A Characterization of Two-Way Deterministic Classes of Languages

A. V. Aho and J. D. Ullman*

Bell Telephone Laboratories, Incorporated, Murray Hill, New Jersey 07974

Received August 18, 1969

It is shown that a class of languages is defined by a class of two-way deterministic balloon automata if and only if that class is closed under marked union, marked Kleene closure and the inverse mappings performed by deterministic GSMs that move two ways on the input. Hence, the context sensitive languages and various time and tape complexity classes are equivalent to classes of two-way deterministic balloon automata.

1. Introduction

There has been much recent work relating closure properties of classes of languages to the structure of classes of automata recognizing the languages. In [HU], classes of balloon automata were defined. A class of balloon automata has an infinite storage of some specified form. The state of the infinite storage can be altered in specified ways, and a finite amount of information can be obtained from storage, also in a specified way. These operations on the infinite storage are under the influence of a finite control. The balloon automaton has also a read-only input tape upon which the word to be recognized appears. Four types of classes of balloon automata have been defined. The types are distinguished by whether the automata are deterministic or nondeterministic, and whether one- or two-way motion of the input head on the input tape is allowed.

Independently of [HU], a concept called an "abstract family of acceptors" (AFA) was defined [GG]. These classes are essentially the one-way nondeterministic classes of balloon automata with certain restrictions removed. It was shown in [GG] that a class of languages is defined by an AFA if and only if that class of languages is a full abstract family of languages (AFL).1 More recently, "abstract families of deterministic

* Current address: Department of Electrical Engineering, Princeton University, Princeton, N.J.

1 A language is a subset of $\Sigma^*$ for some finite alphabet $\Sigma$. A full AFL is a class of languages containing at least one nonempty language and closed under union, concatenation, Kleene closure, homomorphism, inverse homomorphism and intersection with a regular set.
acceptors” (AFDA), a concept corresponding to one-way deterministic balloon automata, were shown to define a class of languages if and only if that class is an “abstract family of deterministic languages” (AFDL).² [C].

In the present paper, we shall give a characterization of the classes of languages defined by classes of two-way deterministic balloon automata. Specifically, these are the classes of languages closed under marked union, marked Kleene closure and inverse two-way (marked) GSM mappings.

While not directly related to the present work, we add that a machine characterization for the full AFLs that are closed under intersection and iterated finite substitution has also been given [R]. These machines, called “abstract families of processors,” differ from balloon automata in that they have a read-write input head.

2. Definitions

We will give a definition of a two-way deterministic balloon automaton that more closely resembles the definition of an AFA than it does the original definition of a balloon automaton in [HU]. The present definition does not restrict the automata to accept only recursively enumerable sets, as did the original.

A two-way deterministic balloon automaton (2DBA) is a 12-tuple

\[ A = (S, K, I, M, f, g, h, q_0, s_0, \$, \$, F), \]

whose components have the following restrictions and interpretations.

1. \( S \) is the set of states of the infinite memory.
2. \( K \) is a finite set, the states of the finite control.
3. \( I \) is the finite input alphabet.
4. \( M \) is the finite information alphabet.
5. \( q_0 \) in \( K \) is the start state.
6. \( s_0 \) in \( S \) is the initial memory state.
7. \( \$ \) and \( \$ \) in \( I \) are the left and right endmarkers, respectively.
8. \( F \subseteq K \) is the set of final states.
9. \( h \) is a function from (possibly a superset of) \( S \) to \( M \), called the information function.

² An abstract family of deterministic languages (AFDL) is a set of languages containing a nonempty language and closed under marked union, marked Kleene closure and inverse marked gsm mappings.
TWO-WAY DETERMINISTIC CLASSES

(10) $g$ is a function from a subset of $K \times I \times M$ to $K \times M \times \{-1, 0, +1\}$, called the finite control function.

(11) $f$ is a function from (a subset of) $M \times S$ to $S$, called the memory control function.

Informally, to make a move the 2DBA first uses the function $h$ to obtain a finite amount of information from the memory. Let $s$ in $S$ be the current memory state and let $h(s) = m$, for some $m$ in $M$. Then, based on the state of the finite control, the input symbol scanned and the symbol $m$ just computed, $g$ determines a new state for the finite control, a motion for $A$'s input head and a piece of information to transmit to the infinite memory. Let $a$, in $I$, be the input symbol scanned and $q$ the current state of the finite control. If $g(q, a, m) = (p, m', d)$, then $p$ becomes the state of the finite control, the input head is moved $d$ symbols right ($d = -1$ thus indicates a move left) and symbol $m'$, in $M$, is transmitted to the memory. Finally, the new state of $A$'s memory is $f(m', s)$.

Formally, a configuration of 2DBA $A$ is triple $(q, w \uparrow x, s)$, where $q$ is in $K$, $s$ in $S$, $w$ is in $I^*$, $x$ in $I^+$ and $\uparrow$ is a new symbol not in any input alphabet. That is, the finite control is in state $q$, the memory in state $s$, $wx$ appears on the input, and the input head scans the leftmost symbol of $x$.

We define the relation $\overset{A}{\rightarrow}$ by

\[
(q, a_1 \cdots a_{i-1} \uparrow a_i \cdots a_n, s) \overset{A}{\rightarrow} (q', a_1 \cdots a_{j-1} \uparrow a_j \cdots a_n, s')
\]

exactly when

1. $h(s) = m$, for some $m$ in $M$,
2. $g(q, a_i, m) = (q', m', d)$, for some $m'$ in $M$ and $d$ in $\{-1, 0, +1\}$ such that $j = i + d$ and $1 \leq j \leq n$,

and

3. $s' = f(m', s)$.

Relation $\overset{A}{\rightarrow}^*$ is the reflexive, transitive closure of $\overset{A}{\rightarrow}$. That is, $Q \overset{A}{\rightarrow}^* Q$ and $Q' \overset{A}{\rightarrow}^* Q'$, if there exists a $Q''$ such that $Q \overset{A}{\rightarrow} Q''$ and $Q'' \overset{A}{\rightarrow} Q'$.

