THE TMG RECOGNITION SCHEMA

Alexander Birman

A DISSERTATION
PRESENTED TO THE
FACULTY OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE BY THE
DEPARTMENT OF
ELECTRICAL ENGINEERING

February, 1970

Acknowledgements

I am deeply indebted to Professor Jeffrey D. Ullman who, as my thesis advisor, guided my research.

Part of this work was done during the summer 1969, at the T.J. Watson Research Center of IBM.

Table of Contents

I. INTRODUCTION

TMG is a compiler writing system reported by McClure in [1]. As part of the TMG system, TMGL is a language for describing translation procedures. A sentence in TMGL can be thought of as a recognition subroutine which performs the syntactic analysis of a string; it is executed dynamically and has definite rules of flow. This is in contrast to the usual syntax directed compiler, in which the order of analysis is not specified by the writer. Once the analysis has been accomplished, the output is then constructed.

The formalization of the syntatic analysis schema used in TMG and the investigation of its properties form the scope of this work. The results presented here can be then extended to include translations into a suitably defined output alphabet.

In Chapter II we define the "TMG Recognition Schema" (TS); then we describe an automaton (TSA) corresponding to a given TS and we show that the TSA accepts exactly the language "recognized" by the TS. In Chapter III we show that the languages recognized by a subclass of TS, the so called "well-formed TS", include all deterministic cfl.

In Chapter IV we study the various ways in which the TS could fail to recognize an input string; we call these "failures". Then, these types of failures are investigated and among other results,

some closure properties and decidability results are obtained.

Chapter V discusses time complexity: it is shown that the languages "recognized" by the TS, the TSL, can be recognized in linear time by a given algorithm.

In Chapter VI the relation between TSL and other classes of languages is investigated. It is shown that the TSL are context sensitive, they include some non-cfl's and the TSL over a one letter alphabet are not regular.

Chapter VII and Chapter VIII relate to generalizations of the TS. Two of these models, the gTS and the (ℓ,m)-TS, which are shown to be equivalent, are studied; it is shown that they compare favorably with the original model. A third model (the eTS) is briefly described and is shown to be equivalent to the gTS. Finally it is shown that there exists a class of one way deterministic balloon automata which accepts the class of languages recognized by the gTS.

II. THE TMG RECOGNITION SCHEMA AND ITS AUTOMATON

First, we will give an informal description of our basic concept.

The "TMG recognition schema", or shortly TS, has a set of variables, a set of terminal symbols, a distinguished symbol and a set of rules, just like a phrase structure grammar. The rules have one

of the forms A..a or A..BC/D where A,B,C,D are variables, "a" is a terminal symbol or the null string ε or the metasymbol "f".

Unlike a grammar there is at most one rule for each variable in a TS. A variable in a TS behaves very much like a subroutine; given a string, a variable is "called". If its rule is of the form A..BC/D, it will "call" another variable. If the rule is of the form A..a, for some input symbol a, it will try to match the next input symbol. The outcome of a variable "call" can be success or failure. Suppose variable A is called on string x$ ($ is a special symbol called endmarker) and the rule for A is A..BC/D. Then A calls B and two cases arise:

Case 1 B succeeds and recognizes a substring x_1, where $x_1y = x$ for some y. Then C is called on string y$.

Case 1a C succeeds and accepts y_1, $y_1y_2 = y$ for some y_2. Then A succeeds and accepts x_1y_1.

Case 1b C fails. D is called on x$ (i.e. we have backtracking). If D succeeds and accepts x_2, $x_2x_3 = x$ for some x_3, then A succeeds and accepts x_2. If D fails then A fails.

Case 2 B fails. Then D is called on x$ and we proceed as in Case 1b.

We should also mention at this point that a string can be rejected by a TS for other reasons beside failure in the sense described above (which we call "recognition failure"), for example,

if the procedure stops because a variable is called for which there is no rule in the TS (we call it a "subroutine failure").

Now we give a formal definition for the TS:

Definition 2.1 A TMG Recognition Schema (TS) R is a 5-tuple

$R = (V,\Sigma,P,S,\$)$ in which:

V is a finite set of variables

Σ is a finite set of terminal symbols

S is an element of V

\$ is a symbol called endmarker, \$ not in Σ. (For any input alphabet Σ we will use the notation Σ_e for the set $\Sigma \cup \{\$\}$).

P is a finite set of rules of the form a) or b):

a) A..BC/D, A,B,C,D in V.

b) A..a, a in $\Sigma_e \cup \{\epsilon, f\}$, f is a metasymbol not in Σ_e and ϵ is the null string.

For any variable A there is at most one rule with A on the lefthand side.

We define the set of relations for each n in N, $A \overset{n}{\underset{R}{\Longrightarrow}} (x\uparrow y, i)$, x,y in Σ_e^*, i in {0,1}, as follows:

1) If A..a is in P, a in Σ_e, then

$A \overset{1}{\underset{R}{\Longrightarrow}} (a\uparrow x, 0)$ for all x in Σ_e^*, and

$A \overset{1}{\underset{R}{\Longrightarrow}} (\uparrow bx, 1)$, for all x in Σ_e^*, b in Σ_e, $b \neq a$.

(In the derivation $A \underset{R}{\overset{1}{\Longrightarrow}} (a\uparrow x,0)$, A has outcome 0 - i.e. success - on ax and it recognizes the substring a, which is indicated by the fact that the marker $\uparrow$ follows the substring which has been recognized. In $A \underset{R}{\overset{1}{\Longrightarrow}} (\uparrow bx, 1)$, A has outcome 1 - i.e. recognition failure - on string bx and, as is always the case for outcome 1, the string recognized is the null string or equivalently the marker $\uparrow$ precedes the string bx).

2) If A..ε is in P, then

$$A \underset{R}{\overset{1}{\Longrightarrow}} (\uparrow x, 0) \text{ for all } x \text{ in } \Sigma_e^*.$$

(The outcome for a rule A..ε is success, the string recognized being ε).

3) If A..f is in P, then

$$A \underset{R}{\overset{1}{\Longrightarrow}} (\uparrow x, 1) \text{ for all } x \text{ in } \Sigma_e^*.$$

(The outcome for a rule A..f is 1, the string recognized being, of course ε).

4) Let A..BC/D be in P. For each x_1, x_2, x_3 and x_4 in Σ_e^*, if:

(a) $B \underset{R}{\overset{\ell}{\Longrightarrow}} (x_1\uparrow x_2x_3,0)$, $C \underset{R}{\overset{m}{\Longrightarrow}} (x_2\uparrow x_3,0)$, then $A \underset{R}{\overset{k}{\Longrightarrow}} (x_1x_2\uparrow x_3,0)$, $k=\ell+m+1$. (B and C succeed; then A succeeds and it recognizes the concatenation of x_1 and x_2).

(b) $B \underset{R}{\overset{\ell}{\Longrightarrow}} (x_1\uparrow x_2,0)$, $C \underset{R}{\overset{m}{\Longrightarrow}} (\uparrow x_2,1)$, $D \underset{R}{\overset{p}{\Longrightarrow}} (x_3\uparrow x_4,0$, and $x_1x_2 = x_3x_4 = x$, then $A \underset{R}{\overset{k}{\Longrightarrow}} (x_3\uparrow x_4, 0)$, $k = \ell+m+p+1$.
(B succeeds on x_1x_2, recognizing x_1, but C fails on x_2. Then D is called; A recognizes whatever D recognizes).

(c) $B \underset{k}{\overset{\ell}{\underset{R}{\Longrightarrow}}} (\uparrow x_1 x_2, 1)$, $D \underset{R}{\overset{m}{\Longrightarrow}} (x_1 \uparrow x_2, 0)$, then $A \underset{R}{\Longrightarrow} (x_1 \uparrow x_2, 0)$, $k = \ell+m+1$. (This case is similar to b). B fails and D is called; A recognizes whatever D recognizes).

(d) $B \underset{R}{\overset{\ell}{\Longrightarrow}} (x_1 \uparrow x_2, 0)$, $C \underset{R}{\overset{m}{\Longrightarrow}} (\uparrow x_2, 1)$, $D \underset{R}{\overset{p}{\Longrightarrow}} (\uparrow x_1 x_2, 1)$, then $A \underset{R}{\overset{k}{\Longrightarrow}} (\uparrow x_1 x_2, 1)$, $k = \ell+m+p+1$. (The same as b), only that D fails; A fails too).

(e) $B \underset{R}{\overset{\ell}{\Longrightarrow}} (\uparrow x_1, 1)$ and $D \underset{R}{\overset{m}{\Longrightarrow}} (\uparrow x_1, 1)$, then $A \underset{R}{\overset{k}{\Longrightarrow}} (\uparrow x_1, 1)$, $k = \ell+m+1$. (The same as c), only that D fails; hence, A fails).

If $A \underset{R}{\overset{n}{\Longrightarrow}} (x \uparrow y, i)$, x,y in Σ_e^*, i in $\{0,1\}$, we say that A <u>derives</u> $(x \uparrow y, i)$ <u>in n steps</u>. (Equivalently, we will say A has outcome i on xy and it recognizes the substring x). We say that A <u>derives</u> $(x \uparrow y, i)$ if A derives $(x \uparrow y, i)$ in r steps for some r, and we write $A \underset{R}{\Longrightarrow} (x \uparrow y, i)$.

<u>The language recognized by R is:</u>

$$T(R) = \{x \mid x \text{ in } \Sigma^*, S \underset{R}{\Longrightarrow} (x\$\uparrow, 0)\} .$$

<u>Remarks</u>

1. Let $A \underset{R}{\Longrightarrow} (x \uparrow y, i)$, for some xy in Σ_e^*. If $i = 0$ we say that A <u>succeeds</u> on xy. If $i = 1$, we have a <u>recognition failure</u> (or simply <u>failure</u>).

2. We generalize the form of the rules in P, to include expressions $A..\alpha_1/\alpha_2/\ldots/\alpha_n$, where α_i in V^+, for $1 \leq i \leq n$ and $n \geq 1$. We do

this recursively in the following way:

a) The expression A..B, A and B in V, stands for the set of rules $A..BX_1/X_2$, $X_1..\varepsilon$, $X_2..f$ where X_1, X_2 are new variables, to be added to V, and which do not appear in any other rule in P.

b) The expression $A..B_1B_2$ stands for the set of rules $A..B_1B_2/X_1$, $X_1..f$ and X_1 is a new variable which does not appear in any other rule in P.

c) The expression $A..B_1B_2.....B_k$, for $k \geqslant 3$ stands for the set of rules $A..X_1B_k$, $X_1..B_1B_2....B_{k-1}$ where X_1 is a new variable which does not appear in any other rule in P.

d) The expression $A..\alpha/B$, A and B in V, α in V^+, stands for the set of rules $A..X_1X_2/B$, $X_1..\alpha$, $X_2..\varepsilon$ where X_1, X_2 are variables which do not appear in any other rule in P.

e) The expression $A..\alpha_1/\alpha_2/..../\alpha_n$, for $n \geqslant 2$ and α_i in V^+, $1 \leqslant i \leqslant n$, stands for the set of rules $A..\alpha_1/X_1$, $X_1..\alpha_2/\alpha_3..../\alpha_n$ and X_1 is a new variable which does not appear in any other rule in P.

3. We generalize further the form of the rules in P to include expressions $A..\alpha_1/\alpha_2/..../\alpha_n$, where α_i in $(V \cup \Sigma_e \cup \{f,\varepsilon\})^*$ for $1 \leqslant i \leqslant n$, $n \geqslant 1$. Let $\alpha_i = \beta a \gamma$ for some $1 \leqslant i \leqslant n$, β and γ in $(V \cup \Sigma_e \cup \{f,\varepsilon\})^*$ and a in $\Sigma_e \cup \{f,\varepsilon\}$. Then, in a recursive manner, the expression above stands for the set of rules $A..\alpha_1'/\alpha_2'/..../\alpha_n'$, $X_1..a$ where $\alpha_j' = \alpha_j$ if $j \neq i$ and $\alpha_i' = \beta X_1 \gamma$ and X_1 is a new variable which does not appear in any other rule in P.

As an example, consider the TS $R = (V,\Sigma,P,S,\$)$ where $V = \{S, A, X_a, X_\$\}$, $\Sigma = \{a\}$, $P = \{S..AX_\$,\ A..X_aX_aA/\epsilon,\ X_a..a,\ X_\$..\$\}$.

Given an input string $a^n\$$, it can be shown that if n is even then $A \underset{R}{\Rightarrow} (a^n\uparrow\$,0)$, otherwise $A \underset{R}{\Rightarrow} (a^{n-1}\uparrow a\$,0)$. Using now the rule for S, $S..AX_\$$ we get: for n even $A \underset{R}{\Rightarrow} (a^n\uparrow\$,0)$, $X_\$ \underset{R}{\Rightarrow} (\$\uparrow,0)$ and $S \underset{R}{\Rightarrow} (a^n\$\uparrow,0)$; for n odd $A \underset{R}{\Rightarrow} (a^{n-1}\uparrow a\$,0) X_\$ \underset{R}{\Rightarrow} (\uparrow a\$,\ 1)$ and $S \underset{R}{\Rightarrow} (\uparrow a^n\$,1)$ hence a^n is in T(R) if and only if n is even:

$$T(R) = \{a^n \mid n \geq 0 \text{ and } n \text{ even}\} .$$

Let $R = (V,\Sigma,P,S,\$)$ be a TS. We will show that the TS can be thought of as an automaton and for this purpose we define the "TS-automaton" A(R):

Definition 2.2 Let $R = (V,\Sigma,P,S,\$)$ be a TS.

The tape alphabet of A(R) is $\Gamma = V \cup \{X_1X_2/X_3 \mid X_i \text{ in } V \text{ or } X_i = \bar{A} \text{ and } A \text{ in } V, \text{ for } i = 1,2,3\}$.

The internal states of A(R) are $\{s,r\}$.

A configuration of A(R) is a 3-tuple (q,x,ω) where q is in $\{s,r\}$, $x = x_1\uparrow x_2$, x_1,x_2 in Σ_e^*, $\uparrow$ is a special symbol (which indicates the position of the read head on string x_1x_2), ω is in $(\Gamma \times N)^*$, N being the set of natural numbers. The position of $\uparrow$ in x is denoted by the function $p: \Sigma_e^*\uparrow\Sigma_e^* \to N$, as follows: if $x = x_1\uparrow x_2$, $|x_1| = k$ (by $|x|$ we denote the length of the string x), then $p(x) = k$.

We define the relation $\vdash_{A(R)}$ between two configurations α, β, and we write $\alpha \vdash_{A(R)} \beta$, as follows: assume $\alpha = (q,x,\omega)$, $\omega = (X_1,i_1)\ldots(X_k,i_k)$,

$\beta = (q', x', \omega')$, $x = x_1 \uparrow x_2$, $x' = x_1' \uparrow x_2'$ and $x_1 x_2 = x_1' x_2'$, q, q' in $\{s,r\}$, $\omega' = (X_1,i_1)\ldots(X_{k-1},i_{k-1})\ \omega''$.

1) Let $q = s$, $X_k = A$, A in V, A..BC/D in P; then $q' = s$, $x' = x$ and $\omega'' = (\bar{B}C/D,\ p(x))\ (B,\ p(x))$.

2) Let $q = s$, $X_k = A$, A in V, A..a in P, a in Σ_e; then if $x_2 = ax_3$ we have $x_1' = x_1 a$, $x_2' = x_3$, $q' = s$ and $\omega'' = \varepsilon$; if $x_2 = bx_3$, $b \neq a$, we have $x' = x$, $q' = r$ and $\omega'' = \varepsilon$.

3) If $q = s$, $X_k = A$, A in V, A..ε in P, then $x' = x$, $q' = s$ and $\omega'' = \varepsilon$.

4) If $q = s$, $X_k = A$, A in V, A..f in P, then $x' = x$, $q' = r$ and $\omega'' = \varepsilon$.

5) Let $X_k = \bar{A}B/C$, A,B,C in V; if $q = s$ then $q' = s$, $x' = x$ and $\omega'' = (A\bar{B}/C,\ i_k)\ (B,\ p(x))$; if $q = r$ then $q' = s$, $p(x') = i_k$ and $\omega'' = (AB/\bar{C},\ i_k)\ (C,\ i_k)$.

6) Let $X_k = A\bar{B}/C$, A,B,C in V; if $q = s$ then $q' = s$, $x' = x$ and $\omega'' = \varepsilon$. If $q = r$ then $q' = s$, $p(x') = i_k$, $\omega'' = (AB/\bar{C},i_k)\ (C,i_k)$.

7) Let $X_k = AB/\bar{C}$, A,B,C in V; if $q = s$ then $q' = s$, $x' = x$ and $\omega'' = \varepsilon$. If $q = r$ then $q' = r$, $p(x') = i_k$, $\omega'' = \varepsilon$.

If $\alpha \vdash_{A(R)} \beta$, we say A(R) makes one <u>move</u> from configuration α to configuration β. We write $\alpha \vdash^{*}_{A(R)} \beta$ if there are $\alpha_1, \alpha_2, \ldots, \alpha_n$, $\alpha = \alpha_1$, $\beta = \alpha_n$ and $\alpha_i \vdash_{A(R)} \alpha_{i+1}$, for $1 \leq i \leq n - 1$ and some n, the number of moves.

The <u>language accepted by A(R)</u> is $\{w | w \text{ in } \Sigma^*,\ (s,\uparrow w\$,(S,0)) \vdash^{*}_{A(R)} (s,\ w\$\uparrow,\varepsilon)\}$.

We will show now that the language accepted by A(R), for a given

R, is exactly T(R).

Theorem 2.1 The language accepted by A(R) is T(R).

Proof Let $R = (V,\Sigma,P,S,\$)$ be a TS.

Part One. First we show:

(a) If $A \overset{n}{\underset{R}{\Rightarrow}} (x_1 \uparrow x_2, 0)$, x_1x_2 in $\Sigma^*\$$, then for all x_3 in Σ^*, γ in $(\Gamma \times N)^*$ we have $(s, x_3 \uparrow x_1x_2, \gamma(A, |x_3|)) \overset{*}{\underset{A(R)}{\vdash}} (s, x_3x_1 \uparrow x_2, \gamma)$.

(b) If $A \overset{n}{\underset{R}{\Rightarrow}} (\uparrow x, 1)$, x in $\Sigma^*\$$, then for all y in Σ^*, γ in $(\Gamma \times N)^*$, we have $(s, y \uparrow x, \gamma(A, |y|)) \overset{*}{\underset{A(R)}{\vdash}} (r, y \uparrow x, \gamma)$. We prove (a) and (b) simultaneously, by induction on n.

n = 1(a) Assume the rule for A in P is A..a, a in $\Sigma_e \cup \{\epsilon\}$ and $A \underset{R}{\Rightarrow} (a \uparrow x_2, 0)$. From configuration $(s, x_3 \uparrow ax_2, \gamma(A, |x_3|))$, A(R) will go in one move to $(s, x_3a \uparrow x_2, \gamma)$.

(b) Assume the rule for A in P is A..a, a in Σ_e and $A \underset{R}{\Rightarrow} (\uparrow bx, 1)$, $b \neq a$. From configuration $(s, y \uparrow bx, \gamma(A, |y|))$, A(R) will go to $(r, y \uparrow bx, \gamma)$, according to Definition 2.2.

Finally, assume the rule for A is A..f, and $A \underset{R}{\Rightarrow} (\uparrow x, 1)$. Then $(s, y \uparrow x, \gamma(A, |y|)) \overset{}{\underset{A(R)}{\vdash}}_n (r, y \uparrow x, \gamma)$.

Induction step (a) $A \overset{n}{\underset{R}{\Rightarrow}} (x_1 \uparrow x_2, 0)$, $n > 1$. The first move must come from a rule of the form A..BC/D in P.

Case 1 $B \underset{R}{\Rightarrow} (y_1 \uparrow y_2x_2, 0)$, $C \underset{R}{\Rightarrow} (y_2 \uparrow x_2, 0)$, $x_1 = y_1y_2$. Consider in A(R), $(s, x_3 \uparrow y_1y_2x_2, \gamma(A, |x_3|))$. Using Definition 2.2, in one move we get the configuration in A(R) $(s, x_3 \uparrow y_1y_2x_2, (\bar{B}C/D, |x_3|))(B, |x_3|)$. Using the inductive hypothesis on (a) and the derivation $B \underset{R}{\Rightarrow} (y_1 \uparrow y_2x_2, 0)$ from above, we get $(s, x_3y_1 \uparrow y_2x_2, \gamma(\bar{B}C/D, |x_3|))$ and then $(s, x_3y_1 \uparrow y_2x_2, (B\bar{C}/D, |x_3|)\ (C, |x_3y_1|))$. Repeating the same argument for C we get

$(s, x_3y_1y_2\lceil x_2, \gamma(B\bar{C}/D, |x_3|))$ and finally the configuration $(s, x_3x_1\lceil x_2, \gamma)$.

Case 2 $B \underset{R}{\Rightarrow} (\lceil x_1x_2, 1)$, $D \underset{R}{\Rightarrow} (x_1\lceil x_2, 0)$. We have in A(R) in one move $(s, x_3\lceil x_1x_2, \gamma(A, |x_3|)) \underset{A(R)}{\vdash} (s, x_3\lceil x_1x_2, \gamma(\bar{B}C/D, |x_3|)\ (B, |x_3|))$. Using the inductive hypothesis on (b) this time, we get $(r, x_3\lceil x_1x_2, \gamma(\bar{B}C/D, |x_3|))$ and then $(s, x_3\lceil x_1x_2, \gamma(BC/\bar{D}, |x_3|)\ (D, |x_3|)) \overset{*}{\underset{A(R)}{\vdash}} (s, x_3x_1\lceil x_2, \gamma(BC/\bar{D}, |x_3|)) \underset{A(R)}{\vdash} (s, x_3x_1\lceil x_2, \gamma)$.

Case 3 $B \underset{R}{\Rightarrow} (y_1\lceil y_2, 0)$, $C \underset{R}{\Rightarrow} (\lceil y_2, 1)$, $D \underset{R}{\Rightarrow} (x_1\lceil x_2, 0)$ and $y_1y_2 = x_1x_2$. This case is treated similarly.

(b) $A \overset{n}{\underset{R}{\Rightarrow}} (\lceil x, 1)$, $n > 1$. The first step must come from a rule of the form $A..BC/D$ in P.

Case 1 $B \underset{R}{\Rightarrow} (\lceil x, 1)$, $D \underset{R}{\Rightarrow} (\lceil x, 1)$. We have in one move in A(R) $(s, y\lceil x, \gamma(A, |y|)) \underset{A(R)}{\vdash} (s, y\lceil x, \gamma(\bar{B}C/D, |y|)\ (B, |y|))$. The inductive hypothesis and $B \underset{R}{\Rightarrow} (\lceil x, 1)$ imply A(R) goes to $(r, y\lceil x, \gamma(\bar{B}C/D, |y|))$ and then $(s, y\lceil x, \gamma(BC/\bar{D}, |y|)\ (D, |y|))$. Using this time the inductive hypothesis on D, $D \underset{R}{\Rightarrow} (\lceil x, 1)$, we get the final configuration of A(R) as $(r, y\lceil x, \gamma)$.

Case 2 $B \underset{R}{\Rightarrow} (x_1\lceil x_2, 0)$, $C \underset{R}{\Rightarrow} (\lceil x_2, 1)$, $D \underset{R}{\Rightarrow} (\lceil x_1x_2, 1)$ and $x = x_1x_2$. This case is treated similarly.

Part Two. We want to prove:

(c) For all y in Σ^*, γ in $(\Gamma \times N)^*$, if $(s, y\lceil x_1x_2, \gamma(A, |y|)) \overset{*}{\underset{A(R)}{\vdash}} (s, yx_1\lceil x_2, \gamma)$, x_1x_2 in $\Sigma^*\$$, then $A \underset{R}{\Rightarrow} (x_1\lceil x_2, 0)$.

(d) For all y in Σ^*, γ in $(\Gamma \times N)^*$, if $(s, y\lceil x, \gamma(A, |y|)) \overset{*}{\underset{A(R)}{\vdash}} (r, y'\lceil x', \gamma)$, x in $\Sigma^*\$$, then $A \underset{R}{\Rightarrow} (\lceil x, 1)$, $y = y'$ and $x = x'$.

We prove (c), (d) by induction on n, the number of moves of the automaton A(R).

n = 1 (c) A one move derivation in this case implies, by Definition 2.2, $x_1 = a$ in $\Sigma_e \cup \{\varepsilon\}$ and A..a is in P. It follows that $A \underset{R}{\Rightarrow} (a \uparrow x_2, 0)$.

(d) By Definition 2.2 we have either $x_1 = b$, b in Σ_e, and A..a which implies $x = bx_2$, $A \underset{R}{\Rightarrow} (\uparrow b\, x_2, 1)$, or we have A..f in P which implies $A \underset{R}{\Rightarrow} (\uparrow x, 1)$.

Induction step: n > 1. (a) First we have: $(s, y\uparrow x_1x_2, \gamma(A, |y|)) \overset{*}{\underset{A(R)}{\vdash}}$ $(s, y\uparrow x_1x_2, \gamma(\bar{B}C/D, |y|)\ (B, |y|))$, where A..BC/D is in P. It is assumed in (c) that the tape cell with (B, |y|) will be eventually replaced by ε in some configuration of A(R).

Case 1 We assume that (B, |y|) is replaced by ε in state s; moreover, when (C, k) is written in its place, for some k, this one is also eventually erased and we assume that (C, k) is replaced by ε in state r. From the first assumption follows, by the inductive hypothesis (c), that if $x = x_1x_2$, and $(s, y\uparrow x, \gamma(\bar{B}C/D, |y|)\ (B, |y|)) \overset{*}{\underset{A(R)}{\vdash}}$ $(s, y\, x_3\uparrow x_4, \gamma(\bar{B}C/D, |y|))$, $x = x_3x_4$, then $B \underset{R}{\Rightarrow} (x_3\uparrow x_4, 0)$. Next A(R) goes to $(s, y\, x_3\uparrow x_4, \gamma(B\bar{C}/D, |y|)\ (C, |yx_3|))$ and finally to $(r, yx_3\uparrow x_4, \gamma(B\bar{C}/D, |y|))$. By the inductive hypothesis (d) follows $C \underset{R}{\Rightarrow} (\uparrow x_4, 1)$. Finally A(R) goes to $(s, y\uparrow x_1x_2, \gamma(BC/\bar{D}, |y|)\ (D, |y|)) \overset{*}{\underset{(A(R)}{\vdash}}$ $(s, yx_1\, x_2,)$. The last part implies $D \underset{R}{\Rightarrow} (x_1\uparrow x_2, 0)$. By Definition 2.1 we have $A \underset{R}{\Rightarrow} (x_1\uparrow x_2, 0)$.

