Chapter 7

Connectionism and Determinism in a Syntactic Parser

STAN C. KWASNY & KANAAN A. FAISAL

The processing of natural language is, at the same time, naturally symbolic and naturally subsymbolic. It is symbolic because ultimately symbols play a critical role. Writing systems, for example, owe their existence to the symbolic nature of language. It is also subsymbolic because of the nature of speech, the fuzziness of concepts, and the high degree of parallelism that is difficult to explain as a purely symbolic phenomenon. Building a processor of natural language, therefore, requires a hybrid approach. This report details a set of experiments which support the claim that natural language can be syntactically processed in a robust manner using a connectionist deterministic parser. The model is trained from patterns derived from a deterministic grammar and tested with grammatical, ungrammatical and lexically ambiguous sentences.

KEYWORDS: Connectionism, determinism, learning, natural language processing, neural networks, parsing.

1. Introduction

Connectionist approaches to natural language processing (NLP), while by most accounts not as successful as symbolic approaches, stand as an important counterpoint. In connectionism, there is the promise of robust decision making, generalization, and other benefits of an extensional style of programming, while symbolic approaches enjoy a history of linguistic study, the application of well-understood methods, and the reassurance that only comes from computing intermediate structures. Clearly there are benefits in both approaches.

Just as clearly there are drawbacks. Symbolic approaches tend to be brittle and intolerant to minor variations. Symbolic systems often contain rules which are difficult to compose or learn symbolically. Subsymbolic NLP is still an exploratory endeavor and generally cuts against conventional wisdom by not supporting hierarchical and

Stan C. Kwasny, Center for Intelligent Computer Systems, Department of Computer Science, Washington University, Campus Box 1045, One Brookings Drive, St Louis, MO 63130, USA; Tel: 314-889-6123 (office); 314-889-6160 (department); Bitnet: sck@wucsl.wustl.edu. The sponsors of the Center are McDonnell Douglas Corporation, Southwestern Bell Telephone Company, and Mitsubishi Electronics America, Inc. Kanaan A. Faisal, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia; Tel: (9663) 860-2175; Bitnet: FACP012@SAUPM00. The second author gratefully acknowledges the support of King Fahd University of Petroleum and Minerals in this research. We thank the reviewers for many insightful comments and suggestions that led to improvements of this paper. The authors also express gratitude to William E. Ball, Steve Cousins, Georg Dorffner, Marios Fourakis, David Harker, Barry Kalman, Dan Kimura, Ron Loui, John Merrill, Gadi Pinkas, Robert Port, and Guillermo Simari for thoughtful discussions and comments concerning this work.

stack-like data structures. These models often impose critical limitations such as limiting the maximum lengths of their input sentences arbitrarily or limiting the iterative nature of the processing.

The processing of natural language is, at the same time, naturally symbolic and naturally subsymbolic. It is symbolic because ultimately symbols play a critical role. Writing systems, for example, owe their existence to the symbolic nature of language. It is also subsymbolic as seen in the nature of speech, the fuzziness of concepts and the high degree of parallelism that is difficult to explain as a purely symbolic phenomenon.

This paper investigates the marriage between connectionism and rule-based deterministic parsing, although the distinction between a subsymbolic component and a symbolic component is maintained. With the correct architecture such a system can retain the benefits of each approach and overcome many of their difficulties.

Any plausible model of language processing should permit alternative linguistic structures to compete while inputs are processed left-to-right. Computer models based on backtracking (e.g. Augmented Transition Networks (ATNs) or Definite Clause Grammars (DCGs)) do not adequately capture the competitive nature of sentence processing. Furthermore, there is no evidence from human experiments that any conscious reprocessing of inputs is routinely performed. The lone exception is perhaps for 'garden path' sentences.

A good example of competition can be found in the TRACE model of speech perception (McClelland & Elman, 1986). In that work, competing interpretations of the pseudo-speech feature vectors are proposed and activation levels rise or fall as each potential interpretation is supported or contradicted. Parsers should permit syntax and other levels of processing to aid in resolving lexical ambiguities just as ambiguous phenomes were resolved in TRACE.

Autonomous syntactic parsing, as Birnbaum (1989) argues, is an activity of questionable value in language processing. A parser which purports to build syntactic structures in isolation from semantic, contextual or other components of the system will have difficulty succeeding completely. This is easily shown by considering examples which are genuinely ambiguous. However, a parser which supports competition in the sense described above is well-positioned to accept influence from these components and cooperate with them in constructing an appropriate structure. Although further work in this direction is necessary, our approach supports competition among structural choices precisely in the cases where such outside influences can and should make the critical difference.

Classically, parsers process inputs iteratively from an unbounded stream of input. Neural network parsers typically do not work iteratively and have limits imposed artificially on the length of the sentence (Fanty, 1985; Selman & Hirst, 1985; Waltz & Pollack, 1985). In classic approaches, natural language processing by computer is performed under the direction of a set of grammar rules. These are often executed as if following instructions in a program. If the intent is to model human sentence processing, then this method is incorrect. Rules should be permitted to play an advisory role only—that is, as descriptions of typical situations and not as prescriptions for precise processing. Control in the application of a rule or variant of a rule should be determined jointly as a data-driven and expectation-driven process.

Symbolic rules are an essential part of most linguistic accounts at virtually all levels of processing, from speech signal to semantics. But systems based literally on rules tend to be brittle since there is no direct way to process linguistic forms that do not strictly adhere to the preconceived rules. If a complete set of rules for all

meaningful English forms existed, then this might be satisfactory. But no such set of rules exists, nor does it seem desirable or even possible to construct such a set. Furthermore, the rules would have a difficult time capturing 'degrees of grammaticalness' (Chomsky, 1965).[1]

Another consequence of a rule-based grammar is that acquisition of new grammar rules often requires tedious re-tuning of existing rules. Rarely can a rule be added to the grammar without it affecting and being affected by other rules in the grammar. To the credit of their creators, some grammars have been continually refined over a period of years, even decades, in an attempt to depict more accurately the processing requirements of English. The only solution to this problem in a practical and realistic manner is through learning.

Chomsky (1965) popularized a fundamental linguistic distinction between competence and performance in language. The field of linguistics has historically centered on the study of competence, while virtually ignoring the more difficult issue of performance except, for example, in certain psycholinguistic studies. Developers of NLP systems have had to deal more with performance issues despite the lack of a solid theoretical foundation for doing so. This has led to some elegant natural language systems which nonetheless contain *ad hoc* components to deal with performance, if they deal with it at all.

The problems this raises for rule-based NLP systems can be attacked through learning. Given sufficient flexibility, a trainable system can be taught syntactic rules and then be led to adapt to those cases where the rules fail. The competence rules of a grammar can be taught in concert with performance examples. While strictly symbolic learning may be possible, such systems typically need to invent new symbols to extend their capabilities and this imposes limitations on the approach. Connectionism holds promise for attacking these problems.

