AD708402

Technical Report 70-111
Nonr-5144(00)

April, 1970

ISOTONIC GRAMMARS, PARALLEL GRAMMARS, AND PICTURE GRAMMARS

Azriel Rosenfeld

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

28

Technical Report 70-111 April, 1970
Nonr-5144(00)

ISOTONIC GRAMMARS, PARALLEL GRAMMARS, AND PICTURE GRAMMARS

Azriel Rosenfeld

The support of the Information Systems Branch, Office of Naval Research, under Contract Nonr-5144(00), is gratefully achnowledged.

RESEARCH PROGRESS REPORT

TITLE: "Isotonic grammars, parallel grammars and picture grammars", A. Rosenfeld, University of Maryland Computer Science Center Technical Report 70-111, April 1970; Contract Nonr-5144(00).

BACKGROUND: The Computer Science Center of the University of Maryland is investigating the theory of image processing by computer. One area under study is the theory of "grammars" whose "languages" are arrays rather than strings.

CONDENSED REPORT CONTENTS: When one attempts to generalize phrase-structure grammars from strings to arrays, difficulties arise which can be avoided if the grammars are required to be isotonic: in any array rewriting rule, the left and right members are congruent subarrays. For strings, such grammars are exactly as powerful as monotonic grammars, provided that derivations can begin with any initial string of the form $\#S^k\#$ rather than always with #S#.

Isotonic context-sensitive array rewriting rules are essentially the same as local digital picture processing operations. Since the latter are often applied to pictures in parallel, it is of interest to study (string) grammars which operate in parallel: that is, when a rule is applied to a string, every instance of the left member is replaced by the right member. (Here again, there are difficulties which can be avoided if all rules are isotonic and context-sensitive.) The sets of sentences which such a parallel grammar generates is not the same as the set of sentences which it parses, nor is either of these the same as the set of sentences generated (or parsed) when rules need not be applied in parallel. However, any parallel language is a sequential language and vice versa.

FOR FURTHER INFORMATION: The complete report is available in the major Navy technical libraries and can be obtained from the Defense Documentation Center. A few copies are available for distribution by the author.

1. Introduction

1.1 String grammars and array grammars

In recent years there has been considerable interest in applying the methods of mathematical linguistics to picture generation and description [1]. In this approach, pictures are regarded as "concatenations" of subpictures, which are in turn built up out of still smaller parts, in analogy with the way that sentences can be broken down into phrases and words.

In mathematical linguistics, the most widely used device for generating and analyzing "sentences" is the phrase structure grammar. Formally, such a grammar is a 5-tuple $G = (V, V_T, P, S, \#)$, where

1) V is a finite set, called the vocabulary of G; the elements of V are called symbols.
2) V_T is a subset of V, called the terminal vocabulary of G.
3) P is a finite set of pairs (α, β), where α and β are strings of elements of V, α nonnull; P is called the set of productions or rewriting rules of G. Elements of P are usually written in the form $\alpha \rightarrow \beta$ (read: "α can be rewritten as β"). Symbols in V_T are never destroyed by these rules; in other words, if $\alpha \rightarrow \beta$ is a rule and $\alpha = \xi_0\eta_1\xi_1\ldots\eta_n\xi_n$, where the ξ's are strings of elements of V_T and the η's are strings on $V-V_T$, then $\beta = \xi_0\zeta_1\xi_1\ldots\zeta_n\xi_n$, where the ζ's are strings on V.

4) S is a special nonterminal symbol (i.e., symbol in $V-V_T$), called the <u>initial symbol</u> of G.

5) # is a special terminal symbol, called the <u>endmarker</u>, which is neither created nor destroyed by any rule of G.

Let γ, δ be strings on V. We say that δ is <u>directly derivable</u> from γ in G (notation: $\gamma \Rightarrow \delta$) if $\gamma = \gamma_1 \alpha \gamma_2$ and $\delta = \gamma_1 \beta \gamma_2$, where $\alpha \rightarrow \beta$ is a rule of G. More generally, we say that δ is <u>derivable</u> from γ in G (notation: $\gamma \overset{*}{\Rightarrow} \delta$) if there exist strings $\theta_1, \ldots, \theta_n$ such that $\gamma \Rightarrow \theta_1 \Rightarrow \ldots \Rightarrow \theta_n \Rightarrow \delta$. A string on V_T is called a <u>sentence</u> of G if it is derivable from #S#. The set of sentences of G is called the <u>language</u> of G (notation: L(G)). [In applying this formalism to natural language, one can think of the terminal symbols as words, the nonterminal symbols as phrases, S as "sentence", and the endmarkers as punctuation marks which indicate the beginning and end of the sentence.] Readily, the sentences of G are just the strings of terminal symbols from which #S# can be derived by applying the rules in reverse (i.e., replacing right members by left members); this reverse procedure is called <u>parsing</u>.

