JOURNAL OF COMPUTER AND SYSTEM SCIENCES 6, 324–342 (1972)

Absolutely Parallel Grammars and Two-Way Finite-State Transducers*

VACLAV RAJLICH[†]

Case Western Reserve University, Cleveland, Ohio 44106

Received August 12, 1970

Absolutely parallel grammars are defined, and it is shown that the family of languages generated is equal to the family of languages generated by two-way deterministic finite-state transducers (abbreviated 2ft). Furthermore it is shown that this family forms a full AFL closed under substitution. It is shown that the family of languages generated by two-way nondeterministic finite-state transducers is equal to the family of checking automata languages and that it properly contains the family of languages generated by 2ft.

INTRODUCTION

Recently there has been an extensive investigation of different types of context-sensitive grammars with the context scattered through the whole word. (See [10], [13], etc.) This paper defines a new class of grammars, called absolutely parallel grammars, belonging to this category. A particular feature of this grammar making it worthwhile is that it describes in simple terms all languages generated by two-way finite-state transducers, which in turn play an important role in the general understanding to two-way deterministic machines [1].

The paper is divided into five sections. In the first two sections, we present definitions of two-way deterministic finite-state transducers and absolutely parallel grammars. In addition, these sections contain some technical lemmas. The equality of the two corresponding families of languages is established in the third section.

In the fourth section, the family of absolutely parallel languages is shown to be a full AFL [4].

* This paper is derived in part from a dissertation [14] submitted in partial fulfillment of the requirements for the Ph.D. degree in mathematics at Case Western Reserve University. A preliminary version of this paper appeared in the *Proc. Third Ass. Comput. Mach. Symp. Theory Computing*, Shaker Heights, Ohio, May 1971.

[†] Present address: Research Institute of Mathematical Machines, Loretanske n. 3, Prague 1, Czechoslovakia.

In the fifth section, the output languages of the nondeterministic two-way finite-state transducers are shown to be the same as the languages accepted by the family of checking automata [8]. The absolutely parallel languages over a one-letter alphabet are shown to be regular, and therefore they form a proper subfamily of the checking automaton languages; hence the deterministic and nondeterministic two-way finite-state transducers differ in their generative power.

1. Two-Way Finite-State Transducers

In this section we shall define two-way finite-state transducers, both deterministic and nondeterministic ones, and provide the necessary notation.

Intuitively, a two-way nondeterministic finite-state transducer consists of an input tape, finite control, and output tape. The control can see at most one letter on each side of the input pointer, which can be moved back and forth. Furthermore, it can insert a word on the end of the output tape. These actions are specified by move-rules. Each move-rule tells exactly the situation in which it applies, i.e., the state of the finite control, possibly the letters on one or both sides of the input pointer, and the action to be taken, i.e., the move of the pointer, the new state of the finite control, and the word added to the end of the output tape.

It has proved expedient to depart from the familiar practice of scanning the input tape with a reading head that can see only one symbol at a time. It is clear, however, that the present results concerning generated languages apply also to the formulation with a reading head. A similar remark applies to the omission of endmarkers.

This completes the informal description. We now proceed formally. For this purpose we use infinite alphabets K, X, and Y of symbols called *states*, *input symbols*, and *output symbols*, respectively, and a symbol $\uparrow \notin K \cup X \cup Y$, called the *pointer*.

Definition 1.1. A *configuration* is any triple $(p, x \uparrow x', w)$ where $p \in K$, x and $x' \in X^*$, and $w \in Y^*$. A *move-rule* is any 5-tuple $\mu = (p, a \uparrow b, q, a_1 \uparrow b_1, y)$ where p and $q \in K$, a and $b \in X \cup \{\lambda\}$, $ab = a_1 b_1$, and $y \in Y^*$. [λ denotes the empty word.] A *two-way nondeterministic finite-state transducer* (abbreviated 2nft) is a triple $F = (M, s, K_1)$ where M is a finite set of move-rules, $s \in K$, and K_1 is a finite set of states. State s is called the *start state*; the states of K_1 are called *accepting states*. We denote by K_F, X_F, and Y_F the finite sets of states, input symbols and output symbols, respectively, that are explicitly mentioned in the description of a particular 2nft F.

For fixed 2nft $F = (M, s, k_1)$, for any move-rule $\mu \in M$ and configurations U, V, $U \vdash_\mu V$ iff there exist $x, x' \in X_F{}^*$ and $z \in Y_F{}^*$ such that $U = (p, xa \uparrow bx', z)$ and $V = (q, xa_1 \uparrow b_1 x', zy)$. Then $U \vdash_M V$ iff there exists $\mu \in M$ such that $U \vdash_\mu V$. $\overset{*}{\vdash}_M$ is the reflexive and transitive closure of $\vdash_M$. If $U_0 \vdash_M U_1 \vdash_M \cdots U_{k-1} \vdash_M U_k$, then and only then $U_0 \vdash_M{}^k U_k$.

If $(s, \uparrow x, \lambda) \overset{*}{\vdash}_M (p, x\uparrow, y)$ with $p \in K_1$, we say x *is accepted* by F and y *is generated* by F (denoted also $y \in F(x)$).

The *accepted language* of F is the set $\Sigma(F)$ of all words accepted by F.

The *generated language* of F is the set $\Gamma(F)$ of all words generated by F.

It is clear in view of [12] that the accepted languages for 2nft are regular. The generated languages will be described in greater detail in the third and fourth sections.

DEFINITION 1.2. A *two-way* (*deterministic*) *finite-state transducer* (abbreviated 2ft) is a 2nft $F = (M, s, K_1)$ with the following restrictions on the set of move-rules:

(i) For each configuration U, there exists at most one move-rule $\mu \in M$ such that for some V, $U \vdash_\mu V$.

(ii) No move-rule of the form $(p, a \uparrow b, q, a_1 \uparrow b_1, y)$ has $p \in K_1$.

With these restrictions it is obvious that to each accepted word there corresponds a unique generated word.

It is easy to show that the family of languages generated by 2ft and 2nft are equal to the analogous families of [1] and [2].

The remainder of this section is devoted to the definitions of some useful notions related to 2ft and to the derivation of some technical results concerning them.

DEFINITION 1.3. A *left* 2*ft* and 2*nft* $E = (N, t, L)$ is analogous to 2ft and 2nft, respectively, with the following difference:
x is accepted and y is generated by E iff $(t, \uparrow x, \lambda) \overset{*}{\vdash}_N (p, \uparrow x, y)$ where $p \in L$.

LEMMA 1.1. *The families of languages generated by* 2*ft and left* 2*ft are equal.*

Proof. Let $F = (M, s, K_1)$ be a 2ft. Then construct a left 2ft $E = (N, s, \{r\})$ where $X_E = X_F \uplus \{¢, \$\}$,[1] $K_E = K_F \uplus \{q, r\}$, $Y_E = Y_F$ and

$$N = M \cup \{(s, \uparrow ¢, s, ¢\uparrow, \lambda)\} \cup \{(p, \uparrow \$, q, \uparrow \$, \lambda) \mid p \in K_1\}$$
$$\cup \{(q, a\uparrow, q, \uparrow a, \lambda) \mid a \in X_F\} \cup \{(q, ¢\uparrow, r, \uparrow ¢, \lambda)\}.$$

Then $\Sigma(E) = \{¢\} \cdot \Sigma(F) \cdot \{\$\} \cdot X_E{}^*$ and $\Gamma(E) = \Gamma(F)$.

Analogously, given a left 2ft $E = (N, t, L)$ construct a 2ft $F = (M, t, \{r\})$ where $X_F = X_E \uplus \{¢, \$\}$, $K_F = K_E \uplus \{q, r\}$, $Y_F = Y_E$ and

$$M = N \cup \{(s, \uparrow ¢, s, ¢\uparrow, \lambda)\} \cup \{(p, ¢\uparrow, q, ¢\uparrow, \lambda) \mid p \in L\}$$
$$\cup \{(q, \uparrow a, q, a\uparrow, \lambda) \mid a \in X_E\} \cup \{(q, \uparrow \$, r, \$\uparrow, \lambda)\}.$$

Then $\Sigma(F) = \{¢\} \cdot \Sigma(E) \cdot \{\$\}$ and $\Gamma(F) = \Gamma(E)$.

