Computer Science Department Technical Report
Artificial Intelligence Laboratory
University of California
Los Angeles, CA 90024-1596

SYMBOLIC NEUROENGINEERING FOR NATURAL LANGUAGE PROCESSING: A MULTILEVEL RESEARCH APPROACH

Michael G. Dyer

September 1988
CSD-880075

Symbolic NeuroEngineering for Natural Language Processing:

A Multilevel Research Approach

Michael G. Dyer

August 1988

Technical Report UCLA-AI-88-14

OUTLINE

Symbolic NeuroEngineering for Natural Language Processing:

A Multilevel Research Approach *

Michael G. Dyer**

3532 Boelter Hall
Computer Science Dept.
UCLA, Los Angeles, CA 90024

Abstract

Natural language processing (NLP) research has been built on the assumption that natural language tasks, such as comprehension, generation, argumentation, acquisition, and question answering, are fundamentally symbolic in nature. Recently, an alternative, subsymbolic paradigm has arisen, inspired by neural mechanisms and based on parallel processing over distributed representations. In this paper, the assumptions of these two paradigms are compared and contrasted, resulting in the observation that each paradigm processes strengths exactly where the other is weak, and vice versa. This observation serves as a strong motivation for synthesis. A multilevel research approach is proposed, involving the construction of hybrid models, to achieve the long-term goal of mapping high-level cognitive function into neural mechanisms and brain architecture. Four levels of modeling are discussed: knowledge engineering level, localist connectionist level, distributed processing level, and artificial neural systems dynamics level. The two major goals of research at each level are (a) to explore its scope and limits and (b) to find mappings to the levels above and below it. In this paper the capabilities of several NLP models, at each level, are described, along with major research questions remaining to be resolved and major techniques currently being used in an attempt to complete the mappings. Techniques include: (1) forming hybrid systems with spreading activation, thresholds and markers to propagate bindings, (2) using extended back-error propagation in reactive training environments to eliminate microfeature representations, (3) transforming weight matrices into patterns of activation to create virtual semantic networks, (4) using conjunctive codings to implement role bindings, and (5) employing firing patterns and time-varying action potentials to represent and associate verbal with visual sequences.

---

* To appear in J. Barnden and J. Pollack (Eds.) (An initial version of this paper was presented at the AAAI & ONR Sponsored Workshop on High-Level Connectionism, held at New Mexico State University, April 9-11, 1988).

** The research reported here was supported in part by grants from the JTF Program of the DoD (monitored by JPL), the ITA Foundation, ONR, the Keck Foundation and the Hughes Artificial Intelligence Center. Hardware grants in support of this research were supplied by Apollo Computer, Hewlitt-Packard, the Keck Foundation and NSF. The computer programs described herein were implemented by my graduate students: Sergio Alvarado (now at UC Davis), Charlie Dolan, Michael Gasser (now at Indiana Univ.), Trent Lange, Risto Miikkulainen, Michael Pazzani (now at UC Irvine), Ron Sumida, Alan Wang and Uri Zernik (now at GE Labs). Helpful suggestions on this paper's content and organization were made by Margot Flowers.

1. Background and Motivation

In the last two decades, the natural language processing (NLP) systems built by Artificial Intelligence (AI) researchers have been designed under the *Physical Symbol Systems Hypothesis (PSSH)* (Newell 1980). Under the PSSH, knowledge is assumed to be represented in terms of relationships among symbols, and processes of comprehension, generation, application, retrieval, planning, reasoning and learning are modeled in terms of the creation and manipulation of symbolic structures, e.g. (Dyer in press). Research by computational linguists has also followed the PSSH, with syntactic knowledge represented as symbolic structures and parsing viewed as various transformations over them. While computational linguists and AI/NLP researchers have often held heated debates over paradigmatic assumptions, such as the role of syntax and universal grammar with respect to a theory of semantics and language acquisition (for example, see (Dresher and Hornstein 1976), (Schank and Wilensky 1977)), the shared assumptions of the PSSH have not been seriously challenged by either group.

Recently, however, what appears to be a radically new hypothesis for knowledge representation and processing has emerged. Research programs inspired by this new hypothesis vary considerably, which is to be expected, given the early stage of development. These research programs go by many names, including: *Neural Information Processing (NIPS), Parallel Distributed Processing (PDP)* (Rumelhart and McClelland 1986), (Touretzky and Hinton 1985), *Connectionism (CM)* (Feldman and Ballard 1982) (Feldman 1986), *Artificial Neural Systems (ANS)* (Grossberg 1982, 1988), *Neurocomputing* (Anderson and Rosenfeld 1988) and *Subsymbolic Processing* (Smolensky 1988). Here, I will use the term *Subsymbolic Processing Hypothesis (SSPH)* to refer to these related research paradigms in their broadest sense. Below I attempt to enumerate some of the major differences between the PSSH and the SSPH, at times perhaps exaggerating their contrasts, but if so, only to highlight the relevant issues.[1] These contrasts are summarized in Table 1.

PSSH (Symbolic)	**SSPH (Subsymbolic)**
Constraints are functional/logical	Constraints are brain/neural
Knowledge is distinct/localized	Knowledge is shared/distributed
Distinct tokens and types	Tokens equal types
Memory is static/structured	Memory is dynamic/reconstructive
Storage capacity unlimited	Fixed storage capacity
Separate data and processing	Data and processing merged
Rules are declarative	Rules emerge statistically
Parallelism is peripheral	Parallelism is central
Basic categories supplied	Basic categories emerge statistically
Models fragile to noise/damage	Models are robust/lesionable
Models are programmed	Models are trained or reinforced
Learning by proof and induction	Learning by incremental adaptation

Table 1: Symbolic vs Subsymbolic Hypotheses

[1] Ultimate computing power is normally not a major issue at the paradigmatic level, since nearly all subsymbolic models are simulated on symbolic architectures. Also, neural networks could be wired up to simulate Turing machines. Therefore, major issues have to do with perspective: i.e. which paradigm is most natural for understanding and modeling cognitive function?

1.1. Physical Symbol System Hypothesis vs *Subsymbolic Processing Hypothesis*

(1) Inspiration and constraints:

Research based on the PSSH is inspired both by introspection and observed problem solving behavior within a given task/domain. Constraints on symbolic models come mainly from logical and functional constraints of the task/domain. This point of view has allowed AI/NLP researchers to "cut the mind loose from the brain", largely ignoring brain organization and neural mechanisms.

SSPH models are inspired by, and get their major constraints from, brain architectures and neural mechanisms. The approach tends to be more bottom up, with researchers building models whose primitive processing units conform to various aspects of neural behavior. The resulting architectures are then explored to discover what they are capable of computing.

(2) Data and processing:

In PSSH models, data structures are kept separate from processing structures and processing is realized by one or more interpreters operating over program instructions. Knowledge is explicitly represented in terms of relationships between symbols. Memory is viewed as static and operated on by processes, which are dynamic.

In SSPH models, there is no separation between data structures and interpreters. Long-term knowledge is represented in terms of weighted links between primitive processing units while short-term knowledge is encoded as transient patterns of activation over these processing units. Memory is viewed as fundamentally dynamic.

(3) Hardware storage and memory management:

In PSSH models, symbols take up distinct locations in hardware memory. Symbols have no special internal structure (i.e. there is an arbitrary encoding, such as ASCII). Memory registers that are used to represent one piece of information are not allowed to represent some other information at the same time. Instances (tokens) are created by instantiating general templates (types) and are kept distinct from one another. Multiple tokens and types coexist without interference. There is unlimited free storage, supplied by a virtual memory management system.

In SSPH models, information is distributed throughout a network. Hardware connections are shared, i.e. used to represent or contribute to multiple pieces of information at the same time. Knowledge is often stored in terms of microfeatures, which allows the internal structure of symbol-like objects to be directly compared to one another. There is no type/token distinction and as a result of hardware sharing, multiple instances interfere with one another. Storage is limited; all information must reside in a fixed set of connections.

(4) Representation and interpretation:

Knowledge in PSSH models is represented in terms of predicates and the relationships among them. Inferences are represented in terms of rules which contain mappings between relationships. What is important is to capture functional and logical constraints of the task/domain in terms of inferences among associated symbols and their relationships.

Knowledge in SSPH models is represented in terms of energy landscapes, formed through a statistical analysis of the data. Rule-like behavior is an emergent property of the models, without the existence of explicit rules or rule interpreters. Inference is performed by associative retrieval and/or pattern completion, via reconstruction of patterns through basins of attraction in the energy landscape.

(5) Categorization and learning:

The basic categories and relationships in PSSH models are not learned, but are supplied by the programmer or knowledge engineer. Any learning that does occur arises through induction over pre-existing symbols and relationships. The result of learning involves: (a) the construction of new instances of existing knowledge structures, (b) the combination of existing structures into novel configurations, and/or (c) the acquisition of new rules to manipulate these structures.

Learning occurs as the result of a stochastic analysis of the data. Rather than learning new rules, SSPH-based models learn through repeated interaction with exemplars. The models cannot be programmed directly via instructions, but rather learn through adaptive modification of weighted links between simple processing elements. Instead of being supplied with symbolic categories, the models form category discriminations automatically, based on feedback, either from a trainer or the internal/external environment

(6) Psychological and neurological plausibility:

Since functional constraints are considered most critical, PSSH models do not make a commitment to a particular hardware architecture. All that matters at the architectural level is that fundamental symbolic operations be supplied for traversal, construction, binding and testing of symbolic structures. Parallelism is considered more an efficiency issue than a theoretical issue.

SSPH models are designed to capture constraints at the neural level. As a result, such models are lesionable and may be compared more directly to biological information processing systems. Massive parallelism is viewed as central, since the distributed nature of the representations requires it.

(7) Logic and adaptation:

In PSSH models, logic is viewed as both a means for representing knowledge and as a method for analyzing systems. Variables are of central importance and a major function is pattern matching and variable binding (e.g. unification). Knowledge is acquired (or applied) by generating proof chains, with constraints propagated along the chains via variable bindings. With recursive rules and variables, the models have infinite generative capacity.

In SSPH models, associative retrieval and parallel constraint satisfaction are considered basic operations. Variables are not currently available in these models. With a fixed memory, the models currently lack infinite generative capacity.

1.2. PSSH and SSPH Models Currently Inhabit Distinct Processing Niches

If either paradigm were superior along all or most processing dimensions, then one could be easily chosen over the other, but this is not the case. In fact, at this point it appears that purely symbolic and purely subsymbolic (distributed) models inhabit two distinct 'niches' in what might be termed an abstract processing 'ecology'. That is, what subsymbolic/PDP models do well, purely symbolic systems do poorly, *and vice versa*. These two niches are summarized in Table 2.

Purely Symbolic	Purely Distributed Subsymbolic	Capability
-	+	knowledge integration
-	+	smooth variation
-	+	intermediate representations
-	+	reconstructive memory
-	+	self-organization
-	+	associative retrieval
-	+	robustness to noise & damage
-	+	associative inference
-	+	adaptive learning
+	-	variables & bindings
+	-	schemas & roles
+	-	constituent, recursive structure
+	-	infinite generative capacity
+	-	defaults & inheritance
+	-	instantiations (types/tokens)
+	-	reference/pointers
+	-	memory management
+	-	intra-task communication
+	-	meta-reasoning
+	-	explanation-based learning
+	-	complex sequential control

Table 2: Distinct Processing Niches

In Table 2, the set of "+ -" pairs forms one niche, while the set of "- +" pairs forms the other. Here, each "+" represents support for the capability in the rightmost column, while each "-" represents lack of support. These capabilities are described below. The term "purely" is used here since there exist many ways to form hybrid systems, sharing features of both symbolic and subsymbolic. These hybrid approaches are discussed at length in later sections of this paper.

1.2.1. Natural for the Subsymbolic and Awkward for the Purely Symbolic

(1) *Knowledge Interaction and Integration* -- Since all information in subsymbolic models is represented homogeneously, in terms of numerical strengths of activation and weights, it is easy to have many distinct sources of knowledge interact through simple numerical operations, such as summation, multiplication, maximization and negation. In purely symbolic models, specialized rules and procedures must be devised for one kind of knowledge source to communicate or influence another knowledge source.

(2) *Smoothly Varying Knowledge Commitment* -- By varying the numeric value of a unit, the commitment to an interpretation can also vary in a smooth and continuous fashion. In purely symbolic systems, rules must either be deductive (given: p and p-implies->q, conclude q) or abductive (given: q and p-implies->q, conclude maybe q). In a subsymbolic system, unidirectional links can connect p to q and q to p, with varying weights. If the firing sequence is p followed by q and (a) the weight from p to q is large, then we have deduction, or (b) if the weight from q to p is large, then we have abduction. In addition, we can have all of the smoothly varying possibilities that lie in between these two extremes.

(3) *Intermediate Representations* -- In SSPH models, patterns of activation represent short-term knowledge. Since each activation value can smoothly vary, allowing the entire representation to enter intermediate states. For instance, if a room is represented as activation levels over units representing pieces of furniture and their properties, then a representation for a 'half' bedroom can

emerge that, in addition to containing bedroom furniture, contains a sofa, small fireplace, a refrigerator and a large dining table in it (Rumelhart and McClelland 1986).

(4) *Reconstructive Memory* -- Memory is by definition reconstructive, since the patterns on output are transient behaviors of the long-term memory traces that have been laid down into the interconnection weights. Interference occurs naturally in such memories, since similar forms of knowledge share more interconnection values. Symbolic models that attempt to explain memory confusions do so by assuming that different knowledge structures can point to shared substructures. For instance, a doctor visit and dentist visit would both point to a generalized office visit (Schank 1982). Such models have been successful at explaining cross-contextual remindings, but the symbolic approach appears unable to adequately explain how pieces of one memory can get intermingled at a microstructure level with pieces of related memories.

(5) *Self-Organization* -- Through autoassociation (i.e. mapping an input pattern to the same pattern on output) SSPH models can categorize an environment automatically. In contrast, knowledge engineers must supply symbolic models with the fundamental categories needed for forming representations. SSPH models thus have the capacity to form basic categories that were not anticipated by knowledge engineers.

(6) *Associative Retrieval* -- To retrieve information, one need only 'clamp' down one or more units on input and the network will settle into a configuration satisfying the largest number of constraints. It is easy to model influence of context in such a model simply by priming (increasing the initial activation of) a set of units before retrieval. Pattern completion results automatically in cases where a complete input pattern is not available. The pattern that arises is the result of all exemplars that were learned by the system. In contrast, symbolic systems operate by explicitly indexing knowledge and then performing a search to select the most restricted piece of knowledge that best matches the input. As a result, in symbolic models it is difficult to (a) represent the varying influence of experience on decision making and (b) handle the myriads of exceptions that always arise for any given rule.

(7) *Robustness to Noise and Damage* -- Since knowledge in subsymbolic models is distributed over a large number of processing units and weights, destruction of any random subset of units, or modification of a random set of weights, will have little effect on input/output behavior. In some models, noise actually improves function, since it serves to keep the model from retrieving local minima in the energy landscape. When noise or damage does alter I/O behavior, degradation is graceful, with the amount of degradation smoothly increasing as a function of the amount of damage. Since the representations are distributed, removal of processing units does not totally eliminate any specific piece of knowledge. In symbolic systems, the removal of a symbol causes permanent and localized damage to a specific piece of knowledge.

(8) *Adaptive Learning Algorithms* -- Most subsymbolic models learn through small, adaptive changes to their interconnection weights. Many learning algorithms operate by calculating an error between the desired and real output of the network. This error is locally computed and causes local adjustments of interconnection weights in parallel, throughout the network. Learning is similarity-based and consists of clustering and/or classifying data by some similarity metric within a very high-dimensional hyperspace. In general, any boolean mapping function can be acquired through learning. Since the learning is by incremental adaptation, it is common to require many weight modification cycles over the training set to arrive at the desired input/output behaviors.

