Multiplicative Binding, Representation Operators & Analogy

Ross W. Gayler (r.gayler@psych.unimelb.edu.au)

Analogical inference depends on systematic substitution of the components of compositional structures. Simple systematic substitution has been achieved in a number of connectionist systems that support binding (the ability to create connectionist representations of the combination of component representations). These systems have used various implementations of two generic composition operators: **bind()** and **bundle()**. This paper introduces a novel implementation of the **bind()** operator that is simple, can be efficiently implemented, and highlights the relationship between retrieval queries and analogical mapping.

A frame of role/filler bindings can easily be represented using **bind()** and **bundle()**. However, typical binding systems are unable to adequately represent multiple frames and arbitrary nested compositional structures. A novel family of representational operators (called **braid()**) is introduced to address these problems. Other binding systems make the strong assumption that the roles and fillers are disjoint in order to avoid ambiguities inherent in their representational idioms. The **braid()** operator can be used to avoid this assumption.

The new representational idiom suggests how the cognitive processes of bottom-up and top-down object recognition might be implemented. These processes depend on analogical mapping to integrate disjoint representations and drive perceptual search.

Analogical Inference by Systematic Substitution

Analogical inference depends on systematic substitution of the components of compositional structures (Gentner, 1983; Halford et al., 1994; Holyoak & Thagard, 1989).

Base	Target	Mappings	Notes
(x (y z z) x)	(a (b c c) a)	x→a, y→b, z→c	
(x (y z z) x)	(a (b c c) ?1)	x→a, y→b, z→c, ?1→a	
(x (y z z) x)	(a (b c ?1) a)	x→a, y→b, z→c, ?1→c	
(x (y z z) x)	(a (?1 c c) a)	x→a, z→c	Unable to infer the unknown
(x (y z z) x)	(a (b c ?1) ?2)	x→a, y→b, z→c, ?1→c, ?2→a	
(x (y z z) x)	(a (b c ?1) ?1)		Structures inconsistent

Table 1: Examples of systematic substitution

Simple systematic substitution has been achieved in a number of connectionist systems that support binding (Halford et al., 1994; Kanerva, 1997; Plate, 1994; Smolensky, 1990). Binding is a generic connectionist operator for combining pattern vectors. Substitution of **x** for **a** in the simple structure **s** by binding can be carried out by: **substitute(x, a, s) = bind(bind(x, $\mathbf{a^{-1}}$), s)**. Substituting **fred** for **joe** in **bind(loves, joe, mary)** we get **substitute(fred, joe, bind(loves, joe, mary)) = bind(bind(fred, $\mathbf{joe^{-1}}$), bind(loves, joe, mary)) = bind(loves, fred, bind($\mathbf{joe^{-1}}$, joe), mary) = bind(loves, fred, mary)**.

Representation Operators

These systems (Kanerva, 1997; Plate, 1994; Smolensky, 1990) have used two types of representation operators (generically renamed here as **bind()** and **bundle()**) for joining and aggregating patterns. This paper introduces a novel implementation of the **bind()** operator.

	Smolensky	Plate	Kanerva	Gayler
bind()	tensor product	circular convolution	elementwise XOR	elementwise product
bundle()	elementwise sum	elementwise sum	thresholded elementwise sum	elementwise sum

Table 2: Implementations of representation operators

The **bundle()** operators are essentially identical: normalised elementwise sums of the vectors. (Kanerva's normalisation is achieved by thresholding.). There are interesting relations between the **bind()** operators. Smolensky's tensor product is the outer product of the vectors to be bound. Binding two vectors of **n** units gives an outer product of $\mathbf{n}^2$ units. Plate's circular convolution is a set of **n** sums over the elements of the outer product. The binding is the same length as each component. The **bind()** operators of Kanerva and Gayler also have this property.

Kanerva restricts his units to have values of 0 or 1. His **bind()** operator is equivalent to Plate's complex-valued Holographic Reduced Representation when the components are restricted to {-1,+1}. Gayler limits the activation of his units to the real range [-1,+1]. The elementwise product of the extreme values is isomorphic to the exclusive-or of {0,1}. Thus Gayler's multiplicative binding can be seen as a continuous generalisation of Kanerva's **bind()** and shares the same relationship to Plate's **bind()**.

