Polynomial Charts for Totally Unordered Languages

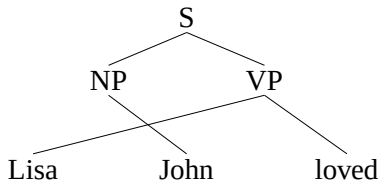
Anders Søgaard
Center for Language Technology
Njalsgade 80
DK-2300 Copenhagen
anders@cst.dk

Abstract

Totally unordered or discontinuous composition blows up chart size on most setups. This paper analyzes the effects of total unordering to type 2 grammars and simple attribute-value grammars (s-AVGs). In both cases, charts turn exponential in size. It is shown that the k-ambiguity constraint turns charts polynomial, even for s-AVGs. Consequently, tractable parsing can be deviced.

1 Introduction

It is common knowledge among linguists that in many languages, the daughters of syntactic constituents can be locally reordered with little or no effect on grammaticality. Certain languages – of which Dyirbal and Warlpiri are often-cited members, but that also include Estonian and Finnish – exhibit a much more radical form of unordering, the kind of unordering that has made linguists propose "crossed brances" analyses, e.g.



yielding $\mathbf{love}(\mathbf{John}, \mathbf{Lisa})$. The Finnish translation is *Liisaa Jussi rakasti*. All six permutations of this sentence are grammatical.

Unordered grammars have been suggested in face of intra-constituent free word order. Similarly, a few authors have proposed *totally* unordered grammars in face of free word order phenomena that involve discontinuous constituents. Dowty (1995), originally published in 1989, is often cited as the original source.

This paper is structured as follows: Sect. 2 defines type 2 grammars and s-AVGs, and their totally unordered pendants. Sect. 3 establishes bounds on chart size for these kinds of grammars. Charts for totally unordered grammars are shown to be worst case exponential. In reply to this, Sect. 4 introduces the k-ambiguity constraint, which turns totally unordered charts polynomial again. Of course this means that polynomial time parsing can be deviced.

2 Grammars and total unordering

Our first task is to properly define the grammars in question:

2.1 Type 2 grammars

Definition 2.1 (Type 2 grammars)**.** $G = \langle N, T, P, \{S\}\rangle$ is a type 2 grammar iff every production rule in P is of the form

$$A \rightarrow \omega$$

where $A \in N$ and $\omega \in \{N \cup T\}^+$.

Definition 2.2 (Derivability)**.** For a type 2 grammar G and $\omega_1, \omega_2 \in (N \cup T)^*$, $\omega_1 \Longrightarrow_1 \omega_2$ iff there is a $A \rightarrow \phi \in P$ and there are $\psi_1, \psi_2 \in (N \cup T)^*$ such that $\omega_1 = \psi_1 A \psi_2$ and $\omega_2 = \psi_1 \phi \psi_2$. $\stackrel{*}{\Longrightarrow}_1$ (the derivability relation) is the reflexive transitive closure of $\Longrightarrow_1$.

Definition 2.3 (Type 2 languages)**.** The language of a type 2 grammar G is defined as

Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek and Mare Koit (Eds.)
NODALIDA 2007 Conference Proceedings, pp. 183–190

$$L(G) \quad = \quad \{x \in T^* : S \stackrel{*}{\Longrightarrow}_1 x\}.$$

Definition 2.4 (Chomsky normal form)**.** A type 2 grammar $G = \langle N, T, P, \{S\}\rangle$ is in Chomsky normal form iff each production has one of the following forms:

- $A \rightarrow BC$
- $A \rightarrow a$
- $S \rightarrow \epsilon$

where $a \in T$ and $B, C \in N - \{S\}$.

Example 2.5. Consider the type 2 grammar with rules $S \rightarrow aXb|ab, X \rightarrow aXb|ab$. The Chomsky normal form of this grammar is obtained by adding the rules $A \rightarrow a, B \rightarrow b$ and by reducing the length of the S, X-rules. Consequently, P' now includes:

$$\begin{array}{llllll} S & \rightarrow & AT|AB & T & \rightarrow & XB \\ X & \rightarrow & AT|AB & A & \rightarrow & a \\ B & \rightarrow & b & & & \end{array}$$

Lemma 2.6 (Equivalence of normal forms)**.** *Say $G = \langle N, T, P, \{S\}\rangle$ is a type 2 grammar. There is an algorithm to construct a grammar $G' = \langle N', T, P', \{S'\}\rangle$ in Chomsky normal form that is weakly equivalent to G.*

Proof. See Sudkamp (2005, 122–3). □

2.2 Totally unordered type 2 grammars

Definition 2.7 (Totally unordered type 2 grammars)**.** $G = \langle N, T, P, \{S\}\rangle$ is a type 2 grammar iff every production rule in P is of the form

$$A \quad \rightarrow \quad \omega$$

where $A \in N$ and $\omega \in \{N \cup T\}^*$.

