COMPLEXITY OF $\omega$-COMPUTATIONS ON DETERMINISTIC PUSHDOWN MACHINES

by

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

Technical Report #56

November 1975

* Department of Computer Science, Technion, Haifa, Israel.

** Faculty of Mathematics, Technion, Haifa, Israel.
Abstract

Deterministic pushdown machines working on \(\omega\)-tapes are considered. The \(\omega\)-languages recognized by such machines are called \(\omega\)-DCFL's. Various recognition mechanisms for \(\omega\)-sequences are considered, yielding a hierarchy of "\(i\)-recognizable" classes of \(\omega\)-DCFL's. \(i\)-recognizability of an \(\omega\)-language can be considered a measure of the complexity of the \(\omega\)-language. These "\(i\)-recognizable" classes are extensively studied and related to other classes of \(\omega\)-CFL's. Decision problems concerning these classes are investigated; in particular, for some of the classes the membership problem is shown to be decidable.
ACKNOWLEDGEMENT

The authors wish to thank Mrs. Raya Anavi for her careful typing of the manuscript.
0. INTRODUCTION

An $\omega$-language is a collection of $\omega$-length strings over some alphabet; an $\omega$-machine is any device capable of processing (or recognizing) $\omega$-length inputs.

One may consider each $\omega$-string as an object parametrized by time, and a set of $\omega$-strings can be viewed as describing a dynamical discrete-time process. In fact, such an approach has been frequently taken in general system theory, where a general system (or process) is formalized as an abstract relation parametrized by time. In a recent book by Windeknecht [Wi] set theoretical (and some algebraic) notions are employed to develop an axiomatic mathematical approach to general system theory. The central notion there is that of a "T-process", where $T$ stands for a linearly ordered "time set". A T-process is defined as a set of functions from $T$ into some arbitrary set $A$. Particular attention is given to the discrete time set $T = \omega$, in which case the notion of an $\omega$-process is obtained, meaning a collection of $\omega$-sequences, i.e. an $\omega$-language, over alphabet $A$. An $\omega$-processor is a special type of $\omega$-process, in which the set $A$ is a set of ordered pairs. Thus what we consider an $\omega$-machine is a particular type of $\omega$-processor, usually with some additional underlying structure.

Many of the concepts discussed in the book can be re-formulated in terms of language theory; particularly the notions concerning time evolution and strong types of causality in processes and processors are closely related to the infinite behaviour of sequences and machines.

We, therefore, believe that the development of a general theory of
\(\omega\)-languages and \(\omega\)-machines may provide new tools for the constructive
specification of \(\omega\)-processes and contribute to the understanding of \(\omega\)-systems
in general. In the development of such a theory, topics which might shed new
light on the nature of \(\omega\)-processes should be emphasized. It seems that the
notions of "type \(i\) recognizability" of \(\omega\)-languages (\(i=1,1',2,2',3\)), as
well as the notion of "\(\omega\)-Kleene Closure" of a family of languages (defined
in Section 1 below) become intuitively meaningful when considered in terms
of \(\omega\)-processes. Before proceeding to discuss our work, we shall attempt to
give an intuitive explanation of these notions in terms of \(\omega\)-processes.

Informally, an \(\omega\)-process is "1-recognizable" if it is recognizable
by the occurrence of some "event", one out of a finite set of "designated
events", at least once during the process, i.e. at least once in each of its
\(\omega\)-sequences (assuming that the occurrence of each such "event" can be recog­
nized by the appropriate type of machine). The \(\omega\)-process in "2-recognizable"
if at least one event out of a finite set of "designated events" occurs
"every once in while" (i.e. infinitely many times) in each of its \(\omega\)-sequences.
One can similarly attach meaning to the other types of "\(i\)-recognizability".

An \(\omega\)-language which belongs to the "\(\omega\)-Kleene Closure" of some family
of (finite string) languages, can be viewed as modelling an infinite computa­
tion in which "every once in a while" all previous information is "forgotten"
e.g. a "rewind" instruction is applied), and the computation re-starts "from
scratch". The above ideas will become clearer after reading Section 1.

Previous work on \(\omega\)-languages and \(\omega\)-automata was concerned mainly with
the infinite behaviour of finite state automata ([Mc], [Lan], [Cho], [Hos] and
A detailed summary of previous work in this direction is found in [Eil], [Tr&Ba] and [Cho]. The main results concerning finite state \(\omega\)-automata and the corresponding family of \(\omega\)-regular languages will be stated in Section 1.

A basis for a general theory of \(\omega\)-languages was laid in the series of two papers [Co&Go] and [Co&Gol]. In the first paper the fundamental notions concerning \(\omega\)-grammars and \(\omega\)-machines were introduced. The paper is devoted mainly to the study of \(\omega\)-context free languages (\(\omega\)-CFL's), their properties and their characterization both by means of \(\omega\)-pushdown automata (\(\omega\)-PDA's) and, more interestingly, by applying the "\(\omega\)-Kleene Closure" operator to the (classical) family of context free languages. The latter characterization generalizes an analogous result by McNaughton concerning the family of \(\omega\)-regular languages ([Mc]; see also Theorem 1.1).

The second paper deals with various models of \(\omega\)-generation in grammars and \(\omega\)-recognition by machines. The effect of certain restrictions on \(\omega\)-derivations (e.g. control sets) is studied. Non-leftmost generation in \(\omega\)-context free grammars is shown to be strictly less powerful than leftmost generation. The notion of "type i recognition" by \(\omega\)-machines is introduced, giving rise to a hierarchy of "i-recognizable" classes of \(\omega\)-CFL's. The classes are studied and characterized. A detailed summary of the results in the above papers, particularly the results relevant to the current paper, is included in Section 1.

This paper is devoted mainly to the study of deterministic \(\omega\)-PDA's and the corresponding family of deterministic \(\omega\)-CFL's (\(\omega\)-DCFL's). Following the preliminaries in Section 1, closure properties of the \(\omega\)-DCFL's are studied in
Section 2. In Section 3 the $\omega$-Kleene Closure operator is applied to the class of (finite string) deterministic CFL's, yielding a subfamily of the $\omega$-CFL's which properly contains the $\omega$-DCFL's; thus the above "$\omega$-Kleene Closure" characterization of the $\omega$-CFL's fails to hold for the $\omega$-DCFL's. However, the $\omega$-Kleene Closure of the family of strict deterministic languages [Ha&Ha] is characterized by means of "$\omega$-empty quasi-deterministic" PDA's.

Section 4 deals with the various modes of i-recognition in $\omega$-DPDA's. The families of $\omega$-CFL's i-recognizable by $\omega$-DPDA's are shown to constitute a hierarchy within the $\omega$-DCFL's. This hierarchy differs in structure from the analogous hierarchy obtained for the non-deterministic $\omega$-PDA's. Particular attention is paid to the classes of $l'$-recognizable and 1-recognizable $\omega$-DCFL's, which are characterized by means of two new operations, "extrapolation" and "non-init", operations which transform finite-string languages into $\omega$-languages. These operations, introduced in Subsection 4.2, shed light on the nature of the $l'$-recognition mode in deterministic $\omega$-automata of any kind.

In Section 5 the relations among the classes of i-recognizable $\omega$-DCFL's and the other classes of non-deterministic $\omega$-PDA languages are established, and the whole rich hierarchy of $\omega$-CFL families is presented. Section 6 deals with various decidability questions concerning both deterministic and non-deterministic $\omega$-CFL's, as well as questions concerning membership in any of the "i-recognizable" classes. In particular, it is shown that, for an arbitrary $\omega$-DCFL $L$, it is decidable whether $L$ can be 1'-recognized (1-recognized) by a deterministic $\omega$-PDA. The analogous questions for non-deterministic $\omega$-PDA's are shown to be unsolvable. The paper terminates with a discussion on
the $\omega$-regularity problem for $\omega$-DCFL's (which, in general, remains open) and its solution within the subfamilies of $1'$-recognizable and $1$-recognizable $\omega$-DCFL's.

A basic knowledge of Formal Language Theory [H&U] is sufficient for the understanding of this paper. Section 1 below contains all the necessary background concerning $\omega$-languages.
1. \(\omega\)-CONTEXT FREE LANGUAGES - PRELIMINARIES AND A SURVEY OF PREVIOUS WORK

We now present a detailed summary of the basic notions and main results concerning \(\omega\)-context free languages and \(\omega\)-pushdown automata, as developed in [Co&Go] and [Co&Gol].

The reader is assumed to be familiar with formal language theory; the terminology and notation used here will be mostly taken from [H&U].

A finite string (word) over \(\Sigma\) is any sequence \(x = \prod_{i=1}^{k} a_i\), where \(a_i \in \Sigma\) for \(i = 1, \ldots, k\), \(k = 0, 1, \ldots\); \(x^R = \prod_{i=1}^{k} a_{k+1-i}\) is the reversal of \(x\). \(k\) is the length of \(x\) and is denoted by \(|x|\). If \(|x| = 0\), \(x\) is the null (empty) word and is denoted by \(\epsilon\). Let \(N\) 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. For any language \(L \subseteq \Sigma^*\), let

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

Thus \(L^\omega\) consists of all \(\omega\)-strings obtained by concatenating words from \(L\) in an infinite sequence. An \(\omega\)-language is any subset of \(\Sigma^\omega\).

For any \(\sigma \in \Sigma^\omega\), if \(\sigma = \prod_{i=1}^{\infty} a_i\), \(a_i \in \Sigma\), define for each \(j \geq 1\), \(\sigma/j = \prod_{i=1}^{j} a_i\), \(\sigma(j) = a_j\) and \(\sigma/0 = \epsilon\).

**Definition 1.2** For any mapping \(\psi : A \rightarrow B\), define \(\psi^{-1}(b) = \{a \mid a \in A, \text{ card}(\psi^{-1}(b)) = \omega\}\), where \(\text{card}(S)\) denotes the cardinality of set \(S\).
Finite state machines working on \(\omega\)-inputs have been considered by several authors; among them [Mc], [Lan], [Cho], [Hos], are concerned with these machines as recognizers of \(\omega\)-languages. A detailed summary of previous work in this direction can be found in [Eil], [Tr&Ba], [Cho].

We now define the notion of \(\omega\)-type finite-state automaton.

**Definition 1.3** A (non-deterministic) finite-state machine (FSM) is the system \(M = (K, \Sigma, \delta, q_0)\), where \(K\) is a finite set of states; \(\Sigma\) is a finite input alphabet; \(q_0\) in \(K\) is the initial state, and \(\delta\) is a mapping from \(K \times \Sigma\) into \(2^K\).

An FSM is called deterministic (DFSM) iff \(\delta : K \times \Sigma \rightarrow K\).

An \(\omega\)-type finite state automaton (\(\omega\)-FSA) is the system \(M_1 = (K, \Sigma, \delta, q_0, \mathcal{F})\), where \(M = (K, \Sigma, \delta, q_0)\) is an FSM and \(\mathcal{F} \subseteq 2^K\) is the collection of designated sets. \(M\) will sometimes be written as \((M, \mathcal{F})\). An \(\omega\)-FSA \((M, \mathcal{F})\) is deterministic (\(\omega\)-DFSA) if \(M\) is deterministic.

We now introduce the notion of "infinite run" of an FSM, representing the computation of the machine on an infinite input.

Let \(\sigma = \prod_{i=1}^{\infty} a_i \in \Sigma^\omega\), where \(a_i \in \Sigma\), \(i \geq 1\). A sequence of states \(r = \{q_i\}_{i \geq 1}\) is called an (infinite) run of \(M\) on \(\sigma\), starting in state \(p\), iff:

1. \(q_1 = p\); 2. for each \(i \geq 1\), \(q_{i+1} \in \delta(q_i, a_i)\). In case a run \(r\) of \(M\) on \(\sigma\) starts in state \(q_0\), we simply say "a run of \(M\) on \(\sigma\)."

Every (infinite) run \(r\) induces a mapping \(f_r\) from \(\mathbb{N}\) into \(K\), where \(\forall i \geq 1, f_r(i) = q_i\). Referring to Definition 1.2, we define \(\text{INS}(r) = \bigcap_{n} (f_r)\). \(\text{INS}(r)\) is the set of all states in \(K\) entered by \(M\) infinitely many times during run \(r\). Define:

\[ T(M_1) = \{\sigma \in \Sigma^\omega | \text{there exists a run of } M \text{ on } \sigma \text{ s.t. } \text{INS}(r) \in \mathcal{F}\} \].
T(M) is the \( \omega \)-language accepted by \( M \).

In [Mc] the equivalence of the models of deterministic and non-deterministic \( \omega \)-FSA was established. This lead to a characterization of the \( \omega \)-FSA languages by means of a new operator, the "\( \omega \)-Kleene closure", applied to the regular sets.

**Definition 1.4** For any family of sets \( \mathcal{L} \) over alphabet \( \Sigma \), the \( \omega \)-Kleene closure of \( \mathcal{L} \), denoted \( \omegaKC(\mathcal{L}) \), is

\[
\omegaKC(\mathcal{L}) = \{ L \subseteq \Sigma^\omega \mid L = \bigcup_{i=1}^{k} U_i V_i^\omega \text{ for some } U_i, V_i \in \mathcal{L}, \ i=1,2,\ldots,k, \ k \geq 1 \}
\]

Let \( \omegaKC(CF) \) (\( \omegaKC(\text{Reg}) \)) denote the \( \omega \)-Kleene closure of the context free (regular) languages.

The following is McNaughton's characterization theorem for \( \omega \)-FSA languages.

**Theorem 1.1** [Mc]. For any \( \omega \)-language \( L \), the following conditions are equivalent:

(a) \( L \) belongs to \( \omegaKC(\text{Reg}) \).

(b) There exists an \( \omega \)-FSA that accepts \( L \).

(c) There exists an \( \omega \)-DFSA that accepts \( L \).

An \( \omega \)-language \( L \) satisfying one of the conditions in Theorem 1.1 above will be called an \( \omega \)-regular language. The name \( \omega \)-regular is justified by the observation that any \( \omega \)-language given as a member of \( \omegaKC(\text{Reg}) \) can be described by what might be called an "\( \omega \)-regular expression" of the form

\[
E = \bigcup_{i=1}^{k} E_i F_i^\omega
\]

where \( E_i \) and \( F_i \) are (ordinary) regular expressions and \( k \geq 1 \). In the sequel such notation will prove very useful for the specification of \( \omega \)-languages.

In [Co&Go] a theory of \( \omega \)-languages of higher complexity than the \( \omega \)-regular languages has been initialized. The basic notions of \( \omega \)-grammars, \( \omega \)-context free
languages and $\omega$-PDA's were first introduced. Various representations of the $\omega$-context free languages were derived; particularly, the $\omega$-Kleene closure characterization of the $\omega$-FSA languages (Theorem 1.1(a)&(b)) has been generalized to $\omega$-PDA languages.

Before proceeding to state the main results, we review the definitions of $\omega$-PDA's and of $\omega$-grammars.

**Definition 1.5** A pushdown machine (PDM) is the system $M = (K, \Sigma, \Gamma, \delta, q_0, Z_0)$, where $K$ is a finite set of states, $\Sigma$ is a finite alphabet called the input alphabet, $\Gamma$ is a finite alphabet called the pushdown alphabet, $q_0$ in $K$ is the initial state, $Z_0$ in $\Gamma$ is the start symbol, initially appearing on the pushdown store, and $\delta$ is a mapping from $K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to finite subsets of $K \times \Gamma^*$. 

Following the notation in [H&U], if $\gamma \in \Gamma^*$ describes the pushdown store contents, the leftmost symbol will be assumed to be on "top" of the store. A configuration of a PDM is a pair $(q, \gamma)$, where $q \in K$ and $\gamma \in \Gamma^*$. If $a \in \Sigma \cup \{\varepsilon\}$, $\gamma, \beta \in \Gamma^*$, $Z \in \Gamma$, and further, if $(p, \beta)$ is in $\delta(q, a, Z)$, then we write $a : (q, Z \gamma) \mid M^\ast (p, \beta \gamma)$, and refer to it as a step of the computation of $M$. $\mid M^\ast$ is the transitive reflexive closure of $\mid M$. The subscript $M$ will be omitted from $\mid M$ and $\mid M^\ast$ whenever understood.

Let $\sigma = \prod_{i=1}^{\infty} a_i \in \Sigma^\omega$, where $a_i \in \Sigma$, $i \geq 1$. An infinite sequence of configurations $r = \{(q_i, \gamma_i)\}_{i \geq 1}$ is called a run of $M$ on $\sigma$, starting in configuration $(p, \gamma)$, iff:

(a) $(q_1, \gamma_1) = (p, \gamma)$;
(b) for each $i \geq 1$ there exists $b_i \in \Sigma \cup \{\varepsilon\}$ s.t. $b_i : (q_i, \gamma_i) \mid M (q_{i+1}, \gamma_{i+1})$.

Technion - Computer Science Department - Technical Report CS0056 - 1975
(c) $\prod_{i=1}^{\infty} b_i$ is identical with $\sigma$, in which case the run is complete, or $\prod_{i=1}^{\infty} b_i$ is a finite prefix of $\sigma$, in which case the run is called incomplete.

In case the starting configuration is $(q_0, z_0)$, $r$ will be simply referred to as a run of $M$ on $\sigma$.

Every run 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 $r$.

Referring to Definition 1.2, we define $\text{INS}(r) = \text{In}(f_r)$. $\text{INS}(r)$ is the set of all states in $K$ entered infinitely many times by $M$ during run $r$.

