Modularity and Information Content Classes in Principle-based Parsing

Paola Merlo*
Université de Genève

In recent years models of parsing that are isomorphic to a principle-based theory of grammar (most notably Government and Binding (GB) Theory) have been proposed (Berwick et al. 1991). These models are natural and direct implementations of the grammar, but they are not efficient, because GB is not a computationally modular theory. This paper investigates one problem related to the tension between building linguistically based parsers and building efficient ones. In particular, the issue of what is a linguistically motivated way of deriving a parser from principle-based theories of grammar is explored. It is argued that an efficient and faithful parser can be built by taking advantage of the way in which principles are stated. To support this claim, two features of an implemented parser are discussed. First, configurations and lexical information are precompiled separately into two tables (an $\overline{X}$ *table and a table of lexical co-occurrence) which gives rise to more compact data structures. Secondly, precomputation of syntactic features (θ-roles, case, etc.) results in efficient computation of chains, because it reduces several problems of chain formation to a local computation, thus avoiding extensive search of the tree for an antecedent or extensive backtracking. It is also shown that this method of building long-distance dependencies can be computed incrementally.*

1. Introduction

In the development of parsers for syntactic analysis, it is standard practice to posit two working levels: the grammar, on the one hand, and the algorithms, which produce the analysis of the sentence by using the grammar as the source of syntactic knowledge, on the other hand. Usually the grammar is derived directly from the work of theoretical linguists. The interest in building a parser that is grounded in a linguistic theory as closely as possible rests on two sets of reasons: first, theories are developed to account for empirical facts about language in a concise way—they seek general, abstract, language-independent explanations for linguistic phenomena; second, current linguistic theories are supposed to be models of humans' knowledge of language. Parsers that can use grammars directly are more likely to have wide coverage, and to be valid for many languages; they also constitute the most economical model of the human ability to put knowledge of language to use. Therefore, postulating a direct correspondence between the parser and theories of grammar is, methodologically, the strongest position, and is usually assumed as a starting point of investigation. However, experiments with parsers that are tightly related to linguistic principles have often been a disappointment, largely because these parsers are inefficient.

Inefficiency is a problem that cannot simply be cast aside. Computationally, it renders the use of linguistic theories impractical, and, empirically, it clashes with the

* Département de Linguistique Générale, Université de Genève, 2 rue de Candolle, 1204 Genève, Switzerland

observation that humans make use of their knowledge of language very effectively. In this paper, I investigate the computational problem related to the tension between building linguistically based parsers and building efficient ones, which, I argue, derives from the particular forms linguistic theories have taken recently. In particular, I explore the issue of what is a good parsing technique to apply to principle-based theories of grammar. I take Government-Binding (GB) theory (Chomsky 1986a,b; Rizzi 1990) to be a suitable illustration of such theories, and also to show in all clarity the problems that might arise. I differ from other investigations on the import of principle-based parsing in not drawing on cognitive issues or psycholinguistic results to justify my assumptions. Indeed, part of the spirit of this work is to explore how far one can go in advocating principle-based parsing, in the absence of motivations given by cognitive modelling.

1.1 The Problem

When generative grammatical theory in the '70s talked about "dative shift," "topicalization," "passive," it meant that each of these constructions was captured in the grammar by a specific rule. Consequently, rules were not only construction-specific, but also language-specific (French, Italian and Spanish, for instance, have no "dative shift"). The conceptual development of the '80s, in many frameworks, consists in having identified the unifying principles of many of these construction-specific rules. For example, according to GB theory, the same set of principles are at work in the "raising" construction, (1a) and in passive, (1b). The principles are $\overline{X}$ theory, the Theta Criterion, and the Case Filter. In both cases, the relation between the underlying position and the surface string is expressed by *chains*. Chains consist of the word that undergoes movement and all the positions this word occupies in the course of a derivation. In (1) the chains are *(John, t)* and *(The children, t)*.

(1) a. John seems $[_{IP}$ t to like Bill]

b. The children are loved t by John.

The advantage of this treatment is that common properties of language, here certain classes of verbs, are expressed by common principles.

This search for generality is not unique to GB theory. Feature-structure formalisms also use rule schemata to capture similarities among grammar rules. Moreover, reentrancy as a notational device to express common features seeks the same type of representational economy that is expressed by the use of "traces" in GB theory.

It is desirable for a syntactic analyser to make use of linguistic theories to obtain, at least in principle, the same empirical coverage as the theory, and to capture the same generalizations. Moreover, a parser that makes direct use of a linguistic theory is more *explanatory*. A guiding belief for the development of the generative framework is that a theory that can derive its descriptions from the interaction of a small set of general principles is more explanatory than a theory in which descriptive adequacy is obtained by the interaction of a greater number of more particular, specific principles (Chomsky 1965). This is because the former theory is smaller. Thus, each principle can generate a set the encoding of which would require a much larger number of bits than the bits needed to encode the principle itself. The classic example is the use of natural classes of distinctive features in phonology, in order to compact several rules into one. A modular theory that encodes universal principles has obtained a greater degree of succinctness than a nonmodular theory, and is considered more explanatory. Since it

is desirable for the parser to maintain the level of explanatory power of the theory, it must maintain its modularity.

It has also been argued (Berwick 1991) that the current shift from rules to a modular system of principles has computational advantages. Principle-based grammars engender compactness: Given a set of principles, $P_1, P_2, \ldots, P_n$, the principles are stored separately and their interaction is computed on-line; the multiplicative interaction of the principles, $P_1 \times P_2 \times \cdots \times P_n$ does not need to be stored. Hence, the size of the grammar is the sum of the sizes of its components: $|G| = P_1 + P_2 + \cdots + P_n$. Consequently, a parser based on such a grammar is compact, and, theoretically, easier to debug, maintain and update.[1] In practice, however, designing and implementing faithful and efficient parsers is not a simple matter.

Defining "faithfulness" to a linguistic theory is not a trivial task, as a direct relation between the grammar and the parser is not the only option (see Bresnan 1978; Berwick and Weinberg 1984; van de Koot 1990, and references therein). In general, it is not necessary for a parser to implement the principles of the grammar directly. Rather, a covering grammar could be used, more suited to the purpose of parsing. However, it is important that such covering be done in such a way that accidental properties of a particular grammar, which would not hold under counterfactual changes, are not used. Otherwise, the covering grammar would not be sufficiently general.

A faithful implementation is particularly difficult in the GB framework, as GB principles are informally expressed as English statements, and can take a variety of forms. For example, $\overline{\text{X}}$ theory (a condition on graphs), the Case Filter (an output filter on strings), and the θ criterion (a bijection relation on predicates and arguments) all fall under the label of principles. Attempts have been made to formalize GB principles to a set of axioms (Stabler 1992).

One possible, extreme interpretation of the direct use of principles is an approach where no grammar compilation is allowed (Abney 1989; Frank 1992; Crocker 1992).[2] This approach is appealing because it reflects, intuitively, the idea of using the grammar as a set of axioms and reduces parsing to a deduction process. This is very much in the spirit of the current shift in linguistic theories from construction-dependent rules to general principles, and it separates quite clearly the grammar from the parsing algorithm.

However, it is not obvious that this approach is efficient. Partial evaluation and variable substitution can increase performance, but, as usual, a space/time trade-off will ensue. Excess of partial evaluation off-line increases the size of the grammar, which might, in turn, slow down the parse. Experimentation with different kinds of algorithms suggests that some amount of compilation of the principles might be necessary to alleviate the problem of inefficiency, but that too much compilation slows down the parser again.

1 Berwick (1982, 403ff.) shows that the size of a cascade of distinct principles (viewed as machines) is the size of its subparts, while if these same principles are collapsed, the size of the entire system grows multiplicatively. Modularity corresponds to maximal succinctness when all independent principles are stated separately. Independent principles are, intuitively, principles that can be computed independently of each other, and therefore whose interactions are all possible. Barton *et al.* (1987) and Berwick (1990) attempt to formalize the concept of independence as *separability*, assuming that the topology of a principle-based theory like GB can be mapped onto a planar graph. In fact, if independent modules are separable modules, there is little reason to think that GB is modular, as it corresponds to a highly connected graph.

2 By compilation, here and below, I mean off-line computation of some general property of the grammar, for example the off-line computation of the interaction of principles, using partial evaluation or variable substitution.

1.2 On-line Computation is Inefficient

Several researchers note that principle-based parsers allowing no grammar precompilation are inefficient. Firstly, Johnson (1989), Stabler (1990), and van de Koot (1991) note that the computation of a multi-level theory without any precompilation might not even terminate. Secondly, experimental results show that an entirely deductive approach is inefficient. Kashket (1991) discusses a principle-based parser, where no grammar precompilation is performed, and which parses English and Warlpiri using a parameterized theory of grammar. The parsing algorithm is a generate-and-test, backtracking regime. Kashket (1991) reports, for instance, that a 5-word sentence in Warlpiri (which can have 5! analyses, given the free word order of the language) can take up to 40 minutes to parse. He concludes that, although no mathematical analysis for the algorithm is available, the complexity appears to increase exponentially with the input size.

Fong (1991, 123) discusses a parsing algorithm. He shows that an initial version of the parser, where the phrase structure rules were expressed as a DCG and interpreted on-line, spent 80% of the total parsing time building structure. In a later version, where rules were compiled into an LR(1) table, structure-building constituted 20% of the total parsing time. This same parser includes a module for the computation of long distance dependencies, which works by generate-and-test. Fong finds that this parsing approach is also inefficient.

Dorr (1987) notices similar effects in a parser that uses an algorithm more parallel in spirit (Earley 1970). Dorr notes that a limited amount of precompilation of the principles speeds up the parse, otherwise too many incorrect alternatives are carried along before being eliminated. For example, in her design, $\overline{X}$ theory and the other principles are coroutined. She finds that precompiling the principles that license empty categories with the phrase structure rules reduces considerably the number of structures that are submitted to the filtering action of the other principles, and thus speeds up the parse.

