bounded transient, automaton, finite, state transition function, minimum state, partition, transient, algorithm

BOUNDDED—TRANSIENT AUTOMATA

BY

S. Winograd
BOUNDDED-TRANSIENT AUTOMATA

S. Winograd

Mathematical Sciences Department
International Business Machines Corporation
Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT

A bounded-transient finite automaton is an automaton for which a single change of an input will not affect the output "far away." This paper investigates the class of events which bounded-transient automata can compute and shows that this class of events is a generalization of the class of definite events. A method of determining whether or not a finite automaton is bounded-transient is described, and the connection between the results of this paper and Kilmer's results is indicated.

I. INTRODUCTION

In a paper, "Transient Behavior in Iterative Combinational Switching Networks," W. L. Kilmer investigated bounded-transient, one-dimensional, unidirectional iterative networks. Since these networks are closely related to finite automata, we will investigate the finite automata which are bounded-transient, and in particular we will attempt to define the class of events which bounded-transient finite automata compute.

II. DEFINITION

A bounded-transient finite automaton is an automaton for which a single change of an input will not affect the output "far away." We will assume, for the sake of simplicity, that the automaton has only two output symbols--0, 1. The generalization of the results to the case of automata with finite output alphabet is trivial. In this context we define an automaton as a 5-tuple \((X, S, f, s_0, g)\) where

- \(X = \{x, y, z, \ldots\}\) is the finite input alphabet,
- \(S = \{s_1, s_2, \ldots\}\) is the finite set of internal states,
- \(f: S \times X \to S\) is the state transition function, \(s_0\) is the initial state, and \(g: S \times X \to \{0, 1\}\) is the output function.

In this paper we will represent a string of inputs by \(T\) and the length of the string by \(L(T)\). The state transition function, \(f(s, x)\), will be written \((s, x)\), and \((s, T)\) will denote the generalized state transition function, defined inductively by

\[
(s, xT) = ((s, x), T)
\]

It is clear from the definition of \((s, T)\) that for any two strings of inputs \(T_1\) and \(T_2\), \((s, T_1 T_2) = (s, T_1 T_2)\). We will also assume that \(S = \{s_1, s_2, \ldots\}\) includes only those states which can be reached from \(s_0\), i.e., \(s_0 \Rightarrow \exists T\) such that \((s_0, T) = s\).

Definition: A finite automaton is bounded-transient if and only if \(\exists n\) such that \(\forall s \in S, \forall x, y \in X, \forall T[L(T) \geq n \Rightarrow (s, xT) = (s, yT)]\).

III. EXAMPLE

Theorem 1: For every event \(E\) which can be expressed in the form

\[
E = F_0 \cup \bigcup_{i=1}^{j} A^{i-1}(A^i F_1)
\]

for some \(j\) and some \((F_0, F_1, \ldots, F_j)\) can be computed by a bounded-transient automaton, where \(F_1(i = 0, 1, \ldots, j)\) is a finite event (a finite set of finite tables) and \(A = \bigcup_{x \in X} x\) designates the finite
event "any single input."

**Proof:** Consider the minimum state automaton which computes \( E \) (in this minimum state machine \( s_i \neq s_j \) if and only if \( \exists T \) such that \( g(s_i, T) \neq g(s_j, T) \), where \( g(s, T) \) is the generalized output function). Let \( \{ t_{ik} \} = \{ t_{11}, t_{12}, \ldots \} \) be the finite set of tables of \( F_i \), and \( L(t_{ik}) \) the length of the table \( t_{ik} \). A string of inputs \( T \) is recognizable by the automaton (is in the event \( E \)) if and only if there exists a \( t_{ik} (i \neq 0) \) such that \( L(T) \geq L(t_{ik}) \) and \( L(T) - L(t_{ik}) = i - 1 \mod j \) and \( \forall T \), the last \( L(t_{ik}) \) inputs of \( T \) are \( t_{ik} \), or \( T = t_{ok} \) for some \( k \). Let \( n = \max \max L(t_{ik}) \); then \( \forall T \), the last \( L(t_{ik}) \) inputs of \( T \) are \( t_{ik} \), or \( T = t_{ok} \) for some \( k \). Let \( n = \max \max L(t_{ik}) \); then \( \forall T \), the last \( L(t_{ik}) \) inputs of \( T \) are \( t_{ik} \), or \( T = t_{ok} \) for some \( k \).

**IV. MAIN THEOREMS**

We will denote the class of regular events which can be expressed in the form (1) by \( C \). In the remainder of this paper we will show that \( C \) is also the class of regular events which can be computed by bounded-transient automata.

