A CRITICAL LOOK AT A FORMAL MODEL FOR STRATIFICATIONAL LINGUISTICS

Alexander T. Borgida
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A7

SUMMARY

We present here a formalization of the stratificational model of linguistics proposed by Sampson [13] and investigate its generative power. In addition to uncovering a number of counter-intuitive properties, the results presented here bear on meta-theoretic claims found in the linguistic literature. For example, Postal [11] claimed that stratificational theory was equivalent to context-free phrase-structure grammar, and hence not worthy of further interest. We show, however, that Sampson's model, and several of its restricted versions, allow a far wider range of generative powers. In the cases where the model appears to be too powerful, we suggest possible alterations which may make it more acceptable.

1. Introduction

Linguistic theories are at least partially interested in presenting the regularities found in natural languages. Given the current dominance of the Transformational Generative (TG) school in the field of linguistics, it seems necessary for theories competing for attention to possess a formal model. In addition to the advantages normally derived from presenting results through a formalism, such as precision, succinctness and verifiability, one can also comment on the veracity of meta-theoretic claims. It was using such formal arguments that Chomsky and his collaborators demonstrated the inability of finite automata and of context-free grammars to describe all natural language constructs. Similarly, the formal work of Peters and Ritchie [8,9] was important in uncovering inadequacies of two notions of TG theory namely, the "recoverability of deletions condition" and the "universal base hypothesis".

Finally, since many generative linguists want grammatical theories which characterize natural languages, they fault any theory which is "too powerful" in the sense of being able to describe languages which clearly cannot be natural languages, such as non-recursive sets. Furthermore, computer scientists working on natural languages will have to give in the future more consideration to the work of linguists, especially on "exotic" languages, in order to be able to observe a wider range of phenomena. Such access will be facilitated if the formalisms in which the grammars

are presented lend themselves to computer implementation for purposes such as parsing, testing, etc. This entails, among other things, that linguists should avoid as much as possible features which make their grammars generate non-recursive sets, and hence it is one of the purposes of the present paper to point out such features and discuss possible ways of avoiding them.

In this paper we will discuss one model proposed for the stratificational theory of linguistics. This theory, advanced by S. Lamb, H.A. Gleason Jr. and their collaborators ([5],[6],[7]), advocates that languages be described in terms of several subsystems, known as strata. Each stratum has its own set of units and a tactics specifying the "correct" ("allowable") structures on that stratum. A specific grammar might for example have strata corresponding roughly to semantics, syntax-morphology and phonology, although this is by no means standard. Furthermore, the strata are linearly ordered as levels, and there is a realization relation which connects adjacent strata by attaching to every well-formed structure on one stratum, zero or more accompanying structures on the adjacent strata. Note therefore that a particular utterance has simultaneous expression on each stratum.

In this paper we examine the formal model for stratificational linguistics proposed by Sampson ([13]). This model uses rewrite grammars $G_1, G_2, \ldots$ to describe the tactics, while the realization relation is essentially a rewrite system R acting as a transducer between the languages of the tactics. More specifically,

realization connects adjacent tactics G_j and G_{j+1} by matching sentences u in the language generated by G_j with those sentences v in the language of G_{j+1} which can be derived from u by using rules from R. An important property of the linguistic realization relation is the fact that every structure on some stratum can have only a finite number of "realizates" on the next stratum. This means that the rewrite system R must be constrained so that it has no recursive symbols. Such a rewrite system will be called acyclic.

We investigate here the effect of acyclic rewrite systems acting as transducers on axiom sets, varying the type of the derivations and rules allowed.

We prove in this paper that regular languages are closed under transduction by acyclic rewrite systems, but that the linear context-free languages are mapped onto the recursively enumerable sets. This implies that stratificational grammars with non-selfembedding tactics would be too weak while those with even one context-free tactics would be too strong. If the realization derivation is restricted to be in some sense "left-most", then we show that the transduction can be performed by a finite state device known as an a transducer. Furthermore, if productions with null right-hand sides are not allowed in an acyclic rewrite system then all the derivations can be made left-most. This provides one possible method of restricting the generative power of acyclic rewrite systems.

By deriving a recursive characterization of the languages generated with n-strata in terms of (n-1)-stratal languages, we can show that if the realization is restricted to being leftmost, then the languages described are homomorphic images of the intersections of the languages generated by the tactics. In particular, this means that we can find natural families of stratificational grammars which generate for example the sets recognized in real time by nondeterministic multitape Turing machines. This result partially confirms a hitherto unproven claim by Sampson, and discredits Postal's [11] classiciation of stratificational grammars as just another variant of context-free phrase-structure grammars.

Finally, we investigate the use of ordered rules in linguistic grammars and prove that in several models they allow the generation of sets which are not even recursively enumerable a clearly unsatisfactory situation.

The remainder of the paper is structured as follows. In Section 2, we present the formal definitions and notation to be used, including the formal model for stratificational grammars. In Section 3, we examine the properties of "acyclic rewrite systems", which form the principal novel component in our definition of stratificational grammars. We then return in Section 4 to examine the generative power of stratificational grammars and relate the results to linguistics.

2. Definitions

We repeat here some important definitions from [12], and assume that the reader is familiar with the other basic notions of formal language theory.

A vocabulary V is a finite set of symbols, and we use V^+ to denote the set of all non-null strings consisting of symbols from V; using e to denote the null string, we also define V^* to be $V^+ \cup \{e\}$.

A rewrite system RW is a pair (V,R) where V is a vocabulary and R is a finite set of rules (productions) of the form $u \to v$, where $u \in V^+$ and $v \in V^*$; u is known as the left hand side of the production (lhs.) and v is its right hand side (rhs.).

