AFOSR TN 59-110
ASTIA AD No. 210 390

PROJECT No. 9769
TASK No. 37659

ZTB-124

A NEW METHOD FOR DISCOVERING THE GRAMMARS OF PHRASE STRUCTURE LANGUAGES

R. J. Solomonoff

APRIL 1959

Paper to be presented at the International Conference on Information Processing, UNESCO, Paris, France, June 13-23, 1959.

CONTRACT AF 49(638)-376

PREPARED FOR

OFFICE OF SCIENTIFIC RESEARCH
AIR RESEARCH AND DEVELOPMENT COMMAND
UNITED STATES AIR FORCE

WASHINGTON 25, D. C.

ZATOR COMPANY

140½ MOUNT AUBURN STREET, CAMBRIDGE 38, MASS.

ABSTRACT

A NEW METHOD FOR DISCOVERING THE GRAMMARS OF PHRASE STRUCTURE LANGUAGES

R. J. SOLOMONOFF

Zator Company, 140½ Mt. Auburn St., Cambridge, Mass., U. S. A.

A new method has been devised for the discovery of the grammars of phrase structure languages. The method has applications in information retrieval, linguistics, pattern discovery and a kind of mechanical translation. A technique is used that is similar to one described by Chomsky and Miller for discovering the grammars of finite state languages.

In finite state languages, a phrase that forms a "cycle" may be successively repeated an arbitrary number of times in an acceptable sentence, because the insertion of that phrase does not change the state that exists at the point of insertion. In phrase structure languages, the corresponding entity that forms a cycle is an ordered pair of phrases. If the first phrase of such a pair be inserted before, and the second phrase be inserted after a suitable type of single phrase in an acceptable sentence, then the sentence will remain acceptable. The insertions may be repeated an arbitrary number of times, because the process does not change the phrase type of the single phrase.

The set of all such cycles and higher order cycles that exist in a phrase structure language along with the insertion rules, constitute a complete grammatical description of the language. These cycles are discovered by a systematic process of deletion and reinsertion of phrases and pairs of phrases. A "teacher" or equivalent, is used to determine if the sentences resulting from the deletions and reinsertions are acceptable sentences.

ZTB
124

A NEW METHOD FOR DISCOVERING THE GRAMMARS OF PHRASE STRUCTURE LANGUAGES

R. J. SOLOMONOFF

Zator Company, 140½ Mt. Auburn St., Cambridge, Mass., U. S. A.

1. INTRODUCTION

This paper describes a method for discovering the grammar of an arbitrary phrase structure language[1, 2, 3] in a suitable training situation. A method for grammar discovery of finite state languages has been devised by Chomsky and Miller.[2] A method for grammar discovery in phrase structure languages was described by Solomonoff.[4] The method described here is considerably different from Solomonoff's earlier method, and is very similar to that used by Chomsky and Miller for finite state languages. The correspondence between the discovery methods for finite state and phrase structure languages will be brought out in the present paper.

Though the problem being dealt with has already been solved by another method, its importance justifies the investigation of several possible solutions. It is likely that in certain cases one of the methods will require a smaller training sequence than the other, and therefore have some advantages in these types of problems.

The grammar discovery problem has been solved only for training situations in which there exists a "teacher" who will answer questions on the acceptability of proposed sentences devised by the "learner". The problem of grammar discovery in which no teacher is present has not been solved in any practical way – though a theoretical solution exists which involves a prohibitive amount of search time. It is hoped that the new method for phrase structure grammar discovery may be of some aid in devising a solution to the problem in which no teacher is available.

Also, the problem of grammar discovery for transformational languages has not yet been solved, and the present method for phrase structure languages sheds some light on this more difficult problem. By suitably generalizing the concept of phrase structure grammar, it is possible to solve some essentially transformational languages by the present technique.

2. THE APPLICATIONS OF GRAMMAR DISCOVERY METHODS.

The importance of grammar discovery stems from the applicability of finite state and phrase structure grammars to a large variety of problems. Chomsky and Miller first used the grammar discovery problem as a tool to study the learning of new concepts by human subjects.[2]

The fact that English is expressible as a phrase structure language – though not in a very economical way[1] – makes phrase structure languages important in linguistic studies.

It has been shown that in certain cases, the problem of learning to translate from one phrase structure language to another is identical with the problem of learning the grammar of a third phrase structure language of a more general sort.[4]

Some problems in information retrieval can be viewed as grammar discovery problems. For example, we may view all the titles of papers about chemistry as sentences in some "language". Our grammar discovery routine would, ideally, be able to devise a set of grammar rules for the language, and use them to determine whether a new paper was about chemistry, by analysis of the title of the paper. The practicality of such a scheme depends on our development of "approximate languages", and our ability to devise "teacherless" grammar discovery schemes for transformational languages.

