Efficient Parsing for Bilexical Context-Free Grammars

and Head Automaton Grammars

Jason Eisner and Giorgio Satta

January 24, 1999

Abstract

Several recent stochastic parsers use bilexical grammars, where each word type idiosyncratically prefers particular complements with particular head words. We present $O(n^4)$ parsing algorithms for two bilexical formalisms, improving the previous upper bounds of $O(n^5)$. Also, for a common special case that was known to allow $O(n^3)$ parsing (Eisner, 1997), we present an $O(n^3)$ algorithm with an improved grammar constant.

1 Introduction

Lexicalized grammar formalisms are of both theoretical and practical interest to the computational linguistics community. Such formalisms specify syntactic facts about each word of the language—in particular, the type of arguments that the word can or must take. Early mechanisms of this sort included categorial grammar (Bar-Hillel, 1953) and subcategorization frames (Chomsky, 1965). Other lexicalized formalisms include (Schabes et al., 1988; Mel'cuk, 1988; Pollard and Sag, 1994).

Besides the possible arguments of a word, a natural-language grammar does well to specify possible head words for those arguments. “Convene” always takes an NP object, but some NPs are more semantically or lexically appropriate here than others, and the appropriateness is largely a function of the NP’s head (e.g., “meeting”). We use the general term bilexical for a grammar that records such facts. A bilexical grammar makes many stipulations about the compatibility of particular pairs of words in particular roles. The acceptability of “Nora convened the party” then depends on the grammar writer’s assessment of whether parties can be convened.

Several recent real-world parsers have improved state-of-the-art parsing accuracy by relying on probabilistic
or weighted versions of bilexical grammars (Alshawi, 1996; Eisner, 1996; Charniak, 1997; Collins, 1997). The rationale is that soft selectional restrictions play a crucial role in disambiguation.\(^1\)

The chart parsing algorithms used by most of the above authors run in time \(O(n^5)\), because bilexical grammars are enormous (the part of the grammar relevant to a length-\(n\) input has size \(O(n^2)\) in practice). Heavy probabilistic pruning is therefore needed to get acceptable runtimes. But in this paper we show that the complexity is not so bad after all:

- For bilexicalized context-free grammars, \(O(n^4)\) is possible.
- The \(O(n^4)\) result also holds for head automaton grammars.
- For a very common special case of these grammars where an \(O(n^3)\) algorithm was previously known (Eisner, 1997), the grammar constant can be reduced without harming the \(O(n^3)\) property.

Our algorithmic technique throughout is to propose new kinds of sub-derivations that are not constituents. We use dynamic programming to assemble such sub-derivations into a full parse.

### 2 Notation for context-free grammars

The reader is assumed to be familiar with context-free grammars. Our notation follows (Harrison, 1978; Hopcroft and Ullman, 1979). A context-free grammar (CFG) is a tuple \(G = (V_N, V_T, P, S)\), where \(V_N\) and \(V_T\) are finite, disjoint sets of nonterminal and terminal symbols, respectively, and \(S \in V_N\) is the start symbol. Set \(P\) is a finite set of productions having the form \(A \rightarrow a\), where \(A \in V_N, a \in (V_N \cup V_T)^*\). If every production in \(P\) has the form \(A \rightarrow BC\) or \(A \rightarrow a\), for \(A, B, C \in V_N, a \in V_T\), then the grammar is said to be in Chomsky Normal Form (CNF).\(^2\) Every language that can be generated by a CFG can also be generated by a CFG in CNF.

In this paper we adopt the following conventions: \(a, b, c, d\) denote symbols in \(V_T\), \(w, x, y\) denote strings in \(V_T^*\), and \(A, \beta, \ldots\) denote strings in \((V_N \cup V_T)^*\). The input to the parser will be a CFG \(G\) together with a string of terminal symbols to be parsed, \(w = d_1d_2\ldots d_n\). Also \(h, i, j, k\) denote positive integers, which are assumed to be \(\leq n\) when we are treating them as indices into \(w\). We write \(w_{i,j}\) for the input substring \(d_i\ldots d_j\) (and put \(w_{i,j} = \epsilon\) for \(i > j\)).

\(^1\) Other relevant parsers simultaneously consider two or more words that are not necessarily in a dependency relationship. These include (Lafferty et al., 1992; Magerman, 1995; Collins and Brooks, 1995; Chelba and Jelinek, 1998).

