Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 221–228
ISBN 978-83-60810-22-4

A comparison of geometric analogues of Holographic Reduced Representations, original Holographic Reduced Representations and Binary Spatter Codes

Agnieszka Patyk-Łońska, Marek Czachor, and Diederik Aerts

***Abstract*—Geometric Analogues of Holographic Reduced Representations (GA HRR) employ role-filler binding based on geometric products. Atomic objects are real-valued vectors in n-dimensional Euclidean space and complex statements belong to a hierarchy of multivectors. The paper reports a battery of tests aimed at comparison of GA HRR with Holographic Reduced Representation (HRR) and Binary Spatter Codes (BSC). Firstly, we perform a test of GA HRR which is analogous to the one proposed by Plate in [13]. Plate's simulation involved several thousand 512-dimensional vectors stored in clean-up memory. The purpose was to study efficiency of HRR but also to provide a counterexample to claims that role-filler representations do not permit one component of a relation to be retrieved given the others. We repeat Plate's test on a continuous version of GA HRR – GA$_c$ (as opposed to its discrete version described in [12]) and compare the results with the original HRR and BSC. The object of the test is to construct statements concerning multiplication and addition. For example, "$2 \cdot 3 = 6$" is constructed as $times_{2,3} = times + operand * (num_2 + num_3) + result * num_6$. To look up this vector one then constructs a similar statement with one of the components missing and checks whether it points correctly to $times_{2,3}$. We concentrate on comparison of recognition percentage for the three models for comparable data size, rather than on the time taken to achieve high percentage. Results show that the best models for storing and recognizing multiple similar statements are GA$_c$ and Binary Spatter Codes with recognition percentage highly above 90.**

***Index Terms*—distributed representations, geometric algebra, HRR, BSC, scaling.**

I. Introduction

HOLOGRAPHIC Reduced Representations (HRR) and Binary Spatter Codes (BSC) are distributed representations of cognitive structures where binding of role–filler codevectors maintains predetermined data size. In HRR [13] binding is performed by means of circular convolution

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

of real n-tuples or, in 'frequency domain', by componentwise multiplication of (complex) n-tuples,

$$(x_1, \ldots, x_n) \circledast (y_1, \ldots, y_n) = (x_1 y_1, \ldots, x_n y_n). \tag{2}$$

Bound n-tuples are superposed by addition, and unbinding is performed by an approximate inverse. A dual formalism, where real data are bound by componentwise multiplication, was discussed by Gayler [6]. In BSC [8], [9] one works with binary n-tuples, bound by componentwise addition mod 2,

$$\begin{aligned}(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) &= (x_1 \oplus y_1, \ldots, x_n \oplus y_n), \\ x_j \oplus y_j &= x_j + y_j \bmod 2, \end{aligned} \tag{3}$$

and superposed by pointwise majority-rule addition; unbinding is performed by the same operation as binding.

One often reads that the above models represent data by *vectors*, which is not exactly true. Given two vectors one does not know how to perform, say, their convolution or componentwise multiplication since the result depends on basis that defines the components. Basis must be fixed in advance since otherwise all the above operations become ambiguous. It follows that neither of the above reduced representations can be given a true and meaningful geometric interpretation. Geometric Analogues of Holographic Reduced Representations (GA HRR) [2] can be constructed if one defines binding by the geometric product, a notion introduced in 19th century works of Grassmann [7] and Clifford [5].

The fact that GA HRR is intrinsically geometric may be important for various conceptual reasons — for example, the rules of geometric algebra may be regarded as a mathematical formalization of the process of *understanding* geometry. The use of geometric algebra in distributed representations has been inspired by a well-known fact, that most people think in pictures, i.e. two- and three-dimensional shapes, not by using sequences of ones and zeroes.

In order to grasp the main ideas behind GA HRR let us consider an orthonormal basis $b_1, \ldots, b_n$ in some n-dimensional Euclidean space. Now consider two vectors $x = \sum_{k=1}^{n} x_k b_k$ and $y = \sum_{k=1}^{n} y_k b_k$. The *scalar*

$$x \cdot y = y \cdot x \tag{4}$$

is known as the *inner product*. The *bivector*

$$x \wedge y = -y \wedge x \tag{5}$$

is the *outer product* and may be regarded as an oriented plane segment (alternative interpretations are also possible, cf. [4]). **1** is the identity of the algebra. The geometric product of x

and y then reads

$$xy = \underbrace{\sum_{k=1}^{n} x_k y_k \, \mathbf{1}}_{x\cdot y} + \underbrace{\sum_{k<l} (x_k y_l - y_k x_l) b_k b_l}_{x\wedge y}. \quad (6)$$

Grassmann and Clifford introduced geometric product by means of the basis-independent formula involving the *multivector*

$$xy = x \cdot y + x \wedge y \quad (7)$$

which implies the so-called Clifford algebra

$$b_k b_l + b_l b_k = 2\delta_{kl}\mathbf{1}. \quad (8)$$

when restricted to an orthonormal basis. Inner and outer product can be defined directly from xy:

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

The most ingenious element of (7) is that it adds two apparently different objects, a scalar and a plane element, an operation analogous to addition of real and imaginary parts of a complex number. Geometric product for vectors x, y, z can be axiomatically defined by the following rules:

$$\begin{aligned} (xy)z &= x(yz), \\ x(y+z) &= xy + xz, \\ (x+y)z &= xz + yz, \\ xx = x^2 &= |x|^2, \end{aligned}$$

