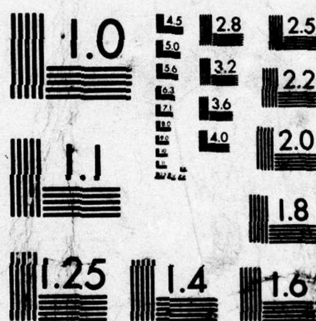
AD-A034 063 PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGI--ETC F/G 12/2
INFERENCE FOR TRANSITION NETWORK GRAMMARS,(U)
1976 S M CHOU, K S FU AF-AFOSR-2661-74
UNCLASSIFIED AFOSR-TR-76-1308 NL

| OF |
AD
A034 063

END
DATE
FILMED
2-77



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Third International Joint Conference on Pattern Recognition, Nov. 8-11, 1976

INFERENCE FOR TRANSITION NETWORK GRAMMARS†

S. M. Chou†† and K. S. Fu
School of Electrical Engineering
Purdue University
W. Lafayette, Indiana 47907

Abstract

This paper gives a brief introduction to transition networks and proposes an approach to the inference of transition network grammars.

I. Introduction

In order to model a language more realistically, it is desirable that the grammar used can be directly inferred from the set of sample sentences. This problem of learning a grammar based on a set of sample sentences is called grammatical inference. The applications of grammatical inference include areas of pattern recognition, information retrieval, artificial intelligence, and translation and compiling of programming languages[1,2].

A unique relationship between a language and a grammar does not exist. Quite often different grammars generate the same language. By definition[3], two grammars are equivalent if and only if they both generate the same language. It is possible to tell if two finite state grammars are equivalent[3]. However, for two grammars of types other than finite state, there is no way of telling their equivalence. Thus, except for finite state grammars, the inference problem does not have a unique solution unless additional constraints are placed upon the grammar being inferred[4]. One of the constraints may be to select a grammar of minimum complexity[5].

In the process of inference, a set of sentences that are known to be in the language must be given. This set is called a positive sample of the language. There may be another set of sentences given called a negative sample of the language that are known not to be in the language. A positive sample of a language L(G) is said to be structurally complete if each rewriting rule of G is used in the generation of a nonempty subset of the sample. In general, assumptions are made for all existing inference techniques as follows:

1. The type of the grammar being inferred is specified,
2. The given sample of the language is finite,
3. The given positive sample of the language is structurally complete,
4. The inferred grammar G is such that $S^+ \subseteq L(G)$ and $S^- \subseteq \bar{L}(G)$, where S^+ and S^- are positive and negative samples of the language, respectively; and $\bar{L}(G)$ is the complement of the language L(G).

†This work was supported by the AFOSR Grant 74-2661.

††S. M. Chou is presently with the Automation and Control Laboratory, General Electric Corporate Research and Development, Schenectady, New York 12301.

A survey of literature in the area of grammatical inference can be found in [2,6]. In this paper, an introduction to transition networks will first be given. Then following a brief review of the problem, an approach to the inference of transition networks will be proposed.

II. Transition Network Grammars

The transition network grammar has been developed as a model of natural language analysis[7-10].

A basic transition network (BTN) is a directed graph with labeled states and arcs, a distinguished state called the start state and a distinguished set of states called final states. It looks essentially like a nondeterministic finite state transition diagram, except that the labels on the arcs may be state names as well as terminal symbols. The interpretation of an arc with a state name as its label is that the state at the end of the arc will be saved on a pushdown store and the control will jump (without advancing the input pointer) to the state that is the arc label. When a final state is encountered, the pushdown store may be "popped" by transferring control to the state which is named on the top of the stack. An attempt to pop an empty stack when the last input symbol has just been processed is the criterion for acceptance of the input string.

The TN described above is a generalized pushdown automaton and is equivalent to a context-free grammar. However, a TN could be augmented into a more powerful machine by adding facilities to each arc. These include arbitrary conditions which must be satisfied in order for the arc to be followed and a set of register-setting actions to be executed if the arc is followed. The power of an augmented transition network (ATN) is determined by the facilities added to the arcs. With certain restrictions of the arcs, the power of the ATN can be modified for any kind of applications needed.

A TN can be described as a generalized pushdown machine consisting of a finite set of finite-state machines and a finite set of pushdown stores. Formally, a TN can be defined as a 6-tuple.

