while in the second we have
\[ M_A(a, \sigma) \in I_A \iff b = M(q, \sigma) \in I \iff M'(q, \sigma) = b' \in I' \]
the last implication coming from the observation: \( b \in I \iff b' \in I \) as is seen using the connectivity of \( B \).

Finally, we remark that \( o' \geq o_A \), the covering being regarded in \( B \). So by the above we have
\[ o_A M_A x = o' M' x \quad \text{whence} \quad M_A(o_A, x) \in I_A \iff M'(o', x) \in I' \]
and we conclude that \( o' \geq o_A \) when regarded in \( B' \) as well. By the Proposition of the previous section, \( B' \) covers \( A \), thus contradicting the minimality of \( B \).

*Theorem (Paull–Unger):* Let \( A \) be an incomplete automaton. Corresponding to each irredundant covering \( B \) of \( A \) is an \( m_A \)-closed set system \( \mu_B \) with the following properties.

**Property 1:** \( \mu_B \leq \gamma_A \).

**Property 2:** \( |\mu_B| = |S_B| \).

Conversely, to each \( m_A \)-closed set system \( \mu \leq \gamma_A \) there corresponds an irredundant covering \( B \) of \( A \) having \(|\mu|\) states.

**Proof:** Given that \( B \) is an irredundant covering of \( A \), put
\[ \mu_B = \{ B_i : b_i \in S_B \} \]
with
\[ a \in B_i \iff b_i \geq a. \quad (6) \]
For these subsets \( B_i \subseteq S_A \) it is easily verified that (2) \( \implies \) (I) and (5) \( \iff \) (II) so that \( \mu_B \) is a set system. To see that it is \( m_A \)-closed, let \( B_i \in \mu_B \). By the Lemma on additive operators, we need only show that \( m_A(B_i) \leq \mu_B \). Now a block of \( m_A(B_i) \) is either a singleton or of the form \( \{ M_A(a, x) : a M_A x \in B_i \} \).

Using (6) and (4) we have \( b_j = M_B(b_i, x) \geq M_A(a, x) \in B_j \). Thus \( \{ M_A(a, x) \} \subseteq B_j \in \mu_B \). Property 2 is immediate. As for Property 1, it is sufficient to show that \( p R_A q \) for any \( p, q \in B_i \in \mu_B \). Suppose conversely that for some \( p, q \in B_i \) there exists \( x \in \Sigma^* \) with
\[ M_A(p, x) \in I_A \quad \text{and} \quad M_A(q, x) \notin I_A \]
(or vice versa). Then \( b_i \geq p, q \) and using (4)
\[ M_B(b_i, x) \geq M_A(p, x), M_A(q, x). \]
Considering (3) we have a contradiction.

Conversely, let \( m \leq m_A \)-closed and \( \mu \leq \gamma_A \). Put
\[ S_\mu = \mu \quad I_\mu = \{ B_j : B_j \subseteq I_A \} \]
and designate one block \( O_\mu \) for which \( o_j \in O_\mu \). Let \( B_i M_\mu \sigma \) for \( \sigma \in \Sigma \) whenever \( a M_\mu \sigma \) for some \( a \in B_i \), setting
\[ M_\mu(B_i, \sigma) = \{ M_\mu(a, \sigma) : a \in B_i, a M_\mu \sigma \} \leq B_j \in \mu \]
the latter being possible because \( \mu \) is \( m_A \)-closed. Then \( m_\mu \) extends to an incomplete automaton \( B_\mu = (S_\mu, \Sigma, M_\mu) \) having \(|\mu|\) states, so we only have to see that \( B_\mu \) is an irredundant covering of \( A \). For this purpose, we first observe that
\[ B_j \notin I_A = B_j \subseteq S_A - I_A \]
or otherwise we would contradict \( \mu \leq \gamma_A \). Clearly \( o_A M_\mu x \) implies \( O_\mu M_\mu x \) for any \( x \in \Sigma^* \) and then
\[ M_A(o_A, x) \in I_A \iff M_A(o_A, x) \in M_\mu(o_A, x) \cap I_A \iff M_\mu(o_A, x) \in I_\mu \]
by the observation just made. Thus \( O_\mu \leq o_A \) and by the Proposition of the previous section \( B_\mu \) covers \( A \). The irredundancy follows from the fact that \( \mu \) is a set system, i.e.,
\[ B_i \geq a \iff B_j \geq a \quad (\text{all} \ a) \imply B_i \subseteq B_j \iff i = j. \]