The language accepted by $A$, denoted $T(A)$ is $\{w \in (I - \{\$\})^* \mid (q_0, \uparrow \$w, s_0) \overset{A}{\rightarrow}^* (q, \$w \uparrow s, s) \text{ for some } q \in F \text{ and } s \in S\}$.

The most interesting feature of balloon automata is that one can define subsets thereof which mimic most of the common classes of automata (pushdown, stack, counter, Turing, etc.). These subsets are called classes of balloon automata and have many properties in common. The resulting theorems then immediately apply to all

\footnotesize
3 $I^+$ stands for $II^*$.
models of automata that are covered. The classes of balloon automata are defined by restricting the ways in which the infinite memory can be probed and altered. We need a few definitions in order to give a formal definition of a class.

A closed class of information functions is a set $\mathcal{H}$ of functions with finite range such that if $h_1$ and $h_2$ are in $\mathcal{H}$, then $h$ is in $\mathcal{H}$ if $h$ obeys the following restriction:

If $h(s) \neq h(t)$ then either $h_1(s) \neq h_1(t)$ or $h_2(s) \neq h_2(t)$.

Informally, if two automata are in a class, then there exists a third automaton in the same class which can obtain from the memory whatever information either original automaton obtains (but no more). This restriction differs somewhat from the ideas of [GG] and [C]. The AFA's [GG] and AFDA's [C] differ from our model, which follows [HU] more closely, in the following respects:

1. [GG] and [C] require that acceptance of languages occur with the infinite memory in the initial state, while we do not.
2. [GG] and [C] insist that the information function be able to distinguish the initial state of memory from all other states.

The equivalence of our model and the others in the one way deterministic and non-deterministic cases follows by showing that the corresponding models accept the same classes of languages. No proofs of these results have appeared in print to our knowledge. We expect that if the AFDA were extended in the natural way to have a two-way deterministic input, the results of this paper would go through in essentially the same way for that model.

Let $f(m, s)$ be a control function. For each $m$, define $f_m$ to be the function $f_m(s) = f(m, s)$. Let $f_1$ be the identity function, $f_1(s) = s$ for all $s$. Let $f_0$ be the $s_0$-reset function, $f_0(s) = s_0$ for all $s$.

We now introduce a notion of a class of 2DBA. The class is defined by specifying the set of states which the memories of the members of the class may take and the information functions and control functions which are authorized for members of the class.

A class definer is a 4-tuple $D = (\mathcal{I}, \mathcal{F}, \mathcal{H}, s_0)$ such that

1. $\mathcal{I}$ is an arbitrary set; $s_0$ is in $\mathcal{I}$;
2. $\mathcal{H}$ is a closed class of information functions; each defined for all $s$ in $\mathcal{I}$.
3. $\mathcal{F}$ is a set of functions of one variable, including the identity and $s_0$-reset functions.

The class of 2DBA defined by $D$ is the set of 2DBA

$$A = (S, K, I, M, f, g, h, q_0, s_0, \emptyset, t, F)$$

such that

1. $h$ is in $\mathcal{H}$;
(2) for each \( m \) in \( M, f_m \) is in \( \mathcal{F} \);

(3) \( S \subseteq \mathcal{F} \).

**Example.** Two-way deterministic pushdown automata are defined by the class definer \( D = (\mathcal{L}, \mathcal{F}, \mathcal{H}, Z_0) \), where the following are true:

(1) Let \( \Sigma = \{Z_0, Z_1, \ldots, Z_i, \ldots\} \) be a countable alphabet. \( \mathcal{F} = \Sigma^* \).

(2) \( \mathcal{H} \) consists of functions \( h \) such that for all \( x \) and \( w \) in \( \Sigma^* \) and \( Z \) in \( \Sigma \), \( h(wZ) = h(xZ) \) (i.e., the automaton can read the top pushdown symbol only). Also, for some \( i \geq 0 \), \( h(wZ_j) = h(xZ_k) \) if \( j \geq i \) and \( k \geq i \), (i.e., each automaton uses a finite pushdown alphabet).

(3) \( \mathcal{F} \) consists of the identity and \( Z_0 \)-reset function, the function \( f_E(wZ) = w \) for all \( w \) in \( \Sigma^* \) and \( Z \) in \( \Sigma - \{Z_0\} \) (\( f_E \) erases a symbol, but cannot erase \( Z_0 \), which serves as an endmarker for the pushdown list) and the functions \( f_{Z_i}(w) = wZ_i, i \geq 1 \). (The later functions print symbols on top of the pushdown list.)

A two-way generalized sequential machine (2GSM) is an 8-tuple

\[ G = (K, \Sigma, \Delta, \delta, q_0, \epsilon, \tau, F) \]

such that

(1) \( K, \Sigma \) and \( \Delta \) are finite sets of states, input and output symbols, respectively;

(2) \( q_0 \) in \( K \) is the start state; \( \epsilon \) and \( \tau \) in \( \Sigma \) are left and right endmarkers, respectively;

\( F \subseteq K \) is the set of final states;

(3) \( \delta \) maps a subset of \( K \times \Sigma \) to \( K \times \Delta^* \times \{-1, 0, +1\} \). \( \delta(q, \epsilon) \) is not defined for \( q \) in \( F \), so that output is unique.

A configuration of 2GSM \( G \) is a triple \( (q, w \uparrow x, y) \), where \( q \) is in \( K \), \( w \) in \( \Sigma^* \), \( x \) in \( \Sigma^+ \), \( y \) in \( \Delta^* \) and \( \uparrow \) is a new symbol, not in \( \Sigma \). Define \( \tau_G \) by \( (q, a_1 \ldots a_{i-1} \uparrow a_i \ldots a_n, y) \tau_G \rightarrow (q', a_1 \ldots a_{i-1} \uparrow a_i \ldots a_n, y') \) exactly when \( \delta(q, a_i) = (q', y', d) \), \( j = i + d \) and \( 1 \leq j \leq n \). \( \tau_G^* \) is the reflexive, transitive closure of \( \tau_G \). Let \( w \) be in \( (\Sigma - \{\epsilon, \tau\})^* \). Then \( G(w) = y \) if and only if \( (q_0, w \uparrow \epsilon \omega, \epsilon) \tau_G^* (q, \epsilon \omega \uparrow \tau, y) \), for some \( q \) in \( F \).