It must be clearly understood that in a complete run $M$ actually scans the whole input $\sigma$, while in an incomplete run $M$ enters an infinite $\epsilon$-loop.

Note that there is also the possibility that in a certain computation the automaton is blocked on $\sigma$ not because of an infinite $\epsilon$-loop. However, subsequently we shall be interested only in complete runs; thus, unless otherwise specified, by a run we shall always mean a complete run.

An $\omega$-type pushdown automaton ($\omega$-PDA) is the system $M = (K, \Sigma, \Gamma, \delta, q_0, z_0, F)$, where $M_1 = (K, \Sigma, \Gamma, \delta, q_0, z_0)$ is a PDM and $F \subseteq 2^K$ is the collection of designated sets. $M$ will sometimes be written as $(M_1, F)$. Define:

$$T(M) = \{ \sigma \in \Sigma^\omega \mid \text{there exists a run of } M_1 \text{ on } \sigma \text{ s.t. } \text{INS}(r) \in F \}.$$  

$T(M)$ is the $\omega$-language accepted by $M$. Every $\omega$-word $\sigma \in T(M)$ is accepted by $M$.

The class of $\omega$-languages accepted by $\omega$-PDA's will be denoted by $\text{PDL}_\omega$.

We now define the notion of an $\omega$-context-free grammar. A similar definition can be used to define type 0,1 or 3 $\omega$-grammars, thus yielding a "Chomsky Hierarchy"
of $\omega$-grammars. Though a few results concerning type 0 $\omega$-grammars have appeared in [Co&Go1], there is a forthcoming paper [Co&Go2] devoted exclusively to the study of general $\omega$-grammars and of $\omega$-computations by Turing machines. (See also [Co&Go3] and [Go&Co].)

**Definition 1.6** An $\omega$-context-free grammar ($\omega$-CFG) (with variable repetition sets) is a quintuple $G = (V_N,V_T,P,S,F)$, where $G_1 = (V_N,V_T,P,S)$ is an ordinary context free grammar (CFG), the rules in $P$ are all of the form $A \rightarrow \beta$, where $A \in V_N$, $\alpha \in V^*$ and $F \subseteq 2^{V_N}$. The sets in $F$ are called the repetition sets of the 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 $G$:

$$d: \alpha = u \dot \gamma \alpha \overset{\omega}{\Rightarrow} u_1 \alpha \overset{\omega}{\Rightarrow} u_1 u_2 \alpha \overset{\omega}{\Rightarrow} \ldots \overset{\omega}{\Rightarrow} u_1 u_2 \ldots = \sigma$$

where for each $i$, $u_i \in V_T^*$, $\alpha_i \in V^*$ and $X_i \in V_N$. Let $\sigma = \prod_{i=1}^{\infty} u_i$. Note that the derivation need not be leftmost, since some of the $u_i$ may be empty. If $\sigma \in V_T^\omega$, we write $d: \alpha \overset{\omega}{\Rightarrow} (G) \Rightarrow \sigma$.

Define two mappings $d_P: N \rightarrow P$ and $d_V: N \rightarrow V$ as follows: $d_P(i) =$ the production used in the $i$-th step of $d$; and $d_V(i) =$ the variable rewritten in the $i$-th step of $d$. Also define $\text{INV}(d) = \text{In}(d_V)$ and $\text{INP}(d) = \text{In}(d_P)$.

$\text{INV}(d)$ ($\text{INP}(d)$) is the set of variables rewritten (productions used) infinitely many times during derivation $d$.

If in derivation $d$ above for each $i \geq 0$, $X_i$ is the variable rewritten in step $i + 1$, then $d$ is called a leftmost derivation. Define:
\[ L_{\omega}^L(G) = \{ \sigma \in \mathcal{V}_T^\omega \mid \text{there exists a derivation } d: S \xrightarrow{G} \omega \sigma, \text{ INV}(d) \in F \} \]

\[ L_{\omega}^N(G) = \{ \sigma \in \mathcal{V}_T^\omega \mid \text{there exists a leftmost derivation } d: S \xrightarrow{G} \omega \sigma, \text{ INV}(d) \in F \} \].

\( L_{\omega}^L(G)[L_{\omega}^N(G)] \) is the \( \omega \)-language generated by \( G \) by leftmost [non-leftmost] derivations.

Let \( \text{CFL}_\omega[nL-\text{CFL}_\omega] \) denote the class of \( \omega \)-languages of the form \( L_{\omega}^L(G)[L_{\omega}^N(G)] \), where \( G \) is an \( \omega \)-CFG.

**Example 1.1** Consider the \( \omega \)-CFG \( G = ( \{ S, T \}, \{ a, b \}, \{ S \rightarrow TS; T \rightarrow aTb; T \rightarrow ab \}, S, \{ \{ S, T \} \} ) \). As for leftmost derivations in \( G \), since there is a single repetition

set containing both variables \( S \) and \( T \), certain leftmost \( \omega \)-derivations, such as:

\[ d: S \Rightarrow TS \Rightarrow aTbS \Rightarrow \ldots \Rightarrow a^{i_1}Tb^{i_2}S \Rightarrow a^{i_1+1}Tb^{i_2+1}S \Rightarrow \ldots \]

though generating \( \omega \)-strings (\( d \) above generates \( a^{\omega} \)), do not contribute any
members to the \( \omega \)-language \( L_{\omega}^L(G) \), because the set of variables rewritten
indefinitely many times in \( d(\text{INV}(d)) \) consists of \( T \) alone. However, considering
non-leftmost derivations, any \( \omega \)-string of the form \( ( \prod_{i=1}^k a^{n_i}b^{n_i})^{\omega}, n_i \geq 1, \)
\( 1 \leq i \leq k, k \geq 1, \) is in \( L_{\omega}^N(G) \), since it is generated by a non-leftmost
derivation \( d \) with \( \text{INV}(d) = \{ S, T \} \) (this is because one can apply the \( S \)-production
"vacuously" indefinitely many times in the rightmost "unreached" part of the
sentential form, not affecting the terminal \( \omega \)-string generated on the left).

Therefore, denoting \( L_o = \{ a^n b^n \mid n \geq 1 \} \), we have \( L_{\omega}^L(G) = L_o^{\omega} \), whereas
\( L_{\omega}^N(G) = L_o^{\omega} \cup L_o^{*} a^{\omega} \).

In [Co&Go1] a study of leftmost versus non-leftmost generation in \( \omega \)-CFG's
was made, yielding the following results.

**Proposition 1.2** The \( \omega \)-language \( L^\omega_o \), where \( L_o = \{ a^n b^n \mid n \geq 1 \} \), cannot be generated by non-leftmost derivations by any \( \omega \)-CFG.

Another important result is:

**Theorem 1.3** \( n^\omega \text{-CFL} \subseteq \text{CFL}^\omega \).

**Corollary 1.4** \( n^\omega \text{-CFL} \neq \text{CFL}^\omega \).

By the above results, leftmost generation is the more powerful generation model in \( \omega \)-CFG's. Moreover, the family \( \text{CFL}^\omega \) possesses several elegant characterizations (see Theorem 1.5 below), whereas there seems to be no natural characterization for \( n^\omega \text{-CFL} \) (or at least we have been unable to find one). Therefore, we have chosen leftmost generation as our standard definition of generation in \( \omega \)-CFG's.

**Definition 1.7** An \( \omega \)-context-free language (\( \omega \)-CFL) is any \( \omega \)-language of the form \( L^\omega(G) \), where \( G \) is an \( \omega \)-CFG.

Thus, \( \text{CFL}^\omega \) denotes the collection of all \( \omega \)-CFL's.

**Remark 1.1** One can also define an \( \omega \)-CFG with production repetition sets, i.e. \( G = (V_N,V_T,P,S,F) \), where \( F \subseteq 2^P \) is the collection of production repetition sets, and \( L^\omega(G) \) \( \subseteq \{(n^\alpha(G) \mid \alpha \in \text{INP}(d) \subseteq F \) is defined as the set of all \( \omega \)-words generated by leftmost [non-leftmost] derivations \( d \) s.t. \( \text{INP}(d) \subseteq F \). However, as was proved in [Co&Gd1], the two models of \( \omega \)-CFG's, with variable repetition sets and with production repetition sets, are equivalent in generation power. This is true both for leftmost as well as for non-leftmost generation. For \( \omega \)-CFG's
it is usually more convenient to use variable repetition sets, whereas for type 0 or 1 \( \omega \)-grammars, only the model of production repetition sets is meaningful.

In [Co&Go] the connection between \( \omega \)-PDA's, \( \omega \)-KC(CF) and \( \omega \)-CFG's was established, leading to the following characterization theorem for \( \omega \)-CFL's.

**Theorem 1.5**  Main Characterization Theorem for \( \omega \)-Context Free Languages

\[ \text{CFL}_\omega \] coincides with \( \omega \)-KC(CF), which in turn, coincides with the class of \( \omega \)-PDA languages (which also equals the class of \( \omega \)-languages recognized by \( \omega \)-PDA's by emptying the pushdown store infinitely many times (see Definition 3.2)).

Thus precisely \( \text{CFL}_\omega \) is obtained by applying the \( \omega \)-Kleene closure operator to the context free languages. This generalizes McNaughton's characterization of the \( \omega \)-regular languages (Theorem 1.1).

Incidentally, one cannot obtain analogous characterizations for families higher in the Chomsky Hierarchy, as shown by the following example.

**Example 1.2** Let \( L_1 \) be the \( \omega \)-language consisting of the single \( \omega \)-string \( \prod 0^i 1 \). We claim that for no family of languages \( \mathcal{L} \), does \( L_1 \) belong to the \( \omega \)-Kleene closure of \( \mathcal{L} \). For suppose \( L_1 \in \omega \text{-KC(} \mathcal{L} \text{)} \) for some collection of languages \( \mathcal{L} \); then there exist, for some \( n \geq 1 \), 2\( n \) languages \( U_i, V_i \in \mathcal{L}, i = 1, \ldots, n \), s.t. \( L_1 = \cup_{i=1}^{n} U_i V_i^\omega \); but because of the special structure of \( L_1 \) such an equation is clearly impossible.

Now \( L_1 \) can be justifiably called a context sensitive \( \omega \)-language, as it is generated by the following context sensitive grammar \( G \) by non-leftmost
derivations and production repetition sets. To describe $G$ we borrow the terminology and notation of $\omega$-CFG’s (see Definition 1.6 and also Remark 1.1). Thus let $G = (\{S,X,X_1,\$,\$,1,2,3\}, \{0,1\}, P_1\{S \rightarrow 01\$X_13\}, S)$ be a context sensitive grammar, where $P_1$ is the following set of rules:

\begin{align*}
(1) \quad & $X_1$ \rightarrow 0$X_11 \\
(2) \quad & X_1X \rightarrow XX_1 \\
(3) \quad & X_1\$ \rightarrow \$X_11 \\
(4) \quad & \$X_1 \rightarrow 021\$2X_1X_1 \\
(5) \quad & \$2X_1 \rightarrow X\$2 \\
\end{align*}

One can easily verify that:

$L_1 = L_{\text{hL}}(G) = \{\sigma \in \{0,1\}^\omega | \text{there exists a derivation } d:S \xrightarrow{\omega}(G) \Rightarrow \sigma, \text{ INF}(d) = P_1\}.$

A variety of modes of $\omega$-recognition by $\omega$-automata will now be defined. The notion of "$i$-acceptance", $i = 1,1',2,2',3$ (first introduced in [Lan]) is general in that it does not refer to any specific type of device, but rather to the mechanism used to recognize $\omega$-length inputs. As will become apparent later, the index $i$ in "$i$-acceptance" can be considered a measure for the complexity of the recognized $\omega$-language.

**Definition 1.8** Let $f: N \rightarrow S$ be a mapping of the set of natural numbers into a set $S$, and let $F \subseteq 2^S$. We say that mapping $f$ is:

- **1-accepting** w.r.t. $F$ if $\forall t \in F_1 \in F$ $(\forall t) f(t) \in F_1$.
- **1'-accepting** w.r.t. $F$ if $\exists t \in F_1 \in F$ $(\forall t) f(t) \in F_1$.
- **2-accepting** w.r.t. $F$ if $\forall F_1 \in F \text{ INF}(f) \cap F_1 \neq \emptyset$.
- **2'-accepting** w.r.t. $F$ if $\exists F_1 \in F \text{ INF}(f) \subseteq F_1$.
- **3-accepting** w.r.t. $F$ if $\text{ INF}(f) \in F$. 
A mapping $f: \mathbb{N} \to S$ is $i$-accepting ($i=1,1',2,2',3$) with respect to a single subset $F \subseteq S$ if it is $i$-accepting with respect to $\{F\} \subseteq 2^S$.

Let $M = (M_1, F)$ be an $\omega$-PDA($\omega$-FSA). For $i=1,1',2,2',3$, define:

$$T_i(M) = \{ \sigma \in \omega^\omega | \text{there exists a (complete) run } r \text{ of } M \text{ on } \sigma \text{ s.t. } f_r \text{ is } i\text{-accepting w.r.t. } F \}.$$  

where for each $j$, $f_r(j)$ is the state entered in the $j$-th step of the computation described by run $r$.

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

As for $i=3$, note that $3$-acceptance coincides with the usual notion of acceptance introduced in Definitions 1.3 and 1.5. Thus subsequently, $3$-acceptance will be simply referred to as acceptance, and $T_3(M)$ will be denoted by $T(M)$.

Informally, an $\omega$-input $\sigma$ is $1$-accepted by $M$ iff there exists a (complete) run of $M$ on $\sigma$ during which $M$ passes through a state belonging to some designated set in $F$ at least once. $\sigma$ is $1'$-accepted by $M$ if, for some designated set $H \in F$, there exists a run of $M$ on $\sigma$ s.t all states entered by $M$ during this run are in $H$. $\omega$-input $\sigma$ is $2$-accepted by $M$ in case there exists a run of $M$ on $\sigma$ during which the set of states entered infinitely many times by $M$ contains at least one state from some designated set. $\sigma$ is $2'$-accepted by $M$ if there exist a designated set $H \in F$ and a run of $M$ on $\sigma$ representing a computation in which, after some finite number of steps, from a certain step on, all states entered by $M$ belong to $H$. Finally, $\sigma$ is $3$-accepted, i.e. accepted, by $M$ if for some run of $M$ on $\sigma$, the set of states entered infinitely many times by $M$ coincides with one of the designated sets.
As follows from the above definition, in the case of 1-acceptance and 2-acceptance, one may assume with no loss of generality that there is only a single designated set in the collection \( F \).

**Notation 1.1** An \( \omega \)-PDA with a unique designated set will be denoted by \( U-\omega \)-PDA. In this case we write \( M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F) \), where \( F \subseteq K \) is the designated set.

**Definition 1.9** Two \( \omega \)-PDA's \( M \) and \( M' \) are \( i \)-equivalent (for \( i = 1, 1', 2, 2' \)) iff \( T_i(M) = T_i(M') \).

The above definitions are illustrated by the following example.

**Example 1.3** Let \( M = (\{q_0, q_1\}, \{a, b\}, \{Z_0, Z_1, a, b\}, \delta, q_0, Z_0) \) be a deterministic PDM, where \( \delta \) is defined as follows: For every \( c \in \Sigma \),
\[
\delta(q_0, c, Z_0) = (q_0, cZ_1), \quad \delta(q_0, c, c) = (q_0, cc), \quad \delta(q_1, c, Z_1) = (q_0, cZ_1);
\]
also
\[
\delta(q_0, \varepsilon, Z_1) = (q_1, Z_1) \quad \text{and} \quad \delta(q_0, a, b) = \delta(q_0, b, a) = (q_0, \varepsilon).
\]

For \( x \in \Sigma^* \) let \( \#_c(x) \) denote the no. of occurrences of letter \( c \) in \( x \).

Define the language \( L = \{ x \in \Sigma^* \mid \#_a(x) = \#_b(x) \} \) and the \( \omega \)-languages \( L_{ab} = \{ \sigma \in \Sigma^\omega \mid \forall n \geq 1, \#_a(\sigma/n) \geq \#_b(\sigma/n) \} \) and \( L_{ba} = \{ \sigma \in \Sigma^\omega \mid \forall n \geq 1, \#_b(\sigma/n) \geq \#_a(\sigma/n) \} \).

Clearly \( T_2(M, \{q_0, q_1\}) = T_2'(M, \{q_0, q_1\}) = \Sigma^\omega \); however, changing the designated set, we obtain \( T_2'(M, \{q_0\}) = L^*(L_{ab} \cup L_{ba}) \) and \( T_1'(M, \{q_0\}) = L_{ab} \cup L_{ba} \).

**Notation 1.2** For \( i = 1, 1', 2, 2' \), an \( \omega \)-language \( i \)-accepted by some \( \omega \)-FSA will be called an \textbf{Ai-\( \omega \)-regular language}. For \( i = 1, 1', 2, 2' \), the class of \( \omega \)-languages \( i \)-accepted by \( \omega \)-PDA's will be denoted by \( \text{Ai-PDL}_{\omega} \).
The families of $A_i$-$\omega$-regular languages were studied in [Lan] and [Hos].

In [Co&Go1] an extensive study of $i$-recognizability by $\omega$-PDA's was made; the classes $A_i$-$PDL_\omega$ were characterized and shown to constitute a hierarchy within $CFL_\omega$. A summary of the main results concerning these classes will now be presented.

We start with two characterization theorems.

