A vector representation of Fluid Construction Grammar using Holographic Reduced Representations

Yana Knight[1] and
Michael Spranger[2] and
Luc Steels[1]
[1] Artificial Intelligence Laboratory,
Free University of Brussels (VUB)
Pleinlaan 2, 1050 Brussels, Belgium
Dr. Aiguader 88, Barcelona 08003, Spain
[2] Sony Computer Science Laboratories
3-14-13 Higashigotanda, 141-0022 Tokyo, Japan

Abstract

The question of how symbol systems can be instantiated in neural network-like computation is still open. Many technical challenges remain and most proposals do not scale up to realistic examples of symbol processing, for example, language understanding or language production. Here we use a top-down approach. We start from Fluid Construction Grammar, a well-worked out framework for language processing that is compatible with recent insights into Construction Grammar and investigate how we could build a neural compiler that automatically translates grammatical constructions and grammatical processing into neural computations. We proceed in two steps. FCG is translated from symbolic processing to numeric processing using a vector symbolic architecture, and this numeric processing is then translated into neural network computation. Our experiments are still in an early stage but already show promise.

Keywords: Vector Symbolic Architectures; Fluid Construction Grammar; Connectionist Symbol Processing

Introduction

Since the early days of cognitive science in the late nineteen fifties, there has been a struggle to reconcile two approaches to model intelligence and cognition: a symbolic and a numeric one. The symbolic approach postulates an abstract layer with symbols, symbolic structures, and operations over these symbolic structures, so that it is straightforward to implement the kind of analysis that logicians, linguists, and psychologists tend to make. AI researchers have built remarkable technology to support such implementations based on high level 'symbolic' languages like LISP.

The numeric approach wants to look at cognitive processing in terms of numeric operations. It is motivated by the fact that biological neuronal networks are dynamical systems and that numeric processing can model self-organizing processes. So the numeric approach tries to get intelligent behavior without needing to postulate symbolic structures and operations explicitly. There have been several waves exploiting this numeric approach under the head of neural networks and most recently deep learning.

The symbolic approach has proven its worth in modeling very large scale language systems, search engines, problem solvers, models of expert knowledge, ontological and episodic memory, etc., but most of these applications rely heavily on a human analyst who identifies the relevant symbols and symbol processing operations. It is usually claimed that the symbolic approach is unable to deal with learning and grounding, but this criticism often ignores work within the large field of (symbolic) machine learning and work on grounding symbolic representations in perception and action by physical robots. While the numeric approach has proven its worth in the domains of pattern recognition which includes feature extraction, category formation, and pattern detection, it has not been equally successful in the implementation of 'true' physical symbol systems (Newell & Simon, 1976). More specifically, it turns out to be non-trivial to represent a group of properties of an object (a feature structure), to compare feature-structures to each other, and to handle variable binding and feature structure merging - all operations which many researchers have argued to be necessary for intelligence. We believe the symbolic and the numeric approach can only be reconciled when they are viewed as two levels of description of the same system whereby the former describes and models natural objects at a higher level than the latter. Each level has its own abstractions at which regularities are revealed and each own laws of operation. It is necessary and highly interesting to find out how the different levels map to each other. This paper sets some small steps in this direction. We do not go immediately from the symbol level to the numeric level but rather use a two-step process: mapping the symbolic level to a symbolic vector layer, as suggested by several researchers (Hinton, 1990; Neumann, 2001; Plate, 1994; Gayler, Levy, & Bod, 2010) and then mapping this layer to a possible neural implementation level in terms of populations of neurons, which has also been explored already in (Eliasmith, 2013).

This paper focuses only on the first step. Experiments have also been done for the second step using the Nengo framework (Eliasmith, 2013) but are not reported here. The paper begins by introducing Fluid Construction Grammar as a challenging test case for studying how to map symbolic processing to numeric processing. It then proceeds to describe a potential approach for the translation of FCG to vector form, namely Holographic Reduced Representations (HRR). Finally, it presents the results of experiments using HRR to produce a vector representation of FCG feature structures and core operators.