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

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

Geometric algebra 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. 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 geometric and 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 Euclidean space there are 2^n different blades. This can be seen as follows. Let $\{x_1, \dots, x_n\}$ be a sequence 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 $b_k^0 = \mathbf{1}$, which shows that blades are in a one-to-one relation with n-bit numbers. A general multivector is a linear combination of blades,

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

with real or complex coefficients $\psi_{x_1 \dots x_n}$. Clifford algebra implies that

$$c_{x_1 \dots x_n} c_{y_1 \dots y_n} = (-1)^{\sum_{k<l} y_k x_l} c_{(x_1 \dots x_n) \oplus (y_1 \dots y_n)}, \quad (10)$$

where $\oplus$ is given by (3). Multiplication of two basis blades is thus, up to a sign, in a one-to-one relation with exclusive alternative of two binary n-tuples. Accordingly, (10) is a projective representation of the group of binary n-tuples with addition modulo 2.

GA HRR is based on binding defined by geometric product (10) of blades while superposition is just addition of blades (9). The discrete GA$_d$ is a version of GA HRR obtained if $\psi_{x_1 \dots x_n}$ in (9) equal ± 1. The first recognition tests of GA$_d$, as compared to HRR and BSC, were described in [12]. In the present paper we go further and compare HRR and BSC with GA$_c$, a version of GA HRR employing "projected products" [2] and arbitrary real $\psi_{x_1 \dots x_n}$. We also repeat Plate's scaling test ([13], Appendix I) and compare test results for GA$_c$, HRR and BSC models.

Throughout this paper we shall use the following notation: "$*$" denotes binding roles and fillers by means of the geometric product and "$+$" denotes the superposition of sentence chunks, e.g.

$$\text{“\textit{Fido bit Pat}”} = bite_{agt} * Fido + bite_{obj} * Pat. \quad (11)$$

Additionally, "$\circledast$" will denote binding performed by circular convolution used in the HRR model and a^* denotes the involution of a HRR vector a. A "$+$"' in the superscript of x^+ denotes the operation of reversing a blade or a multivector x: $(b_{k_1 \dots k_j})^+ = b_{k_j} \dots b_{k_1}$. Asking a question will be denoted with "$\sharp$", as in

$$\begin{aligned} &\text{“\textit{Who bit Pat?}”} \\ &= (bite_{agt} * Fido + bite_{obj} * Pat) \,\sharp\, bite_{agt} \quad (12) \\ &\approx Fido. \end{aligned}$$

The *size* of a (multi)vector means the number of memory cells it occupies in computer's memory, while the *magnitude* of a (multi)vector $V = \{v_1, \dots, v_n\}$ is its Euclidean norm $\sqrt{\sum_{i=1}^{n} v_i^2}$.

For our purposes it is important that geometric calculus allows us to define in a very systematic fashion a hierarchy of associative, non-commutative, and invertible operations that can be performed on 2^n-tuples. The resulting superpositions are less noisy than the ones based on convolutions, say. Such operations are in general unknown to a wider audience, which explains popularity of tensor and convolution algebras. Geometric product preserves dimensionality at the level 2^n-dimensional *multivectors*, where n is the number of bits indexing basis vectors. Moreover, all nonzero vectors are invertible with respect to geometric product, a property absent for convolutions and important for unbinding and recognition. A detailed analysis of links between GA HRR, HRR and BSC can be found in [2]. In particular, it is shown that both GA HRR and BSC are based on two different representations (in group theoretical sense) of the additive group of binary n-tuples with addition modulo 2. Actually, the latter observation was the starting point for studying geometric algebra forms of reduced representations [3].

II. The GA$_c$ model

Multivector (9) associated with n-dimensional Euclidean space can be represented by the 2^n-tuple $(\psi_{0_1\dots 0_n},\dots,\psi_{1_1\dots 1_n})$. Geometric product of two such 2^n-tuples is again a 2^n-tuple. In this sense geometric product is analogous to bindings employed in HRR or BSC, but we can still proceed in several inequivalent ways. For example, since a product of two basis blades is again a basis blade multiplied by ± 1, we can require that $\psi_{x_1\dots x_n} = \pm 1$. Such a discrete version of GA HRR was tested vs. HRR and BSC in [12], and will be denoted here by GA$_d$ (discrete GA HRR).

The continuous GA$_c$ model differs greatly from GA$_d$. First of all, we do not begin with a general 2^n-dimensional multivector. Atomic objects are real-valued vectors in n-dimensional Euclidean space, in practice represented by n-tuples of components taken in some basis. A hierarchy of multivectors is reserved for complex statements, formed by binding and superposition of atomic objects. An n-dimensional vector, when seen from the multivector perspective, is a highly sparse 2^n-tuple: Only n out of 2^n components can be nonzero.