1.2.2. Natural for the Symbolic and Awkward for the Purely Subsymbolic

(1)*Variables Bindings* -- In symbolic models, variables are capable of binding to any type of structure and new symbols can be created dynamically during execution. In subsymbolic models, variables do not (yet) exist and their function can only be partially realized, e.g. by specifying

ahead of time in the architecture the number of available symbols and their potential range of bindings, e.g. (Touretzky 1986).

(2) *Schemas and Roles* -- Schemas (frames/scripts) consist of a number of roles and their values, which can point to other schemas. Role bindings supply a mechanism for propagating information. For instance, when a schema-processing system is told "John entered the restaurant" a restaurant schema (Schank and Abelson 1977), (Dyer et al. 1987) is created with John bound to the diner role *throughout the schema*. Later, when the system is told "He showed him the menu." the system binds "him" to the diner role in a <waiter show menu to diner> event in the schema that matches the "show menu" description in the input. Since "him" is bound to the diner role and since the diner role is *already* bound to John throughout the schema, the system can automatically conclude that "him" refers to John. Since subsymbolic models lack variables and bindings, of course they cannot propagate them.

(3) *Constituent and Recursive Structure* --- Since symbol systems can keep instances separate and form bindings, it is easy to represent constituency and variable-depth recursive structures, such as "John told Bill that Fred told Mary that ...". In this case, each "told" structure can be kept separate, with the object of the inner "told" bound to a role in the structure of the outer "told". Constituency is essential to capture syntactic knowledge, such as the fact that, in passive transformations, the entire subject (noun phrase) becomes the object of "by". This noun phrase can be of variable length, thus causing problems for any PSSH model attempting to map active to passive by being trained on a fixed set of exemplars. Constituency and recursion is also essential in representing semantic information.

(4) *Infinite Generative Capacity* -- With variables, bindings and recursion, symbolic models possess infinite generative capacity (Chomsky 1965). That is, with a finite set of rules, an infinite number of instances can be generated (or recognized). Most subsymbolic models work on a finite number of instances. Subsymbolic models are trained on a finite training set, with a subset of the exemplars reserved. The model's generalization capability is then tested by seeing how well it performs its mapping task on the untested subset. Normally, the percentage of elements in the subset is around 5% - 20% of the entire training set.

(5) *Defaults and Inheritance* -- In symbolic models it is easy to set up "is-a" links from tokens to their types, and from types to supertypes, etc., in order to form inheritance hierarchies. Information can then be inherited from the higher levels. Subsymbolic models perform inheritance only to the extent that one representation shares features with another. For example, if a person (e.g. John) is represented in terms of features (animate, human, male) and another person (e.g. Mary) is represented as (animate, human, female), then any pattern completion operations over the Mary representation (i.e. processing units with those features) will influence the John representation, simply because the features are shared. But since there is no one place where knowledge about humans *in general* is stored in these subsymbolic models, it is not possible to add facts about humans in general and then automatically and immediately make this new knowledge available to all instances of humans, as is possible in symbolic models.

(6) *Instantiations (types/tokens)* -- In symbolic models, multiple instances of knowledge can be kept separate. Without symbols, variable and bindings, subsymbolic models suffer from "crosstalk" (Feldman 1986), in which additions of the patterns (John loves Mary) and (Fred loves Susan) can lead to the creation of the pseudo-memories: (John loves Susan) and (Fred loves Mary).

(7) *Reference and Pointers* -- A symbol gains its referential capability by maintaining the address of another symbol. The underlying (von Neumann) hardware is capable of retrieving the referenced symbol by accessing information at the desired address. In a symbolic model, the creation and augmentation of a memory for a unique entity consists in adding pointers to/from a

unique address in memory. The addition of a pointer allows a symbolic model to dynamically build up virtual structures. In subsymbolic models there are no addresses, since there are no unique locations for information. Consequently, there are no unique building blocks with which to explicitly construct larger structures out of smaller ones.

(8) *Memory Management* -- Symbolic models are supported by a virtual memory, which supplies new, unused storage for the construction of new symbol structures. In subsymbolic models, there is currently no way known to dynamically conscript new processing units and/or layers to allow memory capacity to grow as tasks change.

(9) *Intra-Task Communication* -- In symbolic systems, every interpreter is capable of recognizing symbols and manipulating the structures built out of symbols. Consider LISP; in it each function is built out of the primitive, structure manipulating operations: CAR, CDR, CONS and COND. Thus, the structure built by one function can immediately be traversed, compared and/or used by another function, and for some *other* purpose than that of the function that originally constructed it. Symbolic structures and structure-manipulating functions thus serve as an interlingua for all interprocess communication. In contrast, subsymbolic models live in isolated worlds. The weights formed by adaptation in one associative memory are specific to a given mapping task. It is not currently known how the long-term knowledge encoded in these weights can be interpreted by another subsystem, or applied to a novel task.

(10) *Meta-Reasoning* -- Since knowledge is composed of globally interpretable symbolic structures, one part of a symbolic system can reason, not only about knowledge produced by another subsystem, but also about itself and it own operations, to the extent they are encoded symbolically. The knowledge encoded in the weights in a subsymbolic model are designed for rapid, associative retrieval and recognition; however, this knowledge is unique to a specific task and not available for analysis.

(11) *One-Shot, Explanation-Based Learning* -- In symbolic models, associations between pairs can be memorized in one shot, upon entry. The resulting associations, however, do not lend themselves to generalization. In contrast, subsymbolic models require repeated passes through a corpus of data in order to form the distributed representations that support generalization. However, since the generalizations formed by subsymbolic models are basically the result of stochastic processes, their knowledge can be biased by the number of exemplars of a given type. In a rule-based system, an explanation for a novel input can be constructed from a single exemplar, through a process of rule chaining and theorem proving and statistical biasing does not occur.

(12) *Complex, Sequential Control* -- No matter how much memory and processing parallelism is available, as long as it is finite, the resulting system will lack infinite generative capacity. Therefore, in addition to virtual memory, feedback loops with sequential control are necessary in symbolic models to give them infinite generative capacity. Symbolic models are supported by programming languages developed over the last 40 years that are designed for specifying extremely complex architectures with modularity, hierarchical organization, feedback, communication and processing control. Currently, subsymbolic models lack high-level programming languages, so descriptions of processing are still very close to the "machine language" level.

1.3. Symbolic versus Subsymbolic in Humans and Animals

Watch a cat chase a rat, or a hawk pick its prey out of the air. Billions of years of evolution have produced neural networks that solve extremely complicated problems in learning to categorize and recognize complex sensory information, and to initiate detailed, coordinated motion. Yet, with all of the complexity of the billions of neurons in the neural circuitry of animals, they cannot engage in an argument over religion or politics; they cannot compare or contrast the meanings of words;

they cannot read a newspaper; they cannot play chess or monopoly; they cannot invent a mechanical device or musical composition, etc. Human language and problem-solving capacities appear to require a qualitative difference in the way the neurons we share with animals are organized. Human brains appear to contain additional 'modules', where abstract representations -- ones invariant with respect to sensory experience or modality -- are formed and linked to one another.

Consider current subsymbolic and symbolic representations for the letter "A" (Figure 1). The subsymbolic representation consists of an adaptive network designed for rapid visual recognition and categorization of a widely varying set of exemplars. The symbolic representation cannot begin to handle this kind of variation. However, the symbolic representation captures abstract, structural information.

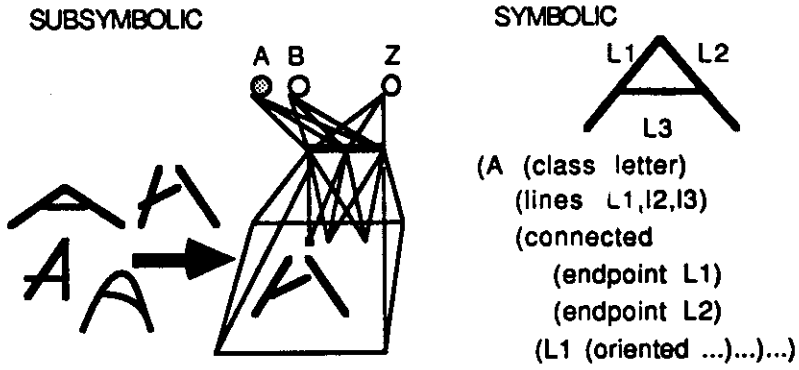

Figure 1.

A symbolic representation of the capital letter "A" consists of symbols representing line segments, their orientations, their regions, and connection relations among them. Thus a symbolic system can communicate what an "A" is to another system, via a symbolic language. A symbolic system can compare representations and answer questions, for example, about how similar or different an "A" is to a "C", "H", or "N", etc. A subsymbolic model forms a richer sensory representation of "A"s by seeing many variations, but the subsymbolic model can neither represent or communicate abstractions concerning that representation, e.g. that the prototypic "A" contains two line segments joined at their midpoints by a horizontally oriented third line.

Animals definitely have subsymbolic capabilities, but lack symbolic capabilities (or possess them in only the most rudimentary form). Humans clearly have both capabilities. From an evolutionary perspective, symbol processing is a very late development and has evolved on top of neural networks with largely subsymbolic capabilities. As a result, two speculative conclusions can be drawn: (1) symbol processing in humans may be fragile in places, since it has not had a very long evolutionary development, and (2) symbol processing may be computationally complex, since it consists of abstracting and coordinating the complex sensory information supplied by subsymbolic levels.

1.4. Symbolic versus Subsymbolic in Humans and Machines

Current computer models of subsymbolic processing exhibit a very different learning behavior than that exhibited by humans. Psychological experiments, e.g. (Bruner et al. 1956), (Bower and Trabasso 1968), (Wells 1963), indicate that people systematically experience different levels of difficulty in discovering conceptual categories, based on: (1) the number of attributes (e.g. blue) that are involved, (2) whether the attributes are positive or negative (e.g. not a square), (3) whether

the attributes are combined conjunctively or disjunctively. Levels of difficulty are summarized in Table 3.

Level of Difficulty	Attribute Type/ Combination	Form	Example
Easiest	1. Affirmation	x	squares
	2. Conjunction	x & y	small blue circles
	3. Disjunction	x or y	small circles or large squares
	4. Exclusive OR	x or y, but not both	circles & blue objects, but not blue circles
Hardest	5. Polymorphous	2-out-of-3 (x,y,z)	

Table 3: Relative ease of concept identification, based on number of learning trials required.

In contrast to human subjects, subsymbolic networks (specifically, PDP networks based on weight adjustment through error back-propagation) show systematic differences in ease of concept identification learning when compared to humans (Pazzani and Dyer 1987). Major differences are: (1) PDP networks learn conjunctive, disjunctive and polymorphous concepts with equivalent ease/difficulty. (2) The average number of trials in all cases is much greater for the networks than for human subjects. For example, when using conjunctions/disjunctions of three two-valued attributes, the PDP networks we applied to the concept identification task require eight times as many learning trials as do human subjects.

Concept formation at the symbolic level tends to follow Occam's razor (Pazzani et al. 1987), i.e. the simplest hypothesis is pursued first and only later abandoned in the face of conflicting evidence. In contrast, subsymbolic systems entertain multiple hypotheses and follow a gradient descent through a high-dimensional space. While subsymbolic systems can recognize complex, polymorphic categories, they are unable to follow the strategy of Occam's razor. As a result, it takes a subsymbolic system just as long to learn a simple category as to learn a complex one. In contrast, humans can apply prior knowledge to speed up the concept learning task, and, in many cases, learn a concept from a single exemplar, e.g. (Pazzani 1988). Suppose one encounters a single event in which a rich, blond-haired person is kidnapped. Immediately we now that the wealth is relevant and not the hair color. In a subsymbolic system, the importance of the wealth attribute could be represented by biasing the initial weights of the network. However, subsequent events might be about rich, blond-haired people getting skin cancer. Now it is the hair color that is relevant. What is needed is a representation that establishes separate causal relationships, between skin cancer and skin color on the one hand, and wealth with attempts at extortion on the other. Symbolic representations are designed for such a task.

It appears that humans use subsymbolic, statistical pattern recognition techniques to form categories in knowledge-poor areas, such as early visual learning in infants. Once some fundamental categories (e.g. colors, shapes) have been formed , humans use these categories symbolically, as building blocks for forming higher-level concepts and generalizations.

1.5. What are Symbols, Anyway?

In the ongoing, symbolic vs subsymbolic debate, e.g. (Dreyfus and Dreyfus 1988) (Smolensky 1988) (Reeke and Edelman 1988) (Pinker and Mehler 1988), there is a spectrum of positions. At one extreme, the argument is made that symbols are not needed and that attempts to recreate them in subsymbolic models are simply imposing over a hundred years of ill-conceived notions from

symbolic logic, e.g. (Churchland 1986). At the other extreme, the subsymbolic approach is viewed as just a disguised form of the simple associationism of Skinnerian behaviorism, that was laid to rest many years ago with Chomsky's (1959) devastating critique of B. F. Skinner's theory of language, along with the rise of symbol processing models in cognitive psychology and artificial intelligence (Fodor and Pylyshyn 1988). One model, around which the battle lines have been drawn, is that of McClelland and Rumelhart's (1986) PDP model for acquisition of the past tense. This model has recently received sharp criticism by (Fodor and Pylyshyn 1988), (Pinker and Prince 1988) and (Lachter and Bever 1988). These authors have argued that natural language requires symbolic structures to support the infinite generative capacity that it exhibits.

Do we need symbols for natural language processing? The answer to this question depends on what we mean by the term "symbol". If we mean the capacity to relate separate sources of abstract knowledge in a structured way, then clearly symbolic processing is needed. If we mean the implementation of unique bit vectors and addressing operations supplied by von Neumann architectures, then such symbols may very well be just conceptual baggage from the past. Perhaps at some point we may have to invent a new term (e.g. "symboid") to distinguish von Neumann symbols from symbols that are implemented in a distributed fashion. In this paper, however, I use "symbol" in its most abstract sense, to refer to the fundamental capacity for dynamically building and manipulating an unlimited number of abstract structures.

One of the most useful aspects of the symbolic vs subsymbolic debate is that it forces researchers to more carefully examine their assumptions concerning symbols and symbolic processing. Upon careful examination, it becomes clear that the notion of symbol and symbolic processing is not monolithic, but consists of a configuration of coupled assumptions and capacities. Subsymbolic models show that these assumptions and capacities need not always be so coupled. For example, it is possible to have a subsymbolic model in which reference and recursion is supported but binding capacity is impoverished. It is also possible to incorporate subsymbolic features of spreading activation, thresholds, weighted links, etc. in symbolic models. The construction of hybrid models is important in expanding our understanding of symbolic and subsymbolic processing.

2. Symbolic NeuroEngineering: Methodology for Synthesis

What we currently appear to have is a situation in which subsymbolic, distributed processing models exhibit massive parallelism, graceful error degradation, robust fault tolerance, and general adaptive learning capabilities, while symbol/rule based systems exhibit powerful reasoning, structural and inferential capabilities. If we could embed symbol representations and structure-manipulating operations within a distributed, subsymbolic architecture, then very powerful, massively parallel, fault tolerant high-level reasoning/planning systems could be created. At the same time, by embedding symbolic reasoning in an architecture closer to the level of neural processing, we might be able to shed light on how higher cognitive functions reside in brain mechanisms. These dual possibilities provide powerful motivations for attempting to synthesize symbolic with subsymbolic processing.

