

$$\begin{aligned} \langle AX \rangle_1 &= x_0 A + \sum_{l,m} A_l x_{lm} b_m = \sum_k \left(x A_k + \sum_l A_l x_{lk} \right) b_k \\ &= \sum_k Y_k b_k = Y \end{aligned}$$

is also a vector. In this way we have arrived at a map $*_{1,2}$ that transforms an n -tuple $A = (A_1, \dots, A_n)$ and a $(1 + n(n-1)/2)$ -tuple $X = (x_0, x_{12}, \dots, x_{n-1,n})$ into an n -tuple $Y = (Y_1, \dots, Y_n)$,

$$Y_k = (A *_{1,2} X)_k = x_0 A_k + \sum_{l=1}^n A_l x_{lk}, \tag{15}$$

where the convention is that for $l > k$ we put in (15) $x_{lk} = -x_{kl}$. More explicitly, we can write

$$Y_k = (A *_{1,2} X)_k = x_0 A_k + \sum_{l=1}^{k-1} A_l x_{lk} - \sum_{l=k+1}^n A_l x_{kl}. \tag{16}$$

The map $*_{1,2}$ is an example of what we call a *projected product*. $*_{1,2}$ exactly reconstructs the vector part of AX without any need of computing the whole of AX . As opposed to convolutions, $*_{1,2}$ is basis independent. As an exercise let us take $X = BC$, $A = B$. Now,

$$\begin{aligned} X &= (x_0, x_{12}, \dots, x_{n-1,n}) \\ &= \left(\sum_k B_k C_k, B_1 C_2 - C_1 B_2, \dots, B_{n-1} C_n - C_{n-1} B_n \right), \end{aligned}$$

and

$$\begin{aligned}
 Y_k &= (B *_{1,2} BC)_k \\
 &= B_k \sum_{l=1}^n B_l C_l + \sum_{l=1}^{k-1} B_l (B_l C_k - B_k C_l) - \sum_{l=k+1}^n B_l (B_k C_l - B_l C_k) \\
 &= C_k \sum_{l=1}^n B_l B_l = |B|^2 C_k,
 \end{aligned}$$

as it should be since $BBC = |B|^2 C$.

Geometric algebra allows us to systematically derive in an analogous way a hierarchy of products, appropriate for multiplying sequences of different length, equipped with geometric interpretation and maintaining those features of geometric algebra that are necessary for concrete applications.

As a second example let us consider the more complicated case of hierarchical structures such as sequences, predicates, or recursive predicates. Plate's HRRs, similarly to various earlier approaches (Lewandowsky & Murdock, 1989; Murdock, 1983; Slack, 1984; Touretzky & Geva, 1987), employ chaining methods where complex structures are modeled by superpositions of objects of 'different rank'. A simplified form of coding the sentence 'Mark ate the fish' in HRR could be Plate (2003), for example,

$$\mathbf{X} = \mathbf{eat} + \mathbf{eat}_{agt} \otimes \mathbf{mark} + \mathbf{eat}_{obj} \otimes \mathbf{the_fish}.$$

Here \mathbf{eat}_{agt} and \mathbf{eat}_{obj} are the roles, \mathbf{mark} and $\mathbf{the_fish}$ are the fillers, and \mathbf{eat} is the frame name – all these objects are n -tuples. The geometric algebra analogues would be given by vectors from an n -dimensional Euclidean space. The record

$$\begin{aligned}
 X &= \mathbf{eat} + \mathbf{eat}_{agt} \mathbf{mark} + \mathbf{eat}_{obj} \mathbf{the_fish} \\
 &= \mathbf{eat} + \mathbf{eat}_{agt} \cdot \mathbf{mark} + \mathbf{eat}_{agt} \wedge \mathbf{mark} \\
 &\quad + \mathbf{eat}_{obj} \cdot \mathbf{the_fish} + \mathbf{eat}_{obj} \wedge \mathbf{the_fish}
 \end{aligned}$$

is now a multivector consisting of scalars, vectors, and bivectors. Decoding of \mathbf{mark} is performed by $\mathbf{eat}_{agt}^{-1} X$, resulting in a multivector consisting of scalars, vectors, bivectors, and trivectors. However, the filler \mathbf{mark} is a vector, hence scalars, bivectors, and trivectors belong to the noise term. In order to reconstruct \mathbf{mark} we can employ the same projected product as before.