In all these cases, the source of inefficiency stems from the principle-based design. Because each principle is formulated to be as general as possible, the "logical" abstraction of each principle from the others causes a lot of overgeneration of structure and, consequently, a very large search space.

1.3 Too Much Precompilation is Inefficient

Simple precompilation is not a solution to the inefficiency of principle-based parsing, however. Experimentation with different amounts of precompilation shows that off-line precompilation speeds up parsing only up to a certain point, and that too much precompilation slows down the parser again.

The logic of why this happens is clear. The complexity of a parsing algorithm is a composite function of the length of the input and the size of the grammar. For the kind of input lengths that are relevant for natural language, the size of the grammar easily becomes the predominant factor. If principles are precompiled in the form of grammar rules, the size of the grammar increases.

As Tomita (1986) points out, input length does not cause a noticeable increase in running time up to 35 to 40 input tokens. For sentences of this length, grammar size becomes a relevant factor for grammars that contain more than approximately 220 rules, in his algorithm (an LR parser with parallel stacks). Both Dorr (1987) and Tomita (1986) show experimental results confirming that there is a critical point beyond which the parser is slowed down by the increasing size of the grammar. In the Generalized Phrase Structure Grammar (GPSG) formalism (Gazdar et al. 1985), similar experiments have been performed, which confirm this result. Parsers for GPSG are particularly interesting, because they use a formalism that expresses many grammatical generalizations in

a uniform format. Therefore, GPSG is, in principle, more amenable to being processed by known parsing techniques. Thompson (1982) finds that expanding metarules, rather than computing them on-line, is advantageous, but that instantiating the variables in the expanded rules is not. Phillips and Thompson (1985) also remark that compiling out a grammar of twenty-nine phrase-structure rules and four metarules is equivalent to "several tens of millions of context-free rules." Phillips (1992) proposes a modification to GPSG that makes it easier to parse, by using propagation rules, but still notes that variables should not be expanded.

In conclusion, the lesson from experimentation is that parsing done totally on-line is inefficient, but that compilation is not always a solution. A parser that uses linguistic principles directly must fulfill apparently contradictory demands: for the parser to be linguistically valid it must use the grammar directly, while a limited amount of off-line precompilation might make the parser more efficient.[3] In the next section, I propose and discuss a solution to this problem that builds on other approaches and relates the parser to the grammar in a principled way.

2. The Proposal

Two avenues have generally been pursued to build efficient GB parsers. In one case, a "covering grammar" is compiled, which overgenerates and is then filtered by constraints. The compilation is done in such a way that the overgeneration is well-behaved. For instance, the correct distribution of empty categories is calculated off-line (Dorr 1993). In the other case, all the principles are applied on line, but they apply only to a portion of the tree, and are therefore restricted to a local computation (Frank 1992).[4] My proposal combines these two approaches: it adopts the idea of compiling the grammar, at least partially, off-line but it attempts to find a principled way of doing so. In this, I differ from Dorr, where the amount of compilation is heuristic and based on practical experimentation. The approach shares Frank's intuition that linguistic principles have a form, which can be exploited in structuring the parser.

This proposal is based on two observations. First, each principle of linguistic theory has a *canonical form*, and second, primitives of linguistic theories can be partitioned into classes, based on their content.

As an illustration of the first observation, we can look at the principle that regulates the distribution of the empty categories in the phrase marker, the Empty Category Principle (ECP), as stated below (adapted from Rizzi 1990, 25).

(2) The Empty Category Principle

An empty category x is licensed if the 3 following conditions are satisfied:

1. x is in the domain of a head H

3 For CF parsers, just how much compilation speeds up the parser is defined precisely by the analysis of the algorithm. No such precise analysis is available for principle-based algorithms.

4 Frank (1992) presents a parsing model that is claimed not to allow any compilation of the linguistic theory, and to operate in linear time. Two objections can be raised to these claims: first, the use of TAG elementary trees to restrict the working space of the parser amounts to a precompilation of phrase-structure and locality constraints, so that locality is not computed in the course of the parse, but basically done as template matching. Second, in the measure of complexity, Frank does not count the cost of choosing which elementary tree to unadjoin or unsubstitute, or the cost of backtracking if the wrong decision is made. There are indeed cases where, in order to perform the correct operation, more than one elementary tree must be spanned. It is not clear that linear time complexity can actually be claimed if all factors are taken into account. For a more detailed discussion, see Merlo 1992, to appear.

2. the category of H $\in$ {A, Agr, N, P, T, V}
3. there is no barrier or head H′ that intervenes between H and x

It can be observed that this principle has an internal structure and can be decomposed into separate pieces of information: (2.1) imposes a condition on configurations, namely, a condition on the shape of the tree; (2.2) imposes a condition on the labelling of the nodes in the tree; and (2.3) imposes a locality condition, as it defines the subtree available for the computation of the principle. These three conditions are independent. For instance, the configuration does not depend on the categorial labelling of the head node. The precompilation of these conditions would require computing all the possible combinations, without any reduction of the space of analysis.

The second observation is based on a detailed inspection of the form of the principles of the grammar. What is presented in (2) as an illustrative example is, in fact, a consistent form of organization of the principles. If one looks at several of the principles of the grammar that are involved in building structure and annotating the phrase marker, one finds the same internal organization.

Theta-assignment occurs in the configuration of sisterhood, it requires a θ-assigning head, and it must occur between a node and its most local assigner. Assignment of Case occurs in a given configuration (according to Chomsky (1988, 1992) it is always a specifier-head configuration), given a certain lexical property of the head ([-N]), and locally, within the same maximal projection). The same restriction occurs again for what is called the *wh*-criterion (Rizzi 1991), which regulates *wh*-movement, where the head must have a *+wh* feature and occur within a specifier-head configuration. Categorial selection and functional selection also occur under the same restrictions, in the complement configuration (i.e., between a head and a maximal projection). The licensing of subjects in the phrase marker, done by predication, must occur in the specifier-head configuration. The licensing of the empty category *pro* also requires the inflectional head of the sentence to bear the feature *Strong Agr*, and it occurs in the specifier-head configuration. The assignment of the feature [$\pm$ barrier] depends on L-marking, which in turn requires that the head is lexical, and that marking occurs in the complement configuration.

Thus, each different "factor" that composes a principle can be considered a separate primitive, and such primitives can be grouped into classes defined according to their *content*. Linguistic information can be classified into five different classes:

(3) a. Configurations: sisterhood, c-command, m-command, $\pm$ maximal

b. Lexical features: $\pm$N, $\pm$V, $\pm$Funct, $\pm$c-selected

c. Syntactic features: $\pm$Case, $\pm\theta$, $\pm\gamma$, $\pm$barrier, $\pm$Strong Agr

d. Locality information: minimality, antecedent government

e. Referential information: $\pm$anaphor, $\pm$pronominal, indices

This qualitative classification forms a partitioning into natural classes based on *information content*. I call these IC Classes.[5]

5 Differently from Crocker (1992, to appear) and Frazier (1985), this partitioning does not rely on the particular representation used. The spirit of the hypothesis is that linguistic theory is formed by heterogeneous types of information, and that the representation used to describe them is a derived concept. Frazier (1990) proposes an *evolutionary* partitioning of the parser based on tasks. This

It can then be hypothesized that the amount of compilation (or, conversely, the modularity of the parser) is captured by the notion of IC classes as follows:

IC Modularity Hypothesis (ICMH)
Precompilation within IC Classes improves efficiency.
Precompilation across IC Classes does not.

In other words, a parser that takes advantage of the structure of linguistic principles will maintain a modular design based on the five classes in (3).

Although the ICMH is not so stringent as to make predictions that converge on a single parsing architecture, it does provide some predictive power about the organization of the parser. First, structural information is encoded separately from lexical information. Standard context-free rules, specified with category, such as VP $\rightarrow$V NP, are not compatible with the ICMH, nor are proposals in the spirit of licensing grammars (Abney 1989, Frank 1992), where information is encoded in each lexical item. Second, the ICMH predicts that long-distance dependencies, represented as chains, are computed in steps. Empty categories are licensed in two computational steps: structural licensing by an appropriate head, and feature instantiation. With respect to feature instantiation in particular, it is predicted that precompiling syntactic features speeds up the parsing process. This is different from functional approaches such as Fong (1991), and Fong and Berwick (1992), in which there is no precompilation.[6]

These predictions seem to be supported (and, consequently, so is the ICMH) by two main results, which are illustrated below:

1. separating $\overline{X}$ from lexical information yields more compact data structures; I propose a parser that uses two compiled tables: one that encodes structural information, and the other that encodes lexical information.

2. using syntactic features to compute empty categories reduces the search space, complex chains can be computed efficiently.

These claims are supported in the next section, where I discuss the properties of an implemented parser, which computes simple, complex, and multiple chain formation, as exemplified in Figure 1. This subset of constructions has been chosen because it constitutes the crucial test set for principle-based parsers: it involves complex interactions of principles over large portions of the tree.[7]

---

perspective is not in opposition to the current proposal, as the specialization of the parser in different tasks is likely to be an adaptive reaction to the different types of inputs.

At first sight it might appear that the notion of types proposed by Fong (1991) is similar to IC classes. In fact, the similarity is superficial. Clearly, both notions constitute an attempt to partition the set of principles into smaller subsets. However, Fong's types are a mechanism to interleave constraints and phrase structure rules automatically. They are a method to schedule the on-line computation of principles that are the direct translation of the theory, and not a way of defining the design of the parser. In Fong's view, all computations are done on-line and the parser reflects the theory as directly as possible.

