EMPIRICAL RESTRICTIONS ON THE POWER OF TRANSFORMATIONAL GRAMMARS *

Royal Skousen
University of Illinois

Abstract

In their article "On the Generative Power of Transformational Grammars", Peters and Ritchie (1971) argue that given any recursively enumerable set there exists a transformational grammar that will produce that set. They claim that the deletion rule used in their proof ($b \rightarrow \phi$) is a recoverable deletion rule. This deletion rule obligatorily deletes all occurrences of a terminal symbol b from every sentence of a context-sensitive language related to the original recursively enumerable set. As a result of such a transformation, there exists no surface sentence with b . The terminal symbol b is only recoverable because in their formulation the grammar that produces the b 's is already determined. There is, however, no evidence in the surface strings of the language for any b .

Given the assumption that speakers do not have rules for which there is no empirical evidence, the type of deletion rule found in Peters and Ritchie's proof would never be found in any human language. In terms of language acquisition, the deletion rule is not recoverable. Given such an empirical condition on grammars, the types of rules found in the grammars of natural languages would be restricted on the basis of empirical evidence. Peters and Ritchie's proof fails to meet this empirical condition and is therefore invalid.

In Aspects of the Theory of Syntax, Chomsky (1965) suggests that human languages are recursive sets. A recursive set is a set of strings such that there exists an algorithm that can effectively determine if a given string is in the recursive set or not. Chomsky assumes that speakers possess just such an algorithm with respect to their native language. Speakers can classify linguistic signals into two well-defined sets: sentences and nonsentences. This assumption has been basic to Chomsky's linguistic analyses.

A weaker assumption would be that human languages are recursively enumerable sets. A recursively enumerable set is a set of strings for which there exists a procedure (or Turing machine) that can effectively enumerate the strings in the set, but may be undecidable with respect to strings not in the set. Of course, for some recursively enumerable sets there will exist procedures that will effectively enumerate the strings not in the set as well as those in the set. Such procedures are, by definition, algorithms. But there are recursively enumerable sets for which no algorithm exists. For such a recursively enumerable set, the procedure may be able to determine that some strings are not in the set, but there will exist at least one string for which the procedure will fail. Such strings are undecidable with respect to the procedure.

* Appeared in: York Papers in Linguistics 2 (1972), 61 - 70 and Papers in Linguistics

Chomsky admits that the transformational theory that he proposes in Aspects may be too powerful in that it might be able to describe any recursively numerable set. Chomsky is looking for a theory that will describe human languages. And if human languages are recursive sets, then a correct theory of transformational grammar should only describe the set of recursive languages, not the larger set of recursively enumerable languages (Chomsky, 1965, p. 62). In such a case the proper solution will be to devise formal ways to restrict the theory of transformational grammars so that they describe only the set of recursive languages (p. 208).

Peters and Ritchie (1971) argue that Chomsky's observation about the power of his theory of transformational grammar is, in fact, correct. Having formalized the Aspects' theory of transformational grammar, they prove that every recursively enumerable language is generated by some context-sensitive transformational grammar. Their proof is based upon the following relation of recursively enumerable languages to context-sensitive languages: If L_1 is any recursively enumerable language (that is, any language generated by an unrestricted rewriting system (a type 0 grammar)), then there exists a context-sensitive language L_2 and a regular set R such that $L_1 = L_2/R$. The only difference between a string in L_1 and the corresponding string in L_2 is that the L_2 string may include a string in R . The following method¹ describes how to construct a grammar G_2 that will generate L_2 , given L_1 and G_1 (the grammar that generates L_1): Let $G_1 = (V_N, V_T, P_1, S_1)$ be the unrestricted rewriting system that generates L_1 . We will construct a corresponding grammar $G_2 = (V_N \cup \{S_2, B\}, V_T \cup \{b, \#\}, P_2, S_2)$, where S_2 and B are not in V_N , and b and $\#$ are not in V_T . The production rules in P_2 are determined as follows: (1) If $\alpha \rightarrow \beta$ is a production rule of G_1 and the length of α is not greater than the length of β , then $\alpha \rightarrow \beta$ is a production rule of G_2 . (2) If $\alpha \rightarrow \beta$ is a production rule of G_1 and the length of β is less than the length of α , where the difference in length is n , then $\alpha \rightarrow \beta B^n$ is a production rule of G_2 . (3) For each symbol X in V_N and x in V_T , $BX \rightarrow XB$ and $Bx \rightarrow xB$ are production rules of G_2 . (4) $B\# \rightarrow b\#$ and $Bb \rightarrow bb$ are in P_2 . (5) $S_2 \rightarrow S_1\#$ is in P_2 .

