Unbounded Dependency: Tying strings to rings

Jon M. SLACK
e-mail: slack@irst.uucp
Istituto per la Ricerca Scientifica e Tecnologica (I.R.S.T.)
38050 Povo (TN)
ITALY

Abstract: This paper outlines a framework for connectionist representation based on the composition of connectionist states under vector space operators. The framework is used to specify a level of connectionist structure defined in terms of addressable superposition space hierarchies. Direct and relative address systems can be defined for such structures which use the functional components of linguistic structures as labels. Unbounded dependency phenomena are shown to be related to the different properties of these labelling structures.

Introduction

One of the major problems facing connectionist approaches to NLP is how best to accommodate the role of structure (Slack, 1984). Fodor and Pylyshyn (1988) have argued that connectionist representations lack combinatorial syntactic and semantic structure. Furthermore, they claim that the processes that operate on connectionist representational states function without regard to the inherent structure of the encoded data. The thrust of their criticism is that mental functions, such as NLP, are appropriately described in terms of the manipulation of combinatorial structures, such as formal languages, and that, at best, connectionism provides an implementation paradigm for mapping NLP structures and proceses onto their underlying neural substrates.

If Fodor and Pylyshyn's arguments are correct then there can be no connectionist principles which influence the nature of theories developed at the level of symbolic representation. However, the present paper shows that it is possible to define a level of connectionist structure, and moreover, that this level is involved in the explanation of certain linguistic phenomena, such as unbounded dependency.

Connectionist Structure

A theory of connectionist representation must show how combinatorial structure can be preserved in passing from the symbolic level of explanation to the connectionist level. One way of achieving this is by positing an intermediate level of description, called the level of *Connectionist Structure* (CS), at which combinatorial structure is preserved but in terms of connectionist combinatory operators rather than the operators of formal languages.

A framework for connectionist representation is illustrated in figure 1. In a connectionist system the formal medium for encoding representations is a numerical vector corresponding to a point in a Vector Space, *V*. Formally, all connectionist representations can be expressed as vectors of length k, defined over some numerical range.

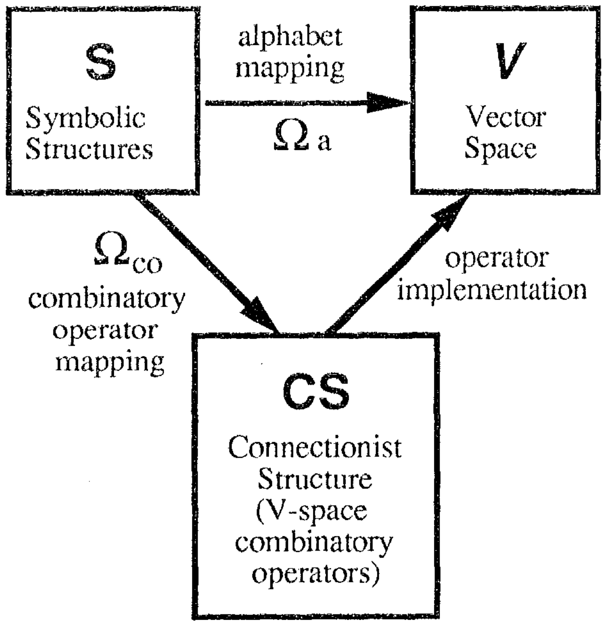
S
Symbolic Structures
alphabet mapping
Ωa
V
Vector Space
Ωco
combinatory operator mapping
operator implementation
CS
Connectionist Structure (V-space combinatory operators)

Figure 1

Symbolic structures comprise an alphabet of atomic symbols, and a set of symbolic combinatory operators; the symbolic alphabet is mapped into *V*-space under the *alphabet mapping*, Ω_a. This mapping might have one or more desirable properties, such as *faithfulness*, *orthogonality*, etc..

