NEWCAT: Parsing
Natural Language Using
Left-Associative Grammar
The verb *to parse* means "to describe grammatically by stating the part of speech and explaining the syntactical relationship" (Webster's New Collegiate Dictionary). The noun *parser* refers to computer programs which grammatically analyze sentences or text of a language. Parser programs have been written for both formal languages (programming languages)\(^1\) and natural languages (e.g. English or German).

Natural language parsers are a precondition of comfortable man-machine communication. Automatic speech recognition, data base interfaces, machine translation, and a host of other important applications require efficient natural language parsers. For this reason natural language parsing has always been a primary goal of non-numeric programming.

The construction of natural language parsers is an interdisciplinary enterprise, requiring the cooperation of linguists and computer scientists. This cooperation is characterized by a convenient division of labor. The linguists take pride in basing their grammars solely on linguistic grounds, such as natural language "universals". Whether or not their grammar is suitable for parsing programs is not considered an issue. The computer scientists, on the other hand, take pride in their ability to implement any grammar as a computer program as long as the grammar is a reasonably explicit formalism. How a grammar is implemented on a computer is considered irrelevant as long as the program runs reasonably fast, and the display of the output closely resembles the syntactic representations envisioned by the linguist.

However, despite great efforts for over thirty years, the parsing of natural language is still an unsolved mystery. There are many different parsing algorithms, each with its own merits and limitations. But somehow the structures found in natural language do not seem amenable to a general and efficient analysis with existing parsing programs. This is taken by many people as evidence that it is simply impossible to build computers which analyze (and understand) natural language with the ease and efficiency of a native speaker.

Why is the computational analysis of natural language such a difficult task? Is natural language or the theoretical approach at fault? So far the widely accepted separation of the "declarative" (grammatical) and the "procedural" (computational) aspects of parsing has prevented the investigating of whether contemporary formal grammars of natural language provide a suitable basis for parsing programs.

In this book it is shown that constituent structure analysis, predominant in today's grammars, induces an irregular order of linear composition which is the direct cause of extreme computational inefficiency. An alternative left-associative grammar is proposed, which operates with a regular order of linear compositions. Left-associative grammar is based on building up and cancelling valencies. Left-associative parsers differ from all other systems in that the history of the parse doubles as the linguistic analysis. The efficiency and descriptive power of left-associative grammar is illustrated with two left-associative natural language parsers: one for German and one for English.

Munich/Stanford, May 1986

R. Hausser

---

\(^1\) For conversion of higher level statements into assembly or machine language in compilers.
Acknowledgements

The research for this book was supported by a Heisenberg grant from the Deutsche Forschungsgemeinschaft, West Germany. The parser programs were written during two 3 month stays at the Center for the Study of Language and Information, Stanford University, in 1984 and 1985. The German parser NEWCAT (a name derived from ‘NEW CATegorial approach’) was first described in the CSLI publication IN-CSLI-85-5 of December 1985.

Conceptually NEWCAT is based on many years of linguistic research in syntax and semantics which would have remained dormant in the form of paper and pencil studies without the opportunity to work with the computing facilities and the people maintaining them at CSLI. I would like to thank Betsy Macken, John Perry, and Stanley Peters for sponsoring my stays there.

At various stages in the development of the programs, I received help from people who were or still are working at CSLI. Doug Cutting, Frederic Vander Elst, Mike Moore, Atty Mullins, Paul Oppenheimer, and Greep (alias Steven Tepper) spent long hours figuring out what I wanted and how to write it in LISP. Brad Horak and Joe Zingheim maintained the dandytiger in my office in top running condition. Emma Pease helped me with formatting this book in Latex. David Brown, Marjorie Maxwell, and Susi Parker helped me with the practical aspects of life at CSLI.

I benefitted especially from discussions with Tryg Ager, Pentti Kanerva, Ron Kaplan, Martin Kay, Joachim Laubsch, Eric Ostrom, Carl Pollard, Stuart Shieber, and Hans Uszkoreit. Steven Tepper and Theo Vennemann made detailed comments on the semifinal draft. I am indebted to Dikran Kargueuzian for help and advice on several occasions. Last, but not least, I would like to thank Deborah Kerman for proofreading the book. All remaining mistakes of style and substance are the responsibility of the author.