To achieve the desired synthesis we must avoid any overly straightforward, direct implementation of symbolic processing; otherwise the result will be a purely symbolic virtual machine. Why bother building a massively parallel architecture, if the symbolic operations that it supports are simply identical to those supported by a standard von Neumann architecture? In such a case, the model would behave exactly like symbolic models and many advantages of the subsymbolic level would be lost. Even the use of massive parallelism does not guarantee increased efficiency at the symbolic level. For instance, Touretzky's BoltzCONS (1986) model is extremely slow at the symbolic level, since it implements some of its primitive LISP list processing functions through simulated annealing (Hinton and Sejnowski 1986). Drew McDermott has dubbed the problem of

the use of massive parallelism at the subsymbolic level, resulting in extremely slow processing at the symbolic level, as the "Touretzky tarpit" problem. Feldman (1986) has criticized PDP models of symbol processing for being overly sequential at the symbolic level, even while remaining highly parallel at the subsymbolic level. A desirable synthesis, therefore, must be more subtle in nature, and result in associative inferential capabilities, with symbolic structures and categories dynamically learned while retaining the best features of distributed representations.

2.1. Need for Complex Architectures

One of the most attractive aspects of subsymbolic, distributed models is their potential for self organization. Unfortunately, there is a limit to the amount of knowledge a homogeneous, PDP architecture can learn through direct interaction with the data. The more complicated the knowledge and processing, the more architectural structure and stages of development/learning are needed. For instance, we cannot feed 20 years of raw sensory input to a 3-layer, feed-forward, back-error propagation network and then expect a college graduate to result. Even if we succeeded, we would still insist on having a *theory* of knowledge and processing. E.g. what architectures support which cognitive tasks? How does changing an architecture or training experience alter the resulting behavior or capabilities of a given model?

Consider the task of recognizing a highly abstract situation, such as natural language descriptions of "irresponsible behavior". Imagine feeding 10,000 paragraphs of text to the input layer of a standard, 3-layer, feed-forward PDP network. Suppose that these texts were broken up into 100 categories, where one category consisted of irresponsible behavior. A PDP network with 100 units on output could then easily 'represent' the 100 categories simply by activating a given output unit for a given input text. What is the chance that such a network could discover the conceptual features involved in recognizing irresponsible behavior and generalize correctly to novel texts? One story might involve a baby-sitter letting the baby fall into the swimming pool. Another story might involve a prosecutor failing to prosecute a criminal. But there could be many other stories involving baby-sitters and prosecutors that did not involve irresponsibility. If the network did manage to discriminate the desired stories in the training set, the chances would be very high that the network would be simply noticing combinations of very low-level features. Perhaps the stories describing irresponsible behavior in the training set just happen to have more letter "p"s and fewer letter "a"s in their input descriptions than stories in other categories. Clearly, without more existing structure, in a higher conceptual space of goals, agreements, beliefs and plans, the network could never successfully generalize to novel stories.

The limitation on the generalization capacity of such networks results from two weaknesses: (1) Appropriate levels of conceptual processing must already have been built up within the network architecture, since even humans cannot understand the concept of irresponsibility without first having learned about communication, agreements, agency, goals and their outcomes. (2) The architecture must support manipulation of representations that are highly abstracted from sensory data. One reason "irresponsibility" gives such networks a hard time is that it is at a very high conceptual level. That is, there are no specific, low-level features that lead to recognition of that category: *any* kind of specific behavior may or may not be irresponsible.

The "abstraction representation" problem exists not just in natural language processing, but in any case where the categories are fundamentally conceptual (versus perceptual) in nature. For instance, why is it that, in the visual domain, nonhuman mammals (e.g. pigeons, dogs) can learn to discriminate, say, different images of fish, but cannot learn to recognize images involving humorous, embarrassing, threatening (or other abstract) situations? Imagine a pigeon trying to categorize political cartoons into those that ridicule various kinds of ideological positions. The pigeon could never do it since it lacks the ability to represent the target categories or the abstract features from which they are constructed. Imagine a dog trained to categorize, e.g. into a "threat" category, images of a man raising his arm above another man's head while holding a club. What

will happen later when the dog sees that man holding a sign above the other man's head in a crowd? The dog will be incapable of correctly categorizing this image because it cannot reason that a sign has a different function than a club and that the man holding the sign, in this context, is raising it simply to make it more visible to view.

2.2. Need for Multiple Levels of Description in Theory Formation

Whatever theory of intelligence we develop, it will have to contain multiple levels of description. There are three general reasons that multiple levels arise:

(1) *Complexity* -- To manage complexity at lower levels of analysis, higher levels must be created, so that one can focus on and reason about major processes without being overwhelmed by the complexity of detail at the lower levels. Suppose, for instance, that emotional states turn out to be complex "trajectory fluctuations" in some high-dimensional energy "landscape". We will still want to be able to explain, for example, why the victim of a theft is mad at the person the victim believes had stolen the object from him (Dyer 1987). That is, we will still need a language that describes how a "trajectory fluctuation" representing anger is brought about by a prior "trajectory fluctuation" representing a belief concerning a "trajectory fluctuation" representing a goal and its violation. At some point it will become more efficient and perspicacious to simply drop the repeated term "trajectory fluctuation" and use directly the language of emotional states, goals, plans and beliefs, with the mapping to the underlying "trajectory fluctuation" calculus understood. As a result, we will end up with languages of description at different levels, along with a theory of how one level maps to the other.

(2) *Many-to-Many Mappings* -- In most models, a theory at a higher level can be implemented in many different ways at a lower level. In addition, an implementation at a lower level can represent many possible higher-level constructs. Consider a vector of bits passing through the bus in a von Neumann architecture. Without knowing the exact programming language, compiler and higher-level instruction being executed at that moment, it would be impossible to interpret those bits. They could be an instruction or data; they could be ASCII or EBCDIC or some other, internally managed coding convention. At the same time, consider a higher-level instruction before compilation or interpretation. Depending upon the available compiler and underlying machine, it could be mapped into any one of an arbitrary number of lower-level codes. Thus, although the higher level is implementable in the lower, the *languages* for talking about the higher (or lower) level may not be reducible, where irreducibility means that one cannot interpret or describe one level without simultaneously knowing the current status and context supplied by the other level. If this state of affairs is true for as simple an architecture as a von Neumann machine, with its relatively straightforward mappings between programming languages and machine operations, how much more so it will tend to be true for complex, distributed, dynamically adaptive learning architectures.

(3) *Tools versus Applications* -- When AI researchers developed property lists in LISP, there was great excitement, since it was realized that any kind of symbolic structure could be encoded using such property lists. However, the existence of property lists did not reveal the content of the structures that should be built out of them. Similarly, physicists, e.g. Hopfield (1982), Sejnowski (Hinton and Sejnowski 1986), have excited PDP researchers by pointing out that memories can be stored as energy minima. But knowing that energy minima are available does not reveal their appropriate content and relationships. This distinction, between the structure of a solution and its content, is similar to the distinction between the semantics of a tool and the semantics of the applications developed using the tool. For instance, logic programming can be used to produce an application with a completely distinct semantics from logic, as in the case of building models influenced by emotional states (Dyer 1987). A related distinction is that of necessary versus sufficient conditions. To explain intelligence we need both. Encoding memories as energy minima may turn out to be necessary for a theory of human and animal intelligence, but

it is definitely not sufficient. In some sense, by showing us that memories follow a Hamiltonian equation, we are being told that the brain does not violate any of the underlying laws of physics (something we had assumed anyway). Once we know the underlying tools for structuring memories, the focus of attention naturally shifts to (a) building memory content: which memories get encoded and when, and (b) modeling memory dynamics: how one memory brings about another (e.g. how emotional states are influenced by beliefs and goals).

2.3. Steps Toward Symbolic NeuroEngineering

Symbolic NeuroEngineering (SNE) is a general approach toward building content-oriented theories of knowledge. It receives its inspiration from two sources: (1) *Knowledge Engineering in the AI tradition*, with its emphasis on introspection and constraints from the logical and functional aspects of specific tasks within restricted domains of knowledge, and (2) *Neural Information Processing*, with constraints taken from currently available information about neural mechanisms and brain organization. Symbolic NeuroEngineering takes the following steps: (1) Specify knowledge structures and processes symbolically, using known knowledge engineering techniques in AI. (2) Encode these knowledge structures into a distributed form, using a variety of approaches, to be described below. The encoding may involve mapping symbolic knowledge through multiple levels of description. (3) Use and/or develop adaptive algorithms to reduce or remove the initial fragility of the knowledge structures originally encoded.

This approach allows one to incorporate the synthetic, task-driven orientation of AI knowledge engineering with the bottom-up, analytic approach that dominates in the neurosciences. The symbolic neuroengineering approach allows one to build complex architectures with knowledge encoded by hand, thus bypassing potentially long and complex developmental stages, while at the same time experimenting with how knowledge at a given stage is dynamically modified in the face of experience.

2.4. Four Levels from Mind to Brain

The research path from mind to brain is too complex to be achieved without passing through a number of intermediate levels. Of course, developing any intermediate level constitutes a risk, in that it may not lie on the ultimate path. This risk can be reduced somewhat by entertaining simultaneously a number of possible paths between varying models of processing at different levels of description. Our current research approach in NLP maps mind to brain through four intervening levels, illustrated in Figure 2.

There are two major goals of the research: (1) to understand the nature and limitations of models constructed at each level and (2) to discover methods for mapping from a given level to the levels above and below it. Although the second goal relies on the first, in many ways the mapping goal is more important, since we can validate or judge our theories at one level by how well it supports mappings to, and constraints from, the levels above or below.

1. Knowledge Engineering Level -- Examination of the structure/content of knowledge/processing needed to support high-level cognitive tasks (e.g. goal/plan analysis for narrative comprehension and belief justification for argument comprehension).

2. Localist Connectionist Network (LCN) Level -- Use of spreading activation, referent structures, thresholds and weighted links over semantic networks to support dynamic reinterpretation of word senses and multi-language generation.

3. Parallel Distributed Processing (PDP) Level -- Use of PDP models and adaptive learning algorithms to support the development of global distributed representations, recursive structures, role bindings, and propagation of bindings during inferencing.

4. Artificial Neural Systems Dynamics (ANSD) Level -- Use of artificial neural elements with real-time neural dynamics, for exploring various aspects of the "grounding" problem -- i.e. how language syntax and semantics can be learned through simulated real-time interaction with other sensory information that contains a temporal dimension.

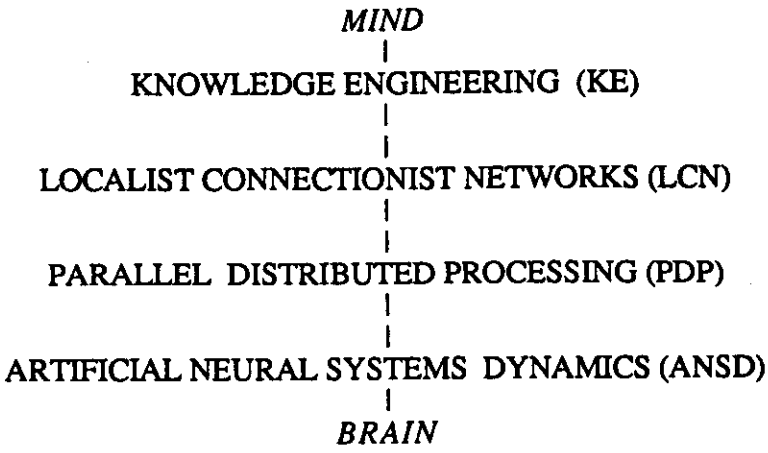

Figure 2: Levels of Research

Our approach is multileveled because each level of research both allows (and/or blocks) functional capabilities available at alternate levels. For instance, at the knowledge level it is easy to implement and bind variables, but difficult to dynamically form generalizations from the input. At the LCN level, it is easy to represent structured knowledge, but difficult to recover gracefully from damage. At the PDP level, it is easy to form distributed representations that are robust against damage, but difficult to represent structure and to implement bindings. At the ANSD level, time dynamics allow sequencing knowledge to be more easily captured than at the PDP level, but the systems are more complex to simulate and analyze.

3. Knowledge Engineering Level

The knowledge engineering level is needed to specify the kinds of structural relationships and inferences required for high-level cognitive tasks. The constraints supplied by this level are just as real as constraints from the neural level, but perhaps more important in the long run, especially if there turn out to be many different architectures capable of supporting high-level cognitive processing. The NLP systems we have constructed at this level are concerned with: (1) the relationship of belief/justifications to goals/planning, for comprehension of arguments and abstract themes, such as irony (Dyer, Flowers, Reeves in press), (2) the organization and acquisition of lexical information (Zernik and Dyer 1987), and (3) knowledge content and use within specific domains, such as law (Goldman, Dyer, Flowers in press) and mechanical engineering (Dyer, Hodges, Flowers in press). Since the knowledge engineering tradition and its AI technology are well known, the research at this level will just be briefly summarized for two models, along with some of the possibilities and problems these models pose for being implemented at lower levels.

3.1. Argument Units for Editorial Comprehension in OpEd

OpEd (Alvarado, Dyer, Flowers 1986) is a editorial text comprehension system designed to read, represent and answer questions about arguments in the domain of economic protectionism. OpEd's knowledge of argumentation is represented in terms of Argument Units (AUs), which

hold abstract belief attack/support structures, where the content of each belief refers to the efficacy of classes of plans and their outcomes. For example, the argument unit AU-NEG-SIDE-EFFECT (Figure 3) contains information about multiple goal outcomes of a plan and its effect on competing beliefs.

```
AU-NEG-SIDE-EFFECT:
 x believes [plan P1 should be executed]
        ---supported-by--->
 x believes [executing plan P1 will achieve G1]

y believes [P1 should not be executed]
        ---supported-by--->
y believes [executing plan P1 will achieve G1,
                executing plan P1 (as a side effect) will
                  cause failure of goal G2,
                G2 is more important than G1]
```

Figure 3. An Argument Unit

This AU commonly appears in economic/political arguments. For example, an economist might argue that printing more money (P1) will initially bolster the economy (G1), but will cause inflation as a side effect, which will hurt both business and consumers more in the long run (failure of G2).

Currently, OpEd can read and answer questions about two single paragraph-length fragments: (1) a Newsweek editorial by Milton Friedman on voluntary limits of Japanese automobiles ,and (2) an editorial by Lance Morrow on protection of U.S. motorcycle manufacturers from Japanese competition. A major aspect of the research involves categorizing and representing AUs, along with developing process models for recognizing and instantiating abstract AUs from lexical input.

3.2. Figurative Phrase Acquisition with a Phrasal Lexicon in RINA

The RINA program (Zernik and Dyer 1987) (Dyer and Zernik 1986) (Zernik 1987) is designed to learn novel figurative phrases in context. The RINA lexicon consists of phrasal patterns with associated conceptual and contextual patterns. For instance, given the entries in Figure 4, RINA is then told "David took on Goliath".

```
<x "took" y  "to" location>  -->  ((ACT  PHYS-TRANSFER)
                                      (OBJECT (GROUP x  y)
                                      (TO  (LOC location)))

<x  "on " y>   -->      ((RELATION ON-SURFACE )
                                      (OBJECT x)
                                      (SURFACE-OF y))
```

Figure 4. Phrasal/Lexical Entries.

Since neither of the relevant patterns matches the input, RINA analyzes the resulting discrepancies, at phrasal, conceptual and contextual levels, and then enters into a dialog with the user to receive more examples of the use of this phrase (e.g. "The researcher took on a hard AI problem"). With each exemplar, RINA refines its syntactic, conceptual and contextual hypotheses, using knowledge in its episodic memory, such as the major events of the David and Goliath story. As a result, RINA constructs a new entry in lexical memory, where <person1 "took on" person2> in a fight context maps to a representation for meeting a challenge.

