DEDUCTIVE PARSING WITH MULTIPLE LEVELS OF REPRESENTATION.*

Mark Johnson,
Brain and Cognitive Sciences, M.I.T.

ABSTRACT

This paper discusses a sequence of deductive parsers, called PAD1 – PAD5, that utilize an axiomatization of the principles and parameters of GB theory, including a restricted transformational component (Move-α). PAD2 uses an inference control strategy based on the 'freeze' predicate of Prolog-II, while PAD3 – 5 utilize the Unfold-Fold transformation to transform the original axiomatization into a form that functions as a recursive descent Prolog parser for the fragment.

INTRODUCTION

This paper reports on several deductive parsers for a fragment of Chomsky's Government and Binding theory (Chomsky 1981, 1986; Van Riemsdijk and Williams 1984). These parsers were constructed to illustrate the 'Parsing as Deduction' approach, which views a parser as a specialized theorem-prover which uses knowledge of a language (i.e. its grammar) as a set of axioms from which information about the utterances of that language (e.g. their structural descriptions) can be deduced. This approach directly inspired by the seminal paper by Pereira and Warren (1983). Johnson (1988a) motivates the Parsing as Deduction approach in more detail than is possible here, and Johnson (1988b) extends the techniques presented in this paper to deal with a more complex fragment.

In this paper I describe a sequence of model deductive parsers, called PAD1 – PAD5, for a fragment of GB theory. These parsers are not designed for practical application, but simply to show that GB deductive parsers can actually be built. These parsers take PF representations as their input and produce LF representations as their output. They differ from most extant GB parsers in that they make explicit use of the four levels of representation that GB attributes to an utterance – namely D-structure, S-structure, PF and LF – and the transformational relationship that holds between them. A 'grammar' for these parsers consists entirely of a set of parameter values that parameterize the principles of GB theory – thus the parsers described here can be regarded as 'principle-based' (Berwick 1987) – and the parsers' top-level internal structure transparently reflects (some of) the principles of that theory; X' and θ theory apply at D-structure, Case theory applies at S-structure, Move-α is stated as a relation between D- and S-structure, and LF-movement relates S-structure and LF. In particular, the constraints on S-structures that result from the interaction of Move-α with principles constraining D-structure (i.e. X' and θ theories) are used constructively throughout the parsing process.

The PAD parsers are designed to directly mirror the deductive structure of GB theory. Intuitively, it seems that deductive parsers should be able to mirror theories with a rich internal deductive structure; these parsers show that to a first approximation this is in fact the case. For example, the PAD parsers have no direct specification of a 'rule' of Passive, rather they deduce the relevant properties of the Passive construction from the interaction of θ theory, Move-α, and Case theory.

It must be stressed that the PAD parsers are only 'model' parsers. The fragment of English they accept could only be called 'restricted'. They have no account of WH-movement, and Move-α is restricted to apply to lexical categories, for example, and they incorporate none of the principles of Bounding Theory.

* Steven Abney, Bob Berwick, Nelson Correa, Tim Hickey, Elizabeth Highleyman, Ewan Klein, Peter Ludlow, Martin Kay, Fernando Pereira and Whitman Richards all made helpful suggestions regarding this work, although all responsibility for errors remains my own. The research reported here was supported by a grant by the Systems Development Foundation to the Center for the Study of Language and Information at Stanford University and a Postdoctoral Fellowship awarded by the Fairchild Foundation through the Brain and Cognitive Sciences Department at MIT.

However, the techniques used to construct these parsers are general, and they should extend to a more substantial fragment.

A SKETCH OF GB THEORY

In the remainder of this section I sketch the aspects of GB theory relevant to the discussion below; for more detail the reader should consult one of the standard texts (e.g. Van Riemsdijk and Williams 1986). GB theory posits four distinct representations of an utterance, D-structure, S-structure, PF and LF. To a first approximation, D-structure represents configurationally the thematic or predicate-argument structure of the utterance, S-structure represents the utterance's surface constituent structure, PF represents its phonetic form, and LF ("Logical Form") is a configurational representation of the scopal relationships between the quantificational elements present in the utterance. The PF and LF representations constitute the interface between language and other cognitive systems external to the language module (Chomsky 1986, p. 68). For example, the PF representation "Everybody is loved" together with the D-structure, S-structure and LF representations shown in Figure 1 might constitute a well-formed quadruple for English.