A word $x \in V^+$ is said to directly derive or generate in R another word $y \in V^*$ (denoted by $x \Rightarrow_R y$) iff there exist words u,v,w,z such that x = wuz, y = wvz and $u \to v$ belongs to R. Let $\overset{+}{\Rightarrow}_R$ be the transitive closure of $\Rightarrow_R$, and $\overset{*}{\Rightarrow}_R$ its transitive reflexive closure. A sequence of words $w_1, w_2, \ldots, w_n$ such that $w_1 \Rightarrow_R w_2 \Rightarrow_R \ldots \Rightarrow_R w_n$ is said to be a (free) R-derivation (or simply a derivation) of w_n from w_1.

Given a rewrite system RW = (V,R) and a subset AX of V^*, the language generated by R from axiom set AX with free derivations is defined to be the set $\mathcal{L}(AX,RW) = \{w \mid u \in AX,\ u \overset{*}{\Rightarrow}_R w\}$.

Given the rewrite system RW = (V,R), define the dominance relation $<$ on $V \times V$ by: $d<b$ iff $xby \to udv$ is one of the productions in R (for some strings x,y,u,v) or if there exists some c in V such that $d<c$ and $c<b$. Then RW is defined to be

acyclic (abbreviated a.r.) iff the relation < is anti-symmetric and anti-reflexive.

If $u \to v$ is a production in a rewrite system, it will be called a null rule if v is the null string e, and it will be called context-free if $|u|$, the length of u, is 1.

A rewrite grammar G is a quadruple (N,T,S,P) where N and T are the sets of nonterminals and terminals respectively, S is a distinguished nonterminal and $\mathcal{G} = (N \cup T; P)$ is a rewrite system. In this case, if $u =*>_{\mathcal{G}} w$ then this is called a G-derivation, or a derivation in G, and the language generated by G, denoted by L(G), is defined to be the set $\{t \mid S =*> t \text{ in } G,\ t \in T^*\}$. We assume the reader is familiar with the terminology of type 0 (recursively enumerable or RE), type 1 (context sensitive), type 2 (context free) and type 3 (regular) languages, and corresponding families of grammars and automata. A type 2 grammar will be called linear if all its productions are of the form $A \to aBb$, where $A,B \in N$, $a,b \in T \cup \{e\}$, and will be called selfembedding if for some $A \in N$ there is a G-derivation $A =*> uAv$ where u and v are not null.

New languages can be obtained from old ones through such set operations as union, intersection and concatenation.

One can also define mappings over strings and then extend them to sets of strings in the obvious way. One such mapping is the substitution s which associates with every symbol b of some alphabet T, a set of words s(b) over another alphabet T'; defining s(xy) = s(x)s(y) and s(e) = e, a substitution can be extended to strings. If the sets s(b), are regular, finite or

e-free then s is said to be regular, finite or e-free respectively; if s(b) contains a single word then s is called a homomorphism, and the braces for sets are dropped. A homomorphism h can also be e-free, or it can be length-preserving, if $|h(b)| = 1$ for all symbols b. If $\mathcal{L}$ is a family of languages then we use $H(\mathcal{L})$ and $H^0(\mathcal{L})$ to represent the families of languages obtained from elements of $\mathcal{L}$ through e-free homomorphisms and homomorphisms respectively.

One final operation on strings is reversal defined by Rev(b) = b if $|b| < 2$ and Rev(xy) = Rev(y)Rev(x).

One can also use automata to perform mappings between strings. The a-transducer $M = (K, T_1, T_2, k^0, F, \tau)$ is an extension of the finite automaton, where T_1 and T_2 are the input and output alphabets, and τ is a finite subset of $K \times T_1^* \times T_2^* \times K$ (the transition set). The relation $\vdash$ is defined on $K \times T_1^* \times T_2^*$ by the rule $(k,uv,z) \vdash (k',v,zx)$ if $(k,u,v,k') \in \tau$. The output of M for input word w is one of the strings in the set $\{z \mid (k^0,w,e) \vdash^* (k,e,z),\ k \in F\}$. An a-transducer is said to be e-output free if for any (r,u,v,s) in τ, the string v cannot be null.

A collection of languages $\mathcal{A}$ is said to be closed under the operation σ if $\sigma(L) \in \mathcal{A}$ whenever $L \in \mathcal{A}$. A (full) trio is a family of languages containing at least one non-empty set, closed under e-free homomorphism (arbitrary homomorphism), inverse homomorphism, and intersection with regular languages.

Finally, omitting detailed justification (see [3]), the following formal definition captures the essential aspects of the notion of stratificational grammar, as presented by Sampson [13]:

Definition An n-stratal rewrite grammar (n-RSTRAT) is a 5-tuple RST = (n,TCT,RLZ,V_C,V_E), where V_C and V_E are the set of "content units" and "expression units" respectively, TCT = $(G_1,G_2,\ldots,G_n)$ is a vector of n rewrite grammars, and RLZ = $(R_0,R_1,\ldots,R_n)$ is a vector of $n+1$ acyclic rewrite systems. The transduction performed by such a grammar will be defined by T-RSTRAT(RST) = $\{(u,v) \mid w_0=u \in V_C^+$, $w_{n+1}=v \in V_E^*$, there exist $w_j \in L(G_j)$ such that $w_j \Rightarrow^* w_{j+1}$ via R_j-derivations for $j = 0,1,\ldots,n\}$. Its language is described by L-RSTRAT(RST) = $\{v \mid (u,v) \in$ T-RSTRAT(RST)$\}$.

