INFORMATION AND CONTROL 23, 1–34 (1973)

Parsing Algorithms with Backtrack*

ALEXANDER BIRMAN

IBM Thomas J. Watson Research Center,
Yorktown Heights, New York 10598

AND

JEFFREY D. ULLMAN

Princeton University, Princeton, New Jersey

Two classes of restricted top-down parsing algorithms modeling "recursive descent" are considered. We show that the smaller class recognizes all deterministic context free languages, and that both classes can be simulated in linear time on a random access machine. Certain generalizations of these parsing algorithms are shown equivalent to the larger class. Finally, it is shown that the larger class has the property that loops and other "failures" can always be eliminated.

1. INTRODUCTION

Many compilers or compiler writing systems use a parsing strategy called "recursive descent." At least two similar strategies can be identified. One is used in the TMG compiler-compiler [McClure, 1965], and we call it the "TMG recognition scheme" (TS for short). A second which is used in the META compiler-compiler [Schorre, 1964] among others is modeled in [Knuth, 1971]. We refer to this scheme as "generalized TS" (gTS for short).

Both schemes analyze strings over some input alphabet by a program which consists of procedures calling one another recursively. Each procedure attempts to recognize a certain set of strings and returns with the outcome "success" if it finds one and "failure" otherwise. If it succeeds, the input head is moved over the input portion as recognized.

* Work supported by NSF Grant GJ-465.

Several observations can be made regarding these schemes:

(a) As recursive descent parsing algorithms involve backtracking, care must be exercised when designing the parser, else the backtracking would cause the time required for parsing to grow exponentially with the input length. We give an "Early-type" tabular parsing method (see [Earley, 1970], [Aho and Ullman, 1972]) which will find a parse in time proportional to the input length for any input recognized by a TS or gTS.

(b) Both the TS and gTS can define noncontext free languages. Thus, it is interesting to investigate the class of languages defined by these schemes. In particular, we show that all deterministic context free languages can be recognized in either TS or gTS. Coupled with the linear recognition time result mentioned above, we thus have one of the largest subclasses of the context free languages known to be linear time recognizable.

In Section 2 we define the "TMG recognition schema" (TS) and we study the relation between the "TS languages" and other known classes of languages. Then in Section 3 we describe the so called "failure types" and prove some closure and decidability results of the TS languages.

In Section 4 a generalized model, the gTS, is defined and the time complexity of recognition for TS and gTS languages is studied. In Section 5 the concept of a reduced gTS is investigated together with the "multiple failure schema" (a model shown to be equivalent to the gTS); it is shown that every gTS is equivalent to a reduced gTS.

2. The TMG Recognition Schema

Definition. A TMG *Recognition Schema* (TS) R is a 5 tuple $R = (V, \Sigma, P, S, \$)$ in which V is a finite set of *variables* (the recursive procedures), Σ is a finite set of *terminal symbols*, $V \cap E = \varphi$, S is an element of V, and \$ is a symbol of Σ, called the *endmarker*. P is a finite set of *rules* of the form

(a) $A \rightarrow BC/D$, A, B, C, D in V;

(b) $A \rightarrow a$, a in $\Sigma \cup \{\epsilon, f\}$, f a metasymbol assumed not to appear elsewhere, and ϵ the null string. For any variable A there is at most one rule with A on the left-hand side. (There may be none.)

Intuitively, the rule $A \rightarrow a$ means "When A is called, see if symbol a is under the input pointer. If so, move the input pointer right and succeed, otherwise fail." The rule $A \rightarrow \epsilon$ means that A always succeeds and $A \rightarrow f$ means that A always fails.

The rule $A \to BC/D$ means that when A is called, it calls B. If B succeeds and recognizes some input, C is called with the input pointer where left by B. If C succeeds, A succeeds and leaves the input pointer where C left it. If either B or C fail, A retracts the input pointer to the position it held when A was called. A then calls D and succeeds or fails as D does.

Define the set of relations, $\Rightarrow_R^n$, from V to $\Sigma^* \uparrow \Sigma^* \times \{0, 1\}$ as follows[1]:

1. If $A \to a$ is in P, a in Σ, then $A \Rightarrow_R^1 (a \uparrow x, 0)$ for all x in Σ^*, and $A \Rightarrow_R^1 (\uparrow bx, 1)$ for all x in Σ^*, b in Σ, $b \neq a$.
2. If $A \to \epsilon$ is in P, then $A \Rightarrow_R^1 (\uparrow x, 0)$ for all x in Σ^*.
3. If $A \to f$ is in P then $A \Rightarrow_R^1 (\uparrow x, 1)$ for all x in Σ^*.
4. Let $A \to BC/D$ be in P. For each x_1, x_2, x_3, and x_4 in Σ^*, if

(a) $B \Rightarrow_R^l (x_1 \uparrow x_2x_3, 0)$, $C \Rightarrow_R^m (x_2 \uparrow x_3, 0)$, then $A \Rightarrow_R^k (x_1x_2 \uparrow x_3, 0)$, $k = l + m + 1$;

(b) $B \Rightarrow_R^l (x_1 \uparrow x_2, 0)$, $C \Rightarrow_R^m (\uparrow x_2, 1)$, $D \Rightarrow_R^p (x_3 \uparrow x_4, 0)$, and $x_1x_2 = x_3x_4$, then $A \Rightarrow_R^k (x_3 \uparrow x_4, 0)$, $k = l + m + p + 1$;

(c) $B \Rightarrow_R^l (\uparrow x_1x_2, 1)$, $D \Rightarrow_R^m (x_1 \uparrow x_2, 0)$ then $A \Rightarrow_R^k (x_1 \uparrow x_2, 0)$, $k = l + m + 1$;

(d) $B \Rightarrow_R^l (x_1 \uparrow x_2, 0)$, $C \Rightarrow_R^m (\uparrow x_2, 1)$, $D \Rightarrow_R^p (\uparrow x_1x_2, 1)$, then $A \Rightarrow_R^k (\uparrow x_1x_2, 1)$, $k = l + m + p + 1$;

(e) $B \Rightarrow_R^l (\uparrow x_1, 1)$ and $D \Rightarrow_R^m (\uparrow x_1, 1)$, then $A \Rightarrow_R^k (\uparrow x_1, 1)$, $k = l + m + 1$.

If $A \Rightarrow_R^n (x \uparrow y, i)$ for some n, then we write $A \Rightarrow_R (x \uparrow y, i)$. We call n the number of steps in the derivation. The language recognized by R is $T(R) = \{x \mid x \text{ in } (\Sigma - \{\$\})^*, S \Rightarrow_R (x\$ \uparrow, 0)\}$.

If $A \Rightarrow_R (x \uparrow y, i)$, for some xy in Σ^*, we say that A *derives* $(x \uparrow y, i)$. If $i = 0$ then A *succeeds* on xy; if $i = 1$ we have a *recognition failure* (or simply *failure*).

The form of the rules given here is actually a "Chomsky normal form" of the rules used in the TMG language [McClure, 1965]. We find it simpler to make the definitions we have made and then define an extended notation that corresponds more closely that that of [McClure, 1965].

DEFINITION. An *extended* TMG rule is a string of the form $A \to \alpha_1/\alpha_2/\cdots/\alpha_n$, where $n \geqslant 1$, and α_i is in $(V \cup \Sigma)^* \cup \{f\}$ for $1 \leqslant i \leqslant n$. We convert this extended rule to a set of rules of the original form as follows.

[1] Intuitively, the relation $A \Rightarrow_R^n (x \uparrow y, i)$ means: If A is called on string xy, it will, after n steps, return outcome i success = 0 or failure = 1 and the input pointer positioned as indicated by the special symbol $\uparrow$.

(i) Replace each instance of a symbol a in $\Sigma \cup \{f\}$ appearing in $\alpha_1\alpha_2, \ldots, \alpha_n$ by X_a, a new variable symbol. (We retain the names α_i for the modified strings.) Add rule $X_a \to a$ for each such a.

(ii) If $n \geqslant 2$, replace the rule $A \to \alpha_1/\cdots/\alpha_n$ by $A \to \alpha_1/B_1$, $B_1 \to \alpha_2/B_2, \ldots, B_{n-2} \to \alpha_{n-1}/B_{n-1}$, $B_{n-1} \to \alpha_n/B_n$, $B_n \to f$, where $B_1, \ldots, B_n$ are new symbols.

(iii) For each rule $A \to \alpha/B$ generated in (ii), where $|\alpha| > 2$,[2] let $\alpha = C_1 \cdots C_m$, each C_i a variable symbol. Replace this rule by $A \to C_1D_1/B$, $D_1 \to C_2D_2/F, \ldots, D_{m-3} \to C_{m-2}D_{m-2}/F$, $D_{m-2} \to C_{m-1}C_m/F$ and $F \to f$. If $A \to C/B$ is generated in (ii), where C is a variable, replace the rule by $A \to CE/b$ and $E \to \epsilon$. Replace $A \to \epsilon/B$ generated in (ii) by $A \to \epsilon$.

EXAMPLE. The rule $A \to aB/CDE/F$, where a is a terminal and the other symbols variables, is replaced by $A \to X_aB/CDE/F$ and $X_a \to a$ in step (i). In step (ii), the first of these is replaced by $A \to X_aB/G$, $G \to CDE/H$, $H \to F/I$, and $I \to f$. In step (iii), we replace $G \to CDE/H$ by $G \to CJ/H$, $J \to DE/K$ and $K \to f$. We replace $H \to F/I$ by $H \to FL/I$ and $L \to \epsilon$.

Informally, the rule $A \to \alpha_1/\cdots/\alpha_n$ means that when A is called, try to recognize all of the characters of α_1, variable and terminal, in order on the input. If not, retract the input pointer and try those of α_2. Continue in this way until all of α_i is found for some i, and succeed. If no α_i completely succeeds, then A fails.

The TS can be thought of as a program for an automaton, and for this purpose we define the "TS-automaton."

DEFINITION. Let $R = (V, \Sigma, P, S, \$)$ be a TS. The *tape alphabet* of the TS-*automaton* $A(R)$ is $\Gamma = V \cup \{X_1X_2/X_3 \mid X_i \text{ in } V \text{ or } X_i = \bar{A} \text{ and } A \text{ in } V, i = 1, 2, 3\}$. The *internal states* of $A(R)$ are $\{s, f\}$. A *configuration* of $A(R)$ is a 3-tuple $(q, x_1 \vdash x_2, \omega)$, where q is in $\{s, f\}$, x_1x_2 in $(\Sigma - \{\$\})^*\$$. $\vdash$ 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. Informally, ω is a list of goals (variables) with information as to what part of the rule for each goal has been found and whence the input head should go if the goal fails.

