A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing

Mats Wirén
Department of Computer and Information Science
Linköping University
S-581 83 Linköping, Sweden

Abstract

Currently several grammatical formalisms converge towards being declarative and towards utilizing context-free phrase-structure grammar as a backbone, e.g. LFG and PATR-II. Typically the processing of these formalisms is organized within a chart-parsing framework. The declarative character of the formalisms makes it important to decide upon an overall optimal control strategy on the part of the processor. In particular, this brings the rule-invocation strategy into critical focus: to gain maximal processing efficiency, one has to determine the best way of putting the rules to use. The aim of this paper is to provide a survey and a practical comparison of fundamental rule-invocation strategies within context-free chart parsing.

1 Background and Introduction

An apparent tendency in computational linguistics during the last few years has been towards declarative grammar formalisms. This tendency has manifested itself with respect to linguistic *tools*, perhaps seen most clearly in the evolution from ATNs with their strongly procedural grammars to PATR-II in its various incarnations (Shieber et al. 1983, Karttunen 1986), and to logic-based formalisms such as DCG (Pereira and Warren 1980). It has also manifested itself in linguistic *theories*, where there has been a development from systems employing sequential derivations in the analysis of sentence structures to systems like LFG and GPSG which establish relations among the elements of a sentence in an order-independent and also direction-independent way. For example, phenomena such as rule ordering simply do not arise in these theories.

This research has been supported by the National Swedish Board for Technical Development.

In addition, declarative formalisms are, in principle, processor-independent. Procedural formalisms, although possibly highly standardized (like Woods' ATN formalism), typically make references to an (abstract) machine.

By virtue of this, it is possible for grammar writers to concentrate on linguistic issues, leaving aside questions of how to express their descriptions in a way which provides for efficient execution by the processor at hand.

Processing efficiency instead becomes an issue for the designer of the processor, who has to find an overall "optimal" control strategy for the processing of the grammar. In particular (and also because of the potentially very large number of rules in realistic natural-language systems), this brings the *rule-invocation strategy*[1] into critical focus: to gain maximal processing efficiency, one has to determine the best way of putting the rules to use.[2]

This paper focuses on rule-invocation strategies from the perspective of (context-free) chart parsing (Kay 1973, 1982; Kaplan 1973).

Context-free phrase-structure grammar is of interest here in particular because it is utilized as the backbone of many declarative formalisms. The chart-parsing framework is of interest in this connection because, being a "higher-order algorithm" (Kay 1982:329), it lends itself easily to the processing of different grammatical formalisms. At the same time it is of course a natural test bed for experiments with various control strategies.

Previously a number of comparisons of rule-invocation strategies in this or in similar settings have been reported:

[1] This term seems to have been coined by Thompson (1981). Basically, it refers to the spectrum between top-down and bottom-up processing of the grammar rules.

[2] The other principal control-strategy dimension, the *search strategy* (depth-first vs. breadth-first), is irrelevant for the efficiency in chart parsing since it only affects the *order* in which successive (partial) analyses are developed.

Kay (1982) is the principal source, providing a very general exposition of the control strategies and data structures involved in chart parsing. In considering the efficiency question, Kay favours a "directed" bottom-up strategy (cf. section 2.2.3).

Thompson (1981) is another fundamental source, though he discusses the effects of various rule-invocation strategies mainly from the perspective of GPSG parsing which is not the main point here.

Kilbury (1985) presents a left-corner strategy, arguing that with respect to natural-language grammars it will generally outperform the top-down (Earley-style) strategy.

Wang (1985) discusses Kilbury's and Earley's algorithms, favouring the latter because of the inefficient way in which bottom-up algorithms deal with rules with right common factors. Neither Wang nor Kilbury considers the natural approach to overcoming this problem, viz. top-down filtering (cf. section 2.2.3).

As for empirical studies, Slocum (1981) is a rich source. Among many other things, he provides some performance data regarding top-down filtering.

Pratt (1975) reports on a successful augmentation of a bottom-up chart-like parser with a top-down filter.