This book was reproduced from a camera-ready copy supplied by the author who gladly acknowledges the generous access to computers provided by CSLI.
# Contents

**Introduction**  

1 Left-associative grammar  
1.1 The constituent structure paradox .......................... 13  
1.2 The irregular left-to-right order of constituent structure ...... 16  
1.3 Left-associative versus categorial grammar .................. 18  
1.4 Left-associative trees as structured lists .................. 22  
1.5 Parsing continuous text .................................... 27  

2 Left-associative parsing  
2.1 A production system with a simple control structure ........... 33  
2.2 Modularity and expansion of the grammar .................... 36  
2.3 The parsing history as linguistic analysis ................... 41  
2.4 Parsing ungrammatical input .................................. 50  
2.5 Some computational contrasts with other parsers .............. 52  

3 The category system of left-associative grammar ............... 57  
3.1 Syntactic categories and syntactic rules .................... 57  
3.2 Determiner-noun agreement in DCAT .......................... 61  
3.3 Adjective agreement in DCAT ................................. 65  
3.4 Combining noun phrases and verbs in DCAT .................... 70  
3.5 Treating different kinds of noun phrases in DCAT .............. 74  

4 The local nature of possible continuations .................... 95  
4.1 The treatment of word order in DCAT .......................... 95  
4.2 Passive and other constructions with auxiliaries in DCAT ....... 112  
4.3 Center-embedded versus extraposed relative clauses in DCAT ... 124  
4.4 Syntactic equivalence in left-associative grammar ............ 131  
4.5 Remarks on the lexicon of left-associative grammar ........... 145
5 A left-associative fragment of English  
  5.1 A distinctive categorization for English noun phrases .............. 159  
  5.2 Combining noun phrases and verbs in ECAT ..................... 172  
  5.3 Passive and other constructions with auxiliaries in ECAT .......... 188  
  5.4 Relative clauses in ECAT ................................... 198  
  5.5 Wh-interrogatives in ECAT ................................. 207  

Appendices

A The LISP functions of DCAT .................................. 219  
  A.1 The motor of a left-associative parser ........................ 220  
  A.2 The linguistic rules and rule packages ...................... 226  
  A.3 Auxiliary functions of the linguistic rules ................... 245  
  A.4 Alphabetical list of DCAT-functions .......................... 249  
  A.5 The definitions of DLEX ................................... 253  

B A selection of DCAT test examples ............................ 273  
  B.1 List of category segments used in DCAT and DLEX ............... 274  
  B.2 Declaratives with finite main verbs ........................... 276  
  B.3 Declaratives with auxiliaries and non-finite main verbs ......... 280  
  B.4 Declaratives with topicalized non-finite verbs ................ 285  
  B.5 Various predicate constructions in declarative main clauses .... 291  
  B.6 Various predicate constructions in subordinate clauses .......... 298  
  B.7 Passive constructions in declarative main clauses ............. 302  
  B.8 Q-Passives in declarative main clauses ........................ 313  
  B.9 Passive in subordinate clauses ................................ 320  
  B.10 Q-passives in subordinate clauses ............................ 326  
  B.11 Multiple modal infinitives in main clauses ................... 334  
  B.12 Multiple modal infinitives in subordinate clauses ............. 338  
  B.13 Obligatory versus optional adverbs ........................... 347  
  B.14 The preposition hinter in various constructions .............. 352  
  B.15 Adsentential clauses and adverbs in various positions ......... 356  
  B.16 Relative clause agreement .................................. 360  
  B.17 Sentential complements .................................... 375  
  B.18 Infinitives with zu in main clauses ........................... 382  
  B.19 Infinitives with zu in subordinate clauses ................... 395  
  B.20 Separable verbal prefixes ................................... 403  
  B.21 Yes/no-interrogatives ...................................... 408  
  B.22 Wh-interrogatives ......................................... 413  