The other major component of the framework, Ω_{co}, maps symbolic combinatory operators onto corresponding vector combinatory operators. The CS level is defined in terms of structured vectors which are generated through applying the *V*-space combinatory operators to the set of vectors in the codomain of the alphabet mapping. The main reason for differentiating the CS level of representation is that only certain combinatory operators are available at this level, the most useful ones being association and superposition, and this restricts the range of symbolic structures that can be encoded directly under a connectionist representation. Essentially, the CS level preserves the connectivity properties of the symbolic structures.

Within this framework the CS level can be defined formally as a *semiring*, as follows

Definition. The CS level comprises the quintuple $(\mathbf{V}, +, **, \mathbf{0}, \partial)$[1] where

1. $(\mathbf{V}, +, \mathbf{0})$ is a commutative monoid defining the *superposition* operator;
2. $(\mathbf{V}, **, \partial)$ is a monoid defining the *association* operator;
3. $**$ distributes over $+$;

The two identity elements correspond to identity vectors, where ∂ is defined for zero-centred vectors (Slack, 1984). The vector combining operations of association and superposition are used to build connectivity configurations in memory. Moreover, using an appropriate threshold function, the superposition operator can simulate a rudimentary form of unification (Slack, 1986). The most general class of structures that can be defined at the CS level using the two combinatory operators are *addressable superposition space hierarchies* (referred to as ASSHs).

CS Address Systems

The Ω_{co} mapping can be used to define a correspondence between the symbolic operations of union and concatenation, and the CS operations of superposition and association, respectively. This allows the following homomorphism to be defined f: S -> CS, mapping the semiring S into the CS semiring, where

$$f(xUy) = f(x) + f(y) \text{ and } f(x.y) = f(x) ** f(y),$$

and the semiring S comprises the quintuple $(L_X, U, ., 0, \{\partial\})$ where L_X is the finite set of strings defined over the symbolic alphabet X, and U and . denote the union and concatenation operators, respectively, with their corresponding identities. The existence of the homomorphism allows symbolic structures to address CS representations. However, the restriction on this mapping is that CS level address systems cannot capture the full expressive power of regular languages. This is because no CS level operator can be defined with the same closure properties as the Kleene star operator at the symbolic level. The implications of this constraint become apparent in describing how symbolic structures can function as structural addresses for the CS level.

The symbolic structures that function as addresses to the CS level can be represented using directed, acyclic graphs (DAGs), and are referred to as *address structure DAGs* (AS-DAGs). AS-DAGs codify the way in which symbolic labels address, or map onto, the nodes and edges of ASSHs. In general, two possible types of address system can be defined, direct addressing and relative addressing. A system of direct addressing involves specifying unique ASSH addresses explicitly. That is, a symbolic label functioning as an address, directly accesses a unique ASSH node. The alternative addressing scheme involves specifying nodes in the configuration in terms of their connectivity paths from some pre-defined origin node. This form of relative addressing requires, (a) a pre-specified origin, or root node, and (b) a labelling system for the connections within the configuration.

The set of symbolic labels that serve as addresses in AS-DAGs can be partitioned into two classes, local and global labels, which are differentiated in terms of their function within an address structure. Global labels map onto the nodes of AS-DAGs providing direct addresses for the superposition spaces in ASSH configurations. Local labels, on the other hand, map onto AS-DAG edges and specify the relative addresses of ASSH spaces. That is, they specify the locations of superposition spaces relative to the addresses of their dominating nodes. This relationship is illustrated in figure 2 showing how AS-DAGs map onto ASSHs.

[1] Characters in bold denote elements of the vector space.