Case 2 When (B, |y|) and (C, k) are replaced by ε, A(R) is in state s.

Case 3 When (B, |y|) is replaced by ε A(R) is in state c.

The cases 2 and 3 are treated similarly.

(d) First we have $(s, y\uparrow x, \gamma(A, |y|)) \vdash^{*}_{A(R)} (s, y\uparrow x, \gamma(\bar{B}C/D, |y|)(B, |y|))$, where A..BC/D is in P.

Case 1 We assume (B, |y|) is replaced by ε in state r. Then we get the configuration $(r, y\uparrow x, \gamma(\bar{B}C/D, |y|))$ which goes in one move to $(s, y\uparrow x, \gamma(BC/\bar{D}, |y|)\ (D, |y|))$. It follows that (D, |y|) must also be replaced by ε in state r (otherwise the symbol (A, |y|) would be finally replaced by ε in state s). By the inductive hypothesis (d) we conclude $B \underset{R}{\Rightarrow} (\uparrow x, 1)$ and also $D \underset{R}{\Rightarrow} (\uparrow x, 1)$. The final configuration is $(s, y\uparrow x, \gamma)$ and hence $y' = y$, $x' = x$. Also it follows, since A..BC/D, $B \underset{R}{\Rightarrow} (\uparrow x, 1)$, $D \underset{R}{\Rightarrow} (\uparrow x, 1)$ that $A \underset{R}{\Rightarrow} (\uparrow x, 1)$.

Case 2 (B, |y|) is replaced by ε in state s. This case is treated similarly.

From (a) if $S \underset{R}{\Rightarrow} (x\uparrow, 0)$, x in $\Sigma^{*}\$$ then $(s, \uparrow x, (S,0)) \vdash^{*}_{A(R)} (s, x\uparrow, \varepsilon)$. From (c) follows the reverse, i.e., if x is accepted by A(R) then it belongs to T(R). QED

We will now prove a "uniqueness" result, which in conjunction with the previous theorem will be used in establishing later results.

Lemma 2.1 Let $R = (V, \Sigma, P, S, \$)$ be a TS. Assume that $(q, x\uparrow y, \gamma) \vdash^{*}_{A(R)} (q_1, x_1\uparrow y_1, \gamma_1)$ and $(q, x\uparrow y, \gamma) \vdash^{*}_{A(R)} (q_2, x_2\uparrow y_2, \gamma_2)$ for some q, q_1, q_2 in {s,r}, $xy = x_1y_1 = x_2y_2$ in $\Sigma^{*}\$$, $\gamma, \gamma_1, \gamma_2$ in $(\Gamma \times N)^{*}$ and in configurations $(q_1, x_1\uparrow y_1, \gamma_1)$, $(q_1, x_2\uparrow y_2, \gamma_2)$ A(R) has no next move. Then $q_1 = q_2$, $x_1 = x_2$ and $\gamma_1 = \gamma_2$.

Proof According to Definition 2.2, in any configuration of A(R) there is at most one possible move and therefore A(R) is deterministic. The lemma follows immediately. QED

III. TS AND DETERMINISTIC PDA'S

In this chapter we define a subclass of TS, the "well-formed TS", and we show that the class of languages they recognize include all deterministic cfl's ([4], [5], [6]).

Definition 3.1 A well-formed TS (or shortly wfTS) $R = (V,\Sigma,P,S,\$)$ is a TS with the property: for all x in Σ^* either $S \underset{R}{\Longrightarrow} (x\$\uparrow,0)$ or $S \underset{R}{\Longrightarrow} (\uparrow x\$, 1)$. (We mention at this point that, as we will show later, given a TS R it is undecidable whether R is also a wfTS).

We first give the definition of a DPDA (the notation used is similar to the one in [14]):

Definition 3.2 A deterministic pushdown automata (DPDA) is a 7-tuple $Q = (K,\Sigma,\Gamma,\delta,Z_o,q_o,F)$, where:

K is a finite set of states.

Σ is a finite set of input symbols.

Γ is a finite set of pushdown symbols.

δ is a mapping from $K \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ into $K \times \Gamma^*$ such that for each q in K and Z in Γ, if $\delta(q, \epsilon, Z) \neq \emptyset$ then $\delta(q,a,Z) = \emptyset$ for all a in Σ.

Z_o is an element of Γ.

q_o is in K (the initial state)

F is a subset of K (the set of final states).

Let $\vdash_Q^*$ be the relation on $K \times \Sigma^* \times \Gamma^*$ defined as follows:

1) For a in $\Sigma \cup \{\varepsilon\}$, Z in Γ, if $\delta(q,a,Z) = (p,\gamma)$ then we write $(q,aw,\alpha Z) \vdash_Q (p,w,\alpha\gamma)$, and this is called a move.

2) For α and β in Γ^* and x_i in $\Sigma \cup \{\varepsilon\}$, $1 \leq i \leq k$, we write $(q, x_1 \ldots x_k w, \alpha) \vdash_Q^* (p,w,\beta)$ if there exist $q_1 = q, \ldots, q_{k+1} = p$ in K and $\alpha_1 = \alpha, \ldots, \alpha_{k+1} = \beta$ in Γ^* such that $(q_i, x_i \ldots x_k w, \alpha_i) \vdash_Q (q_{i+1}, x_{i+1} \ldots x_k w, \alpha_{i+1})$ for $1 \leq i \leq k$.

A word w is <u>accepted</u> by Q if $(q_o,w,Z_o) \vdash_Q^* (q,\varepsilon,\alpha)$ for some q in F and α in Γ^*. The set of all words accepted by Q is denoted by $T(Q)$.

<u>Theorem 3.1</u> Let $Q_1 = (K_1,\Sigma,\Gamma,\delta_1,Z_o,q_o,F_1)$ be a DPDA and let the language accepted by Q_1 be $T(Q_1)$. There exists a wfTS R such that $T(R) = T(Q_1)$.

<u>Proof</u> The language accepted by Q_1 by final state is $T(Q_1) = \{x \mid x \text{ in } \Sigma^* \text{ and } (q_o,x,Z_o) \vdash_{Q_1}^* (q,\varepsilon,\gamma) \text{ for any } \gamma \text{ in } \Gamma^* \text{ and } q \text{ in } F\}$. According to [2], p. 168, we can assume that for all x in Σ^*, there are p in K and α in Γ^* such that $(q_o,x,Z_o) \vdash_{Q_1}^* (p,\varepsilon,\alpha)$, which means that the automaton Q_1 always scans the input string. We will construct from Q_1 a new DPDA Q which will accept $x\$$, for some x in Σ^*, if and only if x is in $T(Q_1)$ and in addition Q will always erase its storage tape before accepting or rejecting. Let $\Sigma = \{a_i \mid 1 \leq i \leq m\}$ and $K_1 = \{q_i \mid 1 \leq i \leq n-2\}$. Then $Q = (K,\Sigma,\Gamma,\delta,Z_o,q_o,F)$ where $K = \{q_i \mid 1 \leq i \leq n\}$, K contains two new states q_{n-1}, q_n, $F = \{q_n\}$. Q will simulate Q_1 until the endmarker is reached; then if Q_1 is in an accepting state, Q goes to q_n which erases the tape

and accepts the string. If, however, Q_1 is in a non-accepting state, then Q goes to q_{n-1} which will erase the tape. Then: $\delta(q,\$,Z) = (q_n, Z)$ if q is in F and $\delta(q,\$,Z) = (q_{n-1}, Z)$ if q is not in F, for all Z in Γ. Also $\delta(q_n,\varepsilon,Z) = (q_n,\varepsilon)$ and $\delta(q_{n-1},\varepsilon,Z) = (q_{n-1},\varepsilon)$ for all Z in Γ. Now consider the TS $R = (V,\Sigma,P,[q_0 Z_0 q_n],\$)$ where: V includes the set $\{[q_i\ Z\ q_j] |\ q_i,q_j$ in K, Z in $\Gamma\}\cup\{[q_i Z q_j;a] | q_i,q_j$ in K, Z in Γ, a in $\Sigma_e\}$; V also contains variables which are implicitly defined in the handshort notation of the rules in P. (A variable of the form $[q_i Z q_j]$ will have outcome 0 and will recognize a string x only if Q, in state q_i and with Z on its storage tape will eventually erase Z in state q_j having scanned substring x on its input tape; for variables $[q_i\ Z\ s]$, where $s \neq q_j$, the outcome will be 1).

P contains the following rules (according to [14] we can assume for Q that if $\delta(q,a,Z) = (p,\gamma)$ for some q, p in K, a in $\Sigma_e\cup\{\varepsilon\}$, Z in Γ, then $|\gamma| \leq 2$):

1) if $\delta(q_i,\varepsilon,Z) = \emptyset$ (i.e. there are no ε-rules for q_i in K, Z in Γ) then P contains

(P1) $[q_i Z q_j] .. a_1[q_i Z q_j;a_1]/a_2[q_i Z q_j;a_2]/...../a_m[q_i Z q_j;a_m]$ for all q_j in K (variables of the form $[q_i Z q_j;a]$ are used to register the fact that a symbol a has been recognized).

2) if $\delta(q_i,a_k,Z) = (q_\ell,X_\ell Y_\ell)$, X_ℓ,Y_ℓ in Γ, then

(P2) $[q_i Z q_j;a_k]..[q_\ell X_\ell q_1][q_1 Y_\ell q_j]/..../[q_\ell X_\ell q_n][q_n Y_\ell q_j]$, for all q_j in K.

3) if $\delta(q_i,a_k,Z) = (q_\ell,X_\ell)$, X_ℓ in Γ, then

(P3) $[q_i Z q_j;a_k]..[q_\ell X_\ell q_j]$ for all q_j in K.

4) if $\delta(q_i,a_k,Z) = (q_\ell,\varepsilon)$, then

(P4) $[q_iZq_\ell;a_k] .. \varepsilon$, and $[q_iZq_j;a_k] .. f$, for all q_j in K, $q_j \neq q_\ell$.

5) if $\delta(q_i,\varepsilon,Z) = (q_\ell,X_\ell Y_\ell)$, then

(P5) $[q_iZq_j] .. [q_\ell X_\ell q_1][q_1 Y_\ell q_j]/...../[q_\ell X_\ell q_n][q_n Y_\ell q_j]$ for all q_j in K.

6) if $\delta(q_i,\varepsilon,Z) = (q_\ell,X_\ell)$, then

(P6) $[q_iZq_\ell] .. [q_\ell X_\ell q_j]$, for all q_j in K.

7) if $\delta(q_i,\varepsilon,Z) = (q_\ell,\varepsilon)$, then

(P7) $[q_iZq_\ell] .. \varepsilon$, and $[q_iZq_j] .. f$, for all q_j in K, $q_j \neq q_\ell$.

Part One .We show that if $(q,x,Z) \overset{*}{\underset{Q}{\vdash}} (p,\varepsilon,\varepsilon)$, x in Σ_e^*, Z in Γ, then $[qZp] \underset{R}{\Longrightarrow} (x\uparrow,0)$ and $[qZs] \underset{R}{\Longrightarrow} (\uparrow x, 1)$ for all s in K and $s \neq p$.

The proof is by induction on n', the number of moves of Q.

Base: $n' = 1$ Case 1 $x = a_i, a_i$ in Σ_e and $\delta(q,a_i,Z) = (p,\varepsilon)$. Then $[qZp;a_i] .. \varepsilon$, $[qZs;a_i] .. f$ for all $s \neq p$ and $[qZp] .. a_1[qZp;a_1]/...../a_m[qZp;a_m]$. It follows $[qZp;a_i] \underset{R}{\Longrightarrow} (\uparrow, 0)$ and $[qZp] \underset{R}{\Longrightarrow} (a_i\uparrow,0)$. Also in P we have $[qZs] .. a_1[qZs;a_1]/...../a_m[qZs;a_m]$. It follows $[qZs; a_i] \underset{R}{\Longrightarrow} (\uparrow, 1)$ for $s \neq p$ and hence $[qZs] \Rightarrow (\uparrow a_i, 1)$.

Case 2 $x = \varepsilon$ and $\delta(q,\varepsilon,Z) = (p,\varepsilon)$. $[qZp] .. \varepsilon$, $[qZs] .. f$ for $s \neq p$ and hence $[qZp] \underset{R}{\Longrightarrow} (\uparrow,0)$, $[qZs] \underset{R}{\Longrightarrow} (\uparrow, 1)$ for all $s \neq p$. These two cases are the only possible ones and the base of the induction is thus proved.

Induction step Case 1 We first assume the first move from configuration (q,x,Z) is not an ε-move. Suppose $x = a_iy$, a_i in Σ, and $\delta(q,a_i,Z) = (p',XY)$, X,Y in Γ, p' in K. Then $(q,a_iy,Z) \underset{Q}{\vdash} (p',y,XY)$. For some

p'' in k we will also have $(p', y_1y_2, XY) \overset{*}{\vdash}_Q (p'', y_2, Y) \overset{*}{\vdash}_Q (p, \varepsilon, \varepsilon)$. By induction $[p'Xp''] \underset{R}{\Rightarrow} (y_1\uparrow y_2, 0)$ and $[p''Yp] \underset{R}{\Rightarrow} (y_2\uparrow, 0)$; also $[p'Xs] \underset{R}{\Rightarrow} (\uparrow y_1y_2, 1)$, $s \neq p''$ and $[p''Ys] \underset{R}{\Rightarrow} (\uparrow y_2, 1)$, $s \neq p$. We have the rule $[qZp; a_i]..[p'Xq_1][q_1Yp]/.../[p'Xq_n][q_nYp]$. Let k be such that $p'' = q_k$. From above $[p'Xq_j] \underset{R}{\Rightarrow} (\uparrow y_1y_2, 1)$ for all $j \neq k$.

We also have $[p''Yp] \underset{R}{\Rightarrow} (y_2\uparrow, 0)$ and together with $[p'Xp''] \underset{R}{\Rightarrow} (y_1\uparrow y_2, 0)$ we get $[qZp; a_i] \underset{R}{\Rightarrow} (y_1y_2\uparrow, 0)$. Finally the rule for $[qZp]$: $[qZp]..$ $a_1[qZp; a_1]/...../a_m[qZp; a_m]$ gives us $[qZp] \underset{R}{\Rightarrow} (a_iy_1y_2\uparrow, 0)$.

Consider now the rule for $[qZs; a_i]$, $s \neq p$. $[qZs; a_i]$ .. $[p'X\, q_1][q_1Ys]/ .... / [p'Xq_n][q_nYs]$. We know already that $[p'Xq_j] \Rightarrow (\uparrow y_1y_2, 1)$ for $j \neq k$, $p'' = q_k$ and $[p'Xq_k] \underset{R}{\Rightarrow} (y_1\uparrow y_2, 0)$. We also have $[q_kYs] \underset{R}{\Rightarrow} (\uparrow y_2, 1)$, $s \neq p$ which implies $[qZs; a_i] \underset{R}{\Rightarrow} (\uparrow y_1y_2, 1)$ and finally $[qZs] \underset{R}{\Rightarrow} (\uparrow a_iy_1y_2, 1)$ for all $s \neq p$.

Next, assume $\delta(q, a_i, Z) = (p', X)$. Then $[qZp; a_i]$ .. $[p'Xp]$ and by induction $[p'Xp] \underset{R}{\Rightarrow} (y\uparrow, 0)$, hence $[qZp] \underset{R}{\Rightarrow} (a_iy\uparrow, 0)$; $[qZs; a_i]$ .. $[pXs]$ and $[pXs] \underset{R}{\Rightarrow} (\uparrow y, 1)$, all $s \neq p$, hence $[qZs] \underset{R}{\Rightarrow} (\uparrow a_iy, 1)$ for all $s \neq p$. If $\delta(q, a_i, Z) = (p', \varepsilon)$ then $[qZp', a_i]$ .. ε and $[qZs; a_i]$ .. f, all $s \neq p'$ and the theorem follows.

Case 2 First move is an ε-move, $\delta(q, \varepsilon, Z) = (p', XY)$. Then let $x = x_1x_2$ such that $(p', x_1x_2, XY) \overset{*}{\vdash}_Q (p'', x_2, Y) \overset{*}{\vdash}_Q (p, \varepsilon, \varepsilon)$. By induction $[p'Xp''] \underset{R}{\Rightarrow} (x_1\uparrow x_2, 0)$ and $[p'Xs] \underset{R}{\Rightarrow} (\uparrow x_1x_2, 1)$, all $s \neq p''$; $[p''Yp] \underset{R}{\Rightarrow}$ $(x_2\uparrow, 0)$ and $[p''Ys] \underset{R}{\Rightarrow} (\uparrow x_2, 1)$, all $s \neq p$. The rule for $[qZp]$: $[qZp]..[p'Xq_1][q_1Yp]/...../[p'Xq_n][q_nYp]$; let k be such that $q_k = p''$. Then $[q'Xq_k] \underset{R}{\Rightarrow} (x_1\uparrow x_2, 0)$, $[q_kYp] \underset{R}{\Rightarrow} (x_2\uparrow, 0)$ and from above follows the

theorem. Other cases are treated similarly.

Part Two We show that if $[qZp] \overset{n'}{\underset{R}{\Rightarrow}} (x\uparrow, 0)$, x in Σ^*, p,q in K, Z in Γ, then $(q,x,Z) \overset{*}{\underset{Q}{\vdash}} (p,\varepsilon,\varepsilon)$.

The proof is by induction on n'.

$n' = 1$. The only possible case is $[qZp]..\varepsilon$ and $[qZp] \underset{R}{\Rightarrow} (\uparrow,0)$, $x = \varepsilon$. Hence, $\delta(q,\varepsilon,Z) = (p,\varepsilon)$ and $(q,\varepsilon,Z) \underset{Q}{\vdash} (p,\varepsilon,\varepsilon)$.

Induction step Case 1 Suppose the rule for [qZp] is: $[qZp]..a_1[qZp; a_1]/$ $...../a_m[qZp; a_m]$; also $[qZp; a_i]..[p'Xq_1][q_1Yp]/...../[p'Xq_n][q_nYp]$; Then there are x_1, x_2 and k such that $x = a_i x_1 x_2$ and $[p'Xq_k] \underset{R}{\Rightarrow} (x_1\uparrow x_2, 0)$, $[q_kYp] \underset{R}{\Rightarrow} (x_2\uparrow, 0)$, $\delta(q,a_i,Z) = (p',XY)$, Z,X,Y in Γ, q,p' in K. By induction $(p',x_1x_2,XY) \overset{*}{\underset{Q}{\vdash}} (q_k,x_2,Y) \overset{*}{\underset{Q}{\vdash}} (p,\varepsilon,\varepsilon)$ which proves the case.

Case 2 The rule for [qZp] is: $[qZp]..[p'Xq_1][q_1Yp]/...../$ $[p'Xq_n][q_nYp]$. Then, there are x_1, x_2, k such that $x = x_1x_2$, $[p'Xq_k] \underset{R}{\Rightarrow}$ $(x_1\uparrow x_2, 0)$, $(q_kYp] \underset{R}{\Rightarrow} (x_2\uparrow, 0)$, $\delta(q,\varepsilon,Z) = (p',XY)$. By induction $(q,x_1x_2,Z) \underset{Q}{\vdash} (p',x_1x_2,XY) \overset{*}{\underset{Q}{\vdash}} (q_k,x_2,Y) \overset{*}{\underset{Q}{\vdash}} (p,\varepsilon,\varepsilon)$.

Now we can show that for all x in Σ^*, x\$ is in T(Q) iff x is in T(R). Assume x\$ is in T(Q). Then $(q_o,x\$,Z_o) \overset{*}{\underset{Q}{\vdash}} (q_n,\varepsilon,\varepsilon)$. Using Part One we get $[q_oZ_oq_n] \underset{R}{\Rightarrow} (x\$\uparrow, 0)$, $[q_oZ_os] \underset{R}{\Rightarrow} (\uparrow x\$, 1)$, for all $s \neq q_n$.

Assume now x is in T(R); then $[q_oZ_oq_n] \underset{R}{\Rightarrow} (x\$\uparrow, 0)$ and using Part Two we get $(q_o,x\$,Z_o) \overset{*}{\underset{Q}{\vdash}} (q_n, \varepsilon, \varepsilon)$.

Part Three We have to show that R is a wfTS, that is for all x in Σ^* either $[q_oZ_oq_n] \underset{R}{\Rightarrow} (x\$\uparrow, 0)$ or $[q_oZ_oq_n] \underset{R}{\Rightarrow} (\uparrow x\$, 1)$. By the definition

of Q we must have for all x in Σ^* either $(q_o,x\$,Z_o) \vdash_Q^* (q_n,\epsilon,\epsilon)$ or $(q_o,x\$,Z_o) \vdash_Q^* (q_{n-1},\epsilon,\epsilon)$ and using new Part One the theorem follows. QED

IV. FAILURE TYPES AND ADDITIONAL PROPERTIES FOR TS

In this chapter we discuss the ways in which the TS could fail to recognize an input string. These "failure types" are: recognition, subroutine, end, partial-acceptance and loop failures. We show that partial-acceptance and end failures can be eliminated; moreover, subroutine failures can be replaced by loop failures so that for any TS, an equivalent TS can be constructed which has only recognition or loop failures. Using these results we prove some closure properties: The TSL are closed under intersection, the complement of a wfTSL is a TSL, the wfTSL are closed under union. Also, we show the following problems to be unsolvable: the Σ^*-problem for wfTS, the emptiness problem for wfTS, the problem of deciding whether a given TS is a wfTS.

We start by defining the various types of failures:

Definition 4.1 Let $R = (V,\Sigma,P,S,\$)$ be a TS. For A in V, x in Σ^*:

1) A has a recognition failure (or simply failure) on x if $(s,\vdash x\$,(A,o)) \vdash_{A(R)}^* (r,\vdash x\$,\epsilon)$.

2) A has a subroutine failure on x if either there is no rule for A in P or $(s,\vdash x\$,(A,o)) \vdash_{A(R)}^* (s,x_1\vdash x_2,\gamma(B,n))$ for some γ in $(\Gamma xN)^*$, n in N, B in V such that $x_1x_2 = x\$$ and there is no rule for B in P.

3) A has an end failure on x if $(s,\uparrow x\$,(A,o)) \vdash^{*}_{A(R)} (s,x\$\uparrow,\gamma(B,n))$ for some γ in $(\Gamma xN)^{*}$, n in N, B in V, and the rule for B in P is B..a, for some a in Σ_1 (In case 2) above A(R) halts because no rule is available for some variable. In case 3) A(R) halts because the read head "falls off" the input tape).

4) R has a partial-acceptance (p-a) failure on x if $(s,\uparrow x_1x_2\$, (S,o)) \vdash^{*}_{A(R)} (s,x_1\uparrow x_2\$,\varepsilon)$ for some x_2 in Σ^{*}. (We will say equivalently, that S has a p-a failure on x. The definition of a p-a failure shows that if S has outcome 0 but has not scanned the whole string, the string is rejected and it does not belong to T(R)).

5) A has a loop failure on x if A(R) in configuration $(s,\uparrow x\$,(A,0))$ can make an unbounded number of moves.

In the following theorem we will show that those are the only possible failures.

Theorem 4.1 Let $R = (V,\Sigma,P,S,\$)$ be a TS and x in Σ^{*}, x is not in T(R). If S does not have on x a subroutine, end, partial-acceptance or loop failure then S has a recognition failure on x.

Proof A(R) in configuration $(s,\uparrow x\$,(S,o))$ can only make a finite number of moves, since otherwise S would have a loop failure on x. By Lemma 2.1 there is a unique configuration $(q,x_1\uparrow x_2,\gamma)$ such that $x_1x_2 = x\$$, $(s,\uparrow x\$,(S,o)) \vdash^{*}_{A(R)} (q,x_1\uparrow x_2,\gamma)$ and A(R) has no move in this configuration.

First, assume $\gamma \neq \varepsilon$; by the definition of A(R) we must have $\gamma = \gamma_1(B,n)$ for some B in V, n in N (since otherwise a move is possible). Also, we must have a rule for B in P, since otherwise we would have a subroutine

failure. The rule for B cannot be B..ε or B..f; if the rule is B..a, for some a in Σ_e, a move is still possible unless $x_2 = \varepsilon$ in which case we have an end failure. Therefore the assumption $\gamma \neq \varepsilon$ leads to a contradiction.

Assume now $\gamma = \varepsilon$; we cannot have q = s, since we would have a p-a failure. If q = r then, according to the definition of A(R), we have backtracking and $x_1 = \varepsilon$. We conclude that $(s,\lceil x\$,(S,o)) \vdash^{*}_{A(R)}$ $(r,\lceil x\$,\varepsilon)$ and we have a recognition failure. QED

Consider a variable A which has the rule A..ε .This variable has the following property: if we start A(R) in configuration $(s,\lceil x,(A,o))$, for some x in $\Sigma^*\$\cup\{\varepsilon\}$, the storage tape is eventually erased and no input symbol is checked for a match (we notice x can be ε). In other words, when A(R) has scanned the input string and has also successfully matched the endmarker, every variable called afterwards has to have this property if A(R) is to accept the input, since the first attempt to match a symbol in Σ_e will cause the read-head to "fall off" the input tape and A(R) will halt.

Next we study the set of variables in V which have this property and which belong to one of the sets U(R), V(R):

<u>Definition 4.2</u> Let R be a TS, R = (V,Σ,P,S,$). Construct <u>U(R)</u>⊆ V and <u>V(R)</u>⊆ V as follows:

1) $U_o = \{A|A \text{ in } V, A..\varepsilon \text{ in } P\}, V_o = \{A|A \text{ in } V, A..f \text{ in } P\}$.

2) $U_{i+1} = U_i \cup \{A|$ if A..BC/D in P for some B,C,D in V then (B,C in U_i) or (B in V_i and D in U_i) or (B,D in U_i and C in V_i)$\}$, $V_{i+1} = V_i \cup \{A|$ if A..BC/D in P for some B,C,D in V then (B,D in V_i) or (B in U_i and C,D in V_i)$\}$, for $i \geq 0$.

3) Let I be the smallest integer such that $U_{I+1} = U_I$ and $V_{I+1} = V_I$ (the existence of I is assured by the finiteness of V). Let $U(R) = U_I$, $V(R) = V_I$.

The following lemma gives a precise description to the mentioned property of the elements of U(R), V(R):

Lemma 4.1 a) $(s,\uparrow,(A,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,\varepsilon)$ iff A in U(R).

b) $(s,\uparrow,(A,o)) \overset{*}{\underset{A(R)}{\vdash}} (r,\uparrow,\varepsilon)$ iff A in V(R).

Proof

The "only if" part. The proof is by induction on n, the number of moves in A(R).

n = 1 In (a) we must have the rule A..ε, hence A in U_o. For (b) we must have the rule A..f, hence A is in V_o.