By a suitable generalization process, it is possible to view many multidimensional patterns as phrase structure languages. Being able to discover grammar rules for such a language is then equivalent to a limited kind of multidimensional pattern discovery.

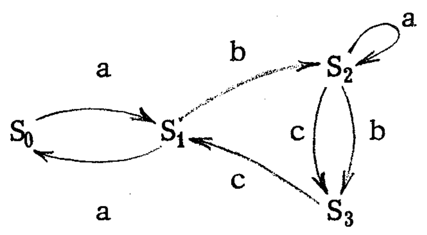
3. THE METHOD OF CHOMSKY AND MILLER FOR DISCOVERING THE GRAMMARS OF FINITE STATE LANGUAGES.

Before describing our system for discovering the grammars of phrase structure languages, we will first briefly describe a similar system that was devised by Chomsky and Miller[2] for finite state languages.

We may view a sentence of a finite state language as an ordered sequence of symbols, with certain unwritten states that occur before the first symbol, between successive symbols, and after the last symbol. Such a sentence might be

$$\begin{array}{ccccccccccc} & a & & b & & c & & c & & a & \\ \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow \\ S_0 & & S_1 & & S_2 & & S_3 & & S_1 & & S_0 \end{array}$$

The sentence itself is a b c c a. The states are $S_0, S_1, S_2, S_3, S_1, S_0$. Here we are using a grammar with the following state diagram:



In the sentence a b c c a, we return to state S_1 after having passed through it once. For this reason the phrase b c c, which goes between the two occurrences of S_1, can be removed, and the resultant string of symbols a a, will still be an acceptable sentence.

We can look upon the string b c c as a "cycle" on the state S_1. If we like, we can repeat this cycle as many times as we like in a suitable context – and the resultant string will still be an acceptable sentence. Thus a a, a b c c a, a b c c b c c a, a b c c b c c b c c a, etc., are all acceptable sentences.

ZTB 124

In the above example, the state S_1 has more than one possible cycle on it. We may thus form the sentences a a, a b b c a, a b b c b b c a, a b b c b b c b b c a, etc. Also, we may mix these two cycles if we like, first using one, and then the other – e.g. a b c c b b c a.

The cycles may in turn have cycles upon them. In the above example, the state S_2 has a cycle "a" upon it. This means that the cycle b c c may have the cycle "a" inserted in it an arbitrary number of times – so b a c c, b a a c c, b a a a c c are all cycles on S_1, and therefore, a b a c c a, a b a a c c a and a b a a a c c a, are all acceptable sentences.

Now suppose that we are given a large set of acceptable sentences from some finite state language, and we are asked to discover a set of grammar rules that could generate the set. We are also given a teacher. If we have any idea of what the grammar is, we are permitted to devise a trial string of symbols, and ask the teacher if it is an acceptable sentence. This is the only permissible type of question.

The strategy devised by Chomsky and Miller consists first of deleting parts of the known acceptable sentences, and asking the teacher if the remaining string is acceptable. In the above example, if a b c c a is one of the known acceptable sentences, then the teacher would be asked if b c c a, a c c a, a b c a, a b c c, c c a, a c a, a b a, a b c, c a, a a, a b, and a were acceptable sentences. Of these, only a a is acceptable. The part omitted, b c c, is reinserted and repeated several times, so the teacher is asked if a b c c b c c a, and a b c c b c c b c c a are acceptable. Since they are, we may conclude that b c c is a cycle that may be inserted between the a's of a a to form acceptable sentences.

By continuing this routine of deletion and repeated reinsertion, we will eventually discover all of the cycles and the cycles upon them, etc. These cycles will then be sufficient to generate any acceptable sentence in the language. This routine have been applied by Chomsky and Miller to finite state languages only. The method that will be described is an extension of these techniques to phrase structure languages.

4. PHRASE STRUCTURE GRAMMARS

This section is a brief review of some properties and definitions relevant to phrase structure languages.

One way to describe a phrase structure language is by means of a set of initial strings of symbols, Σ, and a set of permissible substitutions, F.

