Principle-based Parsing and Logic Programming

Matthew W. Crocker
Centre for Cognitive Science
University of Edinburgh
2 Buccleuch Place
Edinburgh, UK EH8 9LW
E-mail: mwc@cogsci.ed.ac.uk

Keywords: Natural language processing, computational linguistics, deductive parsing

Edited by:

Received: **Revised:** **Accepted:**

While deductive parsing techniques are well-understood for traditional rule-based and lexicalist grammars, they are rather more elusive for current principle-based grammars. In this paper, we argue that a major source of difficulty arises from a fundamental difference in the way such grammars should be axiomatised. While rule-based grammars typically consist of a set of sufficient *'structure-generating' axioms, principle-based grammars are more naturally expressed as a set of* necessary *'structure-licensing' conditions. On this basis we propose a methodology for implementing deductive parsers which is more suitable for this class of 'licensing grammars'. We then argue that current principle-based grammatical theories can be most naturally implemented by decomposing them into representationally homogeneous subsystems, which are axiomatised as licensing 'sub-grammars' – each contributing its own aspect of a global syntactic deduction system. Finally we consider the new range of options this approach offers for developing flexible and possibly distributed control regimes.*

1 Introduction

Syntactic constraints form an important source of knowledge in any natural language processing (NLP) system, be it intended for a practical application requiring deep interpretation, or as theoretical or cognitive model for current theories.[1] However, modern 'principle-based' linguistic theories are complex, abstract, and formally underspecified, making them difficult to incorporate within current NLP systems. To begin exploiting these linguistic developments in computational systems, we require a rigorous framework in which we can formally represent such grammars and then construct systems which use these grammar respresentations transparently. We argue that the logic programming paradigm provides such a framework.

However, while phrase structure grammars are highly amenable to axiomatisation in horn-clause logic, and lexicalist frameworks such as categorial grammar have well-defined logical interpretations, this paper considers how more heterogeneous, principle-based linguistic theories might benefit similarly from the logic programming paradigm. We demonstrate that while deductive parsing techniques can be fruitfully applied to principle-based or 'licensing' grammars, several interesting differences emerge. First, we show that the correspondence between a syntactic analysis and a proof must be reconsidered, and the deductive techniques revised. Secondly, we argue that such grammatical theories may be most effectively axiomatised as several sub-grammars in which particular constraints are defined over the simple representational types to which they apply. We then consider the motivations and implications of such axiomatisations for efficient implementation of principle-based parsing systems.

[1] This is the case regardless of whether or not a syntactic representation is explicitly constructed. That is, such constraints may be used to directly map from an input string to, say, some meaning representation language.

The relationship between logic programming and language processing dates back to the origins of Prolog. Indeed, Definite Clause Grammars (DCGs), now a component of virtually all Prolog implementations, provide a notation explicitly for writing phrase structure grammars. Once these grammar rules are compiled into standard Prolog[2] they receive a procedural interpretation, becoming a top-down, left-to-right, recursive-descent parser. That is to say, by representing the rules of grammar as axioms in Prolog's horn-clause logic, we can use Prolog's theorem proving engine as a parser.

This natural embedding of phrase structure grammars in Prolog suggests that we should be able to cast natural language parsing into the broader logic programming paradigm. That is to say, distinction between *logic* and *control* within logic programming may be naturally inherited by the parsing domain. This constitutes the so-called *Parsing as Deduction* (PAD) hypothesis, wherein a parser is defined as a logical specification of a grammar, in conjunction with some deductive inference engine which realises a particular parsing algorithm for that grammar (Pereira and Warren, 1983) (Pereira and Shieber, 1987). So just as there is a direct correspondence between a grammar and its logical representation as horn clauses, so is there a similar correspondence between the logical inferencing (or theorem proving) strategy and the parsing algorithm. A further, and important, correspondence is that for such logic grammars, the 'proof' that a sentence is indeed a theorem of the grammatical axioms is precisely the parse tree for that sentence.

If we adopt the PAD methodology, the task of developing a parser can be simply broken down into (a) the provision of a logical specification of our grammar, and (b) the construction of a suitable theorem prover. The latter of course, is dependent upon the richness of the logic we have used, and may conceivably take advantage of specific properties of our grammar axiomatisation.[3] While such deductive parsing techniques are well-understood for traditional rule-based, phrase structure grammars, they are rather more elusive for the current wave of principle-based or constraint-based grammars. In this paper, we argue that a major source of difficulty arises from a fundamental difference in the way such grammars should be axiomatised. While rule-based grammars typically consist of a set of *sufficient* 'structure-generating' axioms, principle-based grammars are more naturally expressed as a set of *necessary* 'structure-licensing' conditions which in essence rule-out ill-formed structures, rather than generating well-formed ones. On the basis of this observation we propose a new methodology for implementing deductive parsers for this class of licensing grammars.

We begin below with a brief discussion of principle-based parsing, highlighting the tendency to adapt traditional rule-oriented parsing technology. In particular, we stress that the standard notions of parsing are inherently biased towards the homogeneous nature of rule-based grammars, and are inappropriate given current abstract licensing grammars which consist of a small set of interacting principles/constraints – typified by the principles and parameters paradigm, e.g. *Government-Binding Theory* (Chomsky, 1981) (Lasnik and Uriagereka, 1988). We then cast the parsing problem in deductive terms, to provide a more formal foundation for our discussion. In particular, we observe that construction-oriented, rule-based grammars (such as CFGs, etc.) typically consist of structure specific rules, where one rule is sufficient to license or indeed generate a particular instance of syntactic structure. In the context of licensing grammars, however, particular instances of syntactic structure are often required to meet a number of more abstract conditions which interact.

We will then examine some previous approaches to implementing deductive, principle-based parsers, and the key problems they face. In particular, we consider the relationship between a syntactic analysis and the deductive proof. While the relationship is direct for phrase structure grammars, this is not the case for the current principle-based accounts in which principles in-

[2]The translation from DCG form into Prolog is a trivial one, which requires the simple addition of string-handling difference-lists to the original rules. For a thorough exposition of DCG implementation see (Pereira and Warren, 1980), and for other logic programming formalisms see (Abramson and Dahl, 1989).

[3]Consider, for example, that the inference procedure used by Prolog to parse DCGs restricts one not only to grammars that may be specified as horn-clauses, but also to grammars that contain no left-recursive rules.

teract combinatorially to license annotated structural representations. In contrast with some existing deductive, principle-based techniques, we advocate an approach which *decomposes* syntactic analysis into several uniform representation types. This reduces the logical and deductive complexity of the component systems, permitting the use of simple and well-understood techniques. Within this approach, a complete syntactic analysis consists of a 'tuple' of proofs; each corresponding to a particular representational aspect of the overall analysis. We demonstrate the application of this technique by constructing a grammar fragment which distinguishes the recovery of local and hierarchical constituent structure (or *phrase structure*), from the determination of long distance relationships, called *chains*.