FCG and its key operations

Fluid Construction Grammar is a computational platform for implementing construction grammars (Steels, 2011). It is a typical example of a complex symbol system addressing a core competence of the human brain, namely the representation and processing (comprehension, production, learning) of language. FCG was originally designed for modeling language learning and language change (Steels, 2012), and language-based robot interaction (Steels & Hild, 2012). More recently research has focused on challenging problems in linguistics and broader coverage grammars. The components of FCG are symbols, feature structures, transient structures and constructions.

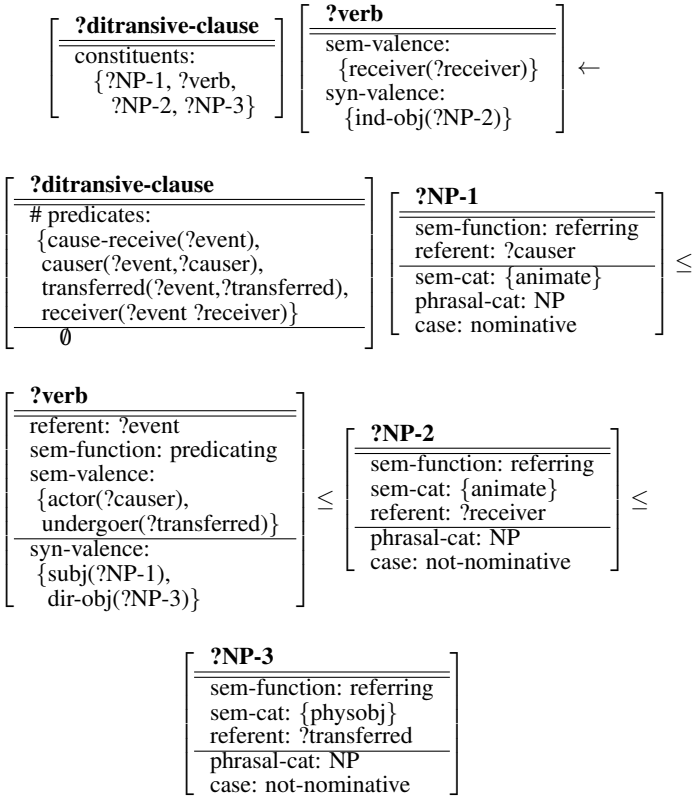
Symbols are the elementary units of information. They stand in for syntactic categories (like 'noun' or 'plural'), semantic categories (like 'animate' or 'future'), unit-names (e.g. 'noun-phrase-17'), grammatical functions (like 'subject' or 'head'), ordering relations of words and phrases (e.g. 'meets' or 'preceeds'), meaning-predicates, etc. A basic grammar of a human language like English would certainly feature thousands of such symbols, and the set of meaning-predicates is basically open-ended. Symbols can be bound to variables, which are written as names with a question-mark in front as: ?unit, ?gender, ?subject, etc. Symbol names are chosen to make sense for us but of course the FCG interpreter has no clue what they mean. The meaning of a symbol only comes from its functions in the rest of the system.

Feature structures are a way to group information about a particular linguistic unit, for example, a word or a phrase. A feature structure has a name to index it (which is again a symbol, possibly a variable) and a set of features and values. Construction grammars group all features of a unit together, whatever the level. So a feature structure has phonetic and phonologic features, morphological information, syntactic and semantic categories, pragmatic information, as well as structural information about the many possible relations between units (constituent structure, functional structure, argument structure, information structure, temporal structure, etc.). All of these are represented explicitly using features and values. The values of a feature can be elementary symbols, sets of symbols (e.g. the constituents of a phrase form a set), sequences or feature structures, thus allowing a hierarchically structured feature structure.

Feature structures are used to represent **transient structures**. These are the structures built up during comprehension and production. The features are grouped into a semantic pole, which contains the more semantic oriented features, including pragmatics and semantic categorisations, and a syntactic pole, which contains the form-oriented features. For comprehension, the initial transient structure contains all the information that could be gleaned from the form of the utterance by perceptual processes and then this transient structure is progressively expanded until it contains enough information to interpret the utterance. For production, the initial transient structure contains the meaning to be expressed and then this structure is transformed until enough information is present to render a concrete utterance. There are often multiple ways to expand a transient structure so a search space is unavoidable.

