OUKA
The University of Osaka
Institutional Knowledge Archive

Title	The Continuing Importance of Peters and Ritchie
Author(s)	Stirk, C. Ian
Citation	大阪外大英米研究. 1990, 17, p. 41-59
Version Type	VoR
URL	https://hdl.handle.net/11094/99140
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

THE CONTINUING IMPORTANCE OF PETERS AND RITCHIE

Ian C. Stirk

Introduction

P. S. Peters and R. Ritchie's paper, "On the Generative Power of Transformational Grammars", was much circulated in photocopied form before it was finally published in 1973, two years after their (1971), which logically follows it.

The stir which it originally caused has largely died down, but maybe it is time to rake over those old coals again. The "Peters-Ritchie results" have never, as far as I know, been exhaustively presented in a purely linguistic way. That is, they have always appeared in a form suited mainly to mathematically minded readers, although there is no reason why the results cannot be put into a shape which is entirely familiar to linguists.

This is what I have tried to do in what follows. After presenting the proofs in a revised form, I have attempted to draw some conclusions from them which may have some pretence to novelty.

The Peters-Ritchie results, it will be remembered, show that transformational grammars are equivalent in generative power to unrestricted rewriting grammars, presumably the most powerful generating devices possible. This is the case even when the transformational grammars meet certain "constraints on deletion", devised expressly to prevent the grammars from being so powerful.

According to these constraints, a transformational rule may only delete items if (i), those items are identical to others not deleted, or (ii), they are members of a certain designated set of terminal symbols. The difficulty of deletion in general is pointed out in my (1988).

The Proofs

If we are permitted to have context-sensitive (CS) phrase structure (PS) rules, it is not difficult to show that transformational grammars (TGs) can generate any unrestricted rewriting (UR) language.

For consider any UR grammar. It is only different from a length-increasing grammar in that some of its rules may involve deletion, that is, their right hand sides may be shorter than their left. Suppose we add a new symbol "B" to the non-terminal vocabulary of the UR grammar : that is, if its original non-terminal vocabulary was V_N, the new one will be $V_N \cup \{B\}$. Now we replace any rule $P \rightarrow Q$, where Q is shorter than P, in the set of productions F of the UR grammar, by

$$P \rightarrow QB^n$$

where n is the difference in length between P and Q. This precisely makes up for the deletion : the right hand side is now just as long as the left. The overall result of this process is that the UR grammar becomes a length-increasing one, with a set of rules F' , say. Unfortunately it may not generate the same terminal strings as before, since not only might B's be added, but those which appear here and there in derivations might block the functioning of some of the rules in the production set F of the original UR grammar.

For example, suppose that F contained the two rules

$$(1) \quad X_1 \; X_2 \; X_3 \rightarrow X_1 \; X_2$$
$$(2) \quad X_2 \; X_4 \rightarrow X_5 \; X_6$$

and that a line in some derivation according to the grammar had the form :

$$(3) \quad \ldots \; X_1 \; X_2 \; X_3 \; X_4 \ldots$$

that is, it contains $X_1\ X_2\ X_3\ X_4$ as a substring. Now, rule (1) could apply to this line, giving

$$(4)\quad \ldots\ X_1\ X_2\ X_4\ \ldots$$

Rule (2) can apply here, giving as the next line :

$$(5)\quad \ldots\ X_1\ X_5\ X_6\ \ldots$$

If we were adding B's to the rules of this grammar, in order to construct a length-increasing one, then rule (1) would become :

$$(1')\quad X_1\ X_2\ X_3 \rightarrow X_1\ X_2\ B$$

Applying it to the line (3) above would yield

$$\ldots\ X_1\ X_2\ BX_4\ \ldots$$

instead of (4). Rule (2), still a part of the length-increasing grammar, cannot apply to this. Adding the B to one rule can quite alter the character of the grammar.

This problem could be avoided if there was a mechanism to move these B's to the right hand side of any string. In that case, they could move "out of the way" of the symbols of the original UR grammar, and permit derivations to continue as usual. A suitable mechanism would be the following set of rules, added to F' :

$$(6)\quad Bx \rightarrow xB, \text{ for all } x\ \varepsilon\ V_N \cup\ V_T$$

There is to be one rule for each member of the whole vocabulary of the UR grammar, terminal and non-terminal. The rule moves B to its right. B's can thus move to the right of anything, and the set of rules (6) is length-increasing, moreover.