3.3. Pure Knowledge Engineering: Pro and Con

Currently, abstract structures such as the Argument Units of OpEd cannot be represented for retrieval in a distributed fashion. The main reason is that the information contained in AUs is mainly structural and also highly abstract. For instance, any particular action could be a plan, and any state of affairs could be a goal, so no particular plan, goal or belief can trigger a given AU. What is required is a particular set of *structural relationships* among classes of goals, plans, and beliefs. In distributed models, a vector of features is used to retrieve a memory. But in AU theory, the same specific set of goals, plans and beliefs will indicated a different AU, depending on how these goals, plans and beliefs are related to one another.

At the same time, retrieval of an AU from a given set of cues can be viewed as a multiple-constraint satisfaction problem, and spreading activation is a very useful technique for satisfying multiple constraints. Therefore, one solution to the problem of retrieval of abstract, structured information is to add spreading activation, thresholds and linked weights to semantic networks, since such networks are already able to represent structural information. This hybrid approach, along with its advantages and disadvantages, is described in section 4 (localist connectionist level).

While symbolic models can capture abstract structure, it is difficult to model learning in situations where every piece of knowledge in memory influences every other piece of knowledge. For instance, RINA is currently incapable of learning subtle connotations that arise from related contexts of use. Consider the phrases "David took on Goliath" and "The AI researcher took on a hard AI problem". The meaning of "took on" in these contexts indicate effort and potential risk. Now consider the phrase: "John took on a new worker at the plant". RINA learns to map the phrase "took on" to HIRING in the context where the subject is an employer and the object an employee. However, there appears to remain a hint of a suggestion of effort/risk involved in this description, which is not evident in the phrase "John *hired* a new worker at the plant". In "John *took on* a new worker" there is the additional sense that John may have to spend more time training the new worker than in the more neutral term "hired". This hint of effort/risk appears to come from the influence of past uses of "took on", as in the fighting (David and Goliath) and problem-solving (AI research) contexts. In subsymbolic models, distributed representations support the ability of every memory to influence every other memory, which is extremely difficult to achieve in purely symbolic models. In section 5 (PDP level) we present a lexical acquisition model potentially able to capture such subtle contextual influences.

4. Local Connectionist Network Level

Localist spreading-activation networks (Cottrell and Small 1983) (Waltz and Pollack 1985) have significant advantages over traditional symbolic language understanding systems. Their conceptual knowledge is stored entirely in an interconnected network of simple computational nodes whose activations are calculated using a spreading-activation mechanism. This is opposed to the large collections of sometimes brittle and ad-hoc rules used in purely symbolic systems. More importantly, however, is that many different inference paths can be pursued simultaneously in localist networks, so that disambiguation and semantic re-interpretation can be automatically handled by having each concept providing evidence continuously to related concepts through spread of activation.

4.1. Advantages of LCNs for Word Interpretation

Consider the three major strategies of word sense disambiguation and reinterpretation used by pure symbol processing systems:

(1) *Delay* -- When an ambiguous word is encountered, interpretation is delayed as long as possible. There are a number of problems with this strategy. Determining how long to delay can be very difficult and in cases where words mutually disambiguate themselves, delaying can cause deadlock, where the interpretation of each word is waiting on the disambiguation of other words.

(2) *Commit and backtrack* -- In this strategy, a word sense is selected and if it later proves to be incorrect, the system backtracks to that branch point and recomputes an interpretation, using an alternate word sense. The major problem with this approach is that useful work is thrown away. Consider the sentence: "The pot on top of the new stove was smoked by the druggie." Suppose the COOKING-POT meaning of "pot" is first selected. When "smoked by the druggie" is encountered, backtracking will occur and MARIJUANA will be selected for "pot". However, as a result of backtracking, the correctly parsed substructure "on top of the new stove" will be thrown away during backtracking and will have to be reparsed with "pot" interpreted as MARIJUANA.

(3) *Commit and error correct* -- A more sophisticated approach than backtracking is to commit to an interpretation and upon discovering a discrepancy, instead of backtracking, perform intelligent error correcting operations. There are two major problems with this approach. First, error correction requires anticipating all of the kinds of errors that might be made, along with specifying their correction strategies. Second, once an error is made, other processes will come into operation, resulting in more errors on top of the original error. By the time the original error is noticed and correction strategies are invoked, a great deal of complex error correction may be required.

An alternative to the above strategies involves spreading continuous positive (and/or negative) activation values in parallel throughout a network of nodes and links. Structures that end up with highest activation values are selected as the correct interpretation. This technique has a number of advantages: (1) An interpretation can be biased or primed for simply by increasing the initial activation values on a selected set of nodes. (2) Each node can smoothly pass activation and/or inhibition to other nodes in a continuous process of feedback, until a stable configuration is found that satisfies the most constraints. (3) Mutual disambiguation can occur without deadlock and the effect of both backtracking and error correction is accomplished as a side effect of the flow of activation. (4) The mechanism is extremely simple and does not require the use of many ad hoc rules to achieve an integration between different sources of knowledge. Knowledge interaction is handled through simple numerical operations over scalar values.

4.2. Unresolved Problems for LCNs

LCNs also have a number of currently unresolved problems:

(1) *Category formation and self-organization* -- In LCNs with symbolic nodes and links, whose organization is specified by the knowledge engineer, it is unresolved as to how the primitive node classes and link types were originally learned. It is also unresolved as to how the initial organization of the network could have come about through learning.

(2) *Weight adjustment* -- In PDP models, a number of general algorithms are now available for automatically adjusting connection weights in response to input/output pairs. No such general algorithms yet exist for networks with semantically interpretable (i.e. symbolic) nodes and links. As a result, in such systems the connection weights must currently be specified by hand.

(3) *Variables, pointers, and bindings* -- LCN models can be divided into two classes: network models that allow nodes to store local bindings and to pass pointers among the nodes, and network models that are restricted only to passing activation. The pure LCNs (i.e. activation only)

are closer to the neural level, but lack the ability to dynamically create and propagate bindings. The LCNs that are augmented with markers (i.e. structures containing pointers) (Charniak 1983, 1986) (Riesbeck and Martin 1986) (Rau 1987) and local memory (i.e. variables) are able to create virtual networks, but are more removed from the neural level, since it is not clear how such symbol processing capabilities would be implemented.

A large portion of the NLP comprehension task involves dynamically building new structures and propagating bindings among these structures. Consider the text: "Mary and her friends were playing baseball. Her mother made her sweep up the broken glass." There are a great many structures of knowledge that must be selected, instantiated and linked up to build a representation of the conceptual information conveyed here. These structures include: (1) The baseball schema -- Mary and her friends are throwing a baseball among themselves; the default location is outside; they have a bat, etc. (2) Physical causality -- Windows break into pieces when hard objects are propelled at them. (3) Object functionality -- Brooms are used to collect small pieces of material on the ground. (4) Ownership and goals -- The person who owns an object will suffer a goal failure if the object is destroyed. (5) Responsibility schema -- The person x who causes a failure for another person y is often the one who must act as an agent for y to recover from the goal failure. (6) Authority -- A mother can request that her daughter perform certain actions without the bargaining required in other agency situations.

These knowledge structures must be selected from among the myriad other, irrelevant structures (e.g. that the mother gave birth to the daughter, that the broom was bought at a store, that baseballs are round, etc.). In addition to selecting them, they must be instantiated with the roles properly bound and these bindings must be propagated into each new structure as it is activated. For example, the one who caused the goal failure is the one obligated to perform recovery operations. In symbolic, rule-based systems, the propagation of bindings is specified by shared variables in the rules, e.g. [RESPONSIBLE (x) FOR GOAL-FAILURE (g) OF (y)] --implies--> [AGENT (x) FOR (y) TO-RECOVER(g)]. Pure LCNs are unable to propagate such bindings since they lack the ability to pass pointers and other symbolic information.

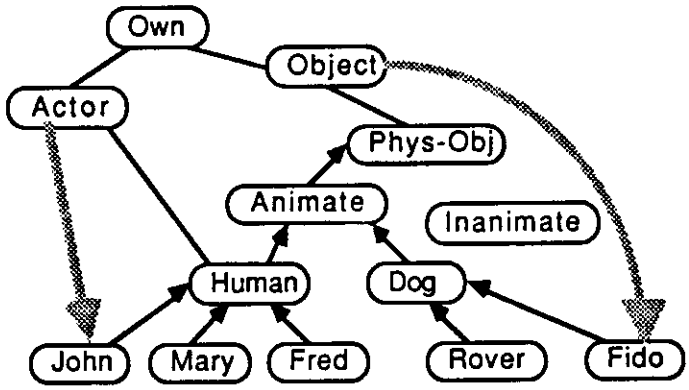

Figure 5: Dark lines represent fixed connections. Virtual connections (gray) are created by placing markers on nodes.

LCNs augmented with markers (pointers) are able to propagate bindings and build new structures. They accomplish these tasks by creating virtual networks over the existing network. Consider the simple network in Figure 5. By placing markers on the existing nodes, a virtual network can be constructed over the given, fixed network.

In Figure 5, the virtual subnetwork [John <-- Own --> Fido] is created by placing a marker on Own:Actor that holds the address of the John node and placing a marker on Own:Object that holds

the address of the FIDO node. This allows the creation of virtual nodes, but relies on the location-based addressing scheme supported by von Neumann architectures.

Markers can only be used to represent short-term bindings. For instance, if new markers are placed on the network without either removing (or distinguishing) older markers, then the problem of "crosstalk" can arise. Suppose others markers are placed on the Own:Actor and Own:Object nodes in Figure 5, pointing to, say, Mary and Rover, then confusion will result as to whether Mary owns Fido or Rover. It is assumed that, over longer periods of time, such transient connections are later turned into long-term memories by adding new instance nodes (Own1) and their roles, as illustrated in Figure 6.

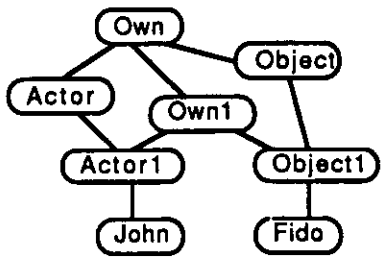

Figure 6: New node instances are indicated by the postfix "1".

In Figure 6, the crosstalk problem is solved by creating Own1, a new instance of Own, along with instances (Actor1, Object1) of each role binding. This technique solves the crosstalk problem, since memory confusions can no longer occur. However, humans do exhibit memory confusions. If such confusions are the result of generalization and learning over shared memory structures, then the solution in Figure 6 is too localist an approach.

In any case, adding long-term memories at the LCN level requires growing the network and it is currently not known how the creation of new nodes would be achieved at the neural level, but the solution at this level requires that neurons (or ensembles of neurons) be somehow conscripted to act as semantic network nodes/links.

4.3. Dynamic Reinterpretation of Words in CAIN

CAIN (Sumida, Dyer, Flowers, 1988) is a natural language conceptual analyzer capable of parsing texts that are conceptually and syntactically ambiguous, and that require dynamic reinterpretation. Consider the following text:

> John put the pot on the stove.... He picked it up and smoked it.

When CAIN reads the first sentence, activation on nodes for "pot" and "stove" cause the COOK-CONTAINER node to receive the highest activation. The subsequent input of "smoked it" spreads activation to MARIJUANA, thus causing "pot" to be reinterpreted.

CAIN operates by passing markers over a semantic network augmented with elements of LCNs, specifically: link weights, continuous activation values, and thresholds. The use of markers allows CAIN to construct virtual bindings between nodes. For example, John can be bound as the smoker by placing a marker on the ACTOR node that connected to the SMOKE node where the marker holds a pointer to the JOHN node. Without such markers, the system must either dynamically build new SMOKE and ACTOR nodes with links to JOHN, or have such a set of nodes already in existence in the network, for every possible combination of bindings. The

former approach is not neurally plausible, while the later approach would cause a severe combinatorial explosion.

Conceptual analysis of the input consists of four concurrent operations: (1) Mark lexical items and their associated meanings by spreading *activation markers* (AMs). (2) Find knowledge structures that connect marked nodes by spreading *search markers* (SMs). (3) Bind the roles of these structures by spreading *role markers* (RMs). (4) Find the most specific, connected structures by spreading *descendant markers* (DMs). Different classes of markers are constrained to spread only over specific types of links, thus restricting the search to relevant portions of the network.

AMs propagate from lexical nodes through a lexical/phrasal portion of the semantic network. The nodes in this portion of the network encode the phrasal-concept pairs of the RINA system (see section 3.2.). As a result, CAIN interprets <x "pick" y "up"> as physical-transer, instead of as initiate-date (e.g. John picked Mary up at the bar) or learn-information (e.g. John picked up tennis quickly).

SMs are propagated over is-a and role links, to activate node configurations representing knowledge structures. SMs receive a different activation strength, depending on the link weights over which they are propagated. Each knowledge representation node has a threshold value. If the incoming activation from the SMs exceeds the threshold, then the knowledge structure is selected, causing RMs to be propagated over role links. When an AM and RM intersect, an AM is placed on the role node. Since an AM holds a pointer back to its source, the role node effectively becomes bound to a concept node originally activated by the input. DMs are propagated down is-a links, to find the most specific nodes for binding. This process allows CAIN to bind JOHN as the ACTOR of SMOKE, rather than binding the more general node HUMAN as the ACTOR.

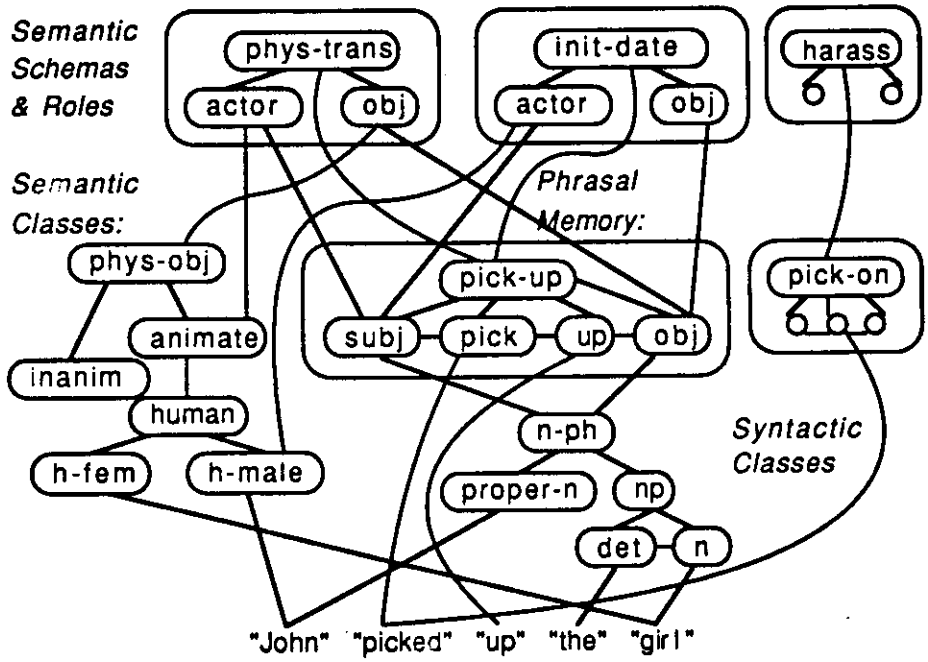

Figure 7. A simplified portion of CAIN's memory structures. (Not shown here are: connection weights, directionality of connections, isa and role connection labels, node thresholds and spread of markers.)