Definition 2.8 (Derivability)**.** If $A \stackrel{*}{\Longrightarrow}_1 \omega$ and $\omega' \in permute(\omega)$, then $A \stackrel{*}{\Longrightarrow}_2 \omega'$.

Definition 2.9 (Totally unordered type 2 languages)**.** The language of a totally unordered type 2 grammar G is defined as

$$L(G) \quad = \quad \{x \in T^* : S \stackrel{*}{\Longrightarrow}_2 x\}.$$

2.3 s-AVGs

s-AVGs are defined over simple attribute-value structures (s-AVSs):

Definition 2.10 (s-AVS)**.** An s-AVS A is defined over a signature $\langle \mathsf{Attr}, \mathsf{Atms}, \rho\rangle$, where $\rho : \mathsf{Attr} \rightarrow 2^{\mathsf{Atms}}$, such that $A \in \mathsf{Attr} \rightarrow 2^{\mathsf{Atms}}$ and $\forall a \in \mathtt{DOM}(A).A(a) \in \rho(a)$.

Definition 2.11 (s-AVG)**.** An s-AVG is a 5-tuple $G = \langle\langle \mathsf{Attr}, \mathsf{Atms}, \rho\rangle, \mathsf{AttrPerc}, T, P, \{S\}\rangle$, where $\mathsf{AttrPerc} \subseteq \mathsf{Attr}$, $\rho : \mathsf{Attr} \rightarrow 2^{\mathsf{Atms}}$, S is an s-AVS, and every production rule in P is of the form $\alpha \rightarrow \omega_i$ or $\alpha_0 \rightarrow \alpha_1 \ldots \alpha_n$ where $n \geq 2$, α_i is an s-AVS, and

(1) $\forall a \in \mathtt{DOM}(\alpha_0) \cap \mathsf{AttrPerc}.\forall 1 \leq i \leq n.f \in \mathtt{DOM}(\alpha_i) \wedge \alpha_i(a) = \alpha_0(a)$

where $\alpha(a)$ is the value of a in the s-AVS α with $\alpha(a) \in \rho(a)$.

Intuitively, the AttrPerc features are agreement features whose values percolate up the tree if defined for every level of it.

Example 2.12. Consider the grammar $G_1 = \langle\langle\{\text{CAT,PLU,PER}\}, \{s, vp, np, v, n, 1, 2, 3, +, -\}, \rho\rangle, \{\text{PLU,PER}\}, \{I, men, John, sleep, sleeps\}, P, S\rangle$ where ρ is the specification of appropriate values of attributes:

$$\begin{array}{lll} \rho(\text{CAT}) & = & \{s, vp, np, v, n\} \\ \rho(\text{PER}) & = & \{1, 2, 3\} \\ \rho(\text{PLU}) & = & \{+, -\} \end{array}$$

and P is the set of production rules:

$$\begin{array}{llll} \left[\text{CAT } s\right] \rightarrow & \left[\text{CAT } np\right], \left[\text{CAT } vp\right] & \left[\text{CAT } vp\right] \rightarrow & \left[\text{CAT } v\right] \\ \left[\text{CAT } np\right] \rightarrow & \left[\text{CAT } n\right] & \begin{bmatrix}\text{CAT } n \\ \text{PER } 1\end{bmatrix} \rightarrow & I \\ \begin{bmatrix}\text{CAT } n \\ \text{PLU } +\end{bmatrix} \rightarrow & men & \begin{bmatrix}\text{CAT } n \\ \text{PLU } - \\ \text{PER } 3\end{bmatrix} \rightarrow & John \\ \begin{bmatrix}\text{CAT } v \\ \text{PLU } +\end{bmatrix} \rightarrow & sleep & \begin{bmatrix}\text{CAT } v \\ \text{PLU } - \\ \text{PER } 3\end{bmatrix} \rightarrow & sleeps \end{array}$$

(1) applies to the subset of attributes $\{\text{PLU,PER}\}$. The start symbol is $S : \left[\text{CAT } s\right]$. The grammar generates exactly the sentences:

(2) I sleep.

(3) Men sleep.

(4) John sleeps.

Definition 2.13 (Subsumption)**.** An s-AVS α subsumes an s-AVS β ($\alpha \sqsubseteq \beta$) iff $\forall a.\texttt{DOM}(a).\alpha(a) = \beta(a)$.