6 It is difficult to separate precisely "lexical" from "syntactic" features. One can consider "syntactic" those features that are used to determine the well-formedness of syntactic trees. In the spirit of more recent developments in syntactic theory, I consider syntactic those features that are involved in some particular incarnation of the "Generalized Licensing Condition" (Sportiche 1992.) These include θ-roles, case, (possibly all ϕ features), and, following Rizzi (1991), Haegemann and Zanuttini (1991), and Laenzlinger (1993), also *wh, neg, adverb*.

7 Many other proposals either do not deal with all types of chains (Frank 1992; Johnson 1989, for

	TYPE	EXAMPLE
1	Simple Transitive	*john loves mary*
2	Simple Intransitive	*john runs*
3	Simple Passive	*mary was loved*
4	Simple Raising	*mary seems to like john*
5	Embedded Transitive	*john thinks that mary loves bill*
6	Embedded Intransitive	*john thinks that mary runs*
7	Embedded Raising	*mary thinks that john seems to like bill*
8	Simple Question	*who does john love ?*
9	Embedded Question	*who do you think that john likes ?*
10	Embedded Question and Raising	*who did you think that john seemed to like ?*
11	Embedded Wh-Question	** who did you wonder why mary liked ?*

Figure 1
Types of Sentences.

In the rest of the paper, I first discuss the advantages of storing $\overline{X}$ information separately from lexical information (section 3). I then turn to the computation of long-distance dependencies. I illustrate two algorithms to compute chains: I show that a particular use of syntactic feature information speeds up the parse, and I discuss the plausibility of using algorithms that require strict left-to-right annotation of the nodes (section 4). In fact, the algorithm I propose appears to be interestingly correlated to a gap in the typology of natural languages.

3. The Computation of Phrase Structure

In order to explore the validity of the proposed hypothesis about the modularity of the parser, an analyzer for English was developed. Each of the data structures is the direct implementation of linguistic objects with different information contents. The input to the algorithm is an unannotated sentence. The output consists of a tree and a list of two chains: the list of $\overline{A}$ chains and the list of A chains, that is, chains formed by *wh*-movement and NP movement, respectively. The main parsing algorithm is a modified LR parsing algorithm augmented by multi-action entries and constraints on reduction.[8]

The structure-building component of the parser is driven by an LR(k) parser (Knuth 1965) which consults two tables. One table encodes $\overline{X}$ information (following Kornai and Pullum 1990). The other table encodes lexical information. Lexical information is consulted only if it is needed to disambiguate a state containing multiple actions in the LR parser. An overview of this design is shown in Figure 2.

instance) or they require extensive backtracking (Fong 1991; Fong and Berwick 1992). In formalism other than GB theory, gaps are encoded directly into the rules. Both GPSG and HPSG use slash features to percolate features to gaps. The use of slash features probably simplifies the computation. There has been a debate on the explanatory adequacy of grammars that employ slash features (see van de Koot 1990, and Stabler 1994). For my purposes, note that, if anything, I am dealing with the worst case for the parser.

8 The ICMH is not sufficient to predict a specific parsing architecture, but rather it loosely dictates the organization of the parser. The choice of an LR parser then is the result of the ICMH (with which the parser's organization must be compatible) and additional independent factors. First, LR parsers have the valid prefix property, namely they recognize that a string is not in the language as soon as possible (other parsing methods have this property as well, for instance Schabes 1991). A parser with this property is incremental, in the sense that it does not perform unnecessary work, and it fails as soon as an error occurs. Second, the stack of an LR parser encodes the notion of c-command implicitly. This is

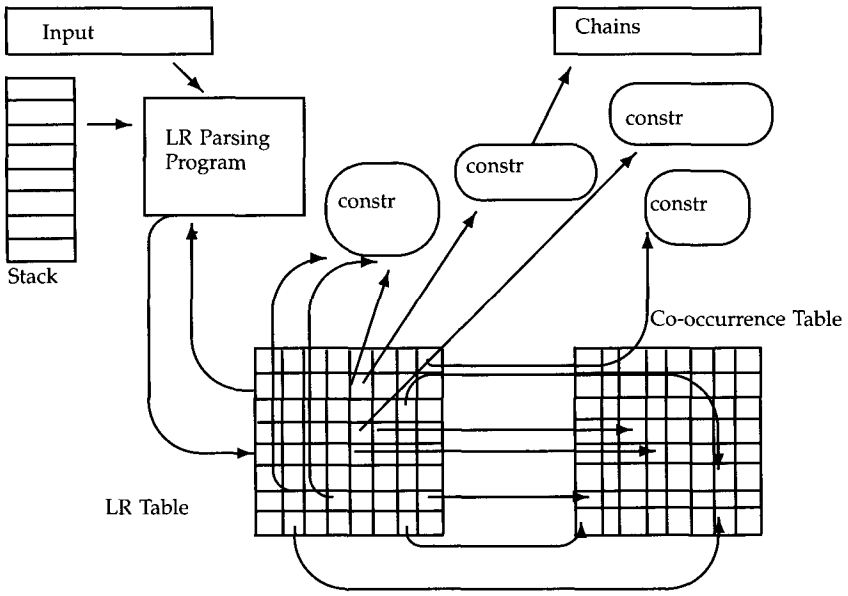

Figure 2
Organization of the Parser: The data structures (tables, stack and chains) are represented as rectangles. Operations on feature annotation are performed by constraints, represented as ovals.

X″	→ Y″ X′	specification
X″	→ X′ Y″	
X′	→ X Y″	complementation
X′	→ Y″ X	
X′	→ Y″ X′	modification
X′	→ X′ Y″	
X′	→ X	unary head
X″	→ X′	unary Xmax
X	→ empty	empty heads
X″	→ empty	empty Xmaxn

Figure 3
Category-Neutral Grammar.

The context-free grammar compiled in the LR table is shown in Figure 3. The crucial feature of this grammar is that nonterminals specify only the $\overline{X}$ projection level, and not the category. Because the LR table is underspecified with respect to the categorial labels of the input, many instances of LR conflicts arise, which can be teased apart by looking at the co-occurrence restrictions on categories. This information would be stored in the rules themselves in ordinary context-free rules. However, ordinary context-free rules do not encode many other types of lexical information also used in parsing. Thus, they lose generality, without exploiting all the available information. As an illustration, consider the following set of context-free rules.

crucial for fast computation of chains. Third, LR parsers are fast.

(4)

1.	C′	→	C_0	IP
2.	I′	→	I_0	VP
3.	V′	→	V_0	NP
4.	V′	→	V_0	e
5.	V′	→	V_0	

Rules 1–4 have the same $\overline{X}$ structure, but they differ in the labels of the nodes. In rules 1 and 2 the heads, C_0 and I_0 respectively, are followed by IP and VP obligatorily. Rules 4 and 5 cover the same string. Clearly, by writing 1–4 as different rules, the fact that they are instances of the same structure is not captured. Similarly, the obligatoriness of IP and VP as complements of C_0 and I_0 is lost. Finally, the choice of rule 4 or rule 5 depends on the actual verb in the string. If the verb is intransitive, rule 4 cannot apply.

In the parser, structural information is separate from information about co-occurrence (rules 1–4), functional selection (rules 1, 2) and subcategorization (rules 4, 5). This information is stored in a table, called a co-occurrence table. The table stores information about obligatory complementation, such as the fact that I_0 must be followed by a VP. It also stores compatible continuations based on subcategorization. For instance, consider the case in which the current token is an intransitive verb. The LR table contains two actions that match the input: one action generates a projection of the input node (V′), without branching, while the other action creates an empty object NP. By consulting the subcategorization information, the parser can eliminate the second option as incorrect.

Using an LR table together with a co-occurrence table is equivalent in coverage to a fully instantiated LR table, but it is more advantageous in other respects. Conceptually, the latter organization encodes $\overline{X}$ theory directly, and it maintains a general design, which makes it applicable to several languages. Practically, there is reason to think that it is more efficient.

3.1 Testing the ICMH for phrase structure

The prediction made by the ICMH is that compiling together $\overline{X}$ theory and categorial information will increase the size of the grammar without reducing the nondeterminism contained in the grammar, because category/subcategory information belongs to a different IC Class than structural (i.e., $\overline{X}$) information.

Method and Materials. The size of the grammar is measured as the number of rules or number of states in the LR table. The amount of nondeterminism is measured as the average number of conflicts (the ratio between the number of actions and the number of entries in a table.)[9]

Three grammars were constructed, constituting (pairwise) as close an approximation as possible to minimal pairs (with respect to IC Classes). They are shown in the Appendix. Grammar 1 differs minimally from Grammar 2, because each head is instantiated by category. The symbol YP stands for any maximal projection admitted by linguistic theory. Grammar 3 differs minimally from Grammar 2, because it also

9 The average number of conflicts in the table gives a rough measure of the amount of nondeterminism the parser has to face at each step. However, it is only an approximate measure for at least two reasons: taking the mean of the conflicts abstracts away from the size of the grammar, which might be a factor, as the search in the table becomes more burdensome for larger tables (but, if anything, it plays against small grammars/tables); moreover, it does not take into account the fact that some states might be visited more often than others.

Table 1
Comparison of the 3 grammars (compiled into LR tables)

	NB OF ENTRIES	NB OF ACTIONS	NB OF RULES	AVERAGE CONFLICTS
GRAMMAR 1	63	123	16	1.95
GRAMMAR 2	793	1319	51	1.78
GRAMMAR 3	251	962	41	3.83

Table 2
Number of actions in the 3 LR tables

	NUMBER OF ACTIONS													
ENTRIES	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GRAMMAR 1	38	6	8	8	1	2								
GRAMMAR 2	465	68	168	6	42									
GRAMMAR 3	144	43			3	8	5					4	30	14

includes some subcategorization information (such as transitive, intransitive, raising), and some co-occurrence restrictions and functional selection. Moreover, empty categories are "moved up," so that they are encountered as high in the tree as possible. These three grammars are then compiled by the same program (BISON) into three (LA)LR tables. The results are shown in Table 1, which compares some of the indices of the nondeterminism in a given grammar to its size, and Table 2, which shows the distribution of actions in each of the grammars.

