ON THE COMPLEXITY OF \( \omega \)-TYPE TURING ACCEPTORS

by

Rina Cohen* and Arie Gold**

Technical Report #89
December 1976
ABSTRACT

Turing machines are considered as recognizers of sets of infinite (ω-type) sequences, so called ω-languages. The basic results on such ω-type Turing acceptors were presented in a preceding paper. This paper focuses on the theory of deterministic ω-type Turing acceptors (ω-DTA's), which turns out to be crucially different from the "classical" theory of Turing machines. It is shown that there exists no ω-DTA which is universal for all ω-DTA's. Two infinite complexity hierarchies for ω-DTA's are established, the "states hierarchy", corresponding to the number of states in the machine, and the "designated sets hierarchy", corresponding to the number of designated sets of states used in the recognition. Concrete examples of ω-languages characterizing each of the complexity classes are exhibited. Two additional examples of interesting ω-languages are presented: (i) An ω-language which is "inherently non-deterministic", i.e. can be recognized by a non-deterministic Turing acceptor but by no deterministic acceptor. (ii) An ω-language which cannot be recognized even by a non-deterministic Turing acceptor. The above examples are constructed without using diagonalization. Oscillating ω-DTA's, i.e. ω-DTA's which are allowed to oscillate on ω-inputs, are also considered and are shown to be strictly more powerful than non-oscillating ω-DTA's, yet strictly less powerful than non-deterministic ω-Turing acceptors.
\( \omega \)-TA's, relevant to the current paper, are presented in Section 2 below.

In this paper we study some complexity issues concerning the various \( \omega \)-TA models. First, it is shown that the number of states in deterministic \( \omega \)-TA's cannot be bounded. Two infinite complexity hierarchies, one corresponding to the number of states and the other to the number of designated state sets in the machines, are established in Section 3. The complexity classes of each hierarchy are then illustrated by concrete and rather simple examples of \( \omega \)-languages, each requiring at least a certain number of states (designated state sets) to be recognized.

It follows that there exists no deterministic \( \omega \)-TA, which is universal for all deterministic \( \omega \)-TA's.

While the original definition of deterministic \( \omega \)-TA's (as studied in [CoGo 4]) allows only for "non-oscillating" machines, i.e. machines whose reading head scans each position on the \( \omega \)-input only finitely many times, in Section 4 we consider the model of "oscillating" deterministic \( \omega \)-TA's. These are deterministic \( \omega \)-TA's which are allowed to oscillate, but which can accept an \( \omega \)-input only via a non-oscillating computation. Such machines can be viewed as equipped with an extra mechanism for recognizing oscillations in an infinite computation. In Section 5, we exhibit a concrete example of an \( \omega \)-language, which is recognizable by an oscillating deterministic \( \omega \)-TA, but which cannot be recognized by a non-oscillating machine of this type. Thus the oscillating deterministic \( \omega \)-TA's are strictly more powerful than the non-oscillating ones, yet they are shown to be still less powerful than the non-deterministic \( \omega \)-TA's.

In Section 6 we exhibit a concrete and simple example (involving no diagonalization) of an \( \omega \)-language LC, which is "inherently non-deterministic".
i.e. is recognizable by a non-deterministic $\omega$-TA, but is not recognizable by any deterministic $\omega$-TA (of either variety).

Finally, in Section 7 an example of an $\omega$-language $L_N$ which is not recognizable by any $\omega$-TA (deterministic or not) is given; this $\omega$-language is defined without using the diagonalization technique.
1. 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 $x = \prod_{i=1}^{k} a_i$, $a_i \in \Sigma$, $i = 1, \ldots, k$, $k = 0, 1, \ldots$. $k = |x|$ is the length of $x$; $\varepsilon$ denotes the empty string and $\Sigma^*$ denotes the set of all finite strings over $\Sigma$.

For any set $S$, let $|S|$ denote the cardinality of $S$. Let $\mathbb{N}$ denote the set of natural numbers.

**Definition 1.1** 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:

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

$L^\omega$ 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 $j \geq 1$,

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

The following is the standard definition of a Turing machine with a single semi-infinite tape ([H&U]).

**Definition 1.2** 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 $K \times T$ to subsets of $K \times \Gamma \times \{L, R, S\}$. A **configuration** of $M$ is the 3-tuple $(q, \sigma, i)$, where $q \in K$, $\sigma \in \Gamma^\omega$ and $i$ is a natural number.

The relations $|_{M}$ and $|_{M}^*$ 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 1.3** Let $M = (K, \Sigma, \Gamma, \delta, q_0)$ be a TM and let $\sigma \in \Gamma^\omega$. An infinite sequence of configurations $r = \{(q_i, \gamma_i, j_i)\}_{i=1}^\omega$ is called a run of $M$ on $\sigma$ iff:

(a) $(q_1, \gamma_1, j_1) = (q_0, \sigma, 1)$;

(b) for each $i \geq 1$ $(q_i, \gamma_i, j_i) |_{M} = (q_{i+1}, \gamma_{i+1}, j_{i+1})$;

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. $\forall k \geq 1 \exists k > \ell$ s.t. $j_k = n_0$.

A complete non-oscillating (abbreviated c.n.o) run is a complete run which does not satisfy condition (d) above; i.e. $\forall n \geq 1 \exists \ell \geq 1$ 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 \( j_0 \geq 1 \) 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.

We now define a special state in which the machine simply traverses the \( \omega \)-input from left to right.

**Definition 1.4** Let \( M = (K, \Sigma, \Gamma, \delta, q_0) \) be a TM. A state \( q_T \in K \) is a traverse state iff \( \forall a \in \Gamma, \delta(q_T, a) = (q_T, a, B) \).

Clearly, if during its computation 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 1.5** Let \( M = (K, \Sigma, \Gamma, \delta, q_0) \) be an m-TM. For any run \( r \) of \( M \), on some \( \omega \)-input \( \sigma \), define \( \text{INS}_M(r) \) as the set of all states entered by \( M \) infinitely many times during run \( r \) (the subscript \( M \) will be omitted whenever \( M \) is understood).

**Definition 1.6** 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 \( (M', F) \).

An m-\( \omega \)-TA \( M = (M', F) \) is deterministic iff \( M' \) is a deterministic m-TM.
There are several distinct ways in which $\omega$-type machines can recognize an $\omega$-input. We now define five $\omega$-recognition modes, so called the "i-acceptance" modes, for $i = 1, 1', 2, 2', 3$, first introduced by Landweber [Lan] w.r.t. finite state $\omega$-acceptors.

**Definition 1.7** Let $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ be an $m$-$\omega$-TA ($\omega$-TA) where $F \subseteq 2^K$.

Define:

1. $T_1(M) = \{ \sigma \in \Sigma^\omega \mid$ there exists a c.n.o. run of $M$ on $\sigma$ during which $M$ enters at least once a state in some set of $F$ \}

2. $T_1'(M) = \{ \sigma \in \Sigma^\omega \mid$ there exists a c.n.o. run of $M$ on $\sigma$ and a designated state set $H \in F$ s.t. all states entered by $M$ during this run are in $H$ \}

3. $T_2(M) = \{ \sigma \in \Sigma^\omega \mid$ there exists a c.n.o. run $r$ of $M$ on $\sigma$ s.t. $\INS(r) \cap H \neq \emptyset$ for some $H \in F$ \}

4. $T_2'(M) = \{ \sigma \in \Sigma^\omega \mid$ there exists a c.n.o run $r$ of $M$ on $\sigma$ s.t. $\INS(r) \subseteq H$ for some $H \in F$ \}

5. $T_3(M) = \{ \sigma \in \Sigma^\omega \mid$ there exists a c.n.o. run $r$ of $M$ on $\sigma$ s.t. $\INS(r) = H$ for some $H \in F$ \}

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

Note that w.r.t. 1-acceptance and 2-acceptance, one may assume w.l.o.g. that the $m$-$\omega$-TA has only a single designated state set.