The non-terminal B 's can be converted to the terminal b 's only after the B 's have been moved to the end of the string since the $\#$ symbol which triggers the change of B to b occurs only at the end of the string. Thus we add production rules to P_2 to allow B to move over any other symbol in the vocabulary of G_1 . Each production rule of G_2 is a type 0.5 rewriting rule. If $\alpha \rightarrow \beta$ is already a type 0.5 rewriting rule in G_1 , then the rule is exactly the same in G_2 . If $\alpha \rightarrow \beta$ is not a type 0.5 rewriting rule in G_1 (that is, the length of β is

¹ Hopcroft and Ullman (1969, pp. 132-133). This method varies somewhat in notation from Peters and Ritchie's. Nonetheless, the form of the proof is the same.

less than the length of α), then a minimal number of B's are added to β to make it a type 0.5 rewriting rule. By a theorem of Kuroda (1964, p. 211), every type 0.5 rewriting rule is convertible into a context sensitive (type 1) rewriting rule. Every rule of G_2 is at least a type 0.5 rewriting rule. Hence, G_2 is a context-sensitive rewriting system. In the derivation of a string in L_2 , B (and correspondingly b) can be introduced only when a rule that is not a type 0.5 rewriting rule is used in deriving the corresponding string in L_1 . The only difference then between a string of L_2 and the corresponding string in L_1 will be that the L_2 string will end in the regular expression b^* . By deleting the expression $b^*\#$ from every string in L_2 , we will have a language equal to our original language L_1 . Thus $L_1 = L_2/b^*\#$.

Peters and Ritchie do not, however, postulate a variable deletion rule that will delete the regular set $b^*\#$ in one step. Rather, they propose a cyclical application of a constant deletion rule that will delete a single b when it is the right-most terminal symbol in a sentence. This deletion transformation will apply an unlimited number of times until all the final b's in a string are deleted¹. Peters and Ritchie make this decision because of their definition of recoverability. A deletion is recoverable (1) if a copy or replica of the string that is deleted is left somewhere in the tree, or (2) if the deleted string is a specific, finite string of words (Peters and Ritchie, 1971, p. 8). Since all b's are deleted, no copy or replica of b is ever left in any sentence of the language. But the single b is a specific, finite string made up of terminal symbols in G_2 . Thus b can be deleted by a recoverable deletion rule: "...although [the transformation] does not necessarily preserve a copy of the symbol b, it satisfies the condition of recoverability of deletions by specifying that the deleted string must in every case be a single b." (Peters and Ritchie, 1971, p. 33) If the deletion rule were allowed to delete any string of b's rather than a single b, the transformation would not satisfy the condition on recoverability of deletions; although the deleted string would always be some sequence of b's, it would be impossible to tell just how many b's had been deleted by a given application of the transformation. In order to preserve their notion of recoverability of deletions, only one b at a time can be deleted. For a given application of the transformation, a single b is always recoverable. The unlimited use of a cyclical deletion rule guarantees that any number of b's can be deleted.

There are other observations that we can make about Peters and Ritchie's proof. For example, B never interferes in the grammar G_2 . In order to understand what it would mean for B to interfere in the grammar G_2 , consider a derivation of a string q in G_1 :

¹ According to the notation used in the proof in this paper, I need a final ad-hoc transformation that will delete the # symbol after all the b's have been deleted.