\( G^{-1}(y) = \{w \mid G(w) = y\} \). If \( L \) is a language, then \( G(L) = \bigcup_{w \in L} G(w) \) and \( G^{-1}(L) = \bigcup_{w \in L} G^{-1}(w) \).

\( G \) is said to be a (one-way) GSM if \( \delta \) maps \( K \times \Sigma \) to \( K \times \Delta^* \times \{0, +1\} \). \( \delta(q, \epsilon) \) is not defined for \( q \) in \( F \), so that output is unique.

Let \( \mathcal{L} \) be a class of languages, \( L_1 \) and \( L_2 \) in \( \mathcal{L} \). Let \( L_1 \subseteq \Sigma_1^* \), \( L_2 \subseteq \Sigma_2^* \) and let \( c_1 \) and \( c_2 \) not be in \( \Sigma_1 \cup \Sigma_2 \). We say \( \mathcal{L} \) is closed under marked union if for all \( L_1, L_2, c_1 \) and \( c_2 \) as above, \( c_1 L_1 \cup c_2 L_2 \) is in \( \mathcal{L} \). We say \( \mathcal{L} \) is closed under marked * (or marked Kleene closure) if \((c_1 L_1)^* \) is always in \( \mathcal{L} \). We say \( \mathcal{L} \) is closed under inverse 2GSM mappings if \( G^{-1}(L_1) \) is in \( \mathcal{L} \) for all \( L_1 \) in \( \mathcal{L} \).

If \( \mathcal{L} \) is a nonempty class of languages closed under marked union, marked * and inverse 2GSM mappings, we say \( \mathcal{L} \) is an abstract family of two-way deterministic languages (AF2DL).\(^4\)

\(^4\) Those given to pronouncing the unpronounceable can call it an “aftoodle.”
3. Equivalence of AF2DL’s and classes of 2DBA languages

We begin with a simple lemma which says that we can assume the memory state to be the initial state whenever acceptance occurs and that no moves occur after acceptance.

**Lemma 1.** Let $C$ be a class of 2DBA and $L = T(A')$ for some $A'$ in $C$. Then $L = T(A)$ for some $A = (S, K, I, M, f, g, h, q_0, s_0, t, F)$, in $C$, such that if $(q_0, t, s_0) \leadsto_A (q, t, s)$ for $q$ in $F$, then $s = s_0$, and for no configuration $Q$ is $(q, t, s) \leadsto_A Q$ true.

**Proof.** Given $A'$, we can construct $A$ by adding a new state $p$ which is the only final state of $A$. $A$ simulates $A'$ and whenever $A$ reaches $t$ in a final state of $A'$, it applies the reset function and enters state $p$. No moves are allowed in state $p$.

**Theorem 1.** Let $L$ be an AF2DL. Then there is a class $C$ of 2DBA such that $L$ is in $\mathcal{L}$ if and only if $L = T(A)$ for some $A$ in $C$.

**Proof.** The proof is an extension of Chandler’s result [C]. Define $D = (\mathcal{L}, \mathcal{F}, \mathcal{K}, \#(\#))$ to be the class definer given by the following:

1. Let $\Sigma$ be the collection of symbols appearing in the languages of $\mathcal{L}$. Then $\mathcal{L} = (\{\#\} \cup \mathcal{L}) \times \Sigma^*$, where $\#$ is a new symbol.
2. $\mathcal{F}$ consists of all functions whose range is a single element.
3. $\mathcal{K}$ contains the identity and $(\#, e)$-reset functions. For each $L$ in $\mathcal{L}$, $\mathcal{F}$ contains $f_L$ and $f^L$ defined by
   a. $f^L((\#, e)) = (L, e)$, and $f_L$ is undefined otherwise.
   b. $f^L((L, w)) = (\#, e)$ if $w$ is in $L$ and $f_L$ is undefined otherwise.

For each $a$ in $\Sigma$, $\mathcal{F}$ contains the function $f^a$ defined by

$$f^a((L, w)) = (L, wa) \quad \text{for all } L \text{ in } \mathcal{L} \quad \text{and } \quad w \text{ in } \Sigma^*;$$

$f^a$ is undefined otherwise.

Let $C$ be the class of 2DBA defined by $D$. Intuitively, the automata in class $C$ select a language $L$ in $\mathcal{L}$ to recognize. $L$ is stored in the first component of the memory;

---

5 The possibility of this result was first suggested by G. F. Rose.

6 $e$ denotes the empty string.


# indicates that no selection has been made. Then, a string is constructed in the second component in memory and it is eventually tested for membership in \( L \) (using the function \( f^L \)). The remainder of the proof is in three parts.

**Part I.** For all \( L \) in \( \mathcal{L} \), \( L = T(A) \) for some \( A \) in \( C \).

Let \( L \subset \Sigma^* \) for some finite alphabet \( \Sigma \). Let \( \xi \) and \( \eta \) not be in \( \Sigma \), and let \( I = M = \{\xi, \eta\} \) and \( K = \{q_0, q_1\} \). Let \( S = \{\#, L\} \times \Sigma^* \) and \( L = \{g_0, g_1\} \). Define \( f \) by \( f_{\xi} = f^L, f_\eta = f^L \) and \( f_a = f_a \) for all \( a \) in \( \Sigma \). Define \( g \) by

1. \( g(q_0, \xi, \xi) = (q_0, \xi, +1) \);
2. \( g(q_0, a, \xi) = (q_0, a, +1) \), for all \( a \) in \( \Sigma \);
3. \( g(q_0, \eta, \eta) = (q_1, \eta, 0) \).

Finally, let \( A = (S, K, I, M, f, g, h, q_0, (\#, \xi), (\xi, \eta), (q_1)) \). Let \( w \) be in \( \Sigma^* \). By (1), 
\[(q_0, \xi w t, (\xi, \eta)) \xrightarrow{A} (q_0, \xi w t, (L, \eta)) \] By (2), 
\[(q_0, \xi w t, (L, \eta)) \xrightarrow{A} (q_1, \xi w t, (\xi, \eta)) \] if and only if \( w \) is in \( L \). Thus \( T(A) = L \).