**Theorem 1.6** $A_1'$-$PDL_\omega$ coincides with the class of $\omega$-languages generated by production repetition sets by $\omega$-$CFG$'s of the form $G = (V_N, V_T, P, S, S')$.

**Theorem 1.7** $A_2'$-$PDL_\omega$ coincides with the class of $\omega$-languages of the form $k \sum_{i=1}^{k} U_i V_i$, where $k \geq 1$ and for each $1 \leq i \leq k$, $U_i$ is a CFL and $V_i \in A_1'$-$PDL_\omega$.

The following theorems establish the hierarchy of families $A_i$-$PDL_\omega$.

**Theorem 1.8** The class of $\omega$-regular languages is properly included in $A_1'$-$PDL_\omega$.

In fact, the $\omega$-language $L_{ab}$ from Example 1.3 is a non-$\omega$-regular member of $A_1'$-$PDL_\omega$.

**Theorem 1.9** (a) $A_1$-$PDL_\omega = A_2'$-$PDL_\omega$. (b) $A_2$-$PDL_\omega = CFL_\omega$.

**Theorem 1.10** For each non-regular language $L$ over alphabet $\Sigma$ and symbol $d \in \Sigma$, we have: (a) $Ld^\omega \notin A_1'$-$PDL_\omega$. (b) $(Ld)^\omega \notin A_2'$-$PDL_\omega$.

Combining the above results, the following hierarchy of $\omega$-CFL families is obtained:
Theorem 1.11 \( \omega\text{-Reg} \subseteq A_1' - \text{PDL}_\omega \subseteq A_1 - \text{PDL}_\omega = A_2' - \text{PDL}_\omega \subseteq n^\omega - \text{CFL}_\omega \subseteq \text{CFL}_\omega = A_2 - \text{PDL}_\omega. \)

Open Question: Is the inclusion of \( A_2' - \text{PDL}_\omega \) in \( n^\omega - \text{CFL}_\omega \) proper?

Properties of \( \omega \)-context free languages were also discussed in [Co&Gol]. Closure properties analogous to those of the "finite-string" context free languages were obtained. Moreover, some new operations typical to \( \omega \)-languages were introduced and their properties studied; however, these will not be discussed here.

We now define two of the basic operations on \( \omega \)-languages which will be needed in the following sections.

Definition 1.10 Let \( L_1, L_2 \) be \( \omega \)-languages over \( \Sigma \). Define the quotient of \( L_1 \) with respect to \( L_2 \) \( (L_1/L_2) \), to be \( L_1/L_2 = \{ x \in \Sigma^* \ | \ y \in L_2 \text{ s.t. } xy \in L_1 \} \).

For any \( \omega \)-language \( L \) over \( \Sigma \), define \( \text{Init}(L) \) to be \( L/\Sigma^\omega \).

Note that the quotient of two \( \omega \)-languages is a finite string language.
Having completed the survey of previous work on \( \omega \)-context-free languages, we now define deterministic \( \omega \)-PDA's, which will be the topic of the following sections.

**Definition 1.11** A PDM \( M = (K, \Sigma, \Gamma, \delta, q_0, Z_0) \) is deterministic (DPDM) if for each \( q \in K, Z \in \Gamma, a \in \Sigma \): (1) \( \delta(q,a,Z) \) contains at most one element; (2) \( \delta(q,\epsilon,Z) \) contains at most one element, and (3) If \( \delta(q,\epsilon,Z) \) is not empty, then \( \delta(q,a,Z) \) is empty for all \( a \in \Sigma \). An \( \omega \)-PDA \( M = (M_1, F) \) is deterministic (\( \omega \)-DPDA) if \( M_1 \) is deterministic. An \( \omega \)-language accepted by an \( \omega \)-DPDA is a deterministic context free \( \omega \)-language (\( \omega \)-DCFL). The class of \( \omega \)-DCFL's will be denoted by \( DCFL_\omega \).

In case \( M \) is an \( \omega \)-DPDA, for every \( \sigma \in \Sigma^\omega \) there is a unique run \( r \) of \( M \) on \( \sigma \) determined by the starting configuration \((p,\gamma)\). In case \( r \) is a complete run we shall use the notation \( INS_M(p,\gamma,\sigma) \) instead of \( INS_M(r) \). In case \( M \) is a DFSM the unique run of \( M \) on \( \sigma \) is determined by the starting state \( p \) and the notation \( INS_M(p,\sigma) \) will be used. The subscript \( M \) will be omitted whenever no confusion arises.

**Definition 1.12** We say that a PDM \( M \) has Property C (the continuity property) iff for every \( \omega \)-tape \( \sigma \), there exists a complete run of \( M \) on \( \sigma \) (in which the whole tape \( \sigma \) is scanned). In case \( M \) is a DPDM with Property C the unique run of \( M \) on \( \sigma \) has to be complete.

**Notation 1.3** For \( i = 1',2,2' \), the class of \( \omega \)-languages \( i \)-accepted by \( \omega \)-DPDA's will be denoted by \( Ai-DPDL_\omega \). The class of \( \omega \)-languages \( 1 \)-accepted by \( \omega \)-DPDA's (\( \omega \)-DPDA's with Property C) will be denoted by \( Ai-DPDL_\omega (Al-DPDL_\omega) \), (see Lemma 2.1 and Remark 2.1 below).
2. CLOSURE PROPERTIES OF THE DETERMINISTIC \( \omega \)-CFL's

As one can expect, most of the well-known closure properties of the deterministic CFL's ([G&G]) can be generalized for \( \omega \)-DCFL's, though, due to the non-terminating nature of the \( \omega \)-strings, some of the proof become here rather complicated.

Exceptional in this section is Theorem 2.10, having no analogue in the classical theory. This theorem deals with a new way of producing \( \omega \)-languages from finite-string languages.

Before proceeding with the results we add the assumption that all \( \omega \)-DPDA's possess Property C. As in the case of non-deterministic \( \omega \)-PDA's [Co&Go], also for \( \omega \)-DPDA's there is no loss of generality in making this assumption. Given an \( \omega \)-DPDA \( M \), the elimination of infinite \( \varepsilon \)-loops is carried out exactly as for a DPDA [G&G], except that whenever \( M \) is blocked or enters an infinite \( \varepsilon \)-loop, the modified automaton will enter a new non-final state, in which it will scan the rest of the input.

Lemma 2.1 For every \( \omega \)-DPDA \( M \) and for every \( i = 1',2,2',3 \), there is an \( i \)-equivalent \( \omega \)-DPDA \( M' \) which has Property C.

Remark 2.1 Note that Lemma 2.1 above does not hold for 1-acceptance by \( \omega \)-DPDA's. Let \( M = (\{q_0\},\{0,1\},\{Z_0\},\delta,q_0,Z_0\{q_0\}) \) be a \( \omega \)-DPDA, where \( \delta(q_0,1,Z_0) = (q_0,Z_0) \); then \( T_1(M) = \{i^0\} \). However, \( \{i^0\} \) cannot be 1-accepted by any \( \omega \)-DPDA which has Property C (see Theorem 4.2.5a). Informally, this is because a 1-accepting \( \omega \)-DPDA decides to accept a string after having scanned only a finite prefix of the string, thus the only way in which such an automaton can later on "regret" its decision is by blocking the run. But if we assume that every run of \( M \) is complete, the automaton cannot change its decision after having passed once through a final state.
It follows that $\omega$-DPDL $\subseteq \bar{\omega}$-DPDL $\subseteq$ Al-DPDL $\subseteq$ Al-DPDL $\subseteq$ A1-DPDL $\subseteq$ A1-DPDL $\subseteq$ A1-DPDL.

For our purposes it turns out to be more convenient to deal with $\omega$-DPDA's with Property C, including the case of 1-acceptance. Therefore, in spite of the above remark, we shall henceforth assume that all $\omega$-DPDA's possess Property C.

By Lemma 2.1 above, we immediately obtain:

**Proposition 2.2** DCFL $\omega$ is closed under complementation.

Using the standard direct product construction we also have the following:

**Proposition 2.3** DCFL $\omega$ is closed under intersection with $\omega$-regular languages.

**Corollary 2.4** If $L$ is an $\omega$-DCFL and $R$ is an $\omega$-regular language, then $L \cup R$, $L - R$ and $R - L$ are $\omega$-DCFL's.

The quotient with respect to an $\omega$-regular language turns out to be an appropriate tool for investigating properties of $\omega$-languages.

Let $M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be an $\omega$-DPDA and $A = (K_A, \Sigma, \delta_A, p_0, F_A)$ be an $\omega$-DFSA. Following [H&U, pp.173], a predicting machine, $\Pi_A(M)$, for $M$ and $A$ will be build to prove that $T(M)/T(A)$ is a DCFL.

The predicting machine $\Pi_A(M)$ will have on its store $[Z_1, \alpha_1][Z_{i-1}, \alpha_{i-1}] \cdots [Z_1, \alpha_1]$, if the store of $M$ is $Z_{i-1} \cdots Z_1$. Here $\alpha_j$ is a mapping from $K \times K_A$ into \{0,1\} defined as follows: for $1 \leq j \leq i$, $\alpha_j(q,p) = 1$ iff there exists $\sigma \in \Sigma^\omega$ s.t. both $\text{INS}_M(q, Z_{j-1} \cdots Z_1, \sigma) \in F$ and $\text{INS}_A(p, \sigma) \in F_A$ (Definition 1.11). Otherwise $\alpha_j(q,p) = 0$. Here too, $\alpha_j$ depends only on the lowest $j-1$ symbols on the pushdown store of $M$ and not on the $j$-th symbol.
Lemma 2.5  For given $\omega$-DFSA $A = (K_A, \Sigma, \delta_A, P_0, F_A)$ and $\omega$-DPDA $M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, there is an effective procedure for constructing a predicting machine $M' = \Pi_A(M)$ as above. Furthermore, $M'$ is a DPDM.

Proof. The proof follows the outlines of the proof in [H&U]. Let $M' = (K, \Sigma, \delta', \Gamma \times \mathcal{C}, q_0, [Z_0, \alpha_0])$ be a DPDM, where $\mathcal{C}$ is the set of all mappings from $K \times K_A$ to the set $\{0, 1\}$ and $\alpha_0$ is the zero mapping. For $q \in K$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$ and $\alpha \in \mathcal{C}$, $\delta'$ is defined as follows:

1. If $\delta(q, a, Z) = (q', \varepsilon)$, then $\delta'(q, a, [Z, \alpha]) = (q', \varepsilon)$

2. If $\delta(q, a, Z) = (q', Z_1 \ldots Z_r)$, then $\delta'(q, a, [Z, \alpha]) = (q', [Z_1, \alpha_1] \ldots [Z_r, \alpha_r])$,
   where $\alpha_i$, $1 \leq i \leq r$, are appropriate mappings.

To show that for $i \leq r$, $\alpha_{i-1}$ can be computed from $Z_i$ and $\alpha_i$ we only mention that $\alpha_{i-1}(q_1, p_1)$ equals 1 in two cases:

(a) $T((K, \Sigma, \Gamma, \delta, q_1, Z_1, F)) \cap T((K_A, \Sigma, \delta_A, p_1, F_A)) \neq \emptyset$ ;

(b) There exist $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^\omega$ s.t. the following conditions hold:

1. $\delta_A(p_1, w_1) = p_2$ for some $p_2 \in K_A$;

2. $\text{INS}_A(p_2, w_2) \in F_A$ ;

3. $w_1: (q_1, Z_1) \xrightarrow[*]{M} (q_2, \varepsilon)$ for some $q_2 \in K$ ;

4. $\text{INS}_M(q_2, \gamma_1, w_2) \in F$, where $\gamma_1$ is the contents of the pushdown store after $Z_1$ is erased.

With the above modifications, it is clear that the DPDM $M' = \Pi_A(M)$ is the appropriate predicting machine.

The effectiveness of the construction follows from the decidability results in [Co&Go].
Following the proof of Theorem 12.4 in [H&H], we have:

**Theorem 2.6** Let $L$ be an $\omega$-DCFL and $R$ be an $\omega$-regular language. Then $L/R$ is a DCFL.

**Corollary 2.7** Let $L$ be an $\omega$-DCFL; then $\text{Init}(L)$ is a DCFL.

The following lemma is obvious.

**Lemma 2.8** For any $L$ in $\text{DCFL}_\omega (\mathbb{A}^{-}\text{DPDL}_\omega, i = 1,1',2,2')$ over alphabet $\Sigma$ and for any $x \in \Sigma^\omega$, $x \backslash L = \{ \sigma \in \Sigma^\omega | x\sigma \in L \}$ is in $\text{DCFL}_\omega (\mathbb{A}^{-}\text{DPDL}_\omega, i = 1,1',2,2' \ \text{resp.})$.

The next theorem is a generalization of a well-known closure result on DCFL's ([G&G]); however, the proof for $\omega$-DCFL's is rather involved and will therefore be omitted and given in full in Appendix A.

**Theorem 2.9** Let $L$ be a DCFL and $R$ be an $\omega$-regular language; then $LR$ is an $\omega$-DCFL.

We now consider inverse GSM mappings. Such mappings provide a new way of producing $\omega$-languages from finite-string languages, as will be shown in Theorem 2.10.

**Definition 2.1** Let $S = (K, \Sigma, \Delta, q_0)$ be a $(\Sigma, \Delta)$-GSM, where $K$ is a finite set of states, $\Sigma$ - the input alphabet, $\Delta$ - the output alphabet, $\delta$ is a mapping from $K \times \Sigma$ to finite subsets of $K \times \Delta^*$ and $q_0$ is the initial state. $S$ is called **deterministic $(\Sigma, \Delta)$-GSM ($(\Sigma, \Delta)$-DGS)$ iff $\delta: K \times \Sigma \rightarrow K \times \Delta^*$.

Let $\sigma = \prod_{i=1}^{\infty} a_i \in \Sigma^\omega$, where $a_i \in \Sigma \ \forall i \geq 1$. An infinite sequence
\[ r = \{ (q_i, x_i) \}_{i \geq 1}, \text{ where } q_i \in K \land x_i \in \Delta^*, \text{ is called a run of } S \text{ on } \sigma \text{ if:} \]

(a) \((q_1, x_1) = (q_0, \epsilon)\); (b) for each \(i \geq 1\), \((q_{i+1}, x_{i+1}) \in \delta(q_i, a_i)\).

Define \(S(\sigma) = \{ \sigma \in \Delta^* \cup \Delta^\omega \mid \text{there exists a run } r = \{ (q_i, x_i) \}_{i \geq 1} \text{ of} \)

\(S \text{ on } \sigma \text{ s.t. } \sigma_1 = \prod_{i=1}^{\infty} x_i \} \). For \(L \subseteq \Sigma^\omega\), let \(S(L) = \bigcup_{\sigma \in L} S(\sigma)\).

Now define for \(\sigma_1 \in \Delta^* \cup \Delta^\omega\), \(S^{-1}(\sigma_1) = \{ \sigma \in \Sigma^\omega \mid \sigma \in S(\sigma) \}. \)

For \(L \subseteq \Delta^\omega\) let \(S^{-1}(L) = \bigcup_{\sigma_1 \in L} S^{-1}(\sigma_1)\); \(S^{-1}(L)\) is an inverse GSM mapping.

For each \(\omega\)-language \(L\) over \(\Sigma\), GSM \(S\) is called \(\omega\)-preserving on \(L\)

if \(S(L) \subseteq \Delta^\omega\). \(S\) is \(\omega\)-preserving if \(S(\Sigma^\omega) \subseteq \Delta^\omega\).

**Theorem 2.10** For any non-\(\omega\)-preserving deterministic \((\Sigma, \Delta)\)-GSM \(S\) and

DCFL \(L\), \(S^{-1}(L) \cap \Delta^\omega\) is in \(A_2'\)-DPDL\(\_\omega\).

**Proof.** Let \(S = (K_S, \Sigma, \Delta, \delta_S, p_o)\) be a \((\Sigma, \Delta)\)-DGSM and let \(M = (K_M, \Delta, \Gamma, \delta_M, q_o, F_M)\)

be a DPDA that accepts \(L\). Following the construction in [H&U, pp.172],

define the \(U\)-\(\omega\)-DPDA \(M' = (K, \Sigma, \Gamma, \delta, q'_o, Z_o, K_2)\), where \(r = \max(\{ |w|, (p, w) \in K \})\)

is in the range of \(\delta\), \(K_1 = \{ [q, p, w] \mid q \in K, p \in K_S, w \in \Delta^i, i = 1 \}\),

\(K_2 = \{ [q, p] \mid q \in K, p \in K_S \}, K = K_1 \cup K_2 \) and \(q'_o = [q_o, p_o, \epsilon]\). For all \(q, q_1 \in K_M, p, p_1 \in K, a, b \in \Sigma, w \in \Delta^*, \gamma \in \Gamma^* \) and \(Z \in \Gamma, \delta\) is defined as follows:

1) If \(\delta_M(q, \epsilon, Z) = \emptyset \) and \(\delta_S(p, b) = (p_1, w)\), then \(\delta([q, p, \epsilon], b, Z) = ([q_1, p_1, w], Z)\).

2) If \(\delta_M(q, \epsilon, Z) = (q_1, \gamma)\), then \(\delta([q, p, w], \epsilon, Z) = ([q_1, p, w], \gamma)\) except when \(w = \epsilon\) and \(q \in F\).

