Predictive Head-Corner Chart Parsing

Klaas Sikkel\(^1\) and Rieks op den Akker\(^2\)

\(^1\) GMD – Forschungszentrum Informationstechnik, FIT.CSCW,
Schloß Birlinghoven, D-53757 Sankt Augustin, Germany
email: sikkel@gmd.de

\(^2\) Dept. of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, the Netherlands
email: infrieks@cs.utwente.nl

Abstract
Head-Corner (HC) parsing has come up in computational linguistics a few years ago, motivated by linguistic arguments. This idea is a heuristic, rather than a fail-safe principle, hence it is relevant indeed to consider the worst-case behaviour of the HC parser. We define a novel predictive head-corner chart parser of cubic time complexity.

We start with a left-corner (LC) chart parser, which is easier to understand. Subsequently, the LC chart parser is generalized to an HC chart parser. It is briefly sketched how the parser can be enhanced with feature structures.

1. Introduction

“Our Latin teachers were apparently right”, Martin Kay (1989) remarks. “You should start [parsing] with the main verb. This will tell you what kinds of subjects and objects to look for and what cases they will be in. When you come to look for these, you should also start by trying to find the main word, because this will tell you most about what else to look for”.

Head-driven or head-corner parsing has been addressed in several publications (Proudian and Pollard, 1985; Kay 1989; Satta and Stock 1989; van Noord 1991; Bouma and van Noord 1993). As the head-driven approach is a heuristic, rather than a fail-safe principle, it is important to pay attention to the worst-case behaviour. This is best taken care of in a tabular approach like the bottom-up head-driven parser by Satta and Stock. We enhance the tabular head-driven parser with top-down prediction.

The algorithmic details of the head-corner parser are not easy. Therefore we will make some effort to convey the intuition behind the parser. To that end, we
first define a left-corner chart parser in Section 3 and afterwards generalize this to a head-corner parser in 4. A complexity analysis is given in 5. We sketch extension with feature structures in 6 and briefly discuss related approaches in 7.

2. Chart parsing

Chart parsing, first introduced by Kay (1980), is a well-known parsing technique in computational linguistics. We will present a conventional Earley chart parser in a slightly unconventional way. Thus the reader who is familiar with the Earley parser will get a feeling for the notation used in this chapter.

We use the following notational conventions. Nonterminals are denoted by $A, B, \ldots \in N$; terminals by $a, b, \ldots \in \Sigma$. We write $V$ for $N \cup \Sigma$ with $X, Y, \ldots$ as typical elements. Strings in $V^*$ are denoted by $a, \beta, \ldots$. A context-free grammar $G$ is a 4-tuple $(N, \Sigma, P, S)$, with $P$ a set of productions and $S$ the start symbol. The sentence to be parsed is denoted $a_1 \ldots a_n$. We make extensive use of place markers $i, j, k, \ldots$, indicating positions in the sentence. The symbol $a_i$ is located between positions $i-1$ and $i$.

A chart parser is characterized by the following entities: a *domain of items*, that can be added to the chart by the parser using an agenda; some *operators* that specify how combinations of items on the chart can lead to recognition of other items; an *initial chart* and an *initial agenda*. Items represent (the existence of) partial parse trees for the given sentence. At each step some current item is selected from the agenda, and moved to the chart. If the chart contains items that, in combination with the current item, allow recognition of other items not yet present on the chart or on the agenda, these are added to the agenda. This continues until the agenda is empty.

A context-free chart parser does not really construct parse trees. But a representation of all parse trees can easily be obtained when items in the chart are annotated with pointers to the items that caused their recognition. Various forms of sophistication can be added by structuring the chart or by providing a strategy to select the next item from the agenda.

The Earley chart parser uses two types of items:

- $[A \Rightarrow \alpha \beta, i, j]$: *Earley items* (for $A \Rightarrow \alpha \beta \in P$ and $0 \leq i \leq j \leq n$),
- $[a, j-1, j]$ : *terminal items*, representing $a_j$ ($1 \leq j \leq n$).