An example is

$$
\begin{aligned}
\Sigma &: S,\ Z \\
F &: S \rightarrow a\ b \\
&\quad Z \rightarrow c\ a \\
&\quad a \rightarrow c\ b \\
&\quad a \rightarrow d \\
&\quad b \rightarrow a\ f \\
&\quad b \rightarrow e
\end{aligned}
$$

A derivation of an acceptable sentence is

1. S
2. a b
3. c b b
4. c b a f
5. c e a f
6. c e d f

We may write this as a "tree":

$$\begin{tikzcd}
 & & & S \arrow[dll] \arrow[drr] & & & \\
 & a \arrow[dl] \arrow[dr] & & & & b \arrow[dl] \arrow[dr] & \\
c & & b \arrow[d] & & a \arrow[d] & & f \\
 & & e & & d & &
\end{tikzcd}$$

The "terminal vocabulary" of the language, from which all acceptable sentences are composed, consists of c, d, e and f. In the present case, our sentence is c e d f. We will call the phrase c e, a "phrase of type a

ZTB
124

in the sentence c e d f." The reason is that there exists a method of deriving this sentence (i.e. the method used above) in which the phrase c e is entirely derived by substitutions from the non-terminal symbol, a.

For similar reasons, in the sentence c e d f, d f is a phrase of type b, e is a phrase of type b, d is a phrase of type a, and c e d f is a phrase of type S.

Given F_1 and Σ_1 for a particular language, L_1, to find all phrases of a particular type, x, we devise a new language, L_2, with F_2 the same as F_1, but with Σ_2 containing only x. If A is a phrase of type x in some sentence S_{1j} of L_1, then A is an acceptable sentence in L_2. Conversely, for each acceptable sentence, A, in L_2, there exists at least one acceptable sentence, S_{1j}, in L_1, such that A is a phrase of type x in S_{1j}. However, if A is an acceptable sentence in L_2, and A occurs as a phrase in an acceptable sentence S_{1j} of L_1, then A <u>need</u> not be a phrase of type x (or of any other type) in S_{1j}.

In general, if α is a phrase of type x in sentence S_1 of a certain phrase structure language, and β is a phrase of type x in the same sentence S_1, or in a different sentence of the same language, then phrase β may be substituted for phrase α in the sentence S_1, and will thereby preserve the acceptability of the resultant sentence. We will use this fact later.

The derivation of a particular sentence may be also indicated by parenthesis. In the previous example, the sentence c e d f can be written [(c(e)) ((d) f)]. Each phrase of any particular type is enclosed by a pair of parentheses. Thus e d f is <u>not</u> a phrase of any particular "type".

In phrase structure languages, it is possible to have "cycles". Since all finite state languages are also phrase structure languages, the cycles of finite state languages will be special cases of cycles that occur in phrase structure languages.

ZTB
124

As an example of a cycle, consider the derivation,

1. S
2. (a b)
3. [(c b) b]
4. [(c{a f}) b]
5. [(c{(d) f}b]
6. [(c{(d) f}) (e)]

$$\begin{tikzcd}
 & & & S \arrow[dll] \arrow[drr] & & \\
 & a \arrow[dl] \arrow[dr] & & & & b \arrow[d] \\
c & & b \arrow[dl] \arrow[drr] & & & e \\
 & a \arrow[d] & & & f & \\
 & d & & & &
\end{tikzcd}$$

The parentheses have been carried along in the derivation. Note that d is a phrase of type a in sentence c d f e. Also c{(d) f} is a phrase of type a, in sentence c d f e.

Since we can substitute phrases of the same type for one another, c [c d f] f e is an acceptable sentence, since we have merely substituted c d f for d in the sentence c d f e. Similarly, c c c d f f f e is acceptable, as is c c c c d f f f f e.

Since surrounding a phrase of type a, by c on the left and f on the right, leaves this phrase still of type a, we will say that "(c, f) forms a cycle about phrases of type a." This cycle leaves the phrase type in a phrase structure language invariant, just as a cycle in a finite state language leaves the state invariant.

In general, a cycle of a certain phrase type need not ostensibly surround its phrase. For example, if α is a phrase of a certain type, then α a b, α a b a b, α a b a b a b may be all phrases of that type. In this case, (U, a b) forms a cycle about phrases of that type. U is the identity. $U\,x \equiv x\,U \equiv x$ for any symbol, x.

Cycles may have cycles on them, in turn, just as in finite state languages. The manner in which this may occur will become clear in the description of the phrase structure grammar discovery method.

It can be shown that in a language defined by a grammar with a particular set of substitution rules, there are only a finite number of acceptable

ZTB
124

sentences that have no cycles on them, and that there are only a finite number of cycles that have no cycles on them.

5. GRAMMAR DISCOVERY OF PHRASE STRUCTURE LANGUAGES

As in the problem of discovering the grammars of finite state languages, we are initially given a set of acceptable sentences, and a teacher who is permitted to tell us whether any sentence we propose is acceptable. We begin by systematically deleting sets of symbols that we suspect of being cycles from our set of initially known acceptable sentences, just as for finite state languages. However, for phrase structure languages, cycles may be pairs of phrases, as well as phrases. After each deletion of a phrase or pair of phrases, we ask the teacher whether the remaining phrase is an acceptable sentence.