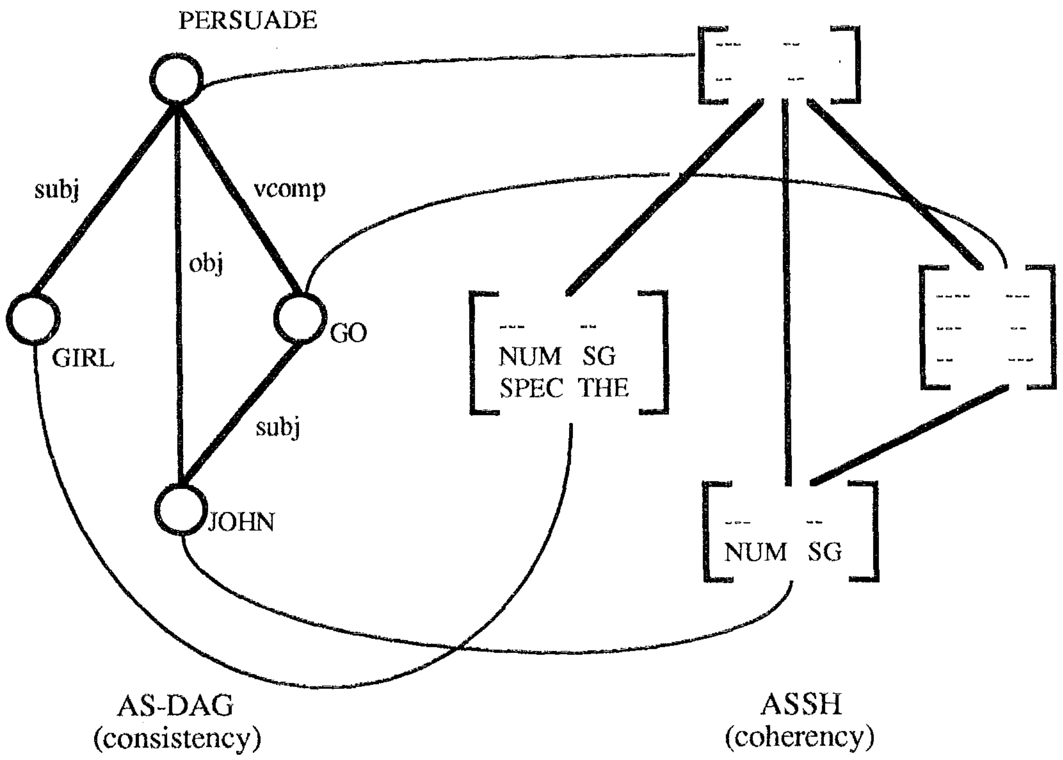

Figure 2

The figure shows the CS level encoding of the LFG representation of the sentence *The girl persuaded John to go* (see Slack, 1990). The superposition space labelled 'JOHN' in the AS-DAG can also be located using the compound address 'PERSUADE.obj'.[2] The obvious question that arises is what possible rationale is there for this system of double addressing? With a system of direct addressing for ASSHs, the relative addressing scheme would appear redundant.

At the symbolic level, local labels specify local structure, that is, how a node relates to its immediate descendents. In situations in which the local structure is uniform and finite, the *V*-space encodings of local labels can be fixed under Ω_a, mapping each label onto a constant vectorial encoding. This allows AS-DAGs to be viewed as configurations of local structures which can be located in *V*-space by fixing the vectorial encodings of their root-nodes. This means that global labels must be assigned dynamically under Ω_a. Putting the emphasis on local structure seems to make the direct addressing system redundant, but there are good reasons for needing direct access to superposition spaces.

Defining arbitrary structural addresses as strings of local labels descending from a root-node can only be achieved under symbolic level control as the representational machinary necessary for interpreting concatenated label strings does not exist at the CS level. A string of local labels can only be encoded as a single AS-DAG edge corresponding to an uninterpreted label string. That is, the CS level comprises a set of superposition spaces which support structured access, and only a single ASSH edge can link two such spaces. This means that CS level access through relative addressing is limited to addresses comprising a single edge leading from an origin node. Building up addresses in this way necessitates an AS-DAG node labelling scheme such that the origin node can be defined iteratively. In other words, because the *V*-space encoding of local structure is fixed under Ω_a, relative addressing can only be specified on a local basis, with the only form of global addressing involving direct access to ASSH nodes, or superposition spaces. This limit on symbolic level access to connectionist representational states is an important source of locality constraints in encoding linguistic structures at the CS level (Slack, 1990)[3].