The above definitions are illustrated by the following example:

**Example:** Consider infinite sequences over $\Sigma = \{a, b, c\}$ of the form $a^2y_1^2a^3y_3^2 \ldots a^iy_1 \ldots$, where $\forall i \geq 1 \ y_i \in \{b, c\}$.

Define the following $\omega$-languages:
\[ L_0 = \{ \prod_{i=1}^{\infty} a^i y_i \mid y_1 = b \text{ or } y_1 = c \} \]

\[ L_1 = L_0 \cap (a^* b^* \Sigma^*)^\omega \quad (\text{"b" must appear infinitely many times}) \]

\[ L_2 = L_0 \cap (a^* b^* \Sigma^*)^\omega \quad (\text{"b" appears infinitely many times but "c" appears only finitely many times}) \]

Let \( M \) be a \( TM \) with a set of working states \( K_1 \) and three special states \( q_b, q_c, q_T \). Given input \( \sigma \in \Sigma^\omega \), \( M \) operates as follows:

1. Using the states of \( K_1 \), \( M \) checks that \( \sigma \) is of the required form and that each section of 'a'\'s is of length one more than the previous section of 'a'\'s; if not, \( M \) enters the traverse state \( q_T \) in which it will keep moving right forever.

2. \( M \) enters state \( q_b \) or \( q_c \) whenever it scans a new square with symbol 'b' or 'c' respectively.

Now let \( F_1 = K_1 \cup \{ q_b, q_c \} \); then the \( \omega \)-TA \( M_1 = (M, \{ F_1 \}) \) 1'-accepts \( L_0 \), while the \( \omega \)-TA \( M_2 = (M, \{ q_b \}) \) 2'-accepts \( L_1 \). If we define \( M_3 = (M, \{ F_2 \}) \) with \( F_2 = K_1 \cup \{ q_b \} \), then \( T_2(M_3) = T_3(M_3) = L_2 \); however, defining \( M_3 = (M, F) \) with \( F = \{ F_1, F_2 \} \), we obtain \( T_2(M_3) = L_0 \) whereas \( T_3(M_3) = L_1 \).

Convention: 3'-acceptance, being the most commonly used as well as the most powerful of the above i'-acceptance modes (e.g. see [Mc], [Cho], [Lan], [CoGo 2-4] and Theorem 2.3 below) will subsequently be our standard definition of \( \omega \)-acceptance, and will be referred to simply as acceptance. \( T_3(M) \), the \( \omega \)-language accepted by \( M \), will be denoted by \( T(M) \) (subscript 3 omitted).

**Definition 1.8** Two \( m \)-\( \omega \)-TA's \( M \) and \( M' \) will be called equivalent (i-equivalent for \( i = 1, 1', 2, 2' \)) iff \( T(M) = T(M') \) \( (T_1(M) = T_1(M')) \).
Note that by the above definition of acceptance (or i-acceptance) in \(\omega\)-TA's, and \(\omega\)-input \(\sigma\) may be accepted only via a c.n.o. run, that is, an incomplete or oscillating run on \(\sigma\) cannot lead to acceptance, regardless of the sequence of states entered by the machine during the run.

**Definition 1.9:** An \(m-\omega\)-TA \(M\) is said to possess Property C iff for every \(\sigma \in \mathbb{L}^\omega\) there exists a c.n.o. run of \(M\) on \(\sigma\).

Note that a non-deterministic \(m-\omega\)-TA with Property C may still have oscillating or incomplete runs. In fact, every \(m-\omega\)-TA \(M\) without Property C may easily be converted into an equivalent \(m-\omega\)-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.

It follows that Property C is not very meaningful w.r.t. non-deterministic \(\omega\)-Turing acceptors. As for deterministic machines, the distinction between machines with and without Property C is most significant as will be explained below.

Let \(M\) be a deterministic \(\omega\)-TA. If \(M\) has Property C, then for each \(\omega\)-input \(\sigma\), the unique run of \(M\) on \(\sigma\) must be c.n.o. Hence \(M\) is a machine which never oscillates on any \(\omega\)-input.

On the other hand, if \(M\) does not have Property C but is non-oscillating, i.e. has no oscillating runs on any \(\omega\)-input, then
an equivalent deterministic $\omega$-TA $M'$ with Property C can be con-
structed from $M$. $M'$ will imitate $M$ while counting the steps, so as to be able to detect an infinite loop in the run. Once $M$ is found to be in an infinite loop, or else if $M$ halts after a finite number of steps, $M'$ will enter a new non-final traverse state, in which it will scan through the rest of the input.

Thus, we have:

**Proposition:** For every non-oscillating deterministic $\omega$-TA, there can be constructed a deterministic $\omega$-TA with Property C which is $i$-equivalent to $M$ for each $i = 1', 2', 3$.

It follows that deterministic $\omega$-TA's which never oscillate are equivalent to those with Property C.

As for deterministic $\omega$-TA's with oscillations (i.e. those which might have oscillating runs), it will be shown in Section 4 below that they are strictly more powerful than the non-oscillating ones. This can be better understood if we view the oscillating machines as equipped with an extra mechanism for distinguishing oscillating runs from non-
oscillating runs (since by definition, acceptance can occur only for non-oscillating runs).

We shall henceforth distinguish between the two families of deter-
ministic $\omega$-TA's - the **non-oscillating** deterministic $\omega$-TA's (i.e. those which never oscillate on any $\omega$-input), bearing in mind that such machines can always be converted into equivalent machines with Property C, and the **oscillating** deterministic $\omega$-TA's.
Notation 1.10  A deterministic $\omega$-TA with Property C will be denoted by $\omega$-DTA, while a deterministic oscillating $\omega$-TA will be denoted by $\omega$-os-DTA. The family of $\omega$-languages accepted by $\omega$-DTA's $[\omega$-os-DTA's] will be denoted by $\text{DTML}_\omega$ $[\text{os-DTML}_\omega]$.

Let $\text{TYPE}_\omega$ denote the family of $\omega$-languages accepted by $\omega$-TA's.

The notation $\text{TYPE}_\omega$ originates from a characterization of the $\omega$-TA languages as the $\omega$-languages generated by type 0 $\omega$-grammars [Co&Go 4].

Definition 1.11  An $\omega$-language is $\omega$-regular iff $L$ is of the form

\[ \bigcup_{i=1}^{k} A_i B_i^\omega, \text{ where } k = 1, 2, \ldots \text{ and } A_i, B_i, 1 \leq i \leq k, \text{ are regular languages.} \]

$\omega$-regular languages have been characterized as the $\omega$-languages accepted by deterministic (non-deterministic) $\omega$-type finite state acceptors ([Mc], [Cho]).
2. A SUMMARY OF PREVIOUS RESULTS

The models of $\omega$-TA and of $\omega$-DTA (with Property C) have been extensively studied in [Co&Go 4]. We shall present here a brief summary of those results which are relevant to the present paper.

The theorems listed in this section are all taken from [Co&Go 4].

For each $i = 1, 1', 2, 2'$, let $\text{Ai-DTML}_i(\omega)$ denote the family of $\omega$-languages $i$-accepted by $\omega$-DTA's.

**Theorem 2.1** Every $\omega$-language in $\text{Ai-DTML}_i(\omega)$ ($i = 1, 1', 2, 2'$) can be $i$-accepted by a three-state $\omega$-DTA with a single designated set.