So if the set F of rules has deletions "filled in" by occurrences of "B", to create the set F', and if the set (6) is added, the result would be a length-increasing grammar, which generates the same sentences as the UR grammar, except that some of them might have a number of B's to their right.

CS grammars only differ from length-increasing ones in that all the symbols on the left hand side of a rule, except one, must appear also on the right. It is very easy, by introducing new symbols into the non-terminal vocabulary, to convert a length-increasing grammar into a CS one. The method is described in many elementary texts, and I have myself outlined it before (as theorem 7 on page 208 of my 1988), so I will not repeat it here. Let us denote by F'' the CS grammar formed from the rules $F' \cup$ (6). The non-terminal vocabulary of this grammar will be $V_N \cup \{B\} \cup U$, where U is the set of symbols needed for the conversion of the rules to CS form.

Of course, F'' could be the CS PS rules of a TG for the UR language, if a transformational rule could be devised to eliminate the unwanted B's.

Unfortunately there might be any number of these extra symbols to be eliminated, which makes the task impossible for any single transformational rule. A possible mechanism would be the transformational cycle, which enables a transformational rule to apply over and over again in successive subsentences of the output of the PS rules. The PS rules must provide a sufficient number of these subsentences. The following method will do. The non-terminal vocabulary is enlarged by two more members, S and Q, becoming altogether $V_N \cup \{B, S, Q\} \cup U$. "S" is to be the familiar initial symbol of a TG. The following set of rules is added to the CS rules already described :

$$
\begin{array}{lll}
(7) & S & \rightarrow SQ \\
 & S & \rightarrow X_0 \\
 & BQ & \rightarrow bb \\
 & Bb & \rightarrow bB
\end{array}
$$

The last two of these are just length-increasing rather than CS, so once again the grammar will need some adjustment to make it a CS one. The result is to be the PS component of a TG, with initial symbol S. The first rule of set (7) ensures that each subsentence generated by the PS rules, except for the inmost one, consists of a Q. The first two rules of (2), then, generate labelled bracketings of this sort :

$$[_S\ [_S\ [_S \ldots\ [X_0]\ Q]\ _S \ldots Q]_S\ Q]_S$$

X_0 is the initial symbol of the original UR grammar, so after rules have applied to it, the inmost subsentence will come to contain a sentence of the UR language followed by some number of B's :

$$[_S\ [_S\ [_S \ldots\ [_S * * * BB..B]_S\ Q]_S \ldots Q]_S$$

The sentence is represented by " * * * ". The third rule in set (7), or rather its conversion into CS form, changes both "B" and "Q", when they are adjacent, into "b", where "b" is a new symbol in the terminal vocabulary : if the UR language has a terminal vocabulary V_T then the new vocabulary is $V_T \cup \{b\}$. The CS equivalent of the fourth rule in (7) interchanges adjacent B's and b's. Continued application of the third and fourth rules will lead to the labelled bracketing becoming :

$$[_S\ [_S\ [_S \ldots\ [_S * * *\ bb..b]_S\ b]_S \ldots b]_S\ b]_S$$

provided we have the same number of B's and Q's at the start of the process. This, of course, is a matter of chance : the form of the CS rules gives no guarantee that the numbers will be equal. If they are not equal, some B's or some Q's will be left over, and it will be impossible to replace them with terminal symbols by the application of any rule. The CS derivation just comes to an end. According to their formal definition, however, languages are to be strings of *terminal* symbols generated by a grammar ; strings with non-terminal symbols remaining,

but to which no rules are applicable, do not form part of the language. In this particular case, we are left with only those CS derivations which end up with equal numbers of B's and Q's and thus an even number of b's.

The following transformational rule obeys the constraints on deletion :

$$\begin{array}{lcccc} SD: & X & - & b & - & b \\ & 1 & & 2 & & 3 \\ SC: & 1 & & \phi & & \phi \end{array}$$

One of the deletions can be considered a deletion under identity ; the other a deletion of a member of a designated set of terminal symbols, in this case the set {b}.

The effect of the transformation is to delete the b's in pairs. It will apply in every cycle provided that the subsentence of that cycle ends with a pair of b's : otherwise it will not apply. You can easily check that if there are 2x b's, there will be $(x+1)$ cycles available. 2x-2 b's will be eliminated in the first $(x-1)$ cycles, that is, all but two of them. On the xth cycle, there will be only one b at the end of the subsentence, so the transformation does not apply. The last two b's are removed on the final, $(x+1)$ th cycle.