Since a picture does not ordinarily have a natural description as a string of subpictures, phrase-structure grammars as such are not a natural tool for picture generation or analysis. In general, a (discrete, e.g., digital) picture is an <u>array</u> of elements having given colors or gray levels; these elements can be regarded as the symbols of a terminal vocabulary. A grammar for a language whose "sentences" are such pictures would have to have rules $\alpha \rightarrow \beta$

which rewrite arrays as arrays; to use such a rule to directly derive the array δ from the array γ, one would have to find the subarray α in γ and replace it by β. A grammar of this type, whose language consists of digital pictures of triangles, has been devised by Kirsch [2] and generalized by Dacey [3]; these seem to be the only examples of array-rewriting grammars in the literature.

1.2 Isotonic grammars

A potentially serious defect of array grammars is that when one subarray is replaced by another, the effects of the replacement may extend far beyond the immediate vicinity of the subarray. In the string case, when the substring α is replaced by β in the string γ, if α and β have different lengths, we simply regard γ as pushed apart or pulled together until β exactly fits the space left by removing α. For arrays, on the other hand, if we wish to replace one subarray by another of a different size or shape, the rows and columns of the host array may have to stretch or shrink by varying amounts, so that "shearing" effects occur which extend all the way out to the edges of the host array, arbitrarily far from the replaced subarray.

Most of the literature on picture grammars deals with line drawings rather than arrays. Here the lines and curves are regarded as joined only at explicitly specified "attaching points", irrespective of whether they actually touch or intersect in the picture. Thus replacement of one subdrawing by another affects the structure of the host drawing only locally, and the shearing problem never arises. Kirsch's right triangle grammar avoided the problem by adding to arrays only at their edges. However, it is not clear that this method could be used for arbitrary array languages.

A simple way of preventing shearing would be to require the left and right members of any array rewriting rule to be congruent. In the string case, the analogous requirement would be that the left and right members of any rule must have the same length. (A grammar whose rules have this property will be called <u>isotonic</u>.) However, this evidently implies that the strings in any derivation must all be of the same length, since no application of a rule can change the length of a string. Since sentences must all be derivable from a single initial S (surrounded by endmarkers, which are never created or destroyed), it follows that sentences can consist only of single symbols, which makes the language rather uninteresting.

This objection to isotonic grammars can be overcome if one is allowed to start not with a single initial S, but with any one of a <u>set</u> of initial strings -- for example, with an arbitrary string of S's (bordered by endmarkers). In fact, it can be shown that if this is permitted, isotonic grammars become exactly as powerful as <u>monotonic</u> grammars (i.e., grammars in which the right member of any rule is at least as long as the left member), which are the most general class of grammars ordinarily studied. It can also be shown that monotonic grammars which change the length of a string only at its ends are as powerful as arbitrary monotonic grammars; this generalizes the method used by Kirsch. These results are presented in Section 2 of this paper.

1.3 Parallel grammars

In digital picture processing, a local operation is one which replaces a given picture element by a new one whose value (if, as is usual, we regard the symbols in a picture array as numbers) depends on the value of the original element and on the values of a set of neighboring elements. This type of operation is analogous to an isotonic, context sensitive string rewriting rule --i.e., a rule of the form $\xi A\eta \rightarrow \xi B\eta$, where A and B are single symbols (A must be nonterminal, since terminals cannot be rewritten). It is well known that context sensitive grammars (i.e., grammars in which every rule is context sensitive) are exactly as powerful as monotonic string grammars; similarly, it is easily shown that isotonic context sensitive grammars are as powerful as arbitrary isotonic grammars (see the end of Section 2).

Local picture processing operations are often applied to every element of a picture "in parallel", i.e., using the original values of the element and its neighbors throughout, rather than using new values for neighbors which have already been processed. It has been shown elsewhere [4] that parallel local operations on pictures are exactly as powerful as "sequential" operations, in which the elements are processed in a fixed order, and in processing each element, new values are used for its already processed neighbors. In view of the analogy between local operations and rewriting rules, the analogous question for grammars is thus of interest: How is the power of an (isotonic, context sensitive) grammar affected if its rules are applied in parallel rather than sequentially -- i.e., when the rule $\xi A\eta \to \xi B\eta$ is applied to a string γ, we simultaneously replace _every_ A which occurs in the context $(\xi, ,\eta)$ by B, rather than just replacing one such A by B?

