ω-COMPUTATIONS ON TURING MACHINES

by

Rina S. Cohen* and Arie Y. Gold**

Technical Report #76

September 1976

* Department of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel.

** Faculty of Mathematics, Technion-Israel Institute of Technology, Haifa, Israel.
ABSTRACT

The paper develops the theory of Turing machines as recognizers of infinite (ω-type) input tapes. Various models of ω-type Turing acceptors are considered, varying mainly in their mechanism for recognizing ω-tapes. A comparative study of the models is made. It is shown that regardless of the ω-recognition model considered, non-deterministic ω-Turing acceptors are strictly more powerful than their deterministic counterparts. Canonical forms are obtained for each of the ω-Turing acceptor models. The corresponding families of ω-sets are studied; normal forms and algebraic characterizations are derived for each family.
1. INTRODUCTION

Infinite computations on Turing machines can be thought of as modelling discrete time continuous dynamical processes. The recognition, or generation, of infinite sequences by Turing machines have been studied before in the literature (e.g. [Ha&St] [Lan]). In this paper we develop the theory of Turing machines as recognizers of $\omega$-type languages.

Several distinct models of (deterministic or non-deterministic) $\omega$-type Turing acceptors are defined, varying mainly in their mechanism (so called "mode") for accepting or rejecting infinite strings. Similar models have been considered previously w.r.t. other types of $\omega$-acceptors, particularly, $\omega$-type finite-state machines ([Bu],[Mc],[Lan], [Cho], [Hos]) and push-down machines ([Co&Go1-3], [Lin]). A comparative study of these models for Turing machines is made in this paper. First, it is established that w.r.t. each of the models, non-deterministic $\omega$-Turing acceptors are strictly more powerful than the deterministic ones. For deterministic $\omega$-Turing acceptors it is shown that the limitations in recognition power of some of the models stem from the way the $\omega$-recognition mode is defined, rather than from the restricted data access of the machines. On the other hand, w.r.t. non-deterministic $\omega$-Turing acceptors, all $\omega$-recognition modes turn out to be of equivalent recognition power. Canonical forms are obtained for each type of $\omega$-Turing acceptor and the corresponding families of $\omega$-languages are studied and characterized.

The paper is divided into six sections. $\omega$-type Turing acceptors ($\omega$-TA's) are introduced in Section 1; Section 2 introduces $\omega$-grammars (i.e. grammars generating $\omega$-languages) and the two notions are related in Section 3, where non-deterministic $\omega$-TA languages are characterized by type 0 $\omega$-grammars, and also by type 1 $\omega$-grammars. Section 4 introduces certain continuous
processes of "folding" the Turing machine tapes, which are later on utilized for constructing \( \omega \)-TA's with desirable properties. Deterministic \( \omega \)-TA's are studied in Section 5; the various \( \omega \)-recognition modes are compared and the corresponding families of \( \omega \)-languages are characterized with the aid of some new algebraic operators, relating them to some well known families of (finite-string) languages. Among other results, it is shown that the Init of a deterministic \( \omega \)-TA language need not be recursively enumerable. In Section 6 it is established that non-deterministic \( \omega \)-TA's are strictly more powerful than the deterministic ones, and that all \( \omega \)-recognition modes are equivalent in so far as non-deterministic \( \omega \)-TA's are concerned. It is also shown that every \( \omega \)-TA can be converted into an equivalent \( \omega \)-TA in which oscillations never occur.
0. PRELIMINARIES

The terminology and notation used in this paper are mostly taken from [H&U].

A finite string (word) over alphabet $\Sigma$ is any sequence $x = \prod_{i=1}^{k} a_i$
where $a_i \in \Sigma$, $i = 1, \ldots, k$, $k = 0, 1, \ldots$. $k = |x|$ is the length of $x$;
$\varepsilon$ denotes the empty string and $\Sigma^*$ denotes the set of all finite strings
over $\Sigma$.

Let $N$ denote the set of natural numbers.

Definition 0.1 For any alphabet $\Sigma$, let $\Sigma^\omega$ denote all infinite
($\omega$-length) strings $\sigma = \prod_{i=1}^{\infty} a_i$, $a_i \in \Sigma$, over $\Sigma$. Any member $\sigma$ of
$\Sigma^\omega$ is called an $\omega$-word or $\omega$-string. An $\omega$-language is any subset of $\Sigma^\omega$.
For any language $L \subseteq \Sigma^*$, define :

$$L^\omega = \{ \sigma \in \Sigma^\omega | \sigma \equiv \prod_{i=1}^{\infty} x_i \text{, where for each } i, \varepsilon \neq x_i \in L \}.$$ 

$L^\omega$ consists of all $\omega$-strings obtained by concatenating words from $L$
in an infinite sequence (note that if $L = \{ \varepsilon \}$ then $L^\omega = \emptyset$).

For any $\sigma \in \Sigma^\omega$, $\sigma = \prod_{i=1}^{\infty} a_i$, $a_i \in \Sigma$, define for each $j \geq 1$,

$$\sigma/j = \prod_{i=1}^{j-1} a_i, \quad j/\sigma = \prod_{i=j+1}^{\infty} a_i, \quad \sigma(j) = a_j \text{ and also } \sigma/0 = \varepsilon, \emptyset/\sigma = \sigma.$$ 

Definition 0.2 [Ra] For sets $A$, $B$ and a mapping $\psi: A \to B$, define

$$\text{Im}(\psi) = \{ b | b \in B, \text{ card}(\psi^{-1}(b)) \geq \omega \},$$

where $\text{card}(D)$ denotes the cardinality of set $D$.

The "$\omega$-Kleene Closure" operator, defined below, turns a family
of (finite string) languages into a family of $\omega$-languages.
Definition 0.3 [Cho] For any family of sets $\mathcal{L}$ over alphabet $\Sigma$, the $\omega$-Kleene Closure of $\mathcal{L}$, denoted $\omega\text{-KC}(\mathcal{L})$, is

$$
\omega\text{-KC}(\mathcal{L}) = \{L \subseteq \Sigma^\omega \mid L = \bigcup_{i=1}^{k} U_i V_i^\omega \text{ for some } U_i, V_i \in \mathcal{L}, i = 1, 2, \ldots, k, k = 1, 2, \ldots \}
$$

1. $\omega$-TYPE TURING ACCEPTORS

Let us first briefly recall the standard definition of a Turing machine with a single semi-infinite tape ([H&U]).

Definition 1.1 A Turing machine (TM) is a 5-tuple $M = (K, \Sigma, \Gamma, \delta, q_0)$ where: $K$ is a finite set of states, $\Sigma$ is a finite input alphabet, $\Gamma$ is a finite tape alphabet s.t. $\Sigma \subseteq \Gamma$, $q_0$ is the initial state, and $\delta$ is a mapping from $K \times \Gamma$ to subsets of $K \times \Gamma \times \{L, R, S\}$. A configuration of $M$ is the 3-tuple $(q, \sigma, i)$, where $q \in K$, $\sigma \in \Sigma^i$ and $i$ is a natural number. The relations $\vdash_M$ and $\vdash_{M^*}$ are defined as usual.

An $m$-tape Turing machine ($m$-TM) ($m \geq 2$) consists of a finite control and $m$ semi-infinite tapes, each with a separate reading head. The moves are defined in the usual way [H&U]. We assume that initially the input appears on the first tape and the other tapes are blank.

In the sequel, unless otherwise specified, by an $m$-TM we shall mean an $m$-tape machine for $m \geq 1$, i.e. a single tape TM ($m=1$) will also be included as a special case.

Definition 1.2 Let $M = (K, \Sigma, \Gamma, \delta, q_0)$ be a TM and let $\sigma \in \Sigma^\omega$. An infinite sequence of configurations $r = \{(q_i, \gamma_i, j_i)\}_{i \geq 1}$ is called a run of $M$ on $\sigma$ iff:

(a) $(q_1, \gamma_1, j_1) = (q_0, \sigma, 1)$ ;
(b) for each \( i \geq 1 \) \( (q_i, \gamma_i, j_i) \xrightarrow{\text{M}} (q_{i+1}, \gamma_{i+1}, j_{i+1}) \);

A run \( r \) is called **complete** if (c) also holds:

(c) \( \forall n \geq 1, \exists k \geq 1 \) s.t. \( j_k > n \);

A complete run is called **oscillating** if, in addition, (d) holds:

(d) \( \exists n_0 \geq 1 \) s.t. \( \forall k \geq 1 \) \( \exists \ell \geq 1 \) s.t. \( j_k > n \) for all \( k > \ell \). Thus a c.n.o. run corresponds to an infinite computation of \( M \) on \( \sigma \) during which \( M \) scans each square on the tape only finitely many times.

A computation which does not correspond to a complete run may be either finite (or blocked), in case the machine halts on \( \sigma \), or else it corresponds to an infinite run for which there exists some \( j_0 > 1 \) s.t. the reading head on input \( \sigma \) never leaves the initial segment \( \sigma/j_0 \).

The notion of c.n.o. run for an \( m \)-TM is defined similarly as for single tape machines. Here a **c.n.o. run** means an infinite computation of the machine on \( \omega \)-input \( \sigma \), during which each square on the first tape (on which the input initially appears) is scanned only finitely many times.

There is no such restriction for the other tapes.

**Definition 1.3** Let \( M = (K, \Sigma, \Gamma, \delta, q_0) \) be a TM. A state \( q_T \in K \) is a **traverse state** iff \( \forall a \in \Gamma, \delta(q_T, a) = \{(q_T, a, R)\} \).

Clearly, if during its computations on \( \omega \)-input \( \sigma \), \( M \) enters a traverse state, then \( M \) will have a c.n.o. run regardless of the contents of the remaining unscanned part of \( \sigma \).
Definition 1.4  Given an \( m \)-TM \( M = (K, \Sigma, \Gamma, \delta, q_0) \), every run \( r \) induces a mapping from \( N \) into \( K \), \( f_r : N \rightarrow K \), where \( f_r(i) = q_i \) - the state entered in the \( i \)-th step of the computation described by run \( r \).