$TN = (\Sigma, Q, A, Q_0, Q_f, q_0)$, where Σ is a finite set of input symbols, Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states of the finite-state machines, $Q_f \subseteq Q$ is the set of final states of the finite-state machines, $q_0 \in Q_0$ is the initial state of the TN, A is a finite set of arcs. Associated with each state there are several arcs for transitions and actions. The arcs can be categorized into five classes:

1. CAT arc: (CAT C). A transition is made from the present state to the state at the end of the arc consuming an input symbol which is in the syntactic class labeled on the arc. The consumed symbol may be saved in a hold list when a HOLD action is required on the arc. This is done for future tests of context

See Form 1473

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

relationship.

2. PUSH arc: (PUSH q_0'), $q_0' \in Q_0$. The destination state of the arc is saved in the pushdown store and the state is transferred to the state shown on the arc which is the initial state of a finite-state machine.

3. POP arc: (POP). The state is transferred to the state shown on the top of the pushdown store. And the stack is popped one element up.

4. VIR arc: (VIRC). A transition from the present state to the state at the end of the arc is made by testing for the symbol shown on the VIR arc in the hold list.

5. JUMP arc: (JUMP). A transition from the present state to the state at the end of the arc is made if the conditions specified on the arc are satisfied. The transition does not consume any of the input string. This is a means of making a transition from one state to another without advancing the input pointer.

The input string is accepted when the TN is popping the empty pushdown store with empty hold list. The language accepted by a TN is denoted as L(TN).

It is interesting to compare the acceptors of phrase structure grammars with the transition networks. Note that a TN consists of a finite set of finite-state machines and pushdown stores. Consider a TN with only one network which is a set of states with CAT and POP arcs. This appears exactly to be a finite-state automaton which accepts type 3 languages. Furthermore, consider the BTN which is a set of finite-state machines and a pushdown store. The BTN is equivalent to an acceptor of a context-free language. With the aid of register-setting actions on the arc and the checking action of the VIR arcs, the ATN could achieve the power of a Turing machine. Suppose that there is a bound on the summing size of all the stores of the ATN such that the size is less or equal to the length of the part of the input string not yet scanned by the input pointer. Then it appears to be like a linear bounded automaton which accepts context-sensitive languages. It is then clear that an ATN is equivalent to a Turing machine. All the acceptors accepting different classes of languages can be derived from special cases of augmented transition networks[11].

A detailed discussion of the relationships between transition network grammars and Chomsky's hierarchy can be found in Ref. 11.

III. Inference for Finite-State Automata[2]

Finite state grammars are the simplest grammars. Most of the questions on the characteristics of this class of grammars are decidable or solvable. Finite-state languages are closed under union, complement, and intersection. If G_1 and G_2 are finite-state grammars generating L_1 and L_2 respectively, then there is an algorithm to determine if the set $(L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$ is empty. If it is empty, L_1 and L_2 are equivalent, and hence G_1 and G_2 are equivalent. There are a number of inference algorithms established for finite-state grammatical inference. Two of them are practical in implementation and easy to apply, and will be illustrated here.

1. Derivative grammars

Definition 1: The formal derivative of a set of strings A with respect to the symbol $a \in V_T$ is defined as[12]

$$D_a A = \{X \mid aX \in A\}$$

The canonical derivative finite-state grammar[6] G_{CD} associated with a positive sample $S^+ = \{X_1, X_2, \ldots, X_V\}$ is defined as follows:

$$G_{CD} = \{V_T, V_N, R, \sigma\}$$

a. Let $U = \{U_1, U_2, \ldots, U_r\}$ be the distinct derivative of S^+ not equal to λ or ϕ where λ and ϕ represent the null string and the empty set respectively. Let $U_1 = D_\lambda S^+$.

b. $\sigma = U_1$

c. $V_N = U$

d. The set of all distinct symbols found in S^+ is V_T

e. R is defined as follows:

$U_i \to a\, U_j$ if and only if $D_a U_i = U_j$

$U_i \to a$ if and only if $\lambda \in D_a U_i$

2. ~~K~~-Tail derived grammars

Definition 2: A derived grammar $G_D = \{V_{N_D}, V_T, R_D, B_\sigma\}$ is a grammar generated from a canonical grammar G_c by the following procedure[6]:

a. The terminal set V_T is the same for G_D and G_C

b. The nonterminal set corresponds to a partition of V_{N_C}, where V_{N_C} is the set of nonterminals of G_C

c. B_σ is the start symbol corresponding to the block in the partition containing σ

d. R_D is defined as follows:

1. $B_i \to a\, B_j$ is in R_D if and only if there exists $Z_\alpha, Z_\beta \in V_{N_C}$ such that

$$Z_\alpha \to a\, Z_\beta,\ Z_\alpha \in B_i,\ Z_\beta \in B_j$$

2. $B_i \to a$ is in R_D if and only if there exists $Z_\alpha \in V_{N_C}$ such that $Z_\alpha \to a$, $Z_\alpha \in B_i$

Definition 3: Let $u = a_1 a_2 \ldots a_r \in V_T^*$, and let $A \subseteq L(G)$. The K-tail of A with respect to u is defined as $g(u,A,K) = \{X \mid X \in D_u A,\ |X| \le K\}$.

Biermann and Feldman[13] have proposed a method of applying the idea of K-tail equivalence to partition the nonterminal set of a derivative grammar in obtaining the derived grammars. Let U_i and U_j be two distinct states of the canonical derivative grammar G_{CD}. These two states are associated with the derivatives $D_{X_i} S^+$ and $D_{X_j} S^+$ respectively where X_i and X_j are sequences from V_T^*. U_i and U_j are K-tail equivalent if and only if $g(X_i, S^+, K) = g(X_j, S^+, K)$. The method of K-tail derived grammars is easy to apply. In obtaining a quick rough approximation of the positive sample set, this method is practical and useful.

Since a simple transition network without any pushdown stores and registers function exactly the same as a finite automaton does, all the techniques of finite-state grammatical inference can be applied to the inference of simple transition network grammars without any difficulty.

IV. Inference for Basic Transition Networks

Context-free grammars are more complicated than finite-state grammars. It has been found that if a context-free grammar is non-self-embedding, then the language generated is a regular language. It is mainly this self-embedding property that distinguishes a context-free grammar from a regular grammar. Hence, most inference techniques for context-free grammars concern the revealing of the self-embeddings from the sample set. The theorem concerning self-embedding of context-free grammars is stated below:

Theorem 1[3]: For any context-free grammar $L(G)$, there exists integers m and n such that if there exists a string Z in $L(G)$ with $|Z| > n$ then Z can be decomposed to the form $Z = uv\omega xy$, where $vx \neq \lambda$ and $|v\omega x| \leq m$ such that for $i \geq 0$, $u v^i \omega x^i y$ is in $L(G)$.

The existing inference techniques for context-free grammars are still limited to some specific types of context-free grammars. Also, they often rely on heuristic methods during the process of inference. Gips has described a method to infer a pivot grammar which is an operator grammar in a very restricted form[5]. Crespi-Reghizzi, through the use of a structured sample set, has developed an inference procedure for K-distinct and K-homogeneous context-free grammar[14]. For the discovery of the self-embedding structures, Solomonoff has proposed a strategy[15]. The idea is to "guess" the elements of self-embedding, "v, ω, x," from the given positive sample set. This heuristic method is not practical in implementation. However, it suggests an important clue for later research in this area. Here, we will present a method of revealing the self-embedding structures using the idea of formal derivatives.

Definition 4: Let $v, x \in \Sigma^*$, and $S \in 2^{\Sigma^*}$, then

$$_v g_x(S,K) = \{\omega \mid \omega \in \Sigma^*,\ v\omega x \in S \text{ and } |\omega| \leq K\} = D_v\, E_x(S,K)$$

where

$$D_v(S,K) = \{\omega \in \Sigma^* \mid v\omega \in S \text{ and } |\omega| \leq K\}$$

$$E_x(S,K) = \{\omega \in \Sigma^* \mid \omega x \in S \text{ and } |\omega| \leq K\}$$

Note that $D_\lambda(S,K) = E_\lambda(S,K) = {}_\lambda g_\lambda(S,K) = S$.