Figure 7 illustrates a portion of CAIN's memory for representing and processing phrases such as <x "pick" "up" y>. Here, each schema (i.e. head plus roles) is enclosed in a large oval. Connections between roles represent constraints. For example, the subject role of the pick-up phrase must be the actor of the physical-transfer or the initiator of the date event. In this illustration, the actor of the physical-transfer can be any animate entity and can transfer any physical object, while the actor of init-date must be a male human and can only date a female human. It is important to realize that the labels within nodes in Figure 7 are for mnemonic and debugging purposes only -- to help set up and analyze the initial network. These labels are not available to CAIN during processing.

As the words in the input are activated (bottom of Figure 7), activation spreads both to semantic and syntactic categories and then on to phrasal and semantic schemas. The final interpretation is represented by those schemas with the highest activation. In Figure 7, since John and Mary spread activation to h-male and h-female, the init-date schema receives more activation than phys-trans. Once init-date's threshold is exceeded, markers are passed down init-date and its role-nodes end up containing pointers to John and Mary. Phys-trans can also have its roles marked (bound) concurrently and, at any point, the bound schema with the highest activation represents the current best interpretation. Thus, as activation levels shift, reinterpretation of the input can occur.

By spreading continuous activation values over weighted links, many nodes can be simultaneously suggested while only those exceeding their thresholds are actually selected. In addition, the use of marker information allows dynamic bindings, thus avoiding combinatorial problems (i.e. pre-existing instance nodes for various instance combinations) while allowing the creation of new structures. For another example of a hybrid system (marker passing with spreading activation), see (Hendler 1987).

4.4. Automatic Formation of LCNs in Neural Networks

In pure semantic networks, both short-term and long-term information are created by dynamically adding new nodes and links to an existing network. These nodes and links are supplied, e.g. in LISP, by selecting new CONS cells from LISP's heap management system and setting the pointer fields of each CONS cell. The resulting structures rely on the underlying addressing-by-location mechanism that von Neumann machines support. In a more neurally realistic model, CONSing is not allowed: new memories do not appear to be constructed in the brain by having neurons sitting "off to the side", unused, and then suddenly becoming "linked up". First, all neurons appear to be active at some level at all times. Second, neurons reside for the most part in a pre-existing network of connections and the time scale for growth of new connections is too large to be useful for instantiation of knowledge that occurs on the order of seconds in NLP tasks. Finally, current models of neuronal behavior do not allow single neurons to send or interpret symbols or pointers.

The construction of a long-term, virtual semantic network (VSN) over a pre-existing neural network could occur in following way: (a) Each VSN node representing a class (e.g. humans) would consist of an ensemble of neurons (EN), with subsets of these nodes containing connections to different subsets of neurons in other ENs. (b) Novel instances (e.g. John) of a VSN would be created by activating different, distributed patterns of a subset of the neurons in an EN (Figure 8).

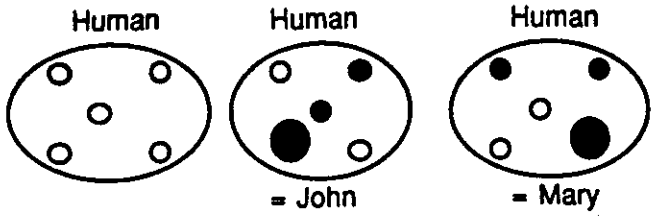

Figure 8: Patterns of activation over the same ensemble of neurons represents different instances of a class.

(c) New VSN links would arise by increasing/decreasing the connection weights between ensembles. (d) New VSN class nodes would be created (or destroyed) by increasing or decreasing the connection weights between subsets of neurons within an EN representing a given VSN node (Figure 9).

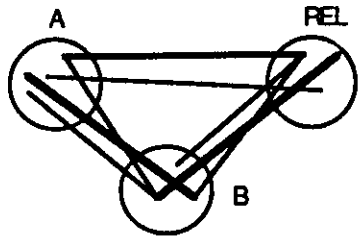

Figure 9: Connection strengths over ensembles establish conceptual relationships over classes.

(e) Connection weights between VSN nodes could be learned adaptively, through training and/or reinforcement. For example, in a training situation, a subset of VSN nodes could be "clamped" as input and activation would spread, causing other VSN nodes to become more highly active, serving as output. This output could then be compared with the desired output of the VSN, and connections would be adjusted incrementally to increase desired activation paths and decrease undesired activation paths in the VSN. Since each VSN node would consist of an ensemble of neurons, the system would have two major advantages: (1) it would be robust against damage and (2) it would be highly parallel at the knowledge level. Furthermore, by having instances be distributed as activation patterns over subsets of ENs, we would retain the PDP-level features of automatic generalization and dynamic categorization, while avoiding the limited memory capacity problem inherent in one-neuron-one-concept models.

The model would also exhibit memory interference for within-class objects while avoiding interference across classes. Humans appear to confuse multiple instances while keeping semantic categories separated. The proposed model would not lose a semantic category unless all of the neurons in its EN were destroyed, which would have a low statistical probability (Feldman 1986). However, destruction of an EN (through stroke or other physical damage) might result in the loss of a specific semantic category, such as relative loss of the ability to comprehend vegetables or small manipulable objects. Some patients exhibit such highly localized deficits (Warrington and McCarthy 1987). Such results are in keeping with the semi-localist view we have sketched above.

4.5. Completely Parallel Natural Language Generation in CHIE

In addition to constructing conceptual analyzers for natural language comprehension, we have built an LCN architecture for language generation. The system, CHIE (Gasser and Dyer 1986,1988) (Gasser 1988), uses representations that are similar to CAIN. However, CHIE is a pure LCN system; there are no markers and links are not discriminated (e.g. role versus is-a connections). CHIE operates completely in parallel, even during the production of a final sequence of words on output. This sequential behavior is achieved by propagating activation along uni-directional links and using inhibition and special winner-take-all (WTA) networks (Feldman and Ballard 1982) to determine which word gets selected for output. Let us reconsider more closely the phrase <x "pick" "up" y>, in Figure 10 (first shown in Figure 7). First, the phrase must contain sequencing information. We cannot say "Up John Mary picked". Second, optionality in the sequence must be representable, e.g. in addition to <x "pick" "up" y>, the pattern <x "pick" y "up"> is allowed.

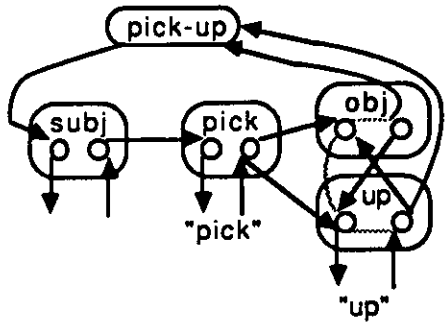

Figure 10: Sequencing in phrasal memory.
(Grey lines represent inhibitory connections.)

Sequencing and optionality is handled by the scheme illustrated in Figure 10. Each role-node of the phrase schema contains two nodes (small circles in the figure) for dealing with sequencing. The leftmost circle (LC) in each node represents initiation of this node in the sequence, while the rightmost circle (RC) represents termination. Assume that activation of a semantic schema (e.g. init-date) has spread activation to the pick-up schema for English generation. Activation spreads as follows: (1) Subj:LC spreads activation to generate the subject. (2) When the subject has been generated (not shown in figure), it passes activation back to subj:RC, which now passes activation on to the pick node. (3) Pick:LC spreads activation to "pick", causing this word (if it receives enough activation) to reach threshold and fire, thus producing English output and also passing activation back up to pick:RC. In this way, the entire sequence is produced in order by simple spread of activation over weighted connections and nodes with thresholds. Optionality within sequences is handled by alternate paths between LCs and RCs, along with inhibitory links between LCs to force the selection of only one path at a time.

The CHIE system is able to represent multi-linguistic constraints and model aspects of language knowledge transfer in humans. For instance, in multilingual speakers, "code switching" can occur, where the speaker inadvertently inserts a word or phrase from language L2 while speaking language L1 (Gasser 1988).

CHIE does not possess markers, so it cannot hold bindings in the way CAIN does. However, CHIE can hold a kind of temporary binding: by leaving a path of primed nodes as the result of activation. Before these primed nodes decay, if a node along this path receives activation, it will cause enough activation to spread back along this path to refire the primed nodes. This technique is

used in CHIE, for example, to influence the selection of pronouns during generation. Instead of generating, "Mary loves John", if "John" is already bound (through a path of primed nodes) to the hearer-role in the conversation, then CHIE can substitute the pronoun "you" for "John" and generate "Mary loves you" (Gasser 1988).

5. PDP Level

The most common PDP model consists of a network of 3-layers of simple processing units (input, hidden, output) that operates in a feed-forward manner. During learning, the connection weights are modified through a process of backward propagation of error, i.e. a difference between the current and desired output (Rumelhart et al. 1986). Tasks in such models are nearly always set up as mapping tasks, where the network is trained on a given corpus of input/output exemplars. The corpus is designed (or assumed) to have some implicit structure and it is the task of the network to discover it. Usually, a percentage of the training corpus is reserved (i.e. not shown to the network during learning). After the network has learned the training corpus, the reserved exemplars are presented to the network. If the network properly maps the reserved exemplars on input to appropriate representations on output, then the network is said to have generalized to novel instances. For natural language mapping tasks, the representations of exemplars on both the input and output layers are usually hand-coded in terms of feature vectors, where one or more units represents a feature, property or class.

5.1. The Syntax-to-Semantics Mapping Task and Semantic Microfeatures

McClelland and Kawamoto (1986) showed how a PDP model could learn to associate a syntactic representation on its input layer with a semantic (case-frame) representation on its output layer, and generalize to novel sentences. The syntax-to-semantic mapping task is non-trivial in English. For example, both sentences S1 and S2 have similar syntax:

S1: The boy ate the pizza with a fork.
S2: The girl are the hamburger with a pickle.

However, the semantic representation of S1 assigns "fork" as the instrument of the eating. In S2, although "pickle" is also the object of the preposition "with", its semantic interpretation is distinct. That is, the girl eats the pickle along with the hamburger, but the boy does not eat the fork along with the pizza.

In addition to similar syntactic cases/roles mapping to distinct semantic roles, distinct syntactic cases will also sometimes map to the same semantic role. Consider sentences S3 - S5:

S3: The man opened the door with a key.
S4: The key opened the door.
S5: The door opened.

Here, the words "door" and "key" change in syntactic role. In S3 "door" is the object of the verb; in S4 "key" is the subject; in S5 "door" is the subject. However, in these sentences, "door" must consistently map to the semantic patient (or recipient of the action) while "key" must map to the instrument used to achieve the action.

To obtain several hundred sentences for their training set, McClelland and Kawamoto used sentence templates, such as <HUMAN ATE FOOD WITH UTENSIL> and generated specific sentences by filling in each class in the template with a word from their lexicon. For example, given lexical entries such as "boy", "hamburger", "fork", the above template produces "The boy ate the hamburger with a fork". McClelland and Kawamoto represented the syntax of each

sentence in terms of a fixed number of processing units in the input layer, where each group of units represented the syntactic roles: SUBJECT, VERB, OBJECT, and PREP-PHRASE. They represented the semantics of each sentence also in terms of groupings of units (on the output layer), with the semantic case roles being: ACTOR, ACTION, PATIENT, INSTRUMENT, and PATIENT-MODIFIER.

Each syntactic/semantic role was filled in with a word, where each word was represented in terms of a vector of processing units. To represent the meaning of a word, they assigned a semantic interpretation to each unit making up the vector. Thus, each unit represented a "semantic microfeature", such as (MALE, FEMALE, INANIMATE). The semantic microfeature method of representation was used to make sure that words with similar meanings would possess similar representations, thus supporting later generalization after training. For instance, given that the PDP network was trained with S6 below, the network should successfully interpret the novel sentence S7, since the semantic microfeature representation for each pair: (John, Mary), (hamburger, pizza) and (fork, spoon) share active processing units.

S6: John ate the hamburger with a fork.
S7: Mary ate the pizza with a spoon.

For example, assuming that semantic microfeatures include (ANIMATE, INANIMATE, HUMAN, MALE, FEMALE), in that order, with binary-valued units, then representations for "John" and "Mary" would share values on 3 out of 5 units:

John: 10110
Mary: 10101

It is this sharing between vectors that supports generalization of the mapping task for novel inputs. In PDP models, since the connection weights were adjusted to accommodate multiple input/output representations, the memory serves to reconstruct the closest output patterns when presented with noisy or incomplete feature vectors on the input layer.

5.2. Automatic Learning of Distributed Word Representations with FGREP

In McClelland and Kawamoto's model, the semantic microfeature representations of the words were encoded by hand and remained static throughout the training and testing phases. Both of these aspects of the model pose general problems. First, having to encode representations by hand presents a knowledge engineering bottleneck; one even worse than that in AI, since feature vectors are impoverished to the extent that they lack the ability to represent constituent/recursive structure. It is no less difficult a task to decide what a correct set of semantic microfeatures should be than to decide what the correct set of primitive predicates and relations should be in an AI system. Second, by maintaining static representations for the exemplars, there is no way for the system to learn (i.e. form representations) about its environment. In McClelland and Kawamoto's model, this means that the system could not learn about words while learning about the syntax-to-semantic mapping task. Children, however, appear to learn the meanings of words while learning how utterances (with syntactic structure) relate to situations (with semantic significance). In general, PDP models form distributed representations in their hidden layers, but the representations used on the input and output layers remain the same during training and performance. But PDP models must be able to form novel representations of the I/O exemplars if global representations are ever to be formed.

To solve these problems, we have developed an extension to back-error propagation, called FGREP (Miikkulainen and Dyer 1987, 1988), that allows a PDP architecture to automatically discover representations for elements in the input/output layers. The basic approach of FGREP is twofold: (1) to represent the input, not as a pattern of activation on the input units, but as an

extra vector of modifiable weights, leading into the input layer, and (2) to extend back-error propagation learning into these representations. Each time the network is trained on an exemplar, the exemplar's representation becomes altered as its weight vector is modified. As a result, the network is, in some sense, "chasing a moving target" since the representations of the exemplars used in the training set are being altered while at the same time the network is attempting to learn the specified mapping task. Thus, the network is learning in a *reactive environment.*

We tried the FGREP method on the McClelland and Kawamoto task, using the same sentence generators and lexical entries. Instead of assigning each word a hand-coded, semantic microfeature representation, we initially assigned to each word a random vector of continuous values. As the FGREP method was applied, these vectors took on new activity profiles. Although the mapping task took a bit longer to learn, when the network reached convergence the lexical entries had formed fully distributed representations. The FGREP architecture for this task is illustrated in Figure 11.

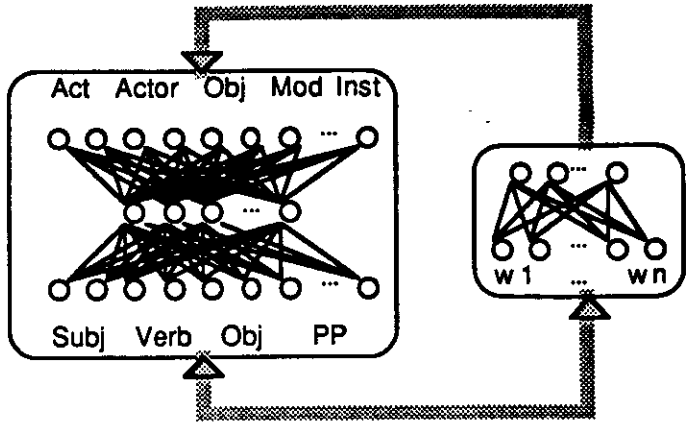

Figure 11: FGREP architecture for lexical acquisition while learning the syntax-to-semantic mapping task.