C A selection of ECAT test examples ............................. 425  
  C.1 List of category segments used in ECAT and ELEX ............... 425  
  C.2 Active voice constructions using the verb give ................. 429  
  C.3 Passive voice constructions using the verb give ............... 434
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.4</td>
<td>Genitive constructions</td>
<td>440</td>
</tr>
<tr>
<td>C.5</td>
<td>Auxiliaries taking noun phrases as the second argument</td>
<td>442</td>
</tr>
<tr>
<td>C.6</td>
<td>Auxiliaries taking an adjective as the second argument</td>
<td>444</td>
</tr>
<tr>
<td>C.7</td>
<td>Nominative agreement and the auxiliary <em>be</em></td>
<td>445</td>
</tr>
<tr>
<td>C.8</td>
<td>Yes/no-interrogatives and related declaratives</td>
<td>448</td>
</tr>
<tr>
<td>C.9</td>
<td>Wh-interrogatives</td>
<td>455</td>
</tr>
<tr>
<td>C.10</td>
<td>The interrogative determiner <em>which</em></td>
<td>465</td>
</tr>
<tr>
<td>C.11</td>
<td><em>That</em>-clauses</td>
<td>467</td>
</tr>
<tr>
<td>C.12</td>
<td>Wh-interrogatives with <em>that</em>-clauses</td>
<td>470</td>
</tr>
<tr>
<td>C.13</td>
<td>Passives in subordinate clauses</td>
<td>474</td>
</tr>
<tr>
<td>C.14</td>
<td>Relative clauses modifying sentence final noun phrases</td>
<td>479</td>
</tr>
<tr>
<td>C.15</td>
<td>Relative clauses modifying mid-sentence noun phrases</td>
<td>486</td>
</tr>
<tr>
<td>C.16</td>
<td>Relative clauses modifying sentence initial noun phrases</td>
<td>492</td>
</tr>
<tr>
<td>C.17</td>
<td>The relative pronoun <em>who as</em> subject and object</td>
<td>495</td>
</tr>
<tr>
<td>C.18</td>
<td>Mixing relative clauses and <em>that</em>-clauses</td>
<td>501</td>
</tr>
<tr>
<td>C.19</td>
<td>Wh-interrogatives with relative clauses</td>
<td>506</td>
</tr>
<tr>
<td>C.20</td>
<td>Declaratives and interrogatives with “Wh-movement”</td>
<td>508</td>
</tr>
<tr>
<td>C.21</td>
<td>Subordinate clauses with and without complementizers</td>
<td>514</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td><strong>List of computer-generated sample derivations</strong></td>
<td>521</td>
</tr>
<tr>
<td>D.1</td>
<td>The DCAT derivations</td>
<td>521</td>
</tr>
<tr>
<td>D.2</td>
<td>The ECAT derivations</td>
<td>532</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>References</strong></td>
<td>539</td>
</tr>
</tbody>
</table>
Introduction

This book describes a **left-associative** approach to the syntax and semantics of natural language. A left-associative system analyzes a sentence from left to right, first combining word 1 and word 2, then adding word 3, then adding word 4, etc., until there are no more ‘next words’. Conceptually, the left-associative approach is based on the notion of **possible continuations**: after word n has been added, the grammar specifies precisely what the categories of word n+1 may be.

The formal description of the possible continuations at the end of a ‘sentence start’ may be used to choose a grammatically compatible ‘next word’ (generation), or it may be used to decide whether a given ‘next word’ is grammatically compatible with the sentence start (parsing). Left-associative grammar is suited equally well for generation and for parsing.

Analyzing language in a linear, left-associative fashion in terms of possible continuations represents a substantial departure from contemporary linguistic analysis, which works in terms of constituent structures. Constituent structure analysis takes place in the theoretical space between the root of the constituent structure tree (usually called the S-node), representing an abstract category, and the leaves of the tree, representing the concrete words of the sentence (called the terminal symbols). Constituent structure analysis views the whole sentence by looking from the root of the tree to the terminal symbols (top-down analysis), or from the terminal symbols to the root of the tree (bottom-up analysis).\(^1\)

Left-associative analysis, on the other hand, takes place in the theoretical space between the first and last words of a sentence or text. The only combinations permitted are between sentence starts and next words. The resulting trees are of a completely regular, left-associative nature (see example 1.3.4 below). The ‘root’ of a left-associative tree is not an abstract start symbol, but the result of the last combination of a sentence start and a next word. Left-associative trees are built only from the bottom up; every combination of a sentence start and a next word results in a new ‘root’.