We must now show that for all \( A \) in \( C \), \( T(A) \) is in \( \mathcal{L} \).

**Part II.** Let \( \mathcal{L}_1 = \{L_1, L_2, \ldots, L_k\} \) be a finite subset of \( \mathcal{L} \), with \( L_i \subset \Sigma^* \) for \( 1 \leq i \leq k \) and some finite \( \Sigma \). Let \( \xi_1, \xi_2, \ldots, \xi_k \) and \( \xi_0, \xi_1, \ldots, \xi_k \) not be in \( \Sigma \) and let \( L' = \bigcup_{i=1}^k \{\xi_i t_i \} \cup \xi_0 t_0 \cup t_0 \}. \) Define \( L^* = (L')^* \). We will show that \( L^* \) is in \( \mathcal{L} \).

In proof, note that if \( \xi_0 \) is a new symbol, then \( L = \xi_0 \xi^* \bigcup \bigcup_{i=1}^k \xi_i L_i \) is in \( \mathcal{L} \), since \( \mathcal{L} \) is closed under marked union. \( \xi^* \) is easily shown to be in every \( \text{AF2DL} \). There is a 2GSM \( G \) such that \( L' = G^{-1}(L) \). \( G \) works as follows on the words of \( L' \). If \( G \) sees \( \xi_0 \) it emits \( \xi_0 \) and accepts. If \( G \) sees \( \xi_i \), it moves right over elements of \( \Sigma \) until it finds \( t_i \) or \( t_0 \) and then returns to \( \xi_i \). \( G \) has made no output during this operation. If \( \xi_i \) was seen, \( G \) emits \( \xi_i \), and if \( t_0 \) was seen, \( G \) emits \( \xi_0 \). Then \( G \) moves right and emits all symbols in \( \Sigma \) that it sees. When it reaches \( t_i \) or \( t_0 \), it emits nothing but accepts.

Hence, \( L' \) is in \( \mathcal{L} \). By marked \( \ast \), \( (\xi_0 L')^\ast \) is in \( \mathcal{L} \), and by inverse 2GSM again, \( L'' \) is in \( \mathcal{L} \).

**Part III.** Let \( A = (S, K, I, M, f, g, h, q_0, (\#, \xi), (\xi, \eta), (q_1)) \) be in \( C \) and assume \( A \) satisfies Lemma 1. Then \( T(A) \) is in \( \mathcal{L} \).

We construct a 2GSM \( G \) which mimics the finite control and input head action of \( A \).

At each step, \( G \) emits the symbol passed by \( A \) to its memory at that step. Let \( \mathcal{L}_1 = \{L_1, L_2, \ldots, L_k\} \) be the finite set of \( L \) in \( \mathcal{L} \) such that for some \( m \) in \( M \), \( f_m = f^L \) or \( f^L \). Let \( \mathcal{L}_1 \) be the finite set of \( a \) in \( \Sigma \) such that \( f_m = f^a \) for some \( m \) in \( M \). Let \( \xi_i, \xi_0 \) and \( t_0 \) be new symbols, \( 1 \leq i \leq k \), and let \( \Delta = \mathcal{L}_1 \cup \{\xi_i, \ldots, \xi_k, t_0, \ldots, t_k\} \).

Define the 2GSM \( G = (K, I, \Delta, \delta, q_0, (\#, \xi), (\xi, \eta), (q_1)) \) as follows: Let \( \{m_0\} \) be the range of \( h \), let \( q \) be in \( K \), \( a \) in \( I \) and let \( g(q, a, m_0) = (p, m, d) \). Then

1. \( f_m = f_b \) for \( b \) in \( \Sigma \), then \( \delta(q, a) = (p, b, d) \).
(2) If \( f_m = f^{L_i} \), for \( L_i \) in \( \mathcal{L}_1 \), then \( \delta(q, a) = (p, \tau_i, d) \).

(3) If \( f_m = f^{L_i} \), for \( L_i \) in \( \mathcal{L}_1 \), then \( \delta(q, a) = (p, \tau_i, d) \).

(4) If \( f_m \) is the \((\#_1, \epsilon)\)-reset function, then \( \delta(q, a) = (p, \tau_0, d) \).

(5) If \( f_m \) is the identity, then \( \delta(q, a) = (p, \epsilon, d) \). We observe that \( w \) is in \( T(A) \) if and only if

1. \( G(w) \) is defined, and
2. the output of \( G \) with input \( w \), interpreted as a sequence of control functions applied to the memory, is such that a next memory state is always defined, and such that the last state reached is \((\#, \epsilon)\).

The latter condition is equivalent to saying that in state \((\#, \epsilon)\) only the identity, reset and \( f^L \) for some \( L \) in \( \mathcal{L} \) may be applied. In state \((L, w)\) only the identity, reset, \( f^a \) for \( a \) in \( \Sigma \), and \( f^t \) may be applied. In the latter case, \( w \) must be in \( L \).

Define \( L' = \bigcup_{i=1}^{k} \{ q_i L_i t_i \cup \epsilon_i \Sigma_i^* \} \) and \( L^* = (L')^* \). Then \( w \) is in \( T(A) \) if and only if there is some \( y \) in \( \Delta^* \) such that \( G(w) = y \) and \( y \) is in \( L^* \). Thus, \( T(A) = G^{-1}(L^*) \), and \( T(A) \) is in \( \mathcal{L} \) by the result of Part II. Theorem 1 is now proven.

We have now to demonstrate that every class of 2DBA defines AF2DL. Each such class is closed under marked union and marked * [GH]. It is also closed under reversal, intersection with a regular set and inverse (one-way) GSM mapping [HU], properties which we shall need for proving closure under inverse 2GSM mappings. This proof is in two parts. Given a class \( C \) of 2DBA, with \( L = T(A) \) for \( A \) in \( C \), and given a 2GSM \( G \), we construct a 2DBA which recognizes words of \( G^{-1}(L) \) that have added information describing the path of \( G \) on that word. An intuitive picture of the words recognized is given in Fig. 1.