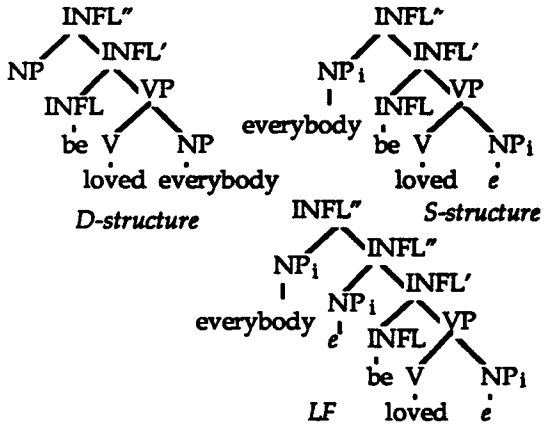

Figure 1: Representations of GB Theory.

In order for such a quadruple to be well-formed it must satisfy all of the principles of grammar; e.g. the D-structure and S-structure must be related by Move-α, the D-structure must satisfy X'-theory and Θ-theory, etc. This is shown schematically in Figure 2, where the shaded rounded boxes indicate the four levels of representation, the boxes indicate relations that must hold simultaneously between pairs of structures, and the ellipses designate properties that must hold of a single structure. This diagram is based on the organization of GB theory sketched by Van Riemsdijk and Williams (1986, p. 310), and represents the organization of principles and structures incorporated in the parsers discussed below.

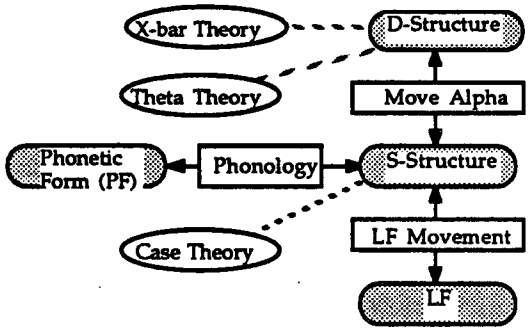

Figure 2: (Some of) The Principles of GB Theory.

The principles of grammar are parameterized; the set of structures they admit depends on the value of these parameters. These principles are hypothesised to be innate (and hence universally true of all human languages, thus they are often called 'Universal Grammar'), so the extra knowledge that a human requires in order to know a language consists entirely of the values (or settings) of the parameters plus the lexicon for the language concerned. The syntax of the English fragment accepted by the parsers discussed below is completely specified by the following list of parameters. The first two parameters determine the X' component, the third parameter determines the Move-α relation, and the fourth parameter identifies the direction of Case assignment.

(1) headFirst.
specFirst.
movesInSyntax(np).
rightwardCaseAssignment.

I conclude this section with some brief remarks on the computational problems involved in constructing a GB parser. It seems that one can only construct a practical GB parser by simultaneously using constraints from all of the principles of grammar mentioned above (excepting LF-Movement), but this involves being able to 'invert' Move-α 'on the fly'. Because of the difficultly of doing this, most

implementations of GB parsers ignore Move-α entirely and reformulate X' and Θ Theories so that they apply at S-structure instead of D-structure, even though this weakens the explanatory power of the theory and complicates the resulting grammar, as Chomsky (1981) points out. The work reported here shows that it is possible to invert a simple formulation of Move-α 'on the fly', suggesting that it is possible to build parsers that take advantage of the D-structure/S-structure distinction offered by GB theory.

PARSING AS DEDUCTION

As just outlined, GB theory decomposes a competent user's knowledge of a language possessed into two components: (i) the universal component (Univeral Grammar), and (ii) a set of parameter values and a lexicon, which together constitute the knowledge of that particular language above and beyond the universal component. The relationship between these two components of a human's knowledge of a language and the knowledge of the utterances of that language that they induce can be formally described as follows: we regard Universal Grammar as a logical theory, i.e. a deductively closed set of statements expressed in a specialized logical language, and the lexicon and parameter values that constitute the specific knowledge of a human language beyond Universal Grammar as a set of formulae in that logical language. In the theory of of Universal Grammar, these formulae imply statements describing the linguistic properties of utterances of that human language; these statements constitute knowledge of utterances that the parser computes.