\(^2\) In order to admit \(\epsilon\) as a string of the language, the production \(S \rightarrow \epsilon\) is also allowed in a CNF grammar so long as \(S\) does not appear on the right-hand side of any production. However, \(S \rightarrow \epsilon\) is not allowed in bilexical CFGs.
A “derives” relation, written ↝, is associated with a CFG as usual. We also use the reflexive and transitive closure of ↝, written ↝*, and define \( L(G) \) accordingly. We write \( a \underbrace{\beta \delta} \Rightarrow \alpha \gamma \delta \) for a derivation in which only \( \beta \) is rewritten.

3 Bilexical context-free grammars

We introduce next a grammar formalism that captures lexical dependencies among pairs of words in \( V_T \). This formalism closely resembles stochastic grammatical formalisms that are used in several existing natural language processing systems (see §1). We will specify a non-stochastic version, noting that probabilities or other weights may be attached to the rewrite rules exactly as in stochastic CFG (Gonzales and Thomason, 1978; Wetherell, 1980). (See §4 for brief discussion.)

Suppose \( G = (V_N, V_T, P, T[8]) \) is a CFG in CNF.\(^3\) We say that \( G \) is **bilexical** iff there exists a set of “delexicalized nonterminals” \( V \) such that \( V_N = \{ A[a] : A \in V, a \in V_T \} \) and every production in \( P \) has one of the following forms:

\[
\begin{align*}
1. & \quad A[a] \rightarrow B[b] \ C[a] \\
2. & \quad A[a] \rightarrow C[a] \ B[b] \\
3. & \quad A[a] \rightarrow a
\end{align*}
\]

A constituent of nonterminal type \( A[a] \) is said to have terminal symbol \( a \) as its **lexical head**.

Notice that the start symbol is necessarily a lexicalized nonterminal, \( T[8] \). Hence \( 8 \) appears in every string of \( L(G) \); it is usually convenient to define \( G \) so that the language of interest is actually \( L'(G) = \{ x : x\hat{8} \in L(G) \} \).

Such a grammar can encode lexically specific preferences. For example, \( P \) might contain the productions

\[
\begin{align*}
1. & \quad VP[\text{solve}] \rightarrow V[\text{solve}] \ NP[\text{puzzles}] \\
2. & \quad NP[\text{puzzles}] \rightarrow DET[\text{two}] \ N[\text{puzzles}] \\
3. & \quad V[\text{solve}] \rightarrow \text{solve} \\
4. & \quad N[\text{puzzles}] \rightarrow \text{puzzles} \\
5. & \quad DET[\text{two}] \rightarrow \text{two}
\end{align*}
\]

\(^3\) We have a more general definition that drops the restriction to CNF, but do not give it here.
in order to allow the derivation VP[solve] ⇒* solve two puzzles, but meanwhile omit the similar productions

- VP[eat] → V[eat] NP[puzzles]
- VP[solve] → V[solve] NP[goat]
- VP[sleep] → V[sleep] NP[goat]
- NP[goat] → DET[two] N[goat]

since puzzles are not edible, a goat is not solvable, “sleep” is intransitive, and “goat” cannot take plural determiners. (A stochastic version of the grammar could implement “soft preferences” by allowing the rules in the second group but assigning them various low probabilities.)

The cost of this expressiveness is a very large grammar. Standard context-free parsing algorithms are inefficient in such a case. The CKY algorithm (Younger, 1967; Aho and Ullman, 1972) takes time \( O(n^3 \cdot |P|) \), where in the worst case \( |P| = |V_T|^3 \). For a bilexical grammar, the worst case is \( |P| = |V|^3 \cdot |V_T|^2 \), which is large for a large vocabulary \( V_T \). We may improve the analysis somewhat by observing that when parsing \( d_1 \cdots d_n \), the CKY algorithm will only consider nonterminals of the form \( A[d_i] \); by restricting to the relevant productions we obtain \( O(n^5 \cdot V^3 \cdot \min(n, |V_T|)^2) \).

We observe that in practical applications we always have \( n \ll |V_T| \). Let us then restrict our analysis to the (infinite) set of input instances of the parsing problem that satisfy relation \( n < |V_T| \). With this assumption, the asymptotic time complexity of the CKY algorithm becomes \( O(n^5 \cdot V^3) \). In other words, it is a factor of \( n^2 \) slower than a comparable non-lexicalized CFG.