Suppose that the grammar G being inferred is context-free, then by the self-embedding theorem, for some u, v, ω, x, y, $uv^i\omega x^i y$ is in $L(G)$, where $u, v, \omega, x, y \in \Sigma^*$, $vx \neq \lambda$ and $i \geq 0$. This implies that for a structurally complete positive sample S^+ and for some K, that $K = |\omega|$ then ${}_{uv^i} g_{x^i y}(S^+,K) \supseteq \{\omega\}$ for $i = 0, 1, 2, \ldots$

It can be seen that the substrings of $\{v^i\omega x^i \mid i \geq 0\}$ are generated by a recursive subnetwork shown in Fig. 1. The subnetwork is a finite-state automaton which can be obtained from the derivative grammar described in Section III using the sample set $S_A^+ = \{\omega, vAx\}$, where A is the name of the subnetwork. If such a subnetwork is found, the substrings of $\{v^i\omega x^i \mid i \geq 0\}$ in S^+ are replaced by a nonterminal which is the name of the subnetwork, i.e., A. The procedure of revealing a recursive subnetwork is then re-applied to the new sample set, treating the nonterminal symbol as a terminal. The procedure is repeated until no more recursive subnetworks can be found. Then the techniques of finite-state automaton inference described in Section III can be applied to complete the network of the sentences.

Given a sample set S^+, the procedure of inferring a BTN, such that $L(BTN) \supseteq S^+$ is described step by step as follows:

Step 1. Construct the derivative table.

a. Put the sample set S^+ in the first column, first row.

b. List all the nonempty $D_{v_i}(S^+,n)$ in the first column, with row number $i = 2,3,\ldots$, where $v_i \in \Sigma^+$, and n is an integer such that for every $X \in S^+$, $|X| \leq n$.

c. List all the nonempty $E_{x_j}(S^+,n)$ in the first row, with column number $j = 2,3,\ldots$, where $x_j \in \Sigma^+$.

d. Let the element of the table at column j, row i be denoted as T_{ij}. Complete the table with $T_{ij} = {}_{v_i} g_{x_j}(S^+,n)$, for $i, j \geq 2$.

Step 2. Find the equivalent classes.

a. For $K = 1,2,\ldots$, list the equivalent classes $U_{K,\ell}$ of the derivatives ${}_v g_x(S^+,K)$. If ${}_{v_{i'}} g_{x_{j'}}$ is in $U_{K,\ell}$ then for any $v_{i''}$, $x_{j''}$ such that ${}_{v_{i''}} g_{x_{j''}}(S^+,K) = {}_{v_{i'}} g_{x_{j'}}(S^+,K)$, ${}_{v_{i''}} g_{x_{j''}}$ is in $U_{K,\ell}$. If there are L distinct equivalent classes for K, then $\ell = 1,2,\ldots,L$.

b. For $K = 1,2,\ldots$, examine $U_{K,\ell}$, $\ell = 1,2,\ldots,L$. If a subset of $U_{K,\ell}$ is found to be a subset of the set $\{{}_{uv^i} g_{x^i y} \mid u, v, x, y \in \Sigma^*, vx \neq \lambda, i \geq 0\}$, go to Step 3, otherwise, to Step 4.

Step 3. Construct a subnetwork for the self-embedding

a. Since $U_{K,\ell} \subseteq \{{}_{uv^i} g_{x^i y} \mid u, v, x, y \in \Sigma^*, vx \neq \lambda, \ell \geq 0\}$, substrings v, x are found. Let ${}_{uv^{i_1}} g_{x^{i_1} y}$ be in $U_{K,\ell}$ such that any ${}_{uv^{i_2}} g_{x^{i_2} y} \in U_{K,\ell}$, $i_2 \geq i_1$. Denote the value of $U_{K,\ell}$ as $\bar{U}_{K,\ell} = {}_{uv^{i_1}} g_{x^{i_1} y} = \{\omega_m \mid m \geq 1, |\omega_m| \leq K\}$. Form a sample set

$S_A^+ = \{v^i{}_1\, \bar{U}_{K,\ell}\, x^i{}_1\} \cup \{vAx\}$ where A is the name of the new subnetwork.

b. Construct a derivative finite-state grammar for S_A^+ using the procedure described in Section III.

c. Obtain the subnetwork from the derivative grammar. Note that an arc in the TN with a nonterminal as its label is a PUSH arc.

d. Replace the substring $\{v^i\, \omega_m\, x^i \mid i \geq 1,\ m>0\}$ in S^+ with A.

e. Go to Step 1.

Step 4. Construct the network for the sentences.

a. Construct the derivative finite-state grammar for S^+.

b. Obtain the subnetwork S from the derivative grammar.

c. The *name* of the subnetwork S is the start state of the inferred BTN.

d. The inferred BTN is the set of all the subnetworks inferred.

e. STOP

The procedure is easy and practical to implement on a computer. Any sample set with a reasonable size which is large enough to imply the self-embedding structures of the language and is not so large as to consume up the computer memory, can be put on a computer to infer its BTN.