Another frame representing 'Hunger caused Mark to eat the fish' can be represented by

$$\begin{aligned}
 Y &= \mathbf{cause} + \mathbf{cause}_{agt} \mathbf{hunger} + \mathbf{cause}_{obj} \mathbf{eat} \\
 &= \mathbf{cause} + \mathbf{cause}_{agt} \mathbf{hunger} + \mathbf{cause}_{obj} \mathbf{eat} \\
 &\quad + \mathbf{cause}_{obj} \mathbf{eat}_{agt} \mathbf{mark} + \mathbf{cause}_{obj} \mathbf{eat}_{obj} \mathbf{the_fish}.
 \end{aligned}$$

The entire multivector $\mathbf{eat}_{agt}^{-1} \mathbf{cause}_{obj}^{-1} Y$ contains in general all the k -vector components, for $k = 0, 1, 2, 3, 4, 5$. However, if we want to reconstruct \mathbf{mark} only, we have to project on vectors. We leave to the readers the exercise of defining an appropriate projected product that generates only the vector part $(\mathbf{eat}_{agt}^{-1} \mathbf{cause}_{obj}^{-1} Y)_1$ without providing us with the redundant scalar, bivector, trivector, 4-vector, and 5-vector components.

It remains to say something about the way we compare the decoded items with the contents of clean-up memory. In general, the similarity of items is checked by means of some metric or scalar product in the space of n -tuples or vectors. Preliminary results on comparison of GA analogues of HRRs with HRRs themselves or BSC will be reported in Patyk (in print). One conclusion of Patyk (in print) is that the most efficient way of doing this is by means of the standard unitary-space scalar product of n -tuples $(X, Y) = \sum_j \bar{X}_j Y_j$, a result that is in GA contexts less obvious than one might expect. The subtlety is that there exist various definition of scalar products in GA (Dorst, 2002) – the one that corresponds to the unitary-space definition has been formalized only recently in Marchuk and Shirokov (2008), and was mentioned (but not really employed) in the context of GA analogues of quantum algorithms in Aerts and Czachor (2008).

9. Between the two extremes: Dimensionality revisited

Circular convolution maps pairs of n -tuples into n -tuples. Tensor products map pairs of n -tuples into n^2 -tuples. We have seen that the issue of geometric products is more subtle. If the two n -tuples (x_1, \dots, x_n) and (y_1, \dots, y_n) represent components of two vectors in some basis, i.e. $x = \sum_{k=1}^n x_k b_k, y = \sum_{k=1}^n y_k b_k$, then

$$xy = \sum_{k=1}^n x_k y_k + \sum_{k < l} (x_k y_l - y_k x_l) b_k b_l \tag{17}$$

can be represented as the m -tuple $(x \cdot y, x_1 y_2 - y_1 x_2, \dots, x_{n-1} y_n - y_{n-1} x_n)$, with $m = n^2/2 - n/2 + 1$. However, if the multivector xy is again multiplied by a vector z , say, then one deals with a product of the m -tuple by an n -tuple, whose result is an l -tuple with $l = \binom{n}{1} + \binom{n}{3} = n^3/3! - n^2/2 + 4n/3$. In general, tensor products of kn -tuples are given by n^k -tuples; geometric products of kn -dimensional vectors correspond to m -tuples with $\lim_{n \rightarrow \infty} m/n^k = (k!)^{-1}$. Keeping n fixed and increasing k , the dimension of tensor products grows as n^k . The corresponding growth of dimension of geometric-product m -tuples is limited from above by 2^n , with the critical value $k = n / \log_2 n$ corresponding to $n^k = 2^n$.

