Predictive Head-Corner Chart Parsing

Klaas Sikkel — Rieks op den Akker

Dept. of Computer Science, University of Twente
P.O. Box 217, 7500 AE ENSCHEDE
email: {sikkel,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 remarks in (Kay, 1989). "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 papers. (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 in (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 article.

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 $\alpha, \beta, \ldots$. A context-free grammar G is a 4-tuple (N, Σ, 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 a *domain of items*, that can be added to the chart by the parser and some *operators* that specify how combinations of items on the chart can lead to recog-

nition of other items. Furthermore, there is an *initial chart* and *initial agenda*.

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:

$[A \to \alpha\bullet\beta, i, j]$: *Earley items* (for $A \to \alpha\beta \in P$ 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 $[A \to \alpha\bullet\beta, i, j]$ is to be recognized by the chart parser iff

$\alpha \Rightarrow^* a_{i+1} \ldots a_j$, and
$S \Rightarrow^* a_1 \ldots a_i A\gamma$ for some $\gamma \in V^*$.

The initial chart contains the terminal items representing the string. 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 $[S \to \bullet\gamma, 0, 0]$ for all productions $S \to \gamma \in P$. The following operators are defined for the Earley chart parser:

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

$$[A \to \alpha\bullet B\beta, i, j] \vdash [B \to \bullet\gamma, j, j];$$

scan:

$$[A \to \alpha\bullet b\beta, i, j], [b, j, j+1] \vdash [A \to \alpha b\bullet\beta, i, j+1];$$

complete:

$$[A \to \alpha\bullet B\beta, i, j], [B \to \gamma\bullet, j, k] \vdash [A \to \alpha B\bullet\beta, i, k].$$

The turnstyle ($\vdash$) 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

$$*det\ *n\ *v\ *det\ *n.$$

The example grammar is

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

Due to lack of ambiguity, the example will nicely illustrate the difference between the chart parsers that are presented in this paper. Initially, the chart contains $\{[0, *det, 1], [1, *n, 2], [2, *v, 3], [3, *det, 4], [4, *n, 5]\}$ and the agenda is $\{[S \to \bullet NP\ VP, 0, 0]\}$. In Figure 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 2 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 than can't be extended to a parse. All trese 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, that is to be generalized to a head-corner parser in the next section. The deterministic LC algorithm originally stems from (Rosenkrantz and Lewis, 1970). We describe a generalized LC parser[1] for context-free grammars

[1]The term "Generalized LC" has been introduced in (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

#	item	recognized by
0	$[S \rightarrow \bullet NP\ VP, 0, 0]$	initial
1	$[NP \rightarrow \bullet *det\ *n, 0, 0]$	*predict*(0)
2	$[NP \rightarrow *det\bullet\ *n, 0, 1]$	*scan*(1)
3	$[NP \rightarrow *det\ *n\bullet, 0, 2]$	*scan*(2)
4	$[S \rightarrow NP\bullet VP, 0, 2]$	*compl*(0,3)
5	$[VP \rightarrow \bullet *v\ NP, 2, 2]$	*predict*(4)
6	$[VP \rightarrow *v\bullet NP, 2, 3]$	*scan*(5)
7	$[NP \rightarrow \bullet *det\ *n, 3, 3]$	*predict*(6)
8	$[NP \rightarrow *det\bullet\ *n, 3, 4]$	*scan*(7)
9	$[NP \rightarrow *det\ *n\bullet, 3, 5]$	*scan*(8)
10	$[VP \rightarrow *v\ NP\bullet, 2, 5]$	*complete*(6,9)
11	$[S \rightarrow NP\ VP\bullet, 0, 5]$	*complete*(4,10)

Figure 1: The final Earley chart

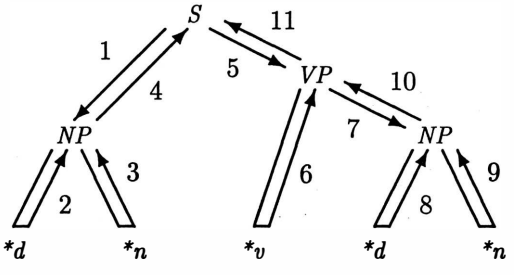

Figure 2: The Earley tree walk

that is similar to, but subtly different from 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 $[*det, 0, 1]$. We can "climb up" from $*det$ to an NP because this moves us nearer to the goal S. Or, to put it more formally, because a leftmost derivation

$$S \Rightarrow^* NP\gamma \Rightarrow *det\ *n\gamma$$

exists, we may extend $[*det, 0, 1]$ to $[NP \rightarrow *det\bullet *n, 0, 1]$.