<u>Step in the derivation</u>	<u>Rule used</u>
S_1	
x_1	R_a
.	.
.	.
.	.
x_{i-1}	R_j
x_i	R_k
x_{i+1}	R_ℓ
.	.
.	.
.	.
x_n	R_y
q	R_z

where R represents the rewriting rule used to derive each step of the derivation from the preceding step. (The use of subscripts is only for notational purposes and does not imply that R_j , R_k , and R_ℓ , for example, cannot in fact be an application of the same rewriting rule.) The corresponding derivation in G_2 will be of the following form:

$$\begin{array}{c}
 S_2 \\
 S_1 \# \\
 . \\
 . \\
 . \\
 x_1 b^* \# \\
 . \\
 . \\
 . \\
 x_{i-1} b^* \# \\
 . \\
 . \\
 . \\
 x_i b^* \# \\
 . \\
 . \\
 . \\
 x_{i+1} b^* \# \\
 . \\
 . \\
 .
 \end{array}$$

$$\begin{array}{c}
 x_n b^* \# \\
 \cdot \\
 \cdot \\
 \cdot \\
 qb^* \#
 \end{array}$$

The first string produced must be $S_1 \#$ because S_2 is the start symbol in G_2 and $S_2 \rightarrow S_1 \#$ is the only rule in G_2 that rewrites S_2 . Of course, $S_1 \#$ is of the form $S_1 b^* \#$. Hence, the first rule that applies produces a string ending in the regular expression $b^* \#$. Suppose that in G_1 the rule R_k used in the derivation of q rewrites α as β . Therefore, representing the string x_{i-1} as $\phi\alpha\psi$, x_i would be $\phi\beta\psi$. Let us assume that the step in the G_2 derivation corresponding to the derivation of x_i in the G_1 derivation ends in the regular expression $b^m \#$, where $m \geq 0$. Then the corresponding string for $\phi\alpha\psi$ would be $\phi\alpha\psi b^m \#$. Now let us suppose that in G_2 , rule R_k' (the rewriting rule in G_2 corresponding to the rewriting rule R_k in G_1) is of the form $\alpha \rightarrow \beta B^n$, where $n \geq 0$. The corresponding string for x_i (or $\phi\alpha\psi$) would be $\phi\beta B^n \psi b^m \#$:

	<u>Derivation in G_1</u>	<u>Derivation in G_2</u>
x_{i-1}	$\phi\alpha\psi$	$\phi\alpha\psi b^m \#$
x_i	$\phi\beta\psi \quad (R_k: \alpha \rightarrow \beta)$	$\phi\beta B^n \psi b^m \# \quad (R_k': \alpha \rightarrow \beta B^n)$

Under what conditions could the next step in the G_1 derivation be blocked in the G_2 derivation? This could only occur if the rule R_ℓ rewrites $\beta\psi$ as γ , but that a B or some derivative of B remains between β and ψ , thus blocking the application of rule R_ℓ' . In such a case, we will say that B interferes in the derivation of the string in G_2 . But B will never interfere in G_2 for the following reasons. First of all, all B 's can be moved over the string ψ by application of the rules in G_2 of the form $BX \rightarrow XB$ or $Bx \rightarrow xB$, where $X \in V_N$ and $x \in V_T$. The only other possibility then is that B be rewritten as something which could not be moved over each of the symbols in ψ . The only rules that rewrite B into another symbol are the rules $B\# \rightarrow b\#$ and $Bb \rightarrow bb$. So if B could be changed to b , then rule R_ℓ' could not apply since b cannot be moved over ψ . The first rule ($B\# \rightarrow b\#$) applies only to a final B . In this case then B could not be followed by ψ . Similarly, b 's can only be followed by b 's or $\#$. Hence, B could not be rewritten as b unless B has already been moved over ψ . Thus, neither B nor b can block the application of the next rule, R_ℓ' . Corresponding to the steps $x_i \rightarrow x_{i+1}$ in G_1 , we have in G_2 the following derivation:

$$\begin{array}{c}
 \phi\beta^n \psi b^m \# \\
 \cdot \\
 \cdot \\
 \cdot
 \end{array}$$

$$\begin{array}{c}
\phi\beta B^{n-1}\psi Bb^m\# \\
\phi\beta B^{n-1}\psi b^{m+1}\# \\
\cdot \\
\cdot \\
\cdot \\
\phi\beta\psi b^{n+m}\#
\end{array}$$

This last string is of the form $x_{i+1}b^*\#$. So given a derivation in G_1 , there exists a corresponding derivation in G_2 . The introduction of B into G_2 never causes the derivation of any string q in G_1 to be blocked in G_2 . B never prevents a rule from G_1 from applying in G_2 since any B can be moved to the end of the string; and B never causes a rule from G_1 to apply in G_2 since B can only be moved to the end of the string. Thus B never interferes in the derivation of strings in G_2 .