Tomita (1985, 1986) introduces a very efficient, extended LR-parsing algorithm that can deal with full context-free languages. Based on empirical comparisons, Tomita shows his algorithm to be superior to Earley's algorithm and also to a modified version thereof (corresponding here to "selective top-down"; cf. section 2.1.2). Thus, with respect to raw efficiency, it seems clear that Tomita's algorithm is superior to comparable chart-parsing algorithms. However, a chart-parsing framework does have its advantages, particularly in its flexibility and open-endedness.

The contribution this paper makes is:

- to survey fundamental strategies for rule-invocation within a context-free chart-parsing framework; in particular
- to specify "directed" versions of Kilbury's strategy; and
- to provide a practical comparison of the strategies based on empirical results.

2 A Survey of Rule-Invocation Strategies

This section surveys the fundamental rule-invocation strategies in context-free chart parsing.[3] In a chart-parsing framework, different rule-invocation strategies correspond to different conditions for and ways of predicting new edges[4]. This section will therefore in effect constitute a survey of different methods for predicting new edges.

2.1 Top-Down Strategies

The principle of top-down parsing is to use the rules of the grammar to generate a sentence that matches the one being analyzed.

2.1.1 Top-Down

A strategy for top-down chart parsing[5] is given below. Assume a context-free grammar G. Also, we make the usual assumption that G is cycle-free, i.e., it does not contain derivations of the form $A_1 \rightarrow A_2$, $A_2 \rightarrow A_3, \ldots, A_i \rightarrow A_1$.

> **Strategy 1**[6] (TD)
> Whenever an active edge is added to the chart, if its first required constituent is C, then add an empty active C edge for every rule in G which expands C.[7]

This principle will apply to itself recursively, ensuring that all subsidiary active edges also get produced.

2.1.2 Selective Top-Down

Realistic natural-language grammars are likely to be highly branching. A weak point of the "normal" top-down strategy above will then be the excessive number of predictions typically made: in the beginning of a phrase new edges will be introduced for all constituents, and constituents within those constituents, that the phrase can possibly start with.

One way of limiting the number of predictions is by making the strategy "selective" (Griffiths

[3] I assume a basic familiarity with chart parsing. For an excellent introduction, see Thompson and Ritchie (1984).

[4] Edges correspond to "states" in Earley (1970) and to "items" in Aho and Ullman (1972:320).

[5] Top-down (context-free) chart parsing is sometimes called "Earley-style" chart parsing because it corresponds to the way in which Earley's algorithm (Earley 1970) works. It should be pointed out that the parse-forest representation employed here does not suffer from the kind of defect claimed by Tomita (1985:762, 1986:74) to result from Earley's algorithm.

[6] This formulation is equivalent to the one in Thompson (1981:4).

[7] Note that in order to handle left-recursive rules without going into an infinite loop, this strategy needs a redundancy check which prevents more than one identical active edge from being added to the chart.

and Petrick 1965:291): by looking at the category/categories of the next word, it is possible to rule out some proposed edges that are known not to combine with the corresponding inactive edge(s). Given that top-down chart parsing starts with a scanning phase, the adoption of this filter is straightforward.

The strategy makes use of a reachability relation $\mathcal{R}$ where $A\mathcal{R}B$ holds if there exists some derivation from A to B such that B is the first element in a string dominated by A. Given preterminal lookahead symbol(s) p_j corresponding to the next word, the processor can then ask if the first required constituent of a predicted active edge (say, C) can somehow start with (some) p_j. In practice, the relation is implemented as a precompiled table. Determining if $\mathcal{R}$ holds can then be made very fast and in constant time. (Cf. Pratt 1975:424.)

The strategy presented here corresponds to Kay's "directed top-down" strategy (Kay 1982:338) and can be specified in the following manner.

> **Strategy 2** (TD_s)
> Let $r(X)$ be the first required constituent of the (active) edge X. Let v be the vertex to which the active edge about to be proposed extends. Let $p_1, \ldots, p_n$ be the preterminal categories of the edges extending from v that correspond to the next word. — Whenever an active edge is added to the chart, if its first required constituent is C, then for every rule in G which expands C add an empty active C edge if for some j $r(C) = p_j$ or $r(C)\mathcal{R}p_j$.