We define the relation $\vdash_R$ between two configurations α, β, and we write $\alpha \vdash_R \beta$, as follows: assume

$$\alpha = (q, x_1 \vdash x_2, \omega), \omega = \gamma(X, i), \beta = (q', x_1' \vdash x_2', \omega'), x_1x_2 = x_1'x_2',$$

for some γ in $(\Gamma \times N)^*$, X in Γ, i in N.

[2] $|\alpha|$ is the length of α.

(i) Let $q = s$, $X = A$, A in V, $A \to BC/D$ in P; then $q' = s$, $x_1' = x_1$ and $\omega' = \gamma(\bar{B}C/D, i)(B, i)$. That is, if goal A is to be found we start by looking for the first part of its rule. (The bar over the B indicates that we should look for B.)

(ii) Let $q = s$, $X = A$, A in V, $A \to a$ in P, a in Σ. If $x_2 = ax_3$ we have $x_1' = x_1a$, $x_2' = x_3$, $q' = s$, and $\omega' = \gamma$. If $x_2 = bx_3$, b in Σ, $b \neq a$, we have $x_1' = x_1$, $q' = f$, and $\omega' = \gamma$. (If a goal is met with a single symbol, we succeed or fail as that symbol is or is not found and complete the goal.)

(iii) If $q = s$, $X = A$, A in V, $A \to \epsilon$ in P, then $x_1' = x_1$, $q' = s$, and $\omega' = \gamma$.

(iv) If $q = s$, $X = A$, A in V, $A \to f$ in P, then $x_1' = x_1$, $q' = f$, and $\omega' = \gamma$.

(v) Let $X = \bar{A}B/C$, A, B, C in V; if $q = s$ then $q' = s$, $x_1' = x_1$, and $\omega' = \gamma(A\bar{B}/C, i)(B, |x_1|)$; if $q = f$ then $q' = s$, $|x_1'| = i$, and $\omega' = \gamma(AB/\bar{C}, i)(C, i)$.

(vi) Let $X = A\bar{B}/C$; if $q = s$ then $q' = s$, $x_1' = x_1$, and $\omega' = \gamma$. If $q = f$ then $q' = s$, $|x_1'| = i$, $\omega' = \gamma(AB/\bar{C}, i)(C, i)$.

(vii) Let $X = AB/\bar{C}$; if $q = s$ then $q' = s$, $x_1' = x_1$, and $\omega' = \gamma$. If $q = f$ then $q' = f$, $|x_1'| = i$, $\omega' = \gamma$.

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

The *language accepted by* $A(R)$ is $\{w \mid w \text{ in } (\Sigma - \{\$\})^*, (s, \vdash w\$, (S, 0)) \overset{*}{\vdash}_R (s, w\$ \vdash, \epsilon)\}$.

The following result of [Birman, 1970] will be given without proof.

THEOREM 2.1. *The language accepted by* $A(R)$ *is* $T(R)$.

EXAMPLE. Consider the language $L = (ab)^*$. A TS that recognizes this language is $R = (V, \Sigma, P, S, \$)$, where $V = \{S, A, B, D, E, F\}$, $\Sigma = \{a, b, \$\}$, $P = \{S \to DS/E, E \to \$, D \to AB/F, A \to a, B \to b, F \to f\}$.

Let us illustrate how the automaton $A(R)$ accepts an input string, say $ab\$$, by a sequence of configurations:

(i) $(s, \vdash ab\$, (S, 0))$, the initial configuration in which the goal is S. In case of failure, backtracking takes place to position 0, that is the present position of the input pointer;

(ii) $(s, \vdash ab\$, (\bar{D}S/E, 0)(D, 0))$, the new goal is D;

(iii) $(s, \vdash ab\$, (\bar{D}S/E, 0)(\bar{A}B/F, 0)(A, 0))$;

(vi) $(s, a \vdash b\$, (\bar{D}S/E, 0)(\bar{A}B/F, 0))$, the input pointer moves to the right, recognizing symbol a;

(v) $(s, a \vdash b\$, (\bar{D}S/E, 0)(A\bar{B}/F, 0)(B, 1))$;

(vi) $(s, ab \vdash \$, (\bar{D}S/E, 0)(A\bar{B}/F, 0))$;

(vii) $(s, ab \vdash \$, (\bar{D}S/E, 0))$;

(viii) $(s, ab \vdash \$, (D\bar{S}/E, 0)(S, 2))$;

(ix) $(s, ab \vdash \$, (D\bar{S}/E, 0)(\bar{D}S/E, 2)(D, 2))$;

(x) $(s, ab \vdash \$, (D\bar{S}/E, 0)(\bar{D}S/E, 2)(\bar{A}B/F, 2)(A, 2))$;

(xi) $(f, ab \vdash \$, (D\bar{S}/E, 0)(\bar{D}S/E, 2)(\bar{A}B/F, 2))$;

(xii) $(s, ab \vdash \$, (D\bar{S}/E, 0)(\bar{D}S/E, 2)(AB/\bar{F}, 2)(F, 2))$;

(xiii) $(f, ab \vdash \$, (D\bar{S}/E, 0)(\bar{D}S/E, 2)(AB/\bar{F}, 2))$;

(xiv) $(f, ab \vdash \$, (D\bar{S}/E,)(\bar{D}S/E, 2))$;

(xv) $(s, ab \vdash \$, (D\bar{S}/E, 0)(DS/\bar{E}, 2)(E, 2))$;

(xvi) $(s, ab\ \$ \vdash, (D\bar{S}/E,)(DS/\bar{E}, 2))$;

(xvii) $(s, ab\$ \vdash, (D\bar{S}/E, 0))$;

(xviii) $(s, ab\$ \vdash, \epsilon)$.

Next 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.

DEFINITION. A *well-formed* TS (or shortly wfTS) $R = (V, \Sigma, P, S, \$)$ is a TS with the property for all x in $(\Sigma - \{\$\})^*$ either $S \Rightarrow_R (x\$ \vdash, 0)$ or $S \Rightarrow_R (\vdash x\$, 1)$.

We first give the definition of a DPDA (the notation used is similar to the one in [Ginsburg, 1966] or [Aho and Ullman, 1972].

DEFINITION. A deterministic pushdown automaton (DPDA) is a 7-tuple $Q = (K, \Sigma, \Gamma, \delta, Z_0, q_0, F)$, 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 $(K \times \Gamma^*) \cup \{\phi\}$ such that for each q in K and Z in Γ, if $\delta(q, \epsilon, Z) \neq \phi$ then $\delta(q, a, Z) = \phi$ for all a in Σ;

Z_0 is an element of Γ;

q_0 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:

(i) For a in $\Sigma \cup \{\epsilon\}$, Z in Γ, if $\delta(q, a, Z) = (p, \gamma)$ then we write $(q, aw, Z\alpha) \vdash_Q (p, w, \gamma\alpha)$.

(ii) For α and β in Γ^* and x_i in $\Sigma \cup \{\epsilon\}$, $1 \leqslant i \leqslant k$, we write

$$(q, x_1 \cdots x_k w, \alpha) \vdash^*_Q (p, w, \beta)$$

if there exist $q = q_1, q_2, \ldots, q_{k+1} = p$ in K and $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_{k+1} = \beta$ in Γ^* such that $(q_i, x_i \cdots x_k w, \alpha_i) \vdash_Q (q_{i+1}, x_{i+1} \cdots x_k w, \alpha_{i+1})$ for $1 \leqslant i \leqslant k$.

A word w is *accepted* by Q if $(q_0, w, Z_0) \vdash^*_Q (q, \epsilon, \alpha)$ for some q in F and α in Γ^*. The set of all words accepted by Q is denoted by $T(Q)$.

THEOREM 2.2. *Let $Q_1 = (K_\Gamma, E, \Gamma, \delta_1, Z_0, q_0, F_1)$ be a* DPDA *and let the language accepted by Q_1 be $T(Q_1)$. There exists a* wfTS *such that $T(R) = T(Q_1)$.*

Proof. The language accepted by Q_1 by final state is $T(Q_1) = \{x \mid x \text{ in } \Sigma^* \text{ and } (q_0, x, Z_0) \vdash^*_Q (q, \epsilon, \gamma) \text{ for } \gamma \text{ in } \Gamma^* \text{ and } q \text{ in } F_1\}$. We can assume [Ginsburg and Greibach, 1966] that for all x in Σ^*, there are p in K_1 and α in Γ^* such that $(q_0, x, Z_0) \vdash^*_Q (p, \epsilon, \alpha)$, which means that the automaton Q_1 always scans the input string. It is elementary to construct from Q_1 a new DPDA, $Q = (K, \Sigma, \Gamma, \delta, Z_0, q_0, F)$ which will accept $x\$$, $\$$ a new symbol, if and only if x is in $T(Q_1)$, and in addition Q will always erase its storage tape before halting. Q has a single final state q_n. We can further assume without loss of generality that Q never increases the length of its list by more than one symbol at a move.

Now consider the TS, $R = (V, \Sigma \cup \{\$\}, P, [q_0 Z_0 q_n], \$)$ where V includes the sets $\{[q_i Z q_j] \mid q_i, q_j \text{ in } K, Z \text{ in } \Gamma\}$ and $\{[q_i Z q_j ; a] \mid q_i, q_j \text{ in } K, Z \text{ in } \Gamma, a \text{ in } \Sigma\}$; V also contains variables which are implicitly defined in the notation of the extended 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 (extended) rules.

(i) If $\delta(q, \epsilon, Z) = \phi$ (i.e., there are no ϵ-rules for q in K, Z in Γ) then P contains (P1): $[qZp] \rightarrow a_1[qZp; a_1]/a_2[qZp; a_2]/\cdots/a_m[qZp; a_m]$ for all p in K where $\Sigma \cup \{\$\} = \{a_1, a_2, \ldots, a_n\}$ (variable of the form $[qZp; a]$ are used to register the fact that symbol a has been recognized).

(ii) If $\delta(q, a, Z) = (p, XY)$, X, Y in Γ, a in $\Sigma \cup \{\$\}$, then (P2): $[qZr; a] \rightarrow [pXq_1][q_1 Y r]/\ldots/[pXq_n][q_n Y r]$, for all r in K, where $K = \{q_1, \ldots, q_n\}$.

(iii) If $\delta(q, a, Z) = (p, X)$, X in Γ, then ($P3$): $[qZr; a] \to [pXr]$ for all r in K.

(iv) If $\delta(q, a, Z) = (p, \epsilon)$, then ($P4$): $[qZp; a] \to \epsilon$, and $[qZr; a] \to f$, for all r in K, $r \neq p$.

(v) If $\delta(q, \epsilon, Z) = (p, XY)$, then ($P5$): $[qZr] \to [pXq_1][q_1Yr]/\cdots/[pXq_n][q_nYr]$ for all r in K.

