Feedback as a Coindexing Mechanism in Connectionist Architectures

Mark A. *Jones*

AT&T Bell Laboratories

Murray Hill, New Jersey 07974

ABSTRACT

Co-occurrence constraints play an important role in rule-based systems such as natural language processing. The constraints appear in many forms including agreement restrictions (e.g.. number agreement between subjects and predicates), selectional restrictions on complement types, and filler-gap movement dependencies. A general account of such constraints is given in a parallel execution model for rule-based systems — Active Production Networks (APNs). The APN model is similar to connectionist (or spreading activation) models, but explicitly provides a *functional* interpretation of rule-based phenomena such as variable binding, multiple instantiations (including recursion), and contextual expectations. Co-occurrence constraints are represented by a theory of coindexing and trace capture, based on a feedback mechanism. Several examples of constraint processing are presented which, surprisingly, also include phenomena such as phonological nulls and contraction.

1. Introduction

There has been an increasing interest by researchers in cognitive science and artificial intelligence in parallel processing models. Examples include Anderson's ACT* system (Anderson 1983), the work on connectionism at Rochester (Feldman and Ballard 1982), the Boltzmann machine model (Fahlman 1979; Fahlman, Hinton and Sejnowski 1983), and the models of Hopfield (1982) and Kirkpatrick (Kirkpatrick, Gelatt and Vecchi 1983). The common theme in this research is that intelligent activity can be modeled by large networks of simple processing elements that use some restricted form of message passing on a massively parallel basis. The models are somewhat similar to earlier neural network or spreading activation theories of human cognitive processes (Selfridge 1958; Minsky and PapErt 1972; Quillian 1968; Collins and Loftus 1975), but benefit from advances in knowledge representation, production systems and learning algorithms. For example, several learning algorithms for multi-stage networks have recently been discovered, including the back-propagation learning algorithm of Rumelhart (Rumelhart and McClelland 1986). The lack of such learning algorithms was one obstacle to previous development of neural network models (Minsky and Papert 1972).

Current connectionist approaches are particularly attractive for computations which can be cast as function optimization. By using distributed energy minimization principles analogous to those of physical systems (e.g. annealing), minima that represent the problem solutions can be computed. A standard technique is to encode problem constraints in such a way that the energy minimum represents a "best-fit" solution; the energy minimization process is thus an iterative relaxation toward a solution. Hopfield and Tank (1985) have worked on schemes to model associative memories and the traveling salesman problem. Simulated annealing has been applied to problems in VLSI design (Kirkpatrick. Gelatt and Vecchi 1983; Otten and van Ginneken 1984). In AI much of the work has concentrated on best-fit problems in machine vision (Sabbah 1982; Hinton 1981) and natural language problems such as word sense disambiguation (Cottrell 1984; Waltz and Pollack 1984). Although relaxation techniques have promise as best-fit categorizers. it is not clear how the current models will deal with the rule-governed behavior of parsers or problem solvers. Selman (1985) has looked at the problem of parsing *bounded length* input strings. Touretzky and Hinton (1985) have proposed a hybrid production system architecture that attempts to capture some aspects of rule-based behavior. The overall operation is reminiscent of an OPS-5 interpreter in which conflict resolution is accomplished by a relaxation process.

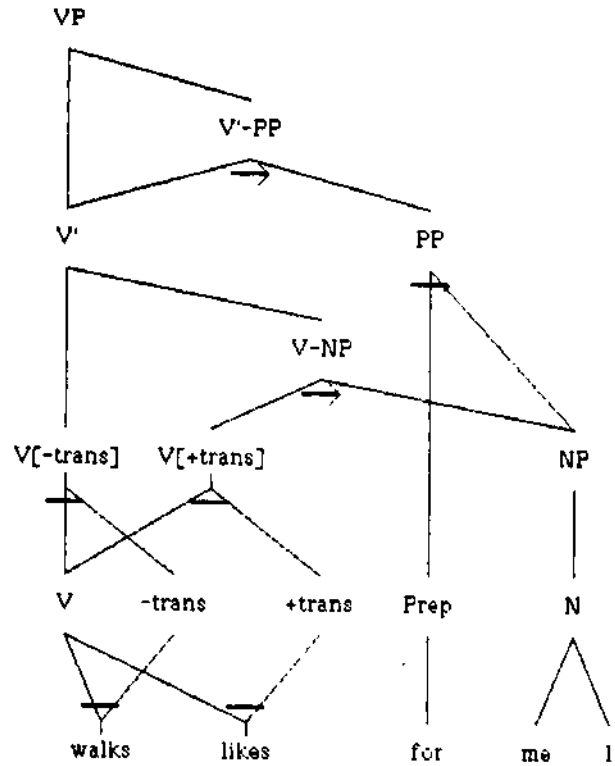
The *Active Production Network (APN)* model is a parallel, rule-based network framework which accounts for properties such as priming, variable binding, and coindexing that are important for tasks such as natural language processing. The APN model describes processing behavior in terms of a distributed activation algorithm as in connectionist models, but it is not centrally concerned with questions of detailed neural modeling, distributed vs. local representations, or fixed vs. dynamic connections. The emphasis is on the *functional* characteristics of context-driven, parallel, rule-based processing. The translation of the APN model into low-level connectionist architectures remains an interesting, open problem.