4 Bilexical CFG in time \( O(n^4) \)

In this section we give a recognition algorithm for bilexical CNF context-free grammars, which runs in time \( O(n^4 \cdot \max(p, |V|^2)) = O(n^4 \cdot |V|^3) \). Here \( p \) is the maximum number of productions sharing the same pair of terminal symbols (e.g., the pair \((b, a)\) in production (1)). The new algorithm is asymptotically more efficient than the CKY algorithm, when restricted to input instances satisfying the relation \( n < |V_T| \).

Where CKY recognizes only constituent substrings of the input, the new algorithm can recognize three types of subderivations, shown and described in Figure 1(a). A declarative specification of the algorithm is given in Figure 1(b–c). The derivability conditions of (a) are guaranteed by (b), by induction, and the correctness of (c)
(a) \( \begin{array}{l}
\text{(i \leq h \leq j, A \in V)} \\
\text{(i \leq j < h, A, C \in V)} \\
\text{(h < i \leq j, A, C \in V)}
\end{array} \) is derived if \( A[h] \Rightarrow * w_{i,j} \)

\( \begin{array}{l}
\text{(i \leq h \leq j, A \in V)} \\
\text{(i \leq j < h, A, C \in V)} \\
\text{(h < i \leq j, A, C \in V)}
\end{array} \) is derived if \( A[h] \Rightarrow B[d_{i,j}] \Rightarrow C[d_{i,j}] \) for some \( B, h' \)

(b) Start: \( A[h] \rightarrow d_h \)

Attach-Left:

\( \begin{array}{l}
\text{B} \\
\text{i \leq h \leq j}
\end{array} \)

\( \begin{array}{l}
A[h] \rightarrow B[d_{i,j}]C[d_{i,j}]
\end{array} \)

Attach-Right:

\( \begin{array}{l}
\text{C} \\
\text{i \leq j < h, A, C \in V}
\end{array} \)

\( \begin{array}{l}
A[h] \rightarrow C[d_{i,j}]B[d_{i,j}]
\end{array} \)

Complete-Left:

\( \begin{array}{l}
\text{C} \\
\text{i \leq j < h, A, C \in V}
\end{array} \)

\( \begin{array}{l}
A[h] \rightarrow C[d_{i,j}]B[d_{i,j}]
\end{array} \)

Complete-Right:

\( \begin{array}{l}
\text{C} \\
\text{i \leq j < h, A, C \in V}
\end{array} \)

\( \begin{array}{l}
A[h] \rightarrow C[d_{i,j}]B[d_{i,j}]
\end{array} \)

(c) \( \begin{array}{l}
T \\
\text{i \leq h \leq j, A, C \in V}
\end{array} \)

is derived for some \( h \) such that \( d_h = \$ \).

Figure 1: An \( O(n^4) \) recognition algorithm for bilexical context-free grammars in Chomsky Normal Form. (a) Types of items in the parse table (chart). The first is syntactic sugar for the tuple \([\Delta, A, i, h, j]\), and so on. The stated conditions assume that \( d_1, \ldots, d_n \) are all distinct. (b) Inference rules. The algorithm derives the item below if the items above have already been derived and any condition to the right of is met. (c) Condition for accepting input \( w = d_1 d_2 \cdots d_n \).

This declarative specification, like CKY, may be implemented by bottom-up dynamic programming. We sketch one such method. For each possible item, as shown in (a), we maintain a bit (indexed by the parameters of the item) that records whether the item has been derived yet. All these bits are initially zero. The algorithm makes a single pass through the possible items, setting the bit for each if it can be derived using any rule in (b) from items whose bits are already set. At the end of this pass it is straightforward to test (c). The pass considers the items in increasing order of width, where the width of an item in (a) is defined as \( \max\{h, i, j\} - \min\{h, i, j\} \). Among items of the same width, those of type \( \Delta \) should be considered last.

The algorithm requires space proportional to the number of possible items, which is at most \( n^3 |V|^2 \). Each of the five rule templates can instantiate its free variables in at most \( n^4 p \) for (Complete rules) \( n^4 |V|^2 \) different ways, each of which is tested once and in constant time; so the runtime is \( O(n^4 \max(p, |V|^2)) \).