2.2 Bottom-Up Strategies

The principle of bottom-up parsing is to reduce a sequence of phrases whose types match the right-hand side of a grammar rule to a phrase of the type of the left-hand side of the rule. To make a reduction possible, all the right-hand-side phrases have to be present. This can be ensured by matching from right to left in the right-hand side of the grammar rule; this is for example the case with the Cocke–Kasami–Younger algorithm (Aho and Ullman 1972).

A problem with this approach is that the analysis of the first part of a phrase has no influence on the analysis of the latter parts until the results from them are combined. This problem can be met by adopting left-corner parsing.

2.2.1 Left Corner

Left-corner parsing is a bottom-up technique where the right-hand-side symbols of the rules are matched from left to right.[8] Once the left-corner symbol has been found, the grammar rule can be used to predict what may come next.

A basic strategy for left-corner chart parsing is given below.

> **Strategy 3**[9] (LC)
> Whenever an inactive edge is added to the chart, if its category is T, then for every rule in G with T as left-corner symbol add an empty active edge.[10]

Note that this strategy will make "minimal" predictions, i.e., it will only predict the *next* higher-level phrases which a given constituent can begin.

2.2.2 Left Corner à la Kilbury

Kilbury (1985) presents a modified left-corner strategy. Basically it amounts to this: instead of predicting *empty* active edges, edges which subsume the inactive edge that provoked the new edge are predicted. A predicted new edge may then be either active or inactive depending on the contents of the inactive edge and on what is required by the new edge.

This strategy has two clear advantages: First, it saves many edges compared to the "normal" left corner because it never produces empty active edges. Secondly (and not pointed out by Kilbury), the usual redundancy check is not needed here since the strategy itself avoids the risk of predicting more than one identical edge. The reason for this is that a predicted edge always subsumes the triggering (inactive) edge. Since the triggering edge is guaranteed to be unique, the subsuming edge will also be unique. By virtue of this, Kilbury's prediction strategy is actually the simplest of all the strategies considered here.

The price one has to pay for this is that rules with empty-string productions (or ϵ-productions, i.e. rules of the form $A \rightarrow \epsilon$), cannot be handled. This might look like a serious limitation since most current linguistic theories (e.g., LFG, GPSG) make explicit use of ϵ-productions, typically for the handling of gaps. On the other hand, context-free grammars can be converted into grammars without ϵ-productions (Aho and Ullman 1972:150).

In practice however, ϵ-productions can be handled in various ways which circumvent the problem. For example, Karttunen's D-PATR system

[8] The *left corner* of a rule is the leftmost symbol of its right-hand side.

[9] This formulation is again equivalent to the one in Thompson (1981:4). Thompson however refers to it as "bottom-up".

[10] In this case, left-recursive rules will not lead to infinite loops. The redundancy check is still needed to prevent superfluous analyses from being generated, though.

does not allow empty productions. Instead, it takes care of fillers and gaps through a "threading" technique (Karttunen 1986:77). Indeed, the system has been successfully used for writing LFG-style grammars (e.g., Dyvik 1986).

Kilbury's left-corner strategy can be specified in the following manner.

Strategy 4 (LC_K)
Whenever an inactive edge is added to the chart, if its category is T, then for every rule in G with T as left-corner symbol add an edge that subsumes the T edge.

2.2.3 Top-Down Filtering

As often pointed out, bottom-up and left-corner strategies encounter problems with sets of rules like $A \rightarrow BC$ and $A \rightarrow C$ (right common factors). For example, assuming standard grammar rules, when parsing the phrase "the birds fly" an unwanted sentence "birds fly" will be discovered.

This problem can be met by adopting *top-down filtering*, a technique which can be seen as the dual of the selective top-down strategy. Descriptions of top-down filtering are given for example in Kay (1982) ("directed bottom-up parsing") and in Slocum (1981:2). Also, the "oracle" used by Pratt (1975:424) is a top-down filter.

Essentially top-down filtering is like running a top-down parser in parallel with a bottom-up parser. The (simulated) top-down parser rejects some of the edges that the bottom-up parser proposes, viz. those that the former would not discover. The additional question that the top-down filter asks is then: is there any place in a higher-level structure for the phrase about to be built by the bottom-up parser? On the chart, this corresponds to asking if any (active) edge ending in the starting vertex of the proposed edge needs this this kind of edge, directly or indirectly. The procedure for computing the answer to this again makes use of the reachability relation $\mathcal{R}$ (cf. section 2.1.2).[11]

Adding top-down filtering to the LC strategy above produces the following strategy.