That completes the proof for the case of TGs with CS PS rules. In presenting it, I have followed for the most part the outline given in Peters and Ritchie (1973). Like those writers, I have adapted the device of adding "Q", to fix the required number of subsentences, from Kuroda (1964).

The proof works by showing how a TG, with a CS base component, may be constructed from an arbitrary UR grammar. The TG meets the constraints on deletion, showing that these constraints are insufficient to prevent TG's from generating the whole class of UR languages. The PS rules have to be CS, of course, since apart from CS rules actually introduced in the construction, any amount of context sensitivity may be brought over from the original UR grammar.

The construction of the TG in the proof is quite brief and simple — *elegant*, to use the usual mathematical term. The proof presents a *counterexample*, although a quite general one, to the proposition that the constraints on deletion are adequate. There is no point in criticising the counterexample by saying that its PS rules and its transformation are totally artificial, and unlike anything proposed for the grammars of natural languages. The constraints on deletion were supposed to prevent any TG from generating a language which is recursively enumerable but not recursive. Given any grammar meeting the constraints, therefore, it would be certain without further check that it generated a recursive language. The counterexample removes this certainty : the constraints are not enough, and a grammar which met them would need some further test for us to be sure that it generated a recursive language. The only reasonable concern about the form of the counterexample grammar might be to wonder what would happen if we insisted that *all* derivations from the CS PS rules resulted in terminal strings : in that case, the "trick" of using the symbol Q to help generate enough subsentences would not work.

Despite the simplicity of that proof, its applicability is limited since most grammars for fragments of natural languages do not make use of CS PS rules. These transformational grammars generally have context-free (CF) bases. So the next step is to show that even TGs with finite-state (FS) PS rules can generate the full set of UR languages. Even the weakest generative capacity will do for the base : the main reason is that transformational rules in themselves are powerful enough to carry out the same functions as rules in a UR grammar. The concern about CS PS rules expressed at the end of the last paragraph becomes irrelevant in the light of this.

The following grammar, or rather grammar schema, will prove the result. Unfortunately it is nothing like as elegant as the preceding one, except in one or two of its basic concepts. The finite state phrase structure rules are as follows :

$$
\begin{array}{llll}
S & \rightarrow \# S & X_1 & \rightarrow \# O X_2 \\
S & \rightarrow \# X_1 & X_2 & \rightarrow O X_2 \\
X_1 & \rightarrow \xi X_1 & X_2 & \rightarrow \# X_1 \\
X_1 & \rightarrow \xi S & X_2 & \rightarrow \# S \\
 & & S & \rightarrow O \# O \# O \# O \# O
\end{array}
$$

If the terminal vocabulary of the UR grammar this TG is to imitate is V_T , then the terminal vocabulary of the FS language of the PS rules is $V_T \cup \{\#, O\}$. "O" is a new terminal symbol, while " # " is the "boundary symbol" introduced by Chomsky. The idea is that boundary symbols are removed as transformations apply to base structures. If a terminal string generated by a TG still has boundary symbols in it, it does not count as a grammatical sentence. This is the *filter function* of transformations, and we shall need to make use of it. In the PS rules, " ξ " is intended to range over all symbols in V_T. Thus "$X_1 \rightarrow \xi X_1$" and the others containing " ξ " are rule schemata rather than rules. If there are r members of V_T, then each rule schema stands for a set of r different rules. "S" is used, as a familiar initial symbol of TGs, replacing the mathematically more consistent "X_0".

It will be seen that in part, the PS rules can generate random strings over V_T, interspersed with blocks of the form $\# O^i \#$, where $i \geqslant 1$. These blocks represent the non-terminal symbols of the UR grammar, but in encoded form. If those non-terminal symbols are Y_j ($j \geqslant 0$), then in general

$$\# O^{j+1} \# \text{ encodes } Y_j$$

The last PS rule generates " $\# O \#$ ", the code for the UR grammar's initial symbol "Y_0", flanked on either side by "$O \# O$". The latter is to be a special sign to control the operation of transformations. Notice that the combination "$O \# O$" can only be generated by this last rule. The other PS rules cannot generate O's with fewer than two #'s between them.