1.1 Parsing with Principles

The computational linguistics community has recently exhibited increasing interest in the development of systems based on the principles and parameters model of current linguistic theory.[4] The adoption of this paradigm remains rather tentative, however, for several reasons. First, principle-based syntactic theories such as GB theory are typically complex, unstable, and – by computational standards – informal. Thus there is no readily available formal specification of what a GB grammar is (but cf. (Stabler,1992)). Secondly, there has been a reluctance to abandon the traditional parsing technology – centered around phrase structure grammars (or, equivalents) – which is rendered largely inadequate by the modular, heterogeneous and abstract nature of principle-based theories. The recent trend towards statistical parsing techniques, has also shifted emphasis away from the explicit formalisation of linguistic knowledge, to its 'approximation' (and acquisition) by data-intensive, stochastic techniques (though Fordham and Crocker (1997) consider how the principle-based and stochastic techniques might be combined).

There are, however, a number of arguments in favour of pursuing principle-based systems. The most obvious is that it allows the exploitation of 'state of the art' syntactic theorising. In this way we might also contribute to the formalisation of syntactic theory as it develops (see (Stabler, 1992) in particular). From an 'engineering' perspective, there are other potential advantages; principle-based systems are much more compact and may well be easier to maintain than construction-based systems. While the interaction of principles is typically more complex, the principles themselves are relatively simple and prohibit the 'yet another special rule' approach which is rife in the development of construction-based systems. Furthermore, given the typically large numbers of rules involved in the latter, there is no explicit notion of an underlying theory which can be used to justify particular rules, while principles must remain consistent with the overall theory of grammar. A further appeal is that principle-based systems may be designed to apply cross-linguistically, sharing the fundamental grammar and parsing machinery, while construction based systems are inherently language specific.

1.2 Parsing as Deduction

In an effort to construct more faithful and transparent realisations of principle-based systems, there has been recent interest in so-called 'deductive' parsing methods. As we sketched above, this approach explicitly separates the axiomatisation of grammatical principles – a purely declarative specification – from the procedures which use these axioms to 'prove' derivations of syntactic analyses. In particular it has been shown that meta-interpreters or program transformations can be used to affect the manner in which a logic grammar is parsed (Pereira and Warren, 1983), while leaving the grammar unaltered (or logically equivalent). That is, for a given logic grammar we can construct a variety of parsers such as LL, LR, and Earley, among others. One problem, however, is that not all parsing strategies are suited to all grammars. A recursive descent parser, for example, may not terminate for a grammar with left recursive rules, while bottom-up parsers are typically unsuitable for grammars with empty productions (Pereira and Shieber (1987) provide a thorough exposition of these issues).

One attempt to extend the PAD hypothesis be-

[4] See Berwick et al (1991) for a good introduction, and a collection of papers on various systems and approaches. Additional systems are presented by Crocker (1991b) and Merlo (1995).

yond its application to simple phrase structure logic grammars is presented by Johnson (1989). In particular, Johnson has developed a prototype parser for a fragment of a GB grammar. The system consists of a declarative specification of the GB model, which incorporates the various principles of grammar and multiple levels of representation. His top-level axiomatisation is as follows:

```
(1) parse(String,LF) :–
                    xBar(infl2,DS),
                    theta(infl2,0,DS),
                    moveAlpha(DS,[],SS,[]),
                    caseFilter(info2,0,SS),
                    phonology(String/[],SS),
                    lfMovements(SS,LF).
```

This specification transparently encodes the standard T-model of transformational syntax: xBar and theta instantiate well-formed D-structure (DS) representations, moveAlpha then transforms these into candidate S-structures (SS), which are in turn mapped onto phonetic (in this case a String) form and logical form (LF). When trying to compute this relation using Prolog's default control mechanism, however, this system has obvious problems. The result will be a naïve generate and test parser, since D-structure and S-structure will first be generated by the grammar and only then matched to the String by the phonology predicate. Furthermore, if xBar were to contain left recursive axioms (Johnson's does not, but this would be required for a more complete axiomatisation) that predicate alone might never halt.

It is at this point that the declarative interpretation of logic programs becomes of practical use. Crucially, Johnson illustrates how the fold/unfold transformation, when (manually) applied to various components of the grammar, can be used to render more efficient implementations derived from the sort of axiomatisation given above. This approach effectively reaxiomatises the grammar into a system more amenable to Prolog's control and inference regimes.[5] Roughly speaking, this can be viewed as a step in the direction of *partially evaluating* a principle-based system into a set of phrase structure rules (although Johnson does not advocate such a move); moving from an abstract specification to a more concrete or 'compiled-out' form (for discussion of this in a non-deductive context see (Merlo, 1995)).

While the transformation approach may be a practical solution for constructing efficient parsers, it loses the appeal of a system which directly exploits a compact, modular system of principles *on-line*. Also, every time a change is made to the underlying grammar, it will need to be re-transformed, and the nature of these transformations may need to be revised for the new grammar. As an alternative, Johnson also demonstrates how goal *freezing*, an alternative Prolog control strategy, can be used to increase efficiency by effectively coroutining the recovery of the various levels of representation, allowing all principles to be applied as soon as possible, at all levels of representation. While attractive, this approach is not without its difficulties; the success of coroutining relies not only on the careful encoding of the representations, but efficiency and indeed termination properties will depend on precisely how and when the various principles apply to successfully constrain what is essentially an *informed* generate and test procedure.

In sum, the deductive approach to parsing is theoretically attractive, but unsurprisingly inherits a number of problems with automated deduction in general. Real automated theorem provers are, at least in the general case, incomplete. That is, they cannot *a priori* be guaranteed to return all (or any) solutions to a given request. One instance of this is the left-recursion example cited above; a perfectly legitimate grammar rule may cause the Prolog inference engine to pursue an infinite path and never halt. While it is possible to solve or at least detect some of these problems, especially for parsing algorithms and grammars formalisms which are well understood,[6] it is certainly not possible in the general case. We can therefore imagine that a true, deductive implementation of GB would present a problem. Unlike traditional, homogeneous phrase structure grammars, GB makes use of abstract, modular principles, each of which may be relevant to only a particular type or level of representation. This modular, heterogeneous organisation makes the task of

[5] See (Johnson, 1991) for further discussion of such techniques.

[6] As another example, many bottom-up algorithms cannot be guaranteed to succeed if there are empty productions in the grammar.

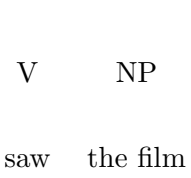
deriving some single, specialised interpreter with adequate coverage and efficiency a very difficult one. For further illumination of these issues, the reader is referred to (Stabler, 1992), although that work is not specifically directed at parsing concerns.

2 Rules *versus* Principles

The traditional characterisation of language in terms of construction-based systems (i.e. systems which use individual rules to describe units of structure, as exemplified by a context-free grammar) has significantly influenced our views about parsing. In particular, there is a tendency for parsers to use the rules of grammar to 'generate' instances of structure. This is possible because of the homogeneous nature of construction-based grammars, where a single rule is sufficient to license a particular unit of syntactic structure.