Definition 2.14 (Derivability)**.** Say $G = \langle\langle \mathsf{Attr}, \mathsf{Atms}, \rho\rangle, \mathsf{AttrPerc}, T, P, \{S\}\rangle$ is an s-AVG. If P contains a production $A \rightarrow \omega$, then for any ϕ_1, ϕ_2, $\phi_1 A' \phi_2 \Longrightarrow_3 \phi_1 \omega' \phi_2$ if $A \sqsubseteq A'$ and $\omega \sqsubseteq \omega'$. $\overset{*}{\Longrightarrow}_3$ is the reflexive, transitive closure of $\Longrightarrow_3$.

Definition 2.15 (s-AVG languages)**.** The language of an s-AVG G is defined as

$$L(G) \quad = \quad \{x \in T^* : \exists S'.S' \sqsupseteq S \wedge S' \overset{*}{\Longrightarrow}_3 x\}.$$

2.4 Totally unordered s-AVGs

Call totally unordered s-AVGs u-AVGs.

Definition 2.16 (u-AVG)**.** A u-AVG is a 5-tuple $G = \langle\langle \mathsf{Attr}, \mathsf{Atms}, \rho\rangle, \mathsf{AttrPerc}, T, P, \{S\}\rangle$.

Definition 2.17 (Derivability)**.** If $A \overset{*}{\Longrightarrow}_3 \omega$ and $\omega' \in permute(\omega)$, then $A \overset{*}{\Longrightarrow}_4 \omega'$.

Remark 2.18. (1) means that no Chomsky normal form can be obtained for s-AVG or u-AVG.

3 Bounds on chart size

3.1 Type 2 grammars

Lemma 3.1 (Size of derivation structure)**.** *Say $D = \langle V, e\rangle$ is a derivation structure for ω, G where G is a type 2 grammar in Chomsky normal form. It now holds that $|V| \leq (3n - 1)$.*

Proof. Since G is in Chomsky normal form, there are only two kinds of production rules: Any derivation of ω of length n needs $n-1$ binary applications, and n unary ones, i.e. of non-branching rules. There are n many terminals. Consequently, the derivation structure is at most $3n - 1$. □

Definition 3.2 (ω-grammar)**.** Say you have a type 2 grammar in Chomsky normal form $G = \langle N, T, P, \{S\}\rangle$ and some string $\omega_1 \ldots \omega_n$. Construct $G_\omega = \langle N_\omega, T_\omega, P_\omega, \{{}_1S_n\}\rangle$ such that

$$T_\omega \quad = \quad \{\omega_1, \ldots, \omega_n\}$$

and, recursively

(a) $(\omega_i \in T_\omega$ and $A \rightarrow \omega_i \in P) \Rightarrow ({}_iA_i \in N_\omega$ and ${}_iA_i \rightarrow \omega_i \in P_\omega)$

(b) $({}_iB_{j,j+1}\, C_k \in N_\omega$ and $A \rightarrow BC) \Rightarrow ({}_iA_k \in N_\omega \wedge {}_iA_k \rightarrow {}_iB_{jj+1}C_k \in P_\omega)$

Example 3.3. Consider $aabb$-grammar of the Chomsky normal form grammar in Example 2.5. First $T_\omega = \{a_1, a_2, b_3, b_4\}$. By (a), the terminal rules are constructed: ${}_1A_1 \rightarrow a_1$, ${}_2A_2 \rightarrow a_2$, ${}_3B_3 \rightarrow b_3$, and ${}_4B_4 \rightarrow b_4$. Nonterminal binary rules can now be constructed:

$$\begin{array}{cc} {}_2S_3 \rightarrow {}_2A_{23}B_3 & {}_2X_3 \rightarrow {}_2A_{23}B_3 \\ {}_2T_4 \rightarrow {}_2X_{34}B_4 & {}_1X_4 \rightarrow {}_1A_{12}T_4 \\ {}_1S_4 \rightarrow {}_1A_{12}T_4 & \end{array}$$

$N_{aabb} = \{{}_1A_1, {}_2A_2, {}_3B_3, {}_4B_4, {}_2S_3, {}_2X_3, {}_2T_4, {}_1X_4, {}_1S_4\}$.

Our construction of G_ω gives us two sets of possible interest, N_ω and P_ω. It is easy to see that

$$|N_\omega| \quad \leq \quad |N| \times (\tfrac{n^2+n}{2})$$

where $|\omega| = n$. In our example above this amounts to $|N_{aabb}| = 9 \leq 4 \times \frac{4\times 5}{2} = 40$.

The chart size is bounded by $|P_\omega| + n$.