Strategy 5 (LC_t)
Let v be the vertex from which the triggering edge T extends. Let $A_1, \ldots, A_m$ be the active edges incident to v, and let $r(A_i)$ be their respective first required constituents. — Whenever an inactive edge is added to the chart, if its category is T, then for every rule C in G with T as left-corner symbol add an empty active C edge if for some i $r(A_i) = C$ or $r(A_i)\mathcal{R}C$.

Analogously, adding top-down filtering to Kilbury's strategy LC_K results in the following.

Strategy 6 (LC_{Kt})
(Same preconditions as above.) — Whenever an inactive edge is added to the chart, if its category is T, then for every rule C in G with T as left-corner symbol add a C edge subsuming the T edge if for some i $r(A_i) = C$ or $r(A_i)\mathcal{R}C$.

One of the advantages with chart parsing is direction independence: the words of a sentence do not have to be parsed strictly from left to right but can be parsed in any order. Although this is still possible using top-down filtering, processing becomes somewhat less straightforward (cf. Kay 1982:352). The simplest way of meeting this problem, and also the solution adopted here, is to presuppose left-to-right parsing.

2.2.4 Selectivity

By again adopting a kind of lookahead and by utilizing the reachability relation $\mathcal{R}$, it is possible to limit the number of edges built even further. This lookahead can be realized by performing a dictionary lookup of the words before actually building the corresponding inactive edges, storing the results in a table. Being analogous to the filter used in the directed top-down strategy, this filter makes sure that a predicted edge can somehow be extended given the category/categories of the next word. Note that this filter only affects *active* predicted edges.

Adding selectivity to Kilbury's strategy LC_K results in the following.

Strategy 7 (LC_{Ks})
Let $p_1, \ldots, p_n$ be the categories of the word corresponding to the preterminal edges extending from the vertex to which the T edge is incident. Let $r(C)$ be defined as above. — Whenever an inactive edge is added to the chart, if its category is T, then for every rule C in G with T as left-corner symbol add a C edge subsuming the T edge if for some j $r(C) = p_j$ or $r(C)\mathcal{R}p_j$.

2.2.5 Top-Down Filtering and Selectivity

The final step is to combine the two previous strategies to arrive at a maximally directed version of Kil-

[11] Kilbury (1985:10) actually makes use of a similar relation encoding the left-branchings of the grammar (the "first-relation"), but he uses it only for speeding up grammar-rule access (by indexing rules from left corners) and not for the purpose of filtering out unwanted edges.

bury's strategy. Again, left-to-right processing is presupposed.

> **Strategy 8** (LC_{Kst})
> Let $r(A_i)$, $r(C)$, and p_j be defined analogously to the previous. — Whenever an inactive edge is added to the chart, if its category is T, then for every rule C in G with T as left-corner symbol add a C edge subsuming the T edge if for some i $r(A_i) = C$ or $r(A_i)\mathcal{R}C$ and for some j $r(C) = p_j$ or $r(C)\mathcal{R}p_j$.

3 Empirical Results

In order to assess the practical behaviour of the strategies discussed above, a test bench was developed where it was made possible in effect to switch between eight different parsers corresponding to the eight strategies above, and also between different grammars, dictionaries, and sentence sets.

Several experiments were conducted along the way. The test grammars used were first partly based on a Swedish D-PATR grammar by Merkel (1986). Later on, I decided to use (some of) the data compiled by Tomita (1986) for the testings of his extended LR parser.

This section presents the results of the latter experiments.

3.1 Grammars and Sentence Sets

The three grammars and two sentence sets used in these experiments have been obtained from Masaru Tomita and can be found in his book (Tomita 1986).

Grammars I and II are toy grammars consisting of 8 and 43 rules, respectively. Grammar III with 224 rules is constructed to fit sentence set I which is a collection of 40 sentences collected from authentic texts. (Grammar IV with 394 rules was not used here.)

Because grammar III contains one empty production, not all sentences of sentence set I will be correctly parsed by Kilbury's algorithm. For the purpose of these experiments, I collected 21 sentences out of the sentence set. This reduced set will henceforth be referred to as sentence set I.[12] The sentences in this set vary in length between 1 and 27 words.