[2] In the figure, global labels are shown in uppercase and local labels in lowercase.

[3] The representational framework has been implemented on a simple associative memory

Unbounded Dependency: Connectivity

One linguistic phenomenon which, more than any other, focuses on the problem of addressing structural configurations is that of unbounded dependency (UBD). Typically, in sentences like *The boy who John gave the book to _ last week was Bill*, the phrase *The boy* is taken as the 'filler' for the missing argument, or 'gap', of the *gave* predicate, as indicated by the underline. At the level of constituent structure there are no constraints on the number of lexical items that can intervene between a filler and its corresponding gap. Such "unbounded dependencies" are typical of a class of linguistic phenomena in which the structural address of an element is determined by information which is only accessible over some arbitrary distance in the structure. To build the appropriate memory configuration, it is necessary to determine the address of the gap to which a filler belongs. However, because gaps and fillers can be separated by arbitrary distance in the input string, it is not possible to specify the set of potential predicate-argument relations that the filler can be involved in, and so a direct address cannot be identified. Instead, it is necessary to generate a relative address through the construction of a chain of global and local labels.

Within the framework of Government-Binding theory, these phenomena have been explained through identifying conditions defined on constituent trees that account for the distribution of gaps and fillers both within and across natural languages. One such principle is based on the structural geometry of constituent trees, in particular, their connectivity properties (Kayne, 1983). Kaplan and Zaenen (1988) have taken a different approach to UBD restrictions arguing that they are best explained at the level of predicate-argument relations, rather than in terms of constituent structure. Working within the LFG framework, their formal system is based on the idea of functional uncertainty expressions. For example, for topic-alization sentences these expessions have the general form (^ TOPIC) = (^ GF* GF) involving the Kleene closure operator, where GF stands for the set of primitive grammatical functions. These equations express an uncertain binding between the TOPIC function and some argument of a distant predicate. The uncertainty relates to the identification of the appropriate predicate. To resolve the uncertainty it is necessary to expand this expression and match it against the functional paths of missing arguments. Different computational strategies can be used to optimise the resolution process (Kaplan & Maxwell, 1988). What is common to both approaches is that they define a system for specifying the structural address of a gap relative to its corresponding filler.

The notion of functional uncertainty, in common with other linguistic feature structures, uses an address system based on regular languages (Kasper & Rounds, 1986). It is impossible, however, to use such addresses to access the CS level directly as the Kleene closure operator cannot be interpreted at this level. In their present form, functional uncertainty algorithms require some kind of 'symbolic level' memory in which to expand uncertainty expressions.

An alternative account of UBD phenomena, also based on predicate-argument relations, can be founded on the notion of symbolic labels functioning as local and global addresses to the CS level. The problem of UBD can be decomposed into two sub-problems, one relating to local structure, the other relating to global indeterminacy. Consider the sentence fragment *The girl John saw Bill talking to _ .....*, where the problem is to specify the structural address of the topicalised NP, *The girl*, as the missing argument of some COMP function[4]. At the level of local structure, the structural address of the filler is minimally uncertain in that it can fulfil

---

system based on a functional partition of *V*-space into an *Address Space* and a *Content Space* (Kanerva, 1988). An important feature of this architecture is that by encoding the elements of both spaces as k-bit vectors, they are potentially interchangeable. This allows elements retrieved from content space to function as addresses to other memory locations, and vice versa. Thus, the memory consists of a set of superposition spaces, where each space has a label (or address), and where labels can be encoded as elements of other spaces resulting in a hierarchical structure. In a hybrid architecture based on a CS level memory, symbolic structures are encoded through symbolic labels addressing elements of the homomorphic ASSH configurations in memory. In other words, symbolic level activity is implemented as the manipulation of address space labels (see Slack, 1990).