Induction step (a) Assume A..BC/D is the rule for A. Then several cases arise. First let A succeed through B and C, that is $(s,\uparrow,(B,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,\varepsilon)$, $(s,\uparrow,(C,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,\varepsilon)$. By induction B is in U(R) and C is in U(R), and by construction of U follows that A is in U(R). All other cases are similarly treated.

(b) Assume A..BC/D is the rule for A. Consider first the case B and D fail: $(s,\uparrow,(B,o)) \overset{*}{\underset{A(R)}{\vdash}} (r,\uparrow,\varepsilon)$, $(s,\uparrow,(D,o)) \overset{*}{\underset{A(R)}{\vdash}} (r,\uparrow,\varepsilon)$. By induction B is in V(R) and D is in V(R). We finally get that A is in V(R). The other case is similar.

"if" Consider the sets U_o, $U_1,\ldots,U_I$ and $V_o,V_1,\ldots,V_I$. Given a variable A in U(R) we can associate with A an integer i such that A is the U_i and not in U_j for $j < i$. Similarly, if

A is in V(R), let i be such that A is in V_i and not in V_j for all $j < i$. The proof is by induction on i associated with A in V.

Base (a) i = o or A in U_o; the theorem follows.

(b) A in V_o and the theorem follows.

Induction step (a) Assume A is in U_i. Consider first the case where A..BC/D in P and A is in U_i because B,C in U_{i-1}. By induction $(s,\uparrow,(B,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,\varepsilon)$ and $(s,\uparrow,(c,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,\varepsilon)$. It follows $(s,\uparrow,(A,o)) \underset{A(R)}{\vdash} (s,\uparrow,(B\bar{C}/D,o)(B,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,(C,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,\varepsilon)$. All other cases are similarly treated.

(b) A in V_i, A..BC/D in P. Consider the case where B is in U_{i-1}, C,D in V_{i-1}. We apply the inductive hypothesis to obtain $(s,\uparrow,(B,o)) \overset{*}{\underset{A(R)}{\vdash}} (s,\uparrow,\varepsilon)$ and also $(s,\uparrow,(C,o)) \overset{*}{\underset{A(R)}{\vdash}} (r,\uparrow,\varepsilon)$ and a similar expression for D. The theorem follows. QED

In order to prove some important properties of the TS, such as closure under intersection, we have to show that partial-acceptance failures can be eliminated. We make use of the previous lemma in order to prove:

Theorem 4.2 Given a TS R = (V,Σ,P,S,$), there is a TS R' which has no p-a failures and T(R) = T(R').

Proof Consider the TS R' = (V',Σ,P',$\bar{S}$,$) where V' = {A,$\bar{A}$| all A in V} ∪{J} and J is not in V. The significance of the pair of variables A, $\bar{A}$ in V' corresponding to the variable A in V is the following: A in R'

behaves like A in R, accepting the same strings; however $\bar{A}$ will accept only strings in $\Sigma^*\$$ and only those which are also accepted by A. For example, assume the rule for S in R is S..AB/C; first we notice that in R', we have to use $\bar{S}$ as the distinguished symbol. Then (assume for the moment that B is not in U(R)) on a recursive argument we have to write the rule for $\bar{S}$ in R' as: $\bar{S}..A\bar{B}/\bar{C}$. In other words we keep track of the variable which accepts the last symbol of the string and make sure this one is the endmarker. Formally, P' is formed as follows:

1) if A..ε is in P, then P' contains $\bar{A}..J$ and A..ε.

2) if A..a is in P, a in Σ, then P' has $\bar{A}..aJ$ and A..a.

3) if A..\$ is in P, then $\bar{A}..\$$, A..\$ are in P'.

4) if A..f is in P, then P' contains $\bar{A}..f$, A..f.

5) if A..BC/D is in P and C is not in U(R), then P' contains $\bar{A}..B\bar{C}/\bar{D}$ and A..BC/D.

6) if A..BC/D is in P and C is in U(R), then P' contains $\bar{A}..\bar{B}C/\bar{D}$ and A..BC/D.

The rest of the proof follows in three parts:

Part One First we show: for all x_1x_2 in Σ_e^*, i in {0,1}, $A \overset{*}{\underset{R}{\Rightarrow}} (x_1\uparrow x_2, i)$ iff $A \overset{*}{\underset{R'}{\Rightarrow}} (x_1\uparrow x_2, i)$. The proof is easily obtained by induction, first on the number of steps in the derivation in R and then for the "if" part, the induction is on the number of steps in the derivation in R'.

Part Two For all x in Σ^*,

(a) $A \overset{*}{\underset{R}{\Rightarrow}} (x\$\uparrow, o)$ iff $\bar{A} \overset{*}{\underset{R'}{\Rightarrow}} (x\$\uparrow, o)$

(b) $A \overset{*}{\underset{R}{\Rightarrow}} (\uparrow x\$, 1)$ iff $\bar{A} \overset{*}{\underset{R'}{\Rightarrow}} (\uparrow x\$, 1)$

The "only if" part By induction on the number of steps in the derivation in R.

Base (a) A one step derivation implies $A \underset{R}{\Rightarrow} (\$\uparrow, o)$, and A..$ in P. We get $\bar{A}$..$ in P', $\bar{A} \underset{R'}{\Rightarrow} (\$\uparrow, o)$.

(b) Several cases arise. If A..f is in P, then $\bar{A}$..f is in P' and the theorem holds. If A..a is in P, a in Σ, then $\bar{A}$..aJ is in P' and in R' we get $\bar{A} \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$. Finally, the case A..$ in P is also easily verified.

Induction step (a) Again, several cases are possible:

Case 1 A..BC/D is in P, C is not in U(R).

1a) $B \underset{R}{\Rightarrow} (x_1 \uparrow x_2\$, 0)$, $C \underset{R}{\Rightarrow} (x_2\$\uparrow, 0)$ for some x_1 in Σ^*, $x_1x_2 = x$. Then by induction $\bar{C} \underset{R'}{\Rightarrow} (x_2\$\uparrow, 0)$. Also, $\bar{A}$..$B\bar{C}/\bar{D}$ is in P', $B \underset{R'}{\Rightarrow} (x_1 \uparrow x_2\$, 0)$, hence $\bar{A} \underset{R'}{\Rightarrow} (x\$\uparrow, 0)$.

1b) $B \underset{R}{\Rightarrow} (\uparrow x\$, 1)$, $D \underset{R}{\Rightarrow} (x\$\uparrow, 0)$. By induction $\bar{D} \underset{R'}{\Rightarrow} (x\$\uparrow, 0)$ and we get $\bar{A} \underset{R'}{\Rightarrow} (x\$\uparrow, 0)$.

1c) Similar.

Case 2 A..BC/D in P, C in U(R). This case is similarly treated.

(b) Given $A \underset{R}{\Rightarrow} (\uparrow x\$, 1)$.

Case 1 A..BC/D is in P, C is not in U(R).

1a) $B \underset{R}{\Rightarrow} (\uparrow x\$, 1)$, $D \underset{R}{\Rightarrow} (\uparrow x\$, 1)$. By induction $\bar{D} \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$. Also we have $\bar{A}$..$B\bar{C}/\bar{D}$ in P' and $B \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$, hence $\bar{A} \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$.

1b) $B \underset{R}{\Rightarrow} (x_1 \uparrow x_2\$, 0)$, $C \underset{R}{\Rightarrow} (\uparrow x_2\$, 1)$, $D \underset{R}{\Rightarrow} (\uparrow x\$, 1)$, for some x_1 in Σ^*, $x_1x_2 = x$. By induction $\bar{C} \underset{R'}{\Rightarrow} (\uparrow x_2\$, 1)$, $\bar{D} \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$. $\bar{A}$..$B\bar{C}/\bar{D}$ is in P' and $B \underset{R'}{\Rightarrow} (x_1 \uparrow x_2\$, 0)$, hence $\bar{A} \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$.

Case 2 A..BC/D in P, C in U(R). The only possibility is: $B \underset{R}{\Rightarrow} (\uparrow x\$, 1)$ and $D \underset{R}{\Rightarrow} (\uparrow x\$, 1)$. We have $\bar{A}..\bar{B}C/\bar{D}$ in P' and applying the inductive hypothesis we get $\bar{A} \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$.

"if" By induction on the number of steps in the derivation in R'. This part of the proof is similar to the one above.

Part three We will show R' has no partial-acceptance failures by showing that for no A in V and no x_1, x_2 in Σ^* we have $\bar{A} \underset{R'}{\Rightarrow} (x_1 \uparrow x_2\$, 0)$. The proof is by induction on the number of steps in the derivation in R'.

Base Assume we have a one step derivation in R'. The rule in P' corresponding to the derivation step cannot be of the form $\bar{A}..aJ$ or $\bar{A}..J$ or $\bar{A}..f$. The only remaining possibility is $\bar{A}..\$$ but this does not apply either because we assumed $x_1 x_2$ in Σ^*.

Induction step Two cases arise:

Case 1 let the rule for $\bar{A}$ in P' be $\bar{A}..B\bar{C}/\bar{D}$.

1a) let $B \underset{R'}{\Rightarrow} (x_3 \uparrow x_4 x_2\$, 0)$, $\bar{C} \underset{R'}{\Rightarrow} (x_4 \uparrow x_2\$, 0)$; but this contradicts the hypothesis, hence it is not possible.

1b) and 1c) are similar.

Case 2 let $\bar{A}..\bar{B}C/\bar{D}$ be in P'. The reasoning from above applies here too and we conclude R' has no partial-acceptance failures. QED

Remarks

1) By using techniques similar to the ones in Theorem 4.2 it can be shown that in a TS the end failures can also be eliminated. Therefore we can replace all p-a and end failures by subroutine failures. Moreover, any subroutine failure can be replaced by a loop failure; for

example, if there is no rule for variable A, we write for A the rule A..AA/A which will produce a loop failure instead. Hence, for any given TS R there exists a TS R' such that T(R) = T(R') and R' has only recognition or loop failures.

2) Another way of eliminating subroutine failure is to replace them by end failure. Assume there is no rule for A in P; we will include in P: $A..a_1A/a_2A/.../a_nA$ where $\{a_i \mid 1 \leq i \leq n\} = \Sigma_e$. It is easy to see that when A is called an end failure will result.

Using the result from above we can now prove that the TS languages are closed under intersection.

Theorem 4.3 Let $R_i = (V_i,\Sigma,P_i,S_i,\$)$, $i = 1,2$, be two TS. There is a TS R such that $T(R) = T(R_1) \cap T(R_2)$.

Proof We will assume R_1, R_2 have no partial-acceptance failures and also that $V_1 \cap V_2 = \phi$. Consider the TS $R = (V,\Sigma,P,S,\$)$ in which:

$V = V_1 \cup V_2 \cup \{S,X,J,Y,Z\}$ where S,X,J,Y,Z are new variables,

$P = P_1 \cup P_2 \cup \{S..XY/S_2, Y..f, X..S_1Z/J, Z..\varepsilon\}$. We will show: $S \underset{R}{\Rightarrow} (x\$\uparrow, 0)$ iff $S_i \underset{R_i}{\Rightarrow} (x\$\uparrow, 0)$, $i = 1,2$.

The "only if" part Assume $S \underset{R}{\Rightarrow} (x\$\uparrow, 0)$. The rule for S is $S..XY/S_2$; we notice that X cannot have a recognition failure (due to the rule $X..S_1Z/J$ and to the fact that J has no rule in P), hence the only way for S to succeed is for X and S_2 to succeed. But X succeeds only if S_1 succeeds and since S_1,S_2 cannot have partial-acceptance failures, it follows $S_1 \underset{R_1}{\Rightarrow} (x\$\uparrow, 0)$ and $S_2 \underset{R_2}{\Rightarrow} (x\$\uparrow, 0)$.

"if" Assume $S_i \underset{R_i}{\Rightarrow} (x\$\uparrow, 0)$, $i = 1,2$. We have $Z \underset{R}{\Rightarrow} (\uparrow, 0)$ and using the rule for X we get $X \underset{R}{\Rightarrow} (x\$\uparrow, 0)$. We have in R: $Y \underset{R}{\Rightarrow} (\uparrow, 1)$ and

$S_2 \underset{R}{\Rightarrow} (x\$\uparrow, 0)$; using the rule for S we finally get $S \underset{R}{\Rightarrow} (x\$\uparrow, 0)$. QED

Theorem 4.4 The complement of a wfTS language is a TS language.

Proof Let $R = (V,\Sigma,P,S,\$)$ be a wfTS. Consider the TS $R' = (V',\Sigma,P',S',\$)$ as follows: V' includes the set $V \cup \{S', J, A\}$, where S', A, J are new variables.

Let $\Sigma = \{a_i \mid 1 \leq i \leq k\}$; P' contains the rules in P and also the rules: $S'..SJ/A$, $A..a_1A/a_2A/...../a_kA/\$$.

First we show: for all x in Σ^*, if x is in T(R) then x is not in T(R'). Indeed, x in T(R) means $S \underset{R}{\Rightarrow} (x\$\uparrow, 0)$; then by the rule $S'..SJ/A$ we have in R' a subroutine failure (there is no rule for J in P'). By Lemma 2.1 the result is unique, hence x is not in T(R').

On the other hand, suppose x is not in T(R). We must have, under the assumption R is wfTS, $S \underset{R}{\Rightarrow} (\uparrow x\$, 1)$. Now $A \underset{R'}{\Rightarrow} (x\$\uparrow, 0)$ for all x in Σ^*, and the rule $S'..SJ/A$ will finally give $S' \underset{R'}{\Rightarrow} (x\$\uparrow, 0)$, hence x is in T(R'). We conclude that $T(R') = \overline{T(R)}$. QED

Next, we show that the wfTSL are closed under union.

Theorem 4.5 Let $R_i = (V_i,\Sigma,P_i,S_i,\$)$, $i = 1,2$, be two wfTS. There exists a wfTS R such that $T(R) = T(R_1) \cup T(R_2)$.

Proof We will assume $V_1 \cap V_2 = \phi$. Consider the wfTS $R = (V,\Sigma,P,S,\$)$ as follows: V includes the set $V_1 \cup V_2$; it also contains S, a new variable. P includes $P_1 \cup P_2$ and it also contains the rules $S..S_1/S_2$. We will show: for all x in Σ^*, x is in T(R) iff x is in $T(R_1) \cup T(R_2)$. First assume x is in T(R); this implies $S \underset{R}{\Rightarrow} (x\$\uparrow, 0)$. The rule for S in R:

$S..S_1/S_2$. The derivation $S \overset{*}{\underset{R}{\Rightarrow}} (x\$\uparrow, 0)$ is produced either by $S_1 \overset{*}{\underset{R}{\Rightarrow}} (x\$\uparrow, 0)$, which implies x is in $T(R_1)$, or by $S_1 \overset{*}{\underset{R}{\Rightarrow}} (\uparrow x\$, 1)$ and $S_2 \overset{*}{\underset{R}{\Rightarrow}} (x\$\uparrow, 0)$, which implies x is in $T(R_2)$. In either case x is in $T(R_1) \cup T(R_2)$. On the other hand, suppose x is in $T(R_1) \cup T(R_2)$. If x is in $T(R_1)$ then we also have $S_1 \overset{*}{\underset{R}{\Rightarrow}} (x\$\uparrow, 0)$ and $S \overset{*}{\underset{R}{\Rightarrow}} (x\$\uparrow, 0)$, i.e. x is in T(R); if x is in $T(R_2)$ and not in $T(R_1)$, we easily get x is in T(R). Finally R is well formed: suppose x is not in T(R). Then x is not in $T(R_1) \cup T(R_2)$ which implies $S_1 \overset{*}{\underset{R_1}{\Rightarrow}} (\uparrow x\$, 1)$ and $S_2 \overset{*}{\underset{R_2}{\Rightarrow}} (\uparrow x\$, 1)$. We also have $S_1 \overset{*}{\underset{R}{\Rightarrow}} (\uparrow x\$, 1)$, $S_2 \overset{*}{\underset{R}{\Rightarrow}} (\uparrow x\$, 1)$ and finally $S \overset{*}{\underset{R}{\Rightarrow}} (\uparrow x\$, 1)$. QED

In the next theorem we show the Σ^*-problem for wfTS is unsolvable (for definitions see [2]).

Theorem 4.6 It is unsolvable whether the language recognized by an arbitrary wfTS contains all the strings over its input alphabet.

Proof We know that the question (Q_1) whether the intersection of two languages accepted by two arbitrary DPDA's is empty, is unsolvable ([2]). Let L_1, L_2 be languages accepted by the arbitrary DPDA's A_1, A_2. We know ([2]) that we can construct two DPDA's, A_1' and A_2' such that the languages accepted by those are $\bar{L}_1$ and $\bar{L}_2$. By Theorem 3.1 we can effectively find two wfTS, R_1 and R_2 such that $T(R_1) = \bar{L}_1$ and $T(R_2) = \bar{L}_2$. Moreover, by Theorem 4.5, we can construct a wfTS R such that $T(R) = \bar{L}_1 \cup \bar{L}_2$. Suppose now that the question whether $T(R) = \Sigma^*$ was solvable. Then we could decide whether $\bar{L}_1 \cup \bar{L}_2 = \Sigma^*$ or, equivalently, whether $L_1 \cap L_2 = \emptyset$. This implies we could solve the given instance of Q_1, which is a contradiction. QED

At this point we can show that the emptiness problem for wfTS is unsolvable.

Theorem 4.7 It is unsolvable whether the language recognized by an arbitrary wfTS is empty.

Proof Consider the question Q_1 as in Theorem 4.6 and the instance of Q_1 with DPDA's A_1 and A_2 which accept the languages L_1 and L_2 respectively.

We will make the following assumption about A_1, A_2: let # be a new symbol; the input alphabet of A_1 and A_2 will contain #, and if this symbol appears in a string, the string is rejected. This assumption does not change the generality of the proof, because given an arbitrary DPDA, an equivalent DPDA with the above property can be constructed. Consider the wfTS R_1, R_2, R for which an effective construction was provided in Theorem 4.6; it was shown that if we could decide whether $T(R) = \Sigma^*$ we would have a solution for the given instance of Q_1.

Next we will describe the effective construction of a wfTS R_3 such that $T(R) = \Sigma^*$ iff $T(R_3) = \phi$. (The existence of a TS R_3 is guaranteed by Theorem 4.4. However, we are about to show that a well-formed R_3 can be found). It follows that if we could decide whether $T(R_3) = \phi$ we then would have a solution for the instance of Q_1.

Let $R = (V, \Sigma \cup \{\#\}, P, S, \$)$. Consider $R_3 = (V_3, \Sigma_1, P_3, S_3, \#)$ where: $\Sigma_1 = \Sigma \cup \{\$\}$; let $\Sigma = \{a_i \mid 1 \leq i \leq m\}$. V_3 includes $V \cup \{S_3, S_4, X, A\}$, S_3, S_4, X, A are new variables. P_3 includes P and also the set of rules:

$S_3 .. XS_4$, $X .. SX/\varepsilon$, $S_4 .. A\#$,

$A .. a_1A/a_2A/\ldots a_mA/\$$.

We first show: if $T(R) = \Sigma^*$ then $T(R_3) = \phi$. Assume $T(R) = \Sigma^*$. Let y in Σ_1^* be an input string for R_3. We write y: $y = x_1\$x_2\$\ldots x_{n-1}\$x_n$ for some integer n and some strings x_i in Σ^*, $1 \leq i \leq n$. We have $S \underset{R_3}{\Rightarrow} (x_1\$\lceil, 0)$ for $1 \leq i \leq n-1$, because $S \underset{R}{\Rightarrow} (x_i\$\lceil, 0)$ as we assumed $T(R) = \Sigma^*$. Since any string with # is rejected we have $S \underset{R_3}{\Rightarrow} (\lceil x_n\#, 1)$, which together with X..SX/ε implies $X \underset{R_3}{\Rightarrow} (x_1\$..x_{n-1}\$\, x_n\#, 0)$. And since $A \underset{R_3}{\Rightarrow} (\lceil x\#, 1)$ for all x in Σ^*, we have $A \underset{R_3}{\Rightarrow} (\lceil x_n\#, 1)$. Then $S_4 \underset{R_3}{\Rightarrow} (\lceil x_n\#, 1)$ and finally $S_3 \underset{R_3}{\Rightarrow} (\lceil y\#, 1)$, which implies that y is not in $T(R_3)$.

Next we show: if $T(R) \neq \Sigma^*$ then $T(R_3) \neq \phi$. Let x in Σ^* and $S \underset{R}{\Rightarrow} (\lceil x\$, 0)$. Then, in R_3, $X \underset{R_3}{\Rightarrow} (\lceil x\$\#, 0)$, $A \underset{R_3}{\Rightarrow} (x\$\lceil\#, 0)$, $S_4 \underset{R_3}{\Rightarrow} (x\$\#\lceil, 0)$ and finally $S_3 \underset{R_3}{\Rightarrow} (x\$\#\lceil, 0)$ which implies $T(R_3) \neq \phi$. We then conclude that $T(R) = \Sigma^*$ iff $T(R_3) = \phi$ and thus the proof is complete. QED

Corollary It is unsolvable whether the language recognized by an arbitrary TS is empty or accepts Σ^*.

The above result, expressed in terms of the TMG system from which the TS was modeled, implies that given a program in the TMG language, we cannot know if this program does anything at all. Since the emptiness problem is decidable for cfl's, the TS languages are, from this point of view, at a disadvantage for practical applications.

As we mentioned earlier, given a TS we can eliminate certain types of failures. For instance, Theorem 4.2 shows that p-a failures can be eliminated if we are willing to tolerate subroutine failures. It was also pointed out that given any TS R, a TS R' can be constructed

such that $T(R) = T(R')$ and any failure in R' is either a recognition or a loop failure. The question which arises is whether we can eliminate the loop failures in a given TS. Moreover, if we could eliminate all the loop failures without introducing other types of failures except recognition failures then we would reduce the given TS to a well-formed TS (In fact to a slightly restricted wfTS since the definition of a wfTS requires only S, the distinguished variable, to have only recognition failures). We suspect that loop failures cannot be eliminated in a TS and the following result seems to support this view.

Theorem 4.8 It is unsolvable whether an arbitrary TS is a wfTS.

Proof It was shown in Theorem 4.6 that it is unsolvable, for an arbitrary wfTS R_1, whether $T(R_1) = \Sigma^*$. Let $R_1 = (V_1,\Sigma,P_1,S_1,\$)$ be a wfTS. Consider the TS $R = (V,\Sigma,P,S,\$)$ where: V includes V_1 and also contains a new variable S. P includes P_1 and the rules: $S..S_1/\epsilon$. We will show that R is a wfTS if and only if $T(R_1) = \Sigma^*$. If we could decide whether R is a wfTS we could decide whether $T(R_1) = \Sigma^*$. First we show that if R is a wfTS then $T(R_1) = \Sigma^*$. The rule for S: $S..S_1/\epsilon$ implies that for no x in Σ^*, $S_1 \underset{R}{\Rightarrow} (\lceil x\$, 1)$, because we would have then $S \underset{R}{\Rightarrow} (\lceil x\$, 0)$ and this is not possible in a wfTS. Hence we must have $S_1 \underset{R}{\Rightarrow} (x\$\rceil, 0)$, for all x in Σ^*, or $T(R_1) = \Sigma^*$.

Now suppose $T(R_1) = \Sigma^*$; then $S \underset{R}{\Rightarrow} (x\$\rceil, 0)$ for all x in Σ^* and therefore R is a wfTS. QED

V. TIME COMPLEXITY OF TSL

We will describe a procedure which, given a TS and an input string of length n, will accept the string if and only if it is in the language recognized by the TS. Then we will prove the procedure halts after at most c.n steps from some constant c.

Algorithm 5.1 Let $R = (V,\Sigma,P,S,\$)$ be a TS and $w = a_1a_2\ldots a_{n-1}\$$, w in $\Sigma^*\$$, the input string.

The recognition matrix M of w is a r x n matrix, r being the number of elements in V, where each entry m(i,j) is in the set $\{(0,k) \mid 1 \leq k \leq n\} \cup \{\epsilon,1,2\}$. Initially for all i,j, $m(i,j) = \epsilon$. Let i,j be integers; i will designate the row, j will designate the column. Let $V = \{A_i \mid 1 \leq i \leq r\}$, $S = A_1$, and write the elements of P as $\{p_i \mid 1 \leq i \leq r\}$ where p_i is the rule for A_i if there is such a rule in p, otherwise $p_i = \phi$. The meaning of m(i,j) is the following:

- if $m(i,j) = (o,k)$, for some integer k, $j-1 \leq k \leq n$, then $A_i \underset{R}{\Rightarrow} (a_ja_{j+1}\ldots a_k\uparrow a_{k+1}\ldots\$, 0)$.
- if $m(i,j) = 1$ then $A_i \underset{R}{\Rightarrow} (\uparrow a_j\ldots\$, 1)$
- if $m(i,j) = 2$ then A_i has on $a_ja_{j+1}\ldots.\$$ a subroutine failure or an end failure or a loop failure. (See Definition 4.1).

The main part of the algorithm consists of filling in the entries of the recognition matrix M of w; this is described by the block diagram in Fig. 5.1:

1) The entries of M are filled in by columns from right to left, the last column is filled in first.

2) The entries of a given column j are filled in as follows: the column is scanned from top down (we call this a "pass") considering all entries $m(i,j) = \varepsilon$, until an entry is found which can be filled in. After this is done, another pass is initiated, starting from the first row. An entry m(i,j) can be filled in in several cases:

a) There is no rule for A_i, i.e. $p_i = \phi$; then we have a subroutine failure and m(i,j) is set to 2.

b) The rule for A_i has the form $A_i..a$, for some a in $\Sigma_e \cup \{\varepsilon, f\}$. if a is in Σ_e, then it is matched against a_j, the symbol corresponding to column j; if $a = a_j$ the m(i,j) is set to (o,j). In case $a \neq a_j$ or if a = f we have a recognition failure and m(i,j) is set to 1. If $a = \varepsilon$ then m(i,j) is set to (o, j-1) which indicates that the symbol a_j has yet to be checked against a symbol in Σ_e.

c) Let the rule for A_i be $A_i..A_k A_\ell / A_s$. In order to fill in m(i,j), we must first have $m(k,j) \neq \varepsilon$. Let $m(k,j) = (o,n_1)$, for some n_1, $j \leq n_1 < n$ (other cases are similar). Next consider $m(\ell, n_1+1)$; under the assumptions made $m(\ell,n_1+1) \neq \varepsilon$. If $m(\ell, n_1+1) = (o, n_2)$ for some n_2, $n_2 \leq n$, then m(i,j) is set to (o,n_2). If $m(\ell, n_1+1) = 2$ then m(i,j) is set to 2 and we have a subroutine failure. If $m(\ell, n_1+1) = 1$ then we consider m(s,j) - if m(i,j) is to be filled in we must have $m(s,j) \neq \varepsilon$ - and we set m(i,j) = m(s,j).