This property of 'rule-to-structure' correspondence does not obtain in principle-based grammars, however, where particular units of structure must satisfy a collection of relevant principles. Furthermore, any given principles might only be concerned with a particular aspect of that structure. Consider, for example, the following VP structure:

(2)



In a typical phrase structure grammar, this structure (excluding the subtrees of V and NP, and henceforth written as $[_{V^2}\ V^0\ N^2]$) is licensed by the single rule; VP → V NP. In a principle-based grammar, the structure is only well-formed if it satisfies a number of principles; $\overline{X}$-theory licenses the basic structural configuration (i.e. the complement NP as sister to the V^{min} projection, and dominated by a higher V projection, namely VP), the θ-criterion is satisfied since the NP both requires a thematic (θ-)role and occupies a θ-marked position, and finally, the NP must satisfy the Case filter (which it does, due to the transitive verb). Given that the principles are each concerned with only a particular aspect of the syntactic structure, none are particularly appropriate for generating the structure. In sum, principles are more naturally viewed as constraints on syntactic structures, rather than generators of them.

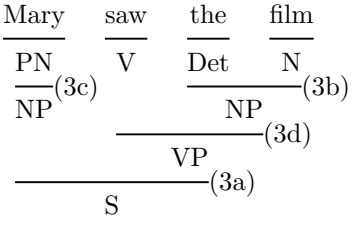
The approach adopted in most existing principle-based parsers is to treat the rules of $\overline{X}$-theory as structure generators, and then apply the principles as constraints on these structures (e.g. see (Crocker, 1991b) and also the discussion of other systems by Berwick (1991)). This technique has a number of potential disadvantages, however. In the first place, $\overline{X}$-theory is a principle of D-structure in most current instantiations of the theory, and hence is not sufficient for generating the set of possible S-structure configurations.[7] Secondly, there has been an increase in support for the abolition of $\overline{X}$ 'rules' *per se*, favouring feature-based constraints derived from more fundamental properties of lexical items, e.g. (Speas, 1990).

The fundamental difference between rule- and principle-based systems – i.e. rules as 'generators' versus principles as 'constraints' – is made more precise when cast in terms of deductive parsing. To begin, consider the following (horn clause) axiomatisation of a simple context-free grammar:

(3)
(a) S ← NP ∧ VP
(b) NP ← Det ∧ N
(c) NP ← PN
(d) VP ← V ∧ NP
(e) PN ← "Mary"
(f) Det ← "the"
(g) N ← "film"
(h) V ← "saw"

Now we can illustrate the derivation or 'proof' of a sentence S for the string *Mary saw the film*, as follows:

(4)



Borrowing a notation widely used in the Categorial Grammar literature (as introduced by Ades

[7] That is, surface structures may be composed of both the configurations licensed by $\overline{X}$-theory and those which result from transformations, such as adjunction.

and Steedman (1982)), this derivation illustrates a complete proof of the theorem: S ← *Mary saw the film*, derived from the axioms given in (3).[8] Furthermore, the derivation transparently represents the (inverted) phrase structure for the sentence, where derivation steps are translated into branches connecting the consequence (below the derivation line) to it premises (above the derivation line) – this follows from the fact that the rules/axioms directly characterise the well-formed units of structure, as discussed above.

Note, it is possible to construct our derivation such that we record the proof/constituent structure as we go along. This is accomplished by writing the derived consequence X as [$_X$ Prem1 ... PremN], where each premise in turn is the structure of the sub-proof:

(5)

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{Mary}}{\text{PN}}\quad\dfrac{\text{saw}}{\text{V}}\quad\dfrac{\dfrac{\text{the}}{\text{Det}}\quad\dfrac{\text{film}}{\text{N}}}{[_{NP}\ \text{Det N}]}(3b)}{[_{VP}\ \text{V}\ [_{NP}\ \text{Det N}]]}(3d)}{[_{NP}\ \text{PN}]}(3c)}{[_S\ [_{NP}\ \text{PN}]\ [_{VP}\ \text{V}\ [_{NP}\ \text{Det N}]]]}(3a)$$

Thus the final derivation is in fact a bracketed list encoding the structure of the original derivation in (4).[9] Crucially, while it is possible to carry the sub-proof as we construct the derivation (as in (5)), this structure is not used or referenced by the axioms of grammar (indeed, (4) does not build such a proof record). In early transformational grammar, however, it was possible write such structure sensitive rules, as in the case of the passive[10]:

(6) [$_S$ NP^1 [$_{VP}$ V NP^2]] ← [$_S$ NP^2 [$_{VP}$ V [$_{PP}$ by NP^1]]]

As we discussed above, current conceptions of the phrase-structure component within the principles and parameters approach are heading away from such a rule-oriented characterisation. This is exemplified by the *Project Alpha* proposal of Speas, where the projection of structure from the lexicon is free, and only subsequently constrained by the syntax (Speas, 1990). This move is essentially the final step in eliminating the rule component in favour of a pure 'licensing' grammar. In sum, the view is one where syntax simply constrains (or, licenses) virtually arbitrary structures, rather than generating them.

Our approach is to characterize the well-formed structures of a language in two stages. First, we define a set of possible structures against which the grammatical principles can apply. Secondly, we specify the grammatical principles which pick out those members of the set of possible structures which are, in fact, well-formed. What are the possible structures? Here, we begin with the view that they are simply binary branching trees. We will further assume that the principles that apply to them are strictly *local* and are defined exclusively with respect to local structure units. That is, our principles will not apply to whole trees but only to branches of trees, e.g. [$_{V^2}$ V^0 N^2]. This restriction permits the close interleaving of the two stages outlined above; as each new binary branch is proposed during the construction of a tree, we can immediately verify the local well-formedness of that branch (Crocker (1991a, 1996) provide more detailed discussion and illustration of this point).

It might be objected that a complete axiomatisation incorporating movement would entail the use of *tree transducing* axioms, permitting the description and constraint of Move-α.[11] This would eliminate the strictly local characterisation of the principles.[12] In recent work, however, Crocker (1992,1996) develops a 'representational' reformulation of the transformational model which de-

[8] In fact, the reader may have noticed that linear order is implicit in this system. That is, NP ← Det ∧ N, implies that Det is adjacent to, and precedes, N. Furthermore, any theorem (i.e. derived result) or premise (i.e. lexical item) may only be used once in the course of the derivation. This may be defined naturally within the general framework of a linear logic. For our purely illustrative purposes, however, a thorough and formal exposition of the assumed logical framework would take us too far afield.

[9] See (Stabler, 1987) for a discussion relating the proof procedure assumed in (4) with an equivalent one which records the proof tree, as in (5).

[10] Note, this transformational rule is a simplified reformulation of the standard passive rule in TG, where the premise is the 'structural description', and the consequence is the resulting 'structural change'.

[11] See (Stabler, 1992) for an elaborate exposition of this approach.