The parsers presented below compute instances of the 'parse' relation, which is true of a PF-LF pair if and only if there is a D-structure and an S-structure such that the D-structure, S-structure, PF, LF quadruple is well-formed with respect to all of the (parameterized) principles of grammar. For simplicity, the 'phonology' relation is approximated here by the S-structure 'yield' function. Specifically, the input to the language processor are PF representations and that the processor produces the corresponding LF representations as output. The relationship between the parameter settings and lexicon to the 'parse' relation is sketched in Figure 3.

Knowledge of the Language

Parameter Settings
headFirst.
specFirst.
movesInSyntax(np).
rightwardCaseAssignment.

Lexicon
thetaAssigner(love).
thetaAssigner(loved).
nonThetaAssigner(sleep).
...

⇓ *imply in the theory of Universal Grammar*

Knowledge of Utterances of the Language.

parse([everybody,-s,love,somebody],
[everybody$_i$ [somebody$_j$ [I'' [NP e_i] [I' [I -s]
[V'' [V' [V love] [NP e_j]]]]]]])

parse([everybody,-s,love,somebody],
[somebody$_j$ [everybody$_i$ [I'' [NP e_i] [I' [I -s]
[V'' [V' [V love] [NP e_j]]]]]]])

.....

Figure 3: Knowledge of a Language and its Utterances.

It is important to emphasise that the choice of logical language and the properties of utterances computed by the parser are made here simply on the basis of their familiarity and simplicity: no theoretical significance should be attached to them. I do not claim that first-order logic is the 'language of the mind', nor that the knowledge of utterances computed by the human language processor are instances of 'parse' relation (see Berwick and Weinberg 1984 for further discussion of this last point).

To construct a deductive parser for GB one builds a specialized theorem-prover for Universal Grammar that relates the parameter values and lexicon to the 'parse' relation, provides it with parameter settings and a lexicon as hypotheses, and uses it to derive the consequences of these hypotheses that describe the utterance of interest. The Universal Grammar inference engine used in the PAD parsers is constructed using a Horn-clause theorem-prover (a Prolog interpreter). The Horn-clause theorem-prover is provided with an axiomatization $\mathcal{U}$ of the theory of Universal

Grammar as well as the hypotheses $\mathcal{H}$ that represent the parameter settings and lexicon. Since a set of hypotheses $\mathcal{H}$ imply a consequence F in the theory of Universal Grammar if and only if $\mathcal{H} \cup \mathcal{U}$ implies F in first-order logic, a Horn-clause theorem-prover using axiomatization $\mathcal{U}$ is capable of deriving the consequences of $\mathcal{H}$ that follow in the theory of Universal Grammar. Thus the PAD parsers have the logical structure diagrammed in Figure 4.

Knowledge of Language

Axiomatization of Universal Grammar

```
parse(String, LF) :-
        xBar(infl2,DS), theta(infl2,0,DS),
        moveAlpha(DS,[],SS,[]),
        caseFilter(infl2,0,SS),
        phonology(String/[],SS),
        lfMovement(SS,LF).
```

Parameter Settings + Lexicon

```
headFirst.
...
thetaAssigner(love).
...
```

..... ⇓ *imply in First-order Logic* .....

Knowledge of Utterances of the Language.

parse([everybody,-s,love,somebody],
[everybody$_i$ [somebody$_j$ [I'' [NP e_i] [I' [I -s]
[V'' [V' [V love] [NP e_j]]]]]])
.....

Figure 4: The Structure of the PAD Parsers.

The clause defining the 'parse' relation given in Figure 4 as part of the axiomatization of GB theory is the actual Prolog definition of 'parse' used in the PAD1 and PAD2 parsers. Thus the top-level structure of the knowledge of language employed by the PAD parsers mirrors the top-level structure of GB theory.