2. Determinism and NLP

The determinism hypothesis which forms the basis for PARSIFAL (Marcus, 1980) imposes important restrictions on natural language processing. It states (p. 11) that:

> Natural Language can be parsed by a mechanism that operates 'strictly deterministically' in that it does not simulate a nondeterministic machine . . .

If we accept this hypothesis, it must follow that NLP need not depend in any fundamental way on backtracking. As a further consequence, no partial structures are produced during parsing which fail to become part of the final structure. PARSIFAL was the first of a number of systems to demonstrate how deterministic parsing of natural language can be performed using a rule-based grammar. Extensions to PARSIFAL have been researched independently including the parsing of ungrammatical sentences in PARAGRAM (Charniak, 1983), the resolution of lexical ambiguities in ROBIE (Milne, 1986), and the acquiring of syntactic rules from examples in LPARSIFAL (Berwick, 1985).

Deterministic, 'wait-and-see' parsers process input sentences primarily left-to-right. Determinism is accomplished by permitting a lookahead of up to three constituents with a constituent buffer designated for that purpose. To permit embedded structures, a stack is also part of the architecture. Rules are partitioned into rule packets which dynamically become active or inactive during parsing, but are usually associated with the current (top-level) node of the structure being built. A single processing step consists of selecting a rule that can fire from an active rule packet,

firing the rule, and performing its action. Conflicts are resolved from the static ordering (priority) of rules within the packet. The action effects changes to the stack and buffer. After a series of processing steps, a termination rule fires and processing is terminated. The final structure is left on top of the stack.

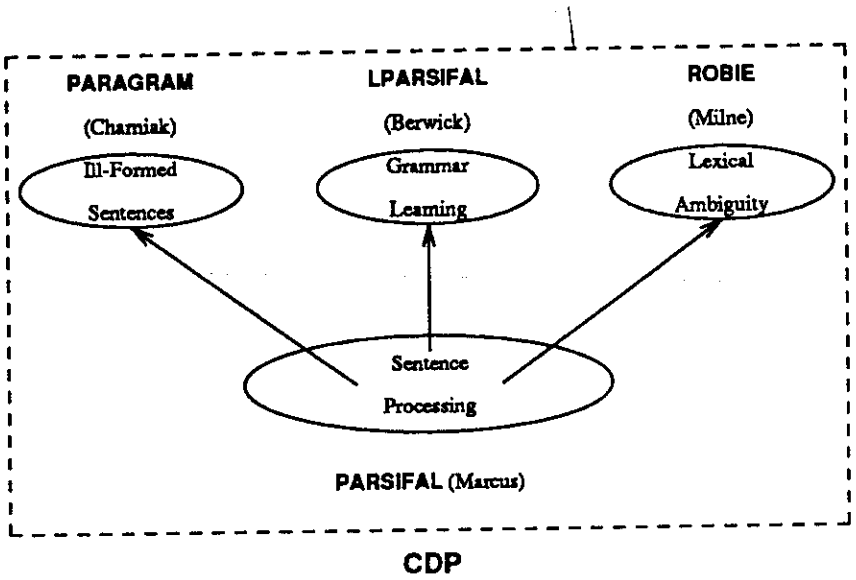

Figure 1. Deterministic parsing systems.

As Figure 1 illustrates, the three extensions to PARSIFAL are all derivatives of deterministic parsing, but represent independent solutions in specific problem areas. The integration of their processing capabilities is one goal of our work (Kwasny *et al.*, 1990). The ultimate goal is to produce a parser that is capable of learning some reasonable facility with language, but does not fail on inputs that are only slightly different from the inputs it is designed to process.

2.1. Connectionist Deterministic Parsing

Exploring the consequences of the determinism hypothesis has led to an architecture for a connectionist deterministic parser called CDP. CDP represents a departure from traditional deterministic parsers by introducing a subsymbolic component. The symbolic component manages the data structures and other components of a traditional parser, while the subsymbolic component is trained with patterns derived from rules of a deterministic grammar to make decisions during parsing. Our results are consistent with the determinism hypothesis in that they provide evidence that natural language interpretation for all but some varieties of 'garden-path' sentences can be deterministically performed with a stack, a buffer for sentence constituents, actions for their manipulation, and a mechanism for deciding which actions to perform. For PARSIFAL (Marcus, 1980), the latter is realized as partitioned packets of rules. In CDP, it is realized as a connectionist neural network.

CDP combines the concepts and ideas from deterministic parsing together with the generalization and robustness of connectionist, adaptive (neural) networks. The combination of these ideas was first suggested by McClelland & Kawamoto (1986, p. 317), but our approach differs somewhat from the case-based approach they advocate. Training sequences are derived from the rules of a deterministic grammar by coding them for use during training.

Parsing experiments are described that permit decision making in the parsing process to be performed by the subsymbolic component. Some simplification over traditional deterministic parsers is realized including the elimination of rule packets and priorities. Furthermore, parsing is performed more robustly and with more tolerance for error. Data are presented which show how a neural network trained with syntactic rules can parse both expected (grammatical) sentences as well as some novel (ungrammatical or lexically ambiguous) sentences. In comparison with symbolic deterministic parsing systems, CDP performs favorably, but its performance depends on the extent and nature of the training. Once trained, the network is efficient, both in terms of representation and execution.

Aside from demonstrating that the symbolic/subsymbolic combination improves NLP, our goal is to duplicate previous results on deterministic parsing and test the generalization capabilities of the system. Our engineering goal is simply to demonstrate that an appropriately designed and trained neural network can capture the same generalities as the rule packets. Another goal is one of economy. Can the effort of grammar design be minimized? Specifically, can traces of processing activity, appropriately coded, serve as training data for grammar learning? Evidence is presented which demonstrates this activity.

Some small modifications to deterministic grammar rules are necessary to ensure the suitability of each rule for use with our 'winner-take-all' network. Many of these changes are simplifications that have been proposed by others and are not essential to the success of our approach. All these changes are made without altering the capabilities represented in the original set of rules. Changes include: elimination of the packet system; removal of attention-shifting rules; removal of rule priorities; reduction of lookahead to two positions instead of three; and revision of the rules so that a single action is performed by each. The reduction to two buffer positions follows the lead of Milne (1986).

Rule **Main-verb**	in packet *parse-vp* priority: 10
IF:	The first element in buffer is a verb
THEN:	DEACTIVATE packet *parse-vp* *if* the active node is a major sentence *then* ACTIVATE packet *ss-final* *else if* the active node is a secondary sentence *then* ACTIVATE *emb-s-final*. CREATE a VP node. ATTACH a VP node to the S. ATTACH the first element in the buffer to the active node as verb. ACTIVATE the clause level packet *cpool* *if* verb is labeled *passive* *then* ACTIVATE the packet *passive* and RUN the grammar rule *passive* next. • •

Rule **Create_VP**	
IF:	current node is S node Attached is AUX node first is a verb
THEN:	CREATE VP node
Rule **Main_verb**	
IF:	current node is VP node Attached is AUX node first is a verb
THEN:	ATTACH as MVB • •

Figure 2. PARSIFAL and CDP rules compared.