For example, if we were initially given the string a b c d e f as an acceptable sentence, then we would ask whether the following were acceptable sentences:

	b c d e f ,	c d e f ,	d e f ,	etc.	deletions of
	a c d e f	a d e f	a e f		single
	a b d e f	a b e f	a b f		
	a b c e f	etc.	a b c		phrases
	etc.				
and					
	b d e f ,	a c e f ,	a b d f ,	a b c e	
	b c e f	a c d f	a b d e		deletions
	b c d f	a c d e			of
	b c d e				pairs
and					of
	c e f ,	a d e ,	etc.		phrases.
	c d f	a d f			
	c d e				

We perform the entire deletion routine for all initially known, acceptable sentences. Then we take each of the resultant acceptable sentences, reinsert several repetitions of the deleted portion, and ask whether the resultant phrases are acceptable sentences. In the above example, if a d e f were an acceptable sentence, we would ask whether a b c b c d e f , a b c b c b c d e f, a b c b c b c b c d e f, etc., were acceptable sentences. If c e f were an acceptable sentence we would ask if a b a b c d d e f, a b a b a b c d d d e f, a b a b a b a b d d d d e f, etc., were acceptable sentences. We would ask for enough cases of repetition of the deleted portion to be reasonably certain that an arbitrary number of such repetitions would leave the sentence acceptable.

Those sentences in which one could not find phrases or phrase pairs that could be deleted and repeated an arbitrary number of times and still leave the sentence acceptable, would be called "basic sentences", and would be retained for future examination.

We would next consider all such shortened sentences resulting from such deletion, that remain acceptable sentences after an arbitrary number of repeated reinsertions of the deleted phrase or phrases. If a b c b c d e f, a b c b c b c def, etc., and a b a b c d d e f, a b a b a b c d d d e f, etc., were acceptable, then a d e f and c e f would be two such sentences.

We would mark the places at which the deletions and repetitions in these sentences occurred. In the case of a d e f and c e f, we could do this by the notation: a α d e f and β c β e f. The phrase (b c) and the phrase pair (a b, d) are called "cycles." The cycle (b c) is insertable at the α position of a α d e f, and the cycle (a b, d) is insertable at the β positions of β c β e f. The symbols α and β are called "cycle markers."

When we have obtained all of the sentences from our initial sample, with their associated cycles and cycle markers, we attempt to break them down further, by looking for more cycles on the sentences and on the cycles themselves.

Suppose that a δ c d δ f is a set of acceptable sentences, for the cycle (b, e), in the δ position. We then try the deletion, repetition strategy.

ZTB
124

For deletion and repetition of single phrases, our strategy is as before.

We ask if the following are acceptable sets of sentences, with the cycle (b, e) : δ c d δ f, a δ d δ f, a δ c δ f, a δ c d δ and a δ δ f.

When we look for cycles of two phrases, our strategy is more restricted. The phrase pairs deleted must both be between the δ's or both not between the δ's. As a result, the only phrase pair deletions possible are (a, f) and (c, d) , resulting in δ c d δ and a δ δ f, respectively. The deletions (a, c), (a, d), (c, f) and (d, f) are all illegal. Note that the deletion of the phrase c d and the phrase pair (c, d) produce the same effect in the set of sentences a δ c d δ f. They both produce the sentence set a δ δ f. However, when c d is reinserted and repeated, we get a δ c d c d δ f, a δ c d c d c d δ f, etc. When c, d is reinserted and repeated, we get a δ c c d d δ f, a δ c c c d d d δ f, etc.

Proceeding in this way, we obtain a large set of strings of symbols, their cycles and cycle markers. When we can find no more cycles on a string and its cycle markers, we shall call this string a "basic sentence."

Occasionally in our analysis, we obtain complex strings and cycle markers like a γ b c δ d e δ f g γ h. Here, our cycles containing two phrases may be (a, h), (b, f), (b, g), (c, f), (c, g), or (d, e), but not (a, b), (a, d), (a, g), (b, e), (c, e), etc.

The rule is that for two phrases to form a permissible cycle, all cycle markers between them must be appropriately paired. Thus c and g have the paired markers δ, δ between them, and so (c, g) is a possible cycle. e and h have the unpaired markers δ and γ between them, and so (e, h) could not be a legal cycle.

The cycles that we obtain are also analysed. Cycles and cycle markers are found for them, just as was obtained for sentences.