(vi) If $\delta(q, \epsilon, Z) = (p, X)$, then ($P6$): $[qZr] \to [pXr]$, for all r in K.

(vii) If $\delta(q, \epsilon, Z) = (p, \epsilon)$, then ($P7$): $[qZp] \to \epsilon$, and $[qZr] \to f$, for all r in K, $r \neq p$.

Part one. We show that if $(q, x, Z) \vdash_Q^* (p, \epsilon, \epsilon)$, x in $\Sigma^*\$$, Z in Γ, then $[qZp] \Rightarrow_R (x \vdash, 0)$ and $[qZs] \Rightarrow_R (\vdash x, 1)$ 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. $x = a$, a in $\Sigma \cup \{\$\}$, and $\delta(q, a, Z) = (p, \epsilon)$. Then $[qZp; a] \to \epsilon$, $[qZs; a] \to f$ for all $s \neq p$ and

$$[qZp] \to a_1[qZp; a_1]/\cdots/a_m[qZp; a_m].$$

It follows that $[qZp; a] \Rightarrow_R (\vdash, 0)$ and $[qZp] \Rightarrow_R (a \vdash, 0)$. (The definition of the extended rules is necessary to verify these contentions.) Also in P we have $[qZs] \to a_1[qZs; a_1]/\cdots/a_m[qZs; a_m]$. It follows that $[qZs; a] \Rightarrow_R (\vdash, 1)$ for $s \neq p$, and hence $[qZs] \Rightarrow (\vdash a, 1)$.

Case 2. $x = \epsilon$ and $\delta(q, \epsilon, Z) = (p, \epsilon)$. $[qZp] \to \epsilon$, $[qZs] \to f$ for $s = p$, and hence $[qZp] \Rightarrow_R (\vdash x, 0)[qZs \Rightarrow_R (\vdash, 1)$ for all $s = p$.

Induction step case 1. The first move from configuration (q, x, Z) is not an ϵ-move. Suppose $x = ay$, a in $\Sigma \cup \{\$\}$ and $\delta(q, aZ) = (p', XY)$, X, Y in Γ, p' in K. Then $(q, ay, Z) \vdash_Q^* (p', y, XY)$. Let p'' be a state for which $(p', y_1y_2, XY) \vdash_Q^* (p'', y_2, Y) \vdash_Q^* (p, \epsilon, \epsilon)$ such that $y_1y_2 = y$ and (p'', y_2, Y) is the first configuration for which X and its descendants have been erased. By induction $[p'Xp''] \Rightarrow_R (y_1 \vdash y_2, 0)$ and $[p''Yp] \Rightarrow_R (y_2 \vdash, 0)$; also $[p'Xs] \Rightarrow_R (\vdash y_1y_2, 1)$, $s \neq p''$ and $[p''Ys] \Rightarrow_R (\vdash y_2, 1)$, $s \neq p$. We have the rule $[qZp; a] \to [p'Xq_1][q_1Yp]/\cdots/[p; Xq_n][q_nYp]$. Let k be such that $p'' = q_k$. Then $[p'Xq_j] \Rightarrow_R (\vdash y_1y_2, 1)$ for all $j \neq k$.

We also have $[p''Yp] \Rightarrow_R (y_2 \vdash, 0)$, and together with $[p'Xp''] \Rightarrow_R (y_1 \vdash y_2, 0)$ we get $[qZp; a] \Rightarrow_R (y_1y_2 \vdash, 0)$. Finally the rule for $[qZp]$,

$$[qZp] \to a_1[qZp; a_1]/\cdots/a_m[qZp; a_m]$$

gives us $[qZp] \Rightarrow_R (ay_1y_2 \vdash, 0)$.

Consider now the rule for $[qZs; a]$, $s \neq p$,

$$[qZs; a] \to [p'Xq_1][q_1Ys]/\cdots/[p'Xq_n][q_nYs].$$

We know already that $[p'Xq_j] \Rightarrow (\vdash y_1y_2, 1)$ for $j \neq k$, $p'' = q_k$, and $[p'Xq_k] \Rightarrow_R (y_1 \vdash y_2, 0)$. We also have $[q_kYs] \Rightarrow_R (\vdash y_2, 1)$, $s \neq p$, which implies $[qZs; a] \Rightarrow_R (\vdash y_1y_2, 1)$ and finally $[qZs] \Rightarrow_R (\vdash ay_1y_2, 1)$ for all $s \neq p$.

Next, assume $\delta(q, a, Z) = (p', X)$. Then $[qZp; a] \to [p'Xp]$ and by induction $[p'Xp] \Rightarrow_R (y \vdash, 0)$, hence $[qZp] \Rightarrow_R (ay \vdash, 0)$; $[qZs; a] \to [p'Xs]$ and $[p'Xs] \Rightarrow_R (\vdash y, 1)$, all $s \neq p$, hence $[qZs] \Rightarrow_R (\vdash ay, 1)$ for all $s \neq p$. If $\delta(q, a, Z) = (p', \epsilon)$ then we have the same behavior as for $n' = 1$, case 1.

Case 2. The first move is an ϵ-move. Suppose $\delta(q, \epsilon, Z) = (p', XY)$. Then let p'' be a state for which $(p', x_1x_2, XY) \overset{*}{\vdash}_Q (p'', x_2, Y) \overset{*}{\vdash}_Q (p, \epsilon, \epsilon)$ such that $x_1x_2 = x$ and (p'', x_2, Y) is the first configuration for which X and its descendants have been erased. By induction, $[p'Xp''] \Rightarrow_R (x_1 \vdash x_2, 0)$ and $[p'Xs] \Rightarrow_R (\vdash x_1x_2, 1)$, all $s \neq p''$; $[p''Yp] \Rightarrow_R (x_2 \vdash, 0)$ and $[p''Ys] \Rightarrow_R (\vdash x_2, 1)$, all $s \neq p$. The rule for $[qZp]$ is

$$[qZp] \to [p'Xq_1][q_1Yp]/\cdots/[p'Xq_n][q_nYp];$$

let k be such that $q_k = p''$. Then $[q'Xq_k] \Rightarrow_R (x_1 \vdash x_2, 0)$, $[q_kYp] \Rightarrow_R (x_2 \vdash, 0)$ and from above follows the desired result. The cases in which the pushdown symbol is replaced by one or zero symbols are treated analogously.

Part Two. We show that if $[qZp] \Rightarrow_R^{n'} (x \vdash, 0)$, x in $(\Sigma \cup \{\$\})^*$, p, q in K, Z in Γ, then $(q, x, Z) \overset{*}{\vdash}_Q (p, \epsilon, \epsilon)$.

The proof is by induction on n'. The details will be omitted here.

Part Three. We have to show that R is a wfTS, that is for all x in Σ^* either $[q_0Z_0q_n] \Rightarrow_R (x\$ \vdash, 0)$ or $[q_0Z_0q_n] \Rightarrow_R (\vdash x\$, 1)$. By the construction of Q we must have for all x in Σ^*, $(q_0, x\$, Z_0) \overset{*}{\vdash}_Q (q_n, \epsilon, \epsilon)$ if $x\$$ is accepted by Q, or $(q_0, x\$, Z_0) \overset{*}{\vdash}_Q (q, \epsilon, \epsilon)$ for $q \neq q_n$ if $x\$$ is not accepted. The result follows from part one.

Next we will show that the TSL are accepted by deterministic linear bounded automata [Myhill, 1960], and that they include some non-cfl's. Also, it will be shown that the TSL over a one letter alphabet are not all regular.

THEOREM 2.3. *Every* TS *language is accepted by a deterministic linear bounded automaton* (DLBA).

Proof. We make the following observations about the TS automaton.

(i) If a TS-automaton $A(R)$ has list $(X_1, i_1)(X_2, i_2) \cdots (X_m, i_m)$ at some step in its computation, then $i_1, i_2, \ldots, i_m$ is a nondecreasing sequence.

(ii) If R has v variables, and more than $3v + 1$ consecutive elements of the sequence $i_1, \ldots, i_m$ are the same, then $A(R)$ is in a loop and will not accept its input.

We construct a DLBA, M, to simulate $A(R)$. If at some step, $A(R)$ has list $(X_1, i_1) \cdots (X_m, i_m)$, then M will store on its jth cell from the left the symbols $X_k, X_{k+1}, \ldots, X_l$, if $i_k, i_{k+1}, \ldots, i_l$ are exactly these i's equal to j.

By (ii), this simulation can be done with a finite-sized tape alphabet. Details of the method of acceptance and loop detection by M are omitted.

It will be shown in Section 4 that any TS language can be recognized in linear time on a suitable random access machine using Algorithm 4.1. 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 languages.

It is easy to see that there are TS languages which are not context free. For example, consider the deterministic cfl, $L_1 = \{a^n b^n a^m \mid n, m \geqslant 1\}$ and $L_2 = \{a^m b^n a^n \mid n, m \geqslant 1\}$. Their intersection $L_3 = \{a^n b^n c^n \mid n \geqslant 1\}$ is not a cfl but according to Theorems 2.2 and 3.4 (closure under intersection), there is a TS R such that $T(R) = L_3$.

Next we will show that the TSL over a one letter alphabet are not regular by producing such a nonregular language. Consider $R = (V, \Sigma, P, A_1, S)$, where $V = \{A_i \mid 1 \leqslant i \leqslant 6\}, \Sigma = \{a, \$\}, P = \{A_1 \to A_2A_6, A_2 \to A_4A_5/A_3, A_3 \to \epsilon, A_4 \to A_5A_2, \to a, A_6 \to \$\}$.

THEOREM 2.4. *Let R be the* TS *described above. Then*

$$T(R) = \{a^{2^{(2n-1)}} \mid n = 0, 1, 2, \ldots\}.$$

For the proof of this theorem see [Birman, 1970]. The mechanism for an inductive proof of this contention will become clearer after Algorithm 4.1.

3. FAILURE TYPES AND ADDITIONAL PROPERTIES FOR TS

In this section 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 failure 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.

DEFINITION 3.1. Let $R = (V, \Sigma, P, S, \$)$ be a TS. For A in V, x in $(\Sigma - \{\$\})^*$:

(i) A has a *recognition failure* (or simply failure) *on* x if

$$(s, \vdash x\$, (A, 0)) \overset{*}{\vdash}_R (f, \vdash x\$, \epsilon).$$

(ii) A has a *subroutine failure on* x if either there is no rule for A in P or $(s, \vdash x\$, (A, 0)) \overset{*}{\vdash}_R (s, x_1 \vdash x_2, \gamma(B, n))$ for some γ in $(\Gamma \times N)^*$, n in N, B in V such that $x_1 x_2 = x\$$ and there is no rule for B in P.