Decoding based on the geometric product in general increases the dimensions of the m -tuples (but limited from above by the maximal value 2^n) due to the excess of noise terms. This is why it is advisable to decode by means of projected products. For example, let the dimension of an Euclidean space be n . The geometric product of $j = 2k$ vectors is in a given basis given by an m -tuple with $m = \sum_{l=0}^k \binom{n}{2l}$; for $j = 2k + 1$ we find $m = \sum_{l=0}^k \binom{n}{2l+1}$. A geometric product of five vectors, say $X = abcde$, has maximal dimension $\binom{n}{1} + \binom{n}{3} + \binom{n}{5}$. Multiplying X by f which is an inverse of either of these five vectors we get a multivector fX of maximal dimension $\binom{n}{0} + \binom{n}{2} + \binom{n}{4}$. If the vector f is not proportional to any of $\{a, \dots, e\}$, the dimension of fX is $\binom{n}{0} + \binom{n}{2} + \binom{n}{4} + \binom{n}{6}$. Replacing the geometric product by an appropriate projected product we always obtain $\binom{n}{0} + \binom{n}{2} + \binom{n}{4}$, no matter which vector f we choose. The part of dimension $\binom{n}{0} + \binom{n}{2} + \binom{n}{4}$ may contain noise, but the part corresponding to $\binom{n}{6}$ is just pure noise. Projected products become the more efficient the more complex the structure that is employed in the decoding (think of the reduction of dimensionality obtained by replacing $\mathbf{eat}_{agt}^{-1} \mathbf{cause}_{obj}^{-1} Y$ by $\langle \mathbf{eat}_{agt}^{-1} \mathbf{cause}_{obj}^{-1} Y \rangle_1$).

10. Cartan representation of Clifford algebras

It is useful to be able to work with matrix representations of GA. Although this is not an efficient way of doing GA computations, matrix representations allow us to perform independent cross-checks of various GA constructions and algorithms. In this section we give an explicit matrix representation of GA. We begin with Pauli's matrices:

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

The GA of a plane is represented as follows: $1 = 2 \times 2$ unit matrix, $b_1 = \sigma_1, b_2 = \sigma_2, b_{12} = \sigma_1 \sigma_2 = i \sigma_3$. Alternatively, we can write $c_{00} = 1, c_{10} = \sigma_1, c_{01} = \sigma_2, c_{11} = i \sigma_3$, and

$$\begin{aligned}
 &\psi_{00} c_{00} + \psi_{10} c_{10} + \psi_{01} c_{01} + \psi_{11} c_{11} \\
 &= \begin{pmatrix} \psi_{00} + i \psi_{11} & \psi_{10} - i \psi_{01} \\ \psi_{10} + i \psi_{01} & \psi_{00} - i \psi_{11} \end{pmatrix}.
 \end{aligned}$$

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

In 3D space we have $1 = 2 \times 2$ unit matrix, $b_1 = \sigma_1, b_2 = \sigma_2, b_3 = \sigma_3, b_{12} = \sigma_1 \sigma_2 = i \sigma_3, b_{13} = \sigma_1 \sigma_3 = -i \sigma_2, b_{23} = \sigma_2 \sigma_3 = i \sigma_1, b_{123} = \sigma_1 \sigma_2 \sigma_3 = i$.

Now the representation of

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

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 (Budinich & Trautman, 1988):

$$b_{2k} = \underbrace{\sigma_1 \otimes \dots \otimes \sigma_1}_{n-k} \otimes \sigma_2 \otimes \underbrace{1 \otimes \dots \otimes 1}_{k-1},$$

$$b_{2k-1} = \underbrace{\sigma_1 \otimes \dots \otimes \sigma_1}_{n-k} \otimes \sigma_3 \otimes \underbrace{1 \otimes \dots \otimes 1}_{k-1}.$$

So let us return to the example of Pat Smith. For simplicity take $n = 4$ so that we can choose the representation

$$\left. \begin{array}{l} \mathbf{Pat} = c_{1100}, \\ \mathbf{male} = c_{1000}, \\ \mathbf{66} = c_{0100}, \end{array} \right\} \text{ fillers}$$

$$\left. \begin{array}{l} \mathbf{name} = c_{1010}, \\ \mathbf{sex} = c_{0111}, \\ \mathbf{age} = c_{1011}. \end{array} \right\} \text{ roles}$$