In this formal model, the grammar is thought of as transducing "meaning" into "sound" in the following manner: starting with a string of "content units" (expressing the meaning of an utterance), the realization rewrite rules are repeatedly applied until a string of "expression units" is obtained. The realization derivation is constrained by the requirement that for each tactics there exists an intermediate stage in the realization derivation which conforms to the tactics specifications (i.e. belongs to the language generated by the tactics). The above formalism is based mainly on Lamb's version of stratificational linguistics; an alternate approach, closer in spirit to Gleason's model, is presented in [3].

3. Generative power of acyclic rewrite systems

To begin with, we remark that the formal definition of stratificational grammars in [13] allows in the realization system rewrite rules with null left-hand sides (i.e. rules of the form e → u). Unfortunately, such rules could be applied to some string an arbitrary number of times. In our stratificational model, this would result in any string having an infinite number of realizates. Furthermore, rules of the form e → u can also be used to establish context-free dependencies in strings generated even from singleton axiom sets. For example, if R = ({c,d},{e→cd}) then $\mathcal{L}(\{e\},R) = \{w | w \in \{c,d\}^*$, w has the same number of "c" and "d" symbols}, which is known to be a non-regular context-free language. The phenomena described above do not appear to have linguistic equivalents, and run counter to the stratificational philosophy which envisages only finitely many realizations for any structure.

As it turns out, in practice rules of the form e → u are only required to introduce in the realization derivation syntactically determined elements, such as "do" in questions. Such insertions need however be performed only once, at the end of every realization derivation between two tactics. Therefore they can be accomplished through normal acyclic rules if each e → u in R is replaced by rules v → uw and v → wu for all v → w in R. For this reason, we will continue to use the definition of rewrite systems which only allows productions with non-null left-hand sides.

We next investigate the effect of a.r. on simple types of axiom sets.

Theorem 3.1 Let AX be a regular set over alphabet T and let E be some alphabet disjoint from T. If RW = (V,R) is an a.r. then $\mathcal{L}(AX,RW) \cap E^*$ is also a regular set.

Proof Let G = (N,T,S,P) be a type 3 grammar generating AX, and without loss of generality assume that $N_G \cap V_R$ is empty. Furthermore, normalize R so that all its rules are of the form a→bc, a→e or bc→d. This can be accomplished in a 3-step process: first, replace rules of the form u → abv ($a,b \in V$, $u,v \in V^*$) by rules u → aa, $\bar{a}$ → bv where $\bar{a}$ is a new symbol; repeat this until all rules have rhs. no longer than two symbols. Next, replace rules of the form abu → v by ab → $\bar{a}$, $\bar{a}$u → v, until all lhs. of rules are at most two symbols. Finally, eliminate rules of the form a → b by adding to R a rule y → zbz' whenever y → zaz' is in R.

Our goal is to produce a type 3 grammar such that R-derivations are "precomputed" in its productions. For example, if the grammar G originally had productions X→aY and Y→bZ, while R contained the rule ab→d, then the final grammar would contain production X→dZ.

For this purpose, consider the following iterative construction:

INITIALIZATION: Let G_1 be G; let $T' = T \cup V_R$.

CONSTRUCTION 1: For every integer i, given grammar $G_i = (N_i,T_i,S_G,P_i)$, construct from it a type 3 grammar $G_{i+1} = (N_{i+1},T',S_G,P_{i+1})$ as follows:

1. for every $a \in T_i$, let P(i,a) be the set of all productions in G_i which have the symbol "a" on the rhs.;
2. to begin with, let P_{i+1} contain P_i, and N_{i+1} contain N_i;
3. IF b→cd is a production in R, THEN for every A→bB in P(i,b), ADD to N_{i+1} a nonterminal [A;B;b→cd], and ADD to P_{i+1} productions A→c[A;B;b→cd] and [A;B;b→cd]→dB;
4. IF b→e is in R, THEN for every A→bB in P(i,b) ADD production A→B to P_{i+1};
5. IF bc→d is in R, THEN for every pair of productions A→bB in P(i,b), and C→cD in P(i,c), ADD to P_{i+1} the new production A→dD <u>if $B \Rightarrow^* C$ in G_i</u>;

END;

Suppose that we were able to establish that

$$\mathcal{L}(L(G),R) = \bigcup_{i=1}^{\infty} L(G_i) \qquad (1)$$

From the construction it is easy to see that P_i is always a subset of P_{i+1} (and hence $L(G_i) \subset L(G_{i+1})$), and if

$$G_m = G_{m+1} \text{ for some index } m \qquad (2)$$

(i.e. no new productions are added to G_m in Construction 1), then G_j would be equal to G_m for every $j > m$.

But, if such an m exists then $\mathcal{L}(L(G);R) = \bigcup_{i=1} L(G_i) = L(G_m)$ and G_m is the type 3 grammar we are looking for. Therefore, it remains to establish equalities (1) and (2).

To prove (1), we first define a new type of derivation ("single, left-right pass") relation "$=o>_R$" as follows:

$u =o>_R v$ iff there exists integer n such that for j = 1,...,n $x_j \to y_j$ is a rule in R and z_j is some string with the property that $u = z_0x_1z_1\ldots x_nz_n$ and $v = z_0y_1z_1\ldots y_nz_n$ (if n = 0, then u = v).

We then claim that

$$L(G_{i+1}) = \{w \mid \exists\ v \in L(G_i) \text{ such that } v =o>_R w\} \qquad (3)$$

This equality can be demonstrated by straightforward inductions on, respectively, the lengths of derivations in G_{i+1}, and the integer n appearing in the definition of $=o>_R$. In both cases the important points are that if $A \Rightarrow bB$ in G_i ($A,B \in N_i$, $b \in T_i$) then either $A \Rightarrow bB$ in G_{i+1} (by step 2 in Construction 1) or $A \Rightarrow uB$ if $b \to u$ is in R (by steps 3 or 4); and if $A \Rightarrow bB =^*> bC \Rightarrow bcD$ in G_1 then $A \Rightarrow dD$ in G_{i+1} in case $bc \to d$ is in R (step 5).

