ON THE WEAK GENERATIVE CAPACITY OF INFINITE GRAMMARS

D. Terence Langendoen

The Graduate Center

1. Linguistic motivation for infinite grammars. It has been known for some time that adequate grammars of human languages must contain infinitely many rules. For example, the base components of such grammars must contain infinitely many type-2 (context-free phrase-structure) simple rewrite rules[1] to express the fact that sentences may have any number of conjoined sentences as immediate constituents (Chomsky and Schützenberger, 1963: 133; Chomsky, 1965: 224). To illustrate, consider the infinite set of type-2 rules given in 1.

(1) S → S C S

S → S S C S

S → S S S C S

...

S → N V

N → {Alice, Bob, Carol, Dave}

V → {jumped, kicked, lunged, moved}

C → {and, or}

A grammar containing this set of rules, unlike any finite type-2 grammar, is capable of generating sentences with n conjuncts, $2 \leq n < \infty$, with flat constituent structures of the type $[_S\ [_S\ X_1\ ]_S \cdots [_S\ X_n\ ]_S\ ]_S$, which is one of the structure-types that adequate grammars for human languages must be able to provide for such sentences.

2. Rule schemata and hypergrammars. One conventionally represents infinite grammars like 1 in finite fashion by replacing all but finitely many of the rules of such grammars by finite rule schemata. Thus 1 could be represented in schematic form as in 2.

(2) $S \rightarrow S^n\ C\ S \qquad (n > 0)$

$S \rightarrow N\ V$

$N \rightarrow$ {Alice, Bob, Carol, Dave}

$V \rightarrow$ {jumped, kicked, lunged, moved}

$C \rightarrow$ {and, or}

However, from the representation of infinite grammars in finite schematic form, it cannot be immediately determined what the weak generative capacity of a given class of infinite grammars is. Thus, consider the class of all infinite type-2 grammars, that is, the class of all infinite grammars all of whose rules are of type-2. While it is the case that some such grammars do generate type-2 languages, not all of them do.[2] For example, the infinite type-2 grammar in 3, and schematized in 4, generates a language $L = \{a^n b^n a^n \mid n > 0\}$ that cannot be generated by any finite type-2 grammar.

(3) $S \rightarrow A\ B\ A$

$S \rightarrow A\ A\ B\ B\ A\ A$

$S \rightarrow A\ A\ A\ B\ B\ B\ A\ A\ A$

...

$A \rightarrow a$

$B \rightarrow b$

(4) $S \rightarrow A^n B^n A^n \quad (n > 0)$

$A \rightarrow a$

$B \rightarrow b$

Obviously, to determine the weak generative capacity of classes of infinite grammars, one must take into consideration more than just the types of rules contained in those grammars. One must also consider the types of devices that are used to construct the rules of those grammars, which the schemata indirectly represent. To determine what those devices are, one cannot simply consider rule schemata as abbreviatory conventions, analogous to the finite abbreviatory conventions represented by the use of curly braces and parentheses. Rather they must be thought of as standing for the rules of another grammar, which enumerate all but finitely many of the rules of the infinite grammars that they are part of. For terminological convenience, let us call a grammar that generates all and only all of the rules of another grammar a <u>hypergrammar</u>.[3] In what follows we will be concerned only with finite simple rewrite hypergrammars that generate infinite simple rewrite grammars.

To illustrate the notion of hypergrammars, we give in 5 a hypergrammar that generates the infinite grammar in 1.[4]

(5) $\Sigma \Rightarrow \{\Delta \ \Theta \ \Lambda \ \Xi \ \Pi\}$

$\Delta \Rightarrow \Phi \ S \ C \ S$

$\Phi \Rightarrow \Phi \ S$

$\Phi \Rightarrow S \rightarrow$

$\Theta \Rightarrow S \rightarrow N \ V$

Λ => N → {Alice, Bob, Carol, Dave}

Ξ => V → {jumped, kicked, lunged, moved}

Π => C → {and, or}

The hypergrammar in 5 is, obviously, a type-3 grammar. The fact that there is a type-3 hypergrammar that generates the grammar in 1 must be the reason why that grammar does not generate a language that is outside the generative capacity of finite type-2 grammars. On the other hand, any hypergrammar that generates the grammar in 3 cannot be a type-2 grammar, and it must be this fact that explains why the grammar in 3 generates a language that is outside the generative capacity of finite type-2 grammars.