[1] The symbol $\uplus$ differs from the standard set-theoretic notation. Used in the statement of the form $A = A_1 \uplus A_2 \uplus \cdots \uplus A_n$ it means $A = A_1 \cup A_2 \cup \cdots \cup A_n$ and simultaneously $A_1, A_2, \ldots, A_n$ are mutually disjoint. It will be frequently used in various constructions.

DEFINITION 1.4. A *normalized left 2ft* is a left 2ft $F = (M, s, \{r\})$ such that

(i) each move-rule is of the form $(p, \uparrow a, q, a\uparrow, y)$ or $(p, a\uparrow, q, \uparrow a, y)$ with $a \neq \lambda$, denoted $R(p, a, q, y)$ or $L(p, a, q, y)$, respectively.

(ii) $K_F \times K_F$ and Y_F are mutually disjoint.

LEMMA 1.2. *There exists an effective procedure to find for each left 2ft F a normalized left 2ft F′ such that* $\Gamma(F) = \Gamma(F')$.

Proof. Let $F = (M, s, K_1)$ be a left 2ft. Without loss of generality we can assume K_1 contains one state only, namely $K_1 = \{r\}$. (This follows easily from the proof of Lemma 1.1; given a left 2ft $F'' = (M'', s, K'')$, we can construct a 2ft $F' = (M', s, \{r'\})$ and finally another left 2ft $F = (M, s, \{r\})$ such that $\Gamma(F'') = \Gamma(F') = \Gamma(F)$.)

Our task here is to construct a new left 2ft $F' = (M', s, \{r\})$ in such a way that each move-rule in M is simulated by new move-rules of the type $R(p, a, q, y)$ and $L(p, a, q, y)$. All information necessary for this simulation must be carried by the new states. Hence let $X_{F'} = X_F$, $Y_{F'} = Y_F$ and

$$K_{F'} = K_F \cup (K_F \times X_F) \cup (K_F \times X_F \times X_F) \cup (K_F \times \{B\}) \cup (K_F \times X_F \times \{B\}).$$

Then note that the set of the state symbols K_F can be split into two disjoint parts $K_F = K_L \cup K_R$, where $K_L = \{p \mid (p, a\uparrow, q, a_1 \uparrow b_1, y) \in M\}$ (K_L can be called the set of left-looking states) and $K_R = K_F - K_L$ (can be called the set of right-looking states). We immediately note that if F is deterministic and $p \in K_L$, then no move-rule of the type $(p, \uparrow b, q, a_1 \uparrow b_1, y) \in M$. The simulation of move-rules with left-looking states differs from the simulation of move-rules with right-looking states in order to preserve determinism of the 2ft.

Each move-rule $\mu \in M$ will be replaced by a set of move-rules of the desired type (in the following, $a, b \in X_F$):

(i) If $p \in K_L$ and $\mu = (p, a\uparrow b, q, ab\uparrow, y), (p, a\uparrow b, q, \uparrow ab, y)$, or $(p, a\uparrow b, q, a\uparrow b, y)$, then

$$\begin{aligned} M_\mu = {} & \{L(p, a, (p, B), \lambda), R((p, B), a, (p, a), \lambda), R((p, a), b, q, y)\}, \\ & \{L(p, a, (p, B), \lambda), R((p, B), a, (p, a), \lambda), R((p, a), b, (p, a, B), \lambda), \\ & L((p, a, B), b, (p, a, b), \lambda), L((p, a, b), a, q, y)\}, \qquad \text{or} \\ & \{L(p, a, (p, B), \lambda), R((p, B), a, (p, a), \lambda), \\ & R((p, a), b, (p, a, B), \lambda), L((p, a, B), b, q, y)\}, \end{aligned}$$

respectively.

(ii) If $p \in K_R$ and $\mu = (p, a\uparrow b, q, ab\uparrow, y), (p, a\uparrow b, q, \uparrow ab, y)$, or $(p, a\uparrow b, q, a\uparrow b, y)$, then

$$\begin{aligned} M_\mu = \{&R(p, b, (p, B), \lambda), L((p, B), b, (p, b), \lambda),\\ &L((p, b), a, (p, b, B), \lambda), R((p, b, B), a, (p, a, b), \lambda), R((p, a, b), b, q, y)\},\\ \{&R(p, b, (p, B), \lambda), L((p, B), b, (p, b), \lambda), L((p, b), a, q, y)\}, \qquad \text{or}\\ \{&R(p, b, (p, B), \lambda), L((p, B), b, (p, b), \lambda),\\ &L((p, b), a, (p, b, B), \lambda), R((p, b, B), a, q, y)\}, \end{aligned}$$

respectively.

(iii) If $\mu = (p, a\uparrow, q, a\uparrow, y)$, $(p, \uparrow b, q, \uparrow b, y)$, or $(p, \uparrow, q, \uparrow, y)$, then

$$\begin{aligned} M_\mu = &\{L(p, a, (p, B), \lambda), R((p, B), a, q, y)\},\\ &\{R(p, b, (p, B), \lambda), L((p, B), b, q, y)\}, \qquad \text{or}\\ &\{R(p, b, (p, B), \lambda), L((p, B), b, q, y) \mid b \in X_F\}, \end{aligned}$$

respectively. (Note that for the last type of move-rule, the construction is suitable strictly for the *left* 2ft only, because for any left 2ft F, if $x \in \Sigma(F)$, then also $xa \in \Sigma(F)$ for any $a \in X_F$ and the same word is generated.)

(iv) For all other $\mu \in M$, let $M_\mu = \{\mu\}$.

Then let $M' = \bigcup_{\mu \in M} M_\mu$ which completes the construction of F'.

It is clear that F' is a normalized left 2ft. To show $\Gamma(F) = \Gamma(F')$, note that the following statements are easily verified:

(a) If $(p, x \uparrow x', w) \vdash_M (q, y \uparrow y', w')$ where p and $q \in K_F$, then

$$(p, x \uparrow x', w) \overset{*}{\vdash}_{M'} (q, y \uparrow y', w').$$

(b) If $(p, x \uparrow x', w) \vdash_{M'} V_1 \vdash_{M'} \cdots \vdash_{M'} V_k \vdash_{M'} (q, y \uparrow y', w')$ where p and $q \in K_F$ and the states of $V_1, \ldots, V_k$ are in $K_{F'} - K_F$, then $(p, x \uparrow x', w) \vdash_M (q, y \uparrow y', w')$.

We are now in the position to introduce a concept which plays a key role in some of the proofs later:

Definition 1.5. Let $F = (M, p_0, K_1)$ be a left 2ft and let us have a finite sequence of configurations $(U_0, \ldots, U_n)$ such that $U_0 = (p_0, \uparrow xy, \lambda)$, $U_n = (p_n, \uparrow xy, w_n)$ with $p_n \in K_1$, $U_i \vdash_{\mu_i} U_{i+1}$ and $\mu_i = (p_i, a_i \uparrow b_i, p_{i+1}, c_i \uparrow d_i, z_i)$. Then we can divide all configurations into two disjoint classes according to the position of the pointer in relation to the boundary between x and y:

$$\begin{aligned} U_i \in l \quad &\text{iff} \quad U_i = (p_i, x_i \uparrow x_i'y, w_i) \qquad \text{and}\\ U_i \in \rho \quad &\text{iff} \quad U_i = (p_i, xy_i \uparrow y_i', w_i) \qquad \text{where} \quad y_i \neq \lambda. \end{aligned}$$

For the purpose of simpler notation, we shall assume $U_{-1}, U_{n+1} \in l$.