Discussion. Consider Grammar 1 and Grammar 2 in Table 1. Grammar 2 has a slightly smaller average of conflicts, while it has three times the number of rules and twelve times the number of entries, compared to Grammar 1. The fact that Grammar 2 is larger than Grammar 1, with only a slightly smaller average of conflicts, confirms the prediction made by the ICMH that compiling $\overline{X}$ theory with categorical information will increase the size of the grammar without decreasing nondeterminism. Since the number of rules is expanded, but no "filtering" constraint is incorporated in Grammar 2 with respect to Grammar 1, this result might not seem surprising.

However, the ICMH is also confirmed by the other pairwise comparisons and by the global results. Grammar 3 has a higher number of average conflicts than Grammar 2, but it is smaller, both by rules and LR entries, so it is more compact. Notice that adding information (subcategory, selection, etc.) has a filtering effect, and the resulting grammar is smaller. However, adding information does not reduce nondeterminism. Compared to Grammar 1, Grammar 3 does not show any improvement on either dimension: Grammar 3 is both larger (four times as many LR entries) and more nondeterministic than Grammar 1. Globally, one can observe that an increase in grammar size, either as a number of rules or number of LR entries, does not correspond to a parallel decrease in nondeterminism.

As Table 2 shows, the distribution of the conflicts in Grammar 3 presents some gaps. This occurs because certain groups of actions go together. Two main patterns of conflict are observed: In those states that have the highest number of conflicts, all rules that cover the empty string can apply; in those states that have an intermediate number

Table 3
Comparison of the 3 grammars (compiled into LL tables)

	NB OF ENTRIES	NB OF ACTIONS	NB OF RULES	AVERAGE CONFLICTS
GRAMMAR 1	19	62	16	3.26
GRAMMAR 1′	19	46	13	2.42
GRAMMAR 2	112	255	51	2.28
GRAMMAR 3	144	368	41	2.62

of conflicts, only some rules can apply, namely, those that have a certain $\overline{X}$ projection level, and that cover the empty string (e.g., all XP's, independent of category, that cover the empty string). This observation confirms that categorial information does not reduce nondeterminism. On the contrary, adding categorical information multiplies nondeterminism by adding structural configurations. Even introducing "filtering" lexical information (co-occurrence restrictions and functional complementation) does not appear to help. In fact, ambiguities caused by empty categories occur according to structural partitions. The qualitative observation supports the numerical results: Introducing categorial information is not advantageous, because it increases the size of the grammar without decreasing significantly the average number of conflicts.

3.2 Extending the test to other compilation techniques

The effects discussed above could be an artifact of the compilation technique. In order to check that this is not the case, the same three grammars (reported in the appendix) were compiled into LL and Left Corner (LC) tables.

LL compilation: Discussion. The LL compilation method yields results similar to those of the LR compilation, although less clear cut. This confirms the intuition that the results reflect some structural property of the grammar, and are not an artifact of the LR compilation.

The results of the compilation of the same grammars into LL tables are shown in Table 3. Grammar 1′ is a modified version of Grammar 1, without adjunction rules. These figures show that there is no relation between the increased specialization of the grammar and the decrease of nondeterminism. Note that the LL compilation does not maintain the paired rankings of actions and rules. So, for the LL table, the co-occurrence of lexical categories does not play a filtering role.

Globally, there appears to be an inverse relation between the size of the grammar, measured by the number of rules, and the average number of conflicts: the larger the grammar the smaller the number of conflicts. This might make one think that there is some sort of relation between grammar size and nondeterminism after all. However, this is not true if we use the number of entries as the relevant measure of size. Moreover, if one looks at Grammar 1′, which is smaller than Grammar 1, one can see that the average number of conflicts decreases quite a bit. This confirms a weaker hypothesis, which is nonetheless related to the initial one, namely that nondeterminism does not vary in an inverse function to "content of information."

Some qualitative observations might help clarify the sources of ambiguity in the tables. In all three grammars, the same ambiguities are repeated, for each terminal item. In other words, all columns of the LL table are identical (with the exception

Table 4
Comparison of the 3 grammars (compiled into LC tables)

	NB OF ENTRIES	NB OF ACTIONS	NB OF RULES	AVERAGE CONFLICTS
GRAMMAR 1	49	136	16	2.77
GRAMMAR 2	1456	4030	51	2.76
GRAMMAR 3	398	610	41	1.53

Table 5
Number of actions in the 3 compiled LC tables

	NUMBER OF ACTIONS								
ENTRIES	1	2	3	4	5	6	7	18	19
GRAMMAR 1	4	18	15	9	3				
GRAMMAR 2	602	702	48					96	8
GRAMMAR 3	282	92		4	4	4	12		

of cell [X0, wp] in Grammar 1.) This suggests that lexical tokens do not provide any selective information. Moreover, as we saw in the LR tables, projections to the same level have the same pattern of conflicts. (In Grammar 2, the number of conflicts is multiplied by the number of categories.)[10]

LC-compilation: Discussion. The same three grammars were compiled in left corner (LC) tables. The result of the compilation are shown in Table 4, and the distribution of the conflicts is shown in Table 5. As can be seen from Table 4, Grammar 2 is three times larger than Grammar 1 and is compiled in a table that has twenty-nine times as many entries, but the average number of conflicts is not significantly smaller.

The interpretation of the LC table derived from Grammar 3 poses a problem for the ICMH. Grammar 3 is larger than Grammar 1, as it contains category and some co-occurrence information, but its average of conflicts is smaller. In this case, it seems that adding information reduces nondeterminism. On the other hand, compared to Grammar 2, both the table and the average number of conflicts are smaller. I take this to mean that the ICMH is confirmed only by a global assessment of the relation between the content of information and the average conflicts, but not by pairwise comparisons of the grammars. Notice however, that the difference in the two pairwise comparisons confirms that simple categorial information does not perform a filtering action on the structure, while lexical co-occurrence does. This is precisely what I propose to compile in the lexical co-occurrence table.

The qualitative inspection of the tables confirms the clustering of conflicts suggested by Table 5. Grammar 1 and Grammar 2 show the same patterns of conflicts as the LR and LL tables: conflicting actions cluster with the bar level of the category. So, for example, in Grammar 2, one finds that when the left corner is a maximal projection

10 In all cases, this is caused by the $\overline{X}$ form of the grammar. Namely, the loci of recursion and gapping are at both sides of the head, and anything can occur there. Eliminating this property would be incorrect, as it would amount to eliminating one of the crucial principles of GB, namely *move-α*, which says that any maximal projection or head can be gapped.

the action is unique, while when the corner is a bar level projection there are multiple actions and they are the same, independently of the input token. In Grammar 3, the same patterns of actions are repeated for each left corner, independently of the goal or of the input token.

The qualitative inspection of the compiled tables is coherent across compilation methods and appears, in general, to support the ICMH, as the interaction of structural and lexical information is the cause of repeated patterns of conflicts. Quantitatively, the results, which are very suggestive in the LR compilation, are less clear in the other two methods. However, in no case do they clearly disconfirm the hypothesis. I conclude that categorial information should be factored out of the compiled table and separate data structures should be used.[11]

4. The Computation of Chains

As a result of using a category-neutral context-free backbone to parse, most of the feature annotation is performed by conditions on rule reduction associated with each context-free rule, which are shown in Figure 4.

The most interesting issues arise in the treatment of filler-gap dependencies, which are represented as chains. Informally, a chain is a syntactic object that defines an equivalence class of positions for the purpose of feature assignment and interpretation.

(5) a. *Mary*$_i$ was loved t_i

b. *Who*$_i$ did John love t_i ?

c. *Mary*$_i$ seemed t'_i to have been loved t_i .

d. *Who*$_i$ did John think t'_i that Mary loved t_i ?

The sentence in (5a), for example, contains the chain (*Mary*$_i$, t_i), which encodes the fact that *Mary* is the object of *love*, represented by the empty category t.

In this parser, empty categories are postulated by the LR parser, when building structure, and their licensing is immediately checked by the appropriate condition on rule reductions, shown in Figure 4.

Many principles regulate the distribution of chains. For the purpose of the following discussion, it is only necessary to recall that a chain can only contain one thematic position and one position that receives case. Moreover, chains divide into two types:

11 It should be noted that, although phrase-structure rules are reduced to the bare bones, they cannot be eliminated altogether. Parsers that project phrase structure and attachments entirely from the lexicon have been presented by Abney (1989) and Frank (1992), using licensing grammars (LS). They suffer from serious shortcomings when faced with ambiguous input, as they do not have enough global knowledge of the possible structures in the language to recover from erroneous parses. Abney alleviates this problem by attaching LR states to the constructed nodes, thus losing much of the initial motivation of the licensing approach. Frank's parser is augmented by a parse stack to parse head-final languages. Frank does not discuss this issue in detail, but it seems that a "shift" operation must be added to the operations of the parser. As there could always be a licensing head in the right context, which would license a left-branching structure, the "shift" operation is always correct. But then, the parser might reach the end of the input (or at least the end of the relevant elementary tree, i.e., the main predicate-argument structure) before realizing either that it pursued an incorrect analysis, in the case of ambiguous input, or that the input is ill-formed. Thus, this augmented parser could not recognize errors as soon as they are encountered. Finally, note that all the augmentation necessary to make the LS grammar work make it equivalent to a phrase-structure grammar, possibly with the disadvantage of being procedurally instead of declaratively encoded. On the other hand, a precompiled table which keeps track of all the alternative configurations guarantees that incorrect parses are detected as soon as possible, and, if alternative parses exist, they will be found.