Definition 1.5  An \( m \)-tape \( \omega \)-type Turing Acceptor (\( m-\omega \)-TA), \( m \geq 1 \), is a 6-tuple \( M = (K, \Sigma, \Gamma, \delta, q_0, F) \), where \( M' = (K, \Sigma, \Gamma, \delta, q_0) \) is an \( m \)-TM and \( F \subseteq 2^K \) is the collection of designated state sets. \( M \) will sometimes be denoted by \( (M', F) \).

Notation 1.6  An \( \omega \)-TA with a unique designated state set will be denoted by \( U-\omega \)-TA. In this case we write \( M = (K, \Sigma, \Gamma, \delta, q_0, F) \) where \( F \subseteq K \) is the unique designated set.

We now proceed to define a variety of \( \omega \)-type recognition modes, so called "\( i \)-acceptance" (\( i = 1, 1', 2, 2', 3 \)), in \( \omega \)-type acceptors; these have been first considered in \([\text{Lan}]\) w.r.t. finite state \( \omega \)-automata, and have also been studied w.r.t. \( \omega \)-type pushdown automata (\([\text{Co&Go2,3}]\)).

Definition 1.7  Let \( f : N \rightarrow S \) be an arbitrary mapping and let \( F \subseteq 2^S \).

We say that \( f \) is:

(a) \( 1 \)-accepting w.r.t. \( F \) if \( (\exists H \in F)(\exists t)f(t) \in H \).

(b) \( 1' \)-accepting w.r.t. \( F \) if \( (\exists H \in F)(\forall t)f(t) \in H \).

(c) \( 2 \)-accepting w.r.t. \( F \) if \( (\exists H \in F)\text{Im}(f) \cap H \neq \emptyset \).

(d) \( 2' \)-accepting w.r.t. \( F \) if \( (\exists H \in F)\text{Im}(f) \subseteq H \).

(e) \( 3 \)-accepting w.r.t. \( F \) if \( \text{Im}(f) \in F \).

\( f \) is \( i \)-accepting (\( i = 1, 1', 2, 2', 3 \)) w.r.t. \( F \subseteq S \) if it is \( i \)-accepting w.r.t. \( \{F\} \subseteq 2^S \).
Definition 1.8  Let $M = (M', F)$ be an $m$-$\omega$-TA ($\omega$-TA). For $i=1, 1', 2, 2', 3$, define:

$$T_i(M) = \{ \sigma \in \Sigma^\omega \mid \text{there exists a c.n.o. run } r \text{ of } M \text{ on } \sigma \text{ s.t. the mapping } f_r$$

is $i$-accepting w.r.t. $F \}.$

$T_i(M) \ (i=1, 1', 2, 2', 3)$ is the $\omega$-language $i$-accepted by $M$.

The above definitions are illustrated by the following example.

Example 1.9  Let $\Sigma_1 = \{b, c\}$, $\Sigma = \{a, b, c\}$ and let $L = a^+ \prod a^1 x_i | x_i \in \Sigma_1^+$ and for infinitely many 1's $x_i \in \Sigma_1^+ b_1^+$. Let $M$ be a TM with a set of working states $K_1$ and three special states $q_a', q_b, q_T$. Given input $\sigma \in \Sigma^\omega$, $M$ first scans through the initial 'a'-s on $\sigma$ in a state from $K_1$ till it first reaches a letter from $\Sigma_1$. From now on $M$ acts according to the following rules:

1. Whenever $M$ scans a new 'b' on $\sigma$ it enters state $q_b$;
2. Whenever $M$ reaches the end of a section from $\Sigma_1^+$ and scans 'a' again, it (i) passes through state $q_a'$, then (ii) checks that the current section of a's is longer by exactly one from the previous section of a's. While (ii) is carried out, $M$ is in the set of working states $K_1$.

In case $\sigma$ is found not to be of the required form, i.e. if for some $j > 0$, $\sigma/j \notin \text{Init}(L)$, $M$ enters the traverse state $q_T$ in which it will keep moving right forever. Now if $F = K_1 \cup \{ q_a, q_b \}$, then $T_3(M, F) = L'$ while

$$T_1(M, F) = T_2(M, F) = a^\omega U \text{Init}(L) \Sigma_1^+ U L',$$

where $L' = \{ a^+ \prod a^1 x_i | x_i \in \Sigma_1^+ \}$.

If $F' = \{ q_a \}$, we have $T_2(M, F') = L'$, while $T_1(M, F') = a^+a_1^+a_2^+\Sigma^\omega$ and $T_i(M, F') = \emptyset$ for $i=1', 2', 3$. □

Among the above defined $i$-acceptance modes, $3$-acceptance has been the most commonly used definition of $\omega$-acceptance in previous papers (e.g. [Mc], [Cho]) and it also turns out to be more powerful (or at least no less powerful) than any of the other $i$-acceptance modes, as was shown for other types of machines ([Lan], [Co&Go2,3]) and for $\omega$-TA's in Section 6 below. It is therefore chosen as our standard definition of $\omega$-acceptance, and the following convention
is made for convenience.

Convention: Subsequently 3-acceptance will be simply referred to as acceptance, and \( T_3(M) \), the \( \omega \)-language accepted by \( M \), will be denoted by \( T(M) \) (with subscript 3 dropped).

Definition 1.10 Two \( m-\omega \)-TAl's \( M \) and \( M' \) will be called equivalent (\( i \)-equivalent for \( i = 1, 1', 2, 2' \)) iff \( T(M) = T(M') \) (\( T_i(M) = T_i(M') \)).

Note that by the above definition of acceptance (or \( i \)-acceptance) in \( \omega \)-TAl's, an \( \omega \)-input \( \sigma \) may be accepted only via a c.n.o. run, that is, incomplete or oscillating runs on \( \sigma \) cannot lead to acceptance.

Definition 1.11 An \( m-\omega \)-TA \( M \) is said to possess Property C iff for every \( \sigma \in \omega \), there exists a c.n.o. run of \( M \) on \( \sigma \).

Remark 1.12 Every \( \omega \)-TA(\( m-\omega \)-TA) without Property C can be transformed into an equivalent \( \omega \)-TA \( M' \) with Property C. \( M' \) is obtained from \( M \) by adding a new traverse state \( q_T \), in which \( M' \) just keeps moving right on the input tape. \( M' \) may choose to enter \( q_T \) at the beginning of its computation and stay in that state forever, or else \( M' \) imitates \( M \) on the given \( \omega \)-input.

In Section 6 we prove a more powerful result, namely that every \( \omega \)-TA can be transformed into an equivalent \( \omega \)-TA in which every run is c.n.o. (Theorem 6.6).

The above remark is no longer true when only deterministic \( \omega \)-TA's are considered. As is shown in [Co&Go4], deterministic \( \omega \)-TA's without Property C are strictly more powerful than deterministic \( \omega \)-TA's with Property C.
2. \( \omega \)-GRAMMARS

In this section \( \omega \)-grammars, i.e. grammars generating \( \omega \)-languages, are introduced and the "S-boundary" normal form for \( \omega \)-PSG's is derived.

**Definition 2.1** An \( \omega \)-phrase structure grammar (\( \omega \)-PSG) is a quintuple \( G = (V_N, V_T, P, S, F) \), where \( G_1 = (V_N, V_T, P, S) \) is an ordinary phrase structure grammar, the rules in \( P \) are all of the form \( \alpha \rightarrow \beta \), where \( \alpha \in V_N^+ \), \( \beta \in V_T^* \) and \( F \subseteq 2^P \). The sets in \( F \) are called the repetition sets of grammar \( G \).

We shall focus our attention on infinite sequences generated by using rules of \( G \) infinitely many times. Let \( d \) be an infinite derivation in \( G \), starting from some string \( \alpha \in V_T^* \):

\[
d: \alpha = \alpha_0 \xrightarrow{G} \alpha_1 \xrightarrow{G} \alpha_2 \xrightarrow{G} \ldots \xrightarrow{G} \alpha = \ldots
\]

where for each \( i = 0, 1, \ldots \), \( \alpha_i \in V_T^* \), \( \alpha \in V_T \). Note that the derivation need not be leftmost, since some of the \( \alpha_i \)'s may be empty.

Let \( \sigma = \prod_{i=0}^{\infty} \alpha_i \). If \( \sigma \in V_T^\omega \), we write \( \sigma \rightarrow \). The assumption that the left-hand side of each rule of \( P \) is in \( V_N^+ \) guarantees that the terminal prefix of each sentential form, up to the first occurrence of a non terminal will never be replaced later in the derivation, and can be considered a prefix of the infinite word generated.

The above derivation \( d \) induces a mapping from \( N \) to \( P \), \( d_p: N \rightarrow P \), where \( d_p(i) \) is the rule in \( P \) used in step \( i \) of derivation \( d \). Define:

\[
L(G) = \{ \sigma \in V_T^\omega \mid \text{there exists a derivation } \sigma \rightarrow \}.
\]

\( L(G) \) is the \( \omega \)-language generated by \( G \).

An \( \omega \)-language generated by an \( \omega \)-PSG is a **TYPEO \( \omega \)-language**. Let \( \text{TYPEO}_\omega \) denote the class of \( \text{TYPEO} \ \omega \)-languages.
Definition 2.2 An \( \omega \)-context sensitive grammar (\( \omega \)-CSG) is an \( \omega \)-PSG in which for each production \( \alpha \rightarrow \beta \), \( |\beta| \geq |\alpha| \) holds.

Example 2.3 Let \( G = \langle \{S,X,X_1,\$,\$,\$,\$,\$,\$,\$,\$,\$\},\{0,1\},P_1 \cup \{S \rightarrow O_1 \$X\$1\$\},S,\{P_1\} \rangle \) be an \( \omega \)-CSG, where \( P_1 \) is:

1. \( \$X \rightarrow O_1 \$X_1 \)
2. \( X_1 \$X_1 \rightarrow X_1 \$X_1 \)
3. \( X_1 \$X_1 \rightarrow X_1 \$X_1 \)
4. \( \$X_1 \rightarrow 001 \$X_1 \$X_1 \)
5. \( \$X_1 \rightarrow X_1 \$X_1 \)
6. \( \$X_1 \rightarrow X_1 \$X_1 \)

One can easily verify that \( L(G) = \{ \sum \sum \}_{i=1}^{\infty} \).