In Section 3 it is shown that the above result cannot be extended to 3-acceptance in $\omega$-DTA's. However, we do have an analogous result for nondeterministic $\omega$-TA's.

**Theorem 2.2** Each $\omega$-TA can be replaced by an equivalent $\omega$-TA with only two states and a single designated set.

A comparison among the various $i$-acceptance modes for $\omega$-DTA's was made, yielding the following hierarchy of families $\text{Ai-DTML}_i(\omega)$.

**Theorem 2.3** (a) The family of $\omega$-regular languages is incomparable with each of the families $\text{Ai-DTML}_i(\omega)$, $i = 1, 1', 2, 2'$. All of the above families are properly contained in $\text{DTML}(\omega)$.

(b) $A2\text{-DTML}_\omega$ and $A2'\text{-DTML}_\omega$ are incomparable and both properly contain $\text{AI-DTML}_\omega$ and $\text{AI'}\text{-DTML}_\omega$.

By contrast, for nondeterministic machines all $i$-acceptance modes, $i = 1, 1', 2, 2', 3$, turned out to be equivalent.
Theorem 2.4  For each \( i = 1,1',2,2' \), the class of \( \omega \)-languages i-accepted by \( \omega \)-TA's coincides with \( \text{TYPEO}_\omega \).

The next result implies that for each oscillating \( \omega \)-TA there can be constructed an equivalent non-oscillating \( \omega \)-TA.

Theorem 2.5  Every \( \omega \)-TA can be replaced by an equivalent (2-equivalent) \( \omega \)-TA in which every run on every \( \omega \)-input is c.n.o.

W.r.t. each machine type, multitape machines are no more powerful than single tape machines.

Theorem 2.6  For \( m \geq 2 \), each \( m-\omega \)-TA [\( m-\omega \text{-DTA} \)] can be replaced by an equivalent (i-equivalent for \( i = 1,1',2,2' \)) \( \omega \)-TA [\( \omega \text{-DTA} \)].

The Folding Process

We now define a process of "folding forward" a Turing machine semi-infinite tape so that all information written on the tape is continuously carried forward, and thus can be retrieved without having to re-scan the initial segment of the tape. This "folding process" (which was also used in [Co&Go 4]), will enable us to simulate an oscillating \( \omega \)-TM by a non-oscillating one. However, the simulation will not always result in an equivalent machine, as will be explained below.

Definition 2.7  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 \( k\backslash\eta \) contains \( \sigma \) with its initial segment \( a_i \).
$a_1 \ldots a_{k-1}$ folded forwards as is shown in Figure 2.1 below

\[ \sigma: \begin{array}{c}
  a_1 & a_2 & \ldots & a_{k-1} & a_k & \ldots & a_{2k-2} & a_{2k-1} \\
  X & X & \ldots & X & a_k & \ldots & a_{2k-2} & a_{2k-1} \\
  X & X & \ldots & X & a_{k-1} & \ldots & a_i & X & \ldots
\end{array} \]

Figure 2.1

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

Lemma 2.8 (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 fixed integer $k > 0$, for every $i \geq 2$, once $M$ reaches $\sigma(i)$, within the next $k$ steps $M_1$ leaves the initial segment $\sigma/i-1$ and never returns to it.

Proof $M_1$ simulates $M$ on $\sigma$; for each $i \geq 2$, whenever $M$ scans $\sigma(i)$ for the first time, $M_1$ will create the $i$-folded version of $M$'s tape, on which 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 Figure 2.2.
Remark 2.9  
(1) Note that machine $M_1$ constructed in Lemma 2.8 above will have a c.n.o. run corresponding to each complete run of $M$. Thus $M_1$ may also accept an $\omega$-input $\sigma$ by simulating an oscillating run of $M$ on $\sigma$. Only in case $M$ is a non-oscillating machine will the construction of $M_1$ above yield a machine equivalent to $M$.

(2) The above folding process can be utilized to show that oscillating $\omega$-DTA's which cannot recognize oscillations are just as powerful as non-oscillating $\omega$-DTA's. That is, if we changed the definition of acceptance s.t. complete oscillating runs would also be considered as possibly accepting runs, then the folding process could be used to convert any oscillating $\omega$-DTA into an "equivalent" (in the new sense) $\omega$-DTA with Property C. \hfill $\Box$

Lemma 2.10 (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 $k > 0$, when given an $\omega$-input $\sigma$, $M_1$ simulates $M$ on $\sigma$ s.t. for each $i \geq 2$, within at most $k$ 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 $\beta/i-1$
3.1 The State Complexity Hierarchy

Definition 3.1 Let DTMS(n)_ω denote the class of ω-languages which are accepted by ω-DTA's with at most n states.

We will show that the classes DTMS(n)_ω constitute an infinite hierarchy of ω-language families within DTML_ω. This hierarchy will be called the "state complexity" hierarchy.

Theorem 3.2 For every integer n > 0 there can be found an integer m > n and an ω-language L s.t. L ∈ DTMS(m)_ω but not DTMS(n)_ω i.e. L is recognized by an m-state ω-DTA but cannot be recognized by any ω-DTA with n or less states.

Proof. Let K be a set of n states and let \{S_j\}_{j=1}^{k(n)} be any enumeration of all subsets of \(2^K\), where \(k(n) = 2^{2n}\). Let \(\{M_i\}_{i \geq 1}\) be an effective enumeration of all pairs \((M',F')\), where \(M'\) is a DTM with K as the set of states and \(F' \subseteq 2^K\). Define \(K_M = \bigcup_{j=1}^{k(n)} K(j)\), where \(K(j) = \{q(j) | q \in K\\}\). For \(1 \leq j \leq k(n)\), define for \(D \subseteq K(j)\) \(\text{proj}(D) = \{q \in K | q(j) \in D\}\). Define the ω-DTA \(M = (K_M,H,\delta,F_0,F_M)\), where H is an auxiliary set of working states s.t. \(H \cap K_M = \emptyset\), \(q_0 \in H\) and \(F_M = \bigcup_{j=1}^{k(n)} \{D \cup D' | D' \subseteq H, D \subseteq K(j)\} \& \text{proj}(D) \notin S_j\). For ω-input \(0^i\omega\), \(M\) generates an encoding of machine \(M_i = (M',F')\), and for \(j_0\) s.t. \(F' = S_{j_0}\), \(M\) simulates \(M_i\) in states from \(K(j_0)\). The tape of \(M\) is divided into two tracks: α - the input track and β - the working track. Every single simulation step on the β track is followed by a move of one square to the right on the α track, which, in turn, is followed by a corresponding folding process of the β track w.r.t. the currently scanned square on the α track (Lemma 2.10). In case \(M_i\) halts on \(0^i\omega\), \(M\) stops the simulation, enters a traverse
state and continues moving to the right on the α track. Otherwise, by
definition \( \sigma = 0^i \omega \in T(M_1) \) iff \( \text{INS}(\sigma) \in S_j \) (Definition 1.5), then if
the run of \( M_1 \) on \( 0^i \omega \) is c.n.o. \( 0^i \omega \in T(M_1) \) iff \( 0^i \omega \in T(M) \).

Hence \( T(M) \neq T(M_1) \) for all \( M_1 \) which have Property C (since \( M \) itself
does not oscillate, the simulation of oscillating machines is irrelevant).

It follows that \( T(M) \in DTMS(m)_\omega \setminus DTMS(n)_\omega \), where \( m = |K_M \cup H| \) is of
order \( n \cdot 2^m \).

\[ \Box \]

Corollary 3.3 For each integer \( n \geq 1 \) there can be found an integer
\( m > n \) s.t. \( DTMS(n)_\omega \subseteq DTMS(m)_\omega \subseteq DTML \).