Ideally the internal structure of the various principles of grammar should reflect the internal organization of the principles of GB (e.g. Case assigment should be defined in terms of Government), but for simplicity the principles are axiomatized directly here. For reasons of space a complete description of the all of the principles is not given here; however a sketch of one of the principles, the Case Filter, is given in the remainder of this section. The other principles are implemented in a similiar fashion.

The Case Filter as formulated in PAD applies recursively throughout the S-structure, associating each node with one of the three atomic values *ass*, *rec* or *0*. These values represent the Case properties of the node they are associated with; a node associated with the property *ass* must be a Case assigner, a node associated with the property *rec* must be capable of being assigned Case, and a node associated with the property *0* must be neutral with respect to Case. The Case Filter determines if there is an assignment of these values to nodes in the tree consistent with the principles of Case assignment. A typical assignment of Case properties to the nodes of an S-structure in English is shown in 5, where the Case properties of a node are depicted by the boldface annotations on that node.[1]

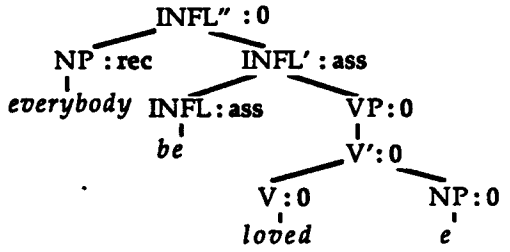

Figure 5: Case Properties.

The Case Filter is parameterized with respect to the predicates 'rightwardCaseAssignment' and 'leftwardCaseAssignment'; if these are specified as parameter settings of the language concerned, the Case Filter permits Case assigners and receivers to appear in the relevant linear order. The lexicon contains definitions of the one-place predicates 'noCase', 'assignsCase' and 'needsCase' which hold of lexical items with the relevant

1 These annotations are reminiscent of the complex feature bundles associated with categories in GPSG (Gazdar et. al. 1986). The formulation here differs from the complex feature bundle approach in that the values associated with nodes by the Case Filter are not components of that node's category label, and hence are invisible to other principles of grammar. Thus this formulation imposes an *informational encapsulation* of the principles of grammar that the complex feature approach does not.

property; these predicates are used by the Case Filter to ensure the associations of Case properties with lexical items are valid.

Specifically, the Case Filter liscences the following structures:

(2a) a constituent with no Case properties may have a Case assigner and a Case receiver as daughters iff they are in the appropriate order for the language concerned,

(2b) a constituent with no Case properties may have any number of daughters with no Case properties,

(2c) a constituent with Case property C may be realized as a lexical item W if W is permitted by the lexicon to have Case property C, and

(2d) INFL' assign Case to its left if its INFL daughter is a Case assigner.

This axiomatization of Universal Grammar together with the parameter values and lexicon for English is used as the axiom set of a Prolog interpreter to produce the parser called PAD1. Its typical behaviour is shown below.[2]

```
:parse([everybody, - s, love, somebody],  LF)
LF = everybody::i^somebody::j^infl2:[np:i,
   infl1:[infl: # (- s), vp:[v1:[v: # love, np:j]]]]
LF = somebody::j^everybody::i^infl2:[np:i,
   infl1:[infl: # (- s), vp:[v1:[v: # love, np:j]]]]
No (more) solutions

:parse([harry, be, loved],  LF)
LF = infl2:[np: # harry, infl1:[infl: # be,
        vp:[v1:[v: # loved, np:[]]]]]
No (more) solutions
```

AN ALTERNATIVE CONTROL STRUCTURE

Because it uses the SLD inference control strategy of Prolog with the axiomatization of Universal Grammar shown above, PAD1 functions as a 'generate and test' parser. Specifically, it enumerates all D-structures that satisfy X'-theory, filters those that fail to satisfy Θ-theory, computes the corresponding S-structures using Move-α, removes all S-structures that fail to satisfy the Case Filter, and only then determines if the terminal string of the S-structure is the string it was given to parse. Since the X' principle admits infinitely many D-structures the resulting procedure is only a semi-decision procedure, i.e. the parser is not guaranteed to terminate on ungrammatical input.