One could attempt to formulate the notion of parallelness for grammars having arbitrary rewriting rules $\alpha \to \beta$: when applying such a rule to a string γ, replace every instance of α in γ by β. However, if α can overlap itself, i.e., $\alpha = A_1 \ldots A_n$ where $A_{k+1} \ldots A_n = A_1 \ldots A_{n-k}$ for some $k < n$, undesirable effects arise. For example, let α = ABA and suppose that γ = ABABA. Since α occurs twice in γ, applying $\alpha \to \beta$ to γ "in parallel" should require us to rewrite γ as $\beta\beta$; but this means that, in effect, one of the β's replaces an ABA and the other replaces only an AB or a BA. As another example, let α = xxx, β = xx, $\gamma = x^m$. Applying $\alpha \to \beta$ to γ sequentially shortens γ by 1 at each application; but applying it in parallel yields $\beta^{m-2} = x^{2(m-2)}$ (where it will be noted that one β replaces an xxx while all the others replace single x's!), so that parallel application of this length-decreasing rule to γ can actually increase its length.

These difficulties do not arise if we restrict ourselves to context-sensitive rules, $\alpha = \xi A\eta$, and replace only the A by the appropriate part of β; here there can be no self-overlap of the substrings which are being replaced, since they have length 1. On the other hand, if we allowed arbitrary context-sensitive rules $\xi A\eta \to \xi\theta\eta$, where θ need not be just a single symbol, the same undesirable effects could arise when the rules are used to parse, since θ's might overlap. Thus parallelness is most easily handled if all rules are required to be isotonic context-sensitive, as suggested by the parallel picture processing analogy.

It will be shown in Section 3 that when a grammar is used "in parallel", the set of sentences which can be derived from #S# is not in general the same as the set obtained when the rules are used in the ordinary "sequential" manner. Moreover, the set of sentences which can be derived "in parallel" need not be the same as the set which can be parsed in parallel. However, we shall show that various classes of "parallel languages" are the same as the corresponding classes of "sequential languages".

2. Isotonic string grammars

In this section we show that isotonic grammars are as powerful as monotonic grammars. More precisely, we prove that if the language L has a monotonic grammar, it also has an isotonic grammar, provided that derivations are allowed to start with initial strings consisting of arbitrary numbers of repetitions of the initial symbol. To this end, we first prove

Proposition 1. Any (monotonic, isotonic) language has a (monotonic, isotonic) grammar in which no rule involves the endmarkers.

Proof: Let the given language have a grammar on the vocabulary $A_1, \ldots, A_n = S$ (where $A_1, \ldots, A_t$ are terminal and the rest nonterminal), and with rules of the form $(\#)B_1 \ldots B_r(\#) \to (\#)C_1 \ldots C_s(\#)$. We use the new vocabulary $A_1^O, \ldots, A_n^O, A_1^L, \ldots, A_n^L, A_1^M, \ldots A_n^M, A_1^R, \ldots A_n^R$, $A_1, \ldots, A_t$, with initial symbol A_n^O, and new rules constructed as follows:

Original rule	Replaced by rule(s)
$\#B_1\# \to \#C_1\#$	$B_1^O \to C_1^O$
$\#B_1\# \to \#C_1 \ldots C_s\#$ $(s > 1)$	$B_1^O \to C_1^L C_2^M \ldots C_{s-1}^M C_s^R$
$\#B_1 \ldots B_r\# \to \#C_1\#$ $(r > 1)$	$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \to C_1^O$
$\#B_1 \ldots B_r\# \to \#C_1 \ldots C_s\#$ $(r, s > 1)$	$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \to C_1^L C_2^M \ldots C_{s-1}^M C_s^R$
$\#B_1 \to \#C_1$	$B_1^L \to C_1^L$
	$B_1^O \to C_1^O$
$\#B_1 \to \#C_1 \ldots C_s$ $(s > 1)$	$B_1^L \to C_1^L C_2^M \ldots C_s^M$
	$B_1^O \to C_1^L C_2^M \ldots C_{s-1}^M C_s^R$
$\#B_1 \ldots B_r \to \#C_1$ $(r > 1)$	$B_1^L B_2^M \ldots B_r^M \to C_1^L$
	$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \to C_1^O$