CONSTRAINT	FUNCTION
θ-criterion	checks if all chains in the chain list have received a θ-role
Case filter	checks if all chains in the chain list have Case
node labelling	determines what kind of chain link the current node is: head, intermediate, foot
chain selection	selects chain to unify with current node
chain unification	unifies node with selected chain
head feature percolation	consults cooccurrence table and determines cooccurrence restrictions among heads
θ-marked	marks node with available θ-role
case marked	marks node with available Case
c-select	categorial selection
is-a barrier	checks if maximal projection is a barrier
license empty head	checks features of closest lexical head
licensing head	finds a lexical head to license a maximal projection
locality	checks that the maximal projections between antecedent and empty category are not barriers

Figure 4
The Constraints.

wh-chains, also called $\overline{\text{A}}$-chains, and NP-movement chains, also called A-chains; the empty categories that occur in these chains have different properties. More than one chain can occur in a sentence. Multiple chains occurring in the same sentence can either be disjoint or intersected.[12] Disjoint chains are nested, as in (6a). If chains intersect, they share the same index and they have exactly one element in common, as in (6b).

(6) a. *Who*$_i$ did Mary$_j$ seem t_j to like t_i?

b. *Who*$_i$ did you think t_i seemed t_i to like Mary?

4.1 The Algorithms

When building chains, several problems must be solved. First of all, the parser must decide whether to start a new chain or not. It must also decide whether to start a chain headed by an element in an argument position (A-chain), such as the head of a passive chain, or a chain headed by an element in a non-argument position ($\overline{\text{A}}$-chain),

12 Actually, chains can also compose. If chains compose they do not have intersecting elements, but they create a new link. This type of chain is exemplified in (i). We will only discuss chains of the types of (6).
(i) *Who*$_i$ did you meet t_i O_i without greeting t_i?

such as the head of a *wh*-chain. Second, on positing an empty element, the parser must decide to which chain it belongs.[13]

The two decisions can be seen as instances of the same problem, which consists in identifying the *type* of link in the chain that a given input node can form (whether head, intermediate or foot, abbreviated as H,I,F in what follows.) One can describe this sequence of decisions as two problems that must be solved in order to form chains: the Node Labelling Problem (NLAB), and the Chain Selection Problem (CSEL), formulated below.

The Node Labelling Problem (NLAB).

> Given a node N to be inserted in a chain, determine its label L, where $L \in \{\overline{A}H, AH, \overline{A}I, AI, \overline{A}F, AF\}$.

This problem defines a relation R: $N \times L$, where N belongs to the set of nodes, and L belongs to the set of labels for the elements of chains. The labels of possible chain links reflect the theoretical distinctions between A-movement and $\overline{A}$-movement, and the fact that links of a chain can be either the first element of the chain, the head (H), or an intermediate element (I) in the case of chains formed by several links, or the last element, the foot (F).[14]

Algorithm 1
Input: Node, Local Configuration
Output: List of Labels

```
If Node is not empty then
  If Node is [+wh] then Label ← Ā H
  else Label ← AH
else
  If Node has θ-role then
    If Node has Case then Label ← Ā F
    else Label ← AF
  else
    If Local Configuration = Spec of C then Label ← Ā I
    else Label ← AI
```

There are six possible outputs for this algorithm. The first case arises when the node N is a lexical *wh*-word, which starts a *wh*-chain. The second possibility is if the head is lexical, but not a question word. In this case, an argument chain (A-chain) is started, as in passives. The last four cases deal with empty categories. The feature

13 Strictly speaking, it must also provide a rescuing procedure. This can be done by checking whether all the chains satisfy the well-formedness conditions. If not all the chains satisfy the well-formedness constraints, the parser can attempt to intersect or compose two or more chains in order to satisfy the well-formedness conditions. These two problems are not treated here. For an illustration, under the name of Chain Intersection Problem and Chain Composition Problem, see Merlo 1992.

14 I present here a simplified version of the algorithm, to avoid technical linguistic details, which are not relevant for the following discussion. However, one should also output a label $\overline{A}$Op, which designates the empty operator that binds, for instance, the empty variable in a parasitic gap construction and other cases of non-overt movement, such as relative clauses. In *the man OP I saw* an empty operator is postulated by analogy to *the man whom/that I saw*. $\overline{A}$Op is licensed by the same conditions that license an intermediate $\overline{A}$ trace.

annotation of the category is inspected: case distinguishes the foot of an A-chain from the foot of an $\overline{A}$-chain, while intermediate traces are characterized by a lack of θ-role and by their configurations (i.e., intermediate A empty categories occur in A positions (spec of I), while intermediate $\overline{A}$ empty categories occur in $\overline{A}$ positions (spec of C)).

Once the potential chain links have been labelled, a second algorithm looks for a chain that can "accept" a node with that label.

The Chain Selection Problem (CSEL)
Given a node N of label L, and an ordered list of chains C, return the chain C_i, possibly none, to which N has unified.

Algorithm 2
Input: Node, Label(s), Ordered List of Chains
Output: Chain or empty set

If Label $\in \{\overline{A}H, AH\}$ then
start new chain
else
If Label $\in \{\overline{A}F, AF, AI\}$ then
choose (nearest) unsaturated chain
else
If Label = $\overline{A}I$ then
choose nearest unsaturated chain,
unless it is the immediately preceding element in the stack.

The list of chains given as input is ordered by the structure-building algorithm: when new chains are started, they are added at the end of the list. The first clause of Algorithm 2 starts a new chain whenever a lexical element is seen. No other type of chain link can start a chain. The second clause selects a chain when the foot is seen. By choosing the nearest chain (i.e., the last one in the list), only nested dependencies are built. The third clause assigns $\overline{A}I$ in a condition that is more complex than the others, to deal with subject-oriented parasitic-gaps.[15]

In Figure 5, I show schematically how these algorithms build chains. A pseudo-Prolog notation is used, which is similar to the output of the parser, where chains are represented as lists enclosed in square brackets. I show the I/O of each algorithm, given the sentence *Who did you think that John seemed to like?*, where a multiple $\overline{A}$-chain and an A-chain must be recovered. NLAB takes an input word and outputs a label, while CSEL takes a triple (Node, Label, Chains) as input, and returns a new chain list.

Note that, in Algorithms 1 and 2, features such as Case and θ-role must be available as input for the correct labelling and chain assignment of the empty category. This is a crucial feature of the algorithms for chain formation proposed here.

In GB theory, empty categories can be freely coindexed with an antecedent, from which they inherit their features. Features that are incompatible with a given context are automatically excluded, since the sentence will be ungrammatical (Brody 1984). This theory is called *functional determination of empty categories*. In GB parsing, there have been two approaches to the implementation of chains: one that mirrors directly

15 This restriction handles sentences such as *A man [that [whenever I meet] looks old.]* This construction, although marginal, like all parasitic gaps, is accepted by many speakers. Parasitic gap constructions have many interesting properties that must be dealt with for the algorithms that treat chains to be fully general.

Who did you think e1 that John seemed e2 to like e3 ?		
NLAB	who	$\overline{A}$ Head
CSEL	who $\overline{A}$ Head	[(who)]
NLAB	you	A Head
CSEL	you A Head	[[(who)][(you)]]
NLAB	e1	$\overline{A}$ Intermediate
CSEL	e1 $\overline{A}$ I [[(who)][(you)]]	[[(who,e1)][(you)]]
NLAB	John	A Head
CSEL	John A Head [[(who,e1)][(you)]]	[[(who,e1)][(you)(John)]]
NLAB	e2	A Foot
CSEL	e2 A Foot [[(who,e1)][(you)(John)]]	[[(who,e1)][(you)(John,e2)]]
NLAB	e3	$\overline{A}$ Foot
CSEL	e3 $\overline{A}$ Foot [[(who,e1)][(you)(John,e2)]]	[[(who,e1,e3))][(you)(John,e2)]]

Figure 5
Chain building example.

the generate-and-test nondeterminism of the theory (Fong 1991; Kashket 1991), and another that takes advantage of structural constraints to limit the space of hypotheses, and therefore is called *structural determination of empty categories* (Correa 1988, 1991; Crocker 1991). Of these two positions, the ICMH predicts that the latter will be more efficient: features belonging to the same class must be compiled.

The algorithms proposed here amount to a "look-up" of all the relevant features that divide all empty categories into classes, as opposed to a functional algorithm, where the empty categories are all the same, and their different syntactic function is determined in a second stage of parsing. An algorithm of this latter type cannot make use of the intrinsic properties of empty categories to direct the search for the antecedent and the construction of chains. Fong (1991) shows experimentally that an algorithm that computes categories functionally slows down the parser by orders of magnitude. Fong and Berwick (1992) also report that functional determination of empty categories causes Japanese to be parsed more slowly than English, as too many categories are posited. I adopt a more indirect implementation of the theory, along the lines proposed by Correa (1991). In particular, I use features that define the typology of empty categories immediately, in the course of the parse.[16]

This position is more indirectly related to linguistic theory, thus it is a weaker theoretical position, but it is preferable because it is more efficient, with full generality. I address these two issues in the next two sections: first, I show that disregarding

16 It should be noted that the algorithms differ from Correa's in some respects, which give them wider empirical coverage. Correa's algorithm for chain evaluation is based on an attribution rule. To restrict attribute percolation, Correa imposes a restriction, such that a node can only participate in one $\overline{A}$-chain and one A-chain at a time. (The theoretical motivation for this limit is that an attribute grammar with unrestricted percolation of attributes corresponds to a type 0 grammar, thus it is too powerful to describe natural languages correctly.) Thus, some locality restrictions such as *wh*-islands and the Complex NP Constraint are modelled. These locality restrictions, though, depend on the language. This attribution rule does not work for less restrictive languages, such as the British variant of English (Grimshaw 1986) and Italian (Rizzi 1982), which allow the types of extractions that Correa's limit is designed to exclude. Finally, as Correa himself notes, this attribution would not work for parasitic gaps in English. Thus, imposing a one-slot limit to the propagation of $\overline{A}$ chains is not the right way to capture restrictions on movement. Reasons of space prevent me from discussing the issues related to locality here, which have been the topic of debate for a rather long time (see Marcus 1980; Berwick and Weinberg 1984,1985; Frank 1992 and references therein). For a proposal which follows current linguistic theory, based essentially on a parametrization of locality restrictions, see Merlo 1992, to appear.

features such as case and thematic roles when building chains leads to an exponential growth of the space of hypotheses; second, I argue that using these features does not restrict the validity of the algorithm to specific constructions or languages.