The procedure we employ was suggested in [2]. The space of 2^n-tuples is split into subspaces corresponding to scalars (0-vectors), vectors (1-vectors), bivectors (2-vectors), and so on. At the bottom of the hierarchy lay vectors $V \in \mathbb{R}^n$, having rank 1 and being denoted as $\overset{1}{V}$. An object of rank 2 is created by multiplying two elements of rank 1 with the help of the geometric product. Let $\overset{1}{V}=\{\alpha_1,\alpha_2,\alpha_3\}$ and $\overset{1}{W}=\{\beta_1,\beta_2,\beta_3\}$ be vectors in $\mathbb{R}^3$. A multivector $\overset{2}{X}$ of rank 2 in $\mathbb{R}^3$ comprises the following elements (cf. [10])

$$\overset{2}{X}=\overset{1}{V}\overset{1}{W}=\begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\end{bmatrix}\begin{bmatrix}\beta_1\\\beta_2\\\beta_3\end{bmatrix}=\begin{bmatrix}\alpha_1\beta_1+\alpha_2\beta_2+\alpha_3\beta_3\\\alpha_1\beta_2-\alpha_2\beta_1\\\alpha_1\beta_3-\alpha_3\beta_1\\\alpha_2\beta_3-\alpha_3\beta_2\end{bmatrix}, \quad (13)$$

the first entry in the array on the right being a scalar and the remaining three entries being 2-blades. For arbitrary vectors in $\mathbb{R}^n$ we would have obtained one scalar (or, more conviniently: $\binom{n}{0}$ scalars) and $\binom{n}{2}$ 2-blades.

Let $\overset{2}{X}=\{\gamma_1,\gamma_2,\gamma_3,\gamma_4\}$ and $\overset{1}{V}=\{\alpha_1,\alpha_2,\alpha_3\}$ be two multivectors in $\mathbb{R}^3$. A multivector $\overset{3}{Z}$ of rank 3 in $\mathbb{R}^3$ may be created in two ways: as a result of multiplying either $\overset{1}{V}$ by $\overset{2}{X}$ or $\overset{2}{X}$ by $\overset{1}{V}$. Let us concentrate on the first case

$$\overset{3}{Z}=\overset{1}{V}\overset{2}{X}=\begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\end{bmatrix}\begin{bmatrix}\gamma_1\\\gamma_2\\\gamma_3\\\gamma_4\end{bmatrix}=\begin{bmatrix}\alpha_1\gamma_1-\alpha_2\gamma_2-\alpha_3\gamma_3\\\alpha_1\gamma_2+\alpha_2\gamma_1-\alpha_3\gamma_4\\\alpha_1\gamma_3+\alpha_2\gamma_4+\alpha_3\gamma_1\\\alpha_1\gamma_4-\alpha_2\gamma_3+\alpha_3\gamma_2\end{bmatrix}. \quad (14)$$

Here, the first three entries in the resulting matrix are 1-blades, while the last entry is a 3-blade. For arbitrary multivectors of rank 1 and 2 in $\mathbb{R}^n$ we would have obtained $\binom{n}{1}$ vectors and $\binom{n}{3}$ trivectors. We cannot generate multivectors of rank higher than 3 in $\mathbb{R}^3$, but it is easy to check that in spaces $\mathbb{R}^{n>3}$ a multivector of rank 4 would have $\binom{n}{0}$ scalars, $\binom{n}{2}$ bivectors and $\binom{n}{4}$ 4-blades. The number of k-blades in a multivector of rank r is described by Table I. It becomes clear that a multivector of rank r over $\mathbb{R}^n$ is actually a vector over a $\sum_{i=0}^{\lfloor \frac{r}{2}\rfloor}\binom{n}{2i+r \bmod 2}$-dimensional space.

As an example let us consider the following roles and fillers being normalized vectors drawn randomly from $\mathbb{R}^n$ with Gaussian distribution $\mathrm{N}(0,\frac{1}{n})$

$$\begin{array}{rcl@{\qquad}rcl} Pat &=& \{a_1,\dots,a_n\}, & name &=& \{x_1,\dots,x_n\},\\ male &=& \{b_1,\dots,b_n\}, & sex &=& \{y_1,\dots,y_n\},\\ 66 &=& \{c_1,\dots,c_n\}, & age &=& \{z_1,\dots,z_n\}. \end{array} \quad (15)$$

$PSmith$, who is a 66 year old male named Pat, is created by first multiplying roles and fillers with the help of the geometric product

$$\begin{aligned} PSmith &=\\ &= name * Pat + sex * male + age * 66\\ &= name\cdot Pat + name\wedge Pat + sex\cdot male +\\ &\quad sex\wedge male + age\cdot 66 + age\wedge 66 \qquad (16) \end{aligned}$$

$$\begin{aligned} &= \begin{bmatrix} \sum_{i=1}^n (a_i x_i + b_i y_i + c_i z_i)\\ a_1x_2 - a_2x_1 + b_1y_2 - b_2y_1 + c_1z_2 - c_2z_1\\ a_1x_3 - a_3x_1 + b_1y_3 - b_3y_1 + c_1z_3 - c_3z_1\\ \vdots\\ a_{n-1}x_n - a_nx_{n-1} + b_{n-2}y_n - b_ny_{n-1} + c_{n-1}z_n - c_nz_{n-1} \end{bmatrix}\\ &= [d_0, d_{12}, d_{13},\dots,d_{(n-1)n}]^T\\ &= d_0 + d_{12}e_{12} + d_{13}e_{13} + \dots + d_{(n-1)n}e_{(n-1)n}, \qquad (17) \end{aligned}$$

where $e_1,\dots,e_n$ are orthonormal basis blades. In order to be decoded as much correctly as possible, $PSmith$ should have the same magnitude as vectors representing atomic objects, therefore it needs to be normalized. Finally, $PSmith$ takes the form of

$$PSmith = [\hat{d}_0, \hat{d}_{12}, \hat{d}_{13},\dots,\hat{d}_{(n-1)n}]^T, \quad (18)$$

where $\hat{d}_i = \frac{d_i}{\sqrt{\sum_{j=0,12}^{(n-1)n} d_j^2}}$.