An Earley item $[A \Rightarrow \alpha \beta, i, j]$ is to be recognized by the chart parser iff

- $\alpha \Rightarrow a_i+1 \ldots a_j$, and
- $S \Rightarrow a_1 \ldots a_n A \gamma$ for some $\gamma \in V^*$.

The initial chart contains the terminal items representing the string.\(^1\)

\(^1\)In general we could allow that items $[X, j, k]$, with $k > j$ and $X$ either a terminal or a nonterminal, are initially on the chart. These “initial” items could be constructed then by preprocessing the input sentence by a lexical analyser, or some other substring parser.
When the $j$-th word belongs to different categories, say $a$ and $b$, then both items $[a, j-1, j]$ and $[b, j-1, j]$ are present in the initial chart. The initial agenda contains items $[S \to \cdot \gamma, 0, 0]$ for all productions $S \to \gamma \in P$. The following operators are defined for the Earley chart parser:

**predict:** for $B \to \gamma \in P$:

$[A \to a \cdot B \beta, i, j] \vdash [B \to \cdot \gamma, j, j]$;

**scan:**

$[A \to a \cdot b \beta, i, j], [b, j, j+1] \vdash [A \to a b \beta, i, j+1]$;

**complete:**

$[A \to a \cdot B \beta, i, j], [B \to \gamma \cdot, j, k] \vdash [A \to a B \beta, i, k]$.

The turnstile notation is a convenient shorthand, meaning that the left-hand side items licence the recognition of the right-hand side item. As a running example we will use the sentence *the cat caught a mouse*, represented by the lexical categories

$*\text{det} *\text{n} *\text{v} *\text{det} *\text{n}$.

The example grammar is

$S \to NP \ VP$,
$NP \to *\text{det} *\text{n}$,
$VP \to *\text{v} NP$.

Due to lack of ambiguity, the example will nicely illustrate the difference between the chart parsers that are presented in this chapter. Initially, the chart contains $[[0, *\text{det}, 1], [1, *\text{n}, 2], [2, *\text{v}, 3], [3, *\text{det}, 4], [4, *\text{n}, 5]]$ and the agenda is $[[S \to \cdot NP \ VP, 0, 0]]$. In Table 1 the completed chart is shown (excluding terminal items), annotated with how the items were recognized. The items can be seen as the trace of a left-to-right walk through the parse tree. This walk is shown in Figure 1, annotated with the item numbers. (In the general case things are rather more complicated. There could be different parse trees and also partial left-to-right walks of valid prefixes that can’t be extended to a parse. All these traces are interlaced and may partly coincide.)

3. **Left-corner chart parsing**

We will now specify a Left-Corner (LC) parser as a chart parser; it is to be generalized to a head-corner parser in the next section. The deterministic LC algorithm originally stems from Rosenkranz and Lewis (1970). We describe a generalized LC parser\(^2\) for context-free grammars that is similar to, but subtly different from

---

\(^2\)The term “Generalized LC” has been introduced by Demers (1977) for a rather different concept. He generalized the notion of Left Corner, deriving a framework that describes a class of parsers and associated grammars ranging from LL(k) via LC(k) to LR(k). We generalize the LC(0) parser by dropping the restriction that the grammar be LC(0). The nondeterminism is handled by a dynamic programming technique, as in Generalized LR parsing (Tomita, 1985). Note that the semantic ambiguity of the noun
the Earley parser. *Scan* and *complete* operations remain unchanged, predicting is handled differently. We start with a *predict* item \([0,S]\) meaning that we are to look for a constituent \(S\) starting at position 0. We proceed bottom-up, starting from \([*\text{det}, 0, 1]\). We can “climb up” from \(*\text{det}\) to an NP because this moves us nearer to the goal \(S\). Or, to put it more formally, because a leftmost derivation

\[
S \Rightarrow^* \text{NP}_\gamma \Rightarrow^* \text{det}*\text{n}_\gamma
\]

exists, we may extend \([*\text{det}, 0, 1]\) to \([\text{NP} \rightarrow *\text{det}*\text{n}, 0, 1]\).

