Cascade Synthesis of Finite-State Machines

H. P. Zeiger

University of Colorado, Boulder, Colorado 80302

One can construct any finite-state machine as a cascade interconnection of machines whose inputs either permute the states or reset them all to one state. Each permutation group needed in the construction is a homomorphic image of a group generated by the action of a set of input sequences on a state subset of the original machine. Proofs of these facts will be given and their application to the Krohn-Rhodes theory described.

LIST OF SYMBOLS

\( A, B, C, D, \) States of finite state machines.
\( E, X, Y, Z, \) Finite state machines
\( 0, 1, 2, 3, \) Cover (see p. 421)
\( p, q, r \) Identity map of the set \( A \)
\( I^* \) Free semigroup generated by the set \( I \)
\( S \) Semigroup of transformations
\( \tilde{S} \) Extension of \( S \) defined on page 427
\( Q_M \) State set of the machine \( M \)
\( I_M \) Input set of the machine \( M \)
\( S_M \) Semigroup of the machine \( M \)
\( x_M \) State transformation produced by the input string \( x \)
\( Z_{M,x} \) Output function for input \( x \)
\( P, R \) Subsets of a state set
\( \cup \) Union
\( \subseteq \) Includes
\( \in \) Belongs to
\( \max C \) Set of all elements of \( C \) that are maximal with respect to set inclusion

419
<table>
<thead>
<tr>
<th>Input</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present State</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

**FIG. 1. A unit delay**

<table>
<thead>
<tr>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>A</td>
</tr>
</tbody>
</table>

**FIG. 2. Permutation of the state subset \{AB\} produced by the input sequence 01.**

**K → L**

Set of all machines that can be built as a series composition of \( K \) followed by \( L \)

\*\*\*\*

INTRODUCTION

We devote Parts I and II of this report to proving the result stated in the summary. Part I gives an intuitive formulation of the ideas of the proof; Part II gives the proof itself; Part III gives the connection with the theorems of Rhodes and Krohn.

A unit delay is a finite-state machine each of whose inputs resets all states to one state, as shown in Fig. 1. It is not, in general, possible to construct every finite-state machine as a cascade connection of unit delays. Such a construction is prevented by the presence of state sub-
PART I. THE IDEAS OF THE CONSTRUCTION

To construct an arbitrary machine as a cascade connection of unit delays, it would (by iteration) suffice to show how to construct the machine in the form shown in Fig. 3, where \( L \) is simpler than the original machine. Let us try to construct, in this form, the machine \( M \) whose state table is shown in Fig. 4. Each state of \( N \) has two coordinates: the state, 0 or 1, of the delay, and the state of \( L \). If the state of the delay is 0, then the previous input was 0; the present state of \( M \) is \( A, B, \) or \( D \). Therefore \( N \) must have at least three states with first coordinate 0, one corresponding to \( A \), one to \( B \), and one to \( D \). Similarly, \( N \) must have at least three states with first coordinate 1, one corresponding to \( B \), one to \( C \), and one to \( E \). Thus \( L \) must have at least three states; call them \( X, Y, \) and \( Z \), and we need a mapping from the states of \( N \) onto the states of \( M \) satisfying the above constraints. One is shown in Fig. 5. From Figs. 4 and 5 we now construct the state table for \( N \). The first-coordinate entries are dictated by the operation of the delay and
State of Delay

FIG. 5. A state-assignment mapping from states of N to states of M

<table>
<thead>
<tr>
<th>State of L</th>
<th>Z</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>


0  1

FIG. 6. First-coordinate state transitions

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X</td>
<td>0</td>
</tr>
<tr>
<td>0Y</td>
<td>0</td>
</tr>
<tr>
<td>0Z</td>
<td>0</td>
</tr>
<tr>
<td>1X</td>
<td>0</td>
</tr>
<tr>
<td>1Y</td>
<td>0</td>
</tr>
<tr>
<td>1Z</td>
<td>0</td>
</tr>
</tbody>
</table>

FIG. 7. State table for N

are shown in Fig. 6. The second-coordinate entries are now dictated by Figs. 4 and 5 to be as shown in Fig. 7. As a reminder, the corresponding states of M are given in parentheses. Thus the machine L of Fig. 3 is described by the state table of Fig. 8, which is simpler than that of M, at least in number of states.

This is the basic construction. Before we make further use of it, we must get rid of its two worst features:

(1) For L to be smaller than M it was necessary that each column of M's state table be a proper subset of the state set.
(2) Not enough constraints were available precisely to determine the mapping of Fig. 5. It is possible for the choice of this mapping unduly to influence later steps in the iteration. To dispose of the first objection, suppose we wish to construct, in the form shown in Fig. 9, the machine $M$ of Fig. 10.