Corollary 3.4 There exists no universal \( \omega \)-DTA.

3.2 The Designated Set Complexity Hierarchy

Definition 3.5 Let \( DTMF(n)_\omega \) denote the class of \( \omega \)-languages which are
accepted by \( \omega \)-DTA's with at most \( n \) designated sets.

The "designated set complexity" hierarchy in \( DTML_\omega \) will be now
established.

Theorem 3.6 For every integer \( n \geq 1 \) there can be found an integer
\( m > n \) and an \( \omega \)-language \( L \) s.t. \( L \in DTMF(m)_\omega \setminus DTMF(n)_\omega \), i.e. \( L \) is
recognized by an \( \omega \)-DTA with \( m \) designated sets but cannot be recognized
by any \( \omega \)-DTA with \( n \) or less designated sets.

Proof. The proof resembles the proof of Theorem 3.2 above. Let \( \{M_i\}_{i=1}^{2^n} \)
be an effective enumeration of all the pairs \( (M', F') \), where \( M' \) is a DTM
and \( F' \) is a collection of designated state-sets of \( M' \) s.t. \( |F'| \leq n \).

Define an \( \omega \)-DTA \( M = (K, \Sigma, \Gamma, \delta, q_0, F_M') \), where \( H \) is an auxiliary set of
working states, \( H \cap K = \emptyset, q_0 \in H \), \( K = \{q^{(j)}_B, q^{(j)}_C | 1 \leq j \leq n\} \) and
P_M = 2^{KH} - \tilde{P}, \text{ where } \tilde{P} = \{D \cup D' \mid D' \subseteq H, D \subseteq K \} \text{ and there exists } j, 1 \leq j \leq n, \text{ s.t. } \phi^{(j)}_G \in D, \phi^{(j)}_B \in D'. \text{ For } \omega \text{-input } 0^1\omega, M \text{ generates an encoding of } M_i = (M', F') \text{ and simulates } M_i \text{ on } 0^1\omega. \text{ Suppose } P' = \{D_j\}_{j=1}^k \text{ where } k \leq n. \text{ The tape of } M \text{ is divided into } 2 \text{ tracks: } \\
\alpha \text{- the input track and } \beta \text{- the working track. The simulation of } M_i \text{ is carried out in the set } H \text{ of working states. Each simulation step is followed by a book-keeping phase, a single move to the right on the } \alpha \text{ track and then a corresponding folding process of the } \beta \text{ track w.r.t. the currently scanned square on the } \alpha \text{ track (Lemma 2.10). We now describe the book-keeping phase. Let } q \text{ be the state entered by } M_i \text{ in the most recent simulation step. Then during the book-keeping phase, } M \text{ will pass through all of the following states in an arbitrary sequence: }

(a) \phi^{(j)}_G \text{ for all } 1 \leq j \leq k \text{ s.t. } q \in D_j \text{ and s.t. in case state } \phi^{(j)}_G \text{ has been entered before, then since the last time } \phi^{(j)}_G \text{ has been entered, } M_i \text{ has accomplished another pass through all states of } D_j;

(b) \phi^{(j)}_B \text{ for all } 1 \leq j \leq k \text{ s.t. } q \notin D_j. \text{ In case } M_i \text{ halts on } 0^1\omega, M \text{ stops the simulation and continues moving right on the } \alpha \text{ track in a traverse state of } H. \text{ By the definition of } M, \text{ if the run of } M_i \text{ on } 0^1\omega \text{ is c.n.o., } 0^1\omega \in T(M_i) \text{ iff } 0^1\omega \in T(M). \text{ As in Theorem 3.2 above, } M \text{ has Property C and } T(M) \in DTMF(m) \omega - DTMF(n) \omega, \text{ where } m = |F_M|.

Corollary 3.7 \quad \text{For each integer } n \geq 1 \text{ there can be found an integer } m > n \text{ s.t. } DTMF(n) \omega \notin DTMF(m) \omega \notin DTM. \omega.

3.3 \quad \text{Concrete Examples Representing the Complexity Classes}

We shall now define two families of } \omega \text{-regular languages which cut across both the state-complexity and the designated set complexity hierarchies.
Definition 3.8  Let $K = \{q_i\}_{i=1}^n$ be a set of $n$ elements. Define a binary enumeration function $\text{BE}: 2^K \to \{1, \ldots, 2^n\}$ as follows: for $H \subseteq K$ let
$$\text{BE}(H) = 1 + \sum_{i=1}^n \delta_H(q_i)2^{i-1},$$
where $\delta_H(q_i) = 1$ if $q_i \in H$ and 0 otherwise.

BE enables us to effectively enumerate all subsets $H$ of $K$, according to increasing value of $\text{BE}(H)$; this enumeration will be called the binary enumeration of $2^K$.

The binary enumeration has the following property: if $K = \{q_i\}_{i=1}^n$ and $S = \{q_i\}_{i=1}^m$, where $m > n$, and $\{s_{j\cdot}^{i\cdot} \}_{j=1}^\ell(m) \subseteq 2^m$, is the binary enumeration of $2^S$, then $\{s_{j\cdot}^{i\cdot} \}_{j=1}^\ell(n) \subseteq 2^n$, is the binary enumeration of $2^K$.

Definition 3.9  Let $\Sigma = \{a, b\}$. For each $\sigma \in (a^+b)^\omega$ define the set of repetition numbers.

$$\text{RN}(\sigma) = \{i \mid \text{the subword } ba^i b \text{ appears infinitely many times in } \sigma\}.$$

Let $N = \{1, 2, \ldots, n\}$ and let $\{K_j\}_{j=1}^\ell(n) \subseteq 2^n$, be the enumeration of all subsets of $2^N$, obtained by using twice the binary enumeration. Define:

$$\text{LS}(n) = \{a^q b^\sigma \mid 1 \leq q \leq \ell(n), \sigma \in (a^+b)^\omega \text{ and } \text{RN}(\sigma) \subseteq K_q\}.$$  

LS($n$) is an $\omega$-regular language which can be recognized by an $\omega$-type deterministic finite state automaton with about $n \cdot 2^n$ states.