A few steps later we have recognized $[S \rightarrow NP\bullet VP, 0, 2]$. Here we set a *sub-goal*, represented by the predict item $[2, VP]$. In Figure 3 it is shown how the LC chart parser steps through a parse tree:

- 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 encorporated in the parser.
- Steps down to *non-leftmost nonterminal* children correspond to setting a new subgoal.

that the semantic ambiguity of the noun phrase "Generalized LC parsing" duly reflects the syntactic ambiguity: we are concerned with [Generalized [Left-Corner Parsing]], whereas Demers discussed [[Generalized Left-Corner] parsing].

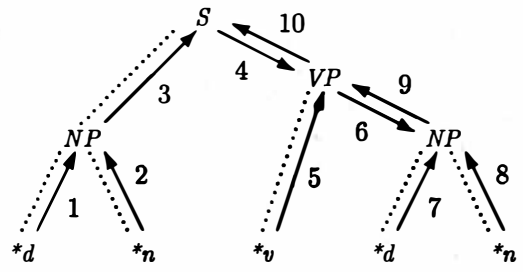

Figure 3: The left-corner tree walk

In Figure 4 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.

#	item	recognized by
0	$[0, S]$	initial
1	$[NP \rightarrow *det\bullet *n, 0, 1]$	$lc(i)$ (0)
2	$[NP \rightarrow *det\ *n\bullet, 0, 2]$	*scan* (1)
3	$[S \rightarrow NP\bullet\ VP, 0, 2]$	$lc(ii)$ (0,2)
4	$[2, VP]$	*predict* (3)
5	$[VP \rightarrow *v\bullet NP, 2, 3]$	$lc(i)$ (4)
6	$[3, NP]$	*predict* (5)
7	$[NP \rightarrow *det\bullet *n, 3, 4]$	$lc(i)$ (6)
8	$[NP \rightarrow *det\ *n\bullet, 3, 5]$	*scan* (7)
9	$[VP \rightarrow *v\ NP\bullet, 2, 5]$	*complete* (5,9)
10	$[S \rightarrow NP\ VP\bullet, 0, 5]$	*complete* (3,9)

Figure 4: The final LC chart

The intuition should be clear now, and we present the formal definition 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 $A >_\ell U$ if A has left corner $U \in (V \cup \{\varepsilon\})$ We write $>_\ell^*$ for the transitive closure of $>_\ell$. Hence, in the running example, we have $S >_\ell^* *det$.
The LC chart parser uses the following kinds of items:

$[i, A]$: predict items,
$[B \rightarrow \alpha\bullet\beta, i, j]$: Earley items (but only those with $\alpha \neq \varepsilon$ or $\alpha = \beta = \varepsilon$),
$[a, j-1, j]$: terminal items as usual.

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 ε.

$lc(i)$: for $A >_\ell^* B$, $B \rightarrow a\beta \in P$:

$$[i, A],\ [a, i, i+1] \vdash [B \rightarrow a\bullet\beta, i, i+1],$$

$lc(ii)$: for $A >_\ell^* B$, $B \rightarrow C\beta \in P$:

$$[i, A], [C \rightarrow \gamma\bullet, i, j] \vdash [B \rightarrow C\bullet\beta, i, j],$$

$lc(iii)$: for $A >_\ell^* B$, $B \rightarrow \varepsilon \in P$:

$$[i, A] \vdash [B \rightarrow \bullet, i, i];$$

predict:

$$[B \to \alpha\bullet C\beta, i, j] \vdash [j, C];$$

scan:

$$[B \to \alpha\bullet a\beta, i, j],\ [a, j, j+1] \vdash [B \to \alpha a\bullet\beta, i, j+1];$$

complete:

$$[B \to \alpha\bullet C\beta, i, j],\ [C \to \gamma\bullet, j, k] \vdash [B \to \alpha C\bullet\beta, i, k].$$

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 Figure 4 for our example sentence. For a proof of the correctness of the algorithm we refer to a more extensive technical report (Sikkel and op den Akker, 1992).

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[2], as shown in Figure 5. With this very simple idea in mind we can fix the formal details.

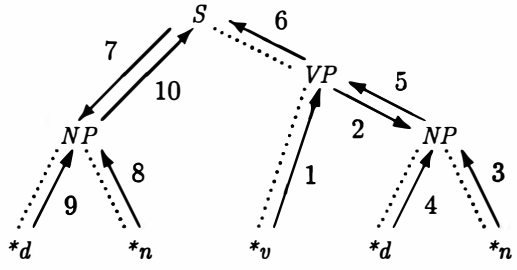

Figure 5: The head-corner tree walk

A *context-free head grammar* is a 5-tuple (N, Σ, 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 ε-production has head ε.