[4] As the present discussion focuses on predicate-argument relations, we will continue to make use of LFG notation and constructs, such as grammatical functions.

only a small set of local roles, for the present case the OBJ function. However, the structural address of the appropriate local structure is maximally uncertain, as the filler item carries no information to constrain it.

Before considering solutions to these two sub-problems, it is necessary to clarify how the informational components of linguistic structures such as *f*-structures function as addresses to the CS level. Elements of the set GF can function as both local and global addresses to memory configurations. Each GF element defines a component of local structure and as such can function as a local label in the relative address chain for an AS-DAG node. In addition, GF labels can be associated with fixed memory locations, thereby functioning as global addresses. For example, the symbol COMP can be used to label an AS-DAG edge forming a constituent of a relative address, and at the same time provide direct access to a fixed location, that is, label an AS-DAG node. These two addressing functions can be distinguished by using the labels COMP_p and COMP_g to denote the local and global addresses, respectively.

In encoding *f*-structures at the CS level, each sub-structure maps onto a separate superpositon space (Slack, 1986). This form of direct addressing requires a set of global symbolic labels that uniquely identify each sub-structure. The predicate names of *f*-structures provide such a labeling system. In this case, each predicate name constitutes a unique origin for defining relative local addresses. Hence, local labels like COMP_p specify locations relative to a predicate 'p', that is, their immediate dominating node in the AS-DAG.

These labeling systems can be used to solve the two UBD sub-problems. On encountering a filler item in the input string, the analyser must allocate some structural location in memory at which to store the information carried by the item[5]. Part of that information specifies the filler's local address. For example, the information carried by the phrase *the girl* might include an encoding of the functional sub-structure [OBJ ** [pred 'girl' + spec the + num sg]][6]. At some later point in processing, this information will be superposed, or unified, with stored information about the structure of some local tree. For example, the predicate *talk* may encode through subcategorisation the local structure *talk(subj, obj, comp)*. If the OBJ function remains unspecified, the filler information will unify at this location in memory, enabling its structural address to be specified relative to the address of the predicate *talk*. The syntactic analyser can solve the memory allocation problem by generating the label TOPIC, enabling the encoding: TOPIC_g -> [OBJ**[NP features]][7]to be created. This encoding solves the local dependency problem.

To solve the problem of global indeterminacy, the analyser must also build an encoding like

$$\mathrm{COMP}_p \text{ -> } \mathrm{TOPIC}_g$$

the effect of which is to move the topicalised information through connected COMP locations. In other words, it corresponds to the control equation TOPIC = COMP* at the symbolic level[8].

The principle underlying this latter encoding is that COMP_p labels a specific location in memory determined relative to the location with the global address 'p'. This means that the local label is automatically reassigned as each new local structure unfolds. The operation of reassignment involves two concurrent actions: 1) *Direct labelling* - the structural location labelled by COMP_p is re-labelled using the new predicate name as a global label; 2) *Build connection* - a new location is connected into the structure with the label COMP_p where 'p' is bound to the new predicate name. The effect of the last action is to *pass* the topicalised information onto the next *connected* level of local structure. Obviously, if the first action occurs without the second, the filler label will not be passed as the COMP_p -> TOPIC_g encoding will become undefined. Once this happens the COMP_p address is no longer retrievable for further processing. However, the second action can only occur if the building of a COMP_p edge is *licensed* by the information carried by the predicate 'p'. For example, consider the partial *f*-structure shown in figure 3 and the corresponding AS-DAG configuration. A topicalised NP originating at the top-level

[5] Problems of structural ambiguity are not being considered at this point.

[6] This encoding utilises the fact that memory addresses can also be encoded as memory contents, and vice versa.

[7] This notation specifies address-content associations;
AS-DAG address -> superposition space contents.

[8] As stated previously, the operator * cannot be mapped directly to the CS level.