Sentence set II was made systematically from the schema

$$\text{noun verb det noun (prep det noun)}^{n-1}.$$

An example of a sentence with this structure is "I saw the man in the park with a telescope...". In these experiments $n = 1, \ldots, 7$ was used.

The dictionary was constructed from the category sequences given by Tomita together with the sentences (Tomita 1986 pp. 185–189).

3.2 Efficiency Measures

A reasonable efficiency measure in chart parsing is the number of edges produced. The motivation for this is that the working of a chart parser is tightly centered around the production and manipulation of edges, and that much of its work can somehow be reduced to this. For example, a measure of the amount of work done at each vertex by the procedure which implements "the fundamental rule" (Thompson 1981:2) can be expressed as the product of the number of incoming active edges and the number of outgoing inactive edges. In addition, the number of chart edges produced is a measure which is independent of implementation and machine.

On the other hand, the number of edges does not give any indication of the overhead costs involved in various strategies. Hence I also provide figures of the parsing times, albeit with a warning for taking them too seriously.[13]

The experiments were run on Xerox 1186 Lisp machines. The time measures were obtained using the Interlisp-D function TIMEALL. The time figures below give the CPU time in seconds (garbage-collection time and swapping time not included; the latter was however almost non-existent).

3.3 Experiments

This section presents the results of the experiments.

In the tables, the fourth column gives the accumulated number of edges over the sentence set. The second and third columns give the corresponding numbers of active and inactive edges, respectively. The fifth column gives the accumulated CPU time in seconds. The last column gives the rank of the strategies with respect to the number of edges produced and, in parentheses, with respect to time consumed (if differing from the former).

Table 1 shows the results of the first experiment: running grammar I (8 rules) with sentence set II (7 sentences). There were 625 parses for every strategy (1, 2, 5, 14, 42, 132, and 429).

[12] The sentences in the set are 1–3, 9, 13–15, 19–25, 29, and 35–40 (cf. Tomita 1986:152).

[13] The parsers are experimental in character and were not coded for maximal efficiency. For example, edges at a given vertex are being searched linearly. On the other hand, grammar rules (like reachability relations) are indexed through precompiled hashtables.

Table 1
Experiment 1: Grammar I, sentence set II

Strategy	Active	Inactive	Total	Time	Rank
TD	1628	3496	5124	62	6
TD_s	1579	3496	5075	58	4 (5)
LC	3104	3967	7071	79	8
LC_t	1579	3496	5075	57	4
LC_K	2873	3967	6840	64	7
LC_{Ks}	697	3967	4664	47	2 (3)
LC_{Kt}	1460	3496	4956	45	3 (2)
LC_{Kst}	527	3496	4023	40	1

Table 2
Experiment 2: Grammar II, sentence set II

Strategy	Active	Inactive	Total	Time	Rank
TD	5015	2675	7690	121	6
TD_s	3258	2675	5933	78	4
LC	7232	5547	12779	192	8
LC_t	3237	2675	5912	132	3 (7)
LC_K	6154	5547	11701	117	7 (5)
LC_{Ks}	1283	5547	6830	70	5 (2)
LC_{Kt}	2719	2675	5394	74	2 (3)
LC_{Kst}	915	2675	3590	41	1

Table 3
Experiment 3: Grammar III, sentence set II

Strategy	Active	Inactive	Total	Time	Rank
TD	13676	5278	18954	910	6 (5)
TD_s	9301	5278	14579	765	4
LC	19522	7980	27502	913	8 (6)
LC_t	9301	5278	14579	2604	4 (8)
LC_K	18227	7980	26207	731	7 (3)
LC_{Ks}	1359	7980	9339	482	2
LC_{Kt}	8748	5278	14026	1587	3 (7)
LC_{Kst}	718	5278	5996	352	1

Table 4
Experiment 4: Grammar III, sentence set I

Strategy	Active	Inactive	Total	Time	Rank
TD	30403	8376	38779	1524	6 (4)
TD_s	14389	8376	23215	1172	4 (2)
LC	42959	19451	62410	2759	8 (6)
LC_t	14714	8376	23090	5843	3 (8)
LC_K	38040	19451	57491	1961	7 (5)
LC_{Ks}	3845	19451	23296	1410	5 (3)
LC_{Kt}	12856	8376	21232	3898	2 (7)
LC_{Kst}	1265	8376	9641	1019	1

Table 2 shows the results of the second experiment: grammar II with sentence set II. This grammar handles PP attachment in a way different from grammars I and III which leads to fewer parses: 322 for every strategy.