3. The weak generative capacity of classes of infinite grammars. There is, in fact, a straightforward relation between the weak generative capacity of a class of infinite grammars, on the one hand, and the weak generative capacity of the class of hypergrammars that generate them and the type of rules contained in them, on the other. This relation is expressed by the following theorem.

Theorem. Let T be the class of hypergrammars of type-m ($0 \leq m \leq 3$) that generate the class Γ of infinite grammars all of whose rules are at most of type-n ($0 \leq n \leq 3$). Then the weak generative capacity of Γ is that of type-p grammars, where $p = \min(m, n)$.

Proof. Clearly, $p \neq m$, since, given any language generated by a type-m grammar, one can construct a type-m hypergrammar that generates an infinite type-3 grammar that simply lists all and only

all of the sentences of that language. Clearly also, $p \not> n$, since all the type-n languages can be generated by a subset of the grammars in Γ, namely those in which only finitely many of the rules are used in the derivation of sentences. Hence $p \leq m$ and $p \leq n$. To show that if $m \leq n$, then $p = m$, and that if $n \leq m$, then $p = n$, we must consider a number of cases, as follows. For simplicity, we assume also that no finite abbreviatory devices are used in the statement of the rules of each grammar G in Γ.

Case 1: $m = 0$, $0 \leq n \leq 3$; or $n = 0$, $0 \leq m \leq 3$. We show that $p = 0$ by showing that the sentences of any language generated by an infinite grammar G in Γ can be recursively enumerated. Clearly the rules of an infinite grammar G of type-n ($0 \leq n \leq 3$) can be recursively enumerated. We enumerate the derivations of sentences generated by G as follows. The first line of a derivation of a sentence generated by G is the initial symbol S. Construct the second line by choosing some rule of G (if any) whose left-hand side consists of that symbol; the right-hand side of that rule is the second line of that derivation. Now let $\tau = \phi\chi\omega$ be the k-th line of the derivation under construction, and let n be the length of τ. Then construct the k+1st line $\tau' = \phi\psi\omega$ of the derivation by selecting any rule of the form $\chi \rightarrow \psi$. If such a rule exists, it will be possible to find it after a finite search, since its left-hand side is of length n or less. Continue in this fashion until either the derivation is terminated or it cannot be continued.[5] By this procedure every derivation of every sentence of L(G) can be enumerated.

Case 2: $m = 1$, $1 \leq n \leq 3$; or $n = 1$, $1 \leq m \leq 3$. We show that $p = 1$ by showing that the amount of computation space required for generating a sentence σ of $L(G)$ is linearly bounded by the length n of σ. If all of the rules of G necessary for the derivation of σ are available to the device that computes that derivation, the amount of computing space it requires to generate σ is linearly bounded by the length of σ, since the rules of G are of type-n ($1 \leq n \leq 3$). If some rule of G needed for the derivation of σ is not immediately available to that device, we assume that it is equipped to search for that rule among the rules of G, and upon finding it to apply it. Since the length of the longest rule of G that could be applicable in the derivation of σ is $2n+1$ (the left- and right-hand sides of that rule may be at most of length n, and the rewrite symbol $\rightarrow$ is of length 1), and since the hypergrammar H in T that generates those rules if of type-m ($1 \leq m \leq 3$), the amount of computing space required to enumerate the potentially applicable rules of G is linearly bounded by the length of σ. The total amount of computing space required for the derivation of σ in $L(G)$ (both for determining what rules are applicable and for carrying them out) is linearly bounded, therefore, by the length of σ.

Case 3: $m = 2$, $2 \leq n \leq 3$. We show that $p = 2$ by using the pumping lemma for type-2 grammars to replace each G in Γ by a weakly equivalent finite type-2 grammar. Let H in T be such that $L(H) = G$. Since H is a type-2 hypergrammar that generates an infinite language, by the pumping lemma for type-2 grammars, there is at least one non-terminal symbol Δ in H such that there is a sentence z_Δ in $L(H)$ of

sufficient length, and strings u, v, w, x, y (w and not both v and x nonnull) over the terminal vocabulary of H, such that z = uvwxy, and $\Sigma \overset{*}{\underset{H}{\Longrightarrow}} u\Delta y$, $\Delta \overset{*}{\underset{H}{\Longrightarrow}} v\Delta x$, and $\Delta \overset{*}{\underset{H}{\Longrightarrow}} w$, and for all $k \geq 0$, $uv^k wx^k y$ is in L(H). Let us call the latter sentences members of the family of z_Δ. All but a finite number of sentences of L(H) must be members of one of a finite number of such families.