[12] Note, the approach presented here would still be applicable, but the axiomatisation would be rather more complicated, and the resulting computational complexity would be increased.

composes syntactic analysis into several representation types – including phrase structure, chains, and coindexation – allowing one to maintain the strictly local characterisation of principles with respect to their relevant representation type. This approach entails the use of multiple inference engines: one dedicated to the construction of each representation.

For expository purposes, we will first consider a single, phrase structure representation. This allows us to illustrate the general licensing approach for a simple grammar. We then consider the treatment of non-local dependencies, which is realised by introducing the representation of *chain structure*. We first extend the phrase structure system to allow empty categories, and a broader range of structures. We then show how the techniques developed for proving phrase structure 'theorems' for a given string can be similarly exploited to prove chain structure theorems for a given phrase structure. A complete, well-formed analysis is obtained precisely when we can prove a phrase structure which covers the string, and a chain structure which covers the phrase structure. Finally, we consider the issue of control in the context of our multiple deductive systems.

2.1 Licensing and Phrase Structure

Given our commitment to *local binary branches*, our procedure is the following: First, we define the set of possible binary branches. Secondly, we apply the principles of grammar to the branches thereby defining the set of proper branches, and thirdly, we define the set of well-formed binary branching trees from of the set of proper branches. This can be formalised straightforwardly as follows:

(7) I. Two sets of nodes:
(a) The set T of all terminals.
(b) The set NT of all non-terminals.

II. The set B of branches:
(a) if $X, Y, Z \epsilon NT$, then $[_X\ Y\ Z\ ] \epsilon B$
(b) if $X \epsilon NT$, $Y \epsilon T$, then $[_X\ Y\ ] \epsilon B$
(c) there is nothing else ϵB

III. The set PB of proper branches:
(a) $\alpha \epsilon PB$ iff $\alpha \epsilon B$, and
(b) α meets all necessary conditions in (8)

IV. The set Tr of well-formed trees:
(a) if $\alpha = [_X\ Y\ ] \epsilon PB$, then $\alpha \epsilon Tr$
(b) if $A_t, B_t \epsilon Tr$ and $[_X\ A\ B\ ] \epsilon PB$, then $X_t = [_X\ A_t\ B_t\ ] \epsilon Tr$
(Z_t denotes a tree rooted at the NT node Z)

To complete the definition, we now give an example of a rather simple set of principles, which state the conditions which are necessary for a branch ($b \in B$) to be a proper branch ($b \in PB$).[13] In the following definitions, X, Y, Z are variables over NT, Word is a variable over T and N, D, $V \in NT$.

(8) $\overline{X}$-theory:
(a)$[_{X^i}\ \mathrm{Y}^j\ \mathrm{Z}^k\ ] \rightarrow$
X = Z, i=j=2, k $\leq$ 1, Y is-spec-of X
or,
X = Y, j=0, k=2, 1 $\leq$ i $\leq$ 2.
(b) $[_{X^n}$ Word$] \rightarrow$ cat(Word,X), 2 $\geq$ n $\geq$ 0.

Case Filter:
(c) $[_Z\ \mathrm{X}\ \mathrm{N}^2\ ] \rightarrow$ case-assigner(X).

θ-Criterion:
(d) $[_Z\ \mathrm{X}^0\ \mathrm{Y}^2\ ] \rightarrow \theta$-marks($\mathrm{X}^0$,$\mathrm{Y}^2$).

Lexicon/Parameters:
(e) cat('the',D).
(f) cat('film',N).
(g) cat('saw',V).
(h) case-assigner(V).
(i) θ-marks(V^0,N^2).
(j) D is-spec-of N.

In these rules, the left-hand-side shows a structure which pattern-matches against structures in B. The right-hand-side states a necessary condition for the structure to be a proper branch. In contrast, our earlier CFG rules stated *sufficient* conditions for structures to be well-formed.

Given this set of axioms, consider the following proof of VP $\rightarrow$ *saw the film*:[14]

[13] The given definitions are only intended as simple approximations for the purposes of exposition, and are not to be construed as axiomatisations of the actual grammatical principles.

[14] We consider only a VP constituent for the moment, since a full sentence would require an axiomatisation of movement, which is not included in the current fragment. Precisely such a grammar is developed in the next section.

(9)

$$\dfrac{\dfrac{\text{saw}}{[_{V^0}\ \text{saw}]^{\{8b,g\}}} \quad \dfrac{\dfrac{\text{the}}{[_{D^2}\ \text{the}]^{\{8b,e\}}} \quad \dfrac{\text{film}}{[_{N^1}\ \text{film}]^{\{8b,f\}}}}{[_{N^2}\ \text{D}^2\ \text{N}^1]^{\{8a,j\}}}(7\text{IIa})}{[_{V^2}\ \text{V}^0\ \text{N}^2]^{\{8a,c,d,h,i\}}}(7\text{IIa})$$

In such a proof, simply drawing a line under one or more structures and writing a new structure underneath the line corresponds to constructing a branch using clause (7IIa) where X and Y are instantiated on the basis of the root node of each structure above the line (or by clause (7IIb), for the lexical cases). The superscript rule numbers next to each branch indicate which constraints are satisfied to show the branch is a proper branch (in accordance with (7III)). By virtue of (7IV) we can then construct the well-formed tree:

(10) $[_{V^2}\ [_{V^0}\ \text{saw}]\ [_{N^2}\ [_{D^2}\ \text{the}]\ [_{N^1}\ \text{film}]]]$

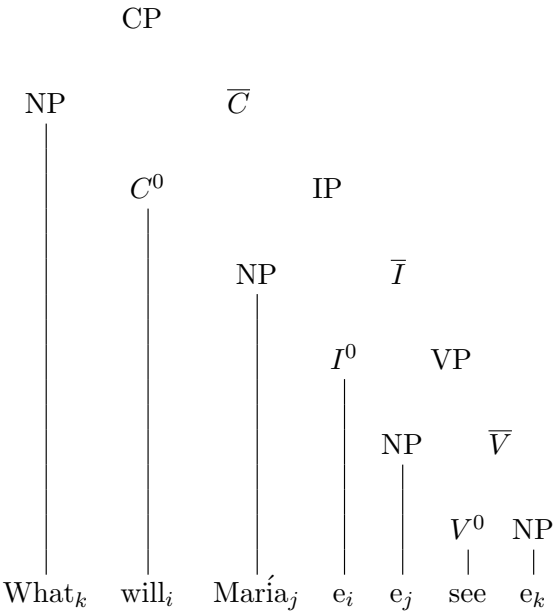
¿From this example, we can see that the axiomatisation and derivations for a principle-based – or, more properly, licensing – grammar are rather more complex than for a traditional 'construction-based' CFG. Most importantly, the derivation of a particular unit of structure is not supported by an individual (sufficient) axiom characterising precisely that construction, but must rather be consistent with the complete set of (necessary) axioms – each of which might only constrain some aspect of that unit of structure.