$PSmith$ is now a multivector of rank 2. The decoding operation

$$\begin{aligned} &name^+ PSmith\\ &\quad = name^+(name\cdot Pat + name\wedge Pat + sex\cdot male\\ &\qquad + sex\wedge male + age\cdot 66 + age\wedge 66) \qquad (19) \end{aligned}$$

will produce a multivector of rank 3 consisting of vectors and trivectors. However, the original Pat did not contain any trivector components — they all belong to the noise part and the only interesting blades in $name^+ PSmith$ are vectors. The expected answer is a vector, therefore there is no point in

TABLE I
NUMBERS OF k-BLADES IN MULTIVECTORS OF VARIOUS RANKS IN $\mathbb{R}^n$

rank	scalars	vectors	bivectors	trivectors	4-blades	…	data size
1	0	$\binom{n}{1}$	0	0	0	…	$O(\binom{n}{1})$
2	$\binom{n}{0}$	0	$\binom{n}{2}$	0	0	…	$O(\binom{n}{0}+\binom{n}{2})$
3	0	$\binom{n}{1}$	0	$\binom{n}{3}$	0	…	$O(\binom{n}{1}+\binom{n}{3})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$2r$	$\binom{n}{0}$	0	$\binom{n}{2}$	0	$\binom{n}{4}$	…	$O(\Sigma_{i=0}^{r}\binom{n}{2i})$
$2r+1$	0	$\binom{n}{1}$	0	$\binom{n}{3}$	0	…	$O(\Sigma_{i=0}^{r}\binom{n}{2i+1})$

calculating the whole multivector $name^{+}PSmith$ and only then comparing it with items stored in the clean-up memory. To be efficient, one should generate only the vector-part while computing $name^{+}PSmith$ and skip the noisy trivectors.

Let $\langle\cdot\rangle_k$ denote the projection of a multivector on k-blades. To decode $PSmith$'s $name$ we need to compute

$$\begin{aligned}\langle name^{+}PSmith\rangle_1&\\ =\ & name^{+}namePat+\langle\ name^{+}(name\wedge Pat\\ & +sex\cdot male+sex\wedge male+age\cdot 66+age\wedge 66)\ \rangle_1\\ =\ & Pat+\text{noise}=Pat'. \qquad (20)\end{aligned}$$

The resulting Pat' will still be noisy, but to a lesser degree than it would have been if the trivectors were present.

Formally, we are using a map $*^{1}_{1,2}$ that transforms a multivector of rank 1 (i.e. an n-tuple) and a multivector of rank 2 (i.e. a $(1+\frac{(n-1)n}{2})$-tuple) into a multivector of rank 1 without computing the unnecessary blades. Let X be a multivector of rank 2

$$X=\langle X\rangle_0+\langle X\rangle_2=x_0+\sum_{l<m}x_{lm}e_le_m, \qquad (21)$$

where $x_{lm}=-x_{ml}$. If $A=(A_1,\ldots,A_n)$ is a decoding vector (actually, an inverse of a role vector), then

$$\begin{aligned}A*^{1}_{1,2}X &= x_0A+\sum_{l,m}A_lx_{lm}e_m\\ &= \sum_k\big(xA_k+\sum_l A_lx_{lk}\big)e_k\\ &= \sum_k Y_ke_k=Y, \qquad (22)\end{aligned}$$

with $Y=(Y_1,\ldots,Y_n)$ being an n-tuple, i.e. a multivector of rank 1. More explicitly,

$$Y_k=(A*^{1}_{1,2}X)_k=x_0A_k+\sum_{l=1}^{k-1}A_lx_{lk}-\sum_{l=k+1}^{n}A_lx_{kl}. \quad (23)$$

The map $*^{1}_{1,2}$ is an example of a *projected product*, introduced in [2], reconstructing the vector part of AX without computing the unnecessary parts. The projected product is basis independent, as opposed to circular convolutions. In general, $*^{m}_{l,k}$ transforms the geometric product of two multivectors $\overset{l}{A}$ and $\overset{k}{B}$ into a multivector $\overset{m}{C}$.

We now need to compare Pat' with other items stored in the clean-up memory using the dot product, and since Pat' is a vector, we need to compare only the vector part. That means, if the clean-up memory contained a multivector $\overset{2t+1}{M}$ of an odd rank, we would also need to compute $Pat'\cdot\langle\overset{2t+1}{M}\rangle_1$ while searching for the right answer.

This method of decoding suggests that items stored in the clean-up memory should hold information about their ranks, which is dangerously close to employing fixed data slots present in localist architectures. However, a rank of a clean-up memory item can be "guessed" from its size. In a distributed model we also should not "know" for sure how many parts the projected product should reject, but it can certainly reject parts spanned by blades of highest grades. Unfortunately, since the geometric product is non-commutative, questions concerning roles and fillers need to be asked on different sides of a sentence, forcing atomic objects to hold information on whether they are roles or fillers and thus, forcing them to be partly hand-generated. We can either ask question always on the same side of a sentence and be satisfied with less precise answers or always ask about only the roles or only the fillers. It becomes clear, that recognition based on the hierarchy of multivectors and the projected product is best applicable to tasks in which questions need to be asked only on one side of the sentence or in which sentences have predetermined structure.