A few steps later we have recognized \([S \rightarrow \text{NP}*\text{VP}, 0, 2]\). Here we set a subgoal, represented by the predict item \([\text{NP} \rightarrow *\text{det}*\text{n}, 0, 1]\). In Figure 2 it is shown how the LC chart parser steps through a parse tree:

phrase “Generalized LC parsing” duly reflects the syntactic ambiguity: we are concerned with \([\text{Generalized} \ [\text{Left-Corner Parsing}]\), whereas Demers discussed [[Generalized Left-Corner] parsing].
Steps up correspond to a scan or complete as in the Earley case.
Steps down to the leftmost child are skipped because these are implicitly encoded in the transitive left-corner relation that is incorporated in the parser.
Steps down to non-leftmost nonterminal children correspond to setting a new sub-goal.

Figure 2: The left-corner tree walk

In Table 2 the final chart is shown of the LC chart parser that will be formally defined next. Each item on the final chart corresponds to an arrow in the tree walk. The intuition should be clear now, and we present the formal definition

<table>
<thead>
<tr>
<th>#</th>
<th>item</th>
<th>recognized by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[0,S]</td>
<td>initial</td>
</tr>
<tr>
<td>1</td>
<td>[NP → *det • *n, 0, 1]</td>
<td>lc(i) (0)</td>
</tr>
<tr>
<td>2</td>
<td>[NP → *det *n •, 0, 2]</td>
<td>scan (1)</td>
</tr>
<tr>
<td>3</td>
<td>[S → NP • VP, 0, 2]</td>
<td>lc(ii) (0,2)</td>
</tr>
<tr>
<td>4</td>
<td>[2, VP]</td>
<td>predict (3)</td>
</tr>
<tr>
<td>5</td>
<td>[VP → *v • NP, 2, 3]</td>
<td>lc(i) (4)</td>
</tr>
<tr>
<td>6</td>
<td>[3, NP]</td>
<td>predict (5)</td>
</tr>
<tr>
<td>7</td>
<td>[NP → *det • *n, 3, 4]</td>
<td>lc(i) (6)</td>
</tr>
<tr>
<td>8</td>
<td>[NP → *det *n •, 3, 5]</td>
<td>scan (7)</td>
</tr>
<tr>
<td>9</td>
<td>[VP → *v NP •, 2, 5]</td>
<td>complete (5,8)</td>
</tr>
<tr>
<td>10</td>
<td>[S → NP VP •, 0, 5]</td>
<td>complete (3,9)</td>
</tr>
</tbody>
</table>

Table 2: The final LC chart

rather terse. The left corner of a production is the leftmost symbol in the right-hand side of that production (and ε for an empty production). We write $A \succ_U U$ if $A$ has left corner $U \in (V \cup \{ε\})$ We write $\succ^*_U$ for the transitive closure of $\succ_U$.

Hence we have, for example, $S \succ^*_i *det$.

The LC chart parser uses the following kinds of items:
The initial chart contains the terminal items representing the sentence, the agenda is initialized to \([0, S]\). The operators of the LC chart parser are defined as follows.

We distinguish separate left-corner \((lc)\) operators for left corners \(a, C, \) and \(\varepsilon\).

- **lc(i):** for \(A \to B, B \to a \beta \in P:\)
  
  
  \([i, A], [a, i, i+1] \vdash [B \to a \beta, i, i+1],\)

- **lc(ii):** for \(A \to B, B \to C \beta \in P:\)
  
  
  \([i, A], [C \to \gamma \beta, i, j] \vdash [B \to C \beta, i, j],\)

- **lc(iii):** for \(A \to B, B \to \varepsilon \in P:\)
  
  \([i, A] \vdash [B \to \varepsilon, i, i],\)

**predict:**

\([B \to a \varepsilon \beta, i, j] \vdash [j, C];\)

**scan:**

\([B \to a a \varepsilon \beta, i, j], [a, j, j+1] \vdash [B \to a a \varepsilon \beta, i, j+1];\)

**complete:**

\([B \to a \varepsilon C \beta, i, j], [C \to \gamma \beta, j, k] \vdash [B \to a C \beta, i, k].\)

Thus we have characterized the LC chart parser by defining the initial chart and agenda and the operators. The reader may verify that these operators produce the chart shown in Table 2 for our example sentence. For a proof of the correctness of the algorithm we refer to Sikkel and op den Akker (1992) or Sikkel (1993).