Table 3 shows the results of the third experiment: grammar III (224 rules) with sentence set II. Again, there were 625 parses for every strategy.

Table 4 shows the results of the fourth experiment: running grammar III with sentence set I (21 sentences). There were 885 parses for every strategy.

4 Discussion

This section summarizes and discusses the results of the experiments.

As for the three undirected methods, and with respect to the number of edges produced, the top-down (Earley-style) strategy performs best while the standard left-corner strategy is the worst alternative.

Kilbury's strategy, by saving active looping edges, produces somewhat fewer edges than the standard left-corner strategy. More apparent is its time advantage, due to the basic simplicity of the strategy. For example, it outperforms the top-down strategy in experiments 2 and 3.

Results like those above are of course strongly grammar dependent. If, for example, the branching factor of the grammar increases, top-down overpredictions will soon dominate superfluous bottom-up substring generation. This was clearly seen in some of the early experiments not showed here. In cases like this, bottom-up parsing becomes advantageous and, in particular, Kilbury's strategy will outperform the two others.

Thus, although Wang (1985:7) seems to be right in claiming that "...Earley's algorithm is better than Kilbury's in general.", in practice this can often be different (as Wang himself recognizes). Incidentally, Wang's own example (:4), aimed at showing that Kilbury's algorithm handles right recursion worse than Earley's algorithm, illustrates this:

Assume a grammar with rules $S \rightarrow Ac$, $A \rightarrow aA$, $A \rightarrow b$ and a sentence "a a a a b c" to be parsed. Here a bottom-up parser such as Kilbury's will obviously do some useless work in predicting several unwanted S edges. But even so the top-down overpredictions will actually dominate: the Earley-style strategy gives 16 active and 12 inactive edges, totalling 28 edges, whereas Kilbury's strategy gives 9 and 16, respectively, totalling 25 edges.

The directed methods — those based on selectivity or top-down filtering — reduce the number of edges very significantly. The selectivity filter here

turned out to be much more time efficient, though. Selectivity testing is also basically a simple operation, seldom involving more than a few lookups (depending on the degree of lexical ambiguity).

Paradoxically, the effect of top-down filtering was to degrade time performance as the grammars grew larger. To a large extent this is likely to have been caused by implementation idiosyncrasies: active edges incident to a vertex were searched linearly; when the number of edges increases, this gets very costly. After all, top-down filtering is generally considered beneficial (e.g. Slocum 1981:4).

The maximally directed strategy — Kilbury's algorithm with selectivity and top-down filtering — remained the most efficient one throughout all the experiments, both with respect to edges produced and time consumed (but more so with respect to the former). Top-down filtering did not degrade time performance quite as much in this case, presumably because of the great number of active edges cut off by the selectivity filter.

Finally, it should be mentioned that bottom-up parsing enjoys a special advantage not shown here, namely in being able to detect ungrammatical sentences much more effectively than top-down methods (cf. Kay 1982:342).

5 Conclusion

This paper has surveyed the fundamental rule-invocation strategies in context-free chart parsing. In order to arrive at some quantitative measure of their performance characteristics, the strategies have been implemented and tested empirically. The experiments clearly indicate that it is possible to significantly increase efficiency in chart parsing by fine-tuning the rule-invocation strategy. Fine-tuning however also requires that the characteristics of the grammars to be used are borne in mind. Nevertheless, the experiments indicate that in general directed methods are to be preferred to undirected methods; that top-down is the best undirected strategy; that Kilbury's original algorithm is not in itself a very good candidate, but that its directed versions — in particular the one with both selectivity and top-down filtering — are very promising.

Future work along these lines is planned to involve application of (some of) the strategies above within a unification-based parsing system.