3) If \(\delta_M(q, a, Z) = (q_1, \gamma)\), then \(\delta([q, p, aw], \epsilon, Z) = ([q_1, p, w], \gamma)\).
4) If \( q \in F \), then \( \delta([q,p,\varepsilon], \varepsilon, Z) = ([q,p], Z) \)

5) If \( \delta_S(p,b) = (p_1, \varepsilon) \), then \( \delta([q,p], b, Z) = ([q,p_1], Z) \)

6) If \( \delta_S(p,b) = (p_1, w) \) and \( w \neq \varepsilon \), then \( \delta([q,p], b, Z) = ([q,p_1, w], Z) \)

One can verify that \( T_2(M') = S^{-1}(L) \).

Similarly, one can prove:

**Proposition 2.11** \( \text{DCFL}_\omega \) is closed under inverse DGSM mapping.

We now discuss some non-closure properties of the \( \omega \)-DCFLs.

**Proposition 2.12** \( \text{DCFL}_\omega \) is not closed under union and intersection.

**Proof** The languages \( L_1 = \{a^ib^i j^i | i, j \geq 1\} \) and \( L_2 = \{a^ib^i j^i | i, j \geq 1\} \)
are DCFL's, but \( L_1 \cap L_2 \) is not a CFL and by [G&G] \( L_1 \cup L_2 \) is not a DCFL.
Let \( R_0 = a^+b^+a^+ \); then \( L_1 b^\omega \) and \( L_2 b^\omega \) belong to \( \text{DCFL}_\omega \), but
\( (L_1 b^\omega \cap L_2 b^\omega) / b^\omega \) and \( R_0 = L_1 \cap L_2 \) and \( ((L_1 b^\omega \cup L_2 b^\omega)/b^\omega) \cap R_0 = L_1 \cup L_2 \);
hence by Theorem 2.6 \( (L_1 \cup L_2)b^\omega \) and \( (L_1 \cap L_2)b^\omega \) are not in \( \text{DCFL}_\omega \).

In the following remarks, let \( L_1, L_2 \) be as in Proposition 2.12 above.

**Remark 2.2** (a) There exists a DCFL \( L \) s.t. \( L^\omega \notin \text{DCFL}_\omega \).
Let \( L = L_2 \cup cL_1 \cup \{c\} \). Then \( L^\omega \cap c^+a^+c^\omega = c(L_1 \cup L_2)c^\omega \subseteq L_0 \',
(L_0 / c^\omega) \cap c^+a^+c^\omega = c(L_1 \cup L_2) \), the latter is not a DCFL, and by Theorem 2.6,
\( L_0 \), and therefore also \( L^\omega \), are not in \( \text{DCFL}_\omega \).

(b) One can construct an \( \varepsilon \)-free homomorphism \( h \) s.t. \( h(cL_1 b^\omega dL_2 b^\omega) = c(L_1 \cup L_2)b^\omega \)
hence \( \text{DCFL}_\omega \) is not closed under homomorphism.

**Remark 2.3** We saw that the Init of an \( \omega \)-DCFL is a deterministic CFL.

The following example shows an \( \omega \)-CFL which is not in \( \text{DCFL}_\omega \) but the Init...
of which is $\Sigma^*$. Let $\Sigma = \{a, b, c\}$, $L_0 = cL_1 \cup L_2$ and $L = \Sigma^*L_0b^\omega$. Clearly $\text{Init}(L) = \Sigma^*$. Moreover, $L_0b^\omega \in \text{DCFL}_\omega$ but $L \cap ca^+b^+a^+b^\omega = c(L_1 \cup L_2)b^\omega$
and $(c(L_1 \cup L_2)b^\omega/b^\omega) \cap ca^+a^+ = c(L_1 \cup L_2)$. The latter is not a DCFL, hence $c(L_1 \cup L_2)b^\omega$ and $L$ are not in $\text{DCFL}_\omega$.

If we take $R = \Sigma^*$ and $L' = L_0b^\omega$ we have:

**Corollary 2.13** There exist a regular language $R$ and an $\omega$-DCFL $L'$ s.t. $RL'$ is not an $\omega$-DCFL.
3. THE $\omega$-KLEENE CLOSURE OF THE DETERMINISTIC CFL's AND VARIANTS OF $\omega$-DPDA's

In [Co&Go] the family of $\omega$-context free languages was characterized as $\omega$-KC(CF), the $\omega$-Kleene closure of the context free languages, and also as the family of $\omega$-PDA ($\omega$-empty-store PDA) languages (Theorem 2.5).

Attempting to derive similar characterizations for the $\omega$-DPDA languages, an interesting hierarchy of the deterministic variants of the above mentioned families is obtained. In particular, the $\omega$-Kleene closure of the strict deterministic languages [Ha&Ha] is characterized by means of quasi-deterministic $\omega$-PDA languages.

**Notation 3.1** Let $\omega$-KC(DCF) denote the $\omega$-Kleene closure of the family of deterministic CFL's.

The next theorem shows that the characterization of CFL $\omega$ as $\omega$-KC(CF) does not hold for DCFL $\omega$.

**Theorem 3.1** $\omega$-KC(DCF) properly contains DCFL $\omega$, and is a proper subset of CFL $\omega$.

**Proof.** (a) First prove DCFL $\omega \subseteq \omega$-KC(DCF). Let $M = (K,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ be an $\omega$-DPDA. Since by definition, $\omega$-KC($\mathcal{L}$), for any family $\mathcal{L}$, is closed under union, we may assume that $F$ consists of only one set, denoted by $F$ itself. Let $F = \{q_i\}_{i=1}^{L}$ and define $B = \{(p,\gamma) \mid p \in F, |\gamma| \geq 1, (p,\gamma) \text{ in the range of } \delta\}$. For every $(p,\gamma)$ in $B$, define $U_{(p,\gamma)}$ to be the language accepted by the DPDA $M_{(p,\gamma)} = (K \cup \{q_F\},\Sigma,\Gamma,\delta,q_0,Z_0,\{q_F\})$, where $q_F \in K$, $\delta(q,a,Z) = \delta(q,a,Z)$ if $\delta(q,a,Z) \neq (p,\gamma)$, $\delta(q,a,Z) = (q_F,Z)$ if $\delta(q,a,Z) = (p,\gamma)$, and $\delta(q_F,\epsilon,Z) = (p,\gamma)$. For each $(p,\gamma) \in B$, define $V_{(p,\gamma)}$ as the language defined
by the DPDA \( M'_{(p,\gamma)} = (K_1 \cup \{q_0\}, \Sigma, \Gamma, \delta_1, q_0, \overline{Z}_0, \{q_F\}) \), where \( \overline{Z}_0 \notin \Gamma \),
\( F(1) = \{q(1)|q \in F\} \) for each \( 0 \leq i \leq \ell \) and \( K_1 = \{q_F\} \cup \{ \cup_{i=0}^{\ell} F(i)\} \).

\( \delta_1 \) is defined as follows: \( \delta_1(q_0, \varepsilon, \overline{Z}_0) = (p(0), \gamma). \) For \( i = 0, \ldots, \ell - 1 \),
\( \delta_1(p(i), a, Z) = (q(i+1), \alpha) \) if \( \delta(p, a, Z) = (q(i+1), \alpha). \) For all \( r, q \in F \) and \( i = 0, \ldots, \ell \), \( \delta_1(r(i), a, Z) = (q(i), \alpha) \) if \( \delta(r, a, Z) = (q, \alpha) \), except when \( i < \ell \) and \( q = q_{i+1} \) or when \( i = \ell \) and \( (p, \gamma) = (q, \alpha) \); in the latter case, \( \delta_1(r^{(\ell)}, a, Z) = (q_F, Z). \) Also let \( \delta(q_F, \varepsilon, Z) = (p^{(\ell)}, Z) \).

Informally, \( M'_{(p,\gamma)} \) enters \( q_F \) whenever the computation of \( M \) completes a cycle through all the states of \( F \) and \( M \) makes a move of the form \((p, \gamma)\). We have \( T(M) = \bigcup_{(p, \gamma) \in B} (p, \gamma)^{\omega} \). Hence \( \text{DCFL}_\omega \subseteq \omega-\text{KC}(\text{DCF}) \).

Remark 2.2 implies that the inclusion is proper.

(b) Clearly \( \omega-\text{KC}(\text{DCF}) \subseteq \omega-\text{KC}(\text{CF}) = \text{CFL}_\omega \). Let \( \Sigma = \{a, b\} \), \( L = \{ww^R | w \in \Sigma^*\} \) and \( L_1 = Ld^\omega \cup \Sigma^\omega \). Suppose \( L_1 \in \omega-\text{KC}(\text{DCF}) \). Then \( L_1 = \bigcup_{i=1}^{n} G_i H_i^{\omega} \), where \( n \geq 1 \) and for each \( n \), \( G_i, H_i \) are DCFL's. Let \( i_j, j = 1, \ldots, \ell \), be those indices s.t. \( G_{i_j} H_{i_j} \subseteq Ld^\omega \); then \( H_{i_j} \subseteq d^+ \) for \( 1 \leq j \leq \ell \). Let \( L_2 = \bigcup_{j=1}^{\ell} \bigcup_{i_j} G_{i_j} \); then \( (L_2 d / d^+ \cap \Sigma^+) = L. \)

The DCFL's are closed under right concatenation, intersection and right quotient with the regular languages and the same holds also for their union closure. It follows that \( L \) is a union of DCFL's, which is false by [G&G]. Hence we obtain \( L_1 \notin \omega-\text{KC}(\text{DCF}). \)

\[ \square \]
We now define the deterministic version of an $\omega$-empty pushdown automaton.

**Definition 3.2** An $\omega$-empty pushdown automaton ($\omega$-EPDA) is a PDM $M = (K, \Sigma, \Gamma, \delta, q_0, X)$ satisfying the following condition: for any $a \in \Sigma \cup \{\varepsilon\}$ and $q \in K$, if $(q', \gamma) \in \delta(q, a, Z)$ then:

(a) $\gamma = \gamma'Z$ for some $\gamma' \in (\Gamma \setminus \{X\})^*$, in case $Z = X$;

(b) $\gamma \in (\Gamma \setminus \{X\})^*$ in case $Z \neq X$;

An $\omega$-EPDA $M$ is called deterministic ($\omega$-EDPDA) if $M$ is a DPDM.

The $\omega$-language accepted by $M$, by $\omega$-empty store, denoted $T_e(M)$, is defined as the set of $\omega$-tapes $\sigma$ s.t. there is a run of $M$ on $\sigma$ during which $M$ reaches $X$ on the pushdown store infinitely many times.

The class of $\omega$-languages accepted by $\omega$-EDPDA's will be denoted by $\text{EDPDL}_\omega$.

We now generalize the model of EDPDA by allowing the automaton to have more than one initial state. Except for the choice of initial state, the automaton remains completely deterministic. We shall call such an automaton quasi-deterministic.

**Definition 3.3** A quasi-deterministic $\omega$-EPDA ($\omega$-EQPDA) is a sixtuple $M = (K, \Sigma, \Gamma, \delta, Q_0, X)$ where $Q_0 = \{q_i\}_{i=1}^\ell \subseteq K$ and $M_i = (K, \Sigma, \Gamma, \delta, q_i, X)$ for $1 \leq i \leq \ell$ are $\omega$-EDPDA's. $T_e(M)$, the $\omega$-language accepted by $M$ by $\omega$-empty store is defined as $\bigcup_{i=1}^\ell T_e(M_i)$.

The class of $\omega$-languages accepted by $\omega$-EQPDA's will be denoted by $\text{EQPDL}_\omega$. 

Remark 3.1 Since every $\omega$-EQPDA can be considered as a set of $\omega$-DPDA's, each with a single initial state, each of the $\omega$-EDPDA's can be modified to have Property C as in Lemma 2.1. Thus we obtain an $\omega$-EQPDA in which every choice of initial state leads to a complete run.

We next derive a characterization of EQPD$\omega$ by means of the family of strict deterministic languages. As will be shown, the latter family constitutes a generator for the quasi-deterministic $\omega$-EPDA languages in the same fashion that the family of CFL's is a generator for CFL$\omega$.

Notation 3.4 Let SDCF denote the family of strict deterministic languages [Ha&Ha].

Theorem 3.2 EQPD$\omega$ = $\omega$-KC(SDCF).

Proof (a) Let $M = (K, \Sigma, \Gamma, \delta, Q_0, X)$ be an $\omega$-EQPDA, where $Q_0 = \{q_i\}_{i=1}^{\ell}$.

With no loss of generality we may assume that the initial states will never be re-entered. Since $T_e(M) = \bigcup_{i=1}^{\ell} T_e(M_i)$, where for each $i = 1, \ldots, \ell$,

$M_i = (K, \Sigma, \Gamma, \delta, \{q_i\}, X)$, and since, by definition, $\omega$-KC($\mathcal{L}$), for any family $\mathcal{L}$, is closed under union, we may also assume that $Q_0 = \{q_0\}$. Let $K = \{q_i\}_{i=0}^{n}$, and define $U_i$, $V_i$, $1 \leq i \leq n$, to be the following languages:

$U_i$ - the set of all prefix-free (finite) input words causing $M$ to transfer from configuration $(q_0, X)$ to $(q_i, X)$;

$V_i$ - the set of all prefix-free (finite) input words causing $M$ to transfer from configuration $(q_i, X)$ back to $(q_i, X)$.

One can verify that $T_e(M) = \bigcup_{i=1}^{\ell} U_i V_i^{\omega}$, where $U_i$ and $V_i$, $1 \leq i \leq n$, Technion - Computer Science Department - Technical Report CS0056 - 1975.
are accepted by empty store by deterministic pushdown automata.

(b) Let \( L_1, L_2 \) be strict deterministic. The proof that \( L_1 L_2 \in \text{EDPDL}_\omega \) follows the lines of the proof of [Co&Go, Lemma 4.1.11]. In this case, however, the \( \omega \)-EPDA constructed will be deterministic, as \( C_1, C_2 \) may be assumed to be deterministic. Since \( \text{EQPDL}_\omega \) is closed under union, we have \( \omega\text{-KC}(\text{SDCF}) \subseteq \text{EQPDL}_\omega \).

\[ \Box \]

Proposition 3.3 \( \omega\text{-KC}(\text{SDCF}) \) is incomparable with the class of \( \omega \)-regular languages.

Proof. Let \( L = \{a^i b^n \mid n \geq 1\} a^\omega \); then \( L \in \omega\text{-KC}(\text{SDCF}) \) but clearly \( L \) is not an \( \omega \)-regular language.

Now consider the \( \omega \)-regular language \( L = \Sigma^* a^\omega \).

Suppose there exist, for some \( n \), strict deterministic languages \( U_i, V_i \), \( 1 \leq i \leq n \), s.t. \( L = \bigcup_{i=1}^{n} U_i \bigcup_{i=1}^{n} V_i^\omega \). Clearly, \( V_i \subseteq \Sigma^* \) for \( 1 \leq i \leq n \). Now, since \( ba^\omega \in L \), there exist natural numbers \( i_1, r_1 \), s.t. \( c_1 = ba^r_1 \in U_{i_1} \).

Also, \( c_1 b a^\omega \in L \), so there exist \( i_2, r_2 \), s.t. \( c_2 = c_1 b a^r_2 \in U_{i_2} \) and \( i_1 \neq i_2 \) because no word in \( U_{i_1} \) is a prefix of another. Since \( c_2 b a^\omega \in L \), there exist \( i_3, r_3 \), s.t. \( c_3 = c_2 b a^r_3 \in U_{i_3} \) and \( i_1 \neq i_2 \neq i_3 \). Proceeding in this fashion, we obtain a word \( c_{n+1} \) s.t. each \( c_i, 1 \leq i \leq n \), is a prefix of \( c_{n+1} \). Then \( c_{n+1} a^\omega \) cannot belong to \( L \), since for all \( r = 0,1,2, \ldots \), \( c_{n+1} a^r \) does not belong to any \( U_i, 1 \leq i \leq n \), a contradiction. \( \Box \)
Theorem 3.4  (a) EDPDL $\omega \not\subset$ DCFL $\omega \not\simeq$ EQPDL $\omega = \omega$-KC(SDCF) $\subset \omega$-KC(DCF).

(b) $\omega$-regular languages $\not\subset$ EDPDL $\omega \not\subset$ EQPDL $\omega$.

Proof. Let $L_1$ and $L_2$ be as in the proof of Proposition 2.12.

(a) Clearly, EDPDL $\omega \subset$ DCFL $\omega$. By Proposition 3.3, $L^* \omega$ implies that the inclusion is proper and that DCFL $\omega \not\simeq$ EQPDL $\omega \subset \omega$-KC(DCF). The language $(L_1 \cup L_2)^\omega$ shows that EQPDL $\omega \subset$ DCFL $\omega$.

(b) From $(L_1 \cup L_2)^\omega$ we also have EDPDL $\omega \not\subset$ EQPDL $\omega$; the other relations follow from Proposition 3.3.

□

Corollary 3.5  (a) EDPDL $\omega$ is not closed under union, intersection and complementation. (b) EQPDL $\omega$ is not closed under intersection and complementation.

Proof. (a) $L_1^\omega, L_2^\omega$ in Prop.2.12 show that EDPDL $\omega$ is not closed under union and intersection. As for complementation, let $\Sigma = \{a, b\}$ and $L = (a^* b)^\omega$; then $L \in$ EDPDL $\omega$ but $\Sigma^\omega - L = \Sigma^* a^\omega \not\subseteq$ EDPDL $\omega$ (Prop. 3.3).

(b) The examples in (a) above will also do for EQPDL $\omega$.