### 4. Head-Corner chart parsing

We introduce the head-corner chart parser by analogy to the left-corner parser. While the LC parser makes a left-to-right walk through a parse tree, the HC parser makes a head-first walk through a parse tree\(^3\), as shown in Figure 3. With this very simple idea in mind we can fix the formal details.

A context-free head grammar is a 5-tuple \((N, \Sigma, P, S, r)\), with \(r\) a function that assigns a natural number to each production in \(P\). Let \(|p|\) denote the length of the right-hand side of \(p\). Then \(r\) is constrained to \(r(p) = 0\) for \(|p| = 0\) and \(1 \leq r(p) \leq |p|\) for \(|p| > 0\). The head of a production \(p\) is the \(r(p)\)-th symbol of the right-hand side; an \(\varepsilon\)-production has head \(\varepsilon\). It is up to the grammar writer to choose heads for the productions.

In a practical notation, we give a head grammar as a set of productions with the heads underlined. We have extended the example grammar to a head grammar with the following head assignments:

\(^3\)In the general case, to be precise, the HC parser makes interlacing and partially coinciding walks through all full parse trees, as well as “valid infixes” starting from a feasible lexical head of the sentence.
The relation $>_h$ is defined by $A >_h U$ if there is a production $p = A \rightarrow a \in P$ with $U$ the head of $p$; the transitive and reflexive closure of $>_h$ is denoted $>_h^*$. Hence we have, for example, $S >_h^* \ast v$. For the head-corner parser we distinguish the following kinds of items:

- $[l, r, A]$: predict items
- $[B \rightarrow \alpha \bullet \beta \bullet \gamma, i, j]$: double dotted (DD) items,
- $[a, j-1, j]$: terminal items.

A predict item $[l, r, A]$ will be recognized if a constituent $A$ is being looked for that must be located somewhere between $l$ and $r$. Such a constituent should either stretch from $l$ to some $j$ (if we are working to the right from the head of some production) or from $r$ down to some $j$ (if we are working to the left from the head of some production), with $l \leq j \leq r$. A double dotted item $[B \rightarrow \alpha \bullet \beta \bullet \gamma, i, j]$ is recognized if there is a goal $[l, r, A]$ such that $A >_h^* B$, $l \leq i, j \leq r$ and $\beta \Rightarrow^* a_{i+1} \ldots a_j$ has been established.

The initial chart contains the terminal items that represent the string as usual; the agenda is initialized with a single predict item $[0, n, S]$. If the string is correct, we will be able to derive an item $[S \rightarrow \ast, 0, n]$. The operators, are defined as follows. We distinguish three different head-corner ($hc$) operators for terminal heads $b$, for nonterminal heads $C$ and for heads being $\varepsilon$.

$hc(i)$: for $A >_h^* B$, $B \rightarrow \alpha \bullet \gamma \in P$, $l < j \leq r$:

$[l, r, A], [l, j-1, j] \vdash [B \rightarrow \alpha \bullet \gamma, j-1, j]$.

$hc(ii)$: for $A >_h^* B$, $B \rightarrow \alpha \bullet \gamma \in P$, $l \leq i \leq j \leq r$:

$[l, r, A], [C \rightarrow \bullet \gamma, i, j] \vdash [B \rightarrow \alpha \bullet \gamma, i, j]$.
complexity analysis and further optimizations

\( k c(iii) \): for \( A \geq B, B \rightarrow \varepsilon \in \mathcal{P}, l \leq j \leq r: \)

\[ \left[ l, r, A \right] \vdash [B \rightarrow \bullet, j, j]; \]

\( \text{predict: for } A \geq B, l \leq i \leq j \leq r: \)

\[ \left[ l, r, A \right], [B \rightarrow \alpha C \bullet \beta \gamma, i, j] \vdash [l, i, C]; \]

\[ \left[ l, r, A \right], [B \rightarrow \alpha \beta C \gamma, i, j] \vdash [j, r, C]; \]