Constructions are also represented as feature structures and they are more abstract than transient structures. They typically contain variables that can be bound to the elements of transient structures and they contain less information about some of the units. Constructions have a conditional part which has to *match* with the transient structure they try to expand and a contributing part which they add to the transient structure if the conditional part matches. The conditional part is decomposed into a production lock which constrains the activation of a construction in production and a comprehension lock which constrains the construction in comprehension. When the lock fits with the transient structure, all information from the construction which is not there yet is *merged* into the transient structure. So match and merge are the most basic fundamental operations of the grammar.

Here is a simplified example of the double object construction (Goldberg, 1995) handling phrases like "she gave him a book". It has a unit for the clause as a whole (?ditransitive-clause) and for the different constituents (?NP-1, ?verb, ?NP-2 and ?NP-3). The conditional part is on the right-hand side of the arrow and the contributing part on the left-hand side. Units in the conditional part have a comprehension lock (on top) and a production lock (below). The $\leq$ sign between units means 'immediately preceeds'.



A regular speaker of a language knows probably something on the order of half a million constructions. So it is not possible to simply throw them in one bag and try constructions randomly. FCG therefore features various mechanisms to fine-tune which construction should be selected and, if more than one construction matches, which one should be pursued further. They include priming networks, organisation of constructions into sets, partial orderings, a scoring mechanism, footprints preventing some constructions from becoming active, etc.

Obviously Fluid Construction Grammar is a sophisticated computational formalism but all the mechanisms it proposes are absolutely necessary to achieve accurate (as opposed to approximate) comprehension and correct production of utterances given a particular meaning. Due to the space limitations, the reader is referred to the rapidly growing literature on FCG for more details (see also emergent-languages.org).

Holographic Reduced Representations

The kind of structures used in FCG can be represented using the AI techniques provided by symbolic programming languages such as LISP. It is very non-trivial to implement FCG but doable and adequate implementations exist. We now come to the key question: Can similar mechanisms also be implemented using a numeric approach? This means that the basic elements of FCG are encoded in a numeric format and the basic FCG-operations are translated into numeric operations over them. Various efforts to implement symbolic systems in neural terms have already been undertaken (Shastri & Ajjanagadde, 1993). The key problem however is scaling: The number of neurons required to represent the millions of symbols in human-scale grammars so far becomes biologically totally unrealistic (Eliasmith, 2013).

Vector-based approaches known as Vector Symbolic Architectures (VSA) have demonstrated promising results for representing and manipulating symbolic structures using distributed representations. Smolensky's tensor product is one of the simplest variants of VSA (Smolensky, 1990). However, the main problem with this approach is that a tensor binding results in an n^2-dimensional vector and in case of recursive representations does not scale well (Eliasmith, 2013). Alternative approaches have been suggested, such as Binary Spatter Codes (Kanerva, 1997) and Holographic Reduced Representations (HRR) (Plate, 1994), where binding is done with circular convolution, which results in a n-dimensional vector, so that the number of dimensions does not increase. Hinton explored distributed representations based on the idea of reduced representations (Hinton, 1990) and later (Neumann, 2001) demonstrated that connectionist representational schemes based on the concept of reduced representation and on the functional composition of hierarchical structures can support structure-sensitive processes which show a degree of systematicity. VSAs provide a means for representing structured knowledge using distributed vector representations and as such provide a way to translate symbolic to vector representations (Eliasmith, 2013). Since vectors can be used by many machine learning methods (for example, neural networks, support-vector machines, etc.), once a symbol system has been translated to a vector space architecture, a subsequent implementation of such a system in numeric terms should give us access to the machine learning methods associated with distributed representations.

Given the claims made by these various authors, we decided to explore VSA, more specifically Holographic Reduced Representations (HRR), for implementing Fluid Construction Grammar, and then further translate this representation using existing neural mappings (Eliasmith, 2013). The remainder of this section reflects on what is required for the mapping from FCG to VSA.