To summarise, the difference between rule-based and licensing grammars can be conceptually characterised as follows: Assuming that the task of a grammar is to define the set of well-formed syntactic analyses of a language, the rule-based grammar provides a set of sufficient conditions for the construction of particular units of structure (e.g. to construct an NP it is sufficient to have a determiner and a noun, NP ← Det N). In a licensing grammar however – and assuming that principle-based grammars are licensing grammars – the axioms do not 'generate' well-formed structures. Rather the axioms are necessary conditions which, given some structure, determine whether or not it is a candidate member of the set of well-formed formulae (wffs) and only once all the relevant axioms are satisfied can we be sure it is indeed a wff.

2.2 Licensing and Chains

It should now be clear that the licensing grammar introduced above is extremely limited in its potential coverage by the fact that only local (proper branch) dependencies can be expressed.[15] Analyses within current principle-based accounts often posit numerous such dependencies, even for superficially simple utterances. Consider the sentence *What will María see?*, which might reasonably be assigned an analysis as follows:

(11)



In comparison with the simple grammar fragment introduced above, such an analysis requires several additions. We need to introduce relevant constraints for the clausal, functional categories C and I including their valid head, complement and specifier selections. We need also permit NPs (and also I) to be empty, dominating *traces* which replace the lexical element which has moved out of that position. None of these additions is particularly onerous, and are achieved by extending the phrase structure constraints given in (8) as follows:

(12) $\overline{X}$-theory:
(a)$[_{X^i}\ \text{Y}^j\ \text{Z}^k\ ] \rightarrow$
X = Z, i=j=2, k ≤ 1, Y is-spec-of X

[15] It is of course possible to implement longer distance relations using feature passing techniques, such as GPSG's slash category. This will, however, compromise the efficiency of the system by introducing a greater degree of backtracking, i.e. proper branches will be licensed contingent upon features successfully being passed between mother and daughters.

or,
X = Y, j=0, k=2, $1 \leq i \leq 2$.

(b) $[_{X^n}$ Word] $\rightarrow$ cat(Word,X), $2 \geq n \geq 0$.

Case Filter:

(c) $[_Z$ V N^2] $\rightarrow$ case-assigner(V)
or,
$[_Z$ N^2 I] $\rightarrow$ case-assigner(I).

θ-Criterion:

(d) $[_Z$ X^0 Y^2] $\rightarrow$ θ-marks(X,Y^2)
or,
$[_Z$ Y^2 V^1] $\rightarrow$ θ-marks(V,Y^2).

Lexicon/Parameters:

(e) cat('What',N).
(f) cat('Maria',N).
(g) cat('see',V).
(h) cat('will',I).
(i) cat('will',C).
(j) cat(e,N).
(k) cat(e,I).
(l) case-assigner(V^0).
(m) case-assigner(I^1).
(n) θ-marks(V,N).
(o) θ-marks(I,V).
(p) θ-marks(C,I).
(q) N is-spec-of V.
(r) N is-spec-of I.
(s) N is-spec-of C.

The main additions to the system are: the introduction of N and I traces (12j,k),[16] the addition of the functional categories I and C and their relevant properties (12h,i,m,o,p,r,s),[17] and the introduction of NP specifiers of V (12q).[18] To simplify the system slightly, we have also removed determiners in favour of a simple proper noun (12f) and Wh-pronoun (12e). This revised grammar axiomatisation is now sufficient to prove the appropriate structure for CP $\rightarrow$ *What will María see*, shown in (14).

As before, we annotate each derivation with reference to the axioms required to license the corresponding branch of structure. Again by virtue of (7IV) we can construct the resulting well-formed tree,shown by the following bracketed list:

(13) $[_{C^2}$ $[_{N^2}$ What] $[_{C^1}$ $[_{C^0}$ will] $[_{I^2}$ $[_{N^2}$ María] $[_{I^1}$ $[_{I^0}$ e_1] $[_{V^2}$ $[_{N^2}$ e_2] $[_{V^1}$ $[_{V^0}$ see] $[_{N^2}$ e_3]]]]]]]

The are, however, two important points about the revised system. Firstly, while the above is the correct parse tree, it is an incomplete syntactic analysis. While the system successfully recovers the phrase structure in (11), the coindexations (e.g. between *what* and the trace in the verbs object position) are not represented (the subscripts on traces are purely for later identification, and not recovered by the system). Furthermore, the phrase structure axioms now substantially overgenerate. In particular, we could generate the above tree with or without traces in any of the positions, parse it as a CP or IP, and all such permutations. Clearly these two points are related, in that only that trees for which there are well-formed coindexations among moved constituents and traces, should receive a parse.

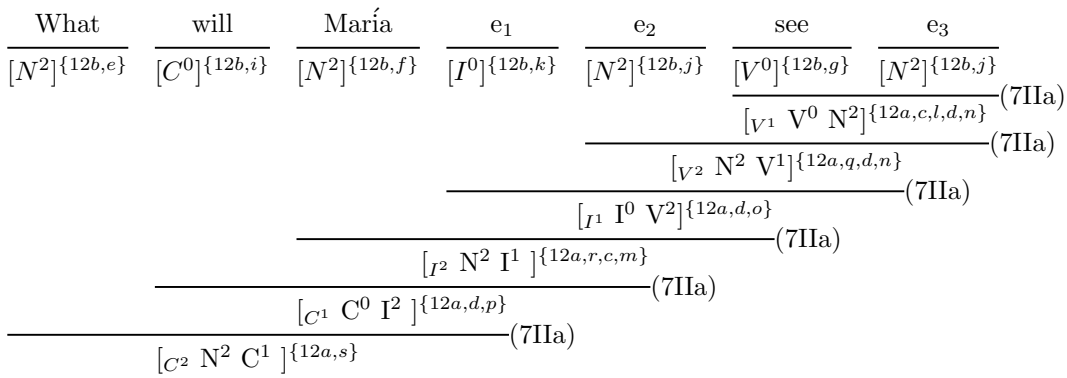
In this section we will argue that the correct solution to this problem is not to augment the phrase structure component described above. Such a move would complicate the simple representational schema for phrase structure, and weaken the notion of locality. That is, it would no longer be possible to define the necessary axioms in terms of a proper branch. Rather, we introduce a new type of licensing grammar, exclusively for the purpose of recovering a representation of long distance relationships among constituents. We will call this representation a *chain*, consisting of a list of constituents which enter into a well-formed long distance relation, and thus capture the coindexations shown in (11). Indeed, just as we have argued for the view of phrase structure as an abbreviated proof that a given string is a valid theorem w.r.t. the axioms over proper branches, we will now propose that the recovery of a chain structure constitutes a similarly abbreviated proof that a given phrase structure is a valid theorem, w.r.t the axioms over the *proper links* of chains. The result is a two-stage deductive parsing system which distinguishes the recovery of phrase structure from the recover of chains. The advantage of such an articulated system is that each component — phrase structure and chains — is kept simple, with axioms defined as strictly local, necessary constraints licensing branches and links, respectively.