□
4. TYPE i RECOGNITION IN DETERMINISTIC \( \omega \)-PDA's

This section is devoted to a study and comparison of the various modes of \( i \)-acceptance in \( \omega \)-DPDA's. After deriving some general results on \( i \)-accepting mappings in Subsection 4.1, attention is focused on the families \( \text{Ai-DPDL}_{\omega} \) for \( i = 1,1',2,2' \). These families are shown to constitute a hierarchy within \( \text{DCFL}_{\omega} \). Moreover, the families \( \text{AL'-DPDL}_{\omega} \) and \( \text{AL-DPDL}_{\omega} \) are characterized with the aid of two new unary operations -"extrapolation" and "non-init", both of which, when applied to finite-string languages, yield \( \omega \)-languages. These operations, introduced in Subsection 4.2, shed light on the nature of the \( l' \)-acceptance mode in deterministic \( \omega \)-automata of any kind.

4.1 Basic Results

We start with two general lemmas concerning the relations among the various types of \( i \)-accepting mappings. In the following, let \( S \) denote an arbitrary finite set.

The first lemma follows directly from Definition 1.8:

**Lemma 4.1.1** Let \( r: \mathbb{N} \rightarrow S \) 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 4.1.2** Let \( r: \mathbb{N} \rightarrow S \) be a mapping and let \( F = \{ F_i \}_{i=1}^{L} \subseteq 2^S \);
(a) There can be constructed a set $S_1$, a set $H \subseteq S_1$ and a mapping $r_1 : N \rightarrow S_1$ s.t. $r$ is $2'$-accepting w.r.t $F$ iff:
1. $r_1$ is not $2$-accepting w.r.t $H$;
2. $r_1$ is $2'$-accepting w.r.t $S_1 - H$.

(b) There can be constructed a set $S_2$, a set $K \subseteq S_2$ and a mapping $r_2 : N \rightarrow S_2$ s.t. $r$ is $1'$-accepting w.r.t $F$ iff:
1. $r_2$ is not $1$-accepting w.r.t. to $K$;
2. $r_2$ is $1'$-accepting w.r.t $S_2 - K$.

Proof. (a) Let $S_1 = S \times \{0,1\}^\ell$, let $H = S \times \{0\}^\ell$ and let $r_1$ be defined as follows: $r_1(1) = (q,0^\ell)$ if $r(1) = q$. In the following, let $v$ and $v'$ denote vectors in $\{0,1\}^\ell$. For $i > 1$, if $r_1(i-1) = (q,v)$ and $r(i) = q'$ then $r_1(i) = (q',v')$, where $v'$ is defined as follows:

(i) if $v' = 1^\ell$, then $v'$ is derived from $0^\ell$ by setting its $j$'th component to 1 for every $1 \leq j \leq \ell$ s.t. $q' \notin F_j$;

(ii) if $v' \neq 1^\ell$, then $v'$ is derived from $v$ by setting its $j$-th component to 1 for every $1 \leq j \leq \ell$ s.t. $q' \notin F_j$.

Clearly $S_1, H$ and $r_1$ satisfy the requirements of the lemma.

(b) Let $S_2 = S \times \{0,1\}^\ell \cup \overline{S}$, where $\overline{S} = \{\overline{s} | s \in S\}$, $K = \overline{S}$ and let $r_2$ be defined as follows: Let $v$ and $v'$ denote vectors in $\{0,1\}^\ell$. Define $r_2(1) = (q,v)$ if $r(1) = q$, where $v$ is derived from $0^\ell$ by setting its $j$-th component to 1 for every $1 \leq j \leq \ell$, s.t. $q \notin F_j$. For $i > 1$, if $r_2(i-1) = (q,v)$ and $r(i) = q'$, then let $r_2(i) = \overline{q'}$ in case $v = 1^\ell$ and otherwise let $r_2(i) = (q',v')$, where $v'$ is derived from $v$ by setting its
j-th component to 1 for every \( 1 \leq j \leq \ell \), s.t. \( q' \notin F_j \). If for \( i > 1 \),
\[
q'(i-1) = q \quad \text{and} \quad q(i) = q',
\]
then define \( q'(i) = q' \).

Clearly, \( q' \) is the required mapping.

From Lemma 4.1.2 we have:

**Proposition 4.1.3** For any \( \omega \)-language \( L \subseteq \Sigma^\omega \);

(a) \( L \in A_1'-DPDL_\omega \iff \Sigma^\omega -L \in A_1' -DPDL_\omega \),

(b) \( L \in A_2'-DPDL_\omega \iff \Sigma^\omega -L \in A_2' -DPDL_\omega \).

**Proof.** Let \( L \) be 1-accepted (2-accepted) by the \( \omega \)-DPDA \((K,\Sigma,\Gamma,\delta,q_0,Z_0,F)\)
then the \( \omega \)-DPDA \( M_1 = (K,\Sigma,\Gamma,\delta,q_0,Z_0,F\cup(\cup F')) \) 1'-accepts (2'-accepts) \( \Sigma^\omega -L \).
If \( L \in A_1' -DPDL_\omega (A_2' -DPDL_\omega) \) then by Lemma 4.1.2, \( \Sigma^\omega -L \in A_1 -DPDL_\omega (A_2 -DPDL_\omega) \).

Recall that a U-\( \omega \)-DPDA is an \( \omega \)-DPDA with a single designated set.

**Proposition 4.1.4** For each \( i = 1,1',2,2' \), every \( \omega \)-DPDA can be replaced by an \( i \)-equivalent U-\( \omega \)-DPDA.

**Proof.** (a) Let \( M = (K,\Sigma,\Gamma,\delta,q_0,Z_0,F) \) be any \( \omega \)-DPDA. The 1-equivalent (2-equivalent) U-\( \omega \)-DPDA will be \( M' = (K,\Sigma,\Gamma,\delta, q_0, Z_0, F'\cup F') \).

(b) Let \( M = (K,\Sigma,\Gamma,\delta,q_0,Z_0,F) \) be an \( \omega \)-DPDA, where \( F' = \{ F' \}_{1}^{\ell} \).

Following Lemma 4.1.2, an \( i \)-equivalent U-\( \omega \)-DPDA can be constructed for \( i=1',2' \).

**4.2 The Operations Ext and Ninit**

We now introduce the operations Extrapolation (Ext) and non-init (Ninit),
which will enable us to characterize 1'-acceptance by deterministic \( \omega \)-automata of any kind.
Definition 4.2.1 For \( L \subseteq \Sigma^* \cup \Sigma^\omega \), define the **extrapolation** of \( L \), \( \text{Ext}(L) \), by \( \text{Ext}(L) = \{ \sigma \in \Sigma^\omega \mid \forall i \geq 0, \sigma/\iota \in \text{Init}(L) \} \). For \( L \subseteq \Sigma^* \), define \( \text{Ninit}(L) = \Sigma^\omega - \Sigma^\omega_L \). For a family of finite-string languages \( \mathcal{L} \), define \( \text{Ninit}(\mathcal{L}) = \{ \text{Ninit}(L) \mid L \in \mathcal{L} \} \).

The following example will clarify the above definitions:

**Example 4.2.1** Let \( L = 0^* 11^* \) over alphabet \( \Sigma = \{0,1\} \); then \( \text{Ext}(L) = 0^\omega \cup 0^* 1^\omega \) and \( \text{Ninit}(L) = 0^\omega \).

The following lemma is easily verified.

**Lemma 4.2.1** For any \( L, L_1, L_2 \subseteq \Sigma^* \cup \Sigma^\omega \):
(a) \( \text{Ext}(\text{Ext}(L)) = \text{Ext}(L) \);
(b) \( \text{Ext}(L) = \text{Ext}(\text{Init}(L)) \);
(c) \( \text{Ext}(L_1 \cup L_2) = \text{Ext}(L_1) \cup \text{Ext}(L_2) \).

**Remark 4.2.1** The following example shows that generally \( \text{Init}(\text{Ext}(L)) \neq \text{Init}(L) \).

Let \( L = \{0^n1 \mid n \geq 1\} \); then \( \text{Init}(L) = \{0^n1^i \mid 0 \leq i \leq n, n \geq 1\} \) but \( \text{Ext}(L) = 0^\omega \) and \( \text{Init}(\text{Ext}(L)) = 0^* \).

**Remark 4.2.2** In [Cho] a new operation on languages, called limit, was introduced, and was used in constructing a simplified proof of the main characterization theorem of \( \omega \)-regular languages. For a language \( L \subseteq \Sigma^* \), \( \lim(L) \) denotes the set of all \( \omega \)-tapes which have an infinite number of initial sections in \( L \). Evidently the limit and extrapolation operations are closely related, as is verified by the following equation: \( \text{Ext}(L) = \lim(\text{Init}(L)) \) for any language \( L \).

The relation between the operations \( \text{Ext} \) and \( \text{Ninit} \) w.r.t. general families of languages is exhibited in the following theorem:

**Theorem 4.2.2** 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. $\mathcal{L} = Ninit(\mathcal{L}_1)$ iff for each
$L \in \mathcal{L}$, $L = Ext(L)$.

**Proof** Let $\mathcal{L}_1$ be a class of finite-string languages for which $\mathcal{L} = Ninit(\mathcal{L}_1)$
and let $L \in \mathcal{L}$. By definition $L \subseteq Ext(L)$. Let $L_1 \in \mathcal{L}_1$ be the language
for which $L = \Sigma^w L_1 \Sigma^w$ and let $\sigma \in Ext(L)$. Since $L_1 \Sigma^* \nsubseteq Init(\Sigma^w L_1 \Sigma^w) \Rightarrow \emptyset$, for all $i \geq 0$, $\sigma/i \notin L_1 \Sigma^*$, thus by definition $\sigma \in \Sigma^w L_1 \Sigma^w$ or $\sigma \in L$.

To prove the other direction, define $\mathcal{L}_1 = \{\Sigma^* Init(L) \mid L \in \mathcal{L}\}$.
Let $L \in \mathcal{L}$; by assumption $L = Ext(L)$. Define $L_1 = \Sigma^* Init(L)$. Since
$Ext(L) = \{\sigma \in \Sigma^w \mid \forall i \geq 0, \sigma/i \notin \Sigma^* Init(L)\}$, $Ext(L) = \Sigma^w L_1 \Sigma^w = Ninit(L_1)$, hence
$L \subseteq Ninit(\mathcal{L}_1)$. Now let $L_1 \in \mathcal{L}_1$ then $L_1 = \Sigma^* Init(L)$ for some $L \in \mathcal{L}$.

**Lemma 4.2.3** (a) For any deterministic CFL $L$, $Ext(L) \in Al'\-DPDL_\omega$;
(b) For any regular language $L$, $Ext(L)$ is an $Al'\-\omega$-regular language.

**Proof.** (a) Let $M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a DPDA which accepts $Init(L)$
([G&G]). Let $\sigma \in Ext(L)$; then for every $i \geq 1$, $\sigma/i \in Init(L)$, i.e.
$\sigma/i: (q_0, Z_0): \frac{\#}{M} (q, \gamma)$ for some $q \in F$; since $M$ is deterministic, there
is a unique run of $M$ on $\sigma$. Define the $\omega$-DPDA $M' = (K, \Sigma, \Gamma, \delta, q_0, Z_0, \{F\})$;
then clearly $T_1(M') = Ext(L)$.
(b) The proof is similar to that of (a).

**Corollary 4.2.4** (a) For any $\omega$-DCFL $L$, $Ext(L) \in Al'\-DPDL_\omega$;
(b) For any $\omega$-regular language $L$, $Ext(L)$ is an $Al'\-\omega$-regular language.

**Proof** (a) By Corollary 2.7, $Init(L)$ is a DCFL. As $Ext(Init(L)) = Ext(L)$,
the result follows from Lemma 4.2.3a above.

(b) Since Init(L) is regular (Ref. [Co&Gol]), the proof follows from Lemma 4.2.3b.

We now obtain characterizations for \( \text{Al-DPDL}_\omega \) and \( \text{Al'}-\text{DPDL}_\omega \).

**Theorem 4.2.5** (a) An \( \omega \)-language \( L \subseteq \Sigma^\omega \) is in \( \text{Al-DPDL}_\omega \) iff \( L \) is of the form \( L = L_1^L \), where \( L_1 \) is a DCFL. (b) \( \text{Al'}-\text{DPDL}_\omega = \text{Ninit}(\text{DCF}) \).

**Proof.** (a) Let \( L_1 \) be the CFL accepted by a DPDA \( M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F) \). Construct a \( \text{U-}\omega \)-DPDA \( M_1 = (K \cup \{q\}, \Sigma, \Gamma, \delta, q_0, Z_0, \{\bar{q}\}) \), where \( \delta_1 \) is defined as follows: \( \forall a \in \Sigma \cup \{\varepsilon\}, \forall Z \in \Gamma, \delta_1(q, a, Z) = \delta(q, a, Z) \) if \( q \in K - F \) and \( \delta(q, a, Z) = (\bar{q}, Z) \) if \( q \in F \). Also let \( \delta_1(q, a, Z) = (\bar{q}, Z) \) for each \( a \in \Sigma \) and \( Z \in \Gamma \). Clearly \( T_1(M_1) = L_1^L \).

As for the other direction, let \( L \) be the \( \omega \)-language \( l \)-accepted by some \( \text{U-}\omega \)-DPDA \( M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F) \). As noted earlier, we assume \( M \) has Property C. If input \( x \in \Sigma^* \) transfers \( M \) from configuration \( (q_0, Z_0) \) to \( (q, y) \) where \( q \in F \), then due to Property C, \( x \in L \). Thus for the DPDA \( M_1 = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) we have \( T(M_1) \subseteq L \).

(b) Follows (a) above and Proposition 4.1.3a.

**Remark 4.2.3** The above characterization of \( \text{Al-DPDL}_\omega \) relies heavily on the assumption that all automata have Property C. Dropping this assumption, we obtain a characterization of \( \text{Al'}-\text{DPDL}_\omega \) as the collection of all \( \omega \)-languages of the form \( L_1^L \subseteq L_2^L \), s.t. there exists a DPDM \( M \) and two sets of final states \( F_1, F_2 \) for which \( L_1 = T(M, F_1) \) and \( L_2 = T(M, F_2) \). To see this, let \( M_1 = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) be a \( \text{U-}\omega \)-DPDA. Define \( L_1 = \{x \in \Sigma^* \mid x: (q_0, Z_0) \xrightarrow{M}(q, y)\} \).
q \in F$ and $L_2 = \{x \in \Sigma^* | M_1, \text{ upon scanning } x, \text{ is blocked or enters an infinite } \\
e$-loop\}. Given $M_1$, a modified DPDA can be constructed, that will accept \ $L_1$ and will enter a new non-final state whenever $M_1$ is blocked or enters an infinite $\varepsilon$-loop ($[G&G], [H&U]$). Hence $L_1$ and $L_2$ are DCFL's of the desired form and clearly $T_\varepsilon(M_1) = L_1 \Sigma^\omega - L_2 \Sigma^\omega$. Now let $L_1 = T(M,F_1)$ and $L_2 = T(M,F_2)$ be two DCFL's where $M$ is a DPDM. Modifying $M$ to be blocked whenever it reaches a state in $F_2$, we get a new DPDM $M_1$ s.t. the \ $U$-$\omega$-DPDA $M' = (M_1,F_1)$ $l'$-accepts $L_1 \Sigma^\omega - L_2 \Sigma^\omega$.

We now show that within the family DCFL$\omega$, the subfamily $A_1'$-DPDL$\omega$ is characterized as the collection of all $\omega$-languages which constitute "fix-points" of the extrapolation operation.

Theorem 4.2.6 An $\omega$-DCFL $L$ is in $A_1'$-DPDL$\omega$ iff $L = \text{Ext}(L)$.

Proof By Theorem 4.2.5, $A_1'$-DPDL$\omega = \text{Ninit}(\text{DCF})$. Let $L \in \text{DCFL}_\omega$.

If $L \in A_1'$-DPDL$\omega$, then $L = \text{Ext}(L)$ (Theorem 4.2.2). Suppose $L = \text{Ext}(L)$.

By Corollary 2.7, $\text{Init}(L)$ is a DCFL, hence by Lemma 4.2.3a, $\text{Ext}(L)$, and therefore also $L$, is in $A_1'$-DPDL$\omega$.

Remark 4.2.4 (a) Theorem 4.2.6 above cannot be further extended, because let $L = \{a^i b^i c^j | i, j \geq 1\} \omega \cup \{a^i b^j a^i | i, j \geq 1\} \omega \cup \{a^i \} \omega \cup \{a^i b^i | i \geq 1\} \omega$. It can be easily verified that $L = \text{Ext}(L)$. $L$ is in EDPDL$\omega$ but is not in DCFL$\omega$ and certainly not in $A_1'$-DPDL$\omega$.

(b) Theorem 4.2.6 characterizes $l'$-acceptance by determinists $\omega$-pushdown automata and cannot be generalized to $l'$-acceptance by non-deterministic
\(\omega\)-pushdown automata. For let \(L = (0^*1)^\omega\); then \(\text{Ext}(L) = (0^*1)^\omega(0^*1)^0^w\).

Since \(L\) is \(\omega\)-regular, by Theorem 1.8, \(L\) is in \(A_1'-\text{PDL}_\omega\); however, \(L \neq \text{Ext}(L)\). Moreover, \(L\) can also be 2-accepted by an \(\omega\)-DPDA; hence the above characterization does not hold for \(A_2\)-DPDL

Theorem 4.2.6 yields some interesting corollaries.