In a practical notation, we give a head grammar as a set of productions with the heads underlined. The head grammar H for our running example is given by

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

The relation $>_h$ is defined by $A >_h U$ if there is a production $p = A \to \alpha \in P$ with U the head of p; the transitive and reflexive closure of $>_h$ is denoted $>_h^*$. In the running example it holds that $S >_h^* *v$. For the head-corner parser we distinguish the following kinds of items:

$[l, r, A]$: *predict items*
$[B \to \alpha\bullet\beta\bullet\gamma, i, j]$: *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 downto some j (if we are

[2]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.

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 with $A >_h^* B$, $l \leq i$, $j \leq r$ and $\beta \Rightarrow^* a_{i+1} \ldots a_j$ has been established. In (Satta and Stock, 1992) it is remarked that, similar to the "valid prefix property" of the Earley and LC parser, our predictive head-corner parser satisfies a "valid infix property". As it is hard enough to get an understanding of what is going on, we will not concentrate on the mathematical details.

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 $[S \rightarrow \bullet\gamma\bullet, 0, n]$. The operators, are defined as follows. We distinguish three different head-corner (hc) operators for heads b, C and ε.

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

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

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

$$[l, r, A],\ [C \rightarrow \bullet\delta\bullet, i, j] \vdash [B \rightarrow \alpha\bullet C\bullet\gamma, i, j],$$

$hc(iii)$: for $A >_h^* B$, $B \rightarrow \varepsilon \in P$, $l \leq j \leq r$:

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

predict: for $A >_h^* B$, $l \leq i \leq j \leq r$:

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

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

scan: for $A >_h^* B$, $l < j \leq k \leq r$:

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

$$[l, r, A],\ [B \rightarrow \alpha\bullet\beta\bullet a\gamma, i, j-1],\ [a, j-1, j] \vdash [B \rightarrow \alpha\bullet\beta a\bullet\gamma, i, j];$$

complete: for $A >_h^* B$, $l \leq i \leq j \leq k \leq r$:

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

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

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 Figure 6. Each item corresponds to a step in the head-corner tree walk for the parse of our example sentence, cf. Figure 5.

#	item	recognized by
0	$[0, 5, S]$	*initial*
1	$[VP \rightarrow \bullet *v\bullet NP, 2, 3]$	$hc(i)$ (0)
2	$[3, 5, NP]$	*predict*(1)
3	$[NP \rightarrow *det\bullet *n\bullet, 4, 5]$	$hc(i)$ (2)
4	$[NP \rightarrow \bullet *det\ *n\bullet, 3, 5]$	*scan* (3)
5	$[VP \rightarrow \bullet *v\ NP\bullet, 2, 5]$	*complete* (1,4)
6	$[S \rightarrow NP\bullet VP\bullet, 2, 5]$	$hc(ii)$ (0,5)
7	$[0, 2, NP]$	*predict*(6)
8	$[NP \rightarrow *det\bullet *n\bullet, 1, 2]$	$hc(i)$ (7)
9	$[NP \rightarrow \bullet *det\ *n\bullet, 0, 2]$	*scan* (8)
10	$[S \rightarrow \bullet NP\ VP\bullet, 0, 5]$	*complete* (6,9)

Figure 6: The HC chart

5 Complexity analysis and further optimizations

The number of items that can be recognized now 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 $[l, r, A]$ is predicted, we write an A in goal table entry (l, r). Furthermore, we write an A in every entry (i, j) with $l \leq i \leq j \leq r$ in which no A is present. A typical case is presented in Figure 7. A goal $[0, 5, A]$ is to be added, the entry (0,5) is indicated * in Figure 7(a). One adds A symbols column by column, stopping each time when an A is found. In Figure 7(b) a * indicates the entries where an A is written and + indicates the entries that were inspected but already contained an A. During the course of the algorithm only $O(n^2)$ A symbols are written, per A only $O(n)$ entries are inspected that already did contain A.

```
      0 1 2 3 4 5 6 (j)
    0 A A A . . * .
    1   A A . . . .
    2     A . . . .
    3       . . . .
    4         A A A
    5           A A
(i) 6             A
```

(a)

```
      0 1 2 3 4 5 6 (j)
    0 A A + * * * .
    1   A A * * * .
    2     A * * * .
    3       * * * .
    4         + + A
    5           A A
(i) 6             A
```

(b)

Figure 7: Adding a goal to the goal table

Using the goal table, the place markers l, r in the *complete* can be replaced by i, k. 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. For an optimal worst-case complexity we have to make a small change to the implementation and represent fully completed DD items $[A \rightarrow \bullet\beta\bullet, i, j]$ 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 $O(|G|rn^3)$, 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 $O(|N||G|rn^3)$. The space complexity is $O(|G|rn^2)$ for the chart and agenda, and $O(|N|^2 \times |N||\Sigma|)$ for storage of the $>_h^*$ relation in tabular form.