Now, the sentences of L(H) are the rules of G. Suppose v in some z_Δ is nonnull. Then u = A→u' for some A in the nonterminal vocabulary of G and some string u' in the vocabulary of G (otherwise the rewrite symbol → of G would appear in v and hence more than once in some rule of G, which is impossible). Thus the members of the family of z_Δ are of the form $A \rightarrow u'v^k wx^k y$. Now let R be a nonterminal symbol that does not appear in G. Form a new grammar from G by deleting all of the infinitely many rules of the form $A \rightarrow u'v^k wx^k y$ from G, and adding in their place the following three rules:

(i) $A \rightarrow u' \; R \; y$

(ii) $R \rightarrow v \; R \; x$

(iii) $R \rightarrow w$

Do the same for all of the finitely many other families in L(H) for which v is nonnull. Similarly, suppose v in some z_Δ is null. Then x cannot be null, and uw = A→t for some nonterminal symbol A in G and some string t over the vocabulary of G. Form a new grammar from G by eliminating the infinitely many rules of the form $A \rightarrow tx^k y$, and adding the following three rules in their place:

(iv) $A \rightarrow R\ y$

(v) $R \rightarrow R\ x$

(vi) $R \rightarrow t$

Do the same for all of the finitely many other families in L(H) for which v is null. The grammar G' that results from all of these substitutions is a finite type-2 grammar that is weakly equivalent to G.

Case 4: $m = 3$, $2 \leq n \leq 3$. We show that $p = n$ (i.e., that $p = 2$ if $n = 2$ and $p = 3$ if $n = 3$) by using the pumping lemma for type-3 grammars to replace each G in Γ by a weakly equivalent finite grammar of the appropriate type.

Let H be in T such that L(H) = G. Since H is a type-3 hypergrammar that generates an infinite language, by the pumping lemma for type-3 grammars, there is at least one nonterminal symbol Δ in H such that there is a sentence z_Δ of sufficient length in L(H), and strings u, v, w (v and either u or w nonull) such that $z_\Delta = uvw$ and (if H is left-linear) $\Sigma \overset{*}{\underset{H}{\Rightarrow}} \Delta w$, $\Delta \overset{*}{\underset{H}{\Rightarrow}} \Delta v$, and $\Delta \overset{*}{\underset{H}{\Rightarrow}} u$, and for all $k \geq 0$, $uv^k w$ is in L(H). As in case 3, call the latter sentences members of the family of z_Δ. All but a finite number of sentences of L(H) must be members of one of a finite number of such families; similarly if H is right-linear.

Since the sentences of L(H) are the rules of G, it must be the case as in case 3 that $u = A \rightarrow u'$ for some A in the nonterminal vocabulary of G and some string u' in the vocabulary of G. Thus members of the family of z_Δ are all of the form $A \rightarrow u'v^k w$. Now let R be a nonterminal symbol that does not appear in G. If G contains type-2

rules, form a new grammar from G by deleting all of the infinitely many rules of the form $A \to u'v^k w$ and adding in their place the following three rules:

(vii) $A \to R\ w$

(viii) $R \to R\ v$

(ix) $R \to u'$

Do the same for all of the remaining finitely many families in L(H). The resulting finite grammar G' is a type-2 grammar that is weakly equivalent to G.

If the rules of G are type-3, then they are all either left-linear or right-linear. Suppose they are left-linear. Then v, w can consist only of terminal symbols in the vocabulary of G, and u' must be of the form x or Bx for some nonterminal symbol B of G and some string x of terminal symbols of G. Replace each of the infinitely many rules of G of the form $A \to u'v^k w$ by the rules vii, viii, and ix above. Do the same for all of the remaining families in L(H). The resulting grammar G' is a finite left-linear type-3 grammar G' that is weakly equivalent to G.

Similarly, if the rules of G are right-linear, then u' and v can consist only of terminal symbols of the vocabulary of G and w must be of the form x or xB for some nonterminal symbol B of G and some string x of terminal symbols of G. Replace each of the infinitely many rules of G of the form $A \to u'v^k w$ by the following three rules.

(x) $A \to u'\ R$

(xi) $R \to v\ R$

(xii) $R \to w$

Do the same for each of the remaining families in L(H). The resulting grammar G' is a finite right-linear type-3 grammar G' that is weakly equivalent to G.

This completes the proof of the theorem.

