

Design of a Hybrid Deterministic Parser

Kanaan A. Faisal

Information and Computer Science Department,
King Fahd University of Petroleum and Minerals[†]
Dhahran 31261
Kingdom of Saudi Arabia

Stan C. Kwasny

Center for Intelligent Computer Systems[‡]
Department of Computer Science
Washington University
St. Louis, MO 63130-4899
U.S.A.

1. Introduction

A deterministic parser is under development which represents a departure from traditional deterministic parsers in that it combines both symbolic and connectionist components. The connectionist component is trained either from patterns derived from the rules of a deterministic grammar. The development and evolution of such a hybrid architecture has led to a parser which is superior to any known deterministic parser.

Experiments are described and powerful training techniques are demonstrated that permit decision-making by the connectionist component in the parsing process. This approach has permitted some simplifications to the rules of other deterministic parsers, 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 connectionist (neural) network trained with linguistic rules can parse both expected (grammatical) sentences as well as some novel (ungrammatical or lexically ambiguous) sentences.

2. Determinism and Parsing

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 processing 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).

[†] The first author gratefully acknowledge the support of King Fahd University of Petroleum and Minerals.

Traditional deterministic 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 within packets from the static ordering (priority) of rules. The action effects changes to the stack and buffer. After a series of processing steps, a termination rule fires and processing ends. The final structure is left on top of the stack.

3. Hybrid Deterministic Parsing

Our parser takes the approach of deterministic parsing and combines it with connectionism. McClelland and Kawamoto (1986, p.317) first suggested the combination of these ideas. Deterministic parsing provides a setting in which no backtracking occurs while connectionism provides generalization and robustness. Our goal is to combine the two in a way that enhances their advantages and minimizes their faults. In simple terms, the rules of the deterministic parser are replaced by a network which is trained from training sequences derived from the grammar rules. The network embodies the decision-making component of the parser and maps a state of the parser to an action. Actions are performed in the traditional way by symbolically manipulating the stack and buffer contents.

Parsing experiments are conducted to determine the effectiveness of training by attempting to process ungrammatical and lexically ambiguous sentence forms. The performance of our hybrid parser depends on the extent and nature of the training. Once trained, the network is efficient, both in terms of representation and execution.

3.1. Design

Some small modifications to deterministic grammar rules are necessary to insure the suitability of each rule for use with our "winner-take-all" network. Many of these changes are simplifications that have been

[‡] The sponsors of the Center are McDonnell Douglas Corporation and Southwestern Bell Telephone Company.

<p>Rule Main-verb in packet parse-vp priority: 10</p> <p>IF: The first element in buffer is a verb</p> <p>THEN:</p> <p>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.</p> <p>•</p> <p>•</p>	<p>Rule Create_VP</p> <p>IF: current node is S node Attached is AUX node first is a verb</p> <p>THEN: CREATE VP node</p> <p>Rule Main_verb</p> <p>IF: current node is VP node Attached is AUX node first is a verb</p> <p>THEN: ATTACH as MVB</p>
--	---

Figure 1: PARSIFAL and Hybrid Parser Rules Compared

proposed by others and are not essential to the success of our approach. All of these changes are made without substantially altering the capabilities represented in the original grammar 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.

As an example, consider part of one sample grammar rule from PARSIFAL and its reformulation in the hybrid parser. Figure 1 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.

Rules in the hybrid parser eliminate many of these details without substantially 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 connectionist component of the hybrid. Actions such as creating and attaching or selecting the argument structure of the verb are carried out symbolically in the hybrid parser. 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.

3.2. 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 the hybrid parser 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 to transparently 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*, can not 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 can 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 the hybrid parser, 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 the hybrid parser 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 defined to uniformly mention items in both buffer cells.

Grammar actions. The repertoire of rule actions is slightly different in the hybrid parser. 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 connectionist 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.

3.3. The Grammar

The hybrid parser is capable of successfully processing a wide variety of 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 sub-symbolic system, which has 73 rules, 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 (like subject-aux inversion) and major movement rules like *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.

4. Architecture of the Hybrid Parser

The hybrid parser is composed of a connectionist network trained using backward propagation (Werbos 1974; Rumelhart et al, 1986) 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 typically represents a large number of patterns. They serve to relate situations that occur during parsing with the action deemed appropriate for that situation. Actions in the hybrid parser are performed symbolically on traditional data structures which are also maintained symbolically.

As Figure 2 illustrates, the hybrid parser is organized into a symbolic component and a connectionist component. The latter component is implemented as a numeric simulation of an adaptive neural network. The symbolic and connectionist components cooperate in a tightly coupled manner since there are proven advantages to this type of organization (Kitzmler and Kowalik, 1987). For the hybrid parser, the advantages are performance and robustness.