Partial *f*-structure

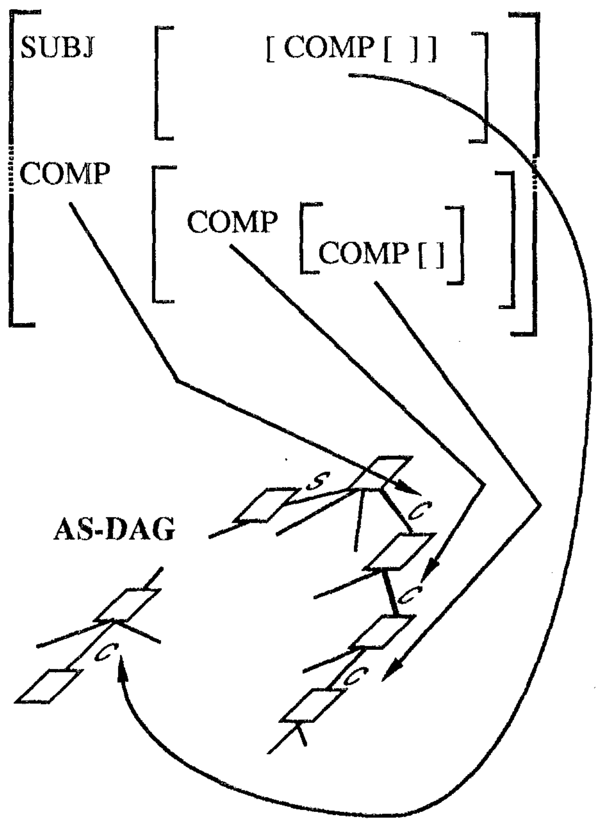

Figure 3

node can be passed down the COMP$_p$ reassignment chain descending from the same node, but it cannot be passed to the COMP embedded in the SUBJ *f*-structure. The COMP$_p$ -> TOPIC$_g$ encoding is undefined at the location addressed by the SUBJ$_p$ label, because the COMP function cannot be licensed by the SUBJ predicate.

In brief, functional uncertainty expressions such as (^ TOPIC) = (^ COMP* OBJ) cannot be mapped directly to the CS level as structural addresses. Instead, the uncertainty is captured by the CS encodings COMP$_p$ -> TOPIC$_g$ and TOPIC$_g$ -> OBJ**[NP features]. As the connectivity structure unfolds in memory, the action of reassigning COMP$_p$ places restrictions on the set of structural addresses to which the topicalised information can be passed. Using predicate-argument structures as address systems for the CS level leads to the conclusion that connectivity, defined at this level of linguistic structure, determines the distribution of fillers and gaps within a language.

References

Fodor, J.A., and Pylyshyn. Z.W. (1988) Connectionism and cognitive architecture: A critical analysis. *Cognition*, **28**, 3-71.

Kanerva, P. (1988) *Sparse distributed memory*. MIT Press, Cambridge, Mass..

Kaplan, R.M. & Maxwell, J.T. (1988) An algorithm for functional uncertainty. *COLING 88*, Budapest.

Kaplan, R.M. and Zaenen, A. (1988) Long-distance dependencies, constituent structure, and functional uncertainty. In M. Baltin and A. Kroch (eds.), *Alternative Conceptions of Phrase Structure*. Chicago: Chicago University Press.

Kasper, R.T. and Rounds, W.C. (1986) A logical semantics for feature structures. In the *Proceedings of the 24th meeting of the Association of Computational Linguistics*, Colombia University, New York.

Kayne, R.S. (1983) Connectedness. *Linguistic Inquiry*, **14**, 223-249.

Slack, J.M. (1984) A parsing architecture based on distributed memory machines. In *Proceedings of COLING-84*, Stanford, California.

Slack, J.M. (1986) Distributed memory: a basis for chart parsing. In *Proceedings of COLING-86*, Bonn, West Germany.

Slack, J.M. (1990) Getting structure from subsymbolic interactions. In G. Adriaens and U. Hahn (eds.), *Parallel Models of Natural Language Computation*. New Jersey: Ablex Publishing Co..