By comparison, the CKY algorithm uses only the first type of item, and relies on rules whose inputs are pairs
Such rules can be instantiated in $O(n^5)$ different ways for a fixed grammar, yielding $O(n^5)$ time complexity. The new algorithm saves a factor of $n$ by combining those two constituents in two steps, one of which is insensitive to $k$ and abstracts over its possible values, the other of which is insensitive to $h'$ and abstracts over its possible values.

It is straightforward to turn the new $O(n^4)$ recognition algorithm into a parser for stochastic bilexical CFGs (or other weighted bilexical CFGs). In a stochastic CFG, each nonterminal $A[a]$ is accompanied by a probability distribution over productions of the form $A[a] \rightarrow \alpha$. A parse is just a derivation (proof tree) of $T = \bigwedge_{n}^{n}$, and its probability—like that of any derivation we find—is defined as the product of the probabilities of all productions used to condition inference rules in the proof tree. The highest-probability derivation for any item can be reconstructed recursively at the end of the parse, provided that each item maintains not only a bit indicating whether it can be derived, but also the probability and instantiated root rule of its highest-probability derivation tree.

5 A more efficient variant

We now give a variant of the algorithm of §4; the variant has the same asymptotic complexity but will often be faster in practice.

Notice that the Attach-Left rule of Figure 1(b) tries to combine the nonterminal label $B[d_b]$ of a previously derived constituent with every possible nonterminal label of the form $C[d_h]$. The improved version, shown in Figure 2, restricts $C[d_h]$ to be the label of a previously derived adjacent constituent. This improves speed if there are not many such constituents and we can enumerate them in $O(1)$ time apiece (using a sparse parse table to store the derived items).

It is necessary to use an agenda data structure (Kay, 1980) when implementing the declarative algorithm of Figure 2. Deriving narrower items before wider ones as before will not work here because the rule Halve derives narrow items from wide ones.
(a) $A_{i \leq h \leq j, A \in V}$ is derived iff $A[d_h] \Rightarrow w_{i,j}$

(b) $A_{i \leq h, A \in V}$ is derived iff $A[d_h] \Rightarrow w_{i,j}$ for some $j \geq h$

(c) $A_{h \leq j, A \in V}$ is derived iff $A[d_h] \Rightarrow w_{i,j}$ for some $i \leq h$

(d) $A_{i \leq j < h, A, C \in V}$ is derived iff $A[d_h] \Rightarrow B[d_{h'}] C[d_h] \Rightarrow w_{i,j} C[d_h] \Rightarrow w_{i,k}$ for some $B, h', k$

(e) $A_{h < i \leq j, A, C \in V}$ is derived iff $A[d_h] \Rightarrow C[d_h] B[d_{h'}] \Rightarrow C[d_h] w_{i,j} \Rightarrow w_{k,j}$ for some $B, h', k$

Figure 2: A more efficient variant of the $O(n^4)$ algorithm in Figure 1, in the same format.
Multiple word senses

Rather than parsing an input string directly, it is often desirable to parse another string related by a (possibly stochastic) transduction. Let $T$ be a finite-state transducer that maps a morpheme sequence $w \in V_T$ to its orthographic realization, a grapheme sequence $\bar{w}$. $T$ may realize arbitrary morphological processes, including affixation, local clitic movement, deletion of phonological nulls, forbidden or dispreferred $k$-grams, typographical errors, and mapping of multiple senses onto the same grapheme. Given grammar $G$ and an input $\bar{w}$, we ask whether $\bar{w} \in T(L(G))$. We have extended all the algorithms in this paper to this case: the items simply keep track of the transducer state as well.

Due to space constraints, we sketch only the special case of multiple senses. Suppose that the input is $\bar{w} = \bar{a}_1 \cdots \bar{a}_n$, and each $\bar{a}_k$ has up to $g$ possible senses. Each item now needs to track its head’s sense along with its head’s position in $\bar{w}$. Wherever an item formerly recorded a head position $h$ (similarly $h'$), it must now record a pair $(h, d_h)$, where $d_h \in V_T$ is a specific sense of $\bar{a}_h$. No rule in Figures 1-2 (or Figure 3 below) will mention more than two such pairs. So the time complexity increases by a factor of $O(g^2)$.