Lemma 3.4 (Chart size)**.** *Say G is a type 2 grammar in Chomsky normal and $\omega \in T^*$. It now holds that $|C_{G,\omega}| \leq (|N_\omega| \times n \times |N|^2) + (|N| \times n) + n$.*

Proof. Each ω-nonterminal ($|N_\omega|$ many) has at most two daughters, and $n \times |N|$ nonterminals are non-branching. Since there are at most n ways to split up the span of a branching terminal in two, and at most $|N|^2$ variable combinations for the two daughters, $((|N_\omega|) \times n \times |N|^2 + (n \times |N|))$ is clearly an upper bound on $|P_\omega|$. In fact, the result is suboptimal, since the ${}_iX_i$-nonterminals count twice. □

The number of trees with n leafs is C_{n-1} (the Catalan number).

3.2 Totally unordered type 2 grammars

Lemma 3.5 (Size of derivation structure)**.** *Say $D = \langle V, e\rangle$ is a derivation structure for ω, G where G is a totally unordered type 2 grammar in Chomsky normal form. It now holds that $|V| \leq (3n - 1)$.*

Proof. Similar to the proof of Lemma 3.1. □

Definition 3.6 (ω-grammar)**.** Say you have a totally unordered type 2 grammar in Chomsky normal form $G = \langle N, T, P, \{S\}\rangle$ and some string $\omega_1 \ldots \omega_n$. Construct $G_\omega = \langle N_\omega, T_\omega, P_\omega, \{S\}\rangle$ such that

$$T_\omega \quad = \quad \{\omega_1, \ldots, \omega_n\}$$

and, recursively

(a) $(\omega_i \in T_\omega$ and $A \rightarrow \omega_i \in P) \Rightarrow (A_{\{i\}} \in N_\omega$ and $A_{\{i\}} \rightarrow \omega_i \in P_\omega)$

(b) $(B_\Sigma, C_{\Sigma'} \in N_\omega$ and $\Sigma \cap \Sigma' = \emptyset$ and $A \rightarrow BC) \Rightarrow (A_{\Sigma\cup\Sigma'} \in N_\omega \wedge A_{\Sigma\cup\Sigma'} \rightarrow B_\Sigma C_{\Sigma'} \in P_\omega)$

Lemma 3.7 (Chart size, upper bound)**.** *Say G is a totally unordered type 2 grammar in Chomsky normal and $\omega \in T^*$. It now holds that $|C_{G,\omega}| \leq |N_\omega|^2 \times n^2 \times |N|^2) + (n \times |N|) + n$.*

Proof. There are n^2 ways to split up a sequence in two discontinuous parts. □

.

Lemma 3.8 (Chart size, lower bound)**.** *Say G is a totally unordered type 2 grammar in Chomsky normal and $\omega \in T^*$. It now holds that $|C_{G,\omega}| \notin \mathcal{O}(n^k)$, i.e. chart size is exponential.*

Proof. It is easy to see this. You only need to consider the upper bound on $|N_\omega|$ in the totally unordered case:

$$|N_\omega| \quad \leq \quad |N| \times 2^n$$

□

3.3 s-AVGs

Lemma 3.9 (Size of derivation structure)**.** *Say $D = \langle V, e\rangle$ is a derivation structure for ω, G where G is an s-AVG. It now holds that $|V| \leq 3n-1 \times (|\mathsf{Attr}|+1)$.*

Definition 3.10 (ω-grammar)**.** Say you have an s-AVG $G = \langle\langle \mathsf{Attr}, \mathsf{Atms}, \rho\rangle, \mathsf{AttrPerc}, T, P, \{S\}\rangle$ and some string $\omega_1 \ldots \omega_n$. Construct $G_\omega = \langle\langle \mathsf{Attr}, \mathsf{Atms}_\omega, \rho\rangle, \mathsf{AttrPerc}, T_\omega, P_\omega, \{{}_1S_n\}\rangle$ such that

$$T_\omega \quad = \quad \{\omega_1, \ldots, \omega_n\}$$

and, recursively

(a) $([\omega]_i \in T_\omega$ and $\alpha \rightarrow [\omega]_i \in P) \Rightarrow ({}_i[\alpha]_i \in N_\omega$ and ${}_i[\alpha]_i \rightarrow [\omega]_i \in P_\omega)$

(b) $({}_i[\alpha_1]_{j,j+1}\,[\alpha_2]_k, \ldots, {}_{m-1}\,[\alpha_n]_m \in N_\omega$ and $[\alpha_0] \rightarrow [\alpha'_1][\alpha'_2]\ldots[\alpha'_n] \in P$ and $\forall 1 \leq i \leq n.[\alpha_i] \sqsubseteq [\alpha'_i] \vee [\alpha'_i] \sqsubseteq [\alpha_i]) \Rightarrow ({}_i[\alpha_0]_m \in N_\omega \wedge {}_i[\alpha_0]_m \rightarrow_i {}_i[\alpha_1]_{j,j+1}\,[\alpha_2]_k, \ldots, {}_{m-1}\,[\alpha_n]_m \in P_\omega)$

I introduce square brackets to enhance readability, i.e. to separate daughter tags from positions. Positions are outside brackets.

