The mechanization of linguistic learning [1]

by Ray J. SOLOMONOFF (U. S. A.),
Physicist, Senior Scientist, Zator Co.

INTRODUCTION

This paper will describe a general technique for inductive inference, followed by a brief description of some simple language types, among them phrase structure languages. We will then describe a routine that has been devised to discover grammar rules of phrase structure languages. A brief discussion will be made of the application of these methods to devising routines for learning to translate between certain pairs of phrase structure languages.

INDUCTIVE INFERENCE

By " inductive inference " I mean a process which involves the observation of many cases of a phenomenon and the formulation of a general rule that describes relationships between particular aspects of all of the observed cases. When the general rule has been formulated, it can be used to predict as yet unobserved parts of the same type of phenomenon, using data on parts that *have* been already observed.

A common form of inductive inference is prediction. We try to predict the nature of a particular event on the basis of our knowledge of events that precede it. The " inductive rule " spoken of above,

1. The research reported in this paper was sponsored by the Air Force Office of Scientific Research, Air Research and Development Command, under Contract No. AF 49(638)-376. The paper is identified as AFOSR-TN-59-246, ASTIA Document No. AD 212 226.

becomes a relationship between events and the events that precede them.

Suppose that John is sick on some days, and well on other days. We would like to know a day in advance whether he will be sick or well. To do this, we categorize days in various ways, and see if we can find a category that correlates well with John's health on the following day. Some possible categories are :

1 — Days on which the peak temperature is 40 degrees F.

2 — Days on which it snows.

3 — Days on which John goes to a movie.

If John is sick one day out of ten — but we find that he is sick on 80 percent of the days following snowy days, then category 2 — is a useful category for this kind of prediction.

In general, all inductive inference problems are quite similar to prediction problems. A kind of inductive inference problem that is a minor variation of prediction, is called " pattern completion ". Here, we are given many instances of completed patterns. These might be correctly worked arithmetic problems, or poems that " scan ". We then formulate general rules relating various parts of the completed patterns. Then we are given an incomplete pattern, and asked to complete it. Again the inductive rules that we devise may all be viewed as methods of categorizing incomplete patterns so that these categories correlate well with the correct completions of these incomplete patterns.

In all inductive inference, our degree of success depends on how well we have constructed our categories. I have devised a general method of category construction that appears to work well in the few types of problems for which I have tested it — namely in certain simple types of arithmetic problems, and in certain simple types of linguistic problems.

The method of category construction that I use starts out with a small set of primordial abstractions. Some possible kinds of primordial abstractions are words, phonemes, phrases, mathematical symbols, sets of symbols, sets of sets of symbols, etc. Rules for combining these abstractions to yield new abstractions may also be regarded as abstractions. The nature of the set of primordial abstractions that is to be used will depend upon whether the inductive inference problem to be solved is in mathematics, linguistics, etc. The primordial abstractions are combined and transformed in accord with certain rules to yield new, more complex abstractions, as well as classification categories. The new abstractions and categories can be combined again with each other and with old abstractions to yield still further abstractions and categories. This process continues as long as it is useful.

Using the method we have outlined, we can create an enormous number of abstractions and categories for prediction. In making a particular prediction, we will often find that there are many different categories that contain events preceding the event to be predicted. As an example, suppose that while John was, indeed, sick on 80 percent of the days following snowy days, he was sick but 1 percent of the days following his visit to a movie. If John went to a movie on a snowy day — what would be the probability of his being sick on the following day ? Suppose, further, that we have too little data on simultaneously snowy and movie days to make a prediction on that basis.

One important method of weighing the two probability estimates, makes the relative weights functions of the sample sizes that gave rise to the probability estimates. Sample size alone, however, is inadequate to resolve the difficulty. We must, in addition, consider the a priori reasonableness or appropriateness of the categories involved. We shall call this a priori appropriateness " utility ". The utility of a category will depend partly upon how useful that category has been in making predictions. It will also depend upon how that category was constructed. It is useful to also assign utilities to abstractions other than prediction categories.

The utility of any abstraction or prediction category will be an increasing function of the utilities of the abstractions that help form it. In addition, prediction categories are given greater utility if they have been particularly successful in making predictions.

