To appear in Behavioral and Brain Sciences.
(Commentary on the target article *Processing Capacity Defined by Relational Complexity: Implications for Comparative, Developmental, and Cognitive Psychology*, by G. Halford, W. Wilson & S. Phillips.)

Chunks, bindings, STAR and Holographic Reduced Representations

Tony A. Plate

School of Mathematical and Computing Sciences, Victoria University, Wellington, New Zealand, 6001.
Electronic mail: Tony.Plate@vuw.ac.nz
Http://www.mcs.vuw.ac.nz/~tap

Abstract: Much of Halford et al.'s discussion of vector models for representing relations concerns the perceived inadequacies of alternative methods with respect to chunking, binding, systematicity, and resource requirements. However, vector-based models for storing relations are in their infancy, and the relative merits of different schemes are not so clearly in favor of their STAR scheme as Halford et al. portray.

Halford, Wilson & Phillips make three strong claims about the processing and representation of relational knowledge in the human brain and in neural network models. First, information processing capacity is more restricted by the complexity of relations that can be processed in parallel than by capacity for short-term storage. Second, in vector-based models, relations are better represented as predicate-argument bindings than role-filler bindings. Third, these bindings are best computed as tensor products. Halford et al.'s goal of producing a vector-based neural-network model of relational processing that incorporates capacity limits is thoroughly worthwhile and the model they propose has many interesting properties. Apart from a slight lack of clarity about the relationship between tensor-product representations for relations and such diverse phenomena as parsing center-embedded sentences and solving word puzzles, my main disagreements are with the second and third claims, which lie at increasingly deep levels of implementation detail. In Halford et al.'s story there is an unwavering path leading from the first through the second to the third claim, constrained by the inadequacies of alternative methods with respect to chunking, binding, systematicity, and resource requirements. However, vector-based models for storing relations are in their infancy, and the relative merits of different schemes are not so clearly in favor of their STAR scheme as Halford et al. portray. I will formulate my discussion as a defense of Holographic Reduced Representations (HRRs) (Plate 1994, 1995) against Halford et al.'s charges, although in many cases the defenses also apply to related methods such as Smolensky's (1990) role-filler tensor product representations and Kanerva's (1996) binary spattercodes.

Chunks. Chunking is one of the most important issues in any model that aspires to process hierarchical relational structure. Such a model must be able to form modest-sized chunks of information which can be presented to its processing machinery. In symbolic architectures cons cells or records can be viewed as chunks, and pointers serve to record relationships among chunks. In neural networks, chunking is less straightforward because there are no addresses to point to. However, neural network chunking techniques are potentially more powerful because a chunk can give some indication of its contents,[1] as Hinton (1990) suggested in his proposal for "reduced representations".

Chunking is intimately bound up with HRRs. Contrary to Halford et al.'s claim in Section 4.2, HRRs easily handle conceptual chunking and satisfy the three general principles of chunking listed in Section 3.4.1. Every vector in a HRR model is already a chunk and no further compression (as sought by Halford et al. in Section 4.2.4) is necessary. For example, the HRR L1=love+loveagt∗john + loveobj∗mary represents the relational instance loves(John, Mary) and is also a chunk that can be used as a filler in other relational instances such as knows(Sam,loves(John,Mary)) represented by L2=know+knowagt∗Sam+knowobj∗L1. An important property of the HRR scheme is that there is only one unitary representation for a relational instance (as opposed to the nonunitary representation, which is the collection of representations for the components), which serves both as a chunk and records relational structure. In contrast, in the STAR model there are two possible unitary representations for a relational instance: the tensor product version, and the chunked version[2]. This is not a fatal flaw, but it does seem to be a case of multiplying entities without necessity.

The claim in Section 4.2.4 that HRRs rely on randomly generated component vectors is not exactly correct: the component vectors merely have to have elements whose statistics are those of a Gaussian distribution, an important difference that allows similar objects to be represented by similar vectors. Chunked representations are easily made similar to their components by adding the components directly into the chunk, as in L1'= love+john+mary+loveagt∗john+loveobj∗mary.