Remark 2.4 As in the case of ordinary CSG's, one can show that the productions of \( \omega \)-CSG's may be restricted to the so called "context sensitive" productions of the form: \( \alpha_1 \alpha_2 + \alpha_1 \beta \alpha_2 \) where \( \alpha_1, \alpha_2 \in V^*, \alpha \in V_N, \beta \in \Sigma^+. \)

Definition 2.5 An \( \omega \)-PSG(\( \omega \)-CSG) with \$-boundary is an \( \omega \)-PSG[\( \omega \)-CSG] \( G = \langle V_N \cup \{\$\}, V_T, P, S, F \rangle \) in which each production is of one of the following forms \( (1)-(4) \) \( [(1)-(3)] \):

1. \( \alpha \rightarrow \beta \), \( \alpha, \beta \in V_N^+ \)
2. \( S \rightarrow \$\alpha \), \( \alpha \in V_N^+ \)
3. \( \$A \rightarrow a\$ \), \( A \in V_N, a \in V_T \)
4. \( A \rightarrow \varepsilon \), \( A \in V_N \)

An \( \omega \)-PSG with \$-boundary has the following property: If \( S \rightarrow^+ \alpha \), then \( \alpha = u\$\alpha' \), where \( u \in V_T^*, \alpha' \in V_N^* \). That is, \$ divides every sentential form into two parts: \( u \) - the terminal part generated so far, never to be rewritten again, and \( \alpha' \) - the workspace string, comprised of non terminals from which the rest of the \( \omega \)-word will be generated.

Theorem 2.6 Every \( \omega \)-PSG(\( \omega \)-CSG) can be converted to an equivalent \( \omega \)-PSG (\( \omega \)-CSG) with \$-boundary.
Proof. Let $G = (V_N, V_T, P, S, F)$ be an $\omega$-PSG. W.l.o.g. we may assume $P = P_1 \cup P_2 \cup P_3$, where $P_1 = \{\alpha + \beta \in P | \alpha, \beta \in V_N^+\}$, $P_2 = \{\alpha \rightarrow a \in P | a \in V_N, a \in V_T\}$, and $P_3 = \{A \rightarrow \epsilon \in P | A \in V_N\}$. Define $V'_N = V_N \cup \bar{V}_T$, where $\bar{V}_T = \{\bar{a} | a \in V_T\}$; let $\$ and $S_1$ be new symbols and define $P' = \{S_1 \rightarrow \$| S \rightarrow a \in P_1\} \cup P_1 \cup \{A \rightarrow \bar{a} | A \rightarrow a \in P_2\} \cup P_3 \cup R$, where $R = \{S\bar{a} \rightarrow a\$ | a \in V_T\}$. Let $G' = (V_N \cup \{S_1, \$\}, V'_T, P', S_1, F')$ be the $\omega$-PSG with $\$-boundary, where $F'$ is defined as follows: for every $H \in F$ s.t. $H = H_1 \cup H_2$, $H_1 \subseteq P_1 \cup P_3$, $H_2 \subseteq P_2$, let $H_2 = \{A \rightarrow \bar{a} | A \rightarrow a \in H_2\}$ and let $H = \{H_1 \cup H_2 U Q | Q \subseteq R\}$; now define $F' = U H$. Then clearly $L(G') = L(G)$.

In case $G$ is an $\omega$-CSG, $P_3$ above will be empty and $G'$ will be an $\omega$-CSG with $\$-boundary.

3. TYPE 0 $\omega$-LANGUAGES: PROPERTIES AND CHARACTERIZATIONS

3.1 Characterization of type 0 $\omega$-languages

As one may expect, type 0 $\omega$-grammars generate precisely the family of $\omega$-languages recognizable by non-deterministic $\omega$-Turing acceptors.

Furthermore, it is shown that also $\omega$-CSG's generate exactly the same family.

Theorem 3.1.1 (a) $\text{TYPE}_0^\omega$ equals the class of $\omega$-languages accepted by $\omega$-TA's. (b) For every $\omega$-PSG there can be constructed an equivalent $\omega$-CSG.

Proof. (a) The proof is an adaptation of the proof for the analogous result in [H&U] (pp. 112). Let $L$ be the $\omega$-language accepted by an $\omega$-TA $M = (K, \Sigma, I, \delta, q_0, F)$. Define the $\omega$-CSG $G = (V_N, \Sigma, P, S, F')$, where $V_N = \Sigma \times \Gamma \cup K \cup \{\$, S, S_1\}$, $\$, S, S_1$ are new symbols and the rules in $P$ are:

1. $S \rightarrow \$q_0S_1$
2. $S_1 \rightarrow [a, a]S_1$ for every $a \in \Sigma$.
(3) For every $q \in K$, $a, b \in \Sigma$ and $A, B \in \Gamma$: $q[a, A] + [a, C]p$ if $(p, C, R) \in \delta(q, A)$; $[b, B]q[a, A] + p[b, B][a, C]$ if $(p, C, L) \in \delta(q, A)$, and $q[a, A] + p[a, C]$ if $(p, C, S) \in \delta(q, A)$.

(4) $[a, A] + a$ for each $a \in \Sigma$, $A \in \Gamma$.

For $1 \leq j \leq 4$ let $R_j$ denote the set of rules of type $(j)$ above.

Since TYPEO is closed under union, we may assume w.l.o.g. that $F$ consists of a single set, denoted by $F$ itself. For every $q \in F$, define $R(q)$ to be the set of rules in $R_j$ in which $q$ appears on the left-hand side. Define $F' = \{ H \subseteq \{ U R(q) \} U R_2 U R_4 \mid \forall q \in F, \ H \cap R(q) \neq \emptyset \land q \in F \}$. Clearly $L(G) = L$.

Now let $G = (V_N, \Sigma, P, S, F)$ be an $\omega$-PSG. By Thm. 2.6 we may assume that $G$ is an $\omega$-PSG with $\$-$boundary. Construct an $\omega$-TA $M = (K, \Sigma, \Gamma, \delta, q_0, F')$, where $K = K' \cup \{ q_{o}, q_{1}, q_T \} \cup \{ q_p \mid p$ a production in $P \}$, $K'$ is a set of working states and $q_T$ is a traverse state. The machine has its tape divided into 2-tracks. The first track contains the input word $\sigma \in \Sigma^\omega$, while on the second track $M$ simulates non-deterministically a derivation in $G$, starting with $S$. For every production $p$ in $P$ there is a corresponding state $q_p$ in $K$, entered by $M$ every time production $p$ is simulated on the second track. Furthermore, each time $M$ simulates a production of the form $\$A + a\$, for some $a \in \Sigma$, letter $a$ is checked against the letter pointed to on the first track. If there is a match, $M$ enters state $q_1$, moves one square to the right on the first track and then proceeds with the simulation. Otherwise $M$ enters the traverse state $q_T$. $M$ may also choose to enter $q_T$ right at the beginning of its computation.

For each $H \in F$ let $\overline{H} = \{ q \cup \{ q_p \mid p \in H \} U \{ q_1 \} \mid q \in F' \}$ and define $F' = U \overline{H}$ as the collection of designated state sets. Then $L(G) = T(M)$.

(b) Follows directly from the proof of (a) above.
Using the standard "padding" technique, one can directly convert an $\omega$-PSG into an equivalent $\omega$-CSG.

3.2 The "Chomsky Hierarchy" for $\omega$-languages

Combining Theorem 3.1.1 with the basic results concerning the $\omega$-context free languages and the $\omega$-regular languages ([Co&Go1],[Mc]) one obtains a "Chomsky hierarchy" for $\omega$-languages, which differs from the well-known Chomsky hierarchy for finite-string languages. Let us first introduce the two lower families in this hierarchy.

Definition 3.2.1 An $\omega$-context free grammar ($\omega$-CFG) ($\omega$-right linear grammar ($\omega$-RLG)) $G$ is an $\omega$-PSG whose productions are context free [right linear] ([H&U]). If $L = L(G)$ for an $\omega$-CFG $G$, then $L$ is an $\omega$-context free language ($\omega$-CFL). An $\omega$-language generated by an $\omega$-RLG is an $\omega$-regular language.

The family of $\omega$-CFL's ($\omega$-regular languages) has been characterized by means of $\omega$-type pushdown acceptors (finite state acceptors) and also with the aid of the $\omega$-Kleene closure operator (Definition 0.3) ([Mc],[Bu], [Co&Go1]). The latter characterization is stated in the next theorem

Theorem 3.2.2 (a) The class of $\omega$-regular languages coincides with $\omega$-KC(Reg), the $\omega$-Kleene closure of the regular languages. (b) The class of $\omega$-CFL's coincides with $\omega$-KC(CF), the $\omega$-Kleene closure of the context free languages.

The singleton $\omega$-language $L = \{ \Pi_{i=1}^{n} 0^{i} \}$, generated by the $\omega$-CSG in Example 2.3, is an example of a TYPEO $\omega$-language which is not $\omega$-context free. This is because, as one can easily verify, $L$ cannot be represented in the form $L = \bigcup_{i=1}^{n} L_{i} \omega$, where $n \geq 1$ and $L_{i}, \omega, 1 \leq i \leq n$, are arbitrary languages. It follows that $L$ cannot belong to the $\omega$-Kleene closure of any family of languages, thus by Theorem 3.2.2 $L$ cannot be an $\omega$-CFL.
Theorem 3.2.3  The family of $\omega$-regular languages is properly included in the family of $\omega$-context free languages, which in turn is properly included in the family of $\omega$-CSG languages; the latter family, however, coincides with TYPE$_0$.