2. The APN Model

Each connector (node) in an Active Production Network is defined by an expression that describes the message patterns to which it responds and the resulting messages that are propagated. For convenience, each non-leaf connector is viewed as being primarily an input connector (multiple inputs, one output) or an output connector (multiple outputs, one input).* Figure I shows a simple APN and the corresponding rule-based description. The connector names in an APN are uninterpreted but are usually chosen to reflect the concepts which they represent. APNs bear some resemblance to rule connection graphs for theorem provers and graphical representations for grammars such as Augmented Transition Networks (ATNs) (Woods 1970). As in connectionist models, however, the APN network simultaneously represents the data (grammar, facts, rules) of the process and the process itself.

A comparison of the rules in Figure 1(b) with the network in Figure 1(a) reveals that the set of connectors has been explicitly designed to correspond to familiar concepts in rule-based systems, while retaining a clear interpretation in the activation-based model. The input connectors, for example, include disjunction and conjunction. The conjunctive connectors, furthermore, divide into synchronous and asynchronous types, and the asynchronous conjunctions may be ordered (sequence) or unordered. Terminal symbols correspond to network leaves having no inputs. Other common rule notations can be built out of these basic building blocks. Optionality. as in the *VP* rule, is expressed by a combination of disjunction and sequence. Repetition can be represented as recursive optionality. The unordered, asynchronous input conjunction (not shown in Figure 1) is useful in grammars for non-configurational (free word order) languages, for specifying order-free, conjunctive, semantic relationships and for representing unbounded movement.

The association of lexical and non-lexical features is often described in first-order rule languages by elevating lexical categories to predicates having associated terms to represent non-lexical features. The synchronous input conjunction performs this feature association or concept *specialization.* Specialization primarily defines feature sets and enforces selectional restrictions by intersecting features from orthogonal feature dimensions. For example, the concepts



(a) a simple APN grammar fragment

$VP :- V'\ (.\ PP).$
$V' :- V\ [-trans\ ].$
$V' :- V\ [+trans\ ].\ NP.$
$PP :- Prep.\ NP.$
$NP :- N.$

$V\ [-trans\ ] :- walks.$
$V\ [+trans\ ] :- likes.$
$Prep :- for.$
$N :- \{me\ I\}.$

(b) a corresponding rule notation

Figure 1. An APN Example

$V\ [-trans\ ]$ and $V\ [+trans\ ]$ specialize the concept V.** The connector. which averages its inputs. is termed "synchronous" because its response is maximal when all of the inputs simultaneously reach their maximum values.

Disjunctive and conjunctive output connectors distribute activation to (possibly) multiple connectors. Competing hypotheses arise from multiple disjunctive outputs. In the lexicon. for example. output disjunctions occur where there are multiple word senses or multiple feature values in a single dimension. Output conjunctions distribute activation along multiple

* In general, we will relax this convention to allow multiple disjunctive outputs (for input connectors) or disjunctive inputs (for output connectors). Prior definitions of the APN framework (Jones 1983; Jones and Driscoll 1985) were equivalently stated in node-based rather than connector-based terms. Nodes were defined with arbitrary amounts of nested input logic. The more atomic, connector-based scheme has proved more flexible and straightforward to implement.

** Of course, *V[-trans]* can also be considered a specialization of *—trans.*

feature dimensions. It is often convenient, as in Figure 1(b). to organize a grammar into rules which convey phrase structure (generally input connector logic) and rules which express lexical properties (output connector logic). Ordered and unordered output feedback connectors (not shown in Figure I) are discussed in a later section on coindexing.

In the basic cycle of execution in an APN parser, each word (or morpheme) in a sentence initiates message passing activity in the network through the corresponding (usually leaf) connector of the network; thus, the words supply *exogenous* inputs to the network. A uniform activation algorithm specifies the message passing behavior of each type of connector in response to the messages that it receives. In general, *activation messages* sweep "upward" in the network to instantiate or further instantiate the rules represented by the connectors in the network. Partially instantiated asynchronous rules initiate *expectation messages* "downward" through the network to contextually prime it. In parsing terms, the expectation messages dynamically compute a left-branching reachability set.

Messages are simple, having only an associated time and value. They do *not* encode complex structures such as entire binding lists, parse trees, feature lists or meaning representations. Similarly, the local processing of messages by connectors consists of simple operations such as maximizing input values. Consequently, the local computation time is bounded and the "result" of a computation consists entirely of the activation trace and the new state of the network. The information present in the activation trace can be interpreted to create a more traditional parse tree representation. For similar connectionist views, see Feidman and Ballard (1982) and Rumelhart and McClelland (1986). A comparison of marker passing, value passing and unrestricted message passing systems is given bv Fahlman, Hinton and Sejnowski (1983).

Message values are subject to the following decay function:

where / (/) is the message value at time /, and / (t_0) is the value of the most recent message, from time r_0. For the sake of efficiency in the current sequential simulation, the implementation numbers the input events and associates these integral "times" with the values sent in messages. The value is dynamically recomputed as necessary according to the decay function.