**Corollary 4.2.7** Let \(L \subseteq L^\omega\) be an arbitrary \(\omega\)-language and let \(L'\) be an \(\omega\)-CFL in \(A_1'-\text{DPDL}_\omega\) s.t. \(L \subseteq L' \subseteq \text{Ext}(L)\). Then \(L' = \text{Ext}(L)\).

*Proof.* Since \(L \subseteq L'\), \(\text{Ext}(L) \subseteq \text{Ext}(L') = L'\) (Theorem 4.2.6), hence \(L' = \text{Ext}(L)\).

**Corollary 4.2.8** For any \(\omega\)-DCFL \(L\), \(\text{Ext}(L)\) is the minimal language in \(A_1'-\text{DPDL}_\omega\) containing \(L\).

**Corollary 4.2.9** For any \(\omega\)-language \(L\) in \(A_1'-\text{DPDL}_\omega\), if there exist two finite words \(x\) and \(y\) s.t. \(xy^* \subseteq \text{Init}(L)\), then \(xy^\omega \in L\).

*Proof.* If \(xy^* \subseteq \text{Init}(L)\) then \(xy^\omega \notin M(L)\) By Theorem 4.2.6 \(L = \text{Ext}(L)\), hence \(xy^\omega \in L\).

4.3 The Hierarchy of Families \(A_i\)-DPDL

We next study the inclusion relations among the families \(A_i\)-DPDL, \(i = 1,1',2,2',3\) and compare each one with the family of \(\omega\)-regular languages. The results are summarized in the following diagram. The lines indicate proper inclusion, and families not shown to be related in the diagram are incomparable.
Note the basic difference between this hierarchy and the corresponding one for the non-deterministic \( \omega \)-PDA families (See Section 1, Theorem 1.11).

We now proceed to prove each of the specific relations.

**Theorem 4.3.1**

(a) \( \text{Al-DPDL}_\omega \) and \( \text{Al'}-\text{DPDL}_\omega \) are incommensurate;

(b) \( \text{Al-DPDL}_\omega \) and \( \text{Al'}-\text{DPDL}_\omega \) are each incomparable with the class of \( \omega \)-regular languages;

(c) \( \text{Al-DPDL}_\omega \cup \text{Al'}-\text{DPDL}_\omega \) is properly included in \( \text{A2-DPDL}_\omega \cap \text{A2'}-\text{DPDL}_\omega \).

**Proof** Let \( \Sigma = \{0,1\} \). For (a), let \( L = 0^{\omega} \). Then \( L \notin \text{Al-DPDL}_\omega \) by Theorem 4.2.5, but clearly \( L \in \text{Al'}-\text{DPDL}_\omega \). By Proposition 4.1.3, \( L' = \Sigma^{\omega} - L \in \text{Al-DPDL}_\omega \) but \( L' \notin \text{Al'}-\text{DPDL}_\omega \).

(b) Neither of the \( \omega \)-regular languages \( 1^+0^\omega \) and \( \Sigma^{\omega} - 1^+0^\omega = (0_11^+0_1^+)\Sigma^{\omega} \cup 1^\omega \) is
of the form \( L \omega \), for any \( L \subseteq \Sigma^* \); therefore by Theorem 4.2.5, \( 1^+ \omega \notin A_1'-DPDL \cup A_1''-DPDL \). On the other hand, let \( L_1 = \{0^n1^n|n \geq 1\} \omega \); clearly \( L_1 \in A_1'-DPDL \) and therefore \( \Sigma^\omega - L_1 \notin A_1'-DPDL \), but \( L_1 \) and \( \Sigma^\omega - L_1 \) are not \( \omega \)-regular languages.

(c) One can easily verify that \( A_1-DPDL \subseteq A_2-DPDL \) and \( A_1-DPDL \subseteq A_2'-DPDL \).

Let \( M = (K,\Sigma,\Gamma,\delta,q_0,Z_0,F) \) be any \( U-\omega-DPDA \). Construct \( U-\omega-DPDA \)

\[ M_1 = (K \cup \{\overline{q}\},\Sigma,\Gamma,\delta_1,q_0,Z_0,F), \]

where \( \overline{q} \notin K \) and \( \forall a \in \Sigma \cup \{\epsilon\}, \forall Z \in \Gamma, \delta_1(q,a,Z) = \delta(q,a,Z) \) if \( q \in F \), \( \delta_1(q,\epsilon,Z) = (\overline{q},Z) \) if \( q \notin F \) and \( \forall a \in \Sigma, \delta_1(\overline{q},a,Z) = (\overline{q},Z). \)

Then \( T_2(M_1) = T_2(M_1) = T_1(M) \). The \( \omega \)-languages \( 1^+ \omega \) mentioned in (b) implies that the inclusion asserted above is proper.

In the next theorem we derive a decomposition of each \( \omega \)-languages in \( A_2'-DPDL \), as a union of products of DCFL's and members of \( A_1'-DPDL \).

Theorem 4.3.2 Every \( L \in A_2'-DPDL \) is of the form \( \bigcup_{i=1}^{\ell} L_i L'_i \), where \( \ell \geq 1 \) and for each \( 1 \leq i \leq \ell \), \( L_i \in A_1'-DPDL \) and \( L'_i \) is a DCFL.

The proof of this theorem follows the same lines of the proof of Theorem 3.1, and is presented in full in Appendix A.

Remark 4.3.1 Unlike in the case of \( A_2'-PDL \) (see Theorem 1.7), Theorem 4.3.2 above does not provide a characterization of \( A_2'-DPDL \). The following is an example of a DCFL \( R \) and an \( \omega \)-language \( L' \) in \( A_1'-DPDL \) s.t. \( RL' \), though in the form given in Theorem 4.3.2, is not in \( A_2'-DPDL \). Let \( L_1 = \{a^i b^i a^j| i, j \geq 1\}, L_2 = \{a^i b^i a^j| i, j \geq 1\}, \Sigma = \{a,b,c,d\}, R = \Sigma^* \) and \( L' = \text{Ext}(cL_1d^\omega L_2d^\omega) \). \( L' \) is in \( A_1'-DPDL \) (Corollary 4.2.4a) but \( RL' \) is not even in \( DCFL \), because as in Remark 2.2, \( RL' \cap c^+a^+b^+a^+d^\omega = c(L_1 \cup L_2)d^\omega \) and the latter is not in \( DCFL \).

The above example shows that there exist a regular language \( R \) and an
\(\omega\)-language \(L'\) in \(A_1'\)-DPDL such that \(RL'\) is not in DCFL\(\omega\).

As a consequence of the last theorem we obtain examples of \(\omega\)-regular languages which are not in \(A_2'-\text{DPDL}_\omega\) or \(A_2-\text{DPDL}_\omega\).

**Lemma 4.3.3** (a) \((0^*1)\omega\) is not in \(A_2'-\text{DPDL}_\omega\).

(b) \(\{0,1\}^*0\omega\) is not in \(A_2-\text{DPDL}_\omega\).

**Proof** Let \(\Sigma = \{0,1\}\). Suppose \(L = (0^*1)\omega \in A_2'-\text{DPDL}_\omega\); then \(L = \bigcup_{i=1}^{l} G_i H_i\), where \(l \geq 1\) and for each \(i\), \(H_i \in A_1'-\text{DPDL}_\omega\) and \(G_i\) is a deterministic CFL.

Let \(R = \{(0^n1^n)\omega | n \geq 1\}\). Since \(R\) is an infinite subset of \(L\), there exists \(r, 1 \leq r \leq l\), s.t. \(G_r H_r\) contains an infinite subset \(R_{r,1}\) of \(R\).

Let \(R_{1,1} = \{(0^i1^i)\omega | j \in I\}\), where \(I\) is an infinite set of indices. Define \(B_k = \{0^i1^i | 0 \leq i \leq k-1\} \cup \{\epsilon\}\) for every \(k \in I\). Then, \(H_r\) contains, for each \(k \in I\), a word of the form \(h_k(0^k1)^\omega\), where \(h_k\) is taken from \(B_k\). We have two cases:

**Case 1:** The length of \(\{h_k | k \in I\}\) is unbounded. Then there are \(h_k\)'s of the form \(0^i1\) for unbounded values of \(i\). But then \(0^* \subseteq \text{Init}(H_r)\) and by Theorem 4.2.6, \(0^\omega \in H_r\).

**Case 2:** The length of \(\{h_k | k \in I\}\) is bounded. Then the number of distinct \(h_k\)'s is finite, and for some fixed \(k_0\), \(h_{k_0}(0^k1) \in H_r\) for infinitely many \(k \in I\). But then by Theorem 4.2.6, \(h_{k_0}0^\omega \in H_r\).

Both cases lead to contradiction; hence \(L \notin A_2'-\text{DPDL}_\omega\).

(b) If \(\Sigma^*0^\omega \in A_2-\text{DPDL}_\omega\), then by Proposition 4.1.3, \(\Sigma^\omega \cdot L = (0^*1)\omega \in A_2'-\text{DPDL}_\omega\) which is false by (a).
Theorem 4.3.4  (a) $A_2$-DPDL$_\omega$ and $A_2'$-DPDL$_\omega$ are incommensurate,

(b) $A_2$-DPDL$_\omega$ and $A_2'$-DPDL$_\omega$ are each incomparable with the class of $\omega$-regular languages,

(c) $A_2$-DPDL$_\omega$ $\cup$ $A_2'$-DPDL$_\omega$ is properly included in DCFL$_\omega$.

Proof. (a) Let $\Sigma = \{0,1\}$. By Lemma 4.3.3, $\Sigma^*0^\omega \not\in A_2$-DPDL$_\omega$ but clearly $\Sigma^*0^\omega \in A_2'$-DPDL$_\omega$; moreover, $(0^*)^\omega \notin A_2'$-DPDL$_\omega$ but obviously $(0^*)^\omega \in A_2$-DPDL$_\omega$.

(b) Follows from (a) and Theorem 4.3.1b.

(c) Let $\mathcal{L} = A_2$-DPDL$_\omega$ $\cup$ $A_2'$-DPDL$_\omega$. Clearly $\mathcal{L} \subseteq$ DCFL$_\omega$. Let $L = 1\Sigma^*0^\omega0(0^*)^\omega$.

By Lemma 4.3.3, $0\backslash L \notin A_2'$-DPDL$_\omega$ and $1\backslash L \notin A_2$-DPDL$_\omega$ thus by Lemma 2.8, $L \notin A_2$-DPDL$_\omega$ and $L \notin A_2'$-DPDL$_\omega$. However, $L$ is an $\omega$-regular language and is clearly in DCFL$_\omega$.
5. THE FULL HIERARCHY OF \( \omega \)-PDA LANGUAGE CLASSES

In the preceding sections, the various deterministic \( \omega \)-PDA classes were studied in detail, yielding an interesting hierarchy of deterministic \( \omega \)-CFL families. In this section we establish the position of each family in the above hierarchy within the general hierarchy of non-deterministic \( \omega \)-PDA classes (see Section 1 and also [Co&Gol]). Combining the two hierarchies, we draw the complete inclusion diagram of the \( \omega \)-PDA language families.

First, the position of EDPDL and of EQPDL (see Section 3) in the hierarchy will be established.

As the proofs of some of the inclusion results below are rather standard, they will be omitted here and appear in Appendix A.

**Lemma 5.1** Let \( \Sigma = \{a, b\} \) and let \( L_{ab} = \{ \sigma \in \Sigma^\omega | \forall n = 1, 2, \ldots, \#_a(\sigma/n) \geq \#_b(\sigma/n) \} \); i.e. \( \sigma \in L_{ab} \) iff in every prefix of \( \sigma \), the no. of a's is greater than or equal to the no. of b's (for \( x \in \Sigma^* \), \( \#_c(x) \) denotes the no. of occurrences of letter c in x). Then \( L_{ab} \notin EQPDL_\omega \) but \( L_{ab} \in A_1'-DPDL_\omega \).

**Theorem 5.2**
(a) \( A_1'-DPDL_\omega \nsubseteq EDPDL_\omega \nsubseteq A_2'-DPDL_\omega \).

(b) EDPDL_\omega is incomparable with both \( A_1'-DPDL_\omega \) and \( A_2'-DPDL_\omega \);

(c) EQPDL_\omega is incomparable with each of \( A_1'-DPDL_\omega \), \( A_2'-DPDL_\omega \) and \( A_2-DPDL_\omega \).

We now turn to the non-deterministic \( \omega \)-PDA classes. The inclusion relations among the families \( A_i-PDL_\omega \), \( i = 1, 1', 2, 2' \), \( n\omega-CFL_\omega \) and \( CFL_\omega \), studied in [Co&Gol], are summarized in Section 1 (Theorem 1.i.i).

The next theorem establishes the relations among the above families and their deterministic counterparts.
Theorem 5.3  (a) $\mathcal{A}l'_\omega$ properly includes $\mathcal{A}l'_\omega$-DPDL and is incomparable both with $\mathcal{A}l\omega$-DPDL and with $\omega$-KC(DCF).

(b) Each of $\mathcal{A}2'_\omega$-PDL and $n\mathcal{CFL}_\omega$ properly includes $\mathcal{A}2'_\omega$-DPDL and is incomparable both with EDPDL and with $\omega$-KC(DCF).

Diagram 5.1 below shows the rich hierarchy of $\omega$-CFL families obtained in this paper and in [Co&Go], [Co&Gol]. The lines indicate proper inclusion, whereas the dashed line between $n\mathcal{CFL}_\omega$ and $\mathcal{A}2'_\omega$-DPDL indicates inclusion, not necessarily proper; as mentioned in Section 1, it is not known whether this inclusion is indeed proper. Any two families not shown to be related in this diagram are incommensurate.
6. DECISION PROBLEMS

This section deals with various decidability questions concerning deterministic and non-deterministic $\omega$-PDA languages, as well as questions concerning membership in the $i$-recognizability classes $\text{Ai-\text{DPDL}}_\omega$ and $\text{Ai-PDL}_\omega$. In particular, Theorem 6.2.4 states that for a given $\omega$-DCFL $L$, it is decidable whether $L$ can be $1'$-accepted (1-accepted) by an $\omega$-DPDA. So far, we have been unable to settle the analogous decidability problem for $2'$-acceptance or 2-acceptance. The section terminates with a discussion on the $\omega$-regularity problem for $\omega$-DPDA's.

The notion of an "effectively given" $\omega$-CFL w.r.t a given class of $\omega$-PDA's (or $\omega$-CFG's) is essential here.

Definition 6.1 We say that:

(a) An $\omega$-CFL $L$ is **effectively given** in each of the following cases:

1. It is given as a member of $\omega$-KC(CF);
2. The $\omega$-PDA accepting (2-accepting) $L$ is given;
3. The $\omega$-CFG generating $L$ is given.

(b) An $\omega$-language $L$ in $\text{DCFL}_\omega$ is **effectively given** whenever the $\omega$-DPDA accepting $L$ is given.

(c) An $\omega$-language $L$ in $\text{nl-CFL}_\omega$ is **effectively given** whenever the $\omega$-CFG $\text{nl}$-generating $L$ is given.

(d) An $\omega$-language $L$ in $\text{Ai-PDL}_\omega$ (for $i = 1,1',2'$) is **effectively given** in $\text{Ai-PDL}_\omega$ whenever the $\omega$-PDA $i$-accepting $L$ is given.

(e) An $\omega$-language $L$ in $\text{Ai-DPDL}_\omega$ (for $i = 1,1',2,2'$) is **effectively given** in $\text{Ai-DPDL}_\omega$ whenever the $\omega$-DPDA $i$-accepting $L$ is given.
The following solvability results are from [Co&Go].

**Theorem 6.1** For any \( \omega \)-regular language \( R \) and \( \omega \)-CFL \( L \) effectively given, it is decidable whether:

(a) \( L \) is empty, finite or infinite; (b) \( L \subseteq R \).

### 6.1 Decidability Results Generalized from The Classical Theory

We start with some general undecidability results for \( \omega \)-CFL's, which follow from analogous results for context free languages.

In [Co&Gol] it was shown that for any \( L \subseteq \Sigma^* \) and \( d \notin \Sigma \), \( L_d^{\omega} \) is an \( \omega \)-CFL (\( \omega \)-regular language) iff \( L \) is a context-free (regular) language.

Utilizing this connection, many undecidability results for \( \text{CFL}_\omega \) are obtained directly from the classical theory ([H&U]).

**Theorem 6.1.1** It is undecidable whether:

(a) The intersection of two \( \omega \)-CFL's is: (1) finite; (2) empty; (3) an \( \omega \)-CFL.
(b) For an \( \omega \)-CFL \( L \) and \( \omega \)-regular language \( R \): (1) \( L=R \); (2) \( R \subseteq L \).
(c) An \( \omega \)-CFL \( L \) is \( \omega \)-regular.

**Proof**

(a) For any \( L_1, L_2 \subseteq \Sigma^* \), \( L_1^{d_1^{\omega}} \cap L_2^{d_2^{\omega}} \) is (1) finite ((2) empty);

(b) For any \( L_1, R_1 \subseteq \Sigma^* \), \( L_1^{d_1^{\omega}} = R_1^{d_1^{\omega}} \) (\( R_1^{d_1^{\omega}} \subseteq L_1^{d_1^{\omega}} \)) iff \( L_1 = R_1 (R_1 \subseteq L_1) \).

(c) Follows in a similar way.

In [B&P&S] (see [Gin]) a language \( L_s \) and a family of languages \( \{L(x,y) | x,y \ n\text{-tuples of words in } \{a,b\}^+\} \), both over \( \Sigma = \{a,b\} \), were constructed with the following properties:
(1) $L_s$ and $L(x,y)$ are deterministic CFL's;

(2) $L_s \cap L(x,y) = \emptyset$ iff $L_s \cap L(x,y)$ is a CFL;

(3) It is undecidable for arbitrary $L(x,y)$ whether $L_s \cap L(x,y) = \emptyset$.