Here, a sentence is entered on the input layer by taking the weight pattern of a word in the lexicon and spreading activation over their weights to the units for that word's syntactic class (Subj, Verb, Obj, PP) in the specified input sentence. The same operation is performed to encode the word in its correct semantic role (Actor, Act, Obj, Mod, Inst) on the output layer. Back propagation of error is then performed on the connections in the hidden layers, with back propagation extended into the weights for the relevant words in lexicon. Each time the same sentence is presented to the mapping network, it is actually different, since the word representations have been altered as a result of encountering previous sentences containing those words.

Most interestingly, *words with similar semantics* (as defined by word use in the syntax-to-semantics mapping task) *end up forming similar distributed representations in the lexicon.* For instance, similar representations are discovered for words in each of the categories: human, predator, prey, utensil, etc. These are the categories originally used to set up the task, but which represent the hidden structure in the data -- initially unknown to the network.

Figure 12 shows the initial and final states of distributed representations for four words. Initially, their patterns are random and unrelated. After many iterations, words used identically have identical representations and words used nearly identically have very similar representations.

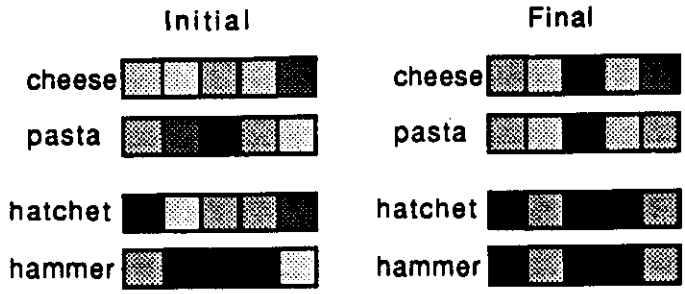

Figure 12: Each word is represented as a distributed pattern of continuous values.

The resulting theory of semantics contained within the FGREP approach is very different from that of traditional NLP, in which word meanings are represented in terms of symbolic structures and their expectations in terms of inference rules, e.g. (Dyer 1983). In the FGREP model, word representations are automatically formed through use. As a result of learning, *the representation of each word carries a memory trace of all the contexts of use that serve to define it.* If two words are used identically, their representations will converge. However, as their uses diverge, so also will their representations. Ambiguous words end up producing composite representations, sharing activity profiles with both classes/uses from which their meanings are drawn. The need to select semantic microfeatures and hand-code word representations is thus eliminated. The representations formed by FGREP are fully distributed and result in a system with better noise handling and generalization capabilities (than in microfeature-based systems), mainly because words with similar meanings have continuously merged representations. At the same time, the concept completion capabilities of PDP networks allows one to place one or more words in the input layer and generate a set of the most likely semantic case-role representations on the output, thus constituting a form of semantic expectations.

We are currently examining how this dynamic lexicon, with such holographically formed representations, can be used for expectation-based conceptual analysis of multi-sentence natural language text.

5.3. Constructing Virtual, Distributed Semantic Networks in DUAL

The semantic network (SN) formalism used by AI researchers to represent knowledge has two major advantages over the "semantic microfeature" vectors used in many PDP models: (1) *SN nodes hold only semantically relevant properties* -- A node in a semantic network will have only those few links (properties, features) emanating from it that explicitly point to other nodes. In a microfeature representation, each vector must contain *all* microfeatures, the majority of which will be inapplicable. For example, the microfeature ANIMATE will be present, although inapplicable, when representing a car; the microfeature METAL will be present, but inapplicable, when representing dog, etc. In general, it is problematic to have a representational formalism in which every feature or property, that something does *not* possess, must be explicitly represented. Such a representational formalism must result in extremely large vectors containing mainly inapplicable microfeatures.

(2) *Representation of constituent and recursive structure* -- Properties/features in a semantic network consist of labelled links connecting one node to another, thus allowing constituent and recursive structure (trees and cycles) to be represented. Since semantic microfeatures in PDP models reside in a fixed-width vector, the representation is "flat". Consequently, representations with variable depth, embedded structure cannot be captured directly in the representation. Furthermore, this "flatness" results in crosstalk: the inability to keep relationships separate. For

example, in a vector with features, A, B, C, D, REL1 and REL2, the representation of (A REL1 B) and (C REL2 D) will activate all of the features, thus making it impossible to avoid the creation of phantom structures, such as (A REL1 D) and (C REL2 B).

Many PDP models attempt to represent structure by assigning case-roles to groups of PDP units on the input/ouput layers. For instance, suppose we designate patterns over 10 units each, say, to represent the ACTOR, ACT, OBJECT, FROM and TO roles of an event; then we can encode with 50 units a case-frame meaning of the sentence "John walked home" as (ACTOR= John, ACT= physical-transfer, OBJECT=John, FROM=unknown, TO=home). But then how do we encode the sentence "Bill believes John walked home"? We could add two more roles (say BELIEF-ACTOR and BELIEF-OBJECT), but this will require changing the width of that layer of our architecture. Even if we do this, then how do we encode "John thinks that Fred wanted Bill to believe that John walked home."? Clearly, what we need is a way to encode variable depth, recursive structures in an architecture with fixed-width layers.

There are a number of ways of achieving this encoding, and all involve the use of recurrent PDP networks (i.e. networks with feedback between layers), where the representations encoded in the hidden units of one PDP network are used to construct novel representations in the input and/or output layers of another (or possibly the same) PDP network. For instance, we can first encode "John walked home" on both the input and output layers (as described in the paragraph above). Once the network is trained on this autoassociative task, we then save the compressed representation, C-R, formed in the hidden layer (Figure 13).

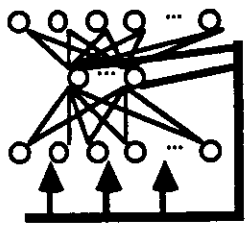

Figure 13: Compressed representation in the hidden layer of units is fed back into one of the groups of units (i.e. case-role) in the input layer.

We then feed C-R back into the input layer (Pollack 1988); this time with ACTOR=Bill, ACT=believe and OBJECT=C-R. In this way we can encode variable depth, recursive structures in a fixed-width PDP architecture.

To encode labeled, directed graphs with cycles, we use a related technique: one that involves manipulating the entire weight matrix in one PDP network as an input vector to the input layer of a larger network. That is, *the long-term information in the weights of one network can be transformed into a transient pattern of activation in another network.* This technique allows us to rapidly store, retrieve and modify entire networks as patterns of activation.

The architecture we have developed for exploring this technique is called DUAL (Dyer, Flowers, Wang 1988) and consists of two PDP networks, labeled STM and LTM. STM is used to represent a single node, Ni, in a semantic network as a set of input/output associations, where each input to STM encodes a semantic link, REL, and each associated output encodes a node Nj such that [Ni --REL--> Nj] occurs in the semantic network. Once all of the links emanating from Ni are encoded, the entire weight matrix (WMx) formed in STM is transferred to the input layer of LTM. The relations between STM and LTM are illustrated in Figure 14.

The transfer is accomplished through the use of sigma-pi connections (Rumelhart and McClelland 1986) (Pollack 1987), i.e. weighted links in one PDP network that connect to weighted links (versus units) in another PDP network. LTM is an autoassociative PDP network which stores STM weight matrices as patterns of activation. The hidden layer in LTM is the same length as the input/output layers in STM. Thus, the LTM hidden layer holds a compression of a given STM WMx that can then be feed back to the input and/or output layers of STM.

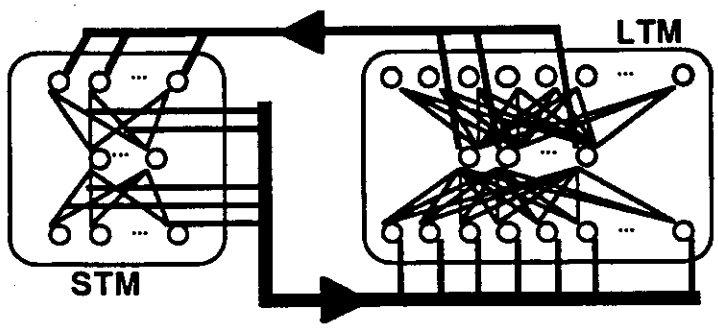

Figure 14: DUAL Architecture. Number of STM weights equals number of input/output units in LTM. Number of hidden units in LTM equals number of input/output units in STM.

Assume that we want to encode in DUAL a semantic node representing John, where the John node includes the following link/node associations:

[John ---OWNS---> Fido]
[John ---GENDER---> Male]
[John ---JOB---> Teacher]
[John ---LAST-NAME---> Dunn]
[John ---WIFE---> Mary]

Once the WMx for John (W-John) has been formed in STM (by associating semantic properties with their values), we train LTM to autoassociate W-John and then save the compression for W-John formed in the LTM hidden layer. Let us call this compression C-John. By entering C-John in the hidden layer of LTM, we can retrieve the WMx W-John, load it into STM and then, for example, place the representation for WIFE in the STM input layer and retrieve the answer "Mary" associatively on the STM output layer.

Recall, however, that "Mary" is also another node in the semantic network. Thus, Mary is represented as a different WMx (W-Mary) that has to be constructed (at a different time) through weight modification in STM, with its compression (C-Mary) also formed in LTM. Suppose that C-Mary has not yet been constructed at the time that the WMx for John is being constructed; then what is the representation for Mary that the John node should point to for the relation [John --WIFE---> Mary]? Clearly, when there are cycles between nodes there will always be at least one undefined node Nj at the time that another node Ni is being constructed that has a link to Nj. In such cases in DUAL, the undefined node is initially given a random pattern of activation as its representation. For [John ---WIFE--->Mary], the WIFE representation on STM input is simply associated with a random representation on STM output. Later, when W-Mary and C-Mary have been constructed, W-John must be updated to associate WIFE on input with C-Mary on output. This update (of the STM WMx for John) will cause all other nodes pointing to John to require updating also. These updates will propagate recursively throughout the entire virtual semantic network until convergence is achieved, at which point the entire semantic network is now

successfully encoded. This semantic network can then be traversed via (1) transfer of WMxs from LTM into STM to retrieve a given SN node Ni, and (2) STM associative retrievals to traverse any semantic relation [Ni ---REL---> Nj].

The process of encoding is described in more detail in (Dyer, Flowers, Wang 1988) and causes DUAL to discover distributed representations for all nodes of the virtual, semantic network that is being encoded. As a result, *SN nodes with similar associational links to similar nodes automatically develop similar distributed representations.*

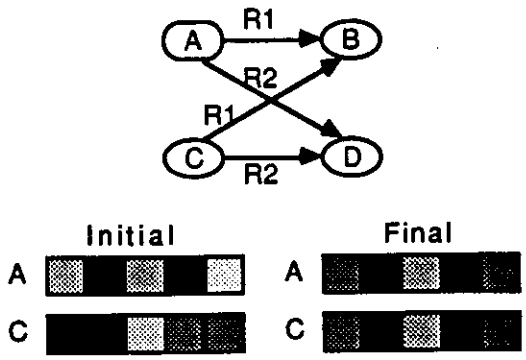

Figure 15: DUAL discovers distributed representations for the nodes of the encoded semantic network.

Figure 15 illustrates the kinds of activation patterns discovered by DUAL as the result of encoding a simple semantic network. Since both A and C point to the identical nodes with identically labelled links, their representations converge as a result of the encoding process.

The DUAL architecture thus allows us to manipulate a virtual semantic network at the symbolic level, while at the same time retaining many of the robust features (of learning, generalization, graceful error degradation, etc.) associated with PDP networks.

Recently, we have extended DUAL to dynamically discover distributed representations for relations/links connecting similar nodes. In addition to the compressions and generalizations exhibited by such distributed encodings of SN links and nodes, what other purpose might they serve? Consider the use of various causal links in semantic networks. Suppose one PDP architecture forms associations between events, where one event caused another. The compression for this entire matrix of event pairs could then become the representation for an abstract CAUSE relation/link at the symbolic SN level. Since the representation of this SN link had been formed via causal experiences, it could then be "opened up" upon demand, resulting in the retrieval of generalizations of all the event pairs that represented experiences underlying the abstract, symbolic notion of causality. Such a system could exhibit symbolic reasoning at one point, by manipulating symbols that are highly abstract and normally "'invariant" with respect to experience, while at another point exhibiting associative learning, through the dynamic modification of these abstractions through interactive experience with the world.

5.4. Implementing Role Bindings in CRAM

Currently, we can design PDP networks that dynamically discover distributed patterns of activity to fill roles in the input/output layers. However, representations of schemas in PDP networks still consist of encoding sequences of roles and their fillers (Touretzky and Hinton 1985). For

example, a case-role and filler in the restaurant schema can be encoded as the triple <predicate role-name role-value>. An instantiated restaurant schema might include the triples shown in Figure 16.

```
<restaurant1 type restaurant>
<restaurant1 diner John>
<restaurant1 food hamburger>
<restaurant1 init-event event0>
        <event0 followed-by event1>
        <event1 followed-by event2>
                ...
        <event1 act ingest1>
                <ingest1 type ingest>
                <ingest1 diner food1>
                        <food1 value hamburger>
        <event2 act trans1>
                <trans1 type transfer>
                <trans1 diner money1>
                <trans1 to  waiter>
                        ...
        ...
```

Figure 16: Encoding a restaurant instance as a set of <pred role filler> triples. (Indenting occurs only for visual clarity.)

In a symbolic system, we can bind every diner role in the restaurant simply by binding the diner role anywhere within the schema. In LISP, for example, diner is either a unique atom or a scoped variable, which makes its address unique within the specified binding environment. In contrast, binding all uses of a role in a PDP schema is a non-trivial task, since roles and values are not represented in terms of unique memory addresses, but are transient patterns of activation. If we encode a given pattern of activation for the diner binding in the ingest event, then we must explicitly find every other triple involving a diner and bind these also to that same pattern of activation. For example, if diner is bound to John within an ingest-food event, then we need to explicitly find all other diner-related triples, such as pay-for food, and bind the diner to the correct pattern. How can this be accomplished in a PDP model?

CRAM (Dolan and Dyer 1987,1988) is a PDP architecture designed to perform role bindings throughout a schema. Binding and rebinding of roles is accomplished through the use of conjunctive codings, probe networks, and projections.

(1) *Conjunctive coding* -- This form of encoding is described briefly in (Hinton et al. 1986) and differs from the standard implementations of distributed representations in PDP networks, which are termed *coarse codings*. In conjunctive coding, a set of roles are encoded as a conjunction of patterns of activation over N**D units, where D is the number of positions in a relation. For instance, to represent a set of <pred role filler> triples, one allocates a cube of units, where each side of the cube contains N units to represent the patterns of activation that will fill that position in the triple.

Suppose <ingest diner food> were to be represented as a conjunctive encoding, with the following (binary valued) patterns of activation: <ingest = (0 1 0 1), diner = (1 1 0 0), food = (1 0 0 1)>.

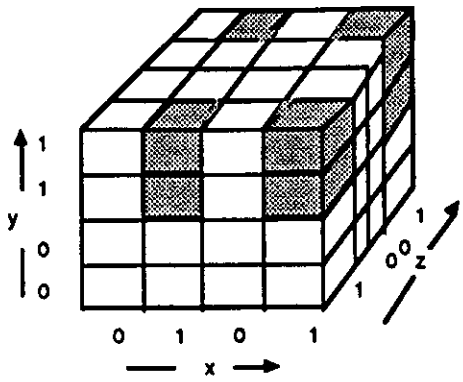

Figure 17. Conjunctive encoding of <ingest diner food> along the axes <x=pred, y=role, z=filler>.