The element B cannot have been in G_1 ; otherwise there would be a rule in G_1 using B and that rule could be used in G_2 to interfere in derivations. If B were rewritten in G_1 , then B 's introduced by rules in G_2 could be rewritten as in G_1 instead of being moved toward the end of the string. If B is produced by rules in G_1 , then such B 's could be changed to b and eventually deleted in G_2 . Similarly, b cannot have been in G_1 . Thus, B and b do not interfere in G_2 in part because they are not found in G_1 .

Peters and Ritchie's deletion transformation deletes only a string-final b . If b could be deleted any place in a surface string, then there wouldn't be any way of determining in what sequence the b 's had been deleted. If a string contained n number of b 's, the deletion transformation could delete the b 's in $n!$ different ways. For example, if a string contains three b 's, the deletion transformation could delete the b 's in six different orders:

b_1	b_2	b_3
b_1	b_3	b_2
b_2	b_1	b_3
b_2	b_3	b_1
b_3	b_1	b_2
b_3	b_2	b_1

(b_i stands for the i th b in $\alpha b^*\#$; for example, in a string containing three b 's: $\alpha b_1 b_2 b_3 \#$.) By specifying that b is deleted only in sentence final position, the b 's must be deleted in the order b_3, b_2, b_1 . By deleting only the final b , the order of the deletion of the b 's is completely recoverable.

Only enough B's are added to each rewriting rule of G_1 so that no string in any derivation in G_2 will decrease in length. What would happen if we didn't restrict the number of B's that could be added to each rewriting rule of G_1 ? Suppose that we only specified that every rule in G_2 had to be a type 0.5 rule. In this case, we wouldn't be able to tell how many b's had been deleted from the string produced in G_2 even if we knew the original grammar G_1 . For example, a string $\phi ad\psi$ rewritten as $\phi c\psi$ in G_1 by the rule $ab \rightarrow c$ would be rewritten as $\phi adB\psi$ in G_2 if the corresponding rule in G_2 were minimally of type 0.5, that is, $ad \rightarrow cB$. But if no such constraint were placed on B—that is, the corresponding rule need only be of type 0.5 and not minimally so—then the rule in G_2 could be $ad \rightarrow cB$ or $ad \rightarrow cBB$ or $ad \rightarrow cBBB$, and so on. Only by setting such a restriction on the rewriting rules of G_2 can the number of b's deleted by the transformation be recovered, given only the original grammar G_1 and not the constructed grammar G_2 .

Of course, the choice of the symbols B and b is totally arbitrary. One could have chosen any two symbols not in the original vocabulary of G_1 . B and b are not unique because they do not interfere in the grammar of G_2 . They are, in essence, dummy symbols and could be replaced by any other symbols not in G_1 , such as letters from the Greek or Russian alphabets, Chinese characters, hieroglyphics, or any other symbols imaginable.

Finally, the deletion rule is always obligatory, never optional, so that every b is deleted; in no surface sentence does b ever show up. There is absolutely no evidence for b from any of the surface strings. The only reason we know how many b's were originally produced in a given surface string is because we know the grammar G_1 and can derive a derivation for that surface string in G_1 and can determine the corresponding derivation in G_2 . In other words, the notion of recoverability depends crucially upon the fact that the original grammar G_1 is given. If G_1 were not known, the b's in G_2 would not be recoverable because the number of B's (and hence b's) produced in G_2 depends upon what rules are in G_1 . If these rules are not specified, we don't know how many B's were originally produced for a given surface string. Consider, for example, a language which contains strings having a square number of a's:

$$L = \{a, aaaa, aaaaaaaaa, \dots, a^{n^2}, \dots\}.$$

Consider two grammars of the form $G = V_n = (\{S_1, A, C, D, E, F, G, H\}, V_T = \{a\}, P, S_1)$ that could generate L. One would generate strings of the form $AF^nGH^{n^2}$:

$$S_1 \rightarrow AFGH$$

$$FG \rightarrow DGH$$

$$FD \rightarrow DF$$

$$AD \rightarrow AC$$

$$CD \rightarrow FC$$

$$CG \rightarrow FFGH$$

To generate the string for the succeeding integer, two H's would be added for every F (by the rule $FG \rightarrow DGH$) and finally an extra H and F would be added (by the rule $CG \rightarrow FFGH$):

$$AF^n GH^{n^2} \rightarrow AF^n FGH H^{2n} H^{n^2}$$

Of course, the last string is equivalent to $AF^{n+1} GH^{(n+1)^2}$. In order to get a^{n^2} , we would need rewriting rules to delete $AF^n G$ and convert H^{n^2} to a^{n^2} :

$$AF \rightarrow E$$

$$EF \rightarrow E$$

$$EG \rightarrow E$$

$$EH \rightarrow H$$

$$H \rightarrow a$$

In other words, these rules would always delete $n+2$ segments from the string $AF^n GH^{n^2}$ in order to get a string with n^2 segments. The context-sensitive grammar corresponding to this grammar would include the following rewriting rules:

$$AF \rightarrow EB$$

$$EF \rightarrow EB$$

$$EG \rightarrow EB$$

$$EH \rightarrow HB$$

$$H \rightarrow a$$

In this grammar, $n+2$ segments would be converted to B's (and eventually b's), while n^2 segments would end up as a's.

Another possible grammar for producing L would generate strings of the form $AF^{2n+1} GH^{n^2}$:

$$S_1 \rightarrow AFFFFGH$$

$$FG \rightarrow DGH$$

$$FD \rightarrow DF$$

$$AD \rightarrow AC$$

$$CD \rightarrow FC$$

$$CG \rightarrow FFFG$$

To generate the string for the succeeding integer, an H would be added for every F (by the rule $FG \rightarrow DGH$) and finally two more F's would be added (by the rule $CG \rightarrow FFFG$):

$$AF^{2n+1}GH^{n^2} \rightarrow AF^{2n+1}FFGH^{2n+1}H^{n^2}$$

This last string is equivalent to $AF^{2(n+1)+1}GH^{(n+1)^2}$. In order to get a^{n^2} , we would need the same rules as in our first grammar to delete $AF^{2n+1}G$ and convert H^{n^2} to a^{n^2} :

$$AF \rightarrow E$$

$$EF \rightarrow E$$

$$EG \rightarrow E$$

$$EH \rightarrow H$$

$$H \rightarrow a$$

In this case, however, $2n+3$ segments must be deleted rather than $n+2$. This means that in the context-sensitive grammar corresponding to this second grammar, $2n+3$ segments would be converted to b's, while as before, n^2 segments would end up as a's. Given a string of the form a^{n^2} , the b-deletion rule would delete $n+2$ b's from the underlying representation if the first grammar were postulated as generating L. But if the second grammars generated L, then $2n+3$ b's would be deleted from the underlying representation.¹ We cannot tell how many b's have been deleted from any given surface string unless we already know the grammar G_1 .

Peters and Ritchie's proof depends upon the fact that a grammar is already given. Obviously, speakers don't have the grammar when they start to acquire the language. The transformation postulated by Peters and Ritchie is not recoverable in terms of language acquisition. Of course, Peters and Ritchie's definition of recoverability allows such a transformation: "...the deleted sequence of factors had a member of finite, preassigned subset of V_T^* as its terminal string." (Peters and Ritchie, 1971, p. 8). As evidence for this part of the definition, Peters and Ritchie could refer to a deletion rule like imperative deletion in English, which deletes the underlying subject "you" and the auxiliary "will" (Lees and Klima, 1963). The rule always deletes the sequence of words "you will" and is therefore recoverable, according to Peters and Ritchie's definition of recoverability. Arguments have also been given for higher performatives in English that never appear in any surface sentence of English (Sadock, 1969). Such performatives are always deleted. Yet in both these cases, there is evidence for postulating such deletion rules. Various co-occurrence arguments have been given to show that imperatives have an underlying "you will". Otherwise, certain rather general conditions on co-occurrence will be unexplainable for imperative sentences except in some ad-hoc way. Similarly, the arguments for higher performatives are based on various

¹ Both grammars will, of course, need the ad-hoc transformation deleting the final $\#$ after all the b's have been deleted.

co-occurrence relations. The point is that for each of these putative rules, there is evidence for such a rule in the surface strings of English. Arguments can be given only on the basis of evidence. There must be some sentences of the language that contain certain surface forms that will lead speakers to postulate such rules.