We no longer have a set $|N|$ to measure chart size. The set of possible category structures is $\mathsf{Atms}^{\mathsf{Attr}}$. However, by inspection of our definition of ω, a tighter bound is obtained:

$$|N_\omega| \quad \leq \quad |P| \times (\tfrac{n^2+n}{2})$$

Unfortunately, no such bound can be placed on $|P_\omega|$. The reason is, of course, that productions ${}_i[\alpha_0]_m \rightarrow {}_i[\alpha_1]\ldots[\alpha_n]_m$, and not ${}_i[\alpha_0]_m \rightarrow {}_i[\alpha'_1]\ldots[\alpha'_n]_m$ are recorded in Definition 3.10.

Lemma 3.11 (Chart size)**.** *Say G is an s-AVG and $\omega \in T^*$. It now holds that $|C_{G,\omega}| \leq (|N_\omega| \times n \times |\mathsf{Atms}|^{|\mathsf{Attr}|\times k}) + (|P| \times n) + n$, if G do not contain m-ary rules such that $m > k$.*

Proof. Compare the situation to Lemma 3.4. $|\mathsf{Atms}|^{|\mathsf{Attr}|\times k}$ is the number of combinations of daughter categories in k-ary productions. □

3.4 Totally unordered s-AVGs

The upper bound on derivation structions in the totally unordered case is the same as for s-AVGs. ω-grammars for u-AVG are built analogously to ω-grammars for totally unordered type 2 grammars. It is easy to see that:

$$|N_\omega| \quad \leq \quad |P| \times 2^n$$

It now holds:

Lemma 3.12 (Chart size)**.** *Say G is an u-AVG and $\omega \in T^*$. It now holds that $|C_{G,\omega}| \leq (|N_\omega| \times n \times |\mathsf{Atms}|^{|\mathsf{Attr}|\times k}) + (|P| \times n) + n$, if G do not contain m-ary rules such that $m > k$.*

In sum,

Theorem 3.13. *Totally unordered 2 grammars, s-AVGs and u-AVGs have worst case exponential charts.*

This leads us to consider complexity and generative capacity.

3.5 Complexity and generative capacity

Consider the universal recognition problem:

Definition 3.14 (Universal recognition)**.** Universal recognition is the decision problem:

INSTANCE:	A grammar $\mathcal{G}$ and a string ω.
QUESTION:	Is ω in the language denoted by $\mathcal{G}$?

Lemma 3.15 ((Barton, 1985))**.** *The universal recognition problem for totally unordered type 2 grammars is NP-complete.*

Proof. The vertex cover problem involves finding the smallest set V' of vertices in a graph $G = \langle V, E\rangle$ such that every edge has at least one endpoint in the set. Formally, $V' \subseteq V : \forall\{a, b\} \in E : a \in V' \vee b \in V'$. The problem is thus an optimization problem, formulated as a decision problem:

INSTANCE:	A graph G and a positive integer k.
QUESTION:	Is there a vertex cover of size k or less for G?

Say $k = 2, V = \{a, b, c, d\}, E = \{(a, c), (b, c), (b, d), (c, d)\}$. One way to obtain a vertex cover is to go through the edges and underline one endpoint of each edge. If you can do that and only underline two vertex symbols, a vertex cover has been found. Since $|V| = 4$, this is equivalent to leaving two vertex symbols untouched. Consequently, the vertex cover problem for this specific instance is encoded by the totally unordered type 2 grammar, where δ is a bookkeeping dummy symbol:

$$
\begin{array}{rcl}
S & \rightarrow & \rho_1\rho_2\rho_3\rho_4 uu\delta\delta\delta\delta \\
\rho_1 & \rightarrow & a|c \\
\rho_2 & \rightarrow & b|c \\
\rho_3 & \rightarrow & b|d \\
\rho_4 & \rightarrow & c|d \\
u & \rightarrow & aaaa|bbbb|cccc|dddd \\
\delta & \rightarrow & a|b|c|d
\end{array}
$$