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 in the connectionist component. On the return side, it evaluates the activations of the output units, decides which action to perform, and performs that action, potentially modifying the stack and buffer in the process. The responsibility of the connectionist 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 back-propagation 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 feed-forward 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.

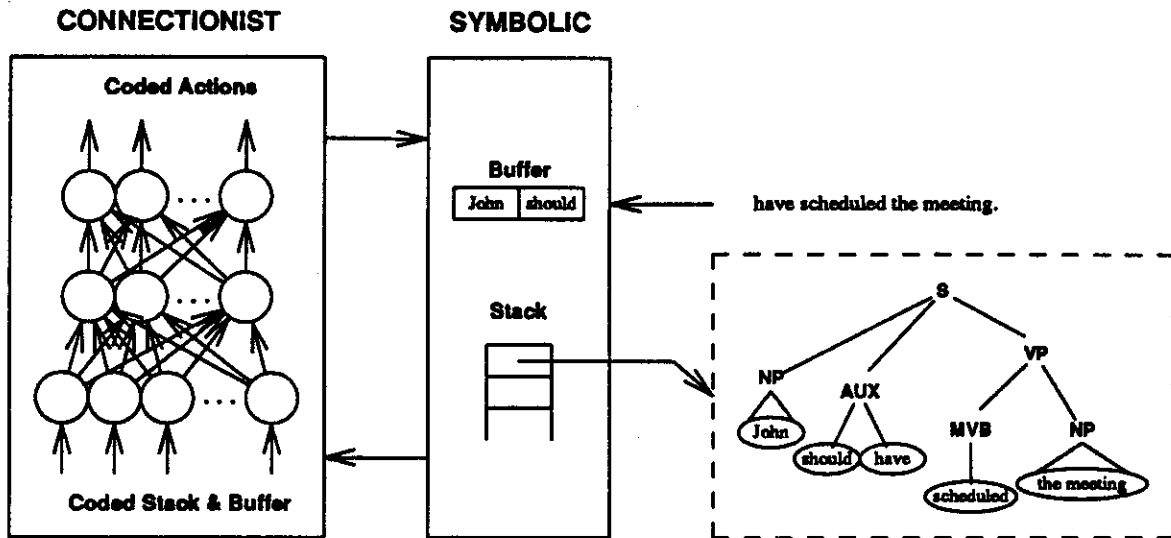


Figure 2: System Overview

In the set of experiments described here, the network has a three-layer architecture, as illustrated, with 66 input units, 40 hidden units, and 40 output units. Each input pattern consists of two feature vectors from the buffer items and one vector from the stack. The first vector activates 26 input units and the second vector activates 12 input units in a pattern vector representing a word or constituent of the sentence. The stack vector activates 28 units representing the current node on the stack and its attachments. One hidden layer has proven sufficient in all of these experiments. The output layer permits the choice of one out of 40 possible actions that can be performed on a single iteration of processing.

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 coded 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 2 illustrates the nature of the processing, although it shows a composite of the initial and final states of the parser. When a sentence form like "John should have scheduled the meeting" appears in the input stream, the first two constituents fill the buffer. These contents along with the contents of the top of the stack and its attachments are encoded and presented to the network. Coding is based on a simple scheme in which those features of the buffer and stack that are actually tested by grammar rules are represented (see Faisal, 1990). The network, in turn, produces a single action. Specification of the action by the network is done by activating one of the output units more than the others thus determining the winner (called "winner-take-all"). This action is then executed symbolically,

yielding changes in the buffer and stack. The process repeats until a stop action is performed at which time the resultant parse structure is left on top of the stack as shown.

4.1. Learning a Grammar

Training of the hybrid parser 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.

Our neural network simulator features a logistic function that computes values in the range of -1 to $+1$. 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. 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 purpose of judging convergence, a zero is substituted for each ? in the rule template to provide testing patterns. For more discussion of the training process, see Faisal and Kwasny (1990).

TABLE 1
Examples of Grammatical Sentences

Sentence Form
(1) Scheduled a meeting for Monday.
(2) John has scheduled the meeting for Monday.
(3) The meeting seems to have been scheduled for Monday.
(4) The jar seems broken.
(5) I persuaded John to do it.
(6) I saw him do it.
(7) Mary wants John to have a party.
(8) Mary wants to have a party.
(9) What will the man put in the corner?
(10) What will the man put the book in?
(11) Who did John see?
(12) Who broke the jar?
(13) Who is carrying the baby?
(14) What is the baby carrying?
(15) What did Bob give Mary?
(16) The man who wanted to meet Mary has disappeared.
(17) The man who hit Mary with a book has disappeared.
(18) The man whom Mary hit with a book has disappeared.
(19) I told that boy that boys should do it.
(20) That mouse that the cat chased had squeaked.
(21) I told Sue you would schedule the meeting.
(22) I told the girl that you would schedule the meeting.
(23) John is scheduling the meeting for Monday.