Figure 17 shows how the triple <(0101),(1100),(1001)> is conjunctively encoded. In this figure, each pattern in one dimension is repeated as the pattern along the next dimension. For instance, the entire pattern (0101) along the x axis is repeated everywhere along the y axis wherever a 1 appears. Likewise, the entire <x,y> plane is repeated everywhere a 1 appears along the z dimension. Although conjunctive coding scales up less well than coarse coding, it retains the advantage that representations are distributed across dimensions, thus supporting merging and generalization of representations. Of course, like coarse-coding schemes, conjunctive coding of too many patterns in memory results in memory interference and creation of spurious memories (i.e. crosstalk).

(2) *Probe networks* -- A probe network is designed to retrieve all of the concepts from another network that satisfy a certain constraint. For example, if (a) the probe network is a cube of units, in one-to-one correspondence with the source network, where the source network encodes the triple <ingest, diner, food>, (b) the constraint (retrieval cue) is an input pattern of activation across the <x=ingest,y=diner, z=*> where * is unspecified, and (c) each unit in the probe network has a threshold high enough that it will become active only if receiving activation from both the source cube and the input pattern <ingest,diner>, then the probe network will retrieve the triple <ingest,diner,food> from the source network. Thus, conjunctive coding supports a relatively straightforward way of performing pattern completion (i.e. retrieval of complete triples, based on partial triples).

(3) *Projections* -- Conjunctive coding also supports retrieving all triples, say, <x=*, y=role, z=*> from memory and replacing them with triples where x and z are bound to different values. It is this capability that is needed to update every role in a schema by accessing all triples that contain that role and replacing memory with all triples (with that role) rebound. Updating all triples with a given binding in a given position (dimension) is accomplished by projecting n-dimensional networks onto networks with higher (or lower) dimensions. The process of projection is depicted in Figure 18.

3-D to 2-D projection is accomplished by taking a 2-D plane of units in the 3-D network and mapping them onto a 2-D network. Thus projection is equivalent to mapping triples into duples. Mapping from a duple to a triple simply requires the inverse projection, but with the pattern from the missing dimension filled in.

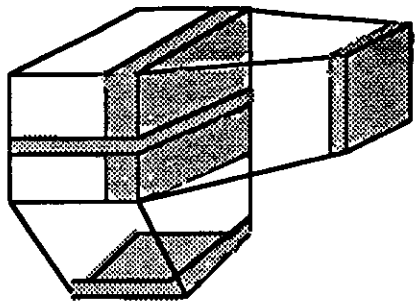

Figure 18: Projecting 3-D network onto 2-D networks.

Rebinding is thus accomplished by: (1) projecting 3-D onto 2-D (this replaces all triples with the desired duples), and (2) projecting the 2-D image, plus the desired 3rd value, onto a 3-D network. Finally, to decide which value to use as an index during projections, probe networks can be used to retrieve portions of existing triples from memory.

6. Level of Artificial Neural Systems Dynamics

There are two major weaknesses that are exhibited by nearly all current NLP systems:

(1) *Lack of grounding* -- Current NLP systems are not grounded in the physical world. That is, NLP systems do not have representations that relate words to sensory information from/about the world. Both symbolic and PDP models for NLP start with representations consisting of primitive syntactic/semantic case-roles (E.g. SUBJECT, ACTOR, OBJECT, MODIFIER) and relationships (e.g. CAUSE, OWNS, WIFE) and then build up more knowledge by linking up objects and relationships with one another internally. But there is no correspondence in these models between internal conceptual structures and sensory experience from the world. Failure to ground language in sensory experience has made it difficult to model language learning, since language in humans is acquired by forming associations between linguistic utterances and other sensory modalities. For instance, small children learn the word "ball" by hearing the word uttered while visually observing and manipulating various examples of balls.

(2) *Lack of temporal dynamics* -- In symbolic NLP systems, temporal relations may be represented declaratively, but the timing relationships between internal processing and the varying speed of arrival of the input are rarely modeled.[2] However, timing is usually extremely important in NLP processes, such as word interpretation. For example, if we do not read (or hear) one word follow another within a given time window, its interpretation will differ, as in S8 versus S9 below:

S8: John needs a car jack... he intends to ask for it.
S9: John needs a car ... [long pause]... jack he intends to ask for it.

In S9 we assume that John bought a car rather than a jack, and that "jack" refers to a person who is the subject of the next sentence. Most symbolic NLP systems simply do not deal with such time dynamics.

Most PDP models for NLP also lack time dynamics. The activation levels of PDP processing units are numbers intended to represent an average firing-rate frequency. Standard PDP models, i.e. in

[2] Speech processing is an exception, but in speech the focus is on word recognition. The focus in NLP is usually at or above the sentence level.

(Rumelhart and McClelland 1986), do not actually model the varying patterns of firing of individual spikes nor the time-dependent modification of action potentials. As a result, PDP models are highly abstracted from real neurons; e.g. weight modification during back-error propagation occurs without regard to timing dynamics and the weight update method used during back-error propagation is not directly realizable in real neurons.

Temporal dynamics and sensory grounding become extremely important when building a realistic model for language acquisition at the word and phrase level. Children hear utterances in time-varying temporal sequences while at the same time receiving time-varying information over other sensory modalities. It is by correlating these temporal sequences that sensory invariant abstractions are slowly formed, with knowledge gained by each modality supplying information to other modalities. Thus, visual and tactile knowledge of balls constrains hypotheses for how the word "ball" will be used syntactically and semantically, while knowledge about the use of the word "ball" in language may later supply information that helps in interpreting ambiguous or incomplete sensory information.

6.1. Grounding Language in DETE

The DETE program (Nenov and Dyer 1988) is designed to explore both the role of grounding and time dynamics in the acquisition of language syntax and semantics. DETE is composed of artificial neurons with temporal dynamics. That is, instead of using a single activation value to represent the average firing rate of a real neuron, the artificial neurons in DETE generate spikes and action potentials over time. The time dynamics of DETE neurons allow the model to associate a temporal sequence of words in its natural language input layer with a temporal sequence of visual images entering its retinal layer. DETE first learns the meaning of "ball" by being presented with different sized circles in different locations in its retinal field. DETE then can learn the meanings of phrases, such as "ball moves up" and "triangle bounced off wall" by associating visual motion sequences of known objects with language input sequences of known words.

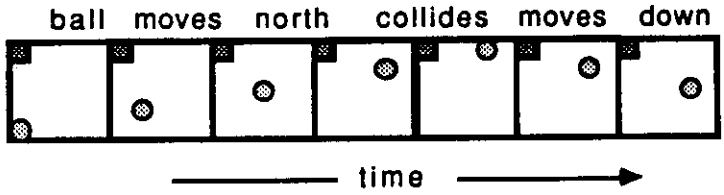

Figure 19: Input sequence of visual/verbal associations.

Figure 19 contains an illustration of the kind of visual/verbal sequences DETE receives as input. The neurons representing a given word may be clamped over the presentation of several visual frames. The model is designed so that the meanings of phrases can be learned even though the phrases are not entered in perfect temporal synchrony with visual motion events. For example, "bounces off" can be entered slightly after the visual event of bouncing and DETE will still ultimately learn the correct visual-to-linguistic mapping.

DETE is composed of four input/output layers:

1. Visual Field (VF) -- The VF consists of an array of processing units that act as an input retina. Information enters the retina as a sequence of binary arrays containing one of more simple objects (e.g. circles, squares) that may change position as the next array is entered as input. Information from the VF is passed in parallel to a number of feature extractors, to segment an object from the background and extract shape, size, intensity, location, speed and direction of motion.

2. Mind's Eye (ME) -- The ME is an output array for visual imagination. For example, once DETE has learned the meaning of "ball", by having seen various circles of various sizes, locations and/or directions of motion, DETE can be given the word "ball" as input and an "image" of a ball-like object will appear in the ME.

3. Sequential Lexicon (SL) -- The SL consists of pairs of artificial neurons, each pair connected to a pair of words from the lexicon. Activation of a given neuron in a pair indicates the order of arrival of the word pair in the input. For example, activation of the "left" neuron for the pair (SMALL, BALL) followed by activation of the "right" neuron for the pair (UP, BOUNCE) represents the input sequence: SMALL BALL BOUNCE UP.

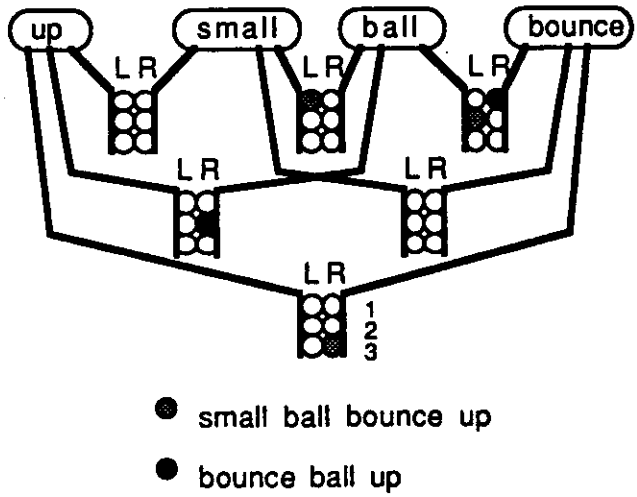

Figure 20: Encoding of two sentences.

Figure 20 illustrates how two sentences are encoded in a small Sequential Lexicon. The numbers indicate the position of each word-pair in the input sentence while L/R indicates which word of a given word-pair appears first.

4. Hidden Articulation Layer (HAL) -- The HAL consists of output neurons connected to lexical entities. Once DETE has learned the meaning of separate words, and then word phrases, DETE can be shown a sequence of a ball moving toward the top of the VF and it will produce the word sequence "ball moves up" in the HAL.

In addition to the I/O layers, DETE contains a number of internal layers, including: (1) A Selective Attention Layer (SAL) -- This module allows DETE to attend to a given object in the VF while receiving motion sequences along with word sequences. The design of the SAL and feature extractors are based on the design of the NeoCognitron (Fukushima 1987) visual recognition and selective attention system. (2) A Visual/Verbal Association Layer (VVAL) -- This layer contains connections between neurons in the Sequential Lexicon and neurons in the feature-extraction layers projecting from the Visual Field. For instance, "ball" is represented in the VVAL by forming strong connections between "ball" and the shape extraction neurons for circular shape, with weak connections to position, size, motion and direction-of-motion feature extraction neurons. As each word (and/or visual object) is input, the corresponding artificial neurons are clamped. Clamping a neuron causes it to generate a burst of spikes and the order and duration of clamping is of particular importance, since associations are formed in the VVAM by the way in which action potentials temporally overlap with one another. For example, if a neuron receives an impulse within a given time duration window of firing, its action potential is boosted.

In some ways the VVAM operates in a manner similar to that proposed by Smolensky (1987). In his model, two (or more) patterns of information are associated by forming tensor products. In the case of DETE, the tensor products formed in the VVAM contain complex timing and nonlinear dynamics which are difficult to analyze, but which allow DETE to associate and generalize temporal sequences across input modalities.

6.2. Future Work on Visual/Verbal Association in DETE

DETE is currently running in *LISP on a 16,834 processor CM2 Connection Machine, acquired by UCLA's interdisciplinary Center for Research in Computational Neurocognition. The Visual Field consists of 1,024 x 1,024 virtual processors. Although DETE contains a visual component for analyzing motion sequences of simple 2-D objects, the focus of our research is not on visual processing, but rather on "the grounding problem" -- how the availability of some temporal/visual information can aid in the acquisition of natural language syntax/semantics. To pursue this research we do not have to solve every difficult problem in vision, but must construct a system that is able to extract and apply some of the kinds of visual information that are useful in understanding verbal descriptions. Our current research is proceeding in several directions:

(1) *Generalization of word meanings and acquisition of syntax* -- DETE learns the meaning of "ball" and "block" by associating these words with circular and square objects in the visual field. A word like "object", however, may sometimes be used to refer to circles; other times, triangles, etc. DETE can learn the meaning of "object" through training sessions that result in weakening the connections between verbal neurons and neurons in the shape detection layer. However, what about the meaning of a function word, such as "it"? Consider sentences S10 - S12:

> S10: The circle collides with the wall and it moves down.
> S11: The circle moves toward the triangle and hits it.
> S12: [The circle is moving slowly toward the square and the square is moving rapidly toward the circle...] It is hit.

In S11, "it" refers to the circle and not the wall (because the wall is not moving in the visual field). In S10, "it" refers to the circle (based on syntactic grounds). In S12, "it" refers to the circle, mainly because English usage tends to treat the slower moving object as the recipient of the collision. Clearly, the interpretation of "it" requires a different level of generalization than that required of "circle" and "object".

(2) *Learning temporal descriptions* -- DETE currently learns to associate "moves" and "bounces" with motion and direction changes in the visual sequence. But consider S13 and S14:

> S13: The circle moved south before it moved north.
> S14: The circle looks like it will collide with the triangle.

S13 and S14 describe past and future events. Consequently, the visual sequence DETE observes will not correspond with the event sequence currently being described in DETE's word sequence layer. A research issue we are currently exploring is: whether DETE's current architecture can learn the semantics of past and future tense in simple English phrases, and if not, in what way the architecture must be modified to allow the learning of such abstractions.

(3) *Use of recurrent networks* -- At the present, DETE does not contain recurrent (feedback) connections, such as are needed to handle input sequences in PDP models, e.g. (Ellman 1988). Instead, DETE implicitly encodes sequences in the temporal dynamics of its neurons. In addition, DETE's mainly localist encoding of word sequences (as ordered word-pairs) eliminates much of the need for recurrence. We are currently examining the possible role of recurrent connections in DETE's architecture.

6.3. Future Research in Dynamic Binding through Temporal Patterns

PDP researchers have focussed on distributed, *spatial* encodings of information. At this point, very little is known about the use of *temporal* encodings. Most PDP models simply pass a single number (i.e. activation) that represents the average firing rate of a neuron. There is some evidence, however, that specific "pacemaker" neurons (Segundo et al. 1964) (Segundo and Kohn 1981) generate unique, preferred temporal patterns (Dayhoff 1988) over time and that these patterns are used to encode information. Recently, B. Richmond and L. Optican have concluded that spike-train patterns in the visual cortex are non-random and carry information encoded as compound, frequency-modulated signals, transmitted in a multiplexed fashion (reported in (Vaughan 1988)). If these conclusions are true, then perhaps ensembles of neurons are sending "tagged" information along with general activation values. Such tags could play the roles that markers play in LCN models.

Recently, we have designed a system, ROBIN (Lange and Dyer 1988), in which numbers are passed along paths in a LCN network. We term these numbers "signatures", since each number represents a unique identification of a node that is imagined to maintain a uniquely identifying activation (or firing pattern) over time. By passing signatures along connections, we can propagate markers. A dynamic binding is created when a node has a signature activation that matches the bound concept's signature. Most importantly, signatures can be passed in the model over very long paths of nodes to handle the non-local bindings necessary for inferencing in natural language understanding.

In general, it appears that high-level cognitive processing with neural networks will continue to pose very difficult practical and theoretical problems until equivalently powerful replacements for symbol-like variables and binding operations are found (Touretzky 1988).

7. Conclusions

The fact that the symbolic and subsymbolic paradigms have generally non-overlapping strengths and weaknesses is a strong motivator toward finding a synthesis. The approach we have presented here consists of generating hybrid systems at multiple levels of description. At each level, the models constructed both gain and lose capabilities with respect to the levels above and below. Research is then directed toward mapping capabilities from one level to another. The mapping techniques we are currently employing are summarized below:

(1) Mapping KE to LCN Level: (a) integrating marker passing with spreading activation, thresholds and weighted connections, to perform disambiguation and dynamic reinterpretation of word senses, (b) priming of paths to represent temporary bindings without markers, and (c) spread of activation over unidirectional, asymmetrical weights and node-pairs to handle sequencing in a parallel fashion for NL generation.