Theorem 3.10  For each integer $n > 0$ there can be found an integer $m_o > n$ such that for each $m \geq m_o$, $\text{LS}(m) \notin \text{DTMS}(n)$, i.e. every $\omega$-DTA recognizing $\text{LS}(m)$ must have more than $n$ states.
Proof. Let \( m_0 \) be the number for which, by Theorem 3.2, \( \text{DTMS}(n+2)_\omega \subseteq \text{DTMS}(m_0)_\omega \). Suppose there exists an \( n \)-state \( \omega \)-DTA \( M = (M',F') \) that accepts \( LS(m_0) \), where \( M' = (K',E,F',\delta,q_0) \). Let \( K = \{ s_i \}_{i=1}^{m_0} \) and let 
\[ \ell(m_0) = 2^K \]
\[ \{ S_j \}_{j=1}^{m_0} \], \( \ell(m_0) = 2^2 \), be the enumeration of \( 2^2 \), obtained by using twice the binary enumeration. With the aid of \( \omega \)-DTA \( M \), for each \( m_0 \)-state \( \omega \)-DTA \( M_1 = (M'_1,F'_1) \) we are now able to construct an \( (n+2) \)-state \( \omega \)-DTA \( M_2 \) that accepts \( T(M_1) \). \( M_2 \) has 2 tracks, \( a \), on which input \( \sigma \) is initially written and \( b \), initially blank. The states of \( M_2 \) are the states of \( M \) with the addition of two working states. Given an input \( \sigma \), \( M_2 \) writes \( a^q b \) on \( b \), where \( F'_1 = S_q \). Then \( M_2 \) simulates \( M_1 \) on the \( a \) track step by step with the working states. While simulating \( M_1 \) on \( a \), \( M_2 \) keeps writing on \( b \) (as described below), and in parallel \( M_2 \) simulates \( M \) on \( b \), with the contents written on \( b \) regarded as input to \( M \). Each simulation step of \( M_1 \) by \( M_2 \) is followed by the addition of string \( a^j b \) to the non-blank prefix of \( b \), where \( j \) is the index of state \( s_j \) entered by \( M_1 \) in that step; then follows the simulation of a single move of \( M \) on \( b \) which is carried out via the corresponding states of \( M_2 \). Adding to each designated set in \( F' \) the two working states of \( M_2 \) we derive a set of designated sets for \( M_2 \) s.t. \( \sigma \in T(M_2) \) iff 
\[ \text{INS}_{M_1}(c) \in S_q \] (Definition 1.5), therefore \( T(M_2) = T(M_1) \). Thus our assumption \( LS(m_0) \in \text{DTMS}(n)_\omega \) yields \( \text{DTMS}(m_0)_\omega \subseteq \text{DTMS}(n+2)_\omega \), which contradicts the choice of \( m_0 \). A similar argument works for all \( m \gg m_0 \). \( \square \)

Corollary 3.11 For each integer \( n \gg 2 \), \( \text{DTMS}(n)_\omega \) is incommensurate with the class of \( \omega \)-regular languages.

Definition 3.12 For each \( m = 1,2,\ldots \) define \( LF(m) = \{ \sigma = \prod_{j=1}^{\infty} x_j c_j \mid \forall j = 1,2,\ldots, x_j = a_j \text{ or } x_j = b_j \text{ for } 1 \leq i_j \leq m, \text{ and } \exists k, \} \)
i \leq k \leq m$, s.t. the subword $ca^kc$ appears infinitely many times in $\sigma$
and the subword $cb^kc$ appears only finitely many times in $\sigma$.

One can easily verify that for each $m$, $LF(m)$ is an $\omega$-regular
language. We now show that for any integer $n > 0$ there can be found an
integer $m > n$, s.t. the $\omega$-set $LF(m)$ cannot be recognized by an $\omega$-DTA
with $n$ or less designated state sets.

**Theorem 3.13** For each integer $n > 0$ there can be found an integer
$m_0 > n$, s.t. for each $m \geq m_0$, $LF(m) \notin DTMF(n)_\omega$.

**Proof.** The proof resembles that of Theorem 3.10 above.

Let $m_0$ be the number for which, by Theorem 3.6, $DTMF(n)_\omega \nsubseteq DTMF(m_0)_\omega$.

Suppose there exists an $\omega$-DTA $M = (M',H)$ accepting $LF(m_0)$, where $|H| = n$.

Given an $\omega$-DTA $M_1 = (M'_1,F')$, where $F' = \{F_{1,i}\}_{i=1}^{m_0}$, we are now able to
construct an $\omega$-DTA $M_2$ in $DTMF(n)_\omega$ accepting $T(M_1)$. Here too $M_2$ has
two tracks: $a$, containing the input word, on which the simulation of $M_1$
is carried out in two working states and $\beta$, initially filled with blanks.

Each time $M_1$ completes a cycle through all the states in $F_{1,i}$, $M_2$ adds
$a^i c$ on $\beta$ (on the right of the non-blank portion) and each time $M_1$
is in a state out of $F_{1,i}$, $M_2$ adds $b^i c$ on $\beta$. $M_2$ operates exactly as
the corresponding $\omega$-DTA in Theorem 3.10 but in case a new letter on $\beta$
is required, which is not yet available, $M_2$ delays the simulation of $M$
on $\beta$ until the next writing, of $a^i c$ or $b^i c$, on $\beta$ takes place.

Adding the two working states to all the designated sets of $M$ we obtain
an $\omega$-DTA $M_2$ with $n$ designated sets that accepts $T(M_1)$. Hence
$LF(m_0) \notin DTMF(n)_\omega$. Similarly $LF(m) \notin DTMF(n)_\omega$ for each $m \geq m_0$. □

**Corollary 3.14** For each integer $n \geq 1$, $DTMF(n)_\omega$ is incommensurate
with the class of $\omega$-regular languages.
4. OSCILLATING DETERMINISTIC $\omega$-TA's

In this section we consider deterministic $\omega$-TA's which may oscillate ($\omega$-os-DTA's) but which accept only via non-oscillating runs. As explained before, we can view these machines as equipped with an extra mechanism for recognizing oscillations in an infinite run. We show here that this extra mechanism does add power to the machines. That is, $\omega$-os-DTA's are indeed more powerful than the non-oscillating $\omega$-DTA's, and differ considerably in their properties.

In studying the variants of $\omega$-os-DTA's we reveal a definite resemblance of their properties with those of non-deterministic $\omega$-TA's (Theorems 2.2 and 2.4). However, the two families are not equivalent; in fact it is shown that non-deterministic $\omega$-TA's are strictly more powerful than oscillating deterministic $\omega$-TA's.

**Theorem 4.1** For every $m$-$\omega$-os-DTA and for each $i, i = 1, 1', 2, 2', 3$, there can be constructed an $i$-equivalent $\omega$-os-DTA.

**Proof.** Let $M$ be an $m$-$\omega$-os-DTA; if $m > 2$, all the working tapes of $M$ can be simulated on its second tape, yielding an $i$-equivalent $2$-$\omega$-os-DTA. Thus we may assume that $M = (M', F)$ is a $2$-$\omega$-os-DTA. Define $M_1'$ to be an $\omega$-os-DTA that simulates $M$ as follows. The single tape of $M_1'$ is divided into two tracks, $a$ and $b$, representing respectively the input tape and the working tape of $M$. For each $\omega$-input $s$, the simulation will be carried out by $M_1'$ while applying the relative folding process (Lemma 2.10) for $b$ w.r.t. $a$. This will guarantee that $M_1'$ oscillates only when $M$ does. For each $i = 1, 1', 2, 2', 3$, one can define in terms of $F$ a set of designated sets $H^{(i)}$ s.t. $T_1((M_1', H^{(i)})) = T_1(M)$. \[\Box\]
Lemma 4.2 For each $\omega$-os-DTA there can be constructed a 1'-equivalent $\omega$-os-DTA with a single designated set.

Proof: Let $M = (M', F)$ be an $\omega$-os-DTA. The $\omega$-os-DTA $M_1$ that 1'-accepts $T_1(M)$ has its whole set of states as its only designated set. For $\omega$-input $\sigma$, $M_1$ simulates $M$ as long as $M$ stays within at least one of the sets in $F$; otherwise $M_1$ is blocked.

Closure under union for $\omega$-os-DTA languages is not so straightforward. This is because when simulating two $\omega$-os-DTA's in parallel on some $\omega$-input $\sigma$, one of the machines may accept $\sigma$ while the other may have an oscillating run on $\sigma$. In such a case the simulation will result in an oscillating run on $\sigma$, and $\sigma$ will not be accepted. To overcome this difficulty a special construction will be used.

Theorem 4.3 The class of $\omega$-languages 1'-accepted by $\omega$-os-DTA's is closed under union.