Then for each i define $P_i(x \uparrow y)$ in the following way:

(i) $P_i(x \uparrow y) = z_i$ iff $U_i, U_{i+1} \in l$,
(ii) $P_i(x \uparrow y) = p_i$ iff $U_{i-1}, U_i \in l$, $U_{i+1} \in \rho$,
(iii) $P_i(x \uparrow y) = p_{i+1}$ iff $U_i \in \rho$, $U_{i+1}, U_{i+2} \in l$,
(iv) $P_i(x \uparrow y) = \lambda$ in all other cases.

Then $L(x \uparrow y) = P_0(x \uparrow y) \cdot P_1(x \uparrow y) \cdots P_{n-1}(x \uparrow y)$ is called the *left part of computation* on xy. Intuitively, $L(x \uparrow y)$ consists of those pieces of the output word which are generated while the pointer moves within the word x, including its boundary. These pieces are separated by the state symbols of the states which are assumed by the finite control while entering the word x nontrivially (i.e., for at least one complete move-rule) or leaving it nontrivially (i.e., after at least one complete move-rule within x). Note that the states in which the pointer touches the boundary of x from the left and immediately departs back to the right are explicitly excluded.

Observe the following properties of $L(x \uparrow y)$:

Observation 1.1. All left parts of computation are of the form $L(x \uparrow y) = w_0 p_1 p_2 w_2 p_3 p_4 \cdots p_{2n-1} p_{2n} w_{2n}$, where $w_i \in Y_F{}^*$, $p_i \in K_F$, $n \geqslant 0$. Moreover $i \neq j$ implies $p_i \neq p_j$ (Otherwise 2ft F "loops" and xy is not an accepted word, hence $L(x \uparrow y)$ is not defined.)

For the relation of the two neighboring left parts of computation, suppose that $y = ay'$ where $a \in X_F$. Then $L(xa \uparrow y') = w_0 v_1 w_2 v_3 \cdots v_{2n-1} w_{2n}$, where $v_i = w_{i,1} q_1 q_2 w_{i,2} q_3 q_4 \cdots q_{2m-1} q_{2m} w_{i,2m}$ with $q_i \in K_F$, $w_{i,j} \in Y_F{}^*$, $m \geqslant 0$, and $j \neq k$ implies $q_j \neq q_k$.

2. Absolutely Parallel Grammars

In this section, we shall define absolutely parallel grammars and provide the necessary notation.

Definition 2.1. An *absolutely parallel grammar* (abbreviated apg) is any 4-tuple $G = (N, T, S, P)$ where N and T are disjoint finite sets of symbols, $S \in N$, and P is a finite set of productions π of the form $(A_1, \ldots, A_n) \to (y_1, \ldots, y_n)$ with $A_i \in N$ and $y_i \in (N \cup T)^*$. Here N, T and S are called the *nonterminal alphabet*, *terminal alphabet* and *start symbol*, respectively. For any production of the above form, we speak of $(A_1, \ldots, A_n)$ and $(y_1, \ldots, y_n)$ as the *left side* and *right side*, respectively.

Then $w \Rightarrow_\pi w'$ iff $w = u_1 A_1 u_2 A_2 \cdots u_n A_n u_{n+1}$ and $w' = u_1 y_1 u_2 y_2 \cdots u_n y_n u_{n+1}$, where $u_i \in T^*$. For any set of productions P, $w \Rightarrow_P w'$ iff there exists $\pi \in P$ such that $w \Rightarrow_\pi w'$. Again $\overset{*}{\Rightarrow}_P$ is the reflexive and transitive closure of $\Rightarrow_P$. If

$$w_0 \Rightarrow_P w_1 \Rightarrow_P \cdots \Rightarrow_P w_k ,$$

then and only then $w_0 \Rightarrow_P{}^k w_k$.

If $S \overset{*}{\Rightarrow}_P w \in T^*$, then w is *word derived by G. The language generated by* an apg G is the set $A(G)$ of all words derived by G. Moreover define $A^N(G) = \{w \mid S \overset{*}{\Rightarrow}_P w\}$. The family of languages generated by an apg is called the family of *absolutely parallel languages* (apl).

As an example, consider the apg $G = (\{S\}, \{a, b, c\}, S, P)$ where P consists of the productions $(S) \to (SSS)$, $(S, S, S) \to (Sa, Sb, Sc)$, $(S, S, S) \to (\lambda, \lambda, \lambda)$. Then the language generated by this grammar is $A(G) = \{a^i b^i c^i \mid i \geqslant 0\}$.

An apg of a special form will play an important role:

DEFINITION 2.2. An apg $G = (N, T, S, P)$ is in *indexed form* iff

(i) For each production $(A_1, ..., A_n) \to (y_1, ..., y_n)$, $i \neq j$ implies $A_i \neq A_j$.

(ii) For each two productions $(A_1, ..., A_n) \to (y_1, ..., y_n)$ and $(B_1, ..., B_m) \to (z_1, ..., z_m)$ either $A_1 \cdots A_n = B_1 \cdots B_m$, or for each i, j, $A_i \neq B_j$.

(iii) For each production $(A_1, ..., A_n) \to (y_1, ..., y_n)$, S does not occur in $y_1 \cdots y_n$.

LEMMA 2.1. *There exists an effective procedure to find for each apg G an apg G′ in the indexed form such that* $A(G) = A(G')$.

Proof. Let $G = (N, T, S, P)$. For every production $\pi \in P$ of the form $(A_1, ..., A_n) \to (y_1, ..., y_n)$, where $y_1 \cdots y_n = z_1 B_1 z_2 B_2 \cdots z_m B_m z_{m+1}$ with $B_i \in N$, $z_i \in T^*$, define $l(\pi) = A_1 \cdots A_n$, $r(\pi) = B_1 \cdots B_m$. Then let $Y = \{\beta \mid$ there exists $\pi \in P$ such that $\beta = l(\pi)$ or $\beta = r(\pi)\}$. Then obviously Y is finite. Let $k = \max\{|\beta| \mid \beta \in Y\}$ Then for the production π construct a new production π' replacing each nonterminal A_i and B_j by the triple $(A_i, i, l(\pi))$ and $(B_j, j, r(\pi))$, respectively. In this way we can get an apg satisfying the condition (i) and (ii) of the Def. 2.2.

The condition (iii) alone can be satisfied very easily by introducing a new start symbol S' and adding a new production of the type $(S') \to (S)$ where S is the old starting symbol.

More formally, let $G' = (N', T, S', P')$, where $N' = (N \times \{1, ..., k\} \times Y) \cup \{S'\}$. For each production π of the above form, let h be the homomorphism over $N' \cup T$ such that $h((B_j, j, r(\pi))) = B_j$ $(j = 1, ..., m)$ and $h(a) = a$ for $a \in T$. Now let $\pi' = ((A_1, 1, l(\pi)), ..., (A_n, n, l(\pi))) \to (y_1', ..., y_n')$ where $(y_1', ..., y_n')$ satisfies the following conditions:

(i) $h(y_i') = y_i \quad (i = 1,..., n)$ and

(ii) $y_1' \cdots y_n' = z_1(B_1, 1, r(\pi)) \cdots z_m(B_m, m, r(\pi))\, z_{m+1}$.

Finally let $P' = \{\pi' \mid \pi \in P\} \cup \{(S') \to ((S, 1, S))\}$.

Then G' is apg in the indexed form.

To prove $\Lambda(G) = \Lambda(G')$, first we can show

$$S \Rightarrow_P^k w_1B_1w_2B_2 \cdots w_nB_nw_{n+1} \text{ where } w_i \in T^*,\ B_i \in N, \text{ iff}$$

$(S, 1, S) \Rightarrow_{P'}^k w_1(B_1, 1, B_1B_2 \cdots B_n)\, w_2(B_2, 2, B_1B_2 \cdots B_n) \cdots w_n(B_n, n, B_1B_2 \cdots B_n)\, w_{n+1}$ by induction on k.