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:

$$\begin{aligned} \mathbf{Pat} &= c_{1100} = b_1 b_2 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3) \\ &\quad \times (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2) \\ &= 1 \otimes 1 \otimes 1 \otimes (-i\sigma_1) \end{aligned}$$

$$\mathbf{male} = c_{1000} = b_1 = \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3$$

$$\mathbf{66} = c_{0100} = b_2 = \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2$$

$$\begin{aligned} \mathbf{name} &= c_{1010} = b_1 b_3 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3) \\ &\quad \times (\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1) \\ &= 1 \otimes 1 \otimes (-i\sigma_2) \otimes \sigma_3 \end{aligned}$$

$$\begin{aligned} \mathbf{sex} &= c_{0111} = b_2 b_3 b_4 \\ &= (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1) \\ &\quad \times (\sigma_1 \otimes \sigma_1 \otimes \sigma_2 \otimes 1) \\ &= \sigma_1 \otimes \sigma_1 \otimes (-i1) \otimes \sigma_2, \end{aligned}$$

$$\begin{aligned} \mathbf{age} &= c_{1011} = b_1 b_3 b_4 \\ &= (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1) \\ &\quad \times (\sigma_1 \otimes \sigma_1 \otimes \sigma_2 \otimes 1) \\ &= \sigma_1 \otimes \sigma_1 \otimes (-i1) \otimes \sigma_3. \end{aligned}$$

In practice, to turn the above expressions into explicit matrices one employs the formula

$$A \otimes B = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}$$

valid for matrices A and B of arbitrary dimensions, supplemented by associativity of \otimes .

The whole record – where the superposition is taken for clarity with arbitrary parameters α , β , and γ – reads

$$\begin{aligned} \mathbf{PSmith} &= \alpha \mathbf{name Pat} + \beta \mathbf{sex male} + \gamma \mathbf{age 66} \\ &= \alpha c_{1010} c_{1100} + \beta c_{0111} c_{1000} + \gamma c_{1011} c_{0100} \\ &= \alpha c_{0110} - \beta c_{1111} + \gamma c_{1111}. \end{aligned}$$

The two noise terms are here linearly dependent by accident, 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 the simplicity of the example. Decoding the name involves two steps. First

$$\begin{aligned} \mathbf{name PSmith} &= c_{1010} \mathbf{PSmith} = c_{1010} [\alpha c_{0110} - (\beta - \gamma) c_{1111}] \\ &= -\alpha c_{1100} - (\beta - \gamma) c_{0101} \\ &= -\alpha \mathbf{Pat} - \underbrace{(\beta - \gamma) c_{0101}}_{\text{noise}} \\ &= -\alpha \underbrace{b_1 b_2}_{\mathbf{Pat}} - \underbrace{(\beta - \gamma) b_2 b_4}_{\text{noise}} = \mathbf{Pat}' \end{aligned}$$

It remains to employ clean-up memory. But this is easy since the noise is perpendicular to \mathbf{Pat} . We only have to project on the set spanned by the fillers (i.e. the blades involving neither b_3 nor b_4), and within this set check which element is closest to the cleaned-up \mathbf{Pat}' .

11. Conclusions

In order to switch from a binary string x , occurring in BSC, to HRR, one employs the exponential map $x \mapsto (-1)^{P_x}$, where x plays a dual role. At the left-hand side x is just a sequence of bits distributed over a n -tuple. At the right-hand side the bits are distributed over the diagonal of a diagonal $n \times n$ matrix P_x . Binary strings equipped with componentwise addition mod 2 (i.e. \oplus , XOR) form a group. The exponential map is a representation of this group in the space of diagonal matrices: $x \oplus y \mapsto (-1)^{P_{x \oplus y}} = (-1)^{P_x} (-1)^{P_y}$.

This group also possesses a projective representation in the space of blades of a GA: $x \oplus y \mapsto c_{x \oplus y} = \pm c_x c_y$. Blades have a straightforward geometric interpretation. As opposed to tensor products, that increase dimensions, the dimensions of c_x , c_y and $c_{x \oplus y}$ are the same. Therefore, $c_{x \oplus y}$ can be used to bind variables. A superposition of such bound variables, a multivector, is a reduced representation analogous to HRR.