We are now in a position to prove (1). First, suppose that w belongs to $\mathcal{L}(L(G),R)$ and w was obtained from $u \in L(G) = L(G_1)$ in an R-derivation with n steps: $u = u_1 \Rightarrow_R u_2 \Rightarrow_R \ldots \Rightarrow_R u_n \Rightarrow w$. If we note that for any strings x,y $x \Rightarrow_R y$ implies $x =o>_R y$ then by (3) we have for i = 1,...,n that $u_i \in L(G_i)$. But then $w = u_n$ must belong to $L(G_n)$, and hence to $\bigcup_{i=1}^{\infty} L(G_i)$. Conversely, if w belongs to $\bigcup_{1=1}^{\infty} L(G_i)$ then there must exist an index m such that $w \in L(G_m)$. Using (3) it is then trivial to prove by induction on m that there exists $v \in L(G)$ such that $v = v_1 =o>_R v_2 =o>_R \ldots =o>_R v_m = w$ for some $v_i \in L(G_i)$ (1 = 1,...,m). But in that case $w \in \mathcal{L}(L(G),R)$ because by definition $x =o>_R y$ implies that $x =^*>_R y$ for any strings x,y. This concludes the proof of identity (1).

To prove (2), one might try to demonstrate that the construction halts after some precomputable number of steps. This approach unfortunately runs into the following problem: the addition of a new production to G_i in step 4, allows new pairs of variables B' and C' to be connected by a derivation $B' \Rightarrow^* C'$; this may allow new production $A' \rightarrow dD'$ to be added to G_{i+1} in step 5, which in turn may eventually allow step 4 to add a new rule to G_j for some $j>i$, etc.

The above compels us to look for an alternative proof of (2): exhibiting a grammar G^o such that every G_i is a subgrammar of G^o. This would mean that the increasing sequence of grammars $G_1, G_2, \ldots$ is bounded above, and hence converges to one of its elements.

To construct G^o, remember that by definition of R there is an anti-symmetric relation $<$ on V_R. Using this, we assign to every symbol in V and every production in R a unique index number, according to the following algorithm:

INDEXING ALGORITHM:

1. I(b) := 0 for every $b \in V$ such that there is no $d \in V$ and $d>b$;
2. FOR i=0 to |V| DO WHILE not all symbols have an index;
   IF I(b)=i & b→cd is in R, THEN I(c) := I(d) := i+1 and
   I(b→cd) := i+1;
   IF I(b)=i & I(c)≤i & bc→d is in R THEN I(d) := i+1 and
   I(bc→d) := i+1;

IF I(b) = i & I(c)≤i & cb→d is in R THEN I(d) := i+1 and

I(bc→d) := i+1;

IF I(b) = i and b→e is in R THEN I(b→e) := i+1;

END

END

By the acyclicity of R, the above algorithm produces a unique value for every symbol and production. Suppose the highest index value assigned is n. Then G^O will be constructed from G by repeated modification in n passes through the following:

CONSTRUCTION 2: Let $G^O = (N^O, T', S, P^O)$ be G initially;

FOR i=1 to n DO

* in the i-th pass, add to G^O all possible productions representing derivations by index i rules */

1. For every symbol 'd' in V_R such that I(d) = i, let P(d) be the set of all productions currently in G^O, with 'd' on the rhs.;
2. IF b→dc (or b→e) is in R and has index i (iff I(b)=i), THEN alter G^O in exactly the same way as in steps 3 (or 4) of CONSTRUCTION 1; (except that P^O and N^O are used instead of P_{i+1} and N_{i+1}).
3. IF bc→d is in R and has index i, THEN for every pair of productions A→bB in P(b) and C→cD in P(c), ADD to P^O the new production A→dD (<u>whether or not B=>*C in G^O</u>).

Note that in the i-th pass the only productions added to G^o have on the rhs: a terminal symbol of index strictly greater than i. Therefore, in successive passes through the loop after the i-th one, P(d) remains unchanged for all symbols "d" with $L(d) \leq i$. Furthermore, the output G^o remains unchanged if passed through CONSTRUCTION 2 a second time.

Secondly, note that if some grammar K remains unchanged by CONSTRUCTION 2 then it does so through CONSTRUCTION 1 as well, because every rule in R is eventually considered in CONSTRUCTION 2, and in each case at least those productions which would have been added by CONSTRUCTION 1 are added by CONSTRUCTION 2.

Therefore, since G is a subgrammar of G^o, G_i will be so for every i greater than 1, and the proof is completed. □

Increasing the range of sets from which we choose the axiom sets, we obtain the following:

Theorem 3.2 Let $G = (N,T,S_G,P_G)$ be an arbitrary type 0 grammar. Then there exists a linear context-free language LIN_G, and an a.r. R_0 (which is dependent only on the set $N \cup T$), such that

$$L(G) = \mathcal{L}(LIN_G,R_0) \cap T^*$$

Proof (Notational convention: let $V = N \cup T$, and if $\epsilon\{-,\sim,\vee\}$ then use V to represent the set $\{\dot{a} | a\epsilon V\}$, and $\dot{w} = \dot{a}_1 \ldots \dot{a}_j$ if $w = a_1 \ldots a_j$.)