The theorem can be expressed graphically by the matrix given in Figure 1, in which the numbers in the cells of the matrix give the weak generative capacity of each class of infinite grammars.

Type of rule in infinite grammar	Type of hypergrammar 0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

Figure 1. Weak generative capacity of classes of infinite grammars as a function of the type of rules contained in them and the type of hypergrammar required to generate them.

As a corollary to this theorem, we note that the categorial subcomponent of the base component of a standard generative-transformational grammar (where either the lexical categories or certain designated lexical items can be thought of as the terminal symbols of that subcomponent) is an infinite type-2 grammar that is weakly equivalent to a finite type-2 grammar. This follows from the fact that there is a type-3 hypergrammar that generates all and only all of the rules of that subcomponent.

Notes

1 By a simple rewrite rule, we mean a rule of the form $\chi \rightarrow \psi$ that enables subderivations ending in lines of the form $\tau = \phi\chi\omega$ to be continued with lines of the form $\tau' = \phi\psi\omega$. A simple rewrite grammar is a grammar that contains solely simple rewrite rules. Throughout this paper we use the Chomsky (1959) hierarchy of simple rewrite grammars, in which unrestricted rewriting systems are called type-0 grammars; grammars in which all rules are such that the length of ψ is at least that of χ are called type-1 grammars (type-1 grammars properly include the class of context-sensitive phrase-structure grammars); context-free phrase-structure grammars are called type-2 grammars; and finite-state (or one-sided linear or regular) grammars are called type-3 grammars. We modify slightly Chomsky's definition of type-3 grammars so as to allow them to contain rules all of the form $A \rightarrow xB$ or $A \rightarrow y$ (right-linear grammars), or all of the form $A \rightarrow Bx$ or $A \rightarrow y$ (left-linear grammars), where A and B are elements of the nonterminal vocabulary of those grammars, and x and y are finite strings (y nonnull) of elements of the terminal vocabulary of those grammars.

2 There is, in fact, a type-3 grammar that is weakly equivalent to 1. A type-3 grammar that generates the same language that is generated by the grammar in 1 is given in i.

(1) S → {Alice, Bob. Carol, Dave} A

A → {jumped, kicked, lunged, moved} ({C, T})

C → {and, or} {S, T}

T → {Alice, Bob, Carol, Dave} B

B → {jumped, kicked, lunged, moved} {C, T}

3 Note that it is legitimate to think of the rules of a grammar as the sentences of a formal language, formed by concatenation over a vocabulary consisting of the terminal and nonterminal vocabulary of the grammar, its rewrite symbol, and the symbols used to express finite abbreviatory conventions in that grammar, such as parentheses, curly and angle braces, and the comma.

Similarly, it is legitimate to think of entire grammars as sentences of a formal language (for example, the language consisting of all type-2 grammars defined over a fixed, finite vocabulary), provided that those grammars do not have infinitely many rules. Thus, one can also conceive of a grammar that generates all and only all of the grammars of a certain type. Such grammars are not hypergrammars in the sense defined here; they may be called, following Levelt (1974), grammar-grammars (or, if they generate hypergrammars, they may be called hypergrammar-grammars).

4 For convenience, we represent the nonterminal vocabulary of a hypergrammar by upper-case Greek letters that are distinct from letters of the Roman alphabet, the initial symbol by Σ, and the rewrite symbol by the double-shafted arrow =>. The terminal vocabulary of a hypergrammar is the entire vocabulary of the infinite grammar that it generates, together with the rewrite symbol $\rightarrow$ of that grammar, and the symbols used to represent finite abbreviatory conventions in that grammar.

5 We assume that one of the ways for a derivation to block before termination is for a line to be constructed that exactly repeats some earlier line. This assumption is necessary to insure that a derivation that could ultimately terminate cannot endlessly cycle before terminating.

References

Chomsky, N. (1959). "On Certain Formal Properties of Grammars." Information and Control, 2, 137-67.

_____ (1965). Aspects of the Theory of Syntax. Cambridge: MIT Press.

Chomsky, N. and M. Schützenberger. (1963). "The Algebraic Theory of Context-Free Languages." Computer Programming and Formal Systems. P. Braffort and D. Hirschberg, eds. Amsterdam: North-Holland. Pp. 118-61.

Levelt, W. J. M. (1974). Formal Grammars in Linguistics and Psycholinguistics. Vol 2: Applications in Linguistic Theory. The Hague: Mouton.