Let $\overline{L_s} = \Sigma^* - L_s$ and $\overline{L(x,y)} = \Sigma^* - L(x,y)$, then from [G&G] we have:

(4) It is undecidable whether $\overline{L_s} \cup \overline{L(x,y)}$ is a deterministic CFL.

**Theorem 6.1.2** Given and $\omega$-CFL $L \subseteq \Sigma^\omega$, it is undecidable whether:

(a) $L = \Sigma^\omega$; (b) $\Sigma^\omega - L = \emptyset$; (c) $\Sigma^\omega - L$ is an $\omega$-CFL.

**Proof** $L'(x,y) = L(x,y)d^\omega$ and $L' = Lsd^\omega$, where $d \in \Sigma$, are $\omega$-DCFL's, hence $L = (\Sigma^\omega - L'(x,y)) \cup (\Sigma^\omega - L')$ is an $\omega$-CFL (Proposition 2.2).

(a) $L = \Sigma^\omega \iff L'(x,y) \cap L' = \emptyset \iff L(x,y) \cap L = \emptyset$ which is undecidable.

(b) $\Sigma^\omega - L = \emptyset \iff L = \Sigma^\omega$ which is undecidable by (a).

(c) For the above $L$, $\Sigma^\omega - L = L'(x,y) \cap L'$; $\Sigma^\omega - L$ is an $\omega$-CFL iff $L(x,y) \cap L$ is a CFL, which is undecidable.

For $\omega$-DCFL's we obtain the following decidability results:

**Theorem 6.1.3** Given an $\omega$-DCFL $L$ and an $\omega$-regular language $R$, it is decidable whether:

(a) $L = R$; (b) $R \subseteq L$; (c) $\Sigma^\omega - L = \emptyset$; (d) $\Sigma^\omega - L$ is an $\omega$-CFL.

**Proof** By Theorem 6.1 it is decidable for an $\omega$-CFL $L$ whether $L = \emptyset$.

(a) $L = R \iff L \cap (\Sigma^\omega - R) = \emptyset$ and $(\Sigma^\omega - L) \cap R = \emptyset$. By Propositions 2.2, 2.3 the result follows;

(b) $R \subseteq L \iff R \cap (\Sigma^\omega - L) = \emptyset$;

(c), (d) $\Sigma^\omega - L$ is always an $\omega$-DCFL (Proposition 2.2), hence it is decidable whether $\Sigma^\omega - L = \emptyset$.  

\[ \square \]
By Theorems 2.6, 2.9, for every $L \in L^*$ and $d \in \Sigma$, $Ld^\omega$ is an $\omega$-DCFL iff $L$ is a DCFL. By the arguments in the proof of Theorem 6.1.1 and results in the classical theory ([G&G]) we have:

**Theorem 6.1.4** For arbitrary $\omega$-CFL $L$, $\omega$-DCFL's $L_1$ and $L_2$ and regular language $R$, it is undecidable whether:

(a) $L_1 \cap L_2 = \emptyset$; (b) $L_1 \cap L_2$ is an $\omega$-CFL; (c) $L_1 \cup L_2$ is an $\omega$-DCFL;

(d) $L_1 \subseteq L_2$; (e) $L$ is an $\omega$-DCFL; (f) $RL_1$ is an $\omega$-DCFL.

In the next theorem, $L^\omega$ plays the role of $L^*$ in an analogous undecidability result for DCFL's ([G&G]).

**Theorem 6.1.5** For an arbitrary $\omega$-DCFL $L$, it is undecidable whether $L^\omega$ is an $\omega$-DCFL.

### 6.2 Decision Problems Concerning $\omega$-Recognizability in $\omega$-PDA's

The next undecidability results rely heavily on theorems from [Co&Gol], (see Section 1).

**Proposition 6.2.1** It is undecidable whether:

(a) An effectively given $\omega$-CFL is in $A2'-PDL_\omega$.

(b) An $\omega$-language effectively given in $A2'-PDL_\omega$ is in $Al'-PDL_\omega$.

(c) An $\omega$-language effectively given in $Al'-PDL_\omega$ is $\omega$-regular.

**Proof** (a), (b). As stated in Theorems 1.11, 1.10, for any CFL $L \in L^*$ and a new symbol $d \in \Sigma$, $(Ld)^\omega \in CFL_\omega$ but $(Ld)^\omega \in A2'-PDL_\omega$ only if $L$ is regular. Moreover, we also have that $Ld^\omega \in A2'-PDL_\omega$ but $Ld^\omega \in A1'-PDL_\omega$ only if $L$ is regular. Since it is, in general, undecidable whether a CFL $L$ is
regular, assertions (a), (b) follow.

(c) Let $L_s$ and $L(x,y)$ be as in Section 6.1 above and let $L'(x,y) = L(x,y) \Sigma^\omega$ and $L' = L_s dL^\omega$, where $\Sigma = \{a,b,d\}$. Then $L'(x,y)$ and $L'_s$ are in $A_l$-DPDL $\omega$ (Theorem 4.2.5), and thus $L = (\Sigma^\omega - L'(x,y)) \cup (\Sigma^\omega - L'_s)$ is in $A_l'$-PDL $\omega$ (Proposition 4.1.1). We have $L(x,y) \cap L_s = \emptyset \iff L'(x,y) \cap L'_s = \emptyset \iff L = \Sigma^\omega$. If $L(x,y) \cap L_s \neq \emptyset$, then $L'(x,y) \cap L'_s$ and $L$ too, are not $\omega$-regular. Hence it is undecidable whether $L$ is $\omega$-regular.

The following result concerns the extrapolation operation defined in Section 4.

Proposition 6.2.2 For any effectively given $\omega$-DCFL $L$, there can be constructed an $\omega$-DPDA which $l'$-accepts $\text{Ext}(L)$.

Proof. Let $M$ be the $\omega$-DPDA that accepts $L$. By Lemma 2.5 and Theorem 2.6 one can effectively construct the DPDA $M' = (K,\Sigma,\Gamma,\delta,q_0,Z,F)$ that accepts $\text{Init}(L) = L / \Sigma^\omega$. Then the $U-\omega$-DPDA $M_L = (K,\Sigma,\Gamma,\delta,q_0,Z,F)$ $l'$-accepts $\text{Ext}(L)$. By Corollary 4.2.8 $\text{Ext}(L)$ is the smallest $\omega$-language in $A_l'$-DPDL $\omega$ containing $L$.

Corollary 6.2.3 For any effectively given $\omega$-DCFL $L$, the minimal $\omega$-language in $A_l'$-DPDL $\omega$ containing $L$ can be effectively constructed.

With the aid of the predicting machine defined in Section 2, we obtain the following:

Theorem 6.2.4 For any effectively given $\omega$-DCFL $L$, it is decidable whether

(a) $L \in A_l'$-DPDL $\omega$; (b) $L \in A_l$-DPDL $\omega$. 
Proof. Let $M = (K, \Sigma, \Gamma, \delta, q_o, Z_0, F)$ be the $\omega$-DPDA which accepts $L$.

By Lemma 2.1 one can effectively construct an $\omega$-DPDA accepting $L^{\omega}_E$.

Since by Proposition 4.1.3, $L \in A_{I}^{*}$-DPDL iff $L^{\omega}_E \in \omega$-DPDL, it suffices to show that one can decide whether $L \in A_{I}^{*}$-DPDL.

By Theorem 4.2.6, an $\omega$-DCFL $L$ is in $A_{I}^{*}$-DPDL iff $L = \text{Ext}(L)^{\omega}$.

Since $L \subseteq \text{Ext}(L)$, $L \in A_{I}^{*}$-DPDL iff $\text{Ext}(L) - L = \emptyset$. We shall now show how an $\omega$-DPDA $\tilde{M}$ accepting $\text{Ext}(L) - L$ can be effectively constructed from $M$.

Since the emptiness problem for $\omega$-CFL's is decidable (Theorem 6.1) the result follows.

Following Lemma 2.5, a modified DPDA $M' = (K, \Sigma, \Gamma, \delta, q_o, [q_o, \alpha_0])$, operating as a predicting machine, can be effectively constructed. Here $C$ is the set of maps from $K$ to the set $\{0, 1\}$. If the pushdown store of $M'$ is $[Z_1, \alpha_1] \ldots [Z_i, \alpha_i]$, for $1 \leq j \leq i$ and $q \in K$, $\alpha_j(q) = 1$ iff there exists $\sigma \in \Sigma^\omega$ s.t. $\text{INS}_M(q, \gamma_j, \sigma) \in F$, where $\gamma_j$ is the contents of the pushdown store after $Z_j$ is erased. $\alpha_0$ is the zero function and $\delta'$ is defined as in Lemma 2.5. For every $(q', Z') \in K \times \Gamma$, define the $\omega$-DPDA $M(q', Z') = (K, \Sigma, \Gamma, \delta, q'_0, [q'_0, \alpha'_0], F)$. Now define the set $B \subseteq K \times (\Gamma \times C)$ as follows:

$(q, [Z, \alpha])$ is in $B$ if either (1) $T(M(q, Z)) \neq \emptyset$, or (2) For some $q_1 \in K$ s.t. $\alpha(q_1) = 1$, there exists a word $w_1 \in \Sigma^*$ s.t. $w_1: (q, [Z, \alpha]) \mid_{M'}^{*} (q_1, \epsilon)$

Let $B_1 = K \times (\Gamma \times C) - B$. Define the DPDA $M_1 = (K \cup q_B, \Sigma, \Gamma \times C, \delta_1, q_0, [q_0, Z_0])$, where $q_B$ is a new state and $\delta_1$ is defined as follows: for every $a \in \Sigma$, and $(q, [Z, \alpha]) \in B_1$, $\delta_1(q, a, [Z, \alpha]) = (q_B, Z)$ and $\forall Z \in \Gamma$, $\delta_1(q_B, a, Z) = (q_B, Z)$; otherwise $\delta_1$ is identical to $\delta'$. $(M_1, F)$ is the $\omega$-DPDA that accepts $L$ and $(M_1, \{K\})$ is an $\omega$-DPDA which $1'$-accepts $\text{Ext}(L)$. Thus $\tilde{M} = (M_1, Z^K, F)$ is the desired $\omega$-DPDA accepting $\text{Ext}(L) - L$. 

□
The question, naturally, arises as to whether one can generalize Theorem 6.2.4 above for \( A_2'-\text{DPDL}_\omega \) and \( A_2\text{-DPDL}_\omega \). This problem is so far unresolved.

Open Problem: Is it decidable, for an effectively given \( \omega\text{-DCFL} \) \( L \), whether \( L \in A_2'-\text{DPDL}_\omega \) (\( A_2\text{-DPDL}_\omega \))?

6.3 The \( \omega\)-Regularity Problem for DCFL \( \omega \)

In [St] the decidability of the regularity problem for deterministic PDA languages was established. The proof of the main theorem in the above paper relies heavily on the finiteness of the input tapes, and thus cannot be readily generalized to \( \omega\text{-DPDA}'s \). So far we have been unable to resolve the general \( \omega\)-regularity problem for \( \omega\text{-DPDA} \) languages. However, utilizing the results of Section 4 and the extension operator, we are able to establish the decidability of this problem within the subfamilies \( A_1'-\text{DPDL}_\omega \) and \( A_1\text{-DPDL}_\omega \).

**Theorem 6.3.1** For any \( L \in A_1'-\text{DPDL}_\omega \), \( L \) is \( \omega \)-regular iff \( \text{Init}(L) \) is a regular language.

**Proof.** If \( L \) is an \( \omega \)-regular language then \( \text{Init}(L) \) is regular ([Co&Go]). If \( \text{Init}(L) \) is regular, then \( \text{Ext}(\text{Init}(L)) = \text{Ext}(L) \) is \( \omega \)-regular (Lemma 4.2.3); but \( L = \text{Ext}(L) \) (Theorem 4.2.6) hence the assertion follows.

**Corollary 6.3.2** It is decidable whether an \( \omega \)-language in \( A_1'-\text{DPDL}_\omega \) (\( A_1\text{-DPDL}_\omega \)) is \( \omega \)-regular.

**Proof.** If \( L \in A_1'-\text{DPDL}_\omega \), then by Theorem 6.3.1 above \( L \) is \( \omega \)-regular iff
Init(L) is regular. If \( L \in \text{Al-DPDL}_\omega \), then \( \Sigma^w - L \in \text{Al'}-\text{DPDL}_\omega \) (Proposition 4.1.3) and since \( L \) is \( \omega \)-regular iff \( \Sigma^w - L \) is \( \omega \)-regular, the assertion follows.

Corollary 6.3.2 above can be strengthened by assuming that the member \( L \) of \( \text{Al'}-\text{DPDL}_\omega (\text{Al-DPDL}_\omega) \) is effectively given only by a 3-accepting \( \omega \)-DPDA, i.e. is effectively given only in DCFL\(_\omega\) and not in \( \text{Al'}-\text{DPDL}_\omega (\text{Al-DPDL}_\omega) \).

For \( \text{Al'}-\text{DPDL}_\omega \) this follows directly from the decidability of the regularity problem for finite string DCFL's [St] and from Lemma 2.5 and Corollary 2.7 in Section 2. For \( \text{Al-DPDL}_\omega \) we also need Lemma 2.1 and Proposition 2.2.

Thus we have:

Corollary 6.3.3 Let \( L \) be an \( \omega \)-language effectively given in DCFL\(_\omega\). If \( L \) is also in \( \text{Al'}-\text{DPDL}_\omega (\text{Al-DPDL}_\omega) \), then it is decidable whether \( L \) is \( \omega \)-regular.

Remark 6.3.1 As it turns out, Theorem 6.3.1 does not generalize to the whole family DCFL\(_\omega\), nor even to the subfamily Al-DPDL\(_\omega\), as is shown by the following examples:

(a) Let \( \Sigma = \{a, b\} \) and let \( L_1 = \{x \in \Sigma^+ | \#_a(x) = \#_b(x) \text{ and for } 1 \leq n \leq |x|, \#_a(x/n) \neq \#_b(x/n)\} \) (\( \#_c(x) \) denotes the no. of occurrences of letter \( c \) in \( x \)). Let \( L = L_1^\omega \); then Init(L) = \( \Sigma^* \) and Ext(L) = \( \Sigma^\omega \). Thus while \( L \) is not \( \omega \)-regular, Ext(L) is, and Init(L) is a regular language. Clearly \( L_1 \) is a DCFL, hence by Theorem 4.2.5a, \( L \in \text{Al-DPDL}_\omega \).

(b) Let \( L = \Sigma^* L_{ab} \), where \( \Sigma = \{a, b\} \) and \( L_{ab} = \{\sigma \in \Sigma^\omega | \forall n \geq 1, \#_a(\sigma/n) \geq \#_b(\sigma/n)\} \). Then Ext(L) = \( \Sigma^\omega \) and Init(L) = \( \Sigma^* \). Also here \( L \) is not \( \omega \)-regular but Ext(L) is and Init(L) is a regular language. To show that \( L \in \text{Al'}-\text{DPDL}_\omega \), we shall describe
informally how an \( \omega \)-DPDA \( M \) which 2'-accepts \( L \) is constructed. Basically, \( M \) operates like \( M' \) in the proof of Lemma 5.1 (see Appendix A) with the exception that whenever \( M' \) enters a non-final state, \( M \) immediately afterwards re-enters the initial state of \( M' \) by an \( \varepsilon \)-move, and starts simulating \( M' \) once more.

As for the \( \omega \)-regularity problem of the whole family \( DCFL_\omega \), we note that any regularity test for \( \omega \)-DPDA languages must depend to a great extent on the type of recognition used in the \( \omega \)-DPDA. The \( \omega \)-DPDA \( M_1 = (M,\{q_0,q_1\}) \) from Example 1.3 is an example of an \( \omega \)-DPDA with a single designated set s.t. the \( \omega \)-language 3-accepted (i.e. accepted) by \( M_1 \) is not \( \omega \)-regular, while the \( \omega \)-language 2'-accepted by \( M_1 \) is \( \omega \)-regular (in fact is \( \Sigma^\omega \)).

Open Problem Is it decidable whether, for a given deterministic \( \omega \)-PDA \( M \), \( T(M) \) is \( \omega \)-regular?
APPENDIX A

Proof of Theorem 2.9 Let \( M = (K_M, \Sigma, \Gamma, \delta_M, q_0, Z_0, F) \) be a loop-free DPDA with \( q_0 \notin F_M \) which accepts \( L \), and let \( A = (K_A, \Sigma, \delta_A, p_0, F_A) \) be an \( \omega \)-DFSA which accepts \( R \). Let \( |K_A| = n \), \( \bar{K}_A = K_A \cup \{0\} \), \( K'_A = \{q' \mid q \in K_A\} \), \( K_1 = \bar{K}_A \cup K'_A \), \( 0^{\times i} = (0,0,\ldots,0) \) and \( K_i = K_1 \times K_1 \times \ldots \times K_1 \) \( i \)-times.

Define: \( \tilde{q}_0 = [q_0,0^{\times n}] \) and \( K = K_M \times K_1 \). Now construct an \( \omega \)-DPDA \( M' = (K, \Sigma, \Gamma, \delta, \tilde{q}_0, Z_0, F) \) as follows. The states of \( M' \) will be mostly of the form: \( [q,p(1),\ldots,p(n),0^{\times(n-1)}] \), where \( p(j) \in K, 1 \leq j \leq i, 1 \leq i \leq n \).