Even though certain finite strings are 'recoverable, this does not mean that any finite string is recoverable. For example, consider a putative transformation that would always delete the impossible finite string *hps* from every sentence of English (or any other language, for that matter). And suppose in addition that one of the rewriting rules of English is:

$$S \rightarrow \text{hps } S$$

Now the problem with this deletion rule is that it is not recoverable. No string of the form *hps* ever appears in the surface of a sentence of English, and *hps* never interacts with the other rules of English. *hps* is never rewritten by another rule nor is it produced by another rule. There is absolutely no evidence for postulating an underlying *hps* nor is there any evidence for deleting all occurrences of the finite string *hps* in English. Yet according to Peters and Ritchie's definition of recoverable deletion, such a rule would be recoverable and hence allowable for a grammar of English. Such an example could be constructed for any impossible sequence of sounds in any language. There is something wrong with such a notion of recoverability when it allows totally unempirical rules in natural languages.

In other words, Peters and Ritchie's definition of recoverability is not strong enough. It allows for totally unempirical transformations, rules that would never be postulated by speakers of natural languages. The fact that some transformation in natural languages delete specific strings of formatives does not mean that the deletion of specific strings is always recoverable. Linguists postulate certain deletion rules for English because there is evidence for those rules in the surface sentences of English. And speakers acquire such rules because there is evidence for the rules in the sentences of English. Recoverability can be defined then in terms of empirical evidence.

Let us postulate then an empirical condition on grammars—that no rule can be postulated unless there is some evidence for the rule. Such a convention would not be too unrealistic. In fact, this convention is the basis of the scientific method. Theories are not constructed in a vacuum, but are grounded in empiricism. Grammars are theories of what speakers know about their language and as such must be based upon empirical evidence. Such a condition on grammars has been assumed in nearly all linguistic work. For example, the empirical condition on grammars is the basis of Kiparsky's (1966) alternation condition; speakers will not postulate underlying segments for which there is no surface evidence. All the arguments against the strong form of Kiparsky's alternation condition simply argue that

in some languages there is evidence for underlying segments that never show up on the surface. The point is, in any case, that evidence is found for such abstract segments. In general, linguists should not argue for rules for which there is no evidence at all.

Peters and Ritchie's proof violates the empirical condition on grammars. There is no surface sentence in which *b* appears. A speaker learning the language would never observe *b*. Moreover, since *B* plays a vacuous role in G_2 (that is, *B* never prevents a rule from G_1 from applying in G_2 , nor does it cause a rule from G_1 to apply in G_2), there is no internal evidence for *B* either. A speaker learning the language would never find even a hint of *B* or *b*. As a consequence, Peters and Ritchie have not shown that for any given recursively enumerable set, there exists a context-sensitive transformational grammar that will produce that set. The problem is that the deletion rule is not really recoverable. Of course, there may be a way to circumvent this empirical condition on grammars and still arrive at Peters and Ritchie's conclusion. However, that remains to be shown. The empirical condition on grammars may be sufficient, in fact, to show that there will always be some recursively enumerable set that cannot be produced by a context-sensitive transformational grammar. That too remains to be shown.

References

- Chomsky, Noam (1965), Aspects of the Theory of Syntax, Cambridge, Mass., MIT Press.
- Hopcroft, John E. and Jeffery D. Ullman (1969), Formal Languages and Their Relation to Automata, Reading, Mass., Addison-Wesley.
- Kiparsky, Paul (1968), "How Abstract is Phonology?", Mimeo.
- Kuroda, S. Y. (1964), "Classes of Languages and Linear-Bounded Automata", Information and Control, 7, 207-223.
- Lees, R. B. and Edward S. Klima (1963), "Rules for English Pronominalization", Language, 39, 17-28.
- Peters, P. Stanley, Jr. and R. W. Ritchie (1971), "On the Generative Power of Transformational Grammars", Mimeo. See Information Sciences, 6, 49-83 (1973)
- Sadock, Jerrold M. (1969), "Super-Hypersentences", Papers in Linguistics, 1, 1-15.