(iii) A has an *end failure on* x if $(s, \vdash x\$, (A, 0)) \overset{*}{\vdash}_R (s, x\$ \vdash, \gamma(B, n))$ for some γ in $(\Gamma \times N)^*$, n in N, B in V, and the rule for B in P is $B \to a$, for some a in Σ. (In case (ii) above, $A(R)$ halts because no rule is available for some variable. (In case (iii), $A(R)$ halts because the read head "falls off" the input tape.)

(iv) R has a *partial-acceptance* (p-a) *failure on* x if

$$(s, \vdash x_1 x_2\$, (S, 0)) \overset{*}{\vdash}_R (s, x_1 \vdash x_2\$, \epsilon)$$

for some x_2 in Σ^*, $x = x_1 x_2$. (We will say equivalently that S has a p-a failure on x. The p-a failure reflects the fact that if S has outcome 0 but has not scanned the whole string, the string is rejected and it does not belong to $T(R)$).

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

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

THEOREM 3.1. *Let* $R = (V, \Sigma, P, S, \$)$ *be a* TS *and* x *in* $(\Sigma - \{\$\})^* - T(R)$. *If* S *does not have on* x *a subroutine, end,* p-a, *or loop failure, then* S *has a recognition failure on* x.

Proof. $A(R)$ in configuration $(s, \vdash x\$, (S, 0))$ can only make a finite number

of moves, since otherwise S would have a loop failure on x. There is thus a unique configuration $(q, x_1 \vdash x_2, \gamma)$ such that

$$x_1x_2 = x\$, (s, \vdash x\$, (S, 0)) \overset{*}{\vdash}_R (q, x_1 \vdash x_2, \gamma)$$

and $A(R)$ has no move in this configuration.

First, assume $\gamma \neq \epsilon$. 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 $B \to \epsilon$ or $B \to f$. If the rule is $B \to a$, for some a in Σ, a move is still possible unless $x_2 = \epsilon$, in which case we have an end failure. Therefore the assumption $\gamma \neq \epsilon$ leads to a contradiction.

Assume now $\gamma = \epsilon$. We cannot have $q = s$, since we would have a p-a failure. If $q = f$ then, according to the definition of $A(R)$, we have backtracking to position 0 (marked on the initial tape as $(s, 0)$), and therefore $x_1 = \epsilon$. We conclude that $(s, \vdash x\$, (S, 0)) \mid \overset{*}{\vdash}_Q (f, \vdash x\$, \epsilon)$ and we have a recognition failure.

Consider a variable A such that $A \Rightarrow_R (\vdash x, 0)$ for all x. This variable has the following property: If we start $A(R)$ in configuration $(s, \vdash x, (A, 0))$, for some x in Σ^*, the storage tape is erased and no input symbol is checked for a match. After $A(R)$ 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 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 $U(R)$, $V(R)$ defined below.

DEFINITION 3.2. Let R be a TS, $R = (V, \Sigma, P, S, \$)$. Construct $U(R) \subseteq V$ and $V(R) \subseteq V$ as follows:

(i) $U_0 = \{A \mid A \text{ in } V, A \to \epsilon \text{ in } P\}$, $V_0 = \{A \mid A \text{ in } V, A \to f \text{ in } P\}$;

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

(iii) 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 [Birman, 1970] gives a precise description to the mentioned property of the elements of $U(R)$, $V(R)$:

LEMMA 3.1.

(a) $(s, \vdash, (A, 0)) \vdash^*_R (s, \vdash, \epsilon)$ *if and only if* A *is in* $U(R)$.

(b) $(s, \vdash, (A, 0)) \vdash^*_R (f, \vdash, \epsilon)$ *if and only if* A *is in* $V(R)$.

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

THEOREM 3.2. *Given a* TS $R = (V, \Sigma, P, S, \$)$, *there is a* TS R' *which has no* p-a *failures and* $T(R) = T(R')$.

Proof. Consider the TS $R' = (V', \Sigma, P', \bar{S}, \$)$ where V' includes $\{A, \bar{A} \mid \text{all } A \text{ in } V\} \cup \{J\}$, where J is a new symbol, and other variables implied by the introduction of some extended rules. 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 $S \to AB/C$; 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 $\bar{S} \to A\bar{B}/\bar{C}$. 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:

(i) If $A \to \epsilon$ is in P, then P' contains $\bar{A} \to J$ and $A \to \epsilon$.

(ii) If $A \to a$ is in P, a in $\Sigma - \{\$\}$, then P' has extended rule $\bar{A} \to aJ$ and $A \to a$.

(iii) If $A \to \$$ is in P, then $\bar{A} \to \$$, $A \to \$$ are in P'.

(iv) If $A \to f$ is in P, then P' contains $\bar{A} \to f$, $A \to f$.

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

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

(vii) There is no rule for J.

The rest of the proof follows in three parts:

Part one. First we show that for all x_1x_2 in Σ^*, i in $\{0, 1\}$, $A \Rightarrow_R (x_1 \vdash x_2, i)$ if and only if $A \Rightarrow_{R'} (x_1 \vdash x_2, i)$. 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 $(\Sigma - \{\$\})^*$,

(a) $A \Rightarrow_R (x\$ \uparrow, 0)$ if and only if $\bar{A} \Rightarrow_{R'} (x\$ \uparrow, 0)$;

(b) $A \Rightarrow_R (\uparrow x\$, 1)$ if and only if $\bar{A} \Rightarrow_{R'} (\uparrow x\$, 1)$.

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

Base. (a) A one-step derivation implies $A \Rightarrow_R (\$ \uparrow, 0)$, and $A \to \$$ in P. We get $\bar{A} \to \$$ in P', $\bar{A} \Rightarrow_{R'} (\$ \uparrow, 0)$.

(b) Several cases arise. If $A \to f$ is in P, then $\bar{A} \to f$ is in P' and the theorem holds. If $A \to a$ is in P, a in $\Sigma - \{\$\}$, then $\bar{A} \to aJ$ is in P' and in R' we get $\bar{A} \Rightarrow_R (\uparrow x\$, 1)$. The case $A \to \$$ in P is easily verified.

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

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

1(a) $B \Rightarrow_R (x_1 \uparrow x_2\$, 0)$, $C \Rightarrow_R (x_2\$ \uparrow, 0)$ for some x_1 and x_2, $x_1x_2 = x$. Then by induction $\bar{C} \Rightarrow_{R'} (x_2\$ \uparrow, 0)$. Also, $\bar{A} \to B\bar{C}/\bar{D}$ is in P', $B \Rightarrow_{R'} (x_1 \uparrow x_2\$, 0)$, hence $\bar{A} \Rightarrow_{R'} (x\$ \uparrow, 0)$.

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

1(c) $B \Rightarrow_R (x_1 \uparrow x_2, 0)$, $C \Rightarrow_R (\uparrow x_2, 1)$, $D \Rightarrow_R (x\$ \uparrow, 0)$, where $x_1x_2 = x\$$. Again, $\bar{D} \Rightarrow_{R'} (x\$ \uparrow, 0)$.

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

Induction step (b): Given $A \Rightarrow_R (\uparrow x\$, 1)$:

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

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

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

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

"*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 $(\Sigma - \{\$\})^*$ do we have $\bar{A} \Rightarrow_{R'} (x_1 \vdash x_2\$, 0)$. The proof is by induction on the number of steps in the derivation in R'. For details see [Birman, 1970]. Q.E.D.

By using techniques similar to the ones in Theorem 3.2 it can be shown that in a TS the end failures can also be replaced by subroutine failures. (We mark variables which can be called after the endmarker is recognized and create a subroutine failure unless they are in $U(R)$ or $V(R)$).

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 $A \rightarrow AA/A$ 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 and loop failures.

Using the results and comments above we can prove the following theorems [Birman, 1970]:

THEOREM 3.3. *Every* TSL *is recognized by a* TS R *with only recognition and loop failures.*

THEOREM 3.4. *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. Assume $V_1 \cap V_2 = \phi$ and S, A, B, and J are new symbols. Merge the rules of R_1 and R_2 and add the extended rules $S \rightarrow AB/S_2$, $A \rightarrow S_1/J$, and $B \rightarrow f$. There is no rule for J. Thus, if the word x is in $T(R_1) \cap T(R_2)$, A will succeed on x, B will fail, of course, but S_2 succeeds, so S succeeds. If x is not in $T(R_1)$, A causes a subroutine failure, so S fails. If x is in $T(R_1)$ but not $T(R_2)$, S obviously fails.

THEOREM 3.5. *The complement of a* wfTS *language is a* TS *language.*

Proof. Let $R = (V, \Sigma, P, S, \$)$ be a wfTS, and $R' = (V', \Sigma, P', S', \$)$ wfTS accepting $(\Sigma - \{\$\})^*$. ($R'$ exists by Theorem 2.2). Assume $V \cap V' = \phi$ and S'' and A are new symbols. Merge P and P', adding the rule $S'' \rightarrow SA/S'$. There is no rule for A.

THEOREM 3.6. *Let* $R_i = (V_i, \Sigma, P_i, S_i, \$)$, $i = 1, 2$, *be* TS, *with* R_1 *a* wfTS. *There exists a* TS R *such that* $T(R) = T(R_1) \cup T(R_2)$.

Proof. Again assume $V_1 \cap V_2 = \phi$ and S is a new symbol. Merge P_1 and P_2, adding the extended rule $S \rightarrow S_1/S_2$.

COROLLARY 3.1. *The union of two* wfTS *languages is a* wfTS *language.*

COROLLARY 3.2. *The finite union of deterministic languages is a* wfTS *language.*

In the next theorem we show the Σ^*-problem for wfTS is undecidable.

THEOREM 3.7. *It is undecidable 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. Let L_1, L_2 be languages accepted by the arbitrary DPDA's A_1, A_2. We know [Ginsburg and Greibach, 1966] that we can construct two DPDA's, A_1', and A_2', accepting $\bar{L}_1$ and $\bar{L}_2$. By Corollary 3.2, 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 $L_1 \cap L_2 = \phi$. This implies we could solve the given instance of Q_1, which is a contradiction.

THEOREM 3.8. *It is undecidable whether the language recognized by an arbitrary* wfTS *is empty.*

Proof. Consider the question Q_1 as in Theorem 3.7 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 DPDA's: Let # be a new symbol; the input alphabet of a DPDA may be assumed to contain #, but 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. Let A_1' and A_2' be two DPDA's accepting $\Sigma^* - L_1$, $\Sigma^* - L_2$, respectively. Consider the wfTS R_1, R_2 recognizing $T(A_1')$ and $T(A_2')$, respectively, and wfTS R recognizing $T(R_1) \cup T(R_2)$ for which an effective construction was provided in Theorem 3.6. Also, it was shown that if we could decide whether $T(R) = \Sigma^*$ we would have a solution for the given instance of Q_1. Note that $T(R)$ is defined over $(\Sigma \cup \{\#\})^*$ but does not include any string that contains the symbol #.