3.3 Properties of TYPE$_0$.  

Theorem 3.3.1  TYPE$_0$ is closed under union and intersection.

Proof.  (a) Closure under union follows from the definition of $\omega$-grammars.  

(b) Let $L_1, L_2$ be two TYPE$_0$ languages accepted by $\omega$-TA's $M_1$ and $M_2$ respectively.  An $\omega$-TA $M$ that accepts $L_1 \cap L_2$ has its tape divided into three tracks.  On the first is written the input $\sigma \in \Sigma^\omega$; on the second and third track $M$ simulates $M_1$ and $M_2$ respectively.  At start, $M$ copies the first $\ell$ input letters from the first track onto the second and third track, where $\ell$ is some fixed positive integer.  Every time $M_i$ ($i = 1, 2$) has to move right of the copied section, $M$ copies $\ell$ more input letters onto both tracks.  Every computation phase of $M$ consists of a simulation of a single move of $M_1$ and a single move of $M_2$.  One can provide $M$ with a collection of designated state sets such that $T(M) = T(M_1) \cap T(M_2)$.  

In Section 6 it will be shown that TYPE$_0$ is not closed under complementation.

Definition 3.3.2  Let $\Sigma, \Delta$ be two finite alphabets.  A substitution $f$ is a mapping $f: \Sigma \rightarrow 2^\Delta$.  $f$ can be extended to strings in $\Sigma^\omega$ as follows:  

For $\sigma = \prod_{i=1}^{\omega} a_i \in \Sigma^\omega$, $a_i \in \Sigma \forall i \geq 1$, define $f(\sigma) = \{ \prod_{i=1}^{\omega} b_i \mid b_i \in f(a_i) \}$.  

For each $L \subseteq \Sigma^\omega$, define $f(L) = \bigcup_{\sigma \in L} f(\sigma)$.  

Substitution $f$ is called an RE substitution if $f(a)$ is a recursively enumerable (RE) set for all $a$ in $\Sigma$.  A class $\mathcal{L}$ of $\omega$-languages is said to be closed under RE substitution if for any $L \subseteq \Sigma^\omega$ in $\mathcal{L}$ and RE substitution $f: \Sigma \rightarrow 2^\Delta$, $f(L) \cap \Delta^\omega \in \mathcal{L}$.  

Technion - Computer Science Department - Technical Report CS0076 - 1976
Theorem 3.3.3  TYPEO is closed under RE substitution.

Proof. Let \( L \) be a \( \text{TYPEO} \) \( \omega \)-language generated by the \( \omega \)-PSG \( G = (V_N, \Sigma, P, S, F) \), where \( \Sigma = \{a_i\}_{i=1}^n \) and \( F = \{ F_j \}_{j=1}^k \). Let \( f: \Sigma + 2^\omega \rightarrow \text{RE} \) be an RE substitution s.t. for \( 1 \leq i \leq n \), \( f(a_i) = L_i = L(G_i) \) where \( G_i = (V_N^{(i)}, V_T, P(i), S(i)) \) is a PSG. W.l.o.g. we may assume that all sets \( V_N^{(i)} \), \( 1 \leq i \leq n \), and \( V_N \) are pairwise disjoint. By Lemma 2.6 we may assume that \( G \) is an \( \omega \)-PSG with \( \$ \)-boundary, i.e. \( V_N = V_N \cup \{ S, \} \), \( P = P_1 \cup P_2 \cup P_3 \cup P_4 \), where
\[
P_1 = \{ \alpha + \beta \in P | \alpha, \beta \in V_N^+ \},
\]
\[
P_2 = \{ S + a S \in P | a \in V_N^+ \},
\]
\[
P_3 = \{ A + e \in P | A \in V_N \}
\]
and \( P_4 = \{ (S + a S) \in P | A \in V_N, a \in \Sigma \} \). Define the \( \omega \)-PSG \( G' = (V_N', V_T, P', S, F') \), where
\[
V_N' = V_N \cup \{ \bigcup_{i=1}^n V_N^{(i)} \} \cup \{ \$, \# \}
\]
and \( P' \) includes: \( R_1 = \{ \bigcup_{i=1}^n P(i) \} \cup P_1 \cup P_2 \cup P_3 \cup P_4 \), \( R_2 = \{ (\alpha + \beta S)^\omega | \alpha \in V_N, \beta \in V_T^* \} \), \( R_3 = \{ \#a + a \$ | a \in V_T \} \) and \( R_4 = \{ \#S \Rightarrow \$ \} \). \( R_2 \) initiates the substitution of \( a_i \) by a word in \( L(G_i) \). Inclusion of at least one rule from \( R_3 \) in each of the repetition sets will guarantee that a word in \( V_T^\omega \) will be generated and the inclusion of \( R_4 \) in each repetition set will assure that each \( a_i \) was substituted by a finite word from \( L(G_i) \). For each \( 1 \leq j \leq \ell \), define \( H(j) = \{ (F_j P_4) \cup Q \cup Q' \cup R_4 \cup Q_j | Q \subseteq \bigcup_{i=1}^n P(i), Q' \subseteq R_3, Q \neq \emptyset \} \), where \( Q_j = \{ (\alpha + \beta S)^\omega | \alpha \in V_N, \beta \in V_T \} \); then let \( F' = \bigcup_{j=1}^\ell H(j) \). Clearly \( L(G') = f(L) \cap V_T^\omega \). \( \square \)

Corollary 3.3.4 For any RE set \( L_1 \) and \( L \in \text{TYPEO} \), \( L_1 \cap L \in \text{TYPEO} \).

The next theorem shows that the \( \omega \)-Kleene closure characterization (Theorem 3.2.2) cannot be generalized to type 0 \( \omega \)-languages.

Theorem 3.3.5 The \( \omega \)-Kleene closure of the family of RE sets is properly included in \( \text{TYPEO} \).

Proof. For any set of symbols \( \{ a_i, b_i \}_{i=1}^n \), \( \{ U a_i b_i \}_{i=1}^n \subseteq \text{TYPEO} \), thus by Theorem 3.3.3 \( \omega \)-KC(RE) \( \subseteq \text{TYPEO} \). The singleton \( \omega \)-language \( L = \{ \prod_{i=1}^\infty 0^i \} \) (Example 2.3) implies that the above containment is proper. \( \square \)
4. THE FOLDING PROCESS

In this section we define a process of "folding forward" a Turing machine semi infinite tape so that all information written on the tape is continuously carried forwards, and thus can be retrieved without having to re-scan the initial segment of the tape. This folding process will subsequently enable us to turn any deterministic \( \omega \)-TM into one with Property C.

Definition 4.1 Let \( \sigma \) and \( \eta \) be infinite tapes over alphabet \( \Gamma \), where \( \eta \) is a two-track tape. We say that \( \eta \) is a \( k \)-folded version of \( \sigma \) iff:

a) for \( k \leq j \leq 2k-2 \), \( \eta(j) \) contains \( \sigma(j) \) on its first track and \( \sigma(2k-j-1) \) on its second track.

b) for \( 2k-2 \leq j \), \( \eta(j) \) contains \( \sigma(j) \) on its first track.

Let \( \sigma = \prod_{i=1}^{\infty} a_i, a_i \in \Sigma \); then \( \eta \) contains \( \sigma \) with its initial segment \( a_1 \cdots a_{k-1} \) folded forwards as is shown in Fig. 4.1 below

\[
\sigma: \begin{array}{cccccccc}
a_1 & a_2 & \cdots & a_{k-1} & a_k & \cdots & a_{2k-2} & a_{2k-1} \\
\end{array}
\]

\[
\eta: \begin{array}{cccccccc}
x & x & \cdots & x & a_k & \cdots & a_{2k-2} & a_{2k-1} & \cdots \\
\end{array}
\]

We say that Turing Machine \( M \) \( k \)-folds \( \sigma \) if \( M \) turns \( \sigma \) into its \( k \)-folded version \( \eta \).

Lemma 4.2 (Folding Process) For every TM \( M \) there can be constructed a TM \( M_1 \) which simulates \( M \) on every \( \omega \)-input \( \sigma \) in such a way that for some \( i \) \( M_1 \) reaches \( \sigma(i) \), within the
it will continue the simulation. Note that for $i \geq 3$, whenever $M$ reaches $\sigma(i)$ for the first time, the tape of $M$ is already $i-1$ folded in $M_1$, thus to obtain the $i$-folded version of the tape, $M_1$ has to shift the initial segment containing squares $1, \ldots, i-1$, which is written backwards on its second track, 2 squares to the right on the second track, and then copy the contents of square $i-1$ of the first track onto square $i$ on the second track, as is illustrated in Fig. 4.2.

\begin{figure}
\centering
\begin{tabular}{c}
\hline
\multicolumn{3}{c}{i-1 folding} \\
\hline
$a_{i-1}$ & $\cdots$ & $a_{2i-4}$ \\
$a_{i-2}$ & $\cdots$ & $a_1$ \\
\hline
\multicolumn{3}{c}{i-1 folding} \\
\hline
$a_{i-1}$ & $a_i$ & $\cdots$ & $a_{2i-3}$ & $a_{2i-2}$ \\
$x$ & $a_{i-1}$ & $\cdots$ & $a_1$ \\
\hline
\end{tabular}
\end{figure}

Remark 4.3 We would like to point out that in the construction of machine $M_1$ above, every complete run of $M$ on $\omega$-input $\sigma$ (including an oscillating run), becomes in $M_1$ a c.n.o. run. Hence $M_1$ may accept $\omega$-inputs which were not accepted by $M$. Only in case $M$ is a machine which never oscillates on any $\omega$-input (e.g. if $M$ is an $\omega$-DTA with property C), $M_1$ is guaranteed to be equivalent to $M$.

