Sequential Machines, Ambiguity, and Dynamic Programming*

RICHARD BELLMAN

The RAND Corporation, Santa Monica, California

Abstract. Given a sequential machine, in the terminology of E. F. Moore, *Annals of Mathematics Studies*, No. 34, 1956, a problem of some interest is that of determining testing procedures which will enable one to transform it into a known state starting from an initial situation in which only the set of possible states is given.

To treat this problem, we introduce the concept of ambiguity, and show how the functional equation approach of dynamic programming can be applied.

1. *Introduction*

Following previous authors, Moore, [6], Huzino, [4], we shall use the terms *sequential machine* to signify a mathematical system consisting of a finite set of quantities, $i_1, i_2, \cdots, i_M$, which we call *inputs*, a finite set of quantities $j_1, j_2, \cdots, j_N$, which we call *outputs*, and a finite set of quantities $x_1, x_2, \cdots, x_K$, which we call the *states* of the system.

These quantities are interrelated by means of the following Markovian properties:

$$\left.\begin{array}{ll} \text{(a)} & \text{The present state of the system depends only upon the past state and the past input.} \\ \text{(b)} & \text{The present output of the system depends only upon the present state and the present input.} \end{array}\right\} \quad (1)$$

We then have two "multiplication tables" describing the states and outputs, given the past states and inputs. Table 1 shows the states, and a similar table would describe the outputs, which we shall call O_{ij}.

A fundamental problem in this field is that of using this information to determine the current state of a sequential machine, assumed constant until an input is applied. One part of this problem is that of determining when this is possible, and another part is that of determining the state in some optimal fashion when this can be done. Some work in this direction has been done; cf. Ginsburg, [3], Mealy, [5].

We wish to show in this paper that the theory of dynamic programming can be utilized to provide a conceptual and analytic basis for problems of this nature, and, in addition, computational algorithms. As a simple example, we shall consider the classical coin-weighing problem, which has by now been solved in a large number of different ways.

An application of dynamic programming techniques to a closely related class of problems may be found in Bellman, Holland, Kalaba [2].

* Received February, 1959

TABLE 1

		Present state			
		x_1	x_2	$\cdots$	x_N
Present output	I_1	x_{11}	x_{12}	$\cdots$	x_{1N}
	I_2	$\vdots$	$\vdots$		$\vdots$
	$\vdots$				
	I_M	x_{M1}	x_{M2}	$\cdots$	x_{MN}

Next state

2. *Information Pattern*

Let us now consider the formulation of the problem of determining optimal testing methods within the framework of the theory of dynamic programming [1]. To begin with, we must introduce state variables which describe the state of our knowledge at any stage of the process.

Suppose that initially it is known that the system is in one of a number of possible states, x_{a1}, x_{a2}, $\cdots$, x_{ak}. The set

$$S = [x_{a1}, x_{a2}, \cdots, x_{ak}] \tag{1}$$

is called the *information pattern*. Conceivably, S may be the set of all possible x_i, which we call Σ.

As we insert inputs and observe outputs, the information pattern, which is to say, the set of possible current states of the machine, will change.

3. *Ambiguity*

There are now two possibilities. Either the structure of the multiplication tables given above is such that by means of some appropriate sequence of inputs we can finally reach a point where we definitely know the state of the machine, or we cannot.

If we cannot, it is of interest to determine the minimum uncertainty attainable, measured perhaps in terms of the minimum number of possible states. If we can, it is of interest to determine the least number of inputs required to attain certainty.

In what follows, we wish to introduce the concept of *ambiguity*, and show how to calculate it by means of functional equations.

4. *Functional Equations*

Given a set of elements S, we introduce the scalar function

$$n(S) \equiv \text{the number of elements in } S. \tag{1}$$

Let us call $n(S)$ the *norm* of S.

Let us now pose the following problem: Given the information that the current state of the sequential machine is a member of a set S, we wish to use a sequence of inputs which will minimize the norm of the set resulting after k stages.

This is to be a sequential, or feedback process, since we allow, and as a matter of fact, encourage observation of the outputs as the testing proceeds.

Let us define

$$\alpha_0(S) = n(S), \tag{2}$$

and

$$\alpha_k(S) \equiv \text{the norm of the set describing the information pattern after } k \text{ trials, using an optimal testing policy.} \tag{3}$$

In order to derive a recurrence relation connecting $\alpha_k(S)$ with the function $\alpha_{k-1}(S)$, let us describe what happens when we use a particular input I. As a result of observing the output O, and the knowledge of the previous information pattern S, we deduce from the multiplication tables that the current state of the system must belong to a new set which we shall call S_{IO}.