3) After r passes, the column to the left is considered. However, if less than r passes have been made and no other entry can be filled in, all entries (i,j) such that $m(i,j) = \varepsilon$ are set to 2 (we have in this case loop failures). When finally M is filled in, the value of m(1,1) is checked. If m(1,1) = (o,n) the string is accepted.

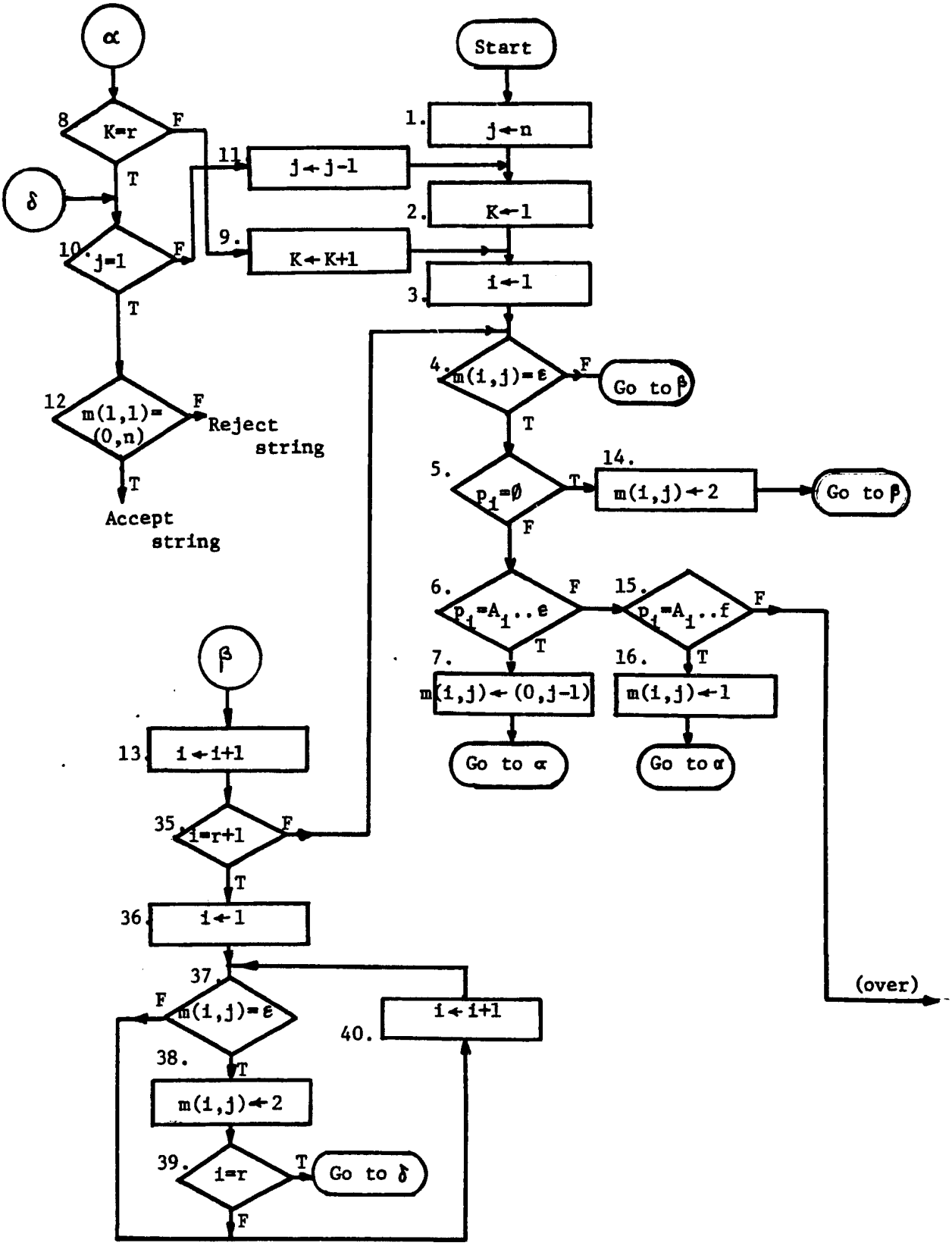
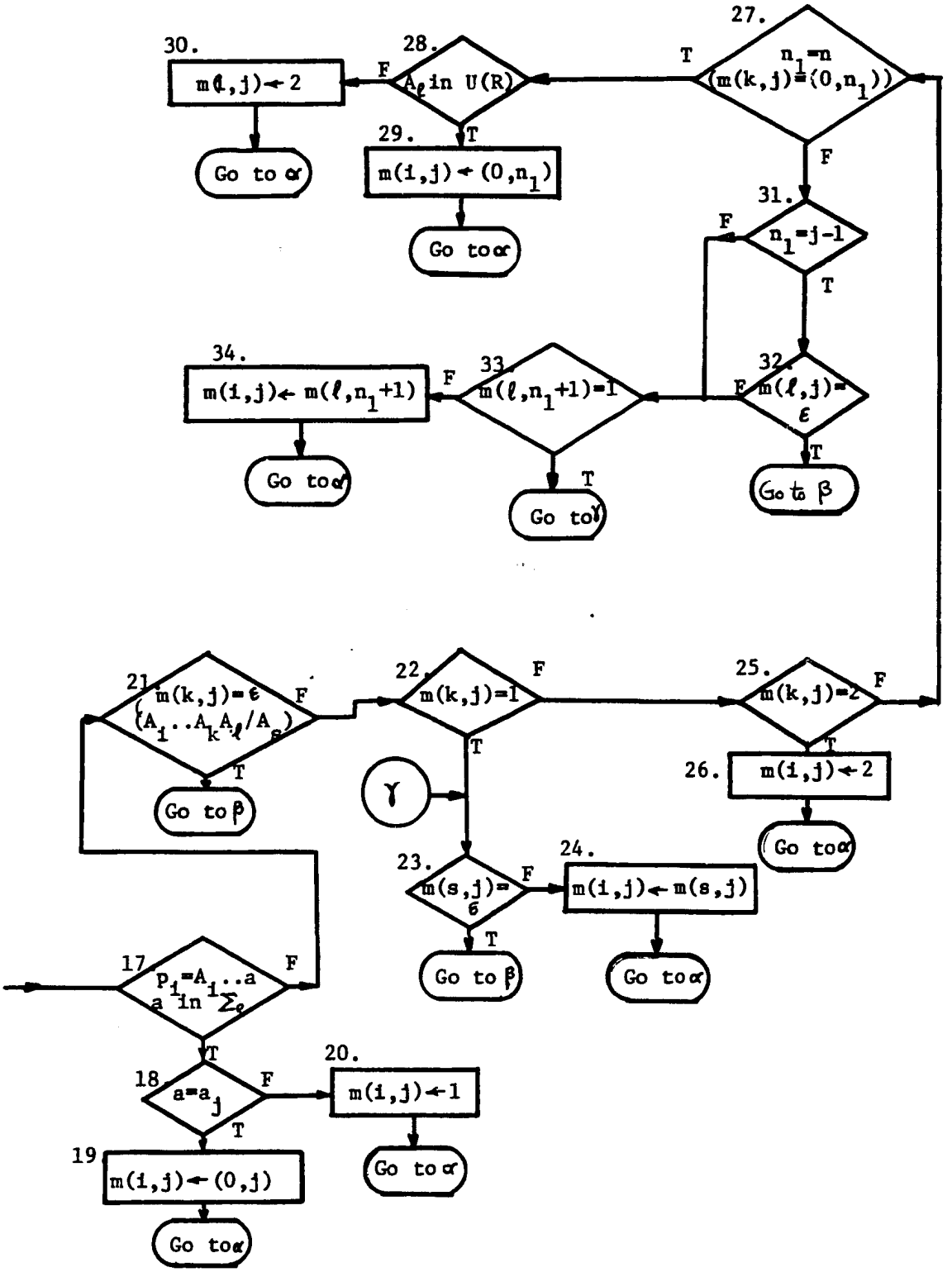


Fig.5.1 Block Diagram for Algoritm 5.1


27.
$n_1=n$
$(m(k,j)=(0,n_1))$
T
F
28.
A_ℓ in U(R)
F
T
30.
$m(\ell,j)\leftarrow 2$
Go to α
29.
$m(i,j)\leftarrow(0,n_1)$
Go to α
31.
$n_1=j-1$
F
T
32.
$m(\ell,j)=\epsilon$
F
T
Go to β
33
$m(\ell,n_1+1)=1$
F
T
Go to γ
34.
$m(i,j)\leftarrow m(\ell,n_1+1)$
Go to α
22.
$m(k,j)=1$
F
T
25.
$m(k,j)=2$
F
T
26.
$m(i,j)\leftarrow 2$
Go to α
γ
23.
$m(s,j)=6$
F
T
24.
$m(i,j)\leftarrow m(s,j)$
Go to α
Go to β
21
$m(k,j)=\epsilon$
$(A_1..A_kA_\ell/A_6)$
F
T
Go to β
17
$p_1=A_1..a$
a in Σ_6
F
T
18
$a=a_j$
F
T
20.
$m(i,j)\leftarrow 1$
Go to α
19
$m(i,j)\leftarrow(0,j)$
Go to α

Example 5.1 Consider $R = (V,\Sigma,P,S,\$)$ where:

$V = \{A_i \mid 1 \leq i \leq 6\}$, $S = A_1$.

$\Sigma = \{a\}$.

$P = A_1..A_2A_6,\ A_2..A_4A_5/A_3,\ A_3..\varepsilon,\ A_4..A_5A_2,\ A_5..a,\ A_6..\$$

Let the input string be $w = aaaaaa\$$. By applying the Algorithm 5.1 we get from M on w:

	1	2	3	4	5	6	7
1	(0,7)	1	1	1	(0,7)	1	(0,7)
2	(0,6)	(0,5)	(0,4)	(0,3)	(0,6)	(0,5)	(0,6)
3	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)
4	(0,5)	(0,4)	(0,3)	(0,6)	(0,5)	(0,6)	1
5	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	1
6	1	1	1	1	1	1	(0,7)

To illustrate how the algorithm works, suppose columns 4 through 7 have been filled in and column 3 is considered next. The first pass will skip over m(1,3), m(2,3); m(3,3) will be set to (0,2). On the next pass m(5,3) is set to (0,3). The third pass will consider m(4,3); as m(5,3) = (0,3) and the rule for A_4 is $A_4..A_5A_2$, we consider m(2,4) = (0,3) and we set m(4,3) to (0,3). When M is completed, we check that m(1,1) = (0,n); in this case (0,7), and the string is accepted. It will be shown later that $T(R) = \{a^{2(2^n-1)} \mid n = 0,1,2,...\}$.

In Fig. 5.1 various steps in the algorithm were designated by numbers from 1 to 45. It can be shown that each of these steps takes a

finite amount of time to perform on any computer. (For a more formal treatment of a "random access machine" see [11], [13]). The next theorem gives a bound for the number of steps, in the sense described above, required by Algorithm 5.1.

Theorem 5.1 For a given TS, Algorithm 5.1 recognizes a string of length n in less than c.n steps, for some constant c.

Proof Consider the three integers n-j, K,i where n is the length of the input string, j,k,i are the contents of the counters for, respectively, column number, number of passes and the row number. The algorithm can make at most r_1 steps, $r_1 < 45$ by Fig. 5.1, before i or K is increased; an upper bound for the number of steps, for a given j, is $r^2.r_1$. Let $c = r^2.r_1$; it follows that the matrix is completely processed after c.n steps. We will show that for all i,j, $1 \leq i \leq r$, $1 \leq j \leq n$:

(a) $m(i,j) = (o,k)$ iff $A_i \underset{R}{\Rightarrow} (a_j \ldots a_k \uparrow a_{k+1} \ldots a_n \$, o)$.

(b) $m(i,j) = 1$ iff $A_i \underset{R}{\Rightarrow} (\uparrow a_j \ldots a_n \$, 1)$.

The "only if" part We will prove (a), (b) simultaneously by induction on the number of passes on the columns of M.

Base m(i,j) is filled in on the first pass (which implies $j = n$).

(a) We have $m(i,n) = (o,k)$ for some k, two cases are possible. We could have $A_i..\$$ and then $k = n$ or we could have $A_i..\varepsilon$ and then $k = n-1$. In both cases the theorem is easily verified.

(b) We have $m(i,n) = 1$; then either $A_i..a$ and $a \neq \$$ for some a in Σ, or $A_i..f$. In both cases $A_i \underset{R}{\Rightarrow} (\uparrow\$, 1)$.

Induction step Let P contain $A_i..A_\ell A_m/A_p$.

(a) We have $m(i,j) = (o,k)$, for some k. Several cases arise:

Case 1 $m(\ell,j) = 1$, $m(p,j) = (o,k)$. By induction $A_\ell \underset{R}{\Rightarrow} (\uparrow a_j....a_n\$, 1)$, $A_p \underset{R}{\Rightarrow} (a_j....a_k\uparrow a_{k+1}...a_n\$, o)$. Using the rule for A_i we get $A_i \underset{R}{\Rightarrow} (a_j....a_k\uparrow a_{k+1}...a_n\$, o)$. Other cases are similarly treated.

(b) $m(i,j) = 1$.

Case 1 $m(\ell,j) = 1$, $m(p,j) = 1$. By induction $A_\ell \underset{R}{\Rightarrow} (\uparrow a_j...a_n\$, 1)$, $A_p \underset{R}{\Rightarrow} (\uparrow a_j....a_n\$, 1)$. Using the rule for A_i: $A_i \underset{R}{\Rightarrow} (\uparrow a_j....a_n\$, 1)$. The other case is treated similarly.

"if" We prove (a), (b) simultaneously, by induction on the number of derivation steps in R (see Definition 2.1).

Base For one step derivation the theorem is easily checked.

Induction step Let P contain $A_i..A_\ell A_m/A_p$. (a) We have $A_i \underset{R}{\Rightarrow} (w_1\uparrow w_2, o)$ where $w_1 = a_j....a_k$, $w_2 = a_{k+1}....a_{n-1}\$$. As before, several cases arise. Suppose $A_\ell \underset{R}{\Rightarrow} (\uparrow w_1w_2, 1)$, $A_p \underset{R}{\Rightarrow} (w_1\uparrow w_2. o)$.Then, by induction $m(\ell,j) = 1$, $m(p,j) = (o,k)$. The algorithm will give $m(i,j) = (o,k)$. Other cases are proved similarly.

(b) Assume $A_i \underset{R}{\Rightarrow} (\uparrow w, 1)$, where $w = a_j....a_{n-1}\$$. Also let $A_\ell \underset{R}{\Rightarrow} (\uparrow w_1, 1)$, $A_p \underset{R}{\Rightarrow} (\uparrow w_1, 1)$. Then by Algorithm 5.1 and using the inductive hypothesis we get $m(\ell,j) = 1$, $m(p,j) = 1$ and $m(i,j) = 1$. The proof is similar for the other case.

QED

VI. TS AND THE PHRASE STRUCTURE GRAMMARS

In this chapter we will show that the TSL are context sensitive and that they include some non - cfl's. Also, it will be shown that the TSL over a one letter alphabet are not regular.

In showing that the TSL are context sensitive we will use the concept of a deterministic linear bounded automaton. The linear bounded automaton (ℓba) was studied in [9], [10]. We will use the following definition of a deterministic ℓba:

Definition 6.1 A deterministic linear bounded automaton M is a 6-tuple $M = (K,\Sigma_M,\Gamma_M,\delta,q_o,F)$ in which:

K is a finite set of states

Σ_M is a finite set of input symbols; Σ_M contains two special symbols: the left marker ¢ and the right marker $.

Γ_M is a finite set of tape symbols, $\Sigma_M \subseteq \Gamma_M$.

q_o is an element of K

F is the set of final states, $F \subseteq K$.

δ is a mapping from $K \times \Gamma_M$ into $K \times \Gamma_M \times \{C,L,R\}$.

A configuration of M is denoted $(q, Z_1Z_2\ldots Z_n, i)$ where q is in K, $Z_1, Z_2, \ldots, Z_n$ in Γ_M, and i is an integer, $1 \leq i \leq n$. We define the relation $\vdash_M$ between two configurations of M as follows:

1) If $\delta(q,Z_i) = (p,Z,L)$ and $i > 1$ then we say that $(q,Z_1Z_2\ldots Z_n,i) \vdash_M (p,Z_1Z_2\ldots Z_{i-1}ZZ_{i+1}\ldots Z_n,i-1)$.

2) If $\delta(q,Z_i) = (p,Z,C)$ then we say that $(q,Z_1Z_2\ldots Z_n,i) \vdash_M (p,Z_1Z_2\ldots Z_{i-1}ZZ_{i+1}\ldots Z_n,i)$.

3) If $\delta(q,Z_i) = (p,Z,R)$ then we say that $(q,Z_1Z_2....Z_n,i) \vdash_M (p,Z_1Z_2....Z_{i-1}ZZ_{i+1}....Z_n, i+1)$. We define the relation $\vdash_M^*$ as the reflexive, transitive closure of $\vdash_M$. The language accepted by M is $\{w|w$ is in $(\Sigma_M-¢,\$)^*$, $(q_o,¢w\$, 1) \vdash_M^* (q,\alpha,i)$ for some q in F, α in Γ_M^*, and integer i$\}$.

Theorem 6.1 For any TS $R = (V,\Sigma,P,S,\$)$ there exists a deterministic ℓba M such that the language accepted by M is T(R).

Proof First we notice that Algorithm 5.1 does not illustrate the point of this theorem; to implement Algorithm 5.1 on a multitape Turing machine in the natural way would require nlogn tape. The reason is the following: an entry has to store elements in the set $\{(o,k)|\ 1 \leq k \leq n$ which encoded will take an amount of tape proportional to logn ; as there are n columns in the matrix, we need nlogn tape. Consider the deterministic ℓba $M = (K,\Sigma_M,\Gamma_M,\delta,q_o,F)$ in which:

$K = \{q_o,q_1,q_2\} \cup \{[p,\alpha]|p$ in $\{s,r\}$, α in $V \cup \{\uparrow,\varepsilon\}\}$.

$\Sigma_M = \Sigma \cup \{¢,\$\}$.

$\Gamma_M = \{\uparrow,\varepsilon\} \cdot \Sigma \cdot (\Gamma \cup \{\varepsilon\})^r$ where $r = |V|$ (by $|X|$ we denote the number of elements in the set X), and Γ is the tape alphabet of A(R). (In the course of the computation, input symbols are not erased, but they can be rewritten followed by a string over Γ^* of length r or less). Let the tape of M contain initially $¢a_1a_2....a_n\$$.

δ is given by the following rules:

1) For all a in Σ_e, $\delta(q_o,a) = ([s,\varepsilon],\uparrow aS,C)$. (The marker moves along the tape in the course of the computation. The rule

above writes the marker at the beginning of the computation, pointing to the first input symbol).

2) If A..BC/D is in P, then for all γ, $\delta([s,\varepsilon],\gamma A) = ([s,\varepsilon], \gamma(\bar{B}C/D)B,C)$. If A..$\varepsilon$ is in P, then for all γ, $\delta([s,\varepsilon], \gamma A) = ([s,\varepsilon], \gamma, C)$. Also, if A..f is in P then for all γ, $\delta([s,\varepsilon], \gamma A) = ([r,\varepsilon],\gamma, C)$. (In a given state in K, the symbols s,r show the last variable called has succeeded or failed, respectively.

3) If A..a is in P, a in Σ_e, then for all a_i in Σ_e, $a_i \neq a$ and all α, $\delta([s,\varepsilon],\uparrow a_i\alpha A) = ([r,\varepsilon],\uparrow a_i\alpha,C)$.

4) If A..a_i in P, a in Σ, then for all α, $\delta([s,\varepsilon],\uparrow a_i\alpha A) = ([s,\uparrow], a_i\alpha,R)$, and also $\delta([s,\uparrow],a_j) = ([s,\varepsilon],\uparrow a_j, L)$ for all a_j in Σ_e. If A..\$ in P, then $\delta([s,\varepsilon],\uparrow \$\alpha A) = ([s,\varepsilon], \$\alpha,C)$. (When there is a match, the marker is moved to the next cell. For \$ the marker is erased and any attempt to match another symbol will cause M to halt).

5) $\delta([s,\varepsilon],a_i) = ([s,\varepsilon],a_i,L)$ for all a_i in Σ_e. $\delta([s,\varepsilon], \gamma(\bar{A}B/C)) = ([s,\varepsilon], \gamma(A\bar{B}/C)B,C)$, $\delta([s,\varepsilon], \gamma(A\bar{B}/C)) = ([s,\varepsilon],\gamma,C)$ and $\delta([s,\varepsilon], \gamma(AB/\bar{C})) = ([s,\varepsilon],\gamma,C)$ for all γ and all A,B,C in V.

6) If A..a is in P, a in Σ_e, then $\delta([s,\varepsilon], a_i\alpha A) = ([s,A], a_i\alpha,R)$ for all a_i in Σ_e. $\delta([s,A], a_i) = ([s,A],a_i,R)$ and $\delta([s,A],\uparrow a_j) = ([s,\varepsilon],\uparrow a_jA,C)$ for all a_i,a_j in Σ_e, A in V. (In state [s,A] M moves to the right searching for the marker $\uparrow$ and when this is found, A is called).

7) For all γ and all B,C,D, in V: $\delta([r,\varepsilon], \gamma(\bar{B}C/D)) = ([s,\varepsilon], \gamma(BC/\bar{D}),C)$, $\delta([r,\varepsilon], \gamma(B\bar{C}/D)) = ([s,\varepsilon], \gamma(BC/\bar{D}),C)$ and $\delta([r,\varepsilon], \gamma(BC/\bar{D})) = ([r,\varepsilon],\gamma, C)$. (In state [r,$\varepsilon$], or [s,$\varepsilon$], M moves to the

left to report a failure, or success). We also have for all a_i in Σ_e and all α: $\delta([r,\varepsilon],\uparrow a_i) = ([r,\uparrow],a_i,L)$, $\delta([r,\uparrow],a_i) = ([r,\uparrow],a_i,L)$, $\delta([r,\uparrow], a_i\alpha) = ([r,\varepsilon],\uparrow a_i\alpha,C)$. (In state $[r,\uparrow]$ M backtracks after a recognition failure and the marker is moved to the place the last rule was called from).

8) $\delta([s,\varepsilon],¢) = (q_1,¢,R)$, $\delta(q_1,a_i) = (q_1,a_i,R)$ for all a_i in Σ and $\delta(q_1,\$) = (q_2,\$,C)$. (The symbol ¢ is reached when S succeeds; then M, in state q_1, verifies the marker $\uparrow$ has been erased which means the string has been accepted. q_2 is the final state).

It remains to show that x in Σ^* is accepted by M iff x in T(R). The following can be shown by induction: let the input string be $¢\, a_1a_2\ldots a_n\$$, let a configuration of M be described as (q,w,i) where q in K, $w = (\gamma_1)(\gamma_2)\ldots$, each γ_i in Γ_M, i an integer, $1 \leq i \leq n$, which shows the location of the read-head; then $([s,\varepsilon],\gamma(\uparrow a_i\alpha A)(a_{i+1})(a_{i+2})\ldots(a_j)\ldots(a_n)\$,\ i) \vdash_M^* ([s,\varepsilon],\ \gamma(a_i\alpha)(a_{i+1})\ldots(\uparrow a_j)(a_{j+1})\ldots\$,\ j)$ for some γ in Γ_M^*, α in Γ^*, A in V, i,j in {1,2,...n}, iff $A \underset{R}{\Rightarrow} (a_i\ldots a_{j-1}\uparrow a_j\ldots a_n\$,\ 0)$ and $([s,\varepsilon],\gamma(\uparrow a_i\alpha A)\ldots\$,\ i) \vdash_M^* ([r,\varepsilon],\ \gamma(\uparrow a_i\alpha)(a_{i+1})\ldots\$,\ i)$ iff $A \underset{R}{\Rightarrow} (\uparrow a_i\ldots\$,\ 1)$. The theorem follows immediately. QED

It was shown in Theorem 5.1 that any TS language can be recognized in linear time on a suitable "machine" using Algorithm 5.1. On the other hand, it is not known if there is a cfl which cannot be recognized in linear time by a suitable algorithm. In view of these facts we conjecture that there are cfl which are not TS.

It is easy to see that there are TS languages which are not context-free. For example consider the deterministic cfl $L_1 = \{a^n b^n a^m \mid n,m \geq 1\}$ and $L_2 = \{a^m b^n a^n \mid n,m \geq 1\}$. Their intersection $L = L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$ is not a cfl but according to Theorem 3.2 and Theorem 4.5 there is a TS R such that $L = T(R)$.

Next we will prove that the TSL over a one letter alphabet are not regular by producing such a non-regular language. (The cfl's over a one letter alphabet are regular, [12]). Consider Example 5.1: $R = (V,\Sigma,P,A_1,\$)$, where $V = \{A_i \mid 1 \leq i \leq 6\}$, $\Sigma = \{a\}$, $P = \{A_1..A_2A_6,\ A_2..A_4A_5/A_3,\ A_3..\varepsilon,\ A_4..A_5A_2,\ A_5..a,\ A_6..\$\}$.

Theorem 6.2 Let R be the TS described above. Then $T(R) = \{a^{2(2^n-1)} \mid n = 0,1,2,....\}$.

Proof Let the input string be $w = a^m$, let N be the largest integer such that $m = 2(2^N-1) + k$ for some positive k; then $0 < k \leq 2^{N+1}$ (It is easy to see there is always such an integer N). We will show that for any m, $A_2 \underset{R}{\Rightarrow} (a^{2(k-1)}\uparrow a^{m-2(k-1)}\$,\ 0)$. The proof follows by induction on m.

Base The cases m = 1 through m = 6 are verified in the example mentioned above (The recognition matrix M).

Induction step Assume the theorem is true for 1,2,...,m-1.

Case 1 $1 < k \leq 2^{N+1}$; then by induction $A_2 \underset{R}{\Rightarrow} (a^{2(k-2)}\uparrow a^{m-1-2(k2)}\$,\ 0)$. We put k-1 for k and m-1 for m). Consider the rule in P, $A_4..A_5A_2$. $A_5 \underset{R}{\Rightarrow} (a\uparrow a^{m-1}\$,\ 0)$ and with the derivation above we get $A_4 \underset{R}{\Rightarrow} (a^{2(k-2)+1}\uparrow a^{m-1-2(k-2)}\$,\ 0)$. Apply now the rule $A_2..A_4A_5/A_3$;

$A_5 \underset{R}{\Rightarrow} (a\uparrow a^{m-2-2(k-2)}\$,\ 0)$ can be written $A_5 \underset{R}{\Rightarrow} (a\uparrow a^{m-2(k-1)}\$,\ 0)$ and finally $A_2 \underset{R}{\Rightarrow} (a^{2(k-1)}\uparrow a^{m-2(k-1)}\$,\ 0)$.