As in the previous construction, we let input 1 reset $K$ to some state, say 1. Since states of $N$ with first coordinate 1 correspond to states $A$ and $B$ of $M$, and since under input 0 $A$ and $B$ go to $C$ and $A$, the machine $K$ needs a new state, say 2, to correspond to $C$ and $A$. Similarly it needs another state, say 3, to correspond to $B$ and $C$. Thus an appropriate state assignment mapping from the states of $N$ to those of $M$ is the one shown in Fig. 11. Next, a state table for $N$ is constructed as before, except that now the first-coordinate entries are fixed by the way the state subsets $AB$, $AC$, $BC$ move under $M$'s inputs (see Fig. 12). Observe that the machine $K$, factored out on the left, is no longer a
unit delay, for in addition to the reset input 1, it also has a permutation input 0. We now take it for granted that this factoring out of a permutation-reset machine can always be done and bend our efforts to iterating the process. Of course, we could iterate the process as it stands and thus realize any machine as a cascade of permutation-reset machines, but this would give only half the desired theorem, since we would have no control over which permutation groups arose in the construction. This control is lost through the arbitrary assignment of mappings like those shown in Figs. 5 and 11.

If we refuse to make such an assignment, the iteration problem takes the form shown in Fig. 13. How can we proceed from Fig. 13(a) to Fig. 13(b) without ever specifying the unlabeled machines?

Observe that in the previous example, the machine $K$ gives partial state data on the machine $M$ in the following sense: With each state of $K$ is associated a set of states of $M$ so that (1) each state of $M$ belongs to at least one of these sets, and (2) the image under any state transformation of one of these sets is included in another.

We now propose to go from 13(a) to 13(b) by requiring that the machine $N$ improve the partial state data given by the machine $K$. That is, each state subset of the cover associated with $N$ should be included in some state subset of the cover associated with $K$, and at least one of the set inclusions should be proper. More briefly, we require that the
cover induced by $N$ should be a proper refinement of that induced by $K$. For example, consider the machine $M$ whose state table is shown in Fig. 14.

Letting $K$ be a unit delay, we get the cover $C_1 = \{\{ABC\}, \{CDE\}\}$. Next we search for a refinement of this cover to associate with $N$. It happens that the avoidance of spurious permutation groups is easiest if we try to refine the cover as little as possible. For example, we might let $C_2 = \{\{AB\}, \{BC\}, \{AC\}, \{CD\}, \{CE\}, \{DE\}\}$. The job that we want $N$ to do, i.e., give the motions of the blocks of $C_2$ under each input, is specified by the flow table of Fig. 15. Thus our problem is to construct a machine $L$ that, when combined with $K$, gives the behavior of Fig. 15. Since each block of states in $C_1$ has three subblocks in $C_2$, $L$ needs three states, say 1, 2, and 3. We are now faced with the same old problem...
in assigning these three states. For example, should we use the assignment shown in Fig. 16(a) or that of Fig. 16(b)?

In our present special situation we can solve this problem. Observe that under input 1, the block \{ABC\} maps invertibly onto the block \{CDE\}. This transition must permute the states of L. Which permutation is produced depends on how we make the assignment. Why not make the permutation as simple as possible—an identity? This forces us to choose the mapping of Fig. 16(b).

Thus the main ideas of our construction of an arbitrary machine as a cascade of permutation-reset machines are as follows:

1. Factor out on the left the first machine, using a suitably coarse cover.
2. Factor out succeeding machines by successively refining covers.
3. Make each refinement so modest that (a) the assignment problem can be solved without introducing spurious groups; (b) the machine factored out is permutation-reset.

**PART II**

The notation used here will coincide with that of the first ten sections of *Naive Set Theory* by Paul Halmos, with the following additions: (1)
If $C$ is a collection of sets, then $\text{max } C$ means the set of all elements of $C$ that are maximal with respect to set inclusion. (2) $E_A$ means the identity map on the set $A$. (3) If $I$ is a set, then $I^*$ is the free semigroup generated by $I$. (4) If $S$ is a semigroup of transformations of a set $Q$, then $S$ is the semigroup got by adjoining $E_Q$ and $\{w: \text{for some } p \in Q \text{ and for each } q \in Q, w(q) = p\}$ to $S$.