Furthermore, from the multiplication table for outputs, we know that O itself must be a member of a set which is determined by I and the set S. Call this set of possible outputs $T(I, S)$. We do not, of course, know the precise output beforehand, since we do not know the precise state of the machine.

Since we do not know the precise output, we must acknowledge the possibility that it will be the worst possible from our point of view. With this in mind the input I must be chosen so as to minimize.

Putting these remarks together, we see that the functional equation satisfied by the functions $\alpha_k(S)$ is

$$\alpha_k(S) = \operatorname*{Min}_I \operatorname*{Max}_{O \in T(I,S)} \alpha_{k-1}(S_{IO}), \qquad k \geqq 1. \tag{4}$$

This is an application of the principle of optimality [1, p. 83].

5. *Limiting Case*

Let us now consider the limiting case obtained as we let the number of allowable tests become infinite. Since the function $\alpha_k(S)$ is clearly weakly monotone decreasing as a function of k as k increases, we can define for each set S the function

$$\alpha(S) = \lim_{k \to \infty} \alpha_k(S). \tag{1}$$

We call this function $\alpha(S)$ the *ambiguity* of S. It satisfies the functional equation

$$\alpha(S) = \operatorname*{Min}_I \operatorname*{Max}_{O \in T(I,S)} \alpha(S_{IO}). \tag{2}$$

If $\alpha(S) = 1$ for every $S \subset \Sigma$, we say that the sequential machine is *unambiguous.*

6. *Discussion*

In passing, let us note that we have used a rather useful analytic device in discussing the problem of determining some state of the machine. Not only have

we posed a feasibility problem, but we have associated it with a variational problem. Frequently, the two in combination are easier to attack than the feasibility problem alone.

7. *Minimum Time*

Along these very same lines, let us consider the problem of determining a testing program that will reduce S to a single element as quickly as possible.

Define the new function

$$f(S) = 0, \text{ if } n(S) = 1,$$
$$(1) \qquad \equiv \text{the minimum number of trials required to reduce } S \text{ to a single element, provided that } n(S) \neq 1 \text{ initially.}$$

If it is not possible to reduce S to a single element, then $f(S)$ will have an infinite value. The function clearly satisfies the functional equation

$$(2) \qquad f(S) = 1 + \operatorname*{Min}_{I} \operatorname*{Max}_{O \epsilon T(I,S)} f(S_{IO}).$$

The problem of determining whether or not a given sequential machine is ambiguous or not has thus been made equivalent to solving the preceding functional equation.

For a discussion of a problem of related type, see [2].

8. *The Coin-Weighing Problem*

As a simple illustration of these techniques, consider the classic puzzle of using an equal-arm balance to detect one heavy coin in a lot of N coins of similar appearance.

Let

$$(1) \qquad f_N \equiv \text{the maximum number of weighings required using an optimal policy.}$$

At each stage, we are allowed to weigh one batch of k coins against another, and observe the outcome. Two situations can arise. Either the two sets of coins will balance, or they will not. If the two sets balance, the heavy coin must be one of the remaining $N - 2k$ coins; if they do not balance, we know that the heavy coin is in one of the sets of k coins. Since we must take account of both contingencies, we obtain the relation

$$(2) \qquad f_N = 1 + \operatorname*{Min}_{1 \leq k \leq N/2} \operatorname{Max} [f_k, f_{N-2k}].$$

To minimize, we clearly want to make k and $N - 2k$ as equal as possible.

Consequently, we take $k = [N/3]$ or $[N/3] + 1$, depending upon whether N has the form $3m + 1$ or $3m + 2$.

REFERENCES

1. R. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, N. J., 1957)

2. R. BELLMAN, J. HOLLAND, AND R. KALABA, On an application of dynamic programming to the synthesis of logical systems, *J. Assoc. Comp. Mach. 6* (1959), 486–493
3. S. GINSBURG, On the length of the smallest uniform experiment which distinguishes the terminal states of a machine, Research Report, The National Cash Register Company, Electronics Division, Hawthorne, California, 1957.
4. S. HUZINO, "On some sequential machines and experiments," *Memoirs of Fac. Sci.*, Kyushu Univ., Ser. A, Vol. XII, No. 2, 1958.
5. G. H. MEALY, A method for synthesizing sequential circuits, *Bell System Tech. J. 34* (1955), 1045–1079.
6. E. F. MOORE, "Gedanken-experiments on sequential machines," in *Automata Studies*, Annals of Mathematics Series, No. 34, 1956, pp. 129–153.