Case 2 $k = 1$. We use now the inductive hypothesis applied to a string of length $m - 1$; we get, $m-1 = 2(2^N-1)$ and $A_2 \underset{R}{\Rightarrow} (a^{m-1}\uparrow\$,\ 0)$. Applying the rule $A_4..A_5A_2$ we get $A_4 \underset{R}{\Rightarrow} (a^m\uparrow\$,\ 0)$ and the rule $A_2..A_4A_5/A_3$ leads to $A_5 \underset{R}{\Rightarrow} (\uparrow\$,\ 1)$ and $A_3 \underset{R}{\Rightarrow} (\uparrow,\ 0)$ and finally $A_2 \underset{R}{\Rightarrow} (\uparrow a^m\$,\ 0)$, which proves the theorem.

Given a string $a^m\$$, A_1 will succeed iff $A_2 \underset{R}{\Rightarrow} (a^m\uparrow\$,\ 0)$ and using the result above, this happens iff $m = 2(k-1)$ which implies $m = 2(2^n-1)$ for some n. QED

VII. GENERALIZATIONS OF TS

In previous chapters we developed and studied the properties of the TS, a formal model for the recognition schema used in the TMG system ([1]). The following question arises: "what extensions or variations of this model are possible and how do they relate to the original model from a practical point of view?".

In this chapter we develop two extensions of the TS, namely the "generalized TS" (gTS) and the (ℓ,m)-TS. We show that these models are equivalent; they include the TS as a particular case and maintain some of its properties such as: (1) they can be recognized in linear time (by a modified version of Algorithm 5.1), (2) to every gTS (or (ℓ,m)-TS) there corresponds an automaton (similar to the one in Definition 2.2) which accepts the language recognized by the gTS.

However, we will show an important result which belongs to the generalized models and does not, we believe, belong to the TS, namely that for any gTS (or (ℓ,m)-TS) an equivalent gTS can be constructed which recognizes the same language and has only recognition failures. Finally, another extension of the TS, the eTS, is briefly discussed.

First we define the "generalized TS":

Definition 7.1 A generalized TS (gTS) R is a 5-tuple $R = (V,\Sigma,P,S,\$)$ in which:

V is a finite set of variables,

Σ is a finite set of terminal symbols,

S is an element of V,

\$ is the endmarker, \$ is not in Σ,

P is a finite set of rules of the form a) or b):

a) $A..B(C,D)$ A,B,C and D in V,

b) $A..a$ a in $\Sigma_e \cup \{\epsilon,f\}, \Sigma_e = \Sigma \cup \{\$\}$.

For any variable A there is at most one rule with A on the lefthand side.

Define the set of relations for each n in N, $A \overset{n}{\underset{R}{\Rightarrow}} (x\lceil y,i)$, x,y in Σ_e^*, i in {0,1}, as follows:

1) If $A..a$ is in P, a in Σ_e, then $A \overset{1}{\underset{R}{\Rightarrow}} (a\lceil x, o)$ for all x in Σ_e^*, and $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil bx,1)$ for all x in Σ_e^*, b in Σ_e, $b \neq a$.

2) If $A..\epsilon$ is in P, then $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil x,o)$ for all x in Σ_e^*.

3) If A..f is in P, then $A \overset{1}{\underset{R}{\Rightarrow}} (\hat{x}, 1)$ for all x in Σ_e^*.

4) Let A..B(C,D) be in P. For each x_1, x_2 and x_3 in Σ_e^*, if:

a) $B \overset{\ell}{\underset{R}{\Rightarrow}} (x_1 \hat{} x_2 x_3, o)$, $C \overset{m}{\underset{R}{\Rightarrow}} (x_2 \hat{} x_3, o)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (x_1 x_2 \hat{} x_3, o)$, $k = \ell+m+1$.

b) $B \overset{\ell}{\underset{R}{\Rightarrow}} (x_1 \hat{} x_2, o)$, $C \overset{m}{\underset{R}{\Rightarrow}} (\hat{} x_2, 1)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\hat{} x_1 x_2, 1)$, $k = \ell+m+1$.

c) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\hat{} x_1 x_2, 1)$, $D \overset{m}{\underset{R}{\Rightarrow}} (x_1 \hat{} x_2, o)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (x_1 \hat{} x_2, o)$, $k = \ell+m+1$.

d) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\hat{} x_1, 1)$, $D \overset{m}{\underset{R}{\Rightarrow}} (\hat{} x_1, 1)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\hat{} x_1, 1)$, $k = \ell+m+1$.

If $A \overset{n}{\underset{R}{\Rightarrow}} (x \hat{} y, i)$, x, y in Σ_e^*, i in {o,1}, we say that A <u>derives</u> $(x \hat{} y, i)$ <u>in n steps</u>. We say A <u>derives</u> $(x \hat{} y, i)$ if A derives $(x \hat{} y, i)$ in r steps for some r, and we write $A \underset{R}{\Rightarrow} (x \hat{} y, i)$. (Equivalently, we will say A has <u>outcome</u> i on x y and it <u>recognizes</u> x). The language recognized by R is: $T(R) = \{x | x \text{ in } \Sigma^*, S \underset{R}{\Rightarrow} (x\$\hat{}, o)\}$.

<u>Remarks</u>

1. The difference between the TS and the gTS can be explained informally as follows: suppose we have a rule A..B(C,D) in a given gTS and an input x in $\Sigma^*\$$; if A is called, A calls B and two cases arise:

<u>Case 1</u> B succeeds and recognizes some substring x_1 of the input string x. Let $x_1 y = x$; then C is called on y. If C succeeds and accepts y_1 for some y_1 and y_2, $y_1 y_2 = y$, then A succeeds and accepts $x_1 y_1$. If C fails then A fails (In the TS, in a similar Case D is called. In fact this is the only difference between the two schemas).

Case 2 B fails; then D is called on x and if D fails then A fails. If D succeeds then A succeeds and accepts whatever D accepts.

2. We generalize, in a recursive manner, the form of the rules in P:

a) The expression A..B,A and B in V, stands for the set of rules A..B(X_1,X_2), X_1..ε, X_2..f, where X_1,X_2 are new variables which do not appear in any other rule in P.

b) The expression A..B_1B_2 stands for the set of rules A..B_1(B_2,X_1) and X_1..f, where X_1 is a new variable which does not appear in any other rule in P.

c) The expression A..B_1B_2....B_k for some k > 2, stands for the set of rules A..X_1B_k, X_1..B_1B_2....B_{k-1} where X_1 is a new variable which does not appear in any other rule in P.

d) Consider the expression A..$\alpha_1\alpha_2...\alpha_k$ where α_i in $V \cup \Sigma_e \cup \{\varepsilon,f\}$ for $1 \leq i \leq k$, $k \geq 2$. If α_k is in V then the expression stands for the set of rules A..$X_1\alpha_k$, X_1..$\alpha_1\alpha_2...\alpha_{k-1}$, where X_1 is a new variable which does not appear in any other rule in P. If α_k is in $\Sigma_e \cup \{\varepsilon,f\}$ then the expression above stands for the set of rules A..$\alpha_1\alpha_2$.... $\alpha_{k-1}X_1$, X_1..α_k and X_1 is a new variable which does not appear in any other rule in P.

e) The expression A..α/B, A and B in V, α in $(V \cup \Sigma_e \cup \{\varepsilon,f\})^+$ stands for the set of rules A..X_1(X_2,B), X_1..α, X_2..ε where X_1,X_2 are new variables which do not appear in any other rule in P. (The expression A..α/B has the same meaning as in TS; a formal proof is given in Theorem 7.1).

f) The expression $A..\alpha_1/\alpha_2/...../\alpha_n$, for $n \geq 2$, α_i in $(V \cup \Sigma_e \cup \{\varepsilon,f\})^+$, $1 \leq i \leq n$, stands for the set of rules $A..\alpha_1/X_1$, $X_1..\alpha_2/\alpha_3/..../\alpha_n$ and X_1 is a new variable which does not appear in any other rule in P.

3. The expression $A..\alpha(\beta,\gamma)$ where α,β and γ in $(V \cup \Sigma_e \cup \{\varepsilon,f\})^+$, stands for the set of rules $A..X_1(X_2,X_3)$ $X_1..\alpha$, $X_2..\beta$ $X_3..\gamma$, where X_1,X_2,X_3 are new variables which do not appear in any other rule in P.

4. The expression $A..(\alpha)(\beta,\gamma)$, where α,β and γ are in $(V \cup \Sigma_e \cup \{\varepsilon,f\})^+$, stands for the set of rules $A..X_1(\gamma,\beta)$, $X_1..\alpha(f,\varepsilon)$, where X_1 is a new variable which does not appear in any other rule in P. Informally, the outcome of α will determine only which expression is processed next, β or γ, but the string recognized by α is irrelevant: if α succeeds we have backtracking and β is "called"; if α fails, γ is called and we have backtracking as usual.

5. As in the case of the TS, a "gTS-automaton" A(R) corresponding to the gTS R can be defined such that the language accepted by A(R) is T(R). Using the same notations for A(R), of a given gTS R, as in the case of the TS we will apply Definition 4.1 for various types of failures in the gTS.

In [15] Knuth describes the "parsing machine" (PM) which is similar to the gTS and the languages accepted by PM are the gTSL. The problem studied in [15] is the following: given a BNF grammar

the "corresponding PM program" is defined; then the question is asked whether the "corresponding PM program" accepts the language generated by the given grammar.

In the next theorem we show that the class of languages recognized by gTS, the gTSL, includes all the TSL.

Theorem 7.1 For any TS $R = (V,\Sigma,P,S,\$)$ there is a gTS R' such that $T(R') = T(R)$.

Proof Consider the gTS $R' = (V',\Sigma, P',S,\$)$ where V' includes the set $V \cup \{X_A \mid \text{all } A \text{ in } V\}$ and also all the variables which are implicitly defined in the shorthand notation of the rules in P'. P' is formed as follows:

1) if $A..a$ is in P, a in $\Sigma_e \cup \{\epsilon,f\}$, then $A..a$ is in P'.

2) if $A..BC/D$ is in P, A,B,C and D in V then P' contains $A..X_A(\epsilon,D)$ and $X_A..B(C,f)$.

It can be shown by induction that $T(R) = T(R')$. QED

Some of the results for the TS can be extended to gTS. One example is the result concerning the time complexity of TSL. The Algorithm 5.1 can be, with little change, applied to any given gTS and so we have the following theorem:

Theorem 7.2 For any gTS R, there is an algorithm which recognizes a string of length n in T(R) in less than c.n steps, for some constant c.

The Theorem 6.1 can also be extended to gTS:

Theorem 7.3 For any gTS R, there exists a deterministic linear bounded automaton M such that the language accepted by M is T(R).

Next we will display a gTS which accepts the language $L = \{a^{n^2} \mid n \geq 1\}$. We conjecture that there is no TS which recognizes L and subsequently the inclusion TSL $\subseteq$ gTSL is proper.

Let $Q = (V,\Sigma,P,S,\$)$ be a gTS where: V contains the set $\{S,A,A',B,C\}$ and also the variables which are implicitly included in the shorthand notation of the rules in P, $\Sigma = \{a\}$; P contains the rules: S..C(ε,A')

A..(S)(f,BS)

B..(S)(ε,aBa)

C..a$(ε,aaaa$)

A'..aaA

Theorem 7.4 Let Q,L as defined above. Then T(Q) = L.

Proof Let the input string be $a^n\$$, $n \geq 0$. For $n \leq 4$ the theorem can be checked with the following recognition matrix (the entries here have the same meaning as in Algorithm 5.1; see also Theorem 7.2):

	1	2	3	4	5	6
S	1	(0,6)	1	1	(0,6)	1
A	1	1	1	1	1	1
A'	1	1	1	1	1	1
B	(0,2)	(0,1)	1	(0,5)	(0,4)	1
C	1	(0,6)	1	1	(0,6)	1

For $n \geq 5$, let k be the largest integer such that $n = k^2 + m$, for some non negative integer m. We will show by induction on n that:

a) $B \underset{Q}{\Rightarrow} (a^{2m}\uparrow a^{n-2m}\$, 0)$

b) if $m = 0$ then $A' \underset{Q}{\Rightarrow} (a^n\$\uparrow, 0)$, otherwise $A' \underset{Q}{\Rightarrow} (\uparrow a^n\$, 1)$.

(These statements do not hold for $n < 5$ and for this reason we start the induction with $n = 5$).

Base The case $n = 5$ can be checked with the matrix above.

Induction step Case 1 Assume $m = 0$; let input string be $a^n\$$, $n=k^2$. On input $a^{n-2}\$$, we had $n-2=(k-1)^2 + 2k-3$; by the inductive hypothesis $B \underset{Q}{\Rightarrow} (a^{2(2k-3)}\uparrow a^{n-2-2(2k-3)}\$, 0)$. We can write $n-2-2(2k-3) = (k-2)^2$; again by induction on $a^{(k-2)^2}\$$ we get $S \underset{Q}{\Rightarrow} (a^{(k-2)^2}\$\uparrow, 0)$ and finally $A' \underset{Q}{\Rightarrow} (a^n\$\uparrow, 0)$, which proves statement b). Statement a) follows immediately using the rule for B and the fact that S succeeds on $a^n\$$; we get $B \underset{Q}{\Rightarrow} (\uparrow a^n\$, 0)$.

Case 2 We assume $m \neq 0$; let $n = k^2 + m$. We claim that A fails on $a^{n-2}\$$, i.e. $A \underset{Q}{\Rightarrow} (\uparrow a^{n-2}\$, 1)$, and this implies A' fails on $a^n\$$ and, finally, S fails on $a^n\$$. To show A fails on $a^{n-2}\$$, there are two cases to consider.

Case 2a: $m = 2$; then $n-2 = k^2$ and using the rule for A we get $A \underset{Q}{\Rightarrow} (\uparrow a^{k^2}\$, 1)$. Case 2b $m \neq 2$; then S fails on $a^{n-2}\$$ and for A to succeed, BS must succeed on $a^{n-2}\$$. We write $n-2 = k^2+m-2$ and by the inductive hypothesis $B \underset{Q}{\Rightarrow} (a^{2(m-2)}\uparrow a^{k^2-(m-2)}\$, 0)$; A will succeed on $a^{n-2}\$$ only if $k^2-(m-2) = p^2$ for some p. Assume such an integer p exists. Then $m = k^2-p^2+2 = (k+p)(k-p)+2 \geq (k+k-1)(k-k+1)+2 > 2k + 1$.

On the other hand, we assumed k takes the largest integer such that $n=k^2+m$, hence $m \leq 2k$, and the contradiction obtained shows BS cannot succeed on $a^{n-2}\$$, therefore A' (and also S) fails on $a^n\$$. By applying the inductive hypothesis on a) we get $B \underset{Q}{\Rightarrow} (a^{2m-2}\vdash a^{n-2m+1}\$,0)$. The rule for B: B..(S) (ε,aBa) applied on string $a^n\$$ gives $B \underset{Q}{\Rightarrow} (a^{2m}\vdash a^{n-2m}\$,0)$ which is the desired result. QED

Next we define the (ℓ,m)-TS. The (ℓ,m)-TS is a generalization of the TS and the gTS; whenever a variable is called over an input string, m outcomes are possible: ℓ of these are considered successful, the rest are failures and we have backtracking.

Definition 7.2 An (ℓ,m)-TS R, $\ell \geq 1$, $m \geq 2$, is a 6-tuple R = (V,Σ,P,S,g,$) in which:

V is a finite set of variables,

Σ is a finite set of terminal symbols,

S is an element of V,

$ is a symbol called endmarker, $ is not in Σ. (We use the notation Σ_e for the set $\Sigma \cup \{\$\}$).

P is a finite set of rules of the form a) or b):

a) $A..(Q_1,Q_2,\ldots,Q_m)$, A in V, for $1 \leq j \leq m$, $Q_j \subseteq \Sigma_e \cup \{\varepsilon\}$ such that for all a in $\Sigma_e \cup \{\varepsilon\}$ a belongs to exactly one Q_j.

b) $A..B(C_1,C_2,\ldots,C_m)$, A,B in V, C_j in V for $1 \leq j \leq m$.

For any A in V there is exactly one rule with A on the lefthand side.

g: $U_m \times U_m \to U_m$, where $U_m = \{1,2,..,m\}$. We define the set of relations, for each positive integer n, $A \overset{n}{\underset{R}{\Rightarrow}} (x\lceil y,i)$, x,y in Σ_e^*, i in U_m as follows:

1) If $A..(Q_1,Q_2,\ldots, Q_m)$ is in P and ϵ belongs to Q_j for some j, then $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil x,j)$, for all x in Σ_e^*. (In this case we can assume that all Q_i, $i \neq j$, are empty).

2) Let $A.. (Q_1,Q_2,\ldots., Q_m)$ be in P and for no j, Q_j contains ϵ; also let a be in Q_i. For all x in Σ_e^*, if $i \leqslant \ell$ then $A \overset{1}{\underset{R}{\Rightarrow}} (a\lceil x, i)$, and if $i > \ell$ then $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil ax, i)$.

3) Let $A..B(C_1,C_2,\ldots.,C_m)$ be in P. For all x_1,x_2,x_3 in Σ_e^*, if:

a) $B \overset{n_1}{\underset{R}{\Rightarrow}} (x_1\lceil x_2x_3, j)$, $C_j \overset{n_2}{\underset{R}{\Rightarrow}} (x_2\lceil x_3, k)$, $g(j,k) = i$ and $i \leqslant \ell$ then $A \overset{n}{\underset{R}{\Rightarrow}} (x_1x_2\lceil x_3, i)$ where $n = n_1 + n_2 + 1$.

b) $B \overset{n_1}{\underset{R}{\Rightarrow}} (x_1\lceil x_2x_3, j)$, $C_j \overset{n_2}{\underset{R}{\Rightarrow}} (x_2\lceil x_3,k)$, $g(j,k) = i$ and $i > \ell$, then $A \overset{n}{\underset{R}{\Rightarrow}} (\lceil x_1x_2x_3, i)$, where $n = n_1 + n_2 + 1$. (Informally, when A is "called" on $x_1x_2x_3$, A first calls B. Let us assume B "returns" with outcome j, recognizing the substring x_1; here, we could have $j > \ell$, in which case $x_1 = \epsilon$. Next C_j is called on x_2x_3 and it is assumed C_j returns with outcome k, recognizing the substring x_2. The function g will determine the outcome for A, specifically the outcome is $i = g(j,k)$.

In case a) the outcome is "success" since $i \leqslant \ell$ and therefore A will recognize the substring x_1x_2. In case b) $i > \ell$ and we have backtracking).

If $A \overset{n}{\underset{R}{\Rightarrow}} (x\lceil y, i)$, x,y in Σ_e^*, i in U_m, we say A derives $(x\lceil y,i)$ <u>in n steps</u>. We say that A derives $(x\lceil y, i)$ if A derives $(x\lceil y,i)$ in r steps for some r, and we write $A \underset{R}{\Rightarrow} (x\lceil y,i)$.

The language recognized by R is: $T(R) = \{x | x \text{ in } \Sigma^*, S \underset{R}{\Rightarrow} (x\$\uparrow, 1)\}$. As in previous cases (for the TS or the gTS) an automaton A(R) can be defined corresponding to an (ℓ,m)-TS R such that the language accepted by A(R) is T(R).

Definition 7.3 Let $R = (V,\Sigma,P,S,g,\$)$ be an (ℓ,m)-TS. The tape alphabet of A(R) is $\Gamma = \{A^{(i)} |$ all A in V, $0 \leq i \leq m\}$. (Whenever variable A is called in R, A(R) will print on its tape $A^{(o)}$, the superscript (o) indicating that no processing has been done yet. However if the rule for A is $A..B(C_1,\ldots,C_m)$ and B "returns" with outcome i, then before C_i is called, the symbol $A^{(o)}$ on the tape is changed to $A^{(i)}$, to indicate the result of the "call" of B). The set of internal states of A(R) is $K = \{q_i \mid 0 \leq i \leq m\}$. (There exists a state for every possible outcome when the head of the storage tape moves inwards, and also the state q_o which indicates a variable call when the head moves outward). A configuration of A(R) is a 3-tuple $(q,x_1\uparrow x_2,\omega)$ where q is in K, x_1x_2 is in Σ_e^*, $\uparrow$ is an abstract symbol (it indicates the position of the read head on string x_1x_2), ω is in $(\Gamma xN)^*$, N being the set of natural numbers.

We define the relation $\underset{A(R)}{\vdash}$ between two configurations α,β, and we write $\alpha \underset{A(R)}{\vdash} \beta$ as follows: assume $\alpha = (q,x_1\uparrow x_2,\omega)$, $\beta = (q',x_1'\uparrow x_2',\omega')$, $x_1x_2 = x_1'x_2'$, $\omega = \gamma(X,i)$ for some γ in $(\Gamma xN)^*$, X in Γ, i in N.

1) Let $q = q_o$, $X = A^{(o)}$ for some A in V, the rule for A in P is $A..B(C_1,\ldots,C_m)$; then $q' = q_o$, $x_1' = x_1$, $\omega' = \omega(B^{(o)}, i)$. (The rule indicates that with a rule of the form $A..B(C_1,\ldots C_m)$ the processing of A starts by having B called).

2) Let $q = q_o$, $X = A^{(o)}$, P contains $A..(Q_1,\ldots,Q_m)$; then if $x_2 = a\,x_3$, a in Q_j, $j \leq \ell$ we have $x_1' = x_1 a$, $q' = q_j$, $\omega' = \gamma$; if $x_2 = a\,x_3$, a in Q_j, $j > \ell$ we have $x_1' = x_1$, $q' = q_j$, $\omega' = \gamma$. (The processing of A in this case consists in matching the input symbol scanned with a set Q_j; if $j > \ell$ we have backtracking of one symbol, i.e the symbol has to be matched again).

3) Let $q = q_o$, $X = A^{(o)}$, P contains $A..(Q_1\ldots,Q_m)$ and ε belongs to Q_j; then $x_1' = x_1$, $q' = q_j$ and $\omega' = \gamma$.

4) Let $q = q_j$, $X = A^{(o)}$, P contains $A..B(C_1,\ldots,C_m)$; then $q' = q_o$, $x_1' = x_1$ and $\omega' = \gamma(A^{(j)}, i)(C_j^{(o)},|x_1|)$.

5) Let $q = q_k$, $X = A^{(j)}$, P contains $A..B(C_1,\ldots,C_m)$, $g(j,k) = h$; if $h \leq \ell$ then $q' = q_h$, $x_1' = x_1$, $\omega' = \gamma$; if $h > \ell$ then $q' = q_h$, $|x_1'| = i$, $\omega' = \gamma$.

If $\alpha \vdash_{A(R)} \beta$, we say A(R) makes a move from configuration α to configuration β. We write $\alpha \vdash^{*}_{A(R)} \beta$ if there are $\alpha_1,\ldots,\alpha_n$ such that $\alpha = \alpha_1$, $\beta = \alpha_n$, and $\alpha_i \vdash_{A(R)} \alpha_{i+1}$ for $1 \leq i \leq n-1$ and some n, the number of moves.

The language recognized by A(R) is $\{w|w \text{ in } \Sigma^*, (q_o,\lceil w\$, (S^{(o)},0)) \vdash^{*}_{A(R)} (q_1,w\$\rfloor,\varepsilon)\}$.

In a way similar to the proof of Theorem 2.1, it can be shown that the language recognized by A(R) is T(R). We notice that by the definition of the (ℓ,m)-TS, there exists a rule for every variable and therefore no subroutine failures are possible. Otherwise we can define various types of failures as for the TS (Definition 4.1).

Definition 7.4 Let $R = (V,\Sigma,P,S,g,\$)$ be an (ℓ,m)-TS. For A in V, x in Σ^*:

1) A has a recognition failure (or simply failure) on x if $A \underset{R}{\Longrightarrow} (\lceil x\$, j)$ for some $j > \ell$.

2) A has an end failure on x if $(q_o, \lceil x\$, (A^{(o)}, 0)) \overset{*}{\underset{A(R)}{\vdash}} (q_o, x\$\rfloor, \gamma(B^{(o)}, n))$ for some γ in $(\Gamma xN)^*$, n in N, B in V, and the rule for B in P is $B..(Q_1,\ldots,Q_m)$, no Q_i contains ε, $1 \leq i \leq m$.

3) R has a partial-acceptance failure (p-a failure) on x if $S \underset{R}{\Longrightarrow} (x_1 \lceil x_2\$, 1)$ for some x_1, x_2 in Σ^*, $x_1x_2 = x$.

4) A has a loop failure on x if A(R) in configuration $(q_o, \lceil x\$, (A^{(o)}, 0))$ can make an unbounded number of moves.

The following theorem shows that any gTS can be "simulated" by an (ℓ,m)-TS.

Theorem 7.5 For any gTS R there exist an (ℓ,m)-TS R' such that $T(R) = T(R')$.

Proof Let $R = (V,\Sigma,P,S,\$)$ be a gTS and consider the (1,3)-TS $R' = (V',\Sigma,P',S,g,\$)$ as follows: V' includes the set $V \cup \{X\}$ where X is not in V (V' contains also variables included in the shorthand notation of the rules in P'). The set of indices for R' is $U_3 = \{0,1,2\}$, g is defined by: $g(i,j) = j$ for all i,j in U_3. P' contains the rules:

1) $X..(\phi,\phi,\{\varepsilon\})$,

2) if $A..B(C,D)$ is in P, then $A..B(C,D,X)$,

3) if $A..\varepsilon$ is in P, then $A..(\{\varepsilon\}, \phi,\phi)$,

4) if A..f is in P, then A..$(\phi,\{\varepsilon\},\phi)$

5) if A..a, a in Σ_e, is in P, then A..$(\{a\},\Sigma_e-\{a\},\phi)$

6) if A has no rule in P, then A..$(\phi,\phi,\{\varepsilon\})$.

Since the (ℓ,m)-TS is defined so that no subroutine failure is possible (there exists a rule for every variable), we simulate this failure by introducing an additional outcome (2). Otherwise, it can be shown $A \overset{}{\underset{R}{\Longrightarrow}} (x\lceil y, i)$ iff $A \overset{}{\underset{R'}{\Longrightarrow}} (x\lceil y,i)$, for all A in V, x,y in Σ_e^*, i in $\{0,1\}$.

The theorem follows. QED

At this point, the following question can be asked: given an (ℓ,m)-TS, is it possible to find a gTS which accepts the same language? We will show that the answer is positive, but we first prove it for a restricted case.

Definition 7.5 An (ℓ,m)-TS, or a gTS, is reduced if it has only recognition failures.

In the definition which follows, a gTS is constructed corresponding to any reduced (ℓ,m)-TS. It will be shown that they recognize the same language.