As an example, consider part of one sample grammar rule from PARSIFAL and its reformulation in CDP. Figure 2 shows the two styles side by side. Rule actions are in capital letters; rule names are in bold. In the PARSIFAL rule, a priority number is given explicitly and the rule contains multiple actions and conditionals similar to a programming language. It explicitly activates and deactivates rule packets, executes rules, creates new phrase structure nodes, and tests for complex properties of the elements in the buffer.

CDP rules eliminate many of these details without changing the capabilities of the grammar. In the figure, two of several rules derived from the *Main-verb* rule are shown. In the first rule, a new VP active node is created on the stack and in the second rule the verb is attached as a main verb to the active node (VP) on top of the stack.

With the elimination of rule packeting, no priorities nor explicit packet activations/deactivations are required. While this mechanism is precisely what is required for efficient design of a symbolic parser, priorities are at the essence of what is learned when training the subsymbolic component of CDP.

Actions such as creating and attaching or selecting the argument structure of the verb are carried out symbolically in CDP. Also, a symbolic lexicon is consulted to determine the properties of words. When a predicate such as a verb is encountered, the requirements or expectations for its arguments are made part of the features of the active VP node, thus affecting which actions will be executed later on.

2.2. Evolutionary Steps from PARSIFAL

Elimination of the packet system. In PARSIFAL, rules are organized into packets. Only those rules in an active packet are considered while processing. Often, more than one packet is active. For example, the packet CPOOL, or clause level packet, is always active. Since CDP has no packets, every rule is considered in parallel with the situation dictating which action should be taken.

Removal of attention-shifting rules. PARSIFAL relies on attention-shifting rules transparently to build certain constituents, particularly NPs, which begin in the second buffer position. For example, in the sentence taken from Marcus: *Have the students who missed the exam taken the makeup today?*, the subject-aux inversion mechanism (switch) must be deliberately postponed until the NP starting in the second position is analyzed as a complete constituent. Only then can the inversion take place. PARSIFAL solves this problem by temporarily shifting buffer positions so that the parser is viewing the buffer beginning in the second position. The second leftmost complete constituent (the NP) is then reduced before the first element constituent. We follow the lead of Berwick (1985) and others in our treatment of such cases by using the parse stack as a 'movement stack' and stack the postponed item. Two actions, PUSH and DROP, are suitable for this purpose. In the example above, the end of the noun phrase, *the students*, cannot be determined without applying the rules to the embedded clause. When complete, the NP is dropped into the buffer and the auxiliary verb can be re-inserted into the buffer allowing the inversion to take place. Note that at no point is the 'monotonic' property of determinism violated by undoing previous actions.

Removal of rule priorities. In PARSIFAL, rules are ordered by priority. In CDP, rules have no priority. They compete with each other and the most relevant rule, based on training, wins the competition. Only one action, corresponding to the firing of one single-action rule, will be performed on each processing step. The current active node and its attachments along with the contents of the two buffer cells is the basis for this

decision. The rules are coded in such a way that every rule has a unique left-hand side and is thus relevant to situations most similar to its left-hand side pattern.

Restriction of grammar rule format. The format of grammar rules in CDP is different from PARSIFAL in two ways. First, grammar rules are forbidden to have more than a single action which is performed on the first buffer cell only; and second, rule patterns are always defined to test items in both buffer positions. The single action rule modification was first demonstrated by Berwick (1982).

Grammar actions. The repertoire of rule actions is slightly different in CDP. Actions such as ACTIVATE and DEACTIVATE have been removed. The basic actions are:

(a) ATTACH as ⟨node⟩: The first item in the buffer is attached through an intermediate descriptive ⟨node⟩ to the current active node.
(b) CREATE ⟨type⟩: Generates a new node of type ⟨type⟩ and pushes it onto the parse stack as the current active node.
(c) DROP: Pops a node or an item off the top of the stack and inserts it into the buffer in the first buffer position. The previous contents of the buffer is shifted back by one position.
(d) INSERT ⟨item⟩: Inserts the designated item into the buffer in the first buffer position. The previous contents of the buffer is shifted back by one position. In the general form, only a small number of designated lexical items (*you, to, be, wh-marker*) can be inserted. The special form INSERT TRACE inserts an (unbounded) NP trace.
(e) LABEL ⟨feature⟩: Adds designated feature to the first buffer item.
(f) PUSH: Pushes an item onto the stack for temporary storage whenever the parse stack is used as a movement stack.
(g) SWITCH: Exchanges the items in the first and second buffer positions.

These are the only actions the grammar rules can perform. The buffer is managed symbolically and if a position is vacated an item is taken from the input stream to fill the position. The subsymbolic component can only examine the current active node, its immediate attachments and the features of the first two buffer items. Once a node is attached to its parent, it can never again be examined.

3. Architecture of CDP

As Figure 3 illustrates, CDP is organized into a symbolic component and a subsymbolic component. The latter component is implemented as a numeric simulation of an adaptive neural network. The symbolic and numeric components cooperate in a tightly coupled manner since there are proven advantages to this type of organization (Kitzmiller & Kowalik, 1987). For CDP, the advantages are performance and robustness.

The subsymbolic component of CDP is composed of a connectionist network trained using backward propagation (Rumelhart *et al.*, 1986; Werbos, 1974) from rule templates which are derived from the deterministic grammar. Rule templates are intermediate between symbolic rules and the training patterns required by the network. Each rule template is composed from one symbolic rule and typically represents a large number of patterns. They serve to relate situations that occur during parsing with the action deemed appropriate for that situation.

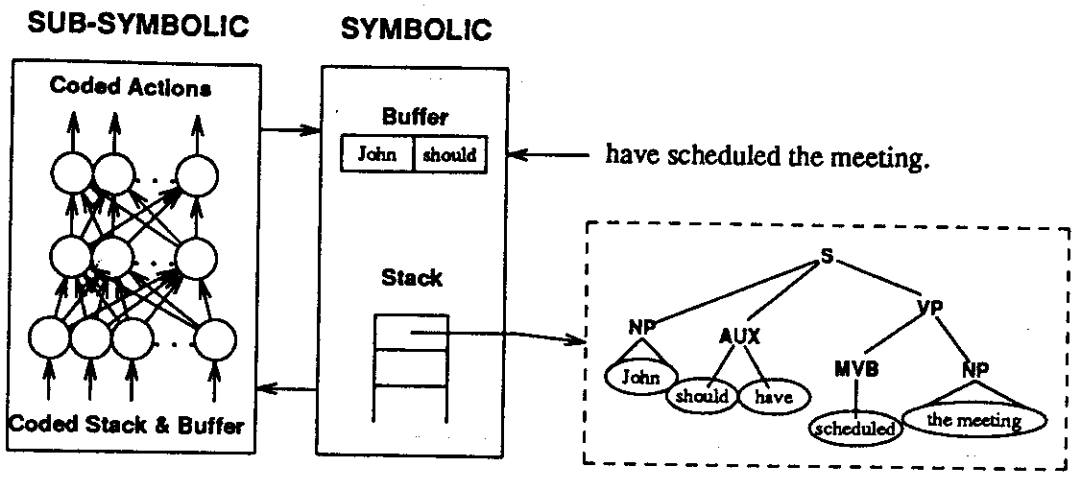

Figure 3. CDP system overview.