Original rule	Replaced by rule(s)
$\#B_1 \ldots B_r \rightarrow \#C_1 \ldots C_s$ $(r, s > 1)$	$B_1^L B_2^M \ldots B_r^M \rightarrow C_1^L C_2^M \ldots C_s^M$
	$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \rightarrow C_1^L C_2^M \ldots C_{s-1}^M C_s^R$
$B_1\# \rightarrow C_1\#$	$B_1^R \rightarrow C_1^R$
	$B_1^O \rightarrow C_1^O$
$B_1\# \rightarrow C_1 \ldots C_s\#$ $(s > 1)$	$B_1^R \rightarrow C_1^M \ldots C_{s-1}^M C_s^R$
	$B_1^S \rightarrow C_1^L C_2^M \ldots C_{s-1}^M C_s^R$
$B_1 \ldots B_r\# \rightarrow C_1\#$ $(r > 1)$	$B_1^M \ldots B_{r-1}^M B_r^R \rightarrow C_1^R$
	$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \rightarrow C_1^O$
$B_1 \ldots B_r\# \rightarrow C_1 \ldots C_s\#$ $(r, s > 1)$	$B_1^M \ldots B_{r-1}^M B_r^R \rightarrow C_1^M \ldots C_{s-1}^M C_s^R$
	$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \rightarrow C_1^L C_2^M \ldots C_{s-1}^M C_s^R$
$B_1 \rightarrow C_1$	$B_1^M \rightarrow C_1^M$
	$B_1^L \rightarrow C_1^L$
	$B_1^R \rightarrow C_1^R$
	$B_1^O \rightarrow C_1^O$
$B_1 \rightarrow C_1 \ldots C_s$ $(s > 1)$	$B_1^M \rightarrow C_1^M \ldots C_s^M$
	$B_1^L \rightarrow C_1^L C_2^M \ldots C_s^M$
	$B_1^R \rightarrow C_1^M \ldots C_{s-1}^M C_s^R$
	$B_1^O \rightarrow C_1^L C_2^M \ldots C_{s-1}^M C_s^R$
$B_1 \ldots B_r \rightarrow C_1$ $(r > 1)$	$B_1^M \ldots B_r^M \rightarrow C_1^M$
	$B_1^L B_2^M \ldots B_r^M \rightarrow C_1^L$
	$B_1^M \ldots B_{r-1}^M B_r^R \rightarrow C_1^R$
	$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \rightarrow C_1^O$

Original rule		Replaced by rule(s)
$B_1 \ldots B_r \to C_1 \ldots C_s$	$r, s > 1$	$B_1^M \ldots B_r^M \to C_1^M \ldots C_s^M$
		$B_1^L B_2^M \ldots B_r^M \to C_1^L C_2^M \ldots C_s^M$
		$B_1^M \ldots B_{r-1}^M B_r^R \to C_1^M \ldots C_{s-1}^M C_s^R$
		$B_1^L B_2^M \ldots B_{r-1}^M B_r^R \to C_1^L C_2^M \ldots C_{s-1}^M C_s^R$

In addition, the new grammar has the rules

$$A_i^O \to A_i \qquad A_i^M \to A_i$$

$$A_i^L \to A_i \qquad A_i^R \to A_i$$

for $1 \le i \le t$. (Note that the new grammar may block if these rules are applied too early.) Readily, the new grammar and the original one have the same language; and if one is monotonic or isotonic, so is the other. //

We can now prove

Theorem 2. Let L be any language having a monotonic grammar; then there exists an isotonic grammar on the set of initial strings $\{T^k \mid k = 1,2,\ldots\}$ whose language is exactly L.

Proof: Let G be a monotonic grammar for L having vocabulary $V = \{S, A_1, \ldots, A_m, b_1, \ldots, b_n\}$ and rewriting rules $\alpha_i \to \beta_i$ $(1 \le i \le k)$, where $0 < |\alpha_i| \le |\beta_i|$, $1 \le i \le k$. We define an isotonic grammar G' for L using the vocabulary $V' = \{T, U, V, A_1, \ldots, A_m, B_1, \ldots, B_n, b_1, \ldots, b_n\}$. For any string γ on V, let γ' be the string on V' defined by replacing S by U and the b's by B's in γ. Then we can take the rules of G' to be

$\#T \rightarrow \#U$

$T \rightarrow V$

$UV \rightarrow VU$

$A_iV \rightarrow VA_i \quad (1 \leq i \leq m)$

$B_iV \rightarrow VB_i \quad (1 \leq i \leq n)$

$\alpha_i V^{|\beta_i|-|\alpha_i|} \rightarrow \beta_i' \quad (1 \leq i \leq k*)$

$B_i \rightarrow b_i \quad (1 \leq i \leq n)$

We first show that every sentence σ of L is a sentence of L(G'). Let

$S = \sigma_0 \Rightarrow \sigma_1 \ldots \Rightarrow \sigma_r = \sigma$

be a derivation of σ in G, where the rule used to rewrite σ_{i-1} as σ_i is $\alpha_{n_i} \rightarrow \beta_{n_i}$; we shall abbreviate $|\beta_{n_i}| - |\alpha_{n_i}|$ by k_i, $1 \leq i \leq r$. Note that $|\sigma_i| - |\sigma_{i-1}| = k_i$, $1 \leq i \leq r$, so that $|\sigma| = |\sigma_r| = |\sigma_{r-1}| + k_r = \ldots = |\sigma_0| + k_r + \ldots + k_1$, where by monotonicity the k's are all nonnegative. To derive σ in G', we begin with the initial string $T^{|\sigma|}$ and use the rules $\#T \rightarrow \#U$, $T \rightarrow V$ to obtain the string $UV^{|\sigma|-1}$. Since this string has at least k_1 V's, and since $\alpha_{n_1}' = U$ or $\#U$, we can apply the rule $\alpha_{n_1}' V^{k_1} \rightarrow \beta_{n_1}'$ to obtain a new string, call it τ_1. Evidently, τ_1 is just σ_1' followed by at least k_2 V's; in particular, τ_1 contains a copy of α_{n_2}'. If we use the rules which shift V's to the left, we can move k_2 of them until they just follow this copy, and then apply the rule $\alpha_{n_2}' V^{k_2} \rightarrow \beta_{n_2}'$ to obtain a new string τ_2.