Example 1: Consider a language $L = \{b^\ell ab^k cb^k ab^\ell \mid \ell, k \geq 1\}$. The substring '$ab^kcb^ka$' is embedded between the b's in the sentence and the symbol 'c' is embedded between the b's in the substring. The language could be the encoded strings of two arms of chromosomes on each side of the centromere constriction. This can be shown in Fig. 2.

A sample set of size 35 which is listed in Table 1 is fed to the computer program implementing the inference procedure as stated. After the table of the derivatives is completed, an equivalent class of value 'c' is found. The class is shown as Table 2(a) which is summarized as Table 2(b). Examining Table 2(b), a *self-embedding 'c' is found with substring v = x = b.* A subnetwork for the sample set S_A^+ = (bcb,bAb) is constructed as Fig. 3.

Table 1. The Sample Set for the BTN Inference Experiment.

babcbab	bbabcbabb
babbcbbab	bbabbcbbabb
babbbcbbbab	bbabbbcbbbabb
babbbbcbbbbab	bbabbbbcbbbbabb
babbbbbcbbbbbab	bbabbbbbcbbbbbab
babbbbbbcbbbbbbab	bbabbbbbbcbbbbbbabb
bbbabcbabbb	bbbbabcbabbbb
bbbabbcbbabbb	bbbbabbcbbabbbb
bbbabbbcbbbabbb	bbbbabbbcbbbabbbb
bbbabbbbcbbbbabbb	bbbbabbbbcbbbbabbbb
bbbabbbbbcbbbbbabbb	bbbbabbbbbcbbbbbabbbb
bbbabbbbbbcbbbbbbabbb	bbbbabbbbbbcbbbbbbabbbb
bbbbbabcbabbbbb	
bbbbbabbcbbabbbbb	bbbbbbabcbabbbbbb
bbbbbabbbcbbbabbbbb	bbbbbb bbcbbabbbbbb
bbbbbabbbbcbbbbabbbbb	bbbbbbabbbcbbbabbbbbb
bbbbbabbbbbcbbbbbabbbbb	bbbbbbabbbbcbbbbabbbbbb
bbbbbabbbbbbcbbbbbbabbbbb	bbbbbbabbbbbcbbbbbabbbbbb

Table 2(a). The Equivalent Class of Value 'c'.

$D_{bab}\ E_{bab}$	$D_{bbabbbb}\ E_{bbbbabb}$
$D_{babb}\ E_{bbab}$	$D_{bbbabbb}\ E_{bbbabbb}$
$D_{bbab}\ E_{babb}$	$D_{bbbbabb}\ E_{bbabbbb}$
$D_{babbb}\ E_{bbbab}$	$D_{bbbbbab}\ E_{babbbbb}$
$D_{bbabbb}\ E_{bbbabb}$	$D_{babbbbbb}\ E_{bbbbbbab}$
$D_{bbbabb}\ E_{bbabbb}$	$D_{bbabbbbb}\ E_{bbbbbabb}$
$D_{bbbbab}\ E_{babbbb}$	$D_{bbbabbbb}\ E_{bbbbabbb}$
$D_{babbbbb}\ E_{bbbbbab}$	$D_{bbbbabbb}\ E_{bbbabbbb}$

Table 2(b). Summary of Table 2(a).

$D_{bab^i}\ E_{b^iab}$	$1 \leq i \leq 6$
$D_{b^2ab^i}\ E_{v^iab^2}$	$1 \leq i \leq 5$
$D_{b^3ab^i}\ E_{b^iab^3}$	$1 \leq i \leq 4$
$D_{b^4ab^i}\ E_{b^iab^3}$	$1 \leq i \leq 3$
$D_{b^5ab^i}\ E_{b^iab^5}$	$i = 1$

Replacing substrings $\{b^icb^i \mid i>1\}$ in the sample set of Table 1 by the nonterminal symbol 'A', we get a new sample set shown in Table 3.

Table 3. The Sample Set After Replacement.

baAab	bbbbaAabbbb
bbaAabb	bbbbbaAabbbbb
bbbaAabbb	bbbbbbaAabbbbbb

The equivalent class of the derivatives of the new sample set is shown in Table 4. A self-embedding of 'aAa' is found with v=x=b. The subnetwork for the sample set S_B^+ = {baAab, bBb} is constructed as Fig. 4.