The symbolic component manages the input sentence and the flow of constituents into the lookahead buffer, coding them as required for the input level of the network. On the return side, it evaluates the activations of the output units, decides which action to perform, and performs that action, modifying the stack and buffer as required in the process. Actions in CDP are performed symbolically on traditional data structures which are also maintained symbolically. The responsibility of the subsymbolic component, therefore, is to examine the contents of the buffer and stack and yield a preference for a specific action. These preferences are garnered from many iterations of backpropagation learning with instances of the rule templates. Learning itself occurs off-line and is a time-consuming process, but once learned the processing times for the system are excellent. Computations need only flow in one direction in the network. The feedforward multiplication of weights and computation of activation levels for individual units produce the pattern of activation on the output level. Activation of output units is interpreted in a winner-take-all manner, with the highest activated unit determining the action to be taken.

During sentence processing, the network is presented with encodings of the buffer and the top of the stack. What the model actually sees as input is not the raw sentence but a canonical feature-based representation of each word in the sentence in a form that could be produced by a simple lexicon, although such a lexicon is not part of the model in its present form. The network produces the action to be taken which is then performed. If the action creates a vacancy in the buffer and if more of the sentence is left to be processed then the next sentence component is moved into the buffer. The process then repeats until a stop action is performed, usually when the buffer becomes empty. Iteration over the input stream is achieved in this fashion.

Figure 3 illustrates the nature of the processing. When a sentence form like 'John should have scheduled the meeting' appears in the input stream, the first two constituents fill the buffer as shown. These contents along with the contents of the top of the stack and its attachments are encoded and presented to the network. The network, in turn, produces a single action which is then executed symbolically, yielding changes in the buffer and stack. This process repeats until a stop action is performed, at which time the resultant parse structure is left on top of the stack as shown.

Training of CDP proceeds by presenting patterns to the network and teaching it to respond with an appropriate action. The input patterns represent encodings of the buffer positions and the top of the stack from the deterministic parser. The output of the network contains a series of units representing actions to be performed during

processing and judged in a winner-take-all fashion. Network convergence is observed once the network can achieve a perfect score on the training patterns themselves and the error measure has decreased to an acceptable level (set as a parameter). Once the network is trained, the weights are stored in a file so that sentences can be parsed. A sentence is parsed by iteratively presenting the network with coded inputs and performing the action specified by the network.

There are two distinct approaches to training a network to parse sentences: deductive and inductive. Each of these training strategies results in a slightly different version of CDP. The difference lies in the nature of the training patterns presented. One approach uses rule templates, training patterns derived from the rules. This type of learning is deductive in the sense that a general form of each rule is learned from which the parser must derive actions specific to individual cases. The second approach uses training data derived from sentence processing traces. This form of training is inductive in the sense that the parser must arrive at general patterns of performance from the specific instances presented.

For deductive training, each constituent is replacing with its coding and a rule template is created. Each grammar rule is coded as a training template which is a list of feature values. In general, each constituent is represented by an ordered feature vector in which one or more values is ON(+1) for features of the form and all other values are either OFF(−1) or DO NOT CARE (?). A rule template is instantiated by randomly changing ? to +1 or −1. The probability of a ? becoming a +1 or −1 is equal and set at 0.5. Thus, each template can be instantiated to give many training patterns and each training epoch is slightly different. It is obviously impossible to test the performance of all these cases, so for the purposes of judging convergence, a zero is substituted for each ? in the rule template to provide testing patterns.

A second type of training for the network uses training patterns derived from traces of the situations encountered and actions performed during the processing of actual sentences. Inductive learning begins with training data derived as 'sentence traces' of deterministic parsing steps. Training proceeds by presenting patterns of these steps to the network and teaching it to respond with an appropriate action. This processing is guided by application of the rules of a deterministic grammar as before.

PARSIFAL is simulated in this way and the task of the inductively-trained CDP parallels that of LPARSIFAL. In LPARSIFAL the object is to learn (symbolic) grammar rules from examples of correct sentences. The success of this task is gauged by directly comparing the rules learned to those of PARSIFAL. In CDP inductive training requires the network to exhibit the correct rule-following behavior after being trained with a sample of sentence traces. Training occurs through the mechanism of backward propagation. No symbolic rules are learned as such, but the behavioral characteristics of the rules are captured within the parameters of the network.

CDP, therefore, can exhibit different properties depending on the patterns used in training. Inductive learning takes longer than deductive learning because more data is required. Also, the range of sentence types handled depends greatly on the completeness of the examples presented. Deductive training imposes an ordering on the training patterns that assures a completeness which is difficult to achieve with inductive training, but inductive training patterns reflect the frequency of rule occurrences seen in actual sentence processing. For more discussion of the training process, see Faisal & Kwasny (1990).

4. Experimentation

Experiments are conducted to determine the effectiveness of training and to investi-

gate whether the connectionist network generalizes properly to ungrammatical and lexically ambiguous cases. In comparison with other deterministic parsing systems, CDP outperforms any single system known. Our experimentation shows results of tests performed with examples drawn from PARSIFAL, PARAGRAM and ROBIE. Much of the performance depends on the extent and nature of the training, of course, but our results show that through proper training a connectionist network can indeed exhibit the same behavioral effect as the rules, but with the advantages of extensional programming.

For each sentence processed, the 'average strength' of the responses made by the network is computed. Average strength for each sentence is computed as the reciprocal of the average error per processing step. On each step, the winning output unit is determined and the error is computed as the Euclidean distance between the actual output vector and an idealized output vector (the corner of the hypercube). The reciprocal of the error is the strength of the step. These errors are summed and averaged to give the average error per processing step. The reciprocal of this average gives the average strength as shown. Average strength reflects the certainty with which individual actions for building structures are being selected. Although there is no real meaning in the values of these numbers, they are a useful means of comparison. They also indicate to what extent other responses are competing with the winner.

Deductive training generally performs well on all generalization tasks and outperforms inductive training by scoring higher on all experiments. Reasons for this include the specificity of the inductive training data as well as the lack of a large amount of training data in the inductive case required to provide sufficient variety.

4.1. Target Grammars

CDP as a system permits experimentation with different grammars which require different networks representing different numbers of units, weights and actions. Our experimentation has examined three different target grammars, which we shall call small, medium and large. Figure 4 shows some of the characteristics of these grammars for comparison. Each grammar contains a set of rules. The number of rules reflects the capability of that grammar and the amount of training necessary to achieve conver-

	Target Grammars		
	Small	Medium	Large
Number of Rules	13	22	73
Number of Actions	5	20	40
Network Size (units)	44-15-5	35-20-20	66-40-40
Network Size (weights)	735	1100	4240
Presentations (× 1000)	200	500	1000
Based On	Example (Winston, 1984)	Appendix C (Marcus, 1980)	Appendix D (Marcus, 1980)

Figure 4. Summary of target grammars used in CDP.

gence so that the network can correctly perform as the rules. The network configurations reflect the increase from five to 20 to 40 actions reflecting the increase in complexity across the three grammars. A variety of training runs have been made with each grammar. Shown is the number of presentations of training patterns made to provide the results discussed.