Next we will describe the effective construction of a wfTS R_3 such that $T(R) = \Sigma^*$ if $T(R_3) = \phi$. (The existence of a TS R_3 is guaranteed by Theorem 3.5. However, we are about to show that a well-formed R_3 can be found).

Let $R = (V, \Sigma \cup \{\#, \$\}, P, S, \$)$. Consider

$$R_3 = (V_3, \Sigma \cup \{\#, \$\}, P_3, S_3, \#).$$

Let $\Sigma = \{a_i \mid 1 \leqslant i \leqslant m\}$. V_3 includes $V \cup \{S_3, S_4, X, A\}$, S_3, S_4, X, A new variables. P_3 includes P and also the set of rules

$$S_3 \to XS_4, X \to SX/\epsilon, S_4 \to A\#, A \to a_1A/a_2A/\cdots a_mA/\$,$$

where $\{a_1, \ldots, a_m\} = \Sigma$.

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

Next we show that if $T(R) \neq \Sigma^*$ then $T(R_3) \neq \phi$. Let x in Σ^* and $S \Rightarrow_R (\upharpoonright x\$, 1)$. Then in R_3, $X \Rightarrow_{R_3} (\upharpoonright x\$\#, 0)$, $A \Rightarrow_{R_3} (x\$ \upharpoonright \#, 0)$, $S_4 \Rightarrow_{R_3} (x\$\# \upharpoonright, 0)$, and finally $S_3 \Rightarrow_{R_3} (x\$\# \upharpoonright, 0)$ 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. Q.E.D.

COROLLARY 3.3. *It is indecidable whether the language recognized by an arbitrary* TS *is empty or whether it accepts* Σ^*.

By Theorem 3.3, we can eliminate all but loop and recognition failures from a TS. 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 nothing but recognition failures). We suspect that loop failures cannot be eliminated in a TS and the following result seems to support this view (the proof is found in [Birman, 1970]).

THEOREM 3.9. *It is undecidable whether an arbitrary* TS *is a* wfTS.

4. gTS: A Generalized Model

We observe that if A, with rule $A \Rightarrow BC/D$, is called in the operation of some TS, then D will be called independent of whether B or C failed. It would be a "nice" additional feature to allow A to call different routines, depending on whether B failed or B succeeded and C failed. We will introduce a new model which is equivalent in concept and parsing power to the parsing machine of [Knuth, 1971] and which can simulate such an action.

Definition 4.1. *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;
\$ in Σ is the *endmarker*;
P is a finite set of *rules* of the form (a) or (b):
(a). $A \to B(C, D)$ A, B, C and D in V,
(b). $A \to a$, a in $\Sigma \cup \{\epsilon, f\}$.

For any variable A there is at most one rule with A on the left-hand side.

Define the set of relations for each n in N, $A \Rightarrow_R^n (x \vdash y, i)$, x, y in Σ^*, i in $\{0, 1\}$, as follows:

(i) If $A \to a$ is in P, a in Σ, then $A \Rightarrow_R^1 (a \vdash x, 0)$ for all x in Σ^*, and $A \Rightarrow_R^1 (\vdash bx, 1)$ for all x in Σ^*, b in Σ, $b \neq a$.

(ii) If $A \to \epsilon$ is in P, then $A \Rightarrow_R^1 (\vdash x, 0)$ for all x in Σ^*.

(iii) If $A \to f$ is in P, then $A \Rightarrow_R^1 (\vdash x, 1)$ for all x in Σ^*.

(iv) Let $A \to B(C, D)$ be in P. For each x_1, x_2, and x_3 in Σ^*, if

(a) $B \Rightarrow_R^l (x_1 \vdash x_2x_3, 0)$, $C \Rightarrow_R^m (x_2 \vdash x_3, 0)$, then $A \Rightarrow_R^k (x_1x_2 \vdash x_3, 0)$, $k = l + m + 1$;

(b) $B \Rightarrow_R^l (x_1 \vdash x_2, 0)$, $C \Rightarrow_R^m (\vdash x_2, 1)$, then $A \Rightarrow_R^k (\vdash x_1x_2, 1)$, $k = l + m + 1$;

(c) $B \Rightarrow_R^l (\vdash x_1x_2, 1)$, $D \Rightarrow_R^m (x_1 \vdash x_2, 0)$, then $A \Rightarrow_R^k (x_1 \vdash x_2, 0)$ $k = l + m + 1$;

(d) $B \Rightarrow_R^l (\vdash x_1, 1)$, $D \Rightarrow_R^m (\vdash x_1, 1)$, then $A \Rightarrow_R^k (\vdash x_1, 1)$, $k = l + m + 1$.

If $A \Rightarrow_R^n (x \vdash y, i)$, x, y in Σ^*, in $\{0, 1\}$, we say that A *derives* $(x \vdash y, i)$ *in* n *steps*. We say A *derives* $(x \vdash y, i)$ if A derives $(x \vdash y, i)$ in r steps for some r,

and we write $A \Rightarrow_R (x \vdash y, i)$. (Equivalently, we will say A has *outcome* i on xy and it *recognizes* x.) The language recognized by R is $T(R) = \{x \mid x \text{ in } (\Sigma - \{\$\})^*,\ S \Rightarrow_R (x\$ \vdash, 0)\}$.

Remarks.

1. The difference between the TS and the gTS can be explained informally as follows: Suppose we have a rule $A \to B(C, D)$ in a given gTS and an input x in $(\Sigma - \{\$\})^*\$$; if A is called, A calls B and assume that 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 schemes.

2. 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 (except for obvious modifications in rules (i) and (v)–(vii) of the definition of $A(R)$, we will apply Definition 3.1 for various types of failures in the gTS.

3. The combination of rules $A \to X(D_1, Y)$, $X \to B(F, E)$, $F \to f$, $E \to \epsilon$, $Y \to Z(E, D_2)$, and $Z \to B(C, F)$ have the effect, when A is called, of calling B then C, and succeeding if both succeed. If B fails D_1 is called and if B succeeds but C fails, then D_2 is called. Note that X never uses any input and succeeds if and only if B fails.

THEOREM 4.1. *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 $V \cup \{X_A \mid \text{all } A \text{ in } V\} \cup \{E, F\}$, where E and F are new symbols. P' is formed

(i) if $A \to a$ is in P, a in $\Sigma \cup \{\epsilon, f\}$, then $A \to a$ is in P';

(ii) if $A \to BC/D$ is in P, A, B, C, and D in V, then P' contains $A \to X_A(E, D)$ and $X_A \to B(C, F)$;

(iii) $E \to \epsilon$ and $F \to f$ are in P'.

A straightforward induction on the number of steps on a derivation shows that $T(R) = T(R')$.

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

ALGORITHM 4.1. *Let* $R = (V, \Sigma, P, S, \$)$ *be a* gTS *and* $w = a_1 \cdots a_n$ *be in* $(\Sigma - \{\$\})^*\$$. *Let* $\# V = k$, *and construct a* $k \times (n+1)$ *matrix* $[t_{ij}]$. *We assume without loss of generality that* $V = \{A_1, \ldots, A_k\}$, *where* $S = A_1$. *The elements* t_{ij} *take on the integers and the symbol* f *as values. If* $t_{ij} = l$, *then we expect* $A_i \Rightarrow_R (a_j \cdots a_{l+j-1} \uparrow a_{l+j} \cdots a_n, 0)$. *If* $t_{ij} = f$, *then* $A_i \Rightarrow_R (\uparrow a_j \cdots a_n, 1)$. *We compute* t_{ij} *as follows*:

(i) *Let* $j = n + 1$.

(ii) *For each* i, $1 \leqslant i \leqslant k$, *if* $A_i \to f$ *is in* P, *set* $t_{ij} = f$. *If* $A_i \to \epsilon$ *is in* P, *set* $t_{ij} = 0$. *If* $A_i \to a_j$ *is in* P, *set* $t_{ij} = 1$, *and if* $A \to b$ *is in* P, $b \neq a_j$ *and* $j \neq n + 1$, *set* $t_{ij} = f$.

(iii) *Do the following for* $i = 1, 2, \ldots$ *until no changes to the* t_{ij}*'s occur in one cycle of values of* i (*from* 1 *to* k).

If the rule for A_i *is of the form* $A_i \to A_p(A_q, A_r)$, *and*:

(a) $t_{pj} = f$ *and* $t_{rj} = x$, *then set* $t_{ij} = x$, *whether* x *is an integer or* f;

(b) $t_{pj} = l_1$ *and* $t_{q(j+l_1)} = l_2$, *set* $t_{ij} = l_1 + l_2$;

(c) $t_{pj} = l_1$ *and* $t_{q(j+l_1)} = f$, *set* $t_{ij} = f$. *In all other cases do nothing for* t_{ij}.

(iv) *Decrease* j *by* 1 *and return to step* (ii), *unless* $j = 0$. *If* j *is now* 0, w *is in* $L(G)$ *if and only if* $t_{11} = n$.

THEOREM 4.2. *Algorithm* 4.1 *correctly determines if the input string* w *is in* $T(R)$.

Proof. We show that after execution of Algorithm 4.1 on $w = a_1 \cdots a_n$, $t_{ij} = f$ if and only if $A_i \Rightarrow_R (\uparrow a_j \cdots a_n, 1)$ and $t_{ij} = l$ if and only if $A_i \Rightarrow_R (a_j \cdots a_{j+l-1} \uparrow a_{j+l} \cdots a_n, 0)$. A straightforward induction on the order in which the t_{ij}'s are computed shows that whenever t_{ij} is given a value, that value is as stated above. Conversely, we show by induction on m that if $A_i \Rightarrow_R^m (\uparrow a_j \cdots a_n, 1)$ or $A_i \Rightarrow_R^m (a_j \cdots a_{j+l-1} \uparrow a_{j+l} \cdots a_n, 0)$, then t_{ij} is given the value f or l, respectively. The details are left to the reader.

THEOREM 4.3. *For each* gTS R *there is a constant* k *such that Algorithm* 4.1 *takes no more than* kn *suitably defined elementary steps of a random access machine on inputs of lengths* $n > 0$.

Proof. It is left to the reader to define the elementary steps reasonably. The crux of the proof is then to observe that in step (iii), at most k cycles are executed for given j.

The following result is similar to Theorem 2.3:

THEOREM 4.4. *For any* gTS R, *there exists a deterministic linear bounded automaton* M *such that the language accepted by* M *is* $T(R)$.