4.2 Restricting the Search Space

As the previous section on phrase structure has shown, computing features is not always profitable, as some features reduce the search space while others do not. To see that checking features does indeed pay off, the cost of checking these features must be compared to the benefit of reducing the search space.

This analysis mostly concerns the first algorithm, NLAB, which is constituted of a series of binary choices. More precisely, recall that the relevant information is: a) whether a node is lexical or not; b) whether it has a θ-role or not; c) whether it has Case or not; d) whether it is a sister of C (hence, in an $\overline{\text{A}}$-position) or not (if not, it counts as an A-position). For the chain selection algorithm (CSEL) there are four main constraints: first, A-nodes can only be inserted in A-chains and $\overline{\text{A}}$-nodes can only be inserted in $\overline{\text{A}}$-chains. Second, empty nodes never start a new chain. Third, the closest head is always chosen as a potential chain to which to unify. Finally, only unsaturated chains are chosen.

Consider what would result if NLAB did not check for all of these factors. If (b) were not checked, NLAB′ would not distinguish between feet and intermediate traces, even in the same type of chain, thus it would output four sets of labels: AH, $\overline{\text{A}}$H, {AF, AI}, {$\overline{\text{A}}$F, $\overline{\text{A}}$I}. If (c) were not checked, NLAB″ would not distinguish between A-feet and $\overline{\text{A}}$-feet, thus it would output AH, $\overline{\text{A}}$H, $\overline{\text{A}}$I, AI, {$\overline{\text{A}}$F, AF}. If (d) were not checked, NLAB‴ would output AH, $\overline{\text{A}}$H, {$\overline{\text{A}}$I, AI}, $\overline{\text{A}}$F, AF. If (b), (c) and (d) together were not checked, NLAB⁗ would output AH, $\overline{\text{A}}$H, {$\overline{\text{A}}$I, AI, $\overline{\text{A}}$F, AF}.

In accounting for the growth rate in the space of hypothesis of these modified algorithms, two factors must be taken into consideration. One factor is the number of active chain types, namely, whether a sentence presents only A-chains, only $\overline{\text{A}}$-chains, or both. This factor encodes the second and third restriction of the CSEL algorithm, with the consequence that not all combinations are attempted. The second factor accounts for the growth rate proper, which is reducible to counting the set of k-strings over an n-sized alphabet, hence n^k. Here, k is the number of relevant links in the sentence (for instance, *feet* in NLAB″), and n is given by the size of the set of features collapsed by lifting some of these checks, hence, 2, 2, 2 and 4, respectively.

The hypothesis space in the three algorithms grows in slightly different ways. In NLAB′, where there is no restriction on the number of active chains, the growth rate is n^k. For NLAB″ and NLAB‴, the formula is ${N_A}^k$, where N_A is the number of active chains. Practically, this amounts to 2^k at most, as the number of active chains is not more than 2, because of the restriction requiring that the nearest unsaturated chain be selected. For NLAB⁗, the restriction for active chains no longer holds. In this algorithm, no features are checked, so it is impossible to establish if a chain is saturated or not until structure building ends. Thus, the growth factor is a function of the number of heads seen up to a certain point in the parse, the number of empty categories, and their respective order in the input. Notice that the different size of the collapsed feature set, which is larger for NLAB⁗, is implicitly taken into account by k, as the number of relevant links varies with the size of the collapsed feature sets. For the same sentence, there are more relevant links if the collapsed feature set is larger.

Now, in all cases, growth is exponential in the number of relevant links, while the possible gain obtained by not checking features can be at most logarithmic in the number of potential empty categories. Since the number of potential empty categories is at most 2^f, for f binary features, this gain is expressed as f. Hence, suppressing

Table 6
Growth of Hypothesis Space: S = sentence; TL = Total number of links; RL= Relevant Links; AC = Number of Active Chains; G = Growth rate

S	TL	NLAB′		NLAB″			NLAB‴			NLAB⁗	
		RL	G	RL	AC	G	RL	AC	G	RL	G
3	2	1	2	1	1	1	–	–	1	1	1
4	3	1	2	1	1	1	–	–	1	1	1
5	3	–	–	0	1	1	–	–	1	–	–
6	2	–	–	0	1	1	–	–	1	–	–
7	4	1	2	1	1	1	–	–	1	1	2
8	3	–	–	1	2	2	–	–	1	1	2
9	5	–	–	1	2	2	1	2	2	2	6
10	6	1	2	2	2	4	1	2	2	3	18

feature checks becomes beneficial only if $kf > n^k$. Now notice that $2 \leq n \leq 2^f$. For $n = 2$ and $f = 3$, the inequality is satisfied for $k < 4$. This means that for algorithms NLAB″ and NLAB‴, all sentences with more than three relevant links are computed faster if features are checked. For $n = 4$, i.e. algorithm NLAB⁗, the inequality is never satisfied.[17]

The results of some calculations are reported in Table 6. The numbers in the "sentence" column refer to the type of construction, as exemplified in Figure 1 (sentence types 1 and 2 are not considered because they contain only trivial chains). If one considers a sentence such as *Who did you say that John thought that Mary seemed to like?*, with four gaps and four heads, there are 96 hypotheses about chain formation to explore using NLAB⁗. Clearly, checking features and using them for building chains, and keeping the hypothesis search space small, is beneficial in most cases.

Extensibility. These algorithms deal in detail with the somewhat neglected problem of what to do when more than one chain has to be constructed. They do not discuss specifically the issues of adjunction or rightward movement. However, they could be extended.

In the unextended algorithm, the postulation and structural licensing of empty categories is always performed by the same mechanism. According to the ECP (as formulated in Rizzi 1990, 25; Cinque 1990; Chomsky 1986b, among others), for an empty category to be licensed, two conditions must be satisfied: the empty category must be within the maximal projection of a lexical head to be licensed structurally, and it must be identified by an antecedent. The structural licenser and the antecedent need not be the same element. In fact, they hardly ever are. Whether movement is to the left or to the right does not affect structural licensing (which is here performed by the conditions that apply to the reduction of an ϵ-rule).

Rightward movement requires an extension of the algorithm to incorporate the empty category in a chain. An empty category that is the foot of rightward movement must be licensed structurally, before its antecedent is seen. When the NP that is the antecedent (head of chain) is found, it starts a new chain, according to CSEL. Therefore, an extension is needed to check if there are any empty categories wait-

17 Note that here I am assuming that checking a feature and checking a chain have the same computational cost, which is an approximation, as a chain cannot be checked with a single operation.

ing to be identified. This requires computing c-command, to check that the NP can be the antecedent of the empty category, if one is found. Explicit computation of c-command is not needed for leftward movement, since it is a property of the stack of an LR parser that it encodes c-command. Only the fact that a constituent contains an "orphan" empty category must be recorded, perhaps by composite categories. If, on the stack, the antecedent immediately follows the element that contains the empty category, c-command obtains (as a consequence of binary branching), and the empty category can be unified to the antecedent.[18]

4.3 Incrementality

For the parser presented here to be able to perform structural interpretation of empty categories, features must be assigned from left to right while scanning the input. In particular, case features must be assigned immediately to an empty element of the chain, as case is crucially used by NLAB to determine if the empty category belongs to an A-chain or an $\overline{\text{A}}$-chain. In general, in order to perform feature assignment incrementally, an LR parser augmented by constraints must be able to assign features at any point in a rule. Compare, for instance, (7a) and (7b).[19]

(7) a. IP → NP I VP { assign Case if I + fin }

b. IP → NP I { assign Case if I + fin } VP

In a parser that uses rule (7a), case assignment to the NP in subject position is performed only after the VP is seen, even if no information about the VP is needed to perform the case-assigning action. This means that if IP is the topmost node, Case assignment to the subject will occur only when the entire tree for the sentence is built. A parser that uses rule (7b), on the other hand, would assign Case as soon as the necessary information, namely, the value of the Inflection node, is available. Formally, this is the problem of transforming an L-attributed grammar into an S-attributed grammar.[20] An L-attributed grammar G_L can be evaluated by an LR parser if G_L is transformed into a grammar G_S such that the actions that perform attribution in G_L always occur at the end of a production in G_S.

Although it is *literally* true that LR parsers can evaluate actions only on reductions (i.e., they only operate on S-attributed grammars), there are techniques to transform an

18 Alternatively, one could adopt the (linguistically radical) position that rightward movement does not exist, as proposed by Kayne (1994). Although this generalization seems to be true for head movement, Kayne's proposal is more controversial for maximal projections. Discussion of these issues falls completely outside of the topic of the present paper.

19 LR parsers have been criticized as possible models of linguistic performance because, supposedly, they cannot perform any parsing action until the end of the sentence is seen, if the structure is right-branching (see Abney 1989; Steedman 1989). Stabler (1991) argues that this criticism adopts an unnecessarily naive view of the interleaving of structure building and interpretation in an LR parser. Shieber and Johnson (1993) show that by using the (set of) left contexts encoded in the LR table, interpretation can be done incrementally. I show that a similar kind of argument can be built with respect to case feature assignment: case features can be assigned to the links of a chain from left to right while scanning the input. Even if, strictly speaking, mother nodes are built only after their children are built.