(2) Mapping LCN to PDP Level: (a) using reactive environments with extended back error propagation to discover distributed representations at the input/output layers, thus eliminating the use of semantic microfeatures, (b) transforming weights into patterns of activation in order to encode labelled, directed graphs, and (c) applying projections to conjunctive codings in order to perform role bindings within schemas.

(3) Mapping PDP to ANDS Level: (a) using action potentials and spike trains to encode temporal information, in order to associate verbal sequences with motion sequences, and (b) design of activation signatures for passing markers, without requiring symbol processing mechanisms.

In pursuing these multiple levels of research, we have: (1) constructed completely parallel language generation and comprehension systems, based on LCNs, (2) developed methods for automatically discovering distributed representations for semantic networks and lexical entities in PDP architectures, and (3) implemented an architecture for exploring the grounding problem by acquiring language syntax and semantics through visual/verbal association.

References

Alvarado, S., Dyer, M.G. and M. Flowers. Editorial Comprehension in OpEd through Argument Units. *Proceedings of American Association of Artificial Intelligence (AAAI-86)*, Philadelphia, PA 1986.

Anderson, J. R. and E. Rosenfeld (eds.) *Neurocomputing: Foundations of Research*. Bradford Book/MIT Press, Cambridge MA. 1988.

Bruner, J. S., Goodnow, J. J. and G. A. Austin. *A Study of Thinking*. Wiley, NY, 1956.

Bower, G. and T. Trabasso. *Attention in Learning: Theory and Research*. John Wiley and Sons, NY. 1968.

Charniak, E. Passing Markers: A Theory of Contextual Influence in Language Comprehension, *Cognitive Science* 7 (3), 1983.

Charniak, E. A Neat Theory of Marker Passing. *Proceedings of the Fifth National Conference on Artificial Intelligence*. Philadelphia, PA. pp. 584-588, 1986.

Chomsky, N. *Aspects of a Theory of Syntax*, MIT Press, 1965.

Chomsky, N. A Review of B. F. Skinner's *Verbal Behavior*. In *Language*, Vol. 35, No.1, pp. 26-58. 1959.

Churchland, P. *Neurophilosophy: Toward a Unified Science of Mind-Brain*. MIT Press, Cambridge MA. 1986.

Cottrell, G. W. and Small, S. L. A Connectionist Scheme for Modelling Word Sense Disambiguation, *Cognition and Brain Theory*, 6 (1), 89-120, 1983.

Dayhoff, J. E. Temporal Structure in Neural Networks with Impulse Train Connections. *Proc. of IEEE Second International Conference on Neural Networks*, Vol. II, pp.33 -45, San Diego, 1988.

Derthick, M. and D. Plaut. Is Distributed Connectionism Compatible with the Physical Symbol System Hypothesis? *Proceedings of the Eighth Annual Conference of the Cognitive Science Society (Cog-Sci-86)*. Amherst, MA,. 1986.

Dolan, C. and M. G. Dyer. Symbolic Schemata, Role Binding and the Evolution of Structure in Connectionist Memories. *Proceedings of the IEEE First Annual International Conference on Neural Networks*. San Diego, CA, June 1987.

Dolan, C. P. and M. G. Dyer. Parallel Retrieval and Application of Conceptual Knowledge. Technical Report UCLA-AI-88-3. Jan. 1988. (Prepared for *Proc. of AAAI Spring Symposium Series*. Parallel Models of Intelligence: How Can Slow Components Think So Fast? Stanford, CA, March 1988.)

Dresher, B. E. and Hornstein, N. On some supposed contributions of artificial intelligence to the scientific study of language, *Cognition*, 4, 321-398, 1976.

Dreyfus, H. L. and S. E. Dreyfus. Making a Mind Versus Modeling a Brain: Artificial Intelligence Back at a Branchpoint. *Daedalus: Journal of the American Academy of Arts and Sciences*. Special Issue on Artificial Intelligence. pp. 15-43. Winter, 1988.

Dyer, M.G. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension.* MIT Press. Cambridge, MA 1983.

Dyer, M.G. Emotions and Their Computations: Three Computer Models. *Cognition and Emotion*, Vol. 1, no. 3, 323-347, 1987.

Dyer, M.G. Knowledge Interactions and Integrated Parsing for Narrative Comprehension. In D. Waltz (ed.), *Advances in Natural Language Processing*, , LEA Press, Hillsdale, NJ. (in press).

Dyer, M.G., Cullingford, R. & Alvarado, S. SCRIPTS. In Shapiro (ed.) *Encyclopedia of Artificial Intelligence.* John Wiley & Sons, 1987.

Dyer, M. G., Flowers, M. and Wang, Y. A. (1988) "Weight-Matrix = Pattern of Activation: Encoding Semantic Networks as Distributed Representations in DUAL, a PDP Architecture" Technical Report UCLA-AI-88-5. CS Dept. UCLA, Los Angeles, CA. 1988.

Dyer, M., Hodges, J. and Flowers, M. Computer Comprehension of Mechanical Device Descriptions. In J. Gero (ed.) *Knowledge-Based Systems in Engineering and Architecture.* Addison-Wesley (in press).

Dyer, M.G. Reeves, J. and M. Flowers. A Computer Model of Irony Recognition in Narrative Understanding. *Advances in Computing and the Humanities*, Vol. 1, No. 1 (in press).

Dyer, M. G. and Zernik, U. Encoding and Acquiring Figurative Phrases in the Phrasal Lexicon, *Proc. of 24th Annual Meeting of the Association for Computational Linguistics*, New York, 1986.

Ellman, J. L. Finding Structure in Time. Technical Report 8801, Center for Research in Language. UCSD, San Diego. 1988.

Feldman, J. A. Neural Representation of Conceptual Knowledge. Technical Report TR189, Dept. of CS, Univ. of Rochester, NY. 1986.

Feldman, J. A. and D. H. Ballard, Connectionist Models and Their Properties. *Cognitive Science.* Vol. 6, 1982.

Fodor, J. A. and Z. W. Pylyshyn. Connectionism and Cognitive Architecture: A Critical Analysis. In Pinker and Mehler (eds.) *Connections and Symbols*, Bradford books/MIT Press, 1988.

Fukushima, K. A Neural Network Model for Selective Attention. *Proceedings of the IEEE First International Conference on Neural Networks.* Vol. 2, pp. 11-18, San Diego, CA. 1987.

Gasser, M. *A Connectionist Model of Sentence Generation in a First and Second Language.* UCLA Ph.D. and Technical Report UCLA-AI-88-13, July 1988.

Gasser, M. and M. G. Dyer. *Speak of the Devil:* Representing Deictic and Speech Act Knowledge in an Integrated Lexical Memory. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society.* Amherst, MA. 1986.

Gasser, M. and M. G. Dyer. Sequencing in a Connectionist Model of Language Processing. *Proceedings of Twelfth Intern. Conf. on Computational Linguistics (COLING-88).* Budapest, Hungary. 1988.

Goldman, S., Dyer, M.G. and M. Flowers. Representing Contractual Situations. In C. Walter (ed.) *Computing Power and Legal Language.* Greenwood/Quorum Publishers (in press).

Grossberg, S. (ed.) *Neural Networks and Natural Intelligence.* Bradford Books/MIT Press, Cambridge MA. 1988.

Grossberg, S. *Studies of Mind and Brain.* Reidel, Boston. 1982.

Hendler, J. A. Marker-passing and Microfeatures. *Proc. of Tenth International Joint Conference on Artificial Intelligence.* pp. 151-154, Milan, Italy. 1987.

Hinton, G. E. and Sejnowski, T. J. Learning and Relearning in Boltzmann machines. In *Parallel Distributed Processing*. Vol. 1, McClelland and Rumelhart. Cambridge, MA: MIT Press/Bradford Books, 1986.

Hinton, G. E., McClelland, J. L. and D. E. Rumelhart. Distributed Representations. In Rumelhart and McClelland. *Parallel Distributed Processing*, Vol. 1, Bradford Book/MIT Press. 1986.

Hinton, G. E. Learning Distributed Representations of Concepts. *Proceedings of the Eighth Annual conference of the Cognitive Science Society.*, pp. 1-12, Amherst, MA. 1986.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. of National Academy of Sciences*. Vol. 79, pp. 2554-2558, 1982.

Lachter, J. and T. G. Bever. The Relationship between Linguistic Structure and Associative Theories of Language learning -- A Constructive Critique of Some Connectionist Learning Models. In Pinker and Mehler (eds.) *Connections and Symbols*, Bradford books/MIT Press, 1988.

Lan[illegible] T. and M. G. Dyer. Dynamic, Non-Local Role Bindings and Inferencing in a Localist Network for Natural Lan[illegible]age Understanding. To appear in *Proceedings of IEEE Conference on Neural Information Processing Systems - Natural and Synthetic (NIPS-88)*, Denver, Colorado, to be held Nov. 28 - Dec. 1, 1988.

McClelland, J. L. and Kawamoto, A. H. Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences. In McClelland and Rumelhart (eds) *Parallel Distributed Processing*. Vol 2, Cambridge, MA: MIT Press/Bradford Books, 1986.

Miikkulainen, R. and Dyer, M. G. Building Distributed Representations without Microfeatures. Tech. Rep. UCLA-AI-87-17, Computer Science Dept. UCLA, August 1987.

Miikkulainen, R. & Dyer, M. G. Forming Global Representations with Extended Backpropagation. *Proceedings of the IEEE Second Annual International Conference on Neural Networks (ICNN-88)*, San Diego, CA. July 1988

Nenov, V. I. and M. G. Dyer. DETE: Connectionist/Symbolic Model of Visual and Verbal Association. *Proceedings of the IEEE Second Annual International Conference on Neural Networks (ICNN-88)*, San Diego, CA July 1988

Newell, A. Physical Symbol Systems. *Cognitive Science* (2), 1980.

Pazzani, M. J. *Learning Causal Relationships: An Integration of Empirical and Explanation-Based Learning Methods*. UCLA Ph.D. and Technical Report UCLA-AI-88-10, Los Angeles, CA. 1988.

Pazzani, M. J. and M.G. Dyer. A Comparison of Concept Identification in Human Learning and Network Learning with the Generalized Delta Rule. *Proceedings of 10th Intl. Joint Conference on Artificial Intelligence (IJCAI-87)*. Milan, Italy, August 1987.

Pazzani, M. J., Dyer, M.G. and M. Flowers. Using Prior Learning to Facilitate the Learning of New Causal Theories. *Proceedings of 10th Intl. Joint Conference on Artificial Intelligence (IJCAI-87)*. Milan, Italy, Aug. 1987.

Pinker, S. and J. Mehler (eds.). *Connections and Symbols*. Bradford Book, MIT Press, 1988. (Special issue of *Cognition: An International Journal of Cognitive Science*, Vol. 28).

Pinker, S. and and Prince, A. On Language and Connectionism: Analysis of a Parallel Distributed Processing Model of Language Acquisition. In Pinker and Mehler (eds.) *Connections and Symbols*, Bradford books/MIT Press, 1988.

Pollack, J. B. Cascaded Back-Propagation on Dynamic Connectionist Networks. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*. Seattle, Wash. 1987.

Pollack, J. Recursive Auto-Associative Memory: Devising Compositional Distributed Representations. Technical Report MCCS-88-124, Computing Research Lab. New Mexico State Univ., Las Cruces, NM, 1988.

Rau, L. F. Spontaneous Retrieval in a Conceptual Information System. *Proc. of Tenth International Conference on Artificial Intelligence.* pp. 155-162, Milan, Italy. 1987.

Reeke Jr., G. N. and G. M. Edelman. Real Brains and Artificial Intelligence. *Daedalus: Journal of the American Academy of Arts and Sciences.* Special Issue on Artificial Intelligence. pp. 143-173. Winter, 1988.

Riesbeck, C. K. and C. E. Martin. Direct Memory Access Parsing. In Kolodner and Riesbeck (eds.) *Experience, Memory, and Reasoning.* Hillsdale, NJ: LEA Press, 1986.

Rumelhart, D. E., J. L. McClelland (Eds.) *Parallel Distributed Processing: Explorations into the Microstructure of Cognition* (Vols. 1 and 2). Bradford Book/MIT Press, Cambridge, MA. 1986a.

Rumelhart, D. E. and McClelland, J. L. On Learning the Past Tenses of English Verbs. In McClelland and Rumelhart (eds) *Parallel Distributed Processing,* Vol 2, Cambridge MA: MIT Press/Bradford Books, 1986b.

Rumelhart, D., Hinton, G. and R. Williams. Learning Internal Representations by Error Propagation. In Rumelhart & McClelland, *Parallel Distributed Processing.* 1986.

Schank, R. C. and R. Abelson, *Scripts, Plans, Goals and Understanding,* Hillsdale, NJ: LEA Press, 1977.

Schank, R. C. *Dynamic Memory,* Cambridge, England: Cambridge University Press, 1982.

Schank, R. C. and R. Wilensky. Response to Dresher and Hornstein. *Cognition,* Vol. 5, pp. 133-145, 1977.

Segundo, J. P., Perkel, D. H., Shulman, J. H., Bullock, T., H. and G. P. Moore. Pacemaker Neurons: Effects of Regularly Spaced Synaptic Input. *Science,* Vol. 145, No. 3627, pp. 61-63, 1964.

Segundo, J. P. and A. F. Kohn. A Model of Excitatory Synaptic Interactions Between Pacemakers: Its Reality, its Generality, and the Principles Involved. *Biological Cybernetics,* Vol. 40, P. 113-126, 1981.

Smolensky, P. A Method for Connectionist Variable Binding. Technical Report CU-CS-356-87, Dept. of Computer Science and Institute of Cognitive Science, Univ. of Colorado, Boulder, CO, 1987.

Smolensky, P. "On the Proper Treatment of Connectionism", *The Behavioral and Brain Sciences,* Vol. 11, No. 1, 1988.

Sumida, R.A., Dyer, M.G. and M. Flowers. Integrating Marker Passing and Connectionism for Handling Conceptual and Structural Ambiguities. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society,* Montreal, Canada, August, 1988.

Touretzky, D. S. and Hinton, G. E. Symbols among the Neurons: Details of a Connectionist Inference Architecture. *Proc. of International Joint Conference on Artificial Intelligence* (IJCAI-85), Los Angeles, CA, 1985.

Touretzky D. S. BoltzCONS: Reconciling Connectionism with the Recursive Nature of Stacks and Trees. *Proceedings of the Eighth Annual Meeting of the Cognitive Science Society.* Amherst, MA. pp. 522-530, 1986.

Touretzky, D. S. Beyond Associative Memory: Connectionists Must Search for Other Cognitive Primitives. *Proc. of AAAI Spring Symposium Series.* Parallel Models of Intelligence: How Can Slow Components Think So Fast? Stanford, CA, 1988.

Vaughan, C. A New View of Vision. *Science News,* Vol. 134, No. 4, pp.58-60, 1988.

Waltz, D. L. and Pollack, J. B. Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. *Cognitive Science,* 9, 51-74, 1985.

Warrington, E. K. and R. A. McCarthy. Categories of Knowledge: Further Fractionations and an Attempted Integration. *Brain*, Vol. 110, pp. 1273-1296, 1987. 4.4.

Wells, H. The effects of transfer in disjunctive concept formation. *Journal of Experimental Psychology*. 65:63-69, 1963.

Zernik, U. *Strategies in Language Acquisitions: Learning Phrases from Examples in Context.* Computer Science Dept. PhD, UCLA 1987.

Zernik, U. and Dyer, M. G. The Self-Extending Phrasal Lexicon, *Computational Linguistics*. Vol. 13, Nos. 3-4, 308-327, 1987.