[16] We assume the parsing engine treats *e* as the empty string appropriately. The prototype parser is a top-down one, so empty categories do not cause a problem. We return to this issue at the end of this section.

[17] We specify the head of I, *will*, also as head of C, since it may move to this position. This a simply a substitute for a more sophisticated treatment of head-movement.

[18] Our examples in this section will assume the VP-internal subject hypothesis that subjects are base-generated and hence θ-marked in the [Spec,VP] position (i.e. the specifier of VP), and move to the [Spec,IP] position to receive Case. Further, we assume that any Case/θ-marked NP can move to [Spec,CP] to form a Wh-question.

(14)



We begin as we did for our phrase structure system, by providing an axiomatisation for well-formed chains. In contrast with the binary branching tree representation of phrase structure, chains can be encoded as lists which contain one lexical antecedent as their head, followed by an arbitrary number of traces, and ending with *nil*. Just as the well-formedness of trees was recursively defined in terms of well-formed (proper) branches, so can well-formed chains be defined in terms of proper links. The details of this formulation are as follows:

(15) I. Two sets of nodes:
(a) The set L of all lexical nodes.
(b) The set NL of all non-lexical (empty) nodes.

II. The set L_{ink} of links:
(a) if $X \epsilon L$, $Y \epsilon NL \cup nil$ then $[X\ Y] \epsilon L_{ink}$
(b) if $X \epsilon NL$, $Y \epsilon NL \cup nil$ then $[X\ Y] \epsilon L_{ink}$
(c) there is nothing else ϵL_{ink}

III. The set PL of proper links:
(a) $\alpha \epsilon PL$ iff $\alpha \epsilon L_{ink}$, and
(b) α meets all necessary conditions in (16)

IV. The set Ch of well-formed chains:
(a) if $\alpha = [X\ Y] \epsilon PL$, then $\alpha \epsilon Ch$
(b) if $Y_c \epsilon Ch$ and $[X\ Y] \epsilon PL$, then $X_c = [X\ Y_c] \epsilon Ch$
(Z_c denotes a chain headed by the node Z)

The formulation above defines a chain to be a list, headed by a lexical node,[19] followed by a list of arbitrary many NL nodes (including zero), and ending in *nil*. This allows for the singleton chain [L nil] to represent constituents which have not moved. Our specification of grammatical well-formedness constraints determining the proper links is given below.[20] Where nodes are prefixed (e.g. by 'lex' (lexical), 'a-pos' (A-postion), and 'a-bar' ($\overline{A}$-position)), we assume these attributes are accessible by whatever mechanism matches postulated links with the constraints. We discuss later how such attributes might be assigned in the first place.

(16) θ-Criterion:
(a) [N^2 nil] → theta-marked(N^2).

Case Theory:
(b) [a-bar:N^2_a a-pos:N^2_b] → case-marked(N^2_b)
or,
[lex:N^2 X] → case-marked(N^2).

A-to-$\overline{A}$ Constraint:
(c) [N^2_a N^2_b] → not(a-pos:N^2_a and a-bar:N^2_b).

Head Movement:
(d) [C^0 I^0] → true.

Category Constraint:
(e) [X Y] → X = Y = N *or* X, Y ϵ {C,I}.

Level Constraint:
(f) [X^i Y^j] → i=j=0 *or* i=j=2.

To briefly summarise these constraints, (16a) states that the last NP in a chain (the one linked to *nil*) must be θ-marked.[21] Case Theory in (16b)

[19] Since (15II) does not permit the links of the form: [NL L].

[20] Again, we stress that this formulation is simplified for expository purposes, and is not intended to represent an actual axiomatisation of current syntactic theory.

[21] We will not give definitions for the theta-marked/1 and case-marked/1 predicates here. We assume that they simply check that the relevant features are instantiated on the

states that the NP in the highest A-position (either [Spec,IP], [Spec,VP] or [Comp,VP]) must be Case marked – the two possible link configurations are either the NP linked to and $\overline{A}$-node or the head of the chain. The A-to-$\overline{A}$ Constraint in (16c) encodes the fact that there is no movement from an $\overline{A}$-position ([Spec,CP]) to an A-position. No constraints on Head movement are required for the current fragment, so this is represented by the trivially satisfied constraint in (16d). And finally, the Category constraint requires that any two linked nodes be either the same category or functional categories (16e), while the Level Constraint requires both nodes to have the same level, either 0 or 2 (16f). As with $\overline{X}$-theory for phrase structure, these last two constraints apply to all links (except those with *nil*).

Chain structure is concerned only with a particular subset of constituents in the phrase structure, namely lexical and empty NPs and the heads of the function categories (C and I). We will therefore assume here, the existence of an *interface* procedure which traverses a particular phrase structure to find these nodes which will constitute the premises of our chain structure derivation. This procedure might be reasonably used to annotate the nodes with the relevant 'lex', 'a-pos', and 'a-bar' information mentioned above (or this could simply be done by the parser). The result of this procedure will recovery of the following set of nodes (shown below with the (non-)lexical material they dominate):

(17)
$$\underline{\text{What}}_{\text{N}^2} \quad \underline{\text{will}}_{\text{C}^0} \quad \underline{\text{María}}_{\text{N}^2} \quad \underline{\text{e}_1}_{\text{I}^0} \quad \underline{\text{e}_2}_{\text{N}^2} \quad \underline{\text{e}_3}_{\text{N}^2}$$

The task of our chain deduction system is to show that this set constitutes a theorem, or rather *theorems*, since more that one chain will be formed. The derivations of each chain (i.e. one for each lexical node), are shown below, along with the its complete chain representation as constructed by (15IV):

(18)
$$\dfrac{\dfrac{\underline{\text{What}}_{\text{N}^2} \quad \underline{\text{e}_3}_{\text{N}^2} \quad \underline{\textit{nil}}}{[\text{N}^2\ \textit{nil}]^{\{16a,c\}}}(15\text{IIb})}{[\text{N}^2\ \text{N}^2]^{\{16b,c,e,f\}}}(15\text{IIa})$$

$\Longrightarrow$ [What, e_3, nil]

(19)
$$\dfrac{\dfrac{\underline{\text{will}}_{\text{C}^0} \quad \underline{\text{e}_1}_{\text{I}^0} \quad \underline{\textit{nil}}}{[\text{I}^0\ \textit{nil}]}(15\text{IIb})}{[\text{C}^0\ \text{I}^0]^{\{16d,e,f\}}}(15\text{IIa})$$

$\Longrightarrow$ [will, e_i, nil]

(20)
$$\dfrac{\dfrac{\underline{\text{María}}_{\text{N}^2} \quad \underline{\text{e}_2}_{\text{N}^2} \quad \underline{\textit{nil}}}{[\text{N}^2\ \textit{nil}]^{\{16a,c\}}}(15\text{IIb})}{[\text{N}^2\ \text{N}^2]^{\{16b,c,e,f\}}}(15\text{IIa})$$

$\Longrightarrow$ [María, e_2, nil]

By successfully deriving a proof of chain structure (i.e. the above set of well-formed chains) for the phrase structure derivation in (14) we have (i) proven that the phrase structure is a theorem of the chain grammar, and (ii) recovered the long distance relationships (i.e. in the representation of chains) which corresponds to the coindexations shown in (11). As a result, we have a complete proof that the string *What will María see* is well-formed, and our 2-tuple of proofs, phrase structure and chain structure, provide the abbreviated proofs which are the complete syntactic analysis. As we saw for the deduction of phrase structure, the links of chains do not correspond to some single grammatical axiom, but rather meet a variety of independent necessary constraints. It should also be apparent that alternative possible phrase structure derivations, such as those with omitted/extra traces would not be provable as theorems of the chain grammar, and thus rejected.