Proof. Let $M', M''$ be two $\omega$-os-DTA's. By Lemma 4.2, there can be constructed two 1'-equivalent $\omega$-os-DTA's with single designated sets $M_1 = (M'_1, \{F_1\})$ and $M_2 = (M''_2, \{F_2\})$ respectively. Construct an $m$-$\omega$-os-DTA $M$ with 6 tapes that simulates both $M_1$ and $M_2$ in parallel, as follows:

Let $\alpha$ be the input tape on which input $\sigma$ is written; $\beta_1, \beta_2$ - two simulation tapes on which $M$ simulates $M_1$ and $M_2$ respectively. $\sigma$ is copied letter after letter onto both $\beta_1$ and $\beta_2$ during the simulation. Tape $\delta$ is the "step counter", i.e. a unary counter which keeps track of the number of steps executed by $M$ up to the current step, i.e. the contents of $\delta$ at step no.i of $M$ is the number i. On the remaining two tapes $\gamma_1, \gamma_2$, $M$ keeps record of its most recent visits to each of the
Consider the simulation of a single move of one of the machines, say $M_t$ ($t = 1$ or $2$), by $M$ during the simulation process. Suppose the head on tape $\beta_t$ in this move scans square no. $i$. Then $M$'s head on tape $\gamma_t$ will be placed on segment no. $i$, in which a number $n_t(i)$ representing the "time" (i.e. step number) when square no. $i$ on $\beta_t$ was last scanned by $M$, is stored. Then the head on the other tape $\gamma_s$ ($s = 1$ or $2$, $s \neq t$) will be placed on the first segment of $\gamma_s$ and will start moving right, scanning segment after segment on $\gamma_s$. For each segment scanned in turn, the number stored in it will be compared with $n_t(i)$, until for the first time a number greater than $n_t(i)$ is encountered. The segment holding this number corresponds to the leftmost square $j$ on $\beta_s$ which has been scanned by $M$ since the last time square $i$ on $\beta_t$ was scanned. Then the reading head on the input tape $\alpha$ is "regressed" to square no. $k = \max(i, j)$, after which it returns back where it was. A new simulation step is then started.

We claim that for each input $\sigma$, $M$ has an oscillating run on $\sigma$ if and only if both machines $M_1$ and $M_2$ have oscillating runs of $\sigma$. To see this, suppose both $M_1$ and $M_2$ have oscillating runs on $\sigma$. Then there exists numbers $m_1$ and $m_2$ s.t. machine $M_t$, $t = 1, 2$, returns infinitely many times to square no. $m_t$ on its tape. By the above construction, $M$'s reading head on the input tape $\alpha$ will be "regressed" to square no. $k = \max(m_1, m_2)$ infinitely many times during the computation; this is because infinitely many times, between two consecutive "visits" of $M$'s head on square $m_1$ on $\beta_1$, $M$ will have visited square
m_2 on \( \beta_2 \). Hence \( M \) will also have an oscillating run on \( \sigma \). On the other hand, if at least one of the machines \( M_1 \) or \( M_2 \) has a c.n.o. run on \( \sigma \), then by the construction above, the reading head on tape \( \alpha \) will not oscillate, hence \( M \) will also have a c.n.o. run.

Defining all the states of \( M \) as its designated state set we have 
\[ T_{1'}(M) = T_1'(M_1) \cup T_1'(M_2). \]

Using Theorem 4.3, we can now prove that, unlike the case of non-oscillating \( \omega \)-DTA's (Theorem 2.3), for \( \omega \)-os-DTA's all i-acceptance modes \( (i = 1, 1', 2, 2', 3) \) are equivalent. An analogous result was obtained for non-deterministic \( \omega \)-TA's (see Theorem 2.4 above).

**Theorem 4.4** For each \( i=1, 1', 2, 2', 3 \), the class \( L_i \) of \( \omega \)-languages i-accepted by \( \omega \)-os-DTA's equals \( \text{os-DTML}_\omega \).

**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 \( \text{os-DTML}_\omega \), it suffices to show that 
\[ \text{os-DTML}_\omega \subseteq L_{1'} \subseteq L_1. \]

Let \( M = (M', F) \) be an \( \omega \)-os-DTA. Since \( L_1 \) is closed under union and \( T(M) = \cup (T(M', \{H\})) \) we may assume w.l.o.g. that \( M = (M', \{F\}) \) is an \( \omega \)-os-DTA with a single designated set. We now construct an \( \omega \)-os-DTA \( M_1 \) which \( 1' \)-accepts \( L = T(M) \). The construction will use the "regression point" technique developed in [CoGo 4] for \( \omega \)-TA's. \( M_1 \) will have a c.n.o. run only on tapes which belong to \( L \). Given input \( \sigma \in \omega \), \( M_1 \) 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 shift the regression point back to the beginning of $\sigma$. If, however, since the last time the regression point was shifted, $M$ has passed through all the states of $F$, $M_1$ will shift 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 $L_1 \subseteq L_{11}$, let $M = (M', F)$ be an $\omega$-os-DTA. The $\omega$-os-DTA $M_1$ that $1$-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.

**Proof.**

In Section 3 we saw that w.r.t. 3-acceptance by non-oscillating deterministic $\omega$-TA's there is an infinite "state complexity" hierarchy corresponding to the number of states in the machines. As for oscillating deterministic $\omega$-TA's, no such hierarchy exists; in fact, two states suffice for recognizing all $\omega$-languages in os-DTML$^\omega$.

**Theorem 4.5** Every $L$ in os-DTML$^\omega$ can be accepted (i-accepted for $d = 1, 1', 2, 2'$) by a two-state $\omega$-os-DTA with a single designated set.
Proof. Let \( L \in \text{os-DTML}_\omega \); then by Theorem 4.4 and Lemma 4.2 \( L \) can be \( 1 \)-accepted by some \( \omega \)-os-DTA \( M \) with a single designated set \( F \).

We can simulate the operation of \( M \), while in \( F \), using only two states \( \langle \text{Sha} \rangle \) and block \( M \) if it moves out of \( F \). Thus a two state \( \omega \)-os-DTA \( M_1 \) with a single designated set, can be constructed s.t. \( L = T_1(M_1) \) for each \( i = 1', 2', 3 \). A similar construction will yield a two state \( \omega \)-os-DTA with a single designated set that \( 1 \)-accepts \( L \). \( \square \)

Theorem 4.6 \( \text{os-DTML}_\omega \) is closed under union and intersection but not under complementation.

Proof. Closure under union follows from Theorems 4.3, 4.4. Closure under intersection is straightforward and is done by a parallel simulation of the two given \( \omega \)-os-DTA's.

As for complementation, let \( \{M_i\}_{i=1}^{\infty} \) be an effective enumeration of all two state \( \omega \)-os-DTA's and let \( L = \{0^i 1^i | 0^i 1^i \in T(M_i) \}; \) then \( L \in \text{os-DTML}_\omega \). By Theorem 4.5 and diagonalization \( \omega^\omega - L \notin \text{os-DTML}_\omega \). \( \square \)

Theorem 4.7 \( \text{os-DTML}_\omega \notin \text{TYPE}_\omega \).

Proof. Let \( \{M_i\}_{i=1}^{\infty} \) be an effective enumeration of all two-state \( \omega \)-os-DTA's with a single designated set consisting of the two states.