Gayler's multiplicative binding is equivalent to taking the diagonal of the outer product and hence is related to Smolensky's **bind()** . Plate conceived of his **bind()** as a way of compressing the outer product and hypothesised that other compressed outer products would also be useful binding operators. Multiplicative binding can be seen as a member of this family in which off-diagonal elements of the outer product receive zero weight. Finally, Plate's circular convolution binding may be rewritten in terms of multiplicative binding as $\mathbf{bind_{Plate}(a,b) = bundle(bind(braid_{x1}(a),braid_{y1}(b)), \ldots ,bind(braid_{xn}(a),braid_{yn}(b)),)}$ where **braid()** is a family of permutation operators to be discussed later.

Multiplicative binding is interesting because: it allows compositional structures to be the same size as their components; can be efficiently implemented because information flows are local; and each pattern is its own inverse (if patterns are restricted to $[-1,+1]^n$). This last property makes the implementation of systematic substitution particularly easy. For example, the following simple systematic substitution architecture relies on the self-inverse property.

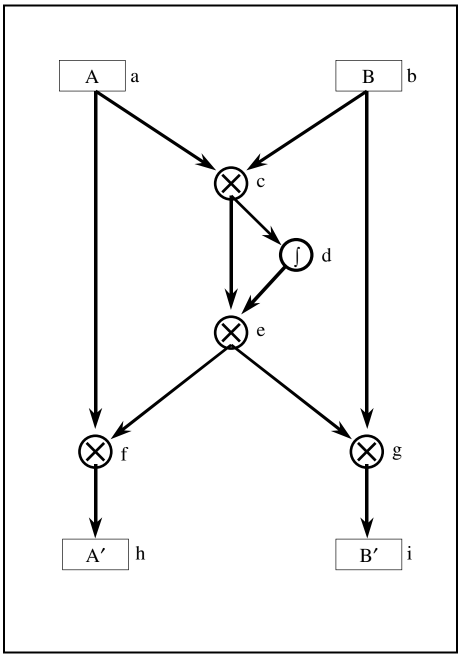

Figure 1: A substitution architecture

a	b	d	h	i
A	B	{[A,B]}	A	B
B	A	{[A,B]}	B	A
A	B	{[A,B],[C,D],…}	{A,noise}	{B,noise}
A	1	{[A,B]}	noise	B
A	X	{[A,B]}	noise	B

Table 3: Examples of retrieval by substitution

The last case is particularly interesting. The pattern **X** acts as a variable marking a query because it is not present in the memory trace (*d*: the sum of previous bindings from *c*) bound to any other pattern, whereas the pattern **A** does appear in the trace bound to another pattern (**B**). During retrieval the pattern **bind(X,B)** is constructed at *e*. Binding with this pattern (at *f* and *g*) substitutes **X** for **B**. Thus, this network performs a very simple systematic substitution. It recognises that the input tuple **[A,X]** is novel and maps **X** to **B** based on the tuple **[A,B]** .

Bind() and Bundle() are an Inadequate Basis for Representation

The **bind()** and **bundle()** operators may be viewed as representational operators because they are used for the construction of compositional representations. The representation of hierarchical structure is of particular interest to analogy because this allows arbitrarily complex structures and the representation of higher-order relations between propositions.

Frame representations have typically been used to represent hierarchical structure as a bundle of role/filler bindings. This representational idiom has limitations that prevent the representation of arbitrary nested structures.

- There is nothing in a role/filler binding that marks it as belonging to a specific bundle. When there are multiple frames represented as bundles over the same set of units there is no way to tell which role/filler pairs belong to which frames.
- A nested structure (constructed by using one frame as the filler of a role in another frame) is equivalent to a bundle of $\text{role}_{\text{outer}}/\text{role}_{\text{inner}}$/filler bindings. With commutative binding operators (Plate, Kanerva, Gayler) the bound components are unordered and it is impossible to tell which of the roles is nested in the other.
- When each pattern is its own inverse (Kanerva, Gayler) the same role cannot appear more than once in a sequence of nested roles because they will cancel out (from the self-inverse property).

Representation of Hierarchical Structure with Braid()

This paper introduces the **braid()** family of unary operators. Each specific **braid()** operator is a permutation of the vector elements. There are as many braid operators as there are permutations of the pattern vector. The major properties of braids are that they are noncommutative, always have an inverse, and generally **braid(x)** is not similar to **x**. The last property is very useful because it makes **braid()** a type of quotation operator that can be used to preserve a pattern from cancellation by self-inverse binding. Thus $\mathbf{braid_i()}$ can be thought of as a type of labeled bracketing (the label being the identity of the specific permutation).