\( M' \) starts in state \( [q_0,0^{\times n}] \). In the first component of its states, \( M' \) mimics the operation of \( M \) on its pushdown store. When \( M \) first reaches some final state \( q_1 \), \( M' \) will enter state \( [q_1,p_0,0^{\times(n-1)}] \) and will start imitating \( A \) on its second component, while continuing the simulation of \( M \) on its first component. Whenever \( M \) reaches a final state, \( M' \) will change the leftmost \( 0 \) component in its current state to \( p_0 \), and will start to mimic \( A \) on this component. Thus rule will be carried on with the following exception: Every time two or more of the last \( n \) components become equal, they will be identified with the leftmost one among them, the other ones to its right will be changed back to \( 0 \), and the imitation of \( A \) will proceed on the leftmost component only.

Formally, \( \delta \) is defined as follows:

1. If \( \delta_M(q,\epsilon,Z) = (q_1,\gamma) \) and \( q_1 \notin F_M \), then for every \( p(i) \in \bar{K}_A, 1 \leq i \leq n \),
\( \delta([q,p(1),\ldots,p(n)],\epsilon,Z) = ([q_1,p(1),\ldots,p(n)],\gamma) \);

2. If \( \delta_M(q,\epsilon,Z) = (q_1,\gamma) \) and \( q_1 \in F_M \), then for every state \( [q,p(1)\ldots p(k),0^{\times(n-k)}] \):
(a) if there exists $i, 1 \leq i \leq k$, for which $p_0 = p(i)$, then define

$$\delta([q, \ldots, p(k), 0^{(n-k)}], a, Z) = ([q_1, \ldots, p(k), 0^{(n-k)}], \gamma)$$

(b) if $p_0 \neq p(i)$ for each $1 \leq i \leq k$, then

$$\delta([q, \ldots, p(k), 0^{(n-k)}], a, Z) = ([q_1, \ldots, p(k), p_0, 0^{(n-k-1)}], \gamma)$$ and for

$k = 0$, $\delta([q, 0^n], a, Z) = ([q_1, p_0, 0^{(n-1)}], \gamma)$. 

3. If $\delta_M(q, a, Z) = (q_1, \gamma)$ for $a \in \Sigma$, then for every state

$[q, p(1), \ldots, p(\ell), 0^{(n-\ell)}]$, $\ell > 0$, let $s_i = \delta_A(p(1), a)$, $i = 1, \ldots, \ell$.

(a) If for each $1 \leq i, j \leq n$, $i \neq j$ implies $s_i \neq s_j$, then define:

$$\delta([q, p(1), \ldots, p(\ell), 0^{(n-\ell)}], a, Z) = ([q_1, s_1, \ldots, s_\ell, s_{\ell+1}, 0^{(n-\ell-1)}], \gamma),$$

where $s_{\ell+1} = 0$ in case $q_1 \notin F_M$ or $q_1 \in F_M$ and $p_0 = s_i$ for some $i$, and $s_{\ell+1} = p_0$ in case $q_1 \in F_M$ and $p_0 \neq s_i$ for each $1 \leq i \leq \ell$.

(b) Otherwise, define "collapse rules" as follows:

Let $\{s_{r_1,1}, \ldots, s_{r_1,m_1}, \ldots, s_{r_t,1}, \ldots, s_{r_t,m_t}\}$ be a partition

of $\{s_1, \ldots, s_\ell\}$ into maximal disjoint groups of identical $s_i$'s.

We may assume that $r_{1,i} < r_{2,i} < \ldots < r_{m_i,i}$ for $i = 1, \ldots, t$ and

that $r_{1,1} < r_{1,2} < \ldots < r_{1,t}$.

(i) In case $q_1 \notin F_M$ or if $q_1 \in F_M$ but $p_0 = s_{1,k}$ for some $1 \leq k \leq t$, define:

$$\delta([q, p(1), \ldots, p(\ell), 0^{(n-\ell)}], a, Z) = ([q_1 c_1, \ldots, c_t, 0^{(n-t)}], \gamma),$$

where for $1 \leq i \leq t$, $c_i = s_{r_{1,i}}$ if $r_{1,i} = i$ and $c_i = s_{r_{1,i}} \in K'$

otherwise.
(ii) If \( q_1 \in F_M \) and \( p_o \neq s_{1,i} \) for all \( 1 \leq i \leq t \), define:

\[
\delta([q_1, p_1(p_1), \ldots, p_t(p_t), 0^{(n-\ell)}], a, z) = ([q_1, c_1, \ldots, c_t, p_0, 0^{(n-t-1)}], \gamma)
\]

where \( c_i \)'s are determined as in (i).

The following \( \varepsilon \)-moves will follow every "collapse":

(iii) For every \( \ell = 1, \ldots, n \), \( q \in K_M \) and every \( c_1, \ldots, c_\ell \in K_A \cup K' \) s.t. at least one of the \( c_i \)'s is in \( K'_A \), let

\[
\delta([q, c_1, \ldots, c_\ell, 0^{(n-\ell)}], \varepsilon, z) = ([q, s_1, \ldots, s_\ell, 0^{(n-\ell)}], z),
\]

where \( s_1 = c_i \) in case \( c_i \in K_A \), and \( s_1 = s \in K_A \) if \( c_i = s' \in K'_A \).

It follows from the above construction that \( \sigma \in LR \) iff during the run of \( M' \) on \( \sigma \), for some \( i, 2 \leq i \leq n+1 \), the projection on component \( i \) of the set of states entered infinitely many times by \( M' \) belongs to \( F_A \). For let \( \sigma \in LR \); then \( \sigma = x_1 x \), where \( x_1 \in L, x \in R \). \( x_1 \) transfers \( M \) to a final state and causes \( M' \) to start simulating \( A \), with input \( x \), on its \( j \)-th component, for some \( 2 \leq j \leq n+1 \). Now this simulation phase of \( A \) can either be continued forever on component \( j \), or else, by a "collapse rule" as in 3(b) above, component \( j \) may be identified with some other component \( j' \), where \( j' < j \). Then the above simulation phase of \( A \) is carried on in component \( j' \), either forever or until some new collapse occurs, whereby the contents of component \( j' \) will be shifted to some component \( j'' < j' \), etc. However, after a finite number of such "shifts" of components, the above simulation phase of \( A \) will stay forever on some component \( j_o \), where \( 2 \leq j_o \leq j \), and the set of states appearing infinitely many times on component \( j_o \) must belong to \( F_A \) in order that \( x \in R \).
The use of $K'_A$ in the above collapse rules will make sure that each component will pass through a state in $K'_A$ or $0$ each time it collapses, and thus no words other than those in LR will cause $M'$ to enter, in one of its components, a designated set of states in $F_A$ and stay in this set forever.

Formally, define $F = \{H \subseteq K_M \times F_A^n \mid \exists i = 2, \ldots, n+1, \text{proj}_i(H) \in F_A\}$

where $\text{proj}_i$ is the projection on the $i$-th component.

By the above argument we have

$$T(M') = LR$$

Proof of Theorem 4.3.2 Let $L$ be any $\omega$-language accepted by a $U$-$\omega$-DPDA $M = (K, \Sigma, \Gamma, \delta, q_0, Z, F)$. For any $\omega$-tape $\sigma$ in $L$, $M$ will scan initial portion of $\sigma$ after which it will enter the set of final states $F$ and will never leave it again. Recalling the argument in [Co&Go, Lemma 4.1.14], whenever $M$ makes a move which is not of the form $(q,\varepsilon)$, there is a possibility that the pushdown store contents beneath the top symbol will never be needed later during the rest of the run on $\sigma$. This happens precisely when the length of the pushdown store reaches a minimum. Bearing this in mind, let $B = \{(p,\gamma) \mid p \in F, |\gamma| \geq 1, (p,\gamma) \text{ in the range of } \delta\}$. For every $(p,\gamma)$ in $B$, define $L(p,\gamma)$ to be the language accepted by the DPDA $M(p,\gamma) = (K \cup \{q_F\}, \Sigma, \Gamma, \delta, q_0, Z, \{q_F\})$, where $q_F \notin K$, $\delta(q,\alpha,\beta) = \delta(q,\alpha,\beta)$ if $(p,\gamma) \in \delta(q,\alpha,\beta)$ and $\delta(q_F,\alpha,\beta) = (p,\gamma)$. Clearly $L(p,\gamma)$ is DCFL. For each $(p,\gamma)$ in $B$, define the $\omega$-language $L'(p,\gamma) = T_1(A(p,\gamma))$, where $A(p,\gamma)$ is the $U$-$\omega$-DPDA $A(p,\gamma) = (K \cup \{q_o, q\}, \Sigma, \Gamma \cup \{Z_o\}, \delta, q_o, Z_o, \{q_o\})$ with $q_o \notin K$, $Z_o \notin \Gamma$, $\delta(q_o,\varepsilon, Z_o) = (p,\gamma Z_o)$.

Technion - Computer Science Department - Technical Report CS0056 - 1975
\[ \delta(q,a,Z) = \delta(q,a,Z) \] for each \( a \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma, q \in K \) and 

\[ \delta(q,a,Z) = (q,Z) \quad \forall a \in \Sigma. \] It can be easily verified that 

\[ L = \bigcup_{(p,\gamma) \in B} L(p,\gamma) L'(p,\gamma). \]

Proof of Lemma 5.1 

(a) \( L_{ab} \) is in \( A_1'\)-DPDL. The \( U_\omega\)-DPDA \( M' \) that \( L' \) accepts \( L_{ab} \) will stay in the set of final states so long as the total no. of \( a \)'s is no less than the total no. of \( b \)'s in the prefix of \( \sigma \) scanned so far. When the reading head scans \( a \) \( b \) and the bottom symbol \( X \) is on the top of the store, the automaton will enter some non-final state.

(b) Suppose \( M = (K,\Sigma,\Gamma,\delta,\sigma,X) \) is an \( \omega\)-EQPDA that accepts \( L_{ab} \), where 

\[ Q_0 = \{q^{(i)}\}_{i=1}^{\omega} \] is the set of initial states. Clearly \( \omega \in L_{ab} \), and we may assume without loss of generality that \( \omega \) is accepted by a run that starts in \( q^{(1)} \). \( M \) reaches \( X \) on the store infinitely many times. So there are \( q \in K \) and positive integers \( k, \ell \) s.t. \( k > \ell \) and both \( a \) and \( a \) transfer \( M \) form the initial configuration \( (q^{(1)},X) \) to \( (q,X) \). Now also \( a_{\ell}^{(k)} b_{\ell}^{(k)} \omega \in L_{ab} \). If a run that accepts \( a_{\ell}^{(k)} b_{\ell}^{(k)} \omega \) starts in \( q^{(1)} \), then also \( a_{\ell}^{(k)} b_{\ell}^{(k)} \omega \) would be accepted by a run that starts in \( q^{(1)} \), a contradiction. By the same argument, no \( \omega \)-word in \( L_{ab} \) of the form \( a_{\ell}^{(k)} b_{\ell}^{(k)} \sigma \) can be accepted by a run starting in state \( q^{(1)} \). Therefore, input \( a_{\ell}^{(k)} b_{\ell}^{(k)} \omega \) is accepted by a run starting in some initial state other than \( q^{(1)} \); let this initial state be \( q^{(2)} \). By the same argument as above, there are positive integers \( k, \ell \), \( k > \ell \) and state \( q' \in K \) s.t. both \( (a_{\ell}^{(k)} b_{\ell}^{(k)}) a_{\ell}^{(k)} \) and \( (a_{\ell}^{(k)} b_{\ell}^{(k)}) a_{\ell}^{(k)} \) transfer \( M \) from \( (q^{(2)},X) \) to \( (q',X) \).

Hence a run that accepts a word in \( L_{ab} \) of the form \( a_{\ell}^{(k)} b_{\ell}^{(k)} \omega \) cannot start in \( q^{(1)} \), \( j = 1,2 \). Repeating the above construction \( \lambda-2 \) more times,
\(l\) numbers \(\{k_i\}_{i=0}^{l-1}\) can be found s.t. the \(\omega\)-word \((\prod_{i=0}^{l-1} a^i b^i a^\omega)\) which is in \(L_{ab}\) cannot be accepted by any run that starts in \(Q_0\), thus is not in \(T_e(M)\). This contradicts our assumption and hence \(L_{ab} \notin \text{EQPDL}_\omega\).

**Proof of Theorem 5.2** (a) Let \(M = (K, \Sigma, \Gamma, \delta, Q_0, Z_0, F)\) be a \(U\)-\(\omega\)-DPDA. Construct an \(\omega\)-EDPDA \(M_1 = (K \cup \{q_1, q_2\}, \Gamma \cup \{x, x_1\}, \delta_1, q_1, x, x_1)\), where \(q_1, q_2 \in K\), \(x, x_1 \in \Gamma\), and \(\delta_1\) is defined as follows: \(\delta(q_1, \varepsilon, x) = (q_0, Z_0 x)\); \(\forall z \in \Gamma, \delta(q, a, Z) = \delta(q, a, Z)\) for \(a \in \Sigma \cup \{\varepsilon\}\) and \(q \notin F\); \(\delta(q_1, a, Z) = (q_1, a)\) for \(a \in \Sigma\) and \(q \in F\); \(\delta(q_1, \varepsilon, Z) = (q_1, \varepsilon)\); \(\delta(q_1, \varepsilon, X) = (q_2, X)\) and \(\forall a \in \Sigma\), \(\delta(q_2, a, X) = (q_2, X)\). Clearly \(T_1(M) = T_c(M_1)\). By Theorem 4.2.5, \(0^\omega \in A1\)-DPDL; hence \(A1\)-DPDL \(\subseteq\) EDPDL \(\omega\). To prove EDPDL \(\omega\) \(\subseteq\) \(A2\)-DPDL \(\omega\), given an \(\omega\)-EDPDA \(M\), construct a \(U\)-\(\omega\)-DPDA \(M_1\) that will simulate \(M\), but will enter a new state \(Q_F\) whenever \(M\) reaches \(X\) on its pushdown store; then define \(F = \{Q_F\}\) to obtain \(T_2(M_1) = T_e(M)\).

\(\omega\)-language \(L_{ab}\) from Lemma 5.1 shows that the inclusion is proper.

(b) By Lemma 5.1, \(L_{ab} \in A1\'-DPDL_{\omega} - \text{EQPDL}_\omega\). By Theorem 4.2.6 the \(\omega\)-language \(1^* 0^\omega\) is not in \(A1\'-DPDL_{\omega}\), but is clearly in EDPDL \(\omega\). \(L_{ab}\) also shows that \(A2\'-DPDL_{\omega} \not\subseteq\) EDPDL \(\omega\). On the other hand, \(L = (0^1)\omega \in \text{EDPDL}_\omega\), but by Lemma 4.3.3 \(L\) is not in \(A2\'-DPDL_{\omega}\).

(c) By Lemma 5.1, each of \(A1\'-DPDL_{\omega}\), \(A2\'-DPDL_{\omega}\), and \(A2\'-DPDL_{\omega}\) is not included in EQPDL \(\omega\). Since \(\{a^i b^i a^j \mid i, j \geq 1\} b^\omega\), \(\{a^i b^i c^i \mid i, j \geq 1\} b^\omega\) is in EQPDL \(\omega\) and not in DCFL \(\omega\) (Proposition 2.12), the assertion follows.

**Proof of Theorem 5.3** Clearly \(A1\'-DPDL_{\omega} \subseteq A1\'-PDL_{\omega}\) and \(A2\'-DPDL_{\omega} \subseteq A2\'-PDL_{\omega}\).

The \(\omega\)-regular languages are included in \(A1\'-PDL_{\omega}\) and in \(A2\'-PDL_{\omega}\), hence by Lemma 4.3.3 the above inclusions are proper.
Now let \( L = \{ww^R \mid w \in \Sigma^*\}d^\omega \cup \Sigma^\omega \), where \( d \notin \Sigma \) is a new symbol. 

\( L \in \omega\text{-}KC(DCF) \) (by Theorem 3.1b) and \( L \in A_1\text{-}DPDL_\omega \) (Theorem 4.2.5a) but clearly \( L \in A_1\text{'-PDL}_\omega \subseteq A_2\text{'-PDL}_\omega \). In [Co&Gol] the following assertions were proved: (i) \( L_1 = \{a^n b^n \mid n \geq 1\}^\omega \in A_1\text{'-PDL}_\omega \); (ii) \( L_2 = \{a^n b^n \mid n \geq 1\}^\omega \in n\text{-}CFL_\omega \), and (iii) \( A_1\text{'-PDL}_\omega \) is closed under GSM mapping. It follows that also \( L_3 = \{a^n b^n \mid n \geq 1\}^\omega \) is not in \( A_1\text{'-PDL}_\omega \). However, one can easily verify that \( L_1 \in \omega\text{-}KC(DCF) \), \( L_3 \in A_1\text{-}DPDL_\omega \) and that \( L_2 \in EDPDL_\omega \), which concludes the proof.

\[\Box\]

**FOOTNOTES**

1. For a set \( \Sigma \) and \( n \geq 1 \), \( \Sigma^{\times n} \) denotes the set of \( n \)-tuples over \( \Sigma \);

   for any member \( a \in \Sigma \), let \( a^{\times n} \) denote the \( n \)-tuple \( (a, \ldots, a) \).
REFERENCES


REFERENCES (cont'd)