4.1.1. The small grammar. In our initial attempt to demonstrate the feasibility of our approach (Kwasny, 1988), a small, simple grammar is used. This grammar is based on an example from the Winston text (1984) and contains 13 rules with five actions. The network requires 44 input units to encode the stack and three-place buffer. This comparatively large number of input units reflects our decision at that point in the project to stay as true as possible to PARSIFAL. Three positions of the buffer are encoded with each position allocated an identical number of units. Our choice of 15 hidden units is determined empirically.

The grammar can produce parse structures containing S, NP, VP and PP nodes. A preprocessing step for noun phrases is assumed, so that NPs have structure before entering the buffer. Coding of three rule packets (S, VP and PP) as rule templates provides training data for the 44–15–5 unit network. Perfect performance, as determined by presentation of a limited number of test sentences, is achieved for the 13 grammar rules coded. With such a limited grammar there are not many different sentence patterns to be tested and therefore no generalization experiments of interest can be reported.

With the small target grammar an attempt is made to keep the coding scheme in line with the rule version, but at the expense of a sparsely coded network and one that consequently takes many training cycles even though what is being learned is somewhat simple. The rule partitioning of the PARSIFAL rules is realized directly in the coding of the data and therefore no hand partitioning is required. Attachments to nodes on the stack are also part of the coding.

Actions are coded as output to be learned by the network. In this example, possible actions are Attach (first item), Create VP, Create PP, Drop and Stop. The task to be learned by the network can be viewed as a simple classification task in which configurations occurring during parsing are classified according to the action needed to continue processing.

4.1.2. The medium grammar. Even though the medium grammar has just under twice the number of rules as the small grammar, it is much more sophisticated than the small one. Appendix C of Marcus (1980) serves as the model for the rules of this grammar. In Appendix C, Marcus collects the set of rules used to illustrate the mechanisms of deterministic parsing in his thesis and thus all the basic mechanisms of deterministic parsing are demonstrated in this grammar. Successful performance with this grammar can be taken as a demonstration of successful realization of the very enterprise of connectionist deterministic parsing. Following the lead of Milne (1986), the buffer size in our version of the grammar has been reduced from three to two constituents.

The medium grammar is capable of processing a variety of simple sentence forms such as simple declarative sentences, simple passive, and imperative sentences as well as yes–no questions. It permits reasonable generalization experiments to be conducted. For testing and comparison purposes several sentences are coded that would parse correctly by the rules of the deterministic parser. Also, several mildly ungrammatical and lexically ambiguous sentences are coded to determine whether the network would

generalize in any useful way. The objective is to test whether syntactic context could aid in resolving such problems.

In the set of experiments conducted with this grammar some of the differences between deductive and inductive training are illustrated. For deductive training, the training templates are derived from the deterministic grammar as before, and for inductive training a small set of positive sentence examples were traced which resulted in 64 unique training patterns (see Faisal & Kwasny, 1990; Kwasny & Faisal, 1989a).

4.1.3. The large grammar. In a third set of experiments, a much larger and more general grammar, based loosely on Appendix D in Marcus (1980), is used. In this case, the grammar consists of 73 rules and represents rules for parsing sentence forms such as simple declarative sentences, passives, imperatives, yes–no questions, wh-questions, wh-clauses and other embedded sentences. The grammar to be learned by the subsymbolic system can be separated into base phrase structure rules and transformational-type rules. The base structure system can be further broken down into rules for NPs, VPs, auxiliaries, main sentence, PPs and embedded sentences. Transformational rules fall into two groups: simple local transformations (e.g. subject-aux inversion) and major movement rules (e.g. *wh* movement). In general, for each type of phrase, creation of the phrase (creating a new node on the active node stack) and completion of the phrase (dropping it into the buffer) is carried out by a separate grammar rule action.

The rules for analyzing verb phrases discriminate among verbs that take different kinds of complements. For example, verbs that take a *wh* complement are discriminated from ones that take a *that* complement. Verbs like *want* that take either a missing or lexical subject in embedded sentential complements are separated from verbs like *try* or *believe* that do not take a lexical subject. Verbs that take one NP object are distinguished from ones that take two NP objects through lexical features.

As with the medium-sized grammar, several sentences are coded that would parse correctly by the rules of the deterministic parser. Additionally, several mildly ungrammatical and lexical ambiguous sentences are coded to determine how the network trained on this larger grammar generalizes. All these examples are derived from those presented in PARAGRAM and ROBIE. Our objective is to determine whether the same generalization properties of the medium grammar would hold when scaled up to a much larger and more realistic set of grammar rules. The large grammar is a major focus of a recent DSc thesis (Faisal, 1990).

4.2. Parsing Grammatical Sentences

Experimentation with grammatical sentences demonstrates the ability of CDP to process sentences exactly as the rules would. Earlier we mentioned that testing for the purpose of establishing convergence is possible from the rule templates by changing the ? to a zero value. The Small grammar converges in this sense and can parse sentences of the form

NP_a moved NP_b to NP_c

With the two larger grammars more extensive testing is performed with actual sentences. the performance of CDP is first examined with grammatical sentences. These are, by our definition, those sentence forms which parse correctly in the rule-based grammar from which we derived the training set. Tables I and II show several

Table I. Grammatical sentences used in testing the medium grammar

Sentence form	Deductive avg. strength	Inductive avg. strength
(1) John should have schedule the meeting.	283.3	84.7
(2) John has schedule the meeting for Monday.	179.3	84.2
(3) Has John schedule the meeting?	132.2	64.4
(4) John is scheduling the meeting.	294.4	83.5
(5) The boy did hit Jack.	298.2	76.2
(6) Schedule the meeting.	236.2	67.8
(7) Mary is kissed.	276.1	84.9
(8) Tom hit(v) Mary.	485.0	80.3
(9) Tom will(aux) hit(v) Mary.	547.5	78.7
(10) They can(v) fish(np).	485.0	80.3
(11) They can(aux) fish(v).	598.2	76.8