Among a great number of others, the PS rules can generate labelled bracketings of this sort :

$$[_s\#[_s\#\ldots[_s\#\,L(m)[_s\#\,L(m-1)\ldots[_s\#\,L(2)[_sO\#O\#O\#O\#O]_s]_s\ldots]_s$$

Here L (i) represents the ith line of a derivation according to the UR grammar. L (1), of course, would normally be "Y_o", but appears here in encoded form. All the non-terminal symbols will be represented by their codes. The derivation runs from right to left, and the last line, L (m), would consist, of course, entirely of members of V_T.

The set of transformations of this TG will be as follows :

T (1)

SD :	# - X -	$[_s$	Y - O # O -	Z	$]_s$	
	1	2	3	4	5	
SC :	O # O	2	ϕ	4	5	

Condition : 2 = 3

This transformation cannot apply on the first cycle, since it mentions the presence of a subsentence, nor will it apply on subsequent cycles unless its identity condition is met. The meeting of that condition depends on subsequent rules. Notice that the deletion of T (1) meets the constraints, as it only applies under the identity condition.

Next comes a transformational schema, standing for a set of k transformational rules. We suppose that the original UR grammar has k rules in its production set, of the form $P_i \rightarrow Q_i$. In the schema below, P_i' and Q_i' represent the encoded forms of P_i and Q_i, that is, forms in which each non-terminal symbol Y_j is replaced by $\#O^{j+1}\#$.

T (i+1)

$(1 \leqslant i \leqslant k)$

SD :	O # O - X -	P_i'	- Y - O # O -	Z		
	1	2	3	4	5	6
SC :	ϕ	2	Q_i'	4	5 + 3	6

In at least one of this set of rules, of course, P_i' will be #O#, the coded form of Y_0, the initial symbol. One rule like this will apply in the first cycle, deleting the initial O#O under identity, quite properly, and placing #O# after the O#O at the right of the string - Z, of course, will be null in this first cycle. No other rule of this set can apply now, since the initial O#O has been deleted, ensuring that the structural description is not met. The effect of an application of one of these transformations is to imitate the application of a rule of the UR grammar, turning one line in a derivation into the next one. In the first cycle, L (1), the first line, Y_0 or #O#, becomes the second line, L (2). Now if, just by chance, the output of the PS rules represented a proper derivation of the UR grammar, L (2) would, after the first cycle, be present in the first subsentence, and also in the next higher subsentence. On the next cycle, then, the structural description of T (1) would be met. As well as deleting one copy of L (2), T (1) replaces the initial # by O#O, giving an opportunity for one of the set T (i +1) to apply. As long as we are dealing with a correct UR derivation, this process will continue. Just one member of T (i+1) will apply on each cycle, to replace L (s), say, of the derivation by L (s+1). In the next cycle, T (1) will delete the superfluous occurrence of L (s+1) and some member of T (i+1) will apply to the remaining occurrence. And so on. Eventually the last line L (m) will be reached. As it consists entirely of terminal symbols of the UR grammar, no member of T (i +1) will apply to it, and T (1) will not apply on the next cycle either, as the sentence being considered begins with #O#O, and the SD of T (1) cannot be met.

There are a number of transformations still to come in the cycle, and of course it is vital that none of these should apply before T (1) through T (k+1) have finished their work of imitating the UR derivation. It is only after this derivation is finished that the sentence being cycled over will start with #O#O, however, so this string is used as part of the SD of the remaining rules. Note that whenever a rule in the set T (2) to T (k+1) applied, the P_i' part of the SD was never deleted, but

adjoined to an occurrence of O # O further along in the string. This was to ensure that any deletions among the original UR rules would not cause the transformations to violate the constraints on deletion. Now, however, these P_i' will have built up to the right of the O # O in the inmost subsentence. All of this unwanted material must be deleted in a way that meets the constraints. The first P_i' to be moved in this way was of course # O #, which is now at the right hand end of the inmost subsentence. T (k+2) includes it in its SD :

T (k+2)

SD :	#	- O -	#	- O -	X	- [$_s$ O	# O	- Y -	#	- # O #]$_s$
	1		2	3	4	5	6	7	8	9
SC :	ϕ		2	3	4	5	6	7	ϕ	9

It will be seen that this transformation deletes any occurrence of # just before the # O # at the end of the inmost sentence. Also, in common with all the remaining transformations except the last two, it deletes the initial #, to ensure that everything works smoothly at a rate of one such transformation per cycle. The following transformation, T (k +3), is very similar, except that it deletes a O instead of a # :