Head automaton grammars in time $O(n^4)$

In this section we show that a length-$n$ string generated by a head automaton grammar (Alshawi, 1996) can be parsed in time $O(n^4)$. We do this by providing a translation from head automaton grammars to bilexical CFGs. This result improves on the head-automaton parsing algorithm given by Alshawi, which is analogous to the CKY algorithm on bilexical CFGs and is likewise $O(n^5)$ in practice (see §3).

A head automaton grammar (HAG) is a function $H : a \mapsto H_a$ that defines a head automaton (HA) for each element of its (finite) domain. Let $V_T = \text{domain}(H)$ and $D = \{-\to, -\cdot\}$. A special symbol $\$ \in V_T$ plays the role of start symbol. For each $a \in V_T$, $H_a$ is a tuple $(Q_a, V_T, \delta_a, I_a, F_a)$, where

- $Q_a$ is a finite set of states;
- $I_a, F_a \subseteq Q_a$ are sets of initial and final states, respectively;
- $\delta_a$ is a transition function mapping $Q_a \times V_T \times D$ to $2^{Q_a}$, the power set of $Q_a$.

Translation in the other direction is possible if the HAG formalism is extended to allow multiple senses per word (see §6). In that case the two formalisms are equivalent.
A single head automaton is an acceptor for a language of string pairs \( \langle z_l, z_r \rangle \in V_T^* \times V_T^* \). Informally, if \( b \) is the leftmost symbol of \( z_r \) and \( q' \in \delta_a(q, b, \epsilon) \), then \( H_a \) can move from state \( q \) to state \( q' \), matching symbol \( b \) and removing it from the left end of \( z_r \). Symmetrically, if \( b \) is the rightmost symbol of \( z_l \) and \( q' \in \delta_a(q, b, \epsilon) \) then from state \( q \) \( H_a \) can move to state \( q' \), matching symbol \( b \) and removing it from the right end of \( z_l \).

More formally, we associate with the head automaton \( H_a \) a “derives” relation \( \vdash_a \), defined as a binary relation on \( Q_a \times V_T^* \times V_T^* \). For every \( q \in Q, x, y \in V_T^*, b \in V_T, d \in D, \) and \( q' \in \delta_a(q, b, d) \), we specify that

\[
(q, xb, y) \vdash_a (q', x, y) \quad \text{if} \quad d = \\epsilon; \\
(q, x, by) \vdash_a (q', x, y) \quad \text{if} \quad d = \\epsilon.
\]

The reflexive and transitive closure of \( \vdash_a \) is written \( \vdash_a^* \). The language generated by \( H_a \) is the set

\[
L(H_a) = \{ \langle z_l, z_r \rangle \mid (q, z_l, z_r) \vdash_a^* (r, \epsilon, \epsilon), q \in I_a, r \in F_a \}.
\]

We may now define the language generated by the entire grammar \( H \). To generate, we expand the start word \( \$$ \in V_T \) into \( x\$$ \ for some \( \langle x, y \rangle \in L(H_\$$) \), and then recursively expand the words in strings \( x \) and \( y \). More formally, given \( H \), we simultaneously define \( L_a \) for all \( a \in V_T \) to be minimal such that

\[
\text{if } \langle x, y \rangle \in L(H_a), x' \in L_{a_x}, y' \in L_{a_y}, \text{ then } x'ay' \in L_a
\]

where \( L_{a_1 \cdots a_s} \) stands for the concatenation language \( L_{a_1} \cdots L_{a_s} \). Then \( H \) generates the language \( L_\$$.

We next present a simple construction that transforms a HAG \( H \) into a bilexical CFG \( G \) generating the same language. The construction also preserves derivation ambiguity. This means that for each string \( w \), there is a linear-time 1-to-1 mapping between (appropriately defined) canonical derivations of \( w \) by \( H \) and canonical derivations of \( w \) by \( G \).