Clearly the PAD1 parser does not use its knowledge of language in an efficient manner. It would be more efficient to co-routine between the principles of grammar, checking each existing node for well-formedness with respect to these principles and ensuring that the terminal string of the partially constructed S-structure matches the string to be parsed before creating any additional nodes. Because the Parsing as Deduction framework conceptually separates the knowledge used by the processor from the manner in which that knowledge is used, we can use an inference control strategy that applies the principles of grammar in the manner just described. The PAD2 parser incorporates the same knowledge of language as PAD1 (in fact textually identical), but it uses an inference control strategy inspired by the 'freeze' predicate of Prolog-II (Cohen 1985, Giannesini et. al. 1986) to achieve this goal.

The control strategy used in PAD2 allows inferences using specified predicates to be delayed until specified arguments to these predicates are at least partially instantiated. When some other application of an inference rule instantiates such an argument the current sequence of inferences is suspended and the delayed inference performed immediately. Figure 6 lists the predicates that are delayed in this manner, and the argument that they require to be at least partially instantiated before inferences using them will proceed.

Predicate	Delayed on
X' theory	D-structure
Θ theory	D-structure
Move-α	S-structure
Case Filter	S-structure
Phonology	not delayed
LF-Movement	S-structure

Figure 6: The Control Strategy of PAD2.

With this control strategy the parsing process proceeds as follows. Inferences using the X', Θ,

[2] For the reasons explained below, the X' principle used in this run of parser was restricted to allow only finitely many D-structures.

Case, Move-α and LF-movement principles are immediately delayed since the relevant structures are uninstantiated. The 'phonology' principle (a simple recursive tree-walking predicate that collects terminal items) is not delayed, so the parser begins performing inferences associated with it. These instantiate the top node of the S-structure, so the delayed inferences resulting from the Case Filter, Move-α and LF-movement are performed. The inferences associated with Move-α result in the instantiation of the top node(s) of the D-structure, and hence the delayed inferences associated with the X' and Θ principles are also performed. Only after all of the principles have applied to the S-structure node instantiated by the 'phonology' relation and the corresponding D-structure node(s) instantiated by Move-α are any further inferences associated with the 'phonology' relation performed, causing the instantiation of further S-structure nodes and the repetition of the cycle of activation and delaying.

Thus the PAD2 parser simultaneously constructs D-structure, S-structure and LF representations in a top-down left-to-right fashion, functioning in effect as a recursive descent parser. This top-down behaviour is not an essential property of a parser such as PAD2; using techniques based on those described by Pereira and Shieber (1987) and Cohen and Hickey (1987) it should be possible to construct parsers that use the same knowledge of language in a bottom-up fashion.

TRANSFORMING THE AXIOMATIZATION

In this section I sketch a program transformation which transforms the original axiomatization of the grammar to an equivalent axiomatization that in effect exhibits this 'co-routining' behaviour when executed using Prolog's SLD inference control strategy. Interestingly, a data-flow analysis of this transformed axiomatization (viewed as a Prolog program) justifies a further transformation that yields an equivalent program that avoids the construction of D-structure trees altogether. The resulting parsers, PAD3 – PAD5, use the same parameter settings and lexicon as PAD1 and PAD2, and they provably compute the same PF–LF relationship as PAD2 does. The particular techniques used to construct these parsers depend on the internal details of the formulation of the principles of grammar adopted here – specifically on their simple recursive structure – and I do not claim that they will generalize to more extensive formulations of these principles.

Recall that the knowledge of a language incorporated in PAD1 and PAD2 consists of two separate components, (i) parameter values and a lexicon, and (ii) an axiomatization $\mathcal{U}$ of the theory of Universal Grammar. The axiomatization $\mathcal{U}$ specifies the deductively closed set of statements that constitute the theory of Universal Grammar, and clearly any axiomatization $\mathcal{U}'$ equivalent to $\mathcal{U}$ (i.e. one which defines the same set of statements) defines exactly the same theory of Universal Grammar. Thus the original axiomatization $\mathcal{U}$ of Universal Grammar used in the PAD parsers can be replaced with any equivalent axiomatization $\mathcal{U}'$ and the system will entail exactly the same knowledge of the utterances of the language. A deductive parser using $\mathcal{U}'$ in place of $\mathcal{U}$ may perform a difference sequence of inference steps but ultimately it will infer an identical set of consequences (ignoring non-termination).