*Corollary:* The problem of finding a minimal covering of an incomplete automaton \( A \) is equivalent to that of finding a set system \( \mu \) with
1. \( \mu \) is \( m_A \)-closed
2. \( \mu \leq \gamma_A \)
3. \( \rho \) satisfies 1, 2 \( \iff \mu \leq |\rho| \).

**Proof:** The lemma ensures that it is sufficient to consider only irredundant coverings.

Such problems, given arbitrary additive \( m \) and arbitrary \( m \)-closed \( \gamma \) are called Paull–Unger problems in [4].

**REFERENCES**


---

**On Autonomous NOR Sequential Machines**

K. C. WANG, MEMBER, IEEE

Abstract—This note considers the analysis of autonomous NOR sequential machines from a linear machine point of view. It is shown that, under certain conditions, the techniques developed for linear machines are directly applicable to NOR machines.

Index Terms—Autonomous NOR machines, linear machines, sequential machines, state diagrams.

I. INTRODUCTION

Recently, King [1] studied the state behavior of autonomous, iterative NOR machines composed of \( n \) NOR gates separated by \( n \) delay elements. The state behavior of such a machine may be described by

\[ s(t + 1) = As(t) \]

(1)

where “+” stands for complementation and \( A = (a_{ij}) \) is the connection matrix in which \( a_{ij} = 1 \) if the output of the \( j \)th gate is, after delay, connected to the input of the \( i \)th gate, and \( a_{ij} = 0 \) otherwise, and the addition in (1) is the usual \( \oplus \). It is shown in [1] that the necessary and sufficient condition for such a machine to generate a state diagram consisting entirely of cycles is that the circuit of the machine forms

---

Manuscript received June 23, 1969; revised October 24, 1970.

The author is with the Department of Computer Science, Washington State University, Pullman, Wash. 99163.
a loop. The resulting state diagram can be described by a cycle set \( C_N = \{a_1(l_1), a_2(l_2), \ldots \} \), where \( a_i(l_i) \) indicates that there are \( a_i \) cycles of the length \( l_i \). It was also shown that \( C_N \) is equal to the linear cycle set \( C_F \) of an \( n \)-stage circulating feedback shift register if \( n \) is even, and is equal to \( C_F \) with every cycle length doubled and every multiplicity halved if \( n \) is odd. In this note, the state behavior of this class of machines will be studied from a linear machine point of view. To be more specific, we shall present an analytical proof to the structures of \( C_N \). This is based on the theory of linear autonomous machines characterized by nonsingular affine transformations [3], [4]. Extensions to tree graphs are also considered. It is shown that the linear machine techniques can be applied very effectively wherever there exists at most one 1 in every row of \( A \).

II. THE STRUCTURE OF \( C_N \)

Let \( A \) be the connection matrix of an \( n \)-stage iterative NOR machine that forms a loop. Then \( A \) has the form [1]

\[
A = \begin{bmatrix}
0 & 0 & \cdots & 0 & 1 \\
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0
\end{bmatrix}
\]  

Since each row of \( A \) has only one 1, the state equation (1) can be rewritten as

\[
s' = As + a
\]  

where \( a \) is the column vector of all 1’s, and the operations are in the field of two elements. Equation (3) describes an affine transformation in which \( A \) is the linear part and \( a \) is the translating vector. Transition graphs of affine transformations have been studied in [3], [4]. We shall apply the results here to explore the structure of \( C_N \) which is the graph of (3).

First, notice that \( A \) is the companion matrix of the polynomial \( x^a - 1 \). We consider the following two cases.