ρ_i captures the ith edge in E. The input string $\omega = aaaabbbbccccdddd$. Generally, the first production has as many ρ_i as there are edges in the graph, $|V|-k$ many u's and $|E|\times|V|-|E|-|E|\times(|V|-k)$ many δ's, i.e. the length of the string minus the number of edges and the extension of $|V| - k$ many u's. The ρ_i productions are simple, u extends into $|E|$ many a's or b's or so on, and δ extends into all possible vertices. Since the grammar and input string can be constructed in polynomial time from an underlying vertex cover problem $\langle k, V, E\rangle$, universal recognition of UCFG must be at least as hard as solving the vertex cover problem. Since the vertex cover problem is NP-complete (Garey and Johnson, 1979), the universal recognition problem for totally unordered type 2 grammars is accordingly NP-hard. It is easy to see that it is also in NP. Simply guess a derivation, polynomial in size by Lemma 3.5, and evaluate it in polynomial time. □

Lemma 3.16. *The universal recognition problem for s-AVGs is NP-complete.*

Proof. The 3SAT problem is a variant of the satisfiability problem of propositional logic for conjunctions of clauses of three literals, e.g. $p \vee p \vee p \wedge \neg p \vee \neg p \vee \neg p$ is *not* satisfiable in any model. Its complexity is the same as its older sister's: It is NP-complete. It is relatively easy to code this problem up in s-AVG. The details are left for the reader. *Hint:* Introduce agreement features for truth assignments and build ternary phrases that ensure at least one propositional variable in the original problem is true. Since AttrPerc must percolate by (1), you need four rules for each propositional variable (true and false for with and without negation). It follows that the universal recognition problem for s-AVGs is NP-hard. It is easy to see that it is also in NP. Simply guess a derivation, polynomial in size by Lemma 3.9, and evaluate it in polynomial time. □

Lemma 3.17. *The universal recognition problem for u-AVGs is NP-complete.*

Proof. Similar to the proof of Lemma 3.16. Extra features can be used for clause bounds. □

Remark 3.18. It is cheap to add linear precedence constraints to totally unordered type 2 grammars and u-AVGs, e.g. to ensure that all verbs precede nouns. Such constraints can be resolved in time $\mathcal{O}(n^2)$ on even the most naïve set-up.

If linear precedence constraints are added, it holds that

Lemma 3.19. *The totally unordered type 2 languages and the totally unordered simple attribute-value languages both are not included in the type 2 languages.*

Proof. Both the totally unordered type 2 languages and the totally unordered simple attribute-value languages include $\{a^m b^n c^m d^n\}$. The simplest way to encode it is to let some rule $S \rightarrow abScd|abcd$ interact with some precedence rule that requires all a's to precede all b's, and so on. Similarly, with s-AVSs. It is just as easy to code up the MIX language, for instance. □

4 k-ambiguity

Our strategy to obtain polynomial charts in the totally unordered cases is to restrict ambiguity. A rigid lexicon is first imposed. In a rigid lexicon every phonological string is associated with at most one lexical entry.

Remark 4.1. Rigidity is a strong constraint in the absence of inheritance. Inheritance provides an alternative to lexical ambiguity, namely underspecification. Such use of inheritance seems necessary for realistic applications of k-ambiguous grammars. Rigidity needs only to apply to open class items. There seems to be some evidence from cognitive neuropsychology that people actually underspecify open class items wrt. morphological features, valence and even syntactic category.

The next step is to restrict ambiguity in parsing.

Definition 4.2. A sign is horizontally k-ambiguous if it only combines with k signs in a sentence. A grammar is horizontally k-ambiguous if all signs are k-ambiguous. A grammar is vertically k-ambiguous if signs are combined unambiguously after k steps.

It is important to remember that our unordered grammars allow signs to combine non-locally. The notion of k-ambiguity can be illustrated by an example from Icelandic:

Example 4.3. Icelandic has nominative objects. Consider, for instance:

(5) *Hún spurði hvort sá*
she asked whether the.NOM
grunaði væri örugglega
suspected.NOM was.3SG.SUBJ surely
þú.
you.SG.NOM
'She asked whether the suspect surely was you.'

In addition, both SVO and OVS constructions occur. So in many cases, a verb that seeks to combine with an object has more than one candidate for doing so, even in sentences with only three constituents:

The V constituent is said to be horizontally 2-ambiguous in this case.

For simplicity, the notion of the order of an s-AVS is introduced:

Definition 4.4. An s-AVS α is said to be of order l iff $|\texttt{DOM}(\alpha)| = l$. If all s-AVSs in a grammar G are of order l, G is itself said to be of order 1.

Lemma 4.5. *Type 2 grammars are equivalent to s-AVGs of order 1. Totally unordered type 2 languages are equivalent to u-AVGs of order 1.*

Proof. Trivial. □

Say s-AVSs are of order 1, and vertical ambiguity 1 (i.e. horizontal ambiguity k). We then have:

$$|C_{G,\omega}| \quad \leq \quad (\tfrac{n^2-n}{2} + \sum_{1<i}^{i<n}(kn(n-i))) + n$$

First all initial combinations $\frac{n^2-n}{2}$ are checked. At this point, there can be at most kn candidate models. For each candidate model, the next set of combinations is checked. Since vertical ambiguity is 1, the set of candidate models remains at most kn.