It is convenient to extend the rules of the gTS. Theorem 4.1 gives a definition of TS rules, and hence extended TS rules as gTS rules.

We can further accept the gTS rule $A \to \alpha(\beta, \gamma)$, where α, β, and γ are in $(V \cup \Sigma \cup \{f\})^*$ for some gTS $R = (V, \Sigma, P, S, \$)$, to stand for the rules $A \to B(C, D)$ and the extended TS rules $B \to \alpha$, $C \to \beta$ and $D \to \gamma$. It informally has the meaning: Look for all the symbols of α on the input and if they are found, look for those of β. If not all are found, backtrack and look for those of γ.

Another useful extension is the rule $A \to [B](C, D)$, which means, informally: see if B succeeds on the input. Backtrack in any event, but call C if B succeeds, D if it doesn't. Formally $A \to [B](C, D)$ stands for the set of rules $A \to X(D, C)$, $X \to B(F, E), F \to f$, and $E \to \epsilon$.

We take $A \to [\alpha](\beta, \gamma)$ to stand for $A \to [B](C, D), B \to \alpha$, $C \to \beta$, and $D \to \gamma$.

We prove the correctness of the informal interpretation of $A \to [B](C, D)$.

LEMMA 4.1. *Let* $R = (V, \Sigma, P, S, \$)$ *be a* gTS *and* $A \to X(D, C)$, $X \to B(F, E), F \to f$, *and* $E \to \epsilon$. *Then* $A \Rightarrow_R (x \vdash y, 0)$ *if and only if* $C \Rightarrow_R (x \vdash y, 0)$ *and for some prefix* z *of* xy, $B \Rightarrow_R (z \vdash, 0)$, *or* $D \Rightarrow_R (x \vdash y, 0)$ *and* $B \Rightarrow_R (\vdash xy, 1)$. $A \Rightarrow_R (\vdash x, 1)$ *if and only if* $C \Rightarrow_R (\vdash x, 1)$ *and* $B \Rightarrow_R (y \vdash z, 0)$, *where* $yz = x$, *or* $D \Rightarrow_R (\vdash x, 1)$ *and* $B \Rightarrow_R (\vdash x, 1)$.

Proof. Let $B \Rightarrow_R (u \vdash v, 0)$. Then $X \Rightarrow_R (\vdash w, 1)$, $w = uv$, and the outcome of A is the same as that of C. If $B \Rightarrow_R (\vdash w, 1)$, then $X \Rightarrow_R (\vdash w, 0)$ and the outcome of A is that of D. If B has no outcome on w, then X has no outcome on w and neither does A.

We claim that the language $L = \{a^{n^2} \mid n \geqslant 1\}$ is a gTS language. It is recognized by the gTS with extended rules:

$$S \to C(\epsilon, D),$$
$$A \to [S](f, BS),$$
$$B \to [S](\epsilon, aBa),$$
$$C \to a\$(\epsilon, aaaa\$),$$
$$D \to aaA.$$

It is conjectured that L is not a TSL, but a proof that the above gTS rules in fact recognizes L appears in [Birman, 1970]. Thus, we conjecture that the inclusion of Theorem 4.1 is proper.

An abstract family of deterministic languages (AFDL) is defined in [Chandler, 1969] as a family of languages closed under the operations of "marked union," "marked*" (marked Kleene closure), and inverse "marked gsm" mapping. These classes are shown in [Chandler, 1969] to be exactly those classes accepted by a class of one way deterministic automata (AFDA).

The gTS automaton is not an AFDA, since 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 distinct restriction in a gTS automaton is on the number of states, two, 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 software. It can be shown, however, that the gTSL form on AFDL. The following theorem [Birman and Ullman, 1970] will be given without proof.

THEOREM 4.5. *The* gTSL *form an* AFDL.

5. MULTIPLE-FAILURE SCHEMA

The multiple-failure schema or 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 cause backtracking. We shall explore the relation between these schemes and the gTS schemes in the current section.

DEFINITION 5.1. An (l, m)-TS R, $1 \leqslant l < m$, is a 6-tuple

$$R = (V, \Sigma, P, S, g, \$)$$

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 of Σ called endmarker;
P is a finite set of rules of the form (a), (b), or (c):

(a) $A \to (Q_1, Q_2, \ldots, Q_m)$, where there is a j, $1 \leqslant j \leqslant m$, such that $Q_j = \{\epsilon\}$ and $Q_i = \varnothing$ for all $i \neq j$.

(b) $A \to (Q_1, Q_2, \ldots, Q_m)$, A in V, for $1 \leqslant j \leqslant m$, $Q_j \subseteq \Sigma$ such that for all a in Σ, a belongs to exactly one Q_j.

(c) $A \to B(C_1, C_2, \ldots, C_m)$, A, B in V, C_j in V for $1 \leqslant j \leqslant m$.
For any A in V there is exactly one rule with A on the left-hand side.

g: $U_m \times U_m \to U_m$, where $U_m = \{1, 2, \ldots, m\}$. We define the set of relations, for each positive integer n, $A \Rightarrow_R^n (x \vdash y, i)$, x, y in Σ^*, i in U_m as follows:

(i) If $A \to (\phi, \ldots, \phi, \{\epsilon\}, \phi, \ldots, \phi)$ is in P and ϵ is in position j, then $A \Rightarrow_R^1 (\vdash x, j)$, for all x in Σ^*.

(ii) Let $A \to (Q_1, Q_2, \ldots, Q_m)$ be in P and let a be in Q_i. For all x in Σ^*, if $i \leqslant l$ then $A \Rightarrow_R^1 (a \vdash x, i)$, and if $i > l$ then $A \Rightarrow_R^1 (\vdash ax, i)$.

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

(a) $B \Rightarrow_R^{n_1} (x_1 \vdash x_2x_3, j)$, $C_j \Rightarrow_R^{n_2} (x_2 \vdash x_3, k)$, $g(j, k) = i$, and $i \leqslant l$, then $A \Rightarrow_R^n (x_1x_2 \vdash x_3, i)$ where $n = n_1 + n_2 + 1$.

(b) $B \Rightarrow_R^{n_1} (x_1 \vdash x_2x_3, j)$, $C_j \Rightarrow_R^{n_2} (x_2 \vdash x_3, k)$, $g(j, k) = i$, and $i > l$, then $A \Rightarrow_R^n (\vdash x_1x_2x_3, i)$, 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. 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 $i \leqslant l$ and therefore A will recognize the substring x_1x_2. In case (b), $i > l$ and we have backtracking.

If $A \Rightarrow_R^n (x \vdash y, i)$, x, y in Σ^*, i in U_m, we say A derives $(x \vdash y, i)$ *in n steps*. We say that A derives $(x \vdash y, i)$ in r steps for some r, and we write $A \Rightarrow_R (x \vdash y, i)$.

The language recognized by R is $T(R) = \{x \mid x \text{ in } (\Sigma - \{\$\})^*, S \Rightarrow_R (x\$ \vdash, 1)\}$. 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 5.2. Let $R = (V, \Sigma, P, S, g, \$)$ be an (l, m)-TS. The *tape alphabet* of $A(R)$ is $\Gamma = \{A^{(i)} \mid \text{all } A \text{ in } V, 0 \leqslant i \leqslant m\}$. (Whenever variable A is called in R, $A(R)$ will print on its tape $A^{(0)}$, the superscript (0) indicating that no processing has been done yet. However, if the rule for A is $A \to B(C_1, \ldots, C_m)$ and B "returns" with outcome i, then before C_i is called, the symbol $A^{(0)}$ on the tape is changed $A^{(i)}$, to indicate the result of the "call"

of B.) The set of *internal states* of $A(R)$ is $K = \{q_i \mid 0 \leqslant i \leqslant m\}$. (There exists a state for every possible outcome when the storage tape is "popped," and also a state q_0, which indicates a variable "call.")

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^*, $\uparrow$ is a metasymbol. (It 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.

We define the relation $\vdash_R$ between two configurations α, β, and we write $\alpha \vdash_R \beta$ 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 $(\Gamma \times N)^*$, X in Γ, i in N.

(i) Let $q = q_0$, $X = A^{(0)}$ for some A in V, the rule for A in P is $A \to B(C_1, \ldots, C_m)$; then $q' = q_0$, $x_1' = x_1$, $\omega' = \omega(B^{(0)}, i)$. (The rule indicates that with a rule of the form $A \to B(C_1, \ldots, C_m)$ the processing of A starts by having B called.)

(ii) Let $q = q_0$, $X = A^{(0)}$, and P contain $A \to (Q_1, \ldots, Q_m)$. Then if $x_2 = ax_3$, a in Q_j, $j \leqslant l$, we have $x_1' = x_1a$, $q' = q_j$, $\omega' = \gamma$; if $x_2 = ax_3$, a in Q_j, $j > l$, 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 $j > l$ we have backtracking of one symbol, i.e., the symbol has to be matched again.)

(iii) Let $q = q_0$, $X = A^{(0)}$, P contain $A \to (Q_1, \ldots, Q_m)$, and Q_j is ϵ. Then $x_1' = x_1$, $q' = q_j$, and $\omega' = \gamma$.

(iv) Let $q = q_j$, $X = A^{(0)}$, and P contains $A \to B(C_1, \ldots, C_m)$. Then $q' = q_0$, $x_1' = x_1$, and $\omega' = \gamma(A^{(j)}, i)(C_j^{(0)}, \mid x_1 \mid)$.

(v) Let $q = q_k$, $X = A^{(j)}$, P contain $A \to B(C_1, \ldots, C_m)$, and $g(j, k) = h$. If $h \leqslant l$ then $q' = q_h$, $x_1' = x_1$, $\omega' = \gamma$; if $h > l$ then $q' = q_h$, $\mid x_1' \mid = i$, $\omega' = \gamma$.

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

The *language recognized by* $A(R)$ is

$$\{w \mid w \text{ in } (\Sigma - \{\$\})^*, (q_0, \uparrow w\$, (S^{(0)}, 0)) \overset{*}{\vdash}_R (q_1, w\$ \uparrow, \epsilon\}.$$

In a manner similar to that 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 3.1).

DEFINITION 5.3. Let $R = (V, \Sigma, P, S, g, \$)$ be an (l, m)-TS. For A in V, x in $(\Sigma - \{\$\})^*$

(i) A has a *recognition failure* (or simply failure) on x if $A \Rightarrow_R (x_1 \vdash x_2\$, j)$, for $j \neq 1$, and $x_1x_2 = x$.

(ii) A has an *end failure* on x if $(q_0, \vdash x\$, (A^{(0)}, 0)) \overset{*}{\vdash}_R (q_0, x\$ \vdash, \gamma(B^{(0)}, n))$ for some γ in $(\Gamma \times N)^*$, n in N, B in V, and the rule for B in P is $B \to (Q_1, \ldots, Q_m)$, where no Q_1 contains ϵ, $1 \leqslant i \leqslant m$.