Representing FCG entities

Symbols A symbol in FCG can be mapped to a randomly generated n-dimensional vector. All the elements of the vectors are drawn from a normal distribution $\mathcal{N}(0,1/n)$ following (Plate, 1994). The symbol and its symbol vector are stored in an error-correction memory as explained later.

Feature-value pairs A feature-value pair (the primary component of a feature structure) can be mapped to a circular convolution of two vectors, the feature vector and the value vector. Following (Plate, 1994), we define convolution as follows:

$$Z = X \otimes Y \tag{1}$$

$$z_j = \sum_{k=0}^{n-1} x_k \, y_{j-k} \qquad for \;\; j = 0, ..., n-1 \tag{2}$$

Once we have a combined feature-value vector, we can, given the feature, extract the value, using circular correlation (Plate, 1994; Neumann, 2001), which convolves the pair with the approximate inverse of the feature vector (Plate, 1994):

$$X = Z \oplus Y \tag{3}$$

$$x_j = \sum_{k=0}^{n-1} z_k \, y_{j+k} \qquad for \;\; j = 0, ..., n-1 \tag{4}$$

Feature-set A feature-set consists of a set of feature-value pairs. This can be mapped to HRR using vector addition (Plate, 1994):

$$Z = X \otimes Y + T \otimes S \tag{5}$$

Feature structures Feature structures in FCG consist of feature-sets combined into units. Each unit has a unique name which is stored as a symbol in the symbol memory. A feature structure is constructed in the same way as a feature-set, i.e by convolution and addition, except that to also include units, we now convolve unit-feature-value *triples* rather than feature-value *pairs*. The feature structure is the addition of all triples:

$$Z = U \otimes X \otimes Y + U \otimes T \otimes S \tag{6}$$

Since we can represent feature structures, we can also represent transient structures as well as constructions of arbitrary length.

Parts of the structure (also called a trace) can be retrieved using the correlation operation (Equation 3). For example, given $U \otimes X$ and the whole structure, we can obtain Y.

However, correlation on traces is noisy. A trace preserves just enough information to recognise the result but not to reconstruct it. Therefore, we need an error-correction memory that stores vectors for possible units, features and values. The memory is used to compare the noisy output of the correlation operation with all vectors known to the system. Various comparison measures can be used, however, the most standard one is dot product, which for two normalized vectors is equal to the cosine of their angle Neumann (2001). We define the following similarity for two vectors:

$$\mathrm{sim}(X,Y) = \frac{X \odot Y}{||X||||Y||} \tag{7}$$

This similarity is used to retrieve the vector stored in the error-correction memory with the highest similarity to the output of correlation. That vector represents the most plausible value of the feature in a particular trace.

Matching and Merging

The promise of distributed representations is that they can do very fast operations over complete feature structures (such as comparing them) without traversing the components as would be done in a symbolic implementation. Let us see how far we might be able to get without decomposition. FCG basically needs (i) the ability to copy a feature structure, (ii) to compare two feature structures (matching) and (iii) to merge a feature structure with another one.

Copying It is not difficult to copy two feature structures because it means to copy the two vectors. However we often need to replace all variables in a feature structure either by new variables (e.g. when copying a construction before applying it) or by their bindings (e.g. when creating the expanded transient structure after matching). It has been suggested that copy-with-variation can be done by convolving the current structure A with a transformation vector T (Plate, 1994):

$$A \otimes T = B \tag{8}$$

The transformation vector is first constructed by convolving the new values with the inverse of the current values, then adding up the pairs by vector addition. For example, in order to set the value of the lex-cat feature with the current value ?x to the new value which is the binding of ?x, e.g. noun, the inverse of ?x should be convolved with noun. The full transformation vector is

$$x' \otimes y + z' \otimes w \tag{9}$$

Such vectors can be hand-constructed (Plate, 1994) – which is not desirable – or learnt from examples as shown in (Neumann, 2002).