20 A grammar G with productions of the sort shown in (7), i.e., a grammar that performs attribute assignment upon reduction of a rule, is called an *attribute grammar* (Irons 1961; Knuth 1968; Correa 1988, 1991.) An attribute grammar is *S-attributed* if all the attribution rules have the form $A.a \rightarrow B.b\ C.c\ \{\ A.a \leftarrow f(B.b, C.c)\ \}$, where the attribute of the parent node is a function of the attributes of the offspring. An attribute grammar is *L-attributed* if all the attribution rules have the form either $A.a \rightarrow \beta.b\ \{\ \beta.b \leftarrow f(A.a)\ \}$ or the form $A.a \rightarrow \beta.b\ \gamma.c\ \{\ \gamma.c \leftarrow f(\beta.b)\ \}$, where the attribute of a node is a function of the attributes of a preceding node in the rule, or of the parent node.

L-attributed rule into an S-attributed rule (Aho, Sethi and Ullman 1977, 282ff discuss the marker nonterminals technique). Such a transformation is possible if the attributes of the tokens on the left of the current token are at a fixed position in the stack.

We can use this S-attribution transformation for Case assignment to the subject (nominative Case or structural case). In English, structural case is assigned to the subject position, if the subject is a sibling of (the projection of) a finite inflectional node. This position can occur both in main and embedded clauses. English is head-initial, and the Specifier precedes the head. These properties interact, so that when the subject NP is reduced, INFL is always the next token in the parsing configuration. Thus, rule (8) can be used.[21]

(8) $IP \rightarrow NP \quad \{\text{Case assign, if I +fin}\} \quad I'$

This rule assigns case correctly only if the attribution is not a function of the subconstituents of I. This is precisely what distinguishes case assigned to the subject (structural case assignment) from other types of case assignments (e.g., case assigned to the object by either a verb or a preposition): it is assigned independently of the properties of the main verb.

The S-attribution transformation is not restricted to languages with the properties of English; it can also be extended to head-final languages. In verb-final languages (German, for example) the subject of the sentence in embedded clauses is not string adjacent to the head of the sentence, as it is in English. However, structural case can be assigned from left to right, since the complementizer, which necessarily marks the left edge of an IP, is obligatory, and the finite complementizer is always different from the infinitival complementizer.

S-attribution could not be performed, however, in parsing a language with all the characteristics given in (9).

(9) a. no overt case marking

b. no distinct finite complementizer

c. verb final

d. right branching in the projections other than the verb

21 At first sight, this might appear as a wild overidealization. In fact, there are both theoretical and empirical reasons to think that this is the right way to idealize the data. A corpus analysis on 111 occurrences of the verb *announce* in the Penn Treebank shows that the subject is followed by an aspectual adverb 11 times, twice by incidental phrases, and 4 times by an apposition. In all other cases the subject and the verb are indeed adjacent. I do not consider appositions and incidentals as challenging for the general claim: incidentals are clearly outside of an $\overline{X}$ structure assigned to the sentence; while appositions are "internal" to the NP, thus when the verb is reached, the phrase sitting on the stack is indeed the NP subject, which can therefore receive Case. The treatment of aspectual adverbs is more complex. There are at least two possible tacks. First, one can notice that adverbs, although they are analysed as maximal projections because they can be modified, never take a complement, thus they are usually limited to a very short sequence of words, and they do not have a recursive structure. A minimum amount of lookahead, even limited to these particular instances of aspectual adverbs, would solve the problem. Clearly, this is an inelegant solution. A more principled treatment comes from recent developments in the theory, that have changed somewhat the representation used for adverbs. Laenzlinger (1993) suggests that all maximal projections have two specifiers, one A and one $\overline{A}$, the higher of the two is the $\overline{A}$-position, which can be occupied by adverbs, if they are licensed by the appropriate head (the Adv-Criterion). For these adverbs, the appropriate head is Asp_0 which we find only with finite verbs. The parser could compile this information and assign case directly, without even waiting to see the (lexical) verb.

Because of property (9a), case could not be inferred from explicit information contained in the input (unlike Japanese or German); because of property (9b) the subject position of an embedded clause would not be unmistakably signalled (unlike German but like Japanese); because of property (9c), the inflectional head would occur after the NP that needs to be assigned case; finally, because of property (9d), an LR parser could give worst-case results (which is not the case for verb-final, but left-branching, languages, like Japanese): it could require the entire sentence to be stacked before starting to assemble it.

Although a problem in principle, this limitation disappears in practice. Inspection of some of the sources on language typology shows that such languages are very difficult to find (Steele 1978; Shopen 1985; Comrie 1981). According to Downing (1978), verb-final languages usually have prenominal relative clauses, which is a sign that they are left branching. Only two verb-final languages have postnominal relative clauses, Persian and Turkish. In Persian, the clause boundary is overtly marked by the suffix *-i* on the antecedent. Moreover, both languages have overt case marking of the subject. Although this is by no means definitive evidence, it suggests that the algorithm for chain formation and feature assignment that I have presented is not obviously inadequate, and that it is applicable to a variety of languages with different properties.

5. Conclusion

The parser described in this paper has been implemented for English. It parses a homogeneous, though small, set of sentences. As a matter of fact, one of the interesting features of this implementation is that it offers a unified treatment of all of the chain types presented above.

The parser has clear limitations due to the fact that it was developed mainly for exploratory purposes. For instance, it deals only with very simple nominal phrases and it does not treat adjunction. In other respects, however, this design lends itself readily to extensions: The structure building and chain formation routines do not rely on characteristics that are found only in English or in a head initial language, as was discussed in the previous section.

In the course of pondering the relation between the grammar and the parser, and mostly how the conceptual *modularity* of current linguistic theories can be implemented, one learns that, in fact, the notion of modular theory is both true and false, at least in its present incarnation. All linguists strive to develop theories that rest on general, abstract principles, which interact in complex ways, so that many empirical facts "fall out" from a few principles. Such a theory is clearly not modular, although highly general and abstract. On the other hand, linguistic concepts operate on different primitives: intuitively, $\overline{X}$-theory, and principles of argument structure or coreference are different objects. Future research must lead in a direction that enables us to define more precisely this basic intuition. Modularity, if it exists, is to be found in the linguistic content, and not in the organization of the theory.

Acknowledgments

I would like to thank those who have helped me in this work: Michael Brent, Bonnie Dorr, Uli Frauenfelder, Paul Gorrell, Luigi Rizzi, Graham Russell, Eric Wehrli, Amy Weinberg, and two anonymous reviewers. All remaining errors are my own.

References

Abney, Steven (1989). "A Computational Model of Human Parsing." *Journal of Psycholinguistic Research*, 18, 129–144.

Aho, Alfred V.; Sethi, Ravi; and Ullman, Jeffrey D. (1977). *Compilers: Principles, Techniques and Tools*. Addison-Wesley

Publishing Company, Reading, MA.
Barton, Edward; Berwick, Robert; and Ristad, Eric (1987). *Computational Complexity and Natural Language*. MIT Press, Cambridge, MA.
Berwick, Robert (1982). "Locality Principles and the Acquisition of Syntactic Knowledge." Doctoral dissertation, MIT, Cambridge, MA.
Berwick, Robert (1990). "Ross was Right: Constraints on Variables." Manuscript, MIT.
Berwick, Robert (1991). "Principle-Based Parsing." In *Foundational Issues in Natural Language Processing*, edited by Peter Sells, Stuart M. Shieber, and Thomas Wasow. MIT Press, 115–226.
Berwick, Robert; Abney, Steven; and Tenny, Carol (1991). *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.
Berwick, Robert, and Weinberg, Amy (1984). *The Grammatical Basis of Linguistic Performance*. MIT Press, Cambridge, MA.
Berwick, Robert, and Weinberg, Amy (1985). "Deterministic Parsing and Linguistic Explanation." *Language and Cognitive Processes*, 1(2), 109–134.
Bresnan, Joan (1978). "A Realistic Transformational Grammar." In *Linguistic Theory and Psychological Reality*, edited by Morris Halle, Joan Bresnan, and George Miller. MIT Press, Cambridge, MA.
Brody, Michael (1984). "On Contextual Definitions and the Role of Chains." *Linguistic Inquiry*, 15(3), 355–380.
Chomsky, Noam (1965). *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.
Chomsky, Noam (1986a). *Knowledge of Language: Its Nature, Origin and Use*. Praeger, New York, New York.
Chomsky, Noam (1986b). *Barriers*. MIT Press, Cambridge, MA.
Chomsky, Noam (1988). "Some Notes on Economy of Derivation and Representation." MIT Working Papers in Linguistics 10, MIT, 43–74.
Chomsky, Noam (1992). "A Minimalist Program for Linguistic Theory." Occasional MIT Working Papers in Linguistics 1. (also appeared in *The View from Building 20*, edited by Ken Hale and Jay Keyser. MIT Press, Cambridge, MA, 1993, 1-52.)
Cinque, Guglielmo (1990). *Types of* $\overline{A}$ *Dependencies*. MIT Press, Cambridge, MA.
Comrie, Bernard (1981). *Language Universals and Linguistic Typology*. Basil Blackwell, Oxford.
Correa, Nelson (1988). *Syntactic Analysis of English with Respect to Government-Binding Theory*. Doctoral dissertation, Syracuse University, Syracuse, NY.
Correa, Nelson (1991). "Empty Categories, Chain Binding and Parsing." In *Principle-Based Parsing*, edited by Robert Berwick, Steven Abney, and Carol Tenny. Kluwer Academic Publishers, Dordrecht, 83–122.
Crocker, Matthew (1991). "A Principle-based System for Syntactic Analysis." *Canadian Journal of Linguistics*, 36(1), 1–26.
Crocker, Matthew (1992). "A Logical Model of Competence and Performance in the Human Sentence Processor." Doctoral dissertation, University of Edinburgh, Edinburgh.
Crocker, Matthew (to appear). *Computational Psycholinguistics: An Interdisciplinary Perspective*. Kluwer Academic Publishers, Dordrecht.
Dorr, Bonnie (1987). "UNITRAN: a Principle-based Approach to Machine Translation." AI Lab Memo 100, MIT, Cambridge, MA.
Dorr, Bonnie (1993). *Machine Translation: A View from the Lexicon*. MIT Press, Cambridge, MA.
Downing, Bruce T. (1978). "Some Universals of Relative Clause Structure." In *Universals of Human Language*, edited by Joseph H. Greenberg. Stanford University Press, 375–418.
Earley, Jay (1970). "An Efficient Context-Free Parsing Algorithm." *Communications of the Association for Computing Machinery*, 14, 453–460.
Fong, Sandiway (1990). "Free Indexation: Combinatorial Analysis and a Compositional Algorithm." In *Proceedings, 28th Meeting of the ACL*, Pittsburgh, PA, 105–110.
Fong, Sandiway (1991). "Computational Properties of Principle-based Grammatical Theories." Doctoral dissertation, MIT, Cambridge, MA.
Fong, Sandiway, and Berwick, Robert (1992). "Isolating Cross-linguistic Parsing Complexity with a Principle-and Parameters Parser: a Case Study of Japanese and English." In *Proceedings, COLING 92*, Nantes, France, 631–637.
Frank, Robert (1992). "Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives." Doctoral dissertation, University of Pennsylvania, Philadelphia, PA.
Frazier, Lyn (1985). "Modularity and the Representational Hypothesis." In *Proceedings of NELS 16*, 131–146.
Frazier, Lyn (1990). "Exploring the Architecture of the Language Processing System." In *Cognitive Models of Speech*