To complete the proof, we will show that from the words of \( G^{-1}(L) \) with path information, it is possible to obtain \( G^{-1}(L) \) itself, using the operations of inverse GSM and reversal. We proceed by first giving a sequence of definitions and lemmas.

Let \( G = (K, \Sigma, \Delta, \delta, q_0, \tau, F) \) be a 2GSM. We say \( G \) is rectified if the set of states \( K \) can be partitioned into two disjoint sets, \( K_R \) (the rightgoing states) and \( K_L \) (the leftgoing states) such that if \( q \) is in \( K_R \), and \( \delta(q, a) = (p, y, d) \), then either \( p \) is in \( K_R \) and \( d = +1 \) or \( p \) is in \( K_L \) and \( d = 0 \). If \( q \) is in \( K_L \), and \( \delta(q, a) = (p, y, d) \),
then either \( p \) is in \( K_L \) and \( d = -1 \) or \( p \) is in \( K_R \) and \( d = 0 \). Also, \( q_0 \) is in \( K_R \). Informally, if \( G \) is rectified, then it stays in rightgoing states for a while, moving right at each step. Then, leaving its head fixed, it changes to a leftgoing state and proceeds to move left for a while, stops, enters a rightgoing state, and so on.

**Lemma 2.** If \( G \) is a 2GSM, then there is a rectified 2GSM \( G' \) such that for all \( w \), \( G(w) = G'(w) \).

**Proof.** This result requires an elementary finite automaton construction, and is left to the reader.

We say the 2GSM \( G = (K, \Sigma, \Delta, \delta, q_0, \delta, F) \) is **single output** if \( \delta(q, a) = (p, y, d) \) for some \( q, p, a \) and \( d \), then \(|y| = 1.7\).

**Lemma 3.** If \( G \) is a 2GSM, then there is a (one-way) GSM \( G_1 \) and a rectified, single output 2GSM \( G_2 \), such that for all languages \( L \), \( G^{-1}(L) = G_2^{-1}(G_1^{-1}(L)) \).

**Proof.** Let \( G = (K, \Sigma, \Delta, \delta, q_0, \delta, F) \). By Lemma 2, assume \( G \) is rectified. Let \( G_2 = (K, \Sigma, \Delta', \delta', q_0, \delta, F) \), where \( \Delta' = \{ [y] \mid \delta(q, a) = (p, y, d) \) for some \( q, p, a \) and \( d \} \), and for all \( q \) in \( K \) and \( a \) in \( \Sigma \), \( \delta'(q, a) = (p, [y], d) \) if and only if \( \delta(q, a) = (p, y, d) \). \( G_1 \) is a one state GSM (homomorphism actually) which upon reading symbol \([y]\) on its input, emits the string \( y \).

Let \( G = (K, \Sigma, \Delta, \delta, q_0, \delta, F) \) be a 2GSM, and let \( w \) be in \((\Sigma - \{ \#, \})^* \), with \( G(w) = y \) for some \( y \) in \( \Delta^* \). Define \( \sigma_{G,w}(i, j) \) or \( \sigma(i, j) \), where \( G \) and \( w \) are understood, to be the state of \( G \) the \( j \)-th time \( G \) visits the \( i \)-th symbol of \( w \). Formally, let \( (q_1, w_1, y_1) \vdash_G (q_2, w_2, y_2) \vdash_G \cdots \vdash_G (q_m, w_m, y_m) \) be the unique sequence of configurations of \( G \) such that \( q_1 = q_0 \), \( y_1 = e \), \( w_1 \) is \( \#w\# \), \( y_m = y \), \( w_m = \#w\# \) and \( q_m \) is in \( F \). Let \( w_k \equiv u_k \mid v_k \), for \( 1 \leq k \leq m \). If \(|u_k| = i - 1 \) and there are exactly \( j - 1 \) values of \( p \), \( 1 \leq p < k \) such that \(|u_p| = i - 1 \), then \( \sigma(i, j) = q_k \). \( \sigma(i, j) \) is undefined otherwise. Note that \( G \) does not loop if it attains a final state at the right endmarker. Hence, \( G(w) \) is not defined if \( G \) enters a loop. Thus, if \( G \) has \( s \) states, \( G(w) \) is only defined if it never visits a symbol of \( \#w\# \) more than \( s \) times. Consequently, \( \sigma(i, j) \) is never defined if \( j > s \).

We here fix our attention on the 2GSM \( G \), as above, with \( s \) states. For \( 0 \leq i \leq s \), let \( \Sigma^{(i)} \) be the set of \( i + 1 \)-tuples \([a, q_1, q_2, \ldots, q_i] \) such that \( a \) is in \( \Sigma \) and \( q_1, q_2, \ldots, q_i \) in \( K \cup \{ e \} \). Note that \( \Sigma^{(0)} \) is isomorphic to \( \Sigma \). Let \( R_G^{(i)} \), or \( R^{(i)} \) where \( G \) is understood, be the set of words \( a_1 a_2 \cdots a_n \), such that

1. \( \alpha_j \) is in \( \Sigma^{(i)} \), for \( 1 \leq j \leq n \).
2. \( \alpha_j = [a_j, q_{j1}, q_{j2}, \ldots, q_{ji}] \).

Then \( a_1 = \# \), \( a_n = \# \) and \( a_2, a_3, \ldots, a_{n-1} \) are in \( \Sigma - \{ \#, \} \). Define \( w = a_2 a_3 \cdots a_{n-1} \).

\( |y| \) is the length of string \( y \).
Then \( q_{jk} = \sigma_{G,w}(j, k) \) for all \( j \) and \( k \), \( 1 \leq j \leq n \), \( 1 \leq k \leq i \), if \( \sigma_{G,w}(j, k) \) is defined, and \( q_{jk} = \epsilon \) otherwise.

Informally, \( R^{(i)} \) is the set of input words for which \( G \) has an output, together with the state of \( G \) the first \( i \) times \( G \) visits each input symbol. Note that \( R^{(i)} \) includes the entire computation of \( G \), and that \( R^{(i)} \) is regular for all \( i \).