Multiplication of multivectors preserves the dimension of the 2^n -dimensional space. This dimension is typically so high that there exists a risk of inefficiency of the GA formalism. In practice, however, one deals with low-dimensional (polynomial in n) subspaces of the 2^n -dimensional space. At this lower-dimensional level the geometric-product binding is a procedure, in a sense, in between the two extreme cases: the circular convolution and the tensor product. What makes GA efficient is, on the one hand, the fact that at the level of geometric products one can consider sums and products of arbitrary numbers of vectors or multivectors. On the other hand, at the level of n -tuples GA automatically defines products of strings of numbers of varying dimensions, a property which is very useful from the point of view of algorithmic implementations of appropriate distributed representations. GA is associative and non-commutative, and in this respect is similar to matrix models (Murdock, 1985; Pike, 1984). One can even make a GA model equivalent to a matrix model by selecting a matrix representation of a Clifford algebra. The corresponding matrices are in typical applications very sparse (contain lots of zeros), which indicates that the matrix algebra is not the most efficient way of dealing with GA. The projected products we have introduced take care of this sparseness by acting only on selected components of multivectors.

The formalism we have proposed can be extended to other domains where binding and superposition of vectors is used. In particular, it seems that GA is a natural language for various kinds of semantic analysis, such as Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer, Foltz & Laham, 1998), Hyperspace Analogue to Language (HAL) (Lund & Burgess, 1996),

Probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 1999), Latent Dirichlet Allocation (Blei, Ng & Jordan, 2003), Topic Model (Griffiths, Steyvers & Tenenbaum, 2007), or BEAGLE (Jones & Mewhort, 2007). The well known difficulty of the approaches à la LSA is the issue of how to encode order of words (the bag-of-words problem). For those who were trained on quantum mechanics the solution seemed obvious and led to the tensor-product binding (Aerts & Czachor, 2004), and the resulting quantum-like structures in LSA. GA leads to an alternative formalism that has not been fully explored in this context as yet.

Another possible field of applications is in modeling of concepts. It is known (Aerts & Gabora, 2005a,b) that paradoxical properties of combinations of concepts are naturally modeled within Hilbert-space frameworks. Since the results from Aerts and Czachor (2008) show how to map Hilbert-space structures into GA, one expects that the ‘quantum’ theory of concepts can be given an entirely geometric representation, in agreement with the general intuitions expressed in Gärdenfors (2003).

GA may also be interpreted as a way of encoding mutual geometric relations between multidimensional geometric objects. For example, a pair containing an oriented plane element and a vector from this plane is mapped in GA into a vector which shows how to move the first vector in order to produce the oriented plane segment in question. The algebraic operation thus reveals a geometric property of the plane segment and resembles the process of *understanding* geometry, a fact suggesting that association of GA with cognitive science is not just a mathematical curiosity but may be deeply rooted in the ways we think.

The fact that geometry is in its deepest essence algebraic, and it is the Grassmann–Clifford formalism that probably correctly phrases this, found in late 1920s unexpected support from relativistic quantum mechanics. Since the discovery of Dirac (1928) we know that the deepest-level relation between energy and mass is $p^2 = (mc)^2$, where p is the energy–momentum 4-vector and the product in p^2 is *geometric* (the so-called Dirac matrices are just the Minkowski space basis vectors, taken by Dirac in a matrix representation of GA). GA is thus the fundamental ‘language of matter’ and one should not be surprised to find it at the roots of other fundamental levels of reality.

Acknowledgments

This work was supported by the Flemish Fund for Scientific Research (FWO) project G.0452.04 and the Polish national network LFPPi.