We adopt the notation above for \( H \) and the components of its head automata. Let \( V \) be an arbitrary set of size \( t = \max\{ |Q_a| : a \in V_T \} \), and for each \( a \), define an arbitrary injection \( f_a : Q_a \rightarrow V \). We define

\[
G = (V_N, V_T, P, T[8]),
\]

where

\[\text{Alshawi (1996) describes HAs as accepting (or equivalently, generating) } z_l \text{ and } z_r \text{ from the outside in. To make Figure 3 easier to follow, we have defined HAs as accepting symbols in the opposite order, from the inside out. This amounts to the same thing if transitions are reversed, } I_a \text{ is exchanged with } F_a, \text{ and any transition probabilities are replaced by those of the reversed Markov chain.}\]
(i) $V_N = \{ A[a]: A \in V, a \in V_T \}$, in the usual manner for bilexical CFGs;

(ii) $P$ is the set of all productions having one of the following forms, where $a, b \in V_T$:

- $A[a] \rightarrow B[b] C[q]$ where $A = f_a(r), B = f_b(q'), C = f_c(q)$ for some $q' \in I_b$, $q \in Q_a, r \in \delta_c(q, b, -)$
- $A[a] \rightarrow C[q] B[b]$ where $A = f_a(r), B = f_b(q'), C = f_c(q)$ for some $q' \in I_b$, $q \in Q_a, r \in \delta_c(q, b, -)$
- $A[a] \rightarrow a$ where $A = f_a(r)$ for some $q \in F_a$

(iii) $T = f_b(q)$, where we assume WLOG that $I_b$ is a singleton set \{q\}.

We omit the formal proof that $G$ and $H$ admit isomorphic derivations and hence generate the same languages, observing only that if $(x, y) = (b_1 b_2 \cdots b_j b_{j+1} \cdots b_k) \in L(H_a)$ — a condition used in defining $I_a$ above — then $A[a] \Rightarrow^* B_1[b_1] \cdots B_j[b_j] a B_{j+1}[b_{j+1}] \cdots B_k[b_k]$, for any $A, B_1, \ldots, B_k$ that map to initial states in $H_a, H_{b_1}, \ldots H_{b_k}$ respectively.

In general, $G$ has $p = O(|V|^3) = O(t^3)$. The construction therefore implies that we can parse a length-$n$ sentence under $H$ in time $O(n^4 t^3)$. If the HAs in $H$ happen to be deterministic, then in each binary production given by (ii) above, symbol $A$ is fully determined by $a, b,$ and $C$. In this case $p = O(t^2)$, so the parser will operate in time $O(n^4 t^2)$.

We note that this construction can be straightforwardly extended to convert stochastic HAGs as in (Alshawi, 1996) into stochastic CFGs. Probabilities that $H_a$ assigns to state $q$'s various transition and halt actions are copied onto the corresponding productions $A[a] \rightarrow a$ of $G$, where $A = f_a(q)$.

8 Split head automaton grammars in time $O(n^3)$

For many bilexical CFGs or HAGs of practical significance, as for the bilexical version of link grammars (Lafferty et al., 1992), it is possible to parse length-$n$ inputs even faster, in time $O(n^3)$ (Eisner, 1997). In this section we describe and discuss this special case, and give a new $O(n^3)$ algorithm that has a smaller grammar constant than previously reported.

A head automaton $H_a$ is called split if it has no states that can be entered on a $\rightarrow$ transition and exited on a $\leftarrow$ transition. Such an automaton can accept $(x, y)$ only by reading all of $y$ — immediately after which it is said to be in a flip state — and then reading all of $x$. Formally, a flip state is one that allows entry on a $\leftarrow$ transition and that either allows exit on a $\rightarrow$ transition or is a final state.
We are concerned here with head automaton grammars $H$ such that every $H_a$ is split. These correspond to bilexical CFGs in which any derivation $A[a] \Rightarrow^* x ay$ has the form $A[a] \Rightarrow^* xH[a] \Rightarrow^* x ay$. That is, a word's left dependents are more oblique than its right dependents and c-command them.

Such grammars are broadly applicable. Even if $H_a$ is not split, there usually exists a split head automaton $H'_a$ recognizing the same language. $H'_a$ exists iff $\{x \neq y : (x, y) \in L(H_a)\}$ is regular (where $\not\in V_T$). In particular, $H'_a$ must exist unless $H_a$ has a cycle that includes both $\rightarrow$- and $\leftarrow$-transitions. Such cycles would be necessary for $H_a$ itself to accept a formal language such as $\{a^n b^n : n \geq 0\}$, but we know of no natural-language motivation for ever using them in a HAG.