We have presented a highly simplified grammatical axiomatisation of both phrase structure and chain systems, with the aim of demonstrating how a set of necessary licensing constraints

nodes, and the counterpart Case Filter and θ-criterion in the phrase structure component do the relevant instantiations (as well as licensing).

can be exploited in a syntactic parsing framework, and how distinct representations may be decomposed, logically, while contributing to a single, complete linguistic analysis. The proposed techniques have, however been exploited in a system of more substantial linguistic coverage. In particular, the techniques outlined here underlie the basis of a larger project on the construction of a bilingual (English-German) system (Crocker, 1992, 1996). The coverage of that systems includes the following construction types (for both languages):

- Subcategorization: complex subcategorization patterns (intransitive, transitive, ditransitive, and sentence complements).
- Case-marking: morphological and abstract case assignment and exceptional case marking phenomena.
- Thematic-role assignment: as determined by subcategorization and case.
- Head movement: subject-aux inversion, V-2 raising in German, and V-to-I raising for inflection.
- Recursive sentences: relative clauses and sentential complements.
- Cyclic movement: long-distance movement from within embedded clauses.

The full system further decomposes the thematic representation from the phrase structure representation, primarily for reasons of cognitive plausibility which we are not concerned with here. Indeed it should be noted that the primary function of the system was as a model of human syntactic parsing, and thus the coverage of the grammar was determined by those constructions which were of linguistic and psychological interest, rather than a goal of wide, practical coverage. The current implementation of the system therefore does not require the coverage of some other principle-based parsers (e.g. (Merlo, 1995) and (Fong, 1991)). One significant omission in the system presented here is the apparent lack of any technique for capturing the so-called *command* relations typically required for constraining long-distance dependencies, e.g. Subjacency. Roughly speaking, command relations make reference to dominance and precedence configurations in the phrase marker. Such information will apparently be unavailable to our chain system which simply receives a set of relevant nodes found in the phrase structure by the interface relation. There are, however, indexing techniques for labelling the nodes of a parse tree, such that various command relations can be determined without re-traversing the tree (Latecki, 1991) (for further discussion see (Merlo, 1993)). If we assume that the parser or interfacing procedure performs such an indexing on phrase structure, then it is possible to compute the command relations necessary for defining constraints such as Subjacency. It is then straightforward to specify such constraints as strictly local conditions on possible proper links.

2.3 Issues of Control

In the exposition to this point, we have focussed on the logical specification of a licensing grammar, and the decomposition of the system on the basis of the representation type being licensed: phrase structure and chains. To construct theorem provers for our licensing grammar axiomatisations, we first require a mechanism that can generate arbitrary formulae (i.e. trees or chains as defined in (7) and (15)) so that we can then apply the axioms of grammar to determine which trees are grammatical. When cast in these terms, the solution to developing an efficient deductive parser for a principle-based, licensing grammar might seem rather elusive. One crucial aspect of this axiomatisation, however, is precisely the locality property mentioned above. By defining the principles of grammar as conditions on proper branches or proper links, we can recursively define the set of well-formed syntactic trees and chains. That is, we need not define the entire set of trees and chains, and then subsequently restrict this set. Rather, the set of possible syntactic structures is defined exclusively in terms of locally well-formed branches and chain links. Given this characterisation of principle-based grammars we can construct structure generating theorem provers which interleave the process of structure building and licensing by the grammatical principles, on a branch-by-branch (or link-by-link) basis.

Thus far, we have treated these as distinct systems, with the latter presumably invoked after the

former. That is, the top level of the system can be specified in Prolog as follows:

```
(21) parse(String,PS,CS) :–
                    phrase-structure(String,PS),
                    interface(PS,List),
                    chain-structure(List,CS).
```

Clearly, however, such a strategy will be wildly inefficient due to its generate and test nature, although unlike Johnson's (1989) system at least the string will be used to inform the construction of phrase structure (PS). As with Johnson's system, however, we have employed coroutining to achieve much improved performance where the chain structure is recovered in tandem with phrase structure. Such a technique permits us to detect invalid phrase structures (i.e. those which are not theorems of the chain system) as early as possible. For example, assuming no rightward movement of constituents, if a trace is postulated by the parser, but cannot be licensed in a chain, then the parser can immediately backtrack and pursue an alternative path. That is, the parser will not sustain the postulation of traces which don't have a potential antecedent in the current, partially constructed chain structure.

The individual theorem provers also provide natural loci for implementing particular controls strategies without requiring modification of the principles themselves. That is, we can adjust the way candidate structures are proposed by the interpreters, but the constraints which apply to these structures remain unchanged. While any suitable parsing algorithm is in theory possible (modulo the relevant constraints imposed by recursion and empty productions), incremental parsing algorithms are necessary for optimal coroutining. That is, if the chain interpreter traverses the phrase structure tree as it is built, then stack based algorithms which delay attachments (e.g. shift-reduce), will delay the invocation of the chain interpreter. In the present system, our prototype uses a simple recursive descent algorithm for postulating trees, but related work has advocated combining top-down and bottom-up techniques (Crocker, 1994) (Stabler, 1994). Since chains are essentially a one-dimensional construct, we simply construct them linearly, from left to right.

It should be apparent that such a coroutining approach will perform as well as a traditional phrase structure grammar using a gap-threading technique, since both procedures will make use of antecedent information during parsing, to constrain the parser's search. Let us now, however, speculate on other potential control regimes. The present approach of decomposing the parsing task into distinct deductive tasks points to the possibility of using other control strategies such as parallel techniques. Since coroutining effectively simulates synchronous parallel computation, it should be relatively straightforward to implement the parallel counterpart. Indeed this should also be possible for the coroutining model proposed by Johnson (1989). That is, while traditional phrase structure grammars permit parallel computation of multiple, distinct syntactic analyses in parallel, the modular licensing model also permits parallel computation *within* a single syntactic analysis. That is, a principle-based system may be distributed, with separate processes computing different representational levels or types.

Interestingly, however, the approach developed here also presents the possibility of asynchronous parallelisation of the chain system. Once the relevant nodes in the phrase structure tree are identified by the interface procedure, they can simply be handed to an asynchronous chain process. That is, the derivation of chain structure, assuming the indexing technique mentioned above, becomes independent of the actual phrase structure parser. The implementation of such systems remains a matter for future investigation.