References

- Aerts, D., & Czachor, M. (2004). Quantum aspects of semantic analysis and symbolic artificial intelligence. *Journal of Physics A*, 37, L123–L132.
- Aerts, D., Czachor, M., & De Moor, B. (2006). On geometric-algebra representation of binary spatter codes. preprint arXiv:cs/0610075[cs.AI].
- Aerts, D., & Czachor, M. (2007). Cartoon computation: Quantum-like algorithms without quantum mechanics. *Journal of Physics A*, 40, F259.
- Aerts, D., & Czachor, M. (2008). Tensor-product versus geometric-product coding. *Physical Review A*, 77, 012316.
- Aerts, D., Czachor, M., & Orłowski, L. (2009). Teleportation of geometric structures in 3D. *Journal of Physics A. Mathematical and Theoretical*, 42, 135307 (8pp). doi: 10.1088/1751-8113/42/13/135307.
- Aerts, D., & Gabora, L. (2005a). A theory of concepts and their combinations I: The structure of the sets of contexts and properties. *Kybernetes*, 34, 167–191.
- Aerts, D., & Gabora, L. (2005b). A theory of concepts and their combinations II: A Hilbert space representation. *Kybernetes*, 34, 192–221.
- Anderson, J. A. (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences*, 12, 197.
- Baylis, W. E. (1996). *Electrodynamics: A modern geometric approach*. Boston: Birkhauser.
- Bayro-Corrochano, E., Danilidis, K., & Sommer, G. (2000). Motor algebra for 3D kinematics: The case of the hand-eye calibration. *Journal of Mathematical Imaging and Vision*, 13, 79–100.
- Bayro-Corrochano, E. J. (2001). Geometric neural computing. *IEEE Transactions on Neural Networks*, 12, 968–986.
- Bennett, C. H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., & Wootters, W. K. (1993). Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. *Physical Review Letters*, 70, 1895.
- Blei, D. M., Ng, A. N., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 993.
- Borsellino, A., & Poggio, T. (1973). Convolution and correlation algebras. *Kybernetik*, 13, 113–122.
- Budinich, P., & Trautman, A. (1988). *The spinorial chessboard*. Berlin: Springer.
- Clifford, W. K. (1878). Applications of Grassmann’s extensive algebra. *American Journal of Mathematics Pure and Applied*, 1, 350–358.
- Czachor, M. (2007). Elementary gates for cartoon computation. *Journal of Physics A*, 40, F753.
- Deerwester, S., et al. (1990). Indexing by latent semantic analysis. *Journal of American Society for Information Science*, 41, 391.
- Dirac, P. A. M. (1928). The quantum theory of the electron. *Proceedings of the Royal Society, A117*, 610–624.
- Doran, C., & Lasenby, A. (2003). *Geometric algebra for physicists*. Cambridge: Cambridge University Press.
- Dorst, L., Doran, C. J. L., & Lasenby, J. (Eds.) (2002). *Applications of geometric algebra in computer science and engineering*. Boston: Birkhauser.
- Dorst, L. (2002). The inner products of geometric algebra. In L. Dorst, C. J. L. Doran, & J. Lasenby (Eds.), *Applications of geometric algebra in computer science and engineering*. Boston: Birkhauser.
- Dorst, L., Fontijne, D., & Mann, S. (2007). *The Morgan Kaufmann series in computer graphics. Geometric algebra for computer science: An object-oriented approach to geometry*. Amsterdam: Morgan Kaufmann.
- Gabor, D. (1968). Holographic model for temporal recall. *Nature*, 217, 1288–1289.
- Gärdenfors, P. (2003). *Conceptual Spaces: The geometry of thought*. Oxford: Oxford University Press.
- Gibbs, J. W. (1906). *The scientific papers of J. Willard Gibbs*. London: Longmans, Green and Company.
- Grassmann, H. (1877). Der Ort der Hamilton’schen Quaternionen in der Ausdehnungslehre. *Mathematische Annalen*, 3, 375–386.
- Griffiths, T. L., Steyvers, M., & Tenenbaum, J. B. (2007). Topics in semantic representation. *Psychological Review*, 114, 211–244.
- Hestenes, D. (1966). *Space-time algebra*. New York: Gordon and Breach.
- Hestenes, D., & Sobczyk, G. (1984). *Clifford algebra to geometric calculus: A unified language for mathematics and physics*. Dordrecht: Reidel.
- Hestenes, D. (1986). *New foundations for classical mechanics*. Dordrecht: Kluwer.
- Hestenes, D. (1994a). Invariant body kinematics: I. Saccadic and compensatory eye movements. *Neural Networks*, 7, 65–77.
- Hestenes, D. (1994b). Invariant body kinematics: II. Reaching and neurogeometry. *Neural Networks*, 7, 79–88.
- Hestenes, D. (2003). Reforming the mathematical language of physics. *American Journal of Physics*, 71, 104–121.
- Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46, 47–76.
- Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of uncertainty in artificial intelligence*.
- Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114, 1–37.
- Kanerva, P. (1996). Binary spatter codes of ordered k -tuples. In C. von der Malsburg, et al., (Eds.), *Lecture notes in computer science: vol. 1112. Artificial neural networks–ICANN proceedings* (pp. 869–873). Berlin: Springer.
- Kanerva, P. (1997). Fully distributed representation. In *Proc. 1997 real world computing symposium* (pp. 358–365). Tsukuba-City: Real World Computing Partnership.
- Kanerva, P. (1998). Large patterns make great symbols: An example of learning from example. In *Hybrid neural systems* (pp. 194–203).
- Kohonen, T. (1972). Correlation matrix memories. *IEEE Transactions on Computers*, 21, 353–359.
- Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to latent semantic analysis. *Discourse Processes*, 25, 259.
- Lasenby, J., Lasenby, A. N., Doran, C. J. L., & Fitzgerald, W. J. (1998). New geometric methods for computer vision. *International Journal of Computer Vision*, 36, 191–213.
- Lewandowsky, S., & Murdock, B. B. (1989). Memory for serial order. *Psychological Review*, 96, 25–57.
- Liepa, P. (1977). Models of content addressable distributed associative memory. Unpublished manuscript.
- Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments and Computers*, 28, 203.
- Marchuk, N. G., & Shirokov, D. S. (2008). Unitary spaces on Clifford algebras. *Advances in Applied Clifford Algebras*, 18, 237–254.
- Metcalfe, J. (1991). Recognition failure and CHARM. *Psychological Review*, 98, 529–553.
- Metcalfe-Eich, J. (1982). A composite holographic associative recall model. *Psychological Review*, 89, 627–661.
- Murdock, B. B. (1982). A theory for the storage and retrieval of item and associative information. *Psychological Review*, 89, 316–338.
- Murdock, B. B. (1983). A distributed memory model for serial-order information. *Psychological Review*, 90, 316–338.
- Murdock, B. B. (1985). Convolution and matrix systems: A reply to Pike. *Psychological Review*, 92, 130–132.