If we fix vertical ambiguity to k (i.e. horizontal ambiguity n):

$$|C_{G,\omega}| \quad \leq \quad (\tfrac{n^2-n}{2} + \sum_{1<i}^{i<n}(n^k(n-i))) + n$$

which is in $\mathcal{O}(n^{k+2})$. Since the order of s-AVSs is bound by $|\texttt{Attr}|$, it holds that:

Theorem 4.6. *k-ambiguous totally unordered 2 grammars, k-ambiguous s-AVGs and k-ambiguous u-AVGs have polynomial charts.*

Proof. See above. The result for s-AVGs is subsumed by the result for u-AVGs. □

Remark 4.7. For unordered type 2 grammars, and possibly for totally unordered ones too, it is an alternative to say that all totally unordered productions in a chart have a yield of at most k. This gives a bound on chart size:

$$\sum_{i<k}^{0\leq i}(|N| \times (n-i) \times \sum_{j<(k-i)}^{0\leq j}(|N|^{k-j}) \times (k-i) \times (n-j)) + \sum_{i<n}^{k<i}(|N|^3 \times (n-i))$$

This fragment no longer generates the MIX language. Such a constraint is obviously not enough for u-AVG, since s-AVG is NP-complete. A third possibility is to restrict the arity of productions.

5 Conclusions and related work

In last year's conference, Søgaard and Haugereid (2006) presented, in a rather informal fashion, a restrictive typed attribute-value structure grammar formalism $\mathcal{T}_f$ for free word order phenomena, equipped with a polynomial parsing algorithm. In $\mathcal{T}_f$, horizontal $k = 1$. The purpose of their paper was mostly philosophical, i.e. in favor of underspecification rather than ambiguity, but many details were left unclear. In a sense, this paper provides a formal basis for some of the claims made in that paper. In particular, types are easily added to s-AVG and u-AVG, and more flexible attribute-value structures can be employed (as long as they are at most polynomial in the size of strings). Unlike $\mathcal{T}_f$, k-ambiguous grammars also admit fixed ambiguity.

Other researchers have tried to single out tractable attribute-value grammars:

Seki et al. (1993) operate in the context of LFG. For a start, they restrict the expressive power of LFG by restricting the syntax of LFG-style functional schemas to:

$$(\uparrow \texttt{attr} = \texttt{val}) \text{ or } (\uparrow \texttt{attr} = \downarrow)$$

Call this fragment non-deterministic copying LFG (nc-LFG). They then proceed to define two tractable fragments of nc-LFG:

Definition 5.1. An nc-LFG is called a dc-LFG (deterministic ...) if each pair of rules $r_1 : A \rightarrow \alpha_1$ and $r_2 : A \rightarrow \alpha_2$ whose left-hand sides are the same is inconsistent in the sense that there exists no f-structure that locally satisfies both of the functional schemata of r_1 and r_2.

Definition 5.2. An nc-LFG is called a fc-LFG (finite ...) if it contains only a finite number of so-called "subphrase nonterminal" (SPN) multisets, i.e. a multiset of nonterminals N such that there exists consistent productions $A_1 \rightarrow \alpha_1 \dots A_n \dots \alpha_n$ and an attribute `attr` such that $N = \{\alpha_i \in \{\alpha_1 \dots \alpha_n\} | (\uparrow \texttt{attr} = \downarrow) \text{ is the FS of } \alpha_i\}$.

A nice example of an nc-LFG that is not an fc-LFG is mentioned in (Seki et al., 1993):

Example 5.3. Let G be an nc-LFG where $N = \{S\}$, $T = \{a\}$, $\texttt{Lbls} = \{\texttt{log}\}$, e the only value, and productions are:

$$\begin{array}{lcll} S & \rightarrow & S & S \\ & & (\uparrow \texttt{log} = \downarrow) & (\uparrow \texttt{log} = \downarrow) \\ S & \rightarrow & a & \\ (\uparrow \texttt{log} = e) & & & \end{array}$$

G is *not* an fc-LFG, since the SPN multisets in G include

$$\{\{S\}\}, \{\{S,S\}\}, \{\{S,S,S,S\}\}, \dots.$$

Both fragments are tractable, and the weak generative capacity of dc-LFG is equivalent to that of finite-state translation systems, while the weak generative capacity of fc-LFG is equivalent to that of linear indexed grammars. It follows that fc-LFG is also equivalent to one-reentrant attribute-value grammar (Feinstein and Wintner, 2006).