Definition 7.6 Let $R = (V,\Sigma,P,S,g,\$)$ be an (ℓ,m)-TS. We define the gTS $M = (V_1,\Sigma,P_1,S^1,\$)$ as follows: V_1 includes the set $\{A^i | A \text{ in } V,\ 1 \leq i \leq m\} \cup \{[A^i,j] | 1 \leq i,j \leq m,\ A \text{ in } V.\}$ (For every variable A in V there are m variables in V_1, namely, A^i for $1 \leq i \leq m$. For any input string x, exactly one variable in this set will succeed and all the others will fail. Moreover, as it will be shown later, if

A has outcome j in R, on some input x, then A^j will succeed in M and it will recognize the same string that A recognizes in R, or if $j > \ell$, A^j will recognize the null string).

P_1 contains the rules:

1) let $A..(Q_1,Q_2,....,Q_m)$ be in P and for no j, Q_j contains ε. If $Q_i = \{a_1,...,a_k\}$, $k \geq 1$, $i \leq \ell$, then $A^i..a_1/a_2/..../a_k$. If $Q_i = \{a_1,...,a_k\}$, $k \geq 1$, $i > \ell$, then $A^i..(a_1/...../a_k)(\varepsilon,f)$. If $Q_i = \phi$ then $A^i..f$.

2) let $A..(Q_1,...,Q_m)$ be in P and suppose there exists j such that Q_j contains ε; then $A^i..f$, for $i \neq j$, and $A^j..\varepsilon$.

3) let $A..B(C_1,....,C_m)$ be in P, let $G_j = \{(p_i , q_i) | 1 \leq i \leq k\}$ be the set of pairs such that $g(p_i,q_i) = j$, for some $j \leq \ell$; then $A^j..B^{p_1}C^{q_1}_{p_1}/B^{p_2}C^{q_2}_{p_2}/...../B^{p_k}C^{q_k}_{p_k}$; if G_j is empty, $A^j..f$. (Since A^j succeeds in M only if A has outcome j in R we need to consider in the rule for A^j only the pairs in G_j).

4) let $A..B(C_1,....,C_m)$ be in P, let $G_j = \{(p_i,q_i) | 1 \leq i \leq k\}$ be the set of pairs such that $g(p_i,q_i) = j$ for some $j > \ell$; then $A^j..(B^{p_1}C^{q_1}_{p_1})(\varepsilon, [A^j,2]);[A^j, i]..(B^{p_i}C^{q_i}_{p_i})\ (\varepsilon,[A^j,i+1])$, $i = 2,3,...,k-1$; $[A^j,k]..(B^{p_k}C^{q_k}_{p_k})(\varepsilon,f)$.

If G_j is empty, then $A^j..f$. (The case is different from the one above in as far as we have backtracking. This is the reason we use rules of the form $A..(\alpha)(\beta,\gamma)$, since α is only checked for outcome and backtracking will always occur on the substring recognized by α).

In order to show T(R) = T(M) we need the following lemmas.

Lemma 7.1 Consider R,M as above. For any y,z in Σ_e^*, if there exists j, $1 \leq j \leq m$, such that $A^j \underset{M}{\overset{n}{\Longrightarrow}} (y\uparrow z, o)$, for some n, then $A^i \underset{M}{\Longrightarrow} (\uparrow yz,1)$ for all $i \neq j$.

Proof By induction on n.

n = 1 The rule for A in P has the form $A..(Q_1,\ldots,Q_m)$.

Case 1 Q_j contains ε. Then, according to the definition of M we have $A^i..f$, for all $i \neq j$; the theorem follows.

Case 2 Q_j does not contain ε. We can write y = a for some a in Σ_e; also a is in Q_j. Using the definition of M it follows that A^i fails on az, for all $i \neq j$.

Induction step Let $A..B(C_1,C_2,\ldots,C_m)$ be in P.

Case 1 $j \leq \ell$; the rule for A^j contains all the pairs $B^{p_i} C_{p_i}^{q_i}$ such that $g(p_i,q_i) = j$. Let (p,q) be such that $B^p \underset{M}{\Longrightarrow} (y_1\uparrow y_2z,o)$, $C_p^q \underset{M}{\Longrightarrow} (y_2\uparrow z,o)$, $y_1y_2 = y$. By induction p and q are unique, that is for no pair (p',q'), $B^{p'} C_{p'}^{q'}$ succeeds on yz and therefore no A^i, for $i \neq j$, can succeed.

Case 2 $j > \ell$; this case is proved in a similar way. QED

Lemma 7.2 Consider R,M as before. For all A in V, xy in Σ_e^*, $A \underset{R}{\Longrightarrow} (x\uparrow y,i)$ iff $A^i \underset{M}{\Longrightarrow} (x\uparrow y, o)$.

Proof First we show: if $A \underset{R}{\overset{n}{\Longrightarrow}} (x\uparrow y,i)$ then $A^i \underset{M}{\Longrightarrow} (x\uparrow y,o)$. The proof is by induction on n.

n = 1 Let P contain $A..(Q_1,\ldots,Q_m)$.

Case 1 Q_i contains ε. Then $x = \varepsilon$, $A^i \ldots \varepsilon$ and $A^i \underset{M}{\Rightarrow} (\uparrow y, o)$.

Case 2 Q_i does not contain ε. Then $x = a$, for some a in Σ_e and a is in Q_i. Finally, in M, $A^i \underset{M}{\Rightarrow} (a \uparrow y, o)$.

Induction step Let the rule for A be $A \ldots B(C_1, \ldots, C_m)$. We will consider the case $i \leq \ell$. (The other case is similar). There are j,k in U_m, such that $B \underset{R}{\Rightarrow} (x_1 \uparrow x_2 y, j)$, $C_j \underset{R}{\Rightarrow} (x_2 \uparrow y, k)$, $x_1 x_2 = x$ and $g(j,k) = i$. Let $G_i = \{(p,q) \mid p,q \text{ in } U_m, g(p,q) = i\}$. By induction $B^j \underset{M}{\Rightarrow} (x_1 \uparrow x_2 y, o)$, $C_j^k \underset{M}{\Rightarrow} (x_2 \uparrow y, o)$. Using now the rule for A^i in M, for all pairs (p,q) others than (j,k), $B^p C_p^q$ will fail on xy according to Lemma 7.1; therefore $A^i \underset{M}{\Rightarrow} (x_1 x_2 \uparrow y, o)$.

Now we show: if $A^i \underset{M}{\overset{n}{\Rightarrow}} (x \uparrow y, o)$ then $A \underset{R}{\Rightarrow} (x \uparrow y, i)$. The proof is by induction on n.

n = 1 This case is easily verified.

Induction step Let the rule for A be $A \ldots B(C_1, \ldots, C_m)$. We will consider the case $i > \ell$ (the other case is similar). Then $x = \varepsilon$; also let (p,q) be such that $B^p \underset{M}{\Rightarrow} (y_1 \uparrow y_2 z, o)$, $C_p^q \underset{M}{\Rightarrow} (y_2 \uparrow z, o)$, $y_1 y_2 z = y$. By induction $B \underset{R}{\Rightarrow} (y_1 \uparrow y_2 z, p)$, $C_p \underset{R}{\Rightarrow} (y_2 \uparrow z, q)$ and $g(p,q) = i$; it follows that $A \underset{R}{\Rightarrow} (\uparrow y_1 y_2 z, i)$. QED

Theorem 7.6 For any reduced (ℓ,m)-TS R, there exists a reduced gTS M such that $T(R) = T(M)$.

Proof Using Lemma 7.2 above: $S \underset{R}{\Rightarrow} (x\$\uparrow, 1)$ if and only if $S^1 \underset{M}{\Rightarrow} (x\$\uparrow, o)$; it follows that $T(R) = T(M)$ and also that M is reduced. QED

We would like to show that given any (ℓ,m)-TS R there exists a reduced (ℓ',m')-TS R' such that $T(R) = T(R')$. For this purpose we will show: a) loop failures can be eliminated; b) p-a and end failures can be eliminated (no loop failures being introduced).

Next we show that loop failures can be eliminated in any given (ℓ,m)-TS. We will proceed in two steps: given an (ℓ,m)-TS R we first construct the (ℓ',m')-TS R_1; from R_1 we will construct later on the desired (ℓ',m')-TS R', which has no loop failures and recognizes T(R).

Definition 7.7 Let $R = (V,\Sigma,P,S,g,\$)$ be an (ℓ,m)-TS. We define the (ℓ',m')-TS R_1, $R_1 = (V_1,\Sigma,P_1,S,g_1,\$)$,as follows: $\ell' = 2\ell$, $m' = \ell+m+1$. (We introduce additional outcomes with the following meaning: if the outcome of a derivation in R is i, $i \leq \ell$, and the string recognized is not the null string, then the same derivation holds in R_1. However if $i \leq \ell$ and the string recognized is the null string or if $i > \ell$ in R, then the outcome in R_1 is $\ell + i$, the string recognized being the null string. The purpose is to separate the derivations in which the null string is recognized and which, as it wil be shown later on, could give rise to a loop. The outcome $\ell+m+1$ is reserved for cases in which a loop occurs in R and though it will be used only later in R', it is introduced here for ease of notation). $V_1 = V \cup \{X\}$, X is a new variable not in V. P_1 contains the following rules:

1) if $A..(Q_1,\ldots,Q_m)$ is in P and for no i Q_i contains ε, then P_1 contains $A..(Q'_1,\ldots,Q'_{m'})$ where $Q'_i = Q_i$ for $i \leq \ell$, $Q'_i = \phi$ for $\ell < i \leq 2\ell$, or $i = \ell+m+1$, $Q'_{i+\ell} = Q_i$ for $\ell < i \leq m$.

2) if $A..(Q_1,\ldots,Q_m)$ is in P and Q_j contains ε for some j, then P_1 contains $A..(Q'_1,\ldots,Q'_{m'})$ where $Q'_{\ell+j} = \{\varepsilon\}$ and $Q'_i = \phi$ for all $i \neq \ell+j$.

3) if $A..B(C_1,\ldots,C_m)$ is in P, then P_1 contains $A..B(X_1\ldots,X_{m'})$ where $X_i = C_i = X_{\ell+i}$ for $1 \leq i \leq \ell$, $X_{\ell+i} = C_i$ for $\ell < i \leq m$, $X_{\ell+m+1} = X$.

Let, for $1 \leq i \leq m$, $\bar{i} = i$ if $i \leq \ell$ and $\bar{i}$ undefined otherwise. Then, g_1 is defined as follows:

1) If $g(i,j) = k$, $k \leq \ell$ then $g_1(\bar{i},\ell+j) = g_1(\ell+i,\bar{i}) = g_1(\bar{i},\bar{i}) = k$ and $g_1(\ell+i, \ell+j) = \ell + k$.

2) If $g(i,j) = k$, $k > \ell$, then $g_1(\bar{i},\bar{j}) = g_1(\bar{i},\ell+j) = g_1(\ell+i,\ell+j) = g_1(\ell+i,\bar{j}) = \ell + k$.

3) $g_1(i, \ell+m+1) = g_1(\ell+m+1, i) = \ell+m+1$ for $1 \leq i \leq \ell+m+1$.

Lemma 7.3 Consider R, R_1 as above.

a) Let $A \overset{n}{\underset{R}{\Longrightarrow}} (x\lceil y,i)$, for some integer n. If $i \leq \ell$ and $x \neq \varepsilon$ then $A \underset{R_1}{\Longrightarrow} (x\lceil y,i)$, otherwise $A \underset{R_1}{\Longrightarrow} (x\lceil y,\ell+i)$.

b) Let $A \overset{n}{\underset{R_1}{\Longrightarrow}} (x\lceil y,i)$. If $i \leq \ell$ then $A \underset{R}{\Longrightarrow} (x\lceil y,i)$. If $i > \ell$, then $x = \varepsilon$ and $A \underset{R}{\Longrightarrow} (\lceil y,i-\ell)$.

Proof a) By induction on n.

n = 1 This case is easily verified.

Induction step Let $A..B(C_1,....,C_m)$ be in P. Also, let $B \underset{R}{\Rightarrow} (x_1\uparrow x_2x_3, p)$, $C_p \underset{R}{\Rightarrow} (x_2\uparrow x_3, q)$, $x_1x_2x_3 = xy$ and $g(p,q) = i$.

Case 1 $p,q,i \leq \ell$; then $x_1x_2 = x$, $x_3 = y$. If $i \leq \ell$ and $x = \varepsilon$ then $x_1 = x_2 = \varepsilon$ and by induction $B \underset{R_1}{\Rightarrow} (\uparrow x_3, p+\ell), C_{p+\ell} \underset{R_1}{\Rightarrow} (\uparrow x_3, q+\ell)$; using the definition of R_1 we get $A \underset{R_1}{\Rightarrow} (\uparrow x_3, g_1(p+\ell, q+\ell))$ or $A \underset{R_1}{\Rightarrow} (x\uparrow y, \ell+i)$.

If $x \neq \varepsilon$ then x_1 or x_2 or both are not the null string, in which case, by induction, $B \underset{R_1}{\Rightarrow} (x_1\uparrow x_2\ x_3, p')$ and $C_p \underset{R_1}{\Rightarrow} (x_2\uparrow x_3, q')$ where $p' = p$ or $q' = q$ or both. Finally by the definition of R_1 $A \underset{R_1}{\Rightarrow} (x\uparrow y, i)$.

Other cases are similarly treated.

b) By induction on n. This proof is similar to a). QED

Corollary $T(R) = T(R_1)$.

In the following definition we construct the (ℓ',m')-TS R' which, as we will show later, has no loop failures and $T(R')=T(R)$.

Definition 7.8 Consider R, R_1 as given in Definition 7.7. We define the (ℓ',m')-TS R' as $R' = (V',\Sigma,P',[S,\phi],g_1,\$)$ where: $V' = \{[A,W] |$ all A in V_1, $W \subseteq V_1\}$, P contains the following rules:

(1) if $A..(Q_1,...,Q_{m'})$ is in P_1, then $[A,W]..(Q_1,...,Q_{m'})$ for all $W \subseteq V_1 - \{A\}$.

(2) if $A..B(C_1,....,C_{m'})$ is in P_1, then for all $W \subseteq V_1 - \{A\}$, $[A,W]..[B,W']$ $([C_1,W_1],....,[C_{m'},W_{m'}])$, where $W_i = \phi$ for $i \leq \ell$, $W_i = W' = W \cup \{A\}$ for $i > \ell$.

(3) $[X,W]..(\phi,\ldots,\phi,\{\varepsilon\})$ for all $W \subseteq V$.

(4) for all A in V, $W \subseteq V$, $[A,W\cup\{A\}]..(\phi,\ldots,\phi,\{\varepsilon\})$.

In R' defined above, every variable has the form [A,W] where A is a variable of R_1 and W is a subset of V_1. The rules in R' are simulating those in R_1 and in addition the set corresponding to each variable (W for the variable [A,W]), keeps track of those variables which could give rise to a loop in R_1. For instance, if a variable [A,W] is called in R', this corresponds to variable A being called in R_1. Further, if variable A in turn, calls in R_1 any variable belonging to the set W, then a loop occurs in R_1. In R' the loop is prevented by rule (4) above which guarantees the derivation will end with outcome m+ℓ+1.

We will show now the relation between R_1,R'. But first, we have the following definition:

Definition 7.9 Let $R = (V,\Sigma,P,S,g,\$)$ be an (ℓ,m)-TS. We define the set L(A,x) for all A in V, x in Σ_e^* as: $L(A,x) = \{Z \mid Z \text{ in } V \text{ and } \exists \gamma$ in $(\Gamma x N)^*$ such that $(q_o,\vdash x,(A^{(o)},0)) \vdash_{A(R)}^{*} (q_o,\vdash x,\gamma(Z^{(o)},0))\}\cup\{A\}$. Informally, the set L(A,x) will contain a variable B if the automaton A(R) starting with x on the input tape and with $A^{(o)}$ on its storage tape will eventually reach a configuration in which B is called and the read-head is in the same position as in the beginning, i.e. it points to the first symbol of x. For instance, if the rule for A is $A..B(C_1,\ldots,C_m)$, then B is in L(A,x), for all x in Σ_e^*.

Lemma 7.4 Consider R_1, R' as given by Definition 7.8. For all xy in Σ_e^*, and all integers i, $1 \leq i < m'$:

a) if $A \overset{n}{\underset{R_1}{\Longrightarrow}} (x\uparrow y, i)$, for some integer n, then $[A,W] \underset{R'}{\Longrightarrow} (x\uparrow y, i)$ for all sets W, $W \subseteq V - L(A,xy)$.

b) if $[A,W] \overset{n}{\underset{R'}{\Longrightarrow}} (x\uparrow y, i)$ for some set $W \subseteq V - L(A,xy)$ and some integer n, then $A \underset{R_1}{\Longrightarrow} (x\uparrow y, i)$.

Proof a) By induction on n.

n = 1 This case is easily verified using Definition 7.8.

Induction step Let the rule for A in P_1 be $A..B(C_1,\ldots,C_m')$; also let $B \underset{R_1}{\Longrightarrow} (x_1\uparrow x_2x_3, p)$, $C_p \underset{R_1}{\Longrightarrow} (x_2\uparrow x_3, q)$, $x_1x_2x_3 = xy$, $g_1(p,q) = i$. Let $W_1 = V\text{-}L(B,xy)$, $W_2 = V\text{-}L(C_p,xy)$, $W_3 = V\text{-}L(A,xy)$; then by induction $[B,W_1] \underset{R'}{\Longrightarrow} (x_1\uparrow x_2x_3,p)$, $[C_p,W_2] \underset{R'}{\Longrightarrow} (x_2\uparrow x_3,q)$. There are two cases possible:

Case 1 $x_1 \neq \varepsilon$ (which implies $p \leq \ell$); we claim that $L(B,xy) = L(A,xy) - \{A\}$. First, if x is in L(B,xy) then X is in L(A,xy), as we have in $A(R_1)$: $(q_o,\uparrow xy,(A^{(o)},0)) \underset{A(R_1)}{\vdash} (q_o,\uparrow xy,(A^{(o)},0)(B^{(o)},0))$. Suppose now that X is in $L(A,xy)-\{A\}$. As $x_1 \neq \varepsilon$, we must have $(q_o,\uparrow xy,(B^{(o)},0) \overset{*}{\underset{A(R_1)}{\vdash}} (q_1,\uparrow xy,\gamma(x^{(o)},0))$ for some γ (otherwise $B^{(o)}$ will be erased and the read-head will move past the first symbol in x_1); therefore X is in L(B,xy). It follows, then, that $W_3 = W_1 - \{A\}$. Further, we observe that for all x_1x_2 in Σ_e^*, $1 \leq i < m'$, Z in V, $W \subseteq V$, if $[Z,W] \underset{R'}{\Longrightarrow} (x_1\uparrow x_2,i)$ then $[Z,W'] \underset{R'}{\Longrightarrow} (x_1\uparrow x_2,i)$ for all $W' \subseteq W$ (since $i < m'$ we do not have a loop in R and by Definition 7.8 the rule for [Z,W] in R' can be written with any set W', $W' \subseteq W$). Thus, we have $[B,W_1] \underset{R'}{\Longrightarrow} (x_1\uparrow x_2x_3,p)$,

$[C_p,\phi] \overset{}{\underset{R'}{\Longrightarrow}} (x_2\uparrow x_3,q)$; using the rule for $[A,W_3]$ we get $[A,W_3] \overset{}{\underset{R'}{\Longrightarrow}} (x\uparrow y,i)$, where $W_3 = V-L(A,xy)$.

Case 2 $x_1 = \epsilon$ (which implies $p > \ell$); using an argument similar to the one before we can show that $L(A,xy)-\{A\} = L(B,xy)\cup L(C_p,xy)$. It follows that $W_3\cup\{A\} = W_1 - L(C_p,xy) = W_2 - L(B,xy)$; also $[B,W_3\cup\{A\}] \overset{}{\underset{R'}{\Longrightarrow}} (x_1\uparrow x_2x_3, p)$, $[C_p,W_3\cup\{A\}] \overset{}{\underset{R'}{\Longrightarrow}} (x_2\uparrow x_3, q)$ and finally $[A,W_3] \overset{}{\underset{R'}{\Longrightarrow}} (x\uparrow y,i)$.

b) The proof is similar to a).

QED

Corollary $T(R_1) = T(R')$.

It remains to be shown that R' has no loop failures.

Lemma 7.5 Let R' be given by Definition 7.8. R' has no loop-failures.

Proof Assume the contrary; then we must have a variable [A,L] in V' and a string x in $\Sigma^*\$$ such that $(q_o,\uparrow x, ([A,L]^{(o)},0)) \overset{*}{\underset{A(R')}{\vdash}} (q_o,\uparrow x,\gamma ([A,L]^{(o)},0))$, for some γ in $(\Gamma x N)^*$. (The notation $[A,L]^{(o)}$ is derived from [A,L] as a variable in V' and the superscript (o) in Γ , as in Definition 7.3). Let the rule for A in P_1 be $A..B(C_1,\ldots,C_{m'})$; then P' contains $[A,L]..[B,L']([C_1,L_1],\ldots,[C_{m'},L_{m'}])$ where $L_i = \phi$ for $i \leq \ell$ and $L_i = L' = L\cup\{A\}$ for $i > \ell$. Two cases are possible:

Case 1 $[B,L']^{(o)}$ is written on the tape and though its superscript (in this case (o)) might change the symbol is never replaced by ϵ ; then, the second symbol of the string γ in $(\Gamma x N)^*$ is $([B,L']^{(i)},1)$ for some i, $0 \leq i \leq m'$, and since $L' = L\cup\{A\}$ we have $L'\supset L$.

Case 2 $[B,L'] \overset{}{\underset{R'}{\Longrightarrow}} (\uparrow x,j)$ for some j, $\ell < j \leq m'$; in this case the second symbol of γ is $([C_j,L']^{(i)}, 1)$ for some i, $1 \leq i \leq m'$, and $L' = L\cup\{A\}$.

We notice that in both cases the set of variables associated with the second symbol of γ(namely L') properly includes the set corresponding to the first symbol (L). Since the length of γ is finite, we can repeat the same argument for every symbol in γ and therefore the set of variables associated with the last symbol in $\gamma([A,L]^{(o)},1)$, i.e. L, must properly include the set corresponding to the first symbol, L, which is a contradiction.

QED

Theorem 7.7 Given any (ℓ,m)-TS R there exists an (ℓ',m')-TS R' such that T(R) = T(R') and R' contains no loop failures.

Proof The theorem follows from lemmas 7.3, 7.4, 7.5.

QED

Remark We define an "ε-free TS" as a TS $R = (V,\Sigma,P,S,\$)$ which has no rule of the form $A..\varepsilon$, A in V. Using techniques similar to the ones used in lemmas 7.4, 7.5 it can be shown that for any ε-free TS the loop failures can be eliminated (some subroutine failures might be introduced at this time). In what follows we will show that p-a and end failures can also be eliminated from any given (ℓ,m)-TS. First we have the definition:

Definition 7.10 Let $R=(V,\Sigma,P,S,g,\$)$ be an (ℓ,m)-TS; we define the (ℓ',m')-TS $R' = (V',\Sigma,P',\bar{S},g',\$)$ as follows: $\ell' = 2\ell$, $m' = \ell+m+1$, $V' = \{A,\bar{A} \mid \text{all A in V}\}\cup\{X \mid X \text{ is not in V}\}$, (Intuitively, for every successful outcome $i \leq \ell$ in R there are two successful outcomes in R', namely i and $i + \ell$; also for every variable A in V there are two variables, A and $\bar{A}$, in V'. The variable $\bar{A}$ in R' simulates A in R as follows: if $A \underset{R}{\Longrightarrow} (x\lceil y,i)$ for some xy in Σ^* then $\bar{A} \underset{R'}{\Longrightarrow} (x\lceil y,i+\ell)$;

if $A \underset{R}{\Rightarrow} (x\lceil y,i)$ for some x in $\Sigma^*\$$, then $\bar{A} \underset{R'}{\Rightarrow} (x\lceil y,i)$. In other words, if the string recognized in R with outcome i does not contain \$, the outcome in R' is ℓ+i; once \$ is recognized the outcome is i. Moreover, for any A in V, A is called in R' only after the endmarker has been scanned successfully by the read-head. The purpose of this schema is twofold: firstly, if \$ is not recognized and the outcome in R is 1, the outcome in R' is ℓ+1 (and not 1) and thus p-a failures are avoided; secondly, after \$ has been scanned successfully only variables of the form A, A in V, are called and we can see to it that no attempts are then made to match an input symbol which would cause an end failure). P' contains the following rules:

1) If $A..(Q_1,\ldots,Q_m)$ is in P and no Q_i contains ε for $1 \leq i \leq m$, then let k be such that \$ is in Q_k. In P' we have the rule $\bar{A}..(Q'_1,Q'_2,\ldots,Q'_m)$ where $Q'_{p+i} = Q_i$ for $1 \leq i \leq m$ and $i \neq k$; if $k > \ell$ then $Q'_{k+\ell} = Q_k$, otherwise $Q'_{k+\ell} = Q_k - \{\$\}$, $Q'_k = \{\$\}$. Also in P' we have $A..(\phi,\ldots,\phi,\{\varepsilon\})$. (According to the rule for $\bar{A}$, an outcome $i \leq \ell$ will occur in R' only when the endmarker has been successfully matched. We notice that all sets Q'_i, $1 \leq i \leq m'$, which have not been specifically described are implicitly defined as empty according to the restrictions in Definition 7.2).

2) If $A..(Q_1,\ldots,Q_m)$ is in P and Q_k contains ε, for some k, then P' contains $\bar{A}..(Q'_1,Q'_2,\ldots,Q'_m)$ and $A..(Q''_1,Q''_2,\ldots,Q''_m,)$ where $Q'_{k+\ell} = \{\varepsilon\}$, $Q'_i = \phi$ for $1 \leq i \leq m'$ and $i \neq k+\ell$; $Q''_p = \{\varepsilon\}, Q''_i = \phi$ for $1 \leq i \leq m'$ and $i \neq p$, where $p = k$ if $k \leq \ell$, $p = k+\ell$ if $k > \ell$.

3) If $A..B(C_1,\ldots,C_m)$ is in P, P' contains $\bar{A}..\bar{B}\ (X_1,X_2,\ldots,X_{m'})$ and $A..B(Y_1,Y_2,\ldots,Y_{m'})$ where $X_i = C_i$ for $1 \leq i \leq \ell$, $X_{\ell+i} = \bar{C}_i$ for $1 \leq i \leq m$, $X_{m'} = X$; $Y_i = C_i$ for $1 \leq i \leq \ell$, $Y_{\ell+i} = C_i$ for $1 \leq i \leq m$, $Y_{m'} = X$.

4) $X..(\phi,\ldots,\phi,\{\varepsilon\})$. g' is defined as follows:

1) If $g(i,j) = k$, $i,j,k \leq \ell$, then $g'(\ell+i,\ell+j) = \ell+k$, $g'(\ell+i,j) = g'(i,j) = g'(i,j+\ell) = k$.