Systematicity. Halford et al. point out that HRRs do not provide a complete solution to systematicity (Section 4.2.9). This is neither surprising nor a significant drawback – contextualized representations[3] were only intended to support an estimation of the degree of systematicity present. In general, detecting whether systematicity is possible is equivalent to solving the graph-isomorphism problem, for which no polynomial-time algorithm is known (Garey & Johnson 1979). Thus, it would be very surprising if any polynomial time algorithm operating on fixed width vectors (such the dot-product of HRRs, or some tensor operation in STAR) were able to detect systematicity in general. It is a bonus that the degree of systematicity between small structures represented in contextualized HRRs can be estimated using a vector dot-product. As regards the general case, even uncontextualized HRRs do represent the structural information required to make a decision about whether systematicity is present. What is missing is a computational process that can act on that information.

Given the computational complexity of detecting systematicity, this process must involve sequential computations in addition to parallel vector operations.

Role-filler and predicate-argument bindings. Halford et al. are right in claiming that role-filler binding schemes are susceptible to confusion if vectors representing relations are added in memory. However, this is a minor problem. There is no compelling reason why relation vectors must be added. This allows a very simple solution: keep relation vectors separate in an associative content-addressable memory. There is no shortage of associative-memory models which can store a few or many items, e.g., the memories described in Baum, Moody and Wilczek (1988) and in Kanerva (1988).

There is considerable room for discussion of the computational and psychological implications of different types of bindings. One consequence of predicate-argument bindings is that there is no explicit representation of roles, which makes it difficult to represent similarity between roles of different relations, as we might want to do for the relations (X buys Y from Z) and (U sells V to W). Another consequence is that roles must be known at storage and recall time, which may not always be the case. Furthermore, access based on one missing argument requires a different computation depending on which argument is missing, which seems cumbersome. In addition, recall based on more than one missing argument is difficult. Finally, dealing with relations having optional or symmetric roles also seems troublesome. The two schemes result in quite different patterns of similarity among relations: the similarity of two relations with role-filler bindings is the average of the similarities of their arguments, but with predicate-argument bindings it is the product of the similarities of their arguments. The two schemes are not mutually exclusive – it would be possible to use both together and thus combine their respective advantages.

A role-filler binding scheme could lead to resource bounds similar to the ones that Halford et al. identify. If looking up a relation in content-addressable memory is what is involved in making a single decision, then the number of independent pieces of information that could influence a decision would be the same as the size of chunks, i.e., the number of role-filler bindings which can be reliably superimposed in the same vector. This explanation leads to a soft bound on processing capacity rather than the hard bound implied by predicate-argument binding. This capacity would also depend on the similarity of the particular pieces of information involved – one can reliably superimpose more dissimilar items than similar ones.

The issues regarding the relative merits of role-filler versus predicate-arguments bindings are largely orthogonal to those regarding the vector operations for implementing the bindings: either type of binding can be implemented with different vector operations (Halford et al. discuss only tensor products and convolution but there are other possibilities, see Plate (1994) and Kanerva (1996)).

Resource requirements. Halford et al. cite the requirement of a clean-up memory as a major computational cost unique to HRRs. However, the space complexity of the clean-up memory is linear in the number of entities and relations stored, which is hardly unreasonable. Suitable fast auto-associative memories are easily implemented in neural architectures. The simplest scheme uses a “grandmother cell” to store each item (Baum, Moody &Wilczek, 1988): one item is encoding in the incoming and outgoing weights of a cell and retrieval involves a winner-take-all competition over the grandmother cells. In any case, a realistic implementation of the STAR model will almost certainly need to be equipped with a clean-up memory, to clean up the results of processing nonorthogonal vectors for items and chunks. Realistic implementations demand nonorthogonal vectors for two reasons. The first is based on simple capacity considerations: denser, nonorthogonal representations must be used if thousands of different possible items (say of the order of number of nouns in the English language) are to be represented. It is unreasonable to suppose that a rank-4 tensor might be formed over vectors 10,000 elements wide, for it would require 10^{16} bits of storage. The second reason is that representing similarity among items (e.g., that John is more similar to Bob than Flying-Saucer) requires non-orthogonal vectors and is essential for reasoning and generalization.