It is known ([1]) that there exist two linear context-free languages L_1 and L_2, as well as a homomorphism h, such that $L(G) = h(L_1 \cap L_2)$. We have constructed ([3]) pairs of new such languages:

$$L_1 = (\{\bar{\%}\bar{S}_G\bar{\%}\$\tilde{\%}\} \cup \{\bar{\%}\bar{v}_1 \ldots \bar{\%}\bar{v}_{m-1}\bar{\%}\bar{S}_G\bar{\%}\$\tilde{\%}\tilde{v}_m \ldots \tilde{\%}\tilde{v}_{2m-2}\tilde{\%}\check{z} \mid m>0,\ z\epsilon T^*,$$
$$v_i \epsilon V_G^+, \mathrm{Rev}(v_{i-1}) = v_{m+i-1} \text{ for } i = 1,\ldots,m-1\}$$

and

$$L_2 = \{\bar{\%}\bar{w}_1\bar{\%} \ldots \bar{w}_n\bar{\%}\$\tilde{\%}\tilde{w}_{n+1}\tilde{\%} \ldots \check{w}_{2n} \mid n>0,\ w_i \epsilon V^+ \text{ for } i<n,$$
$$\text{and } \mathrm{Rev}(w_{n-i}) \Rightarrow_G w_{n+i+1} \text{ for } i = 0,\ldots,n-1\}$$

In this case, the homomorphism h is defined as $h(\check{x}) = x$ if $x\epsilon T$, null otherwise. Observe that L_1 is dependent solely on the vocabulary V and it only checks whether the strings around the central '$\bar{\%}\bar{S}_G\bar{\%}\$\tilde{\%}$' are mirror images of each other. But the following rewriting system does exactly the same job, and, in addition, performs the homomorphism h:

$$R_0 = \{\bar{\%}\bar{S}_G\bar{\%}\$\tilde{\%} \to e,\ \bar{\%}\tilde{\%} \to e,\ \bar{x}\tilde{x} \to e \text{ for all } x\epsilon V_G,\ \check{a} \to a \text{ for } a\epsilon T\}$$

Then $T^* \cap \mathcal{L}(L_2, R_0) = h(L_1 \cap L_2) = L(G)$ and by observation it is clear that R^0 is acyclic. □

This result is surprising, especially from a linguistic point of view, and demonstrates the power of acyclic rewrite systems. Since it is undesirable that linguistic mechanisms be so powerful, we will attempt to put bounds on them. One way to do so is to restrict the places where steps in derivations can occur.

Essentially, in a k-leftmost derivation there is a k-symbol wide "window" on the derivational forms where rewriting can occur, and this window is only allowed to move to the right.

Definition 3.1 Let $w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$ be an R-derivation, where for $i = 1,\ldots,n$ productions $u_i \rightarrow v_i$ are used to obtain $w_i = x_i v_i u_i$ from $w_{i-1} \triangleq x_i u_i y_i$ $(x_i, y_i, w_i \in V^*)$. For any integer k, this is said to be a k-leftmost derivation if for all $i = 1,\ldots,n-1$ there exist s_i and t_i in V^* such that $x_i = t_i s_i$ with $|s_i| \leq k$ and $|t_i| \leq |t_{i+1}|$. This definition of R-derivations gives rise to the new language

$$\mathcal{L}(AX,RW,k\text{-ld}) = \{w \mid x \in AX,\ x \Rightarrow_R^* w \text{ in k-leftmost derivation}\}.$$

Theorem 3.3 Given an a.r. RW = (V,R), there exists an a-transducer Θ_R such that $\mathcal{L}(AX,R,k\text{-ld}) = \hat{\Theta}_R(AX)$.

Proof By the acyclic nature of the rules in R, any string of length k can be rewritten into a string of length at most kd^t where d is the length of the longest rhs. of a rule in R, and t is the number of symbols in V. Therefore, if we define a Turing machine transducer Θ_R which simulates on its working tape k-leftmost R-derivations, then it need have only a bounded, finite-length tape. But this can obviously be kept in a finite memory, and hence Θ_R can be made into an a-transducer. □

Therefore, leftmost constraints on R-derivations lead to a much more restricted version of rewrite systems because all trios (in particular LINEAR-CFL) are closed under a-transduction.

A second method of bounding the power of a.r. is to restrict the form of the productions allowed in R.

Theorem 3.4 If RW = (V,R) is an a.r. with no null productions, then for every axiom set AX, $\mathcal{L}(AX,RW) = \mathcal{L}(AX,RW,k\text{-ld})$ for some constant k.

Proof Suppose R has r rules involving t symbols and let c be the length of the longest lhs. of a production. Now observe that if $v \Rightarrow_R^* w$, then no symbol in w can have more than $k=c^t$ ancestors in v, all of which must be adjacent in v (i.e. the presence of a symbol in w can depend only on the presence of at most k adjacent symbols in v). This value of k can be obtained as follows: since the rules are acyclic, new symbols must appear after every application of a production and hence every symbol in w can be the result of applying at most t productions; since each of these uses at most c symbols as context, we get the value of c^t. The adjacency requirement comes from the condition that there be no e-rules in R. Consider now some R-derivation $tXv \Rightarrow^* t\tilde{t}$ $(t,\tilde{t},v\in V^*, X\in V)$, where no symbol in t is rewritten, but X is. We will prove by induction on the length of the derivation that there is an equivalent k-leftmost derivation.

Basis. If the derivation has 0 or 1 steps then it is clearly k-leftmost.

Induction step. Break up the derivation into steps, to see where X is rewritten:

$$tXv \underset{①}{\Rightarrow^*} tXw\tilde{v} \underset{②}{\Rightarrow} tYzv \underset{③}{\Rightarrow^*} t\tilde{t}$$

where we used rule $Xw \rightarrow Yz$ in step ②.

Now find the last production in ① which produces only non-ancestors of Y.

(a) If there is no such production, then by our opening remarks ① must be k-leftmost, and hence ③ can be made k-leftmost by induction.