*This assumes that no rule of G involves the right endmarker, which is a legitimate assumption by Proposition 1.

This argument can be repeated, so that eventually we obtain a string τ_r which evidently is just $\sigma'_r = \sigma'$. Applying the rules $B_i \to b_i$ to σ' then gives us the desired string σ.

Conversely, let $T^{|\rho|} = \rho_0 \Rightarrow \rho_1 \Rightarrow \ldots \Rightarrow \rho_s = \rho$ be a non-blocking derivation in G'. Without loss of generality, we may assume that $T^{|\rho|}$ is first rewritten into $UV^{|\rho|-1}$. Let $\rho_{u_i-1} \Rightarrow \rho_{u_i}$ $(1 \le i \le p)$ be the steps at which rules of the form $\alpha'_{v_i} V^{w_i} \to \beta'_{v_i}$ are used. We shall show how to construct a derivation for ρ in G. Specifically, we shall construct a G-derivation $S = \nu_0 \Rightarrow \nu_1 \Rightarrow \ldots \Rightarrow \nu_p = \rho$ in which $\nu_i = \rho_{u_i}$ with U's replaced by S's and B's by b's, and ignoring V's, $1 \le i \le p$ (and similarly $\nu_0 = \rho_{u_1-1}$ with the same changes). Note first that ρ_{u_1-1} can contain no A's, B's or b's, so that the rule used to rewrite it as ρ_{u_1} must be of the form $UV^{w_1} \to \beta'_{v_1}$ (or $\#UV^{w_1} \to \#\beta'_{v_1}$); let us take $\nu_1 = \beta_{v_1}$ in our G-derivation. Thus ν_1 and ρ_{u_1} satisfy our requirements. Suppose that ν_{i-1} and $\rho_{u_{i-1}}$ do so. Now ρ_{u_1-1} must contain a copy of α'_{v_i}, and since the other rules of G' cannot change the order of U's, A's, B's or b's, it follows that (ignoring V's) $\rho_{u_{i-1}}$ must also contain such a copy. Thus by induction hypothesis, ν_{i-1} contains a copy of α_{v_i}; apply the rule $\alpha_{v_i} \to \beta_{v_i}$ to this copy, and call the resulting string ν_i. Evidently, ν_i and ρ_{u_i} satisfy our requirements; thus in particular, ν_p and ρ_{u_p} do. But since ρ_{u_p} is the last step of the given G'-derivation at which U's, V's or A's can be eliminated, and since this derivation is non-blocking, we see that ρ_{u_p} must consist entirely of B's and b's, and that $\rho = \rho_{u_p}$ with B's (if any) replaced by b's; thus ν_p must be just ρ. //

Conversely, let G', as in the proof of Theorem 2, be an isotonic grammar which, given the set of initial strings $\{T^k \mid k = 1, 2, \ldots\}$, yields the language L. Let G be the grammar defined by adding a new initial symbol S to the vocabulary of G', and the new rules $S \rightarrow ST$, $S \rightarrow T$ to the rules of G'. Evidently G is monotonic and $L(G) = L$. Note also that G is isotonic except at the left end of a string, which is the only place that the rule $S \rightarrow ST$ can ever apply. We have thus proved

<u>Theorem 3</u>. The following three classes of grammars have the same set of languages:

a) The monotonic grammars, with initial string #S#

b) The monotonic grammars which are isotonic except at the ends of strings, with initial string #S#

c) The isotonic grammars, with set of initial strings $\{\#S^k\# \mid k = 1, 2, \ldots\}$.

It is not difficult to verify that the isotonic grammars are also equivalent to the isotonic context sensitive grammars, i.e., grammars in which every rule is of the form $\xi A \eta \rightarrow \xi B \eta$. Indeed, if $A_1 \ldots A_m \rightarrow B_1 \ldots B_m$ is the kth rule in an isotonic grammar (which we can assume not to involve endmarkers) we can replace it by the rules $A_1 \ldots A_m \to A_1^{(k)} A_2 \ldots A_m$; $A_i^{(k)} A_{i+1} = A_i^{(k)} A_{i+1}^{(k)}$, $1 \le i \le m$; $A_i^{(k)} A_{i+1}^{(k)} \to B_i^{(k)} A_{i+1}^{(k)}$, $1 \le i < m$; and $B_{m-1} A_m^{(k)} \to B_{m-1} B_m$. Since the special nonterminals $A_1^{(k)}, \ldots, A_m^{(k)}$ are only created or rewritten by these rules, the only way that they can be eliminated once the $A_1 \ldots A_m$ has been rewritten is to finish rewriting it as $B_1 \ldots B_m$.