Lemma 4.4 (Relative Folding Process) Let $M$ be an $m$-TM and let $\alpha$ and $\beta$ be two of $M$'s working tapes. Then there can be constructed an equivalent $m$-TM $M_1$ with the following property: For some fixed integer $\ell > 0$, when given an $\omega$-input $\sigma$, $M_1$ simulates $M$ on $\sigma$ s.t. for each $i \geq 2$, within at most $\ell$ computation steps after position $\alpha(i)$ has been reached for the first time on tape $\alpha$, $M_1$'s reading head on $\beta$ will be to the right of position $\beta(i-1)$ and will never again return to the initial segment $\beta(i-1)$.
Proof. The proof resembles that of Lemma 4.2 above; however, here whenever $M$ reaches $a(i)$ for the first time, $M_i$ i-folds $\beta$. □

Note that the relative folding process described in Lemma 4.4 may also be applied in case $\alpha$ and $\beta$ are two tracks of the same working tape rather than two tapes. In both cases we say that the relative folding process is applied to $\beta$ w.r.t. $\alpha$.

5. DETERMINISTIC $\omega$-TURING ACCEPTORS

Definition 5.1 An $m$-$\omega$-TA[$\omega$-TA] $M = (M', F)$ is deterministic (denoted $m$-$\omega$-DTA[$\omega$-DTA]) iff $M'$ is a deterministic $m$-TM.

An $\omega$-DTA ($m$-$\omega$-DTA) has a unique run on every $\omega$-input $\sigma$. We shall focus in this section only on $\omega$-DTA's in which every run is c.n.o., i.e. with Property C. $\omega$-DTA's without Property C turn out to be more powerful and rather different in their properties from those with Property C, and their study will be included in another paper [Co&Go4].

Convention: Every $m$-$\omega$-DTA ($\omega$-DTA) considered in the sequel is assumed to have Property C.

Notation 5.2: Let $\text{DTML}_{\omega}$ denote the class of $\omega$-languages accepted by $\omega$-DTA's with Property C. For $i = 1, 1', 2, 2'$, let $A_i$-$\text{DTML}_{\omega}$ denote the class of $\omega$-languages $i$-accepted by $\omega$-DTA's with Property C.

5.1 Basic Results on $\omega$-DTA's

We first show that w.l.o.g. we may restrict ourselves to studying $i$-acceptance in single tape $\omega$-DTA's.

Theorem 5.1.1 For each $i = 1, 1', 2, 2', 3$ and for every $m$-$\omega$-DTA, $m \geq 2$, there can be constructed an $i$-equivalent $\omega$-DTA.
Proof. Let $M$ be an $m$-$\omega$-DTA; if $m > 2$, all the working tapes of $M$ can be simulated on its second tape, yielding an $i$-equivalent $2$-$\omega$-DTA. Thus we may assume that $M = (M', F)$ is a $2$-$\omega$-DTA. Define $M_1$ to be an $\omega$-DTA that simulates $M$ as follows. The single tape of $M_1$ is divided into two tracks, $\alpha$ and $\beta$, representing respectively the input tape and the working tape of $M$. For each $\omega$-input $a$, the simulation will be carried out by $M_1$ while applying the relative folding process (Lemma 4.4) for $\beta$ w.r.t. $\alpha$. This will guarantee that $M_1$ also has Property C. For each $i = 1, 1', 2, 2', 3$, one can define in terms of $F$ a set of designated sets $H(i)$ s.t. $T_1((M_1, H(i))) = T_1(M)$. □

We now state two fundamental lemmas concerning the various types of $i$-accepting mappings; the first lemma follows directly from the definition.

Lemma 5.1.2 Let $r: N \to S$ be a mapping and let $F \subseteq S$;

(a) $r$ is $1$-accepting w.r.t. $F$ iff it is not $1'$-accepting w.r.t. $S-F$.
(b) $r$ is $2$-accepting w.r.t. $F$ iff it is not $2'$-accepting w.r.t. $S-F$.

Lemma 5.1.3 [Co&Go3] Let $F = \{F_i\}_{i=1}^\infty$, $i=1, 2, 3\ldots$, be a collection of subsets of $S$. Then there can be defined sets $S_1, S_2$ and subsets $K_1 \subseteq S_1, K_2 \subseteq S_2$ s.t. for any given mapping $r: N \to S$, there can be constructed two corresponding mappings $r_1: N \to S_1$ and $r_2: N \to S_2$ satisfying the following conditions:

(a) for $j = 1, 2$, and for each $i > 1$, $r_j(i-1)$ and $r(i)$ uniquely determine $r_j(i)$.
(b) $r$ is $1'$-accepting w.r.t. $F$ iff (1) $r_1$ is not $1'$-accepting w.r.t. $\{K_1\}$, and (2) $r_1$ is $1'$-accepting w.r.t. $\{S_1-K_1\}$.
(c) $r$ is $2'$-accepting w.r.t. $F$ iff (1) $r_2$ is not $2'$-accepting w.r.t. $\{K_2\}$, and (2) $r_2$ is $2'$-accepting w.r.t. $\{S_2-K_2\}$.
By Lemmas 5.1.2, and 5.1.3 we have:

**Proposition 5.1** For any \( \omega \)-language \( L \subseteq \Sigma^\omega \):

(a) \( L \in \text{A}1-\text{DTML}_\omega \) iff \( \exists^\omega - L \in \text{A}1'-\text{DTML}_\omega \).

(b) \( L \in \text{A}2-\text{DTML}_\omega \) iff \( \exists^\omega - L \in \text{A}2'-\text{DTML}_\omega \).

It follows from Lemma 5.1.3 above that every \( \omega \)-DTA can be replaced by an i-equivalent (i = 1, 1', 2, 2') \( \omega \)-DTA with a single designated set (U-\( \omega \)-DTA).

**Theorem 5.1.5** Every \( L \in \text{A}i-\text{DTML}_\omega \) (i = 1, 1', 2, 2') can be i-accepted by a U-\( \omega \)-DTA.

Utilizing Shannon's construction of a universal two-state Turing machine ([Sha]) and Theorem 5.1.5 above we obtain the following:

**Theorem 5.1.6** (a) Every \( L \in \text{A}1-\text{DTML}_\omega \) (i = 1, 1', 2) can be i-accepted by a three-state U-\( \omega \)-DTA; (b) Every \( L \in \text{A}2'-\text{DTML}_\omega \) can be 2'-accepted by a four-state U-\( \omega \)-DTA.

**Proof.** For i = 1, 1', 2, three states suffice to simulate any \( \omega \)-DTA; e.g. w.r.t. 1'-acceptance, two states will do for simulating the machine within the designated state set \( F \), plus an auxiliary "dead state" for non-acceptance. As for 2'-acceptance two states are necessary to simulate the machine inside \( F \), and two more states for simulating the machine outside \( F \).

As is shown in [Cq&Go4], no analogous result holds w.r.t. 3-acceptance in \( \omega \)-DTA's.

5.2 **Characterization of Type i Recognition in \( \omega \)-DTA's**

**Notation 5.2.1** Let \( \text{RE} \) denote the class of \( \text{RE} \) sets. Let \( \text{CSDCS} \) denote the class of context-sensitive languages (CSL's) [deterministic CSL's (DCSL's), i.e. CSL's accepted by deterministic LBA's].
In this section we show that each of the families $A_i-DTM_{\omega}$, $i=1,1',2,2'$, is obtained from the families RE, CS and DCS with the aid of some new algebraic operators. Three unary operators will now be defined, which turn a (finite-string) language into an $\omega$-language. The first of these operators, limit ($\lim$), appears in [Cho], [Eit] and [Lin], and is used mainly for characterizing 2-acceptance in deterministic $\omega$-acceptors. The other two operators, Extrapolation (Ext) and Non-init (Ninit), are utilized for the characterization of $1'$-acceptance and 1-acceptance in deterministic $\omega$-acceptors.

**Definition 5.2.2** For $L \subseteq \Sigma^* \cup \Sigma^\omega$, define the **extrapolation** of $L$, Ext($L$), by:

$$\text{Ext}(L) = \{\sigma \in \Sigma^|\forall i \geq 0, \sigma/i \in \text{Init}(L)\}$$

where $\text{Init}(L) = \{x \in \Sigma^* | \exists y \in \Sigma^* \cup \Sigma^\omega \text{ s.t. } xy \in L\}$.

For $L \subseteq \Sigma^*$ define $\lim(L)$, the **limit** of $L$, by:

$$\lim(L) = \{\sigma \in \Sigma^|\forall i \geq 0 \exists j \geq i \text{ s.t. } \sigma/j \in L\}$$

and also define $\text{Ninit}(L) = \Sigma^\omega - \Sigma^\omega$ for any $L \subseteq \Sigma^*$.

For a family $\mathcal{L}$ of subsets of $\Sigma^* \cup \Sigma^\omega$, define $\text{Ext}(\mathcal{L}) = \{\text{Ext}(L) | L \in \mathcal{L}\}$. For a family $\mathcal{L}$ of subsets of $\Sigma^*$ define $\text{lim}(\mathcal{L}) = \{\lim(L) | L \in \mathcal{L}\}$ and $\text{Ninit}(\mathcal{L}) = \{\text{Ninit}(L) | L \in \mathcal{L}\}$.

The following example will clarify the above definitions:

**Example 5.2.3** Let $\Sigma = \{0,1\}$. (a) Let $L = 0^*1^+$; then $\text{Ext}(L) = 0^\omega \cup 0^*1^\omega$, $\lim(L) = 0^*1^\omega$ and $\text{Ninit}(L) = 0^\omega$; (b) Let $L = \{0^n1^n | n \geq 1\}$; then $\text{Ext}(L) = 0^\omega$ but $\lim(L) = \emptyset$; (c) Let $L = \Sigma^*0^\omega$; then $\text{Ext}(L) = \Sigma^\omega$.

The operations Ext and Ninit are related by the following theorem:
Theorem 5.2.4 ([Co&Go3]) Let $\mathcal{L}$ be a class of $\omega$-languages over $\Sigma$.
Then there exists a class $\mathcal{L}_1$ of finite string languages s.t. $\mathcal{L} = \text{Ninit}(\mathcal{L}_1)$
iff for each $L \in \mathcal{L}$, $L = \text{Ext}(L)$.