3. An Example

Before we enter a more formal discussion, a simple example should help to convey a feel for the execution behavior of the APN model. Figure 2 illustrate the operation of the APN in Figure 1 on the input string *likes me.* The subscripted connectors and links which appear to the right of the defined network represent the activation trace. The trace may be thought of as either an embedded *state* in the defined network or as a newly constructed copy or *instance* of a subgraph of the defined network. For expository purposes, the following discussion adopts the latter view. The widths of the links in Figure 2 crudely approximate activation levels which are actually represented as real numbers in the range [-MAX, +MAX]. The width of the output links from a connector indicates whether its pattern is fully (level = +MAX) or partially (0 < level < +MAX) satisfied. Expectations are shown by dashed lines in widths which analogously illustrate full (level = -MAX) or partial (-MAX < level < 0) activation levels. Unactivated links are shown by a thin solid line.

Figure 2(a) shows the network after the message passing activity spawned by the exogenous input *likes.* The activation messages passed upward in the network cause new connectors (or states) *V0. +trans0. V[+trans]0** etc. to be instantiated. Disjunctive input connectors (e.g.. *VPO)* maximize their inputs. Input sequences (e.g., *V-NPO)* respond weakly to the first input, using it primarilv to gate the second input value. Expectation messages are propagated downward from *V-NPO* (opposite in sign, but equal in magnitude to the value of the first input) to condition the network for an *NP.* At the end of each cycle of message passing, the most highly-expected leaves represent the new, primed context.

In Figure 2(b). *me* is introduced and activation proceeds upward in the presence of the expectations. Expectation messages do not instantiate rules, but they can dynamically alter the behavior of the network. At *NP,* the expectations from *V-NPO* cause it (rather than *PP.* for example) to receive the activation message. Thus the network exhibits nonmonotonic behavior dependent upon the processing context. In different expectation environments, the same input sequence can lead to different activation patterns.

It may help to compare the APN execution model with that of Prolog. Prolog uses backward-chaining, static rule ordering and backtracking for nondeterminism. Roughly, APNs use forward-chaining, dynamic rule ordering (based on expectations which are passed by backward-chaining) and parallelism. The message passing activity in the network has a formal inference analog. From an activation-based perspective, the rules in Figure 1(b) may be understood as universally quantified by an input event. For example, the rule *V[+trans]:-likes* is formally represented as (VA) *likes* (.*)-VCO^{A} *+trans (x).* An input event at time *o will activate *likes (XQ)* and effectively instantiate the rule by universal specializa-

tion and *modus ponens.* Once an input has been bound, its activation value can modulate in response to further input events which it dominates. Jones (1986) lists inference schemas that describe the semantics of activation messages for APN connectors.

The value of *VPO* and hence the expectations from any input sequences being driven above *VPO* are continuously modulated up and down as the constituents below it are started and progress to completion. The final value of *VPO,* in Figure 2(b). is +MAX. indicating that the phrase was successfully parsed. If a *PP* attachment were to commence at this point, the value of *VPO* would be lowered until the *PP* "popped" by raising its value. It is this raising and lowering of activation levels, a kind of distributed pushing and popping of context, that promotes generally context-free descriptions (augmented by coindexing constraints) of the syntactic base of natural languages. A similar phenomenon may extend to other levels of description (e.g.. see Litman (1985) for a discussion of nested discourse contexts).

Another interesting property of the APN execution model is that an ungrammatical input may be recognized, but the overall score computed for the phrase will be penalized. Unlike rigid rule-based systems where rules discretely succeed or fail, APN rules are "leaky" and may partially succeed. This situation arises, for example, when obligatory sequences or conjunctions are only partially satisfied. Also, a small percentage of an expectation value received by an input sequence connector is distributed to the second input; consequently, the nearest attachment can be found even when constituents are missing.

4. Coindexing

The grammar writing style exemplified in Figure 1 introduced a number of important representational paradigms including the widespread use of features and specialization in an *X* style grammar. In *X* theory, rules generally have the form $X^C_1)\ldots(C,)-X^{n}\text{-}^{r}\text{-}(C_{;}+i)\bullet\bullet\bullet(Q)$, where each *C,* is either a full projection of a lexical category or a grammatical formative (Jackendoff 1977). The remainder of this paper addresses the problem of how to represent the numerous co-occurrence constraints that must be added to the context-free skeleton. We will focus primarily on agreement restrictions and selectional restrictions on complements, although a related application of the techniques may apply to