Table II. Grammatical sentences used in testing the large grammar

Sentence form	Deductive avg. strength
(12) John should have schedule the meeting for Monday.	56.9
(13) Scheduled a meeting for Monday.	29.4
(14) Has John schedule the meeting for Monday?	36.8
(15) John has scheduled the meeting for Monday.	55.1
(16) The meeting has been scheduled for Monday.	565.5
(17) The meeting seems to have been scheduled for Monday.	70.8
(18) The jar seems broken.	5.3
(19) I persuaded John to do it.	39.7
(20) I saw him do it.	38.2
(21) Mary wants John to have a party.	46.5
(22) Mary wants to have a party.	57.9
(23) What will the man put in the corner?	376.2
(24) What will the man put the book in?	23.7
(25) Who did John see?	427.3
(26) Who broke the jar?	38.3
(27) Who is carrying the baby?	61.0
(28) What is the baby carrying?	11.5
(29) What did Bob give Mary?	32.1
(30) The man who wanted to meet Mary has disappeared.	33.0
(31) The man who hit Mary with a book has disappeared.	29.2
(32) The man whom Mary hit with a book has disappeared.	254.5
(33) I told that boy that boys should do it.	19.9
(34) That mouse that the cat chased had squeaked.	8.8
(35) I told Sue you would schedule the meeting.	4.3
(36) I told the girl that you would schedule the meeting.	5.8
(37) John is scheduling the meeting for Monday.	54.7
(38) The boy did hit Jack.	137.7
(39) Tom hit(v) Mary.	29.5
(40) Tom will(aux) hit(v) Mary.	125.8
(41) The will(noun) gave the money to Mary.	61.9
(42) They can(v) fish(np).	30.0
(43) They can(aux) fish(v).	6.3

examples of grammatical sentences which are parsed successfully in both the medium and the large grammars along with their average strengths.

Each example shows a relatively high average strength value, indicating that the training data has been learned well. The medium grammar shows data for the two forms of training, deductive and inductive. The deductive average strength value is higher, in almost all cases, than the corresponding inductive average strength. Although comparisons are difficult to make due to variations in the number of unique training patterns and other factors, the deductively-trained network exhibits uniformly more definitive decisions than the inductively-trained network.

Many of these sentences are used in testing the symbolic deterministic parsing systems described earlier. Parse trees are developed which are identical with ones produced by those systems. Sentences (8)–(11) and (39)–(43), which contain ambiguous words, are presented unambiguously with lexical choices shown in parentheses.

Experimentation with grammatical sentences confirms that indeed the rules from these grammars have been learned sufficiently to parse sentences as with the small grammar. However, capabilities described thus far only duplicate what can be done rather comfortably using a symbolic approach. What other features does the model possess? Importantly, how robust is the processing?

4.3. Parsing Ungrammatical Sentences

PARAGRAM extends PARSIFAL to handle ungrammatical sentences. In PARAGRAM, this is accomplished by considering all rules in parallel and scoring each test performed on the left-hand side of a rule according to predefined weights. The rule with the best score fires. In this way, processing will always have some rule to fire. Reported experimentation with PARAGRAM shows this to be an effective method of stretching the inherent capabilities of the grammar.

For CDP, an important test of its generalization capabilities is its response to novel sentences. This is strictly dependent upon its training experiences since in deductive training no relaxation rules (Kwasny & Sondheimer, 1981), meta-rules (Weischedel & Sondheimer, 1983), or other special mechanisms are added to any of the sets of rules to handle ungrammatical cases. Likewise, in inductive training no ungrammatical sentences are used. Experiments consist of testing a few ungrammatical, but meaningful sentences that are close to the training data and within the scope of our encoding for each grammar.

Selected ungrammatical sentences are properly processed as a direct result of the generalization properties of learning. When presented an ungrammatical sentence, i.e. one for which the rules of the grammar would fail to find a parse, the network automatically relates the situations arising during parsing to similar situations on which it has been trained. The tendency is for it to select the closest situation. Very often this generates precisely the response required to relate the deviant form to one that is not.

Table III. Ungrammatical sentences used in testing the medium grammar

Sentence form	Deductive avg. strength	Inductive avg. strength
(44) *John have should schedule the meeting.	25.1	6.6
(45) *Has John schedule the meeting?	38.1	18.2
(46) *John is schedule the meeting.	4.7	4.9†
(47) *The boy did hitting Jack.	26.6	7.5‡

In Table III a few ungrammatical sentences are shown that were tested with the medium grammar. These examples have produced reasonable structures in our judgment when presented to our system. They are shown in the table with their response strengths. Note that overall average strength is lower for ungrammatical sentences when compared to similar grammatical ones.

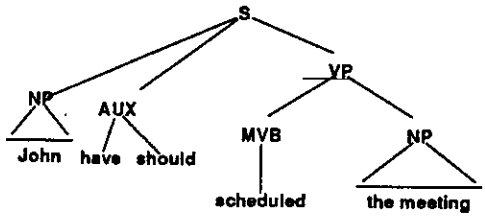

Figure 5. Parse of '*John have should schedule the meeting'.

In sentence (44), for example, the structure produced is identical to that produced while parsing sentence (1), but with lower strength in the inductive case. The only difference is that the two auxiliary verbs, *have* and *should*, are reversed as shown in Figure 5. Sentence (45) contains a disagreement between the auxiliary *has* and the main verb *schedule* and yet the comparable grammatical sentence (3) parsed identically in both approaches, but with lower strength again in the inductive approach.

Sentence (46) can be compared with sentence (4). In the deductive case, a structure similar to that built for sentence (4) is indeed constructed. However, in the inductive case (marked with †), the network attempts to process 'is' as if it were indicating the passive tense. Although this is incorrect for this sentence, it is not an unreasonable choice given the particular training data used. Sentence (47) can be compared with sentence (5), but there is not one clear choice in how the sentence should appear if grammatical. The deductive-trained network processes sentence (47) as sentence (5), while the inductive result (marked with ‡) shows the sentence processed as if it were progressive tense ('The boy is hitting Jack'). In PARAGRAM, a nonsensical parse structure is produced for sentence (47), as reported by Charniak (1983, p. 137). The problems associated with using a syntax-based approach to handling ungrammatical sentences are well-known (see, for example, Kwasny, 1980).

To demonstrate generalization after scaling up, CDP is trained with the large grammar and again tested with several examples of ungrammatical sentences. As with the medium grammar, no special mechanisms were added to handle ungrammatical cases. In Table IV ungrammatical sentences used in testing are shown along with their average strengths. As before, these examples produce reasonable structures when presented to our system and overall average strength is lower for ungrammatical sentences when compared to similar grammatical ones.

In sentence (48), for example, the structure produced is identical to that produced while parsing sentence (12). The only difference is that the two auxiliary verbs, *have* and *should*, are reversed in the parse tree. Such scrambling of words is beyond the capabilities of PARAGRAM as pointed out by Charniak (1983, p. 138). Sentence (49) contains a disagreement between the auxiliary *has* and the main verb *schedule*, and but parsed identical to sentence (14). Sentences (50) and (51) parse comparable to sentence (37). Sentence (51), in fact, demonstrates how the presence of extra words does not deter CDP as it did PARAGRAM (Charniak, 1983, p. 138). Another example from PARAGRAM is sentence (52) which is processed as if it were progressive tense

Table IV. Ungrammatical sentences tested with large target grammar

Sentence form	Avg. strength
(48) *John have should scheduled the meeting for Monday.	14.4
(49) *Has John schedule the meeting for Monday?	32.3
(50) *John is schedule the meeting for Monday.	9.5
(51) *John is is scheduling the meeting for Monday.	7.2
(52) *The boy did hitting Jack.	14.8
(53) *The meeting is been scheduled for Monday.	559.6

('The boy is hitting Jack'). When this sentence is presented to PARAGRAM, a nonsensical parse structure is produced for this sentence as reported by Charniak (1983, p. 137). In CDP it produced a structure like that of sentence (38), but, as we saw in the medium grammar, there is not one clear choice for how the sentence should appear if grammatical. Finally, sentence (53) produced a very strong response, but the comparable grammatical sentence (16) produces an even stronger response. Sentences like (53) are commonly misspoken forms of English.