With the aid of the above operations we are now able to characterize
each of the families $A_i\text{-DTML}_\omega$ ($i = 1,1',2,2'$).

Theorem 5.2.5
(a) An $\omega$-language $L \subseteq \Sigma^\omega$ is in $A_1\text{-DTML}_\omega$ iff $L$ is of the form $L = L_1 \Sigma^\omega$,
where $L_1$ is a deterministic CSL;
(b) $A_1\text{-DTML}_\omega = \text{Ninit}(\text{DCS})$;
(c) $A_2\text{-DTML}_\omega = \text{lim}(\text{DCS})$;
(d) $A_2'\text{-DTML}_\omega = \{\Sigma^\omega - L | L \in \text{lim}(\text{DCS})\}$.

Proof. (a) Let $L \in A_1\text{-DTML}_\omega$; then $L$ is $1$-accepted by some $U_\omega$-DTA
$M = (K,\Sigma,\Gamma,\delta,q_o,F)$. Define a $\text{DTA}^3$ $M_1 = (K,\Sigma,\Gamma U\{\$,\}$,\delta_1,q_o,F)$, where
$\$ $\notin \Gamma$, $\delta_1(q,A) = \delta(q,A)$ for each $q \in K$, $A \in \Gamma$ and otherwise $\delta_1$ is
undefined. Let $L_1 = \{x \in \Sigma^* | x: (q_o,\$,x$,1) \overset{*}{\rightarrow} M_1 (q,A,I) \text{ for some } q \in F\}$; then
clearly $L_1$ is a DCSL. Since $M$ has Property C, it follows that $L = L_1 \Sigma^\omega$.

Now let $L$ be a DCSL; then there exists a $\text{DLBA}$ $M = (K,\Sigma,\Gamma U\{\$,\}$,\delta,q_o,F)$
s.t. $L = \{x \in \Sigma^* | x: (q_o,\$,x$,1) \overset{*}{\rightarrow} M (q,A,I), q \in F\}$ and $\Sigma^\omega - L = \{x \in \Sigma^* | x: (q_o,\$,x$,1) \overset{*}{\rightarrow} M (q_B,A,I)\}$, where $q_B$ is a designated failure state. The
$\omega$-DTA $M_1$ accepting $L \Sigma^\omega$ will operate as follows: Scanning an input $\sigma \in \Sigma^\omega$,
for each $i = 1,2,\ldots$, $M_1$ will make a guess that $\sigma/i \in L$ and try to verify
each guess in turn by simulating $M$ on $\sigma/i$. When for some $i$, the guess
$\sigma/i \in L$ turns out to be wrong, $M_1$ $i$-folds $\sigma$ as in Lemma 5.2 and then
proceeds to make the next guess $\sigma/i+1 \in L$. If for some $i$, $\sigma/i \in L$ turns
out to be true, $M_1$ will enter a final traverse state in which it will scan
the rest of the input tape. Due to the continual folding process $M_1$ will
have Property C. Clearly \( T(M_1) = T(M) \Sigma^\omega \).

(b) Follows by (a) above and Proposition 5.1.4a.

(c) Let \( L \in A^2\text{-}DTML^\omega \) be 2-accepted by some \( U\omega\text{-}DTA \) \( M \). Let \( M_1 \) be the DTA constructed from \( M \) precisely as in (a) above. Then \( T_2(M) = \lim(L_1) \), where \( L_1 \) is the DCSL defined by \( M_1 \) as above.

Now let \( L \) be a DCSL. The \( \omega\text{-}DTA \) \( M' \) 2-accepting \( \lim(L) \) will operate similarly to the \( \omega\text{-}DTA \) \( M_1 \) in (a) above, but for each \( i \) s.t. \( \sigma/i \in L \), \( M' \) will first enter a designated state, then will \( i \)-fold \( \sigma \) and proceed to make the next guess \( \sigma/i+1 \in L \). Clearly \( T_2(M') = \lim(L) \).

(d) Follows by (c) above and Proposition 5.1.4b.

As a corollary of Theorems 5.2.4 and 5.2.5 we obtain:

**Corollary 5.2.6** For each \( L \) in \( A^1\text{-}DTML^\omega \), \( L = \mathcal{E}x(t\ell)(L) \).

**Lemma 5.2.7**

(a) \( \{L\Sigma^\omega|L \in \text{DCS}\} = \{L\Sigma^\omega|L \in \text{CS}\} = \{L\Sigma^\omega|L \in \text{RE}\} \).

(b) \( \lim(\text{DCS}) = \lim(\text{CS}) = \lim(\text{RE}) \).

**Proof.** (a) Let \( L \) be an RE set accepted by a DTA \( M \). For every \( x \in L \), let \( S(x) \) denote the space used by \( M \) on its tape in recognizing \( x \). Define \( L_1 = \bigcup_{x \in L} x \Sigma^\omega S(x)-|x| \); then clearly \( L_1 \Sigma^\omega = L \Sigma^\omega \). To show that \( L_1 \in \text{DCS} \), note that the DLBA accepting \( L_1 \) will, for each input word \( y \), make a sequence of "guesses" as to where the end of \( x \) is located s.t. \( y \in x \Sigma^\omega S(x)-|x| \) and \( x \in L \), trying to verify each guess in turn. Since \( \text{DCS} \subseteq \text{CS} \subseteq \text{RE} \) the assertion follows.

(b) Let \( L \) be an RE set, and let \( L_1 \) be as in (a) above. Let \( \sigma \in \lim(L) \); then there exists an infinite set of indices \( I \) s.t. \( \sigma/i \in L \) for each \( i \in I \). For each \( i \in I \), \( \sigma/i \Sigma^\omega S(\sigma/i)-1 \subseteq L_1 \), and in particular, \( \sigma/S(\sigma/i) \in L_1 \). Since \( I \) is infinite, so must be \( J = \{S(\sigma/i)|i \in I\} \); hence \( \sigma \in \lim(L_1) \). If \( \sigma \in \lim(L_1) \) then clearly \( \sigma \in \lim(L) \), hence \( \lim(L) = \lim(L_1) \). Since \( \text{DCS} \subseteq \text{CS} \subseteq \text{RE} \) the assertion follows. \( \square \)
By the above lemma and Theorem 5.2.5 we obtain the following characterization theorem for the families $A_i$-

**Theorem 5.2.8** (a) $A_i$-DTML$_\omega = \{L \Sigma^\omega | L \in \text{RE}\} = \{L \Sigma^\omega | L \in \text{CS}\} = \{L \Sigma^\omega | L \in \text{DCS}\}$;

(b) $A_1'$-DTML$_\omega = \text{Ninit(RE)} = \text{Ninit(CS)} = \text{Ninit(DCS)}$;

(c) $A_2$-DTML$_\omega = \text{lim(RE)} = \text{lim(CS)} = \text{lim(DCS)}$;

(d) $A_2'$-DTML$_\omega = \{\Sigma^\omega - L | \text{Elim(RE)}\} = \{\Sigma^\omega - L | \text{Elim(CS)}\} = \{\Sigma^\omega - L | \text{Elim(DCS)}\}$.

### 5.3 THE FAMILIES $A_i$-DTML$_\omega$ AND DTML$_\omega$

The characterizations in Section 5.2 help establish the inclusion relations among the families $A_i$-DTML$_\omega$.

**Lemma 5.3.1** Let $\Sigma = \{0,1\}$, then: (a) $0^\omega \in A_1$-DTML$_\omega$; (b) $\Sigma^* 1^\omega \in A_1'$-DTML$_\omega$;

(c) $\Sigma^* 0^\omega \in A_2$-DTML$_\omega$; (d) $\{0^* 1\}^\omega \in A_2'$-DTML$_\omega$.

**Proof.** Let $L = 0^\omega$; then $L \notin A_1$-DTML$_\omega$ by Theorem 5.2.8, and by Proposition 5.1.4a, $\Sigma^\omega - L = \Sigma^* 1^\omega \notin A_1$-DTML$_\omega$. As is shown in [Eil, p.390] for no $\omega$-language $L_1$ does $\text{lim}(L_1) = \Sigma^* 0^\omega$, hence by Theorem 5.2.8 $\Sigma^* 0^\omega \notin A_2$-DTML$_\omega$.

By Proposition 5.1.4b $\Sigma^\omega - \Sigma^* 0^\omega = \{0^* 1\}^\omega \notin A_2'$-DTML$_\omega$.

The following lemma is obvious:

**Lemma 5.3.2** For each $L$ in DTML$_\omega$ ($A_i$-DTML$_\omega$, $i = 1,1',2,2'$) over alphabet $\Sigma$, and for any $x \in \Sigma^*$, $x \backslash L = \{s \in \Sigma^* | x s \in L\}$ is in DTML$_\omega$ ($A_i$-DTML$_\omega$, $i = 1,1',2,2'$ respectively).

The hierarchy of the families $A_i$-DTML$_\omega$ is summarized in the following theorem:

**Theorem 5.3.3**

(a) Each of the pairs ($A_1$-DTML$_\omega$, $A_1'$-DTML$_\omega$) and ($A_2$-DTML$_\omega$, $A_2'$-DTML$_\omega$) is a pair of incommensurate families.
(b) \(\text{A}1-\text{DTML}_\omega, \text{A}1'-\text{DTML}_\omega, \text{A}2-\text{DTML}_\omega, \text{A}2'-\text{DTML}_\omega\) are each incomparable with the class of \(\omega\)-regular languages;

(c) \((\text{A}1-\text{DTML}_\omega \cup \text{A}1'-\text{DTML}_\omega)\) is properly included in \((\text{A}2-\text{DTML}_\omega \cap \text{A}2'-\text{DTML}_\omega)\);

(d) \((\text{A}2-\text{DTML}_\omega \cup \text{A}2'-\text{DTML}_\omega)\) is properly included in \(\text{DTML}_\omega\);

(e) The \(\omega\)-regular languages are properly included in \(\text{DTML}_\omega\).