We assume the reader is familiar with finite-state machines. If $M$ is a finite-state machine, $Q_M$ will denote its state set and $I_M$ its input set. For each $x \in I_M$, $x_M$ will be the transformation of $Q_M$ that takes each state onto its successor under the input $x$. $S_M$ will be the semigroup generated by these transformations. For each $x \in I_M$, $Z_{x,M}$ will be the mapping that takes each state into the output it produces, given input $x$. If the outputs are state determined, we shall write $Z_M$ instead of $Z_{x,M}$.

For technical reasons, we shall often append maps that reset to each state and the identity map to the state transformations of a machine, i.e., use $S_M$ instead of $S_N$.

If $M$ is a machine, then $C$ is a cover for $M$ means $C$ is a nonempty collection of nonempty subsets of $Q_M$ for which for each $w \in S_M$ and $R \in C$, $w(R)$ is a subset of an element of $C$ (hereafter written $w(R) \subseteq C$). Note that $Q_M = \bigcup C$.

If $C$ is a cover for $M$ and $N$ is a machine, then $N$ tells where in $C M$ is means:

1. $I_N = I_M$ and $Z_N$ maps $Q_N$ onto $C$.
2. For each $x \in I_M$ and $q \in Q_N$, $x_M(Z_N(q)) \subseteq Z_N(x_N(q))$.

Note that by state reduction on $N$, $Z_N$ can be converted to an identity.

If $K$, $L$, and $N$ are machines, then $N \subseteq K \rightarrow L$ (i.e., $N$ is a series composition of $K$ followed by $L$) means:

1. $Q_N \subseteq Q_K \times Q_L$ and $I_N \subseteq I_K$ and $I_L \subseteq I_K \times Q_K$.
2. For each $x \in I_N$ and $(p, r) \in Q_N$, $x_N(p, r) = (x_K(p), (x, p)_L(r))$.

(No requirements are imposed on $N$’s output map.)

We first prove that the construction shown in Fig. 17 can be made.

**Proposition 1.** For each machine $M$ there is a cover $C$, not containing $Q_M$, and a machine $N$ for which (1) $N$ tells where in $C M$ is, (2) for each $x \in I_N$, ran $x_N$ is either $Q_N$ or a singleton, and (3) the permutations of $Q_N$ are uniquely determined.

**Proof.** Let $C = \text{max } \{w(Q_M): w \in \tilde{S}_M\} - Q_M$. Let $Q_N = C$ and $Z_N = E_C$. For each $x \in I_N$ (or $I_M$) define $x_N$ as follows:

1. If $x_M(Q_M) \neq Q_M$, then $x_M(Q_M) \subseteq R' \in C$ for some $R'$. For each
R ∈ C, let xN(R) = R'. (Note. There may be more than one R' that will do.)

(2) If xM(QM) = QM, then we assert that xM permutes the elements of C.

(a) For each R ∈ C, xM(R) ∈ C, not just ⊆ C, for if xM(R) were a proper subset of R' then some power of xM = xM⁻¹ when applied to R' would give a set properly containing R, which is impossible, for since C is max of something, it cannot contain any proper subset of R.

(b) Since xM(R) = R' implies xM⁻¹(R') = R the map on C produced by xM is invertible. Let xN be this map.

Finally, just observe that N tells where in C M is, N is permutation-reset, and the permutations of Qx are uniquely determined.

Remark. It is also obvious that the group of permutations in Sx is a homomorphic image of a subgroup of SM.

Next we prove that the construction can be continued as shown in Fig. 18.

PROPOSITION 2. If K and M are machines, C is a cover for M not consisting entirely of singleton sets, and K tells where in C M is, then there are machines N and L and a cover C' for M for which:

(1) C' is a proper refinement of C, i.e., if R ∈ C', then R ⊆ C and for some T in C, T is not a subset of any element of C'.

(2) N tells where in C' M is.

(3) N ∈ K → L.

(4) For each (x, p) ∈ IL, ran (x, p)L is either QL or a singleton.

(5) There is an R ⊆ QM for which there is a homomorphism from the group of all permutations of R produced by state transformations of M.
onto the group in $S_L$. (It follows that the group in $S_L$ is a homomorphic image of a subgroup of $S_M$.)