5. Performance

For testing purposes, 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 if the network would generalize in any useful way. Most of these examples were drawn from work cited earlier by Charniak and Milne. The objective is to discover exactly how syntactic context can aid in resolving such problems. In previous work, a simpler (23-rule) grammar was tested with similar results (Kwasny and Faisal, 1989).

5.1. Parsing Grammatical Sentences

Experimentation with grammatical sentences confirms that indeed the rules from the grammar have been learned sufficiently to parse sentences. When training with the rule templates, testing for convergence is possible by changing each ? to a zero value. Here the performance of the hybrid parser is examined with actual sentences and the claim that the parser simulates both PARSIFAL and LPARSIFAL is substantiated.

Grammatical sentences, by our definition, are those which parse correctly in the rule-based grammar from which the training set is derived. Table 1 shows several examples of grammatical sentences which are parsed successfully. Parse trees are developed which are identical with ones produced by other deterministic parsing systems.

5.2. Parsing Ungrammatical Sentences

Capabilities described above only duplicate what can be done rather comfortably symbolically. Of course, the feedforward network in the hybrid parser allows very fast decision-making due to the nature of the model. But what other features does the model possess? Importantly, how robust is the processing? As a

TABLE 2
Ungrammatical vs. Grammatical Sentences

Sentence Form	Strength
(1a) *John have should scheduled the meeting for Monday.	14.4
(1b) John should have scheduled the meeting for Monday.	56.9
(2a) *Has John schedule the meeting for Monday?	32.3
(2b) Has John scheduled the meeting for Monday?	36.8
(3a) *John is schedule the meeting for Monday.	9.5
(3b) John is scheduling the meeting for Monday.	54.7
(4a) *John is is scheduling the meeting for Monday.	7.2
(4b) John is scheduling the meeting for Monday.	54.7
(5a) *The boy did hitting Jack.	14.8
(5b) The boy did hit Jack.	137.7
(6a) *The meeting is been scheduled for Monday.	559.6
(6b) The meeting has been scheduled for Monday.	565.5

symbolic model, PARAGRAM extends PARSIFAL to handle ungrammatical sentences. 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 extending the inherent capabilities of the grammar.

To demonstrate its generalization capabilities, the hybrid parser is tested with several examples of ungrammatical sentences. Its performance is strictly dependent upon its training experiences since no relaxation rules (Kwasny and Sondheimer, 1981), meta-rules (Weischedel and Sondheimer, 1983), or other special mechanisms were added to the original grammar rules to handle ungrammatical cases. In Table 2, ungrammatical sentences used in testing are shown along with their strengths. These strengths are computed as the reciprocal of the average error per processing step for each sentence and reflect 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. These examples produce reasonable structures when presented to our system. Note that overall average strength is lower for ungrammatical sentences when compared to similar grammatical ones.

In sentence (1a), for example, the structure produced was identical to that produced while parsing sentence (1b). The only difference is that the two auxiliary verbs, *have* and *should*, were reversed in the parse tree. Sentence (2a) contains a disagreement between the auxiliary *has* and the main verb *schedule* and yet the comparable grammatical sentence (2b) parsed identically. Sentences (3a) and (4a) parse comparable to sentence (3b). Sentence (5a) is processed as if it were progressive tense ('The boy is hitting Jack'). In PARAGRAM, a nonsensical parse structure is produced for this sentence, as reported by Charniak (p. 137). It can be compared with sentence (5b), but there is not one clear choice for how the sentence should appear if grammatical. The problems with using a syntax-based approach to handling ungrammatical sentences are well-known (see, for example, Kwasny, 1980).

TABLE 3
Lexically Ambiguous vs. Unambiguous Sentences

Sentence Form	Strength
(1a) <Will> John schedule the meeting for Monday?	5.0
(1b) Will(aux) John schedule the meeting for Monday?	57.46
(2a) Tom <will> hit Mary.	29.8
(2b) Tom will(aux) hit Mary.	125.8
(3a) Tom <hit> Mary.	13.6
(3b) Tom hit(v) Mary.	29.5
(4a) The <will> gave the money to Mary.	16.6
(4b) The will(noun) gave the money to Mary.	61.9
(5a) They <can> fish(np).	20.6
(5b) They can(v) fish(np).	30.0
(6a) They can(aux) <fish>.	2.9
(6b) They can(aux) fish(v).	6.3

5.3. Lexical Ambiguity

As a further test of the generalization properties of the hybrid parser, sentences containing lexically ambiguous words are tested. Some of these sentences are shown in Table 3. Of course, ROBIE takes a symbolic approach in extending PARSIFAL to address these issues by requiring additional rules and lexical features. Note that in the deterministic approach, it is essential for lexical items to be properly disambiguated or backtracking will be required.