Suppose \( \alpha = [a, q_1, q_2, \ldots, q_i] \) is in \( \Sigma^{(i)} \) for some \( i \). Let \( \text{First}(\alpha) = a \). Define \( \text{First}(\alpha_1 \alpha_2 \cdots \alpha_n) = \text{First}(\alpha_1) \text{First}(\alpha_2) \cdots \text{First}(\alpha_n) \). If \( L \) is a language, let \( L^{(i)} \), or \( L^{(i)} \), where \( G \) is understood, be \( \{ w \in R^{(i)} \mid \text{First}(w) = \text{ey}, \text{and } y \text{ is in } G^{-1}(L) \} \). Thus, \( L^{(i)} \) is \( G^{-1}(L) \) with endmarkers and the state of \( G \) the first \( i \) times \( G \) visits each input symbol.

**Lemma 4.** Let \( G \) be a rectified 2GSM with \( s \) states, and let \( L \) be a language. For \( 1 \leq i \leq s \), there exists a GSM \( G_i \) such that either \( L^{(i-1)} = G^{-1}_i(L^{(i)}) \) or \( [L^{(i-1)}] = G^{-1}_i([L^{(i)}]) \).

**Proof.** Suppose that \( G = (K, \Sigma, \Delta, \delta, q_0, \epsilon, t, F) \). If \( w \) is in \( \Sigma^+ \), and \( q \) is in \( K \), define \( \mu(w, q) = p \) if \( G \), started in state \( q \) and reading the rightmost symbol of \( w \) will first move right from the rightmost symbol of \( w \) in state \( p \). If, under those conditions \( G \) never leaves \( w \), then \( \mu(w, q) = \varphi \). A simple construction (concerned with two-way finite automata in [S], for example) shows that a (one-way) GSM can, after reading \( w \) on its input, have \( \mu(w, q) \) for all \( q \) in \( K \), "stored" in its finite control.

We will first discuss the construction of \( G_1 \). \( G_1 \) must map words in \( \Sigma \), or \( \Sigma^{(0)} \), to the proper word in \( \Sigma^{(i)} \). Let the input to \( G_1 \) be \( a_1 a_2 \cdots a_n \). (Note that \( a_1 \) and \( a_n \) are \( G \)'s endmarkers. \( G_1 \) has its own superfluous endmarkers.) \( G_1 \) skips over its left endmarker and upon reading \( a_1 \), emits the symbol \( [a_1, q_0] \) and moves right. Upon reading \( a_j, j > 1 \), if \( G_1 \) has last emitted \( [a_j-1, q] \), \( G_1 \) next emits \( [a_j, p] \) if \( p = \mu(a_1 \cdots a_{j-1}, q) \) is in \( K \). \( G_1 \) halts if \( \mu(a_1 \cdots a_{j-1}, q) = \varphi \).

Next, we consider construction of \( G_i \) for \( i = 3, 5, 7, \ldots \). Note that since \( G \) is rectified, \( \sigma(j, i) \) is a rightgoing state for all odd \( i \) and a leftgoing state for even \( i \). Suppose the input to \( G_i \) is \( \alpha_1 \alpha_2 \cdots \alpha_n \). Upon scanning \( \alpha_j = [a, q_1, q_2, \ldots, q_{i-1}] \), if \( q_{i-1} = \epsilon \), then \( G_i \) emits \( [a, q_1, q_2, \ldots, q_{i-1}, \epsilon] \). (\( G \) does not scan the \( j \)-th symbol \( i \) times.) If \( \delta(q_{i-1}, a) = (p, y, 0) \), \( G_i \) emits \( [a, q_1, q_2, \ldots, q_{i-1}, p] \). (Here, \( G \) scans the \( j \)-th symbol twice in succession.) If \( \delta(q_{i-1}, a) = (p, y, -1) \), let \( w = \text{First}(\alpha_1 \alpha_2 \cdots \alpha_{j-1}) \) and \( p' = \mu(w, p) \). Then \( G_i \) emits \( [a, q_1, q_2, \ldots, q_{i-1}, p'] \). If \( \mu(w, p) = \varphi \), then \( G \) emits \( [a, q_1, q_2, \ldots, q_{i-1}, \epsilon] \). (Here, \( G \) moves left the \( i \)-th time it scans the \( j \)-th symbol. The function \( \mu \) determines the state of \( G \) the next time the \( j \)-th symbol is scanned.)

The construction of \( G_i \) for \( i = 2, 4, 6, \ldots \) is similar, but since GSMS move left to right, \( G_i \) works on \( [L^{(j-1)}] \) and emits strings in \( [L^{(j)}] \).

As a consequence of Lemma 3 and the closure of 2DBA classes under inverse GSM mappings and reversal [HU], we have that \( G^{-1}(L) \) is in class \( C \) whenever \( L^{(i)} \) is in \( C \).
THEOREM 2. Every class $C$ of 2DBA defines an AF2DL.

Proof. Closure under marked union and marked * is given by [GH]. We must show that if $L = T(A)$ for some $A$ in $C$ and $G$ is a 2GSM, then $G^{-1}(L) = T(A')$ for some $A'$ in $C$. By Lemma 3 and the fact that 2DBA classes are closed under inverse GSM mappings, it is sufficient to consider the case in which $G$ is a rectified single output 2GSM. By Lemma 4 and the closure of 2DBA classes under intersection with a regular set [HU], if $G$ has $k$ states, then it suffices to show that $L^{(k)} = T(A') \cap R^{(k)}$ for some $A'$ in $C$. For we will then have that $G^{-1}(L)$ surrounded by an extra pair of endmarkers is defined by a 2DBA in $C$. It is a simple matter to remove these endmarkers using an inverse 2GSM mapping, since the 2GSM has its own endmarkers. The plan of $A'$ is shown in Fig. 2.

![Diagram of 2DBA $A_1$.](image)

$A_1$ will simulate $A$, as though $A$ were working on the output of $G$ (surrounded by endmarkers). However, this string cannot be stored in $A_1$'s control, and we endeavor to have available only the symbol actually scanned by $A$'s input head. The problem is to compute the symbol $A$ scans when $A_1$ simulates a shift of $A$'s head. To assist, $A_1$ has its own input head at the symbol $G$ scanned when it produced the symbol currently scanned by $A$. $A_1$ also records the state of $G$ at this time.