3. Parallel string grammars

Even for isotonic context-sensitive grammars, the sets of sentences generated and parsed when the grammar is used "in parallel" is not the same as the set generated or parsed when the grammar is used "sequentially". In fact, these sets can even be disjoint, as shown by the following example. Let the initial string be #XXXX#, and let the rules be

XXX → XYX	#XX → #cX
XXY → XaY	XX# → Xc#
XYY → XbY	cXc → ccc
aX → aa	bX → bb
Xa → aa	Xb → bb
aY → aa	bY → bb

It is not hard to verify that the "sequential language" of this grammar consists of the single sentence #aaaa#, while its "parallel languages" consist of the single sentences #bbbb# (generated) and #cccc# (parsed), respectively. [If we were not restricted to context-sensitive grammars, a simpler example would be provided by the grammar whose sole rule is XX → aa; readily, from the initial string #XXXX# this yields the sequential language {#aaaa#}, but its "parallel languages" are {#aaaaaa#} (generated) and {#aaa#} (parsed), respectively.]

In spite of the fact that the sequential and parallel languages of a given grammar need not be the same, we can show that various classes of such languages are the same. Here we shall not restrict ourselves to isotonic context-sensitive grammars. We begin by showing that for any grammar G, there exists a grammar G' such that L(G') = L(G), and such that in any step of any derivation of a sentence using G', no rule can apply in more than one place. It follows that the parallel languages of G' -- by any definition -- must be the same as its sequential language L(G). Moreover, if G is monotonic or isotonic, so is G'. Thus any (monotonic, isotonic) sequential language is a (monotonic, isotonic) parallel language.

Theorem 4. For any grammar G, there exists a grammar G' such that L(G') = L(G), and where at any step of any G'-derivation, no rule applies at more than one place. Moreover, if G is monotonic or isotonic, so is G'.

Proof: We may assume that no rule of G involves the endmarkers. In every rule of G, replace each terminal x by a new nonterminal $\bar{x}$, and then replace the leftmost symbols A,B in the left and right members of the rule by new nonterminals A*,B*. (This assumes that the right member of any rule of G is nonnull, which is certainly true if G is monotonic or isotonic; the contrary case will be treated later.)

Also add the new rules

(1) $\#S\# \rightarrow \#S^*\#$

(2) $A^*B \rightarrow AB^*$ | $AB^* \rightarrow A^*B$ — for all pairs of nonterminals A,B, one starred and the other not

(3) $\bar{x}^*\bar{y} \rightarrow x\bar{y}^*$ | $\bar{x}^*\# \rightarrow x\#$ — for all terminals x,y

Readily, any derivation in G corresponds to a derivation in this new grammar G': initially, the S is changed to S^*, and the modified rules of G in conjunction with rules (2) can then be used to yield any string in L(G), but with bars on its symbols and a star on its first symbol. Rules (3) can then be used to erase the bars and star. Note that G' may block if this is done too soon. Conversely, a derivation in G' can yield a terminal string only by eliminating all nonterminals except the barred ones, which can only be done using the modified rules of G, so that the resulting terminal string must be in L(G). Thus L(G') = L(G).

Clearly, at any step in any nonblocking derivation in G' except the first and last, there is exactly one starred symbol. Thus no rule of G' can ever apply to a string in such a derivation at more than one place. Moreover, since rules (1-3) are all isotonic, if G was monotonic or isotonic, so is G'.

The treatment is analogous if G is isotonic and uses the set of initial strings $\{\#T^k\# \mid k = 1,2,\ldots\}$, except that (1) is replaced by $\#T \to \#U^*$. Note also that if G is context-sensitive, (2-3) can be modified to insure that G' is context-sensitive (e.g., replace $A^*B \to AB^*$ by $A^*B \to A^*B' \to A'B' \to A'B^* \to AB^*$).

If G can have rules with null right members, we can use a slightly different trick to insure that no rule applies in more than one place. Let Z be a new nonterminal, and replace each rule $\alpha \to \beta$ of G by the rule $Z\bar{\alpha} \to Z\bar{\beta}$ (where the bars indicate that terminals have been replaced by barred nonterminals). Also add the new rules

1) #S# $\rightarrow$ #ZS#

2) AZ $\rightarrow$ ZA   for all nonterminals A

   ZA $\rightarrow$ AZ

3) Z$\bar{x}$ $\rightarrow$ xZ

   Z# $\rightarrow$ #. //

Note that if L is finite state (or more generally, linear), it has a linear grammar G, which implies that at any step of any derivation of a sentence in L, only one nonterminal is present. In particular, no rule can apply except to that nonterminal. Thus the analog of Theorem 4 holds with "monotonic" replaced by "finite state" or "linear". (On the analogous questions for the context free case, see the end of this section.)