From this $w \in \Lambda(G)$ iff $S \overset{*}{\Rightarrow}_P w \in T^*$ and this is true iff $S' \overset{*}{\Rightarrow}_{P'} (S, 1, S) \overset{*}{\Rightarrow}_{P'} w \in T^*$ and this holds iff $w \in \Lambda(G')$. Hence $\Lambda(G') = \Lambda(G)$ which proves the Lemma.

To illustrate briefly the generative power of apg, let us introduce the following definition and lemma:

DEFINITION 2.3. A *context-free grammar* (abbreviated cfg) is any 4-tuple $G = (N, T, S, P)$, where N and T are disjoint finite sets of symbols, $S \in N$, and P is a finite set of productions of the form $\pi = A \to y$ with $A \in N$, $y \in (N \cup T)^*$. $w \Rightarrow_\pi w'$ iff $w = uAv$, $w' = uzv$ with $uv \in (N \cup T)^*$. $w \Rightarrow_P w'$ iff there exists $\pi \in P$ such that $w \Rightarrow_\pi w'$. $\overset{*}{\Rightarrow}_P$ is the reflexive and transitive closure of $\Rightarrow_P$. If $w_0 \Rightarrow_P w_1 \Rightarrow_P \cdots \Rightarrow_P w_k$, then and only then $w_0 \Rightarrow_P^k w_k$. A *context-free language* (abbreviated cfl) is $\Lambda(G) = \{w \mid S \overset{*}{\Rightarrow}_P w \in T^*\}$, where G is a cfg of the above form.

A cfg is called *nonexpansive*, if, for every $A \in N$ and $w \in (N \cup T)^*$, $A \overset{*}{\Rightarrow}_P w$ implies w does not contain two occurences of A.

LEMMA 2.2. *The family of nonexpansive context-free languages is properly contained in the family of apl.*

Proof. We shall use a theorem of [7]: Any nonexpansive context-free language is equivalent to a derivation bounded set X for some cfg $G = (N, T, S, P)$, i.e., $w \in X$ iff $S \Rightarrow_P w_1 \Rightarrow_P \cdots \Rightarrow_P w_h \Rightarrow_P w \in T^*$ and the number of nonterminal symbols in $w_1, ..., w_h$ is less than some fixed integer k.

Then we can easily construct an apg $G' = (N, T, S, Q)$, where

$$Q = \{(A_1, ..., A_{i-1}, A_i, A_{i+1}, ..., A_n) \to (A_1, ..., A_{i-1}, y_i, A_{i+1}, ..., A_n) \mid$$
$$1 \leqslant n < k,\ 1 \leqslant i \leqslant n \text{ and } (A_i \to y_i) \in P,\ A_j \in N\ (j = 1,..., n)\}.$$

Then $\Lambda(G') = X$.

Nonexpansive cfl are properly contained in apl, because they are a subfamily of cfl and hence do not contain the language $\{a^ib^ic^i \mid i \geqslant 0\}$ (cf. example in Def. 2.1).

3. Main Result

In this section, we present the main theorem of the paper and its proof.

Theorem 3.1. *The family of languages generated by 2ft is equal to the family of apl.*

The proof of the theorem follows with the aid of the constructions and lemmas of this section.

Construction 3.1. Let $G = (N, T, S, P)$ be an apg in indexed form. We construct a left 2ft F with generated language $A(G)$. The idea is that the set of productions P becomes the input alphabet of F. If $\pi_1, ..., \pi_k$ is the sequence of productions in the derivation of a word z by G, then given input word $\pi_1 \cdots \pi_k$, F generates output word z. To describe informally how this is accomplished, consider the word $w_1 A_1 \cdots w_n A_n w_{n+1}$ obtained after the hth production, where the w_i are terminal words and the A_i are nonterminal symbols. Then $z = w_1 x_1 \cdots w_n x_n w_{n+1}$ (cf. Fig. 1). The

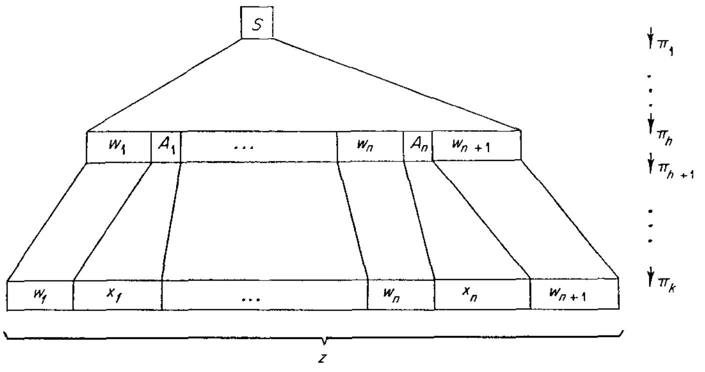

Figure 1

parts $w_1, ..., w_{n+1}$ of the output are produced by F while the pointer is within the initial input subword $\pi_1 \cdots \pi_h$; the parts $x_1, ..., x_n$ contributed by $A_1, ..., A_n$, respectively, are produced while the pointer is within $\pi_{h+1} \cdots \pi_k$. The pointer crosses the $(h + 1)$th input symbol π_{h+1} twice for each A_i—once to the right from state A_i and once to the left to state A_i^L.

More formally, let $F = (M, S, \{S^L\})$ be a left 2ft where

(i) $K_F = \{A, A^L, A^E \mid A \in N\}$, $Y_F = T$, $X_F = P$.

(ii) In the following, let $\pi \in P$ be of the form $(A_1, ..., A_n) \rightarrow (y_1, ..., y_n)$, where

$y_i = u_{i,1}B_{i,1} \cdots u_{i,m_i}B_{i,m_i}u_{i,m_i+1}$ and $B_{i,j} \in N$, $u_{i,j} \in T^*$, $m_i \geqslant 0$. Then define

$$M_1 = \{R(A_i, \pi, B_{i,1}, u_{i,1}), L(B_{i,m_i}^L, \pi, A_i^L, u_{i,m_i+1}) \mid \pi \in P, y_i \notin T^*\},$$
$$M_2 = \{(B_{i,j}^L, \pi\uparrow, B_{i,j+1}, \pi\uparrow, u_{i,j+1}) \mid \pi \in P, y_i \notin T^*(N \cup \{\lambda\})\, T^*\},$$
$$M_3 = \{R(A_i, \pi, A_i^E, y_i), L(A_i^E, \pi, A_i^L, \lambda) \mid \pi \in P, y_i \in T^*\}.$$

Let $M = M_1 \cup M_2 \cup M_3$.

The resulting automaton is well defined and deterministic because the grammar G was said to be in the indexed form.

Lemmas 3.1 and 3.2 give the equivalence of the corresponding families of languages.

LEMMA 3.1. $\Lambda(G) \subset \Gamma(F)$.

Proof. The Lemma is proved by the aid of the statement:

(1) If $w_1A_1 \cdots w_nA_nw_{n+1} \Rightarrow_{\pi_k} \cdots \Rightarrow_{\pi_1} z \in T^*$, where $\pi_j \in P$, $k \geqslant 0$, $w_i \in T^*$, $A_i \in N$, then $z = w_1x_1 \cdots w_nx_nw_{n+1}$ and $(A_i, \uparrow\pi_k \cdots \pi_1, \lambda) \overset{*}{\Rightarrow}_M (A_i^L, \uparrow\pi_k \cdots \pi_1, x_i)$.

This statement can be proved by induction on k:

For $k = 0$, (1) is trivially true.