3 Conclusion

In this paper we have advanced two principal arguments. Firstly we highlighted a fundamental difference between rule-based and principle-based grammars with regards to deductive parsing. Rule-based grammars may be more accurately termed 'construction-based' grammars, in that an individual rule is *sufficient* to license a particular unit of structure. This one-to-one, rule-to-structure correspondence means that rules may be efficiently used to *propose* candidate structures during parsing. Principle-based grammars, however, fall into a category we have termed 'licensing-grammars' (following

Speas (1990)), whose axiomatisation possesses rather different characteristics. In the case of licensing grammars, principles are typically *necessary* conditions on particular structural configurations, and indeed numerous conditions may apply to a particular unit of structure. Furthermore, none of these principles is particularly suited to the task of proposing candidate structures, since a given constraint is (typically) only defined with respect to some isolated aspect of the structure (i.e. some subset of features). Since the individual principles are not *sufficient* to license some structure, such an axiomatisation also entails the existence of axioms which generate the space of possible structure, such as binary-branching trees, for example.

The second contribution is to suggest that the various informational dependencies involved in current principle-based analyses be decomposed into simple, homogeneous representation types. In so doing, we permit the generalised invocation of the above methodology; that is, just as we define phrase structure as a set of structure licensing conditions over the branches of a binary-branching tree, so can we define chain structure and a set of structure licensing conditions over the links in a chain (or, list). While it is an empirical issue as to whether or not the locality constraints imposed by such a formalisation are sufficiently powerful, they are certainly in the spirit of current proposals in syntactic theory (see Manzini (1992) as an example).

In the context of the decomposed, modular architecture we have constructed, we argue that the generate and test nature of the system can be overcome using coroutining techniques similar to Johnson (1989). It has also been our experience, however, that by separating the grammatical principles along representational lines it is much easier to monitor the informational dependencies among various constraints, a matter of crucial importance to the performance in systems which use the *goal freezing* mechanism which underlies coroutining. The 'decomposition' approach also means that theorem proving strategies can potentially be adapted to individual representation types, in a manner that would be virtually impossible for systems such as Johnson's, and finally, we have suggested that our approach is potentially much more amenable to distribution and parallelisation of the parsing task. Whether or not this turns out to be valuable remains a topic for future research.

Acknowledgments

The author would like to thank Ian Lewin for valuable comments on this work. This paper was written while the author was supported by ESRC research fellowship #H52427000394.

References

[1] Abramson, H. and Dahl, V. (1988). *Logic Grammars*. Symbolic Computation Series. Springer Verlag.

[2] Ades, A. and Steedman, M. (1982). On the Order of Words. *Linguistics and Philosophy*, **4**, 517–558.

[3] Berwick, R. C. (1991). Principle-Based Parsing. In P. Sells, S. Sheiber, and T. Wasow, editors, *Foundational Issues in Natural Language Processing*. MIT Press, Cambridge, Massachusetts.

[4] Berwick, R. C., Abney, S. P., and Tenny, C., editors (1991). *Principle-Based Parsing: Computation and Psycholinguistics*. Studies in Linguistics and Philosophy. Kluwer Academic Publishers, Dordrecht.

[5] Chomsky, N. (1981). *Lectures on Government and Binding*. Foris Publications, Dordrecht.

[6] Crocker, M. W. (1991a). Multiple Interpreters in a Principle-Based Model of Sentence Processing. In *Proceedings of the 5th Conference of the European ACL*, Berlin, Germany.

[7] Crocker, M. W. (1991b). *A Principle-based System for Syntactic Analysis. Canadian Journal of Linguistics*, **3**.

[8] Crocker, M. W. (1992). *A Logical Model of Competence and Performance in the Human Sentence Processor*. Ph.D. thesis, Dept. of Artificial Intelligence, University of Edinburgh, Edinburgh, U.K.

[9] Crocker, M. W. (1994). On the Nature of the Principle-Based Sentence Processor. In

C. Clifton Jr., L. Frazier, and K. Rayner, editors, *Perspectives on Sentence Processing*, chapter 11, pages 245–266. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

[10] Crocker, M. W. (1996). *Computational Psycholinguistics: An Interdisciplinary Approach to the Study of Language*. Studies in Theoretical Psycholinguistics 20. Kluwer Academic Publishers.

[11] Fong, S. (1991). *Computational Properties of Principle-Based Grammatical Theories*. Ph.D. thesis, MIT, Cambridge, Massachusetts.

[12] Fordham, A. J. and Crocker, M. W. (1997). A Stochastic Government-Binding Parser. In D. B. Jones and H. Somers, editors, *New Methods in Language Processing*, UCL Press, London, UK.

[13] Johnson, M. (1989). Use of Knowledge of Language. *Journal of Psycholinguistic Research*, **18**(1).

[14] Johnson, M. (1991). Program Transformation Techniques for Deductive Parsing. In C. Brown and G. Koch, editors, *Natural Language Understanding and Logic Programming, III*. Elsevier Science Publishers (North-Holland).

[15] Lasnik, H. and Uriagereka, J. (1988). *A Course in GB Syntax: Lectures on Binding and Empty Categories*. Current Studies in Linguistics. MIT Press, Cambridge, Massachusetts.

[16] Latecki, L. (1991). An Indexing Technique for Implementing Command Relations. In *Proceedings of the 5th Conference of the European ACL*, Berlin, Germany.

[17] Manzini, R. (1992). *Locality: A Theory and Some of Its Empirical Consequences*. Linguistic Inquiry Monograph Nineteen. MIT Press, Cambridge, Massachusetts.

[18] Merlo, P. (1993). For an Incremental Computation of Intra-sentential Coreference. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Chambery, France.

[19] Merlo, P. (1995). Modularity and Information Content Classes in Principle-based Parsing. *Computational Linguistics*, **21**(4).

[20] Pereira, F. and Shieber, S. (1987). *Prolog and Natural-Language Analysis*. CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, California.

[21] Pereira, F. and Warren, D. (1980). Definite Clause Grammars for Language Analysis. *Artificial Intelligence*, **13**, 231–278.

[22] Pereira, F. and Warren, D. (1983). Parsing as Deduction. In *Proceedings of Twenty-First Conference of the ACL*, Cambridge, Massachusetts.

[23] Speas, M. (1990). *Phrase Structure in Natural Language*. Studies in Natural Language and Linguistic Theory. Kluwer, Dordrecht.

[24] Stabler, E. P. (1987). Logic Formulations of Government-Binding Principles for Automatic Theorem Provers. Cognitive Science Memo 30, University of Western Ontario, London, Ontario.

[25] Stabler, E. P. (1992). *The Logical Approach to Syntax: Foundations, Specifications, and Implementations of Theories of Government and Binding*. MIT Press, Cambridge, Massachusetts.

[26] Stabler, E. P. (1994). Syntactic Preferences in Parsing for Incremental Interpretation. Unpublished manuscript, UCLA.