(b) Otherwise, suppose that the last such rule was $\alpha \to \rho$. Then we claim that the derivation in further detail is

$$tXv \underset{④}{=^*>} tXu_1\alpha u_2 \underset{⑤}{=>} tXu_1\rho u_2 \underset{⑥}{=^*>} tXw\tilde{u}_1\rho u_2 \underset{②}{=>} tYzu_1\rho u_2 \underset{③}{=^*>} t\tilde{t}$$

The significant part of this claim is that no production in ⑥ affects the string ρu_2, and this is true by our choice of $\alpha \to \rho$ as the last production generating non-ancestors of Y, hence of Yz, and the necessary contiguity of ancestors. But now note that step ⑤ can be postponed to yield the following reordered derivation:

$$tXv \underset{④}{=^*>} tXu_1\alpha u_2 \underset{⑥}{=^*>} tXw\tilde{u}_1\alpha u_2 \underset{②}{=>} tYz\tilde{u}_1\alpha u_2 \underset{⑤}{=>} tYz\tilde{u}_1\rho u_2 \underset{③}{=^*>} t\tilde{t}$$

By repeating the construction in part (b) on ① = ④⑥ this time (instead of ④⑤⑥) we will eventually (by a second induction, if desired) achieve case (a), and thus complete the proof. #

Note that in the above proof we had only excluded the use of null rules (the symbol ρ could not be null) so that other productions with left-hand sides longer than right-hand sides are still allowed in R.

Finally, we include for completeness the following result whose proof is trivial.

Proposition 3.5 Let RW = (V,R) be an a.r. which has only context free rules. Then there exists a finite substitution s_R such that for every axiom set AX, $\mathcal{L}(AX,RW) = s_R(AX)$. #

4. Stratificational Grammars

We now return to the notion of stratificational grammar which led us originally to consider acyclic rewrite systems. To begin with, note that the original definition of n-RSTRAT grammar has no constraint on the derivations occuring on the tactics, while in practice linguists appear to view the derivations as being leftmost (i.e. the leftmost nonterminal is the one rewritten). Therefore, throughout the following discussion we will examine the differences arising out of this variation.

First, we present a recursive characterization of the n-RSTRAT languages. For this purpose, define the language generated by a 0-RSTRAT grammar $RST^0 = (0,(),(R^0),V_C,V_E)$ as L-RSTRAT(RST^0) $\mathcal{L}(V_C^*,R^0) \cap V_E^*$. Then the following theorem is an obvious consequence of the definition of L-RSTRAT:

Theorem 4.1 If $RST = (n,(G_1,\ldots,G_n),(R_0,\ldots,R_n),V_C,V_E)$ is an n-RSTRAT grammar, and TOP(RST) is the (n-1)-RSTRAT grammar $(n-1,(G_1,\ldots,G_{n-1}),(R_0,\ldots,R_{n-1}),V_C,T_n)$, then L-RSTRAT(RST) = $\mathcal{L}(\text{L-RSTRAT(TOP(RST))} \cap L(G_n),R_n) \cap V_E^*$.

Using Theorems 4.1, 3.1 and the known closure properties of the regular languages, it is easy to see that if all the tactics $G_1,\ldots,G_n$ of an n-RSTRAT grammar are non-selfembedding then the stratificational grammar can generate only a regular language.

On the other hand, as soon as one of the tactics is allowed to be of type 2 and selfembedding, then by Theorems 4.1 and 3.2

the RSTRAT grammar can generate an arbitrary RE set. Even more surprisingly, this can be accomplished using a "universal realization relation", meaning that to obtain any RE set we need only vary the tactics, not the realization rewrite system. This situation is similar to that found for TG in [9], where the transformational component can be varied while the base grammar is kept fixed.

Therefore, in this stratificational model there seems to be no alternative between the insufficient descriptive power of finite automata and the excessive power of arbitrary Turing machines. These results hold even if the derivations on the tactics are constrained to be leftmost. We must therefore search for further limitations on the realization process. In section 3 we considered several possible ways of doing this, namely eliminating null or context-dependent rules, or making the realization derivation leftmost. In linguistic grammars there is a clear need for context-dependent realization rules, hence these cannot be eliminated. Although in Sampson's model null realization rules appear to be needed (more on this below), it is possible to envisage alternative models which avoid them. By Theorem 3.4, the absence of null rules is equivalent to restricting the realization derivation to being k-leftmost. Furthermore, based on current linguistic literature there appears to be no objection to limiting the realization to being k-leftmost. Therefore, we will examine the generative power of n-RSTRAT grammars under this constraint.

Theorem 4.2 If STR = $(n,(G_1,\ldots,R_n),(R_0,\ldots,R_n),V_C,V_E)$ is an n-RSTRAT grammar with realization derivations restricted to be k-leftmost for some integer k, then there exist homomorphism h and languages $L_1,\ldots,L_n$ such that for $i = 1,\ldots,n$ $L(G_i)$ is of the same type[1] as L_i, and L-RSTRAT(STR) = $h(L_1 \cap \ldots \cap L_n)$.

Proof The proof is based on a number of results about trios, which we summarize here from [4]:

(a) For $i = 1,\ldots,n$ the families of languages of the same type as $L(G_i)$ are trios.

(b) If L is a trio then $H(L)$ is a trio and $H^o(L)$ is a full trio.

(c) If $L_1,\ldots,L_n$ are trios then $H(H(L_1 \cap \ldots \cap L_{n-1}) \cap L_n)$ is a trio and it is equal to $H(L_1 \cap \ldots \cap L_{n-1} \cap L_n)$; similarly $H^o(H(L_1 \cap \ldots \cap L_{n-1}) \cap L_n) = H^o(L_1 \cap \ldots \cap L_{n-1} \cap L_n)$ is a full trio.