The obtained worst-case complexity is optimal, in the sense that all complexity factors are properly accounted for (i.e., the factors r and $|N|$ in addition to an optimal Earley parser are evidently necessary). Yet, on a practical level, a large percentage of computing time can be saved by adding some more sophistication to the algorithm. We will not formally introduce an optimized algorithm, as the definitions grow rather complicated, but simply state some principles that can be implemented straightforwardly.

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

- 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 $A >_h^* X$ 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 article.

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 transitively 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 *agreement* 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 $[B \rightarrow \alpha \bullet \beta \bullet \gamma, i, j]$ are replaced by items $[l, r, A; B \rightarrow \alpha \bullet \beta \bullet \gamma, i, j]$, with A the predicted goal symbol. The transitive features of A are shared with B. If we have an item $[l, r, A; A \rightarrow \bullet\alpha\bullet, l, j]$ 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 in (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 in (Matsumoto et al, 1983). This BUP parser is limited to acyclic, ε-free grammars. In (Nederhof, 1992)a generalized LC parser is defined by analogy to Tomita's generalized LR parser (Tomita, 1985). Nederhof's parser is more efficient than our LC chart parser, but it doesn't generalize to HC.

The head-driven parser in (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 in (Kay, 1989).

The context-free head grammars in Section 4 should not be confused with Head Grammars as introduced in (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 have experimented with various parsing strategies for unification grammars (Bouma and van Noord, 1993) 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.

8 Conclusions and future work

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 ε-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 happens 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.

We have indicated how the algorithm can be extended with feature information. An implementation of a head-corner chart parser for PATR-like unification grammars is (nearly) finished. It is currently being tested on a natural language grammar developed for a knowledge representation research project at our institute. We intend to make an extensive comparison of

the efficiency of the head-corner and left-corner parser.

Acknowledgements

We are grateful to Anton Nijholt, Mark-Jan Nederhof, Gertjan van Noord and Giorgio Satta for constructive comments on earlier drafts and Margriet Verlinden for implementing the HC chart parser for PATR.

References

Bouma, G. and G. van Noord (1993). Head-driven Parsing for Lexicalist Grammars: Experimental Results. *6th European Chapter of the ACL*, 71–80.

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

Kay, M. (1980). Algorithm Schemata and Data Structures in Syntactic Processing. Report CSL-80-12, Xerox PARC, Palo Alto, Ca. Reprinted in: GROSZ, B.J. et al. (Eds.), *Readings in Natural Language Processing*, Morgan Kaufmann, Los Altos, Ca. (1982).

Kay, M. (1989). Head Driven Parsing. *Int. Workshop on Parsing Technologies (IWPT'89)*, 52–62.

Matsumoto, Y., H. Tanaka, H. Hirakawa, H. Miyoshi and H. Yasukawa (1983). BUP: a bottom-up parser embedded in Prolog. *New Generation Computing* **1**, 145–158.

Nederhof, M.-J. (1993). Generalized Left-Corner Parsing. *6th European Chapter of the ACL*, 305–314.

Noord, G. van (1991). Head Corner Parsing for Discontinuous Constituency. *29th Ann. Meeting of the ACL*, 114–121.

Pollard, C. (1984). *Generalized Context-Free Grammars, Head Grammars and Natural Languages*. Ph.D. Thesis, Dept. of Linguistics, Stanford University.

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

Proudian, D. and C. Pollard (1985). Parsing head-driven phrase structure grammar. *23th Ann. Meeting of the ACL*, 167–171.

Rosenkrantz, D.J. and P.M. Lewis (1970). Deterministic Left Corner Parsing. *11th Ann. Symp. on Switching and Automata Theory*, 139–152.

Satta, G. and O. Stock (1989). Head-Driven Bidirectional Parsing: A Tabular Method. *Int. Workshop on Parsing Technologies (IWPT'89)*, 43–51.

Satta, G. and O. Stock (1992). BiDirectional Context-Free Grammar Parsing for Natural Language Processing. Technical Report IRCS-92-13, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia.
To appear in *J. of Artificial Intelligence.*

Sikkel, K. and R. op den Akker (1992). Left-Corner and Head-Corner Chart Parsing. Memoranda Informatica 92-55, University of Twente, Enschede, the Netherlands.

Tomita, M. (1985). *Efficient Parsing for Natural Language*. Kluwer, Boston, Mass.