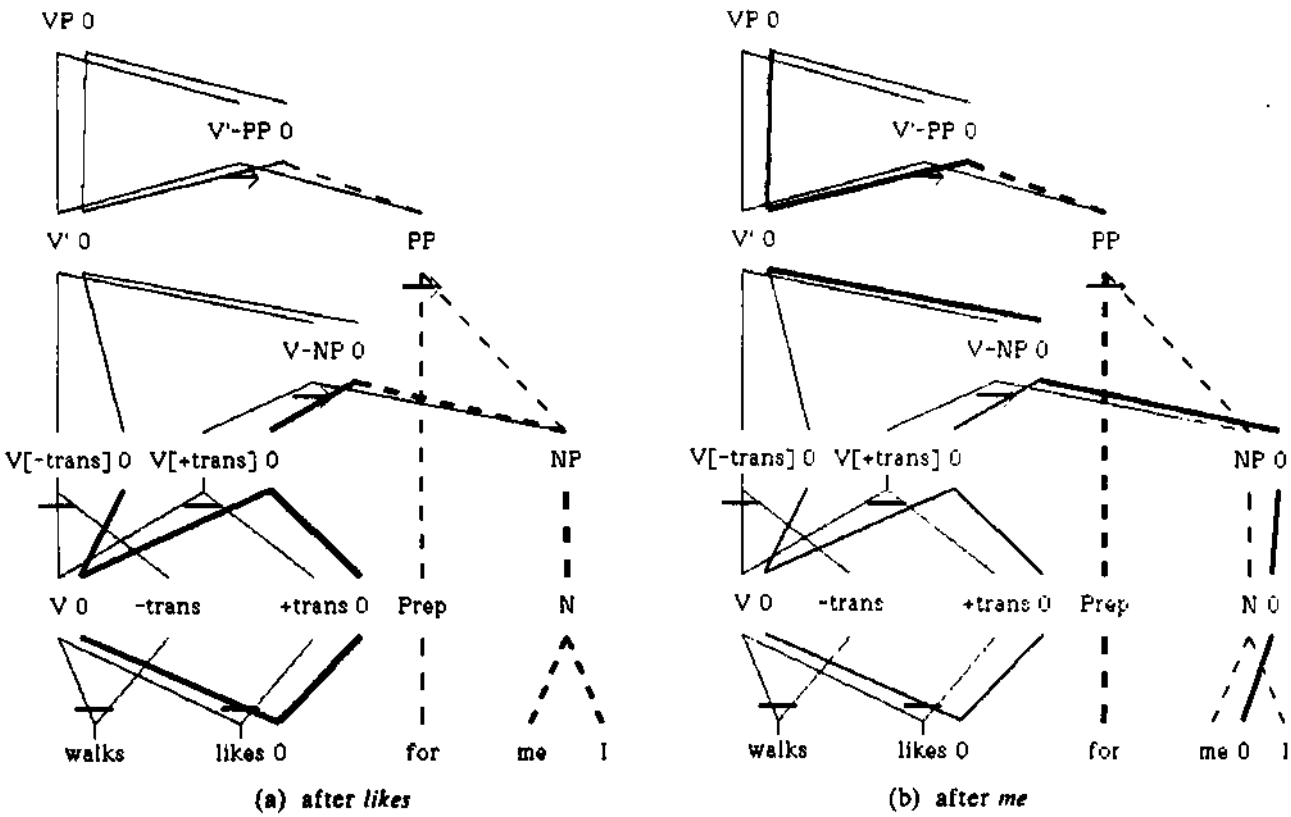


Figure 2. Stages in the Parsing of *likes me*

filler-gap movement dependencies as well (Jones 1985).

A common type of constraint, technically distinguished here *as feature agreement,* occurs when the values of features in two constituents may vary, but must match. In logic programming, feature agreement is implicitly specified by using the same variable name and requiring that the variable assignments unify. For example, determiner-noun number agreement is written as:

NP :- *Det [number*]. *N [number*].

Another type of co-occurrence constraint, *feature selection,* occurs in situations in which the presense of a particular feature constrains the value of another feature in a complement. For example, the rule for transitive VPs in Figure 1 fails to capture the case constraint on the object NP. A more accurate description is:

VP :- *V[+trans]. NP[+obj).*

One account is that the +trans feature controls the case of the NP. A similar (if not the same) feature of prepositions controls case in PPs. The same method can be used to define complement and adjunct conditions in general (e.g., voice, mood, tense). Unfortunately, the usual logic programming style does not make the dependence of *+obj* upon *+trans* explicit as in the feature agreement case. The rule language being developed in an APN compiler project syntactically distinguishes several types of feature dependency.

Feature agreement and feature selection both specify *coindexing* relationships. The general method for coindexing in the APN framework requires an activation *trace* along a feedback path between the coindexed feature, represented by an output feedback connector, and an asynchronous input conjunction which dominates it. As expectations feed back to the coindexed point, the coindexed feature is released into the new expectation environment creating a new activation trace. The trace effectively pre-instantiates the next input in the asynchronous input conjunction, constraining it to possess the selected features. Note that this trace capture occurs during the time frame of the current input without advancing in the input string. The feedback response threshold can be tuned to disallow feedback through distant (weak, non-head) paths; this effectively implements a version of the head-feature convention prevalent in linguistic theories.

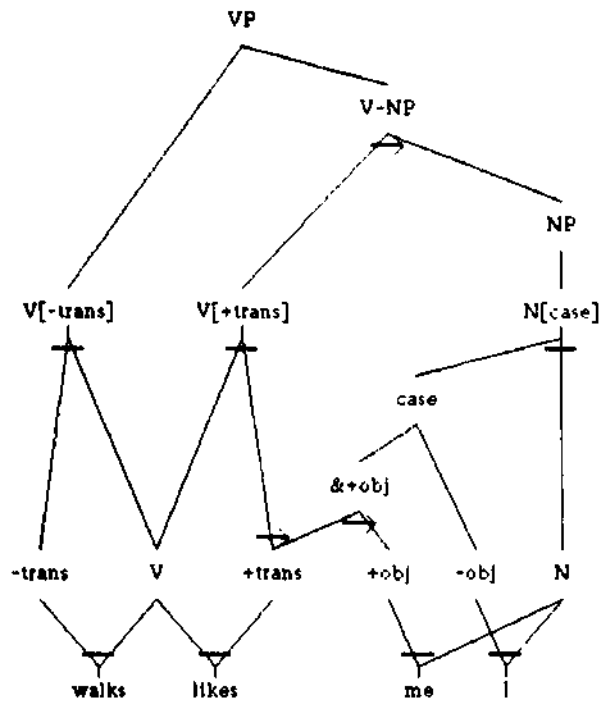
The captured trace produces different effects depending on the form of the grammar that contains it. In the feature selection case, shown in Figure 3 and discussed below, the expectations are transferred to the selected feature(s). For feature agreement, as in Figure 4, the expectations feed back directly to the coindexed feature. Feature control is thus a lexical property; for example, the *+obj* feature must conjoin to the NP head at the level of the lexicon, below any phrase structure specifier sequences. The expectation levels for the selected feature(s) will be modulated together with those of the head (N) by any intervening context.