By diagonalization \( L = \{0^i 1^i | 0^i 1^i \notin T_1(M_i) \} \) is not in \( \text{os-DTML}_\omega \) (Theorem 4.5). We now describe a \( 2 \)-\( \omega \)-TA \( M \) that \( 2 \)-accepts \( L \). For given input \( \sigma = 0^i 1^i \), \( M \) generates an encoding of machine \( M_i \). Then, while reading the input \( \sigma \) on-line fashion, \( M \) copies \( \sigma \) onto the working tape and simulates \( M_i \) on \( 0^i 1^i \). After every single step of the simulation, \( M \) moves to the current position on the input tape and copies the next letter of \( \sigma \) onto the working tape. Now, \( M_i \) can reject \( \sigma \) in two ways: First, by not scanning the whole tape \( \sigma \) and second, by returning infinitely many
times to a certain square on \( \sigma \). While simulating \( M_1 \), \( M \) non-deterministically guesses why \( 0^1 \omega \) is rejected by \( M_1 \) and operates accordingly. \( M \) non-deterministically chooses a square on its working tape and marks it, guessing that \( M_1 \) will return to it infinitely many times. \( M \) simulates \( M_1 \) as described above but passes through a special state \( q_B \) each time \( M_1 \) scans the marked square. \( M \) accepts \( 0^1 \omega \) iff either \( M_1 \) gets blocked during its computation on \( \sigma \) (in which case \( M \) enters a final traverse state), or if \( M \) passes infinitely many times through state \( q_B \).

Clearly, \( L \) is accepted by \( M \).

**Corollary 4.8** Non-deterministic \( \omega \)-TA's are strictly more powerful than oscillating deterministic \( \omega \)-TA's, which in turn are strictly more powerful than non-oscillating deterministic \( \omega \)-TA's, i.e.

\[
\text{DTML}_\omega \not\subseteq \text{os-DTML}_\omega \not\subseteq \text{TYPE}_\omega .
\]

**Proof.** By definition \( \text{DTML}_\omega \) is closed under complementation. The result then follows from Theorems 4.6 and 4.7.

**5. TWO "INHERENTLY OSCILLATING" DETERMINISTIC \( \omega \)-TA LANGUAGES**

Without using the diagonalization technique we are now able to exhibit two \( \omega \)-languages, each of which can be recognized by an oscillating deterministic \( \omega \)-TA but cannot be recognized by any non-oscillating deterministic \( \omega \)-TA(\( \omega \)-DTA).

Let \( LS(n) \) and \( LF(n) \) be as in Definitions 3.9 and 3.12.
Theorem 5.1: Each of the \( \omega \)-languages

\[
LS = \bigcup_{n=1}^{\infty} a^n b LS(n)
\]

and

\[
LF = \bigcup_{n=1}^{\infty} a^n b LF(n)
\]

is in \( os-DTML_\omega \) but not in \( DTML_\omega \).

Proof. Suppose \( LS \) is in \( DTML_\omega \); then for some \( k > 0 \) there exists a \( k \)-state \( \omega \)-DTA accepting \( L \). For every \( t = 1,2, \ldots \) \( a^t b LS(t) = LS \cap a^t b (a+b)^{\omega} \), hence it is possible to construct for each \( t \) a \( k \)-state \( \omega \)-DTA accepting \( LS(t) \) (by extending the tape alphabet \( \Gamma \)), which contradicts Theorem 3.10. It follows that \( LS \notin DTML_\omega \). Similarly by Theorem 3.13 \( LF \notin DTML_\omega \).

We will now describe the \( \omega \)-os-DTA that \( l^i \)-accepts \( LF \). Using the "regression point" technique of Theorem 4.4, one can effectively construct for each \( m \geq 1 \) and \( \ell, 1 \leq \ell \leq m \), an \( \omega \)-os-DTA that \( l^i \)-accepts \( LF(m, \ell) = \{ \sigma \in LF(m) \mid \text{the subword } ca^\ell c \text{ appears infinitely many times in } \sigma \text{ and the subword } cb^\ell c \text{ appears only finitely many times} \} \). Following the method in the proof of Theorem 4.3, one can construct an \( \omega \)-os-DTA \( M_\sigma \) which, given an \( \omega \)-input of the form \( \sigma = a^m b \sigma_1 \), simulates on \( \sigma_1 \) the \( \omega \)-os-DTA \( M_\sigma \) that \( l^i \)-accepts \( \bigcup_{i=1}^{\infty} LF(m, i) \) and thus \( l^i \)-accepts \( LF \).

Similarly, one can construct an \( \omega \)-os-DTA that \( l^i \)-accepts \( LS \). \( \Box \)
6. AN "INHERENTLY NON-DETERMINISTIC" $\omega$-TA LANGUAGE

To illustrate the limitations of deterministic $\omega$-TA's we now exhibit a simple $\omega$-language, involving no diagonalization, which is "inherently non-deterministic" i.e. is recognized by a non-deterministic $\omega$-TA but not recognizable by any type of deterministic $\omega$-TA.

Theorem 6.1 The $\omega$-language

$$LC = \{ \sigma \in (a^+b)^\omega \mid \exists j_0 \text{ s.t. the subword } b^{j_0}a \text{ appears infinitely many times in } \sigma \}$$

is an $\omega$-TA language which cannot be recognized by any deterministic $\omega$-TA (with or without oscillations); i.e.

$$L \notin \text{TYPE}_\omega \ominus \text{os-DTML}_\omega.$$

Proof. The $\omega$-TA which accepts $LC$ guesses $j_0$ at the beginning of its computation and then scans the input from left to right, entering a designated state $\bar{q}$ each time the subword $b^{j_0}a$ is encountered.

If $LC$ belonged to $\text{os-DTML}_\omega$ then, as in Theorems 3.10 and 3.13 we could follow the argument in the proof of $\text{os-DTML}_\omega \subseteq \text{TYPE}_\omega$ (Theorem 4.7) and build an $\omega$-os-DTA $\tilde{M}$ that accepts $L = \{ 0^i1^\omega \mid 0^i1^\omega \in T_1, (M_i) \}$, where $(M_i)_{i=1}^\omega$ is an effective enumeration of all two state $\omega$-os-DTA's with single designated set. $\tilde{M}$ would write $a^j b$ on the second track $\beta$ whenever the simulated $M_i$ entered square $j$ on its tape, and in parallel $\tilde{M}$ would simulate the $\omega$-os-DTA for $LC$ on $\beta$. Thus, via $LC$ $\tilde{M}$ would check whether there is a square to which $M_i$ returned infinitely many times. Since $L \notin \text{os-DTML}_\omega$ (Theorem 4.5) we have a contradiction.

Remark 6.2 The $\omega$-languages $LC$ and $LF$ above can each be recognized by an $\omega$-type non-erasing non-deterministic stack automaton ([H&U]).
7. WHAT NON-DETERMINISTIC ω-TA's CANNOT RECOGNIZE

Lastly, we define, again without diagonalization, an ω-language outside TYPEOω, i.e. not recognizable by any ω-TA.

Definition 7.1 Let \( M = (K, Σ, Γ, δ, q_0, F) \) be an ω-TA. Define the degree of non-determinism of \( M \) to be \( \max\{|δ(q,a,A)| \mid (q,a,A) ∈ K × (Σ ∪ \{ε\}) × Γ\} \).

The next lemma states that the degree of non-deterministic in ω-TA's can be bounded by 2.

Lemma 7.2 Every ω-language \( L \) in TYPEOω can be 2-accepted by an ω-TA with degree of non-determinism at most 2.

Proof. By Theorem 2.4 every \( L \) in TYPEOω can be 2-accepted by an ω-TA. Using the standard method, one can reduce the degree of non-determinism by successively splitting the sets \( δ(q,a,A) \) into two halves and adding transient states. By definition of 2-acceptance there is no need to change the designated state sets.