Recursion equations to implement quantitative utility computations have been devised, but they have not been adequately tested.

SUCCESSFUL APPLICATIONS OF INDUCTIVE INFERENCE TECHNIQUES

A set of primordial abstractions and abstraction combination rules has been devised for the problem of programming a general purpose digital computer to " learn " to solve simple types of arithmetic problems, after having been given a suitable training sequence of correctly worked examples [3].

Abstractions and combination rules have also been devised for the problem of programming a computer to discover the grammar rules of a phrase structure language, in a suitable training situation.

The inductive inference routines for arithmetic and grammar using these abstractions and combination rules have not yet, however, been programmed on a computer.

Closely related to grammar discovery, is the problem of programming a computer to discover the translation rules between two languages, after having been given a large set of pairs of equivalent sentences in the two languages. An example will be given in the present paper of a pair of intertranslatable languages, in which discovering the translation rules is equivalent to discovering the grammar of a suitably generalized phrase structure language.

CHOMSKY'S THREE GRAMMAR TYPES

A brief outline of some of the work of N. Chomsky will provide the essential background for understanding my methods. In the paper " Three models for the description of language ", [1] Chomsky describes three grammar types of increasing complexity and increasing power for approximating the grammars of natural languages. The first of these, called the " finite state language ", is a language that is capable of being generated by a finite state machine.

The second language (of which the first is a special case) is called the " phrase structure language ". Sentences in this language consist of concatenations of phrases such as might be associated with various parts of speech. Examples are noun phrases and verb phrases. Sentences in a phrase structure language can be generated by starting with a certain word or phrase and making progressive substitutions from a stated list. For example, starting with the initial symbol " S ", the list of permissible substitutions might be :

S → NP : VP (i.e., noun phrase : verb phrase)
NP → The : N
N → man, boy, toy, airplane
VP → V : NP
V → saw, sees, is, was

Using these substitutions, we can generate a sentence in the following manner :

1. S
2. NP : VP
3. The : N : VP
4. The : N : V : NP
5. The : N : V : the : N
6. The boy : V : the : N
7. The boy saw the : N
8. The boy saw the man

Arbitrarily long sentences can be produced by this method if we allow substitutions that form " loops ". For example :

S → a
a → b c
c → d a

A permissible generation sequence might then be :

1. S
2. a
3. b c
4. b d a
5. b d b c
6. b d b d a
7. b d b d b c ..... etc.

In certain cases it is useful to have an identity element, I, such that $Ia \equiv aI \equiv a$, for all possible phrases, *a*.

The most powerful of Chomsky's languages is his " transformational language ". In this language, he starts with a set of " kernel sentences " that have been generated by a phrase structure grammar. He performs various transformations on such kernel sentences to obtain new sentences. Some of the transformations involve permutations of the words and phrases, with possible deletions or possible additions of words and phrases. There is, for example, a simple transformation that relates the sentence. " John has eaten the pie " to the passive form of the same sentence, i. e. " The pie has been eaten by John ".

DISCOVERING A GRAMMAR OF A LANGUAGE

In another paper [2], Chomsky and Miller deal with the problem of devising a method for discovering the grammatical rules of a language in a suitable training situation. They give one solution of this problem for finite state languages.

I have found that some of the methods of mechanized inductive inference, that had been successfully applied to arithmetic, could be extended and applied to the discovery of the grammars of finite state and phrase structure languages. While the mechanisms used seem to be adequate for the description of transformational languages, it is not known whether the mechanisms could be used to discover a transformational grammar adequate for a particular body of text.

The method devised for discovering the grammar rules of finite state languages may not need as large a sample size for analysis as the method of Chomsky and Miller [2], though the evidence for this is not yet very conclusive.

The method used for phrase structure languages consists of " factoring " the set of acceptable sentences into the (Boolean) union of " products " of certain sets of phrases. Here, we use the the term " product " to denote concatenation. If a_i is a member of the set of phrases α, and b_j is a member of the set of phrases β, then $a_i\ b_j$ (the concatenation of a_i and b_j) will be a member of the set of phrases designated by $\alpha \times \beta$. For example, suppose the set of acceptable sentences was $a\ b$, $a\ c$, $b\ b$, and $b\ c$. We could completely factor this set into the " product " of the sets $\alpha \equiv (a, b)$ and $\beta \equiv (b, c)$.