Proof. Step 1: Construction of $C'$. Call two elements of $C$ similar if each is the image of the other under some element of $S_M$. Call an element $R$ of $C$ initial in $C$ if it is not the image, under any element of $S_M$, of any element of $C$ not similar to $R$. (If $R$ is a singleton and is initial in $C$, then every element of $C$ is a singleton.) If $R$ is initial in $C$, so are all elements of $C$ that are similar to $R$. Let $D$ be any such similarity class of initial elements. Now let $C'$ be the cover got from $C$ by replacing each $B$ in $D$ with $\max \{w(R) : w(R) \text{ is a proper subset of } B \text{ and } w \in S_M \text{ and } R \in C\}$. Observe that $C'$ will always be a proper refinement of $C$. $C'$ will be a cover so long as $C$ is not composed entirely of the singletons (in which case $C'$ would be empty).

Before proceeding with the next step, we need the following lemma: If $P$ and $R \subseteq D$, there are $v_P \in S_M$ and $v_R \in S_M$ for which $v_P \in P$ and $v_R \in R$. Then there are integers $n, m$ for which $(v_P)^n \in P$ and $(v_R)^m \in R$. Let $v_P^R = w$ and $v_R^P = (yw)^{nm-1}y$ and observe that they are inverses.

Remark. If $A$ and $B$ are elements of $C'$ but not of $C$, then $A$ and $B$ are subsets of elements of $D$ (since, if not, they would have to be subsets of elements of $C - D$, which is a subset of $C'$). Suppose $A \subseteq R$ and $B \subseteq P$ and $v_P^R(A) = B$; then $v_P^R(B) = A$.

Step 2: Construction of $N$ and $L$. Pick some fixed element $q$ of $D$. Let $Q_N$ be $\{R \subseteq C' : R \subseteq q\}$. Let $I_L = I_K \times Q_K$; let $Q_N$ be $Q_L \times Q_L$; let $I_N = I_K = I_M$. Let $Z_N$ map $Q_N$ onto $C'$ so that for each $(p, r) \in Q_N$:

1. If $p \notin D$ then $Z_N(p, r) = p$.
2. If $p \in D$ then $Z_N(p, r) = v_q^p(r)$, where for each $p \in D$ we select a $v_p^p$ and a $v_q^q$ as in the lemma above, and stick to this selection henceforth. This imposes within each element of $D$ a common coordinate system, the state space of $L$. Using the remark above, check that (1) for each $r$ in $Q_L$ and $p$ in $D$, $v_q^p(r) \in C'$, and (2) $Z_N$ is onto $C'$.

Now it remains to define the $x_N$'s so that $N$ tells where in $C'$ $M$ is.

For each $x \in I_N$ and $(p, r) \in Q_N$ define $x_N(p, r) = (s, t)$ by

1. $s = x_K(p)$,
2. if $s \in C \cap C'$ let $t = r$,
3. if $p \in C \cap C'$ and $s \in D$ let $t = v_q^p x_M(p)$,
4. if $p \in D$ and $s \in D$ let $t = v_q^p x_M v_q^p(r)$.

This completes the construction. Conclusions 1 and 3 of Proposition
2 are fulfilled trivially. Conclusion 2 is verified by direct substitution of the definitions of \( x_N \) and \( Z_X \) into the definition of \( N \) tells where in \( C' M \) is. To verify Conclusion 4 observe that part 2 of the definition of \( x_N \) produces identity permutations of \( Q_L \), part 3 produces state transformations that reset all states to one, and part 4—since \( v_q^e x_M v_q^p \) permutes \( q \)—produces permutations of \( Q_L \) by the same argument used in Proposition 1. These permutations generate a group homomorphic to the group of permutations of \( q \) generated by state transformations of \( M \).

PART III

A theorem of Krohn-Rhodes states that each finite-state machine \( M \) can be built as a cascade connection of 2-state machines with no non-identity permutations and permutation machines whose groups are simple groups that are composition factors of subgroups of \( S_M \). In the light of what we have already shown, to prove the Krohn-Rhodes result it will suffice to show that each permutation-reset machine can be built as a cascade connection of 2-state machines and permutation machines whose (simple) groups are composition factors of the permutation group of the permutation-reset machine.

The main problem in proving the above result is to show that each permutation machine can be built as a cascade connection of permutation machines whose (simple) groups are composition factors of the group of the permutation machine. By iteration, it will suffice to show that if \( M \) is a permutation machine with group \( G \), \( H \) is a normal subgroup of \( G \), and \( G/H \) is the factor group, then there is a machine \( N \) equivalent to \( M \) for which \( N \in K \rightarrow L \) where \( S_K \cong G/H \) and \( S_L = H \). The idea of the proof is to give \( L \) the same state set as \( M \), but only allow elements of \( H \) as state transformations of \( L \). Whenever \( L \) is unable to imitate \( M \) because of an external input that does not belong to \( H \), the machine \( K \) comes to the rescue by storing a permutation in \( G \) that, when applied to the “mistaken” state of \( L \), turns it into the state of \( M \) required by the simulation. Luckily, \( K \) needs as states only a set of leaders of the cosets in \( G/H \), and the state transformations of \( K \) generate \( G/H \).