Replacing substrings $\{b^iaAab^i \mid i \geq 1\}$ in the sample set of Table 3 by the symbol 'B', we get the set {B}. The inference procedure is completed with the inferred transition network B shown in Fig. 5. The language generated by this transition network is

$$L = \{b^\ell ab^k cb^k ab^\ell \mid k, \ell \geq 1\}.$$

Table 4. The Equivalent Classes of the Derivatives of the Sample Set in Table 3.

'A'	'Aa'
$D_{ba}\ E_{ab}$	$D_{ba}\ E_{b}$
$D_{bba}\ E_{abb}$	$D_{bba}\ E_{bb}$
$D_{bbba}\ E_{abbb}$	$D_{bbba}E_{bbb}$
$D_{bbbba}\ E_{abbbb}$	$D_{bbbba}\ E_{bbbb}$
$D_{bbbbba}\ E_{abbbbb}$	$D_{bbbbba}\ E_{bbbbb}$
$D_{bbbbbba}\ E_{abbbbbb}$	$D_{bbbbbba}\ E_{bbbbbb}$

'aA'	'Aab'
$D_b\ E_{ab}$	$D_{ba}\ E_{\lambda}$
$D_{bb} E_{abb}$	$D_{bba}\ E_b$
$D_{bbb} E_{abbb}$	$D_{bbba}\ E_{bb}$
$D_{bbbb}\ E_{abbbb}$	$D_{bbbba}\ E_{bbb}$
$D_{bbbbb}\ E_{abbbbb}$	$D_{bbbbba}\ E_{bbbb}$
$D_{bbbbbb}\ E_{abbbbbb}$	$D_{bbbbbba}\ E_{bbbbb}$

'baA'	'aAa'
$D_{\lambda}\ E_{ab}$	$D_b\ E_b$
$D_b\ E_{abb}$	$D_{bb}\ E_{bb}$
$D_{bb}\ E_{abbb}$	$D_{bbb}\ E_{bbb}$
$D_{bbb}\ E_{abbbb}$	$D_{bbbb}\ E_{bbbb}$
$D_{bbbb}\ E_{abbbbb}$	$D_{bbbbb}\ E_{bbbbb}$
$D_{bbbbb}\ E_{abbbbbb}$	$D_{bbbbbb}\ E_{bbbbbb}$

An analysis of CSL in terms of transformational grammars can be found in Ref. 11. A CSG can be seen as a CFG (base) and a set of transformational rules. The CSL is obtained by applying a sequence of transformations to the CFL generated by the CFG. For a given sample of CSL, suppose that some reverse transformations are assumed, then a sample set of CFL can be obtained by applying the reverse transformations. To complete an ATN for the CSL, the technique of BTN inference is used to construct the BTN for the CFL and then augmented arcs are added to achieve the transformations. It can be seen that the set of assumed reverse transformations plays an important role in the resulting ATN. The system of ATN inference may be operated under a supervisor in the fashion of trial and error. For example, the sketch of an ATN inference system shown in Fig. 6 may be a possible solution. For the illustration of the ATN inference technique, Example 2 is given below.

Example 2: A sample set shown in Table 5 is given to the ATN inference system sketched in Fig. 6. A reverse transformation 'bbc→bcb' to the strings is assumed. The sample set obtained after applying the reverse transformation is shown in Table 6.

Table 5	Table 6
abc	abc
$a^2b^2c^2$	$a^2(bc)^2$
$a^3b^3c^3$	$a^3(bc)^3$
$a^4b^4c^4$	$a^4(bc)^4$
$a^5b^5c^5$	$a^5(bc)^5$
$a^6b^6c^6$	$a^6(bc)^6$
$a^7b^7c^7$	$a^7(bc)^7$

Now, for the new sample set, a BTN shown in Fig. 7 is inferred using the technique described previously in this Section. To complete the augmented transition network, two augmented arcs, 7 and 8, are added. The resulting ATN is shown in Fig. 8. It is clear that the given sample set is a subset of the language generated by the inferred ATN.

V. Conclusions and Remarks

In this paper, the inferences of transition network grammars are presented. In particular, a strategy to reveal the self-embedding structures in context-free lanaguages has been proposed. It is found to be a rather systematic and practical approach compard to the existing heuristic methods.