Keller and Weir (1995) go beyond linear indexed grammars on their way toward attribute-value grammar. The first step on this path is to replace the stacks of indeces in linear indexed grammars with trees. Tractability is ensured by the requirement that subtrees of any mother that are passed to daughters that share subtrees with one another must appear as siblings in the mother's tree. The following such grammar generates $\{a^n b^n c^n\}$:

$$\begin{array}{lcl} S_1[\sigma_0] & \rightarrow & A[x]S_2[\sigma(x,x)] \\ S_2[\sigma(x,y)] & \rightarrow & B[x]S_3[y] \\ S_3[x] & \rightarrow & C[x] \\ A[\sigma_2(x)] & \rightarrow & aA[x] \\ B[\sigma_2(x)] & \rightarrow & bB[x] \\ C[\sigma_2(x)] & \rightarrow & cC[x] \\ A[\sigma_1] & \rightarrow & a \\ B[\sigma_1] & \rightarrow & b \\ C[\sigma_1] & \rightarrow & c \end{array}$$

In a sense, this is much like s-AVG, except that reentrancies replace (1) and roots cannot be reentered. Keller and Weir argue this is no problem if

the entire structure is seen as the derivational output, rather than just the AVS of the mother. In addition, reentrancy is interpreted intensionally in their set-up, rather than extensionally. This is similar to ours.

Both formalisms are stronger than k-ambiguous u-AVG in some respects. This is easy to see. Both nc-LFG and Keller and Weir's richer fragment of attribute-value grammar are superfinite, i.e. they generate all finite languages. k-ambiguous u-AVG doesn't. It holds that:

Lemma 5.4. *The k-ambiguous u-AVG languages do not include (all of) the regular languages.*

The proof is omitted, but consider the simpler proof of:

Lemma 5.5. *The 1-ambiguous u-AVG languages do not include (all of) the regular languages.*

Proof. Consider the language

$$\underset{\text{but not } j}{a\{b|\ldots|n\}} \quad \cup \quad \underset{\text{but not } i}{p\{b|\ldots|n\}} \quad \cup \quad \{b|\ldots|n\}$$

in which ab, ai, pj, b are strings, while aj, pi, bb are not. This language is regular, but cannot be generated by a 1-ambiguous u-AVG. □

It should be relatively easy to see how this generalizes to k-ambiguous u-AVG.

In sum, it was shown that the exponential worst case complexity of totally unordered charts is drammaticaly reduced by the k-ambiguity constraint. In particular k-ambiguous charts are in $\mathcal{O}(n^{k+2})$. Since subsumption is linear time solvable, the recognition problem for k-ambiguous u-AVGs is also solvable in polynomial time. Efficient algorithms and their complexity are the topic of future publications. k-ambiguous u-AVG differs in significant ways from other polynomial time attribute-value grammars. In particular, k-ambiguous u-AVG was designed for analyses of discontinuous constituency. It provides the formal machinery needed for "crossed branches" analyses. In addition, k-ambiguous u-AVG is not superfinite. It is conjectured – also by one of the reviewers – that this has interesting consequences for learnability.

References

Edward Barton. 1985. The computational difficulty of ID/LP parsing. In *Proceedings of the 23th Annual Meeting of the Association for Computational Linguistics*, pages 76–81, Chicago, Illinois.

David Dowty. 1995. Toward a minimalist theory of syntactic structure. In Harry Bunt and Arthur van Horck, editors, *Discontinuous constituency*, pages 11–62. de Gruyter, Berlin, Germany.

Daniel Feinstein and Shuly Wintner. 2006. Highly constrained unification grammars. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1089–1096, Sydney, Australia.

Michael Garey and David Johnson. 1979. *Computers and intractability*. W. H. Freeman & Co., New York, New York.

Bill Keller and David Weir. 1995. A tractable extension of linear indexed grammars. In *Proceedings of the 7th European Chapter of the Association for Computational Linguistics*, pages 75–82, Dublin, Ireland.

Hiroyuki Seki, Ryuichi Nakanishi, Yuichi Kaji, Sachiko Ando, and Tadao Kasami. 1993. Parallel multiple context-free grammars, finite-state translation systems, and polynomial-time recognizable subclasses of lexical-functional grammars. In *Proceedings of the 31st Annual Meeting on the Association for Computational Linguistics*, pages 130–139, Columbus, Ohio.

Anders Søgaard and Petter Haugereid. 2006. Functionality in grammar design. In Stefan Werner, editor, *Proceedings of the 15th Nordic Conference of Computational Linguistics*, pages 180–189, Joensuu, Finland.

Thomas Sudkamp. 2005. *Languages and machines*. Pearson, Boston, Massachusetts, 3rd edition.