Halford et al.’s “worst case” situation for a clean-up memory where the majority of instances in a relation are stored (Section 5.2.3) will not occur when there are even a moderate number of items. Consider populating the space of rank 4 tensors where there are 1,000 possibilities for the predicate name and each of the three arguments. Even after storing one instance per second for 100 years, only 3.15% of the relational space would be filled (and this is a gross overestimate – Simon (1974) quotes an estimate of 5 to 10 seconds to store a chunk in long-term memory). In fact, for storing sparse relations a HRR scheme with a clean-up memory is far more economical than a STAR scheme because the HRR scheme only requires resources proportional to the actual number of relational instances stored, whereas the STAR scheme requires resources proportional to the total number of possible relational instances.

NOTES:

1. A pointer usually gives no indication of what it points to; it must be dereferenced to find out anything about the contents. A simple example of the idea of having a pointer say something about its contents can be seen in some LISP architectures where a pointer indicates the type of the object, integer, cons-cell, etc., that it points to.

2. Halford et al.'s two proposals for implementing chunking appear to be representationally inadequate. The first proposal, in which items are chunked by concatenating vectors (illustrated in Fig 1E), can result in confused argument bindings: the result of superimposing the tensors for the two unary relations p(a/b) and p(d/e) (i.e., $p \times c(a,b) + p \times c(d,e)$, where c(x,y) is a function which concatenates the vectors x and y end-to-end) is indistinguishable from the superposition of the vectors for the relations p(a/e) and p(d/b) ($p \times c(a,e) + p \times c(d,e)$). The second proposal, in which items are chunked by convolving vectors loses the order of components within the chunk: p(a/b) has the same tensor product as p(b/a). This is because convolution is commutative: $a*b = b*a$. Both these objections to Halford et al.’s proposals are minor because there are other

operations which have the required properties, e.g., the noncommutative variants of convolution discussed in Plate (1995).

3. Halford et al.'s describe contextualization incorrectly: what contextual features are added depends on the situation, so in one situation features for smiling might be added to the representation for Jane, but in another features for frowning might be added. This does not change the general representation of Jane, only the representation in a particular situation. One of the motivations for the particular implementation of contextualization was that over the long term, if Jane was consistently frowning, the frowning feature might become part of the representation of Jane. However, this would not prevent the smiling feature being added in some particular situation.

References:

Baum, E. B., J. Moody, and F. Wilczek (1988). Internal representations for associative memory. Biological Cybernetics, 59 , 217--228.

Garey, M. and D. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: Freeman.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. Artificial Intelligence. 46 (1-2), 47--76.

Kanerva, P. (1988). Sparse Distributed Memory. Cambridge, MA: MIT Press.

Kanerva, P. (1996). Binary spatter-coding of ordered k-tuples. In C. von der Malsburg, W. von Seelen, J. Vorbruggen, and B. Sendhoff (Eds.), Artificial Neural Networks--ICANN Proceedings , Volume 1112 of Lecture Notes in Computer Science, Berlin, pp.869--873. Springer.

Plate, T. A. (1994). Distributed Representations and Nested Compositional Structure. Ph.D. thesis, Department of Computer Science, University of Toronto. Available at http://mcs.vuw.ac.nz/~tap.

Plate, T. A. (1995). Holographic reduced representations. IEEE Transactions on Neural Networks 6 (3), 623--641.

Simon, H. (1974). How big is a chunk? Science 183 , 482--488.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artificial Intelligence 46 (1-2), 159--216.