Matching In general, matching two feature structures can be done by the same principle that is used in the error-correcting memory, i.e. similarity (Equation 7). Since we use the dot product as our similarity measure, we have a computationally fast operation, which is well understood mathematically. Using dot product provides us with a way to compare a feature structure to every structure in a pool of structures and to find the structure with the highest similarity as the closest match. However this ignores two problems we have not tackled yet: (i) Match in FCG is an includes rather than similarity operation: if the source structure is a subset of the target structure, they should still match, even if the similarity between the two structures is low. In fact, this is very common because the lock of a construction is always a subset of the transient structure, and (ii) This does not take variable bindings yet into account.

Merging Merging two feature structures is straightforward because their respective vector representations can simply be added. It is possible to deal with variable bindings by first transforming both feature structures by replacing the variables by their bindings as discussed earlier. However, there are also some tricky issues to be resolved (e.g. variables may be bound to other variables making a variable-chain and then one of these has to be substituted for all the others).

Preliminary implementation experiments

We now report on first steps in implementing the FCG→VSA mapping described above. Experiments were carried out in Python using the mathematical extension numpy.

Feature encoding and retrieval First, we tested the precision of value retrieval from a feature set and a feature structure. We were particularly interested in the relationship between HRR vector dimensionality, length of FCG feature structure and retrieval accuracy. We therefore tested different lengths of FCG feature sets/structures (5, 50, 500) vs dimensionality of HRR vectors (10, 100, 1,000 etc). We did 100 runs for each combination (results averaged). Each time HRR vectors for individual features were random-initialized and combined into a feature-structure representing HRR vectors using convolution and addition. Then we attempted to retrieve all feature values and measured the number of correct retrievals divided by the original FCG feature sets length. Figure 1 (top) illustrates how precision score increases with vectors of higher dimensionality, consistent with previous experiments with HRR (Neumann, 2001). To encode FCG feature sets with an average length of about 50-100 features, we required around 3,000-dimensional HRR vectors. This figure also illustrates how differences in HRR vector dimensionality are related to the cardinality of the feature-set. For example, in order to represent and successfully retrieve all values from a 5-pair set, around 300 dimensions appears to be sufficient, while a 500-pair feature-set requires just over 30,000. Our feature-values pairs behave in accordance with (Plate, 1994), which can described as follows:

$$n = 3.16(k - 0.25) ln \frac{m}{q^3} \tag{10}$$

where n is a lower bound for the dimensionality of the vectors in order for retrieval to have a probability of error q; k is the number of pairs in a trace and m is the number of vectors in the error-correction memory. For example, to have a q of 10^{-1} in a 5-pair trace with 1,500 items in the error-correction memory, n should be approximately 213. For a smaller q of 10^{-2}, around 300 dimensions is required. This roughly follows the n and q observed and illustrated in Figure 1 (top).

Feature structures (triples) behave similarly to pairs although dimensions required to encode triples increase. Figure 1 (bottom) illustrates how both feature sets and feature structures scale for various structure sizes (5;10;50;100;500;1,000 pairs/triples). These results can be directly translated to FCG. A toy grammar starts at 1-5 units per construction with 1-10 feature-value pairs in each. A more complex grammar can have around 10 units with approximately the same number of pairs in each unit. Represented as triples, such structures can be encoded in vectors of around 6,000 dimensions. Really large grammars of 30 units and 30 feature-values pairs in each unit require roughly 100,000 dimensions.