Before providing formulas for encoding and decoding a complex statement we need to introduce additional notation for the projected product and the projection. We have already introduced the projected product $*^{m}_{l,k}$ transforming the geometric product of two multivectors of ranks l and k into a multivector of rank m. This will not always be the case for complex statements, since we can produce a multivector that will not be of any given rank. Let $*^{m}_{l,\{\alpha_1,\alpha_2,\ldots,\alpha_k\}}$ denote the projected product transforming the geometric product of a multivector $\overset{l}{A}$ and a multivector B containing α_1-blades, α_2-blades,… and α_k-blades into a multivector $\overset{m}{C}$. In this way, the projected product $*^{1}_{1,2}$ may be written down as $*^{1}_{1,\{0,2\}}$. By analogy, let $\langle\cdot\rangle_{\{\alpha_1,\alpha_2,\ldots,\alpha_k\}}$ denote the projection of a multivector on components spanned by α_1-blades, α_2-blades,… and α_m-blades.

Let Ψ denote the normalized multivector encoding the sentence "*Fido bit PSmith*", i.e.

$$\Psi = \underbrace{bite_{agt} * Fido}_{\text{rank 2}} + \underbrace{bite_{obj} * \underbrace{PSmith}_{\text{rank 2}}}_{\text{rank 3}}. \tag{24}$$

Multivector Ψ will contain scalars, vectors, bivectors and trivectors and can be written down as the following vector of dimension $\sum_{i=0}^{3} \binom{n}{i}$

$$\Psi = \underbrace{\alpha}_{\text{a scalar}} + \underbrace{\sum_{i=1}^{n} \beta_i e_i}_{\text{vectors}} + \underbrace{\sum_{1=i<j}^{n} \gamma_{ij} e_{ij}}_{\text{bivectors}} + \underbrace{\sum_{1=i<j<k}^{n} \delta_{ijk} e_{ijk}}_{\text{trivectors}} \tag{25}$$

The following example illustrates how to ask questions in the GA$_c$ architecture.

"*Who was bitten?*"

The answer to that question will be a multivector of rank 2

$$\begin{aligned} \Psi \sharp\, bite_{obj} &= \langle bite_{obj}^{+} \Psi \rangle_{\{0,2\}} = bite_{obj}^{+} *^{2}_{1,\{0,1,2,3\}} \Psi \\ &= PSmith' \approx PSmith. \end{aligned} \tag{26}$$

Let $bite_{obj} = \{y_1, \ldots, y_n\}$, $PSmith'$ will then have the form

$$\begin{aligned} PSmith' &= (y_1 e_1 + \cdots + y_n e_n) *^{2}_{1,\{0,1,2,3\}} \\ &\quad (\sum_{i=1}^{n} \beta_i e_i + \sum_{1=i<j<k}^{n} \delta_{ijk} e_{ijk}) \qquad (27) \\ &= \underbrace{\sum_{k=1}^{n} y_k \beta_k}_{\text{a scalar}} + \underbrace{\sum_{1=i<j}^{n} \theta_{ij} e_{ij}}_{\text{bivectors}}, \qquad (28) \end{aligned}$$

where

$$\theta_{ij} = y_i \beta_j - y_j \beta_i + \sum_{\substack{t=1 \\ t \notin \{i,j\}}}^{n} y_t \delta_{ijt} \tag{29}$$

with $\delta_{ijt} = \delta_{tij} = -\delta_{itj}$. As previously, $PSmith'$ should be compared with appropriate items from the clean-up memory to produce the most probable answer.

III. Overview of Plate's scaling test

Plate [13] describes a simulation in which approximately 5000 HRR 512-dimensional vectors were stored in the clean-up memory. The purpose of his simulation was to study efficiency of the HRR model but also to provide a counterexample to the claim that role-filler representations do not permit one component of a relation to be retrieved given the others. We will repeat Plate's test on several models and compare the results.

Let us consider the following atomic objects

$$\left.\begin{array}{l} num_x \text{ (for } x = 0, \ldots, 2500), \\ times, \\ plus, \end{array}\right\} \text{fillers}, \tag{30}$$

$$\left.\begin{array}{l} result, \\ operand. \end{array}\right\} \text{roles} \tag{31}$$

At the beginning of the scaling test, relations concerning multiplication and addition are constructed. For example, "$2 \cdot 3 = 6$" is constructed as

$$times_{2,3} = times + operand * (num_2 + num_3) + result * num_6. \tag{32}$$

Generally, relations are constructed in the following way

$$\begin{aligned} times_{x,y} &= times + operand * (num_x + num_y) \\ &\quad + result * num_{x \cdot y}, \qquad (33) \\ plus_{x,y} &= plus + operand * (num_x + num_y) \\ &\quad + result * num_{x+y}. \qquad (34) \end{aligned}$$

x and y range from 0 to 50 with $y \leq x$ making a total of 2501 number vectors and 2652 instances of each $times_{x,y}$ and $plus_{x,y}$. As one can notice, the same $operand$ role is used for both x and y to preserve commutativity of multiplication and addition.

Plate writes, that a relation can be "looked up" by supplying enough information to distinguish a specific relation from others. For example, to look up "$2 \cdot 3 = 6$" one needs to find the most similar relation R to any of the following fragmentary statements

$$\begin{array}{lll} \text{(case 1)} & times + operand * num_2 & \\ & +operand * num_3, & (35) \\ \text{(case 2)} & times + operand * num_2 & \\ & +result * num_6, & (36) \\ \text{(case 3)} & times + operand * num_3 & \\ & +result * num_6, & (37) \\ \text{(case 4)} & operand * num_2 + operand * num_3 & \\ & +result * num_6. & (38) \end{array}$$

Retrieving the missing piece of information in the first three cases can be done by asking any of the subquestions

$$\begin{array}{lll} \text{(case 1)} & R \sharp\, result, & (39) \\ \text{(case 2)} & R \sharp\, operand, & (40) \\ \text{(case 3)} & R \sharp\, operand. & (41) \end{array}$$

Case 4 is somewhat more problematic — to look up a missing relation name ($times$ or $plus$) one needs to have a separate clean-up memory containing only relation names or to use an alternative encoding in which there is a role for relation names. We will alter Plate's test by using the latter method.