2) If $g(i,j) = k$, $i,k \leq \ell$, $j > \ell$, then $g'(\ell+i,\ell+j) = \ell+k$, $g'(i,\ell+j) = k$.

3) If $g(i,j) = k$, $j,k \leq \ell$, $i > \ell$, then $g'(\ell+i,\ell+j) = \ell+k$, $g'(\ell+i,j) = k$.

4) If $g(i,j) = k$, $k \leq \ell$, $i,j > \ell$ then $g'(\ell+i,\ell+j) = \ell+k$.

5) If $g(i,j) = k$, $k > \ell$, then $g'(i,j) = g'(i+\ell,j) = g'(i,j+\ell) = g'(i+\ell,j+\ell) = k+\ell$.

6) $g'(i,\ell+m+1) = g'(\ell+m+1,i) = \ell+m+1$ for $1 \leq i \leq \ell+m+1$.

Lemma 7.6 Consider R,R' as given by Definition 7.10. Then for all A in V, x in Σ^*, y in $\Sigma^*\$$:

a) $A \underset{R}{\Rightarrow} (x\uparrow y,i)$, $1 \leq i \leq m$ iff $\bar{A} \underset{R'}{\Rightarrow} (x\uparrow y, \ell+i)$

b) $A \underset{R}{\Rightarrow} (y\uparrow,i)$, $i \leq \ell$ iff $\bar{A} \underset{R'}{\Rightarrow} (y\uparrow,i)$

c) $A \underset{R}{\Rightarrow} (\uparrow,i)$, $1 \leq i \leq \ell$ iff $A \underset{R'}{\Rightarrow} (\uparrow,i)$ or $A \underset{R'}{\Rightarrow} (\uparrow,\ell+i)$

d) $A \underset{R}{\Rightarrow} (\uparrow,i)$, $\ell < i \leq m$ iff $A \underset{R'}{\Rightarrow} (\uparrow,\ell+i)$.

Proof "Only if" We prove these statements simultaneously, by induction on n, the number of steps in the derivation in R.

n = 1 Let the rule for A be $A..(Q_1,Q_2,\ldots,Q_m)$.

a) $A \overset{1}{\underset{R}{\Longrightarrow}} (x\uparrow y,i)$, x in Σ^*, y in $\Sigma^*\$$.

Case 1 $i \leq \ell$ and no Q_j contains ε; then $x = a$ for some a in Σ, also a is in Q_i. Using the rule for $\bar{A}$ we get $\bar{A} \underset{R'}{\Longrightarrow} (a\uparrow y,\ell+i)$.

Case 2 $i \leq \ell$ and Q_i contains ε; then $\bar{A}..(Q'_1,Q'_2,\ldots,Q'_m,)$ in R' where $Q'_{i+\ell} = \{\varepsilon\}$. Thus, we have $\bar{A} \underset{R'}{\Longrightarrow} (\uparrow y,\ell+i)$.

Case 3 $i > \ell$; since the outcome is a failure we have $x = \varepsilon$. Using now the rule for $\bar{A}$ we get $\bar{A} \underset{R'}{\Longrightarrow} (\uparrow y,\ell+i)$.

b) $A \overset{1}{\underset{R}{\Longrightarrow}} (y\uparrow,i)$, $i \leq \ell$; we must have $y = \$$ and $\$$ is in Q_i. In R' $\bar{A}..(Q'_1,Q'_2,\ldots,Q'_m,)$ where $Q'_i = \{\$\}$ and therefore $\bar{A} \underset{R'}{\Longrightarrow} (\$\uparrow,i)$.

c,d) $A \overset{1}{\underset{R}{\Longrightarrow}} (\uparrow,i)$ implies Q_i contains ε; if $i \leq \ell$ $A \underset{R'}{\Longrightarrow} (\uparrow,i)$ otherwise $A \underset{R'}{\Longrightarrow} (\uparrow,\ell+i)$.

Induction step Let the rule for A in P be $A..B(C_1,\ldots,C_m)$.

a) $A \overset{n}{\underset{R}{\Longrightarrow}} (x\uparrow y,i)$, x in Σ^*, y in $\Sigma^*\$$. We assume $B \underset{R}{\Longrightarrow} (x_1\uparrow x_2x_3, j)$, $C_j \underset{R}{\Longrightarrow} (x_2\uparrow x_3,k)$, $x_1x_2x_3 = xy$, $g(j,k) = i$.

Case 1 $x_3 \neq \varepsilon$; then x_1,x_2 in Σ^* and we can use the inductive hypothesis a) as follows: $\bar{B} \underset{R'}{\Longrightarrow} (x_1\uparrow x_2x_3,\ell+j)$, $\bar{C}_j \underset{R'}{\Longrightarrow} (x_2\uparrow x_3,\ell+k)$. Using now the rule for $\bar{A}$ we get $\bar{A} \underset{R'}{\Longrightarrow} (x\uparrow y,\ell+i)$.

Case 2 $x_3 = \varepsilon$, $x_2 \neq \varepsilon$; by induction $\bar{B} \underset{R'}{\Longrightarrow} (x_1\uparrow x_2,\ell+j)$. Using the inductive hypothesis b) we have $\bar{C}_j \underset{R'}{\Longrightarrow} (x_2\uparrow, k)$. However, we must have in this case $i > \ell$ (otherwise $y = \varepsilon$) and applying the definition for g' we get $\bar{A} \underset{R'}{\Longrightarrow} (x\uparrow y,\ell+i)$.

Case 3 $x_3 = x_2 = \varepsilon$; this case is similarly treated.

b) $A \overset{n}{\underset{R}{\Longrightarrow}} (y\uparrow,i)$, $i \neq \ell$; let $B \underset{R}{\Longrightarrow} (y_1\uparrow y_2, j)$, $C_j \underset{R}{\Longrightarrow} (y_2\uparrow,k)$, $y_1y_2 = y$, $g(j,k) = i$.

Case 1 $y_2 = \varepsilon$; by induction $\bar{B} \underset{R'}{\Longrightarrow} (y_1\uparrow,j)$. Applying the inductive hypothesis c) we have $C_j \underset{R'}{\Longrightarrow} (\uparrow,k)$ or $C_j \underset{R'}{\Longrightarrow} (\uparrow,k+\ell)$. In both cases, the rule for $\bar{A}$ and the definition of g' lead to $\bar{A} \underset{R'}{\Longrightarrow} (y\uparrow,i)$.

Case 2 $y_2 \neq \varepsilon$; this case is similar to case 1.

c) $A \overset{n}{\underset{R}{\Longrightarrow}} (\uparrow,i)$, $i \leq \ell$; $B \underset{R}{\Longrightarrow} (\uparrow,j)$, $C_j \underset{R}{\Longrightarrow} (\uparrow,k)$, $g(j,k) = i$. By induction $B \underset{R'}{\Longrightarrow} (\uparrow,\bar{j})$, $C_j \underset{R'}{\Longrightarrow} (\uparrow,\bar{k})$ where $\bar{j}$ is either j or $\ell+j$, $\bar{k}$ is either k or $\ell+k$; by the definition of g' we get $g'(\bar{j},\bar{k})$ is either i or $\ell+i$.

d) $A \overset{n}{\underset{R}{\Longrightarrow}} (\uparrow,i)$, $i > \ell$; using the definition of g' and the inductive hypothesis we get $A \underset{R'}{\Longrightarrow} (\uparrow,\ell+i)$.

The "if" part can be proved in the same way.

QED

Lemma 7.7 Consider R,R' as given by Definition 7.10. Then R' has no p-a or end failures.

Proof First we prove that for no A in V, x in Σ^*, y in $\Sigma^*\$$, integer i, $i \leq \ell$, we have $\bar{A} \overset{n}{\underset{R'}{\Longrightarrow}} (x\uparrow y,i)$. The proof is by induction on n.

n = 1 The rule for A in P must have the form: $A..(Q_1,\ldots,Q_m)$; it is easily checked, with the rule for $\bar{A}$ in R', that the claim made is true.

Induction step Let the rule for $\bar{A}$ be $\bar{A}..\bar{B}(X_1,X_2,\ldots,X_{m'})$. If the derivation $\bar{A} \overset{n}{\underset{R'}{\Longrightarrow}} (x\uparrow y,i)$ holds, we must have $\bar{B} \underset{R'}{\Longrightarrow} (x_1\uparrow x_2x_3, j)$, $X_j \underset{R'}{\Longrightarrow} (x_2\uparrow x_3, k)$, $x_1x_2x_3 = xy$, $g(j,k) = i$.

Case 1 $x_3 = \varepsilon$; then i must be a failure (otherwise we have $y = \varepsilon$) and if so it is not true that $i \leq \ell$.

Case 2 $x_3 \neq \varepsilon$; we cannot have $j \leq \ell$, by induction. If $j > \ell$ then $X_j = \bar{C}_j$ for some C_j in V; again by induction we cannot have $k \leq \ell$ and by the definition of g' we cannot have $i \leq \ell$.

Thus, for no x in Σ^*, y in $\Sigma^*\$$ we have $\bar{S} \underset{R'}{\Rightarrow} (x\lceil y, 1)$ and therefore p-a failures are not possible. It remains to be shown that R' has no end failures.

We notice that in the definition of g', whenever one of the two variables of g' is smaller or equal to ℓ, the value of g' is also smaller or equal to ℓ unless it is a failure. In A(R') an outcome smaller or equal to ℓ is obtained for the first time only when the endmarker has been successfully matched; moreover, if no backtracking occurs afterwards, then according to the observation above the outcome will always be smaller or equal to ℓ . Now, by the rules in P', only variables of the type A, for some A in V, are called following these outcomes and, as it is easy to verify, these variables will never attempt to match an input symbol. In other words, after the endmarker has been successfully scanned (for the last time, if this happens more than once) only variables of the type A, for some A in V, are called and, as it was shown before, these variables never attempt to match an input symbol. Hence, no end failures are possible. QED

Theorem 7.8 Given any (ℓ,m)-TS R there exists an (ℓ',m')-TS R' such that T(R)=T(R') and R' has no p-a or end failures. Moreover, if R has no loop failures, then R' will have none either.

Proof From Lemma 7.6, statement b), we conclude that T(R)=T(R'). By Lemma 7.7 R' has no p-a or end failures. Assume that R has no loop failures. If R' has loop failures, suppose $\bar{A}$ has a loop failure on x, x in $\Sigma^*\$$. In R we cannot have $A \overset{*}{\underset{R}{\Rightarrow}} (x_1 \uparrow x_2, i)$ for $x_1 x_2 = x$, $1 \leq i \leq m$ (this will contradict Lemma 7.6). The only possibility is that A has (in R) on x an end failure; the attempt in R to match an input symbol after the endmarker has been successfully scanned corresponds in R' to outcome ℓ+m+1. By the rules in R', this will also be the outcome of the derivation and no loop is thus possible in R'.

QED

The previous results can now be summarized in the following two theorems:

Theorem 7.8 Given any (ℓ,m)-TS R there exists a reduced (ℓ',m')-TS R' such that T(R) = T(R').

Proof The theorem follows from Theorem 7.7, 7.8 and the observation by by Definition 7.2 an (ℓ,m)-TS does not have any subroutine failures.

QED

As a corollary to this theorem we get the following important result:

Corollary Given any (ℓ,m)-TS R, there exists a gTS R' such that T(R)=T(R').

Proof Using the theorem above we go from R to R_1, which is reduced. By Theorem 7.6 we get a (reduced) gTS R' such that $T(R')=T(R_1)$. QED

Theorem 7.9 Given any gTS R, R = (V,Σ,P,S,$), there exists a reduced gTS R' such that T(R) = T(R').

Proof This theorem follows from Theorem 7.5, 7.6, 7.8. QED

Theorem 7.9 shows that in a gTS all loop, p-a and end failures can be eliminated and the proof provides a procedure for constructing the reduced gTS. From a practical point of view, a recognition schema which is "reduced" seems to be desired since the program will always terminate and it will never loop or halt before coming up with the final answer.

As was mentioned before, the TS is a formalization of the recognition schema used in the TMG system. The concept of a "well-formed" was introduced in order to describe the same desirable features in a TS that the concept of "reduced" underlines for the gTS model. By comparing the definition 3.1 and 7.5 we notice that the property of being "well-formed" for a recognition schema is a restriction of the "reduced" property to be distinguished symbol S. Since the distinguished symbol is our main interest in this case, the concept of "reduced" was introduced only to illustrate a stronger result in the case of the gTS. We do not know if a given TS can be made "well-formed" and various observations, and among them Theorem 4.8, ("It is undecidable whether an arbitrary TS is a wfTS") lead us to believe that this is not possible. The fact, illustrated by Theorem 7.9, that every gTS can be made "reduced" seems to indicate an advantage in using the gTS instead of the TS. The gTSL include,properly - we believe, the TSL and moreover, there is no apparent difficulty in using the gTS instead of the TS.

Next, we will describe the eTS, another abstract model of a recognition schema and a generalization of the TS.

Definition 7.11 An extended TS (eTS) R is a 5-tuple R = (V,Σ,P,S,$) in which:

V is a finite set of variables,

Σ is a finite set of terminal symbols,

S is an element of V,

$ is the endmarker, $ is not in Σ,

P is a finite set of rules of the form a) or b):

a) A..B(C,D), A,B,C and D in V,

b) A..a, a in $\Sigma_e \cup \{\epsilon\} \cup \{f_A | \text{all A in V}\}$, where f_A are metasymbols not in Σ_e and ε is the null string. (Instead of one metasymbol f, we have in eTS one metasymbol f_A for every variable A in V. The outcome of a derivation can be in eTS either in {0,1} or in $\{f_A | \text{A in V}\}$. If the outcome is f_A, for some A in V, this outcome "propagates" until it reaches the rule for A; in other words, an outcome f_A will cause every rule, of which it is part of, to have outcome f_A, until the rule involved is the rule for A, in which case the outcome becomes 1).

For any variable A there is at most one rule with A on the lefthand side.

We define the set of relations for each n in N, $A \overset{n}{\underset{R}{\Rightarrow}} (x \lceil y, p)$, xy in Σ_e^*, p in $\{0,1\} \cup \{f_A | \text{A in V}\}$, as follows:

1) If A..a is in P, a in Σ_e, then $A \overset{1}{\underset{R}{\Rightarrow}} (a \lceil x, o)$ for all x in Σ_e^*, and $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil bx, 1)$ for all x in Σ_e^*, b in Σ_e, $b \neq a$.

2) If $A..\varepsilon$ is in P, then $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil x,o)$ for all x in Σ_e^*.

3) If $A..f_A$ is in P, then $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil x,1)$ for all x in Σ_e^*

4) If $A..f_B$ is in P, $B \neq A$, then $A \overset{1}{\underset{R}{\Rightarrow}} (\lceil x, f_B)$ for all x in Σ_e^*.

5) Let $A..B(C,D)$ be in P. For each x_1, x_2, and x_3 in Σ_e^*, if:

a) $B \overset{\ell}{\underset{R}{\Rightarrow}} (x_1 \lceil x_2 x_3, 0)$, $C \overset{m}{\underset{R}{\Rightarrow}} (x_2 \lceil x_3, 0)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (x_1 x_2 \lceil x_3, 0)$, $k = \ell + m + 1$.

b) $B \overset{\ell}{\underset{R}{\Rightarrow}} (x_1 \lceil x_2, 0)$, $C \overset{m}{\underset{R}{\Rightarrow}} (\lceil x_2, 1)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1 x_2, 1)$, $k = \ell + m + 1$.

c) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\lceil x_1 x_2, 1)$, $D \overset{m}{\underset{R}{\Rightarrow}} (x_1 \lceil x_2, 0)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (x_1 \lceil x_2, 0)$, $k = \ell + m + 1$.

d) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\lceil x_1, 1)$, $D \overset{m}{\underset{R}{\Rightarrow}} (\lceil x_1, 1)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1, 1)$, $k = \ell + m + 1$.

e) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\lceil x_1, f_A)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1, 1)$, $k = \ell + 1$.

f) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\lceil x_1, f_E)$ and $E \neq A$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1, f_E)$, $k = \ell + 1$.

f) $B \overset{\ell}{\underset{R}{\Rightarrow}} (x_1 \lceil x_2, 0)$, $C \overset{m}{\underset{R}{\Rightarrow}} (\lceil x_2, f_E)$, $E \neq A$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1 x_2, f_E)$, $k = \ell + m + 1$.

g) $B \overset{\ell}{\underset{R}{\Rightarrow}} (x_1 \lceil x_2, 0)$, $C \overset{m}{\underset{R}{\Rightarrow}} (\lceil x_2, f_A)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1 x_2, 1)$, $k = \ell + m + 1$.

h) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\lceil x_1, 1)$, $D \overset{m}{\underset{R}{\Rightarrow}} (\lceil x_1, f_E)$, $E \neq A$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1, f_E)$, $k = \ell + m + 1$.

i) $B \overset{\ell}{\underset{R}{\Rightarrow}} (\lceil x_1, 1)$, $D \overset{m}{\underset{R}{\Rightarrow}} (\lceil x_1, f_A)$ then $A \overset{k}{\underset{R}{\Rightarrow}} (\lceil x_1, 1)$, $k = \ell + m + 1$.

If $A \overset{n}{\underset{R}{\Rightarrow}} (x \lceil y, p)$, xy in Σ_e^*, p in $\{0,1\} \cup \{f_A \mid A \text{ in } V\}$, we say A <u>derives</u> $(x \lceil y, p)$ <u>in n steps</u>. We say A <u>derives</u> $(x \lceil y, p)$ if A derives $(x \lceil y, p)$ in

r steps for some r, and we write $A \overset{*}{\underset{R}{\Rightarrow}} (x\uparrow y,p)$. The language recognized by R is: $T(R) = \{x \mid x \text{ in } \Sigma^*, S \overset{*}{\underset{R}{\Rightarrow}} (x\$\uparrow,0)\}$.

Theorem 7.10 Given any eTS R there exists a gTS R_1 such that $T(R_1) = T(R)$.

Proof Let $R = (V,\Sigma,P,S,\$)$. We will construct an (ℓ,m)-TS R' such that $T(R') = T(R)$. The theorem will follow according to the corollary of Theorem 7.8.

Consider the (ℓ,m)-TS $R' = (V',\Sigma,P',S,g,\$)$ in which: $\ell = 1$, $m = r+4$ where r is the number of variables in V. (There are in R' r+2 new outcomes, all of them failures; outcome r+4 will correspond to subroutine failures in R. Outcomes 3 through r+2, one for every variable in V, are designated for carrying over the failures of type f_A for A in V). Let $V = \{X_i \mid 1 \leq i \leq r\}$; then $V' = \{X_i, X_i' \mid 1 \leq i \leq r\} \cup \{Z,W\}$, where Z, W are not in V.

P' contains the following rules:

1) If A..a is in P, a in Σ_e, then P' contains $A..(Q_1,Q_2,\ldots,Q_m)$ where $Q_1 = \{a\}$, $Q_2 = \Sigma_e - \{a\}$, $Q_j = \emptyset$ for $j > 2$. (The outcome 1 stands for success, outcome 2 corresponds to a recognition failure in R).

2) If $A..\varepsilon$ is in P then P' contains $A..(Q_1,\ldots,Q_m)$ where $Q_1 = \{\varepsilon\}$, $Q_j = \emptyset$ for $j > 1$.

3) If $A..f_B$ is in P, then let $A = X_i$, $B = X_j$; if $i = j$ then P' contains $A..(Q_1,\ldots Q_m)$ where $Q_2 = \{\varepsilon\}$; if $i \neq j$ then $A..(Q_1,\ldots,Q_m)$ is in P' where $Q_j = \{\varepsilon\}$.

4) Let A..B(C,D) be in P and let A = X_i; in P' A'..B(Y_1,...,Y_m) where Y_1 = C, Y_2 = D, and Y_j = Z for j not in {1,2}. Also in P' A..A'(Y'_1,...,Y'_m) where Y'_i = W, Y'_j = Z for j ≠ i. (For every rule in P there are two rules in P'; let A be in V and A..B(C,D) in P. Then the rule for A' in P' will register success or recognition failure as outcomes 1 or 2; however, if the outcome for A is f_D, for some D in V - let D = X_i, A' will register outcome i+2. The rule for A in P' is intended to propagate all outcomes i, save outcome j+2 where j is such that A = X_j. For this reason variable W was introduced and the definition of g insures that in this case the outcome is 2, i.e. recognition failure).

5) If there is no rule for A in P, P' contains A..(Q_1,...,Q_m) where Q_m = {ε}; also in P', W..(Q'_1,...,Q'_m) where Q'_{m-1} = {ε}and Z..(Q''_1,...,Q''_m) where Q''_m = {ε}.

g is defined as follows:

1) g(k,m) = k for $1 \leq k \leq m$

2) g(1,k) = g(2,k) = k for $1 \leq k \leq m-2$

3) g(k,m-1) = 2 for $1 \leq k \leq m-2$

It can be shown by induction that T(R) = T(R').

QED

Remark It is straightforward to show that the converse of Theorem 7.10 also holds, i.e. for any gTS R there exists an eTS R_1 such that T(R)=T(R_1).

VIII. THE gTS AND THE ABSTRACT FAMILIES OF DETERMINISTIC LANGUAGES

An abstract family of deterministic languages (AFDL) is defined in [3] as a family of languages closed under the operations of "marked union", "marked*" (marked kleene closure) and inverse "marked gsm" mapping. It was shown that a family of languages is an AFDL if and only if there exists a 1DBA (one-way deterministic balloon automaton) which accepts exactly that family. (The concept of balloon automaton is found in [7]).

In a previous chapter we have shown that corresponding to a gTS is an automaton which accepts exactly the languages recognized by gTS's. However, this automaton is not similar to any class of balloon automata in the sense that it cannot be viewed as having an (unrestricted) finite control which operates on a read-only input tape and some sort of additional memory. In fact a distinctive restriction in a gTSA is on the number of states in the finite control. Such a restriction, regarded from a practical point of view, does not seem justified since in most cases a finite control unit is easily implemented in hardware. For this reason it is desirable to find out whether there exists a class of balloon automata which accepts the gTSL. We will show the answer to this question is positive by showing that the gTSL form an AFDL and using the mentioned result from [3].

First we will recall the definitions of "marked union, "marked*" and "marked gsm" mapping.

Definition 8.1 Let $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$, and the symbols a,b not in $\Sigma_1 \cup \Sigma_2$. The sets $(aL_1) \cup (bL_2)$ and $(aL_1)^*$ are called, respectively, a marked union of L_1 and L_2, a marked * of L_1.

A marked generalized sequential machine (mgsm) is a 7-tuple $G = (K,\Sigma,\Delta,\delta,\lambda, q_o,\$)$ where:

K is a finite set of states,

Σ is a finite set of input symbols,

Δ is a finite set of output symbols,

q_o is an element of K, the initial state,

$\$$ is a symbol called endmarker, $\$$ is not in $\Sigma \cup \Delta$,

δ is a mapping from $K \times \Sigma_e$ into K, (we use the notation Σ_e for $\Sigma \cup \{\$\}$),

λ is a mapping from $K \times \Sigma_e$ into $\Delta^* \cup \Delta^*\$$, such that if a is in Σ, q is in K and $\lambda(q,a) = x$ then x is in Δ^*; also if $\lambda(q,\$) = y$ then y is in $\Delta^*\$$.

The functions δ and λ are extended to mappings from $K \times \Sigma^*$ by defining $\delta(q,\varepsilon) = q$, $\lambda(q,\varepsilon) = \varepsilon$, $\delta(q,wa) = \delta[\delta(q,w),a]$ and $\lambda(q,wa) = \lambda(q,w) \cdot \lambda[\delta(q,w),a]$ for all q in K, w in Σ^*, a in Σ.

Let $G = (K,\Sigma,\Delta,\delta,\lambda,q_o,\$)$ be a mgsm: the function $G:2^{\Sigma^*} \to 2^{\Delta^*}$ defined by $G(X) = \{y \mid \lambda(q_o,x\$) = y\$, x \text{ in } X\}$ is called a mgsm mapping. The function $\bar{G}^{-1}: 2^{\Delta^*} \to 2^{\Sigma^*}$ defined by $\bar{G}^{-1}(Y) = \{x \mid \lambda(q_o,x\$) \text{ in } Y\$\}$ is called an inverse mgsm mapping.

Lemma 8.1 The gTSL are closed under marked union.

Proof Let R_1 and R_2 be two gTS, $R_1 = (V_1,\Sigma_1,P_1,S_1,\$)$ and $R_2 = (V_2,\Sigma_2,P_2,S_2,\$)$; assume $V_1 \cap V_2 = \phi$. Consider the gTS $R = (V,\Sigma,P,S,\$)$ and the symbols a,b not in $\Sigma_1 \cup \Sigma_2 \cup \{\$\}$. $V = V_1 \cup V_2 \cup \{S\}$ where S is a new variable, not in $V_1 \cup V_2$, $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{a,b\}$, P includes $P_1 \cup P_2$; it also contains $S..aS_1/bS_2$.

We claim $T(R) = a.T(R_1) \cup b.T(R_2)$. Let x be in T(R), x in Σ^*. Using the rule for S, we have either $x = ax_1$ and $S_1 \underset{R}{\Rightarrow} (x_1\$\uparrow,0)$ or $x = bx_2$ and $S_2 \underset{R}{\Rightarrow} (x_2\$\uparrow,0)$; it follows that either x is in $a.T(R_1)$ or x is in $b.T(R_2)$. Also, if x is in $T(R_1)$, x in Σ_1^*, then $S_1 \underset{R}{\Rightarrow} (x\$\uparrow,0)$ and $S \underset{R}{\Rightarrow} (ax\$\uparrow,0)$. Similarly for x in $T(R_2)$. QED

Lemma 8.2 The gTSL are closed under marked *.

Proof Let $R_1 = (V_1,\Sigma_1,P_1,S_1,\$)$ be a gTS; according to Theorem 7.9 we can assume R_1 is reduced and therefore there are no partial-acceptance or end failures. Consider also the gTS $R_1' = (V_1',\Sigma_1,P_1',S_1',a)$ which is identical to R_1 except for a different notation for variables ($V_1 \cap V_1' = \phi$) and the endmarker (a is not in Σ_1); otherwise the rules are similar and $T(R_1) = T(R_1')$.

Consider the gTS $R = (V,\Sigma,P,S,\$)$ where: V includes the set $V_1 \cup V_1' \cup \{S,A,B\}$, S,A,B not in $V_1 \cup V_1'$; V also contains the variables which are implicitly defined by the shorthand notation of the rules in P. $\Sigma = \Sigma_1 \cup \{a\}$. Let $\Sigma_1 = \{a_i \mid 1 \leq i \leq m\}$. P includes $P_1 \cup P_1'$ and also the following rules:

S..$/aA

A..(B)(S_1,S_1'A)

B..a_1B/a_2B/...../a_mB/$.