Suppose (1) holds for $k - 1 \geqslant 0$. Let π_k be of the form $(A_1, \ldots, A_n) \to (y_1, \ldots, y_n)$. If $y_i \in T^*$, then $x_i = y_i$ and

$$(A_i, \uparrow\pi_k \cdots \pi_1, \lambda) \vdash_{M_3} (A_i^E, \pi_k\uparrow \cdots \pi_1, y_i) \vdash_{M_3} (A_i^L, \uparrow\pi_k \cdots \pi_1, y_i).$$

If $y_i \notin T^*$ and $y_i = u_{i,1}B_{i,1} \cdots u_{i,m_i}B_{i,m_i}u_{i,m_i+1}$ then $x_i = u_{i,1}x_{i,1} \cdots u_{i,m}x_{i,m_i}u_{i,m_i+1}$, and by the induction assumption and Constr. 3.1,

$$\begin{aligned}
&(A_i, \uparrow\pi_k \cdots \pi_1, \lambda) \vdash_{M_1} (B_{i,1}, \pi_k\uparrow \cdots \pi_1, u_{i,1}) \\
&\overset{*}{\vdash}_M (B_{i,1}^L, \pi_k\uparrow \cdots \pi_1, u_{i,1}x_{i,1}) \\
&\vdash_{M_2} (B_{i,2}, \pi_k\uparrow \cdots \pi_1, u_{i,1}x_{i,1}u_{i,2}) \overset{*}{\vdash}_M \cdots \\
&\overset{*}{\vdash}_M (B_{i,m_i}^L, \pi_k\uparrow \cdots \pi_1, u_{i,1}x_{i,1}u_{i,2} \cdots x_{i,m_i}) \\
&\vdash_{M_1} (A_i^L, \uparrow\pi_k \cdots \pi_1, u_{i,1}x_{i,1}u_{i,2} \cdots x_{i,m_i}u_{i,m_i+1})
\end{aligned}$$

and (1) is proved.

For the proof of the Lemma, suppose $z \in \Lambda(G)$, then $S \overset{*}{\Rightarrow}_P z \in T^*$. Hence by (1), there exists $v \in X_F^*$ such that $(S, \uparrow v, \lambda) \overset{*}{\vdash}_M (S^L, \uparrow v, z)$ and hence $z \in \Gamma(F)$.

LEMMA 3.2. $\Lambda(G) \supset \Gamma(F)$.

Proof. First verify the following statement:

(2) If $xy \in \Sigma(F)$, then $L(x \uparrow y) = w_1 A_1 A_1^L \cdots w_k A_k A_k^L w_{k+1}$ where $k \geqslant 0$, $w_i \in T^*$ and $A_i \in N$.

This statement is proved by induction on $|y|$:

For $|y| = 0, L(x\uparrow) \in T^*$ and (2) holds.

For the induction step, suppose that $(S, \uparrow xay, \lambda) = U_0 \vdash_{\mu_0} \cdots \vdash_{\mu_{n-1}} U_n = (S^L, \uparrow xay, z)$ where $a \in X_F$, each $\mu_i \in M$, and (2) holds with xa in the role of x. Recalling Def. 1.5, we note that $L(xa \uparrow y) = P_0(xa \uparrow y) \cdots P_{n-1}(xa \uparrow y)$ where $P_i(xa \uparrow y)$ is either the ith output (if the pointer is within xa during the ith move), the ith state (if the pointer is within xa during the $(i-1)$th move and the ith move takes the pointer outside of xa), the $(i+1)$th state (if the pointer moves right to the boundary of xa during the ith move and stays within xa during the $(i+1)$th move), or λ (otherwise). Similarly, $L(x \uparrow ay) = P_0(x \uparrow ay) \cdots P_{n-1}(x \uparrow ay)$. We shall split $L(xa \uparrow y)$ into subwords in such a way that each $P_i(xa \uparrow y)$ is a part of one and only of them. We know that each $P_i(xa \uparrow y)$ contains at most one occurrence of a state symbol and hence $L(xa \uparrow y)$ can be split into the subwords in such a way that each one belongs to one and only one of the following cases:

(i) $P_i(xa \uparrow y) \cdots P_{i+h}(xa \uparrow y) = v_1 B_1 B_1^L \cdots v_m B_m B_m^L v_{m+1}$, where $B_i, B_i^L \in K_F$, $v_i \in T^*, m \geqslant 1, (B_j^L, a\uparrow, B_{j+1}, a\uparrow, v_{j+1}) \in M_2$ and $R(A, a, B_1, v_1), L(B_m^L, a, C, v_{m+1}) \in M_1$ for some $A, C \in K_F$. According to Constr. 3.1 this is possible only if there is a production of the type $(..., A,...) \to (..., v_1 B_1 \cdots v_m B_m v_{m+1},...)$ in P. Then $C = A^L$ and $P_i(x \uparrow ay) \cdots P_{i+h}(x \uparrow ay) = AA^L$.

(ii) $P_i(xa \uparrow y) \cdot P_{i+1}(xa \uparrow y) \in T^*$ and $U_{i+1} = (A^E, xa \uparrow y, w)$. This is possible only if there is a production of the form $(..., A,...) \to (..., P_i(xa \uparrow y) \cdot P_{i+1}(xa \uparrow y),...)$ in P. Then $R(A, a, A^E, P_i(xa \uparrow y) \cdot P_{i+1}(xa \uparrow y)), L(A^E, a, A^L, \lambda) \in M_3$. Therefore $P_i(x \uparrow ay) \cdot P_{i+1}(x \uparrow ay) = AA^L$.

(iii) $P_i(xa \uparrow y) \cdots P_{i+h}(xa \uparrow y) \in T^*$ and for every j, $i \leqslant j \leqslant i+h$, $U_j = (p_j, x_1 \uparrow x_2 ay, u_j)$. Then $P_i(x \uparrow ay) \cdots P_{i+h}(x \uparrow ay) = P_i(xa \uparrow y) \cdots P_{i+h}(xa \uparrow y)$, and the proof of (2) is completed.

The Lemma is proved with the aid of the following statement:

(3) Let $L(x \uparrow y) = w_1 A_1 A_1^L \cdots w_k A_k A_k^L w_{k+1}$. Then $w_1 A_1 \cdots w_k A_k w_{k+1} \in \Lambda^N(G)$.

This statement if proved by induction on $|x|$:

From Constr. 3.1. and statement (2) it follows that $L(\uparrow y) = SS^L$, and we know that $S \in \Lambda^N(G)$.

For the induction step, suppose (3) holds for $L(x \uparrow ay) = w_1 A_1 A^L \cdots w_k A_k A_k^L w_{k+1}$ where $a \in X_F$. For $k = 0$ (that means $L(x \uparrow ay \in T^*)$ the induction step is trivial.

If $k \geqslant 1$, then we claim there exist $\pi \in P$ such that $\pi = a$ and

$$w_1 A_1 \cdots w_k A_k w_{k+1} \Rightarrow_\pi w_1 y_1 \cdots w_k y_k w_{k+1}.$$

For suppose there is no such π. Then from Def. 2.2, condition (ii) and Constr. 3.1, for every B, z, $R(A_1, a, B, z) \notin M$ and consequently $L(x \uparrow ay) \neq w_1 A_1 A_1{}^L \cdots w_k A_k A_k{}^L w_{k+1}$ which contradicts the assumption.

Suppose again that π is of the form $(A_1, ..., A_k) \to (y_1, ..., y_k)$. Then let $L(xa \uparrow y) = w_1 v_1 \cdots w_k v_k w_{k+1}$, where $v_i \in (T \cup K_F)^*$. (cf. Observ. 1.1.) Then we have for each $i = 1, ..., k$ the following possibilities:

(i) $v_i \in T^*$. Then from Constr. 3.1 this is true only if $R(A_i, a, A_i{}^E, v_i)$, $L(A_i{}^E, a, A_i{}^L, \lambda) \in M_3$ and this in turn is true only if $y_i = v_i$.

(ii) $v_i \notin T^*$. Then by (2) and Observ. 1.1, $v_i = u_1 B_1 B_1{}^L \cdots u_m B_m B_m{}^L u_{m+1}$. In view of the form of M, this is true only if $R(A_i, a, B_1, u_1), L(B_m{}^L, a, A_i{}^L, u_{m+1}) \in M_1$ and for every j, $1 \leqslant j \leqslant m$, $(B_j{}^L, a\uparrow, B_{j+1}, a\uparrow, u_{j+1}) \in M_2$. But this is true only if $y_i = u_1 B_1 \cdots u_m B_m u_{m+1}$.