Let $G = (K, \Sigma, \delta, q_0, \epsilon, \overline{F}, F)$. Suppose $A_1$ scans the symbol $[a, q_1, q_2, ..., q_k]$ on its input and believes that $G$ is in state $q_i$. (If the input is in $R^{(k)}$, those of $q_1, ..., q_k$
which are not \( \epsilon \) must be distinct.) If \( A \) moves right on the output of \( G \), \( A_1 \) must determine the configuration of \( G \) one time unit later. That is, \( A_1 \) must decide what motion \( G \) makes on its input and what the new state of \( G \) is. \( \delta(q_i, a) \) gives this information, so \( A_1 \) can move its input head to the symbol now scanned by \( G \) and adjust its notion of the current state of \( G \).

If \( A \) moves left on the output of \( G \), \( A_1 \) must determine the configuration of \( G \) at the previous time unit. If \( i > 1 \) and \( \delta(q_i-1, a) = (p, y, 0) \) for some \( p \) and \( y \), then clearly \( p = q_i \) and before the previous move, \( G \)'s head was where it is now, and \( q_{i-1} \) was its state. Otherwise, if \( q_i \) is a left-going state, \( A_1 \) moves its head right. If the symbol there is \( [b, p_1, p_2, ..., p_k] \), \( A_1 \) determines the unique \( j \) such that \( \delta(p_j, b) = (q_i, y, -1) \). This \( j \) must be unique, else the computation of \( G \) represented on the input of \( A_1 \) is one in which \( G \) enters a loop and has no output. The case in which \( q_i \) is a rightgoing state is analogous.

Observe that the input to \( A_1 \) may be any string in \( [\Sigma^{(k)}]^* \), not only those in \( R^{(k)} \), which represent valid computations of \( G \). So \( T(A_1) \) is a superset of \( L^{(k)} \). However, \( L^{(k)} = T(A_1) \cap R^{(k)} \), and \( R^{(k)} \) is easily seen to be a regular set. Since classes of 2DBA are closed under intersection with a regular set, \( L^{(k)} \) is defined by an automaton in the class, as was to be shown. We will now give a formal construction of \( A_1 \).

Let \( A = (S, P, I, M, f, g, h, p_0, s_0, \sigma, \tau) \), and \( G = (K, \Sigma, \Delta, \delta, q_0, r, \tau, F) \). \( G \) has \( k \) states. We can assume that \( A \subset I - \{ \tau \} \). Let

\[
A_1 = (S, P', \Sigma^{(k)}, M', f', g', h, p_0', s_0, \sigma, \tau, P_F'),
\]

where

1. \( P' = \{ [p, q, Y] \mid p \in P, q \in K \cup \{ \sigma, \tau \} \) and \( Y \in \Delta \cup \{ \varnothing, \emptyset, \epsilon \} \), where \( \varnothing \) and \( \emptyset \) are new symbols;
2. \( p_0' = [p_0, \sigma, \epsilon] \);
3. \( P_F' = \{ [p, \sigma, \epsilon] \mid p \in P_F \} \);
4. \( M' = M \cup \{ \# \} \), where \( \# \) is not in \( M \);
5. \( f_m' = f_m \) and \( f_m' \) is the identity;
6. \( g' \) is defined as follows, for all \( p \in P, q \in K, m \) and \( m' \) in \( M, Y \) in \( \Delta \) and \( \alpha = [a, q_1, ..., q_k] \) in \( \Sigma^{(k)} \).

(i) Suppose \( g(p, Y, m) = (p', m', 0) \). Then \( g'(\alpha, m) = (\alpha, m', 0) \).

\( A_1 \) directly simulates moves of \( A \) in which \( A \)'s input head is fixed. The third component of \( A_1 \)'s state gives the symbol that \( A \) is scanning.

(ii) Suppose \( g(p, Y, m) = (p', m', +1) \). Let \( \alpha = [a, q_1, ..., q_k] \) as defined above, and let \( \delta(q, a) = (q', Y, d) \). Then \( g'(\alpha, m) = (\alpha, m', d) \). In addition, if \( q \) is in \( F \) and \( a = \sigma \), then \( g'(\alpha, m) = (\alpha, m', +1) \).
If $A$ moves right, $A_1$ determines the next move of $G$. $A_1$ updates the state of $G$ in its second component and changes the third component to $\mathcal{F}$, which indicates that a move of $G$ in the forward direction (in time) has just been simulated. On the next move, $A_1$ must determine the output of $G$ for the state $q'$ and the symbol $G$ will scan next. Note that if $A_1$ records $q$ as the state of $G$ and is scanning $a$ on its input, then the second component of $\delta(q, a)$, which is $Y$ will appear in the third component of $A_1$'s state. As a special case, if $A$ moves right from the last symbol of $G$'s output (indicated by $q$ being a final state of $G$, while $a$ is $G$'s right endmarker) then $A_1$ moves to its own right endmarker and will stimulate $A$ scanning $\epsilon$.

(iii) If $\beta$ is in $\Sigma^{(k)}$, $\text{First}(\beta) = b$ and $\delta(q', b) = (q'', Z, d)$, then $g'([p', q', Y], \beta, m) = ([p', q', Z], \#, 0)$.

When the third component of $A_1$'s state is $\mathcal{F}$, $A_1$ picks up the symbol which $G$ will next emit. $A_1$ does not change its memory or move its input head.

(iv) Suppose $g(p, Y, m) = (p', m', -1)$. Let $\alpha = [a, q_1, \ldots, q_k]$ and let $q = q_i$. If $i > 1$ and $\delta(q_{i-1}, a) = (q_i, Z, 0)$, then $g'([p, q, Y], \alpha, m) = ([p', q_{i-1}, Z], m', 0)$. Otherwise, if $q$ is a leftgoing state, let $d = +1$ and let $d = -1$ if $q$ is rightgoing. Then $g'([p, q, Y], \alpha, m) = ([p', q, \mathcal{B}], m', d)$.