Figures 3(b)-(d) illustrate the APN feature selection technique applied to object case control for transitive verbs with the phrase *likes me.* Figure 3(b) shows the network state after activation from the verb. Note that an object NP instance has already been created and an input sequence is used to transfer the expectations to *+obj* feature. In Figure 3(c). after *me.* the *+obj* feature was correctly bound. Note that a lexical entry not possessing a feature of type *+obj* (such as /) would fail to satisfy the expectations of *caseO* and would be penalized accordingly.

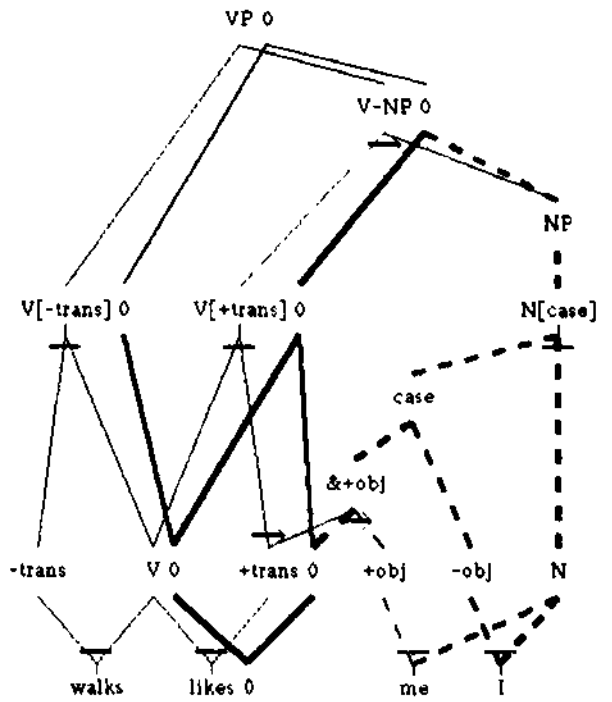
Figures 4(b)-(d) shows an example of feature agreement with the phrase *a deer.* In Figures 4(b)-(c), the activation from *numberO* is integrated with the signal from *DetO* and fed back to to the *N* side of the *NP.* where it is captured. The expectations "hop off" the trace at the disjunctive input connector *numberO.* which posts expectations for the connector *(-plur)* that spawned its currently active input. The expectation feedback prevents the full binding of the *numberO* connector to an instance, but results in a pattern specialization. Effectively, this causes the rule (Vx) *-plur (x)-+number* (x_0) to become active. In Figure 4(d), *numberO* is rebound to the new feature value *-plurJ.*

The unordered output feedback connector, *xand>.* captures more general feature agreement relationships than *seq>*\ multiple, matching input sequences can specify different input orders. In English subject-verb agreement, for example, the tensed verb element precedes the subject in questions and follows the subject in declaratives. As in the case of the input connectors *seq<* and *xand<. seq>* is really just a special case of *xand>* in which the ordering is strict.

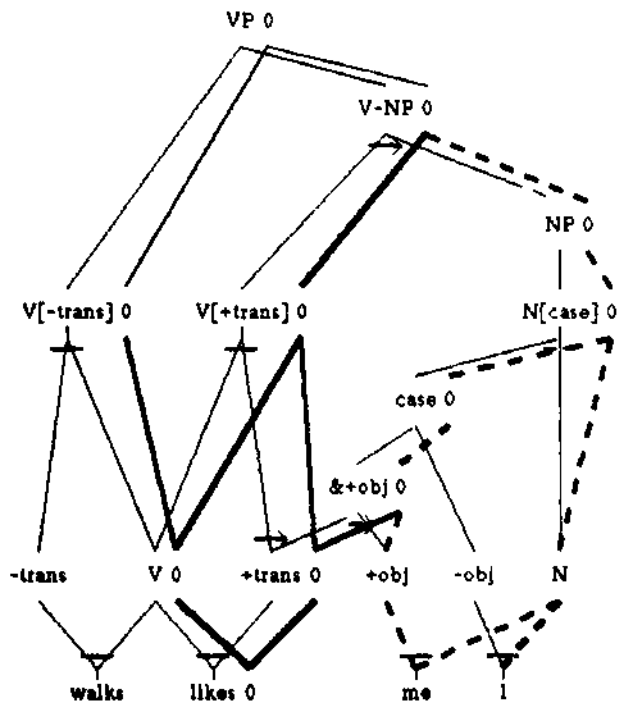
The reader should note that the feature dimensions of number and lexical category in the NP grammar of Figure 4(a). for example, are orthogonal. The syntactic number agreement can be layered onto an initial, over-generalized grammar that permitted any determiner and noun combination on the basis of lexical category alone. The number feature can also be coindexed to verbs for subject-verb agreement. This kind of error correction is non-destructive, i.e.. previous network fragments do not have to be eliminated or completely rewritten. This property may be especially relevant for learning algorithms or incremental compiling approaches. Since the computations in each feature dimension proceed in parallel, the accuracy is improved without a penalty in (parallel) performance. The intersection of relatively simple feature systems thus gives rise to complex behavior. Woods (1980) offers a similar example of cascaded