Although the research of finite-state grammatical inference has been reasonably successful, the research in the whole area of grammatical inference is still in its infancy. In many cases, finite-state grammars are not powerful enough to fully characterize the language under study. Therefore, more complex grammars are needed. The subject of grammatical inference for grammars of types other than finite state is of increasing importance. It has been proven that transformational grammars are as powerful as type 0 grammars. The close relationship between transition network grammars and transformational grammars has been shown.[11] Clearly, a context-sensitive grammar can be represented as a context-free grammar plus a set of transformation rules. These can be fully expressed in terms of basic transition networks with a set of augmented arcs. It turns out that the basic transition networks, which correspond to the context-free grammar, are the foundations of grammars of different complexities. The inference of basic transition network grammars becomes a key to the area of grammatical inference. Much work remains to be done; however, it is hoped that more interest and research will be stimulated on this subject.

References

1. Crespi-Reghizzi, S., Melkanoff, M. A., and Lichten, L., "The Use of Grammatical Inference for Designing Programming Languages," CACM, February 1973, pp. 83-90.

2. Fu, K. S., Syntactic Methods in Pattern Recognition, Academic Press, 1974.

3. Hopcroft, J. E. and Ullman, J. D., Formal Languages and Their Relation to Automata, Addison-Wesley, 1969.

4. Solomonoff, R. L., "A Formal Theory of Inductive Inference," Information and Control 7, 1964, pp. 11-22, 224-254.

5. Feldman, J. A., Gips, J., Horning, J. J., and Reder, S., "Grammatical Complexity and Inference," Tech. Rept. No. CS-125, Computer Science Department, Stanford University, Stanford, California, 1969.

6. Fu, K. S. and Booth, T. L., "Grammatical Inference: Introduction and Survey - Part I," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-5, No. 1, January 1975, pp. 95-111.

7. Thorne, J., Bratley, P., and Dewar, H., "The Syntactic Analysis of English by Machine," Machine Intelligence 3, Michie, D., editor, American Elsevier Press, New York, 1968, pp. 281-297.

8. Bobrow, D. and Fraser, B., "An Augmented State Transition Network Analysis Procedure," Proc. International Joint Conference on Artificial Intelligence, Washington, D.C., 1969, pp. 557-567.

9. Woods, W. A., "Transition Network Grammars for Natural Language Analysis," CACM, Vol. 13, No. 10, October 1970, pp. 591-606.

10. Woods, W. A., "An Experimental Parsing System for Transition Network Grammars," BBN Report No. 2362, Computer Science Division, Bolt Beranek and Neyman Inc., Cambridge, Massachusetts, May 1972.

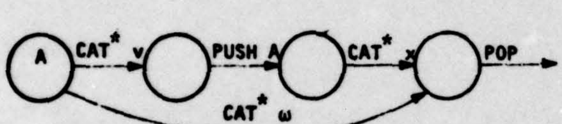
11. Chou, S. M. and Fu, K. S., "Transition Networks for Pattern Recognition," Tech. Rept. TR-EE 75-39, December 1975, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana.

12. Brzozowski, J. A., "Derivatives of Regular Expressions," Journal of ACM, Vol. 11, No. 4, pp. 481-494, 1964.

13. Biermann, A. W. and Feldman, J. A., "On the Synthesis of Finite-State Acceptors," Stanford Artificial Intelligence Project Memo, AIM-114, Stanford University, Stanford, California, 1970.

14. Crespi-Reghizzi, S., "Reduction of Enumeration in Grammar Acquisition," 2nd International Joint Conference on Artificial Intelligence, London, 1971, pp. 546-552.

15. Solomonoff, R. J., "A New Method for Discovering the Grammars of Phrase Structure Languages," Information Processing, June 1959, pp. 285-290.