T (k+3)

SD :	# - O -	#	- O -	X	- [$_s$ O	# O	- Y -	O	- # O #]$_s$	
	1	2	3	4	5	6	7	8	9	
SC :	ϕ	2	3	4	5	6	7	ϕ	9	

Between them, T (k+2) and T (k+3) remove all the unwanted remains of non-terminal symbols of the UR grammar, for they were all encoded by means of # and O only. However, it is quite possible for the P_i', the left hand sides of the UR rules, to include terminal symbols of the UR grammar also. There is a restriction even on UR rules, though, that such terminal symbols must also appear on the right hand side of the rule - they may not be deleted (see my (1988), page 191). All such terminal

symbols must therefore appear in the final string of the derivation, and the following set of transformations for deleting the unwanted ones takes advantage of that fact. We suppose that there are r terminal symbols, a_1 to a_r. There are also r transformations in the set, one for each terminal symbol :

T (k+j+3)

$(1 \leqslant j \leqslant r)$

SD :	#	- O -	#	- O -	X	- a_j -	Y	- [$_s$	O # O	- Y -	a_j	- # O #]$_s$
	1	2	3	4	5	6	7		8	9	10	11
SC :	ϕ	2	3	4	5	6	7		8	9	ϕ	11

Each unwanted a_j is deleted under identity with an occurrence in the sentence being cycled over, that is, an occurrence in the sentence of the UR language.

The preceding rules, T (k+2) to T (k+r+3), will delete all the superfluous remains of the P_j', until nothing remains in the lowest subsentence but O # O and the first moved # O #, which has so far remained untouched. The following transformation, whose SD is of course met now for the first time, deletes all this material under a condition of identity :

T (k+r+4)

SD :		#	- O #	- O -	X	- [$_s$	O #	- O #	- O #]$_s$	
		1	2	3	4		5	6	7	
SC :	0 + 1		2	3	4		ϕ	ϕ	ϕ	

The transformation also adds a O to the beginning of the string, which cunningly allows the tidying up to be completed by one final transformation :

T (k+r+5)

SD :	# O -	# O -	# O -	X	
	1	2	3	4	
SC :	ϕ	ϕ	ϕ	4	

Two occurrences of #O are deleted under the condition of identity, the remaining one as an instance of the deletion of members of a finite set of terminal symbols. In this case, the designated set is just {O, #}.

These transformations seem to work smoothly enough in the case of a particular sort of output from the PS rules : one which contains a "correct" derivation of the UR grammar in encoded form, and also a sufficient number of subsentences consisting only of #, which enable all unwanted material to be deleted, leaving only a sentence of the UR language. Any other variety of PS output should fail to result in a terminal string not containing #, and thus not count as part of the language. The most difficult part of constructing a grammar like this is to be sure that no stray strings get through - strings which are not in the UR language, but which somehow succeed in having all #'s removed by some overlooked fault among the transformations.

I don't *think* any error of this sort has crept into the grammar I've presented here, but maybe some astute reader will find one. No doubt it would be easy enough to remove the error without a major change to the grammar as a whole.

Another possibility is that the grammar might fail to generate sentences that are actually in the UR language. The only sort I can think of here would be null sentences, that is, the case where the sentence of zero length is in the language. T (1) above may cause trouble here : in the case of a null sentence, the variables X and Y of that transformation would both be null, and it is not quite clear if the transformation could be said to apply in such a case. Never mind, an extra transformation could be inserted to take of this. I'll leave the details to any interested reader.

As well as these matters, some might complain of the cavalier use of the boundary symbol "#" throughout the grammar. It appears and disappears as the rules apply, instead of remaining quietly at each end of a subsentence until being deleted, as is the case in transformational grammars of natural languages. If anyone is really worried about this, I would suggest using a new terminal symbol, "1", say, to do the work

of the #'s in the grammar. Boundary symbols could be just in their "natural" position at the ends of subsentences, and be deleted only when no 1's remained. Again, I'll leave the details to you, with the warning that fiddling with these transformations can be as addictive as a daily crossword puzzle, and maybe just as irrelevant to the progress of humankind.

No doubt it is the niggling detail that has to be inserted into proofs like the one above that makes them so unappealing to all but a few. Maybe this one could be of use to linguistics students learning how to write transformations correctly, at least if anyone needs to learn how to do that nowadays.