Two particularly useful braid operators are **up()** and its inverse, **down(),** which are used to move up and down levels of a hierarchical structure. They may be implemented by a one unit shift of the vector pattern relative to the units. These can be used in the representation of nested structure by expanding each role/filler pair to be an object/role/filler triple (where the object pattern is **up(f)** and **f** is the frame represented as a bundle of role/filler pairs). This is equivalent to **bind(up(f),f)** and is an implementation of Hinton's (1990) reduced representations. The effect of this operation is to bind each role/filler pair to the image under **up()** of every other role/filler pair in the same frame. This allows role/filler pairs from different frames to be bundled together without losing their frame identity and allows every role/filler pair to act as a cue for the retrieval of every other role/filler pair in the entire frame.

Braid operators have other potential uses. It may be useful to extend the object/role/filler triples to include the total context (say, the bundle of all the currently represented objects). This could be accomplished by extending the triples to be context/object/role/filler 4-tuples. It is not yet clear whether this can be accomplished by using the **up()** operator with the context. We may need a new pair of **braid()** operators (say, **left()** and **right()**) to represent the bundles of all current objects as a context.

Carrying this idea further, it is not clear that the domain of roles will always be disjoint from the domain of fillers. Most researchers using role/filler representations have assumed the concept of role as self-evident, simply invented role decompositions to suit their purpose, and assumed that roles and fillers are disjoint. However, it is not at all clear what constraints might exist on roles and how a system might learn new roles (Blank & Gasser, 1992; Cohen, 1998). One possibility is that roles are action and attention plans and a role/filler binding may be read as the prediction: "If I perform the plan **role** then I will perceive the value **filler**". It is possible that the filler may also be an action plan (that is, the perceived outcome of the role is a potential for further action). If the role and filler domains are not disjoint the commutativity of the **bind()** operator (Plate, Kanerva, Gayler) will introduce ambiguity as to which pattern is the role and which the filler. This (and other similar ambiguities) could be overcome by using a separate braid operator to encode each facet of the bound tuple. A 4-tuple might be encoded as $\mathbf{bind(braid_{context}(c),braid_{object}(o),braid_{role}(r),braid_{filler}(f))}$.

Bottom-Up and Top-Down Object Recognition

The representation of frames with object/role/filler triples suggests how the processes of bottom-up and top-down object recognition might work. Consider a situation in which the system has just been presented a new instance of a previously encountered object type. The perceptual apparatus is scanning the environment and encoding the results as object/role/filler tuples. Each role is a motor plan that resulted in direction of the perceptual apparatus and the filler is the perceptual input that is received as a consequence. Initially each role/filler pair is treated as a separate, unique object.

Because the role/filler bindings have arisen from the same external object there will be structural relations between the roles and fillers of the role/filler pairs. The analogical retrieval process (cued by the literal roles and fillers) attempts to find a systematic substitution for the object and role facets that will make the transformed input tuples maximally coherent with tuples already stored in memory. That is, the system will attempt to substitute a single object pattern for all the separate object patterns in the tuples. That single pattern will be the braided image of the bundle of all the role/filler bindings integrated by that object. Thus, analogical mapping by systematic substitution can be seen as the thread that is used to sew independent percepts into a coherent object representation by bottom-up recognition.

Now consider a top-down perceptual process. The construction of the new object representation has activated the representations of previously encountered objects (those it has mapped to). Some of these objects may contain roll/filler pairs with no current analogue in the perceived object (recall that the reduced representation of a frame consists of a bundle of all the role/filler pairs in that frame). All the role/filler pairs retrieved from long term memory are mapped back to the current representation. Therefore, the extra role/filler pairs will appear as predictions of properties of the current object.

References

Blank, D., & Gasser, M. (1992). Grounding via scanning: cooking up roles from scratch. *In Proceedings of the 1992 Midwest Artificial Intelligence and Cognitive Science Society Conference*.

Cohen, P. R. (1998). *Growing ontologies* (Computer Science Technical Report 98-20). Amherst: University of Massachusetts, Computer Science Department.

Gentner, D. (1983). Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, *7*, 155-170.

Halford, G. S., Wilson, W. H., Guo, J., Gayler, R. W., Wiles, J., & Stewart, J. E. M. (1994). Connectionist Implications for Processing Capacity Limitations in Analogies. In K. J. Holyoak & J. A. Barnden (Eds.), *Advances in connectionist and neural computation theory, Vol. 2: Analogical connections*. Norwood, NJ: Ablex.

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

Holyoak, K. J. & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, *13*, 295-355.

Kanerva, P. (1997). Fully distributed representation. In *Proceedings RWC Symposium 1997 (Tokyo, Japan)*, 358-365.

Plate, T. A. (1994). *Distributed representations and nested compositional structure*. Ph.D. thesis, Department of Computer Science, University of Toronto.

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