In other respects, however, left-associative grammar is very traditional. Linguistically, left-associative analyses are based solely on the concepts of **valency**

---

\(^1\) Constituent structure trees are usually drawn upside down, with the root at the top and the terminal symbols at the bottom. See 1.2.1 as an example.
and agreement. These, in turn, are implemented on the basis of such traditional notions as case (nominative, genitive, dative, accusative), number (singular, plural), gender (masculine, feminine, neuter), and person. Other notions used are traditional word classes such as verb (of a certain valency), noun (of a certain number, gender, and - in German - case), adjective, adverb, preposition, etc.

Left-associative grammar resulted from attempts to build a parser for the context-free categorial system presented in Surface Compositional Grammar (SCG, Hausser 1984). SCG argues that categorial grammar\(^2\) has formal properties which differ intuitively, methodologically, and heuristically from phrase structure grammar.\(^3\) The point of departure for this argument is the weak equivalence between certain categorial grammars and certain phrase structure grammars proven informally in Bar-Hillel (1953).

This weak equivalence in generative power has been interpreted by linguists working in the paradigm of phrase structure grammar as if there were no important differences between categorial grammar and phrase structure grammar. They saw no reason to explore possible differences in the descriptive potential of the two kinds of formal grammar. Instead, systems which had originated within categorial grammar, such as the various systems of Montague (1974), where quickly redesigned: one kept their characteristic model-theoretic semantics but replaced their categorial surface syntax with a corresponding phrase structure system (with or without a transformational component).

SCG sets out to show that "the choice between phrase structure grammar and categorial grammar is not merely a matter of terminological habit or professional expedience", but has far reaching consequences on the resulting linguistic analyses. For instance, the categories of categorial grammar are combinatorially and denotationally transparent, while those of phrase structure grammar are opaque.\(^4\) SCG illustrates the descriptive potential of pure (i.e. context-free) categorial grammar by presenting the syntax and the semantics of a relatively large fragment of English.

The implementation of the SCG system as a parser was intended to further explore the descriptive potential of pure categorial grammar. Rather than attempting to cast the English fragment defined in SCG into an existing parser framework, such as an ATN, a chart parser, or an Early algorithm, it seemed more suitable for our purposes to program the categorial principles of SCG directly. In the course of this project it became apparent that, well-motivated as the grammatical system of SCG seemed from a linguistic point of view, it was not a very suitable basis for an efficient parsing program. The reason for this

---

\(^2\)In the tradition of Lesniewski (1929), Ajdukiewicz (1935), Bar-Hillel (1953), and Montague (1974).

\(^3\)In the tradition of Post (1936), Chomsky (1957), and Jackendoff (1977). Phrase structure grammars are a linguistic application of Post's rewriting rule systems.

\(^4\)These notions are briefly explained in section 3.1. For a more detailed discussion see SCG.
is not an inherent property of categorial grammar but the irregular nature of conventional constituent structure trees\(^5\), which are common to both categorial grammar and phrase structure grammar.

After this experience we switched to building a parser for German. We wanted to apply the methodological and heuristic principles of SCG in a new context. The use of a different natural language as the object of formal description and the goal of a computationally efficient implementation of this description soon made the syntactic formalism develop independently of the earlier pencil and paper system. We realized that SCG had not gone far enough. In hindsight, the formalism of SCG may be regarded as a last ditch attempt to save constituent structure analysis, albeit in the form of ‘orthogonal trees’\(^6\), while the new research has led us to the conclusion that constituent structure analysis should be abandoned completely.

The basic idea of left-associative parsing was implemented as a LISP-program in December 1984. After returning to CSLI in March, 1985, the linguistic scope of the parser expanded very quickly. NEWCAT (for ‘NEW CATegorical approach’) handles the word order of German in declarative and interrogative main clauses with and without auxiliaries, as well as in subordinate clauses. It handles all free word order variations, center embedded relative clauses of arbitrary depth, extraposed relative clauses, auxiliaries, modals, passive voice in main and subordinate clauses, multiple infinitives, conjunction, gapping, obligatory and optional adverbs, adverbial clauses, prepositional clauses, discontinuous elements, and the agreement between determiners, adjectives, nouns and verbs. In May 1985 the parser NEWCAT was demonstrated at Stanford University and the Stanford Research Institute.