(d) trios are closed under intersection with regular sets and e-output bounded a-transductions, while full trios are also closed under arbitrary a-transduction.

We now prove the theorem by induction on n.

Basis. For n=1, by Theorems 4.1 and 3.3 there exist a transducers Θ_1 and Θ_0 such that L-RSTRAT(RST) = $\Theta_L(\Theta_0(V_C^*) \cap L(G_1)) \cap V_E^*$; then our theorem holds by notes (a) and (d) above with h being the identity map.

Induction step. For the case n+1, by Theorems 4.1 and 3.3 there exists a-transducer Θ_{n+1} such that L-RSTRAT(RST) is equal to

$$\Theta_{n+1}(\text{L-RSTRAT}(\text{TOP}(\text{RST})) \cap L(G_{n+1})) \cap V_E^*. \qquad (4)$$

---

[1] Meaning type 3, type 2, type 1, type 0, linear language.

But by induction, there exist homomorphism h'' and languages $L''_1,\ldots,L''_n$ such that L-RSTRAT(TOP(RST)) = $h''(L''_1 \cap \ldots \cap L''_n)$. Substituting this in (4) and applying notes (a), (b) and (c) we find a homomorphism h and languages $L_1,\ldots,L_{n+1}$ of the same type as $L''_1,\ldots,L''_n$ and $L(G_{n+1})$ such that L RSTRAT(RST) = $h(L_1 \cap \ldots \cap L_{n+1})$. #

Remark that by Theorems 3.4 and 3.5 the same result holds in the case when the realizations do not contain null rules, and by examining the above proof it can be seen that the homomorphism h can be restricted to being e free in this case.

The following converse to Theorem 4.2 can be easily established:

Theorem 4.3 Given homomorphism h from T to T', and rewrite grammars $G_1,\ldots,G_n$ with terminal alphabets T, then for i = 0,...,n there exist context-free acyclic rewrite systems R_i such that L-RSTRAT((n,$(G_1,\ldots,G_n)$,$(R_0,\ldots,R_n)$,T,T') = $h(L(G_1)\cap\ldots\cap L(G_n))$.

Proof For j<n, define R_j to be $\{a\rightarrow a \mid a\in T\}$, by the definition of RSTRAT-derivations this will simulate the intersection of the languages generated by the tactics. Finally, define R_n to be $\{a\rightarrow h(a) \mid a\in T\}$, thus performing the homomorphism on the intersection. #

To begin with, the above theorems partially confirm Sampson's hitherto unproven claim ([13: page 11]) that stratificational languages are the result of intersecting the languages of the tactics. Note however two important qualifications to this claim:

the realization derivation must be k-leftmost and a homomorphism must be applied to the intersection of the languages.

Theorems 4.2 and 4.3 show that with k-leftmost realization derivations, the type i languages (i = 1,2) can be obtained by using a type i grammar on one of the tactics, and making the other ones non-selfembedding. If all the tactics generate context-free languages (as in the case when tactic-derivations are leftmost) then n-RSTRAT grammar can generate the homomorphic intersections of the CFLs. For n≥2, this is known to equal the RE sets if null realizations are allowed; if null realization rules are not allowed then for n≥3 the n-RSTRAT grammars generate the family QUASI of sets recognized by nondeterministic Turing machines in real or linear time ([2]). These observations demonstrate that n-RSTRAT grammars can be appropriately modified so that they generate various language families intermediate between the regular and RE sets. Unfortunately, even when the realization derivation is restricted to being k-leftmost, 1-RSTRAT grammars with context-sensitive tactics and 2-RSTRAT grammars with context-free tactics can generate the RE sets, unless null realizations are restricted. The basic problem with restricting null rules lies in the pronounced bias of this model towards the realization of terminal units from one tactics to the next. In practice, in order to describe linguistic phenomena it is necessary to have information about the entire derivation process on some tactics. Sampson accomplishes this by introducing "pseudo-

terminals" into strings; for example, if the application of production x→y is to be noted for later use, then either rule x→py or x→yp would be used in the tactics to introduce p as a marker of the occurrence of x→y. The chief drawback of this approach is that the "pseudo-terminals" such as p must eventually be deleted, making null rules necessary. One possible solution may be to discover some bound on the number of null rule applications needed, resembling the "cycling function" proposed by Peters and Ritchie ([9]). Another solution is to consider a new formal model which allows realization to access uniformly all parts of the derivations on tactics; this approach is considered in [3].

Before concluding, we take a brief look at the problems raised by one addition to the basic model discussed so far, namely ordered rules. It has often been found useful in linguistic descriptions to use rules of the form "A→u if some condition C holds, otherwise A→v"; basically, these types of rules avoid stating the negation of condition C, which may be cumbersome. In certain stratificational descriptions, this has lapsed into the use of rules of the form "A→u if this can lead to a completed derivation, otherwise A→v". This notion is formalized by Sampson through the assignment of "weights" or "preference values" to certain rules. Thus A→u may be given value 1 while A→v receives value 0, and these values are accumulated throughout the derivation. At the end, only those expression strings resulting from some content string are taken

which have derivations with maximal preference values. The fundamental problem with this use of "ordered rules" is that even in context-sensitive grammars it is in general recursively undecidable whether a certain derivation can be successfully completed or not. In fact, we show that using "ordered rules" we can generate even non-recursively enumerable sets, an obviously undsirable situation.

Theorem 4.4 There exists a context-sensitive grammar G with one "ordered rule" which generates a non-RE language.