The method of factoring that is used here involves a " teacher " If it is suspected that all of the members of $\alpha \times \beta$ are acceptable sentences, and only a few of these members have been given to the machine — (say $a\ c$ and $b\ b$) — then to verify this " suspicion ", the machine would have to ask the teacher if $a\ b$ and $b\ c$ were acceptable sentences. In the present method, the machine is allowed to ask the teacher only questions of the type " Is γ an acceptable sentence ? " A " teacherless " training situation is discussed in Appendix I.

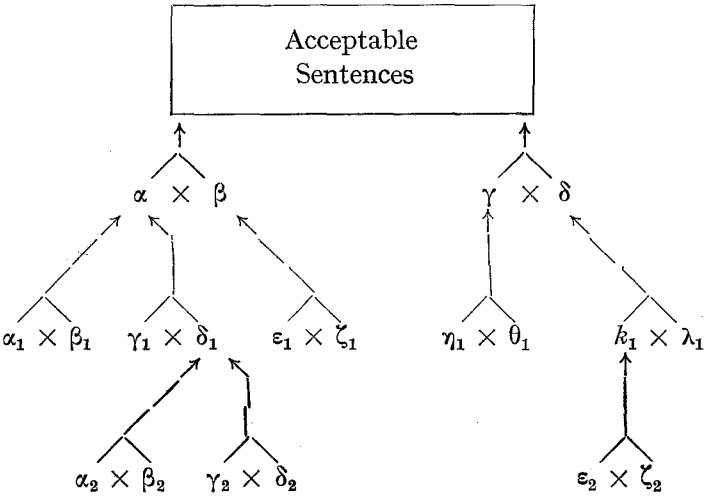
Usually it will not be possible to find a single pair of factors that will yield the entire set of acceptable sentences, so we will then use the union of several such " products ". For example, the set of sentences $a\ b$, $a\ c$, $b\ b$, $b\ c$, $d\ e$, $d\ f$, can be written as :

$$\alpha \times \beta \cup \gamma \times \delta, \text{ where } \alpha \equiv (a, b) : \beta \equiv (b, c), \gamma \equiv (d) \text{ and } \delta \equiv (e, f)$$

It will be noted that in all cases of interest the set of acceptable sentences is an infinite set, and that we will be given only a finite sample of it. The factor sets that we form will at first have only finite numbers of members, and so we cannot expect them to generate all acceptable sentences. Later we will show how to generate factor sets with infinite numbers of members.

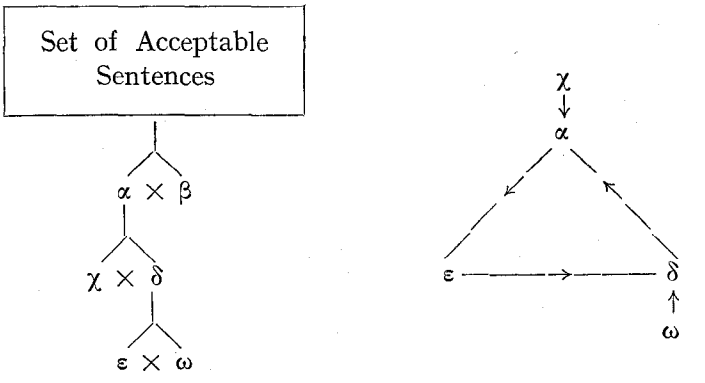
Usually we can factor the acceptable sentence set in many different ways. We shall assign more utility to those factor pairs that produce the largest numbers of acceptable sentences. This method of utility computation is described in Appendix II.

If α is one of the factors of the set of acceptable sentences, we will factor α in all possible ways. Each of these subfactors will, in turn, be factored, and so on, thereby forming a large, highly branched tree. The subfactors will also be assigned utilities in a manner similar to that used for the factors of the set of acceptable sentences.



It is clear that we will soon have an enormous number of factor sets. We will use the utility concept to reduce this number to manageable proportions by giving prior consideration to sets of high utility.