During a third stay at CSLI from late September to December 1985, the principles of NEWCAT were extended in three directions: a revised left-associative German parser, called DCAT, a new left-associative English parser, called ECAT, and a new left-associative parser for propositional logic with truth-value assignment, called LOGCAT. The purpose of these extensions was to demonstrate that left-associative grammar is a general approach, applicable to different natural languages as well as to formal languages.

Since left-associative grammar was abstracted from comprehensive and efficiently running parsing programs, it will be convenient and illuminating to describe the linguistic theory at least in part by explaining how the programs proceed in the analysis of sentences. When we discuss general principles of left-associative parsing, we will use the name of the first left-associative parsing system, NEWCAT. When we turn to the specifics of the parsers for German and English, we will use the names DCAT and ECAT, respectively.

Chapter 1 provides the linguistic background of left-associative grammar. It describes the problems of constituent structure analysis, and explains the

\(^5\)Cf. section 1.2.

\(^6\)See SCG, chapter 3.
relationship between categorial grammar and left-associative grammar. The potential of left-associative grammar is illustrated with some examples of parsing continuous text.

Chapter 2 explains the relationship between the motor, the linguistic rules, the rule packages, and the lexicon of a left-associative parser. NEWCAT is shown to be a highly modular system, with regard to both the relation between the grammar and the parsing procedure, and the different parts of the grammar itself. In NEWCAT the parsing history doubles as the linguistic analysis. This aspect of left-associative parsing is illustrated by going through the derivation of a sentence word by word.

Chapter 3 presents the category system of left-associative grammar. The principles of categorization and the relationship between categories and rules, are explained in terms of a detailed analysis of German noun phrases. It is shown that in German the noun should not be defined as the 'head' of the noun phrase.

Chapter 4 illustrates the descriptive power of left-associative grammar in the analysis of complex syntactic constructions, such as word order phenomena in German main and subordinate clauses, passive and complex auxiliary constructions, center-embedded and extraposed relative clause with arbitrary stacking depth, syntactic ambiguity, etc.

Chapter 5 describes the category system and the basic combinatorial properties of a left-associative analysis of English. It explains the treatment of agreement and word order, and discusses a number of constructions such as passive voice, relative clauses, and interrogatives. The linguistic analysis is implemented in the left-associative parser ECAT/ELEX.

In appendix A the LISP code of the parser program DCAT/DLEX is documented. For many years, linguists have used formal languages to describe natural languages, for methodological and heuristic reasons. LISP, like other programming languages, is a formal language. But in contrast to formalisms defined only on paper, LISP has the advantage of actually running on computers. By printing the LISP code of DCAT/DLEX we provide an explicit formal description of the grammatical system. The descriptive scope of DCAT/DLEX is further documented in appendix B, where 164 additional sample derivations are presented. Appendix C presents 114 sample derivations of the parser ECAT. Appendix D contains a list of all 335 computer-generated derivations presented in this book.

The development of left-associative grammar would not have been possible without the sophisticated programming tools of today's LISP work-stations. They sharpen the perceptions of the researcher in two ways: by demanding the explicit formulation of linguistic hypotheses in the simple and precise code of the programming language, and by being a powerful tool for testing these hypotheses over a much wider range of data than could be managed with paper and pencil.

A closer study will doubtlessly reveal much room for improvement. By
documenting left-associative grammar as completely as possible, we hope to provide a solid foundation for future team efforts to expand the system, as well as continuing in the best tradition of science.
1. Left-associative grammar

The linguistic theory underlying left-associative grammar differs from other contemporary approaches in that the traditional notions of a constituent structure or a dependency structure (cf. Matthews 1981, p. 71-88) are not employed. But like any theory of natural language, left-associative grammar is based on the concept of constituents. Constituents and constituent structures are not the same: the notion of a constituent refers to intuitions of the native speaker, while constituent structure is one of the formalisms invented by linguists to account for these intuitions.

The intuitions underlying the notion of a constituent may be formalized either in terms of possible continuations or in terms of substitutions. For example, the sentence start The man who saw a movie yesterday is similar to the sentence starts The man and John in that each may be continued in the same way, e.g. sleeps on the couch. The expressions in question are also similar in that they may be substituted for each other in a sentence without changing its grammaticality. So on this level the definitions of constituents, in terms of continuations and substitutions, are roughly equivalent.