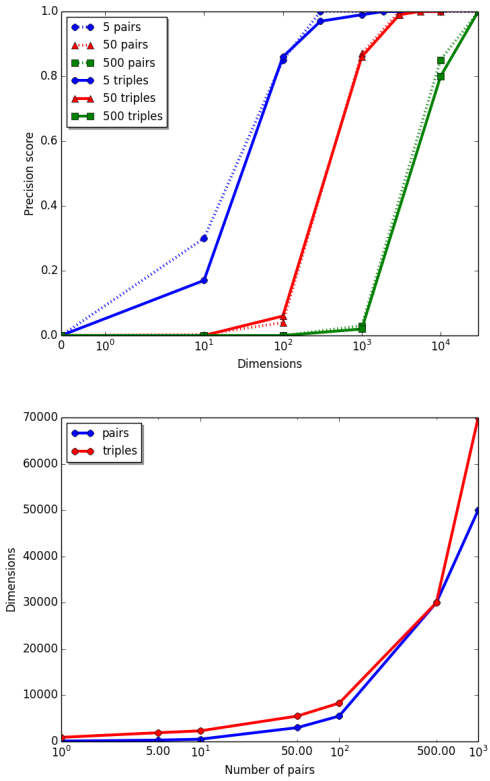


Figure 1: Top: The effects of dimensionality on precision scores in feature sets and structures of various length. Bottom: Scaling of sets and structures (vector dimensionality at which precision scores become 1.0).

Matching We numerically investigated if HRR representations can be used to implement the FCG match operation in two phases: We investigated changes in sim(X,Y) under various conditions working with feature sets (rather than feature structures) for simplicity.

First, we investigated how similarity (Equation 7) between two HRR vectors responds to structural changes vs changes in underlying feature values. Figure 2 (top, bottom) shows that changes to feature values result in a greater decrease in similarity (reaching 0.0 after 10^2 for a 100-pair structure) than structural changes i.e. adding new pairs, which led to a more gradual similarity degradation (reaching 0.0 after 10^5 for the same structure size). The difference between these two types of changes is important in FCG, where structures can be structurally different and still match, while structures that for example, differ in feature values, should not. This finding is also in line with previous experiments comparing HRR structures using dot product (Plate, 1994), where similarity was more sensitive to the content of feature-value pairs rather than the quanitiy of pairs.

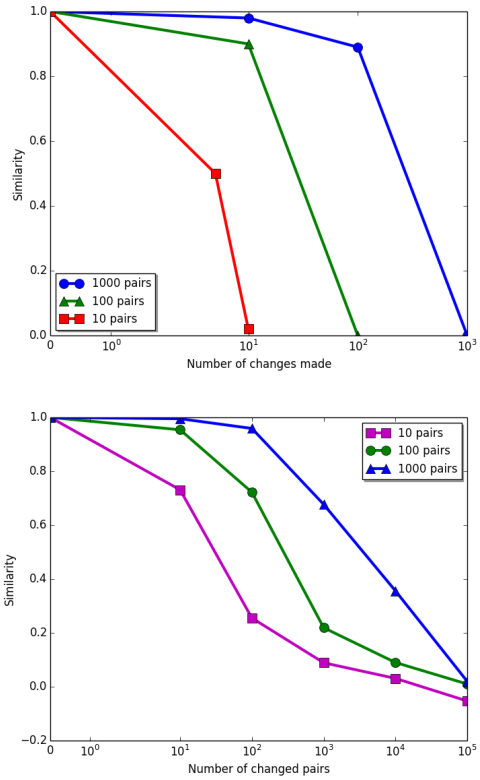


Figure 2: Top: Comparison of similarity values for changes in structure vs changes in bindings for a structure of 10,000 pairs. Bottom: Comparison of similarity as structures of different original length are extended.

When adding or removing new pairs, similarity (Equation 7) is affected as illustrated in Figure 3. Initially, both structures contained 1,000 pairs; the second structure was subsequently changed by an order of magnitude at a time. Thus the first structure gradually became a subset of the second. The graph illustrates that as the structure becomes extended with new pairs, similarity of the two structures begins to drop, de-

spite the fact that the structures share the initial 1,000 pairs. However, this degradation is very gradual, and similarities reach 0.0 only after 10^5 new pairs have been added. Furthermore, it can be seen that for larger structures such degradation is more gradual than for smaller ones (see Figure 2 bottom). For example, for 10-pair structures similarity is almost 0.0 after 10^3 new pairs have been added. This is still, however, fairly gradual, considering that a 10-pair structure had 1,000 pairs added to it before becoming dissimilar to the original. The drop in $\mathrm{sim}(X,Y)$ appears to be asymmetrical: removing pairs gives lower similarity than adding the same number of pairs. This is expected since removing pairs results in less shared information between the structures than adding pairs.