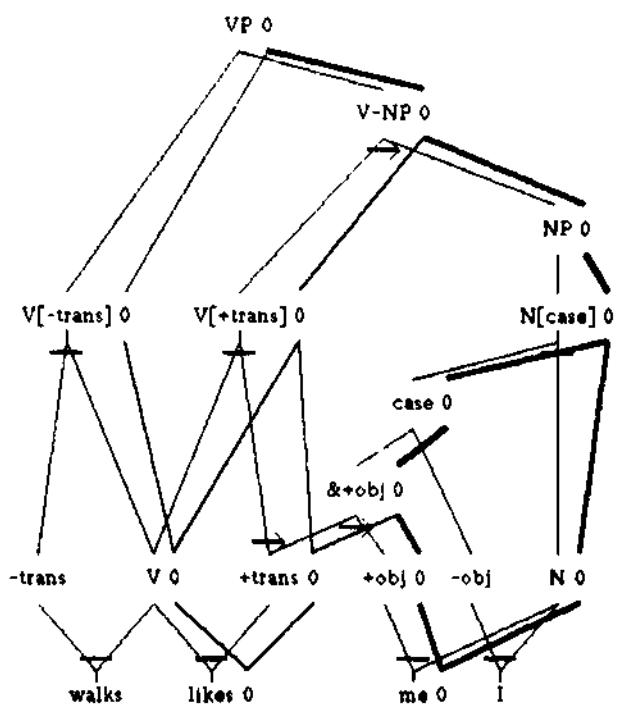
(a) object case control



(b) just before *case* feedback



(c) after *likes*



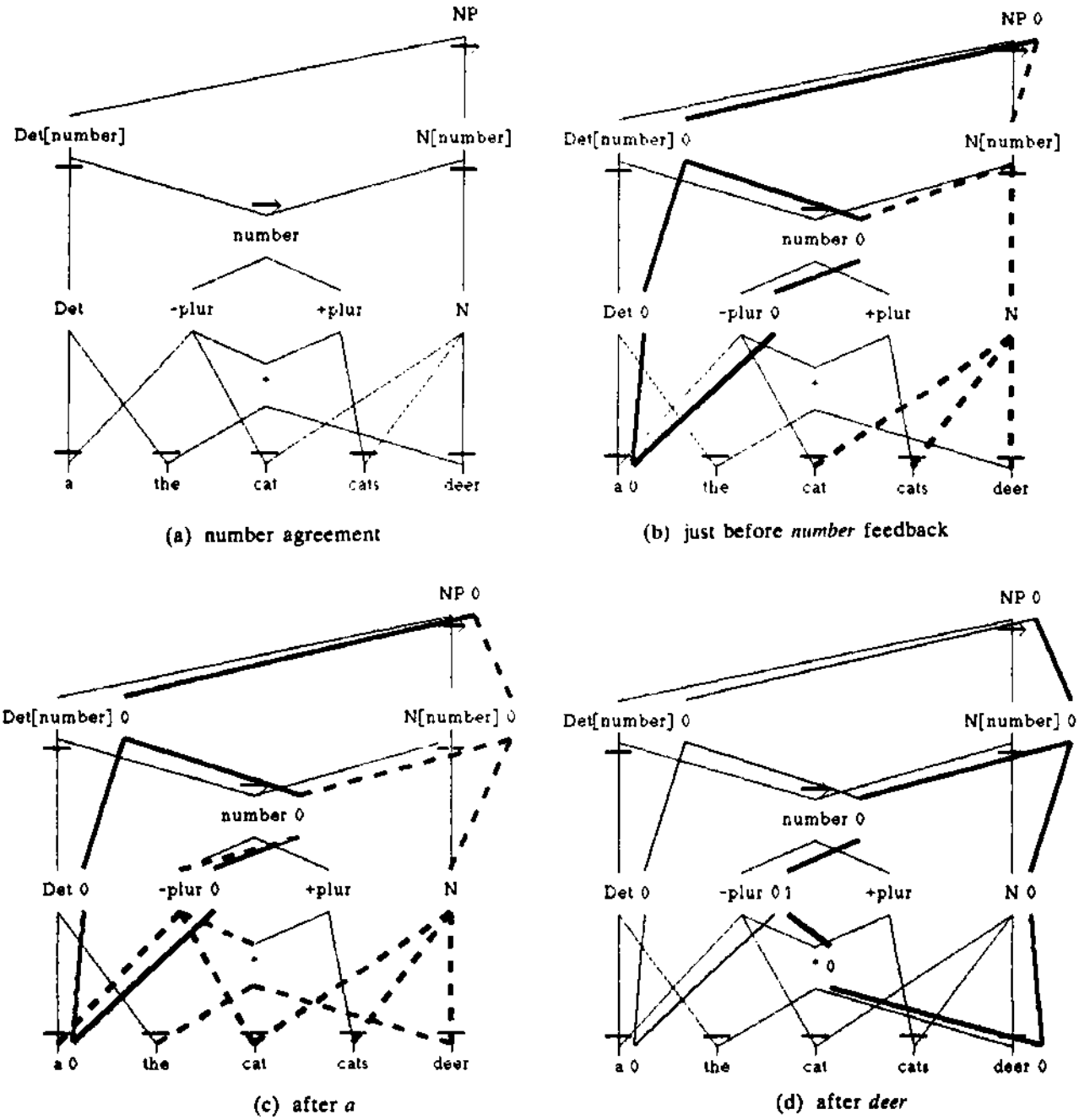
(d) after *me*

Figure 3. An Example of Feature Selection

context-free ATN grammars which recognize a context-sensitive language.

4.1 Phonological Nulls

Linguistic theories have postulated the existence of phonologically null elements which contribute to the parsing of a sentence. Government-Binding Theory (Chomsky 1982), recognizes four different types of empty categories. In contrast, Generalized Phrase Structure Grammar (GPSG) (Gazdar 1985), and Lexical Functional Grammar (LFG) (Bresnan 1982) use a single empty category. Empty categories typically relate to missing NPs and *wh* movement (discussed in the following section), but there are also other applications. In some accounts, for example, a phonologically null preposition, Φ, appears before the object NP to assign case to it. Figure 5 illustrates how the network in Figure 3(a) could be modified to



(a) number agreement

(b) just before *number* feedback

(c) after *a*

(d) after *deer*

Figure 4. An Example of Feature Agreement

represent this theory. The same general principle used for feature selection is applied to preinstantiate the lexical category itself (instead of one of its features). Upon feedback to *+trans,* it will insert itself as the null preposition. In turn, as a preposition, it will then pre-select the *+obj* feature for the object NP. This has the advantage of localizing NP case control to the preposition only.

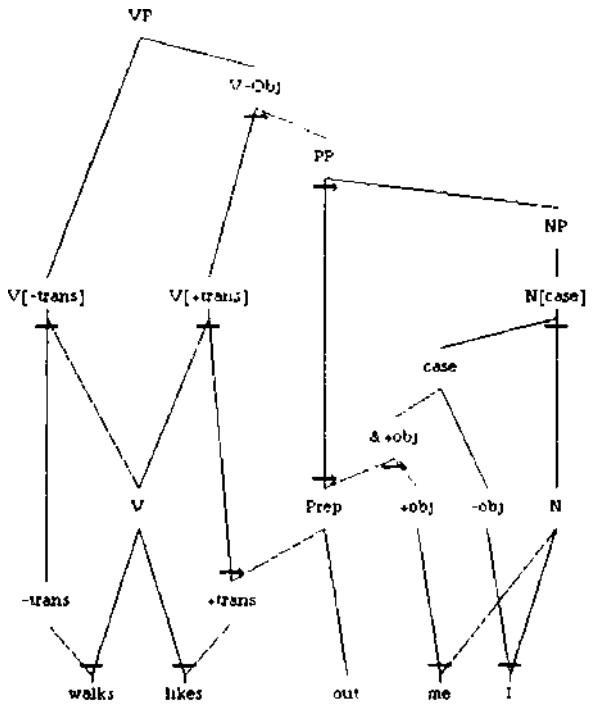


Figure 5. An Example of Phonological Nulls

Interestingly, a similar principle can be used in the representation of phonological contractions or compounds. Figure 6 illustrates the idea for auxiliary-negation contractions. The feedback onto the contraction completes the respelling of *can't,* for example, into *can not.*

5. Conclusions