**Proof.** Let \(\Sigma = \{0,1\}\). (a) Let \(L = 0^\omega\); then \(L \in \text{A}1'-\text{DTML}_\omega\) and \(\Sigma^\omega-L = \Sigma^*0^\omega \in \text{A}1-\text{DTML}_\omega\), but \(L \notin \text{A}1-\text{DTML}_\omega\) and \(\Sigma^\omega-L \in \text{A}1'-\text{DTML}_\omega\) (Lemma 5.3.1 (a),(b)). Let \(L = \Sigma^*0^\omega\); then \(L \in \text{A}2'-\text{DTML}_\omega\) and \(\Sigma^\omega-L \in \text{A}2-\text{DTML}_\omega\), but \(L \notin \text{A}2-\text{DTML}_\omega\) and \(\Sigma^\omega-L \notin \text{A}2'-\text{DTML}_\omega\) (Lemma 5.3.1 (c),(d)).

(b) Let \(L = \{0^n1^n | n > 1\}\); clearly \(L \in \text{A}1-\text{DTML}_\omega\) and therefore \(\Sigma^\omega-L \notin \text{A}1'-\text{DTML}_\omega\), but \(L \) and \(\Sigma^\omega-L\) are not \(\omega\)-regular languages. On the other hand the examples of \(\omega\)-languages in Lemma 5.3.1 show that the \(\omega\)-regular languages are not included in \(\text{A}1-\text{DTML}_\omega\) for \(i = 1,1',2,2'\). (c) One can easily verify that \((\text{A}1-\text{DTML}_\omega \cup \text{A}1'-\text{DTML}_\omega) \subseteq (\text{A}2-\text{DTML}_\omega \cap \text{A}2'-\text{DTML}_\omega)\) and \((\text{A}2-\text{DTML}_\omega \cup \text{A}2'-\text{DTML}_\omega) \subseteq \text{DTML}_\omega\). Neither of the \(\omega\)-regular languages \(1^*0^\omega\) and \(\Sigma^\omega-1^*0^\omega = (0 \cup 1^*0^\omega)\Sigma^\omega U 1^\omega\) is of the form \(L\Sigma^\omega\), for any \(L \subseteq \Sigma^*\); therefore by Theorem 5.2.8, \(1^*0^\omega \in \text{A}1-\text{DTML}_\omega \cup \text{A}1'-\text{DTML}_\omega\). Clearly \(1^*0^\alphaW \in \text{A}2-\text{DTML}_\omega \cap \text{A}2'-\text{DTML}_\omega\). (d) Let \(L = 1^*0^\omega \cup 0\{0^*1\}^\omega\). By Lemma 5.3.1 \(0 \not\in \text{A}2'-\text{DTML}_\omega\) and \(1 \not\in \text{A}2-\text{DTML}_\omega\), thus by Lemma 5.3.2 \(L \in \text{A}2-\text{DTML}_\omega\) and \(L \in \text{A}2'-\text{DTML}_\omega\), but clearly \(L \notin \text{DTML}_\omega\). (e) Obvious.

**Theorem 5.3.4** (a) \(\text{A}i-\text{DTML}_\omega (i = 1,1',2,2')\) is closed under union and intersection but not under complementation. (b) \(\text{DTML}_\omega\) is closed under union, intersection and complementation.

**Proof.** (a) Let \(L_1, L_2\) be in \(\text{A}i-\text{DTML}_\omega\) for some \(i = 1,1',2,2'\). By Theorem 5.1.1 \(L_j = T_j((M_j,F^{(j)}))\) for some \(\omega\)-DTA's \(M_j\), \(j = 1,2\). As in

Technion - Computer Science Department - Technical Report CS0076 - 1976
Theorem 3.3.1, one can construct a DTM \( M \) which, for each \( \sigma \in \Sigma^\omega \), simulates step by step both \( M_1 \) and \( M_2 \) on \( \sigma \). Since \( M_1 \) and \( M_2 \) have Property C, \( M \) has it too. For each \( i = 1,1',2,2' \), one can define, in terms of \( F^{(1)} \) and \( F^{(2)} \), two collections of designated state sets, \( H_i, D_i \) for \( M \) s.t.
\[
L_1 \cap L_2 = T_i((M,H_1)) \quad \text{and} \quad L_1 \cup L_2 = T_i((M,D_1)).
\]
The examples in Lemma 5.3.1 show that each of the families \( \text{Al-DTML}_\omega \), \( i = 1,1',2,2' \), is not closed under complementation. (b) By the same argument as in (a) above \( \text{DTML}_\omega \) is also closed under union and intersection. As for complementation, if \( M = (K,\Sigma,\Gamma,\delta,q_0,F) \) is an \( \omega \)-DTA, then \( T(M_1) = \Sigma^\omega \cap T(M) \), where \( M_1 = (K,\Sigma,\Gamma,\delta,q_0,2^K-F). \)

A typical and most natural result relating \( \omega \)-languages to finite string languages is the following:

**Init Lemma.** If \( L \) is an \( \omega \)-language recognizable by a type \( X \) \( \omega \)-automaton [generated by a type \( X \) \( \omega \)-grammar], then the (finite string) language \( \text{Init}(L) \) is recognizable by an automaton [generated by a grammar] of the same type \( X \) (e.g. the Init of an \( \omega \)-CFL is a CFL).

In previous papers [Co&Gol-3] the Init Lemma was shown to hold for the following types of automata: (1) finite state automata; (2) pushdown automata, and (3) deterministic pushdown automata. Using similar techniques, one can also establish the Init Lemma for more powerful types of automata, e.g. \( 1 \)-way non-deterministic and deterministic stack automata.

However, it turns out that the Init Lemma no longer holds for deterministic \( \omega \)-Turing machines; in fact, even the Init of an \( \omega \)-language in \( \text{Al}^1\text{-DTML}_\omega \) need not be recursively enumerable, as is shown in next theorem.

**Theorem 5.3.5** (a) For each \( L \in \text{Al}^1\text{-DTML}_\omega \), \( \text{Init}(L) \) is recursively enumerable.
(b) There exists an \( \omega \)-language \( L \) in \( \text{Al}^1\text{-DTML}_\omega \) s.t. \( \text{Init}(L) \) is not recursively enumerable.
Proof. (a) $L = L_1\Sigma^\omega$ for some RE set (Theorem 5.2.8), hence $\text{Init}(L) = \text{Init}(L_1) \cup L_1\Sigma^*$ is an RE set. (b) Let $\{M_i\}_{i \geq 1}$ be an effective enumeration of all DTA's and let $L = \{0_i^i\omega | i = 1, 2, \ldots, 0_i^i \in T(M_i)\} \cup \{0^\omega\}$. Construct a 2-$\omega$-DTA $M$ with a single designated set $F$ that $1^i$-accepts $L$ as follows: Given an $\omega$-input $\sigma$, $M$ first checks that $\sigma$ starts with $0$ and then moves right until the first $1$ is reached, thus obtaining the prefix $0_i^i$ for some $i \geq 1$. $M$ then generates a description of machine $M_i$ and starts simulating $M_i$ on input $0_i^i$ on its working tape. In the meanwhile, $M$ keeps moving right on the input tape, checking that the remaining part of $\sigma$ is $1^\omega$. All the above operations are carried out while $M$ is in states of the designated set $F$. If indeed $\sigma = 0_i^i\omega$ and $0_i^i$ is not accepted by $M_i$, then either the simulation of $M_i$ on $0_i^i$ will never end, in which case $M$ will stay forever in states of $F$, or else $M_i$ halts on $0_i^i$ in a non-final state, in which case $M$ will enter a new state $q_F \in F$ in which it will continue scanning the rest of $\sigma$, checking that $\sigma \in 0^i+1^\omega$. If, however, it turns out that either $\sigma \notin 0^i+1^\omega U 0^\omega$ or $0_i^i \notin T(M_i)$, then $M$ enters some traverse state $q_T \notin F$ in which it scans the rest of $\sigma$. If $\sigma = 0^\omega$, $M$ keeps moving right forever looking for a $1$ in a state from $F$, hence $0^\omega$ is accepted. Clearly $T_{1_i}(M) = L$. Since $\text{Init}(L) \cap 0^+1 = = \{0_i^i | 0_i^i \notin F \}$ is not an RE set, $\text{Init}(L)$ is not an RE set either. \hfill $\Box$

6. NON-DETERMINISTIC $\omega$-TA's

In Section 3 the class of $\omega$-languages recognized by non-deterministic $\omega$-DTA's was characterized as TYPE$_\omega^0$ - the class of $\omega$-languages generated by $\omega$-PSG's. In this section we focus our attention on variants of non-deterministic $\omega$-TA's, show that here all $i$-acceptance modes are equivalent and establish the existence of $\omega$-languages in TYPE$_\omega^0$ which cannot be recognized by deterministic $\omega$-TA's.
Theorem 6.1 For every m-\omega-TA there can be constructed an equivalent \omega-TA.

Proof. The proof resembles the proof of Theorem 5.1.1. The only difference is that here \( M_2 \) has one extra state \( q_R \), which it enters each time a new square is scanned on the input tape \( \sigma \). \( q_R \) belongs to all the designated state sets of \( M_2 \), thus guaranteeing that \( \omega \)-inputs \( \sigma \) for which \( M \) has no c.n.o. run will not get accepted by \( M_2 \).

It turns out that for non-deterministic \( \omega \)-TA's, 1'-acceptance is as powerful as 3-acceptance. It will be now shown that for each \( i = 1, 1', 2, 2' \), the family of \( \omega \)-languages 1'-accepted by \( \omega \)-TA's coincides with the whole family \( \text{TYPE}_\omega \).

Theorem 6.2 For each \( i = 1, 1', 2, 2' \), the class \( L_i \) of \( \omega \)-languages 1'-accepted by \( \omega \)-TA's equals \( \text{TYPE}_\omega \).