(iii) R has a *partial-acceptance failure* (p-a failure) on x if $S \Rightarrow_R (x_1 \vdash x_2\$, 1)$ for some x_1, x_2 in $(\Sigma - \{\$\})^*$, $x_1x_2 = x$.

(iv) A has a *loop failure* on x if $A(R)$, from configuration

$$(q_0, \vdash x\$, (A^{(0)}, 0)),$$

can make an unbounded number of moves.

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

THEOREM 5.1. *For any* gTS R *there exists a* (1, 3)-TS R' *such that* $T(R) = T(R')$.

Proof. Let $R = (V, \Sigma, P, S, \$)$ and consider the (1, 3) — TS $R' = (V', \Sigma, P', S, g, \$)$ as follows. $V' = V \cup \{X\}$, where X is not in V. The set of indices for R' is $U_3 = \{1, 2, 3\}$, g is defined by: $g(i, j) = j$ for all i, j in U_3. P' is defined as follows:

(i) $X \to (\phi, \phi, \{\epsilon\})$ is in P'.

(ii) If $A \to B(C, D)$ is in P, then $A \to B(C, D, X)$ is in P'.

(iii) If $A \to \epsilon$ is in P, then $A \to (\{\epsilon\}, \phi, \phi)$ is in P'.

(iv) If $A \to f$ is in P, then $A \to (\phi, \{\epsilon\}, \phi)$ is in P'.

(v) If $A \to a$, a in Σ, is in P, then $A \to (\{a\}, \Sigma - \{a\}, \phi)$ is in P'.

(vi) If A has no rule in P, then $A \to (\phi, \phi, \{\epsilon\})$ is in P'. 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 (ii). Otherwise, it can be shown in a straightforward way that $A \Rightarrow_R (x \vdash y, i)$ iff $A \Rightarrow_{R'} (x \vdash y, i + 1)$ for all A in V, x, y in Σ^*, i in $\{0, 1\}$.

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 5.4. 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 5.5. Let $R = (V, \Sigma, P, S, g, \$)$ be an (l, m)-TS. We define the gTS $M = (V_1, \Sigma, P_1, S^1, S)$ as follows: V_1 includes the set $\{A^i \mid A \text{ in } V,\ 1 \leqslant i \leqslant m\} \cup \{\langle A^i, j\rangle \mid 1 \leqslant i, j \leqslant m, A \text{ in } V.\}$ (For every variable A in V there are m variables in V_1, namely, A^i for $1 \leqslant i \leqslant m$. 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 $j > l$, A^j will recognize the null string). V_1 has other variables introduced by extended rules.

P_1 contains the following rules.

1. Let $A \to (Q_1, Q_2, \ldots, Q_m)$ be in P and $Q_j \neq \{\epsilon\}$ for any j. If $Q_i = \{a_1, \ldots, a_k\}$, $k \geqslant 1$, $i \leqslant l$, then $A^i \to a_1/a_2/\cdots/a_k$ is in P_1. If $Q_i = \{a_1, \ldots, a_k\}$, $k \geqslant 1$, $i > l$, then extended rule $A^i \to [a_1/\cdots/a_k](\epsilon, f)$ is in P_1. If $Q_i = \varnothing$ then $A^i \to f$.

2. Let $A \to (Q_1, \ldots, Q_m)$ be in P and suppose there exists j such that $Q_j = \{\epsilon\}$; then $A^i \to f$, for $i \neq j$, and $A^j \to \epsilon$ are in P_1.

3. Let $A \to B(C_1, \ldots, C_m)$ be in P, let $G_j = \{(p_i, q_i) \mid 1 \leqslant i \leqslant k\}$ be the set of pairs such that $g(p_i, q_i) = j$, for some $j \leqslant l$. Then

$$A^j \to B^{p_1}C^{q_1}_{p_1}/B^{p_2}C^{q_2}_{p_2}/\cdots/B^{p_k}C^{q_k}_{p_k}$$

is in P_1. If G_j is empty, $A^j \to f$. (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 $A \to B(C_1, \ldots, C_m)$ be in P, let $G_j = \{(p_i, q_i) \mid 1 \leqslant i \leqslant k\}$ be the set of pairs such that $g(p_i, q_i) = j$ for some $j > l$; then

$$A^j \to [B^{p_1}C^{q_1}_{p_1}](\epsilon, \langle A^j, 2\rangle),$$

$$\langle A^j, i\rangle \to [B^{p_i}C^{q_i}_{p_i}](\epsilon, \langle A^j, i+1\rangle), \qquad i = 2, 3, \ldots, k-1;$$

$$\langle A^j, k\rangle \to [B^{p_k}C^{q_k}_{p_k}](\epsilon, f).$$

If G_j is empty, then $A^j \to f$. (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

$A \to [\alpha](\beta, \gamma)$, 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 [Birman, 1970]:

LEMMA 5.1. *Consider R, M as above. For any y, z in Σ^*, if there exists j, $1 \leqslant j \leqslant m$, such that $A^j \Rightarrow_M^n (y \vdash z, 0)$, for some n, then $A^i \Rightarrow_M (\vdash yz, 1)$ for all $i \neq j$.*

LEMMA 5.2. *Consider R, M as before. For all A in V, xy in Σ^*, $A \Rightarrow_R (x \vdash y, i)$ iff $A^i \Rightarrow_M (x \vdash y, 0)$.*

THEOREM 5.2. *For any reduced (l, m)-TS R, there exists a reduced* gTS *M such that $T(R) = T(M)$.*

Proof. Using Lemma 5.2 above $S \Rightarrow_R (x\$ \vdash, 1)$ if and only if $S^1 \Rightarrow_M (x\$ \vdash, 0)$; it follows that $T(R) = T(M)$ and also that M is reduced.

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.)

First 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 with a useful property. From R_1 we will construct later the desired (l', m')-TS R', which has no loop failures and recognizes $T(R)$.

DEFINITION 5.6. 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: $l' = 2l$, $m' = l + m + 1$. (We introduce additional outcomes with the following meaning: If the outcome of a derivation in R is i, $i \leqslant l$, and the string recognized is *not* the null string, then the same derivation holds in R_1. However, *if* $i \leqslant l$ and the string recognized is the null string, or if $i > l$ in R, then the outcome in R_1 is $l + i$, the string recognized being the null string. The purpose is to separate the derivation in which the null string is recognized and which, as it will be shown later on, could give rise to a loop. The outcome $l + m + 1$ 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 a new variable not in V. P_1 contains the following rules:

1. If $A \to (Q_1, \ldots, Q_m)$ is in P and for no i does Q_i contain ϵ, then P_1 contains $A \to (Q_1', \ldots, Q_{m'}')$, where $Q_i' = Q_i$ for $i \leqslant l$, $Q_i' = \phi$ for $l < i \leqslant 2l$, $Q_{i+l}' = Q_i$ for $l < i \leqslant m$, and $Q_{m'}' = \phi$.

2. If $A \to (Q_1,\ldots,Q_m)$ is in P and Q_j contains ϵ for some j, then P_1 contains $A \to (Q_1',\ldots,Q_{m'}')$ where $Q_{l+j}' = \{\epsilon\}$ and $Q_i' = \phi$ for all $i \neq l+j$.

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

For $1 \leqslant i \leqslant m$, let $\bar{i} = i$ if $i \leqslant l$ and $\bar{i}$ undefined otherwise. Then, g_1 is defined as follows:

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

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

3. $g_1(i, l+m+1) = g_1(l+m+1, i) = l+m+1$ for $1 \leqslant i \leqslant l+m+1$.

LEMMA 5.3. *Consider R, R_1 as above.*

(a) *Let $A \Rightarrow_R^n (x \vdash y, i)$, for some integer n. If $i \leqslant l$ and $x \neq \epsilon$, then $A \Rightarrow_{R_1} (x \vdash y, i)$; otherwise $A \Rightarrow_{R_1} (x \vdash y, l+i)$.*

(b) *Let $A \Rightarrow_{R_1} (x \vdash y, i)$. If $i \leqslant l$, then $A \Rightarrow_R (x \vdash y, i)$. If $i > l$, then $x = \epsilon$ and $A \Rightarrow_R (\vdash y, i - l)$.*

Proof. By induction on n. For details see [Birman, 1970].

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 failure and $T(R') = T(R)$.

DEFINITION 5.7. Consider R, R_1 as given in Definition 5.6. We define the (l', m')-TS R' as $R' = (V', \Sigma, P', \langle S, \phi\rangle, g_1, \$)$ where $V' = \{\langle A, W\rangle |$ all A in V_1, $W \subseteq V_1\}$ and P contains the following rules:

1. If $A \to (Q_1,\ldots,Q_{m'})$ is in P_1, then $\langle A, W\rangle \to (Q_1,\ldots,Q_{m'})$ for all $W \subseteq V_1 - \{A\}$.

2. If $A \to B(C_1,\ldots,C_m)$ is in P_1, then for all $W \subseteq V_1 - \{A\}$, $\langle A, W\rangle \to \langle B, W'\rangle(\langle C_1, W_1\rangle,\ldots,\langle C_{m'}, W_{m'}\rangle)$, where $W_i = \phi$ for $i \leqslant l$, $W_i = W' = W \cup \{A\}$ for $i > l$.

3. $\langle X, W\rangle \to (\phi,\ldots,\phi,\{\epsilon\})$ for all $W \subseteq V$.

4. For all A in V, $W \subseteq V$, A in W, $\langle A, W\rangle \to (\phi,\ldots,\phi,\{\epsilon\})$.

In R' defined above, every variable has the form $\langle A, W\rangle$ 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 $\langle A, W\rangle$) keeps track of those variables which could give rise to a loop in R_1. For instance, if a variable $\langle A, W\rangle$ 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 now show the relation between R_1, R'. But first, we have the following definition:

DEFINITION 5.8. *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^* as $L(A, x) = \{Z \mid Z \text{ in } V \text{ and } \exists\gamma \text{ in } (\Gamma \times N)^* \text{ such that } (q_0, \vdash x, (A^{(0)}, 0)) \overset{*}{\vdash}_R (q_0, \vdash x, \gamma(Z^{(0)}, 0))\} \cup \{A\}$. 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 $A^{(0)}$ 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 $A \to B(C_1, \ldots, C_m)$, then B is in $L(A, x)$, for all x in Σ^*.