While we are forming the various factor sets, we also look for inclusion relationships among them in the following manner. Suppose that $\alpha \times \beta$ is included in the set of acceptable sentences, and $\chi \times \delta \subset \alpha$ and $\varepsilon \times \omega \subset \delta$.



Furthermore, suppose we notice that many phrases of α are also phrases of ε. We may then *suspect* that ε includes α. If this were true, we would have the recursion relations :

$\chi \times \delta \subset \alpha$; $\varepsilon \times \omega \subset \delta$; $\alpha \subset \varepsilon$, **and hence** $\chi \times \alpha \times \omega \subset \alpha$

Note that every member of ε is " part of " some member of α. However, ε may still include α if they are infinite sets — even though χ and ω contain members other than identity element.

Using the few known members of α, ω and χ we can use these recursion relations to obtain as many new members of α as we like. We can test some of these new members of α, to see if $\alpha \times \beta$ still gives only acceptable sentences. If it does, then we may tentatively conclude that ε does indeed include α.

If not, we will look for inclusion relations between *other* pairs of phrase sets, e. g. α and ω.

When a large enough number of such inclusion relations have been found, it will be possible to generate all of the acceptable sentences in the language, by using the resultant recursion relations, and a small, but adequate set of initially known members of certain of the sets. The set inclusion rules correspond directly to Chomsky's substitution rules, and they generate the set of all acceptable sentences in the same manner that the substitution rules do. Some modifications of the utility evaluation scheme that are brought about by the recursion relations, are discussed in Appendix III.

It had been mentioned earlier that the number of factor sets increased very rapidly, and that it was expedient to consider as few of them as possible, providing one could still obtain the proper recursion relations.

This is done through use of the utility concept. If $\varepsilon \times \omega \subset \delta$ and δ is of high utility, then both ε and ω are given high utilities. As was mentioned before, factors that have high utilities are factored earliest. Also, we first look for inclusion relations between factors of high utility. If utility evaluation is not used, the above method is an exhaustive technique for trying all possible phrase structure grammars that might fit. Since there are only a finite number of grammar rules, such a procedure must, eventually, yield the correct grammar — providing, of course, that we have an adequate training sequence of examples. The utility evaluation yields a reasonable means of ordering this exhaustive search, so as to greatly increase the probability that the correct set of grammar rules will be found after a relatively short search.

An analysis has been made of the effectiveness of utility evaluation in reducing search time. So far, the effectiveness seems very likely, but has not been proved conclusively. The effectiveness is to some extent corroborated by the fact that the above method has been successfully used to obtain the grammar rules of a few simple phrase structure languages.

DISCOVERING MECHANICAL TRANSLATION METHODS

The method described above is of interest not only as a technique for the discovery of grammar rules, but may, in certain cases, be used to implement the learning of language translation by machine.

Consider two languages L_1 and L_2, such that there exists a translation between L_1 and L_2. In the simplest case, this will mean that there is a one-to-one correspondence between the acceptable sentences in L_1 and the acceptable sentences in L_2.

Let us then construct a third " language ", L_3 in the following way : The " acceptable sentences " in L_3 consist of ordered pairs of sentences. The first sentence in each pair is a sentence from L_1 ; the second is the corresponding sentence in L_2.

To speak of L_3 as a " language " implies a certain generalization of the concept. The " words " of L_3 may be pairs of ordinary words ; the " phrases " of L_3 may be pairs of ordinary phrases — or there may be no useful way to divide the sentences of L_3 into words and phrases.

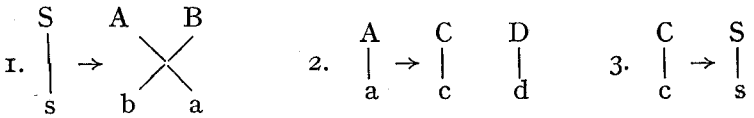
Knowing the grammar of L_3 will then be equivalent to being able to translate from L_1 to L_2. Here, we interpret " knowing the grammar " to mean that if a proposed sentence pair is given us, we can use the grammar rules to determine whether that sentence pair is in L_3 or not. Another, more useful kind of grammar would be one that would give methods to determine all possible legal sentences that could contain a certain fixed phrase as part of them. In this latter case, we need only present the grammar rules with a sentence from L_1, and if the grammar rules are to complete this partial sentence of L_3 to form an acceptable complete sentence in L_3, they must give us the translation in L_2 of the sentence in L_1.