\( \text{scan: for } A \geq B, l < j \leq k \leq r: \)

\[ \left[ l, r, A \right], [a, j-1, j], [B \rightarrow \alpha \alpha \beta \gamma, i, j] \vdash [B \rightarrow \alpha \alpha \beta \alpha \gamma, j-1, k], \]

\[ \left[ l, r, A \right], [B \rightarrow \alpha \beta \alpha \gamma, i, j], [l, j-1, j] \vdash [B \rightarrow \alpha \beta \alpha \gamma, i, j]; \]

\( \text{complete: for } A \geq B, l \leq i \leq j \leq k \leq r: \)

\[ \left[ l, r, A \right], [C \rightarrow \bullet \bullet \bullet \gamma, i, j], [B \rightarrow \alpha C \beta \gamma, j, k] \vdash [B \rightarrow \alpha C \gamma, i, k], \]

\[ \left[ l, r, A \right], [B \rightarrow \alpha \beta \beta \gamma, i, j], [C \rightarrow \bullet \bullet \bullet \gamma, j, k] \vdash [B \rightarrow \alpha \beta C \gamma, i, k]. \]

Head-corner analysis of the example sentence illustrates how the chart parser may jump up and down the sentence. The completed HC chart is shown in Table 3. Each item corresponds to a step in the head-corner tree walk for the parse of our example sentence, cf. Figure 3.

<table>
<thead>
<tr>
<th>#</th>
<th>item</th>
<th>recognized by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>([0, 5, S])</td>
<td>initial</td>
</tr>
<tr>
<td>1</td>
<td>([VP \rightarrow \bullet \bullet \bullet NP, 2, 3])</td>
<td>(hc(i)(0))</td>
</tr>
<tr>
<td>2</td>
<td>([3, 5, NP])</td>
<td>(predict(0, 1))</td>
</tr>
<tr>
<td>3</td>
<td>([NP \rightarrow \bullet \bullet \bullet n, 4, 5])</td>
<td>(hc(i)(2))</td>
</tr>
<tr>
<td>4</td>
<td>([NP \rightarrow \bullet \bullet \bullet n, 3, 5])</td>
<td>(scan(2, 3))</td>
</tr>
<tr>
<td>5</td>
<td>([VP \rightarrow \bullet \bullet \bullet NP, 2, 5])</td>
<td>(complete(0, 1, 4))</td>
</tr>
<tr>
<td>6</td>
<td>([S \rightarrow NP \bullet VP, 2, 5])</td>
<td>(hc(ii)(0, 5))</td>
</tr>
<tr>
<td>7</td>
<td>([0, 2, NP])</td>
<td>(predict(0, 6))</td>
</tr>
<tr>
<td>8</td>
<td>([NP \rightarrow \bullet \bullet \bullet n, 1, 2])</td>
<td>(hc(i)(7))</td>
</tr>
<tr>
<td>9</td>
<td>([NP \rightarrow \bullet \bullet \bullet n, 0, 2])</td>
<td>(scan(7, 8))</td>
</tr>
<tr>
<td>10</td>
<td>([S \rightarrow \bullet \bullet \bullet VP, 0, 5])</td>
<td>(complete(0, 6, 9))</td>
</tr>
</tbody>
</table>

5. Complexity analysis and further optimizations

The number of items that can be recognized is \(O(n^2)\), but the work involved for an arbitrary current item is more than linear. The most problematic operation is \(complete\) (with \(scan\) as a special sub-case) with 5 place markers involved. \(Complete\) can be reduced to 3 place markers with some special extra bookkeeping. As a consequence, the number of place markers involved in \(scan\) and \(predict\) will drop from 4 to 3. We keep a \(goal\ table\), in the form of a CYK table, for storage of the predicted goals. Whenever an item \([l, r, C]\) is predicted, we write a \(C\) in goal table entry \((i, r)\). Furthermore, we write a \(C\) in every entry \((i, j)\) with \(l \leq i \leq j \leq r\) in which no \(C\) is present. A typical case is presented in Figures 4 and 5. A goal
[1,7,C] is to be added, the entry (1,7) is marked * in Figure 4. One adds C symbols column by column, stopping each time when an C is found. In Figure 5 a * indicates the entries where an C is written and + indicates the entries that were inspected but already contained an C. During the course of the algorithm only \( O(n^2) \) C symbols are written, per symbol only \( O(n) \) entries are inspected that already did contain C. Using the goal table, the place markers l, r in the complete can be replaced by i, k.