(CAT* x) means a sequence of CAT arcs accepting a substring x

Fig. 1. The BTN generating $\{v^i \omega x^i \mid i \geq 0\}$

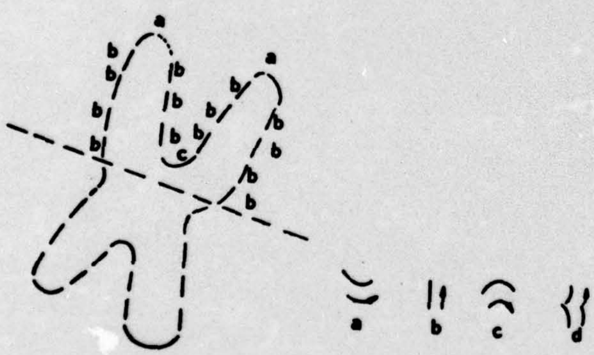


Fig. 2. A sequential embedding example

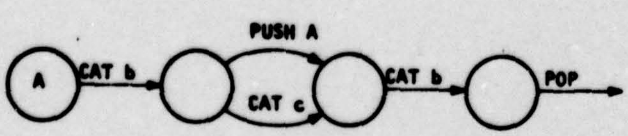


Fig. 3. The subnetwork for $(b^i c b^i \mid i \geq 1)$

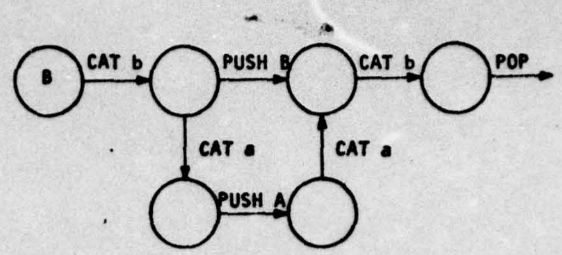


Fig. 4. The subnetwork for {baAab, bBb}

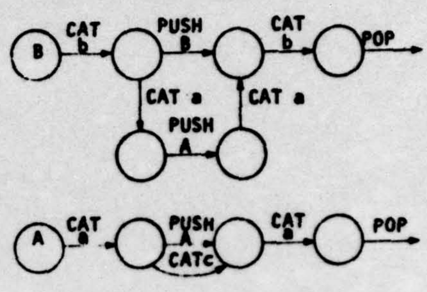


Fig. 5. The Inferred BTN for S^+

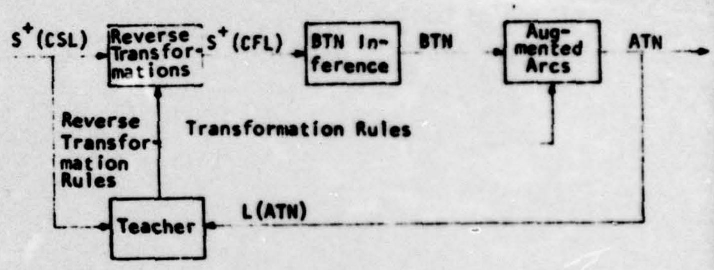


Fig. 6. An ATN Inference System

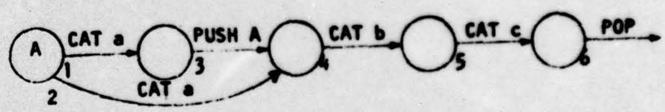


Fig. 7. The Inferred BTN

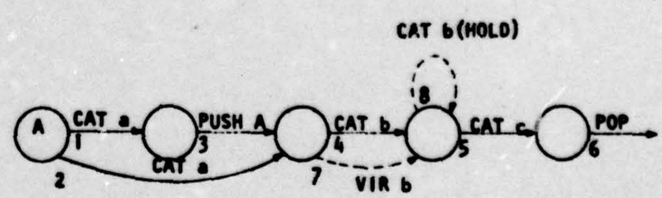


Fig. 8. The Inferred ATN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 18 AFOSR - TR - 76 - 1308	2. GOVT ACCESSION NO.
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): 6 INFERENCE FOR TRANSITION NETWORK GRAMMARS,	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): 10 S. M. Chou [redacted] K. S. Fu	8. CONTRACT OR GRANT NUMBER(s): 15 AF-AFOSR [redacted]-2661-74
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Purdue University, School of Electrical Engineering, West Lafayette, IN 47907	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304 A2 (16) (17)
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332	12. REPORT DATE: 11 1976
	13. NUMBER OF PAGES: 6
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): 12 7p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

ACCESSION for		
NTIS	White Section	☑
DDC	Buff Section	☐
UNANNOUNCED		☐
JUSTIFICATION		
BY		
DISTRIBUTION/AVAILABILITY CODES		
Dist.	AVAIL. and/or SPECIAL	
A		

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper gives a brief introduction to transition networks and proposes an approach to the inference of transition network grammars.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 292 000

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)