The PAD3 parser uses the same parameter values and lexicon as PAD1 and PAD2, but it uses a reaxiomatization of Universal Grammar obtained by applying the Unfold/Fold transformation described and proven correct by Tamaki and Sato (1984) and Kanamori and Horiuchi (1988). Essentially, the Unfold/Fold transformation is used here to replace a sequence of predicates each of which recursively traverses the same structure by a single predicate recursive on that structure that requires every node in that structure to meet all of the constraints imposed by the original sequence of predicates. In the PAD3 parser the X', Θ, Move-α, Case and Phonology principles used in PAD1 and PAD2 are folded and replaced by the single predicate 'p' that holds of exactly the D-structure, S-structure PF triples admitted by the conjunction of the original principles.

Because the reaxiomatization technique used here replaces the original axiomatization of PAD1 and PAD2 with an equivalent one (in the sense of the minimum Herbrand model semantics), the PAD3 parser provably infers

exactly the same knowledge of language as PAD1 and PAD2. Because PAD3's knowledge of the principles of grammar that relate D-structure, S-structure and PF is now represented by the single recursive predicate 'p' that checks the well-formedness of a node with respect to all of the relevant principles, PAD3 exhibits the 'co-routining' behaviour of PAD2 rather than the 'generate and test' behaviour of PAD1, even when used with the standard SLD inference control strategy of Prolog.[3]

PAD3 constructs D-structures, just as PAD1 and PAD2 do. However, a simple analysis of the data dependencies in the PAD3 program shows that in this particular case no predicate uses the D-structure value returned by a call to predicate 'p' (even when 'p' calls itself recursively, the D-structure value returned is ignored). Therefore replacing the predicate 'p' with a predicate 'p1' exactly equivalent to 'p' except that it avoids construction of any D-structures does not affect the set of consequences of these axioms.[4] The PAD4 parser is exactly the same as the PAD3 parser, except that it uses the predicate 'p1' instead of 'p', so it therefore computes exactly the same PF – LF relationship as all of the other PAD parsers, but it avoids the construction of any D-structure nodes. That is, the PAD4 parser makes use of exactly the same parameter settings and lexicon as the other PAD parsers, and it uses this knowledge to compute exactly the same knowledge of utterances. It differs from the other PAD parsers in that it does not use this knowledge to explicitly construct a D-structure representation of the utterance it is parsing.

This same combination of the Unfold/Fold transformation followed data dependency analysis can also be performed on all of the principles of grammar simultaneously. The Unfold/Fold transformation produces a predicate in which a data-dependency analysis identifies both D-structure and S-structure values as ignored. The PAD5 parser uses the resulting predicate as its axiomatization of Universal Grammar, thus PAD5 is a parser which uses exactly the same parameter values and lexicon as the earlier parsers to compute exactly the same PF–LF relationship as these parsers, but it does so without explictly constructing either D-structures or S-structures.

To summarize, this section presents three new parsers. The first, PAD3, utilized a re-axiomatization of Universal Grammar, which when coupled with the SLD inference control strategy of Prolog resulted in a parser that constructs D-structures and S-structures 'in parallel', much like PAD2. A data dependency analysis of the PAD3 program revealed that the D-structures computed were never used, and PAD4 exploits this fact to avoid the construction of D-structures entirely. The techniques used to generate PAD4 were also used to generate PAD5, which avoids the explicit construction of both D-structures and S-structures.

CONCLUSION.

In this paper I described several deductive parsers for GB theory. They knowledge of language that they used incorporated the top-level structure of GB theory, thus demonstrating that parsers can actually be built that directly reflect the structure of this theory.