Proof The proof rests on the well known result that there exists an RE language L^o over some alphabet T, whose complement is not RE, and that there is a type 1 grammar $G^o = (N^o, T^o, S^o, P^o)$, where $T^o = T \cup \{b,\#\}$, such that $L(G^o) = \{w\#b^{i(w)} \mid w \in L^o$, $i(w)$ is some integer, depending on w $\}$ ([12]). Consider the grammar $G' = (N', T', S', P')$ where $N' = N^o \cup T^o \cup \{Y, S', Z\}$, $T' = \tilde{T} \cup T$, P' contains P^o and additional productions as described below. The grammar G' behaves informally as follows:

(a) from S', we generate some string wS' such that $w \in T^*$, using productions from $\{S' \to aS' \mid a \in T\}$;

(b) then we apply the ordered rule " $S' \to YS^o$ with weight 1, $S' \to Z$ with weight 0"; the plan is that the new nonterminal Y can be rewritten into a terminal, #, if and only if Y appears in a string belonging to $\{wYw\#\}\{b\}^*$ (i.e. iff rules of G^o can be used to rewrite S^o into some $w\#b^j$, where w is the same as the guess made in (a)). Once some derivation from S^o is completed, it is

clear that context-sensitive rules can be used to check out the above condition for Y. In addition, the same rules can place "bars" over all the symbols thus checked, resulting, if successful, in a sentence of the form $\bar{w}\#\bar{w}\#b^j$.

(c) Z on the other hand simply travels across the string w and places "dots" on top of every symbol, using rules from $\{sZ \rightarrow Z\dot{s} | s \in T\}$

The result will be that $L(G') = \{\bar{w}\#\bar{w}\#b^j | w \in L^o\} \cup \{\dot{w} | w \notin L^o\}$. Suppose that L(G') is RE, and let h be the homomorphism which deletes all symbols not in T, and removes the dots from the others. Then h(L(G')) is also RE because the RE sets are closed under homomorphism; but h(L(G')) is the complement of L^o, and thus not in RE by our choice of L^o. Therefore by contradiction, L(G') is not RE.

A similar proof can be given for stratificational grammars with two or more context-free tactics. These results draw attention to the need to redefine the notion of "ordered rule" in stratificational usage, and point out that care must be taken whenever formalizing aspects of linguistic practice.

In conclusion, our investigation of the formal properties of the stratificational model proposed by Sampson revealed certain unintuitive properties which make it less desirable as a tool for natural language description. Thus, the use of realization rules with null left hand sides was shown to allow unbounded number of

realizations for certain strings. More significantly, we showed that n-RSTRAT grammars with even one tactics allowing self-embedding could generate all RE sets. Since there are well known problems raised by this possibility, most significant being the inability to decide grammaticality, we identified a linguistically acceptable restriction on the realization, namely k-leftmost derivations, which led to improvements in some situations. Under this additional constraint, classes of n-RSTRAT grammars were shown to variously generate the context-free languages, the Quasi-realtime languages and the context-sensitive languages. Unfortunately, even in this case n-RSTRAT grammars could generate non-recursive sets, unless null realizations were restricted, and we discussed the problems inherent in this approach. Finally, we examined the definition of "ordered rules" used in some stratificational grammars, and formalized by Sampson, showing that it allowed the generation of even non-RE sets with type 1 tactics.

The above formal results about the generative power of stratificational grammars hopefully answer the requests of critics such as Pit'ha ([10]), and demonstrate the inaccuracy of Postal's classification of stratificational grammars as simply variants of context-free phrase structure grammars. The results also indicate some of the problem areas in this formal model for stratificational linguistics. We emphasize though that the problems are specific to this particular formalism, and should not be taken as condemnations of stratificational linguistics in general, since there are other stratificational models which avoid these pitfalls.

Acknowledgements

I would like to thank Professor Ray Perrault for his much appreciated advice and help both during my graduate student career and after it. I am also grateful to Teresa Miao for typing this paper and to Peter Schneider for proof reading it.

References

[1] Baker, B. and R. Book (1974). "Reversal Bounded Multi-pushdown Machines", J. Computer Systems Science - 8, 1974, 315-332.

[2] Book, R.V. and S.A. Greibach (1970). "Quasi-realtime Languages", Math. Systems Theory - 4, 1970, 97-111.

[3] Borgida, A.T. (1977). "Formal Studies of Stratificational Grammars", Ph.D. Dissertation, University of Toronto, also Technical Report No.112.

[4] Ginsburg, S. (1975). Algebraic and Automata-Theoretic Properties of Formal Languages, North-Holland Publishing Co.

[5] Gleason, H.A. Jr. (1964). "The organization of language: a stratificational view", Monograph Series on Language and Linguistics - 17, p.75-95, Georgetown University.

[6] Lamb, S. (1966). Outline of Stratificational Grammar, Georgetown University Press, Washington.

[7] Lockwood, D.G. (1972). Introduction to Stratificational Linguistics, Harcourt Brace Jovanovich, Inc.

[8] Peters, P.S. and R.W. Ritchie (1971). "On restricting the base component of Transformational Grammars", Information and Control - 18.

[9] Peters, P.S. and R.W. Ritchie (1973). "On the generative power of Transformational Grammars" Information Sciences - 6, 49-83.

[10] Pit'ha, P. (1974). "On a new form of Lamb's stratificational grammar", Slovo a Slovesnost - 35, p.208-218; translated from the original Czech by D.G. Lockwood.

[11] Postal, P. (1964). Constituent Structure: A Study of Contemporary Models of Syntactic Description, Indiana University, Bloomington, Ind. First appeared in Int. J. Amer. Linguistics - 30.1, part 3.

[12] Salomaa, A. (1973). Formal Languages, Academic Press, New York.

[13] Sampson, G. (1970). Stratificational grammar: a Definition and an Example, Janua Linguarum, Series Minor: 88, The Hague Mouton.