Definition 7.3 Define the ω-language \( LN \) over alphabet \( Σ = \{a,b\} \) as follows:

\[
LN = \{σ = \prod_{k=1}^{∞} c_k \mid ∀ k, c_k ∈ Σ, \text{ and there exist no two sequences of increasing integers } \{r(j)\}_{j=1}^{∞} \text{ and } \{ℓ(j)\}_{j=1}^{∞} \text{ satisfying for each } j > 1: \ (ℓ(j) + 1)2^{r(j+1)} - r(j) - 1 ≤ ℓ(j+1) ≤ (ℓ(j) + 2)2^{r(j+1)} - r(j) - 2 \text{ and such that for each } j > 1, c_k(j) = b\}.
\]

Theorem 7.4 \( LN ∈ TYPEOω \), i.e. \( LN \) cannot be recognized by a non-deterministic ω-TA.

Proof. By Lemma 7.2 and Theorem 2.5 every ω-language in TYPEOω can be 2-accepted by an ω-TA with degree of non-determinism no greater than 2,
in which each run on every input is c.n.o. Let \( L = \{0^i 1^j | 0^i 1^j \in T_2(M_1) \} \),
where \( \{M_1\}_{i=1}^{\infty} \) is an effective enumeration of all \( \omega \)-TA's with degree
of non-determinism 2 at most. By diagonalization \( L \notin \text{TYPE}_\omega \). We shall
now show how, by assuming \( L \) to be in \( \text{TYPE}_\omega \), one can construct an
\( \omega \)-TA accepting \( L \), leading to a contradiction.

Thus, suppose \( L \in \text{TYPE}_\omega \), and let \( M_{LN} \) be an \( \omega \)-TA that 2-accepts \( L \).

First, let us construct a 2-\( \omega \)-TA \( M \) that for any given input \( \sigma \) and
\( \omega \)-TA \( M_1 = (K_1, L_1, \Gamma_1, \delta_1, q_0') \), simulates simultaneously all possible runs of
\( M_1 \) on \( \sigma \), assuming the degree of non-determinism of \( M_1 \) is at most 2.
\( M \) leaves the input unchanged. The working tape \( \alpha \) will contain a description
of a breadth-first spanning of the infinite binary tree \( T \), representing
all possible runs of \( M_1 \) on \( \sigma \). Tape \( \alpha \) is divided into "blocks"; each
node in the tree \( T \) will be represented by a block on tape \( \alpha \). Each node
at level \( n+1 \) of \( T \) corresponds to a particular sequence of \( n \) moves of
\( M_1 \) (the beginning of a run on \( \sigma \)), represented by the path from the root
leading to that node. The configuration of \( M_1 \) after this particular sequence
has been completed is described in the corresponding block on tape \( \alpha \). The
block contains: (a) the state \( q \) of \( M_1 \); (b) the position \( j \) of \( M_1 \) on
the tape; (c) the first \( n \) squares on \( M_1 \)'s tape. Each node in the tree
has precisely two sons, representing the two possible next moves. In case
\( |\delta_1(q, a, A)| = 1 \), where \( A \) is the symbol currently scanned by \( M_1 \), then the
first son will represent the single next move while the second son, and
also all of its descendants, will be marked by a special symbol \( X \). If
\( |\delta_1(q, a, A)| = 0 \), then both sons as well as their descendants will be marked
by \( X \). A node marked by \( X \) will be represented on \( \alpha \) by a block containing \( X \) alone.

In phase \( n \) of the simulation \( M \) has to deal with up to \( 2^n \) possible
next moves corresponding to level \( n+1 \) in the tree; thus \( M \) adds \( 2^n \)
new blocks on the right of the non-written portion of $a$.

Let us enumerate the blocks on tape $a$ in order of their appearance; we have:

1. For $n = 1, 2, \ldots$, the $2^n$ possible configurations of $M_1$ after $n$ moves (level $n+1$ in the tree) are represented by blocks numbered

$$\sum_{t=1}^{n-1} 2^t + 1 \text{ to } \sum_{t=1}^{n} 2^t,$$

i.e. from $2^{n-1}$ to $2^{n+1}-2$.

2. For $1 \leq k \leq 2^n$ and $m > n$, the $k$-th block at level $n+1$ (numbered $\sum_{t=1}^{n-1} 2^t + k$) has $2^{m-n}$ "descendant" blocks after $m-n$ further moves of $M_1$. The numbers of these blocks are in the range from

$$2^m + (k-1)2^{m-n} \text{ to } 2^m + k \cdot 2^{m-n}-2.$$

Now, every 2-accepting run in $M_1$ defines two sequences: First, $A = \{r(j)\}_{j=1}^{\infty}$ - the set of indices of all steps in the run in which $M_1$ was found in its designated state set. This sequence in turn defines another sequence $B = \{k(j)\}_{j=1}^{\infty}$ - the set of block numbers in $M$ which correspond to sequence $A$, i.e. $k(j)$ is the number of the block that represents $M_1$'s configuration at step $r(j)$ of the run. This block corresponds to node number $k(j) - (2^r(j)-2)$ at level $r(j)+1$ of the tree $T$.

By the above, the following inequality ties the $B$-sequence elements:

for $j \geq 1$,

$$2^{r(j+1) + (k(j)-2^r(j)+1)} 2^{r(j+1)-r(j)-1} \leq k(j+1) \leq 2^{r(j+1) + (k(j)-2^r(j)+2)} 2^{r(j+1)-r(j)-2},$$

i.e.

$$k(j+1) - 2^{r(j+1) - r(j) - 1} \leq (k(j)+2) 2^{r(j+1) - r(j) - 2}.$$
It can be easily seen that \( M \) can organize the information on \( \alpha \), so that it will be able to move constantly to the right on the input \( \sigma \), while the simulation is carried out on the working tape \( \alpha \) in a deterministic fashion.

With the aid of \( M \) we are now able to construct a 3-\( \omega \)-TA \( \tilde{M} \) that accepts \( L \). For any \( \sigma = 0^i \omega \), \( \tilde{M} \) mimics the operation of \( M \) on \( M_1 \) and \( \sigma \) in its first working tape \( \alpha \) (i.e. simulates on \( \alpha \) all possible runs of \( M_1 \) on \( \sigma \)). For each \( n \geq 1 \), after phase \( n \) of the simulation (corresponding to level \( n+1 \) of the tree), \( \tilde{M} \) adds to the right of the non-blank prefix of its second working tape \( \beta \), a word \( y_n \) of length \( 2^n \), which is derived from the new \( 2^n \) blocks which were written by \( \tilde{M} \), on tape \( \omega \) in phase \( n \). Specifically, \( y_n = \tilde{M} c_j \), where for each \( j \), \( c_j = 1 \) in case the \( j \)-th block written at phase \( n \) contains a state from the designated state set of \( M_1 \); otherwise, if the state is outside the designated set, or if the block contains \( X \), \( c_j \) equals 'a'. Each phase \( n \) in the simulation of \( M_1 \) on \( \sigma \) and the corresponding addition of the word \( y_n \) on \( \beta \) is followed by a step of simulation of the \( \omega \)-TA \( M_{LN} \) that 2-accepts \( LN \), with the word written on \( \beta \) regarded as input.

Define the designated state set of \( \tilde{M} \) to be that of \( M_{LN} \). It follows that \( 0^i \omega \) is not in \( T_2(M_1) \) iff in \( \tilde{M} \), during the simulation in parallel of all possible runs of \( M_1 \) on \( 0^i \omega \), as described above, the \( \omega \)-word generated on \( \beta \) belongs to \( LN \), i.e. if and only if \( 0^i \omega \in T_2(\tilde{M}) \).

Hence, \( T_2(\tilde{M}) = L \). \qed
REFERENCES


[Co&Go3] Cohen, R.S. and A.Y. Gold: "w-Computations on Deterministic Pushdown Machines". To appear in JCSS.


REFERENCES (cont'd)