The actual proof accomplishes these objectives in reverse order. First we design \( K \) so that \( S_K \cong G/H \). Then we pick \( Q_L \) and require that \( Z_N \) translate the state of \( L \) by the permutation stored in \( K \). This determines how \( L \) must operate in order that \( N \) imitate \( M \), and we finish by observing that the required operation of \( L \) uses only elements of \( H \) as state transformations. Incidentally, the ideas behind this construction were
invented by Frobenius before 1900; for automata, they were rediscovered by Krohn in 1962.

**Proposition 3.** Let $M$ be a permutation machine with $S_M = G$. Suppose $G$ has a normal subgroup $H$ and factor group $G/H$. Then we can construct machines $K$, $L$, and $N$ so that,

1. $N$ is equivalent to $M$,
2. $N \cong K \rightarrow L$,
3. $S_K \cong G/H$,
4. $S_L = H$.

**Proof.** Let $Q_K$ be a set of coset leaders in a left coset decomposition of $G$ by $H$. (Thus $Q_K \subset G$.) For each $x \in I_M$ and $p \in Q_K$ let $x_K(p)$ be the leader of the coset containing $x_Mp$. Thus $S_K = G/H$. Let $Q_L = Q_M$. Let $Q_N = Q_K \times Q_L$. Let, for each $(p, r) \in Q_N$, $Z_N(p, r) = p(r)$. Next force $N \rightarrow K \rightarrow L$ by requiring that for each $x \in I_M$, $x_N(p, r) = (x_K(p), (x, p)_L(r))$. But $(x, p)_L$ is not yet defined; we now define it so that $N$ will be equivalent to $M$. For each $(p, r) \in Q_N$ and $x \in I_M$ we want:

$$x_M(Z_N(p, r)) = Z_N(x_N(p, r))$$

so

$$x_M(p(r)) = Z_N(x_K(p), (x, p)_L(r)) = x_K(p)((x, p)_L(r)).$$

We need only solve in the group $G$, the equation

$$x_Mp = x_Kp(x, p)_L$$

by

$$(x, p)_L = [x_K(p)]^{-1}x_Mp.$$ 

This completes the construction; it only remains to check that $S_L = H$. Every element of $S_L$ is of the form $(x_K(p))^{-1}x_Mp$. But $x_K(p)$ is the leader of the left $H$-coset containing $x_Mp$, so $(x_K(p))^{-1}x_Mp \in H$.

We now use this construction to show that each permutation-reset machine can be built as a cascade connection of 2-state machines with no nonidentity permutations and permutation machines whose (simple) groups are composition factors of the permutation group of the permutation-reset machine. First, discard the reset inputs of the machine and apply the Frobenius construction with $H = \{\text{identity}\}$, $G/H = G$. Then observe that the resets can be put back in by letting each reset (1) produce the identity state transformation on the machine $K$, and (2)
reset \( L \) to the state \( p^{-1}(r) \) where \( r \) is the desired reset state and \( p \) is the state of \( K \). Now \( L \) is a machine having only resets and the identity map as state transformations. *Any* binary coding of the states makes \( L \) into a parallel connection of 2-state machines with no nonidentity permutations. On the other hand, \( K \) is a group machine and can be built as a cascade of its (simple) composition factors by repeated application of the Frobenius construction.

**Acknowledgment**

Most of the research described in this paper was done at the Research Laboratory of Electronics, M.I.T., under joint contract to the Army, Navy, and Air Force. The remainder was supported by the Ford Foundation, through an Engineering Postdoctoral Fellowship at M.I.T., and by Air Force Contract AF 49(638)-1440 at Stanford University.

Encouragement, editorial assistance, and illuminating technological discussion were contributed by Doctors Trenchard More, Seymour Papert, John Rhodes, Juris Hartmanis, and Michael Arbib.

The reader can well imagine the difficulties faced by Mrs. Kathy Omura and Miss Heather Thomson, who typed the manuscript.

The author would like to thank Mr. W. F. Cutlip and Dr. Abraham Ginsburg for ferreting out errors in the construction.

**Received: November 16, 1966**

**References**

*Finite-State Machines*


Mathematics


More, T., Jr., M.I.T. Course Notes, Seminar in Algebra, Fall 1963.