**Definition:** Let \( M \) be a finite automaton. Two states \( s_i, s_j \in S \) are \( m \)-equivalent (denoted by \( s_i \sim_m s_j \)) if and only if \( \forall T \) \( [L(T) = m \Rightarrow (s_i, T) = (s_j, T)] \). \( (m = 1, 2, \ldots) \)

This is an equivalent relation because:

(i) \( s_i \sim_m s_i \)

(ii) \( s_i \sim_m s_j \Rightarrow s_j \sim_m s_i \)

(iii) \( s_i \sim_m s_j \), \( s_j \sim_m s_k \) \( \Rightarrow s_i \sim_m s_k \).

Each of the \( m \)-equivalent relations induces a partition \( P^{m} \) on \( S \) which partitions it into subsets \( P^{m}_{a}, P^{m}_{b}, \ldots, P^{m}_{i} \). We say that \( P^{a} \preceq P^{b} \) if and only if \( P^{a} \) refines \( P^{b} \).

**Lemma I:** Let \( P^{m} \) be the partition induced by the \( m \)-equivalent relation. Then:

(i) \( s_i \sim^{i+1} s_j \Leftrightarrow \forall x \in X \left[ (s_i, x)^i \sim (s_j, x) \right] \)

(ii) \( P^1 \supset P^2 \supset \cdots \supset P^i \supset P^{i+1} \supset \cdots \)

(iii) \( P^i = P^{i+1} \Rightarrow P^{i+1} = P^{i+2} \).

**Proof:**

(i) \( s_k \sim s_j \Leftrightarrow \forall x \in X \left[ VT[L(T) = i \Rightarrow (s_k, x) \sim (s_j, x)] \right] \)

(ii) \( s_k \sim s_j \Rightarrow VT[L(T) = i \Rightarrow (s_k, T) = (s_j, T)] \Rightarrow VT, \forall x \in X \left[ L(T) = i \Rightarrow (s_k, Tx) = (s_j, Tx) \right] \Rightarrow VT \left[ L(T) = i + 1 \Rightarrow (s_k, T) = (s_j, T) \right] \Rightarrow i + 1 \)

(iii) \( s_k \sim s_j \Rightarrow \forall x \in X \left[ \left[ (s_k, x)^{i+1} \sim (s_j, x) \right] \right] \Rightarrow \forall x \in X \left[ (s_k, x)^i \sim (s_j, x) \right] \Rightarrow s_k \sim s_j \Rightarrow P^{i+2} \supset P^{i+1} \). But by (ii) \( P^{i+1} \supset P^{i+2} \) and
therefore $P_{i+1} = P_{i+2}$.

Since the number of states in $S$ is finite, Lemma 1 guarantees the existence of a minimal (crudest) partition. Let $r$ be the smallest integer such that $P_r$ is the minimal partition. (Note that $r$ cannot exceed the number of states in $S$.)

Theorem 2: Let $M$ be a finite automaton. Then $V s_i, s_j \in S, \forall x, y \in X \{s_i \sim s_j \Rightarrow (s_i, x) \sim (s_j, y)\}$ if and only if $M$ is bounded-transient.

Proof: Let $M$ be a bounded-transient automaton, and then $V s_i, s_j \in S, \forall x, y \in X \{s_i \sim s_j \Rightarrow (s_i, x) \sim (s_j, y)\} \Rightarrow \forall T, L(T) = r + n [(s_i, xT) = (s_{i+n}, xT)] \Rightarrow (s_i, x) \sim (s_j, y) \Rightarrow (s_i, x) \sim (s_j, y)]$. (Here $n$ is the same number used in the definition of a bounded-transient automaton.) To prove the reverse implication note that $V s \in S \{s \sim s\} \Rightarrow V s \in S, \forall x, y \in X, \forall m \geq 0 [(s, x) \sim (s, y)] \Rightarrow V s \in S, \forall x, y \in X, \forall T [L(T) \geq r \Rightarrow (s, xT) = (s, yT)] \Rightarrow M$ is bounded-transient.

Theorem 2 says that a necessary and sufficient condition for a finite automaton to be bounded transient is that regardless of what state $s_i \in P_r$ the automaton is in, and regardless of what input arrives, the next state of the automaton will be in $P_r$, where $P_r$ depends only on $P_r$. In this case we say that $P_r$ precedes $P_r$. Theorem 2 thus gives us an algorithm for deciding whether or not an automaton is bounded-transient. The algorithm is described in the appendix.