The grammar above is rather different from the one used by Peters and Ritchie (1971), although the principles of its operation are the same. Peters and Ritchie developed a transformational grammar which imitates the working of a Turing machine, rather than a UR grammar. This makes it less accessible to linguists, which is unfortunate, as they should constitute the main audience for these results. Mathematically, the whole business is a very messy and inelegant means of obtaining a not very exciting result - the importance is entirely linguistic.

Another difference between Peters and Ritchie's grammar and the one here involves lexical insertion. I have assumed that, in common with most transformational grammars, lexical insertion should take place in the base. No lexical insertion is carried out by the transformations of the grammar above. This makes the filter function of the transformations extremely important. Only a small proportion of the output of the PS rules will result in sentences, for the FS rules, of necessity, throw symbols together almost at random, when compared to an orderly derivation of the UR grammar.

If I had stuck more closely to Peters and Ritchie's method of construction, then the transformational rules would have imitated successive steps in a derivation according to the UR grammar, instead of checking whether the output of the PS rules constituted a proper derivation. This would have been more economical from the point of view of output string

rejection, but causes a problem with lexical insertion. The only way to avoid most lexical insertion being done by the transformations seems to be to have the PS rules generate the complete terminal vocabulary as a list. Then, instead of inserting items from nowhere, the transformations copy them from the list. At the end of the derivation, superfluous terminal symbols in the list, that is, any that are not present in the terminal string generated, are deleted. That is quite proper, though it is odd, perhaps, that the set of designated terminal items which may be deleted consists of the entire terminal vocabulary ! Peters and Ritchie choose that method, and it has the consequence that the TG's only rejected strings, that is, those that come through the grammar still containing boundary symbols, correspond to derivations of the UR grammar which end up containing non-terminal symbols which cannot be rewritten by any rule.

That in itself would not be so important, but it is possible to envisage UR grammars every derivation of which ends up with a terminal string. That is, no derivation would ever get "stuck" by ending in a line containing at least one non-terminal symbol to which no rule could apply, because of its context. UR grammars of this kind cannot generate all possible UR languages, but only a somewhat restricted, though still large, subset. This subset is discussed in Peters and Ritchie (1973a). I have chosen to ignore it here, since the subset is still unreasonably large, as I hope to show in the next section.

Conclusion

Even if transformational grammars are not allowed to employ a filter function, they may still generate a large subset of UR languages, it appears.

I will not explore the nature of this subset further, for its linguistic importance is small. In fact, it is in many ways surprising that linguists have continued to pay attention to any of the Peters-Ritchie results. Almost all have ignored, it appears, the consequence of Peters (1973).

Any derivation of the sort of transformational grammar presented in the last section will obviously contain a number of subsentences whose contents totally vanish on the way to the terminal string and its structural description. With the usual tree pruning conventions, all trace of those subsentences will disappear. Suppose TG's were restricted to those in which every subsentence must contain at least one "survivor" which has a reflex in the terminal strings generated. Subsentences may not totally vanish in the course of a derivation. Not so surprisingly, Peters (1973) was able to show that only recursive languages may be generated by such grammars, since, given any sentence, the maximum number of subsentences that could have been involved in its derivation can be determined. Thus, only a certain finite number of rule applications could have resulted in the derivation of that sentence, and they may all be checked to see whether or not the sentence is grammatical.

Grammars proposed for human languages have always been of this type, or at least with only harmless exceptions, as Peters (1973) points out. There seems to be no good reason to introduce subsentences, clauses, into the deep structure of sentences, only to have them totally eliminated before the surface structure is reached. Nevertheless, linguists seem totally to have ignored this straw offered by Peters to prevent them drowning in the boundless sea of UR languages.

The usual reaction of linguists to the Peters-Ritchie results is in fact contained in the title of Sampson's (1973) paper : "The Irrelevance of Transformational Omnipotence". The basic idea of this paper, and many since, is that weak generative capacity is quite unimportant : only the strong generative capacity of grammars is worth investigating, and the Peters-Ritchie results affect this not at all. Maybe the most recent of such opinions is expressed by Weinberg (1988), in an overview of present day mathematical linguistics. If weak generative capacity is irrelevant to any linguistic concern, then of course a restriction to non-filtering languages, as in Peters and Ritchie (1973a), or to recursive languages, as in Peters (1973), is not worth even a glance.