One more definition will help us bound the complexity. A split head automaton $H_a$ is said to be $g$-split if its set of flip states, denoted $Q_a \subseteq Q_a$, has size $\leq g$. The languages that can be recognized by $g$-split HAs are those that can be written as $\bigcup_{i=1}^g L_i \times R_i$, where the $L_i$ and $R_i$ are regular languages over $V_T$. Eisner (1997) actually defined ($g$-split) bilexical grammars in terms of the latter property.\(^6\)

We now present our result: Figure 3 specifies an $O(n^3 g^2 t^2)$ recognition algorithm for a head automaton grammar $H$ in which every $H_a$ is $g$-split. For deterministic automata, the runtime is $O(n^3 g^2 t)$—a considerable improvement on the $O(n^3 g^2 t^2)$ result of (Eisner, 1997), which also assumes deterministic automata. As in §4, a simple bottom-up implementation will suffice. For a practical speedup, add $\bigtriangleup_{h,j}$ as an antecedent to the Min rule (and fill in the parse table from right to left).

Like our previous algorithms, this one takes two steps (Attach, Complete) to attach a child constituent to a parent constituent. But instead of full constituents—strings $xd_iy \in L_d$—it uses only half-constituents such as $xd_i$ and $d_iy$. Where CKY would combine $\bigtriangleup_{i,R,j,j+1,h,k}$, we save two degrees of freedom $i,k$ (thereby improving $O(n^5)$ to $O(n^3)$) and combine $\bigtriangleup_{i,R,j,j+1,h}$. The other halves of these constituents can be attached later, because to find an accepting path for $(z_i,z_r)$ in a split head automaton, one can separately find the half-path before the flip state (which accepts $z_r$) and the half-path after the flip state (which accepts $z_i$). These two half-paths can subsequently be joined into an accepting path if they have the same flip state $s$, i.e., one path starts where the other ends. Annotating our left half-constituents with $s$ makes this check possible.

\(^6\)That paper associated a product language $L_i \times R_i$, or equivalently a 1-split HA, with each of $g$ senses of a word (see §6). One could do the same without penalty in our present approach: confining to 1-split automata would remove the $g^2$ complexity factor, and then allowing $g$ senses would restore the $g^2$ factor. Indeed, this approach gives added flexibility: a word's sense, unlike its choice of flip state, is visible to the HA that reads it.
(a) \( \frac{q}{h} \) \( \frac{q}{j} \) \( h \leq j, q \in Q_{d_h} \) is derived iff \( d_h : I \xrightarrow{x} q \) where \( w_{h+1,j} \in L_x \)

\( \frac{q}{i} \) \( \frac{q}{h} \) \( i \leq h, q \in Q_{d_h} \cup \{ F \}, s \in \tilde{Q}_{d_h} \) is derived iff \( d_h : q \xrightarrow{x} s \) where \( w_{i,h-1} \in L_x \)

\( \frac{q}{h} \) \( \frac{q}{h'} \) \( h < h', q \in Q_{d_h}, s' \in \tilde{Q}_{d_{h'}} \) is derived iff \( d_h : I \xrightarrow{x_{d_{h'}}} q \) and \( d_{h'} : F \xrightarrow{y} s' \) where \( w_{h+1,h'-1} \in L_{xy} \)

\( \frac{q}{h} \) \( \frac{q}{h'} \) \( h' < h, q \in Q_{d_h}, s \in \tilde{Q}_{d_{h'}}, s' \in \tilde{Q}_{d_h} \) is derived iff \( d_{h'} : I \xrightarrow{x} s' \) and \( d_h : q \xrightarrow{d_{h'} y} s \) where \( w_{h+1,h'-1} \in L_{xy} \)

(b) Start: \( \frac{q}{h} \) \( q \in I_{d_h} \)  
Mid: \( \frac{s}{h} \) \( s \in \tilde{Q}_{d_h} \)  
Finish: \( \frac{q}{F} \) \( q \in F_{d_h} \)

Attach-Right: \( \frac{q}{h} \) \( \frac{F}{i} \) \( i \in \delta_h (q, d_{h'}, \rightarrow) \)

Complete-Right: \( \frac{q}{h} \) \( \frac{s'}{h} \) \( h' \rightarrow \)

Attach-Left: \( \frac{s'}{h'} \) \( \frac{q}{h} \) \( i \in \delta_{d_{h'}} (q, d_{h'}, \rightarrow) \)

Complete-Left: \( \frac{F}{h} \) \( \frac{q}{i} \) \( h' \rightarrow \)

(c) \( \frac{q}{h} \) and \( \frac{s}{n} \) are derived for some \( h, s \) such that \( d_h = \$ \).