Lemma 2: Let $A$ be a bounded-transient finite automaton, then for all $i$, $P_i$ has only one equivalence class preceding it.

Proof: Assume that for some $i$, $P_i$ has two equivalence classes, $P_j$ and $P_k$, preceding it. Then $V s_j \in P_j, \forall s_k \in P_k, \forall x \in X [(s_j, x) \sim (s_k, x)] \Rightarrow V s_j \in P_j, \forall s_k \in P_k [s_k \sim s_j] \Rightarrow P_r = P_r$.

The transition from one equivalence class to another, then, is as shown in Fig. 1.

![Fig. 1](image)

Theorem 3: The class of regular events which can be computed by bounded-transient automata is $C$.

Proof: Theorem 1 proved that $C$ is a subclass of the class of regular events which can be computed by a bounded-transient automaton. Therefore it is left to show that any event which is computed by a bounded-transient automaton can be expressed in the form (1). Let $M$ be a bounded-transient automaton, and let $j$ be the number of equivalence classes in $P_r$. Let the numbering of $P_r$ be such that the initial state $s_0 \in P_r$ and the transition from one equivalence class to another is $P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_j \rightarrow P_r$. Define the (finite) event $F_i (i = 1, 2, \ldots, j)$ to be the set of tables (strings of inputs) $Tx$ such that $L(T) = r$ and $g(s_i, Tx) = 1$ for some $s_i \in P_r$ (and therefore for all $s_i \in P_r$). Define $F_0$ to be the (finite) event which is computed...
by $M$ at times $t = 1, 2, \ldots, r$, then the (regular) event which $M$ computes is $F_0 \cup \bigcup_{i=1}^{r} A_i^{-1}(A_i)^r F_i$ which is of the form (1).

Q.E.D

As a final point, notice that the equivalent classes $P_r^r$ are the phases in Kilmer's paper and that therefore his results hold for connected automata, not only for strongly connected ones.

Appendix

THE ALGORITHM

The algorithm for determining whether a given finite automaton $M$ is bounded-transient consists of two phases: Phase 1--Find the minimal (crudest) partition $P^r$. Phase 2--Determine whether the condition of Theorem 2 is satisfied.

Phase 1 proceeds by finding $P^1, P^2,$ and so on until $P^r$ is reached. This is done in three steps: (1) Describe the state-transition function of $M$ by a matrix in which the rows designate the states and the columns designate the inputs; the entry corresponding to state $s_i$ and input $x$ is the state $(s_i, x)$. (2) Test whether any two rows are identical. If the row corresponding to $s_i$ is identical to the row corresponding to $s_j$ (we assume $i < j$), merge the two rows and designate the merged row by $s_t$; and replace every appearance of $s_i$ by $s_t$. (3) Repeat step 2 on the new matrix until no two rows can be merged. Lemma 1 guarantees that the rows of the matrix $N$ thus obtained correspond to the equivalent classes of $P^r$.

Theorem 2 says that $M$ is bounded-transient if and only if all the entries in each row are the same.

Example: Consider the automaton shown in Fig. 2.

We notice next that rows $s_1, s_2, s_3$ can be merged to yield the matrix of Table 3. We finally merge $s_0$ and $s_4$ to obtain Table 4.

Table 1 shows the state transition matrix. The rows $s_5$ and $s_6$ can be merged, and the result of the merge is shown in Table 2.

Table 1

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_5$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$s_5$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_6$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$s_6$</td>
<td>$s_4$</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_5$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_6$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$s_6$</td>
<td>$s_4$</td>
</tr>
</tbody>
</table>

Table 3

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_5$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$s_4$</td>
</tr>
</tbody>
</table>

Table 4

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_5$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$s_0$</td>
</tr>
</tbody>
</table>

Since no two rows can be merged, phase 1 is completed. The equivalent classes of $P^r$ are: $\{s_0, s_4\}$ (the states merged to form row $s_0$ in Table 4), $\{s_1, s_2, s_3\}$ (the rows merged to form row $s_1$), and $\{s_5, s_6\}$ (the merged rows to form row $s_5$). In phase 2 we notice that all the entries in row $s_0$ are $s_1$, all the entries in row $s_1$ are $s_5$ and all the entries in row $s_5$ are $s_0$. Therefore the automaton shown in Figure 2 is bounded-transient.

ACKNOWLEDGMENT

The author wishes to thank Dr. C. C. Elgot and Jorge Mezei for their many helpful comments.

REFERENCE