Although more work is needed to understand how to model syntactic and semantic processing in activation-based schemes, several key representational techniques have been developed. The techniques include the intersection and synchronization of modular rule systems, specialization, feature selection, feature agreement and phonological nulls. In addition, a general grammar writing methodology based on the features and phrase structure style of *X* theory, modularity, and feature coindexing has been proposed. This choice is consistent with trends in linguistics away from transformational theories toward modular theories such as Government-Binding Theory or perhaps some form of phrase structure grammar such as GPSG. It is our hope that adjustments in linguistic theory and in the APN model will produce a parsimonious account of the expressive power of natural languages. The current APN model has evolved over the last three years and should probably be taken as representative of a class of possible approaches.

The parallel execution model of APNs has several advantages over serial models. Besides the ability to maintain multiple hypotheses in parallel, the bandwidth among collections of modules (knowledge sources) can be quite high. Large numbers of features can be activated simultaneously in multiple modules; conversely, the collective context from these knowledge sources is also simultaneously projected in the form of expectations to reduce the nondeterminism engendered from any single source alone. Furthermore, the model can be used for both recognition and generation processes.

Unlike most connectionist models, we have tried to model explicitly the functional characteristics of rule based systems such as variable binding, feature selection and agreement, multiple instantiations (including recursion), and contextual expectations. Although we are supportive of connectionist efforts to develop neural models and learning algorithms, we feel that there is also a need for research efforts such as the APN model to reach toward them from what is known about rule-based systems. Efforts from both directions can help prune the space of possible models, offer additional levels of description, and bridge the "symbol" gap.