![Figure 4: Adding [1,7,C] to the goal table...](image)

![Figure 5: ...successful (*) and unsuccessful (+) accesses](image)

Using the goal table as explained above, one can straightforwardly verify that the overall complexity in the length of the sentence is \( O(n^2) \) space and \( O(n^3) \) time as usual, assuming that the chart is structured likewise as a triangular matrix. When the size of the grammar is taken into account the analysis gets somewhat more complicated. For the sake of brevity we only state the results. A detailed analysis is given by Sikkel (1993). For an optimal worst-case complexity we have to make a small change to the implementation and represent fully completed DD
items \([A \rightarrow \bullet \beta, i, j]\) by CYK items \([A, i, j]\). An Earley chart parser — using the same optimization — has a time complexity \(O(G n^3)\), with \(|G|\) the number of productions multiplied by the average length of a production. Without prediction the HC parser has a time complexity of \(O(G|n|^3)\), with \(r\) the size of the longest production. This extra factor \(r\) is because we use double dotted, rather than single dotted items. Prediction usually speeds up a parser but may slow it down in pathological cases. In the worst case it adds a factor \(|N|\), the number of nonterminal symbols, yielding a time complexity of \(O(|N||G|n^3)\). The space complexity is \(O(|G||n^3|)\) for the chart and agenda, and \(O(|N|^3 \times |N||\Sigma|)\) for storage of the \(\Rightarrow^*_h\) relation in tabular form.

The obtained worst-case complexity is optimal, in the sense that all complexity factors are properly accounted for. The factor \(|N|\) in addition to an optimal Earley parser is a necessary consequence of (the worst-case behaviour of) prediction. The factor \(r\) for right-hand side length can be eliminated by a simple optimization:

- When an item \([A \rightarrow \alpha \bullet B \gamma, i, j]\) has been recognized by the head-corner rule, with \(\alpha \neq \varepsilon \neq \gamma\), it should be expanded either to \([A \rightarrow \alpha B \gamma, h, j]\) or to \([A \rightarrow \alpha B \gamma, i, k]\) but not both; from either one a completed item \([A \rightarrow \alpha B \gamma, h, k]\) can be obtained. This idea is taken from Satta and Stock (1989).

By imposing the restriction that right expansion is to be allowed only for items that cannot further expand to the left, the time complexity is reduced by a factor \(r\) to \(O(|N||G|n^3)\).

A large percentage of computing time can be saved by adding some more sophistication to the algorithm. Another optimization that can be added straightforwardly is the following:

- A predicted item should fit to the left, fit to the right, or both. This can be expressed by using predict items of the form \([= l, = r, A]\), \([\geq l, = r, A]\) and \([= l, \leq r, A]\) with the obvious interpretation. For example, if \(A \Rightarrow^*_h X\) and \(X\) occurs only at the left (i.e., if \(A \Rightarrow X\beta\)) then \([X, i, j]\) fits to \([= l, \leq r, A]\) only if \(i = l\). The head-corner operator can be adapted accordingly.

If the grammar is limited to Chomsky Normal Form, the first saving doesn’t apply but the second is more effective. Further optimizations are possible, but beyond the scope of this chapter.

6. Extension with feature structures

Feature information according to any unification based grammar can be added to recognized items. For bottom-up composition of constituents this is straightforward. Using feature information for top-down prediction is more subtle, as we can only use so-called transitive features. A feature is transitive if every constituent shares the feature with its head. Hence, a constituent transitivity shares such a feature with its lexical head. This is important, because sub-goals are parsed from the lexical head upwards. If, for example, a VP has been found with agreement 3sg (third person singular), then the NP that is set as a goal must also have agree-
ment 3sg. Because *agreement* is a transitive feature, only a noun with *agreement* 3sg can be the lexical head.