These findings are good and bad news at the same time. On the one hand it is good that feature value changes have a more drastic effect on $\mathrm{sim}(X,Y)$ than structure changes. On the other hand, the system will have to be able to *autonomously* find out whether two HRR vectors represent feature sets which differ structurally or in terms of feature values. Possibly this distinction can be learnt. But finding a solution that is invariant to HRR vector dimension size and feature set cardinality is likely not an easy task. Another problem is the commutative nature of $\mathrm{sim}(X,Y)$ which essentially does not allow to determine which is the more inclusive feature set.

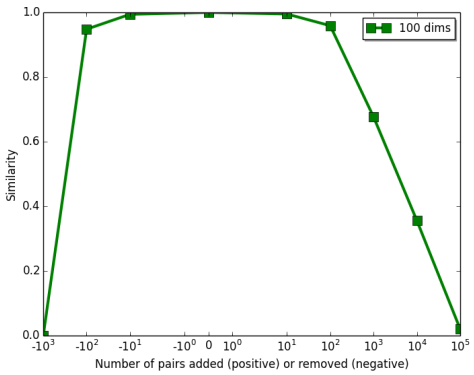


Figure 3: Gradual changes to a feature set of 1,000 feature-value pairs and their effects on similarity values.

Conclusions

This paper has speculated how a linguistically and computationally adequate formalism for language, namely Fluid Construction Grammar, could be represented in a Vector Symbolic Architecture, more specifically Holographic Reduced Representations, as a step towards a neural implementation. We proposed a number of steps and reported some preliminary implementation experiments with promising results. The main conclusion so far is that a number of fundamental issues remain to be solved to make the FCG→VGA mapping fully operational, particularly the issue of implementing a matching operation that uses a binding list and possibly extends it while matching takes place.

Acknowledgments

Research reported in this paper was funded by the Marie Curie ESSENCE ITN and carried out at the AI lab, Vrije Universiteit Brussel and the Institut de Biologia Evolutiva (UPF-CSIC), Barcelona, financed by the FET OPEN Insight Project and the Marie Curie Integration Grant EVOLAN. We are indebted to comments from Johan Loeckx and Emilia Garcia Casademont.

References

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. New York, NY: Oxford University Press.

Gayler, R. W., Levy, S. D., & Bod, R. (2010). Explanatory aspirations and the scandal of cognitive neuroscience. In *Proceedings of the first annual meeting of the bica society* (pp. 42–51). Amsterdam: IOS Press.

Goldberg, A. (1995). *Constructions: A construction grammar approach to argument structure*. Chicago: University of Chicago Press.

Hinton, G. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, *46*, 47–75.

Kanerva, P. (1997). Fully distributed representation. In *Proc. real world computing symposium* (pp. 358–365). Tsukuba-city, Japan: Real World Computing Partnership.

Neumann, J. (2001). *Holistic processing of hierarchical structures in connectionist networks*. Doctoral dissertation, School of Informatics,The University of Edinburgh UK.

Neumann, J. (2002). Learning the systematic transformation of holographic reduced representations. *Cognitive Systems Research*, *3*, 227–235.

Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, *19*, 113–126.

Ohlsson, S., & Langley, P. (1985). *Identifying solution paths in cognitive diagnosis* (Tech. Rep. No. CMU-RI-TR-85-2). Pittsburgh, PA: Carnegie Mellon University, The Robotics Institute.

Plate, T. A. (1994). *Distributed representations and nested compositional structure*. Doctoral dissertation, Graduate Department of Computer Science, University of Toronto,.

Shastri, L., & Ajjanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings. *Behavioral and Brain Sciences*, *16*, 417?494.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*, 159–216.

Steels, L. (Ed.). (2011). *Design patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, L. (Ed.). (2012). *Experiments in cultural language evolution*. Amsterdam: John Benjamins.

Steels, L., & Hild, M. (Eds.). (2012). *Language grounding in robots.* New York: Springer-Verlag.