- Nielsen, M. A., & Chuang, I. L. (2000). *Quantum computation and quantum information*. Cambridge: Cambridge University Press.
- Pavšič, M. (2001). *The landscape of theoretical physics: A global view. From point particles to the brane world and beyond, in search of a unifying principle*. Boston: Kluwer.
- Patyk, A. (2009). Geometric algebra model of distributed representations. In E. Bayro-Corrochano & G. Scheuermann (Eds.), *Geometric algebra applications in computer science and engineering*. Berlin: Springer (in print).
- Penrose, R. (1994). *Shadows of the mind*. Oxford: Oxford University Press.
- Pike, R. (1984). Comparison of convolution and matrix distributed memory systems for associative recall and recognition. *Psychological Review*, 91, 281–294.
- Plate, T. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6, 623–641.
- Plate, T. (2003). *Holographic reduced representation: Distributed representation for cognitive structures*. Stanford: CSLI Publications.
- Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46, 159–216.
- Somaroo, S., Cory, D. G., & Havel, T. F. (1998). Expressing the operations of quantum computing in multiparticle geometric algebra. *Physics Letters A*, 240, 1–7.
- Sommer, G. (Ed.) (2001). *Geometric computing with Clifford algebras*. Berlin: Springer.
- Slack, J. N. (1984). The role of distributed memory in natural language processing. In T. O'Shea (Ed.), *Advances in artificial intelligence: Proceedings of the 6th European conference on artificial intelligence*. Elsevier.
- Touretzky, D. S., & Geva, S. (1987). A distributed connectionist representation for concept structures. In *Proceedings of the 9th annual conference of the cognitive science society*. Hillsdale: Erlbaum.
- Widdows, D. (2004). *Geometry and meaning*. Stanford: CSLI Publications.
- Widdows, D., & Higgins, M. (2004). Geometric ordering of concepts, logical disjunction, and learning by induction. In *AAAI fall symposium series, Compositional connectionism in cognitive science*.
- Willshaw, D. (1989). Holography, associative memory, and inductive generalization. In G. E. Hinton, & J. A. Anderson (Eds.), *Parallel models of associative memory* (updated edition) (pp. 103–124). Hillsdale: Erlbaum.