Thus the proof of (3) is completed.

For the proof of the Lemma, suppose $w \in \Gamma(F)$. Then for some x,

$$(S, \uparrow x, \lambda) \overset{*}{\vdash}_F (S^L, \uparrow x, w).$$

Hence $L(x\uparrow) = w$ and by (3), $w \in \Lambda^N(G)$ and therefore $w \in \Lambda(G)$.

CONSTRUCTION 3.2. Let $E = (M, s, \{r\})$ be a normalized left 2ft. We want to construct an apg G_E such that for each $xay \in \Sigma(E)$, $L(x \uparrow ay)$ is "simulated" by a word in $\Lambda^N(G_E)$. In fact, the grammar will "guess" in each step on the basis of $L(x \uparrow ay)$ (a is a symbol) what $L(xa \uparrow y)$ is. There are always only finitely many possibilities. If any of the guesses is wrong, then no word is derived.

For the purpose of convenient notation, nonterminals of G_E will be couples of state symbols. The terminals will be the output symbols of E.

First for each $p, q \in K_E$, and $b \in X_E$ construct the following sets:

$$\begin{aligned} V(p, q, b) &= \{y \mid (p, \uparrow b, \lambda) \overset{*}{\vdash}_M (q, \uparrow b, y) \text{ and for no } U, (q, \uparrow b, y) \vdash_M U\}, \\ W(p, q, b) &= \{\{y_1(p_1, p_2) y_3(p_3, p_4) \cdots (p_{2n-1}, p_{2n}) y_{2n+1} \mid (p, \uparrow b, \lambda) \overset{*}{\vdash}_M (p_1, b\uparrow, y_1), \\ &\qquad (p_{2i}, b\uparrow, \lambda) \overset{*}{\vdash}_M (p_{2i+1}, b\uparrow, y_{2i+1}) \qquad (i = 1, ..., n-1), \\ &\qquad (p_{2n}, b\uparrow, \lambda) \overset{*}{\vdash}_M (q, \uparrow \mathrm{b}, y_{2n+1}), i \neq j \text{ implies } p_i \neq p_j \text{ and for all } U, \\ &\qquad \text{neither } (p_{2i+1}, b\uparrow, y_{2i+1}) \vdash_M U, \text{ nor } (q, \uparrow b, y_{2n+1}) \vdash_M U\}. \end{aligned}$$

Then $V(p, q, b)$ and $W(p, q, b)$ are finite.

Let P_b denote the set of all productions of the form

$$((p_1, p_2), (p_3, p_4), ..., (p_{2n-1}, p_{2n})) \to (z_1, z_2, ..., z_n)$$

where $i \neq j$ implies $p_i \neq p_j$ and $z_i \in V(p_{2i-1}, p_{2i}, b) \cup W(p_{2i-1}, p_{2i}, b)$. Then $G_E = ((K_E \times K_E), Y_E, (s, r), P)$ is an apg, where $P = \bigcup_{b \in X_E} P_b$.

This completes the construction. Remaining lemmas of this section give the equality of the languages generated by E and G_E.

LEMMA 3.3. $\Lambda(G_E) \subset \Gamma(E)$.

Proof. The Lemma is proved with the help of the statement:

(4) If $w_0(q_1, q_1')\, w_1(q_2, q_2') \cdots (q_h, q_h')\, w_h \Rightarrow_P{}^k w_0 v_1 w_1 v_2 \cdots v_h w_h \in Y_E{}^*$, then there exists $u \in X_E{}^*$ such that $(q_i, \uparrow u, \lambda) \overset{*}{\vdash}_M (q_i', \uparrow u, v_i)$ $(i = 1,\ldots, h)$.

Proof is by induction on k:

For $k = 0$ the statement (4) is trivially true.

Suppose now that (4) holds for $k - 1$. Then let

$$w_0(q_1, q_1')\, w_1(q_2, q_2')\, w_2 \cdots (q_h, q_h')\, w_h \Rightarrow_\pi w_0 z_1 w_1 z_2 w_2 \cdots z_h w_h$$
$$\Rightarrow_P^{k-1} w_0 v_1 w_1 v_2 \cdots v_h w_h = w \in Y_E{}^*.$$

Consider any z_i.

(i) $z_i \in Y_E{}^*$. Then $z_i \in V(q_i, q_i', b)$ where $\pi \in P_b$, hence

$$(q_i, \uparrow b, \lambda) \overset{*}{\vdash}_M (q_i', \uparrow b, z_i).$$

(ii) $z_i \notin Y_E{}^*$. Then $z_i = y_1(p_1, p_2)\, y_3(p_3, p_4) \cdots (p_{2n-1}, p_{2n})\, y_{2n+1} \in W(q_i, q_i', b)$ where $\pi \in P_b$ and $v_i = y_1 v_2' y_3 v_4' \cdots v_{2n}' y_{2n+1}$. By the induction assumption, there exists u such that $(p_{2j-1}, \uparrow u, \lambda) \overset{*}{\vdash}_M (p_{2j}, \uparrow u, v_{2j}')$ $(j = 1,\ldots, n)$. From the construction of $W(q_i, q_i', b)$ we have

$$(q_i, \uparrow b, \lambda) \overset{*}{\vdash}_M (p_1, b\uparrow, y_1),$$
$$(p_{2j}, b\uparrow, \lambda) \overset{*}{\vdash}_M (p_{2j+1}, b\uparrow, y_{2j+1})$$

and

$$(p_{2n}, b\uparrow, \lambda) \overset{*}{\vdash}_M (q_i', \uparrow b, y_{2n+1}).$$

Hence combination of both gives

$$(q_i, \uparrow bu, \lambda) \overset{*}{\vdash}_M (p_1, b \uparrow u, y_1) \overset{*}{\vdash}_M (p_2, b \uparrow u, y_1 v_2')$$
$$\overset{*}{\vdash}_M (p_3, b \uparrow u, y_1 v_2' y_3) \overset{*}{\vdash}_M \cdots \overset{*}{\vdash}_M (p_{2n}, b \uparrow u, y_1 v_2' y_3 \cdots v_{2n}')$$
$$\overset{*}{\vdash}_M (q_i', \uparrow b\, u, y_1 v_2' y_3 \cdots v_{2n}' y_{2n+1})$$

which completes the proof of (4).

For the proof of the Lemma, $w \in \Lambda(G_E)$ implies $(s, r) \overset{*}{\Rightarrow}_P w \in Y_E{}^*$. Then from (4) there exists $u \in X_E{}^*$ such that $(s, \uparrow u, \lambda) \overset{*}{\vdash}_M (r, \uparrow u, w)$ and hence $w \in \Gamma(E)$.

LEMMA 3.4. $\Lambda(G_E) \supset \Gamma(E)$.

Proof. The Lemma is proved with the help of the statement:

(5) If $L(x \uparrow y) = w_0 p_1 p_1' w_1 p_2 p_2' \cdots p_h p_h' w_h$ where p_i and $p_i' \in K_E$, $w_i \in Y_E^*$, then $w_0(p_1, p_1') w_1(p_2, p_2') \cdots (p_h, p_h') w_h \in \Lambda^N(G_E)$.

Proof is by induction on $|x|$:

If $|x| = 0$, then $L(\uparrow y) = sr$ and (5) holds.

Let $a \in X_E$ and $L(x \uparrow ay) = w_0 p_1 p_1' w_1 p_2 p_2' \cdots p_h p_h' w_h$, then from Observ. 1.1, $L(xa \uparrow y) = w_0 v_1 w_1 v_2 \cdots v_h w_h$, where $v_i = w_{i,0} q_1 q_1' w_{i,1} q_2 q_2' \cdots q_m q_m' w_{i,m}$ with $m \geqslant 0$ $(i = 1,..., h)$.