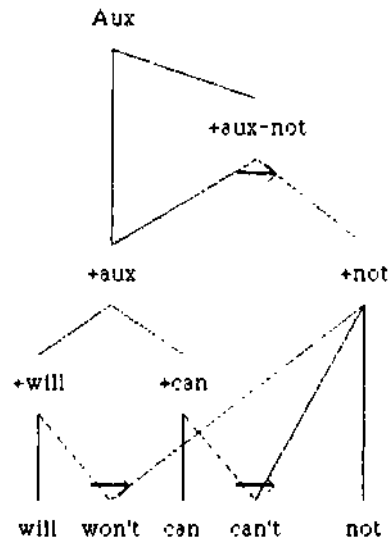


Figure 6. An Example of Phonological Contraction

ACKNOWLEDGEMENTS

I owe a particular debt to Guy Story who. besides providing programming support for the APN system, has also contributed valuably to the insights presented in this paper.

REFERENCES

Anderson, J. R. 1983 *The Architecture of Cognition.* Harvard University Press. Cambridge. Massachusetts.

Bresnan, J. (Ed.) 1982 *The Mental Representation of Grammatical Relations.* MIT Press. Cambridge. Massachusetts.

Chomsky, N. 1982 *Lectures on Government and Binding.* Foris Publications, Dordrecht. Holland.

Collins, A. and Loftus. E. F. 1975 A Spreading-Activation Theory of Semantic Processing. *Psychological Review* 82: 407-428.

Cottrell, G. W. 1984 A Model of Lexical Access of Ambiguous Words. *AAAI-84.* Austin, Texas: 61-67.

Fahlman. S. 1979 *NETL: A System for Representing and Using Real-World Knowledge.* MIT Press, Cambridge, Massachusetts.

Fahlman. S. E.; Hinton, G. E.; and Sejnowski, T. J. 1983 Massively Parallel Architectures for AI: NETL, Thistle, and Boltzmann Machines. *AAAI-83.* Washington, D.C.: 109-113.

Feldman, J. A. and Ballard. D. H. 1982 Connectionist Models and their Properties. *Cognitive Science* 6: 205-254.

Gazdar, G.; Klein, E.; Pullum, G.; and Sag, I. 1985 *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Massachusetts.

Hinton, G. F. 1981 Shape Representation in Parallel Systems. *Proceedings of the 7th UCAI.* Vancouver, B.C.: 1088-1096.

Hopfieid, J. J. 1982 Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proc. Natl. Acad. Sci.* 79: 2554-2558.

Hopfieid, J. J. and Tank, D. W. 1985 "Neural" Computation of Decisions in Optimization Problems. *Biological Cybernetics* 52: 141-152.

Jackendoff. R. 1977 *X-Bar Syntax: A Study of Phrase Structure.* MIT Press. Cambridge, Massachusetts.

Jones, M. A. 1983 Activation-Based Parsing. *Proceedings of the 8th IJCAI.* Karlsruhe, West Germany: 678-682.

Jones. M. A. and Driscoll. A. S. 1985 Movement in Active Production Networks. *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics.* Chicago: 161-166.

Jones, M. A. 1986 Active Production Networks — An Activation-Based Parsing Model, released paper (submitted for publication), AT&T Bell Laboratories.

Kirkpatrick, S.; Gelatt, Jr., C. D.; and Vecchi, M. P. 1983 Optimization by Simulated Annealing. *Science* 220: 671-680.

Litman, D. 1985 Linguistic Coherence: A Plan-Based Alternative. *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics.* New York: 215-223.

Minsky, M. and Papert, S. 1972 *Perceptrons.* MIT Press, Cambridge, Massachusetts.

Otten, R. H. J. M. and van Ginneken, L. P. P. P. 1984 Floorplan Design Using Simulated Annealing. *International Conference on Computer-Aided Design.* Santa Clara, California: 96-98.

Quillian, M. R. 1968 Semantic Memory. In: Minsky, M. L.. Ed.. *Semantic information Processing.* MIT Press, Cambridge, Massachusetts: 227-270.

Rumelhart, D. E. and McClelland, J. L. 1986 *Parallel Distributed Processing.* MIT Press, Cambridge, Massachusetts.

Sabbah, D. 1982 A Connectionist Approach to Visual Recognition. Technical Report 107. University of Rochester, Rochester, New York.

Selfridge, O. G. 1958 Pandemonium: A Paradigm for Learning. In: Uhr, L.. Ed., *Pattern Recognition.* Wiley, New York: 237-250.

Selman, B. 1985 Rule-Based Processing in a Connectionist System for Natural Language Understanding. Technical Report CSRI-168. University of Toronto, Toronto. Canada.

Touretzky. D. S. and Hinton. G. E. 1985 Symbols Among the Neurons: Details of a Connectionist Inference Architecture. *Proceedings of the 9th UCAI.* Los Angeles, California: 244-248.

Waltz. D. L. and Pollack, J. B. 1984 Phenomenoiogically Plausible Parsing *AAAI-84.* Austin. Texas: 335-339.

Woods. W. A. 1970 Transition Network Grammars for Natural Language Analysis. *Communications of the ACM* 13(10): 591-606.

Woods, W. A. 1980 Cascaded ATN Grammars. *AJCL* 6(1): 1-12.