Substitution tests are an essential tool of linguistic research and constitute the methodological basis for establishing different classes or categories of words, and of word groups or sentence parts. But in the context of left-associative grammar we prefer the definition of constituents in terms of possible continuations rather than in terms of substitutions because it is psychologically more natural: hearers often continue incomplete sentences, but nobody outside of linguistics has any cause to substitute constituents from one sentence into another. The substitution process runs counter to the linear nature of coding and decoding natural language.

1.1 The constituent structure paradox

Constituent structures are defined as trees which fulfill the following two conditions. First, words or constituents which belong together semantically are to be dominated directly and exhaustively by a node; thus gave Mary in the sentence John gave Mary a book.

is directly dominated by the node VP, while there is no node that directly and
1.1 The constituent structure paradox

exhaustively dominates John gave.\(^1\) Second, the branches of the constituent structure tree may not cross. This condition is also known as the 'non-tangling' condition. The term "constituent structure analysis" refers to grammars based on tree structures (or equivalent representations) which attempt to fulfill these two conditions whenever possible.

Constituent structure analysis is subject to a deeply rooted paradox, caused by the attempt to express linguistic intuitions in a way that is formally inadequate. The paradox of constituent structure analysis appears whenever two parts of a sentence which belong together semantically are not adjacent to each other in the sentence. Constructions which cannot be represented in standard constituent structure terms are not a problem of natural language but rather of the descriptive apparatus.

A classic example of the constituent structure paradox is 'discontinuous elements', as in 1.1.1.

1.1.1 John looked the word up.

Given the intuition that looked and up belong more closely together semantically than look and the word, the constituent structure tree for this example should be 1.1.2.

1.1.2 An illustration of the constituent structure paradox:

\[ S \]
\[ VP \]
\[ VP \]
\[ NP \]
\[ NP \]
\[ V \]
\[ looked \]
\[ DET \]
\[ the \]
\[ N \]
\[ word \]
\[ DE \]
\[ up \]

\(^1\)See 1.2.1 below for the constituent structure tree of this example.
A tree like 1.1.2 is prohibited, because it violates the requirement that the lines of the tree may not cross. Yet an alternative tree without the crossing would violate the assumption that parts which belong together semantically must be dominated directly and exhaustively by a node characterizing them as a constituent. The paradox of constituent structure analysis is caused by the empirical fact of natural language that expressions which belong together from the viewpoint of semantic intuition need not be adjacent in the surface.

One way of salvaging constituent structure analysis from this dilemma is by postulating two separate structures, one for each of the two incompatible assumptions. For example, in transformational grammar the semantic intuitions of constituent structure are captured in the 'deep structure', and the facts of the surface serialization in the 'surface structure'.

Another way to resolve the dilemma is to give up one of the two incompatible assumptions. LFG\(^2\) and GPSG\(^3\) abandon the requirement that discontinuous elements must be directly and exhaustively dominated by a node in the constituent structure. Instead, LFG encodes the semantic intuition in the F-structure, defined as a directed acyclic graph, or DAG, while GPSG encodes it in the form of a feature system. Abandoning the semantic aspect of constituent structure analysis in those cases where it leads to problems is an \textit{ad hoc} procedure.\(^4\) It seriously weakens the status of constituent structure, and one might well ask why constituent structure should not be relinquished altogether.

There remains the possibility of abandoning or circumventing the condition that the lines of a tree may not cross in a constituent structure. The latter approach is taken by 'tree-linking grammar', which proposes three-dimensional trees. Given the linear, one-dimensional nature of natural language, three-dimensional constituent structures have a charming touch of extravagance. But the problem with any weakening of the 'non-tangling condition' is that the well-studied mathematical properties of two-dimensional trees, as well as the linguistic concepts and constraints based on the notions of dominance and precedence, are lost.

Left-associative grammar does not propose a new way to circumvent the paradox of constituent structure analysis.\(^5\) Instead left-associative grammar avoids the use of constituent structures altogether. A natural language analysis without constituent structures may seem inconceivable to most contemporary

---

\(^2\) Cf. Bresnan (1982).

\(^3\) Cf. Gazdar et al. (1985).