If $m \geqslant 1$, then $(p_i, \uparrow a, \lambda) \overset{*}{\vdash}_M (q_1, a\uparrow, w_{i,0})$ and $(q_j', a\uparrow, \lambda) \overset{*}{\vdash}_M (q_{j+1}, a\uparrow, w_{i,j})$ $(j = 1,..., m)$, $(q_m', a\uparrow, \lambda) \overset{*}{\vdash}_M (p_i', \uparrow a, w_{i,m})$ and for all configurations U, neither $(q_j, a\uparrow, w_{i,j-1}) \vdash_M U$, nor $(p_i', \uparrow a, w_{i,m}) \vdash_M U$.

If $m = 0$, then $(p_i, \uparrow a, \lambda) \overset{*}{\vdash}_M (p_i', \uparrow a, v_i)$, where for no configuration U, $(p_i', \uparrow a, v_i) \vdash_M U$.

Therefore $((p_1, p_1'), (p_2, p_2'),..., (p_h, p_h')) \to (z_1, z_2,..., z_h)$ is a production of P, where each $z_i = w_{i,0}(q_1, q_1') w_{i,1}(q_2, q_2') \cdots (q_m, q_m') w_{i,m}$. Then

$$w_0(p_1, p_1') w_1(p_2, p_2') \cdots (p_h, p_h') w_h \Rightarrow_P w_0 z_1 w_1 z_2 \cdots z_h w_h \in \Lambda^N(G_E),$$

which completes the proof of (5).

For the proof of the Lemma, take $w \in \Gamma(E)$. Then there exists $u \in X_E^*$ such that $(s, \uparrow u, \lambda) \overset{*}{\vdash}_M (r, \uparrow u, w)$, hence $L(u\uparrow) = w$. Then by (5) we have $w \in \Lambda^N(G_E)$ and, therefore, $w \in \Lambda(G_E)$.

Proof of Theorem 3.1. In Constr. 3.1 and Lemmas 3.1 and 3.2 we found for every apg in indexed form a left 2ft with equal generated language. Considering Lemma 2.1 there is for every apg a left 2ft with equal generated language.

Similarly in Constr. 3.2 and Lemmas 3.3 and 3.4 we found for every normalized left 2ft an apg with equal generated language. Hence by Lemma 1.2, there is for every left 2ft an apg with equal generated language.

This establishes the equality of the family of apl with the family of languages generated by left 2ft. Finally by Lemma 1.1 we get Theorem 3.1.

4. Closure Properties of apl

The main purpose of this section is to establish that the family of apl is a full AFL closed under substitution. From the other most important closure properties, it remains open whether the family of apl forms an abstract family of two-way deterministic languages [1] and whether it is closed under two-way gsm mapping. Some closure properties are also investigated in [2].

We shall start with the following definitions.

DEFINITION 4.1. Let Y be a family of languages. A *substition* by Y is any operation τ whose domain consists of words and sets over a finite alphabet X, such that $\tau(\lambda) = \{\lambda\}$, $\tau(a) \in Y$ for all $a \in X$, $\tau(a_1 \cdots a_k) = \tau(a_1) \cdots \tau(a_k)$ for all $k \geqslant 1$ and $a_1, \ldots, a_k \in X$, and $\tau(L) = \bigcup_{x \in L} \tau(x)$ for all languages L over X.

We say Y is *closed under substitution* if $L \in Y$ and τ is a substitution by Y implies $\tau(L) \in Y$.

DEFINITION 4.2.
A set R is *regular* iff for some alphabet A, R is a member of the least class C such that:

(i) every finite subset of A belongs to C,
(ii) if $X, Y \in C$, then $X \cup Y \in C$,
(iii) if $X, Y \in C$, then $X \cdot Y \in C$,
(iv) if $X \in C$, then $X^* \in C$.

DEFINITION 4.3. A family of languages is a *full AFL* if it contains a nonempty language and is closed under union, $*$, concatenation, intersection with any regular set, homomorphism and inverse homomorphism.

THEOREM 4.1. *The family of apl is a full AFL closed under substitution.*

Proof. Ehrich and Yau [2] obtained closure under substitution for the family of languages generated by 2ft. Hence in view of Theorem 3.1, apl are closed under substitution. For a direct "grammar" proof, see [14]. Also, the family of apl is closed under intersection with any regular set [2], [14]. Moreover it is obvious that a^* is an apl for every symbol a. Hence by a theorem of [9], the family of apl is a full AFL.

5. SOME PROPERTIES OF 2nft

In this section, we shall investigate some properties of 2nft.

We begin by defining the notion of checking automaton (abbreviated ca). For this purpose we use the alphabets K, X and Y and the symbol $\uparrow$ introduced in Section 1. In connection with checking automata, however, we refer to the symbols of Y as *stack symbols*.

DEFINITION 5.1. A *ca-configuration* is any triple $(p, x, y \uparrow y')$ where $p \in K$, $x \in X^*$, and $yy' \in Y^*$. A *ca-move-rule* is a 5-tuple $\mu = (p, q, x, a \uparrow b, a_1 \uparrow b_1)$ where p and $q \in K$, $x \in X^*$, a and $b \in Y \cup \{\lambda\}$, and $ab = a_1 b_1$. A *checking automaton* (abbreviated ca) is a one-way nonerasing stack automaton which, once it enters its stack, never writes on it again. More formally it is a triple $C = (M, s, K_1)$ where M is a finite set of ca-move-rules, $s \in K$ and K_1 is a finite set of states. We denote by K_C, X_C and Y_C the finite sets of states, input symbols and stack symbols, respectively, that are explicitly mentioned in the description of C. For fixed ca $C = (M, s, K_1)$, for any ca-move-rule $\mu \in M$ and ca-configurations U, V, $U \vdash_\mu V$ iff there exist $w \in X_C{}^*$, $yy' \in Y_C{}^*$ such that $U = (p, xw, ya \uparrow by')$ and $V = (q, w, ya_1 \uparrow b_1 y')$. $U \vdash_M V$ iff there exists $\mu \in M$ such that $U \vdash_\mu V$. $\overset{*}{\vdash}_M$ is the transitive and reflexive closure of $\vdash_M$.

x is an *accepted word* of C iff there exists $y \in Y_C{}^*$ such that $(s, x, \uparrow y) \overset{*}{\vdash}_M (q, \lambda, \uparrow y)$ where $q \in K_1$. The *accepted language* of ca C (abbreviated cal) is the set $\Sigma(C)$ of all words accepted by C.

Our notation differs from the notation of [8], which uses the reading head instead of the pointer. All possible actions of a ca with the reading head can be described in terms of ca-move-rules as $(p, q, x, \uparrow b, a_1 \uparrow b_1)$ and $\{(p, q, x, a \uparrow b, \uparrow ab) \mid a \in Y_C\}$, where p, q are states, x is an input word, and b is a stack symbol. Hence the reading head does not add any additional power.

To show inverse, suppose we have a ca C in our notation with the pointer. Consider a ca-move-rule of the type $(p, q, x, \uparrow b, a_1 \uparrow b_1)$. Then this ca-move-rule can be simulated by the following action of the reading head: In the state p, reading the letter b (or any letter if $b = \lambda$), change the state p to the state q, erase x on the input, and move the reading head $| a_1 |$ letters to the right.

Consider a ca-move-rule of the type $(p, q, x, a \uparrow b, a_1 \uparrow b_1)$, where $a \neq \lambda$. Then the ca-move-rule can be replaced by the following two actions of the reading head:

(i) In the state p, reading the letter b (or any letter if $b = \lambda$), change the state p to the state (p, b), and move the reading head one letter to the left.

(ii) In the state (p, b), reading the letter a, change the state (p, b) to the state q, erase x on the input, and move the reading head $| a_1 |$ letters to the right.

These two intuitively described simulations show, that the pointer notation and the reading head notation are equal.

THEOREM 5.1. *The family of languages generated by 2nft is the family of* cal.