The above discussion would be of little interest if the grammar rules of L_3 were unreasonably complex. However, if L_1 and L_2 are phrase structure languages, then it is possible for L_3 to be a phrase structure language. We shall give an example of such a situation. The meaning of " phrase structure language " in the case of L_3 will become clear in the following development.

In L_3, the sentences are ordered pairs of equivalent sentences from L_1 and L_2. The phrases and words of L_3 consist of paired equivalent phrases and paired equivalent words from L_1 and L_2. Although there will be many phrase equivalence pairs and word equivalence pairs in L_1 and L_2 in general, all words and all phrases in L_1 need not have equivalences in L_2.

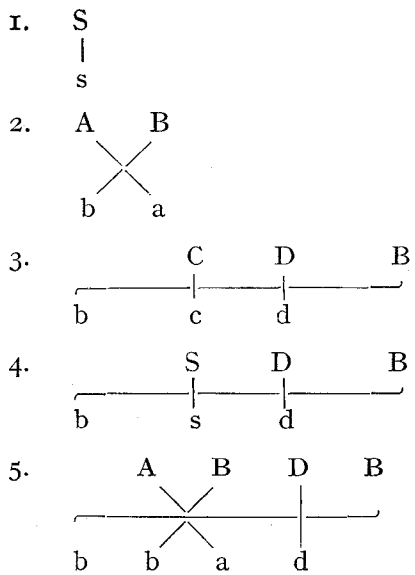
To show how L_3 may be a phrase structure language, we will give

a set of substitution rules for L_3, and show how sentences in L_3 may be generated by using them.



Here, the capital letters refer to words in L_1, or to words that will eventually become phrases in L_1. Similarly, the small letters refer to L_2. The lines between the capital and small letters connect equivalent words or phrases.

The generation of an acceptable sentence in L_3 could be :



Here (A B D B, b b a d) is an acceptable sentence in L_3, and b b a d of L_2 is the translation of A B D B of L_1.

It is easy to show that L_1, (the upper line of L_3) and L_2 (the lower line of L_3) are phrase structure languages. The substitution rules of of L_1 are :

1. S → A B 2. A → C D 3. C → S

Those of L_2 are :

1. s → b a 2. a → c d 3. c → s

The derivation of A B D B, the sentence in L_1, is :

1. S
2. A B
3. C D B
4. S D B
5. A B D B

The dérivation of b b a d, the sentence in L_2, is :

1. s
2. b a
3. b c d
4. b s d
5. b b a d

If, as in the above case, L_3 is a phrase structure language, then, through a large sample of equivalent pairs of sentences from L_1 and L_2, we can discover the grammar rules of L_3 using a generalization of the methods of grammar rule discovery that have been described above.

The grammar rules obtained by the procedure outlined, will be in a form that gives rules for creating all acceptable sentences. We must transform these rules into a different set of rules for completing parts of acceptable sentences. In the case of phrase structure languages, a transformation method has been found which appears to be adequate.

It should be noted that if L_1 and L_2 are phrase structure languages, L_3 *may* be a phrase structure language, but it *need not* be. It is easy to devise counter examples, since a translation rule may be *any* correspondence between the sentences of L_1 and L_2. On the other hand, if L_3 is a phrase structure language, L_1 and L_2 *must* be phrase structure languages. This may be readily seen by noting the manner in which the derivations and grammars of L_1 and L_2 are related to those of L_3 in the example above.

It might appear from the above example that if L_3 is a phrase structure language, then the translation rule that L_3 implies, always consists of a word for word translation, followed by changes in word order. In general, however, L_3 is capable of describing translation rules of significantly greater complexity. Words or phrases of L_1 may be made to correspond to phrases or non-adjacent pairs of phrases in L_2. Also, L_3 can resolve certain cases of ambiguity of word-for-word translation, by using contextual information.