Case 1—(\( n \) odd): If \( n \) is odd, then \( x^a - 1 \) has only one factor of \( (x - 1) \). This follows from the fact that

\[
x^a - 1 = (x - 1)(x^{a-1} + x^{a-2} + \cdots + x + 1)
\]

and that \( b(1) \equiv 0 \) in the binary field. Denote by \( C_b \) the linear cycle set of \( b(x) \). Then the linear cycle set \( C_F \) is equal to the product [2] of \( C_b \) and \( \{2(1)\} \), the linear cycle set of \( (x - 1) \). The result is simply two copies of \( C_F \).

Now we observe that \( (A - I)a = 0 \) for \( (A - I)a \). Therefore, \( a \) is in the null space of \((x - 1)\), or \( a = 1 \) in this subspace. From (4), there exists a basis of \( V^a \) in which (3) reduces to

\[
s' = \begin{bmatrix}
1 & 0 \\
0 & B
\end{bmatrix} s + \begin{bmatrix}
1 \\
0
\end{bmatrix},
\]

The graph of the transformation of \( u' = [1] \cdot u + [1] \) on the one-dimensional space is a cycle of length 2. Therefore, \( C_N \) is equal to the product of the cycle sets \( \{2\} \) and \( C_b \). To show that \( C_N \) is equal to \( C_F \) with every cycle length doubled and every multiplicity halved, it suffices to show that the cycle lengths of \( C_N \) are relatively prime to 2. From (4), \( n \) is the period of \( b(x) \). Since the cycle lengths of \( C_N \) must all divide \( n \), which is assumed to be odd, they are not divisible by 2. Hence \( C_N \) has the kind of structure as claimed.

Case 2—(\( n \) even): If \( n \) is even, \( x^a - 1 \) can be factored as

\[
x^a - 1 = (x - 1)^k \cdot p(x)
\]

where \( k \geq 2 \) and \( p(1) \equiv 0 \). Then there exists a basis of \( V^a \) in which \( A \) is the direct sum of \( A_k \) and \( P \), which are the companion matrices of \((x - 1)^k \) and \( p(x) \), respectively. As before, the vector \( a \) is in the null space of \((x - 1) \) because \( Aa = a \). But this also implies that \( A' a = a \) for any \( i \). Therefore, the set of vectors \{\( a, Aa, A^2a, \ldots, A^{k-1}a \)\} cannot be independent. In this case we say that \( a \) is not a cyclic vector of \( A_k \).

With respect to the new basis (3) becomes

\[
s' = \begin{bmatrix}
A_k & 0 \\
0 & P
\end{bmatrix} s + \begin{bmatrix}
a_k \\
0
\end{bmatrix}
\]

where \( a_i = a \) in the new basis. It was shown in [3] that the graph of the transformation \( u' = A_ku + a_i \) is identical to that of \( A_k \) if \( a \) is not a cyclic vector of \( A_k \), which is the case here. Thus, \( C_N \) is identical to \( C_F \).

III. GENERAL CASE

We now consider the nor machines that do not form a single loop. In general, the \( i \)th equation of (1) can be written as

\[
Y_i = \sum_{j=1}^{n} a_{ij}y_j + 1 + \sum_{k_1 < k_2 < \cdots < k_l} (a_{i k_1} \cdots a_{i k_l}) \cdot (y_{k_1} \cdots y_{k_l})
\]

where "+" denotes the modulo-2 sum. It can be seen that (8) reduces to a linear equation if \( i \) is at most one \( a_{ij} \neq 0 \).

Theorem: If each row of \( A \) has at most one \( 1 \), the state diagram of the nor machine characterized by \( A \) is identical to the direct sum of a linear tree and a cycle set.

When the rows of \( A \) contain more than one \( 1 \), the state equation (1) becomes nonlinear and it seems that there exists no analytical way of determining the state diagrams. In fact, it is possible for a non-nilpotent \( A \) to generate a single tree.

REFERENCES


Measures of Op-Code Utilization

CAXTON C. FOSTER, ROBERT H. GONTER, AND EDWARD M. RISEMAN, MEMBER, IEEE

Abstract—The static and dynamic utilization of a set of machine op-codes is examined. Two measures of the effective use of machine instructions are discussed and applied to samples of hand-coded programs and object code.