Proof. Since by definition \( L_1 \) and \( L_1' \) are both included in \( L_2 \) and in \( L_2' \), which in turn are subsets of \( \text{TYPE}_\omega \), it suffices to show that \( \text{TYPE}_\omega \subseteq L_1' \subseteq L_1 \).

Let \( M = (M', F) \) be an \( \omega \)-TA. Since \( L_1' \) is closed under union and \( T(M) = U ((M', \{H\})) \) we may assume w.l.o.g. that \( M = (M', F) \) is a U-\( \omega \)-TA. We now construct a U-\( \omega \)-TA \( M_1 \) which 1'-accepts \( L = T(M) \). \( M_1 \) will have a c.n.o. run only on tapes which belong to \( L \). Given input \( \sigma \in \Sigma^\omega \), \( M_1 \) simulates \( M \) on \( \sigma \). During the simulation \( M_1 \) marks on its tape a "regression point", changing its location from time to time as is described below. In the beginning the regression point is on the first symbol of \( \sigma \). \( M_1 \) returns to the regression point after each step of the simulated computation. If in the most recent step \( M \) has passed through a state not in \( F \), \( M_1 \) will move the regression point back to the beginning of \( \sigma \). If, however, since the last time the regression point was moved, \( M \) has passed through all the states of \( F \),
$M_1$ will move the regression point to the rightmost symbol of $\sigma$ reached so far by $M$. In all other cases the regression point will remain where it is.

Obviously if $\sigma \in T(M)$, then for each run of $M$ on $\sigma$, the regression point in the corresponding run of $M_1$ will never reach beyond a certain point on $\sigma$, thus the run of $M_1$ on $\sigma$ will not be c.n.o. On the other hand, if $\sigma \in T(M)$, then after some finite number of steps, the regression point will never move left to the beginning of $\sigma$, but will move right an unbounded number of times, and the resulting run of $M_1$ on $\sigma$ will be c.n.o. It follows that $T_{11}(M_1) = T(M)$.

To prove that $L_{11} \subseteq L_1$, let $M = (M', F)$ be an $\omega$-TA. The U-$\omega$-TA $M_1$ that $I$-accepts $T_{11}(M)$ has the starting state of $M$ as its singleton designated set. For $\omega$-input $\sigma$, $M_1$ simulates $M$ so long as it stays within at least one of the sets in $F$; otherwise $M_1$ is blocked.

In Section 5 we saw that in the case of deterministic $\omega$-TA's, for $i = 1, 1', 2$ all $\omega$-languages in $A_i$-DTML can be $i$-accepted by three state $\omega$-DTA's, and all $\omega$-languages in $A_2$-DTML can be $2'$-accepted by four state deterministic machines. As for $3$-acceptance by deterministic $\omega$-TA's, the situation is rather different as there is an infinite "state complexity" hierarchy of the machines, i.e. for each number $n > 0$ there exists $m > n$ s.t. there exists an $\omega$-language $L$ accepted by an $m$-state $\omega$-DTA, which cannot be accepted by any $\omega$-DTA with $n$ or less states ([CoGo4]). As far as non-deterministic $\omega$-TA's are concerned, since by the above theorem all $I$-acceptance modes, $i = 1, 1', 2, 2'$ are equivalent to the $1'$-acceptance mode, two states suffice for recognizing all $\omega$-languages in TYPEO.$\omega$.

**Corollary 6.3** Every $L$ in TYPEO.$\omega$ can be accepted ($I$-accepted for $i = 1, 1', 2, 2'$) by a two-state U-$\omega$-TA.
Proof. Let $L \in \text{TYPE}_0^\omega$; then by Theorem 6.2 and Lemma 5.1.3 $L$ can be
1'-accepted by some U-$\omega$-TA $M = (M', F)$. We can simulate the operation of
$M$, while in $F$, using only two states ([Sha]) and block $M$ if it moves
out of $F$. Thus a two-state $U$-$\omega$-TA $M_1$ that 1'-accepts $L$ can be con-
structed s.t. $L = T_1(M_1)$ for each $1 = 1', 2, 2', 3$. A similar construction
will yield a two-state $\omega$-TA that 1-accepts $L$. □

In Theorem 3.6 we saw that $\text{TYPE}_0^\omega$ is closed under union and intersection.

Theorem 6.4 $\text{TYPE}_0^\omega$ is not closed under complementation.

Proof. Let $\{M_i\}_{i \geq 1}$ be an effective enumeration of all two-state $U$-$\omega$-TAs
and let $L = \{0^11^\omega|0^11^\omega \in T(M_i)\}$. One can easily build an $\omega$-TA that accepts
$L$. By Theorem 6.3 above $L = T(M_j)$ for some $j$. But by the usual diagonal-
alization argument, $\Sigma^\omega L \in \text{TYPE}_0^\omega$. □

An important corollary of the above and Theorem 5.3.4b is that non-
deterministic $\omega$-TAs are strictly more powerful than the deterministic ones.

Theorem 6.5 $\text{DTML}^\omega \not\subseteq \text{TYPE}_0^\omega$.

As shown in the next theorem oscillating runs can be altogether avoided
in $\omega$-TAs.

Theorem 6.6 For every $\omega$-TA $M$ there can be constructed an equivalent $\omega$-TA
$M_1$ in which every run is c.n.o.

Proof. Let $M$ be an $\omega$-TA. The desired $\omega$-TA $M_1$ has its tape divided into
2 tracks: $\alpha$ - the input track and $\beta$ - the working tape on which $M_1$ simulates
$M$. $M_1$ has an auxiliary symbol $X$ and a new state $q_X$. Each simulation step
may be followed by an optional "X-marking" phase; then follows a single
move to the right on the $\alpha$ track and then a corresponding folding process
of the $\beta$ track w.r.t. the currently scanned square on the $\alpha$ track.
In the "X-marking" phase, $M_1$ enters state $q_X$ and marks with $X$ the $(i+1)$-th square on the $\beta$ track, where $i$ is the total number of times the "X-marking" phase has taken place before during the run. The "X-marking" of a square designates a guess that $M$ will never again return to that square. If $M$ reaches a marked square on $\beta$, $M_1$ stops the simulation and scans the rest of the input in a traverse state which does not belong to any designated set. Adding $q_X$ to the designated sets of $M$ we obtain an $\omega$-TA $M_1$ equivalent to $M$, in which every run is c.n.o.

Fig. 6.1 below illustrates the hierarchy of $\omega$-language families corresponding to the various types of $\omega$-TA's.
CONCLUSION

As we have shown in this paper, the theory of $\omega$-type Turing acceptors differs considerably from the classical theory of Turing acceptors. Non-deterministic and deterministic $\omega$-TA's were found to be rather different both in their recognition power (the non-deterministic machines being strictly more powerful w.r.t. each i-acceptance mode) and in their properties. The i-acceptance modes, $i=1,1',2,2',3$, were shown to be pairwise inequivalent w.r.t $\omega$-DTA's, with the corresponding families $A_i$-DTML$_{\omega}$ forming a hierarchy within DTML$_{\omega}$, whereas for non-deterministic $\omega$-TA's all i-acceptance modes turned out to be of the same recognition power.

Many of the results were proved by using some new techniques specific for $\omega$-tapes; for instance, the "folding process" and "relative folding process" for $\omega$-tapes were applied for converting $\omega$-TA's into $\omega$-TA's with Property C.

From the results in Sections 5 and 6 it follows that for non-deterministic $\omega$-TA's one can construct a universal $\omega$-TA, and similarly w.r.t. i-acceptance, $i=1,1',2,2'$ in deterministic $\omega$-TA's, a universal $i$-accepting $\omega$-DTA can be constructed.

In a forthcoming paper [Co&Go4] it is shown that w.r.t. 3-acceptance there exists no $\omega$-DTA which is universal for all $\omega$-DTA's. In the above paper, two infinite complexity hierarchies for $\omega$-DTA's are exhibited, one corresponding to the number of states and the second corresponding to the number of designated state sets in the machine. Concrete (and rather simple) examples of $\omega$-languages characterizing each of the complexity classes, and
also one example of an "inherently non-deterministic" \( \omega \)-language and another of an \( \omega \)-language outside \( \text{TYPEO}_\omega \) are constructed without using the standard diagonalization technique. \( \omega \)-DTA's without Property C are studied and shown to be strictly more powerful than those with Property C, yet less powerful than the non-deterministic \( \omega \)-TA's.
1. In [Co&Go2] it was shown that for every $\omega$-PSG $G$, the $\omega$-language generated by $G$ by leftmost derivations only is an $\omega$-CFL.

2. Note that if we changed the definition of acceptance (i-acceptance) in $\omega$-TM's s.t. oscillating runs would also be considered as possibly accepting runs (i.e. in the definition of $T_i(M)$, 'c.n.o. run' would be replaced by 'complete run'), then the folding process would provide a way of converting any $\omega$-TA ($\omega$-DTA) into an equivalent $\omega$-TA ($\omega$-DTA) in which every run is c.n.o. (because infinite loops can be easily detected in single tape machines). As is shown in [Co&Go4], this is not so w.r.t. our original definition of acceptance, since not every $\omega$-DTA can be converted into an equivalent $\omega$-DTA with property C.

3. A [deterministic] Turing acceptor (TA) [DTA] is a couple $M = (M', F)$ where $M' = (K, \Sigma, \Gamma, \delta, q_0)$ is a TM[DTM] and $F \subseteq K$. The language accepted by $M$ is: $L(M) = \{ w \in \Sigma^* | (q_0, \omega, 1) \xrightarrow{\star M} (q, \alpha, 1) \text{ for some } q \in F, \alpha \in \Gamma^* \text{ and natural number } l \}$.

A [deterministic] linear bounded automaton (LBA)[DLBA] is a TA[DTA] which never leaves those squares on which the finite input is placed.
REFERENCES


[C&Go3] Cohen, R.S. and A.Y. Gold: "ω-Computations on Deterministic Pushdown Machines", TR #56, Computer Science Dept., Technion-IIT, Haifa, November 1975.