\(^4\) Despite appeal in GPSG to an 'autonomy of syntax' hypothesis offered as theoretical justification. Curiously, the 'autonomy of syntax' hypothesis was first championed in what GPSG considers its main opponent, i.e. Chomsky's later theories. LFG uses the C-structure only for treating language dependent-facts of surface-serialization and morphology. Thus constituent structure has a comparatively low status in LFG. What LFG takes to be the universal semantic properties of natural language are encoded in the F-structure.

\(^5\) This was done in SCG. In hindsight orthogonal trees may be regarded as a rather abstract attempt to save the basic intuitions of constituent structure analysis by giving up the traditional tree structure.
formal linguists. But this reaction would be based on mistakenly equating syntactic structure and constituent structure — a terminological confusion similar to equating generative grammar and transformational grammar.

Constituent structure analysis is by no means the only kind of syntactic analysis proposed in the literature. There is also, for example, the tradition of dependency structure analysis. "A dependency grammar is a set of grammatical rules, stating the controlling and dependent relations that each class of units can, and by implication cannot, enter into." (Matthews, 1981, p. 81). However, dependency grammar, like constituent structure grammar, views the complete sentence as a two-dimensional hierarchical structure. This leads to problems similar to the dilemma of constituent structure analysis described above. Left-associative grammar differs from dependency grammar in that dependency relations are not defined between individual words or constituents, but solely between the 'sentence start' and the 'next word'.

1.2 The irregular left-to-right order of constituent structure

The descriptive problems of constituent structure analysis are only one reason for adopting a substantially different approach. The second and perhaps more important reason is the fact that constituent structure trees induce an irregular order of left-to-right combinations. Consider the standard constituent structure analysis of a simple example which does not involve the constituent structure paradox, e.g. *John gave Mary a book*.

1.2.1 Example of a constituent structure tree:
Natural language is encoded and decoded in a linear order, from beginning to end. But within constituent structure analysis, intermediate structures like *John gave* or *John gave Mary a* are not acceptable; they do not satisfy the kinds of substitution tests on which this analysis is conceptually based. Thus there are no categories within constituent structure analysis that could be assigned to such expressions.

Left-associative grammar, on the other hand, is based on the notion of 'possible continuations'. *John gave* and *John gave Mary a* are acceptable intermediate structures because these expressions may be continued into complete well-formed sentences. In left-associative grammar *John gave* has the category (V D A), indicating a sentential expression (V) that still needs a dative (D) and an accusative (A) to be a complete sentence. Similarly, *John gave Mary a* has the category (V SN), indicating a sentential expression which needs a singular noun to be a complete sentence (see 1.3.4 below for the left-associative tree). The categories and the rules of left-associative grammar are designed to cancel off as well as build up valencies, and to encode the relevant agreement features.

From a linear point of view, constituent structures induce an irregular order of combinations that is very difficult to predict, and therefore extremely costly computationally. Consider the work of a conventional parser which uses the above constituent structure tree as the basis of its linguistic analysis. Such a parser would first combine *gave* and *Mary* into *gave Mary*. Then it would combine *a* and *book* into *a book*. Then it would combine *gave Mary* and *a book* into *gave Mary a book*. And finally it would combine the first word of the sentence, *John*, with *gave Mary a book*. Every time a conventional parser scans a new word, it is an open question (to be decided by the grammar) whether this word should be combined with expressions already analyzed, or set aside to be combined later with the words not yet scanned.

But each theoretical framework fosters its own kinds of intuitions and rationalizations. Linguists committed to constituent structure analysis refuse to consider the resulting irregular order of left-to-right composition as an issue, by claiming that the combination order of the words in a sentence is generally irrelevant for the essence of constituent structure analysis: the trees are supposed to represent grammatical relations in abstracto.\(^7\)

That parsing necessarily involves a certain combination order of the words in the input string is treated as a 'procedural' problem and as such regarded as irrelevant for, or extraneous to, theoretical linguistic analysis. Natural language parsers which make no use of constituent structure are often dismissed as 'hacks', i.e. pieces of ad hoc programming which lack generality and have no theoretical foundation. While it is true that a natural language parser should be based on a general and clear linguistic theory, it is by no means certain that the theory must be a version of constituent structure analysis.

---

\(^6\) Or left to right, assuming the writing conventions of most Western cultures.