It is also sometimes possible to devise translation rules utilizing a translation language L_4, which is a phrase structure language, in which neither L_1 nor L_2 are phrase structure languages. In all such cases $L_3 \subset L_4$. A simple example of such a situation is one in which L_1 and L_2 are identical, and are essentially non-phrase structure languages. In such a case L_3 is not a phrase structure language, yet it can easily be imbedded in a larger L_4 that *is* a simple phrase structure language.

APPENDIX I

THE "TEACHERLESS" TRAINING SITUATION.

If we are not allowed to ask questions, the problem of finding a grammar that is consistent with a given fixed body of text is complicated by the fact that there are always an infinite number of such grammars. It is possible, however, to define a " simplest " grammar from among all possible consistent grammars. Another important condition is that the language defined should contain as " few " sentences as possible, in addition to the fixed body of text. The meaning of " few " must be suitably defined, since most languages of interest contain an infinite number of sentences. A theoretical method for finding such optimum gammars without the services of a " teacher " has been devised, but the method involves an excessively long search. No really practical solution to this problem has been found for either finite state or phrase structure languages, although a solution for either language type would be extremely useful.

APPENDIX II

THE METHOD OF UTILITY EVALUATION.

If α and β are sets of phrases such that $\alpha \times \beta$ is included in the set of acceptable sentences, and if α has n_α members, and β has n_β members, then α and β will each be given utilities of :

$$U_\alpha = U_\beta = F_1(n_\alpha, n_\beta)$$

F_1 is a symmetric function of its arguments. At present, it appears that letting $F_1(n_\alpha, n_\beta) = n_\alpha n_\beta$ will be satisfactory, but this functional form has not been tested sufficiently, and its limitations, if any, are uncertain.

If χ and δ are sets of phrases such that $\chi \times \delta \subset \alpha$, with n_χ and n_δ being the numbers of phrases in χ and δ respectively, then :

$$U_\chi = U_\delta = F_2 (n_\chi, n_\delta, n_\alpha, U_\alpha)$$

F_2 is a symmetric function of n_χ and n_δ.
A tentative functional form for F_2 is :

$$F_2(n_\chi, n_\delta, n_\alpha, U_\alpha) = \frac{n_\chi n_\delta U_\alpha}{n_\alpha}$$

In this particular functional form, F_2 is the fraction of α's phrases that are contributed by $\chi \times \delta$, multiplied by U_α.

APPENDIX III

It should be noted that after one or more inclusion relationships have been found, it becomes possible to generate an arbitrarily large number of members of some of the sets of phrases. Using the utility evaluation scheme that has been outlined, we would find that we then had arbitrarily large utilities for these arbitrarily large sets of phrases. One method to avoid the difficulties associated with these arbitrarily large utilities is to limit the number of phrases generated by an inclusion relation through its corresponding recursion relation. The limit should be at a level that is just large enough to give adequate corroboration to the hypothesis that the inclusion relationship does, indeed, exist.

Discovery of new inclusion relationships will not otherwise modify the utility evaluation scheme that has been described. If $\chi \times \delta \subset \alpha$ then $U_\chi = F_2(n_\chi, n_\delta, n_\alpha, U_\alpha)$. If it were later discovered that $\chi \subset \varepsilon$, where $\epsilon \neq \delta$ and $\epsilon \neq \alpha$, then U_χ would still be a direct function of neither n_ϵ nor U_ϵ. However, the relation $\chi \subset \varepsilon$ will have been found by noticing that it implements some sort of recursion relation. Such a recursion relation would, in general, cause n_χ to be an increasing function of n , and so U_χ would be functionally related to n_ϵ in an indirect way. This consideration will not, however, modify the functional relationship between U_χ, n_χ, n_δ, n_α and U_α.

REFERENCES.

[1] CHOMSKY, N. *Three models for the description of language*. IRE, Transactions on Information Theory, Vol I. T-2; Proceedings of the Symposium on Information Theory, pp. 113-124, September 1956.

[2] CHOMSKY, N. and MILLER, G. A. *Pattern conception*. Report No. AFCRC-TN-57-57, (ASTIA Document No. AD 110076), August 7, 1957.

[3] SOLOMONOFF, R. J. *An inductive inference machine*. IRE Convention Record, Section on Information Theory, 1957.