Plate states that he had tried one run of the system making a query for each component missing in every relation — this amounted to 10608 queries. A further 7956 queries had been made to decode the missing component except for the relation name. Plate goes on to claim, that the system made no errors.

There appear to be two misstatements in Plate's claims. Firstly, we cannot treat subquestions regarding cases 2 and 3 separately, as there are two equally probable answers to each of these subquestions, provided that relations R_2 and R_3 point correctly to $times_{x,y}$. Secondly, consider a fragmentary piece of information

$$times + operand * num_0 + result * num_0. \quad (42)$$

In this situation, the missing component can be any of the numbers num_x where $x \in \{0, \dots, 50\}$ and thus, there are 51 atomic objects that are equally probable to be the right answer. This suggests that Plate regards several answers as valid ones, as long as the similarity of these answers exceeds some threshold. To work out the missing component, one then needs to check which of those potential answers is not in the original set used for retrieval.

Such a method of investigating scaling properties has more than a few advantages:

- Inaccuracies mentioned above act as a test if all atomic objects are created and treated equally. Ideally, every atomic object of the num_x form should be recognized as a correct answer to the "zero problem" for $\frac{\text{number of trials}}{51}$. 100% of the time.
- Prime numbers greater than 100 do not appear in any of $times_{x,y}$ and $plus_{x,y}$ relations, therefore they test if the model is immune to garbage data.
- Numbers ranging from num_0 to num_{100} may be constructed in a multitude of ways by addition (num_0 by multiplication) and any given sentence chunk $result * num_z$ will appear quite often in the $plus_{x,y}$ relation. Hence, this is a great way of checking if the model deals with excessive similarity of a number of complex statements.
- Atomic objects bound with $operand$ and $result$ range in variety. On the other hand, there are just two atomic objects acting as an $operation$ — does it affect in any way the recognition of $operation$ filler? Indeed, it will be shown in Section V that recognition of the $operation$ chunk turns out to be quite interesting depending on the choice of the architecture.

IV. NOTATION

For the purpose of explaining test results, let us introduce the following notation. Let $S^*_{x,y}$ and $S^+_{x,y}$ denote relations

$$\begin{aligned} S^*_{x,y} &= operation * times + operand * (num_x + num_y) \\ &\quad + result * num_{x \cdot y}, \quad (43) \\ S^+_{x,y} &= operation * plus + operand * (num_x + num_y) \\ &\quad + result * num_{x+y}, \quad (44) \end{aligned}$$

for $y \leq x$. We chose to use a separate role for a relation name to enable encoding the information given only operands and the result. Let $F^{op}_{i,x,y}$ denote fragmentary statements for $i \in \{1, 2, 3, 4\}$ and $op \in \{*, +\}$

$$\begin{aligned} F^{op}_{1,x,y} &= S^{op}_{x,y} - result * num_{x\ op\ y}, \quad (45) \\ F^{op}_{2,x,y} &= S^{op}_{x,y} - operand * num_x, \quad (46) \\ F^{op}_{3,x,y} &= S^{op}_{x,y} - operand * num_y, \quad (47) \\ F^{op}_{4,x,y} &= S^{op}_{x,y} - operation * op. \quad (48) \end{aligned}$$

If v is an element of the clean-up memory, then let $N(v)$ denote the closest $neighbor$ of v, i.e. an element of the clean-up memory that is most similar to v. If v has more than one neighbor, then all subquestions during the test are asked to all of v's neighbors. In HRR, GA$_d$ (with the Hamming measure of similarity) and GA$_c$ it is extremely unlikely for an element of the clean-up memory to have more than one neighbor due to the continuous nature of data in these architectures. Let $Q^{op}_{i,x,y} = N(F^{op}_{i,x,y})$ for $i \in \{1, 2, 3, 4\}$ and $op \in \{*, +\}$. During the test we asked subquestions concerning components missing in $F^{op}_{i,x,y}$ and obtained the following (sets of) answers

$$\begin{aligned} q^{op}_{1,x,y} &= N(Q^{op}_{1,x,y} \sharp\ result), \quad (49) \\ q^{op}_{2,x,y} &= N(Q^{op}_{2,x,y} \sharp\ operand), \quad (50) \\ q^{op}_{3,x,y} &= N(Q^{op}_{3,x,y} \sharp\ operand), \quad (51) \\ q^{op}_{4,x,y} &= N(Q^{op}_{4,x,y} \sharp\ operation). \quad (52) \end{aligned}$$

We assume that a missing component is identified correctly if it is the only neighbor to appropriate answer $q^{op}_{\cdot,x,y}$ or it belongs to the set of neighbors of $q^{op}_{\cdot,x,y}$.

V. TEST RESULTS

The software for all tests was developed by A. Patyk-Łońska in Java language. All tests were performed on an ordinary PC with dualcore AMD processor with 2 GB RAM.

Tables II through IV compare scaling test results for

- GA$_c$ and HRR, both using dot-product as a similarity measure.
- BSC using Hamming distance as a similarity measure.