Proof. To simulate a 2nft, the ca guesses the input of the 2nft and places it on its stack. Then during the simulation, the ca treats its stack and input in the same way as the original 2nft treats its input and output, respectively. After finishing this simulation, the ca ends on the right end of the stack, hence new move-rules have to be added which return the pointer to the left end of the stack.

More formally, let $F = (M, s, K_1)$ be a 2nft. Then construct a ca $C = (M', s, \{q\})$ in the following way: $X_C = Y_F$, $Y_C = X_F \cup \{\$\}$, $K_C = K_F \cup \{q\}$. For each $\mu = (p, a \uparrow b, p', a_1 \uparrow b_1, x) \in M$, let $\mu' = (p, p', x, a \uparrow b, a_1 \uparrow b_1)$. Then let $M' = \{\mu' \mid \mu \in M\} \cup \{(p, q, \lambda, \uparrow\$, \uparrow\$) \mid p \in K_1\} \cup \{(q, q, \lambda, a\uparrow, \uparrow a) \mid a \in Y_C\}$.

For the constructed ca C, we can show $(p, \uparrow xy, \lambda) \vdash_M^k (p', x \uparrow y, z)$ iff

$$(p, z, \uparrow xy\$) \vdash_{M'}^k (p', \lambda, x \uparrow y\$)$$

by induction on k. Hence $w \in \Gamma(F)$ iff for some $u \in X_F^*$, $(p, \uparrow u, \lambda) \overset{*}{\vdash}_M (r, u\uparrow, w)$ where $r \in K_1$. Then by the statement above this is true iff

$$(p, w, \uparrow u\$) \overset{*}{\vdash}_{M'} (r, \lambda, u \uparrow \$) \vdash_{M'} (q, \lambda, u \uparrow \$) \overset{*}{\vdash}_{M'} (q, \lambda, \uparrow u\$)$$

and this holds iff $w \in \Sigma(C)$.

Hence $\Sigma(C) = \Gamma(F)$. Thus every language generated by a 2nft is accepted by a ca.

To prove inverse, let $C = (M, s, K_1)$ be a ca. Then construct a 2nft $F = (M', s, \{r\})$ where $X_F = Y_C \cup \{¢\}$, $Y_F = X_C$, $K_F = K_C \cup \{r\}$. For each

$$\mu = (p, q, x, a \uparrow b, a_1 \uparrow b_1) \in M$$

define $\mu' = (p, a \uparrow b, q, a_1 \uparrow b_1, x)$. Then let

$$M' = \{\mu' \mid \mu \in M\} \cup \{(s, \uparrow ¢, s, ¢\uparrow, \lambda)\} \cup \{(q, ¢\uparrow, r, ¢\uparrow, \lambda) \mid q \in K_1\}$$
$$\cup \{(r, \uparrow a, r, a\uparrow, \lambda) \mid a \in X_F\}.$$

Again the simulation is straightforward and similarly as above, we have $\Gamma(F) = \Sigma(C)$.

THEOREM 5.2. *The family of languages generated by 2ft is properly contained in the family of languages generated by 2nft.*

This Theorem is proved with the aid of Lemma 5.1.

DEFINITION 5.2.

A *left-linear context-free grammar* (abbreviated llg) is a cfg $G = (N, T, S, P)$ where P is a finite set of productions of the form

$$A \to z \quad \text{with} \quad A \in N \quad \text{and} \quad z \in (\{\lambda\} \cup N) \cdot T^*.$$

LEMMA 5.1. *If $L \subset a^*$ is an apl, then L is regular.*

Proof. Let $G = (N, \{a\}, S, P)$ be an apg which generates L. Let $l(\pi)$ and $r(\pi)$ be defined in the same way as in the proof of Lemma 2.1. Then construct a llg $G' = (N', \{a\}, S, Q)$ where $N' = \{l(\pi), r(\pi) \mid \pi \in P\}$. For every production

$$\pi = (A_1, ..., A_n) \to (y_1, ..., y_n) \in P$$

let $\pi' = (A_1 \cdots A_n) \to (B_1 \cdots B_m) z_1 \cdots z_m z_{m+1}$ where $y_1 \cdots y_n = z_1 B_1 \cdots z_m B_m z_{m+1}$ and $B_i \in N$, $z_i \in a^*$. Then let $Q = \{\pi' \mid \pi \in P\}$.

We can show the following statement by induction on k: $S \Rightarrow_P^k w_1 A_1 \cdots w_m A_m w_{m+1}$

with $w_i \in a^*$, $A_i \in N$ iff $S \Rightarrow_Q^k (A_1 \cdots A_m) w_1 \cdots w_m w_{m+1}$. Then $w \in \Lambda(G)$ iff $S \overset{*}{\Rightarrow}_P w \in a^*$. Then by the statement above, this is true iff $S \overset{*}{\Rightarrow}_Q w \in a^*$, and this holds iff $w \in \Lambda(G')$. Hence $\Lambda(G) = \Lambda(G')$.

It is known [3] that the languages generated by the llg are regular, and therefore L is regular.

Proof of Theorem 5.2. The inclusion is obvious. For the proper inclusion, consider the language $L = \{a^n \mid n \text{ is not a prime}\}$. Now L is a cal [8]. Hence by Theorem 5.1, L is generated by some 2nft. Moreover L is not regular [11], and therefore by Lemma 5.1, it is not an apl. Then by Theorem 3.1, L is not generated by any 2ft.

This theorem is also proved in a different way in [2].

Acknowledgment

I wish to thank my advisor Professor Gene F. Rose for help and encouragement. I also wish to thank Professor William Rounds.

The research was partially supported by a research grant of the Case Western Reserve University chapter of the Society of Sigma *Xi*.

References

1. A. V. Aho and J. D. Ullman, A characterization of two-way deterministic classes of languages, *J. Comput. System Sci.* **4** (1970), 523–538.
2. R. Ehrich and S. S. Yau, Two-way sequential transducers and stack automata, *Inform. Control* **18** (1971), 404–446.
3. S. Ginsburg, "The Mathematical Theory of Context-Free Languages," McGraw-Hill, New York, 1966.
4. S. Ginsburg and S. A. Greibach, Abstract families of languages, *Mem. Amer. Math. Soc.* **87** (1969), 1–32.
5. S. Ginsburg, S. A. Greibach, and M. A. Harrison, One-way stack automata, *J. Assoc. Comput. Mach.* **14** (1967), 389–418.
6. S. Ginsburg and G. F. Rose, Preservation of languages by transducers, *Inform. Control* **9** (1966), 153–176.
7. S. Ginsburg and E. H. Spanier, Derivation-bounded languages, *in* "IEEE Conference Record of 1968 Ninth Annual Symposium on Switching and Automata Theory," pp. 306–314. Institute of Electrical and Electronics Engineers, New York, 1968.
8. S. A. Greibach, Checking automata and one-way stack languages, *J. Comput. System Sci.* **3** (1969), 196–217.
9. S. A. Greibach and J. Hopcroft, Independence of AFL operations, *Mem. Amer. Math. Soc.* **87** (1969), 33–40.
10. S. A. Greibach and J. Hopcroft, Scattered context grammars, *J. Comput. System Sci.* **3** (1969), 233–247.
11. J. Hartmanis and H. Shank, On the recognition of primes by automata, *J. Assoc. Comput. Mach.* **15** (1968), 382–389.

12. J. Hopcroft and J. Ullman, "Formal Languages and their Relation to Automata," Addison-Wesley, Reading, MA, 1969.
13. B. O. Nash and R. S. Cohen, Parallel leveled grammars, *in* "IEEE Conference Record of 1969 Tenth Annual Symposium on Switching and Automata Theory," pp. 263–276. Institute of Electrical and Electronics Engineers, New York, 1969.
14. V. Rajlich, Absolutely parallel grammars and two-way finite-state transducers, Ph.D. Dissertation, Case Western Reserve University, 1971.
15. G. F. Rose, Personal communication.