Acknowledgements

I would like to thank Lars Ahrenberg, Nils Dahlbäck, Arne Jönsson, Magnus Merkel, Ivan Rankin, and an anonymous referee for the very helpful comments they have made on various drafts of this paper. In addition I am indebted to Masaru Tomita for providing me with his test grammars and sentences, and to Martin Kay for comments in connection with my presentation.

References

Aho, Alfred V. and Jeffrey D. Ullman (1972). *The Theory of Parsing, Translation, and Compiling. Volume I: Parsing.* Prentice-Hall, Englewood Cliffs, New Jersey.

Dyvik, Helge (1986). Aspects of Unification-Based Chart Parsing. Ms. Department of Linguistics and Phonetics, University of Bergen, Bergen, Norway.

Earley, Jay (1970). An Efficient Context-Free Parsing Algorithm. *Communications of the ACM* 13(2):94–102.

Griffiths, T. V. and Stanley R. Petrick (1965). On the Relative Efficiences of Context-Free Grammar Recognizers. *Communications of the ACM* 8(5):289–300.

Kaplan, Ronald M. (1973). A General Syntactic Processor. In: Randall Rustin, ed., *Natural Language Processing.* Algorithmics Press, New York, New York: 193–241.

Karttunen, Lauri (1986). D-PATR: A Development Environment for Unification-Based Grammars. *Proc. 11th COLING*, Bonn, Federal Republic of Germany: 74–80.

Kay, Martin (1973). The MIND System. In: Randall Rustin, ed., *Natural Language Processing.* Algorithmics Press, New York, New York: 155–188.

Kay, Martin (1982). Algorithm Schemata and Data Structures in Syntactic Processing. In: Sture Allén, ed., *Text Processing. Proceedings of Nobel Symposium 51.* Almqvist & Wiksell International, Stockholm, Sweden: 327–358. Also: CSL-80-12, Xerox PARC, Palo Alto, California.

Kilbury, James (1985). Chart Parsing and the Earley Algorithm. KIT-Report 24, Projektgruppe Künstliche Intelligenz und Textverstehen, Technische Universität Berlin, West Berlin. Also in: U. Klenk, ed. (1985), *Kontextfreie Syntaxen und verwandte Systeme. Vorträge eines Kolloquiums in Grand Ventron im Oktober, 1984.* Niemeyer, Tübingen, Federal Republic of Germany.

Merkel, Magnus (1986). A Swedish Grammar in D-PATR. Experiences of Working with D-PATR. Research report LiTH-IDA-R-86-31, Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Pereira, Fernando C. N. and David H. D. Warren (1980). Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence* 13(3):231–278.

Pratt, Vaughan R. (1975). LINGOL — A Progress Report. *Proc. 4th IJCAI*, Tbilisi, Georgia, USSR: 422–428.

Shieber, Stuart M., Hans Uszkoreit, Fernando C. N. Pereira, Jane J. Robinson, and Mabry Tyson (1983). The Formalism and Implementation of PATR-II. In: Barbara Grosz and Mark Stickel, eds., *Research on Interactive Acquisition and Use of Knowledge.* SRI Final Report 1894, SRI International, Menlo Park, California.

Slocum, Jonathan (1981). A Practical Comparison of Parsing Strategies. *Proc. 19th ACL*, Stanford, California: 1–6.

Thompson, Henry (1981). Chart Parsing and Rule Schemata in GPSG. Research Paper No. 165, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland. Also in: *Proc. 19th ACL*, Stanford, California: 167–172.

Thompson, Henry and Graeme Ritchie (1984). Implementing Natural Language Parsers. In: Tim O'Shea and Marc Eisenstadt, *Artificial Intelligence: Tools, Techniques, and Applications.* Harper & Row, New York, New York: 245–300.

Tomita, Masaru (1985). An Efficient Context-free Parsing Algorithm For Natural Languages. *Proc. 9th IJCAI*, Los Angeles, California: 756–764.

Tomita, Masaru (1986). *Efficient Parsing for Natural Language. A Fast Algorithm for Practical Systems.* Kluwer Academic Publishers, Norwell, Massachusetts.

Wang, Weiguo (1985). Computational Linguistics Technical Notes No. 2. Technical Report 85/013, Computer Science Department, Boston University, Boston, Massachusetts.