Although BSC and HRR models need only n-dimensional vectors, this is not quite the case for and GA$_c$, which needs $1 + \frac{n(n-1)}{2}$ numbers to represent multivectors of rank 2 over $\mathbb{R}^n$. We present recognitions test results close to 100% and comment on vector length required for each model to achieve such percentage. The real number of memory cells used up by each model is given in brackets in the table headings.

The answers to subquestions $Q^{op}_{2,x,y} \sharp\ operand$ and $Q^{op}_{3,x,y} \sharp\ operand$ were considered to be correct if any of the two possible operands came up as the item most similar to those subquestions. In case of other questions and subquestions only exact answers were taken into consideration.

50 runs of the test were performed on each model. Unlike in Plate's test, x and y ranged from 0 to only 20. Hence, there are 401 number vectors and 462 relation vectors.

The "zero problem" is clearly visible in each tested model, as the recognition percentage of $Q^*_{3,x,y}$ barely exceeds 90%. Nevertheless, $Q^*_{3,x,y}$ almost always contains at least one of the operands from the original sentence $S^*_{x,y}$ since the recognition percentage of $q^*_{3,x,y}$ reaches 100% for sufficiently large data size. On the whole, the recognition percentage of $q^*_{2,x,y}$ and $q^*_{3,x,y}$ does not differ greatly from the recognition percentage of $q^+_{2,x,y}$ and $q^+_{3,x,y}$ in any model.

Table entries marked with a "$^\Delta$" indicate that despite the wrong recognition of a fragmentary sentence, the missing component has been identified correctly. In all tested models such situations arise for sentences with one of the operands missing.

TABLE II
RECOGNITION PERCENTAGE FOR GA_c.

Questions	$\mathbf{R}^{10}$ (46)	$\mathbf{R}^{20}$ (191)	$\mathbf{R}^{30}$ (436)	$\mathbf{R}^{40}$ (781)
$Q^*_{1,x,y}$	89.76%	99.98%	99.99%	100.0%
$q^*_{1,x,y}$	39.44%	95.28%	99.58%	99.88%
$Q^*_{2,x,y}$	91.12%	99.73%	99.98%	100.0%
$q^*_{2,x,y}$	36.24%	83.86%	97.92%	99.81%
$Q^*_{3,x,y}$	83.97%	91.15%	91.33%	91.34%
$q^*_{3,x,y}$	41.27%	84.92%	98.05%$^{\triangle}$	99.82%$^{\triangle}$
$Q^*_{4,x,y}$	98.90%	99.60%	99.63%	99.59%
$q^*_{4,x,y}$	42.01%	95.56%	99.24%	99.52%
$Q^+_{1,x,y}$	89.39%	99.99%	100.0%	100.0%
$q^+_{1,x,y}$	39.09%	95.99%	99.76%	99.95%
$Q^+_{2,x,y}$	86.96%	99.59%	99.96%	100.0%
$q^+_{2,x,y}$	35.32%	83.84%	97.97%	99.79%
$Q^+_{3,x,y}$	87.00%	99.63%	99.96%	100.0%
$q^+_{3,x,y}$	35.12%	83.84%	97.98%	99.79%
$Q^+_{4,x,y}$	99.05%	99.53%	99.51%	99.54%
$q^+_{4,x,y}$	45.84%	94.73%	99.14%	99.49%

TABLE III
RECOGNITION PERCENTAGE FOR HRR.

Questions	$N = 200$	$N = 300$	$N = 400$	$N = 500$
$Q^*_{1,x,y}$	29.1%	27.06%	26.28%	28.51%
$q^*_{1,x,y}$	31.08%$^{\triangle}$	30.03%$^{\triangle}$	30.30%$^{\triangle}$	32.23%$^{\triangle}$
$Q^*_{2,x,y}$	54.72%	52.06%	53.10%	53.32%
$q^*_{2,x,y}$	98.99%$^{\triangle}$	99.92%$^{\triangle}$	99.98%$^{\triangle}$	100.0%$^{\triangle}$
$Q^*_{3,x,y}$	50.53%	47.93%	49.80%	51.21%
$q^*_{3,x,y}$	98.92%$^{\triangle}$	99.90%$^{\triangle}$	99.97%$^{\triangle}$	100.0%$^{\triangle}$
$Q^*_{4,x,y}$	89.23%	90.56%	90.51%	90.29%
$q^*_{4,x,y}$	90.28%$^{\triangle}$	92.69%$^{\triangle}$	92.42%$^{\triangle}$	92.31%$^{\triangle}$
$Q^+_{1,x,y}$	28.26%	29.46%	28.03%	28.81%
$q^+_{1,x,y}$	27.32%	29.37%	28.02%	28.80%
$Q^+_{2,x,y}$	53.91%	54.48%	55.26%	54.68%
$q^+_{2,x,y}$	98.72%$^{\triangle}$	99.90%$^{\triangle}$	99.99%$^{\triangle}$	99.99%$^{\triangle}$
$Q^+_{3,x,y}$	53.73%	55.23%	55.34%	54.62%
$q^+_{3,x,y}$	98.67%$^{\triangle}$	99.91%$^{\triangle}$	99.98%$^{\triangle}$	100.0%$^{\triangle}$
$Q^+_{4,x,y}$	98.70%	98.75%	98.66%	98.75%
$q^+_{4,x,y}$	97.16%	98.55%	98.64%	98.74%

TABLE IV
RECOGNITION PERCENTAGE FOR BSC.