In testing the hybrid parser, normal sentences are presented, except that selected words are coded ambiguously (here indicated by angle brackets < > around the word). Sentences containing words followed by parentheses are presented to the hybrid parser unambiguously, even though these words have ambiguous uses. The lexical choices are shown in parentheses. In the cases shown, the lexically ambiguous words were correctly interpreted and reasonable structures resulted, although lower strengths were observed. The hybrid parser utilizes syntactic context to resolve these ambiguities and automatically works to relate novel situations to training cases through the generalization capability of the network. As before, no additional rules or mechanisms are required to provide this capability.

Sentence (1a) 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 as (1b). A similar result was obtained for sentence (2a) which parses as (2b). In sentence (3a), *hit* is coded to be ambiguous between an NP (as in a baseball hit) and a verb. The network correctly identifies it as the main verb of the sentence as in sentence (3b). Sentence (4a) is constructed as for sentence (4b). Sentence (5a) 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 (5b). Likewise, sentence (6a), which contains *fish* coded ambiguously as a verb/NP and *can* coded uniquely as an auxiliary verb, produces the same structure as sentence (6b). In the cases shown, the lexically ambiguous words were disambiguated and reasonable structures resulted. Note that the overall average strengths were lower than comparable grammatical sentences discussed, as expected.

6. Summary

Our hybrid deterministic parser is based on a deterministic grammar modified slightly from that found in traditional systems. Our grammar is derived from one used by Marcus, but with much inspiration from the work of Milne, Berwick, and Charniak. The rules of the grammar are utilized in training a connectionist component. The result is a hybrid system which exhibits characteristics from several well-known extensions of the basic deterministic parser. In particular, some ungrammatical and lexically ambiguous inputs can be successfully processed although no special provisions are made for them. These extended properties come essentially for free due to the coupling of a symbolic component with connectionism.

References

- Berwick, R.C. 1985. *The Acquisition of Syntactic Knowledge*. MIT Press, Cambridge, MA.
- Charniak, E. 1983. "A Parser with Something for Everyone." In *Parsing Natural Language*, M. King, ed. Academic Press, New York, NY, 117-150.
- Faisal, K.A. 1990. Connectionist Deterministic Parsing. D.Sc. Dissertation, Department of Computer Science, Washington University, St. Louis, Missouri.
- Faisal, K.A. and S.C. Kwasny. 1990. Deductive and Inductive Learning in a Connectionist Deterministic Parser. *Proc Intl Joint Conf Neural Networks*, II-471-474.
- Kitzmler, C.T., and J.S. Kowalik. 1987. Coupling Symbolic and Numeric Computing in Knowledge-Based Systems. *AI Magazine* 8, no. 2, 85-90.
- Kwasny, S.C. and K.A. Faisal. 1989. Competition and Learning in a Connectionist Deterministic Parser. *Proc 11th Conf Cog Sci Society*, 690-697.
- Kwasny, S.C. and N.K. Sondheimer. 1981. Relaxation Techniques for Parsing Ill-Formed Input. *Am J Comp Ling* 7, no. 2, 99-108.
- Kwasny, S.C. 1980. "Treatment of Ungrammatical and Extra-Grammatical Phenomena in Natural Language Understanding Systems." Indiana University Linguistics Club, Bloomington, Indiana.
- Marcus, M. P. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.
- McClelland, J. L., & A. H. Kawamoto. 1986. "Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences." In *Parallel Distributed Processing*, D.E. Rumelhart and J.L. McClelland, MIT Press, Cambridge, MA, 272-325.
- Milne, R. 1986. Resolving Lexical Ambiguity in a Deterministic Parser. *Comp Ling* 12, No. 1, 1-12.
- Rumelhart, D. E., G. Hinton, and R.J. Williams. 1986. "Learning Internal Representations by Error Propagation." In *Parallel Distributed Processing*, D.E. Rumelhart and J.L. McClelland, MIT Press, Cambridge, MA, 318-364.
- Weischedel, R.M. and N.K. Sondheimer. 1983. Meta-Rules as a Basis for Processing Ill-Formed Input. *Am J Comp Ling* 9, No. 3-4, 161-177.
- Werbos, P. 1974. "Beyond Regression: New Tools for Prediction and Analysis in Behavioral Science." Ph.D. Thesis. Harvard University, Cambridge, Ma.