We will show $T(R) = [aT(R_1)]^*$. Let x be in T(R). We write $x = x_o\, ax_1\, ax_2 \ldots ax_k\$$, for some integer k and x_i in Σ_1^*, $0 \leq i \leq k$. First we notice $x_o = \epsilon$ since the rule for S shows any word in T(R) must have $ or a as the first letter. We can write $S_1' \underset{R}{\Rightarrow} (x_i\, a\uparrow,0)$ for $1 \leq i \leq k-1$; also $S_1 \underset{R}{\Rightarrow}$

$(x_k\$\uparrow,0)$. We conclude that x_i is in $T(R_1)$ for $1 \leq i \leq k$ and therefore x is in $[aT(R_1)]^*$.

Suppose now x is in $[aT(R_1)]^*$; it can be shown by simply applying the rules in P that x is in T(R). QED

Next we will show the gTSL are closed under inverse mgsm mapping. But first, we have the following definitions:

Definition 8.2 Let $G' = (K',\Delta,\Sigma,\delta',\lambda',q_o,\$)$ be a mgsm. We define the mgsm $G = (K,\Delta,\Sigma,\delta,\lambda,q_o,\$)$ as follows:

1) If $\delta'(q,b) = p$, $\lambda'(q,b) = \varepsilon$ for q,p in K', b in Σ_e, then K contains q,p and $\delta(q,b) = p$, $\lambda(q,b) = \varepsilon$.

2) If $\delta'(q,b) = p$, $\lambda'(q,b) = a_1a_2....a_k$, q,p in K', b in Δ_e and a_i in Σ_e for $1 \leq i \leq k$ and some k, $k \geq 1$, then K contains $q,q_1,....,q_{k-1}$ where q_i for $1 \leq i \leq k-1$, are new states, not in K', and $\delta(q,b) = q_1$, $\lambda(q,b) = a_1$; $\delta(q_i,\varepsilon) = q_{i+1}$, $\lambda(q_i,\varepsilon) = a_{i+1}$ for $1 \leq i \leq k-2$; also $\delta(q_{k-1},\varepsilon) = p$, $\lambda(q_{k-1},\varepsilon) = a_k$.

The mappings, $\bar{\lambda}$, $\bar{\delta}$, extensions of λ,δ over the domain $K \times \Delta_e^*$, are defined as follows

1) If $\delta(q,\varepsilon)$ is not defined above, then $\bar{\delta}(q,\varepsilon) = q$, $\bar{\lambda}(q,\varepsilon)=\varepsilon$.

2) If $\delta(q_i,\varepsilon) = q_{i+1}$, $\lambda(q_i,\varepsilon) = a_i$ for $1 \leq i \leq k-1$ and some $k > 1$, and $\delta(q_k,\varepsilon)$ is undefined, then $\bar{\delta}(q_1,\varepsilon) = q_k$ and $\bar{\lambda}(q_1,\varepsilon) = a_1a_2....a_{k-1}$.

3) For all x in Δ_e^*, b in $\Delta_e \cup \{\varepsilon\}$: $\bar{\delta}(q,xb) = \delta(\bar{\delta}(q,x),b)$ and $\bar{\lambda}(q,xb) = \bar{\lambda}(q,x).\ \lambda(\bar{\delta}(q,x),b)$. (We notice that the range of $\bar{\delta}$ includes no state q which has an ε-move).

We define the set of mappings $\{\delta_p \mid \text{all p in K}\}$ from $K \times \Delta_e^*$ to K,

and the set of mapping $\{\lambda_p \mid \text{all } p \text{ in } K\}$ from $K \times \Delta_e^*$ to Σ_e^*, (These mappingsare intended to facilitate the notation in later proofs. The subscript p, for some p in K, implies that δ_p has the value p or else is undefined; when, given an input string, $\bar{\delta}$ takes an initial state (q_1) to a final state (q_n) such that a number of ε-moves occur before reaching the final state q_n, then δ_p will take q_1 to p if p is one of the states on which the ε-moves have occurred):

1) If $\delta(q,\varepsilon)$ is undefined, then $\delta_q(q,\varepsilon) = q$, $\delta_p(q,\varepsilon)$ is undefined for all $p \neq q$, $\lambda_q(q,\varepsilon) = \varepsilon$, $\lambda_p(q,\varepsilon)$ is undefined for all $p \neq q$.

2) Let $\delta(q_i,\varepsilon) = q_{i+1}$, $\lambda(q_i,\varepsilon) = a_i$ for $1 \leq i \leq k-1$ and some $k > 1$, and $\delta(q_k,\varepsilon)$ is undefined; if $q_j = p$ for some $1 \leq j \leq k$, then $\delta_p(q,\varepsilon) = p$ and $\lambda_p(q,\varepsilon) = a_1 a_2 \ldots a_{j-1}$. If $q_j \neq p$ for all j, $1 \leq j \leq k$, then $\delta_p(q,\varepsilon)$, $\lambda_p(q,\varepsilon)$ are undefined.

3) If $\delta(q,b) = q_1$, b in Δ_e, then $\delta_p(q,b) = \delta_p(q_1,\varepsilon)$ and $\lambda_p(q,b) = \lambda(q,b).\ \lambda_p(q_1,\varepsilon)$.

4) For all x in Δ_e^*, b in Δ_e: $\delta_p(q,xb) = \delta_p(\bar{\delta}(q,x),b)$, $\lambda_p(q,xb) = \bar{\lambda}(q,x).\ \lambda_p(\bar{\delta}(q,x),b)$.

Definition 8.3 Let $R = (V,\Sigma,P,S,\$)$ be a gTS, let G be the mgsm in Definition 8.2, $\Sigma = \{a_i \mid 1 \leq i \leq m\}$, $\Delta = \{b_i \mid 1 \leq i \leq n\}$, $K = \{q_i \mid 1 \leq i \leq r\}$. We define the gTS $R' = (V',\Delta,P',S',\$)$ as follows: $V' = \{[qAp],[qAp;\lambda] \mid q,p \text{ in } K, A \text{ in } V\} \cup \{S'\} \cup \{[qAp;i] \mid q, p \text{ in } K, A \text{ in } V, i = 2,3,\ldots,r\}$. (The variables in V' of the form [qAp] will simulate derivations in R on strings in $\Delta^*\$$ and in the same time, they will keep track of the states in G. Variables $[qAp;\lambda]$ are intended to recognize strings in Δ^* which map into ε. Variables [qAp;i] are introduced to

facilitate the writting of the rules in P').

P' contains the following rules:

(1) Let K_1 be $K_1 = \{q \mid q \text{ in } K \text{ and } \delta(q,\varepsilon) \text{ undefined}\}$; we denote the elements of K_1 by $K_1 = \{p_i \mid 1 \leq i \leq \ell\}$. Then, $S'..[q_oSp_1]/[q_oSp_2]/..... /[q_oSq_\ell]$. ($K_1$ contains only states with no ε-move).

(2) Let P contain $A..B(C,D)$; for all q,p in K,P' contains: $[qAp]..[qBq_1]([q_1Cp], [qAp; 2])$, $[qAp;i]..[qBq_1]([q_iCp],[qAp;i+1])$, for $i = 2,3,....r-1$. $[qAp;r]..[qBq_r]([q_rCp],[qDp])$.

(3) Let $\Delta_q = \{b \mid b \text{ in } \Delta_e \text{ and } \lambda(q,b) = \varepsilon\}$; we denote the elements of Δ_q by $\Delta_q = \{b^{(i)} \mid 1 \leq i \leq \ell\}$. Then, $[qAp;\lambda]..b^{(1)}[q^{(1)}Ap]/b^{(2)}[q^{(2)}Ap]/ ...../b^{(\ell)}[q^{(\ell)}Ap]$ where $q^{(i)} = \delta(q,b^{(i)}), 1 \leq i \leq \ell$.

(4) Let $A..a$ be in P, a in Σ_e; if there are p,q in K such that $\delta(q,\varepsilon) = p$, $\lambda(q,\varepsilon) = a$, then $[qAp]..\varepsilon$ and $[qAs]..f$ for all s in k, $s \neq p$. If there is b in Δ_e, q,p in k such that $\delta(q,b) = p$, $\lambda(q,b) = a$, then $[qAp]..[qAp;\lambda]/b$. For any other case $[qAp]..f$.

(5) If $A..\varepsilon$ is in P, then for all q,p in k and $q \neq p$, $[qAq]..\varepsilon$, $[qAp]..f$.

(6) If $A..f$ is in P, then $[qAp]..f$ for all q,p in K.

Now we can prove the following lemma:

Lemma 8.3 The gTSL are closed under inverse mgsm mapping.

Proof Consider R,R',G as given by definitions 8.2, 8.3.

Part one Let x be in T(R'), x in Δ^*; we will show $\lambda(q_o,x\$) = y\$$ and y is in T(R). First we prove the following statements:

a) If $[qAp] \overset{n'}{\underset{R'}{\Rightarrow}} (x_1 \uparrow x_2, 0)$, x_1,x_2 in Δ_e^*, then $A \overset{}{\underset{R}{\Rightarrow}} (y_1 \uparrow y_2, 0)$ where $y_1 = \lambda_p(q,x_1)$, $y_2 = \bar{\lambda}(p,x_2)$.

b) If $[qAp] \overset{n(p)}{\underset{R'}{\Rightarrow}} (\uparrow x, 1)$ for all p in K and some integers n(p), then $A \underset{R}{\Rightarrow} (\uparrow y, 1)$ where $y = \bar{\lambda}(q,x)$.

Let $n' = \max_p [n(p)]$. We prove these statements simultaneously, by induction on n'.

n' = 1 a) $[qAp] \overset{1}{\underset{R'}{\Rightarrow}} (x_1 \uparrow x_2, 0)$ implies a rule $[qAq]..\varepsilon$ was used in the derivation and therefore $x_1 = \varepsilon$, $p = q$. Then $\lambda_q(q,\varepsilon) = \varepsilon = y_1$ and using the rule $A..\varepsilon$ in P we get $A \underset{R}{\Rightarrow} (\uparrow y_2, 0)$.

b) We must have in this case $[qAp]..f$ for all p in K. This implies A..f in P and therefore $A \underset{R}{\Rightarrow} (\uparrow y, 1)$.

Induction step a) Assume $[qAp] \overset{n'}{\underset{R'}{\Rightarrow}} (x_1 \uparrow x_2, 0)$, $n' > 1$. There are only three cases possible:

Case 1 The first rule used in the derivation is of the form $[qAp]..[qAp;\lambda]/b$. Assume first that $[qAp;\lambda]$ succeeds on x_1x_2, $(qAp;\lambda] \underset{R'}{\Rightarrow} (x_1 \uparrow x_2, 0)$. Then using the rule for [qAp;], there is $b^{(j)}$ in Δ_e and $x_1 = b^{(j)}x_3$; also $[q^{(j)}Ap] \underset{R'}{\Rightarrow} (x_3 \uparrow x_2, 0)$. Using the inductive hypothesis and the relation $\lambda(q,b^{(j)}) = \varepsilon$ we get $A \underset{R}{\Rightarrow} (y_1 \uparrow y_2, 0)$.

If $[qAp;\lambda]$ fails, then $x_1 = b$ and $\delta(q,b) = p$; we have $\lambda(q,b) = a$ and $A..a$ in P. Therefore $A \underset{R}{\Rightarrow} (a \uparrow y_2, 0)$.

Case 2 The rule for [qAp] has the form (2) in the definition of P' and there is an integer j such that $[qBq_j]$ succeeds on x_1x_2. Let $[qBq_j] \underset{R'}{\Rightarrow} (x_3 \uparrow x_4x_2, 0)$, for some x_3, $x_3x_4 = x_1$; then $[q_jCp] \underset{R'}{\Rightarrow} (x_4 \uparrow x_2, 0)$. By induction $B \underset{R}{\Rightarrow} (y_3 \uparrow y_4y_2, 0)$, $C \underset{R}{\Rightarrow} (y_4 \uparrow y_2, 0)$ where $y_3 = \lambda_{q_j}(q,x_3)$, $y_4 = \lambda_p(q_j,x_4)$; using the rule for A, $A..B(C,D)$ we get $A \underset{R}{\Rightarrow} (y_3y_4 \uparrow y_2, 0)$ where $y_3y_4 = \lambda_p(q,x_3x_4) = \lambda_p(q,x_1)$.

Case 3 $[qBq_j]$ fails, for all q_j in K, and $[qDp]$ succeeds on x_1x_2: $[qBq_j] \underset{R'}{\Rightarrow} (\uparrow x_1x_2, 1)$, $[qDp] \underset{R'}{\Rightarrow} (x_1 \uparrow x_2, 0)$. By the inductive hypothesis b) we have $B \underset{R}{\Rightarrow} (\uparrow x_1x_2, 1)$; finally we get $D \underset{R}{\Rightarrow} (y_1 \uparrow y_2, 0)$ where $y_1 = \lambda_p (q, x_1)$ and using the rule A..B(C,D) we get $A \underset{R}{\Rightarrow} (y_1 \uparrow y_2, 0)$.

b) Assume $[qAp] \underset{R'}{\overset{n(p)}{\Rightarrow}} (\hat{x}, 1)$ for all p in K and some integers n(p), $n' > 1$. There are three possible cases:

Case 1 The rule for $[qA_p]$ has the form (4) in the definition of P', that is the corresponding rule in P has the form A..a, a in Σ_e. Suppose $y = ay_1$; then there is b in Σ_e, b is mapped into a by λ. It follows, according to rules (4) that in this case we will have $[qAs] \underset{R'}{\Rightarrow} (x_1 b \uparrow x_2, 0)$, for some s in k and x_2, x_3 such that $X = x_1 b x_2$. Since this contradicts the hypothesis, we must have $y = a'y_1$, $a' \neq a$; we conclude that $A \underset{R}{\Rightarrow} (\uparrow y, 1)$.

Case 2 The rule for [qAp] has the form (2) and $[qBq_i]$ fails on x, $1 \leq i \leq j-1$, $[qBq_j]$ succeeds on x. Let $[qBq_j] \underset{R'}{\Rightarrow} (x_1 \uparrow x_2, 0)$; then $[q_jCp]$ will fail on x_2 for all p in K. By the inductive hypothesis follows $B \underset{R}{\Rightarrow} (y_1 \uparrow y_2, 0)$, where $y_1 = \lambda_{qj}(q, x_1)$, $y_2 = \bar{\lambda}(q_j, x_2)$, also $C \underset{R}{\Rightarrow} (\uparrow y_2, 1)$. Finally the rule A..B(C,D) leads to $A \underset{R}{\Rightarrow} (\uparrow y, 1)$, $y = y_1y_2 = \bar{\lambda}(q, x)$.

Case 3 $[qBq_i]$ fails on x for all q_i in K. Then [qDp] will fail on x too, for all p in K; by induction we get $B \underset{R}{\Rightarrow} (\uparrow y, 1)$, $y = \bar{\lambda}(q, x)$ and finally $A \underset{R}{\Rightarrow} (\uparrow y, 1)$.

Consider a string x in T(R'); $[q_o S \bar{q}_i] \underset{R'}{\Rightarrow} (x\$\uparrow, 0)$ for some $\bar{q}_i$ in K such that $\delta(\bar{q}_i, \varepsilon)$ is not defined. Then $S \underset{R}{\Rightarrow} (y\$\uparrow, 0)$, where $y\$ = \lambda_{\bar{q}_i}(q_o, x\$) = \bar{\lambda}(q_o, x\$)$ and therefore $\bar{\lambda}(q_o, x\$)$ is in T(R).

Part two We will show that if x is in Δ^* and $\bar{\lambda}(q_o,x\$)$ is in T(R)\$ then x is in T(R'). Consider, x_1 in Δ_e^+, x_2 in Δ_e^* and q in K such that if $x_1 = x_1' b$ for some b in Δ_e and $\bar{\delta}(q,x_1') = p$ for some p in K then $\lambda(p,b) \neq \epsilon$. (This restriction on all 3-tuples (x_1,x_2,q) means that when x_1 is the input on G from state q, the last symbol of x_1 will produce some output). For all x_1 in Δ_e^+, x_2 in Δ_e^*, q in K with the above restriction:

a) Let $y_1 = \lambda_p(q,x_1)$ for some p in K, $y_2 = \bar{\lambda}(p,x_2)$; if $A \overset{n'}{\underset{R}{\Rightarrow}} (y_1 \uparrow y_2,0)$ then $[qAp] \underset{R'}{\Rightarrow} (x_1 \uparrow x_2,0)$ and $[qAs] \underset{R'}{\Rightarrow} (\uparrow x_1 x_2,1)$ for all s in K and $s \neq p$.

b) Let $y_1 = \bar{\lambda}(q,x_1)$; if $A \overset{n'}{\underset{R}{\Rightarrow}} (\uparrow y_1,1)$ then $[qAp] \underset{R}{\Rightarrow} (\uparrow x_1,1)$ for all p in K.

The proof is by induction on n'.

n' = 1 a) $A \overset{1}{\underset{R}{\Rightarrow}} (y_1 \uparrow y_2,0)$ implies $y_1 = a$, a in $\Sigma_e \cup \{\epsilon\}$ and A..a in P.

Case 1 a in Σ_e; let $|x_1| = \ell$(by $|x|$ we designate the length of the string x). We prove this case by induction on ℓ. First, if $\ell = 0$ we have $b = \epsilon$; also $\delta(q,\epsilon) = p$ and $\lambda(q,\epsilon) = a$. It follows that [qAp]..ε and [qAs]..f, for all s in K, $s \neq p$; this leads to the desired result. For the induction step let $|x_1| = \ell$, $\ell > 0$. There is b in Δ_e such that $x_1 = x_3 b x_4$, $\bar{\delta}(q,x_3) = p_1$. $\bar{\lambda}(q,x_3) = \epsilon$, $\lambda(p_1,b) = a$; also if $\delta(p_1,b) = p_2$ then $\bar{\lambda}(p_2,x_4) = \epsilon$ from which we conclude that $x_4 = \epsilon$. Therefore $x_1 = x_3 b$, and $\bar{\lambda}(q,x_3) = \epsilon$. Applying the rule for [qAp], of the form (4), $[qAp;\lambda]$ succeeds on $x_3 b$; let $x_3 = b^{(1)} x_3'$. Then, by induction on $x_3' b$ we get $[q^{(1)}Ap] \underset{R'}{\Rightarrow} (x_3' b \uparrow x_2,0)$ and finally $[qAp] \underset{R'}{\Rightarrow} (x_1 \uparrow x_2,0)$.

Case 2 a = ε is treated similarly.

b) $A \overset{1}{\underset{R}{\Longrightarrow}} (\uparrow y_1, 1)$; two cases arise:

Case 1 A has a rule A..f; then according to rules (6) in the definition of P', [qAp]..f for all p in K and the result follows.

Case 2 A has a rule A..a, a in Σ_e and $y_1 = a'y_2$ for some a' in Σ_e, $a' \neq a$. First assume $\delta(q,\varepsilon) = p_1$. Then $\lambda(q,\varepsilon) = a'$ and according to rules (4), [qAp]..f for all p in K; the result follows. Now, suppose $\delta(q,\varepsilon)$ is undefined; we have x_3 in Δ_e^*, b' in Δ_e such that $x_1 = x_3 b' x_4$, $\bar{\delta}(q,x_3) = p_1$, $\bar{\lambda}(q,x_3) = \varepsilon$, $\lambda(p_1,b') = a'$. Let $|x_3| = \ell$; if $\ell = 0$ then applying rules (4) we get [qAs]..f for all s in K. For $\ell > 0$ it can be shown inductively the result still holds.

Induction step a) $A \overset{n'}{\underset{R}{\Longrightarrow}} (y_1 \uparrow y_2, 0)$, $n' > 1$. Then the rule for A has the form A..B(C,D). Two cases are possible:

Case 1 $B \underset{R}{\Longrightarrow} (y_3 \uparrow y_4 y_2, 0)$, $C \underset{R}{\Longrightarrow} (y_4 \uparrow y_2, 0)$ and $y_1 = y_3 y_4$. Consider the rule (2) for [qAp]; let x_3 be the shortest string such that $x_1 = x_3 x_4$ and $y_3 = \lambda_{p_1}(q,x_3)$ for some p_1 in K. Then if $x_3 = x_3' b$, for some b in Δ_e, $\lambda(\bar{\delta}(q,x_3'), b) \neq \varepsilon$ because otherwise the assumption x_3 is the shortest string is contradicted. By the inductive hypothesis $[qBp_1] \underset{R'}{\Longrightarrow} (x_3 \uparrow x_4 x_2, 0)$ and $[qBs] \underset{R'}{\Longrightarrow} (\uparrow x_1 x_2, 1)$ for all s in K, $s \neq p_1$; also by induction $[p_1 Cp] \underset{R'}{\Longrightarrow} (x_4 \uparrow x_2, 0)$ which establishes the claim.

Case 2 $B \underset{R}{\Longrightarrow} (\uparrow y_1 y_2, 1)$, $D \underset{R}{\Longrightarrow} (y_1 \uparrow y_2, 0)$. Applying the inductive hypothesis a) and b) we get $[qBp] \underset{R'}{\Longrightarrow} (\uparrow x_1 x_2, 1)$ for all p in K and $[qDP] \underset{R'}{\Longrightarrow} (x_1 \uparrow x_2, 0)$. Using the rule (2) for [qAp] we get the desired result.

b) $A \overset{n'}{\underset{R'}{\Rightarrow}} (\uparrow y_1, 1)$, $n' > 1$. The rule for A: A..B(C,D). Two cases are possible:

Case 1 $B \underset{R}{\Rightarrow} (y_3 \uparrow y_4, 0)$, $C \underset{R}{\rightarrow} (\uparrow y_4, 1)$, $y_1 = y_3 y_4$. We define x_3, x_4 as above and by induction we get $[qBp_1] \underset{R'}{\rightarrow} (x_3 \uparrow x_4, 0)$ for some p_1 in K and $[qBs] \underset{R'}{\Rightarrow} (\uparrow x_3 x_4, 1)$ for all s in K, $s \neq p_1$. Also by induction we have $[p_1 C p] \underset{R'}{\Rightarrow} (\uparrow x_4, 1)$. Using the rule (2) for [qAp] we get $[qAp] \underset{R'}{\Rightarrow} (\uparrow x_1, 1)$ for all p in K.

Case 2 $B \underset{R'}{\Rightarrow} (\uparrow y_1, 1)$, $C \underset{R'}{\Rightarrow} (\uparrow y_1, 1)$. This case is similarly treated.

Consider x in Δ^*; let $y\$ = \bar{\lambda}(q_o, x\$)$ and $S \underset{R}{\Rightarrow} (y\$\uparrow, 0)$. By applying the result above we get $[q_o S \bar{q}] \underset{R'}{\Rightarrow} (x\$\uparrow, 0)$, where $\bar{q} = \bar{\delta}(q_o, x\$)$, and $[q_o S p] \underset{R'}{\Rightarrow} (\uparrow x\$, 1)$ for all p in K, $p \neq \bar{q}$, which imply x is in T(R').

QED

Theorem 8.1 The gTSL form an AFDL.

Proof Follows from the definition of an AFDL and lemmas 8.1, 8.2, 8.3.

QED

REFERENCES

[1] R.M. McClure, "TMG - A syntax directed compiler", Proc. ACM 20th National Conference, pp. 262 - 274, 1965.

[2] J.E. Hopcroft and J.D. Ullman, Formal Languages and Their Relation to Automata, Addison-Wesley, Reading, Mass., 1969.

[3] W.J. Chandler, "Abstract families of deterministic languages", Proc. ACM Symp. on Theory of Computing, pp. 21-30, Marina del Rey, California, May, 1969.

[4] P.C. Fisher, "On computability by certain classes of restricted Turing machines", Proceedings Fourth Annual Symposium on Switching Circuit Theory and Logical Design, Chicago, Ill., pp. 23-32, 1963.

[5] S. Ginsburg and S.A. Greibach, "Deterministic context-free languages", Inf. and Control, Vol. 9, pp. 620-648, 1966.

[6] L.H. Haines, Generation and recognition of formal languages, Doctoral Thesis, MIT, Cambridge, Massachusetts, 1965.

[7] J.E. Hopcroft and J.D. Ullman, "An approach to a unified theory of automata", Bell System Tech. Journal, vol. 46, pp. 1763-1829, 1967.

[8] D.E. Knuth, "On the translation of languages from left to right", Inf. and Control, vol. 8, pp. 607-639, 1965.

[9] S.Y. Kuroda, "Classes of languages and linear-bounded automata", Inf. and Control, vol. 7, pp. 207-223, 1964.

[10] P.S. Landweber, "Three theorems on phrase structure grammars of type 1", Inf. and Control, vol. 6, pp. 131-136, 1963.

[11] J. Earley, An efficient context-free parsing algorithm, Doctoral Thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1965.

[12] S. Ginsburg and H.G. Rice, "Two families of languages related to ALGOL", JACM, vol. 9, pp. 350-371, 1962.

[13] J.C. Shepherdson and M.E. Sturgis, "Computability of recursive functions", JACM, vol. 10, pp. 217-255, 1963.

[14] S. Ginsburg, The Mathematical Theory of Context-Free Languages, McGraw-Hill, New York.

[15] D.E. Knuth, Lecture notes, International Summer School on Computer Programming, Copenhagen, Denmark, (August 1967).

THE TMG RECOGNITION SCHEMA

Abstract

In [1] McClure describes TMG, a compiler writing system. Although TMG is a syntax directed system, its recognition schema differs in some respects from previously known schemas. The formalization of this recognition schema and the study of its properties form the goal of this work.

First, the "TMG Recognition Schema", shortly TS, is defined; also, an automaton (TSA) corresponding to a given TS is described and it is shown that the TSA accepts exactly the language "recognized" by the TS.

A subclass of TS, the so called "well-formed TS", is then defined and it is shown to have the property: the class of languages it "recognizes" includes all deterministic context-free languages.

A given string can be recognized by a TS or it can be rejected, in which case we have a "failure". The various types of failures are described and the relations between them are investigated. Some closure properties and decidability results follow.

Next, the time complexity of the languages "recognized" by the TS, the TSL, is considered; it is shown that the TSL can be recognized in linear time by a suitable algorithm.

The question how the TS relates to the phrase structure grammars is also investigated; it is shown that the TSL are context sensitive, they include some non-cfl's and is conjectured that there are cfl's which are not TSL. It is also shown that the TSL over a one letter alphabet are not regular.

Another question of interest is whether the TS, our model of a recognition schema, can be improved. Several extensions and generalization of the TS are studied and it is shown that they compare favorably with the original model. One of these extensions, the so called "generalized TS" (gTS), has the following property: there exists a class of balloon automata which accepts exactly the gTSL, i.e. the class of languages recognized by the gTS.