Processing, edited by Gerry Altmann, MIT Press, 409–433.
Gazdar, Gerald; Klein, Ewan; Pullum, Geoffrey; and Sag, Ivan (1985). *Generalized Phrase Structure Grammar*. Blackwell, Oxford.
Grimshaw, Jane (1986). "Subjacency and the S/S' Parameter." *Linguistic Inquiry*. 17(2), 364–369.
Haegemann, Liliane, and Zanuttini, Raffaella (1991). "Negative Heads and the *Neg* Criterion." *The Linguistic Review*, 8 (2/4), 233–251.
Irons, Edgar (1961). "A Syntax Directed Compiler for ALGOL 60." *Communications of the Association for Computing Machinery*, 4(1) : 51–55.
Johnson, Mark (1989). "Parsing as Deduction: The Use of Knowledge of Language." *Journal of Psycholinguistic Research*, 18(1), 233–251.
Kashket, Michael (1991). "A Parameterized Parser for English and Warlpiri." Doctoral dissertation, MIT, Cambridge, MA.
Kayne, Richard (1994). "The Antisymmetry of Syntax." *Linguistic Inquiry Monograph 25*, MIT Press, Cambridge, MA.
Knuth, Donald E. (1965). "On the Translation of Languages from Left to Right." *Information and Control*, 8: 607–639.
Knuth, Donald E. (1968). "Semantics of Context-free Languages." *Mathematical Systems Theory*, 2: 127–145.
van de Koot, Hans (1990). *Essay on the Grammar-Parser Relation*. Foris, Dordrecht.
van de Koot, Hans (1991). "Parsing with Principles: on Constraining Derivations." UCL Working Papers in Linguistics, University College, London, 369–396.
Kornai, Andrais, and Pullum, Geoffrey (1990). "The X-bar Theory of Phrase Structure." *Language*, 66, 24–50.
Laenzlinger, Christopher (1993). "Principles for a Formal Account of Adverb Syntax." Geneva Generative Papers, 1(2), University of Geneva, 47–75.
Marcus, Mitchell (1980). *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge MA.
Merlo, Paola (1992). "On Modularity and Compilation in a Government and Binding Parser." Doctoral dissertation, University of Maryland, College Park, MD.
Merlo, Paola (to appear). *Parsing with Principles and Classes of Information*. Kluwer Academic Publishers, Dordrecht.
Phillips, John (1992). "A Computational Representation for Generalized Phrase Structure Grammars." *Linguistics and Philosophy*, 15(3): 255-287.
Phillips, John, and Thompson, Henry (1985). "GPSGP: A Parser for Generalized Phrase Structure Grammars." *Linguistics*, 23(2), 245–262.
Rizzi, Luigi (1982). *Issues in Italian Syntax*. Foris, Dordrecht.
Rizzi, Luigi (1990). *Relativized Minimality*. MIT Press, Cambridge, MA.
Rizzi, Luigi (1991). "Residual Verb Second and the Wh-Criterion," Technical Reports in Formal and Computational Linguistics, 2, University of Geneva, Geneva.
Schabes, Yves (1991). "Polynomial Time and Space Shift-reduce Parsing of Arbitrary Context-free Grammars." In *Proceedings, 29th Meeting of the ACL*, Berkeley, CA, 106–113.
Shieber, Stuart M., and Johnson, Mark (1993). "Variations on Incremental Interpretations." *Journal of Psycholinguistic Research*, 22(2), 287–319.
Shopen, Timothy (1985). *Language Typology and Syntactic Description*. Cambridge University Press, Cambridge, England.
Sportiche, Dominique (1992). "Clitic Constructions." Manuscript, UCLA.
Stabler, Edward (1990). "Relaxation Techniques for Principle-based Parsing." Manuscript, UCLA.
Stabler, Edward (1991). "Avoid the Pedestrian's Paradox." In *Principle-Based Parsing*, edited by Robert Berwick, Steven Abney, and Carol Tenny. Kluwer Academic Publishers, Dordrecht, 199–238.
Stabler, Edward (1992). *The Logical Approach to Syntax*. MIT Press, Cambridge, MA.
Stabler, Edward (1994). "The Finite Connectivity of Linguistic Structure." In *Perspectives on Sentence Processing*, edited by Charles Clifton, Lyn Frazier, and Keith Rayner. Lawrence Erlbaum, Hillsdale, NJ, 303–336.
Steedman, Mark (1989). "Grammar, Interpretation and Processing from the Lexicon." In *Lexical Representation and Processes*, edited by William Marslen-Wilson. MIT Press, Cambridge, MA, 463–504.
Steele, Susan (1978). "Word Order Variation." In *Universals of Human Language*, edited by Joseph H. Greenberg. Stanford University Press, 585–623.
Thompson, Henry (1982). "Handling Metarules in a Parser for GPSG." Research Paper 175, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK.
Tomita, Masaru (1986). *Efficient Parsing for Natural Language*. Kluwer, Hingham, MA.

Appendix A

A.1 Grammar 1

```
s       : x2;                   /* 1 */
x2      : y2    x1;             /* 2 */
x2      : y2    x2;             /* 3 */
x2      : x1    y2;             /* 4 */
x2      : x2    y2;             /* 5 */
x1      : x0    y2;             /* 6 */
x1      : y2    x0;             /* 7 */
x1      : x1    y2;             /* 8 */
x1      : y2    x1;             /* 9 */
y2      : x2                    /* 10 */
        | w2                    /* 11 */
        | /*empty*/ ;           /* 12 */
x0      : w0                    /* 13 */
        | /* empty */;          /* 14*/
x2      : x1;                   /* 15*/
x1      : x0;                   /* 16 */
```

A.2 Grammar 2

```
s       : c2;                   /* 1 */
c2      : y2    c1;             /* 2 */
c1      : c0    y2;             /* 3 */
c1      : y2    c1;             /* 4 */
c1      : c1    y2;             /* 5 */
c2      : y2    c2;             /* 6 */
i2      : y2    i1;             /* 7 */
i1      : i0    y2;             /* 8 */
i1      : y2    i1;             /* 9 */
i1      : i1    y2;             /* 10 */
i2      : y2    i2;             /* 11 */
v2      : y2    v1;             /* 12*/
v1      : v0    y2;             /* 13 */
v1      : y2    v1;             /* 14 */
v1      : v1    y2;             /* 15*/
v2      : y2    v2;             /* 16*/
a2      : y2    a1;             /* 17*/
a1      : a0    y2;             /* 18*/
a1      : y2    a1;             /* 19 */
a1      : a1    y2;             /* 20 */
a2      : y2    a2;             /* 21 */
p2      : y2    p1;             /* 22*/
p1      : p0    y2;             /* 23 */
p1      : y2    p1;             /* 24 */
p1      : p1    y2;             /* 25 */
p2      : y2    p2;             /* 26 */
d2      : y2    d1;             /* 27 */
d1      : d0    y2;             /* 28*/
d1      : y2    d1;             /* 29 */
d1      : d1    y2;             /* 30 */
d2      : y2    d2;             /* 31*/
c2      : c1;                   /* 32 */
c1      : c0;                   /* 33 */
i2      : i1;                   /* 34 */
i1      : i0;                   /* 35 */
v2      : v1;                   /* 36 */
v1      : v0;                   /* 37 */
a2      : a1;                   /* 38 */
a1      : a0;                   /* 39 */
```

```
p2      : p1;                    /* 40 */
p1      : p0;                    /* 41 */
d2      : d1;                    /* 42 */
d1      : d0;                    /* 43 */
y2      : /*empty*/|n2|c2|i2|v2|a2|p2; /* 44-51 */
```

A.3 Grammar 3

```
s       : i2 | c2 ;
c2      : y2 c1| c1| /* empty */ ;
c1      : c0  i2 | c0;
c0      : c | /*empty*/ ;
i2      : y2 i1 | i1 | /* empty */;
i1      : i0 v2 | i0;
i0      : i | /*empty*/ ;
v2      : y2 v1 | /*empty */|v1 ;
v1      : v0int p2 ;
v0int   : vint| /* empty */;
v1      : v0t n2;
v0t     : vt | /*empty*/ ;
v1      : v0rais i2;
v0rais  : vrais | /*empty*/ ;
v1      : v0int c2;
y2      : c2 | i2 | n2 | v2 | p2
        | /* empty */ ;
n2      : n | /* empty */;
p2      : p0 y2 | /* empty */;
p0      : p |/*empty*/;
```