Figure 3: An \( O(n^3) \) recognition algorithm for split head automaton grammars. The format is as in Figure 1. The following notation indicates that a head automaton can consume a string \( x \) from its left or right input: \( a : q \xrightarrow{x} q' \) means that \( (q, \epsilon, x) \xrightarrow{a} (q', \epsilon, \epsilon) \), and \( a : I \xrightarrow{x} q' \) means this is true for some \( q \in I_a \). Similarly, \( a : q' \xrightarrow{x} q \) means that \( (q, x, \epsilon) \xrightarrow{a} (q', \epsilon, \epsilon) \), and \( a : F \xrightarrow{y} q \) means this is true for some \( q' \in F_a \). The special symbol \( F \) also appears as a literal in some items, and effectively means “an unspecified final state.”
9 Final remarks

We have formally described, and given faster parsing algorithms for, three practical grammatical rewriting systems that capture dependencies between pairs of words. All three systems admit naive \(O(n^3)\) algorithms. We give the first \(O(n^4)\) results for the natural formalism of bilexical context-free grammar, and for Alshawi’s (1996) head automaton grammars. For the usual case of split head automaton grammars, we replace the \(O(n^3)\) algorithm of (Eisner, 1997) by one with a smaller grammar constant.

The dynamic programming techniques we have presented for cheaply attaching head information to derivations can also be exploited in parsing formalisms other than rewriting systems. The authors have developed an \(O(n^7)\)-time parsing algorithm for bilexicalized tree adjoining grammars, improving the naive \(O(n^8)\) method.

The results mentioned in §6 are related to the closure property of CFGs under generalized sequential machine mapping (Hopcroft and Ullman, 1979). This property also holds for our class of bilexical CFGs.

References

A. V. Aho and J. D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

Hiyan Alshawi. 1996. Head automata and bilingual tiling; Translation with minimal representations. In *Proc. of ACL*, pages 167–176, Santa Cruz, CA.

Y. Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

A. Bossi, N. Cocco, and L. Colussi. 1983. A divide-and-conquer approach to general context-free parsing. *Information Processing Letters*, 16:203–208.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proc. of COLING-ACL*.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.

M. Collins and J. Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proc. of the Third Workshop on Very Large Corpora*, Cambridge, MA.

Michael J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th ACL and 8th European ACL*, Madrid, July.

Jason Eisner. 1996. An empirical comparison of probability models for dependency grammar. Technical Report IRCS-96-11, Institute for Research in Cognitive Science, Univ. of Pennsylvania.

Jason Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the 1997 International Workshop on Parsing Technologies*, MIT, Cambridge, MA, September.

R. C. Gonzales and M. G. Thomason. 1978. *Syntactic Pattern Recognition*. Addison-Wesley, Reading, MA.

M. A. Harrison. 1978. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, MA.

J. E. Hopcroft and J. D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA.

M. Kay. 1986. Algorithm schemata and data structures in syntactic processing. Technical report CSL-80, Xerox Palo Alto Research Center, Palo Alto, CA. Also in: B. J. Grosz, K. Sparck Jones and B. L. Webber, editors, *Natural Language Processing*, pages 35-70, Kaufmann, Los Altos, CA, 1986.
John Lafferty, Daniel Sleator, and Davy Temperley. 1992. Grammatical trigrams: A probabilistic model of link grammar. In Proc. of the AAAI Conf. on Probabilistic Approaches to Natural Language, October.

David Magerman. 1995. Statistical decision-tree models for parsing. In Proceedings of the 33rd Annual Meeting of the ACL.

Igor A. Mel'cuk. 1988. Dependency Syntax: Theory and Practice. State University of New York Press.

Carl Pollard and Ivan Sag. 1994. Head-Driven Phrase Structure Grammar. University of Chicago Press.

Yves Schabes, Anne Abeillé, and Aravind Joshi. 1988. Parsing strategies with ‘lexicalized’ grammars: Application to Tree Adjoining Grammars. In Proceedings of COLING-88, Budapest, August.

C. S. Wetherell. 1980. Probabilistic languages: A review and some open questions. Computing Surveys, 12(4):361–379.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time $n^3$. Information and Control, 16(2):189–208, February.