This work might be extended in several ways. First, the fragment of English covered by the parser could be extended to include a wider range of linguistic phenomena. It would be interesting to determine if the techniques described here to axiomatize the principles of grammar and to reaxiomatize Universal Grammar to avoid the construction of D-structures could be used on this enlarged fragment – a program transformation for reaxiomatizing a more general formulation of Move-α is given in Johnson (1988b).

Second, the axiomatization of the principles of Universal Grammar could be reformulated to incorporate the 'internal' deductive structure of

---

3 Although in terms of control strategy PAD3 is very similiar to PAD2, it is computationally much more efficient than PAD2, because it is executed directly, whereas PAD2 is interpreted by the meta-interpreter with the 'delay' control structure.

4 The generation of the predicate 'p1' from the predicate 'p' can be regarded an example of static garbage-collection (I thank T. Hickey for this observation). Clearly, a corresponding run-time garbage collection operation could be performed on the nodes of the partially constructed D-structures in PAD2.

the components of GB theory. For example, one might define c-command or goverment as primitives, and define the principles in terms of these. It would be interesting to determine if a deductive parser can take advantage of this internal deductive structure in the same way that the PAD parsers utilized the deductive relationships between the various principles of grammar.

Third, it would be interesting to investigate the performance of parsers using various inference control strategies. The co-routining strategy employed by PAD2 is of obvious interest, as are its deterministic and non-deterministic bottom-up and left-corner variants. These only scratch the surface of possibilities, since the Parsing as Deduction framework allows one to straightforwardly formulate control strategies sensitive to the various principles of grammar. For example, it is easy to specify inference control strategies that delay all computations concerning particular principles (e.g. binding theory) until the end of the parsing process.

Fourth, one might attempt to develop specialized logical languages that are capabale of expressing knowledge of languages and knowledge of utterances in a more succinct and computationally useful fashion than the first-order languages.

BIBLIOGRAPHY

Berwick, R. (1987) *Principle-based Parsing.* MIT Artificial Intelligence Laboratory Technical Report No. 972. Also to appear in *The Processing of Linguistic Structure,* The MIT Press, Cambridge, Mass.

Berwick, R. and A. Weinberg. (1984) *The Grammatical Basis of Linguistic Performance.* The MIT Press, Cambridge, Mass.

Chomsky, N. (1981) *Lectures on Government and Binding.* Foris, Dordrect.

Chomsky, N. (1986) *Knowledge of Language, Its Nature, Origin and Use.* Praeger, New York.

Cohen, J. (1985) Describing Prolog by its Interpretation and Compilation. C. *ACM.* 28:12, p. 1311-1324.

Cohen, J. and T. Hickey. (1987) Parsing and Compiling Using Prolog. *ACM Trans. Programming Languages and Systems.* 9:2, p. 125-163.

Gazdar, G., E. Klein, G. Pullum and I. Sag. (1985) *Generalized Phrase Structure Grammar.* Basil Blackwell, Oxford.

Giannesini, F., H. Kanoui, R. Pasero, and M. v. Caneghem. *Prolog.* Addison-Wesley, Reading, Mass. 1986.

Johnson, M. (1988a) Parsing as Deduction, the Use of Knowledge of Language, ms.

Johnson, M. (1988b) Computing with Move-α using the Unfold-Fold Transformation, ms.

Kanamori, T. and K. Horiuchi (1988) Construction of Logic Programs Based on Generalized Unfold/Fold Rules, in Lassez, ed., *Proceedings of the Fourth International Conference of Logic Programming,* p. 744 - 768, The MIT Press, Cambridge, Mass.

Pereira, F. and S. Shieber. (1987) *Prolog and Natural Language Processing.* CSLI Lecture Notes Series, distributed by Chicago University Press. Chicago.

Pereira, F. and D. Warren. (1983) Parsing as Deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics,* MIT, Cambridge, Mass.

Tamaki, H. and T. Sato. (1984) Unfold/Fold Transformation of Logic Programs. In *Proceedings of the Second International Logic Programming Conference,* p. 127-138, Uppsala University, Uppsala, Sweden.

Van Riemsdijk, H. and E. Williams. (1986) *Introduction to the Theory of Grammar.* The MIT Press, Cambridge, Mass.