Our final goal is to show that for any grammar G there exists a grammar G* which, in effect, applies the rules of G in parallel, so that L(G*) is the same as the parallel language generated by G; thus any parallel language is a sequential language. Here again, we need not restrict ourselves to any particular definition of "parallel language". The proof given below assumes that $\alpha \rightarrow \beta$ is applied to γ by replacing every instance of α in γ by β; an analogous proof can be given for the context sinsitive version of "parallel" in which $\xi A \eta \rightarrow \xi \theta \eta$ is applied to γ by replacing every A in γ by θ.

Let the vocabulary of G be $\{S = A_1, \ldots, A_n, \#\}$, and its rules be $\{\alpha_i \rightarrow \beta_i \mid 1 \le i \le m\}$, where $\alpha_i = A_{i1} \ldots A_{ir_i}$, $\beta_i = A_{ir_i+1} \ldots A_{is_i}$. We define G* to have vocabulary $\{(a,b,c,d) \mid a = A_1, \ldots, A_n,\ b = 0, A_1, \ldots, A_n,\ c = 0, \ldots, m$

or 1*,...,m*, d = 0,M,N,M*,N* or P} and initial symbol (S,0,0,0), and to have the following rules:

1) $\#(a_1,0,0,0)\ldots(a_{r_i},0,0,0) \rightarrow \#(a_1,A_{i_1},i^*,0)$
$(a_2,A_{i_2},i,0)\ldots(a_{r_i},A_{ir_i},i,0)$, $1 \le i \le m$

These rules copy the left member of a rule of G into the second terms of the left end of a string, and the rule number into the third terms; the first element of the rule has its rule number starred. <u>In all of the following rules, a, b, c are A's</u>:

2) $(a,b,i^*,0) \rightarrow (a,b,i^*,M)$ if $a = b$
$(a,b,i^*,0) \rightarrow (a,b,i^*,N)$ if $a \neq b$
(a,b,i or i*, M) (c,d,i,0) → (a,b,i or i*,M) (c,d,i,M)
if c = d, or with the last M replaced by N if c ≠ d
(a,b,i or i*,N) (c,d,i,0) → (a,b,i or i*,N) (c,d,i,N)

These rules test to see whether the left member of the rule matches the corresponding string of first terms, starting from the left end. M's are created as long as the match continues, N's otherwise.

3) $(a,b,i^*,M)\ (c,0,0,0) \rightarrow (a,0,i,P)\ (c,b,i^*,0)$
$(a,b,i,M)\ (c,0,0,0) \rightarrow (a,0,i,M)\ (c,b,i,0)$
$(a,b,i,M)\ (c,o,i,M) \rightarrow (a,0,i,M)\ (c,b,i,0)$
$(a,b,i^*,M)\ (c,0,i,M) \rightarrow (a,0,i,P)\ (c,b,i^*,0)$

If a match reaches the right end of the string of second terms, the string is shifted one step to the right, and a P is created at its former left end, while the M's and N's are erased.

4) (a,b,i*,N) (c,0,0,0) → (a,0,0,0) (c,b,i*,0)
(a,b,i,N) (C,0,0,0) → (a,0,i,N) (c,b,i,0)
(a,b,i,N) (c,0,i,N) → (a,0,i,N) (c,b,i,0)
(a,b,i,M) (c,0,i,N) → (a,0,i,N) (c,b,i,0)
(a,b,i*,N) (c,0,i,N) → (a,0,0,0) (c,b,i*,0)
(a,b,i*,M) (c,0,i,N) → (a,0,0,0) (c,b,i*,0)

Otherwise, the string is shifted and the M's and N's erased, but no P is created. In either case, once the i* is shifted, (2-4) can now be repeated. Note that when it has shifted sufficiently, and there are no P's left behind, (1) can be re-initiated too.

5) (a,b,i*,M)# → (a,0,i,P)#
(a,b,i,M)# → (a,0,0,M*)#
(a,b,i,M) (c,0,0,M*) → (a,0,0,M*) (c,0,0,0)
(a,b,i*,M) (c,0,0,M*) → (a,0,i,P) (c,C,0,0)

6) (a,b,i*,N)# → (a,0,0,0)#
(a,b,i,N)# → (a,0,0,N*)#
(a,b,i,N) (c,0,0,N*) → (a,0,0,N*) (c,0,0,0)
(a,b,i,M) (c,0,0,N*) → (a,0,0,N*) (c,0,0,0)
(a,b,i*,N) (c,0,0,N*) → (a,0,0,0) (c,0,0,0)
(a,b,i*,M) (c,0,i,N*) → (a,0,0,0) (c,0,0,0)

When the right endmarker is reached, the string of second terms is erased, M's and N's are erased, and a P created if what reached the right endmarker was an M. Note also that (3-4) turned all the i's to the left of the i* into 0's, and left i's only where there are P's; now (5-6) have turned all the i's to the right of the i* into 0's, and have turned the i* into i if it has a P, into 0 otherwise.