If $A$ moves left, $A_1$ determines the previous move of $G$ and establishes the condition of $G$ before that move. In the first case, the previous move of $G$ left the head of $G$ stationary. $A_1$ can determine the previous output of $G$ directly. In the contrary case, $A_1$ determines from whence the head of $G$ came by seeing whether $q$ is left- or right-going. $A_1$ sets the third component of its state to $\mathcal{B}$, to indicate backward motion (in time). Unlike (ii), the state of $G$ is retained, as it is needed on the next move.

(v) Let $\beta = [b, r_1, \ldots, r_k]$ be in $\Sigma^{(k)}$. If $q$ and $d$ are as in (iv), let $i$ be the unique integer such that $\delta(r_i, b) = (q, Z, -d)$ for some $Z$ in $\Delta$. Then $g'([p', q, \mathcal{B}], \beta, m) = ([p', r_i, Z], \#, 0)$.

After a move of type (iv), $A_1$ determines the new state of $G$ to be that state which could have caused $G$ to be in state $q$ on the next move. No change in the memory occurs. Note that if integer $i$ is not unique, then the input to $A_1$ is not in $R^{(k)}$. We do not care what happens, since this input will later be "filtered out" by intersection with $R^{(k)}$.

We now have to consider the special cases in which $A$ (and $A_1$) reaches one of its endmarkers.

(vi) $g'([p, q_0, \mathcal{B}], \epsilon, m) = ([p, \epsilon, \epsilon], \#, 0)$.

If, in rule (iv), $A_1$ is required to find the output symbol of $G$ to the left of the first symbol it emitted, $A_1$ will wind up scanning its left endmarker. $A_1$ goes to a state in which it will simulate $A$ scanning $\epsilon$.

(vii) If $g(p, \epsilon, m) = (p', m', 0)$, then $g'([p, \epsilon, \epsilon], \epsilon, m) = ([p', \epsilon, \epsilon], m', 0)$. If $g(p, \epsilon, m) = (p', m', +1)$, then $g'([p, \epsilon, \epsilon], \epsilon, m) = ([p', q_0, \mathcal{F}], m', +1)$.

$A_1$ simulates $A$ on $\epsilon$. If $A$ moves right, $A_1$ will make use of rule (iii) to compute the first output of $G$. 
(viii) If \( g(p, t, m) = (p', m', 0) \), then \( g'([p, t, e], t, m) = ([p', t, e], m', 0) \). If \( g(p, t, m) = (p', m', -1) \), then \( g'([p, t, e], t, m) = ([p', t, \emptyset], m', -1) \).

\( A_1 \) simulates \( A \) on \( t \). If \( A \) moves left, \( A_1 \) will make use of the next rule to compute the last output symbol of \( G \).

(ix) Let \( \alpha = [t, q_1, \ldots, q_k] \) and let \( i \) be the unique integer such that \( q_i \) is in \( F \). If \( q_{i-1} \) is rightgoing and \( \delta(q_{i-1}, \overline{t}) = (q_i, Y, 0) \), then \( g'([p', t, \emptyset], \alpha, m) = ([p', q_{i-1}, Y], \#, 0) \). Otherwise, \( g'([p', t, \emptyset], \alpha, m) = ([p', q_i, \emptyset], \#, -1) \).

The last output symbol of \( G \) occurs with the move causing it to enter a final state. \( A_1 \) finds the final state and runs \( G \) backwards in time, as in rule (v).

A proof that \( T(A_1) \cap R^{(k)} = L^{(k)} \) is a straightforward result when one bears in mind the intuitive meaning of the various components of \( A_1 \)'s state. We have now shown that the languages defined by a class of 2DBA is closed under inverse 2GSM mappings, and consequently forms an AF2DL.

### IV. Further Consequences and Comments

A first observation is that if we extend the 2DBA to two-way nondeterministic balloon automata in the natural way, as in [HU], then every class of these automata is an AF2DL.

Since every AF2DL is an AFDL[C], as is every full AFL[GG], the classes of languages defined by the four types of balloon automata are related as shown in Fig. 3.

\[ \begin{array}{ccc}
1 \text{DBA} & & \text{INBA} \\
(A) & & (F) \\
2 \text{DBA} & & \\
(B) & & (E) \\
2 \text{NBA} & & \\
(C) & & (D) \\
\end{array} \]

**FIG. 3.** Relation Between Types of Balloon Automata.

There are thus six regions in which AFDL's may be. Region (A) is nonempty. Consider the one-way deterministic pushdown automata, which are not closed under 1 homomorphism or reversal. The context-free languages show that (F) is non-
empty. The two-way nondeterministic pushdown languages are in (C). We conjecture that the two-way deterministic pushdown languages are in (B), since they are probably not closed under (unmarked) union. Moreover, by [GHI], they are not closed under full homomorphism, and hence are not a 1NBA class. (D) contains the regular sets. It is open whether region (E) is empty.

There are many interesting classes of languages which are AF2DL's, including the context sensitive languages.

**Theorem 3.** Let \( f(n) \) be a monotonically nondecreasing function such that for some constant \( k \), \( f(2n) \leq kf(n) \), i.e., \( f(n) \) is at most a polynomial in \( n \). The following are AF2DL's:

1. The languages of tape complexity or nondeterministic tape complexity \( f(n) \).
2. The languages of time complexity \( f(n) \) if there exists \( c > 0 \) such that \( f(n) \geq (1 + c) n \).

We now have characterizations for each of the four types of balloon automata except the 2N. Moreover, the 1D, 1N and 2D classes can each be phrased as the classes of languages closed under marked union, marked * and inverse marked GSM mappings of the corresponding type.

Note that the full AFL's (the 1N classes) are the classes of languages closed under marked union, marked * and inverse one-way nondeterministic GSM mappings. However, our GSM is really the sequential transducer of [GG], not what is normally referred to as a GSM. The models merge in the deterministic case, so no difference was noted until this point.

One would naturally conjecture that the 2N classes are those which are closed under marked union, marked * and inverse two-way nondeterministic GSM mappings, and surely, any class so closed is a 2NBA class. However, the conjecture must be rejected, since then all the 2N classes would be full AFLs, which we know not to be the case. (The 2N pushdown automaton languages [GHI] is one of several known counterexamples.)

**References**