4.4: Processing Lexically Ambiguous Sentences

ROBIE extends PARSIFAL to address issues of lexical ambiguity. ROBIE requires additional rules and number agreement tests to handle these items properly. In the deterministic approach, it is essential that lexical items be properly disambiguated to permit processing to proceed without backtracking.

In another set of experiments with the medium grammar, the parser is tested for this. Normal sentences are presented, except that selected words are coded ambiguously (here, indicated by angle brackets ⟨ ⟩ around the word) to represent an ambiguously stored word from the lexicon. Some of these sentences are shown in Table V. The numbers again indicate the average strength for each sentence. In the cases shown, the lexically ambiguous words are correctly interpreted and reasonable structures result.

Table V. Lexically ambiguous sentences used in testing the medium grammar

Sentence form	Deductive avg. strength	Inductive avg. strength
(54) They ⟨can⟩ fish.	4.5	2.6
(55) They can ⟨fish⟩.	172.2	4.9
(56) ⟨Will⟩ he go?	83.6	14.3
(57) Tom ⟨will⟩ hit Mary.	118.7	19.9
(58) Tom ⟨hit⟩Mary.	39.0	2.5

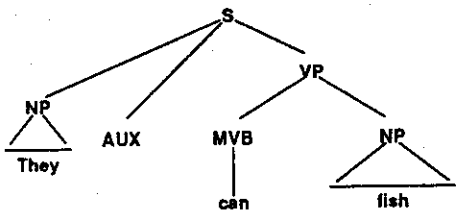

Figure 6. Parse of 'They can(v) fish(np)'.

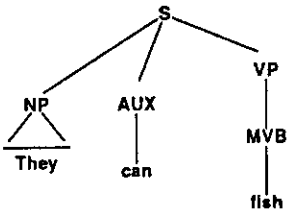

Figure 7. Parse of 'They can(aux) fish(v)'.

CDP utilizes syntactic context to resolve these ambiguities. Again, the generalization capability of the network automatically works to relate novel situations to its training cases. For lexically ambiguous situations, some inputs may contain features which confuse its identity as expected by the parser. The context provided by the buffer and stack of the deterministic parser works well in aiding the resolution of many types of lexical ambiguities. An important fact is that, as before, no additional rules or mechanisms are required to provide this capability.

For example, sentence (54) presents *can* ambiguously as an auxiliary, modal and main verb, while *fish* is presented uniquely as an NP. *Can* is processed as the main verb of the sentence and resulted in the same structure as sentence (10) of Table I. This is shown in Figure 6. Here, each word is presented unambiguously with *can* coded as a verb and *fish* coded as an NP. The same structure results in each case, with the average strength level much higher in the unambiguous case. Likewise, sentence (55), by coding *fish* ambiguously as a verb/NP and coding *can* uniquely as an auxiliary verb, produced the same structure as sentence (11). This is the structure in Figure 7.

Sentence (56) contains the word *will* coded ambiguously as an NP and an auxiliary, modal verb. In the context of the sentence, it is clearly being used as a modal auxiliary and the parser treats it that way. A similar result is obtained for sentence (57). In sentence (58), *hit* is coded to be ambiguous between an NP (as in playing cards) and a verb. The network correctly identifies it as the main verb of the sentence.

In the cases shown, the lexically ambiguous words are disambiguated and reasonable structures result. Note that the overall average strengths are lower than comparable grammatical sentences discussed, as expected. Also, deductive average strength is higher than inductive average strength.

Additional sentences containing lexically ambiguous words are tested with the large grammar. Some of these sentences are shown in Table VI. Ambiguously coded words within the context of otherwise normal sentences are shown to be interpreted correctly in this set of experiments. In the cases shown, the lexically ambiguous words are correctly interpreted and reasonable structures result, although lower strengths are

Table VI. Lexically ambiguous sentences tested with large target grammar

Sentence form	Avg. strength
(59) ⟨Will⟩ John schedule the meeting for Monday?	5.0
(60) Tom ⟨will⟩ hit Mary.	29.8
(61) Tom ⟨hit⟩ mary.	13.6
(62) The ⟨will⟩ gave the money to Mary.	16.6
(63) They ⟨can⟩ fish(np).	20.6
(64) They can(aux) ⟨fish⟩.	2.9

again observed. As before, no additional rules or mechanisms are required to provide this capability.

Sentence (59) contains the word *will* coded ambiguously as an NP and an auxiliary, modal verb. In the context of the sentence, it is clearly being used as a modal auxiliary and the parser treats it that way. A similar result is obtained for sentence (60) which parses as (40). In sentence (61), *hit* is coded to be ambiguous between an NP (as in playing cards) and a verb. The network correctly identifies it as the main verb of the sentence as in sentence (39). Sentence (62) is constructed as for sentence (41). Sentence (63) presents *can* ambiguously as an auxiliary, modal and main verb, while *fish* is presented uniquely as an NP. *Can* is processed as the main verb of the sentence and results in the same structure as sentence (42). Likewise, sentence (64), which contains *fish* coded ambiguously as a verb/NP and *can* coded uniquely as an auxiliary verb, produces the same structure as sentence (43). In the cases shown, the results are very comparable to those of the medium grammar.

One final type of lexical ambiguity should be mentioned. In CDP, the word *to* is always coded ambiguously as both a preposition (prep) and as the word that introduces an infinitive (to-inf). For the word *that*, coding always represents it as a complementizer (comp), but often it should play the role of a determiner (det). Through training, CDP is able properly to resolve the roles of these words according to context. Examples showing how *to* is properly handled are sentences like (22) and (41), while ones illustrating disambiguation of *that* are (33) and (34).

5. Discussion

With our architecture for connectionist deterministic parsing, the functionality of symbolic deterministic parsing is demonstrated. Training becomes difficult as the complexity of the grammar increases, but our data show that many of the advantages of the connectionist approach can be realized by modifying a deterministic parser as we have described. Likewise, many of the drawbacks of previous connectionist attempts at NLP can be overcome, while addressing some of the deficiencies of symbolic models. Thus, the marriage is a good one from our perspective.

While deductive training exhibits better performance than inductive training for all experiments, there are tradeoffs in the two approaches. Deductive training requires rules as the basis for rule templates while inductive training requires a large amount of data to be successful. Fortunately there is a middle ground which allows mixtures of the two training strategies. Training can be performed using rule templates as well as patterns based on sentence traces. In a recent set of experiments in which the two

types of training data are combined, the network is capable of generalizing in ways similar to deductive learning, but also shows particularly good performance on the specific cases reflected in the inductive data. This is further discussed in Faisal & Kwasny (1990).