Practically this works as follows. Double dotted items $[B \rightarrow \alpha \beta \gamma, i, j]$ are replaced by items $[l, r, A; B \rightarrow \alpha \beta \gamma, i, j]$, with $A$ the predicted goal symbol. The transitive features of $A$ are shared with $B$. If we have an item $[l, r, A; A \rightarrow \alpha \beta \gamma, l, j]$ we can unify the predicted and parsed $A$ if the nontransitive features match as well.

7. Related approaches

The left-corner chart parser of Section 3, although rather different in style of presentation, is closely related to the predictive chart parsing framework introduced by Kay (1980). First ideas of generalized LC parsing, although not under that name, can be traced back to Pratt (1975). A left-corner style parser in Prolog was presented by Matsumoto et al. (1983). This BUP parser is limited to acyclic, $\varepsilon$-free grammars. Nederhof (1992) defines a generalized LC parser by analogy to the generalized LR parser of Tomita (1985). Nederhof's parser is more efficient than our LC chart parser, but it doesn't generalize to HC.

The head-driven parser of Satta and Stock (1989) is similar to ours, but does not make use of prediction. The use of a head-driven approach to enhance the efficiency of prediction was first suggested by Kay (1989).

The context-free head grammars in Section 4 should not be confused with Head Grammars as introduced by Pollard (1984), that handle discontinuous constituents by means of "head wrapping". (Van Noord, 1991) describes a Prolog implementation of a head-corner parser for languages with discontinuous constituents.

Bouma and van Noord (1988) have experimented with various parsing strategies for unification grammars and conclude that for important classes of grammars it is fruitful to apply parsing strategies that are sensitive to the linguistic notion of a head.

A Head-Corner parser for typed feature structures has been implemented by Moll (1995). It is used as a parser in an experimental natural language dialogue System. See op den akker et al. (1995) for more details.

8. Conclusions

We have given a formal treatment of a predictive head-corner parser. The item-based description of (predictive) chart parsers is a useful formalism for such a formal treatment. This is exemplified by the fact that we cover grammars with $\varepsilon$-productions with hardly any additional effort, while these are usually left out for the sake of simplicity. Enhancing a head-corner chart parser with prediction is new.

It cannot be stated in general that the head-corner approach is more efficient than the (generalized) left-corner approach or other parsers. It is indeed a heuristic, that can be expected to be effective when most of the feature information of a constituent is located in the head. Hence, *because* it is a method based on a heuristic, rather than a fail-safe principle, it is important to consider what hap-
pens if the heuristic doesn’t pay off. Therefore we have made some effort to make
sure that the worst-case behaviour conforms to the usual complexity bounds for
context-free parsing algorithms: $O(n^3)$ time and $O(n^2)$ space.

Acknowledgements

We are grateful to Anton Nijholt, Mark-Jan Nederhof, Gertjan van Noord and
Giorgio Satta for constructive comments on earlier drafts of this chapter. We are
grateful to Margriet Verlinden, Gert Veldhuijzen van Zanten, and Marc Moll for
their work on the implementation of the Head-Corner Parser.

References

systems using typed feature structures. Memoranda Informatica 95-25, Department
of Computer Science, University of Twente, Enschede, the Netherlands.

Bonma, G. and van Noord, G. (1993) Head-driven Parsing for Lexicalist Grammars:

Demers, A.J. (1977) Generalized Left Corner Parsing. 4th ACM Symp. on Principles of
Prog. Lang., 170–182.

CIS-86-12, Xerox PARC, Palo Alto, Ca.

Reprinted in: Grosz, B.J. et al. (Eds.), Readings in Natural Language Processing,

52–62.


University of Twente, Dept. of Computer Science, Enschede, the Netherlands.

the ACL, 305–314.

Ann. Meeting of the ACL, 114–121.


Pratt, V.R. (1975) LINGOL — A Progress Report. Proc. 4th Int. Joint Conf. on Artificial
Intelligence (IJCAI’75), 422–428.


Ann. Symp. on Switching and Automata Theory, 139–152.


of Twente, Enschede, the Netherlands.

Memoranda Informatica 92-55, University of Twente, Enschede, the Netherlands.