\(^7\) Instead the discussion within linguistics has centered on the ordering of rules, e.g. intrinsic versus extrinsic ordering of transformations and of phonological rules.
Today's natural language parsers vary widely as to whether they analyze constituent structures top-down, bottom-up, left-to-right, right-to-left, left-corner, right-corner, etc. Each ordering has its own merits and faults depending on the constructions handled by the grammar\(^8\). So it may seem that the order of combination is something of dubious theoretical status, best handled by purely computational considerations.

But one may interpret this situation quite differently: that no dominant order of combination has emerged in contemporary natural language parsing is a direct result of the fact that the underlying linguistic analysis imposes an irregular order of left-to-right composition. The widely acknowledged lack of substantial progress in natural language parsing during the last 25 years\(^9\) must be largely attributed to the overwhelming dominance of constituent structure analysis in theoretical and computational linguistics.

### 1.3 Left-associative versus categorial grammar

Left-associative grammar developed in the course of attempting to program the orthogonal syntax of Surface Compositional Grammar (SCG, Hausser 1984), which uses the insights of categorial grammar from Lesniewski 1929 to Montague 1974 in describing the syntax and semantics of a fairly large fragment of natural language. But SCG is still a conventional categorial system in that it makes great efforts to satisfy traditional concepts of constituent structure, although in the form of orthogonal trees. The purpose of the orthogonal trees is to solve the constituent structure paradox by separating the surface order (specified vertically) and the semantic order (specified horizontally).

In our initial project to implement the SCG grammar as a parser, the constituent structure nature of SCG was reflected in the bracketing structure of the input string:

**1.3.1 Bracketing structure of a categorial SCG analysis:**

\[
\text{(John, ((gave, Mary), (the, book)))}
\]

This attempt at writing a categorial parser, a system called CATG (for CATegorical Grammar), worked very nicely on the basis of categorial cancelling rules. But it would analyze inputs only if the proper bracketing was provided, as in 1.3.1. Since a bracketing that reflects constituent structure is of an irregular nature which cannot be predicted, we had to either provide the bracketing of the input string by hand, or test the string automatically for all possible bracketings. Both possibilities where unacceptable, the first because of its dependence on human preprocessing, and the second because of its staggering computational inefficiency.

\(^8\)See Winograd (1984) for a description of the different systems proposed in the literature.

As an alternative we explored the idea of using a completely regular bracketing structure, conceptually based on the notion of possible continuations.

1.3.2 Bracketing structure of a left-associative analysis:

$$(((\text{John,gave}), \text{Mary}), \text{the}), \text{book})$$

Rewriting the rules of CATG to accommodate this regular bracketing structure turned out to be surprisingly easy. Further study of this new approach led to the development of left-associative grammar and the parser NEWCAT.

To clarify the conceptual difference between a left-associative grammar and a categorial grammar, consider the following linguistic analyses of the sentence "John gave Mary the book." (1.3.3 and 1.3.4 below).

1.3.3 Analysis in traditional categorial grammar

```
V
   /
  /  
V S3
   /  
V S3 A
   /  
 John gave Mary the book
   /  
S3 V S3 D A
   /  
 D
   /  
A
   /  
the A SN
   /  
SN
```
1.3.4 Analysis in left-associative grammar

Which grammar is preferable? 1.3.3 and 1.3.4 assign exactly the same categories to the words. These categories are well-motivated. S3 indicates a nominative of third person singular, V S3 D A indicates a verb that takes an S3, a D (for dative noun phrase) and an A (for accusative noun phrase) as its obligatory arguments, and A SN indicates a determiner which takes a singular noun to render an accusative. On the basis of these categories, quite similar categorial combination rules may be defined in 1.3.3 and 1.3.4. So what is the difference?

The difference is in the order of the combinations of words in the sentence. Consider a bottom-up analysis of the two trees. In 1.3.4 the grammar combines John with gave, takes that and combines it with Mary, takes that and combines it with the, and so on. In 1.3.3, on the other hand, the grammar cannot combine John with anything until it has reached the end of the sentence and built up the (V S3)\textsuperscript{10} constituent gave Mary the book. As far as parsing is concerned, the structure in 1.3.4 is computationally more efficient than the structure in 1.3.3, for reasons that are intuitively obvious. Whenever a new word is scanned in 1.3.3, it is difficult to decide whether this word should be combined with

\textsuperscript{10} or VP, to use a more familiar notation