The potential for exploiting the notion of extensional programming is also good. The dependency on inductive learning will be greatest in those domains which are difficult to analyze. These are exactly the domains in which symbolic techniques break down. We have begun to examine an approach to NLP based almost exclusively on inductive training sequences (Kwasny & Faisal, 1989b). This looks promising, but is a topic for further research.

6. Open Problems

Several open problems exist which need to be resolved before satisfactory NLP is possible with neural networks. The most important of these relates to the issue of overcoming the 'finiteness' constraint on inputs. Methods need to be developed which permit unbounded streams of input to be processed by a neural network. Our approach is to apply the network iteratively in a manner identical to the rules in deterministic parsing. The iteration is managed symbolically and external to the network itself. This makes our solution far from completely satisfying. Furthermore, the action portion of the rule is identical to that of symbolic deterministic parsers. Ideally, the actions should be incorporated into what is learned in the network. Requiring an action as output introduces a point of fragility just as symbolic systems lack robustness due to the fragility of their individual symbols.

Finally, methods of representation need to be developed for unbounded structures which correspond to the resultant structures of language processing. Pollack (1989) is a step in that direction, but more work is required. Some ideas for how a more fully connectionist parser based on deterministic parsing might be realized are contained in Kwasny & Faisal (1989b). Language processing, it is suggested, may turn out to be best viewed in its relation to more behaviorally-defined outputs. However, it is easy to show that what is commonly viewed as syntax is undoubtedly an important component of language processing and, therefore, should be investigated.

7. Summary

CDP implements a deterministic parser based on the rules from a deterministic grammar. The result is a combined symbolic/subsymbolic system which exhibits characteristics from several well-known extensions to the basic deterministic parser. These extended properties come essentially for free due to the use of connectionism. The grammar used in these experiments is derived, with minor modifications, from one used by Marcus, but with much inspiration from Milne, Berwick and Charniak. Only small modifications are required in the grammar to accommodate our particular architecture in CDP.

Notes

1. There are several expressions of this idea in the literature. Several psycholinguistic studies attempt to measure the reality of this notion, both from a use as well as an interpretation perspective. Chomsky is selected as an important reference and one that illustrates a classic viewpoint.

References

Berwick, R.C. (1982) *Locality Principles and the Acquisition of Syntactic Knowledge.* PhD thesis. Cambridge, MA: Massachusetts Institute of Technology.

Berwick, R.C. (1985) *The Acquisition of Syntactic Knowledge.* Cambridge, MA: MIT Press.

Birnbaum, L. (1989) *A Critical Look at the Foundations of Autonomous Syntactic Analysis. Proceedings of the 11th Annual Conference of Cognitive Science Society,* 99–106. Hillsdale, NJ: Erlbaum.

Charniak, E. (1983) A parser with something for everyone. In: King (Ed.) *Parsing Natural Language.* New York: Academic Press.

Chomsky, N. (1965) *Aspects of the Theory of Syntax.* Cambridge, MA: MIT Press.

Faisal, K.A. (1990) CDP: connectionist deterministic parser. DsC thesis, May 1990. St Louis, MO: Department of Computer Science, Washington University.

Faisal, K.A. & Kwasny, S.C. (1990) *Deductive and Inductive Learning in a Connectionist Deterministic Parser.* The International Joint Conference on Neural Networks. Washington, DC, January.

Fanty, M. (1985) Context-free parsing in connectionist networks. Technical Report 174. Rochester, NY: Computer Science Department, University of Rochester.

Kitzmiller, C.T. & Kowalik, J.S. (1987) Coupling symbolic and numeric computing in knowledge-based systems. *AI Magazine,* 8, 2, 85–90.

Kwasny, S.C. (1980) Treatment of ungrammatical and extra-grammatical phenomena in natural language understanding systems. Bloomington, IN: Indiana University Linguistics Club, November.

Kwasny, S.C. (1988) A PDP approach to deterministic natural language parsing. *Neural Networks,* 1, Supplement 1, 305.

Kwasny, S.C. & Faisal, K.A. (1989a) *Competition and Learning in a Connectionist Deterministic Parser.* 11th Annual Conference of Cognitive Science Society. Ann Arbor, MI, August.

Kwasny, S.C. & Faisal, K.A. (1989b) Connectionism and determinism in a rule-based natural language system. Technical Report WUCS-89-31. St Louis, MO: Department of Computer Science, Washington University.

Kwasny, S.C., Faisal, K.A. & Ball, W.E. (1990) Unifying several natural language systems in a connectionist deterministic parser. In: W. Webster & R. Uttamsingh (Eds) *AI and Simulation: Theory and Applications, Simulation Series,* Vol. 22, pp. 28–33 (San Diego, CA: Society for Computer Simulation).

Kwasny, S.C. & Sondheimer, N.K. (1981) Relaxation techniques for parsing ill-formed input. *American Journal of Computational Linguistics,* 7, 99–108.

McClelland, J.L. & Elman, J.L. (1986) The TRACE model of speech perception. *Cognitive Psychology,* 18, 1–86.

McClelland, J.L. & Kawamoto, A.H. (1986) Mechanisms of sentence processing: assigning roles to constituents of sentences. In: D. E. Rumelhart & J. L. McClelland (Eds) *Parallel Distributed Processing,* 272–325. Cambridge, MA: MIT Press.

Marcus, M.P. (1980) *A Theory of Syntactic Recognition for Natural Language.* Cambridge, MA: MIT Press.

Milne, R. (1986) Resolving lexical ambiguity in a deterministic parser. *Computational Linguistics,* 12, 1–12.

Pollack, J.B. (1989) Recursive distributed representations, Report 89-JP-RECURSIVE, Laboratory for Artificial Intelligence Research, Columbus, OH: Ohio State University, August.

Rumelhart, D.E., Hinton, G. & Williams, R.J. (1986) Learning internal representations by error propagation. In: D. E. Rumelhart & J. L. McClelland (Eds) *Parallel Distributed Processing,* 318–362. Cambridge, MA: MIT Press.

Selman, B. & Hirst, G. (1985) A rule-based connectionist parsing system. *Proceedings of the 7th Annual Conference of the Cognitive Science Society,* 212–221. Irvine, CA.

Waltz, D.L. & Pollack, J.B. (1985) Massively parallel parsing: a strongly interactive model of natural language interpretation. *Cognitive Science,* 9, 51–74.

Weischedel, R.M. & Sondheimer, N.K. (1983) Meta-rules as a basis for processing ill-formed input. *American Journal of Computational Linguistics,* 9, 161–177.

Werbos, P. (1974) *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Science.* PhD thesis, Harvard University, Cambridge, MA.

Winston, P.H. (1984) *Artificial Intelligence.* Reading, MA: Addison-Wesley.