Questions	$N = 200$	$N = 300$	$N = 400$	$N = 500$
$Q^*_{1,x,y}$	86.71%	91.65%	93.78%	94.74%
$q^*_{1,x,y}$	82.82%	90.62%	93.87%$^{\triangle}$	94.95%$^{\triangle}$
$Q^*_{2,x,y}$	94.42%	97.60%	99.03%	99.44%
$q^*_{2,x,y}$	99.68%$^{\triangle}$	99.97%$^{\triangle}$	99.98%$^{\triangle}$	100.0%$^{\triangle}$
$Q^*_{3,x,y}$	86.87%	89.43%	90.50%	90.97%
$q^*_{3,x,y}$	99.15%$^{\triangle}$	99.47%$^{\triangle}$	99.65%$^{\triangle}$	100.0%$^{\triangle}$
$Q^*_{4,x,y}$	94.39%	95.58%	95.39%	95.50%
$q^*_{4,x,y}$	90.78%	94.89%	95.22%	95.44%
$Q^+_{1,x,y}$	86.38%	91.59%	93.65%	94.71%
$q^+_{1,x,y}$	81.71%	90.28%	93.27%	94.57%
$Q^+_{2,x,y}$	94.23%	97.77%	99.19%	99.52%
$q^+_{2,x,y}$	99.36%$^{\triangle}$	99.94%$^{\triangle}$	100.0%$^{\triangle}$	100.0%$^{\triangle}$
$Q^+_{3,x,y}$	94.54%	97.39%	98.77%	99.48%
$q^+_{3,x,y}$	99.41%$^{\triangle}$	99.94%$^{\triangle}$	100.0%$^{\triangle}$	100.0%$^{\triangle}$
$Q^+_{4,x,y}$	95.40%	95.38%	95.65%	95.66%
$q^+_{4,x,y}$	91.81%	94.27%	95.02%	95.27%

For HRR, however the missing item has been "accidentally" correctly identified also in cases of missing $operation * times$ and $result * times_{x,y}$ components. Such recognition did not occur in cases of missing $operation*plus$ and $result*plus_{x,y}$ components, which is distressingly asymmetric.

HRR turned out to be the worst model during this experiment. The recognition percentage of $Q^*_{1,x,y}$ and $Q^+_{1,x,y}$ is dangerously low when compared to other Q's. Both $Q^*_{1,x,y}$ and $Q^+_{1,x,y}$ are retrieved from the clean-up memory given only two operands and the operation type. Since we have only two operation types, $Q^*_{1,x,y}$ and $Q^+_{1,x,y}$ will not differ greatly from each other. This phenomenon is also observable in BSC (but not in GA$_c$), where the recognition percentage of Q_1's is only slightly lower than that of the other Q's. Apart from that weakness, BSC performs as well as GA$_c$ for adequate data size.

VI. Conclusion

Authors developed a new model of distributed representations based on geometric algebra. Although the data representations of sentences encoded in this model may have varying lengths (as opposed to HRR and BSC), it can be justified by the fact that it is quite logical for sentences that hold more information to have larger "volume".

Tedious calculations presented in Section 2 imply that the GA$_c$ model is best applicable to sentences having a similar or identical complexity structure, otherwise it may be hard to make the process of asking questions and retrieving answers automatic. Because of this limitation, this construction seems to be a promising candidate for a holographic database.

Acknowledgment

This work was supported by grant G.0405.08 of the Research Programme of the Research Foundation-Flanders (FWO, Belgium)

References

[1] D. Aerts and M. Czachor, "Tensor-product versus geometric-product coding", *Physical Review A*, vol. 77, id. 012316, 2008.

[2] D. Aerts, M. Czachor, and B. De Moor, "Geometric Analogue of Holographic Reduced Representation", *J. Math. Psychology*, vol. 53, pp. 389-398, 2009.

[3] D. Aerts, M. Czachor, and B. De Moor, "On geometric-algebra representation of binary spatter codes". preprint arXiv:cs/0610075 [cs.AI], 2006.

[4] D. Aerts, M. Czachor, and Ł. Orłowski, "Teleportation of geometric structures in 3D ", *J. Phys. A* vol. 42, 135307, 2009.

[5] W.K. Clifford, "Applications of Grassmann's extensive algebra", *American Journal of Mathematics Pure and Applied*, vol. 1, 350–358, 1878.

[6] R. W. Gayler, "Multiplicative binding, representation operators, and analogy", *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*, K. Holoyak, D. Gentner, and B. Kokinov, eds., Sofia, Bulgaria: New Bulgarian University, p. 405, 1998.

[7] H. Grassmann, "Der Ort der Hamilton'schen Quaternionen in der Ausdehnungslehre", *Mathematische Annalen*, vol. 3, 375–386, 1877.

[8] P. Kanerva, "Binary spatter codes of ordered k-tuples". In C. von der Malsburg et al. (Eds.), *Artificial Neural Networks ICANN Proceedings, Lecture Notes in Computer Science* vol. 1112, pp. 869-873, 1996.

[9] P. Kanerva, "Fully distributed representation". *Proc. 1997 Real World Computing Symposium (RWC97, Tokyo)*, pp. 358-365, 1997.

[10] N.G. Marchuk, and D.S. Shirokov, "Unitary spaces on Clifford algebras", *Advances in Applied Clifford Algebras*, vol 18, pp. 237-254, 2008.

[11] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2000.

[12] A. Patyk, "Geometric Algebra Model of Distributed Representations", in *Geometric Algebra Computing in Engineering and Computer Science*, E. Bayro-Corrochano and G. Scheuermann, eds. Berlin: Springer, 2010. Preprint arXiv:1003.5899v1 [cs.AI].

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