7) $(a_1,0,i,P)\ (a_2,0,0,0)\ldots(a_{r_i},0,0,0) \rightarrow (a_{ir_i+1},0,0,0)\ldots$
$(A_{is_i},0,0,0)$, $(1 \leq i \leq m)$;
$(a_1,0,i,P)\ (a_2,0,i,0)\ldots(a_k,0,i,0)\ (b,0,i,P) \rightarrow$
$(A_{ir_i+1},0,i,0)\ldots(A_{is_i},0,i,0)\ (b,0,i,P)$, $(1 \leq i \leq m,\ 1 \leq k < r_i)$

These rules replace each run of up to r_i 4-tuples which starts with P, contains no other P, and (if shorter than r_i) is immediately followed by a P, by 4-tuples containing the right member of the ith rule. Even if a new rule left member has been copied in, its processing cannot cross P's; i.e., one rule's P's must be processed before a second rule can move into the same part of the string.

In summary: Rules (1-7) of G* cause the ith rule of G to be applied to the given string "in parallel".

8) $\#(a,0,0,0) \rightarrow \#a$
$a(b,0,0,0) \rightarrow ab$

These rules turn the string of 4-tuples into a string of symbols of G's vocabulary; unless these are all terminals, the derivation blocks.

Note that if G is monotonic or isotonic, so is G*. To modify the proof to handle the context sensitive type of parallelness, we would need to mark the symbol to be re-written (A, in $\xi A\eta$) rather than the leftmost symbol.

The analog of these results in the context free case is false -- a context free parallel language is not necessarily a context free sequential language. For example, the context free grammar with rules $S \rightarrow TT$, $T \rightarrow aT$, $T \rightarrow bT$, $T \rightarrow a$, $T \rightarrow b$, if applied in parallel, generates the set of sentences $\omega\omega$, where ω is any string of a's and b's; as is well known, this is not a context free sequential language. It is an open question whether conversely, any context free sequential language is a context free parallel language.

REFERENCES

1. W. F. Miller and A. C. Shaw, Linguistic methods in picture processing - a survey, Proc. 1968 Fall Joint Comp. Conf., 279-290.

2. R. A. Kirsch, Computer interpretation of English text and picture patterns, IEEE Trans. EC-13, August 1964, 363-376.

3. M. F. Dacey, The syntax of a triangle and some other figures, Pat. Recog. 2, January 1970, 11-31.

4. A. Rosenfeld and J. L. Pfaltz, Sequential operations in digital picture processing, JACM 13, October 1966, 471-494.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* University of Maryland	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b GROUP n.a.

3. REPORT TITLE

Isotonic grammars, parallel grammars and picture grammars

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report

5. AUTHOR(S) *(Last name, first name, initial)*

Rosenfeld, Azriel

6. REPORT DATE April 1970	7a. TOTAL NO. OF PAGES 24	7b. NO. OF REFS 4
8a. CONTRACT OR GRANT NO. Nonr-5144(00)	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Report 70-111	
b. PROJECT NO. n.a.		
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	
d.		

10. AVAILABILITY/LIMITATION NOTICES

n.a.

11. SUPPLEMENTARY NOTES n.a.	12. SPONSORING MILITARY ACTIVITY Information Systems Branch Office of Naval Research Washington, D. C.

13. ABSTRACT When one attempts to generalize phrase-structure grammars from strings to arrays, difficulties arise which can be avoided if the grammars are required to be isotonic: in any array rewriting rule, the left and right members are congruent subarrays. For strings, such grammars are exactly as powerful as monotonic grammars, provided that derivations can begin with any initial string of the form $\#S^k\#$ rather than always with #S#.

Isotonic context-sensitive array rewriting rules are essentially the same as local digital picture processing operations. Since the latter are often applied to pictures in parallel, it is of interest to study (string) grammars which operate in parallel: that is, when a rule is applied to a string, every instance of the left member is replaced by the right member. (Here again, there are difficulties which can be avoided if all rules are isotonic and context-sensitive.) The sets of sentences which such a parallel grammar generates is not the same as the set of sentences which it parses, nor is either of these the same as the set of sentences generated (or parsed) when rules need not be applied in parallel. However, any parallel language is a sequential language and vice versa.

DD FORM 1 JAN 64 1473

Security Classification