Index Terms—Gibson mix, information measure, instruction set, op-code frequency, op-codes, programming, programming style.

INTRODUCTION

The trends in computer design have been towards providing greater flexibility for those who utilize the machines. In many instances this has resulted in an increase in the number of machine instructions that are made available. In some cases, the machine language programmer has at his disposal over 200 distinct operations. This flexibility is obtained only through increased cost in machine hardware. In addition, increases in the size of the set of op-codes may require more space in each instruction word to be assigned to hold the op-code. This note examines the effective utilization of the diversity of machine instructions that are provided. An analysis is made of the static and dynamic use of the op-codes in both hand-coded programs and object code on a CDC-3600 computer.

Even though the cost of a central processing unit (CPU) is becoming a small part of the cost of a large computer system, it is still a substantial part of the cost of a mini. While the advent of large-scale integrated circuits will reduce these costs even further there are still the matters of power requirements, space, weight, maintainability, and overall complexity to be considered.

According to information theory [1], the maximum utilization of the information in op-codes is achieved if all op-codes are equally likely. However, this situation is quite improbable. As any assembly language programmer knows, some instructions are used much more frequently than others. Under these conditions, one can devise a variable length code which uses, on the average, fewer bits than the standard approach of an equal number of bits per mnemonic. Thus, more frequently used mnemonics would be encoded into short sequences of bits. However, this system would require a rather sophisticated CPU to handle these variable length instructions that would have little relationship to the standard word boundaries of memory. The complexities introduced by this procedure outweigh the advantages. A further consideration is the nonuniform probability of transition from one instruction to another. It has been shown that this information can be used to reduce the size of the op-code field by means of “conditional instruction decoding” [2].

MEASURES OF OP-CODE USAGE

In this note two measures are proposed as initial methods for analyzing the use of operations that are provided in machine hardware. The first is Shannon’s information measure, to be applied to the usage of the set of op-codes in representative samples of machine code. If there are $T$ instructions and $p_i$, $i = 1, \ldots, T$, represents the probability of the $i$th instruction being used, then the average number of bits of information contained in each op-code is

$$I = - \sum_{i=1}^{T} p_i \log_2 p_i.$$ 

As mentioned before, this measure reaches a maximum value of $I_{\max} = \log_2 T$ when all instructions are used an equal number of times; i.e., $p_i = 1/T$, $i = 1, \ldots, T$. Thus, the relative difference between an observed value of $I$ and $I_{\max}$ is a measure of the utilization of any particular set of op-codes.

The second measure examines the effect of decreasing the number of machine instructions available to the programmer. We limit the number of “usable” op-codes to the $N$ most popular ones and then determine the fraction of the instructions in the original program that would have to be rewritten (because they did not occur in this subset). More formally, we order the instructions from most frequently used to least frequently used in the original program and let $C_k$ equal the number of times the $k$th instruction was used where $C_k \geq C_{k+1}$ for all $k$; and $P$ equal the total number of instructions in the original program. Now, function $f(N)$ is defined to be the fraction of all instruction occurrences included in the set $N$, where $f(N)$ is given by

$$f(N) = \frac{1}{P} \sum_{k=1}^{N} C_k$$

and $g(N) = 1 - f(N)$ is then a measure of the effort necessary to recode the original program onto a machine with a repertoire that is more limited than the original. It will be convenient to plot $g(N)$ versus $N$ on semilogarithmic paper since it has been observed that $g(N)$ is approximately proportional to $e^{-an}$. This measure bears some similarity to Gibson’s analysis [3] of machine execution speed. A running count is kept of the types of instructions executed, such as arithmetic instructions, index register operations, etc. Each type of instruction is weighted by its respective speed of execution to arrive at an overall measure of the speed of the machine.

Both of the measures that have been discussed can be applied to the static and dynamic usage of the machine code. The static case is a measure of the use of op-codes in specifying the logic of the program, where the count is carried out over the instructions written; the dynamic case is a measure of the use of op-codes in executing the logic of the program, with the count over the set of instructions executed.