LEMMA 5.4. *Consider* R_1, R' *as given by Definition* 5.7. *For all* xy *in* Σ^*, *and all integers* i, $1 \leqslant i \leqslant m'$:

(a) *if* $A \Rightarrow_{R_1}^{n} (x \vdash y, i)$, *for some integer* n, *then* $\langle A, W\rangle \Rightarrow_{R'} (x \vdash y, i)$ *for all sets* W, $W \subseteq V - L(A, xy)$;

(b) *if* $\langle A, W\rangle \Rightarrow_{R'}^{n} (x \vdash y, i)$ *for some set* $W \subseteq V - L(A, xy)$ *and some integer* n, *then* $A \Rightarrow_{R_1} (x \vdash y, i)$.

Proof. By induction on n. The details [Birman, 1970] are omitted here.

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

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

LEMMA 5.5. *Let* R' *be given by Definition* 5.7. R' *has no loop failures.*

Proof. Assume the contrary; then we must have a variable $\langle A, L\rangle$ in V' and a string x in $(\Sigma - \{\$\})^*\$$ such that $(q_0, \vdash x, (\langle A, L\rangle^{(0)}, 0)) \vdash_{R'} (q_0, \vdash x, \gamma(\langle A, L\rangle^{(0)}, 0))$, for some γ in $(\Gamma \times N)^*$. (The notation $\langle A, L\rangle^{(0)}$ is derived from $\langle A, L\rangle$ as a variable in V' and the superscript (0) in Γ, as in Definition

5.2.) Let the rule for A in P_1 be $A \to B(C_1, \ldots, C_{m'})$; then P' contains $\langle A, L\rangle \to \langle B, L'\rangle(\langle C_1, L_1\rangle, \ldots, \langle C_{m'}, B_{m'}\rangle)$, where $L_i = \phi$ for $i \leqslant l$ and $L_i = L' = L \cup \{A\}$ for $i > l$. Two cases are possible:

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

Case 2. $\langle B, L'\rangle \Rightarrow_{R'} (\uparrow x, j)$ for some j, $l < j < m'$; in this case the second symbol of γ is $(\langle C_j, L'\rangle^{(i)}, 1)$ for some i, $1 \leqslant i \leqslant m'$, 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 $\gamma(\langle A, L\rangle^{(0)}, 0)$, i.e., L, must properly include the set corresponding to the first symbol, L, which is a contradiction. Q.E.D.

THEOREM 5.3. *Given any* (l, m)-TS R *there exists an* (l', m')-TS R' *such that* $T(R) = T(R')$ *contains no loop failures.*

Proof. The theorem follows from Lemmas 5.3–5.5. Q.E.D.

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 5.9. 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: $l' = l$, $m' = l + m + 1$, $V' = \{A, \bar{A} \mid A \text{ in } V\} \bigcup \{X\}$, where X is a new symbol.

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 $A \Rightarrow_R (x \mid y, i)$ for some xy in $(\Sigma - \{\$\})^*$, then $\bar{A} \Rightarrow_{R'} (x \restriction y, i + l)$; if $A \Rightarrow_R (x \restriction y, i)$ for some x in $(\Sigma - \{\$\})^*\$$ then $\bar{A} \Rightarrow_{R'} (x \restriction y, i)$. 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 (i.e., not $\bar{A}$), 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 $A \to (Q_1, ..., Q_m)$ is in P and no Q_i contains ϵ for $1 \leqslant i \leqslant m$, then let k be such that \$ is in Q_k. In P' we have the rule $\bar{A} \to (Q_1', Q_2', ..., Q_{m'}')$ where $Q'_{l+i} = Q_i$ for $1 \leqslant i \leqslant m$ and $i \neq k$; if $k > l$ then $Q'_{k+l} = Q_k$, otherwise $Q'_{k+l} = Q_k - \{\$\}$ and $Q_k' = \{\$\}$. Also in P' we have $A \to (\phi, ..., \phi, \{\epsilon\})$. (According to the rule for $\bar{A}$, an outcome $i \leqslant l$ will occur in R' only when the endmarker has been successfully matched. We notice that all sets Q_i', $1 \leqslant i \leqslant m'$, which have not been specifically described are implicitly defined as empty according to the restrictions in Definition 5.1.)

2. If $A \to (Q_1, ..., Q_m)$ is in P and Q_k contains ϵ, for some k, then P' contains $\bar{A} \to (Q_1', Q_2', ..., Q_{m'}')$ and $A \to (Q_1'', Q_2'', ..., Q_{m'}'')$, where $Q'_{k+l} = \{\epsilon\}$, and $Q_p'' = \{\epsilon\}$, where $p = k$ if $k \leqslant l$ and $p = k + l$ if $k > l$.

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

4. $X \to (\phi, ..., \phi, \{\epsilon\})$.

g' is defined as follows:

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

2. If $g(i, j) = k$, $i, k \leqslant l$, $j > l$, then $g'(l + i, l + j) = g'(i, l + j) = k$. (Since $j > l$ in R, there is only one corresponding outcome in R': $j + l$.)

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

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

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

6. $g'(i, l + m + 1) = g'(l + m + 1, i) = l + m + 1$ for $1 \leqslant i \leqslant l + m + 1$.

LEMMA 5.6. *Consider R, R' as given by Definition 5.9. Then for all A in V, x in $(\Sigma - \{\$\})^*$, y in $(\Sigma - \{\$\})^*\$$:*

(a) $A \Rightarrow_R (x \restriction y, i)$, $1 \leqslant i \leqslant m$, *iff* $\bar{A} \Rightarrow_{R'} (x \restriction y, l + i)$.

(b) $A \Rightarrow_R (y \restriction, i)$, $1 \leqslant i \leqslant l$, *iff* $\bar{A} \Rightarrow_{R'} (y \restriction, i)$.

(c) $A \Rightarrow_R (\restriction, i)$, $1 \leqslant i \leqslant l$, *iff* $A \Rightarrow_{R'} (\restriction, i)$ *or* $A \Rightarrow_{R'} (\restriction, l + i)$.

(d) $A \Rightarrow_R (\restriction, i)$, $l < i \leqslant m$, *iff* $A \Rightarrow_{R'} (\restriction, l + i)$.

For a complete proof see [Birman, 1970].

LEMMA 5.7. *Consider R, R' as given by Definition* 5.9. *Then R' has no* p — a *or end failures.*

Proof. First we prove that for no A in V, x in $(\Sigma - \{\$\})^*$, y in $(\Sigma - \{\$\})^*\$$, integer i, $i \leqslant l$, do we have $\bar{A} \Rightarrow_{R'}^{n} (x \vdash y, i)$. The proof is by induction on n.

$n = 1$. The rule for A in P must have the form $A \to (Q_1, \ldots, Q_m)$; 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 $\bar{A} \to \bar{B}(X_1, X_2, \ldots, X_{m'})$. If the derivation $\bar{A} \Rightarrow_{R'}^{n} (x \vdash y, i)$ holds, we must have $\bar{B} \Rightarrow_{R'} (x_1 \vdash x_2 x_3, j)$, $X_j \Rightarrow_{R'} (x_2 \vdash x_3, k)$, $x_1 x_2 x_3 = xy$, $g'(j, k) = i$.

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

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

Thus, for no x in $(\Sigma - \{\$\})^*$, y in $(\Sigma - \{\$\})^*\$$, we have $\bar{S} \Rightarrow_{R'} (x \vdash y, 1)$ 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 unbarred variables are called, and as it was shown in Lemma 5.6, these variables never attempt to match an input symbol. Hence, no end failures are possible.

THEOREM 5.4. *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 5.6(b), we conclude that $T(R) = T(R')$. By Lemma 5.7, R' has no p-a or end failures. Assume that R has no loop failures, suppose $\bar{A}$ has a loop failure on x, x in $(\Sigma - \{\$\})^*\$$. In R we cannot have

$A \Rightarrow_R (x_1 \uparrow x_2, i)$ for $x_1x_2 = x$, $1 \leqslant i \leqslant m$ (this will contradict Lemma 5.6). 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'.

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

THEOREM 5.5. *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 Theorems 5.3, 5.4, and the observation that by Definition 5.1 an (l, m)-TS does not have any subroutine failures.

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 5.2 we get a (reduced) gTS R' such that $T(R') = T(R)$.

THEOREM 5.6. *Given any* gTS R, $R = (V, \Sigma, P, S, \$)$, *there exists a reduced* gTS R' *such that* $T(R) = T(R')$.

Proof. This theorem follows from Theorems 5.1, 5.2, and 5.5.

Theorem 5.6 shows that in a gTS all loops, 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.

THEOREM 5.7. *If a language is recognized by an* (l, m)-TS *for any* l *and* m, *it is recognized by a* $(1, 3)$-TS.

Proof. Immediate from Theorems 5.5 and 5.1.

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 definitions 2.3 and 5.4 we notice that the property of being "well-formed" for a recognition schema is a restriction of the "reduced" property to the 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, among them Theorem 3.8, ("It is undecidable whether an arbitrary TS is a wfTS") lead us to believe that this is not possible. The fact, illustrated by Theorem 5.6, that every gTS can be made "reduced" seems to indicate an advantage in using the gTS instead of the TS. The gTSL includes, properly we believe, the TSL and moreover, there is no apparent difficulty in using the gTS instead of the TS.

Acknowledgment

The authors are indebted to Doug McIlroy and to the referee for many helpful comments.

References

1. A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation and Compiling, Vol. I, "Parsing," Prentice Hall, Englewood Cliffs, N. J., 1972.
2. A. Birman, The TMG Recognition Schema, Ph.D. Thesis, Princeton University, 1970.
3. A. Birman and J. D. Ullman, "Parsing Algorithms with Backtrack," *Symposium* on Switching and Automata Theory, pp. 153–174, Santa Monica, Calif., October, 1970.
4. W. J. Chandler, "Abstract families of deterministic languages," *Proceedings of ACM Symposium on Theory of Computing*, pp. 21–30, Marina del Rey, Calif., May, 1969.
5. J. Earley, An Efficient Context Free Parsing Algorithm, *CACM* **13** (1970), 94–102.
6. S. Ginsburg, "The Mathematical Theory of Context-Free Languages," McGraw-Hill, New York.
7. S. Ginsburg and S. A. Greibach, "Deterministic Context Free Languages," *Inf. Control* **9** (1966), 620–648.
8. D. E. Knuth, Lecture notes, International Summer School on Computer Programming, Copenhagen, Denmark (August 1967).
9. R. M. McClure, "TMG—A syntax directed compiler," *Proc. ACM 20th National Conference*, pp. 262–274, 1965.
10. J. Myhill, Linear Bounded Automata, WADD Technical Note 60-165, Wright Patterson AFB, Ohio, 1960.
11. D. V. Schorre, "META II, a syntax oriented compiler writing language, *Proc. 19th ACM National Conference*, pp. D1.3-1–D1.3-11, 1964.