Yet it seems to me that linguists have generally ignored the connec-

tions between weak and strong generative capacity, perhaps influenced by the "unnatural" appearance of the devices used in discussing mathematical linguistics. It was observed before that such "unnaturalness" is irrelevant to the force of an argument through counterexample.

Suppose that it is decided, quite *a priori*, that natural languages are, say, context-free. Such a claim has a great effect on the kinds of rules that may be included in a grammar for a natural language, in other words, a great effect on the *strong generative capacity* of such grammars. Context may be mentioned in such rules, deletion may even be carried out by such rules, but in a peculiarly restricted way. These matters have been stressed recently by proponents of generalised phrase structure grammar, and I have summarised some of the formal points in my (1987). A decision about weak generative capacity can profoundly influence the form of rules, and force hard decisions upon grammar writers. Concentration on strong generative capacity only gives too much freedom : types of rule can be decided on quite arbitrary grounds. There are infinitely many ways of generating a set of sentences if any type of rule can be introduced.

I feel there is a certain, if not very strong, analogy between linguistic argument in the present day and certain events in the history of astronomy. Until after the time of Copernicus, astronomers were concerned mainly with the prediction of the places of the planets in the heavens. They devised the system of epicycles, a complex yet internally consistent and somehow compelling method of determining planetary positions to a certain degree of accuracy. Copernicus constructed an alternative system, in which the sun, rather than the earth, occupied the centre of the mechanism. On the other hand, Copernicus' system too remained in the epicyclic framework. Let us compare these systems with those linguistic ones in which strong generative capacity is paramount. There are many reasons why certain grammars should be chosen above others, although they all generate a similar range of sentences, and may be adapted to generate others. Similarly different epicyclic systems have certain advantages and disadvantages, though they all predict planetary positions and get them more or less right.

The end of that particular astronomical tale came when Kepler began to ask questions not just about a planet's position in the sky, but also about how far away it might be. This led to an astonishingly simple mathematical relation between planetary distances and positions, and ultimately to the theory of gravitation and much more. The relation between Kepler's ideas and those of Copernicus is made elegantly clear in Hoyle (1973).

The end of the linguistic tale is not yet in sight. Yet a simple mathematical relation between human languages and formal systems is already detectable : it is remarkable that virtually nothing has appeared in any natural language which is certainly not context-free. It seems to me that linguistics should concentrate on that strange fact, and thoroughly investigate grammars obeying that powerful constraint of context-freeness. Simple mathematical properties of linguistic objects are much more likely to link up with profound theories in other areas like psychology, than are musings about the superiority of one system of epicycles to another.

Bibliography

M. Gross, M. Halle and M-P. Schutzenberger, (eds) "The Formal Analysis of Natural Languages" (Mouton, 1973)

K. Hintikka, J. Moravcsik and P. Suppes (eds) "Approaches to Natural Languages" (Reidel, 1973)

F. Hoyle (1973) "Nicolaus Copernicus : an Essay on his Life and Work" (Heinemann)

S-Y. Kuroda (1964) "Classes of Languages and Linear Bounded Automata" (Information and Control 7, pp 207 - 223)

F.J. Newmeyer (ed) "Linguistics : The Cambridge Survey" (Vol.1) (Cambridge University Press, 1988)

P.S. Peters (1973) "On Restricting Deletion Transformations" (in Gross, Halle and Schutzenberger, eds)

P.S. Peters and R. Ritchie (1971) "On Restricting the Base Component of Transformational Grammars" (Information and Control 18, pp 483 - 501)

P.S. Peters and R. Ritchie (1973) "On the Generative Power of Transformational Grammars" (Information Sciences 6, pp 49 - 83)

P.S. Peters and R. Ritchie (1973a) "Nonfiltering Grammars" (In Hinkikka, Moravcsik and Suppes, eds)

G. Sampson (1973) "The Irrelevance of Transformational Omnipotence" (Journal of Linguistics 9, pp 299 - 302)

Ian C. Stirk (1987) "Context-Free Languages Revisited Yet Again" (大阪外大 英米研究 15, pp 103 - 132)

Ian C. Stirk (1988) "Counting Languages" (大阪外大 英米研究 16, pp 191 - 209)

Amy S. Weinberg (1988) "Mathematical Properties of Grammars" (in Newmeyer, ed, pp 416 - 429)