For example, suppose we have the cycle (a b c d e , f g h i j k). We try systematic deletion of phrases and phrase pairs. If the resultant phrase

ZTB
124

pair remains a cycle for the same environments as before, we try repetition and reinsertion of the deleted portions. If (a d e, i j k) and (a b d, f g h i j k) and (a b c d e, f g k) are loops for the same environments as before, then we try (a b c b c d e, f g h f g h i j k), etc.; (a b c c d e e, f g h i j k), etc.; and (a b c d e, f g h i j h i j k) etc., in those same environments. If these all remain successful cycles, then we write them as (a δ d e, δ i j k) with cycle (b c, f g h) in the δ position, (a b γ d γ, f g h i j k) with cycle (c, e) in the γ position, and (a b c d e, f g ϵ k) with cycle (h i j) in the ϵ position.

The resultant cycles with their cycle markers are analysed further, in attempts to remove more cycles from them. When, finally, cycles are found from which it is impossible to extract more cycles, we will have what we call "basic cycles." We will be able to generate all of the sentences in the language, when we have obtained an adequate set of basic sentences with all of the cycle markers and basic cycles that may be attached to them, along with an adequate set of basic cycles with all of the cycle markers and basic cycles that may be attached to them.

In general, while there exist phrase structure languages in which the number of basic sentences is infinite, it can be shown that there always exists at least one set of basic sentences and basic cycles that are adequate to generate all sentences in the language.

A very simple phrase structure language with an infinite number of basic sentences, is the language whose sentences are a b, a a b b, a a a b b b, a a a a b b b b, etc. Some of its basic sentences with their cycle markers are α a b α, β a β b, γ a a γ b b, δ a a a δ b b b, etc. In all cases, the basic cycle is (a, b), in the α or β or γ or δ, etc. positions.

However, the basic sentence α a b α with cycle (a, b) is adequate to generate the entire language.

We can also find infinitely many basic cycles for these same cycle

ZTB
124

markers. Some of these basic cycles with their cycle markers, are : (ϵaa, ϵbb), (ωaaa, ωbbb) , ... In all cases, the cycle (a, b) may be inserted at the cycle marker positions.

Although the set of basic sentences and cycles constitute in themselves a complete description of the grammar of a phrase structure language, one always can convert this if one likes, to the (Σ, F), "substitution form" of grammar. The following are five typical examples:

"Basic Cycle" Form of Grammar	Corresponding "Substitution" Form of Grammar
1. Basic sentence: a α b α c Cycle in α position: (d, e)	Σ : S F : S $\rightarrow$ a A c A $\rightarrow$ d A e A $\rightarrow$ b
2. Basic sentence: a α b α c Cycles in α position: (d, e) and (f, g)	Σ : S F : S $\rightarrow$ a A c A $\rightarrow$ d A e A $\rightarrow$ f A g A $\rightarrow$ b
3. Basic sentence: a α β b β α c Cycle in α position: (f, g) Cycle in β position: (d, e)	Σ : S F : S $\rightarrow$ a A c A $\rightarrow$ f A g A $\rightarrow$ B B $\rightarrow$ d B e B $\rightarrow$ b
4. Basic sentence: a α b α c Cycle in α position: (d β f, g β e) Cycle in β position: (h, k)	Σ : S F : S $\rightarrow$ a A c A $\rightarrow$ d B e B $\rightarrow$ h B k B $\rightarrow$ f A g A $\rightarrow$ b
5. Basic sentence: a α b α c Cycle in α position: (f, g γ h γ k) Cycle in γ position: (d, e)	Σ : S F : S $\rightarrow$ a A c A $\rightarrow$ f A g B k A $\rightarrow$ b B $\rightarrow$ d B e B $\rightarrow$ h

ZTB
124

This work is supported by the United States Air Force Office of Scientific Research under Contract No. AF 49(638)-376.

BIBLIOGRAPHY:

1. Chomsky, N., "Three Models for the Description of Language," IRE Transactions on Information Theory, Vol. IT-2, Proceedings of the Symposium on Information Theory, September 1956, pp 113-124.

2. Chomsky, N., and Miller, G. A., "Pattern Conception," Report No. AFCRC-TN-57-57, August 7, 1957. (ASTIA Document No. AD110076)

3. Chomsky, N., and Miller, G. A., "Finite State Languages" Informa- and Control Vol. 1, No. 2, May 1958, pp 91-112.

4. Solomonoff, R., "The Mechanization of Linguistic Learning," Proceedings of the 2nd International Congress on Cybernetics, Namur, Belgium, Sept. 3-10, 1958.

ZTB
.124